CHIPKIT UNO 32: DEBOUNCE

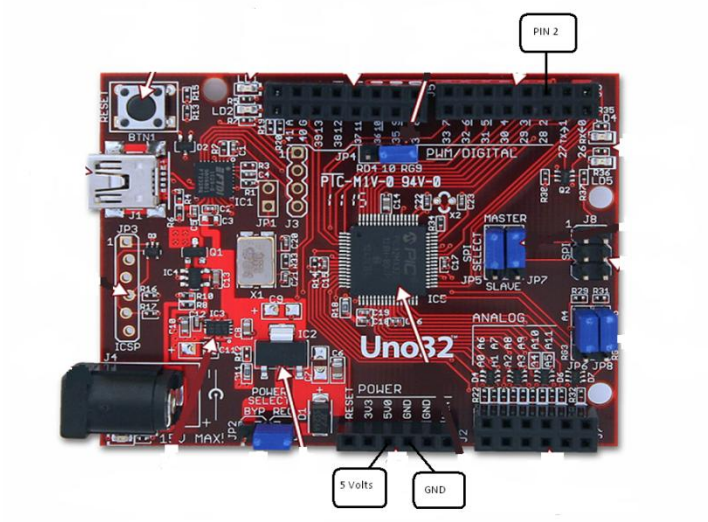
Author: Umashankar Shetty C

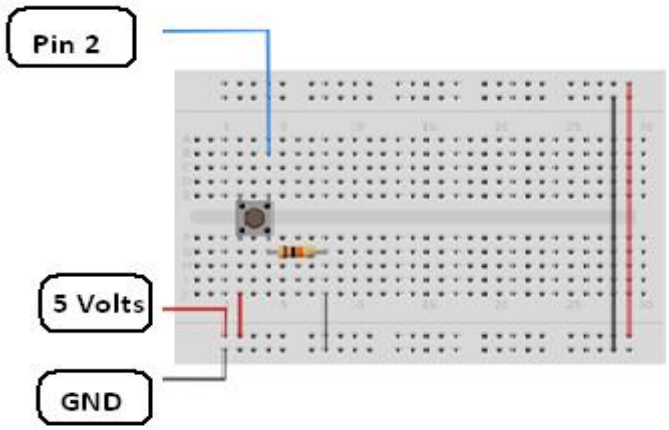
This example demonstrates the use of a pushbutton as a switch: each time you press the button, the LED (or whatever) is turned on (if it's off) or off (if on). It also **debounces** the input, which means checking twice in a short period of time to make sure it's definitely pressed. Without debouncing, pressing the button once can appear to the code as multiple presses. Makes use of the millis() function to keep track of the time when the button is pressed.

Hardware Required:

- Chipkit UNO32 Board
- momentary button or switch
- 10K ohm resistor
- breadboard
- hook-up wire

Hardware Connection:





Code:

The code below is based on Limor Fried's version of debounce, but the logic is inverted from her example. In her example, the switch returns LOW when closed, and HIGH when open. Here, the switch returns HIGH when pressed and LOW when not pressed.

```
/*
 Debounce

 Each time the input pin goes from LOW to HIGH (e.g. because of a push-button  press), the
output pin is toggled from LOW to HIGH or HIGH to LOW.  There's  a minimum delay between
toggles to debounce the circuit (i.e. to ignore noise).

 The circuit:
 * LED attached from pin 13 to ground
 * pushbutton attached from pin 2 to +5V
 * 10K resistor attached from pin 2 to ground

 * Note: On most Chipkit UNO32 boards, there is already an LED on the board connected to pin
13, so you don't need any extra components for this example.
  */

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// Variables will change:
int ledState = HIGH;        // the current state of the output pin
int buttonState;            // the current reading from the input pin
```

```
int lastButtonState = LOW;   // the previous reading from the input pin

// the following variables are long's because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long lastDebounceTime = 0;  // the last time the output pin was toggled
long debounceDelay = 50;    // the debounce time; increase if the output flickers

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH),  and you've waited
  // long enough since the last press to ignore any noise:

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current state:
    buttonState = reading;
  }

  // set the LED using the state of the button:
  digitalWrite(ledPin, buttonState);

  // save the reading.  Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}
```