Does Anybody Really Know What Time It Is?

Your computer needs to know the time almost as often as you do. In business applications, in particular, you can improve your record-keeping if the computer, knowing the time and date, can print reports comprising chronological information.

Your TRS-80's crude time and date-keeping function (with expansion interface required on the Model I) has several drawbacks. Besides the obvious annoyance of the date/time prompt every time you turn the computer on, many computer functions require that you disable interrupts to keep the 25 millisecond (ms) "heartbeat" pulse from intervening and properly updating the time and date values.

A relatively simple external circuit and some software can eliminate such problems. This month's project, a real-time clock board (using National Semiconductor's MM58174 real-time clock chip), has battery back-up so the clock always keeps the correct time and date, whether or not primary power is supplied to the board. You only have to set time and date once, and they'll be available whenever you turn on the system. You can build the board (see the Photo) for the Model I, III, or 4, with any amount of memory. Model I users don't need an expansion interface, either.

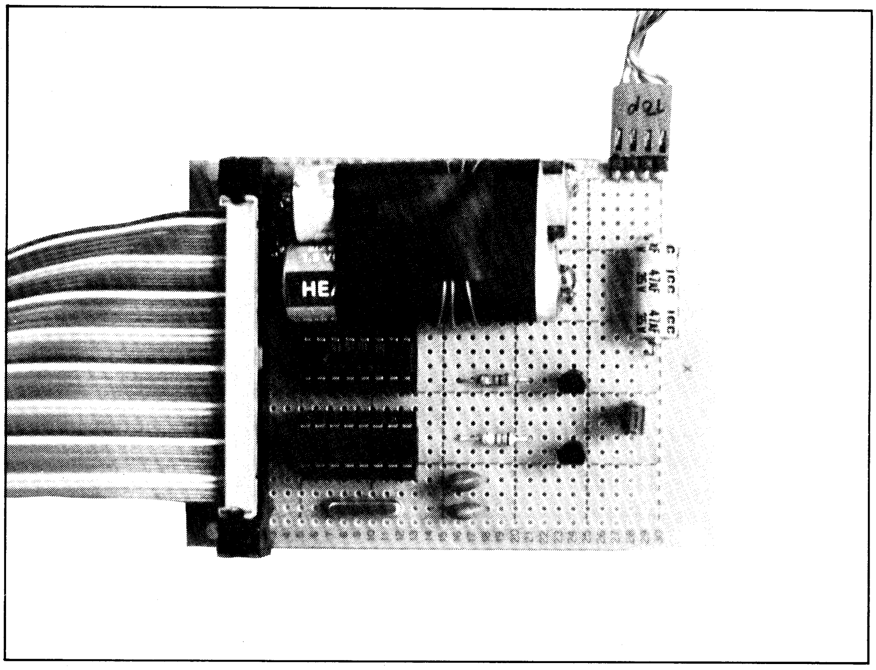

Photo. Real-time clock board.

A Real-Time Clock Chip

The MM58174 I use in this project is a relatively inexpensive device that allows precise time-keeping with minimal power requirements (see Fig. 1 for a block diagram). The CMOS technology behind the 58174 results in its low power requirements—typically 1 milliampere (mA) when active—and permits it to have a low-power standby mode, requiring only 10 microamps (μA). This small current draw lets you back up the date and time by a single 3-volt (V) lithium battery (with 1,200 mA-hour) for over 13 years. Even a 50 mAh lithium battery (watch-battery size) can back up the 58174 for nearly seven months.

The 58174 exists in a 16-pin DIP (dual in-line package), making it relatively small and economical. It has internal registers for all time values from hours to tenths of seconds and date registers including day of the week, month, and day of the month. While the date registers do not support the year, there is a leap-year register that receives a value telling the chip when February gets 29 days.

The 58174 was designed for simple interfacing to microprocessor buses, so you need very little circuitry to build this month's project. Because of the chip's timing specifications, however, Model 4 users must run in Model III mode (slower processor speed) unless they add special circuitry to support the faster timing.

Set only hours and minutes when you set the 58174; the seconds are automatically cleared to zero; seconds and tenths of seconds can, however, be read from the device. The 58174 can also generate interrupts on an interval of 0.5 seconds, 5.0 seconds, or 60 seconds (+/− 16.6 ms). Although I didn't use the interrupt feature in this project, the board schematic shows the interrupt connection if you want to use it (see Fig. 2).

National Semiconductor has noted the deficiencies of the 58174 and has corrected them in their new MM58274. The 58274 is a pin-compatible (identical functional pinout)

The Key Box LOAD 80

Models I and III
32K RAM Disk Basic
Assembly Language
EDTASM

---

replacement for the 58174. It provides a year register, lets you set seconds, and provides much greater flexibility in interrupt interval selection. Since the 58274 is pin-compatible with the 58174, it can directly replace the 58174 in this month's project when it becomes readily available. The 58274 does have a somewhat different register arrangement, however, so software for the two devices is not compatible.

Constructing the Board

The real-time clock board schematic shows an optional jumper coming off the battery that removes it from the circuit. If the battery is in its holder when you don't want it backing up the real-time clock chip, you can remove the jumper to keep the battery from draining. If you intend always to use the battery for back-up while in its holder, don't put the jumper on the board.

The capacitor on pin 15 of the 58174 is a critical timing capacitor that permits you to set the 58174 precisely. The schematic shows a 10 pF capacitor in this position, but National recommends a 6–36 pF variable capacitor for precise frequency tuning. If you use a variable capacitor you can set the crystal frequency precisely to the desired 32.768 kHz in a number of ways; refer to "The MM58174 Applications Note," by Steve Munich of National Semiconductor.

You use the 74LS138 shown in the schematic for address decoding. Although you can use other port addressing, the decoding for the board as shown uses port addresses 20 hex–2F hex (32–47 decimal). The software shown later assumes you used this address decoding.

The back-up battery must be approximately 3V, but that's the only requirement. I used two AA-size batteries in series to create the 3V. For longer life, you can also use a 3V lithium battery.

Aside from the two integrated circuits (ICs) mentioned (the 58174 and the 74LS138) and the batteries described above, you need only seven other components for the board: a 32.768 kHz crystal, two resistors, two capacitors, and two PNP transistors (see Table 1 for parts list and ordering information). Finally, you need a +5V power supply (at 15 mA) to operate the board.

Theory of Operation

You use the two PNP transistors as switches. When the +5V power supply is on, Q1 is forward-biased, allowing the 58174 to draw its current from the +5V source. When you turn off the power supply and the voltage from the 100 μF capacitor falls below the battery voltage, Q1 becomes reverse-biased, switching Q2 on. The current for the 58174 then comes from the back-up battery. At this point VDD of the 58174 will be approximately 2.7V and the real-time clock will be in standby mode (VDD is less than 4V). When in standby mode, Q2 also keeps the 58174's chip select pin (pin 1) high, as required by its power-down timing.

Operating the Board

To access the board, either to read it or set it, be sure that the main power supply (+5V) is on. Otherwise, the board is in standby mode and the processor can't access it. Also, without the +5V, the 74LS138 doesn't function, so the 58174 could never be selected.

Table 2 shows the 58174 registers as addressed on the real-time clock board. I have included the test register (port 20 hex) primarily for a produc-

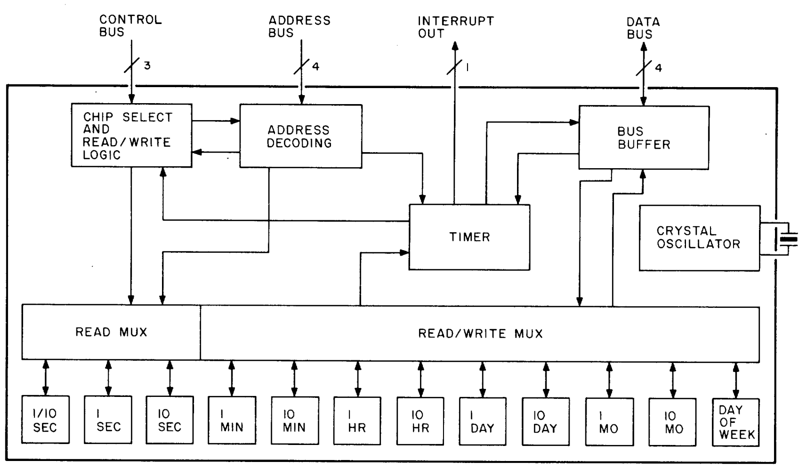

Figure 1. The 58174 real-time clock block diagram.

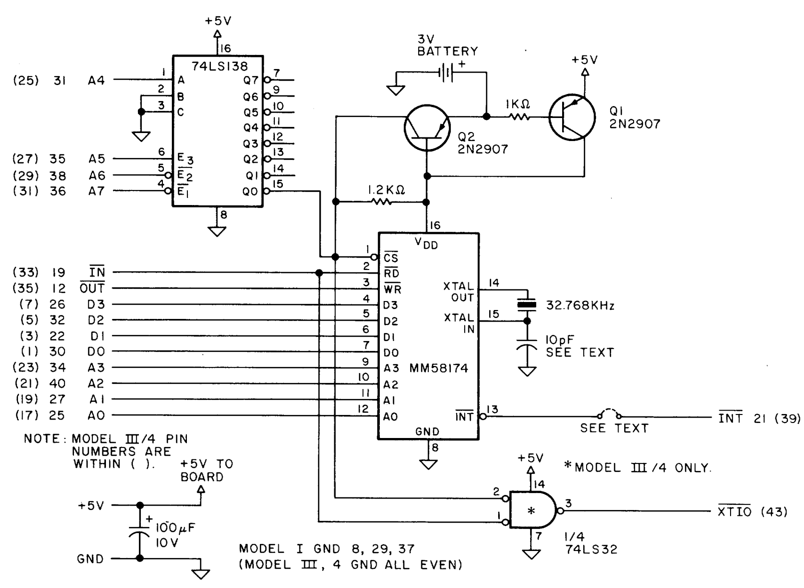

Figure 2. Real-time clock board schematic.

Port Number	Register Function	Access Mode
20 hex	Test register	Write only
21 hex	Tenths of seconds	Read only
22 hex	Units of seconds	Read only
23 hex	Tens of seconds	Read only
24 hex	Units of minutes	Read/write
25 hex	Tens of minutes	Read/write
26 hex	Units of hours	Read/write
27 hex	Tens of hours	Read/write
28 hex	Units of days	Read/write
29 hex	Tens of days	Read/write
2A hex	Day of week	Read/write
2B hex	Units of months	Read/write
2C hex	Tens of months	Read/write
2D hex	Leap year	Write only
2E hex	Stop/start	Write only
2F hex	Interrupt/status	Read/write

Table 2. Registers of the 58174 real-time clock chip.

tion test of the device. For proper operation of the real-time clock, bit 3 of the register should be set to zero during chip initialization, placing the chip in non-test mode.

The start/stop register, port 2E hex, starts and stops the internal timing operations of the real-time clock. When the processor writes a zero to bit zero of this register, the clock stops. A 1 written to the same bit starts the clock. This is a necessary register, since you should stop the time when setting the real-time clock with time and date information. As soon as the processor writes all values to the chip, you can start the clock again.

The interrupt and status register (port 2F hex) sets up interrupt-generation information. By writing various values to this register, the processor can enable and disable the interrupt function, as well as select the interrupt time interval.

Most of the remaining registers are reasonably straightforward. Since there is only a 4-bit data bus on the 58174 to communicate with the TRS-80's Z80, all time and date values are communicated (written and read) as single decimal digits. Thus, the high-order digit (in units of tens) and low-order digit (units of ones) are stored in separate registers for the seconds, minutes, hours, days, and months. The tenths-of-seconds and day-of-week values have their own registers.

One register that can be confusing is the years register. As mentioned earlier, the 58174 doesn't hold a year value. The years register is merely a 4-bit shift register that keeps track of leap year. Exactly 1 bit should be set in this register at any time. The position of the bit indicates the relative position of the current year to leap year. If bit 3 is set, the current year is leap year (February has 29 days); if bit 2 is set, the current year is one year before leap year, and so on. This lets the 58174 know when to base its calculations on 28 days in February and when to base its calculations on 29 days.

Also note that the 58174 does not provide error-checking features. If you give the date as January 32, the real-time clock believes you and stores that date in its internal registers.

Because the 58174 keeps track of time constantly—to a tenth of a second—you have to provide the time read functions in Assembly-language software. From a software standpoint, the entire time read must take place within a tenth of a second. You also must ensure that the time didn't change between the beginning and the end of the register reads.

The 58174 does have an internal data-changed flip-flop to indicate a tenth-of-a-second change. Whenever the tenths-of-seconds counter changes, this flip-flop sets all 58174 data lines high; it is cleared by the next low-to-high transition of the read strobe. There are several reasons, however, why a program may never see the "F" on the data lines. First, if other peripheral reads (to other devices) take place during the time a program reads the registers (because of an interrupt, for example), the flip-flop would be

Quantity	Description	Distributor	Part Number	Price (each)
1	Real-time clock IC	JE	MM58174AN	$7.95
1	3-to-8 decoder IC	JDR	74LS138	.55
1	Quad 2-input OR gate (LS TTL) IC*	JDR	74LS32	.29
2	2N2907 low-power switching transistor (PNP)	RS	276-2023	.79
1	1k resistor (¼ watt)	RS	271-1321	.08
1	1.2k resistor (¼ watt)	DK	1.2KQ	.05
1	32.768 KHz crystal	JDR		1.95
2	1.5V battery			—
1	10 pF/500V disk capacitor	DK	P4000	.08
1	100 μF/10V electrolytic capacitor (PC mount)	JDR		.18
1	.1" matrix grid prototype board	RS	276-158	1.95
1	40 pos. cable header (w/w)†	DK	R241-ND	5.58
1	40 pos. ribbon cable edge connector†	DK	R503-ND	3.80
1	40 pos. ribbon cable socket connector†	DK	R306-ND	3.73
1 foot	40 cond. ribbon cable†	DK	R007-ND	—
1	50 pos. cable header (w/w)*	DK	R247-ND	6.93
1	50 pos. ribbon cable edge connector*	RS	276-1566	4.95
1	50 pos. ribbon cable socket connector*	DK	R307-ND	4.65
1 foot	50 cond. ribbon cable*	DK	R008-ND	—

*Model III/4
†Model I

Note: You can substitute Radio Shack's 1.2k ½ watt resistor for the ¼ watt resistor specified, and a 3V lithium battery for the 1.5V battery. Use a variable capacitor for greater accuracy (see text).

Addresses:

Jameco Electronics (JE), 1355 Shoreway Road, Belmont, CA 94002, 415-592-8097.

JDR Microdevices, 1224 S. Bascom Ave., San Jose, CA 95128, 800-538-5000 or 408-995-5430 outside California; 800-662-6279 within California.

Radio Shack (RS), National Parts Division, 900 East Northside Drive, Forth Worth, TX 76102, 817-870-5662.

Digi-Key Corp. (DK), Highway 32 S., P.O. Box 677, Thief River Falls, MN 56701, 800-346-5144 or 218-681-6674.

Table 1. Parts list and ordering information.

reset before the program could read it. Also, the data-changed flip-flop can be set while the read strobe is low; again, the program would never see the "F."

Because of the unreliability of the high-bus method of detecting a time change, National recommends that a read of the 58174 begin and end with a read of the tenths-of-seconds register. The two values are then compared; if they differ, the read was unsuccessful and the values are read again. When the two values agree, a successful read is assured.

My short Assembly-language program (see Listing 1) contains two real-time clock routines, one to read the clock chip and one to set it. The program begins at 0FFC0 hex, which is at the top of memory for a 48K system. Other systems should set the starting address where appropriate. Don't forget to reserve the top of memory by setting HIMEM when using Assembly-language routines you can call from Basic.

I designed the two routines in List-

Program Listing 1. Two real-time clock routines.

```
      ;***********************************************
      ;           58174 Real-Time Clock              *
      ;       Read Time and Set Time Routines        *
      ;                                              *
      ;    The two routines included here allow the  *
      ; National Semiconductor 58174 real-time clock *
      ; to be set and reset via USR calls from Basic. *
      ; A specific area of memory is set aside as a  *
      ; data area, where values read from the 58174  *
      ; are stored, and where values to be written to *
      ; the 58174 are found.                         *
      ;                                              *
      ;    No parameters need to be passed to either *
      ; routine, and no result information is        *
      ; returned. Also, no error checking takes place *
      ; here. All information to be written to the   *
      ; real-time clock chip is assumed to be        *
      ; correct.                                     *
      ;                                              *
      ;          Written by Roger C. Alford          *
      ;***********************************************
      ;
      ;*** Define Equates:
0020  TEST    EQU    20H    ;58174 TEST REGISTER
0021  TNTHSC  EQU    21H    ;58174 TENTHS OF SECONDS
0022  UNITSC  EQU    22H    ;58174 UNITS OF SECONDS
0023  TENSSC  EQU    23H    ;58174 TENS OF SECONDS
0024  UNITMN  EQU    24H    ;58174 UNITS OF MINUTES
0025  TENSMN  EQU    25H    ;58174 TENS OF MINUTES
0026  UNITHR  EQU    26H    ;58174 UNITS OF HOURS
0027  TENSHR  EQU    27H    ;58174 TENS OF HOURS
0028  UNITDY  EQU    28H    ;58174 UNITS OF DAYS
0029  TENSDY  EQU    29H    ;58174 TENS OF DAYS
002A  DYOFWK  EQU    2AH    ;58174 DAY OF WEEK
002B  UNITMO  EQU    2BH    ;58174 UNITS OF MONTHS
002C  TENSMO  EQU    2CH    ;58174 TENS OF MONTHS
002D  LPYEAR  EQU    2DH    ;58174 LEAP YEAR CODE REGISTER
002E  STPSTR  EQU    2EH    ;58174 STOP/START REGISTER
002F  INTSTS  EQU    2FH    ;58174 INTERRUPT/STATUS REGISTER
      ;
FFC0          ORG    0FFC0H
      ;
      ;
```

Listing 1 continued

Listing 1 continued

```
                ;*****************************************************
                ; RDTIME is the routine to read the time and date      *
                ; values from the 58174 and place them into the proper *
                ; locations in the time table (TIMTBL) below. All      *
                ; values are read from tenths of a second to tens of   *
                ; months. The calling Basic program can then read these *
                ; values using its PEEK function, then combine the     *
                ; values as desired to create time and date strings.   *
                ;*****************************************************
                ;
FFC0 21ECFF     RDTIME  LD      HL,TIMTBL       ;POINT TO TIME TABLE
FFC3 0E20               LD      C,TNTHSC-1      ;PNT TO 1ST READ REG - 1
FFC5 060C               LD      B,TENSMO-TNTHSC+1
                                                ;PUT # OF READ REG'S IN B
FFC7 0C         RDLOOP  INC     C               ;POINT TO NEXT READ REG.
FFC8 EDA2               INI                     ;PUT REG VALUE IN TABLE
FFCA 20FB               JR      NZ,RDLOOP       ;LOOP UNTIL ALL READ
FFCC DB21               IN      A,(TNTHSC)      ;GET TENTHS OF SEC AGAIN
FFCE 47                 LD      B,A             ;SAVE TEMP IN B REGISTER
FFCF 3AECFF             LD      A,(TIMTBL)      ;GET ORIG TENTHS OF SEC
FFD2 B8                 CP      B               ;ARE THEY STILL THE SAME?
FFD3 20EB               JR      NZ,RDTIME       ;IF NOT, READ VALUES OVER
FFD5 C9                 RET                     ;ELSE, DONE - RETURN
                ;
                ;
                ;*****************************************************
                ; SETIME is the routine to set the time and date in the *
                ; 58174 real-time clock chip. The time table (TIMTBL)  *
                ; below is where the values to be written are expected *
                ; to be. Since nothing smaller than minutes can be     *
                ; written to the 58174, the first three table locations *
                ; are ignored by this routine. The calling Basic       *
                ; program must first POKE the proper time and date     *
                ; values into the appropriate TIMTBL locations, then   *
                ; call this routine. The values are then written to the *
                ; 58174, then control is returned to the Basic program. *
                ;*****************************************************
                ;
FFD6 AF         SETIME  XOR     A               ;CLEAR THE ACCUMULATOR
FFD7 D320               OUT     (TEST),A        ;MAKE SURE NOT IN TEST MD
FFD9 D32E               OUT     (STPSTR),A      ;STOP 58174 OPERATION
FFDB 21EFFF             LD      HL,TIMTBL+3     ;PT TO 1ST USED TBL LOC.
```

Listing 1 continued

ing 1 so that you can call them from Basic; Listing 2 is a Basic program that uses these routines. They reserve a 13-byte table for a buffer area for the time and date values. The locations correspond with the registers of the 58174, with the first location representing the tenths-of-seconds register. The RDTIME routine of Listing 1 reads the time and date values from the 58174 and stores them in the table (TIMTBL). The calling Basic program can then access these values (using PEEK).

Likewise, the Basic program can store set-up values in the appropriate locations (using POKE), and call the SETIME routine to set the time and date in the real-time clock. Since the 58174 doesn't check for incorrect time and date input, the Basic program in Listing 2 takes on this responsibility. If you want to set the time, the program prompts you for the time and date information, including the year. When you enter the information, the program checks its validity. If it detects an error—February 30, for ex-

✓ See List of Advertisers on Page 195

ample—the set-up procedure starts over; otherwise, the program returns you to the main menu. The program uses the year you enter to determine the proper value for the years register (leap-year calculation).

If you select the read-time option when running the Basic program in Listing 2, the program calls the RDTIME routine, then converts the information into time and date strings and displays them on the screen, spelling out the actual month and day of the week.

Those of you whose systems have a 25 ms interrupt (Model I's with an expansion interface, for example) can use it to keep track of time and date after you've set them. Since the real-time clock board circumvents the date and time prompts each time you turn on the computer, my other Assembly-language program (see Listing 3) takes advantage of the 25 ms interrupt.

The program starts at 0FF60 hex for a 48K system, so make the appropriate adjustments for other configurations. I'm not sure about precise memory addresses for Model III and 4 for this program (I assume they're the same), so I can't guarantee this program will operate as is on those systems. To set up the interrupt interception, you must first call the main code section (RTIMER) from Basic, although the type of Basic determines how you'll implement this precisely. With Disk Basic, you can run the RTIMER code using two instructions:

DEFUSR = &HFF60
N = USR(0)

Once you make the interrupt interception, the program calls the TIMSRV routine whenever an interrupt occurs. If the 25 ms timer causes the interrupt, the interrupt servicing is forwarded to the normal interrupt service routine. Otherwise, the program reads the time and date from the real-time clock chip and stores the information in the reserved memory locations. The routine then returns from the interrupt. Since the 58174 does not support the year function, you have to POKE the year into location 4044 hex (16452 decimal) from Basic. ■

Write to Roger C. Alford at Washtenaw Digital Systems, P.O. Box 2014, Ann Arbor, MI 48106. Please include a self-addressed, stamped envelope for a reply.

Listing 1 continued

```
FFDE 0E23                 LD      C,UNITMN-1        ;PT TO 1ST WRITE REG - 1
FFE0 060A                 LD      B,LPYEAR-UNITMN+1
                                                    ;PUT # OF WRITE REG IN B
FFE2 0C         STLOOP    INC     C                 ;PT TO NEXT WRITE REG.
FFE3 EDA3                 OUTI                      ;TRANSFER VALUE TO REG
FFE5 20FB                 JR      NZ,STLOOP         ;LOOP UNTIL DONE
FFE7 3E01                 LD      A,01H             ;GET 58174 START VALUE
FFE9 D32E                 OUT     (STPSTR),A        ;START 58174 OPERATION
FFEB C9                   RET                       ;DONE - RETURN
                ;
                ;
                ;Define Tables:
000D            TIMTBL    DEFS    13        ;TABLE OF TIME READ/WRITE VALUES
                ;
FFC0                      END     RDTIME
00000 Total errors
```

End

Program Listing 2. Real-time and set-time Basic code (for use with Program Listing 1).

```
10 DIM A$(7),B$(12),A(12)
20 FOR I=1 TO 7
30 READ A$(I)
40 NEXT I
50 FOR I=1 TO 12
60 READ B$(I)
70 NEXT I
80 DEFUSR0=&HFFC0:'   READ TIME ROUTINE
90 DEFUSR1=&HFFD6:'   SET TIME ROUTINE
100 CLS
110 INPUT"READ TIME (R) OR SET TIME (S)";A$
120 IF A$="R" THEN GOSUB 1000 ELSE IF A$="S" THEN GOSUB 2000
130 GOTO 110
1000 '------------------------ READ TIME ROUTINE -------------
1005 N=USR0(0):'    READ CURRENT TIME
1010 FOR I=1 TO 12
1020 A(I)=PEEK(&HFFEB+I) AND 15:'   READ IN TIME VALUES
1030 NEXT I
1040 PRINT A$(A(10));", ";
1050 PRINT B$(A(12)*10+A(11));A(9)*10+A(8);"    ";
1060 A$=STR$(A(7)*10+A(6))+":"
1062 B$=STR$(A(5)*10+A(4)):A$=A$+RIGHT$("0"+RIGHT$(B$,LEN(B$)-1)
,2)+":"
1064 B$=STR$(A(3)*10+A(2)):A$=A$+RIGHT$("0"+RIGHT$(B$,LEN(B$)-1)
,2)+"."
1066 A$=A$+RIGHT$(STR$(A(1)),1)
1070 PRINT A$
1080 PRINT
1090 RETURN
2000 '---------------------- SET TIME ROUTINE ----------------
2005 CLS:GOSUB 1000
2010 INPUT "ENTER MONTH (1-12) ";MO%
2020 INPUT "ENTER DAY OF MONTH ";DT%
2030 INPUT "ENTER DAY OF WEEK  ";DY%
2040 INPUT "ENTER YEAR       19";YR%
2050 INPUT "ENTER HOUR         ";HR%
2060 INPUT "ENTER MINUTE       ";MN%
2070 IF MN%>59 OR MN%<0 THEN 2000
2080 IF HR%>23 OR HR%<0 THEN 2000
2090 IF DY%<1 OR DY%>7 THEN 2000
2100 A=(YR%/4-INT(YR%/4))*4
2110 IF MO%<1 THEN 2000
2120 IF MO%=4 OR MO%=6 OR MO%=9 OR MO%=11 THEN IF DT%>30 THEN 20
00
2130 IF MO%=2 AND A=0 THEN IF DT%>29 THEN 2000
2140 IF MO%=2 AND A<>0 THEN IF DT%>28 THEN 2000
2150 IF DT%>31 THEN 2000
2160 A(2)=INT(MN%/10)
2170 A(1)=INT((MN%/10-A(2))*10+.1)
2180 A(4)=INT(HR%/10)
2190 A(3)=INT((HR%/10-A(4))*10+.1)
2200 A(6)=INT(DT%/10)
2210 A(5)=INT((DT%/10-A(6))*10+.1)
2220 A(7)=DY%
2230 A(9)=INT(MO%/10)
2240 A(8)=INT((MO%/10-A(9))*10+.1)
2250 IF A=0 THEN A(10)=8 ELSE IF A=3 THEN A(10)=4 ELSE A(10)=A
2260 FOR I=1 TO 10
2270 POKE &HFFEE+I,A(I)
2280 NEXT I
2290 N=USR1(0)
2300 CLS:RETURN

5000 DATA "SUNDAY","MONDAY","TUESDAY","WEDNESDAY","THURSDAY"
5010 DATA "FRIDAY","SATURDAY"
5020 DATA "JANUARY","FEBRUARY","MARCH","APRIL","MAY","JUNE"
5030 DATA "JULY","AUGUST","SEPTEMBER","OCTOBER","NOVEMBER"
5040 DATA "DECEMBER"
```

End