MICROCONTROLLER-BASED DC MOTOR SPEED CONTROLLER

■ AKSHAY MATHUR

DC motor speed controllers are very useful for controlling the motion of robotic and industrial automation systems. The controller presented here uses the pulse-width modulation (PWM) technique. The PWM wave for speed control is generated using Atmel AT89C52 microcontroller.

To control the speed of the DC motor, you need a variable-voltage DC power source. When the DC motor is switched on, it takes certain time to reach the full speed. As soon as the power supply is switched on, the DC motor starts gaining speed and if you switch off the power supply before it reaches the maximum rated speed, it starts to slow down.

In case switching on and switching off are done in quick succession, the motor rotates at a slower speed between zero and full rated speed. This is what a PWM technique based controller does: it switches the motor 'on' and 'off' with a pulse train. To control the motor speed, it varies (modulates) the width of the pulses—hence the pulse-width modulation. When the motor is 'on' for a short period and 'off' for a long one, it will rotate slowly. When the motor is 'on' for most of the time and 'off' only for a short while, it will rotate at higher speed, say, nearly at full (maximum) rated speed.

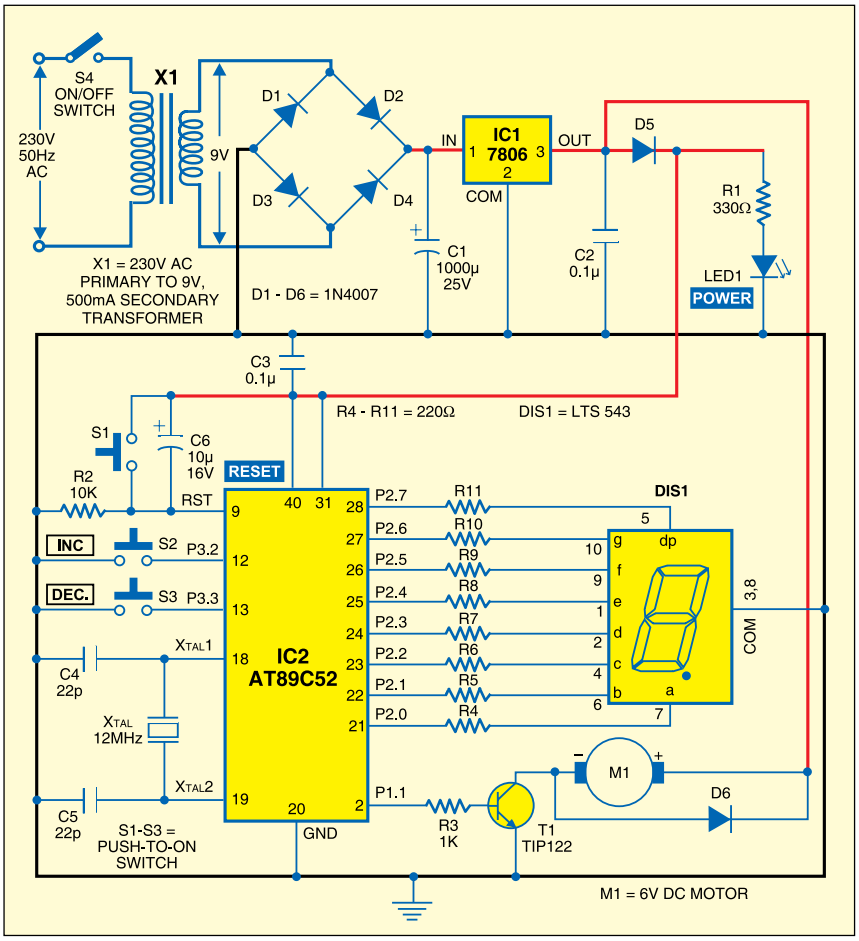

Fig. 1: Circuit of microcontroller-based DC motor speed controller

Circuit description

Fig. 1 shows the circuit of the DC motor speed controller. 230V AC mains is stepped down by transformer X1 to deliver secondary output of 9V, 500 mA. The secondary output is rectified by a full-wave bridge rectifier comprising diodes D1 through D4, filtered by capacitor C1 and regulated by IC 7806

PARTS LIST

Semiconductor:

IC1	- 7806, 6V regulator
IC2	- AT89C52 microcontroller
T1	- TIP122 npn transistor
D1-D6	- 1N4007 rectifier diode
LED1	- 5mm light-emitting diode
DIS1	- LTS543 common-cathode 7-segment display

Resistors (all ¼-watt, ±5% carbon):

R1	- 330-ohm
R2	- 10-kilo-ohm
R3	- 1-kilo-ohm
R4-R11	- 220-ohm

Capacitors:

C1	- 1000µF, 25V electrolytic
C2, C3	- 0.1µF ceramic disk
C4, C5	- 22pF ceramic disk
C6	- 10µF, 16V electrolytic

Miscellaneous:

X1	- 230V AC primary to 9V, 500mA secondary transformer
S1-S3	- Push-to-on switch
S4	- On/off switch
X_{TAL}	- 12MHz crystal
M1	- 6V DC motor
CON1	- Connector for power supply

Signet Electronics Pvt Ltd is
NO longer marketing our products

SIGNET
Measurable Quality
Test Instruments Manufacturer
Tel: 24011079, 32523577
Mob:9322509190 Fax:24070266
Contact : M Loynmoon signet_instruments@yahoo.com
201/202 Champaklal Ind Est.
105,Sion(E) Mumbai 400022.
Website : www.signet-instruments.com

Colour Pattern Generators
S-1044 VHF/IF
S-1053
S-9144 Synthesized 30-860 MHz Rhombus Patt
50-1300 MHz S-99 Antenna Trainer
IIT DESIGNED
50-860 MHz S-54 Antenna Trainer
800MHz S-45 Antenna Trainer
5-850 MHz S-5022 Spectrum/Network Analyser
RF Lab EXPERIMENTS: Tuned Amp,RF Osc,
RF Xtal Osc,IF Amp,RF Mixer,RF Filter
50-1300 MHz S-18 Slotted Transmission Line
Trainer
850-1300 MHz S-36
Microstrip Trainer
IIT Mumbai Design

S-8955 Noise Generator 5-1300 MHz 70dBuV
S-5022A 1.3GHz Spectrum/Network Analyzer + Software
S-4068 465MHz TV Sweep Generator+Marker (Wobbulator)
S-909 1 Hz - 110 MHz Signal Generator (AM/FM SG+PS+FC)
S-4050 Service Test Station (AM/FM SG+PS+FG+FC)
S-990 300MHz AM/FM(Stereo) Sig Gen+Sweep+Marker
S-945 300 MHz AM/FM(Stereo) Synthesized Signal Gen
S-1611 Frequency Counter 1.3 GHz
Enquire regarding other available Instruments
Servicing of all SIGNET instruments available.
DEALERS WELCOME

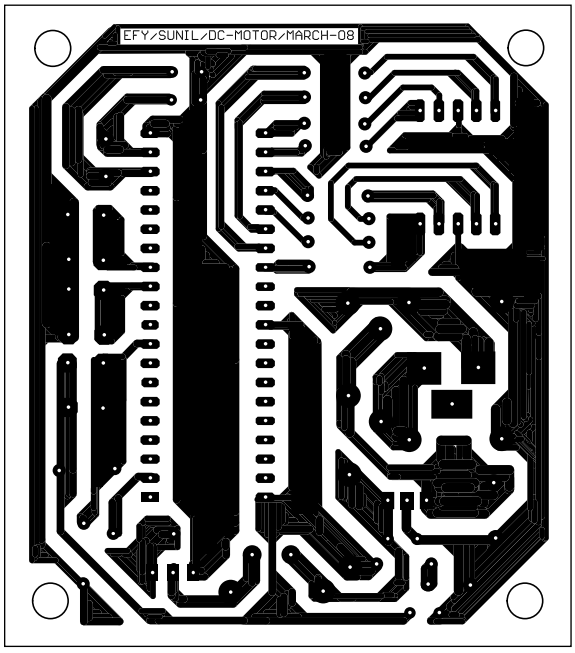

Fig. 2: A single-side, actual-size PCB layout for microcontroller-based DC motor speed controller

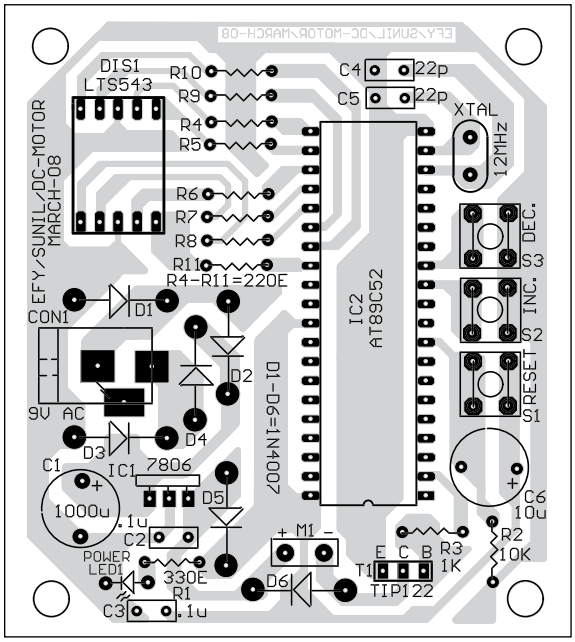

Fig. 3: Component layout for the PCB

(IC1). Capacitor C2 bypasses any ripple present in the regulated output. LED1 acts as the power-'on' indicator. Resistor R1 limits the current passing through LED1. Diode D5 causes a voltage drop of 0.6V and, as a result, the final output of the circuit is approximately 5.4V.

IC AT89C52 (IC2) is a low-power, high-performance, 8-bit microcontroller with 8 kB of Flash programmable and erasable read-only memory (PEROM), 256 bytes of RAM, 32 input/output (I/O) lines, three 16-bit timers/counters, a six-vector two-level interrupt architecture, a full-duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C52 is designed with static logic for operation down to zero frequency and supports two software-selectable power-saving modes. The idle mode stops the CPU while allowing the RAM, timers/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next hardware reset is activated.

At the heart of the speed controller system is microcontroller AT89C52 (IC2), which creates (using timer 0) pulses of varying width for pulse-width modulation and controls the motor speed. To change the speed of the motor, switches S2 and S3 are interfaced to interrupt the input to pins P3.2 and P3.3 of IC2, respectively.

Whenever any of switches S2 and S3 is pressed, an interrupt is generated, which changes the duty cycle of the pulse train. Switch S2 interfaced to Interrupt-0 increases the duty cycle of the pulse waveform, whereas switch S3 interfaced to Interrupt-1 decreases the duty cycle of the pulse waveform. Power-on reset for the microcontroller is achieved through capacitor C6 and resistor R2. Switch S1 provides manual reset to the microcon-

troller. A 12MHz crystal (X_{TAL}) is used for basic clock frequency.

Port 2 is an 8-bit, bidirectional, input/output (I/O) port with internal pull-ups. Port-2 output buffers can sink/source four TTL inputs. The duty cycle of the pulse waveform is displayed on common-cathode 7-segment display LTS543 (DIS1). Segments 'a' through 'g' and decimal 'dp' are connected to port pins P2.0 through P2.6 and P2.7 via current-limiting resistors R4 through R11, respectively.

DIS1 displays one-tenth value of the duty cycle. If duty cycle is 50 per cent, the value displayed on DIS1 is 5. If duty cycle is 100 per cent, the value displayed on DIS1 is decimal point '•' only (as it is only a single-digit display system and '10' is displayed as '•'). The software is written such that the duty cycle for PWM is increased in discrete intervals of '10'. Hence the speed of the DC motor is divided into eleven steps from 0 to 10.

Port pin P1.1 is internally pulled up. It is used as the output to control the motor with driver transistor T1. Whenever timer-0 overflows, the status of pin P1.1 is complemented and hence a square wave with appropriate duty cycle is generated. This pin is interfaced to power transistor TIP122 (T1), which is used to drive the motor.

When the transistor is driven into saturation, current flows through the motor. When the transistor is cut off, the motor current keeps flowing because of the motor's inductance. Diode D6 connected across the motor coil prevents reverse current flow. A heat-sink is used with power transistor T1.

An actual-size, single-side PCB for the DC motor speed controller is shown in Fig. 2 and its component layout in Fig. 3.

Software

The software is written in 'C' language and compiled using Keil C compiler, which generates Intel hex code for the microcontroller. The µVision3 integrates all tools including the 'C' compiler, micro assembler, linker/locator and hex file generator. The generated hex code is burnt into the micrcontroller using a suitable programmer.

Whenever any switch is pressed, the duty cycle of PWM varies. The software then calculates the appropriate values for TH0 and TL0 for 'on' and 'off' time of the output, which are copied in TH0 and TL0 on timer interrupts.

In this circuit, we have used timer-0 of the microcontroller for generating PWM pulses, which is clocked using a 12MHz crystal oscillator. The base frequency is kept constant at 1 kHz and the duty cycle of this wave is varied to change the analogue level at output pin P1.1 of the microcontroller.

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

MOTOR.C

```
#include <AT89x52.h>
int i=0; int t_on=0, t_off=100;
int  k[]={63,06,91,79,102,109,125,07,12
7,111,128};
bit t_val=0;
unsigned int count1, count2, TH0_on,
TH0_off, TL0_on, TL0_off;

void increase (void) interrupt 0
{
t_on = t_on + 10;
if(t_on>100) {t_on = 100;}
t_off = 100 - t_on;
count1 = t_on * 10;
count2 = t_off * 10;
TH0_on = (65535-count1)/256;
TL0_on = (65535-count1)%256;
TH0_off = (65535-count2)/256;
TL0_off = (65535-count2)%256;
}
void pwm (void) interrupt 1
{
TR0 = 0;
P1_1 = ~P1_1;
if(t_val==1)
{TH0 = TH0_off; TL0 = TL0_off;}
else
{TH0 = TH0_on; TL0 = TL0_on;}
t_val = ~t_val;
TR0 = 1;
}
void decrease (void) interrupt 2
{
t_on = t_on - 10;
if(t_on<0) {t_on = 0;}
t_off = 100 - t_on;
count1 = t_on * 10;
count2 = t_off * 10;
TH0_on = (65535-count1)/256;
TL0_on = (65535-count1)%256;
TH0_off = (65535-count2)/256;
TL0_off = (65535-count2)%256;
}
void main()
{
P1=1;
IE = 135;
IP=5;
TCON = 21;
TMOD = 1;
count1 = t_on * 10;
count2 = t_off * 10;
TH0_on = (65535-count1)/256;
TL0_on = (65535-count1)%256;
TH0_off = (65535-count2)/256;
TL0_off = (65535-count2)%256;
TH0 = TH0_on;
TL0 = TL0_on;
TR0=1;
while(1)
{
P2 = k[t_on/10];
}
}
```