REAL-TIME CLOCK

Richard Sargent

If you want to know the time, ask an MM58174. OK, it hasn't got quite the same ring about it but it can be very useful to your micro.

These days most computers have internal timers which can be configured to count and display in hours, minutes and seconds, but they can't tell you the date and the day of the week. Yes, I know you probably know what day of the week it is, but your software doesn't. Futhermore, if you disable your computer's interrupts, the chances are that it will lose all sense of time completely.

This project describes the construction of a Real-Time Card with battery back-up. Designed to interface with an eight-bit output latch and an eight-bit input buffer, it should prove suitable for a host of micros. The software and port addresses given in this project are for the powerful MTX 500/512 Micro, which has the advantage of an uncommitted 16-bit I/O socket nestling on its PCB!

THE MM58174

The MM58174 has a total of 16 internal registers which hold time and control data (see Table 1). Normally you would use the CPU address, data and READ and WRITE lines to access these registers as though they were any normal memory location, but in the design presented here it is software acting through the I/O port which communicates with the chip. For the present, though, it is easier to assume that straightforward READs and WRITEs are being made to the timer chip.

When the timer is first powered up it is necessary to enter the correct data into the device registers and start the clock running. A nibble (four bytes) on the data pins is used to pass BCD data to a correctly addressed register and in this manner all the internal counters are set to the desired time. When the time-keeping registers have all been set, the clock is started by sending a high on D0 to register 14. Conversely, a low written there will stop the clock counting. Incidentally, all starts reset the seconds counter to zero.

Don't forget to write data 8 to register 15 if you're building this card in 1984 (Table 2, the leap year status register). By some strange oversight there is no readout capability on this register, so the micro can't tell if the current year is a leap year! However, there is a hardware solution to this problem which will be considered later.

Reading data from the other registers is done by interrogating the low nibble for a valid clock or calendar value. A simple READ is not quite sufficient because a timer register might be updated during the actual read operation, and when this happens the MM58174 will deliberately return the illegal BCD code 1111. This enables the detection of a faulty READ situation. Software running slowly under BASIC will tend to pick up quite a few "1111" codes, and they must, of course, all be trapped.

Register 15 can be programmed as an interrupt timer giving 0.5, 5.0 or 60 second intervals and can be coded for single or repeated operations. The open drain interrupt output is pulled to ground when the timer times out and reading the interrupt register provides status and internal selected information. See Table 3.

TABLE 1

MM58174 INTERNAL REGISTERS

No	Name	Mode
0	Not used	
1	Tenths of sec	R
2	Units of sec	R
3	Tens of sec	R
4	Units of mins	R/W
5	Tens of mins	R/W
6	Units of hours	R/W
7	Tens of hours	R/W
8	Units of days	R/W
9	Tens of days	R/W
10	Day of week	R/W
11	Units of month	R/W
12	Tens of month	R/W
13	Years	W
14	Stop/start	W
15	Interrupt	R/W

TABLE 2

YEARS STATUS REGISTER

	DB3	DB2	DB1	DB0
Leap year	1	0	0	0
Leap year + 1	0	1	0	0
Leap year + 2	0	0	1	0
Leap year + 3	0	0	0	1

This register is a shift register and the contents are rotated to the right every 31st December.

TABLE 3

INTERRUPT READ/WRITE REGISTER

Function	DB3	DB2	DB1	DB0
No interrupt	0	0	0	0
Interrupt at 60sec intervals	0/1	1	0	0
Interrupt at 5.0s intervals	0/1	0	1	0
Interrupt at 0.5s intervals	0/1	0	0	1

Write mode:
DB3=0 single interrupt DB3=1 repeated interrupt
Read mode:
DB3=0 no interrupt DB3=1 no interrupt

CIRCUIT DESCRIPTION

Points T0-T7 are the eight inputs of the host computer's input port which will typically be a 74LS244 (tri-state buffer) or a PIA/VIA configured as a tri-state buffer. The MTX uses a 74LS373, a transparent latch, and this may be considered as an ordinary tri-state buffer in this application.

Points L0-L7 are the eight outputs from the host computer's output port. This will usually be a 74LS374 (octal latch), as is the case of the MTX, or it might be the other half of the PIA etc. The outputs must be latched so they maintain the same state until told by software to change.

The outputs L0-L3 pass through another latch, IC4, which holds their value and thus stabilises an address on the A0-A3 pins of the timer, IC1. An ALE (address latch enable) pulse achieves this. While this is happening, READ and WRITE will be held high, thus preventing the same data from entering at the D0-D3 pins of IC1. Writing data to IC1 involves enabling the four tri-

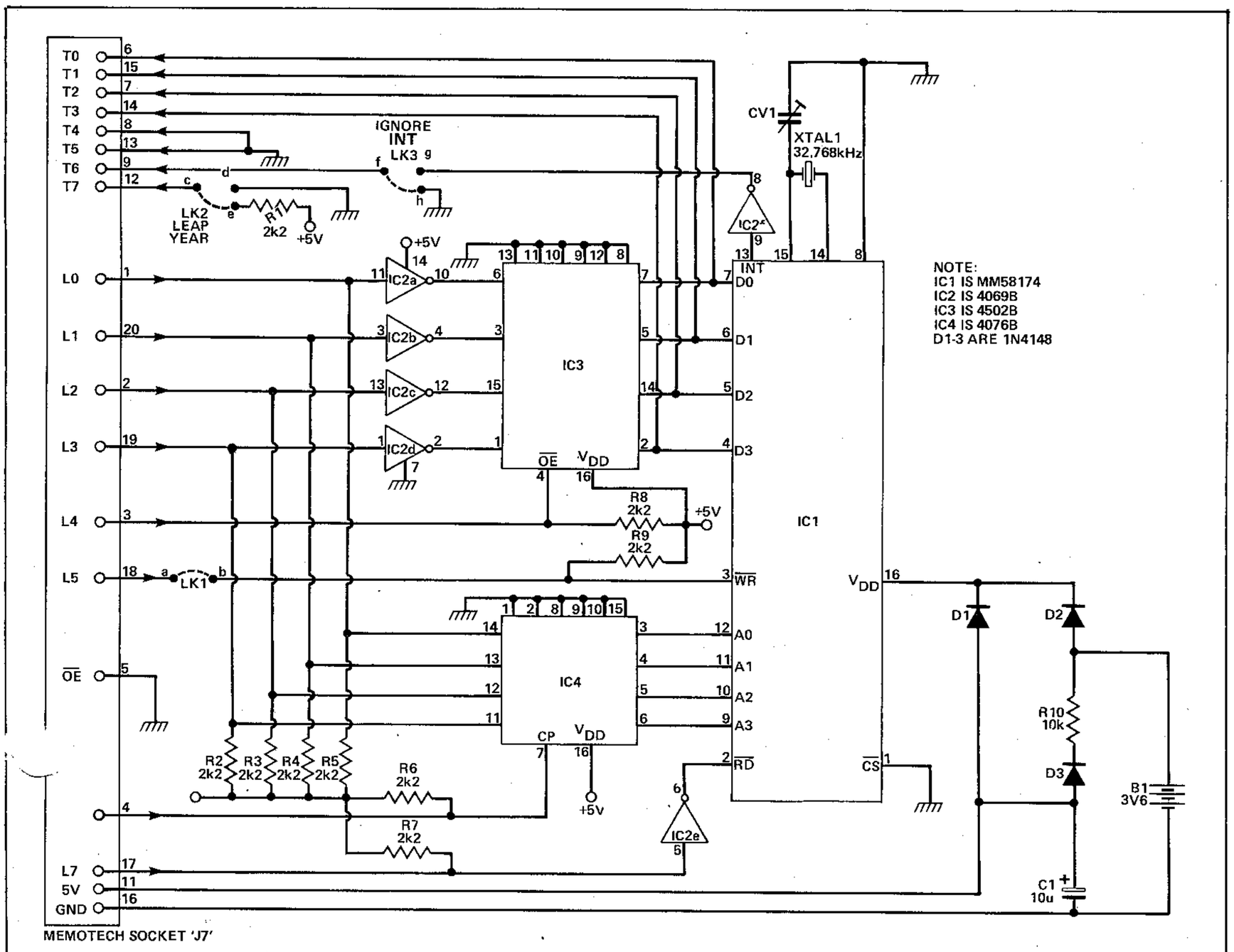

Fig. 1 Circuit diagram of the Real-time clock.

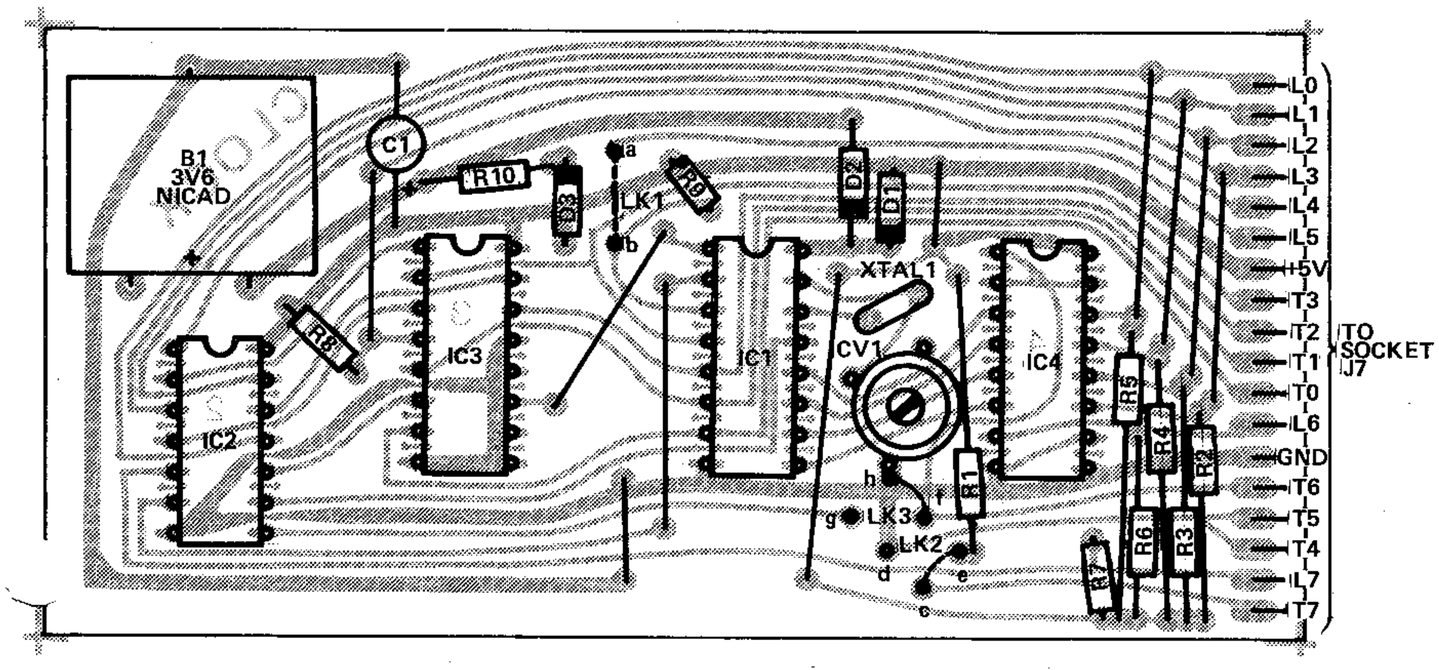

Fig. 2 Component overlay.

PARTS LIST

IC1	MM58174 CMOS timer
IC2	4069B CMOS inverter
IC3	4502B CMOS tri-state inverter
IC4	4076B CMOS latch
R1	10k ¼W 5%
R2-10	2k2 ¼W 5%
C1	10*u*F 16V tantalum capacitor
CV1	5-60pF trimmer (Cirkit 06-60001)
XTAL1	32,768 kHz crystal
D1-3	1N4148 diodes
NiCad	3V6 (Cirkit 01-03100)
IC sockets	three off 16-pin, one off 14-pin
One 20-pin header	
18 or 20-way ribbon cable	
PCB	

is also denied but periodic interrupts may still be read.

The T0-T7 lines read the timer directly, and it may prove convenient to tie the four most-significant bits to ground so that the complete eight-bit number represents the time value. Personally, I would rather allocate T7 as a leap-year indicator, and T6 as the interrupt reader, in which case the software must unscramble the information contained in those eight bits.

The 2k2 pull-up resistors on the L0-L7 are a necessary requirement when driving CMOS from TTL. The timer's clock is formed using a standard 32.768 kHz crystal across pins 14 and 15 and it is recommended that a 5-6pF trimmer be used to fine-tune the oscillator.

The remaining gate in IC2 is used as a buffer on the interrupt line, and its output may be fed to external equipment, if desired.

IC1's V_{DD} is connected to the battery supply line and the chip will maintain its data in standby mode on voltages down to 2V2. The MTX 5 V rail is capable of supplying the 1 mA needed to trickle-charge the small NiCad battery which provides a nominal 3V6 standby voltage to IC1. Running the computer for a few hours every week will keep the battery charged, and should you go on holiday the timer will retain its data. A fully charged NiCad should be capable of running the MM58174 for three months.

CONSTRUCTION

The integrated circuits are all CMOS devices so take extra care of them until the board is fully built. They should, of course, be the last items to go onto the board, and they must all go into IC sockets. The best way to bend the pins prior to pushing the chips into their sockets is to lay the IC on a sheet of kitchen-foil and exert the necessary pressure, having first removed any harmful static electricity by grounding yourself and the foil on the nearest water-pipe.

A PCB foil pattern is given for this project, so you shouldn't have any difficulties with the board itself, providing that you purchase the specified components. What you will have difficulty in making or obtaining is a 20-pin header to

state buffes in IC3 with a PWR (pre-write) pulse and then allowing the data through to IC1 with a WR pulse.

The inverter is needed on the READ line to guard against a situation where zero is written to the output latch L0-L7. The host computer's ROM might do this as an initialisation, or it might occur after a system crash. Without IC2 an all-zeros situation would set the MM58174 to a simultaneous READ and WRITE and that's not a good idea . . . .

If you want to permanently disable the WRITE line you should cut link LK1 on the PCB. The clock data cannot now be accidentally overwritten. Access to the interrupt register

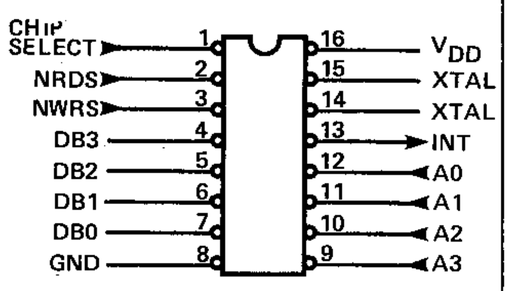

Fig. 3 The pin designations of the MM58174.

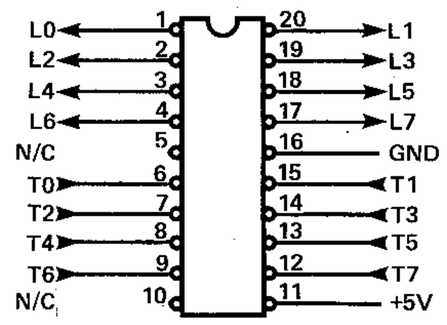

Fig. 4 Pin connections of the MTX parallel I/O port (port 07).

go into the 20-pin I/O socket on the MTX PCB. The socket is a standard 20-pin IC socket but it seems there isn't much call for 20-pin headers. I would suggest making one by splitting a 24-pin header into two 10-pin lengths. A short length of ribbon cable connects across to the Real Time board which is small enough to fit somewhere in the case of an unexpanded MTX.

Alternatively, it is but a simple matter to pass the ribbon cable out through the convenient gaps at the rear of the MTX case, in which case the timer can be housed in a tiny box Blu-Taked to the computer. The pin-outs of the MTX I/O socket are given in the user manual, but in case you have mislaid that, they are reproduced here as Fig. 4. Note that the +5 V pin on the MTX I/O port is only guaranteed to deliver 20 mA, so don't be tempted to replace IC2, IC3 and IC4 with TTL chips with their higher supply current requirements.

SOFTWARE

To read just one register of the clock/calendar from BASIC is a fiddle and a waste of time since for virtually the same length of code the whole set can be read at once. A numeric array, V(), accepts the data from the clock and a subroutine does the work: see Listing 1.

The numbers sent to the port on the OUT command are made up as follows:
R is a number between 0 and 15 representing one of the 16 possible registers in IC1.
16 is added whenever a high impedance output from IC3 is required.
32 is added whenever a WRITE to IC1 is *not* required.
64 is added whenever the ALE signal is required.
128 is added whenever a READ to IC1 *is* required.

Because of the hardwired leap-year indicator, the data received by INPut 7 will either be in the range 0-15 or 128-143. The code at line 8062 corrects this, and also sets V(13) to zero if it isn't a leap year.

Note the check for valid date at line 8070. If the data is 15, then the read occurred during one of the timer's update cycles and another reading must be taken immediately, line 8080.

For computers other than the Memotech, you will have to change INP to IN and the port values L and T must have port numbers appropriate to the machine. If IN and OUT commands don't exist, the port facilities may be mapped into memory and so a POKE will replace OUT and a PEEK will replace IN.

All that remains to be done now is to set the timer chip to GMT. For this a WRITE subroutine is required. Variable R passes the register number, and D passes the data, to the subroutine given in Listing 2. A simple loading routine can then be used to set the timer. See Listing 3.

The response to Register 14 should be 1, the start-timer code, and the carriage-return you make after that starts the clock "on the pips" if you're listening to TIM on the 'phone. Seconds are automatically zeroed whenever register 14 is used, so in theory total accuracy can be achieved. However, don't necessarily expect your timer to be 100 % accurate from the word go. You must be prepared to test it over a period of a few days and to tweak the trimmer if necessary, although I should think any software using the clock would accept an accuracy of ±60 seconds a week.

The whole board is best left outside the Memotech case for a few days until you are happy with the clock's timekeeping. When it's OK, I would suggest you cut the link on the WRITE line, and pop the unit inside the computer case or in its own little plastic box. The link on input line D6 should not be made unless you are going to experiment with interrupts, in which case you will need machine-code driving routines rather than the simple BASIC routines presented here.

Finally, a reminder to all computer buffs — it's late evening on December 31st and the revellers are abroad in Trafalgar Square and Scottish programmes are on all television channels:
what must you do to link LK2 on your real-time board?
Answers please on a postcard . . .

```
REM -- READ THE TIME
GOSUB 8000
PRINT V(7);V(6);":";V(5);V(4)
REM HOURS AND MINUTES DISPLAYED



 100 DIM V(13): LET L=7: LET T=7
8000 REM - TIME READING SUBROUTINE
8005 LET V(13)=0        :REM - LEAP YEAR FLAG ZEROED
8010 FOR R=1 TO 12      :REM - 12 TIME REGISTERS
8020 OUT L,R+16+32      :REM - OFFER ADDRESS TO IC4
8030 OUT L,R+16+32+64 :REM - LATCH ADDRESS INTO IC4
8040 OUT L,16+32+128   :REM - READ IC1
8050 LET V(R)=INP(T)   :REM - ACCEPT DATA FROM IC1
8060 OUT L,16+32        :REM - FINISH READ
8061 REM - STRIP AWAY BIT 7, SETTING LEAP YEAR FLAG
8062 IF V(R)>15 THEN  LET V(R)=V(R)-128:LET V(13)=1
8070 IF V(R)<15 THEN  GOTO 8090 :REM - DATA VALID ?
8080 OUT L,16+32+128 : LET V(R)=INP(T) : OUT L,16+32
8085 IF V(R)>15 THEN LET V(R)=V(R)-128
8090 NEXT R
8095 RETURN
```

Listing 1. Subroutine for reading the time.

```
8100 OUT L,R+16+32      :REM - OFFER ADDRESS TO IC4
8110 OUT L,R+16+32+64 :REM - LATCH ADDRESS INTO IC4
8120 OUT L,D+16+32+64 :REM - PRESENT DATA TO IC3
8130 OUT L,D+32         :REM - PASS DATA
8140 OUT L,D            :REM - WRITE DATA
8150 OUT L,D+32         :REM - FINISH WRITE
8160 OUT L,D+16+32      :REM - IC3 TRISTATE AGAIN
8170 RETURN
```

Listing 2. Subroutine to write to the timer.

```
10 LET R=14:LET D=0:GOSUB 8100:REM - STOP THE TIMER
20 FOR R=4 TO 14
30 PRINT "REGISTER ";R
40 INPUT "VALUE -- ";D
50 GOSUB 8100
60 NEXT R
70 STOP
```

Listing 3. A loader program using the subroutine in Listing 2 to load the GMT values.

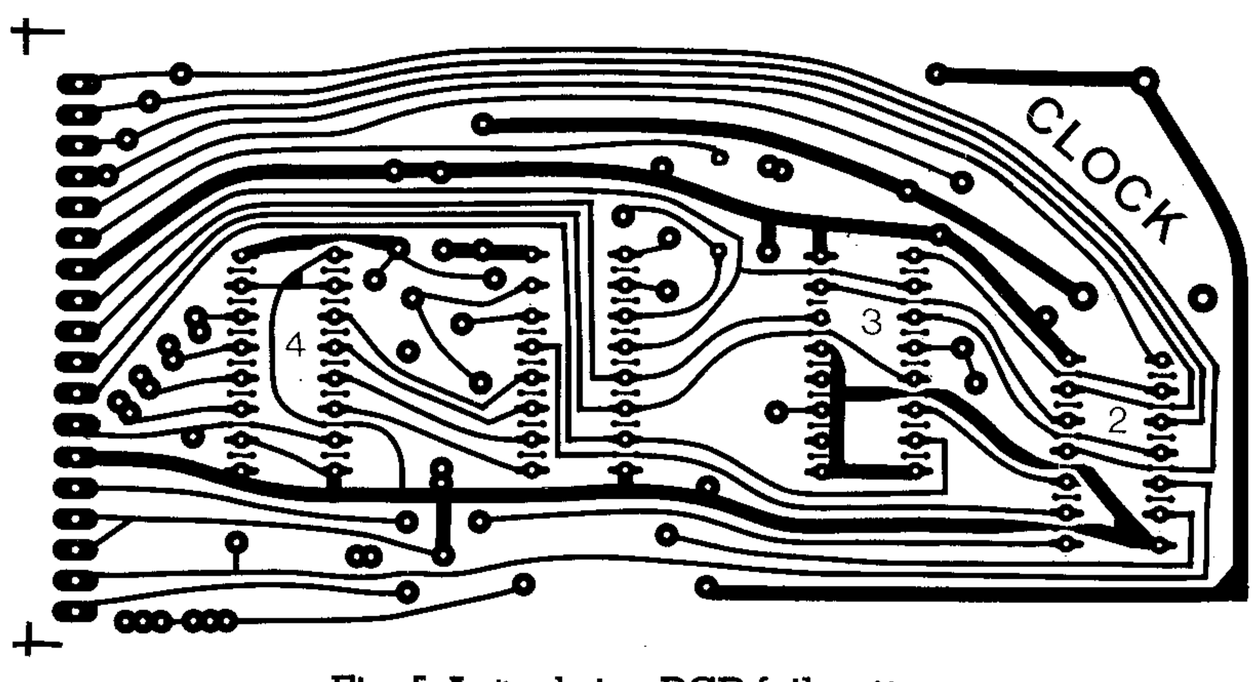

Fig. 5 Actual size PCB foil pattern.