MICROCONTROLLER-BASED TEMPERATURE INDICATOR

■ K. PADMANABHAN

Temperature measurement is now more common than ever before. Just look around and you will find even air-conditioners having built-in temperature indicators. Also, with a temperature probe, you can measure how hot is the water for bath.

Temperature is measured either in degrees Celsius or Fahrenheit, though the former is now standard. The ambient temperature keeps varying during different times of the day and night of any place. Here we describe a temperature indicator using AT89C2051 microcontroller, temperature sensor and other components.

Sensor

LM35 and LM34 are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Centigrade and Fahrenheit temperature values, respectively. The rate of change of the output voltage of both sensors with respect to the rise in temperature is 10 mV/degree. LM35 and LM34 have a full range of –55ºC to +150ºC and –50°F to +300°F, respectively. Both sensors are like three-pin transistors in plastic devices.

Microcontroller AT89C2051 does not have any analogue-to-digital converter to read the analogue signals (millivolts) as digital equivalents (digital numbers). However, it possesses two pins—pins 12 (P1.0) and 13 (P1.1)—that have the feature of a precision analogue voltage comparator. The comparator compares the two signal voltage levels at pins 12 and 13. When pin 12 (port-1 pin P1.0) is high with respect to pin 13 (port-1 pin P1.1), pin 3.6 of port 3 within the device goes high. (Pin 3.6 is not made available outside the device on any of its external ports.) Through an instruction that checks whether pin 3.6 is high, you can then infer that the signal level at pin 12 is higher than at pin 13.

We use this technique to measure the voltage rise due to the rise in sensor temperature. It is possible by applying the increasing voltage to one pin and the sensor's output voltage to the other pin. When the rising voltage crosses the sensor output voltage, pin 3.6 of microcontroller AT89C2051 will go high. Thus the instruction will understand that the voltage of the rising signal is just above that of your temperature sensor.

PARTS LIST

Semiconductors:	
IC1	- 7806 6V regulator
IC2	- LM35 Centigrade temperature sensor
IC3	- LM34 Fahrenheit temperature sensor
IC4	- CD4066 analogue switch
IC5	- AT89C2051 microcontroller
IC6	- ULN2003 current buffer
T1-T5	- BC557 pnp transistor
ZD1	- 5.1V, 1W zener diode
D1-D4	- 1N4007 rectifier diode
D5, D6	- 1N4148 switching diode
DIS1-DIS5	- LTS542 common-anode, 7-segment display
Resistors (all ¼-watt, ±5% carbon):	
R1	- 5-ohm, 2W
R2, R4	- 10-kilo-ohm
R3	- 1-kilo-ohm
R5-R8	- 1.2-kilo-ohm
R9, R10	- 220-ohm
VR1	- 10-kilo-ohm preset
RNW1	- 10-kilo-ohm resistor network
Capacitors:	
C1	- 1000µF, 25V electrolytic
C2, C3, C5	- 0.1µF ceramic disk
C4	- 10µF, 16V electrolytic
C6, C7	- 22pF ceramic disk
Miscellaneous:	
X1	- 230V AC primary to 9V, 500mA secondary transformer
S1	- On/off switch
S2	- Push-to-on switch
X_{TAL}	- 12MHz crystal
CON1, CON2	- 3-pin bergstick male connector
CON3	- Connector for power supply

EFY SUNIL KUMAR TESTED

For temperature measurement, you need to know the value of the rising signal voltage, which will give the value of the equivalent temperature at that moment. This is done with the technique of charging a capacitor with constant current. When you charge a capacitor (a non-leaky capacitor made of polyester metal film) with constant current, the voltage will rise linearly. First, discharge the capacitor by short-circuiting its terminals, then open the shorting switch and allow the constant current to flow through it. It will raise the voltage across the capacitor uniformly or charge it linearly.

Here, we have used transistor T5 with an emitter resistor and fixed base bias to provide constant current, since we know that a high-gain transistor has a constant-current output, no matter how the voltage across its collector and emitter varies. The bias voltage to its base together with the emitter resistor will decide the value of current. Use two diodes (D5 and D6) in the biasing circuit and preset resistor VR1 for limiting the value of current. By using diodes, the constancy of the base voltage is somewhat assured. Calibrate by setting preset VR1 so that you get a linear rise of 1 mV per

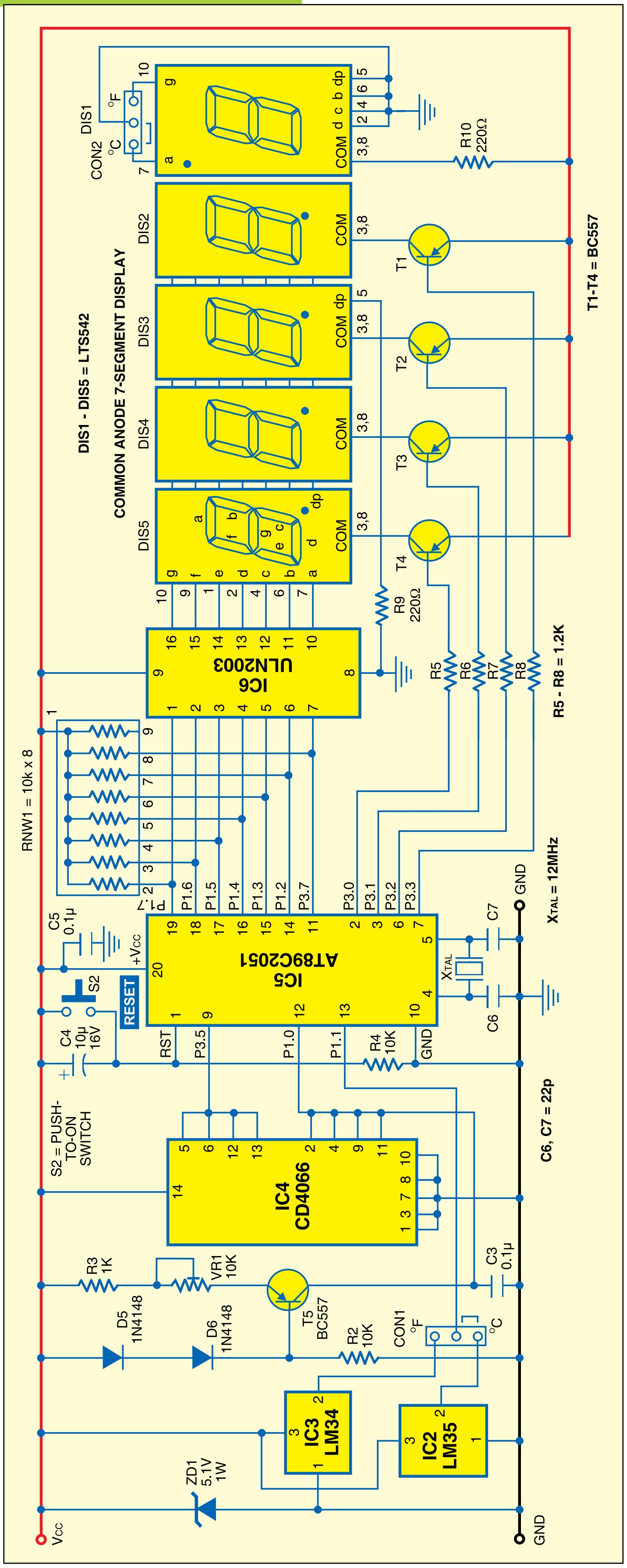

Fig. 1: Circuit of microcontroller-based temperature indicator

microsecond. Synchronisation of starting of the capacitor-charging cycle and measurement of voltage and time elapsed is done through microcontroller AT89C2051 (IC5) and analogue switch CD4066 (IC4).

AT89C2051 has two internal hardware timers. Timer-0 ticks and counts every microsecond once it is started. So start this timer as soon as you open the capacitor-shorting switch and thus reckon time in microseconds till its pin 3.6 goes high. That tells you the value of millivolts in numerals.

Continuous shorting and de-shorting of the capacitor for charging and discharging displays the temperature measurement value online. For this continuous activity of shorting and de-shorting, you need an electronic switch. This function of the switch is performed by CD4066.

CD4066 has four such switches inside. Each switch has two leads, one of which is connected to ground and the other to capacitor C3, which would be charging. There is a control voltage pin for each switch. For example, pin 13 of IC4 is the control voltage pin for the switch that is formed with pins 1 and 2 of the same IC. This signal originates from pin 9 of IC5, which if at high logic, will short the capacitor.

Wait until the capacitor is completely discharged. Then open the switch by making control pin 9 low. Simultaneously, the timer starts counting from zero.

Circuit description

Fig. 1 shows the circuit of the microcontroller-based temperature indicator. Microcontroller AT89C2051 is at the heart of the circuit. Port-1 pins P1.7 through P1.2, and port-3 pin P3.7 are connected to input pins 1 through 7 of IC ULN2003 (IC6), respectively. These pins are pulled up with a 10-kilo-ohm resistor network RNW1. They drive all the segments of the 7-segment display with the help of inverting buffer IC6.

Port-3 pins P3.0 through P3.3 of IC5 (microcontroller) are connected to the base of transistors T4 through

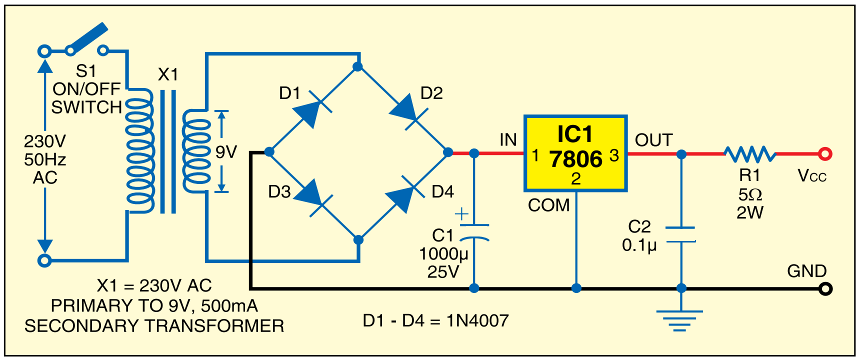

Fig. 2: Power supply for microcontroller-based temperature indicator

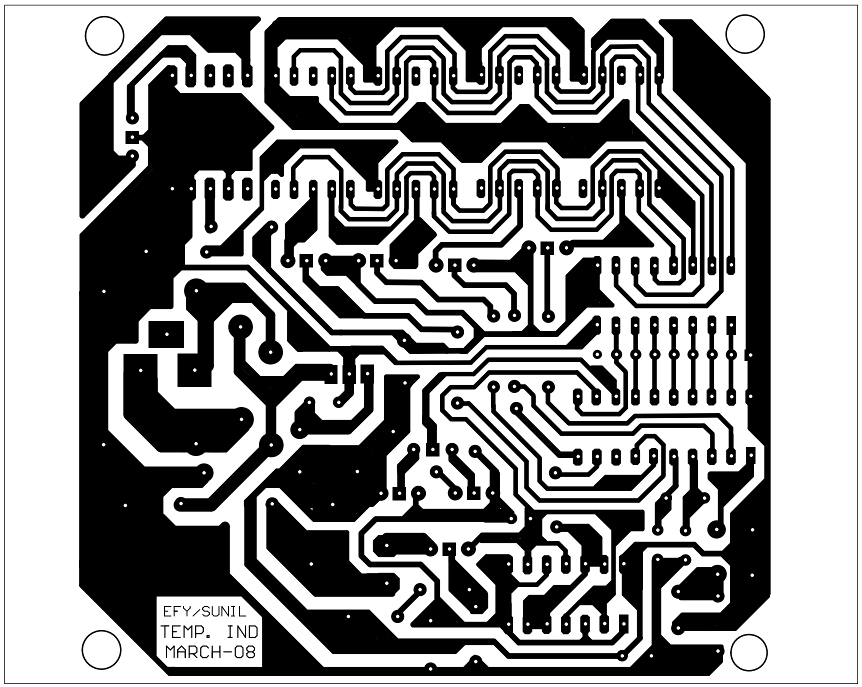

Fig. 3: Actual-size, single-side PCB for the microcontroller-based temperature indicator

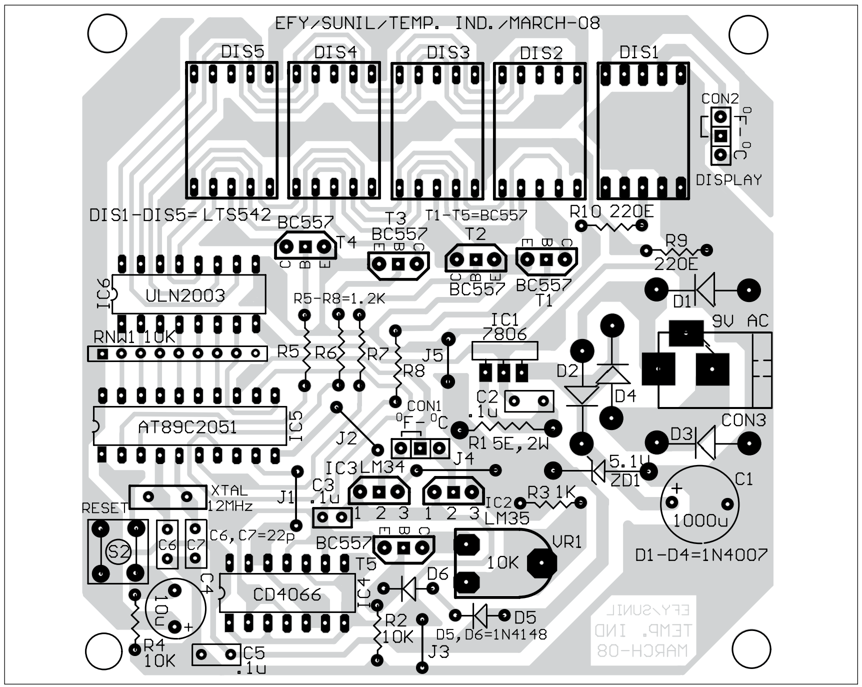

Fig. 4: Component layout for the PCB

T1 to provide the supply to 7-segment displays DIS5 through DIS2, respectively. Pin P3.0 of IC5 goes low and drives transistor T4 into saturation, which provides supply to either common-anode pin 3 or pin 8 of 7-segment display DIS5. Similarly, transistors T1 through T3 provide supply to the common anodes of the 7-segment display.

IC5 provides the segment data and display-enable signals simultaneously in time-division-multiplexed mode for displaying a particular number on the 7-segment display unit. Segment-data and display-enable pulse for the display are refreshed every 5 ms. Thus, the display appears to be continuous, even though it lights up the LEDs sequentially.

Switch S2 is used to manually reset the microcontroller, while the power-on-reset signal for the microcontroller is derived from the combination of capacitor C4 and resistor R4. A 12MHz crystal is used to generate the basic clock frequency for the microcontroller. Resistor R9 connected to pin 5 of DIS3 enables the decimal point. R10 is connected to the common pin 3 or 8 of DIS1 to provide the supply to it. DIS1 is used to indicate the °C or °F.

Fig. 2 shows the circuit for power supply. 230V AC mains is stepped down by transformer X1 to deliver the secondary output of 9V, 500 mA. The transformer output is rectified by a full-wave bridge rectifier comprising diodes D1 through D4, filtered by capacitor C1 and then regulated by IC 7806 (IC1). Capacitors C2 bypasses any ripple present in the regulated power supply. Zener diode ZD1 provides the constant 5.1V to the circuit.

Construction and testing

An actual-size, single-side PCB for the microcontroller-based temperature indicator is shown in Fig. 3 and its component layout in Fig. 4. The sensor (LM35 or LM34) should be kept inside a closed-end metallic tube. Use bases for ICs AT89C2051, ULN2803 and CD4066. Handle CD4066 with care. Don't touch its pins as it may damage the metal-oxide field-effect transistors inside with the static electric charge developed with touch.

Program AT89C2051 and put it into the IC base after soldering all other components. Press switch S2 to reset the system. The 7-segment display will show a random reading. If the reading is not steady, check for zener voltage at 5.1V and for loose connections or dry soldering joints. When the reading is steady, adjust preset VR1 at the collector of transistor T5 in the capacitor-charging circuit. By shorting the jumper, you can select between measurement in Centigrade and Fahrenheit as per indication.

Software

The software is written in Assembly language and assembled using 8051 cross-assembler. Burn the generated hex code into the microcontroller using a suitable programmer. (You may refer to the programmer for 89C51/89C52/8902051 microcontrollers published in EFY's May 2004 issue.) The software is well commented and easy to understand. Set pin P3.5 to discharge the capacitor and clear pin P3.5 to charge the capacitor. Timer 0 starts as soon as the capacitor starts charging and thus reckons time in microseconds until pin 3.6 goes high.

EFY note. The source code and all the relevant files of this article have been included in this month's EFY-CD.

TEMP.ASM

```
$MOD51
ORG 0
RESET:   AJMP 30H
ORG 0BH ;TIMER 0 INTERRUPT VECTOR
         AJMP  220H  ;Timer  0  Interrupt
service routine address
ORG 30H
START: MOV SP,#60H ;set stack pointer
 MOV P3,#0FFH ;set all port 3 bits high
to enable inputs also
BEG: MOV P1,#03 ;set port 1 to all zeros
BUT D0,D1 HIGH FOR ANALOG INPUT
 MOV TMOD,#01100001B ;TIMER 1 - MODE 2
COUNTER,TIMR-0 TO MODE 1
 MOV  TH0,#0H  ;TIMER  REG.0  IS  SET  TO
0000, increments each 1 us (12mc XTL)
 MOV TL0,#0H ; timer low reg. is zero
 SETB P1.0
 SETB P1.1
 setb p3.5 ; short capacitor
 mov r6,#6h
DISCHARGE_WAIT:
         acall delay1
 DJNZ R6,DISCHARGE_WAIT
 clr p3.5 ; allow it to charge up
 SETB TR0 ;START TIMER 0
CHECK:   JNB P3.6,CHECK
         setb p3.5 ;discharge it
 MOV A,TL0
 MOV 40H,A
 MOV A,TH0
 MOV 41H,A ; MOVE TIMER COUNT TO MEMORY
 clr tr0 ;yes, stop timer
 MOV 30H,#14H
 C1: CALL DISP1 ;show the value
 DJNZ 30H,C1
 AJMP BEG
DISP1:
REFRESH: ;The content of 18 to 1B memory
locations are output on LEDs
 ;only  numbers  0  to  9  and  A  to  F  are
valid data in these locations
         mov r1,41h
 mov r2,40h
 CALL HEXTOBCD
 MOV 18H,r3 ; SHOWS centigrade as "C"
 MOV 19H,r4 ;least significant digit
 MOV 1AH,r5
 MOV 1BH,R6 ;most significant digit
 MOV R0,#18h ; 18,19,1A,1B, holds values
for 4 digits
 MOV R4,#8 ; pin p3.3 made low first
 mov dptr,#ledcode
 PQ2: CALL SEGDISP
 INC R0
 CLR C
 MOV A,R4
 RRC A
 MOV R4,A
 JNC PQ2
 PV3:    RET
SEGDISP:MOV A,@R0
 ANL A,#0FH
 k: MOVC A,@A+dptr
 MOV R5,A
 ORL A,#03H ;WE WANT TO USE PORT 1 BITS
0 AND 1 FOR INPUT ANLOG
 ; so retain them high
 S3: MOV P1,A ;SEGMENT_PORT
 S1: MOV A,R4 ;get digit code from r4
 CPL A ;but dont disturb pin 3.5,3.7
 anl a,#5fh
 MOV P3,A
 MOV A,R5 ;we use p3.7 for the segment
'a' of display
 RRC A ;so get that bit D1 into carry
 RRC A ; by moving right twice

 JNC  S44  ;  and  if  carry  is  there,  set
high pin P3.7
 SETB P3.7
 SJMP S5
S44: CLR P3.7 ;else clear it
S5:
S4: ACALL DELAY1 ; let it burn for some
time
 MOV A,#0fFH       ;extinguish the digit
after that time
 MOV P3,A ; to prevent shadow
 s6: RET
ledcode: ;these are code for the numbers
0 to 9 and A to F
         DB 7EH,0CH,0B6H,9EH,0CCH,0DAH,
0FAH
         DB 0EH,0FEH,0CEH,0EEH,0F8H,72H
,0BCH,0F6H,0E2H
DELAY1: MOV R1,#0ffH
 N: NOP
 NOP
         DJNZ R1,N
 RET
hextobcd: ;2 BYTE HEX TO 5 DIGITS BCD
;16 Bit Hex to BCD Conversion for 8051
Microcontroller
;This routine is for 16 bit Hex to BCD
conversion
;Accepts a 16 bit binary number in R1,R2
and returns 5 digit BCD in
;R7,R6,R5,R4,R3(upto 64K )
 Hex2BCD: ;r1=high byte,R2 = LSByte
         MOV R3,#00D
         MOV R4,#00D
         MOV R5,#00D
         MOV R6,#00D
         MOV R7,#00D
         MOV B,#10D
         MOV A,R2
         DIV AB
         MOV R3,B ;
         MOV B,#10 ; R7,R6,R5,R4,R3
         DIV AB
         MOV R4,B
         MOV R5,A
         CJNE R1,#0H,HIGH_BYTE ; CHECK
FOR HIGH BYTE
         SJMP ENDD
HIGH_BYTE:
         MOV A,#6
         ADD A,R3
         MOV B,#10
         DIV AB
         MOV R3,B
         ADD A,#5
         ADD A,R4
         MOV B,#10
         DIV AB
         MOV R4,B
         ADD A,#2
         ADD A,R5
         MOV B,#10
         DIV AB
         MOV R5,B
         CJNE R6,#00D,ADD_IT
         SJMP CONTINUE
ADD_IT: ADD A,R6
CONTINUE:
         MOV R6,A
         DJNZ R1,HIGH_BYTE
         MOV B, #10D
         MOV A,R6
         DIV AB
         MOV R6,B
         MOV R7,A
ENDD:    ret
         END
```

●