MICROCONTROLLER-BASED TACHOMETER

■ **PROF. K. PADMANABHAN**

A tachometer is nothing but a simple electronic digital transducer. Normally, it is used for measuring the speed of a rotating shaft. The number of revolutions per minute (rpm) is valuable information for understanding any rotational system. For example, there is an optimum speed for drilling a particular-size hole in a particular metal piece; there is an ideal sanding disk speed that depends on the material being finished. You may also want to measure the speed of fans you use.

This easy-to-make photoelectric tachometer measures the rpm of most shop-floor tools and many household machines without any mechanical or electrical interface.

How it works?

Just point the light-sensitive probe tip atop the spinning shaft towards the spinning blade, disk or chuck and read the rpm. The only requirement is that you first place a contrasting colour mask. A strip of white adhesive tape is ideal on the spinning object. Position it such that the intensity of light reflected from the object's surface changes as it rotates.

Each time the tape spins past the probe, the momentary increase in reflected light is detected by the phototransistor. The signal processor and microcontroller circuit counts the increase in the number of such light reflections sensed by it and thereby evaluates the rpm, which is displayed on the 4-digit, 7-segment display.

The phototransistor is kept inside a plastic tube, which has a convex lens fitted at one end. A convex lens of about 1cm diameter and 8-10cm focal length is a common item used by watch repairers and in cine film viewer toys. It can be obtained from them to set up the experiment. The phototransistor is fixed on a piece of cardboard such that it faces the lens at a distance of about 8 cm. The leads from the phototransistor are taken out

PARTS LIST

Semiconductors:

IC1	-	AT89C2051 microcontroller
IC2	-	ULN2003 current buffer
IC3	-	CA3140 operational amplifier
T1-T4	-	BC557 pnp transistors
T5	-	2N2222 npn transistor
T6	-	L14F1 photo-transistor
D1	-	1N4007 rectifier diode
DIS1	-	KLQ564 4-digit, 7-segment display

Resistors (all ¼-watt, ±5% carbon):

R1-R4	-	1-kilo-ohm
R5, R6	-	10-kilo-ohm
R7-R11	-	1.2-kilo-ohm
RNW1	-	10-kilo-ohm resistor network
VR1	-	4.7-kilo-ohm preset

Capacitors:

C1	-	10µF, 16V electrolytic
C2, C5, C6	-	0.1µF ceramic disk
C3, C4	-	22pF ceramic disk

Miscellaneous:

S1	-	Push-to-on switch
S2	-	On/Off Switch
X_{TAL}	-	12MHz
BATT.	-	6V battery

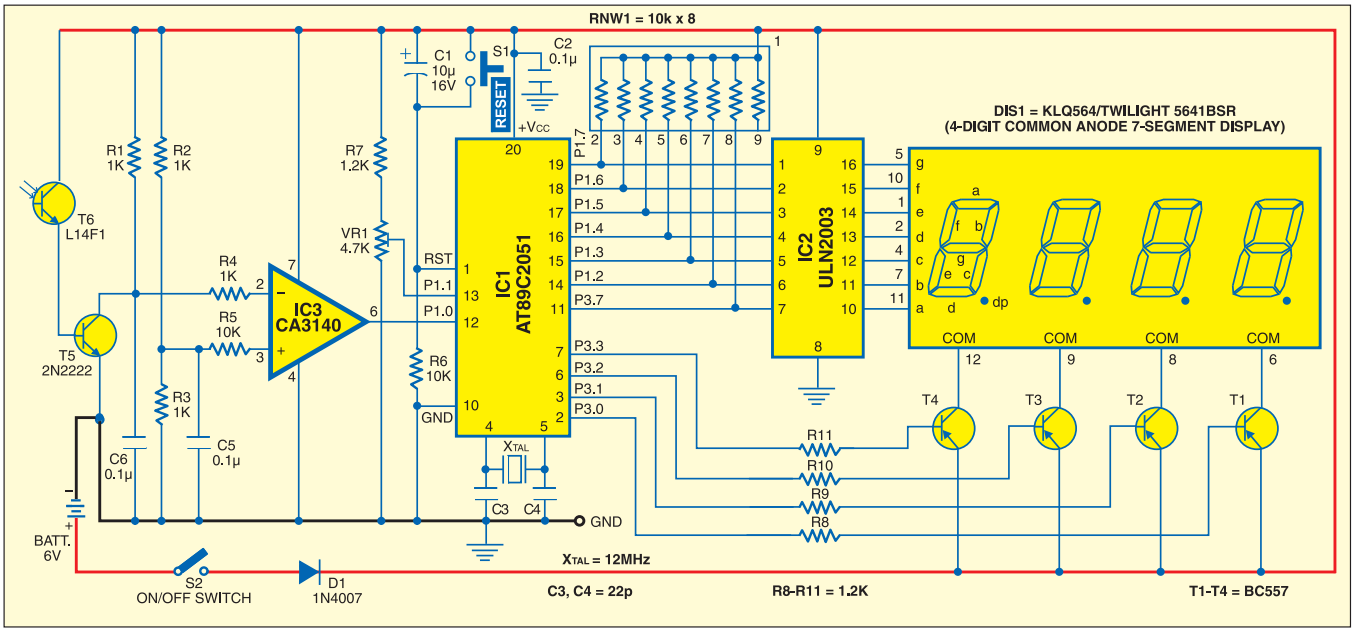

Fig. 1: Circuit of microcontroller-based tachometer

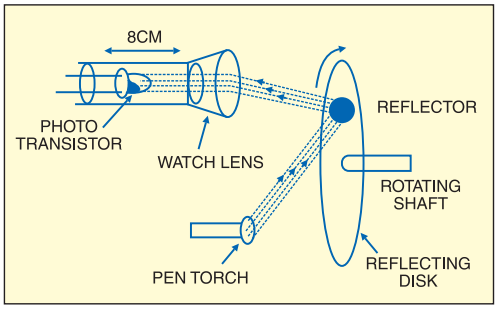

Fig. 2: Suitable arrangement of phototransistor

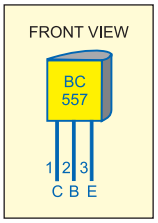

Fig. 3: Pin configuration of transistor BC557

and connected in the circuit shown in Fig. 1. Fig. 2 shows the suitable arrangement of phototransistor.

The detected signal is amplified by transistor 2N2222 (T5) and further amplified by operational amplifier CA3140 (IC3). The reference voltage point for the operational amplifier is obtained by resistor divider network comprising R2 and R3. The output from pin 6 of IC3 is fed to pin 12 of microcontroller AT89C2051. Note that pins 12 and 13 of microcontroller AT89C2051 are the inputs (+ and -) of its internal analogue comparator. Pin 13 is adjusted to nearly half the supply voltage using a potential divider comprising resistor R7 and preset VR1 across the supply.

The pulses picked up by the phototransistor are sensed by the internal comparator of AT89C2051 and, through software, each pulse representing one rotation of the object is detected. By counting the number of such pulses, on an average per minute basis, the RPM is evaluated. It is displayed by a software routine to light up the LED segments of the 4-digit, 7-segment display.

Circuit description

Fig. 1 shows the circuit of the microcontroller-based tachometer. The tachometer comprises AT89C2051 microcontroller, ULN2003 high-current Darlington transistor array, CA3140 operational amplifier, common-anode 7-segment (4-digit multiplexed) display and its four anode-driving transistors.

The AT89C2051 is a 20-pin, 8-bit microcontroller of Intel's 8051 family made by Atmel Corporation. Port-1 pins P1.7 through P1.2, and port-3 pin P3.7 are connected to input pins 1 through 7 of ULN2003. Port-1 pins are pulled up with 10-kilo-ohm resistor network RNW1. They drive all the seven segments of the display with the help of internal inverters.

Port-3 pins P3.0 through P3.3 of the microcontroller are connected to the base of transistors T1 through T4, respectively, to select one digit out of the four at a time and to supply anode-drive currents to the common anode pin of respective digit. Pin configuration of transistor BC557 is shown in Fig. 3.

When pin P3.0 of microcontroller IC1 goes low, it drives transistor T1 into saturation, which provides the drive current to anode pin 6 of 4-digit, 7-segment, common-anode display DIS1. Similarly, transistors T2 through T4, respectively, provide supply to common-anode pins 8, 9 and 12 of DIS1. Thus microcontroller IC1 drives the segment in multiplexed manner using its port pins. This is time-division multiplexing process.

Segment data and display-enable pulse for display are refreshed every 5 ms. Thus, the display appears to be continuous even though it lights up one by one.

Switch S1 is used to manually reset the microcontroller, while the power-on-reset signal for the microcontroller is given by C1 and R6. A 12MHz crystal is connected to pins 4 and 5 of IC1 to

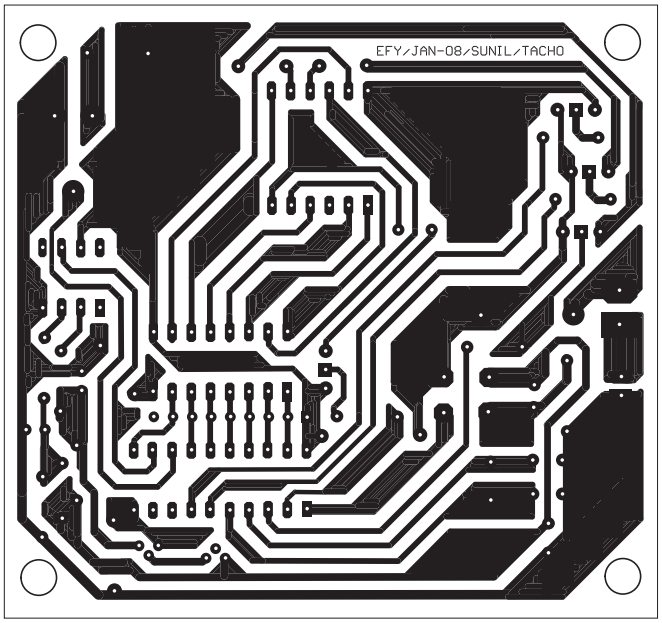

Fig. 4: A single-side, actual-size PCB layout for microcontroller-based tachometer

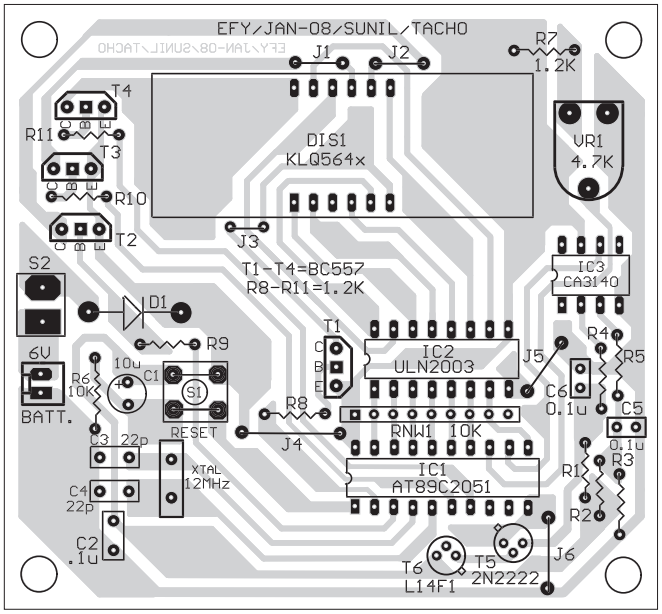

Fig. 5: Component layout for the PCB

generate the basic clock frequency for the microcontroller. The circuit uses a 6V battery for power supply or alternatively a mains derived low voltage supply. An actual-size, single-side PCB layout for the tachometer (Fig. 1) is shown in Fig. 4 and its component layout in Fig. 5.

Testing

The source code of this article is available at http://www.electronicsforu.com/efycodes/efy-codes.zip at code file 'tacho.hex.' Using a programmer, load the code into the new chip AT89C2051. (Refer the May 2005 issue of EFY for article on programmer for 89C51 and 2051.) Then, fit it into the circuit board and after powering up the circuit, test it.

For testing, point the probe using torchlight for illumination of the rotating object. For fans, use the light from behind. Hold the probe firmly so as to provide a steady, bright illumination on the object. Even an LED pen torch could be used here. Avoid the fluctuating background light from sources such as tubelight.

Software

The software is written in Assembly language and assembled using 8051 cross-assembler. It is well commented and easy to understand. It uses AT89C2051's internal timer for measuring the period of one cycle of the rotation in units of 100 microseconds. Thus if the speed is 1500 rpm, it is 25 rps, and the time taken for one cycle is 40 ms.

The timer uses an interrupt to count overflows every 100 microseconds and so the number counted by the timer program in this case will be '400.' This is divided by '600,000' (so many 100/µs present in a minute), giving a result of '1500.' This gives the rpm. These digits are displayed on the 4-digit, 7-segment display. To perform the division, subroutine UDIV32 is employed, which is a standard subroutine available for 8051 family for 32-bit number by 16-bit number division. It has an accuracy of 5 rpm in a 6000rpm count.

EFY note. The source code of this article is available at http://www.electronicsforu.com/efycodes/efy-codes.zip and will also be included in EFY-CD of February 2008 issue. ●

Prof. K. Padmanabhan retired from Alagappa College of Technology, Guindy, Chennai

TACHO.ASM

```
$mod51
ORG 0H
 AJMP 30H
ORG 0BH ;TIMER 0 INTERRUPT VECTOR
      AJMP TIMER0ISR ;Timer 0 Interrupt service routine address
ORG 30H
 MOV SP,#60H ;set stack pointer
 MOV P3,#0FFH ;set all port 3 bits high to enable inputs also
 MOV P1,#03 ;set port 1 to all zeros expect bits 0,1
  MOV TMOD,#01100001B ;TIMER 1 - MODE 2 COUNTER,TIMR-0 TO 16 bit timer
BEG: MOV TH0,#0ffH ;TIMER REG.0 IS SET TO 0, GIVES 64ms
 MOV TL0,#-99 ; timer low reg. is also so
 setb et0
 setb ea
 mov 44h,#0
 mov 45h,#0
 acall delay
 ajmp lowsig
delay: mov r2,#10
 djnz r2,$ ;wait 20 us
 ret
lowsig: jb p3.6,lowsig
 call delay
 jnb p3.6,$
 setb tr0 ; start timer
 mov c,p3.6 ;high begins
 mov p3.5,c
 acall delay
 jb p3.6, $
 mov c,p3.6 ;low now
 mov p3.5,c
 acall delay
 jnb p3.6,$
 mov c,p3.6 ;high begins again
 mov p3.5,c
 clr tr0 ;stop timer
 clr et0 ;and interrupt by timer
 mov r3,#0 ;number 600000 or 927c0 hex as Dividend
 mov r2,#09h ; 9
 mov r1,#27h ;27
 mov r0,#0c0h ; c0
 mov r5,45h ;divisor is time for one cycle
 mov r4,44h
 call UDIV32 ;divide 60000/t
 mov 40h,r0
 mov 41h,r1
 mov r1,41h
 mov r2,40h
 CALL HEX2BCD
      mov 50h,#0FFH
      call refresh
disp: call refresh1
 djnz 50h,disp ; so many times for a visible time limit
 jmp beg
;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, r7 most significant digit, 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
DISP1:
REFRESH:; 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 18H,r3 ; least significant digit
 MOV 19H,r4 ; next significant digit
 MOV 1AH,r5
 MOV 1BH,R6 ; most significant digit (max:9999)
      RET
refresh1:
      MOV R0,#1bh ; 1b,1a,19,18, holds values for 4 digits
 MOV R4,#8 ; pin p3.3_ 0 made low one by one starts wth 18
  mov r7,#2 ; decimal pt.on 3rd digit from left (2 nd fromright)
PQ2: CALL SEGDISP
 dec R0
 mov a,r4
 rrc a
 mov r4,a
 jnc pQ2
 PV3:RET
SEGDISP:mov dptr,#ledcode
 MOV A,@R0
 ANL A,#0FH
 MOVC A,@A+dptr
segcode: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
 MOV A,R5 ;we use p3.7 for the segment 'a' of display
 RRC A ;so get that bit D0into carry
 rrc a
 mov p3.7,c ;segment 'a;
 S1: MOV A,R4 ; get digit code from r4 00001000
 cpl a ;11110111
 rrc a ;11111011-1
 mov p3.0,c ; output to drive transsitors for digit lighting
 rrc a ;11111101-1
 mov p3.1,c
 rrc a ;11111110-1
 mov p3.2,c
 rrc a ;1111111-0 yes low makes left most digit show msdigit
```

```
 mov p3.3,c
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:DB 7EH,0CH,0B6H,9EH,0CCH,0DAH,0FAH
      ;these are code for the numbers 0 to 9 and A to F
      DB 0EH,0FEH,0CEH,0EEH,0F8H,72H,0BCH,0F6H,
0E2H
 DELAY1:MOV 55h,#0ffH ; 1ms
 N: NOP
 DJNZ 55h,N
 RET
TIMER0ISR:mov th0,#0ffh
 mov tl0,#-90 ; in 100 us steps
 push acc
 mov a,#1
 clr c
 add a, 44h ;count time btwn pulses
 mov 44h,a
 mov a,#0
 addc a,45h ;add carry to most sign. byte
 mov 45h,a
 pop acc
 reti
; subroutine UDIV32
;32 bit /16 bit to 32 bit quotient and remainder un-
signed
;input r3,r2,r1,r0 = dividend X
;input r5,r4 = divisor y
;output r3-r0 = quotient Q of X/Y
;r7,r6,r5,r4 =remainder
;alters acc, flags
UDIV32: push 08 ;save reg. bank 1
      push 09
 push 0AH
 push 0BH
 push 0CH
 push 0DH
 push 0EH
 push 0Fh
 push dpl
 push dph
 push B
 setb RS0 ;select reg.bank 1
 mov r7,#0
 mov r6,#0
 mov r5,#0
 mov r4,#0
 mov B,#32 ;set loop count
div_lp32:clr RS0 ;selet reg.bank 0
 clr C
 mov a,r0 ;shift highestbit of X
 rlc a
 mov r0,a
 mov a,r1 ;shift next bit of X
 rlc a
 mov r1,a
 mov a,r2 ;shift next bit of X
 rlc a
 mov r2,a
 mov a,r3 ;shift next bit of X
 rlc a
 mov r3,a
 setb rs0 ;reg. bank 1
 mov a,r4 ;lowest bit of remainder
 rlc a
 mov r4,a
 mov a,r5 ;shift next bit of rem
 rlc a
 mov r5,a
 mov a,r6 ;shift next bit of rem
 rlc a
 mov r6,a
 mov a,r7 ;shift next bit of rem
 rlc a
 mov r7,a
 mov a,r4
 clr C
 subb a,04
 mov dpl,a
 mov a,r5
 subb a,5
 mov dph,a
 mov a, r6
 subb a,#0
 mov 06,a
 mov a,r7
 subb a,#0
 mov 07,a
 cpl C
 jnc div_321
 mov r7,7
 mov r6,6
 mov r5,dph
 mov r4,dpl
div_321: mov a,r0
 rlc a
 mov r0,a ; shift result bit into partial quotient
 mov a,r1
 rlc a
 mov r1,a
 mov a,r2
 rlc a
 mov r2,a
 mov a,r3
 rlc a
 mov r3,a
 djnz B,div_lp32
 mov 7,r7
 mov 6,r6
 mov 5,r5
 mov 4,r4
 mov 3,r3
 mov 2,r2
 mov 1,r1
 mov 0,r0
 clr rs0
 pop B
 pop dph
 pop dpl
 pop 0Fh
 pop 0EH
 pop 0Dh
 pop 0Ch
 pop 0bh
 pop 0ah
 pop 09
 pop 08
 ret
END
```