REAL TIME CLOCK/CALENDAR

It seems strange that many microcomputers cannot tell the time of day or the date when such a facility can be so useful to the programmer. Never fear, ETI is here, with a simple peripheral for 6502-based machines. Design by M.D. Bedford

Programmers who are familiar with mainframe or minicomputers will probably be aware that it is generally possible to access the actual time and date from within a program. Such a facility is known as a real time clock and is often not available on the more modest microcomputers. It is not difficult to see that a real time clock would enhance any system — applications range from control programs, to the determination of the elapsed time between occurrences, to giving listings that professional touch by using the time and date in the header.

Two approaches are possible for the implementation of a real time clock — software or hardware. Traditionally, a software solution has been used in which a hardware interrupt is generated at regular intervals, probably every 20 milliseconds, these being counted by the interrupt handling routine which then calculates the time and date. Such a system obviously requires initialising and would prompt the user for the time and date each time the computer was switched on *(our own word processor uses this system — Ed).* Quite apart from the possible inconvenience, this method is probably unsuitable for most microcomputer users as it would require modification of the monitor program in ROM to prompt for the time and date. On the other hand, it is possible to devise a hardware alternative with battery back-up which is transparent to the system when not being accessed and doesn't lose the time and date on power-down of the main system.

For these reasons a hardware approach is presented here. The design is primarily intended for the Tangerine Microtan system, the PCB given here being of such a size that it will plug directly into the system rack. From an electronic point of view, however, there is no reason why the board may not be used with any 6502-based computer.

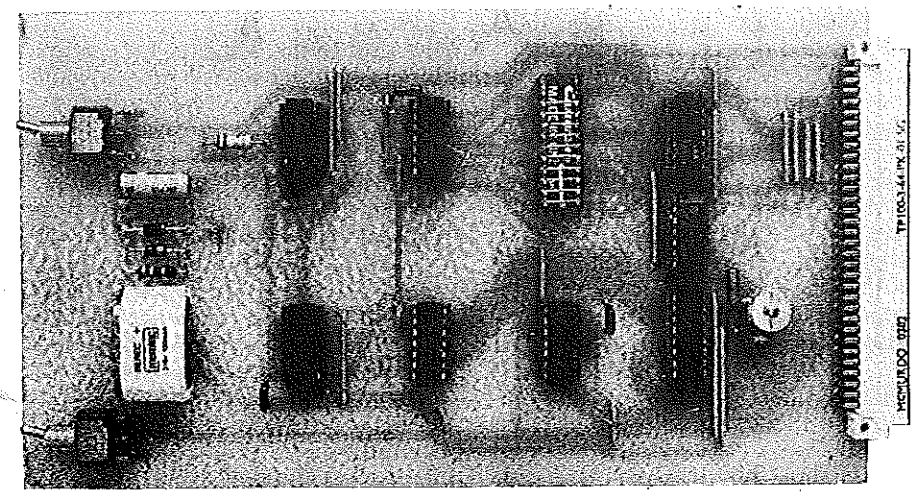
Functional Description

The real time clock, which may be configured to occupy any 16-byte block within the Tangerine I/O area, has 16 registers as specified in Table 1. It will be noticed that 12 of the registers are used to store the time and date, two registers being used to store (in BCD format) any number which may take a value greater than nine. For example, the value of the minutes is calculated as (10*REGISTER 5) + REGISTER 4. Of these 12 registers, registers 1-3 are read-only, these 'seconds' registers being automatically set to zero on starting the clock.

Each time the clock is updated, ie every tenth of a second, a flip-flop is set, writing a value of 15 to all the readable registers to indicate that an update has taken place since the last read. Reading a register under these conditions resets the flip-flop so that a further read will produce a valid result.

This board may also be used to generate interrupts at regular intervals, this function being controlled by register 15 as described in Table 2. Switch SW2 may be used to disable interrupts, a facility which is especially useful in view of the fact that this board does not reset at switch-on.

The remaining registers are write-only and have various control functions. Register 0 should have a value of 0 written to it to select non-test mode for normal operation. A value of 1, 2, 4 or 8 should be written into register 13 to indicate 'leap year, leap year + 1, leap year + 2, or leap year + 3 respectively. A value of 1 written to register 14 will start the clock, whereas a value of 0 will stop it. Switch SW1 gives the board write-proteciton, hence obviating the accidental overwriting of the time and date once initialised. This facility does not affect register 15 so that interrupts may still be selected when the



A bird's eye view of the completed project.

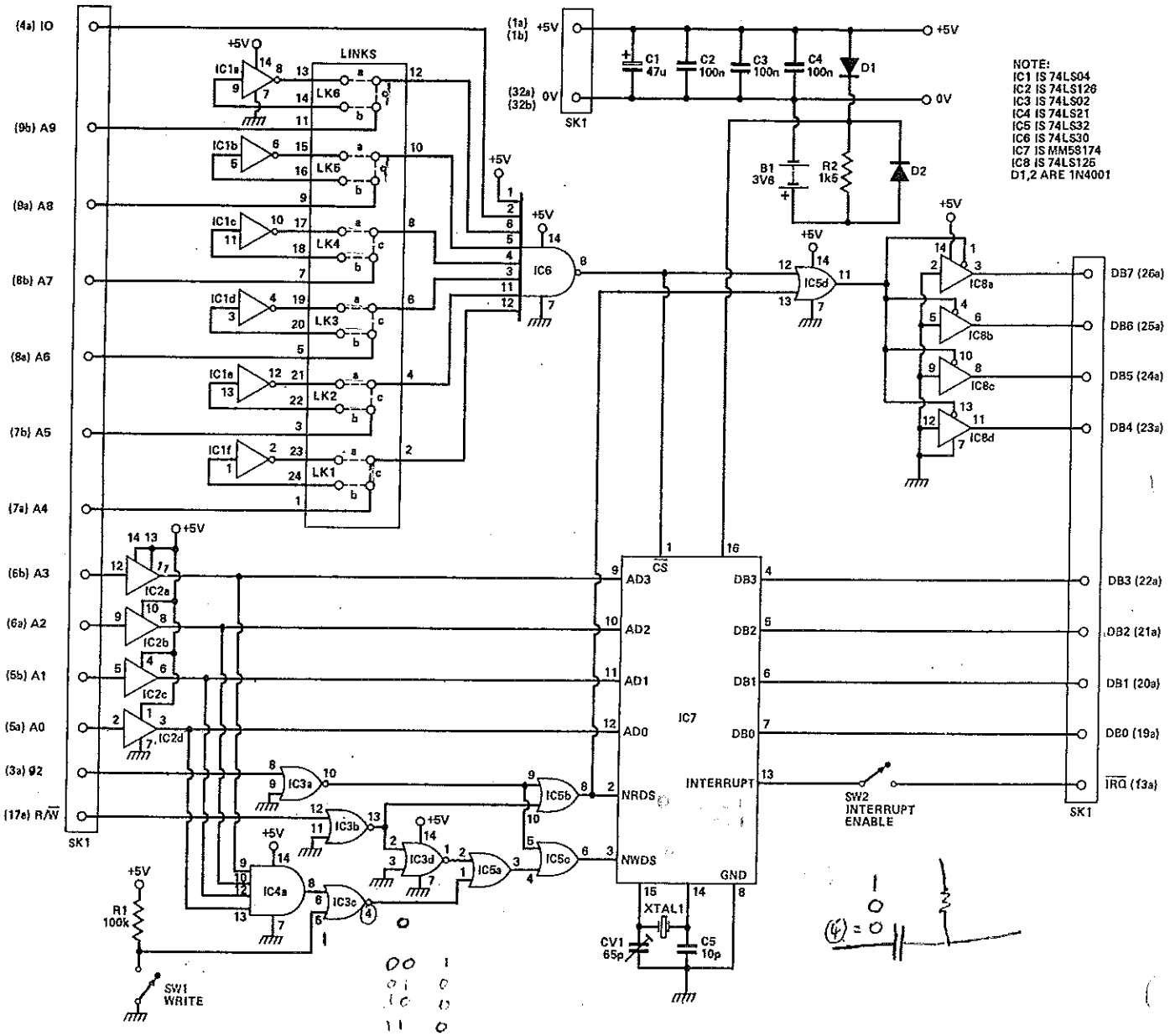


Fig. 1 Circuit diagram of the real time clock/calendar. Non-Microtan owners will find a circuit to generate the IO signal in last month's ETI.

board is write-protected. Both switches are mounted so that 'down' selects the enabling of the appropriate function.

The battery back-up facility allows data to be retained when the computer is not switched on, hence avoiding the need to initialise the clock at power-on. The time and date will be retained for about three months with a fully charged battery and a minimum of one hours use every nine days will ensure that the battery remains in a state of full charge.

Construction

If the printed circuit board layout presented here is adhered to, construction should present no difficulties. Since the board is of a single-sided design, a number of wire links need to be fitted as shown on the component layout diagram. Sockets should be used throughout for the integrated circuits. It should be noted that the MM58174 IC is fabricated in CMOS and accordingly the usual precautions of not touching the pins of the IC and not soldering the board while the IC is in its socket should be adhered to.

We suggest that DIL headers plugged into DIL sockets should be used for the wiring of the selectable address links. A 16-pin and an 8-pin socket should be used to make up the 24-pin by 0.3" socket used for these links. The required start address should be set up as follows: the start of the board is 16* (the binary number represented by links 1-6) from the start of the Tangerine I/O area, where link 1 is the least significant bit. Making links a and b gives a 0, making link c gives a 1. So, for example, the following links will set up the board to start at 48 bytes from the start of the I/O area: link 6 ab, link 5 ab, link 4 ab, link 3 ab, link 2 c, link 1 c. If the board is to be constructed to a different layout to suit non-Tangerine systems, the only points to be borne in mind are that C2, C3 and C4 should be well distributed around the board and that XTAL1, CV1 and C5 should be mounted close to IC7.

HOW IT WORKS

The heart of the circuit, IC7, is the MM58174 real time clock which reads and writes four bits of data onto DB0-DB3. ALthough not absolutely necessary (since the top four bits could be masked out by programming), a neat hardware solution is provided by the use of IC8 to zero DB4-DB7 during read operations. The circuitry comprising IC1, IC6 and the DIL links gives a chip select for IC7 and IC8 when an address in the range selected by the links is accessed.

Since the MM58174 is specifically intended to interface with microprocessors such as the 8080 or Z80, the circuitry comprising IC3 and most of IC5 is required to generate the NRDS and NWDS signals from the 6502 R/$\overline{W}$ and $\phi 2$. Hence write protection may be provided by blocking NWDS when SW1 is in the closed position. IC4 is used to detect when register 15 is being addressed (A0-A3 all high) and under these circumstances overrides the write protection.

IC2 is to buffer A0-A3 — in fact, the whole circuit is designed to present no more than one TTL load to any bussed signal.

D1 is used to pass the +5 V supply to IC7 when it is present, the battery being trickle-charged through R2 under these conditions. When the +5 V supply is not present, D1 prevents the battery from discharging through the power supply and IC7 is supplied with sufficient voltage to operate in standby mode via D2.

PARTS LIST

Resistors (all ¼W, 5%)	
R1	100k
R2	1k5
Capacitors	
C1	47u 16 V axial electrolytic
C2-4	100n disc ceramic
C5	10p ceramic plate
CV1	5-65p trimmer
Semiconductors	
IC1	74LS04
IC2	74LS126
IC3	74LS02
IC4	74LS21
IC5	74LS32
IC6	74LS30
IC7	MM58174 (see Buylines)
IC8	74LS125
D1,2	1N4001
Miscellaneous	
SW1,2	SPCO PCB-mounting toggle switches (see Buylines)
B1	3V6 PCB-mounting Nicad battery (see Buylines)
SK1	2 x 32 way A + B DIN Euro connector (male, angled pins) — see Buylines

PCB (see Buylines); DIL sockets to suit.

XTAL1 32.768

TABLE 1

List of Real Time Clock Registers

Reg No	Function	Access Mode
0	test	write
1	tenths of seconds	read
2	units of seconds	read
3	tens of seconds	read
4	units of minutes	read/write
5	tens of minutes	read/write
6	units of hours	read/write
7	tens of hours	read/write
8	units of days	read/write
9	tens of days	read/write
10	day of week	read/write
11	units of months	read/write
12	tens of months	read/write
13	year status	write
14	start/stop	write
15	interrupt	read/write

TABLE 2

DESCRIPTION OF INTERRUPT MODES

Function	Value in Register 15
no interrupts	0 or 8
single interrupt after 60 seconds	4
repeated interrupts at 60 second intervals	12
single interrupt after 5 seconds	2
repeated interrupts at 5 second intervals	10
single interrupt after 0.5 seconds	1
repeated interrupts at 0.5 second intervals	9

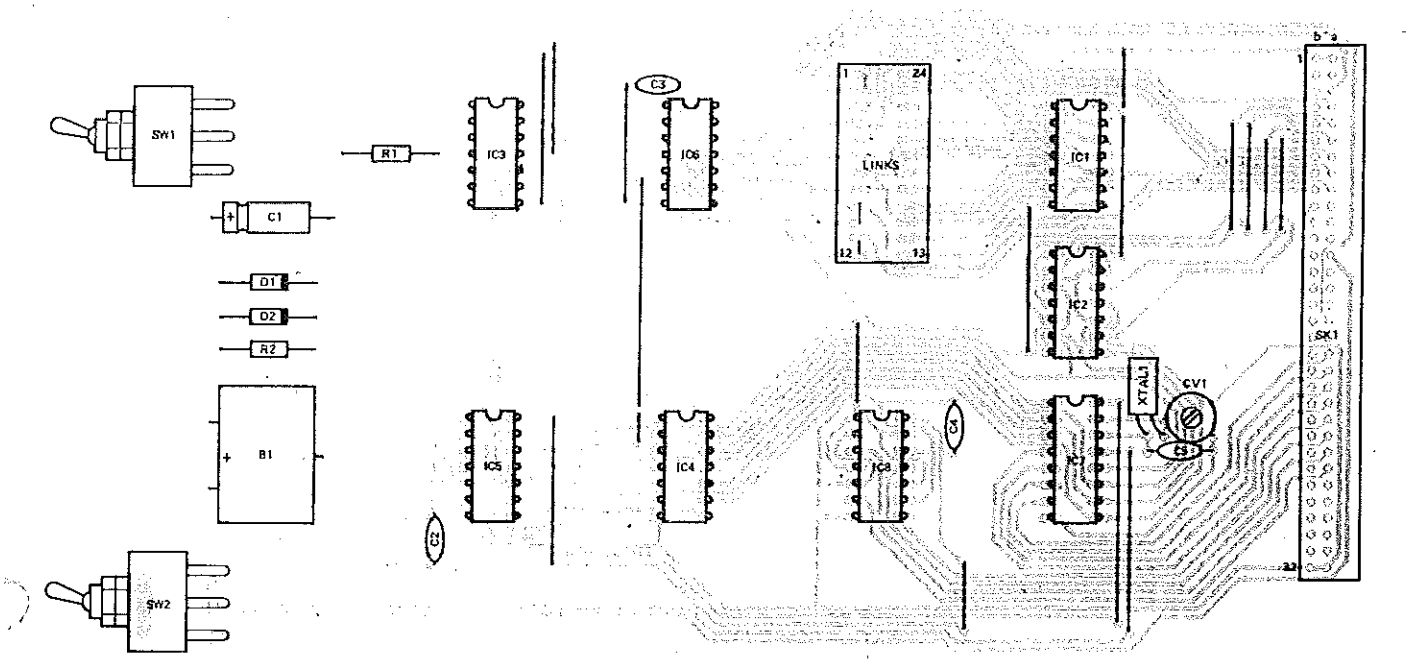
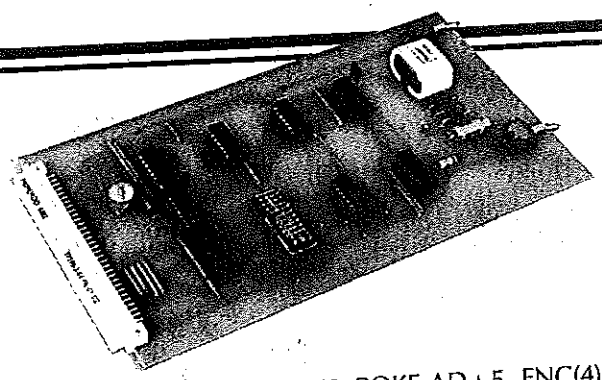


Fig. 2 Component overlay for the real time clock.

3 19



Programming

The following BASIC program is used for initialising the real time clock/calendar. The board should be write-enabled before running the program — however, if this is not done the user will be instructed to do so by the program. The program will fully validate the information given before writing it to the clock, to reduce the likelihood of human errors. We suggest that a time and date a few minutes ahead of the actual time is entered, the RETURN following the day of the week request being pressed exactly as this time arrives.

```
10 REM . . .MM58174 REAL TIME
   CLOCK INITIALISATION
   PROGRAM
20 DEF FNC(I) = VAL(MID$(TD$,
   I, 1))
30 DEF FNN(I) = 10 * FNC(I) +
   FNC(I+1)
40 DIM DM(12)
50 DATA 31, 28, 31, 30, 31, 30,
   31, 31, 30, 31, 30, 31
60 FOR I=1 TO 12: READ
   DM(I):NEXT I
70 PRINT "MM58174
   INITIALISATION"
80 INPUT "ENTER START
   ADDRESS OF BOARD"; AD
90 POKE AD, 0:REM . . . NON
   TEST MODE
100 POKE AD+15, 0:REM . . .
   DISABLE INTERRUPTS
110 POKE AD+14, 0:REM . . .
   STOP CLOCK
120 I=PEEK(AD+4):I=PEEK(AD+4)
130 J=15 AND (I+1):POKE AD+4,
   J
140 I=PEEK(AD+4)
150 IF I=J THEN 180
160 PRINT "WRITE ENABLE REAL
   TIME CLOCK — RETURN
   WHEN DONE"; :GET A$
170 GOTO 120
180 INPUT "ENTER TIME AND
   DATE IN THE FORM HH MM
   DD/MM/YY"; TD$
190 HH = FNN(1)
200 IF HH<0 OR HH>23 THEN
   180
210 POKE AD+7, FNC(1): REM . . .
   HOURS * 10
220 POKE AD+6, FNC(2): REM
   . . .HOURS
230 MM = FNN(4)
240 IF MM<0 OR MM>59 THEN
   180
250 POKE AD+5, FNC(4):REM . . .
   MINUTES * 10
260 POKE AD+4, FNC(5):REM
   . . .MINUTES
270 YY = FNN(13)
280 IF YY<0 OR YY>99 THEN 180
290 YR =
   2↑(3-(YY-4*INT(YY/4)))
300 IF YR=8 THEN
   DM(2)=29:GOTO 320
310 DM(2)=28
320 POKE AD+13, YR:REM . . .
   YEAR STATUS
330 MM = FNN(10)
340 IF MM<1 OR MM>12 THEN
   180
350 POKE AD+12, FNC(10):REM
   . . . MONTH * 10
360 POKE AD+11, FNC(11):REM
   . . . MONTH
370 DD = FNN(7)
380 IF DD<1 OR DD>DM(MM)
   THEN 180
390 POKE AD+9, FNC(7):REM . . .
   DAY * 10
400 POKE AD+8, FNC(8):REM . . .
   DAY
410 INPUT "ENTER DAY OF WEEK
   (1-7, 1=MONDAY)"; DW
420 IF DW<1 OR DW>7 THEN
   410
430 POKE AD+10, DW:REM . . .
   DAY OF WEEK
440 POKE AD+14, 15:REM . . .
   START CLOCK
450 PRINT "WRITE DISABLE REAL
   TIME CLOCK"
460 STOP
470 END
```

To access the time and date from within a program, the following BASIC subroutine may be used: a few words of explanation are probably appropriate. Line 1040 clears the update flip-flop by reading the clock once. The following two lines then loop until a value of 15 is read, indicating that an update has just taken place and that a tenth of a second is available to read the registers before the next update. It is the requirement to read 11 registers in this 100 milliseconds time slot (in order to avoid the possiblity of an update occurring between the reading of two registers) which accounts for the strange appearance of much of the rest of the subroutine. The inherent slowness of BASIC on an eight-bit microcomputer dictated the avoidance of FOR-NEXT loops, subscripted variables and numerical constants in the time-critical portion. The routine returns with numeric values of seconds, minutes . . . months in R2-R7 respectively, an ASCII representation of the time in TM$ and an ASCII version of the date in DT$.

```
1000 REM . . .MM58174 READING
   ROUTINE
1010 R2=AD+2:R3=AD+3:R4=
   AD+4:R5=AD+5
1020 R6=AD+6:R7=AD+7:R8=
   AD+8:R9=AD+9
1030 RA=AD+10:RB=AD+11:
   RC=AD+12
1040 Z = PEEK(AD+2)
1050 Z = PEEK(AD+2)
1060 IF Z<>15 THEN 1050
1070 R2=PEEK(R2):R3=PEEK(R3):
   R4=PEEK(R4)
1080 R5=PEEK(R5):R6=PEEK(R6):
   R7=PEEK(R7)
1090 R8=PEEK(R8):R9=PEEK(R9):
   RA=PEEK(RA)
1100 RB=PEEK(RB):RC=PEEK(RC)
1110 TM$=CHR$(48+R7)+CHR$
   (48+R6)+":"+CHR$(48+R5)
   +CHR$(48+R4)
1120 TM$=TM$+":"+CHR$(48+-
   R3)+CHR$(48+R2)
1130 DT$=CHR$(48+R9)+CHR$
   (48+R8)+" "+MM$(RB+
   10*RC)
1140 R2 = R2 + 10*R3
1150 R3 = R4 + 10*R5
1160 R4 = R6 + 10*R7
1170 R5 = R8 + 10*R9
1180 R6 = RA
1190 R7 = RB + 10*RC
1200 RETURN
```

Prior to calling the above subroutine, the following portion of program should be executed to store the names of the months in the array MM$:

```
10 DIM MM$(12)
20 DATA "JANUARY",
   "FEBRUARY", "MARCH",
   "APRIL", "MAY", "JUNE"
30 DATA "JULY", "AUGUST",
   "SEPTEMBER", "OCTOBER",
   "NOVEMBER", "DECEMBER"
40 FOR N=1 TO 12: READ
   MM$(N):NEXT N
```

BUYLINES

The MM58174 real time clock/calendar IC is available from Cricklewood Electronics, Technomatic or Watford Electronics. The PCB-mounting switches and Nicad battery might be a bit tricky to find unless you have industrial contacts, but non-PCB types could be used and wires taken to the PCB pads; there's enough room on the PCB, which is available from our PCB Service as usual. See page 87. The Euro connector is stocked by Watford Electronics.