μCOMPAC

AppleTime

User's Manual

March, 1981

MicroComPac
P.O. Box 163
Golden, CO 80401

Introduction

The MicroComPac AppleTime is a versatile Clock/Calendar for the Apple II* computer. The features of this device include:

- o Time of day clock
- o Date of year calendar
- o Day of week
- o Recharging battery backup
- o Complete software formatting
- o Offset time/date/day readout
- o Program timer

The user of the AppleTime needs only to issue simple PRINT and INPUT statements to the device to fully utilize the clock's features. No extra programming is required, all of the programming is done on the board itself.

*Apple, Applesoft are trademarks of Apple Computer, Inc.

Installation

The installation of the AppleTime clock card into your Apple II computer is quite simple. Just follow these steps:

1) - Turn the computer's power supply off.

2) - Remove the top cover from the computer.

3) - Plug the AppleTime module into any one of the unused peripheral slots on the back of the computer's printed circuit board. Do not plug the card into slot Ø (the one closest to the power supply), as this is a special slot, reserved for other devices.

4) - Replace the top cover.

5) - You are ready to use the AppleTime.

Operation

Overview

The clock connects to the Apple II as any other 'intelligent' peripheral device does. That is, it plugs into one of the slots in the rear of the computer, and is addressed via the IN and PR commands of BASIC. If the board is to be used outside of the DOS environment, the program may directly utilize these commands. The format of these commands is IN (for input) or PR (for output) followed by a pound sign (#) followed by the slot number of the board. For example:

```
10  IN # 3
20  PR # 3
```

(All examples in this document assume that the clock is plugged into slot 3.)

This command will direct all of the computer input and output to the clock card. In order for the user to do normal input and output, the program must be told to switch the operations to the 'normal' device (usually slot Ø, the keyboard and screen of the Apple II). For example:

```
80  IN # Ø
90  PR # Ø
```

If the program is running under the control of the Disk Operating System (DOS), there is a slight modification to the above procedure. This is true because DOS hooks itself into the computer via these input and output mechanisms, and must do the alteration of these functions itself, or it will get lost and fail to perform properly. The program instructs DOS to do the clock alteration in the same manner as all DOS requests are made. That is, a PRINT command is formatted with a leading Control "D" and the command follows. For example:

```
5   D$=CHR$(4)              (Applesoft BASIC is assumed)
10  PRINT D$;"IN # 3"
20  PRINT D$;"PR # 3"
```

To exit from the clock:

```
80  PRINT D$;"IN # Ø"
90  PRINT D$;"PR # Ø"
```

Try the following program:

```
5   D$=CHR$(4)
10  PRINT D$;"IN # 3"
20  PRINT D$;"PR # 3"
30  INPUT "A";T$
40  PRINT D$;"IN # 0"
50  PRINT D$;"PR # 0"
60  PRINT "THE TIME IS ";T$
70  END
```

This program will setup the clock as the input/output device for the Apple II, ask for the time in Applesoft format, reset the input/output back to the primary devices and print the time of day.

Now add the following lines to the above program:

```
35  INPUT "D";C$
65  PRINT "THE DATE IS ";C$
```

These lines ask for the date in format number one, and print the date.

Note that the above examples assume the use of Applesoft BASIC. If Integer BASIC is used, a slight modification must be made to the program. These changed lines are:

```
3   DIM T$(20)        : REM MUST DIMENSION STRINGS
5   D$=""             : REM NO CHR$ FUNCTION IN INTEGER
                        (PUT CONTROL D IN THE QUOTES)
30  INPUT "B",T$      : REM USE ',' RATHER THAN ';' AND
                        'B' FORMAT RATHER THAN 'A'
35  INPUT "D",C$      : REM USE ',' RATHER THAN ';'
```

The only really important change here (although subtle) is the use of the comma rather than the semicolon in the INPUT command.

Reading the clock

The next topic is the use of the formatting controls. In the above examples, three formats were illustrated. These are "A" (Applesoft time), "D" (Date format 1) and "B" (Integer time). There are more formats, each will be discussed here.

The clock has three basic functions, that is, Time, Date and Day. Each of these functions has an unformatted form and at least one formatted form.

The unformatted form is the raw numeric data input with the most significant numbers first. That is, the date is

presented in the form of YYMMDD. For example, June 21, 1981 would be returned as 81Ø621. The time is returned in HHMMSS. For example 6:32 and 23 seconds (in the AM) would be returned as Ø63223 (or as Ø63223 AM if the AM/PM mode is being used). The day has only a single numeric digit and is returned as D (Sunday is Ø, Saturday is 6). The unformatted returns are handy for internal storage of data (for example, the date in the form of YYMMDD is 'sortable') or for direct numerical operations.

The formatted types of inputs are handy for printing on the screen or on reports. The time has two of these formats, although they are here to compensate for the differences between Integer and Applesoft BASIC, rather than to display the data differently. In either BASIC the data look the same. The format of the time is HH:MM:SS with an additional AM or PM tagged onto the end if this mode is desired. The date has two formats. Format number one is DD-MMM-YY where the month is alphabetic. Format two is MM/DD/YY, the more classic way of displaying a date. The day of week has only one formatted form, that is DDD, where the data are alphabetic.

To tell the clock board how to return the formatted data, all you need to do is use "prompted" input. That is, the user prompts the clock with a character of data and the ROM code in the clock does the rest. The above examples include this prompting.

The following table shows all of the clock prompts and what each character will do.

	Normal mode	Offset mode	Program Timer mode
Time, Unformatted	@	H	P
Time, Applesoft format	A	I	Q
Time, Integer format	B	J	R
Date, Unformatted	C	K	
Date, Format 1	D	L	
Date, Format 2	E	M	
Day, Unformatted	F	N	
Day, Formatted	G	O	

Prompts

Control of the clock

Up to now, we have only discussed the "Normal" mode of collecting the clock data. There are two other modes, Offset and Timer. In order to use these modes, the user must change the way the clock "thinks". This is done by the PRINT command.

There are four PRINT commands which can be used to set the clock, load the offset time, pulse the minute adjust and set the timer. The first character of the string "printed" to the clock card determines the function. These characters are:

Ø - Set the time and date
1 - Load the offset
2 - Pulse the minute adjust
3 - Set the timer (synchronize to the next second)

Setting the clock:

The string to set the clock is formatted as ØYYMMDDWHHMMØØ where:

YY - Year
MM - Month
DD - Day
W - Day of week (Sunday is Ø, Saturday is 6)
HH - Hour
MM - Minute
ØØ - Two zero characters (seconds are not settable)

The user may specify that the next February has 29 days by adding a 4 to the 1Ø's digit of the month. Additionally, the clock will keep time in 24 hour format or 12 hour format. If the user wants 24 hour time, an 8 is added to the 1Ø's digit of the hour. If the 12 hour mode is desired, a 4 is added to the 1Ø's digit of the hour to signify PM. The example BASIC program called SETTER included in this document handles all of these special conditions. The only thing to remember is to set the clock sometime between March 1 of the year before a leap year, and February 28 of the leap year. This will cause the leap year to have 29 days in February.

Loading the offset:

The offset time feature allows the user to determine the date and time anywhere in the world. To do this the clock must be told the hours and minutes to be offset from the local time. Additionally, if the International Date Line

is crossed by the offset, the clock must be informed of this as well. The first byte of an offset load is a "1". The next character must be either a plus sign (+) or a minus sign (-). This will tell the clock to add time (going east) or subtract time (going west). Up to a 12 hour offset can be specified for either direction. (Indeed, the plus 12 and minus 12 offset come to the same time zone.) After the sign, an optional "D" may be put into the string. This will signify that the Date Line has been crossed by the offset and the clock will adjust the date accordingly. The next four characters of the string specify the hours and minutes to offset. They are in the form of HHMM. Note that leading zeros are required to pad out the hours. Seconds cannot be offset. For example:

Assume that the clock is set to Pacific Standard Time and the desired offset is GMT. The command is as follows:

```
PRINT "1+0800"
```

(This and the following examples assume that the PR and IN commands have been done)

Now the offset Applesoft time may be read as:

```
INPUT "I"; GT$
```

and the offset date may be read as:

```
INPUT "L"; GD$
```

NOTE! - When offsets are done, the time must be read first in order to properly offset the date. You must do this even if you don't plan to use the returned time. However, once you have done one time offset, you may request as many dates or date formats as you wish without requesting another time. (This assumes that the time in the offset zone does not become 12AM while the dates are being read.)

Minute adjust:

The crystal time base of the clock has been very carefully adjusted at the factory. However, because of age and temperature variations inside of the computer the clock may drift off of the exact time. In order to correct for this slight drift, there is a command which will set the clock to the nearest minute. (This assumes that the clock is within 30 seconds of the correct time when this operation is done.) To do this minute adjust issue the clock command:

```
PRINT "20"
```

If the clock has fewer than 30 seconds in its seconds counter the seconds will simply be set to 00. If the clock has 30 seconds or more in its counter the minutes will be incremented (Hours too if necessary) and the seconds will be set to 00. The SETTER program also has this function as an option.

Set the timer:

In order to do some simple timing tasks, there is a built-in timer in the clock program. To set the timer issue the command:

```
PRINT "30"
```

This causes the program to wait for the next second to tick past and then it issues an internal clock read. It is now assumed that some event, external to the clock program, will be started. At the conclusion of this event the user may issue a "timer" input request. For example:

```
INPUT "Q";DT$
```

At this point the clock program waits for the next second to tick past (timing the wait in 10's of milliseconds). When the second transition is seen the difference in time between the PRINT and INPUT will be output in one of the three formats. Appended to the normal string of characters will be ".XX", where XX is the fractional seconds between the calls. Note that if the unformatted time request is used, that the time data may be read directly into a numeric variable (in Applesoft) and used as numeric data. For example:

```
INPUT "P"; T
```

Another valuable feature for this function is to create known delays in the user's program. In the example:

```
FOR I=1 TO 10: PRINT "30": NEXT I
```

The program will be delayed for 10 transitions of the second counter.

It is important to point out that this function should not be considered to be an accurate stopwatch for external events, as the setting function must wait for a fraction of a second to elapse prior to the start of the timing operation. It is a very valuable feature, however, for timing program execution or controlling the program on a time delay basis.

Operational Notes

This is a collection of hints to help you get the most from your AppleTime.

Removing the 5832 clock IC:

The battery is constantly connected to the 5832 clock chip. If this chip is to be removed for any reason, the battery must be disconnected from the card. Unsolder one of the battery leads, then remove the chip. Failure to do this may result in destruction of the IC.

Adjusting the internal frequency:

The small trimmer located next to the crystal is used to make fine adjustments of the time base. When the small dot on the trimmer is at the bottom the clock is running at its fastest. When the dot is at the top the clock is the slowest. Note that although the trimmer has a 360° travel, that the two 180° portions are the same.

RAM Usage:

The Apple II's intelligent interface conventions allow for eight RAM locations in the primary text buffer to be used by the interface's program. The AppleTime uses all eight of these locations. Additionally, eight more locations are shared between all of the interfaces. The AppleTime uses one of these ($678) for the Program Timer function. This should cause no problems with the normal operation of other interfaces.

Cursor during INPUT:

Due to the Input/Output logic of the Apple II, the flashing cursor appears during AppleTime's INPUT operations. The duration of the cusrsor flash is quite short as the code in the clock card returns its data very quickly. However, please note that the text beyond the cursor is erased at the conclusion if the INPUT operation (just as it is from keyboard INPUT statements).

Setting the clock:

The National Bureau of Standards operates a series of radio stations (WWV) transmitting (among other things) the exact time of day. This service is also available over the telephone, the number is (303)-499-7111 (a toll call outside of the Denver area). For those users who demand very precise clock settings, this may be handy information.

APPENDIX A - SAMPLE PROGRAMS

```
PR#0
]LIST

1  REM   DIAGNOSE                    TEST THE APPLETIME
3  HOME
4  PRINT : PRINT " APPLESOFT CLOCK DIAGNOSTIC": PRINT : INPUT "IN WHICH SLOT IS THE APPLETIME CARD "}SL$
5 D$ =  CHR$ (4)
6  VTAB 4
100  PRINT D$}"PR$"}SL$
110  PRINT D$}"IN#"}SL$
120  PRINT "3 "
130  PRINT "1+0800": REM  PST TO GMT
140  INPUT "@"}TU$: INPUT "H"}GU$
150  INPUT "A"}TA$: INPUT "I"}GA$
160  INPUT "C"}DA$: INPUT "K"}HA$
170  INPUT "D"}D1$: INPUT "L"}H1$
180  INPUT "E"}D2$: INPUT "M"}H2$
190  INPUT "F"}DN$: INPUT "N"}GN$
200  INPUT "G"}DT$: INPUT "O"}GT$
300  PRINT D$}"PR#0"
310  PRINT D$}"IN#0"
320  PRINT
330  HTAB 18: PRINT "LOCAL:";: HTAB 28: PRINT "GMT:": PRINT
350  PRINT "UNFORMATTED DAY   "}DN$}: HTAB 28: PRINT GN$
360  PRINT "FORMATTED DAY     "}DT$}: HTAB 28: PRINT GT$: PRINT
370  PRINT "UNFORMATTED TIME  "}TU$}: HTAB 28: PRINT GU$
380  PRINT "APPLESOFT TIME    "}TA$}: HTAB 28: PRINT GA$: PRINT
400  PRINT "UNFORMATTED DATE  "}DA$}: HTAB 28: PRINT HA$
410  PRINT "DATE FORMAT 1     "}D1$}: HTAB 28: PRINT H1$
420  PRINT "DATE FORMAT 2     "}D2$}: HTAB 28: PRINT H2$
800  GOTO 6
32767  END
```

```
]

]LIST

1  REM  SETTER
100  REM  CLOCK SETTING ROUTINE
105  HOME : VTAB 3
110  PRINT "THIS ROUTINE WILL SET THE COMPAC": PRINT "APPLETIME CLOCK."
115  PRINT : INPUT "IN WHICH SLOT IS THE APPLETIME CLOCK ";SL$
117  PRINT : PRINT "DO YOU WANT A FULL OR 30 SECOND SET ";: INPUT C$: IF  LEFT$ (C$,1) = "3" THEN C$ = "20": GOTO 280
120  PRINT : PRINT "ENTER THE DATE IN THE FORM:": PRINT "MM/DD/YY ";
125 D$(0) = "SUNDAY":D$(1) = "MONDAY":D$(2) = "TUESDAY":D$(3) = "WEDNESDAY":D$(4) = "THURSDAY":D$(5) = "FRIDAY":D$(6) = "SATURDAY"
130  INPUT D$:P = 1:P0 = 1:I = 1
140  IF  MID$ (D$,P,1) = "/" THEN T$(I) =  MID$ (D$,P0,P - P0):I = I + 1:P0 = P + 1
145 P = P + 1: IF I = 3 GOTO 160
150  IF P )  LEN (D$) THEN  PRINT " ERROR - YOU MUST USE SLASHES!": GOTO 120
155  GOTO 140
160 T$(3) =  MID$ (D$,P0, LEN (D$) - P + 1)
170  FOR I = 1 TO 3: IF  LEN (T$(I)) ( 1 THEN  PRINT " ERROR - YOU LEFT ONE OUT": GOTO 120
175  IF  LEN (T$(I)) ) 2 THEN  PRINT " ERROR - ONLY 2 CHARACTERS": GOTO 120
180  IF  LEN (T$(I)) = 1 THEN T$(I) = "0" + T$(I)
185  NEXT I
190  IF  VAL (T$(3)) ( 1 THEN  PRINT "  ERROR - YEAR MUST BE NON-ZERO": GOTO 120
200 Y =  VAL (T$(3)):D =  INT (365 * (Y - 2) + (Y - 1) / 4) +  VAL (T$(2)) + 2
205  RESTORE
210  FOR I = 1 TO  VAL (T$(1)): READ X:D = D + X: NEXT I
215  DATA  0,31,28,31,30,31,30,31,31,30,31,30
220 D = D - ( INT (D / 7) * 7)
230  PRINT : PRINT "IT IS ";D$(D);"."
235 DD = 28:T1 = 0: IF  INT (Y / 4) * 4 = Y THEN  IF  VAL (T$(1)) ( 3 THEN T1 = 4:DD = 29
237  IF Y -  INT (Y / 4) * 4 = 3 THEN  IF  VAL (T$(1)) ) 2 THEN T1 = 4:DD = 29
238  PRINT "NEXT FEBRUARY HAS ";DD;" DAYS.": PRINT
240  PRINT "DO YOU WANT 24 OR 12 HOUR TIME ";: INPUT C$
245 T = 8: IF  LEFT$ (C$,1) = "1" THEN T = 0: PRINT "AM OR PM ";: INPUT C$: IF  LEFT$ (C$,1) = "P" THEN T = 4
250  PRINT "ENTER THE TIME IN THE FORM:": PRINT "HHMM ";: INPUT C$: IF  LEN (C$) ( 4 THEN C$ = "0" + C$
255  IF  LEN (C$) (  ) 4 THEN  PRINT " ERROR - ENTER 3 OR 4 DIGITS": GOTO 250
260 C$ =  CHR$ ( ASC ( LEFT$ (C$,1)) + T) +  RIGHT$ (C$,3)
265 T$(2) =  CHR$ ( ASC ( LEFT$ (T$(2),1)) + T1) +  RIGHT$ (T$(2),1)
270 C$ = "0" + T$(3) + T$(1) + T$(2) +  STR$ (D) + C$ + "00"
280  PRINT : INPUT "PRESS RETURN AT THE NEXT MINUTE";T$
300 D$ =  CHR$ (4): PRINT D$;"PR#";SL$: PRINT D$;"IN#";SL$: PRINT C$
305  INPUT "A";C$: INPUT "D";C1$: INPUT "G";C2$
310  PRINT D$;"PR#0": PRINT D$;"IN#0"
315  PRINT : PRINT "PLEASE VERIFY:": PRINT C$;"   ";C1$;" ";C2$
320  FOR I = 1 TO 2000: NEXT I
330  PRINT D$;"RUN DIAGNOSE"
```

APPENDIX B - ROM CODE LISTING

```
:88F 29 40              AND#    INTYP    ;SEE IF OUTPUT IS EXPECTED
:891 F0 04              BEQ     DO08#
:893 A9 9D              LDA#    <REJOUT  ;NO - SET TO IGNORE OUTPUT
:895 D0 02              BNE     DO08C
:897 A9 A8   DO08#      LDA#    <MOROUT  ;YES - SETUP TO GET MORE OUTPUT
:899 85 36   DO08C:     STA     CWSL
:89      B8             BNE     EXIT     ;ALWAYS BRANCH TO EXIT

:89D C9 8D   REJOUT:    CMP#    #8D      ;IS <CR>?
:89F D0 06              BNE     REJ08    ;NO - IGNORE THE DATA
:8A1 A9 08              LDA#    #08      ;RESET THE OUTPUT VECTOR
:8A3 85 36              STA     CWSL
:8A5 A9 8D              LDA#    #8D      ;RESTORE THE AC
:8A7 60      REJ08:     RTS              ;GO BACK

:8A8 B8      MOROUT:    CLV              ;USE INPUT'S REGISTER STUFF
:8A9 50 08              BVC     MORIXX
:8AB A9 80   MOROXX:    LDA#    #80      ;TURN ON THE 2K
:8AD 9D 80C0            STAX    P#
:8B0 9D 80C8            STAX    P#
:8B3 20 00C9            JSR     0DISP#   ;GO TO THE OUTPUT DISPATCH
:8B6 18                 CLC
:8B7 90 9C              BCC     EXIT     ;ALWAYS

          ; THE CALLS TO THE 2K ROM MUST
          ;  RTS BACK TO THIS PAGE SO THAT THE
          ;  2K ADDRESSING MAY BE SWITCHED OFF.

                  .LOC    MOVLOC+$E7, LODLOC+$E7
                                   ;FORCE THIS TABLE TO
                                   ;THE END OF PAGE $C8

          ; THE FOLLOWING JUMP TABLE WILL DISPATCH
          ; THE OPERATION.  ALL OF THESE ROUTINES
          ; ARE SUBROUTINES.

          ; IT IS REQUIRED THAT THE POSTIION OF THE
          ; JUMPS REMAIN THE SAME ACROSS DIFFERENT
          ; VERSIONS OF THE ROM CODE.  THE TARGETS
          ; OF THE JUMPS ARE ALLOWED TO CHANGE.

3E7 4C F9C9 ID0:    JMP     READSW   ;READ STOPWATCH
3EA 4C 16CC ID1:    JMP     TIME     ;READ TIME
3ED 4C F9CC ID2:    JMP     DATE     ;READ DATE
3F0 4C 00C9 ID3:    JMP     DAY      ;READ DATE

3F3 4C E9CD OD0:    JMP     SET      ;SET THE CLOCK
3F6 4C D0C9 OD1:    JMP     SETSW    ;START THE STOPWATCH
3F9 4C C5CD OD2:    JMP     OFFOFF   ;SET THE OFFSET TIME
3FC 4C 30C9 OD3:    JMP     SET30    ;DO THE 30 SECOND ADJUST

3FF 00      ID:     .BYTE   BRD+COD  ;STORE THE REVISION OF
                                     ;THE CODE AND BOARD
                                     ;THIS BYTE MUST BE THE
                                     ;LAST BYTE IN THIS PAGE
```

```
* ASM65 ***  (V3.0-AP)  OCT-79

                  .PAGE

          ;  THE FOLLOWING IS THE DISPATCH FOR
          ;  THE LEGAL INPUT/OUTPUT ROUTINES.  ALL OF
          ;  THE ROUTINES ARE LOCATED IN THE
          ;  $C0XX REGION.

          ; DISPATCH A PRINT OPERATION

:00 A5 45   0DISP#: LDA     #BUF     ;SEE IF UNEXPECTED TERMINATION
:02 29 7F           AND#    $7F
:04 F0 17           BEQ     DONE     ;A NULL ENDS
:06 C9 0D           CMP#    $0D
:08 F0 13           BEQ     DONE     ;SO DOES A <CR>

:0A B9 F804         LDAY    TYPE     ;GET THE TYPE
:0D C9 30           CMP#    TSET     ;IS SET?
:0F F0 E2           BEQ     0D0
:11 C9 32           CMP#    TSEC30   ;IS 30 SECOND ADJUST?
:13 F0 E7           BEQ     0D3
:15 C9 33           CMP#    TSETSW   ;SET STOPWATCH?
:17 F0 DD           BEQ     0D1
:19 C9 31           CMP#    TOFF     ;IS OFFSET?
:1B F0 DC           BEQ     0D2
                                     ;IF NO DECODE, FALL THRU TO IGNORE
                                     ; THE OUTPUT

          ; END A PRIT OPERATION

:1D A9 9D   DONE:   LDA#    <REJOUT  ;SET TO IGNORE THE REST OF THE OUTPUT
:1F 85 36           STA     CWSL
:21 60              RTS              ;AND RETURN

          ; DISPATCH AN INPUT OPERATION

:22 B9 F804 IDISPA: LDAY    TYPE
:25 29 10           AND#    TRSW     ;IS STOPWATCH READ?
:27 D0 BE           BNE     ID0
:29 B9 F804         LDAY    TYPE
:2C 29 07           AND#    $07      ;ONLY LOOK AT FIRST 8
:2E C9 03           CMP#    $03      ;0,1 OR 2 (TIME)
:30 30 B8           BMI     ID1
:32 C9 06           CMP#    $06      ;3,4 OR 5 (DATE)?
:34 ** 07           BMI     ID2
:36     3           BPL     ID3      ;6 OR 7 (DAY)

          ; THE LEAST SIGNIFICANT 3 BITS OF THE TYPE
          ; ARE USED TO DETERMINE THE INPUT DISPATCH.
          ; THEREFORE SOME ROUTINE WILL BE DISPATCHED
          ; REGARDLESS OF THE DATA TYPE.
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                  .PAGE

          ; DAY OF THE WEEK OUTPUT
          ;
C948 B9 F807 DAY:   LDAY    STATE    ;GET THE STATE
C94B D0 3E          BNE     DAY00    ;START?
C94D A9 06          LDA#    #06      ;DAY ADDRESS
C94F 20 03CB        JSR     READC    ;READ THE CLOCK
C952 99 7007        STAY    TEMP     ;HOLD THE DAY
C955 A9 00          LDA#    00       ;TURN OFF HOLD
C957 9D 02C0        STAX    P#
C95A B9 F804        LDAY    TYPE     ;SEE IF ANY OFFSET
C95D 29 08          AND#    #08      ;EITHER ONE
C95F F0 2A          BEQ     DAY00    ;IF NOT, BYPASS

C961 B9 7004        LDAY    STATUS   ;ADD?
C964 29 10          AND#    ADDDAY
C966 F0 0F          BEQ     DAY00A   ;NO
C968 B9 7007        LDAY    TEMP     ;YES, BUMP IT
C96B 18             CLC
C96C 69 01          ADC#    #01
C96E C9 07          CMP#    #07      ;TOO FAR?
C970 D0 02          BNE     DAY00B   ;NO
C972 A9 00          LDA#    #00      ;MAKE 0

C974 99 7007 DAY00B: STAY   TEMP     ;PUT BACK

C977 B9 7004 DAY00A: LDAY   STATUS   ;SEE IF SUBTRACT
C97A 29 20          AND#    SUBDAY
C97C F0 0D          BEQ     DAY00    ;NO
C97E B9 7007        LDAY    TEMP     ;GET IT
C981 38             SEC
C982 E9 01          SBC#    #01      ;DROP BY 1
C984 10 02          BPL     DAY00C   ;IF WENT MINUS ADJUST
C986 A9 06          LDA#    #06
C988 99 7007 DAY00C: STAY   TEMP


C98B B9 F807 DAY00: LDAY    STATE    ;GET THE STATE
C98E C9 03          CMP#    3        ;3 IS END
C990 F0 14          BEQ     FINJ0
C992 B9 F804        LDAY    TYPE     ;IS UNFORMATTED?
C995 29 07          AND#    #07      ;LOW 7
C997 C9 06          CMP#    TUDAY
C999 D0 0E          BNE     DAY01    ;GO DO FORMATTED
C99B B9 F807        LDAY    STATE    ;GET STATE
C99E D0 06          BNE     FINJ0    ;NON 0 IS UNFORMATTED END
C9A0 B9 7007        LDAY    TEMP     ;GET THE DAY
C9A3 4C 92CB        JMP     ASCII    ;END WITH ASCII

C9A6 4C A0CB FINJ0: JMP     FINISH   ;END

C9A9 8A      DAY01: TXA              ;HOLD THE X
C9AA 48             PHA
C9AB B9 7007        LDAY    TEMP     ;GET THE DAY
C9AE 0A             ASLA             ; X2
C9AF 18             CLC
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                  .PAGE

          ; THIS IS THE 30 SECOND ADJUST CODE.
          ; THE CLOCK WILL BE SET TO THE NEAREST
          ; MINUTE WHEN THIS CODE IS RUN.


C9B0 A9 08   SET30: LDA#    SEC30    ;SET THE 30 SECOND RESET
C9B2 9D 02C0        STAX    P#
C9B5 A9 74          LDA#    $74      ;DO A (RATHER LONG) DELAY
C9B7 20 A0FC        JSR     WAIT     ;  (RETURNS 0 IN A)
C9BA 9D 02C0        STAX    P#       ;TURN OFF 30 SEC BIT
C9BD 4C 1DC9        JMP     DONE     ;REJECT THE REST OF THE OUTPUT
```

```
;       C L O C K
;
;    CODE FOR MICROCOMPAC CLOCK.
;
;  THIS CODE RESIDES IN THE ROM ON THE
;  COMPAC CLOCK CARD.  THE ROUTINES
;  INTERFACE THE CLOCK WITH BOTH APPLE
;  BASICS.
;
;       DEFINES

        .DEF    COD     =$00        ;REVISION OF CODE (IN HIGH NIBBLE)
        .DEF    BRD     =$00        ;REVISION OF BOARD (IN LOW NIBBLE)

        .DEF    IOBASE=$C080        ;IO ADDRESSES
        .DEF    HOLD    =$01        ;HOLD BIT
        .DEF    READ    =$02        ;READ BIT
        .DEF    WRITE   =$04        ;WRITE BIT
        .DEF    SEC30   =$08        ;30 SECOND SET BIT

        .DEF    IOSAVE=$FF4A        ;APPLE'S REGISTER SAVE ROUTINE
        .DEF    ABUF    =$0045      ; SAVE'S A REG BUFFER
        .DEF    XBUF    =$0046      ; SAVE'S X REG BUFFER
        .DEF    IOREST=$FF3F        ;APPLE'S RESTORE ROUTINE

        .DEF    XRET    =$FF58      ;RTS IN APPLE'S ROM CODE
        .DEF    DOSSLT=$07F8        ;BUFFER FOR SLOT IN THE FORM OF $CN
        .DEF    WAIT    =$FCA8      ;APPLE'S DELAY
        .DEF    KWSL    =$0038      ;INPUT LOW VECTOR
        .DEF    KWSH    =KWSL+1     ; HI BYTE
        .DEF    CWSL    =$0036      ;OUTPUT LOW VECTOR
        .DEF    CWSH    =CWSL+1     ; HI BYTE

;  THE BASE ADDRESS (IOBASE) ADDRESSES
;  THE 6021 IO PORT.  THE ADDRESSES ARE
;  AS FOLLOWS:
;       IOBASE+0 = PORT A DDR* IF APCR BIT 2=0
;                  PORT A IF APCR BIT 2=1
;       IOBASE+1 = APCR (A PORT CONTROL REGISTER)
;       IOBASE+2 = PORT B DDR* IF BPCR BIT 2=0
;                  PORT B IF BPCR BIT 2=1
;       IOBASE+3 = BPCR
;
;  *DDR=DATA DIRECTION REGISTER

;  PORT B CONTROLS THE 5032'S ADDRESS AND CONTROL LINES.
;  PORT A CONTROLS THE 5032'S DATA LINES
;  THESE LOCATIONS ARE ADDRESSED WITH THE
;  SLOT IN X (AS $N0).

        .DEF    PAC     =IOBASE+1 ;PORT A CONTROL
        .DEF    PA      =IOBASE+0 ;PORT A
        .DEF    PBC     =IOBASE+3 ;PORT B CONTROL
        .DEF    PB      =IOBASE+2 ;PORT B


;  RAM LOCATIONS FOR INTERNAL BUFFERS
;  ARE ADDRESSED BY INDEXED OPERATIONS
;  WITH THE SLOT IN Y (AS $0N).



        .DEF    STATUS=$0478    ;STATUS BIT BYTE
        .DEF    TYPE    =$04F8  ;TYPE OF PROCESSING
        .DEF    OFFHRS=$0578    ;HOUR OFFSET
        .DEF    OFFMIN=$05F8    ;MINUTE OFFSET
        .DEF    OFFTMP=OFFMIN   ; USE FOR TEMP OFFSET STORAGE AS WE
        .DEF    MMYY    =$0678  ;HOURS-YEARS BUFFER
        .DEF    HHMM    =$06F8  ;MINUTES-MONTHS BUFFER
        .DEF    SSDD    =$0778  ;SECONDS-DAYS BUFFER
        .DEF    TEMP    =SSDD   ; USE FOR TEMP STORAGE AS WELL
        .DEF    STATE   =$07F8  ;STATE OF PROCESS

;  FOR STOPWATCH OPERATIONS THERE MUST
;  BE ONE EXTRA BUFFER (FOR FRACTIONS).
;  THE "SHARED" BUFFER SPACE IS USED
;  FOR THIS BYTE.

        .DEF    FRAC    =$0678  ;STOPWATCH FRACTION BUFFER


;  DEFINE THE STATUS BITS FOR THE STATUS BYTE

        .DEF    OFFSGN=$01      ;SIGN OF OFFSET
        .DEF    DATLIN=$02      ;DATELINE CROSS IF SET
        .DEF    AMPM    =$04    ;1=PM, 0=AM
        .DEF    C1224   =$08    ;1=24, 0=12
        .DEF    ADDDAY=$10      ;1=ADD A DAY
        .DEF    SUBDAY=$20      ;1=SUB A DAY
        .DEF    SWATCH=$40      ;STOPWATCH OPERATION
        .DEF    SFLOW   =$80    ;SPECIAL FLOW LOGICAL

;  DEFINE THE TYPES

        .DEF    TSET    ='0     ;SET THE CLOCK
        .DEF    TOFF    ='1     ;OFFSET
        .DEF    TSEC30='2       ;30 SECOND ADJUST
        .DEF    TSETSW='3       ;SET STOPWATCH

        .DEF    INTYP   =$40    ;WILL BE SET FOR INPUT TYPES
        .DEF    TRSW    =$10    ;WILL BE SET FOR STOPWATCH READS
        .DEF    TUTI    =$00    ;'0' UNFORMATTED TIME
        .DEF    TATI    =$01    ;'A' APPLESOFT TIME
        .DEF    TITI    =$02    ;'B' INTEGER TIME
        .DEF    TUDA    =$03    ;'C' UNFORMATTED DATE
        .DEF    TDA1    =$04    ;'D' DATE FORMAT 1
        .DEF    TDA2    =$05    ;'E' DATE FORMAT 2
        .DEF    TUDAY   =$06    ;'F' UNFORMATTED DAY OF WEEK
        .DEF    TDAY    =$07    ;'0' FORMATTED DAY OF WEEK

;  NOTE THAT WITH THE ABOVE (0-8) TYPES,
;  THE N-0 TYPES RETURN OFFSET DATA.
```



```
                        .PAGE

                        .DEF    LODLOC=$6000    ;LOAD LOCATION OF THIS CODE
                        .DEF    MOVLOC=$C800    ;LOCATION WHERE CODE EXECUTES

                        .LOC    MOVLOC, LODLOC

                ; THE FOLLOWING CODE WILL BE ENTERED ON
                ; EVERY "FIRST CALL", EITHER FOR INPUT
                ; OR OUTPUT.
                ;
                ; UP UNTIL THE JSR TO THE DISPATCHING
                ; ROUTINES THIS CODE IS EXECUTED IN
                ; PAGE $CX, WHERE X IS THE SLOT NUMBER.
                ; THIS PAGE OF CODE ALSO APPEARS IN
                ; PAGE $C8 AFTER THE 2K ADDRESSING
                ; IS ENABLED.

C800 20 4AFF ENTRY1:    JSR     IOSAVE  ;SAVE THE REGISTERS
C803 78                 SEI             ;NO INTERRUPTS
C804 20 58FF            JSR     XRET    ;GO TO A KNOWN RTS
C807 BA                 TSX             ;GET THE STACK POINTR
C808 BD 0001            LDAX    $100    ;GET THE STACK DATA
C80B 8D F807            STA     DOSSLT  ;BE SURE DOS'S POINTER IS SET
C80E 29 0F              AND#    $0F     ;GET JUST THE SLOT
C810 A8                 TAY             ;HOLD THE SLOT AS $0N
C811 0A                 ASLA            ;GET SLOT INTO HI NIBBLE
C812 0A                 ASLA
C813 0A                 ASLA
C814 0A                 ASLA
C815 AA                 TAX             ;HOLD FOR INDEXING

                ;  THIS CODE SETS UP THE DDR'S IN
                ;  THE 6021 TO THE PROPER STATE FOR
                ;  A CLOCK INPUT/OUTPUT.  NOTE THAT
                ;  THE 2K ADDRESSING IS NOT ENABLED
                ;  AT THIS TIME.

C816 A9 00              LDA#    $00
C818 99 F807            STAY    STATE   ;RESET STATE
C81B 9D 01C0            STAX    PAC     ;RESET CONTROLS
C81E 9D 03C0            STAX    PBC
C821 A9 80              LDA#    $80     ;HI BIT IS OUTPUT
                                        ;LO 7 ARE INPUTS
C823 9D 00C0            STAX    PA      ;SET A DDR FOR INPUT
C826 A9 FF              LDA#    $FF
C828 9D 02C0            STAX    PB      ;SET B DDR FOR OUTPUT
                                        ; ON ALL 8 BITS
C82B A9 04              LDA#    $04     ;SETUP FOR DATA
C82D 9D 01C0            STAX    PAC
C830 9D 03C0            STAX    PBC
C833 A5 37              LDA     CWSH    ;TEST THE OUTPUT VECTOR
C835 CD F807            CMP     DOSSLT  ;IS SAME?
C838 D0 04              BNE     DOIN    ;IF NOT, MUST BE INPUT
C83A A5 36              LDA     CWSL    ;IF OUTPUT, HAS IT BEEN SETUP?
C83C F0 3E              BEQ     DOOUT   ;IF NOT, IS FIRST OUTPUT CALL
```



```
                        .PAGE

                ; SETUP THE INITIAL ENTRY STUFF

C83E B9 7804 DOIN:      LDAY    STATUS  ;SET FLOW TRUE
C841 09 80              ORA#    SFLOW   ;AS THE CLOCK READING CODE MAY NEED IT
C843 99 7804            STAY    STATUS
C846 A9 60              LDA#    <MORIN  ;SET THE INPUT VECTOR
C848 85 38              STA     KWSL


C84A A9 00    NORDIS:   LDA#    $00     ;TURN ON THE 2K
C84C 9D 00C0            STAX    PA      ;TWO ACCESSES ARE REQUIRED
C84F 9D 00C0            STAX    PA      ; FOR THE SWITCH
C852 20 22C9            JSR     IDISPA  ;DO THE INPUT DISPATCH
                                        ;FALL TO EXIT

C855 A9 00    EXIT:     LDA#    $00     ;
C857 9D 00C0            STAX    PA      ;TURN OFF 2K (SECOND ONE REALLY DOES IT)
C85A 9D 00C0            STAX    PA
C85D 4C 3FFF            JMP     IOREST  ;RESTORE THE REGS

                ;  THE FOLLOWING SETS UP X AND Y FOR
                ;  PROPER USE.  THE ROUTINES MUST
                ;  PRESERVE THESE REGISTERS!

C860 2C 58FF MORIN:     BIT     XRET    ;SET THE V BIT
C863 08      MORIXX:    PHP             ;REMEMBER THE STATUS
C864 20 4AFF            JSR     IOSAVE  ;SAVE THE REGS
C867 78                 SEI             ;NO INTERRUPTS
C868 20 58FF            JSR     XRET    ;GO TO AN RTS
C86B BA                 TSX             ;GET THE STACK POINTR
C86C BD 0001            LDAX    $100    ;GET THE STACK DATA
C86F 29 0F              AND#    $0F     ;GET JUST THE SLOT
C871 A8                 TAY             ;HOLD THE SLOT AS $0N
C872 0A                 ASLA            ;MAKE INDEX FOR IO
C873 0A                 ASLA
C874 0A                 ASLA
C875 0A                 ASLA
C876 AA                 TAX             ;USE X FOR IO
C877 28                 PLP             ;GET THE V BIT BACK
C878 50 31              BVC     MORORX  ;BACK TO OUTPUT
C87A 70 CE              BVS     NORDIS  ;BACK TO INPUT


C87C A9 BF    DOOUT:    LDA#    ~SWATCH-1 ;BE SURE THE STOPWATCH BIT IS OFF
C87E 39 7804            ANDY    STATUS
C881 99 7804            STAY    STATUS
C884 A5 45              LDA     ABUF    ;GET THE TYPE OF INPUT
C886 29 7F              AND#    $7F     ;STRIP HI BIT
C888 C9 0D              CMP#    $0D     ;INTEGER BASIC GIVES US SOME OF THESE
C88A F0 C9              BEQ     EXIT    ;WE DON'T WANT THEM
C88C 99 F804            STAY    TYPE    ;STORE IT
```

```
C9B0 79 7007          ADCY   TEMP     ; X3
C9B3 79 F007          ADCY   STATE    ;ADD IN THE STATE
C9B6 AA               TAX             ;MAKE INDEX
C9B7 BD C3C9          LDAX   DAYS     ;GET THE DAY
C9BA 09 80            ORA#   $80      ;SET THE HI BIT
C9BC 85 45            STA    ABUF     ;HOLD FOR RETURN
C9BE 68               PLA             ;RESTORE X
C9BF AA               TAX
C9    96CB            JMP    INCSTA   ;END THIS WAY

C9C3 53 554E DAYS:    .ASCII "SUNMONTUEWEDTHRFRISAT"


                      .LINK  RAM3
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                      .PAGE

            ; SET STOPWATCH.  WAIT FOR A SECOND
            ; TRANSACTION, THEN READ THE TIME
            ; AND EXIT.  THIS ROUTINE MAY ALSO
            ; BE USED FOR PROGRAM SYNCHRONIZATION.

C9D0 A5 36    SETSW:  LDA    CWSL     ;BUFFER A BYTE FOR USE IN PAGE 0
C9D2 48               PHA
C9D3 A9 02            LDA#   READ     ;TURN ON THE READ LINE
C9D5 9D 82C0          STAX   PB
C9D8 BD 80C0          LDAX   PA       ;GET SECONDS
C9E0 29 0F            AND#   $0F      ;CLEAR HI NIBBLE
C9E2 85 36            STA    CWSL     ;STORE THE TEST
C9E4 BD 80C0 SETS00:  LDAX   PA       ;HAS TIME CHANGED?
C9E7 29 0F            AND#   $0F      ;STRIP BARBAGE
C9E9 C5 36            CMP    CWSL
C9EB F0 F7            BEQ    SETS00   ;WAIT
C9ED 20 99CA          JSR    READCC   ;STORE THE TIME
C9F0 68               PLA             ;RESTORE PAGE 0
C9F1 85 36            STA    CWSL
C9F3 4C 1DC9          JMP    DONE     ;EXIT QUICKLY
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                      .PAGE

            ; READ STOPWATCH.  FIRST READ THE TIME
            ; CORRECTING THE BUFFERS TO DELTA TIME.
            ; THEN COUNT THE TIME TO THE "NEXT
            ; SECOND" TRANSITION.  THIS TIME IS
            ; THE "FRACTIONAL" SECOND.


C9F6 B9 F007 READSW:  LDAY   STATE    ;FIRST ENTRY?
C9F9 D0 05            BNE    RSW05    ;NO - GO DIRECTLY TO TIME
C9FB B9 7004          LDAY   STATUS   ;SEE IF SECOND STATE 0 ENTRY
C9FE 30 03            BMI    RSW00    ;FIRST ENTRY - DO THE STOPWATCH THING

CA00 4C 16CC RSW05:   JMP    TIME     ;ELSE GO TO TIME FOR OUTPUT

CA03 A9 40   RSW00:   LDA#   SWATCH   ;SET THE STOP WATCH BIT
CA05 20 A9CB          JSR    SETSTA
CA08 A5 38            LDA    KWSL     ;BUFFER PAGE ZERO BYTE
CA0A 48               PHA

CA0B 20 99CA          JSR    READCC   ;READ THE CLOCK
                                      ; (COLLECTING DIFFERENCES)
                                      ; (KWSL LOADED WITH SECOND(1))
CA0E A9 02            LDA#   READ     ;TURN ON READ
CA10 9D 82C0          STAX   PB

CA13 98               TYA             ;NOW BUFFER Y
CA14 48               PHA

CA15 A9 99            LDA#   $99      ;SET DECIMAL ACCUMULATOR
CA17 8D 7006          STA    FRAC

CA1A A0 9D   RSW04:   LDY#   $9D      ;SET Y COUNTER
                                      ; VALUE DETERMINED TO BE CORRECT
CA1C 20 50FF RSW02:   JSR    XRET     ; 40 U SEC DELAY
CA1F 20 50FF          JSR    XRET
CA22 20 50FF          JSR    XRET
CA25 20 50FF          JSR    XRET
CA28 BD 80C0          LDAX   PA       ;GET DATA
CA2B 29 0F            AND#   $0F      ;STRIP
CA2D C5 38            CMP    KWSL     ;SAME?
CA2F D0 10            BNE    RSW01    ;DONE IF NOT
CA31 88               DEY             ;COUNT THE TEST
CA32 D0 E8            BNE    RSW02    ;LOOP 'TILL OVERFLOW
CA34 F8               SED             ;ACCUMULATE IN DECIMAL
CA35 AD 7006          LDA    FRAC     ;SETUP SUBTRACT
CA38 38               SEC
CA39 E9 01            SBC#   $01
CA3B D8               CLD             ;NO DECIMAL
CA3C 8D 7006          STA    FRAC
CA3F D0 D9            BNE    RSW04    ;LOOP IF NO OVERFLOW

CA41 68       RSW01:  PLA             ;RESTORE Y
CA42 A8               TAY
CA43 68               PLA             ;RESTORE KWSL
CA44 85 38            STA    KWSL
```

```
CA46 A9 00            LDA#   $00      ;TURN OFF THE READ
CA48 9D 82C0          STAX   PB
CA4B 20 B9CB          JSR    SFTRU    ;MAKE FLOW TRUE
CA4E A9 40            LDA#   SWATCH   ;RESET THE STOPWATCH STATUS
CA50 20 B0CB          JSR    CLRSTA
CA53 4C 16CC          JMP    TIME     ;GO PROCESS WITH THE TIME ROUTINE


                      .LINK  RAM2
```

```
                        .PAGE

                ;  READ THE TIME

CA59 A5 39   READCC:  LDA   KWSH     ;SAVE THE BUFFER
CA5B 48               PHA
CA5C A5 46            LDA   XBUF     ;SAVE THE X BUFFER
CA5E 48               PHA
CA5F A9 05            LDA#  $05      ;SET ADDRESS FOR CLOCK
CA61 85 46            STA   XBUF     ;HOLD IT
CA63 20 BFCB          JSR   SFFAL    ;SPECIAL FLOW TO FALSE

CA66 A9 01   READX:   LDA#  HOLD     ;PUT THE CLOCK INTO HOLD MODE
CA68 9D 02C0          STAX  PB
CA6B A9 04            LDA#  $04      ;DELAY FOR HOLD
CA6D 20 A0FC          JSR   WAIT

CA70 A5 46   RC00:    LDA   XBUF     ;GET ADDRESS
CA72 38               SEC            ;SUBTRACT THE STATE
CA73 F9 F007          SBCY  STATE
CA76 20 03CB          JSR   READC    ;GET THE DATA

CA79 48               PHA            ;HOLD FOR TESTS
CA7A B9 7004          LDAY  STATUS   ;SET STATUS
                                     ;WHICH FLOW PATH?
CA7D 10 0E            BPL   RC01     ;CLEAR IS CLOCK, SET IS CALENDAR

CA7F B9 F007          LDAY  STATE    ;CALENDAR TEST STARTS HERE
CA82 C9 04            CMP#  4        ;IS IT DAY(10)
CA84 D0 1C            BNE   RC02
CA86 68               PLA            ;GET DATA
CA87 29 03            AND#  $03      ;STRIP LEAP YEAR BIT
CA89 48               PHA            ;REPLACE
CA8A 4C A2CA          JMP   RC02     ;BYPASS CLOCK CODE

CA8D B9 F007 RC01:    LDAY  STATE    ;CLOCK CODE STARTS TESTS HERE
CA90 D0 10            BNE   RC02     ;ON PASS 1, DO THE FOLLOWING
CA92 A9 3C            LDA#  C1224+AMPM+ADDDAY+SUBDAY
                                     ;STRIP 1224, AMPM, ADDDAY & SUBDAY
CA94 20 B0CB          JSR   CLRSTA
CA97 68               PLA            ;GET DATA
CA98 48               PHA            ;BUT KEEP ON STACK
CA99 29 0C            AND#  $0C      ;GET AMPM 1224
CA9B 20 A9CB          JSR   SETSTA   ;PUT THEM IN STATUS
CA9E 68               PLA            ;NOW GET DATA
CA9F 29 03            AND#  $03      ;STRIP ALL BUT DATA
CAA1 48               PHA            ;REPLACE

CAA2 B9 F007 RC02:    LDAY  STATE    ;GET STATE
CAA5 6A               RORA           ;BIT 0 TO CARRY
CAA6 B0 0C            BCS   RC03     ;IS ODD, BYPASS MPY
CAA8 68               PLA            ;GET THE DATA
CAA9 0A               ASLA           ;MPY BY 16
CAAA 0A               ASLA
CAAB 0A               ASLA
CAAC 0A               ASLA

CAAD 85 39            STA   KWSH     ;BUFFER THIS
CAAF 20 96CB RC04:    JSR   INCSTA   ;BUMP THE STATE
CAB2 D0 BC            BNE   RC00     ;ALWAYS

CAB4 68      RC03:    PLA            ;GET THE CURRENT READ
CAB5 05 39            ORA   KWSH     ;OR IN THE LAST READ
CAB7 48               PHA            ;HOLD ON THE STACK
CAB8 B9 F007          LDAY  STATE    ;GET THE STATE
CABB C9 03            CMP#  3        ;WHERE ARE WE?
CABD F0 19            BEQ   RC06     ;(MINUTES-MONTHS)
CABF B0 2E            BCS   RC05     ;LAST ONE (SECONDS-DAYS)
                                     ;FALL THRU (HOURS-YEARS)
CAC1 B9 7004          LDAY  STATUS   ;STOPWATCH?
CAC4 29 40            AND#  SWATCH
CAC6 F0 09            BEQ   RC03A    ;BYPASS IF NOT
CAC8 F8               SED            ;DECIMAL
CAC9 68               PLA            ;GET DATA
CACA 38               SEC            ;SETUP SUBTRACT
CACB F9 7006          SBCY  MMYY     ;DELTA TIME
CACE 08               PHP            ;HOLD THE STACK'S STATUS
CACF D8               CLD            ;DROP DECIMAL
CAD0 48               PHA            ;HOLD NEW DATA
CAD1 68      RC03A:   PLA            ;GET DATA

CAD2 99 7006          STAY  MMYY     ;HOLD DATA
CAD5 4C AFCA          JMP   RC04     ;CONTINUE

CAD8 B9 7004 RC06:    LDAY  STATUS   ;STOPWATCH?
CADB 29 40            AND#  SWATCH
CADD F0 09            BEQ   RC06A    ;BYPASS IF NOT
CADF F8               SED            ;DECIMAL
CAE0 68               PLA            ;GET DATA
CAE1 38               SEC            ;SETUP SUBTRACT
CAE2 F9 F006          SBCY  MMMM     ;DELTA TIME
CAE5 08               PHP            ;HOLD THE STACK'S STATUS
CAE6 D8               CLD            ;DROP DECIMAL
CAE7 48               PHA            ;HOLD NEW DATA
CAE8 68      RC06A:   PLA            ;GET DATA

CAE9 99 F006          STAY  MMMM
CAEC 4C AFCA          JMP   RC04

CAEF B9 7004 RC05:    LDAY  STATUS   ;STOPWATCH?
CAF2 29 40            AND#  SWATCH
CAF4 F0 3A            BEQ   RC05A    ;BYPASS IF NOT
CAF6 F8               SED            ;DECIMAL
CAF7 68               PLA            ;GET DATA
CAF8 48               PHA            ;STILL HOLD ON

CAF9 29 0F            AND#  $0F      ;KEEP SECONDS
CAFB 85 38            STA   KWSL     ;STORE FOR CLOCK
CAFD 68               PLA            ;GET WHOLE THING BACK

CAFE 38               SEC            ;SETUP SUBTRACT
CAFF F9 7007          SBCY  SSDD     ;DELTA TIME
CB02 99 7007          STAY  SSDD     ;HOLD THE RESULT
CB05 B0 14            BCS   RCW00    ;NO ERROR
CB07 38               SEC            ;IF ERROR, ADJUST SECONDS
CB08 E9 40            SBC#  $40      ;BY 40
CB0A 99 7007          STAY  SSDD     ;RESTORE

CB0D 38               SEC
CB0E B9 F006          LDAY  MMMM     ;BORROW ONE MINUTE
CB11 E9 01            SBC#  $01
CB13 99 F006          STAY  MMMM
CB16 B0 03            BCS   RCW00    ;NO ERROR - CONTINUR
CB18 20 46CB          JSR   RCWDM    ;DECREMENT MINUTE

CB1B 28      RCW00:   PLP            ;GET THE STACK AFTER MINUTES
CB1C B0 03            BCS   RCW01    ;NO ERROR HERE
CB1E 20 46CB          JSR   RCWDM    ;ERROR - DEC MONTH

CB21 B0 03   RCW01:   BCS   RCW02    ;NO ERROR - CONTINUE
CB23 20 59CB          JSR   RCWDH    ;ERROR - DEC HOUR

CB26 28      RCW02:   PLP            ;STATUS AFTER HOURS
CB27 B0 03            BCS   RCW03    ;NO ERROR - CONTINUE
CB29 20 59CB          JSR   RCWDH    ;ERROR - DEC HOUR

CB2C D8      RCW03:   CLD            ;NO MORE DECIMAL
CB2D 4C 34CB          JMP   RC05B    ;CONTINUE PROCESSING

CB30 68      RC05A:   PLA            ;GET DATA

CB31 99 7007          STAY  SSDD
CB34 A9 00   RC05B:   LDA#  $00      ;RESET THE STATE
CB36 99 F007          STAY  STATE
CB39 9D 02C0          STAX  PB        ;TURN HOLD OFF
CB3C 20 BFCB          JSR   SFFAL     ;EXIT WITH FLOW FALSE
                                      ; (OTHERWISE WE WILL READ IT AGAIN)
CB3F 68               PLA             ;RESTORE THE X BUFFER
CB40 85 46            STA   XBUF
CB42 68               PLA             ;GET BACK THE BUFFER
CB43 85 39            STA   KWSH
CB45 60               RTS             ;THE TIME HAS BEEN READ

CB46 B9 F006 RCWDM:   LDAY  MMMM      ;GET MINUTES
CB49 38               SEC
CB4A E9 40            SBC#  $40       ;ADJUST BY 40
CB4C 99 F006          STAY  MMMM
CB4F 38               SEC
CB50 B9 7006          LDAY  MMYY      ;BORROW AN HOUR
CB53 E9 01            SBC#  $01
CB55 99 7006          STAY  MMYY
CB58 60               RTS

CB59 B9 7004 RCWDH:   LDAY  STATUS    ;12 OR 24 (DIFFERENT ADJUST VALUES)
CB5C 29 08            AND#  C1224
CB5E F0 09            BEQ   RCWDH0    ;DO 12 HOUR ADJUST

CB60 B9 7006          LDAY  MMYY      ;GET HOURS
CB63 38               SEC
CB64 E9 76            SBC#  $76       ;ADJUST
CB66 4C 6FCB          JMP   RCWDH1

CB69 B9 7006 RCWDH0:  LDAY  MMYY      ;GET HOURS
CB6C 38               SEC
CB6D E9 88            SBC#  $88       ;ADJUST

CB6F 99 7006 RCWDH1:  STAY  MMYY      ;RESTORE

CB72 60               RTS

                ;  READ THE CALENDAR

CB73 A5 39   READCD:  LDA   KWSH      ;SAVE THE BUFFER
CB75 48               PHA
CB76 A5 46            LDA   XBUF      ;SAVE THE X BUFFER
CB78 48               PHA
CB79 A9 0C            LDA#  $0C       ;CLOCK ADDRESS IS #C-STATE
CB7B 85 46            STA   XBUF      ;HOLD FOR ABOVE CODE
CB7D 20 B9CB          JSR   SFTRU     ;SPECIAL FLOW TO CALENDAR
CB80 4C 66CA          JMP   READX     ;USE THE ABOVE CODE
```

```
                        .PAGE
                ; GENERAL UTILITY ROUTINES

                ;  THE FOLLOWING ROUTINE READS THE DATA
                ;  FROM THE CLOCK.  ENTER WITH THE CHIP
                ;  ADDRESS IN A, EXIT WITH THE DATA IN A.

C803 0A         READC:  ASLA            ;MAKE CLOCK ADDRESS
C804 0A                 ASLA
C805 0A                 ASLA
C806 0A                 ASLA
C807 09 03              ORA#  READ+HOLD ;MAKE COMMAND
C809 9D 82C0            STAX  PB        ;DO THE COMMAND
C80C BD 80C0            LDAX  PA        ;GET THE DATA
C80F 29 0F              AND#  $0F       ;STRIP THE BAD STUFF
C811 60                 RTS             ;GO TO CALLER

C812 09 B0      ASCII:  ORA#  '0+$80    ;OR IN AN ASCII ZERO
C814 85 45      ASCII1: STA   ABUF      ;HOLD IN THE RETURN BUFFER
                                        ;FALL THRU TO BUMP STATE

                ; INCREMENT THE STATE VARIABLE

C816 B9 F807 INCSTA:  LDAY  STATE       ;GET STATE
C819 18               CLC               ;INCREMENT IT
C81A 69 01            ADC#  $01
C81C 99 F807          STAY  STATE       ;PUT IT BACK
C81F 60               RTS

                ; END AN INPUT OPERATION

C820 A9 0D      FINISH: LDA#  $0D       ;RETURN A <CR>
C822 85 45              STA   ABUF
C824 A9 00              LDA#  00        ;RESET INPUT
C826 85 38              STA   KNSL
C828 60                 RTS

C829 19 7004 SETSTA:  ORAY  STATUS      ;SET THE A INTO STATUS
C82C 99 7004          STAY  STATUS
C82F 60               RTS

C830 49 FF      CLRSTA: EOR#  $FF       ;COMPLEMENT THE A
C832 39 7004            ANDY  STATUS    ;CLEAR THE STATUS OF A'S NULL BITS
C835 99 7004            STAY  STATUS
C838 60                 RTS

C839 A9 80      SFTRU:  LDA#  SFLON      ;SET THE SPECIAL FLOW BIT
C83B 20 A9CB            JSR   SETSTA
C83E 60                 RTS

C83F A9 80      SFFAL:  LDA#  SFLON      ;RESET THE SPECIAL FLOW BIT
C841 20 B0CB            JSR   CLRSTA
C844 60                 RTS
```



```
                        .PAGE
                ; LOAD THE OFFSET

C845 B9 F807 OFFOFF: LDAY  STATE        ;IS FIRST BYTE?
C848 D0 12           BNE   OFF000
C84A A9 03           LDA#  OFFSGN+DATLIN ;STRIP THE SIGN AND DATLIN
C84C 20 B0CB         JSR   CLRSTA
C84F A5 45           LDA   ABUF         ;GET THE SIGN
C851 C9 AB           CMP#  '++$80       ;IS A "+"
C853 F0 3E           BEQ   OFF010       ;IF SO IT IS OK NOW
C855 A9 01           LDA#  OFFSGN       ;ELSE STORE THE "-"
C857 20 A9CB         JSR   SETSTA
C85A D0 37           BNE   OFF010       ;ALWAYS

C85C C9 01      OFF000: CMP#  $01       ;FIRST AFTER SIGN?
C85E D0 0A              BNE   OFF001
C860 A5 45              LDA   ABUF      ;IS A <D>?
C862 C9 C4              CMP#  'D+$80
C864 D0 09              BNE   OFF01A
C866 A9 02              LDA#  DATLIN    ;IF SO, STORE THE DATE LINE BIT
C868 D0 0F              BNE   SETSTA    ;EXIT THIS WAY, NO INCREMENT OF STATE

C86A 6A         OFF001: RORA            ;IS STATE ODD?
C86B 90 0C              BCC   OFF002    ;IF NOT WE HAVE LOWER NIBBLE
C86D A5 45              LDA   ABUF      ;GET THE HI NIBBLE
C86F 0A         OFF01A: ASLA            ;MPY BY 16
C870 0A                 ASLA
C871 0A                 ASLA
C872 0A                 ASLA
C873 99 F805            STAY  OFFTMP    ;HOLD THE RESULT
C876 4C 96CB            JMP   INCSTA    ;GO ON

C879 A5 45      OFF002: LDA   ABUF      ;GET THE LO NIBBLE
C87B 29 0F              AND#  $0F       ;ISOLATE THE LOW NIBBLE
C87D 19 F805            ORAY  OFFTMP    ;PUT THEM TOGETHER
C880 48                 PHA             ;HANG ON

C881 B9 F807            LDAY  STATE     ;IS STATE 4?
C884 C9 04              CMP#  $04
C886 D0 07              BNE   OFF003    ;IF NOT STORE HOURS
C888 68                 PLA             ;IF SO STORE MIN
C889 99 F805            STAY  OFFMIN
  8C 4C 1DC9            JMP   DONE      ;THAT'S ALL WE NEED

  8F 68         OFF003: PLA
C890 99 7005            STAY  OFFHRS    ;STORE HOURS

C893 4C 96CB OFF010:  JMP   INCSTA      ;GO BACK
```



```
                        .PAGE


                ;  THE FOLLOWING CODE FORMATS THE TIME
                ;  OF DAY OR STOPWATCH.  (3 WAYS)

CC16 B9 F807 TIME:    LDAY  STATE       ;GET STATE
CC19 D0 1C            BNE   TIM00       ;BYPASS SETUP AFTER FIRST ENTRY
CC1B B9 F804          LDAY  TYPE        ;DON'T INIT ON STOPWATCH EITHER
CC1E 29 10            AND#  TRSW
CC20 D0 15            BNE   TIM00
CC22 B9 7004          LDAY  STATUS      ;GET STATUS
                                        ;IS FLOW ON?
CC25 10 10            BPL   TIM00       ;CLOCK IS READ IF FALSE
CC27 20 59CA          JSR   READCC      ;READ THE CLOCK
CC2A 20 B9CB          JSR   SFTRU       ;SET FLOW

CC2D B9 F804          LDAY  TYPE        ;SEE IF OFFSET DESIRED
CC30 29 08            AND#  $08
CC32 F0 03            BEQ   TIM00
CC34 20 20CE          JSR   DOTOFF      ;DO THE TIME OFFSET

CC37 B9 F807 TIM00:   LDAY  STATE       ;GET STATE
CC3A C9 06            CMP#  $06         ;END?
CC3C 30 59            BMI   TIM01
CC3E B9 F804          LDAY  TYPE        ;SEE IF STOPWATCH
CC41 29 10            AND#  $10
CC43 F0 1F            BEQ   TIM00A      ;NO - DO NORMAL END STUFF

CC45 B9 F807          LDAY  STATE       ;YES - SEE WHERE WE ARE
CC48 C9 07            CMP#  $07
CC4A F0 0F            BEQ   TIS00       ;AT 7, SIMPLE OUTPUT
CC4C 10 41            BPL   TIM04A      ;AT 8, FINISH
CC4E B9 7004          LDAY  STATUS      ;AT 6, SEE IF WE NEED A DOT
CC51 10 08            BPL   TIS00       ;IF ALREADY OUT, DO SIMPLE OUTPUT
CC53 20 BFCB          JSR   SFFAL       ;TURN OFF FLAG
CC56 A9 AE            LDA#  '.+$80      ;<.>
CC58 85 45            STA   ABUF        ;STORE FOR RETURN
CC5A 60               RTS               ;OUT WITHOUT STATE INCREMENT

CC5B 20 B9CB TIS00:   JSR   SFTRU       ;FLAG ON
CC5E AD 7006          LDA   FRAC        ;GET FRACTION
CC61 4C E6CC          JMP   TIM12       ;GO OUTPUT IT


CC64 B9 7004 TIM00A:  LDAY  STATUS      ;IS AM/PM?
CC67 29 08            AND#  C1224
CC69 D0 24            BNE   TIM04A      ;IF NOT, FINISH

CC6B B9 F807 TIM02:   LDAY  STATE       ;WHERE ARE WE?
CC6E C9 07            CMP#  $07
CC70 F0 06            BEQ   TIM03       ;=7, DO <A> OR <P>
CC72 B0 14            BCS   TIM04       ;>7, CHECK WHERE
CC74 A9 A0            LDA#  ' +$80      ;<7, DO SPACE
CC76 D0 0D            BNE   TIM06       ;ALWAYS

CC78 B9 7004 TIM03:   LDAY  STATUS      ;SEE IF A OR P
CC7B 29 04            AND#  AMPM
CC7D F0 04            BEQ   TIM05
CC7F A9 D0            LDA#  'P+$80      ;DO <P>
CC81 D0 02            BNE   TIM06       ;ALWAYS

CC83 A9 C1    TIM05:  LDA#  'A+$80      ;DO <A>
CC85 4C 94CB TIM06:   JMP   ASCII1

CC88 B9 F807 TIM04:   LDAY  STATE       ;WHERE?
CC8B C9 08            CMP#  $08
CC8D F0 03            BEQ   TIM07       ;AT 8 DO <M>
CC8F 4C A0CB TIM04A:  JMP   FINISH      ;ELSE END

CC92 A9 CD    TIM07:  LDA#  'M+$80      ;<M>
CC94 4C 94CB TIM07A:  JMP   ASCII1

CC97 B9 F807 TIM01:   LDAY  STATE       ;NEED WE CHECK FOR LEADING QUOTE?
CC9A D0 13            BNE   TIM00       ;NO
CC9C B9 7004          LDAY  STATUS      ;HAS IT BEEN SENT?
CC9F 10 0E            BPL   TIM00       ;YES
CCA1 20 BFCB          JSR   SFFAL       ;TURN OFF FLOW
CCA4 B9 F804          LDAY  TYPE        ;OK, IS IT APPLESOFT?
CCA7 6A               RORA
CCA8 90 05            BCC   TIM00       ;OH, WELL
CCAA A9 A2            LDA#  '"+$80      ;<">
CCAC 85 45            STA   ABUF        ;HOLD THE BYTE
CCAE 60               RTS               ;GO BACK WITH NO STATE INCREMENT

CCAF B9 F807 TIM00:   LDAY  STATE       ;IS ODD
CCB2 6A               RORA
CCB3 B0 14            BCS   TIM09       ;YES, SKIP
CCB5 B9 7004          LDAY  STATUS      ;NO, IS IT TIME FOR A COLON
CCB8 10 12            BPL   TIM10
CCBA B9 F804          LDAY  TYPE        ;OK, IS IT FORMATTED?
CCBD 29 03            AND#  $03
CCBF F0 0B            BEQ   TIM10       ;NOPE
CCC1 20 BFCB          JSR   SFFAL       ;FLOW OFF
CCC4 A9 BA            LDA#  ':+$80      ;<:>
CCC6 85 45            STA   ABUF        ;STORE
CCC8 60               RTS               ;BACK

CCC9 20 B9CB TIM09:   JSR   SFTRU       ;SET FLOW

CCCC B9 F807 TIM10:   LDAY  STATE       ;WHERE ARE WE?
CCCF C9 02            CMP#  $02         ;0 OR 1?
CCD1 10 06            BPL   TIM11
CCD3 B9 7006          LDAY  MMYY        ;GET HOURS
CCD6 4C E6CC          JMP   TIM12

CCD9 C9 04    TIM11:  CMP#  $04         ;2 OR 3?
CCDB 10 06            BPL   TIM13
CCDD B9 F806          LDAY  MMMM        ;GET MINUTES
CCE0 4C E6CC          JMP   TIM12

CCE3 B9 7007 TIM13:   LDAY  SSDD        ;GET SECONDS

CCE6 48       TIM12:  PHA               ;BUFFER THE DATA
CCE7 B9 F807          LDAY  STATE       ;FIND THE DIGIT
```

```
CCEA 6A                 RORA
CCEB B0 06              BCS     TIM16
CCED 68         TIM15:  PLA             ;GET DATA
CCEE 6A                 RORA            ;DIVIDE BY 16
CCEF 6A                 RORA
CCF0 6A                 RORA
CCF1 6A                 RORA
CCF2 48                 PHA             ;PUSH SO WE CAL FALL THRU

CCF3 68         TIM16:  PLA             ;GET DATA
CCF4 29 0F      TIM14:  AND#    #0F     ;STRIP HI STUFF
CCF6 4C 92CB            JMP     ASCII   ;GO DO IT
```

```
CD55 B9 7006 DAT10:     LDAY    HHYY    ;YEAR
CD58 48         DAT07:  PHA             ;BUFFER IT
CD59 B9 F807            LDAY    STATE   ;WHICH HALF?
CD5C F0 0F              BEQ     TIM15   ;0, 3 OR 6?
CD5E C9 03              CMP#    #03
CD60 F0 0B              BEQ     TIM15
CD62 C9 06              CMP#    #06
CD64 F0 07              BEQ     TIM15
CD66 D0 0B              BNE     TIM16   ;1,2,4 OR 5

                ; RETURN DATE FORMAT 1

CD68 B9 F807 DAT03:     LDAY    STATE   ;WHERE ARE WE?
CD6B C9 09              CMP#    #09     ;DONE?
CD6D F0 82              BEQ     DAT01A

CD6F C9 02      DAT12:  CMP#    #02     ;TIME FOR DASH? (2 OR 6)
CD71 F0 04              BEQ     DAT16
CD73 C9 06              CMP#    #06
CD75 D0 04              BNE     DAT13
CD77 A9 AD      DAT16:  LDA#    '-+$80  ;(-)
CD79 D0 C3              BNE     DAT09A

CD7B C9 02      DAT13:  CMP#    #02     ;0 OR 1?
CD7D 10 06              BPL     DAT14
CD7F B9 7007            LDAY    SSDD    ;GET DAYS
CD82 4C D9CD            JMP     DAT17

CD85 C9 06      DAT14:  CMP#    #06     ;3,4 OR 5?
CD87 10 4D              BPL     DAT15
CD89 8A                 TXA             ;GET X
CD8A 48                 PHA             ;HOLD IT AS INDEXING IS NEEDED
CD8B B9 F806            LDAY    MMMM    ;GET THE MONTH
CD8E C9 0F              CMP#    #0F     ;IS THE DECIMAL TEN ON?
CD90 30 02              BMI     DAT14A
CD92 E9 06              SBC#    #06     ;ADJUST (THE C IS CLEAR UPON ENTRY)
CD94 99 F806 DAT14A:    STAY    MMMM    ;AND HOLD IT HERE

CD97 B9 F806 DAT18:     LDAY    MMMM    ;GET MONTH
CD9A 0A                 ASLA            ; MONTHS*2
CD9B 18                 CLC
CD9C 79 F806            ADCY    MMMM    ; MONTHS*3
CD9F 79 F807            ADCY    STATE   ; MONTHS*3+STATE
CDA2 38                 SEC
CDA3 E9 06              SBC#    #06     ; MONTHS*3+STATE-6
CDA5 AA                 TAX             ;MAKE INDEX
CDA6 BD B2CD            LDAX    MONTHS  ;GET BYTE
CDA9 09 80              ORA#    #80     ;HI BIT
CDAB 85 45              STA     ABUF    ;HOLD IT FOR THE OUTPUT
CDAD 68                 PLA             ;GET THE BUFFERED X
CDAE AA                 TAX             ;PUT IT BACK
CDAF 4C 96CB            JMP     INCSTA  ;EXIT

CDB2 4A 414E MONTHS:    .ASCII "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"

CDD6 B9 7006 DAT15:     LDAY    HHYY    ;YEAR
CDD9 48         DAT17:  PHA             ;HOLD THE DATA
CDDA B9 F807            LDAY    STATE   ;SEE WHERE WE ARE
```

```
CDDD F0 07              BEQ     DAT19   ;ON 0 AND 7 USE TIME15
CDDF C9 07              CMP#    #07
CDE1 F0 03              BEQ     DAT19
CDE3 4C F3CC            JMP     TIM16   ;ELSE USE TIME 16

CDE6 4C EDCC DAT19:     JMP     TIM15

                        .LINK   RAM1    ;FOR NEXT MODULE
```

```
*** RSM65 ***  (V3.0-AP)  OCT-79

                        .PAGE

                ; THE FOLLOWING FORMATS THE DATE, ONE
                ; OF THREE WAYS.

CCF9 B9 F807 DATE:      LDAY    STATE   ;GET STATE
CCFC D0 15              BNE     DAT00   ;BYPASS SETUP AFTER FIRST ENTRY
CCFE B9 7004            LDAY    STATUS  ;GET STATUS
                                        ;IS FLOW ON?
CD01 10 10              BPL     DAT00   ;CLOCK IS READ IF FALSE
CD03 20 73CB            JSR     READCD  ;READ THE CALENDAR
CD06 B9 F804            LDAY    TYPE    ;GET TYPE
CD09 29 10              AND#    #10     ;ANY OFFSET?
CD0B F0 03              BEQ     DAT00A  ;SKIP IF NONE
CD0D 20 06CF            JSR     ADJDAY  ;GO ADJUST THE DAY
CD10 20 09CB DAT00A:    JSR     SFTRU   ;SET FLOW

CD13 B9 F804 DAT00:     LDAY    TYPE    ;WHAT KIND OF DATE?
CD16 29 04              AND#    #04     ;EITHER FORMAT?
CD18 D0 0A              BNE     DAT02   ;IF SO, GO ON

                ; RETURN THE UNFORMATTED DATE

CD1A B9 F807            LDAY    STATE   ;WHERE ARE WE?
CD1D C9 06              CMP#    #06
CD1F D0 A8              BNE     TIM10   ;USE TIME FOR UNFORMATTED RETURN
CD21 4C A0CB DAT01A:    JMP     FINISH  ;ELSE END


CD24 B9 F804 DAT02:     LDAY    TYPE    ;GET TYPE
CD27 6A                 RORA            ;PUT LSB IN C
CD28 90 3E              BCC     DAT03   ;GO DO IT

                ; RETURN DATE FORMAT 2

CD2A B9 F807            LDAY    STATE   ;WHERE ARE WE?
CD2D C9 08              CMP#    #08     ;DONE?
CD2F F0 F8              BEQ     DAT01A  ;YES

CD31 B9 F807 DAT04:     LDAY    STATE   ;TIME FOR SLASH?
CD34 C9 02              CMP#    #02     ;2 OR 5
CD36 F0 04              BEQ     DAT09
CD38 C9 05              CMP#    #05
CD3A D0 05              BNE     DAT05
CD3C A9 AF      DAT09:  LDA#    '/+$80  ;(/)
CD3E 4C 94CB DAT09A:    JMP     ASCII1

CD41 C9 02      DAT05:  CMP#    #02     ;0 OR 1?
CD43 10 06              BPL     DAT06
CD45 B9 F806            LDAY    MMMM    ;GET MONTH
CD48 4C 58CD            JMP     DAT07

CD4B C9 05      DAT06:  CMP#    #05     ;3 OR 4
CD4D 10 06              BPL     DAT10
CD4F B9 7007            LDAY    SSDD    ;DAY
CD52 4C 58CD            JMP     DAT07
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                        .PAGE

                ; CLOCK SETTING CODE

C*   B9 F807 SET:     LDAY    STATE     ;FIRST PASS?
     D0 14           BNE     SET00     ;NO
     A9 00           LDA0    #00       ;SETUP PIA FOR WRITES
CDF8 9D 01C0         STAX    PAC
CDF3 A9 0F           LDA0    #0F       ;PORT A (LOWER 4) OUTPUT
CDF5 9D 00C0         STAX    PA
CDF8 A9 04           LDA0    #04       ;MAKE PORT A IO AGAIN
CDFA 9D 01C0         STAX    PAC
CDFD A9 01           LDA0    HOLD      ;SET THE HOLD BIT
CDFF 9D 02C0         STAX    PB

CE02 A5 45   SET00:  LDA     ABUF      ;GET THE CHARACTER
CE04 9D 00C0         STAX    PA        ;OUTPUT IT (OUTPUT IS ONLY DONE ON LOWER 4
CE07 A9 0C           LDA0    #0C       ;CLOCK ADDRESS IS C-STATE
CE09 38              SEC
CE0A F9 F807         SBCY    STATE
CE0D 0A              ASLA              ;TIMES 16
CE0E 0A              ASLA
CE0F 0A              ASLA
CE10 0A              ASLA
CE11 09 05           ORA0    HOLD+WRITE ;PUT IN THE COMMAND
CE13 9D 02C0         STAX    PB        ;DO IT
CE16 A9 01           LDA0    HOLD      ;TOGGLE OFF THE WRITE
CE18 9D 02C0         STAX    PB

CE1B B9 F807         LDAY    STATE     ;ARE WE DONE?
CE1E C9 0C           CMP0    #0C
CE20 D0 03           BNE     SET10     ;NO, CONTINUE
CE22 4C 1DC9         JMP     DONE

CE25 4C 96CB SET10:  JMP     INCSTA
```

```
*** ASM65 ***  (V3.0-AP)  OCT-79

                        .PAGE

                ; THIS ROUTINE APPLIES THE OFFSET TO
                ; THE TIME IN THE BUFFER

CE28 F8      DOTOFF: SED               ;DECIMAL MODE ON
CE29 B9 7804         LDAY    STATUS    ;GET STATUS
CE2C 29 01           AND0    OFFSGN    ;+ OR -
CE2E D0 7B           BNE     DOT10     ;-

CE30 B9 F806         LDAY    MMMM      ;GET OUR MINUTES
CE33 18              CLC
CE34 79 F805         ADCY    OFFMIN    ;ADD THE OFFSET
CE37 99 F806         STAY    MMMM      ;STORE THE RESULT
CE3A 50 05           BVC     DOT00     ;NO OVERFLOW, GO CHECK
CE3C 18              CLC               ;ADJUST FOR OVERFLOW
CE3D 69 40           ADC0    #40
CE3F D0 07           BNE     DOT01     ;GO ON

CE41 C9 60   DOT00:  CMP0    #60       ;HOUR WRAP?
CE43 30 09           BMI     DOT02
CE45 38              SEC               ;ADJUST THE HOUR
CE46 E9 60           SBC0    #60
CE48 99 F806 DOT01:  STAY    MMMM      ;STORE THE MINUTE
CE4B 38              SEC               ;SETUP HOUR INCREMENT
CE4C B0 01           BCS     DOT03

CE4E 18      DOT02:  CLC               ;SETUP ADD FOR HOURS
CE4F B9 7806 DOT03:  LDAY    HHYY      ;GET HOURS
CE52 79 7805         ADCY    OFFHRS    ;ADD THE OFFSET
CE55 99 7806         STAY    HHYY      ;HOLD

CE58 C9 24           CMP0    #24       ;IS OVER AT 24?
CE5A 30 19           BMI     DOT04     ;OK, CHECK 12/24
CE5C 38              SEC               ;LOWER BY 24
CE5D E9 24           SBC0    #24
CE5F 99 7806         STAY    HHYY      ;HOLD
CE62 A9 10   DOT07:  LDA0    ADDDAY    ;PUT IN THE ADD DAY BIT
CE64 20 A9CB         JSR     SETSTA
CE67 D8      DOT05:  CLD               ;DROP DECIMAL
CE68 B9 7804         LDAY    STATUS    ;SEE IF THE DATE LINE HAS BEEN CROSSED
CE6B 29 02           AND0    DATLIN
CE6D F0 05           BEQ     DOT05A    ;IF NOT, EXIT
CE6F A9 20           LDA0    SUBDAY    ;IF SO, FORCE SUB DAY
CE71 20 A9CB         JSR     SETSTA

     60      DOT05A: RTS               ;GO BACK

CE75 C9 12   DOT04:  CMP0    #12       ;IS 12?
CE77 30 EE           BMI     DOT05     ;<12, EXIT
CE79 D0 15           BNE     DOT06     ;>12, PROCESS
                                       ;=12, FALL THRU

                ; AT EXACTLY 12, WE ASSUME THAT WE MUST
                ; CHANGE THE AM/PM STUFF.  IF NEW IS AM,
                ; THEN SET ADDDAY.

CE7B B9 7804         LDAY    STATUS    ;GET STATUS
CE7E 49 04           EOR0    AMPM      ;CHANGE STATE OF AMPM
CE80 99 7804         STAY    STATUS    ;PUT IT BACK
CE83 29 08           AND0    C1224     ;SEE IF REALLY 12/24
CE85 D0 E0           BNE     DOT05     ;IF 24, GO BACK
CE87 B9 7804         LDAY    STATUS    ;OK, SEE IF AM NOW
CE8A 29 04   DOT08:  AND0    AMPM
CE8C D0 D9           BNE     DOT05     ;IF PM, RETURN
CE8E F0 D2           BEQ     DOT07     ;IF AM, BUMP DAY

CE90 B9 7804 DOT06:  LDAY    STATUS    ;SEE IF 12
CE93 29 08           AND0    C1224
CE95 D0 D0           BNE     DOT05     ;IF 24, EXIT
CE97 B9 7806         LDAY    HHYY      ;GET HOURS
CE9A 38              SEC
CE9B E9 12           SBC0    #12       ;LOWER BY 12
CE9D 99 7806         STAY    HHYY      ;STORE
CEA0 B9 7804         LDAY    STATUS    ;GET STATUS
CEA3 49 04           EOR0    AMPM      ;CHANGE AM/PM BIT
CEA5 99 7804         STAY    STATUS    ;PUT BACK
CEA8 4C 8ACE         JMP     DOT08     ;GO CHECK IF DAY NEEDS BUMPING
```

```
                ; MINUS OFFSET

CEAB B9 F806 DOT10:  LDAY    MMMM      ;MINUTES
CEAE 38              SEC
CEAF F9 F805         SBCY    OFFMIN    ;SUBTRACT OFFSET
CEB2 99 F805         STAY    OFFMIN
CEB5 10 09           BPL     DOT11     ;NO OVERFLOW
CEB7 38              SEC               ;ADJUST BY 40
CEB8 E9 40           SBC0    #40
CEBA 99 F806         STAY    MMMM
CEBD 18              CLC               ;SETUP BORROW
CEBE 90 01           BCC     DOT12

CEC0 38      DOT11:  SEC               ;SETUP HOUR SUBTRACT
CEC1 B9 7806 DOT12:  LDAY    HHYY      ;GET OUR HOURS
CEC4 F9 7805         SBCY    OFFHRS    ;SUBTRACT OFFSET
CEC7 99 7806         STAY    HHYY      ;PUT BACK
CECA 10 15           BPL     DOT15     ;NO OVERFLOW?, EXIT

CECC B9 7804         LDAY    STATUS    ;ARE WE 12 HOUR?
CECF 29 08           AND0    C1224
CED1 F0 1C           BEQ     DOT13     ;GO DO THAT
CED3 38              SEC               ;ADJUST BY 79
CED4 B9 7806         LDAY    HHYY
CED7 E9 76           SBC0    #76
CED9 99 7806         STAY    HHYY

CEDC A9 20   DOT14:  LDA0    SUBDAY    ;PUT IN THE SUB DAY BIT
CEDE 20 A9CB         JSR     SETSTA
CEE1 D8      DOT15:  CLD               ;NO MORE DECIMAL
CEE2 B9 7804         LDAY    STATUS    ;SEE IF THE DATE LINE HAS BEEN CROSSED
CEE5 29 02           AND0    DATLIN
CEE7 F0 05           BEQ     DOT15A    ;IF NOT, EXIT
CEE9 A9 10           LDA0    ADDDAY    ;IF SO, FORCE ADD DAY
CEEB 20 A9CB         JSR     SETSTA

CEEE 60      DOT15A: RTS               ;EXIT
```

```
CEEF 38      DOT13:  SEC               ;ADJUST BY 88
CEF0 B9 7806         LDAY    HHYY
CEF3 E9 88           SBC0    #88
CEF5 99 7806         STAY    HHYY
CEF8 B9 7804         LDAY    STATUS    ;FLIP AM/PM
CEFB 49 04           EOR0    AMPM
CEFD 99 7804         STAY    STATUS
CF00 29 04           AND0    AMPM      ;IS AM NOW?
CF02 D0 D8           BNE     DOT14     ;IF SO, SUB DAY
CF04 F0 DB           BEQ     DOT15     ;IF NOT, EXIT
```

```
                        .PAGE
                ; ADJUSTING THE CALENDAR DATE CODE

CF06 F8         ADJDAY: SED             ;SET DECIMAL
CF07 8A                 TXA             ;HOLD X FOR THIS ROUTINE
CF08 48                 PHA
CF09 B9 7004            LDAY    STATUS  ;SEE IF NEED TO ADD DAY
CF0C 29 10              AND#    ADDDAY
CF0E F0 3C              BEQ     ADJ00   ;NO

CF10 18                 CLC
CF11 B9 7007            LDAY    SSDD    ;GET DAYS
CF14 69 01              ADC#    #01     ;ADD 1
CF16 99 7007            STAY    SSDD    ;STORE RESULT

CF19 20 0BCF            JSR     AMONTH  ;GET THE DAY

CF1C D9 7007            CMPY    SSDD    ;HOW DID WE DO?
CF1F 10 1C              BPL     ADJ03   ;OUT

CF21 18                 CLC
CF22 A9 01              LDA#    #01
CF24 99 7007            STAY    SSDD    ;RESET DAY
CF27 79 F806            ADCY    MMMM    ;BUMP MONTH
CF2A 99 F806            STAY    MMMM
CF2D C9 13              CMP#    #13     ;TOO FAR?
CF2F D0 0C              BNE     ADJ03

CF31 18                 CLC
CF32 A9 01              LDA#    #01
CF34 99 F806            STAY    MMMM    ;RESET MONTH
CF37 79 7006            ADCY    MMYY    ;BUMP YEAR
CF3A 99 7006            STAY    MMYY

CF3D 20 A6CF ADJ03:     JSR     LEAP    ;TEST THE LEAP YEAR
CF40 F0 0A              BEQ     ADJ00   ;IF RETURNS 0, IS OK
CF42 A9 01              LDA#    #01     ;SET THE DATE TO MARCH 1
CF44 99 7007            STAY    SSDD
CF47 A9 03              LDA#    #03
CF49 99 F806            STAY    MMMM

CF4C B9 7004 ADJ00:     LDAY    STATUS  ;SEE IF NEED TO SUB A DAY
CF4F 29 20              AND#    SUBDAY
CF51 F0 34              BEQ     ADJ01

CF53 38                 SEC             ;DROP THE DAY
CF54 B9 7007            LDAY    SSDD
CF57 E9 01              SBC#    #01
CF59 99 7007            STAY    SSDD
CF5C D0 29              BNE     ADJ01   ;NO WRAP, DONE

CF5E 38                 SEC             ;DROP MONTH
CF5F B9 F806            LDAY    MMMM
CF62 E9 01              SBC#    #01
CF64 99 F806            STAY    MMMM



CF67 D0 0E              BNE     ADJ04   ;NO WRAP, SO DO DAY

CF69 A9 12              LDA#    #12     ;DO DECEMBER
CF6B 99 F806            STAY    MMMM
CF6E 38                 SEC
CF6F B9 7006            LDAY    MMYY    ;DROP YEAR
CF72 E9 01              SBC#    #01
CF74 99 7006            STAY    MMYY    ;A WRAP HERE IS NOT PROCESSED

CF77 20 0BCF ADJ04:     JSR     AMONTH  ;GET MONTH
CF7A 99 7007            STAY    SSDD    ;HOLD IN DAY

CF7D 20 A6CF            JSR     LEAP    ;SEE ABOUT LEAP YEAY
CF80 F0 05              BEQ     ADJ01
CF82 A9 28              LDA#    #28     ;MAKE FEB 28
CF84 99 7007            STAY    SSDD

CF87 D8         ADJ01:  CLD             ;DROP DECIMAL
CF88 68                 PLA             ;RESTORE X
CF89 AA                 TAX
CF8A 60                 RTS             ;BACK


CF8B D8         AMONTH: CLD             ;CLEAR DECIMAL FOR THIS
CF8C B9 F806            LDAY    MMMM    ;GET MONTH
CF8F 29 10              AND#    #10     ;HAS A 10?
CF91 F0 0A              BEQ     AM000   ;NO, GO ON
CF93 B9 F806            LDAY    MMMM    ;GET MONTH
CF96 29 0F              AND#    #0F     ;SANS 10
CF98 18                 CLC
CF99 69 0A              ADC#    #0A     ;ADD A NORMAL DAY
CF9B D0 03              BNE     AM001   ;EXIT

CF9D B9 F806 AM000:     LDAY    MMMM    ;GET THE MONTH
CFA0 F8      AM001:     SED             ;BACK TO DECIMAL
CFA1 AA                 TAX             ;MONTH IS NOW INDEX
CFA2 BD C6CF            LDAX    MONLEN  ;GET THE LAST DAY OF THIS MONTH
CFA5 60                 RTS             ;GO BACK

CFA6 B9 F006 LEAP:      LDAY    MMMM    ;GET MONTHS
CFA9 C9 02              CMP#    #02     ;FEB?
CFAB D0 17              BNE     LEA00   ;IF NOT, ALL OK
CFAD B9 7007            LDAY    SSDD    ;DAYS
CFB0 C9 29              CMP#    #29     ;29?
CFB2 D0 10              BNE     LEA00   ;IF NOT, ALL OK
CFB4 18                 CLC             ;GET READY TO ADD
CFB5 B9 7006            LDAY    MMYY    ;YEAR
CFB8 29 10              AND#    #10     ;IS YEAR(10) ODD?
CFBA F0 02              BEQ     LEA01
CFBC A9 02              LDA#    #02     ;IF SO, ADD A 2 TO TEAR(1)
CFBE 79 7006 LEA01:     ADCY    MMYY    ;NOW HAVE NON-BCD IN LOW 2 BITS
CFC1 29 03              AND#    #03     ;IS YEAR EVENLY DIVISIBLE BY 4?
CFC3 60                 RTS             ;THE Z BIT TELLS THE ANSWER

CFC4 A9 00 . LEA00:     LDA#    #00     ;FORCE THE Z BIT OFF
CFC6 60                 RTS


                        .DEF    MONLEN=.-1 ;ADDRESS FOR INDEXING
```

```
CFC7 31                 .BYTE   #31     ;TABLE OF BDC MONTH LENGTHS
CFC8 29                 .BYTE   #29     ;NOTE IMPLIED LEAP YEAR
CFC9 31                 .BYTE   #31
CFCA 30                 .BYTE   #30
CFCB 31                 .BYTE   #31
CFCC 30                 .BYTE   #30
CFCD 31                 .BYTE   #31
CFCE 31                 .BYTE   #31
CFCF 30                 .BYTE   #30
CFD0 31                 .BYTE   #31
CFD1 30                 .BYTE   #30
CFD2 31                 .BYTE   #31
```



```
                        .PAGE
                        .END

.       CFD3    ABUF    0045    ADDDAY  0010
ADJ00   CF4C    ADJ01   CF87    ADJ03   CF3D
ADJ04   CF77    ADJDAY  CF06    AM000   CF9D
AM001   CFA0    AMONTH  CF8B    AMPM    0004
ASCII   CB92    ASCII1  CB94    BRD     0008
C1224   0008    CLRSTA  CB88    COD     0008
CWSH    0037    CWSL    0036    DAT00   CD13
DAT00A  CD18    DAT01A  CD21    DAT02   CD24
DAT03   CD60    DAT04   CD31    DAT05   CD41
DAT06   CD4B    DAT07   CD58    DAT09   CD3C
DAT09A  CD3E    DAT10   CD55    DAT12   CD6F
DAT13   CD78    DAT14   CD85    DAT14A  CD94
DAT15   CDD6    DAT16   CD77    DAT17   CDD9
DATE    CCF9    DATLIN  0002    DAY00   C90B
DAY00A  C977    DAY00B  C974    DAY00C  C988
DAY01   C9A9    DAYS    C9C3    DOIN    C83E
DONE    C91D    DOOO0   C897    DOOOC   C899
DOOUT   C87C    DOSSLT  87F8    DOT00   CE41
DOT01   CE48    DOT02   CE4E    DOT03   CE4F
DOT04   CE75    DOT05   CE67    DOT05A  CE74
DOT06   CE90    DOT07   CE62    DOT08   CE8A
DOT10   CEAB    DOT11   CEC0    DOT12   CEC1
DOT13   CEEF    DOT14   CEDC    DOT15   CEE1
DOT15A  CEEE    DOTOFF  CE20    ENTRY1  C800
EXIT    C855    FINISH  CBA8    FINJ0   C9A6
FRAC    0670    HMYY    0678    HOLD    0001
ID      C8FF    ID0     C8E7    ID1     C8EA
ID2     C8ED    ID3     C8F0    IDISPA  C922
INCSTA  CB96    INTYP   0040    IOBASE  C000
IOREST  FF3F    IOSAVE  FF4A    KWSH    0039
KWSL    0038    LEA00   CFC4    LEA01   CFBE
LEAP    CFA6    LODLOC  6000    MMMM    06F8
MONLEN  CFC6    MONTHS  CDB2    MORDIS  C84A
MORIN   C860    MORIXX  C863    MOROUT  C8A8
MOROXX  C8AB    MOVLOC  C000    OD0     C8F3
OD1     C8F6    OD2     C8F9    OD3     C8FC
ODISPA  C900    OFF01A  CBEF    OFFHRS  0578
OFFMIN  05F8    OFF000  CBDC    OFF001  CBEA
OFF002  CBF9    OFF003  CC0F    OFF010  CC13
OFFOFF  CBC5    OFFS8N  0001    OFFTMP  05F8
PA      C080    PAC     C081    PB      C082
PBC     C083    RC00    CA78    RC01    CA8D
RC02    CAA2    RC03    CAB4    RC03A   CAD1
RC04    CAAF    RC05    CAEF    RC05A   CB38
RC05B   CB34    RC06    CAD8    RC06A   CAE8
RCW00   CB18    RCW01   CB21    RCW02   CB26
RCW03   CB2C    RCWDH   CB59    RCWDH0  CB69
RCWDH1  CB6F    RCWDN   CB46    READ    0002
READC   CB03    READCC  CA59    READCD  CB73
READSW  C9F9    READX   CA66    REJ00   C8A7
REJOUT  C89D    RSH00   CA06    RSH01   CA44
RSH02   CA1F    RSH04   CA1D    RSH05   CA03
```

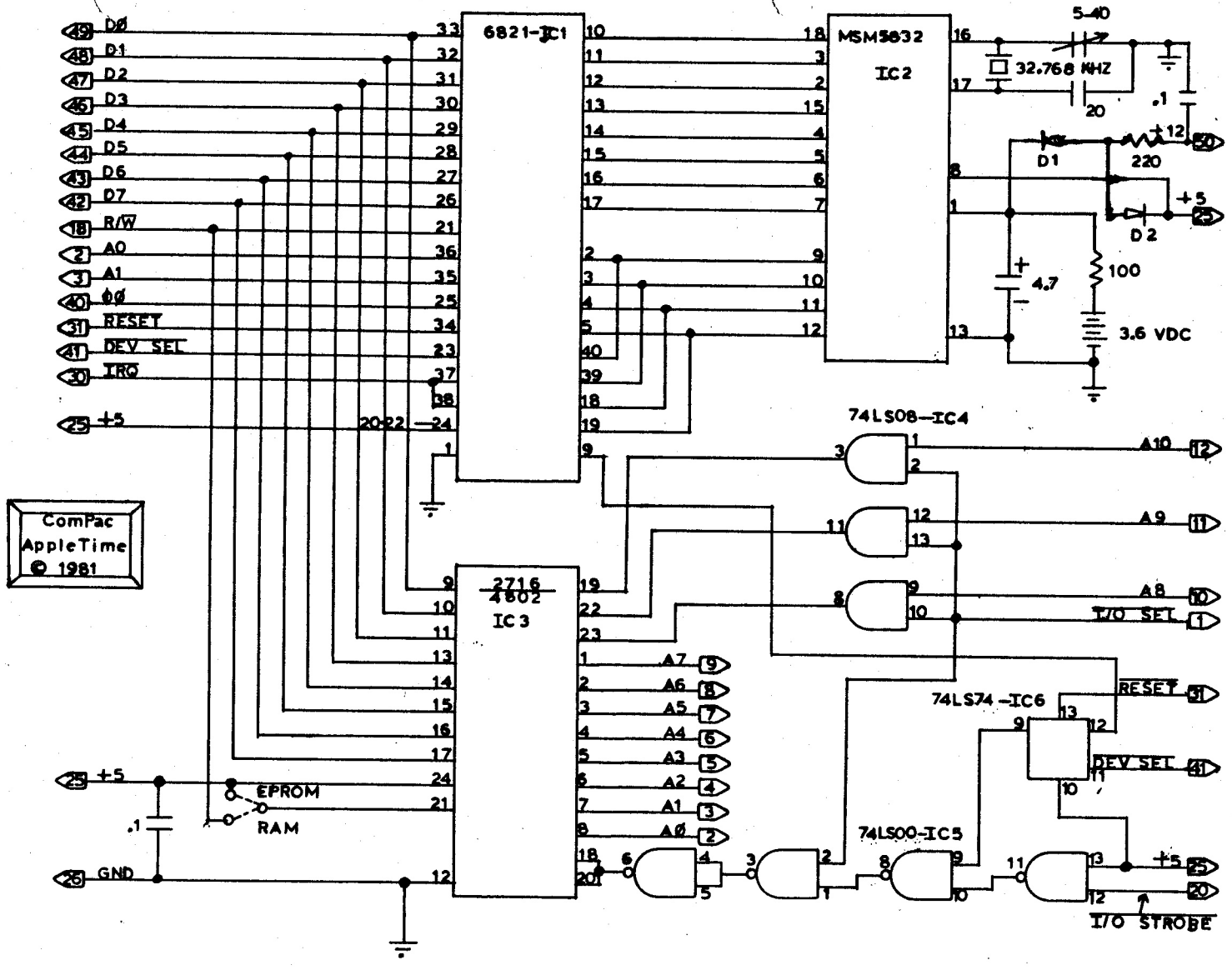
```
SEC30  0000          SET    CDE9          SET00  CE02
SET10  CE25          SET30  C930          SETS00 C9E7
SETSTA CBA9          SETSW  C9D8          SFFAL  CD8F
SFLOW  0000          SFTRU  CBB9          SSDD   0778
STATE  07F8          STATUS 0478          SUBDAY 0020
SWATCH 0040          TATI   0001          TDA1   0004
TD     0005          TDAY   0007          TEMP   0778
TI     CC27          TIM00A CC64          TIM01  CC97
TIM02  CC6B          TIM03  CC78          TIM04  CC88
TIM04A CC8F          TIM05  CC83          TIM06  CC85
TIM07  CC92          TIM07A CC94          TIM08  CCAF
TIM09  CCC9          TIM10  CCCC          TIM11  CCD9
TIM12  CCE6          TIM13  CCE3          TIM14  CCF4
TIM15  CCED          TIM16  CCF3          TIME   CC16
TIS00  CC5B          TITI   0002          TOFF   0031
TRSH   0010          TSEC30 0032          TSET   0038
TSETSW 0033          TUDA   0003          TUDAY  0006
TUTI   0000          TYPE   04F8          WAIT   FCA8
WRITE  0004          XBUF   0046          XRET   FF58

       CFC7
ABUF   0000 C004 C900 C98C C994 C9A2 C9CF C9E0 C9ED C9F9 CC50 CCAC CCC6 CDAB
  CE02
ADDDAY 0000 C964 CA92 CE62 CEE9 CF0C
ADJ00  CF0E CF40 CF4C
ADJ01  CF51 CF5C CF00 CF07
ADJ03  CF1F CF2F CF3D
ADJ04  CF67 CF77
ADJDAY CD0D CF06
AM000  CF91 CF9D
AM001  CF9B CFA0
AMONTH CF19 CF77 CF80
AMPM   0000 CA92 CC7B CE7E CE0A CEA3 CEFB CF00
ASCII  C9A3 CB92 CCF6
ASCII1 CB94 CC05 CC94 CD3E
BRD    0000 C0FF
C1224  0000 CA92 CB5C CC67 CE83 CE93 CECF
CLRSTA CA53 CA94 CB00 CBC1 CBCC
COD    0000 C0FF
CWSH   0000 C033
CWSL   0000 0000 C03A C099 C0A3 C91F C900 C9E5 C9EC C9F4
DAT00  CCFC CD01 CD13
DAT00A CD0B CD10
DAT01A CD21 CD2F CD6D
DAT02  CD18 CD24
DAT03  CD28 CD68
DAT04  CD31
DAT05  CD3A CD41
DAT06  CD43 CD4B
DAT07  CD48 CD52 CD90
DAT09  CD26 CD3C
DAT09A CD3E CD79
DAT10  CD4D CD55
DAT12  CD6F
DAT13  CD75 CD7B
DAT14  CD7D CD85
DAT14A CD98 CD94
DP     CD87 CDD6
D      CD71 CD77
```

```
LODLOC 0000 C000 C0E7
MMMM   0000 CAE2 CAE9 CB0E CB13 CB46 CB4C CCDD CD45 CD8B CD94 CD97 CD9C CE30
  CE37 CE48 CEAB CEBA CF27 CF2A CF34 CF49 CF5F CF64 CF6B CF8C CF93 CF9D CFA6
MONLEN CFA2 CFC7
MONTHS CDA6 CD82
MORDIS C04A C07A
MORIN  C046 C060
MORIXX C063 C0A9
MOROUT C097 C0A0
MOROXX C070 C0AB
MOVLOC 0000 0000 C0B9
OD0    C0F3 C90F
OD1    C0F6 C917
OD2    C0F9 C91B
OD3    C0FC C913
ODISPA C0B3 C900
OFF01A C0E4 C0EF
OFFHRS 0000 CC10 CE52 CEC4
OFFMIN 0000 0000 CC09 CE34 CEAF CEB2
OFF000 CBC8 CBDC
OFF001 CBDE CBEA
OFF002 CBEB CBF9
OFF003 CC06 CC0F
OFF010 CBD3 CBDA CC13
OFFOFF C0F9 CBC5
OFFSGN 0000 CBCA CBD5 CE2C
OFFTMP 0000 CBF3 CBFD
PA     0000 C023 C04C C04F C057 C05A C0AD C0B0 C9E0 C9E7 CA2B CB0C CDF5 CE04

PAC    0000 C01B C02D CDF0 CDFA
PB     0000 C028 C93A C942 C957 C9DD CA13 CA4B CA68 CB39 CB89 CDFF CE13 CE18

PBC    0000 C01E C030
RC00   CA70 CAB2
RC01   CA7D CA8D
RC02   CA84 CA8A CA90 CAA2
RC03   CAA6 CAB4
RC03A  CAC6 CAD1
RC04   CAAF CAD5 CAEC
RC05   CABF CAEF
RC05A  CAF4 CB30
RC05B  CB2D CB34
RC06   CABD CAD0
RC06A  CADD CAE0
RCW00  CB05 CB16 CB1B
RCW01  CB1C CB21
RCW02  CB21 CB26
RCW03  CB27 CB2C
RCWDH  CB23 CB29 CB59
RCWDH0 CB5E CB69
RCWDH1 CB66 CB6F
RCWDM  CB1B CB1E CB46
READ   0000 C9DB CA11 CB87
READC  C94F CA76 CB03
READCC C9F0 CA0E CA59 CC27
READCD CB73 CD03
READSW C0E7 C9F9
READX  CA66 CB00
REJ00  C09F C0A7
REJOUT C093 C09D C91D
```

```
AT17   CD02 CDD9
AT18   CD97
AT19   CDDD CDE1 CDE6
ATE    C0ED CCF9
ATLIN  0000 C0CA C0E6 CE6B CEE5
AY     C0F0 C948
AY00   C94B C95F C97C C988
AY00A  C966 C977
AY00B  C978 C974
AY00C  C984 C988
AY01   C999 C9A9
AYS    C9B7 C9C3
DOIN   C030 C03E
DONE   C904 C900 C91D C945 C9F6 CC0C CE22
DO000  C091 C097
DO00C  C095 C099
DOOUT  C03C C07C
DOSSLT 0000 C000 C035
DOT00  CE3A CE41
DOT01  CE3F CE40
DOT02  CE43 CE4E
DOT03  CE4C CE4F
DOT04  CE5A CE75
DOT05  CE67 CE77 CE05 CE0C CE95
DOT05A CE6D CE74
DOT06  CE79 CE90
DOT07  CE62 CE8E
DOT08  CE8A CEA0
DOT10  CE2E CEAB
DOT11  CEB5 CEC0
DOT12  CEBE CEC1
DOT13  CED1 CEEF
DOT14  CEDC CF02
DOT15  CECA CEE1 CF04
DOT15A CEE7 CEEE
DOTOFF CC34 CE20
ENTRY1 C000
EXIT   C055 C00A C09B C0B7
FINISH C9A6 CBA0 CC0F CD21
FINJ0  C990 C99E C9A6
FRAC   0000 CA1A CA38 CA3F CC9E
HHYY   0000 CAC8 CAD2 CB50 CB55 CB60 CB69 CB6F CCD3 CD55 CDD6 CE4F CE55 CE5F
  CE97 CE9D CEC1 CEC7 CED4 CED9 CEF0 CEF5 CF37 CF3A CF6F CF74 CFB5 CFBE
HOLD   0000 CA66 CB07 CDFD CE11 CE16
ID     C0FF
ID0    C0E7 C927
IF     C0EA C930
I      C0ED C934
IL     C0F0 C936
IDISPA C052 C922
INCSTA C9C0 CAAF CB96 CBF6 CC13 CDAF CE25
INTVP  0000 C00F
IOBASE 0000 0000 0000 0000 0000
IOREST 0000 C05D
IOSAVE 0000 C000 C064
KWSH   0000 CA59 CAAD CAB5 CB43 CB73
KWSL   0000 0000 C048 CA0B CA30 CA47 CAFB CBA6
LEA00  CFA8 CFB2 CFC4
LEA01  CFBA CFBE
LEAP   CF3D CF7D CFA6
```

```
RSW00  CA01 CA06
RSW01  CA32 CA44
RSW02  CA1F CA35
RSW04  CA1D CA42
RSW05  C9FC CA03
SEC30  0000 C930
SET    C0F3 CDE9
SET00  CDEC CE02
SET10  CE20 CE25
SET30  C0FC C930
SETS00 C9E7 C9EE
SETSTA CA00 CA9B CBA9 CBBB CBD7 CBE0 CE64 CE71 CEDE CEEB
SETSW  C0F6 C9D8
SFFAL  CA63 CB3C CB8F CC53 CCA1 CCC1
SFLOW  0000 CB41 CBB9 CB8F
SFTRU  CA4E CB7D CBB9 CC2A CC5B CCC9 CD10
SSDD   0000 0000 CAFF CB02 CB0A CB31 CCE3 CD4F CD7F CF11 CF16 CF1C CF24 CF44
  CF54 CF59 CF7A CF84 CFAD
STATE  0000 C010 C940 C98B C99B C9B3 C9F9 CA73 CA7F CA8D CAA2 CAB8 CB36 CB96
  CB9C CBC5 CC01 CC16 CC37 CC45 CC6B CC88 CC97 CCAF CCCC CCE7 CCF9 CD1A CD2A CD31
  CD59 CD60 CD9F CDDA CDE9 CE0A CE1B
STATUS 0000 C03E C043 C07E C081 C961 C977 C9FE CA7A CAC1 CAD8 CAEF CB59 CBA9
  CBAC CBB2 CBB5 CC22 CC4E CC64 CC78 CC9C CCB5 CCFE CE29 CE68 CE7B CE80 CE87 CE90
  CEA8 CEA5 CECC CEE2 CEF8 CEFD CF09 CF4C
SUBDAY 0000 C97A CA92 CE6F CEDC CF4F
SWATCH 0000 C07C CA06 CA51 CAC4 CADB CAF2
TATI   0000
TDA1   0000
TDA2   0000
TDAY   0000
TEMP   0000 C952 C968 C974 C97E C988 C9A0 C9AB C9B0
TIM00  CC19 CC20 CC25 CC32 CC37
TIM00A CC43 CC64
TIM01  CC3C CC97
TIM02  CC6B
TIM03  CC78 CC78
TIM04  CC72 CC88
TIM04A CC4C CC69 CC8F
TIM05  CC7D CC83
TIM06  CC76 CC81 CC85
TIM07  CC8D CC92
TIM07A CC94
TIM08  CC9A CC9F CCA0 CCAF
TIM09  CCB3 CCC9
TIM10  CCB8 CCBF CCCC CD1F
TIM11  CCD1 CCD9
TIM12  CC61 CCD6 CCE0 CCE6
TIM13  CCDB CCE3
TIM14  CCF4
TIM15  CCED CD5C CD60 CD64 CDE6
TIM16  CCEB CCF3 CD66 CDE3
TIME   C0EA CA03 CA56 CC16
TIS00  CC4A CC51 CC5B
TITI   0000
TOFF   0000 C919
TRSH   0000 C925 CC1E
TSEC30 0000 C911
TSET   0000 C90D
TSETSW 0000 C915
TUDA   0000
```

```
TUDAY   0000 C997
TUT1    0000
TYPE    0000 C80C C90A C922 C929 C95A C992 CC1B CC2D CC3E CCA4 CCBA CD06 CD13
 CD24
WAIT    0000 C93F CA6D
WRITE   0000 CE11
XBUF    0000 CA5C CA61 CA78 CB40 CB76 CB7B
XRET    0000 C804 C860 C868 CA1F CA22 CA25 CA28

NO ERRORS DETECTED
```



The circuitry consisting of IC3, IC4, IC5 and IC6 is proprietary to uComPac, Inc. This circuit may not be used or copied in any form without written permission from uComPac, Inc.

SOME HARDWARE NOTES ON THE APPLETIME CLOCK CARD.

The APPLETIME clock card EPROM can be replaced with some of the newer 2K x 8 RAMS now becoming available. In particular, the card has been tested and used with the MOSTEK 4802-P1 RAM. The use of RAM in place of EPROM allows software to be tailored to a specific task, rather than depending on or trying to "work around" the firmware in the 2716 EPROM.

To use RAM, change the jumper located below the 2716 EPROM from its' present position to the lower hole. This connects the R/W line to pin 21 of the EPROM/RAM socket.

CAUTION: When changing back to EPROM the jumper should be changed back to its' upper position.

When using RAM, the first page (256 bytes) of the 2716 EPROM should be copied into lower memory and used with the RAM code. The first page is primarily control for the clock card. Be aware that when shifting the RAM from the first page ($Cn00) into the upper 2K of the APPLE addressing space ($C800-$CFFF), the first page will now begin at $C800. To turn off the RAM when it is in its upper addressing area, exit must take place through the first page of code. Failure to "turn-off" the RAM will result in bus contention and the possibility of trying to access more than one peripheral at a time. Reference should be made to the EPROM code listing for specific details on how the switching is accomplished.

The battery on the APPLETIME clock card is capable of 3500 to 4000 hours of life. The battery has been charged at the factory and will be recharged each time the computer is turned on. The charge rate is approximately 7 MA., so to completely charge the battery from a discharged condition will take approximately 16 hours. However, because of the low drain from the battery in "computer-off" times, a normal session of "computer-on" time, (2 to 4 hours) should keep the battery in good condition.

A fully charged battery will measure about 3.98 to 4.1 volts. The nominal battery voltage is 3.6 volts, and time keeping will be retained down to about 2.2 to 2.3 volts.

MicroComPac
Software
Registration

Name _____

Address _____

_____

_____

City State Zip

Name of Product _____

Mail to: MicroComPac
P.O. Box 163
Golden, CO 80401