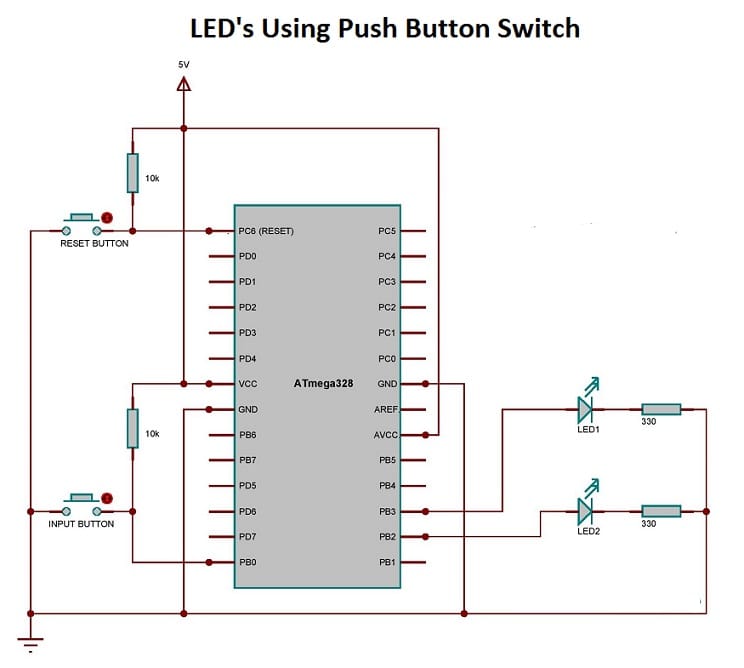
**Project : Control Two Led Using Push Button Switch**

In this project, we are going learn how to control the working of two LED’s using a push button switch, first we will connect the 2 LEDs with PB2 and PB3 of  PORTB of the microcontroller. A push button switch is then attached to PB0 pin and pulled-up using a 10K resistor. The remaining terminal of the switch is grounded. The function of a pull-up resistor is to insure that while leaving the switch as not pressed, the status of the PB0 pin should remain high. There are 20K pull-up resistors built into the ATmega chip that can be accessed from software also. But here  we are using an external pull-up circuit. When the switch is pressed, the two LED’s will glow and will turn off while we release the switch. This is how the circuit will work.

****

***Toggle 2 LED’s Using Push Button Switch – Circuit Diagram***

We have already discussed about the pre-processors and libraries at the earlier section. The only thing that is newly included in this code is the assigning of PB0 pin as input port and the “if” loop associated with that pin.

“DDRB &= ~(1 << DDB0)” is the code used for assigning PB0 as input port. An “if” loop  is included in the “while” loop, which will continuously monitor the status of the PB0 pin and alter the status of the two LED’s accordingly.

/\*

\* pushButton LED.c

\*

\*

\*/

/\*\*@file control\_led.c

@mainpage Control Two LED�s using a Push button switch

@bug No bug found

@author Prafull Varshney

@brief Here we are going learn how to control the working of two LED�s using a push button switch.

@param No parameters are given

@return No return is expected

\*/

#define F\_CPU 1000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB |= (1 << DDB2)|(1<<DDB3);

DDRB &= ~(1 << DDB0);

while (1)

{

if (!(PINB&(1<<PINB0)))

{

PORTB |= (1<<PINB2)|(1<<PINB3);

}

else

{

PORTB &= (~(1<<PINB2))&(~(1<<PINB3));

}

}

}