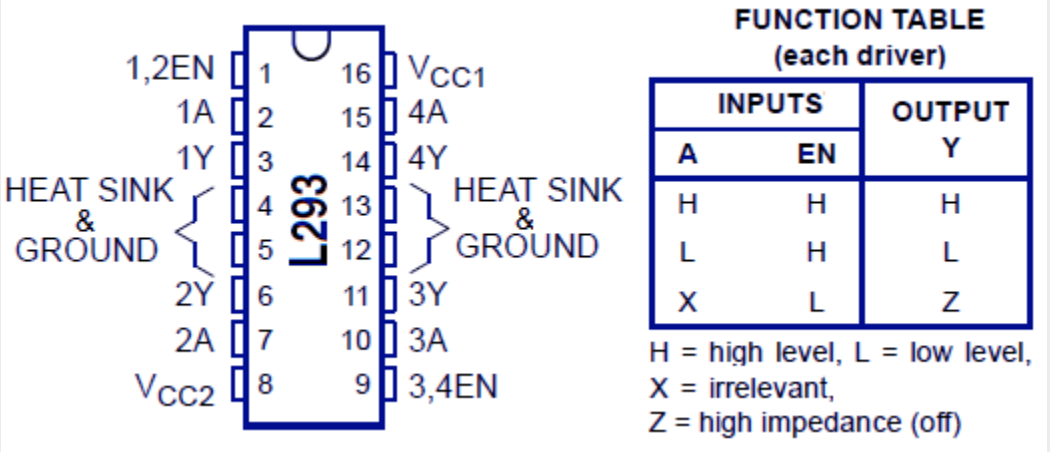
INTERFACING DC MOTOR TO 8051

This article shows how to interface a DC motor to an 8051 microcontroller. Interfacing DC motor to 8051 forms an essential part in designing embedded robotic projects. A well designed 8051-DC motor system has essentially two parts. Firstly an 8051 with the required software to control the motor and secondly a suitable driver circuit. Most of the DC motors have power requirements well out of the reach of a microcontroller and more over the voltage spikes produced while reversing the direction of rotation could easily damage the microcontroller. So it is not wise to connect a DC motor directly to the microcontroller. The perfect solution is to use a motor driver circuit in between the microcontroller and the DC motor.

L293 motor driver.

L293 is a dedicated quadruple half H bridge motor driver IC available in 16 pin package. To know more about H bridge, check this link. H bridge motor driver circuit. L293 has a current capacity of 600mA/channel and has supply voltage range from 4.5 to 36V DC. They are fitted with internal high speed clamp diodes for inductive spike protection. Other good features of L293 are high noise immunity, internal ESD protection, thermal shutdown, separate input supply for each channel etc. The pinout and truth table of an L293 motor driver is shown in the figure below.



FUNCTION TABLE (each driver)

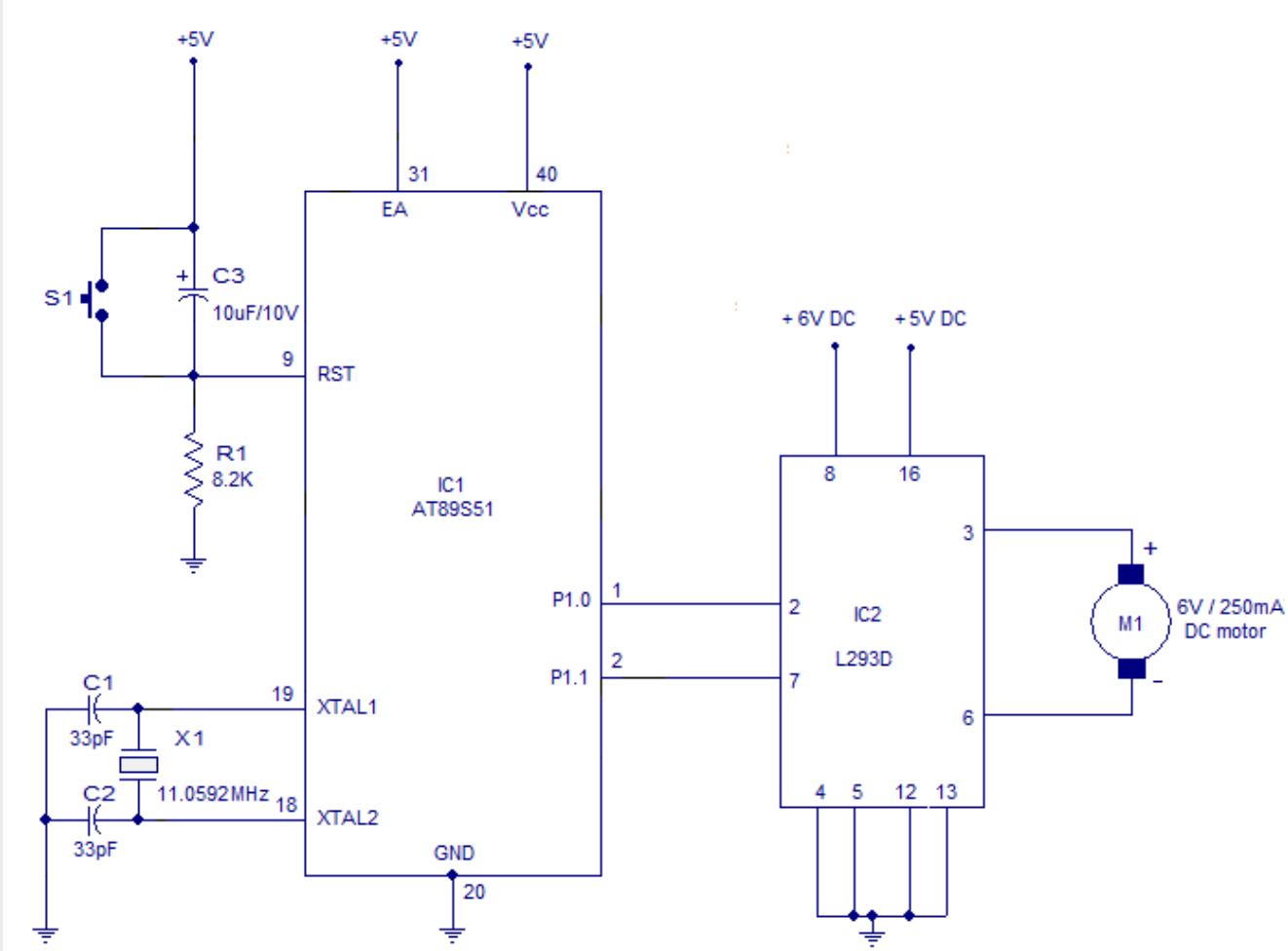
INPUTS		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level,
X = irrelevant,
Z = high impedance (off)

L293 pinout and function diagram

Bi directional DC motor using 8051.

This project describes a bidirectional DC motor that changes its direction automatically after a preset amount of time (around 1S). AT89S51 is the microcontroller used here and L293 forms the motor driver. Circuit diagram is shown below.



Bi directional DC motor using 8051

In the circuit components R1, S1 and C3 forms a debouncing reset circuitry. C1, C2 and X1 are related to the oscillator. Port pins P1.0 and P1.1 are connected to the corresponding input pins of the L293 motor driver. The motor is connected across output pins 3 and 6 of the L293. The software is so written that the logic combinations of P1.0 and P1.1 controls the direction of the motor. Initially when power is switched ON, P1.0 will be high and P1.1 will be low. This condition is maintained for a preset amount of time (around 1S) and for this time the motor will be running in the clockwise direction (refer the function table of L293). Then the logic of P1.0 and P1.1 are swapped and this

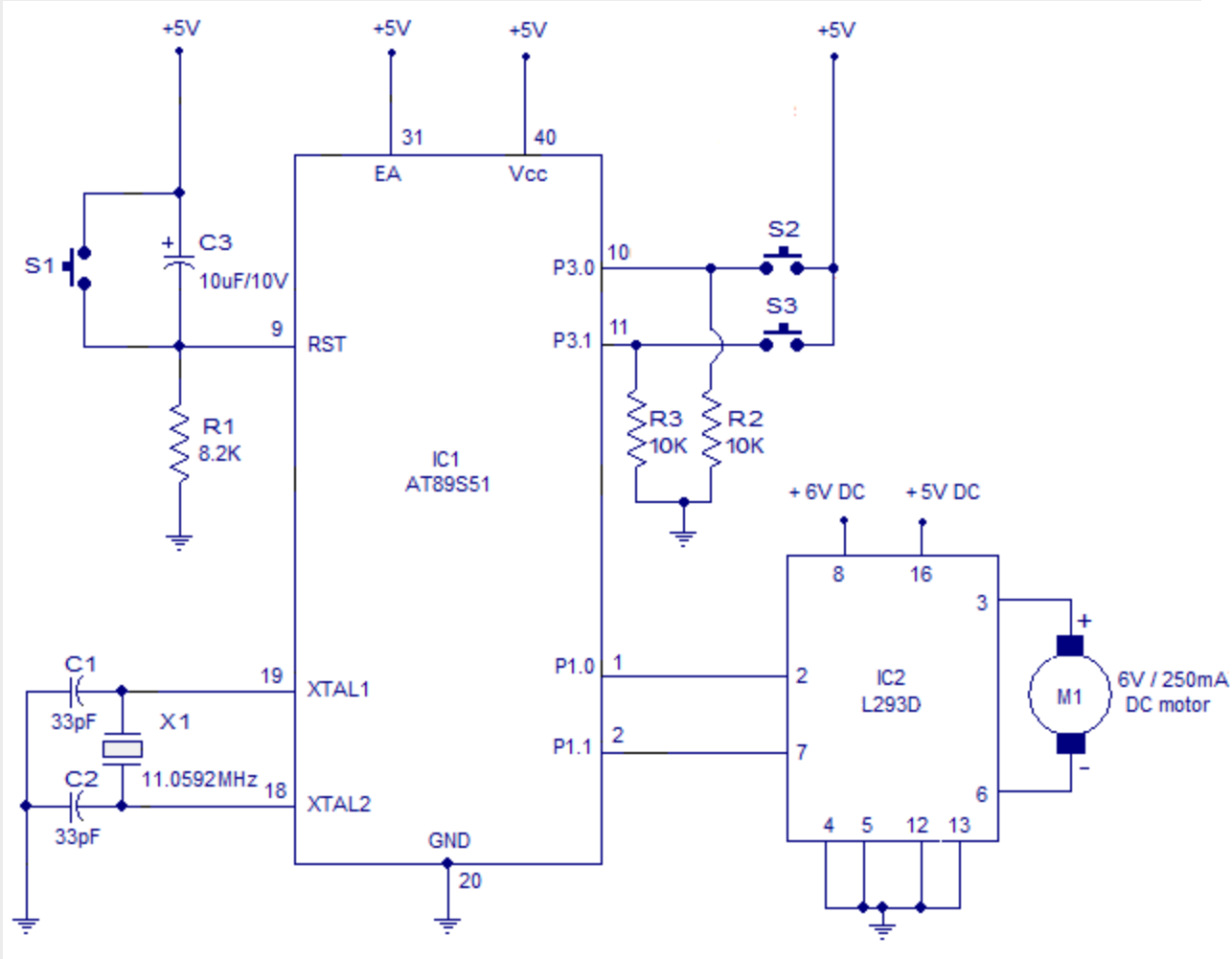
condition is also maintained for the same duration . This makes the motor to run in the anti clockwise direction for the same duration and the entire cycle is repeated.

Program.

```
ORG 00H // initial starting address
MAIN: MOV P1,#00000001B // motor runs clockwise
ACALL DELAY // calls the 1S DELAY
MOV P1,#00000010B // motor runs anti clockwise
ACALL DELAY // calls the 1S DELAY
SJMP MAIN // jumps to label MAIN for repaeting the cycle
DELAY: MOV R4,#0FH
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
DJNZ R3,WAIT2
DJNZ R4,WAIT1
RET
END
```

Bidirectional motor with pushbutton control.

The circuit shown below is of an 8051 based bi directional motor whose direction can be controlled using 2 push button switches. The circuit is very similar to the previous one except for the two push button switches . These pushbutton switches are interfaced to P0rt 3 of the microcontroller. Resistors R2 and R3 are the pull down resistors for P3.0 and 3.1 respectively.



Push button controlled bi-directional DC motor

The code for the above project is so written that initially when power is switched ON, the motor remains OFF. When push button switch S2 is pressed P1.0 goes high and P1.1 remains low. The motor runs in the clockwise direction and this condition is maintained until S3 is pressed. When push button switch S3 is pressed the logic of P1.0 and P1.1 toggles making the motor to run in the opposite direction and this condition is maintained until the next press of S2.

Program.

```
ORG 00H // initiall starting address
MOV P3,#00000000B // initiates P3 as the pushbutton interface
MOV P1,#00000000B // clears P1 for keeping the motor OFF initially
```

```
MAIN:MOV A,P3 // moves the current state of P3 to Accumulator
CJNE A,#00000001B,LABEL1 // checks whether S2 is pressed
MOV P1,#00000001B // makes the motor run clockwise
LABEL1:CJNE A,#00000010B,LABEL2 // checks whether S3 is pressed
MOV P1,#00000010B // makes the motor to run anti clockwise
LABEL2:SJMP MAIN // jumps back to the MAIN loop
END
```

About the program.

Checking whether a particular push button is pressed is done using the CJNE (compare and jump if not equal) instruction. In simple words the CJNE instuction compares two operands and jump to a predefined LABEL if the operands are not equal. If the two operands are equal nothing happens and the next instruction is executed. Whenever push button S2 is pressed the status of P3 will be 00000001B .This status is moved to accumulator A and compared with 00000001B using the CJNE instruction. Both operands are equal means S2 is pressed and the next instruction (MOV P1,#00000001B) which makes the motor run clockwise is executed. If the operands are not equal that means the S2 is not pressed and the controller jumps to LABEL1 which is to check the S3. To check S3, status of P3 is moved to A again and it is compared with 00000010B using the CJNE instruction. Both operands are equal means the S3 is pressed and the next instruction (MOV P1,#00000010B) which makes the motor run anti clockwise is executed. Both operands are not equal means S3 is not pressed and the controller goes to check S2 again and this cycle is repeated.

Source : http://www.circuitstoday.com/interfacing-dc-motor-to-8051