TIMEMASTER II H.O.

USER MANUAL

APPLIED ENGINEERING
P.O. Box 798, Carrollton, Texas 75006

CONTENTS

NOTICE:

THE TIMEMASTER II H.O. DESIGN, PCB LAYOUT AND ROM FIRMWARE WERE
COPYRIGHTED IN 1984 BY APPLIED ENGINEERING. THIS MANUAL IS ALSO
COPYRIGHTED. A PATENT IS PENDING ON THE TIMEMASTER II H.O.

REV 1.2

Thank you very much for selecting the H.O. for use in your computer. We feel confident that you will enjoy owning it and using it as much as we've enjoyed designing it and constructing it.

Because of the fact that all different kinds of people buy Apple computers and clock cards for them, for literally thousands of different types of applications, we have tried to categorize the sections in this manual with regard to computing skill. In this way, if you're just beginning at computing, you can avoid the more technical and programming related sections of the manual. On the other hand, if you possess more computer skills, you can quickly skim over the beginning sections and move right into the necessary information for programming. Each section will fall into one of four skill levels. The skill level is indicated with the numbers 1 through 4 in parentheses just to the right of each section heading. Please do not be intimidated by reading skill levels that you may think are beyond you because chances are you're a lot smarter about computers than you think. Of course, no programming skill is required to use the Timemaster II H.O. because hundreds of commercially prepared software programs are designed to use it automatically and nearly all serious software under development today is designed to read your clock card. In general, the manual begins in the easier skill levels and moves to the more advanced. However, this order is not strictly adhered to. A description of these four skill levels follows below:

(1)   You should know how to get inside your Apple, make a copy of a disk, be able to CATALOG a disk, and RUN a program. If you do not know how to do this, please see pages 57 through 69 in the Apple IIe owners manual or pages 15 and 16 in the Apple II+ Applesoft Tutorial.

(2)   Simple BASIC Programming Skills – If you do not currently possess these skills, a few hours spent with the Applesoft Tutorial will give them to you.

(3)   Advanced BASIC Programming Skills and small knowledge of electronics. You should know what is meant by "when this line is pulled low and be able to convert numbers from hex to decimal and back again. This skill level is rarely shown because the TimeMaster II H.O. is so easy to use in BASIC.

(4)   Machine Language Programmer – As the title would indicate, you need to know how to program in machine language. You'll find many source listings in this manual as well as on disk. All machine language programs on disk were written with the S—C Macro Assembler. We have experimented with many assemblers and we agree with the software reviewers that the S—C assembler is the best. We regret that we cannot provide source code for other assemblers. We remind our customers that competitive clock cards come with little or no source code at all. The S—C Macro assembler may be purchased at low cost from S—C Software, telephone number 214-324—2050.

Welcome to the world of real time!

Your Applied Engineering TIMEMASTER II H.O. will greatly expand
the use of your Apple computer by adding the dimension of real
time and date in intervals of 1 millisecond to 99 years.

Applications of the H.O. are limited only to the imagination.
Anywhere you need to know the time or date you can use the
H.O. Applications include time and date stamping of reports,
checks, letters, file updates, calculating time intervals, and
recording measurements (data logging).

As you read this manual, you will soon learn why the TIMEMASTER
series is the favorite among professional programmers.

```
*****************************************************************************************************************************
*                                                                                                                         *
*                                        IMPORTANT! (1)                                                                   *
*                              Make a backup copy of your H.O. disk                                                       *
*                                                                                                                         *
*                              before installing the clock or doing                                                      *
*                                                                                                                         *
*                              anything else.                                                                            *
*                                                                                                                         *
*****************************************************************************************************************************
```

The H.O. has many options. These options can sometimes confuse a
computer novice. The ONLY way we could have made the H.O. easier
to use would be to have limited its performance. Some of the
programs on the H.O. disk were written for just one of the
modes (the TIMEMASTER II H.O. is really 8 clocks in one) so be
sure you are in the right mode for the sample program you are
experimenting with.

Your new clock uses the latest in CMOS and NMOS large scale
integration ([.51) technology to bring to you a unique peripheral
which retains the day, date, and time even when your Apple
computer is turned off. The on—board ROM allows the highest
degree of programming simplicity as well as a very high degree of
compatibility with existing software. The TIMEMASTER was
designed to meet or exceed all of Apple's high quality standards.

Your new TIMEMASTER II H.O.is a fourth generation of clock
cards designed by Applied Engineering. Our first clock card was
designed in 1980. It was called the Time II. Our second clock
card was the Timemaster and it had many improvements over the old
Time II. In the Timemaster II, we added PRODOS compatibility.
With the Timemaster II H.O., we've added BSR control and other
features and yet maintained software compatibility with both the
older Timemaster and Timemaster II. We consider the TIMEMASTER
II H.O. (just H.O. for short) to represent the zenith in computer
clock technology. It has received rave reviews from both
professional programmers, beginning programmers, as well as the
average computer user. Again, welcome to the world of real time!

1)   TURN OFF THE APPLE'S POWER SWITCH. This is very important to
        prevent damaging the Apple as well as your H.O.

2)   Remove the cover from the Apple.

3)   Plug your H.O. into any slot except 0. (The   Apple //e has
        no slot 0.) Slot 4 is the preferred slot   (because   some
        commercial software expects a clock to   be   in   slot   4)
        but any slot is fine. Insert the fingers of   the   circuit
        board into the slot you have chosen. The fingers will
        enter the slot with some friction, and will then seat firmly.

4)   Make certain switch #1 on the H.O.is in the ON (closed)
        position.
5)   Replace the Apple's cover.

H.O. SWITCHES   (1)

There is a device in the upper left corner of the H.O. card
containing 4 switches. These provide control over the display
format, enable/disable interrupts, and more. Please never use
pen or pencil to change switch settings. A bent out paper clip
works best.

O	1	SET	Must be closed when setting the clock.
P	2	MODE	Selects display mode (see below)
E	3	NMI	Enables Non—Maskable Interrupt
N	4	IRQ	Enables Interrupt Request

Note: OPEN = OFF. Push down toward "OPEN" to turn switch off.

The normal settings are switches 1 and 4 closed and 2 and 3 open.

Your H.O. has two switch—selectable modes, controlled by
switch 2 on the card. The most powerful of these modes is called
the TIMEMASTER mode, which includes compatibility with both DOS
3.3 and PRO DOS. The other mode is included for compatibility
with   certain   older   commercial   software   products   which   were
designed   around   older   generation   clocks   and   do   not   allow
modification for direct usage with the 11.0.

MODE	Sw 2	Format
TIMEMASTER	open	various, see below
Apple Clock	closed	MO/DD HH:MM:SS.WYY

TIMEMASTER MODE (SW 2 open) (1)

This is the most powerful mode and can supply the time/date
information formatted in seven different ways. The desired format
is selected by PRINTing a single character to specify which
format to use.

In the following table, the character between quotation marks is
the format selection character. Each format is illustrated using
Friday, December 14, 1984, 3:30 PM.

```
        " "     MO/DD HH:MI:SS.WYY     APPLE CLOCK MODE
                12/14 15:30:23.384

        ":`  W MO/DD/YY HH:MI:SS       TIMEMASTER MODE
             3 12/14/84 15:30:23
        "%" WWW MMM DD HH:MI:SS PM     THUNDER CLOCK
             FRI DEC 14 03:30:23         APPLESOFT

        "&" WWW MMM DD HH:MI:SS        THUNDER CLOCK
             FRI DEC 14 15:30:23         APPLESOFT
         "#" MO,OW,DD,HH,MI,SS         THUNDER CLOCK
             12,03,14,15,30,23           APPLESOFT
        ">" WWW MMM DD HH:MI:SS PM     THUNDER CLOCK
             FRI DEC 14 03:30:23 PM        INTEGER

        "<" WWW MMM DD HH:MI:SS        THUNDER CLOCK
             FRI DEC 14 15:30:23           INTEGER
```

So   that   the   colons   can   be   accepted   by   an   Applesoft   INPUT
statement,   formats   which   contain   colons   are   preceded   by   a
quotation   mark.   The   quotation   mark   tells   Applesoft   to   allow
colons   in   the   input   string.   The   first   four   formats   above   are
designed for use with Applesoft and include the leading quotation

mark. The last two formats are designed for use with Integer BASIC, and begin with a space rather than a quotation mark.

Even though there are two formats which display the time in AM/PM format using a 12—hour clock, the internal hardware is still keeping time in 24—hour format.

APPLE CLOCK MODE (SW 2 closed) (1)

The Apple Clock mode emulates the format of the Mountain Hardware Apple Clock, for compatibility with commercial software which specifies the Mountain Hardware card and does not allow other brands. Certain "signature" bytes in the firmware for this mode should be recognized by such software, allowing you to use **it** with your H.O.

The Mountain Hardware card delivers the time/date in this format:

        MO/DD HH;MI;SS.MMM

Note the semicolons separating hours, minutes, and seconds. The last three digits are milliseconds. Hours are given for a 24—hour clock (00—23) .Note also that day—of—week and year values are not present. The Mountain Hardware clock does not provide any day—of—week or year values.

The H.O. card substitutes day-of-week and year values for the millisecond values. (If you need time to the nearest millisecond, the H.O can deliver it in a much more accurate way. Please see page 10 for more information.) So, the H.O. substitutes colons for semicolons, giving the standard display of time:

        MO/DD HH:MI:SS.WYY
        12/14 15:30:23.384 (Fri, Dec 14, 1984, 3:30 PM)

So that the colons can be accepted by an Applesoft INPUT statement, the format   is preceded by a quotation   mark. The quotation mark tellsApplesoft   to   allow colons in   the input string.
                    SETTING THE CLOCK   (1)

To set the clock, the switch marked "SET" (switch 1) must be closed (ON). Once the time and date have been properly set, you may wish to protect the clock by placing switch 1 in the open (off) position. However, as of yet, no accidental time change has ever occurred.

The program called SET CLOCK conveniently reads the current clock setting, allows you to enter new date and time values, and then sets the clock to these new values at your signal. RUN the program and follow the directions on your screen. The program automatically finds your TIMEMASTER II H.O. card by searching slots 1 through 7 for certain "signature" bytes. The program also determines which mode you have selected with switch 2.

It is very easy to read the date and time from the H.O. The
H.O. contains 2048 bytes of on-board ROM. The ROM contains the
software that makes it easy to obtain the date and time.

The simplest way to read the date and time involves three steps:

        1)   set input and output to the H.O.
        2)   INPUT the date/time info
        3)   restore input and output to keyboard and screen

Here is a simple example:
        100   SLOT = 4
        110   PRINT CHR$(4) "IN#"SLOT
        120   PRINT CHR$(4) "PR#"SLOT
        130   INPUT A$
        140   PRINT CHR$(4) "IN#0"
        150   PRINT CHR$(4) "PR#0"

The program will receive the formatted date/time information in
the string A$, in the format determined by the MODE switch on the
H.O. The easiest way to tell what mode you have is to add one
more line to the program:

        160   PRINT A$

Type in the program above and run it. Remember to put in line 100
the slot number that your H.O. is in.

If you have switch-selected the TIMEMASTER mode, you can specify
which of the formats to use. Change line 130 to one of the
following, and try running the program again:

        130   INPUT ":";A$
        130   INPUT " ";A$
        130   INPUT "%";A$
        130   INPUT "&";A$

The TIMEMASTER mode also includes a format which returns numeric
values rather than a string, and happens to be the mode used
internally by Apple PRO DOS. Change line 130 and 160 like this,
and RUN again:
        130   INPUT "it ";MNTH,WEEK,DAY,HR,MIN,SEC
        160   PRINT MNTH,WEEK, DAY,HR,MIN,SEC

It is a good programming technique to put CHR$(4) in a string
variable D$, rather than including it "spelled-out" in each DOS
command line. Another useful change to our demonstration program
would be to keep reading the display the time in a loop, until
you press any key on the keyboard. Combining these changes
produces the program on the next page. TRY IT!!!

7

```
100   SLOT = 4                          clock slot
110   D$ = CHR$(4) : HOME               D$ = "control D"
120   PRINT D$"IN#"SLOT                 set input to clock
130   PRINT D$"PR#"SLOT                 set output to clock
140   INPUT ":";A$                      get the time
150   PRINT D$"IN#0"                    restore keyboard in
160   PRINT D$"PR#O"                    restore keyboard out
170   VTAB 12 : HTAB 10                 center the display
180   PRINT A$                          print the time
190   IF PEEK (−16384) < 128 THEN 120   key pressed?
200   POKE −16368,0                     reset key test
```

If you want to rearrange the time/date to a different format, you
can use the MID$ function to pick any part out of the formatted
string.   The following examples show how to access various
portions of the TIMEMASTER and APPLECLOCK formats. please see
your APPLESOFT TUTORIAL if you are unfamiliar with string
manipulation. (It's really quite easy.) TRY IT!!!

```
      TIMEMASTER format:  W MO/DD/YY HH.MI.SS
       WK$ = LEFT$(A$,l)        day of week  ("0" — "6")
       MO$ = MID$(A$,3,2)       month ("01" — "12")
       DA$ = MID$(A$,6,2)       day of month ("01" — "31")
       YR$ = MID$(A$,9,2)       year ("00" — "99")
       HR$ = MID$(A$,12,2)      hour ("00" — "23")
       MI$ = MID$)A$,15,2)      minute ("00" — "59")
       SE$ = RIGHT$(A$,2)       second ("00" — "59)

      APPLECLOCK format:  MO/DD HH:MI:SS.WYY

       WK$ = MID$(A$,16,l)      day of week ("0" — "6")
       MO$ = LEFT$(A$,2)        month ("01" — "12")
       DA$ = MID$(A$,4,2)       day of month ("01" — "31")
       YR$ = RIGHT$(A$,2)       year ("00" — "99)
       HR$ = MID$(A$,7,2)       hour, ("00" — "23")
       MI$ = MID$(A$,1~,l)      minute ("00" — "59")
       SE$ = MID$(A$,13,2)      second ("00" — "59")
```

So if you are only interested in seconds, we need only add or
change these lines to the last program.
```
      180   SE$ = RIGHT$(A$,2)
      185   PRINT SE$
```

If you want the values as numbers, simply use the VAL function.
For example, the day of week as a number would be WK = VAL (WK$).
Using the day of week as a number allows you to look up the full
spelling of the day name. One easy way to program it would be
like this:

```
   WK = VAL(WK$)                   IF WK=3 THEN WK$ = "WEDNESDAY"
   IF WK=O THEN WK$ ="SUNDAY"      IF WK=4 THEN WK$ = "THURSDAY"
   IF WK=l THEN WK$ = "MONDAY"     IF WK=5 THEN WK$ = "FRIDAY"
   IF WK=2 THEN WK$ = "TUESDAY"    IF WK=6 THEN WK$ = "SATURDAY"
```

You can obviously do a similar thing to the spelling of the month names. You can prefix the year with a "19" or a "20" using any method you like.

If you have the clock set in one of the 24—hour modes and still want to read AM/PM times, you can use the following program segment to convert the hour and append "AM" or "PM" as appropriate:

```
HR    VAL(HR$)
IF HR>ll THEN AP$ = "PM"
IF HR<12 THEN AP$ = "AM"
IF HR=0 THEN HR = 12
IF HR>12 THEN HR = HR—12
```

There are a number of programs included on disk with your H.O. which read and display the clock. Try them and then analyze how they are programmed.

FINDING THE CLOCK SLOT BY SOFTWARE   (2)

Each switch—selected mode of the TIMEMASTER II has a unique firmware "signature" in ROM. Some of the signature bytes are embedded in certain locations for compatibility with commercial software written for different clock cards. For example, special effort was made to make the APPLE CLOCK mode appear to software as though you have the Mountain Hardware card.

As well, a simple to use signature was added to distinguish the Applied Engineering card from other brands, and to distinguish the various switch—selected modes.

The following subroutine will search slots 1 through 7 for a TIMEMASTER II H.O. card. If no H.O. is found, the subroutine will return with SLOT 0. If found, SLOT will equal the slot number, and AP will equal a 1 or 3 indicating which switch—selected mode is set.

```
1000 REM FIND TIMEMASTER II H.O. SLOT
1010 SLOT = 0
1020 FOR I = 1 TO 7
1030 ADDR = 12*4096 + 1*256: REM $CS0O
1040 IF PEEK(ADDR) = 8 AND PEEK (ADDR + 1) = 120 AND PEEK
     (ADDR + 254) = 178 THEN SLOT    I : I = 7
1050 NEXT
1060 IF SLOT = 0 THEN RETURN
1070 AP = PEEK(ADDP+255)
1080 PRINT "SLOT = ";SLOT;" AP = ";AP
```

The values of AP can be interpreted easily:

AP	MODE	AP	MODE
1	APPLECLOCK	3	TIMEMASTER

Before we begin our discussion of millisecond time, we remind the
user not to go through the extra steps required for millisecond
accuracy if the events being measured or program lengths will not
allow you to realize this accuracy in a true sense. To quote an
old and wise lab technician, "Don't measure it with a micrometer,
mark it with chalk, then cut it with an ax." But if it's
milliseconds you want, a few additional commands will enable
milliseconds on the H.O. If you want to know the time to the
nearest millisecond, please read on; however, most people are
really trying to measure the elapsed time in milliseconds between
two events. If this is what you're trying to do, please see the
program on the Timemaster II H.O. disk called Millisecond Timer.

Any use of the H.O. involving milliseconds requires the use of
interrupts.   DOS 3.3 has an inconsistency in that it can
sometimes cause interrupt software to crash. This particular bug
has been discussed in several Apple magazines.   Applied
Engineering has written a program called PATCH DOS 3.3 FOR
INTERRUPTS (found on Timemaster II H.O. disk) . This program
will fix DOS 3.3 to work with interrupts. After running this
program, you can INITialize as many disks as you'd like. Always
run the PATCH DOS 3.3 FOR INTERRUPTS before using any millisecond
or interrupt capabilities or use a boot disk that has the
modification already made to it. Your Timemaster II H.O. disk is
such a disk so after booting up on the Timemaster II H.O. disk,
you can simply INITialize new disks. Commercial programs that use
interrupts make this patch automatically. The use of interrupts
with PRODOS are extremely complex and therefore we recommend the
use of DOS 3.3 until the many bugs in PRODOS are worked out.

The millisecond feature is enabled by printing a "." (period) to
the clock card after selecting the format. (Format selection
clears the interrupt selection.) The firmware sets up the IRQ
vector at $3FE and $3FF to point to an interrupt handler in ROM,
and sets a flag that will cause interrupts on the clock card to
be set up. One additional step has to be done from your program,
which is to enable the IRQ interrupt with a CLI instruction. In
Applesoft, you can issue a CLI instruction by POKEing a CLI and
RTS, and then calling them, as in line 100 below. The following
program shows how; it assumes your clock card is in slot 4:

```
100  POKE 768,88: POKE 769,96: CALL 768      set up interrupts
110  TEXT: HOME                              good programing
120  PRINT "PRESS A KEY TO END"              ref to line 200
130  SLOT = 4                                or any slot
                                             set clock input
140  PRINT CHR$ (4)  "IN#";SLOT              set clock output
150  PRINT CHR$ (4)  "PR#";SLOT              get the time
160  INPUT  ":.";A$
170  PRINT CHR$ (4) "IN#0"                   restore keyboard
                                             restore output
180  PRINT CHR$ (4) "PR#0"                   print the time
190  VTAB 4: PRINT A$
200  IF PEEK ( −16384) < 128 THEN 120        key pressed??
                                             reset key ck.
210  POKE ─16368,0
```

The Applied Engineering BSR Remote Control option allows your
Apple to send control signals to your BSR ultrasonic command
console. The command console then sends these signals along your
existing 120 volt AC wiring. To remotely control appliances or
lights, or almost any electrical device, you plug into a BSR
remote module. The BSR interface is designed to operate with the
BSR/x—l0 ultrasonic command console, model UC30l or model GDP—
1510 from Heathkit (NOT a kit) This is the same command console
that operates with the BSR/x—10 cordless controller. Be sure you
have the right model of command console or it will not be able to
hear the signals from your Timemaster II H.O. BSR interface.
The command consol sold by Radio Shack is not compatible with the
BSR option because it has no ultrasonic receiver.

You will find that the command console has 22 command buttons.
Sixteen for setting devices 1 through 16 and 6 function buttons.
Your BSR interface can simulate the pushing of all 22 command
buttons. Remote modules can be purchased in two types; an
appliance module, which simply turns things on and off and can
handle heavy loads. The second type is a lighting module, which
comes in several styles, a plug in lamp type, wall mount single
pole, and 3—way style that fit neatly where your old light switch
was.

Please read the owners manual that comes with the BSR command
console and remote modules before using the Timemaster II H.O.
BSR interface. it is also a good idea to experiment a little bit
with the command console and a few remote modules.

To connect the BSR interface to your H.O., slip the interface
connector over the lower 4 pin plug, taking care that the two
wires going into the connector go over the lowest two pins of
this connector. Please refer to the drawing below.
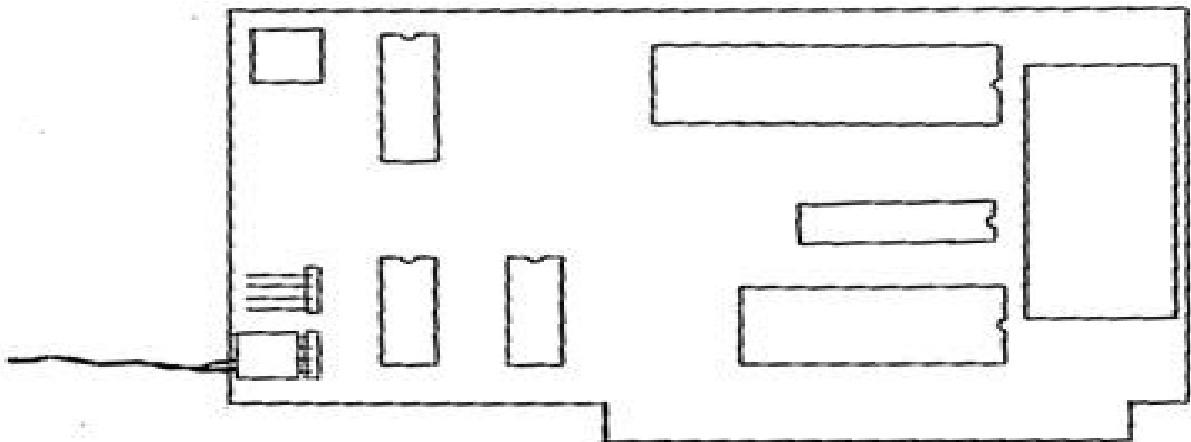
The transducer itself should be placed near enough to the controller to operate. This can be as far away as 6 or 8 feet if the transducer is aimed directly at the pickup inside the controller box. After installing the H.O. with the BSR interface attached, you can type in the program below to test the positioning of the transducer. When the program is running, you should see the red light on the controller flashing on and off.

```
100   PR#4              (or whatever slot)
110   PRINT "U"
120   GOTO 110          (press CTRL RESET to stop)
```

The H.O. firmware includes the ability to send signals to the BSR system. After selecting the H.O. with PR~s C (s is the slot) , you can print code letters to cause the various commands to be sent. The code letters correspond to the command console buttons as follows:

CODE LETTER	COMMAND BUTTON	CODE LETTER	COMMAND BUTTON
A	1	L	12
B	2	M	13
C	3	N	14
D	4	0	15
E	5	P	16
F	6	Q	ON
G	7	R	OFF
H	8	S	BRIGHT
I	9	T	DIM
J	10	U	ALL LIGHTS ON
K	11	V	ALL OFF

For example, to turn on light #1, do this: (2)

```
100 PRINT CHR$(4)"PR#4" (or whatever slot)
110   PRINT "AQ"
120   PRINT CHR$(4) "PR#0"
```

The "A" is button "1" and "Q" is "ON". Each command code is automatically transmitted 5 times to be certain it is correctly understood by the BSR controller. This is equivalent to holding down the button on the command console for about 1/2 second.

The BRIGHT and DIM commands are special. The range from fully OFF to fully ON is divided into many small steps. If you press and hold down the DIM button, it takes about 5 seconds to run through enough steps to fully dim the selected light; correspondingly, 5 seconds of BRIGHT will bring it back to fully on. The firmware simulates holding down the button, and you have the option of telling it how long to "hold it down". For the following sample program, you will need a light connected to button #1. Use the BSR controller manually and push (1) then

12

(ON). If the light does not come on, it may be necessary to hold down the BRIGHT button. Now type in the following program:

```
10 PRINT CHR$ (4)"PR#4"
20 PRINT "ATTTTTTTTTTTTT"
30 PRINT CHR$ (4)"PR#o"
```
The "A" is for button "1" and the "T" means "DIM", the more "T",s

the more dimming you'll get. Now change line 20 to:

```
20 PRINT "ASSSSSSSSSSSSSS"
```

Now reRUN the program and notice that the light will brighten. A shorthand way of controlling the number of DIM or BRIGHT commands issued each time a "T" or "5" is printed is available. By printing a **"i"** followed by a letter "A" through "Z" to select the number of times the DIM or BRIGHT command will be transmitted for each "T" (DIM) or "5" (BRIGHT), "A" for a little, "Z" for a lot. So let's change line 20 in the program above to use the "*" command to dim the light on button "1". Now change line 20 to:

```
20 PRINT "A*NTII
```

Now reRUN the program, by changing the "N" to other letters you can vary how long the "BUTTON" is pressed. When the H.O. sees the "i" it knows that the next character, "N", is the duration code for the following DIM ("Tin') or BRIGHT ("S") commands. The letters "A" thru "Z" used here for duration control should not be confused with the letters "A" thru "V'1 used as button commands.

To use the next sample program, you will need a lamp on "1" with a lamp module and a appliance module on "2". Plug both remotes into wall outlets nearby. If you have small radio to plug into the appliance module, it would help you know when the module is turned on. Be sure your COMMAND CONSOLE is plugged in. To be sure everything is set up properly, press button "1" then press "ON" on the COMMAND CONSOLE. Then press "2", "ON" on the COMMAND CONSOLE. Now press "1", "OFF", "2", "OFF". The light and the radio should have turned on and then off. If not, check to see if you have things set up properly and try again.

Now enter the program below that will allow us to exercise the H.O.'s BSR capability.

```
90 INPUT "WHAT SLOT IS THE H.O. IN ";S
100 INPUT "COMMAND =
110 IF B$ = "" THEN END
120 PRINT CHR$ (4) "PR#";S
130 PRINT B$
140 PRINT CHR$ (4) "PR#0"
150 GOTO 100
```

When we RUN this program using the commands shown, we get the following results:

```
    RUN
    COMMAND = AQ                    turns on the light
    COMMAND = AR                    turns the light off
    COMMAND = BQ                    turns on the radio
    COMMAND = BR                    turns off the radio
    COMMAND = AQTTTTTTTTTT          turns light on then dims it
    COMMAND = ASSSSSSSSSSS          brightens the light
    COMMAND = AR                    turns off the light
    COMMAND = AQBQ                  turns on the light and radio
    COMMAND = ARBR                  turns them both off
```

And now what you've all been waiting for, a program that will send commands to your REMOTE MODULES at specific times using the clock function of the H.O. This program will dim and brighten a light connected to REMOTE MODULE "1" every 30 seconds.

```
 10  TEXT: HOME                        just good programing
 20  SLOT = 4                          or any slot
 30  PRINT CHR$ (4)"IN#";SLOT          set input to clock
 40  PRINT CHR$ (4)"PR#";SLOT          set output to clock
 50  INPUT ":";A$                      get time from clock
 60  PRINT CHR$ (4)"IN#0"              restore keyboard
 70  PRINT CHR$ (4)"PR#0"              restore output
 80  SEC$ = RIGHT$ (A$,2)             throw away all but sec.
 90  VTAB 4: PRINT "SEC. = ";SEC$     display seconds
100  PRINT CHR$ (4)"PR#";SLOT          get ready to send code
110  IF SEC$ = "00" THEN PRINT "A*RT"  dim if sec 00
120  IF SEC$ = "15" THEN PRINT "A*RS"  brighten if sec = 15
130  IF SEC$ = "30" THEN PRINT "A*RT"  dim if sec = 30
140  IF SEC$ = "45" THEN PRINT "A*RS"  brighten if sec 45
150  PRINT CHR$ (4)"PR#0"              restore keyboard
160  GOTO 30                           get new time
```

BSR set ups can vary from the simple to the complicated. For example, you could have two H.O.'s in your Apple that went to two different COMMAND CONSOLES, each with its own separate house code. This would give you control over 32 different devices. It is also possible to have one COMMAND CONSOLE in the study with your computer, and another COMMAND CONSOLE in the bedroom, set to the same house code. Each COMMAND CONSOLE would control the same devices.

TIME BASE CALIBRATION   (1)

Your H.O. has a quartz crystal time base which oscillates at 32,768 HZ.   This frequency can be adjusted up or down approximately 2 HZ by the trimmer capacitor which is next to the DIP switches at the rear of the board.   Your H.O. was calibrated at the factory to 32,768.0 HZ. +/— 0.002%. The manufacturer of the crystal specifies that the frequency may age up to 0.005% or 5 parts per million in one year. If the clock is consistently gaining or losing time, you may wish to adjust the trimmer. Using a small screwdriver, turn the trimmer SLIGHTLY clockwise to speed up the clock, or counter clockwise to slow it down.

+1— 30 SECOND ADJUST   (2)

Your H.O. has the ability to adjust the seconds +— 30. By momentarily connecting the two top pins on the upper four pin connector marked "5V" and marked "ADJ" (for adjust) you can reset the seconds to 00, adding one minute if the seconds were greater than 30. It should be reminded that the set clock routine on the Timemaster II H.O. disk can set the clock as accurately as you can push the return key ; however , in some scientific applications, it may be necessary to trim the time more accurately using the +— 30 second adjust feature.

THE BATTERY   (1)

The H.O. is supplied with a rechargeable Ni—Cad battery to keep the clock running when the computer is turned off, or when power fails. The H.O. will automatically detect power going off and switch to the on—board battery. The clock will continue keeping time while the on-board battery supplies power. The battery is charging whenever power is applied to the APPLE. The time to fully charge a discharged battery is approximately 10 hours. Your H.O. may be used while charging the battery. A fully charged battery can keep the clock running for 5 to 7 months. In order to keep the battery charged, your APPLE should be turned on at least 0.5% of the time or about 1 hour a week or 4 hours a month. The battery CANNOT be overcharged and under normal use it will not require attention.

TIME TO FULLY CHARGE BATTERY (IN HOURS)

15
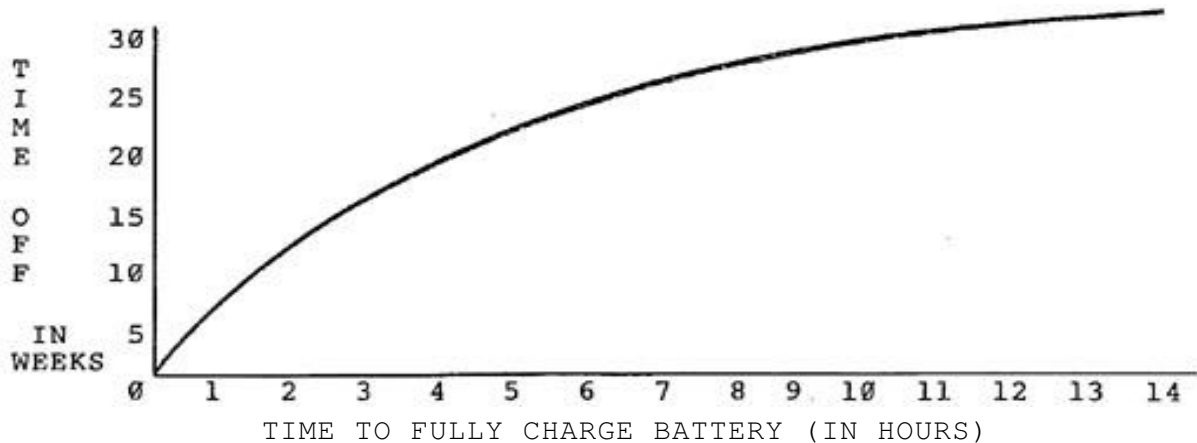
One nice application of the H.O. is to automatically record the current date and time on file names in the catalog when you SAVE, BSAVE, or RENAME files. This feature is found in many large computer systems, as well as Apple pascal, sos, and PRO DOS.

Some small but significant patches must be made to DOS 3.3 to allow automatic time/date stamping of your files. First, a routine to read the clock and overlay the time/date string on your file name must be called for every SAVE, BSAVE, or RENAME command. Second, the two places inside DOS 3.3 where file names are compared must be modified to ignore the right end of the names (where the time/date string may or may not be found).

A program on the TIMEMASTER II H.O. disk called INSTALL DOS DATER will automatically make these patches for you. This program allows you to specify what portion of the time/date string you want to use in stamping file names. The DOS DATER patches require that the H.O. be in the Timemaster mode (switch 2 open) . The Timemaster format must be used because other clock cards contain no year information. INSTALL DOS DATER searches slots 1 through 7 to find your 1-1.0. and checks that you have the right mode selected.

You can select any contiguous portion of the formatted date and tIme:

    W MO/DD/YY HH:MI:SS

The most commonly used stamp is the MO/DD/YY portion, with the leading space.

Using INSTALL DOS DATER, you can select the portion of the string you wish to use with the arrow keys and the "12' and "R" keys. When the string is as you like it, press RETURN and the patches will be recorded. Simply follow the directions in the program to modify as many DOS 3.3 disks as you'd like. Once the patches have been applied to the DOS booted, any disks you initialize with the INIT command will have the patched DOS on them.

After you become familiar with the various features of time/date stamping your files, you may wish to update the DOS on the older disks. The slot your H.O. was in at the time INSTALL DOS DATER was executed is frozen into the installed patches, so do not try to move the card to another slot! If you do move your H.O. to a different slot, you will have to reRUN INSTALL DOS DATER.

There is normally no problem with using the patched DOS with any existing programs on any disks as long as they are not copy protected. Any new files SAVEd, BSAVEd, or RENAMEd will have the current date added. The INIT command will also cause the date to be added to the greeting prOgram's name. Should you ever be running with an unpatched DOS and want to use files that have time/date stamps, you will probably want to temporarily patch the

DOS in memory so that it ignores the time/date stamp. (You don't have to; but if you don't,you will have to type all 30 characters of the file name, which includes the time/date stamp.)

There are two places that file names are compared inside DOS 3.3, and both of them must be patched to shorten the EFFECTIVE file name. The effective file name is 30 characters long, minus the length of your time stamp. If you are using the recommended stamp of " MO/DD/YY", your effective file name is 30—9, or 21 characters. You need the following POKES:
```
     POKE —19965,21       in the catalog search routine
     POKE —22653,20       in the OPEN command handler
```

(Note that the second POKE puts a value one less than the effective length into DOS.)

AUTOMATIC TIME/DATE STAMPING FILES IN PRODOS (1)

PRO-DOS will automatically time and date stamp files with the H.O. in the TIMEMASTER mode (switch 2 open). PRODOS goes one step further than the patches for DOS 3.3, in that the date and time of original creation of a file is kept along with the date and time of last modification.

A word of caution should be mentioned regarding date stamping of files in PRODOS. PRODOS was written for an older design clock made by another company. This clock does not have any year information so PRODOS calculates the year based on the date and day of week. In other words, if it's the 16th and it's Monday, it must be 1984? So the TIMEMASTER II H.O. had to take a small step backwards to be compatible with Apple's new PRODOS. The PRODOS look up table goes from 1981 to 1987. After 1987, it will restart again with 1981 Even though your H.O. will show the correct year of 1988, PRO-DOS will not read the year from the H.O. Apple Computer is expected to directly support the H.O. in future releases of PRODOS. Until then, you will have to live with the same limitations that the buyers of other clock cards will be stuck with forever.

For those of you that wish to use the H.O. in a machine language program, keep reading. If this is not you, turn the page quickly!

The easiest way to read the H.O. from a machine language program is to call on the built-in firmware the same way a BASIC program does. The technique to use depends upon the switch—selected mode you have chosen.

If you have NOT chosen the TIMEMASTER mode (switch 2 open) then you can use the following short subroutine. There are four elementary steps involved:

```
        1)    save the current input hook ($38,$39)
        2)    set input hook to $Cs00(where s=slot of
        TIMEMASTER II
        3)    read the time, storing in a buffer
        4)    restore the original input hook
                1040  *----------------------------------
                1050  READ.TIME
0300— A5  38    1060          LDA $38      KSWL      PUT CURRENT
0312— 48        1070          PHA                    INPUT HOOK
0303— A5  39    1080          LDA $39      KSWH      ON THE STACK
0365— 48        1090          PHA
                1100  *----------------------------------
0306— A9  C4    1110          LDA #$C4     $CN   (N=SLOT CLOCK IS IN)
0308— 85  39    1120          STA $39      KSWH
030A— A2  00    1130          LDX #0
030C— 86  38    1140          STX $38      KSWL
                1150  *----------------------------------
030E— 20  18 FD 1160  .1      JSR $FD18    MONITOR KEY INPUT ROUTINE
0311— 9D  00 02 1170          STA $200,X   SAVE CHARACTER FROM CLOCK
0314— E8        1180          INX
0315— C9  8D    1190          CMP #$8D     END OF INPUT?
0317— DO  F5    1200          BNE .1
                1210  *----------------------------------
0319— 68        1220          PLA          RESTORE INPUT HOOK
031A— 85  39    1230          STA $39      KSWH
031C— 68        1240          PLA
0310— 85  38    1250          STA $38      KSWL
031F— 60        1260          RTS
```

After the subroutine has executed, the time/date string is in the buffer at $200, ending with a carriage return.

The above subroutine is on your TIMEMASTER II H.O. disk. The source code is in the S.ML (OLD MODES) in the format for the S—C Macro Assembler. The object code is in B.ML (OLD MODES). An Applesoft BASIC program which loads B.ML (OLD MODES) and CALLS it is called ML (OLD MODES).

If you have chosen to use the TIMEMASTER mode, your job is even
easier. There are two direct entries in the TIMEMASTER mode
firmware which you can use from machine language. Their use is
illustrated in the following subroutine:

```
0300— A9 BA    1000         LDA #":"      SELECT TIME MASTER MODE
0302— 20 OB C4 1010         JSR $C4ØB     MODE SELECTION
0305— 20 08 C4 1020         JSR $C408     READ TIME TO $200
0305— 60       1030         RTS
```

If your H.O. is not in slot 4, modify lines 1010 and 1020
appropriately. The entry at $CsOB receives the format selection
character. The entry at $CsØ8 reads the clock and stores the
correctly formatted string at $200, terminated by a carriage
return.

If you want to write a machine language program which can read
the clock no matter how switch 2 is set, you can merely look at
$CsFF to see which mode the H.O. is in. The following program
does just that, and proceeds to read the clock according to its
mode:

```
           1040 *---------------------------
           1050 READ.TIME
0300— A9 C4       1060           LDA #$C4   (FILLEDIN BY CALLER IF
DIFFERENT)
0302— 8D 10 03 1070           STA SLOTl
0305— 8D 19 03 1080           STA SLOT2
0309— 8D LC 03 1090           STA SLOT3
030B— 8D 25 03 1100           STA SLOT4
               1101 *-----------------------
030E— AD FF C4 1110           LDA $C4FF TIMEMASTERII   WITH 5W3, 5W4
OFF?
0310—          1120  SLOT1 .EQ *—1        $CN BYTE
0311— C9 03    1130           CMP #$03
0313— DO 09    1140           BNE OLD.MODES NOT TIMEMASTER II MODE
0315— A9 BA    1150           LDA #":"     SELECT TIME MASTER MODE
0317— 20 OB C4 1160           JSR $C4OB    MODE  SELECTION
0319—          1170  SLOT2 .EQ *-l        $CN BYTE
031A— 20 08 C4 1180           JSR $C408    READ  TIME TO $200
031C—          1190  SLOT3 .EQ *-1
031D— 60       1200
               1210*-------------------------
               1220  OLD.MODES
031E— A5 38    1230           LDA $38    KSWL   PUT CURRENT
0320— 48       1240           PHA                 INPUT HOOK
0321— A5 39    1250           LDA $39    KSWH   ON THE STACK
0323— 48       1260           PHA
               1270*-------------------------
0324— A9 C4    1280           LDA #$C4     $CN  (N~SLOT CLOCK IS IN)
0325—          1290  SLOT4 .EQ *-1
0326— 85 39    1300           STA $39    KSWH
0328— A2 00    1310           LDX #0
032A— 86 38    1320           STX $38    KSWL
               1330 *-----------------------
```

```
032C— 20 18 FD 1340 .1      JSR $FD18   MONITOR KEY INPUT ROUTINE
032F— 9D 00 02 1350         STA $200,x  SAVE CHARACTER FROM CLOCK
0332— E8       1360         INX
0333— C9 8D    1370         CMP #$8D    END OF INPUT?
0335— DO F5    1370         BNE .1
               1390  *-------------------------------
0337— 68       1400         PLA         RESTORE INPUT HOOK
0338— 85 39    1410         STA $39     KSWH
033A— 68       1420         PLA
033B— 85 38    1430         STA $38     KSWL
033D— 60       1440         RTS
```

The above subroutine is on your TIMEMASTER II H.O. disk. The
source code is in S.ML (ALL MODES) in the format for the S—C
Macro Assembler. The object code is in B.ML (ALL MODES). An
Applesoft BASIC program which loads B.ML (ALL MODES) and CALLS it
is called ML (ALL MODES).

        READING THE CLOCK WITHOUT USING THE ON—BOARD FIRMWARE (3)

Sometimes you may want to read the H.O. without using the
programs in the on—board ROM. This is not too difficult, and the
job can be broken down into the following tasks:

        1.   Initialize the PIA chip.
        2.   "Hold" the clock.
        3.   Address a digit.
        4.   Read the digit.
        5.   Repeat steps 3 and 4 until all relevant
                digits have been read.
        6.   "Release" the clock, and allow interrupts.

The following Applesoft program shows how it can be done. (The
program is on your H.O. disk.) The program is intended to be
instructive in nature, as Applesoft does not quite run fast
enough to update the time in precise one second intervals.

```
    10    DIM FMT(30)
    20    HOME : INPUT "SLOT:";SLOT: IF SLOT < 1 OR SLOT > 7
            THEN PRINT CHR$ (7);: GOTO 20
    40    GOSUB 2300: REM READ FORMAT DATA
    50    GOSUB 2000: REM INITIALIZE PIA
    60    GOSUB 2100: REM READ CLOCK
    70    VTAB 10: HTAB 10: PRINT T$
    80    IF PEEK ( — 16384) < 128 THEN 60
    90    POKE — 16368,0: END

    2000   REM SETUP PIA
    2010   PA = 49280 + SLOT * 16: REM $C080 + $NO
    2020   CA = PA + l:PB = PA + 2:CB PA + 3
    2030   IF PEEK (CB) < > 0 THEN 2060
            :REM PIA ALREADY SET UP
    2040   POKE CA,0: POKE CB,0: POKE PA,0: POKE PB,255
            :REM SET DIRECTION REGISTERS
```

```
2050    POKE CA,4: POKE CB,4: REM POINT AT DATA REGISTERS
2060    RETURN
2100    REM READ CLOCK USING FORMAT
2110    T$ = "": FOR I = 1 TO NC
2120    POKE PB,16: REM   HOLD CLOCK
2130    D = FMT(I): IF D > 127 THEN 2170
2140    M = 16: IF D = 56 OR D = 53 THEN M = 4
2150    POKE PB,D:T    PEEK (PA): REM READ THE DIGIT
2160    D = T — INT (T / M) * M + 176
2170    T$= T$ + CHR$ (D)
2180    NEXT
2190    POKE PB,47: RETURN
2300    REM "W MM/DD/YY HH:MM:SS" FORMAT
2310    DATA 54,160,58,57,175,56,55,175,60,59,160,
        53,52,186,51,50,186,49,48,0
2320    NC = 0
2330    READ D: IF D = 0 THEN RETURN
2340    NC = NC + 1:FMT(NC) = D: GOTO 2330
```

In the program above, the PIA chip is initialized by the
subroutine at line 2000. The subroutine at line 2300 reads a
format definition into the array ENT. The numbers in the list
are either clock digit addresses (values from 48 through 60) or
ACSII characters (values) 127):

	Clock Registers		ASCII Character
	Tens	Units	
	Digit	Digit	
Year	60	59	160 space
Month	58	57	175 = slash
Day	56	55	186 = colon
Day of week	none	54	0 = end of format
Hour	53	52	
Minute	51	50	
Second	49	48	

The subroutine at line 2100 reads the clock by stepping through
the format array. Line 2120 "holds" the clock, and line 2190
"releases" it.

The Applesoft program above is very similar in concept to the
programs in the on—board ROMs. By varying the format definition,
many different arrangements of the clock data can be displayed.

An assembly language program which reads the clock without using
on—board firmware is on the TIMEMASTER II H.O. disk. The source
code is in the file named S.TIME BY INTERRUPT, in the format
of the S—C Macro Assembler. The object code, which can be BRUN,
is in the file named B.TIME BY INTERRUPT.

INTERRUPTS (3)

One of the main features of the H.O. is the ability to generate
interrupts at set intervals. Interrupts can add new dimensions to
your Apple. For instance, background and foreground programming is
possible by letting the interrupt handler routine initiate the
background program. Also, data can be sampled at precise
intervals.

The following interrupt intervals are available: 1/1024 second, 1
second, 1 minute, or 1 hour. The interval is selected by storing
values from the table below in the control registers on the clock
card.

Time	Control	Register
	CRA	CRB
none	$04	$04
1024 Hz	$05	$04
1 Sec	$OC	$04
1 Min	$04	$05
1 Hour	$04	$0C

SAMPLE INTERRUPT PROGRAMS (3)

This and other details of interrupt handling are best done with
machine language routines. The source listing of a program that
uses interrupts is on the last page of this manual. The name of
this program is MILLISECONDS. Both source and object files are
on your H.O. disk, in S.MILLISECONDS and B.MILLISECONDS
respectively. The source file is in the format for the S—C Macro
Assembler.

There are several programs on your H.O. disk which load
B.MILLISECONDS and use the subroutines to illustrate the use of
interrupts. MILLISECONDS will display a running count of
milliseconds, using the 1024 Hertz interrupt. MILLISECOND
STOPWATCH will allow you to precisely time the interval between
two presses of the space bar, using your Apple as a stopwatch
with accuracy to the nearest millisecond. TIME LIMIT QUIZ
illustrates using interrupts to provide a time limit to some
activity by the user (in this case an interesting multiple choice
quiz)

It is interesting to listen to the Apple 'bell" while millisecond
interrupts are being processed. Try it, and you will hear what
happens when you slow down timed loops to process interrupts.

Note that the interrupt interval generated on the card is
1/1024th of a second, which is not a millisecond (it is about
0.977 milliseconds) Programs such as those above which need real
milliseconds must multiply the 1024 Hz counter value by
1000/1024.

The two files S.TIME BY INTERRUPT and B.TIME BY INTERRUPT are the source and object files for another program which uses interrupts. Again, the source file is in the format of the S-C Macro Assembler. If you BRUN B.TIME BY INTERRUPT, the top two lines of the screen display will be used to display the current date and time. The time will be updated every second.

H.O. INTERRUPT UTILITIES (3)

A binary file named B.CLOCK UTILITIES is on the H.O. User Disk. This file contains seven very useful programs which can be called from an Applesoft program to work when you want to use interrupts.

The file is designed to be BLOADed at $300, and uses all of the space from $300 through $3CF. Five bytes from $EB through $EF, not used by Applesoft or DOS, are used for communication between the UTILITIES and your Applesoft program. Include the following statement at the beginning of your Applesoft program:

        PRINT CHR$(4)"BLOAD B.CLOCK UTILITIES":CALL 785

The "CALL 785" searches from slot 7 down toward slot 1 looking for the H.O. Even if you know ahead of time which slot is being used, you still must CALL 785 to set up the utilities.

After "CALL 785", you can PEEK(235) for the slot number. If PEEK(235) is zero, no H.O. was found. PEEK(236) will be 3 if the H.O. is in the normal mode, or 1 in the Appleclock mode.

The other six subroutines deal with interrupts:
        CALL 768 -- Initialize interrupts
        CALL 771 -- Turn off interrupts
        CALL 774 -- Clear interrupt counter
        CALL 777 -- Read interrupt counter
        CALL 780 -- Enable interrupts
        CALL 783 -- Disable interrupts

CALL 768 -- Initialize interrupts: Four interrupt intervals are available. You select which one you want by POKE 237:

        POKE 237,0:CALL 768 1024 times per second
        POKE 237,1:CALL 768 once per second
        POKE 237,2:CALL 768 once per minute
        POKE 237,3:CALL 768 once per hour

CALL 768 puts the address of an interrupt handler into #3FE and 3FF, clears an interrupt counter, selects the interrupt interval you have specified, and enables interrupts with a CLI instruction.

CALL 771 -- Turn off interrupts: This CALL disables interrupts with an SEI instruction, and de-selects interrupts on the clock.

CALL 774 -- Clear interrupt counter: Zeroes the 32-bit counter.

CALL 777 —— Read interrupt counter: Disables interrupts, copies the interrupt 32—bit counter into 236, 237, 238, and 239 and re—enables interrupts. You may follow CALL 777 with CNT = PEEK(239) + 256 * PEEK(238) + 65536 * PEEK(237) + 16777216 * PEEK(236)

CALL 780 —— Enable interrupts: Enables interrupts with the CLI instruction, without changing the clock card setup.

CALL 783 —— Disable interrupts: Disables interrupts with the SEI instruction, without de—selecting clock card interrupts. Interrupts may later be re—enabled with CALL 780.

                      CLOCK UTILITIES SOURCE CODE (4)

```
                1010 *SAVE S.CLOCK UTILITIES
                1020 *-------------------------------
                1030           .OR $300
                1040           .TF B.CLOCK UTILITIES
EB—             1050 *-------------------------------
                1060 SLOT        .EQ $EB PEEK(235)
EC—             1070 MODE        .EQ $EC PEEK(236)
ED—             1080 INTERVAL    .EQ $ED POKE(237) 0=MS,1=SEC
                                              2=M1N,3=HR
EC—                 1090 COUNT.SAFE .EQ $EC — $EF PEEK (239)
                1100 *
                1110 *
                1120 *-------------------------------
 0300—4C 49  03 1130 CALL.768 JMP INTERRUPT.INIT
 0303—4C 9B  03 1140 CALL.77l JMP INTERRUPT.CLEAR
 0306—4C 83  03 1150 CALL.774 JMP INTERRUPT.ZERO
 0309—4C 8E  03 1160 CALL.777 JMP INTERROPT.READ
 030c—4C 99  03 1170 CALL.780 JMP INTERRUPT.ENABLE
 030F—78        1180 CALL.783 SEI          .DISABLE
 0310— 60 1190 RTS
 0311—          1200 CALL.785 .EQ * FIND SLOT
                1210 *-------------------------------
                1220 FIND.SLOT
 0311—AO C7     1230         LDY #$C7 START WITH SLOT 7
 0313—8C AA  03 1240 .1      STY GET.ROM.BYTE+2
 0316—AD FF  CF 1250         LDA $CFFF RELEASE $C800 SPACE
 0319—A2 FE     1260         LDX #$FE LOOK FOR $B2 AT $CNFE
 031B—20 A8  03 1270         JSR GET.ROM.BYTE
 031E—C9 B2     1280         CMP ~t$B2
 0320—DO 13     1290         BNE .2
 0322—20 A8  03 1300         JSR GET.ROM.BYTE MODE BYTE
 0325—85 EC     1310         STA MODE
 0327—20 A8  03 1320         JSR GET.ROM.BYTE
 032A—C9 08     1330         CMP #$08 "PHP" AT $CNOO
 032C—DO 07     1340         BNE .2 NOT THIS SLOT
 032E—20 A8  03 1350         JSR GET.ROM.BYTE
 0331—C9 78     1360         CMP #$78 "SEI" AT $CNO1
 0333—FO 07     1370         BEQ .3 THIS SLOT
 0335—88        1380 .2      DEY            NEXT SLOT DOWN
 0336—CO Cl     1390         CPY #$Cl
 0338—B0 D9     1400         BCS .1
```

```
033A— A000          1410          LDY #0
                    1420 *---STUFF SLOT $NO VALUE--------
033C— 98            1430 .3       TYA             $CN
033D— 29 OF         1440          AND #$OF        $ON
033F— 85 EB         1450          STA SLOT
0341— OA            1460          ASL
0342— OA            1470          ASL
0343— OA            1480          ASL
0344— 0A            1490          ASL
0345— 8D B0 03      1500          STA SLOT16     $NO
0348— 60            1510          RTS
                    1520 *--------------------------------
                    1530 INTERRUPT.INIT
0349— 78            1540
                    1550 *---LOAD INTERRUPT VECTOR---------
034A— A9 AD         1560          LDA #IRQ.HANDLER
034C— 8D FE 03      1570          STA $3FE
034F— A9 03         1580          LDA /IRQ.HANDLER
0351— 8D FF 03      1590          STA $3FF
                    1600 *---SET UP INTERRUPTIONS-----------
0354— AE BO 03      1610          LDX SLOT16
0357— A9 00         1620          LDA #0       POINT AT DIRECTION REGS
0359— 9D 81 CO      1630          STA $C081,x
035C— 9D 83 CO      1640          STA $C083,X
03SF— 9D 80 CO      1650          STA $C080,X    PORT A ALL INPUT
0362— A9 FF         1660          LDA #$FF
0364— 9D 82 CO      1670          STA $CO82,x    PORT B ALL OUTPUT
0367— AS ED         1680          LDA INTERVAL 0... 3
0369— 29 03         1690          AND #$03
036B— A8            1700          TAY
036C— B9 CA 03      1710          LDA INTERVAL.CRA,Y
036F— 9D 81 CO      1720          STA $C081,X   5=MS 12~SEC 4M1N 4=HR
0372— B9 CC 03      1730          LDA INTERVAL.CRB,Y
037S— 9D 83 CO      1740          STA $C083,X   4=MS 4=SEC 5=MIN 12=HR
0378— A9 2F         1750          LDA #$2F ENABLE INTERRUPTS OUT PIA
037A— 9D 82 CO      1760          STA $C082,X
037D— BD 80 CO      1770          LDA $C080,X   CLEAR PREVIOUS INTERRUPT
0380— BD 82 CO      1780          LDA $C082,X
                    1790 *----------------------------------
                    1800 INTERRUPT.ZERO
0383— 78            1810          SEI          SET INTERRUPT DISABLE
0384— A9 00         1820          LDA #0       ZERO THE 4 BYTE COUNTER
0386— A2 03         1830          LDX #3
0388— 9D C6 03      1840 .1       STA COUNT,X
038B— CA            1850          DEX          DECREMENT X
038C— 10 FA         1860          BPL .1
                    1870 *----------------------------------
                    1880 INTERRUPT.READ
038E— 78            1890          SEI          DISABLE INTERRUPTS WHILE
038F— A2 03         1900          LDX #3
0391— BD C6 03      1910 .1       LDA COUNT,X
0394— 95 BC         1920          STA COUNT.SAFE,X
0396— CA            1930          DEX          DECREMENT X
0397— 10 F8         1940          BPL .1
                    1950 *----------------------------------
```

```
                        1960   INTERRUPT.ENABLE
0399— 58                1970          CLI          ALLOW INTERRUPTS AGAIN
039A— 60                1980          RTS
                        1990   *-------------------------
                        2000   INTERREJPT.CLEAR
039B— 78                2010          SEI DISABLE INTERRUPTS
039C— AE B0 03          2020          LDX SLOT16
039F— A9 04             2030          LDA #04
03A1— 9D 81 CO          2040          STA $C081,X
03A4— 9D 83 CO          2050          STA $C083,X
03A7— 60                2060          RTS
                        2070   *-------------------------
                        2080   GET.ROM.BYTE
03A8— BD 00 C7          2090          LDA $C700,x
O3AB— E8                2100          INX
O3AC— 60                2110          RTS
                        2120   *-------------------------
                        2130   IRQ.HANDLER
O3AD— 8A                2140          TXA          SAVE X-REG
O3AE— 48                2150          PHA
O3AF— A2  BO            2160          LDX #SLOT16 $NO
03B0—                   2170   SLOT16 .EQ ~—1
03B1— BD 80 CO          2180          LDA $C080,X CLEAR INTERRUPT
03B4— BD 82 CO          2190          LDA $C082,X
03B7— A2 03             2200          LDX #3
03B9— FE C6 03          2210   .1     INC COUNT,X
03BC— DO 03             2220          BNE .2
03BE— CA                2230          DEX
03BF— 10 F8             2240          BPL .1
03C1— 68                2250   .2     PLA          RESTORE X AND A REGS
03C2— AA                2260          TAX
03C3— AS 45             2270          LDA $45      GET SAVED A REG
03C5— 40                2280          RTI          RETURN
                        2290   *-------------------------
03C6—                   2300   COUNT     .BS 4
                        2310   *-------------------------
O3CA— 05 OC             2320   INTERVAL.CRA .DA #5,#12
03CC— 04 04 05
O3CF— OC                2330   INTERVAL.CRB .DA #4,#4,#5,#12
                        2340   * -------------------------
0300— CALL 768                          0399— INTERRUPT.ENABLE
0303— CALL 771                          0349— INTERRUPT.INIT
0306— CALL 774                          038E— INTERRUPT.READ
0309— CALL 777                                 .01=0391
030C— CALL 780                          0383— INTERREJPT.ZERO
030F— CALL 783                                 .01=0388
0311— CALL 785                            ED— INTERVAL
03C6— COUNT                             03CA— INTERVAL.CRA
  EC— COUNT.SAVE                        03CC— INTERVAL.CRB
0311— FIND.SLOT                         O3AD— IRQ.I—IANDLER
     .01=0313 .02=0335 .03=033C           SC— MODE
03A8— GET.ROM.BYTE                        SB— SLOT
039B— INTSRRUPT.CLSAR                   03B0— SLOT16
                                               .01=03139 .02=03C1
```

CONTROLLING INTERRUPTS WITHOUT MACHINE LANGUAGE (3)

In order to illustrate how to program interrupts using Applesoft, without separate machine language files, the following program was written by Bill Goodwill. Bill's program sets up the clock to interrupt once per second. Three tiny machine language programs are POKEd into memory: at $300, a routine which the IRQ interrupt; and at CALL 782 a program to enable the IRQ interrupt. At each interrupt the program at $300 stores a non—zero value at 779, so that the Applesoft program will know an interrupt occurred.

Bill's Applesoft program is in a tight loop waiting for a non—zero value (line 2400) . For grins, a period is printed every trip through this loop. When PEEK(77) is finally non—zero, Bill zeroes it and counts the event in C, and prints a slash, for grins again. When C finally reaches the limit T, the program reads the clock and displays it on the screen.

You could substitute your own list of activities to be performed after each interrupt, and after each time the counter C reaches the limit T.

PATCHES FOR DOS 3.3 WHEN USING INTERRUPTS (3)

The Apple and DOS designers must not have really expected users to take advantage of interrupts. The IRQ interrupt handler inside the Apple monitor ROM saves the A—register at location $45 in RAM. This would be all right, except that DOS 3.3 uses location $45 in 25 different places as a temporary variable. An interrupt at the wrong time could greatly confuse DOS, and has the potential of clobbering a disk or at least a running program.

The easiest way around the problem is to be certain that the IRQ interrupt is disabled before any/every DOS command, and enabled when the DOS commands are finished. Another approach (tried in a product called Doubletime Printer from Southwestern Data Systems) is to replace the monitor ROM chip with one which does not use location $45. The best solution is to patch DOS 3.3 50 that it does not use location $45.

The program called PATCH DOS 3.3 FOR INTERRUPTS on the H.O. disk installs patches which make DOS 3.3 completely compatible with the use of the IRQ interrupt.

If you are expecting to use the millisecond interrupt for precise interval counting or other purposes, you may still have a conflict with DOS. Because disk I/O operations are critically timed with software, DOS 3.3 disables the IRQ interrupt during the reading or writing of the disk data. The IRQ interrupt will be ignored during the time it takes to bring a drive up to speed, find the proper track and sector, and read or write the data.

HOW TO READ THE TIME/DATE UNDER AN INTERRUPT (3)

It is usually not a good idea to try to use the built-in firmware to read the time and date in an interrupt driven program. The H.O. firmware uses the system input buffer at $200 to build the time/date string.

It is possible that your use of the clock interrupt does not require all of the clock data. Perhaps only certain clock registers need to be read, so that a separate machine language program could be much faster than an attempt to read the whole clock using the on-board firmware. To keep interrupt overhead to a minimum your interrupt routine could read only selected registers.

The on-board firmware "forgets" your interrupt interval selection in most cases. If you don't want to reestablish the interrupt interval after each clock read, you may want to use a separate clock read program.

A program is on your H.O. disk which reads the clock and formats the date and time. The source code is in the file named S.TIME BY INTERRUPT, in the format of the S-C Macro Assembler. The object code, which can be BRUN, is in B.TIME BY INTERRUPT.

HOW TO DISABLE INTERRUPTS (4)

Once you begin using interrupts, it is important to know how to disable them. The IRQ interrupt can be disabled using the SEI instruction from machine language programs.

Pressing RESET (or Ctrl-RESET in newer Apples) will disable the IRQ interrupt. RESET will also clear the interrupt selections in the clock card, so that both IRO and NMI interrupts will cease.

Both IRQ and NMI interrupts from the clock can be cleared and disabled by storing a zero value in the control registers ($C081+NO AND ~C083+N0).

Since the clock interrupts must be connected to the Apple bus by switches 3 or 4 on the clock card, flipping the switches off will obviously disconnect the interrupts.

```
1990 REM BASIC INTERRUPT ROUTINE BY WILLIAM P. GOODWILL
2000 TEXT : HOME : INPUT "WHAT SLOT IS THE TIMEMASTER IN?
      ";SLOT: IF SLOT < 1 OR SLOT > 7 THEN 2000
2010 REM **LOAD 3 SMALL ASSY LANGUAGE ROUTINES**
2020 REM **;COME HERE UPON INTERRUPT
2030 REM **768 LDA #01 ;SET FLAG FOR BASIC
2040 REM **       STA 779
2050 REM **       LDA PIA ;CLEAR PIA
2060 REM **       LDA ~45 ;RESTORE ACCUMULATOR
2070 REM **       RTI     ;THEN RETURN
2080 REM **779 BYTE 00    ;BASIC FLAG
2090 REM **;USE "CALL 780" TO DISABLE INTERRUPTS
```

```
2100 REM **780 SET
2110 REM **RTS
2120 REM **;USE "CALL 782" TO ENABLE INTERRUPTS
2130 REM **782 CLI
2140 REM ** RTS
2150 DATA 169, 1,141,11,3,173,208,192,165,69,64,0,120,96,88,96
2160 FOR A = 768 TO 783: READ X: POKE A,X: NEXT A
2170 CALL 780: REM TURN OFF INTERRUPTS
2180 POKE 1022,0: POKE 1023,3: REM POINT TO INTERRUPT HANDLER
2190 A = -16256 +16 *SLOT: REM PTA BASE ADDRESS = $C080+$NO
2210 POKE 774,128 +SLOT *16
2220 POKE A +1,0: POKE A +3,0: REM DATA DIRECTION REGS
2240 POKE A,0: REM PORT A INPUT,8 BITS
2250 POKE A +2,255: REM PORT B OUTPUT,8 BITS
2260 REM THE NEXT 2 POKES DETERMINE INTERRUPT RATE
2270 REM (THIS BASIC PROGRAM IS TOO SLOW FOR 1024 HZ INTERRUPTS)
2280 POKE A +1,12: REM CONTROL REGISTER A (1 PER SECOND)
2290 POKE A +3,4: REM CONTROL REGISTER B
2300 POKE A +2,47: REM TELL PTA TO PASS INTERRUPTS
2310 X = PEEK (A): REM CLEAR PTA FLAGS
2320 F = 779: REM BASIC FLAG ADDRESS
2330 Z = 0: REM FLAG VALUE WHEN CLEAR
2340 CALL 782: REM     ENABLE INTERRUPTS
2350 T = 3: REM NUMBER OF INTERRUPTS PER CYCLE
2360 ONERR GOTO 2380
2370 GOTO 2460: REM TWIT
2380 CALL 780: STOP :REM DISABLE INTERRUPTS BEFORE HALTING
2400 IF PEEK (F) Z THEN PRINT ".";:GOTO 2400:REM WAIT FOR TNT
2410 PRINT "/";:REM DO THE FOLLOWING ON EVERY INTERRUPT
2420 POKE F,Z: REM     RESTORE FLAG AFTER INTERRUPT
2430 C = C +1: IF C<T THEN GOTO 2400 REM TNCR CTR,COMPARE TO MAX
2450 REM PERFORM THIS OPERATION WHENEVER COUNT REACHES MAX
2460 C = 0: REM INIT COUNTER
2470 PRINT
2485 D$ = CHR~ (4): PRINT
2490 PRINT D$"IN#SLOT": PRINT D$"PR#"SLOT
2500 VTAB 5: INPUT ":";K$
2510 PRINT D$"IN#0": PRINT D$"PR#0"
2520 PRINT K$
2530 CALL -958
2540 POKE A +1,12: REM CONTROL REGISTER A (1 PER SECOND)
2550 POKE A +3,4: REM CONTROL REGISTER B
2560 GOTO 2400
```

The TIMEMASTER II H.O. may be used under the CP/M operating system with the appropriate software. When using MBASIC or GEASIC your H.O. can be used m ost easily by POKEing a short interface routine. A sample program that shows how this is done may be found on the H.O. disk. The program is called T.CPMDE1 and is stored as a text file which can be converted to the CP/M system by using the APDOS utility on the CP/M master disk. This program was written for the TIMEMASTER mode.

To convert the APPLE DOS text file to a CP/M file, do the following:

1)   Put a copy of your H.O. disk in drive 2 (B).

2)   Put a CP/M disk wi-th BASIC and APDOS on it in drive 1 (A)

3)   Turn on computer or do a PR#6 (RETURN> SO YOU GET AN A>.

4)   Type APDOS <RETURN> YOU SHOULD NOW HAVE AN *.

5)   Type TMl.BAS~T.CPMDE1 <RETURN> , this will convert T.CPMDEl.

6)   Type <control> C , this will put you back in CP/M.

You should now have a program on your CP/M disk, called TMl
This program can be run from MBASIC or GBASIC.

In Pascal, an intrinsic unit has been supplied to enable the access of the H.O. from within a Pascal program. The unit is already linked into a standard Pascal library and can be accessed with the statement:

Uses TIMEMASTER

in a programs heading.

The unit supplies the following functions and procedures:

PROCEDURE GETCLOCK;

FUNCTION CLDAY;
FUNCTION CLMONTH;
FUNCTION CLDATE;
FUNCTION CLYEAR;
FUNCTION CLHOURS;
FUNCTION CLMINUTES;
FUNCTION CLSECONDS;

AND THE VARIABLES
TIME   STRING   13
TODAY   STRING   18

The functions listed return the integer value of the function name. Note that the CLDAY is the day of week and the CLHOURS is the hours 0 to 23 format. These return the same information that is returned when using the input statement in BASIC.

The procedure GETCLOCK sets the strings time and TODAY with the current time and date. The time string returns the time in the following format:

    12:30:08 PM

while the TODAY string is in the format of

    SUN MAR 20, 1983

and both have leading and trailing blanks.

To use these utilities, the system library on your boot disk will need to be replaced with the system library on the suppied disk.

The file system start up will also need to be included on your boot disk if the automatic dating feature is desired.

The three text files supplied are the programs that were used to make the TIMEMASTER II unit.

Your TIMEMASTER II H.O. disk contains many useful and instructive example programs. Some of these have been mentioned in the preceding pages. The disk is not protected (however it is copyrighted) and you can and should make a backup copy using any standard Apple disk copy program (for example, COPYA on the DOS 3.3 System Master Diskette).

Side one of the diskette, which is labeled, contains DOS 3.3 (Copyright 1981 by Apple Computer, Inc.). A small patch has been made to this DOS, to speed up the LOAD, BLOAD, RUN, and BRUN commands.

The programs on this side include a file which can be moved to a CP/M environment. The others can be used either with DOS 3.3, and many of them with Apple PRO DOS. Side two of the diskette contains files intended to be used with Apple Pascal.

There are seveal files on side one of the H.O. diskette which are type thu. Normally file type I signifies Integer BASIC files, but in this case it signifies source assembly language files to be used with the S—C Macro Assembler. You may also have other programs on the H.O. disk.

S.MILLISECONDS   Source file (S—C Macro Assembler) of
                          B.MILLISECONDS

B.MILLISECONDS   Object file used by next three programs
                          which sets up and
                          processes millisecond interrupts.

MILLISECONDS   Uses B.MILLISECONDS to give a running
                          display of
                          elapsed milliseconds.

MILLISECOND STOPWATCH   Uses B.MILLISECONDS to provide a very
                          precise stopwatch.

THE LIMIT QUIZ   Uses B.MILLISECONDS to give a fun
                          multiple choice quiz with time limits
                          on your responses.

S.TIME BY INTERRUPT   Source file (S—C Macro Assembler) of
                          B.TIME BY INTERRUPT

B.TIME BY INTERRUPT   BRUN this to put current date/time
                          on top line of screen. It will stay
                          there while you work, and be updated
                          every second!

BASIC INTERRUPT ROUTINE   Bill Goodwill's program which sets up
                          interrupts and demonstrates them
                          entirely within Applesoft.

PATCH DOS 3.3 FOR INTERRUPTS

Installs patches which make DOS 3.3 safe for interrupts.   (See text)

SET CLOCK   The program to use when you need to set a new time and/or date in your TIMEMASTER II.

EXAMPLE HOOKS   Three different ways to hook in the TIMEMASTER II and read it.

DATE & TIME DISPLAY   Nicely displays date and time, updating every second.

TIME II DEMO   Reads the clock and displays date and time in all available formats according to selected mode.

ANALOG CLOCK   Displays an old fashioned clock with moving hands and an audible "tick—tock".

FIND TIMEMASTER SLOT & MODE

Program which finds the slot and mode of your TIMEMASTER II.

READ TIME (NO ROM)   Program which reads and displays the date and time without using the on—board ROM firmware.

ML (ALL MODES)   Calls B.ML (ALL MODES) to read the date/time string regardless of the mode selected by 5W3 and SW4.

S.ML (ALL MODES)   Source file (S—C Macro Assembler) of B.ML (ALL MODES), a program which reads the date/ time string in assembly language.

B.ML (ALL MODES)   Object file used by ML (ALL MODES).

ML (OLD MODES)   Calls B.ML (OLD MODES) to read the date/time string, which works only when NOT in TIMEMASTER II mode.

S.ML (OLD MODES)   Source file (S-C Macro Assembler) of B.ML (OLD MODES), a program which reads the date/ time string in assembly languages.

B.ML (OLD MODES)   Object file used by ML (OLD MODES).

INSTALL DOS DATER   Modifies DOS 3.3 in memory to stamp date and/or time on files with SAVE, BSAVE, RENAME, or INIT.

```
S.DOS DATER               Source file (S-C Macro Assembler)

B.DOS DATER               Object file used by INSTALL DOS DATER
TALKING CLOCK             If you have an Echo II, this program
                          will speak the time once per minute.

TALK                      Used by TALKING CLOCK, do  not   BRUN

T.CPMDE1                  CP/M demo
```

The files which can be moved to PRO DOS (using CONVERT on the
     PRO DOS Users Disk) are:

```
     B.MILLISECONDS              ANALOG CLOCK
     MILLISECONDS                FIND TIMEMASTER II SLOT & MODE
     MILLISECOND STOPWATCH       READ TIME (NO ROM)
     TIME LIMIT QUIZ             ML (ALL MODES)
     BASIC INTERRUPT ROUTINE     B.ML (ALL MODES)
     SET CLOCK                   ML (OLD MODES)
     EXAMPLE HOOKS               B.ML (OLD MODES)
     DATE & TIME DISPLAY         TIME II DEMO
```

The file names will be changed by CONVERT so that non-letters
become periods, and truncated to 15 characters. With two
exceptions, these programs will function under PRO DOS without
modification. The two programs ML (ALL MODES) and ML (OLD MODES)
must be slightly modified, to correct the names of the binary
files loaded in line 30 of both programs.

          COMMON QUESTIONS ABOUT THE TIMEMASTER II H.O. (1)

Q.   Which slot should I plug the TIMEMASTER II H.O. into?

A.   Any empty slot from slot 1 through slot 7 will do, but we
      recommend slot 4 if it's available. Some commercial software
      expects a clock to be in slot 4 without looking anywhere
      else.

Q.   The program I have says it works with a clock card but I
      can't get it to work with the TIMEMASTER II H.O.

A.   You may have purchased a dud program. The   H.O. has
      been tested with dozens of clock type programs. You may want
      to contact the software publishers for help. Applied
      Engineering is very cooperative with programming companies
      and most software houses will be eager to solve your problem
      because of the large volume of TIMEMASTER sales. By
      helping you, they will be helping their other customers as
      well.

Q.   What mode is recommended for my own programming?

A.   The TIMEMASTER mode (switch 2 open)

Q.   I'm thinking of buying a program that says it will work with
      a real-time clock. Does this automatically mean that it will
      work with the H.O.?

A.   usually, but not always. Try to check with the author of the
      program   to   make   sure that he   supports TIMEMASTER.
      Over *300* software developers have purchased the TIMEMASTER,
      so compatibility problems should be rare.

Q.   With some of   the sample programs, I will see a cursor
      looking thing   flashing on part of the screen. What causes
      this?

A.   The cursor is caused whenever control is restored to the
      screen.   This is part of the Apple monitor ROM and cannot
      be avoided when using PR#'s. The cursor appears to flash
      because the clock is constantly being reread. However,
      most programs only read the clock periodically thereby
      preventing the cursor from reappearing. If you are going to
      write a program that must constantly read the clock, you
      might want to not use the on-board firmware or do it with
      interrupts.

Q.   I've written this really neat program that uses the H.O.
      Should I give it to Applied Engineering?

A.   Depending on the quality of the program, Applied Engineering
      may compensate you. If your program is really spectacular,
      you may want a software publisher to sell it for you.

Q.   The TIMEMASTER II H.O. seems to be way out in front of its
      competition, in both features and price. How did you do it?

A.   Apple peripherals are our only business that's why we're so
      good at it!

                    MILLISECONDS SOURCE CODE

                    1020   .OR $300
                    1030   .TF B.MILLISECONDS
                    1040 *------------------------------

```
0300— 4C 11 03  1050              JMP INT.INTERRUPT      CALL 768
0303— 4C 6C 03  1060              JMP RESET.COUNT CALL 771
0306— 4C 7A 03  1070              JMP SAVE.COUNT CALL 774
0309— 4C A6 03  1080              JMP STOP.INTERRUPT CALL 777
                1090  *-----------------------------------
030C—           1100  MSCNT.SAFE .B54        780.. .783, MSB FIRST
                1110  *-----------------------------------
0310—           1120  SLOT 16 .BS 1          $NO (ZERO IF NOT FOUND)
                1130  *-----------------------------------
                1140  INIT.INTERRUPT
0311— 78        1150              SEI          DISABLE IRQ
                1160  * -FIND TIMEMASTER        II SLOT
0312— AO C7     1170              LDY #$C7     START WITH SLOT 7
0314— 8C 8A 03  1180  .1          STY GET.ROM.BYTE+2
0317— A2 FE     1190              LDX #$FE     LOOK FOR $B2 AT $CNFE
0319— 20 88 03  1200              JSR GET.ROM.BYTE
031C— C9 B2     1210              CMP #$B2
031E— DO OF     1220              BNE .2
0320— E8        1230              INX          NEXT CHECK $CNOO
0321— 20 88 03  1240              JSR GET.ROM.BYTE
0324— C9 08     1250              CMP #$08     "PHP" AT $CNOO
0326— DO 07     1260              BNE .2       NOT THIS SLOT
0328— 20 88 03  1270              JSR GET.ROM.BYTE
032B— C9 78     1280              CMP #$78     "SEI" AT $CNO1
032D— FO 06     1290              BEQ .3       THIS SLOT
032F— 88        1300  .2          DEY          NEXT SLOT DOWN
0330— CO Cl     1310              CPY 4b$C1
0332— BO EQ     1320              BCS .1
0334— 60        1330              RTS
                1340  *---STUFF SLOT $NO VALUE---------
0335— 98        1350  .3          TYA          $CN
0336— OA        1360              ASL
0337— OA        1370              ASL
0338— OA        1380              ASL
0339— OA        1390              ASL
033A— 8D 10 03  1400              STA SLOT 16  $NO
033D— FO 48     1410              BEQ RETURN... NO TIMEMASTER II FOUND
033F— AA        1420              TAX
                1430  *---LOAD INTERRUPT VECTOR ---------
0340— A9 8D     1440              LDA #IRQ.HANDLER
0342— 8D FE 03  1450              STA $3FE
0345— A9 03     1460              LDA/I RQ HANDLER
0347— 8D FF 03  1470              STA $3FF
                1480  *---SET UP  INTERRUPTIONS---------
034A— A9 00     1490              LDA #0       POINT AT DIRECTION REGS
034C— 9D 81 CO  1500              STA $C081,X
034F— 9D 83 CO  1510              STA $C083,X
0352— 9D 80 CO  1520              STA $C080,X PORT A ALL INPUT
0355— A9 FF     1530              LDA #$FF
0357— 9D 82 CO  1540              STA #C082,X PORT B ALL OUTPUT
035A— A9 05     1550              LDA #~05     SELECT INTERVAL (CRA)
035C— 9D 81 CO  1560              STA $C081,X
03SF— A9 04     1570              LDA #~04    (CRB)
0361— 9D 83 CO  1580              STA $C083,X
0364— A9 2F     1590              LDA #$2F     ENABLE INT. OUT OF PIA
```

36

```
0366— 9D 82 CO    1600          STA $C082,X
0369— BD 80 CO    1610          LDA $C080,X CLEAR LAST INT. USE
              $C080 FOR 1024 & iSEC. USE $C082 FOR iMIN & 1 HOUR
                  1620 *——————————————————————————
                  1630 RESET.COUNT
036C— A9 00       1640          LDA itO ZERO THE 4-BYTE COUNTER
036E— 8D A8 03    1650          STA MSCNT
0371— 8D A9 03    1660          STA MSCNT+1
0374— 8D AA 03    1670          STA MSCNT+2
0377— 8D AB 03    1680          STA MSCNT+3
                  1690 *——————————————————————————
                  1700 SAVE.COUNT
037A— 78          1710          SEI DISABLE INTERRUPTS
037B— A2 03       1720          LDX #3 WHILE COPYING TUE COUNT
037D— BD A8 03    1730 .1       LDA MSCNT,X TO A SAFE LOCATION
0380— 9D 0C 03    1740          STA MSCNT.SAFE,X
0383— CA          1750          DEX
0384— 10 F7       1760          BPL .1
0386— 58          1770          CLI ALLOW INTERRUPTS AGAIN
0387— 60          1780 RETURN RTS
                  1790 *——————————————————————————
                  1800 GET.ROM.BYTE
0388— BD 00 C7    1810          LDA $C700,X
0388— E8          1820          INX
038C— 60          1830          RTS
                  1840 *——————————————————————————
                  1850 *——————————————————————————
                  1860 IRQ.HANDLER
038D— AS 45       1870          LDA $45 SAVE A—REG
038F— 48          1880          PHA
0390— 8A          1890          TXA SAVE X—REG
0391— 48          1900          PHA
0392— AE 10 03    1910          LDX SLOT 16 $NO
0395— BD 80 CO    1920          LDA $C080,X CLEAR INTERRUPT
0398— A2 03       1930          LDX #3 iNCREMENT COUNTER
039A— FE A8 03    1940 .1       INC MSCNT,X
039D— DO 03       1950          BNE .2
039F— CA          1960          DEX
03A0— 10 F8       1970          BPL .1
03A2— 68          1980 .2       PLA RESTORE X AND A REGS
03A3— AA          1990          TAX
03A4— 68          2000          PLA
03A5— 40          2010          RTI RETURN
                  2020 *——————————————————————————
                  2030 STOP.INTERRUPT
03A6— 78          2040          SEI
03A7— 60          2050          RTS
                  2060 *——————————————————————————
03A8—             2070 MSCNT  .BS 4
                  2080 *——————————————————————————
```