The CompuTime/QT Clock Boards

by Leo Biese and Emilio Iannuccillo

At last fall's Boston Computer show we picked up what surely seemed a good bargain—a couple of QT "S100-Clock/Calendar+" kits (List $100 a kit/$150 A/T). Unfortunately, it turned out that the bargain boards did not quite work as implied by the designation "S-100," since they don't work with an 8080 CPU that meets the IEEE-696 standard! A letter to Don Smith, President of Q.T. Computer Systems, Inc. went unanswered for nearly one and a half months and, when received, indicated that he was unaware of this problem and would be interested in hearing about our fix. By this time the present review was underway and we did not follow-up his letter. While we were waiting for an answer we had noted the external similarity of the CompuTime ComputerWatch (their name is also on the QT board in small print, a point which we missed the first time around) and contacted them for further information. CompuTime president Gail Beaver was most helpful and kindly supplied their current board (marked S-100 880 REV B) for evaluation. CompuTime turned out to be the manufacturer of both boards and was well aware of the incompatibility problem, having revised the whole board some time ago. We will discuss the incompatability in this review, since many of the earlier versions are still around.

The Board

This full-function clock board is remarkably simple and requires only a backup battery and few support chips for the OKI MSM5832 monolithic "Microprocessor Real-Time Clock/Calendar" chip. This 18-lead CMOS integrated circuit contains its own oscillator and divider chain, 13 four-bit I/O registers for the seconds, minutes, hours, day-of-the-week, date, and year as well as the required chip-select, read, write, and test circuits and a +/-30 sec. correction feature we use programatically. A "hold" input maintains the time while preventing rollover of the clock during a read. Leap year correction is automatic, and either 12- or 24-hour time format can be selected. Details of the registers are covered in the documentation and need not be discusssed here. The board requires four consecutive I/O ports which can conveniently be selected by a DIP switch over the entire range of 0-255.

This full-function clock board is remarkably simple and requires only a backup battery and few support chips for the OKI MSM5832 monolithic "Microprocessor Real-Time Clock/Calendar" chip.

The very low power dissipation of this chip (90 micro Watts @ 3V) allows safe battery backup for several months with as little as 2.2 volts, in this case supplied by a 3.6 volt G.E. "Data Sentry" miniature Ni-Cad battery with on-chip automatic power-loss switching. The oscillator is driven by an external 32.768 Hz crystal (about the size of a 1/8 watt resister!); and a trimming capacitor is provided to "pull" the oscillator frequency. The frequency stability for the 5832 crystal oscillator is given as +/-2 ppm for an approximate two-fold change in operating temperature or a voltage drop to as low as two volts from the nominal five volts. This is an order of magnitude of only about one second per week, so obviously there are factors that effect the clock stability other than oscillator frequency. Since the chip runs at five volts (from the standard 7805 regulator) and is warmer when on-line; and then drops to 3.6 volts and a cooler environment when on standby, the accuracy of the clock is significantly affected. This is not a real problem with our use, dating print-outs, but it would

Biese/Iannuccillo, RFD1 Murray Hill Rd., Hill, NH 03243.

have been a considerable enhancement to have the board designed so that the alternate power sources were more closely matched. One of the boards tested lost about ten seconds per day despite repeated "tweaking" of the variable capacitor. The second board lost over an hour when it was removed from the computer for about two months.

A significant design flaw is the use of a horizontal access trimmer capacitor. Since the oscillator must be touched-up daily over a period of a week or more to maintain accuracy, the board has to reside atop an extender board until this is done. A top-mounted capacitor would have been better.

In addition to the basic clock/calendar function of the 5832 chip, the CompuTime/QT boards provide four hardware interrupt times at one hour, one minute, one second, and one millisecond (approximately) which are potentially useful in real-time process control if the board is kept activated.

In addition to the basic clock/calendar function of the 5832 chip, the CompuTime/QT boards provide four hardware interrupt times at one hour, one minute, one second, and one millisecond (approximately) which are potentially useful in real-time process control if the board is kept activated. As long as the computer is turned on, the accuracy is very good; not a single second was lost during a six hour session with the National Bureau of Standards (WWV) time signals coming into the computer room. Accuracy suffers only when the computer is turned off and the board goes into the 3.6 volt stand-by mode.

The board as supplied is clean, solder-masked, silk-screened and quite up to current manufacturing standards. There is plenty of kludge area available for your own special projects. The 35 page manual supplied is, if anything, too simply written and redundant—the register descriptions are presented in the theory of operation, in the programming section, and again in the appendix. The schematic is poorly done, but usable. Potential users would benefit by obtaining the OKI MSM5832 data sheet which is not supplied. Board-level manufacturers should follow the lead of the disc-controller providers and include the data sheets for "uncommon" chips with their documentation, since these can sometime be very difficult to obtain.

Several redundant sample programs in Basic are provided in the manual to set and read the clock, but we prefer our own version given in Listing 1. The program, "SETCLOCK," allows the clock to be synchronized with the national standard when accurate time measurements are needed. The U.S. National Bureau of Standards broadcasts time signals continuously on the 2.5, 5, 10 and 15 MHz shortwave frequencies that are readily received by even the most simple receiver anywhere in the continental U.S. (station WWV) and in the Pacific (station WWVH in Hawaii). In addition, the Canadian government also broadcasts universal time signals over its CHU channels on 3.33, 7.335 and 14.67 Hz. The details of the format for these signals can be found in any one of the many amateur radio or shortwave listener's handbooks. Essentially, the world-wide time standard is kept very accurately and announced by a distinctive tone on the second, and by a voice on the minute. SETCLOCK is self-prompting, and makes use of the +/- 30 second adjust input (pin 15) provided on the clock chip. When pulsed high, this pin zeroes the seconds counter and, if over 30 seconds are on the clock, also adds one minute. First you must set the Year, Month, Day/Date, and Hours according to a menu selection, you are then advised to set the clock ahead at least one minute. Unlike the programs supplied with the board, the time is always visible on the screen (by using the direct cursor addressing of the ADM3a and similar terminals) while entries are being made. At this point you simply wait for the minute mark and hit the return key. We use a compiled version of this program. The delay (Line 1490) can be adjusted so that Line 950 rings the terminal bell synchronously with the time signals.

The main use we have for this board is to print the date on program listings and runs, thereby getting rid of some of the confusion we have been living with over the years (since we never can remember when anything was printed). While the Basic programs provided are adequate, we also wanted this facility for assembly language listings and developed the program PTIME as a CP/M transient command given in Listing 2. Constant paranoia about whether or not the clock was correct led to a nearly identical program, directed to the console, which we incorporated into our CPM as an auto-load program (See cf. James J. Franz, "Turn-key CP/M Systems," *Creative Computing*, December 1979). With this modification, the time and the date is printed right below the CP/M sign-on message each time the system is brought up, and whenever a warm start is done. This allows, for example, correcting the clock if we have been on vacation and would have forgotten to reset it before printing.

Problems

The inability of this "S-100" board to work with many 8080 CPUs reminds us again that if the manufacturer doesn't say the board conforms to the IEEE-696 standard, it almost certainly doesn't and the potential user should proceed with caution.

Refer to this quote from the standard:

> "2.2.3 Status Bus. The status bus consists of eight lines which identify the nature of the bus cycle in progress, and qualify the nature of the address on the address bus."

One of the the status signals is sOUT. This signal, according to the standard, indicates the type of I/O in progress. Nowhere in the standard does sOUT 'time' the type of I/O.

Another excerpt from the standard:

> "2.7.5.2 The Write Strobe. The generalized write strobe, pWR*, is used to write data from the data bus into the addressed bus slave..... Data out on the data bus must be guaranteed valid for a specified period both before and after the activation of the write strobe. Hence, either the leading or the trailing edge of the write strobe may be used to strobe data into the addressed slave."

Herein lies the problem with the QT board (and the earlier CompuTime version). The design completely ignored the IEEE standard relating to the pWR* strobe, and does not even have that signal coming into the board. Instead, it uses the sOUT signal not only to indicate an OUT operation to the board, but also (and improperly) to time the data onto the board. Consequently an 8080, such as the Imsai which produces valid IEEE signals, cannot strobe the power data into the clock board. The 8080 needs both pWR* and sOUT to operate properly. The sOUT signal is latched by the 8212 (on the CPU board) from long before until long after the data is valid, but the pWR* signal is needed to indicate the period when the data is valid.

These timing problems should be obvious from the diagrams supplied by the IEEE standard and the CPU manufacturers. (We will not reproduce them here.) The pWR* signal is active for a much shorter time than sOUT. The clock board latches trigger on the release of sOUT; by this time pWR* has already gone away and the data is no longer stable. (For the non-hardware types, this means that you set the clock correctly, and the next day, it reads something ridiculous like: "December 66, 1999 18:48:91 PM"—because the data wasn't caught, or latched, at the proper time in the CPU cycle.)

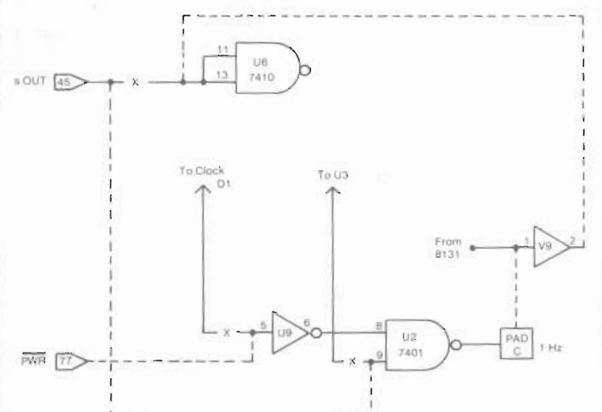
So why does it work with the Z80? It just so happens that the sOUT signal from the Z80 occurs at just about the same time as the pWR* signal. The Z80 has both signals and they look about the same. The Z80, however, does not latch the sOUT signal—in fact, the Z80 does not need to latch any signals, reminding us that while it may run 8080 code, the Z80 is an entirely different chip!

Note: *This problem does not apply to the "Revision B" board supplied by CompuTime. They have added an extra chip (U12) to pick up pWR* in a manner very similar to our Mod-2 below; the difference being that they AND pWR* with sOUT one step later. In addition they have provided pads for connection to the bus interrupt lines.*

Modifications

There are several ways to modify the board to make it IEEE compatible. (Note: while this article was in progress a board fix, without comment, was published by Zoso in *Lifelines* Vol.1, No.10. It is rather cryptic to say the least.)

Figure 1: *Modification #1; This requires no additional components but sacrifices the interrupt timers.*



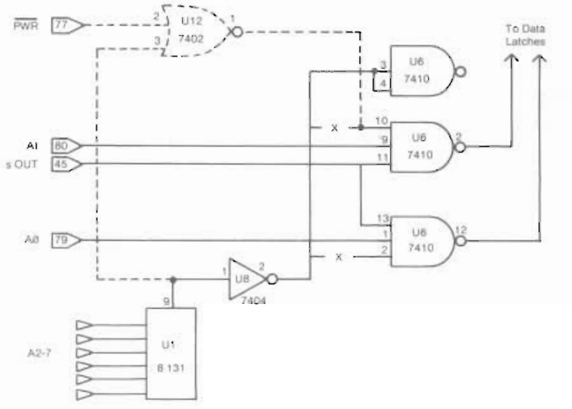
MOD #1 (Figure 1): We had no need of the interrupt features, so the fix was quite easy, mainly because we then had 1/2 of U3 (7421) available as well as all of U2 (7401) and a section of U9 (7404).

1. Cut the trace from Bus #45 (sOUT) to U6 (11,13).
2. Cut the trace leading to U9 (5) at the chip.
3. Cut the traces to U2 (9) on both the top and bottom of the board next to the chip.
4. Run a wire from Bus #77 (pWR*) to U9 (5).
5. Connect Bus #45 (sOUT) to U2 (9).
6. Connect pad C to U9 (1).
7. Connect U9 (2) to U6 (11,13).

MOD #2 (Figure 2): If you desire to keep all functions of the board intact you will have to add a gate. We used a 72LS02 and called it U12.

1. Tie U12 (14) to +5 volts and (8) to ground.
2. Cut the trace from U6 (2) to U6 (3) on the solder side of the board.
3. Cut the trace from U6 (10) to U6 (4). This trace is on the component side under the socket. It must be done before assembly or the socket will have to be removed.
4. Tie U12 (2) to Bus #77 (pWR*).
5. Tie U12 (3) to U1 (9).
6. Tie U12 (1) to U6 (10 and 2).

Figure 2: *Modification #2; All functions of the original board can be retained by adding a gate (U12).*



Summary

To reemphasize, the modifications pertain to the QT and earlier CompuTime boards only; the current CompuTime revision B works perfectly as is. Despite the problems we encountered, we consider each of the boards to be a bargain and a valuable addition to the system. After approximately six months of use, it is hard to think of using a computer without them. The assembled and tested boards are the same price but you can save $25 by getting the QT kit and making the changes suggested here.

For real-time process control the boards are highly accurate and it seeems hard to justify some of the other clock boards currently on the market for several hundred dollars. Indeed, Gail Beaver at CompuTime tells us that this is a major use of their board; for everything from timing rides at amusement parks to controlling grain elevators in Australia!

As a simple time/data board, it is a 'must' for every S-100 bus computer. Any sort of billing or reporting use of the microcomputer, such as a professional practice, requires the addition of at least a date—and many types of reports also require a time entry as well. This facility has long been standard on virtually all mini- and mainframe computers and is now available for the micro user at a reasonable cost. ■

For more information contact:

Compu/Time, P.O. Box 5343, Huntington Beach, CA 92646; (714)536-5000.

QT Computer Systems, Inc., 15620 Inglewood Ave., Lawndale, CA 90260; (800)421-5150.

OKI Semiconductor, 1333 Lawrence Expressway, Suite 104, Santa Clara, CA 95051.

```
 ######  #######  ######    #####  ##          #####    #####  ##   ##
##       ##          ##    ##   ## ##         ##   ##  ##   ## ##  ##
##       ##          ##    ##      ##         ##   ##  ##      ## ##
 #####   #####       ##    ##      ##         ##   ##  ##      ####
     ##  ##          ##    ##      ##         ##   ##  ##      #####
     ##  ##          ##    ##   ## ##         ##   ##  ##   ## ##  ##
######   #######     ##     #####  #######     #####    #####  ##   ##

100 'SETCLOCK: Program to set the parameters of the Computime/QT
110 '          Clock/Calendar boards
120 '
130 'Programed in Microsoft 5.2 BASIC by E.D.I and L.P.B. based
140 'on the manuals supplied (November 1980)
150 '
160 ' 2/01/81 Revised for better screen presentation and a more rapid
170 '         keyboard scan approximating realtime seconds. (LPB)
180 '
190 ' 2/14/81 Input for 'seconds' correction eliminated since BASIC
200 '         is not fast enough to keep up. The PROMPT$(X) now
210 '         begins at 3 (LPB)
220 '
230 ' 6/06/81 Universal time 'SYNC' revised with added prompting (LPB)

240 '========= SET PARAMETERS AND STORE TEXT FOR OUTPUT ===============

250 CLEAR.SCREEN$=CHR$(26)              'Clear ADM3a screen code
260 RING.BELL$=CHR$(7)                  'Ring ADM3a terminal bell
270 ADDR=130                            'Clock input port
280 DAT =129                            'Clock output port
290 KEYBOARD=17                         'System (MITS 2SIO) port

300 DIM DAY$(6),MONTH$(12),PROMPT$(13),T(13),T$(13),U$(13)

310 FOR I=0 TO 6: READ DAY$(I): NEXT
320 FOR I=1 TO 12: READ MONTH$(I): NEXT
330 FOR I=2 TO 12: READ PROMPT$(I): NEXT

340 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
350 DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST
360 DATA SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER

370 DATA MINUTE UNITS,MINUTE TENS,HOUR UNITS,HOUR TENS
380 DATA DAY OF THE WEEK [Sunday=0  Monday=1  Sat=6]
390 DATA DAY UNITS,DAY TENS,MONTH UNITS,MONTH TENS
400 DATA YEAR UNITS,YEAR TENS

410 '============== PRINT THE CRT SCREEN ==============================

420 PRINT CLEAR.SCREEN$
430 PRINT TAB(12);
440 PRINT" -----------------------------------------------------"
450 PRINT TAB(12);
460 PRINT"|   To reset the system clock enter the function #    |"
470 PRINT TAB(12);
480 PRINT"|                                                     |"
490 PRINT TAB(12);
500 PRINT"|  #1 for MINUTES    #3 for DAYS       #5 for YEARS   |"
510 PRINT TAB(12);
520 PRINT"|  #2 for HOURS      #4 for MONTHS     #6  to SYNC    |"
530 PRINT TAB(12);
540 PRINT"|                                                     |"
550 PRINT TAB(12);
560 PRINT"|             #7 to END and return to CP/M            |"
570 PRINT TAB(12);
580 PRINT"|                                                     |"
590 PRINT TAB(12);
600 PRINT"|  Do NOT use CTR/C to end as it may stop the clock   |"
610 PRINT TAB(12);
620 PRINT" -----------------------------------------------------"
630 PRINT
640 PRINT TAB(12);STRING$(54,"=")

650 '================= READ THE CLOCK REGISTERS =======================

660 'The clock registers can be read directly,  unfortunatly we need
670 'to change them to ASCII text for the printout.  Converting puts
680 'in an obligatory leading blank (the implied sign) which must be
690 'stripped off.
700 '
710 ' Variables used:
720 '
730 '     T(x) = a matrix holding the numeric contents of the registers
740 '    U$(x) = temporary matrix holding the string values of T(x)
750 '    T$(x) = ASCII string values of the registers for printout
760 '
770 'See the product literature for register assignments

780 OUT DAT,16                               'Stop the clock
790 I=0                                      'Set Register counter
800 FOR D=32 TO 44                           'For all registers
810     OUT ADDR,D                           '  point to register
820     T(I)=INP(ADDR)                       '  read it
830     I=I+1                                '  bump the counter
840     NEXT D                               '  and get the next
850 OUT DAT,0                                'Now restart the clock

860 '--- Each pass thru the read routine checks for a leap year flag
         and the 12/24 hour format request

870 IF T(8)>3 THEN T(8)=T(8)-4               'Adjust for FEB 29
```

```
880 B=T(5):IF B >7 THEN P$=" ": T(5)=T(5)-8
890        IF B <8 AND T(5) >3 THEN P$=" PM" ELSE P$=" AM"
900        IF T(5) >3 THEN T(5)=T(5)-4
910 '--- Change numeric register data to string and strip off the
      obligatory leading blank

920 FOR I=0 TO 12:U$(I)=STR$(T(I)):T$(I)=RIGHT$(U$(I),1):NEXT

930 '============= PRINT THE CLOCK DATA ============================

940 PRINT CHR$(27)+CHR$(61)+CHR$(43)+CHR$(53)            'Set cursor
950 PRINT RING.BELL$                                      'Tick seconds
960 PRINT
970 PRINT TAB(14)"today is ";DAY$(T(6));"   ";
980 X=T(10)*10+T(9):PRINT MONTH$(X);" ";
990 PRINT T$(8);T$(7);",19";T$(12);T$(11);"   ";
1000 PRINT T$(5);T$(4);":";T$(3);T$(2);":";T$(1);T$(0);P$
1010 PRINT
1020 PRINT TAB(12);STRING$(54,"=")
1030 PRINT

1040 '================ CHANGE THE CLOCK REGISTERS ==================


1050 '--- Check keyboard for input

1060 X = INP(KEYBOARD)
1070 X = X AND 127 - 48
1080 IF X <> 6 THEN 1100

1090 PRINT TAB(12)"SYNC: Hit return just BEFORE zero tone ";:GOTO 1480

1100 PRINT TAB(12)"Enter function #                       ";:
1110 ON X GOTO 1140,1150,1190,1230,1240,1480,1520

1120 GOTO 780                            'Reprint clock till keypressed

1130 '--- Response to keboard entry:
        X = the register to change
        Y = the desired contents

1140 X=3: PRINT:PRINT TAB(7);:GOTO 1260                  'Get minutes

1150 PRINT
1160 X=5: INPUT "        AM or PM";PM$                   'Afternoon?
1170       IF PM$="PM" THEN PM=-1 ELSE PM=0              ' Flag it
1180       GOTO 1260                                     'Get hours

1190 PRINT:PRINT
1200 X=8: INPUT"       IS THIS LEAPYEAR ";LEAP$
1210       IF LEFT$(LEAP$,1)="Y" THEN LEAP= -1
           ELSE LEAP=0                                   ' Flag it
1220       GOTO 1260                                     'Get day

1230 X=10: GOTO 1260                                     'Get month

1240 X=12: GOTO 1260                                     'Get year

1250 '-------- Data input for the clock registers

1260 PRINT:PRINT TAB(8);PROMPT$(X);
1270 INPUT Y
1280 IF X = 5 AND PM THEN Y = Y + 4
1290 IF X = 8 AND LEAP THEN Y = Y + 4
1300 GOSUB 1400                                  'Send it

1310 '--- Return and get the next register. We load them backwards
        ie. tens and then units

1320 X=X-1
1330 PRINT TAB(8);PROMPT$(X);
1340 INPUT Y
1350 GOSUB 1400                                  'Send it

1360 IF X = 7 THEN GOTO 1320            'User wants to synch the clock
1370 OUT DAT,0
1380 GOTO 420

1390 '-------- The actual clock register is changed here

1400 OUT ADDR,X                         'Point to the register
1410 OUT DAT,Y + 16                     'Send it the new data
1420 OUT ADDR,X + 16
1430 OUT ADDR,X
1440 RETURN

1450 '======= ROUTINE TO SYNC THE CLOCK WITH WWV OR CHU ============

1460 '  On the west coast: WWV or WWVH on 5.0, 10.0, or 15.0 MHz
1470 '  On the east coast: CHU (Canada) on 3.33, 7.335 or 14.67 MHz

1480 OUT DAT,32                         'Raise the +/- 30 sec.
1490 FOR A=1 TO 20: NEXT A              'We need a delay here
1500 OUT DAT,0                          'Restart the clock
1510 GOTO 780                           'and read the clock again

1520 PRINT CHR$(26):OUT DAT,0:END
```

[LOG: JULY 13,1981   11:34:07 PM]

```
; PTIME: Prints time and calendar on the CP/M console (ADM3a)
; and lister (Diablo 1640) using the Computime/QT clock board
;
; By Emilio D. Iannuccillo, 825 Hope St, Bristol, R.I.02809
;     January 1981
;
; Restructured by L.P. Biese,Hill,N.H. 03243   May 24, 1981
```

```
CLEARSCREEN     EQU     26      ;ADM3
HOME            EQU     30      ;ADM3
CLOCKADDRESS    EQU     82H     ;Port for clock address
CLOCKDATA       EQU     81H     ;Port for clock data

; The following equates represent
; appropriate address to the clock chip.
; Invoked by outputting to port CLOCKADDRESS above.

YEARTENS        EQU     12
YEARUNIT        EQU     11
MONTHTENS       EQU     10
MONTHUNIT       EQU     9
DAYTENS         EQU     8
DAYUNIT         EQU     7
WEEKDAY         EQU     6
HOURTENS        EQU     5
HOURUNIT        EQU     4
MINUTETENS      EQU     3
MINUTEUNIT      EQU     2
SECONDTENS      EQU     1
SECONDUNIT      EQU     0

;To the above: ADD 32 to RAISE CLOCK READ LINE
;              ADD 16 to RAISE CLOCK WRITE LINE

READLINE        EQU     32
WRITELINE       EQU     16

;The CLOCKDATA port is used to read and write time,
;however, 16 must be added to Data port to stop clock
;from advancing while each register is being accessed.

HOLD            EQU     16


;-----------------------------------------------------------
; The operation of this program is simply to:
;     a) Stop the clock from advancing
;     b) Read clock data into memory
;     c) Restart the clock
;     d) Convert clock data into ascii
;     e) Fill a print-buffer with the ascii data, and
;     f) Print out the buffer contents.
;     g) Exit back to CPM
;-----------------------------------------------------------


; INITALIZE. Program begin here. Save old CPM stack for
; re-entry to system at conclusion of the program

        ORG     100H            ;beginning of CPM TPA area
        LXI     H,0             ;clear HL
        DAD     SP              ;load with CPM stack pointer
        SHLD    RETURNSTACK     ;and save it.
        LXI     SP,STACK        ;set up our own stack

; RAISE CLOCK HOLD LINE to stop the clock from advancing

        MVI     A,HOLD          ;get code
        OUT     CLOCKDATA       ;and send it

; READ CLOCK DATA. Set up a loop to latch the clock
; addresss, read the data and save it. Repeat until
; the 13 clock registers are read.

        MVI     B,32            ;Set B for 0, the first
                                ; clock register + 32 (for read)
        MVI     C,13            ;number of registers
        LXI     H,TIMETABLE     ;Area to store raw data
L1:     MOV     A,B             ;Start of the read loop
        OUT     CLOCKADDRESS    ;Output & latch register wanted
        PUSH    PSW             ;Need 6 microsec delay at 2mh for
        POP     PSW             ; clock chip to catch up
        IN      CLOCKDATA       ;Read the register
        MOV     M,A             ;Store it
        INX     H               ;Bump storage location
        INR     B               ;Bump to next register address
        DCR     C               ;One less to go!
        JNZ     L1              ;Get the next one
        XRA     A               ;Done.
        OUT     CLOCKDATA       ;Restart clock

; CONVERT raw data just read into asci and stuff it
; into a print buffer for later output to devices

        LXI     H,TIMETABLE     ;Start of raw data
        MOV     A,M             ;Pick up seconds
        ORI     30H             ;Convert to ascii
        STA     S1              ;Store at proper point in
        INX     H               ; print buffer
        MOV     A,M             ;Pick up tens digit of seconds
        ORI     30H             ;Convert to ascii
        STA     S10             ;Store in print buffer
        INX     H               ;Advance to minute units
        MOV     A,M             ;Get it
        ORI     30H             ;Convert it
        STA     M1              ;Store it
        INX     H               ;minute tens
        MOV     A,M             ;Repeat
        ORI     30H             ;for
        STA     M10             ;minute tens digit
        INX     H               ;oh,hum
        MOV     A,M
        ORI     30H
        STA     H1
        INX     H               ;ah, hour tens, a little
        MOV     A,M             ;more interesting because it
        ANI     3               ;contains am/pm or 24 hr flag
        ORI     30H             ;First set to ascii and
        CPI     30H             ;Check to see if it is a 0
        JNZ     OVER6           ;Go forward if not a 0, else
        XRA     A               ;wipeout 0 for sake of print looks
OVER6:  STA     H2              ;Store it
        MOV     A,M             ;Get hour tens data again
        ANI     8               ;check if 24 hr time
        JZ      NOT24HR         ;Go forward if not 24 hr,
        XRA     A               ;else if it is 24 hr format
        STA     AM$PM           ;then wipe out AM in the print
```

```
        STA     AM$PM+1         ; buffer. It's meaningless.
        JMP     DAY$ROUTINE

NOT24HR:
        MOV     A,M             ;If 12 hr format, then
        ANI     4               ;test if AM or PM
        JZ      DAY$ROUTINE     ;Ø=AM so leave as is
        MVI     A,'p'           ;else change 'A' in AM
        STA     AM$PM           ;to P

DAY$ROUTINE:
        INX     H               ;Pick up
        MOV     A,M             ;clock data for day
        RLC                     ;Adjust it to make a
        RLC                     ;table pointer
        LXI     D,DAYTABLE      ;Point to ascii day table
        ADD     E               ;Adjust pointer
        MOV     E,A             ;to point to proper day
        JNC     OVER2           ;Check to see if table
        INR     D               ;crossed a page boundry

OVER2:  LXI     B,DAY           ;Point to print buffer
        LDAX    D               ;and move
        STAX    B               ;the day
        INX     D               ;pointed to
        INX     B               ;into the
        LDAX    D               ;proper position
        STAX    B               ;in the
        INX     D               ;print buffer
        INX     B
        LDAX    D
        STAX    B
        INX     D
        INX     B
        LDAX    D
        STAX    B
        INX     H               ;Next point to and
        MOV     A,M             ;get day units
        ORI     3ØH             ;Here's ascii again
        STA     D1              ;And of course storage
        INX     H               ;Day tens is next
        MOV     A,M             ;Get it
        ANI     3               ;Only lower bits for day
        ORI     3ØH             ;Change to ascii
        CPI     3ØH             ;Check if Ø because we
        JNZ     OVER5           ;don't want a Ø to print
        XRA     A               ;wipe out the Ø, if any
OVER5:  STA     D1Ø             ;store it
        INX     H               ;Point to month units
        LXI     D,MONTHTABLE    ;Again we have
        MOV     A,M             ;a table for ascii month
        RLC                     ;Multiply data
        RLC                     ;by 16
        RLC                     ;Each month in monthtable
        RLC                     ;takes up 16 bytes
        ADD     E               ;This is more than really
        MOV     E,A             ;needed. But 16* makes for
        JNC     OVER4           ;ease in adjusting the
        INR     D               ;pointer

OVER4:  INX     H               ;While holding above
        MOV     A,M             ;position, get
        ORA     A               ;month tens,
        JZ      OVER3           ;If Ø go jump ahead
        MVI     A,16Ø           ;Else adjust pointer
        ADD     E               ;by a factor of 1Ø * 16
        MOV     E,A             ;Again check for
        JNC     OVER3           ;crossing page
        INR     D               ;boundry

OVER3:  LXI     B,MONTH         ;Point B to print buffer

MONTHLOOP:
        LDAX    D               ;Move
        CPI     Ø               ;ascii month
        JZ      MONTHDONE       ;pointed to
        STAX    B               ;by DE to
        INX     D               ;print buffer
        INX     B               ;position as
        JMP     MONTHLOOP       ;pointed to by BC

MONTHDONE:
        INX     H               ;For year, we're
        MOV     A,M             ;back to GET
        ORI     3ØH             ;Convert
        STA     Y1              ;Store

; PRINT IT. Print buffer is now the picture wanted
; for output. So send it to the console and lister

        LXI     D,THE$DATE      ;Point to print buffer
        MVI     C,9             ;Send entire line to console
        CALL    5               ;using CPM lineprint conventin
        MVI     E,ØDH           ;next for listing
        CALL    LISTER          ;device, first
        MVI     E,ØAH           ;output a
        CALL    LISTER          ;CRLF, then
        LXI     D,THE$DATE      ;reset pointer to print table

NEXTLETTER:
        LDAX    D               ;and loop
        CPI     '$'             ;through the buffer
        JZ      LISTERDONE      ;printing until $ is reached
        CPI     Ø               ;Also check for Ø
        JNZ     OK2PRINT        ;Do not print Ø
        INX     D
        JMP     NEXTLETTER

OK2PRINT:
        PUSH    D               ;Save print buffer pointer
        MOV     E,A             ;Get character to print
        PUSH    D               ;save it

;--------- Hardware specific for Diablo 164Ø ----------
; This block is code for my Diablo printer. It prints
; the character twice, but offset 1/12Ø of an inch
; from each other. It gives the appearance of being
; a bold print. For a straight forward listing on any
; CPM listing device, eliminate the code in this block
; down to End Diable Code.

; Put Diablo in 1/12Ø spacing mode by sending 'ESC 31 2'

        MVI     E,27
        CALL    LISTER
        MVI     E,31
        CALL    LISTER
        MVI     E,2
        CALL    LISTER

        POP     D               ;get print character
        PUSH    D               ;save chr again
        CALL    LISTER

; Restore Diablo to normal spacing mode

        MVI     E,27            ;Sequence is
        CALL    LISTER          ;ESC  83
        MVI     E,'S'
        CALL    LISTER

;--------------- End of Diablo 164Ø code -----------------


        POP     D               ;Get print character
        CALL    LISTER          ;Print it
        POP     D               ;and go on
        INX     D               ;to next character
        JMP     NEXTLETTER


;This is a subroutine that makes a CPM call
;to output byte in E register to the listing
;device.

LISTER: MVI     C,5
        CALL    5
        RET

LISTERDONE:
        MVI     E,ØDH           ;Now print a CRLF
        CALL    LISTER
        MVI     E,ØAH
        CALL    LISTER

BACK2CPM:
        LHLD    RETURNSTACK     ;Get CPM stack pointer
        SPHL                    ;Put it where it belongs
        RET                     ;and BACK TO CPM we go


;=========================================================
;                 TABLES and STORAGE AREA
;=========================================================

DAYTABLE:
        DB      'Sun '  ;Ø as read
        DB      'Mon '  ;1 from clock
        DB      'Tue '  ;2
        DB      'Wed '  ;3
        DB      'Thu '  ;4
        DB      'Fri '  ;5
        DB      'Sat '  ;6

MONTHTABLE:
        DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø ;ØØ as read
        DB      'January',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø      ;Ø1 from the
        DB      'February',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø        ;Ø2 two month
        DB      'March',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø     ;Ø3 registers
        DB      'April',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø     ;Ø4 on clock
        DB      'May',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø   ;Ø5
        DB      'June',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø    ;Ø6
        DB      'July',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø    ;Ø7
        DB      'August',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø      ;Ø8
        DB      'September',Ø,Ø,Ø,Ø,Ø,Ø,Ø         ;Ø9
        DB      'October',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø       ;1Ø
        DB      'November',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø        ;11
        DB      'December',Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø        ;12

THE$DATE:

;---------------------------------------------------------
;   Printing  format:
;          [LOG: SEPTEMBER 21, 1981  12:3Ø:3Ø PM]
;          Ø123456789ABCDEFØ123456789ABCEDFØ12345    << byte
;          Ø               1               2         << count
;---------------------------------------------------------

        DB      '[LOG: '
MONTH   DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,' '
D1Ø     DB      '2'
D1      DB      '1, 19'
Y2      DB      '8'
Y1      DB      '1   '
H2      DB      '1'
H1      DB      '2'
        DB      ':'
M1Ø     DB      '3'
M1      DB      'Ø'
        DB      ':'
S1Ø     DB      '3'
S1      DB      'Ø'
        DB      ' '
AM$PM   DB      'a'
        DB      'm'
        DB      ']'
        DB      '$'
DAY     DS      4                       ;Day was not used
                                        ;Simply put before $
                                        ;if wanted in the printout


TIMETABLE       DS      16
                DS      128
STACK           EQU     $
RETURNSTACK     DS      2

                END
```