CHIPKIT UNO 32: BUTTON STATE CHANGE DETECTION

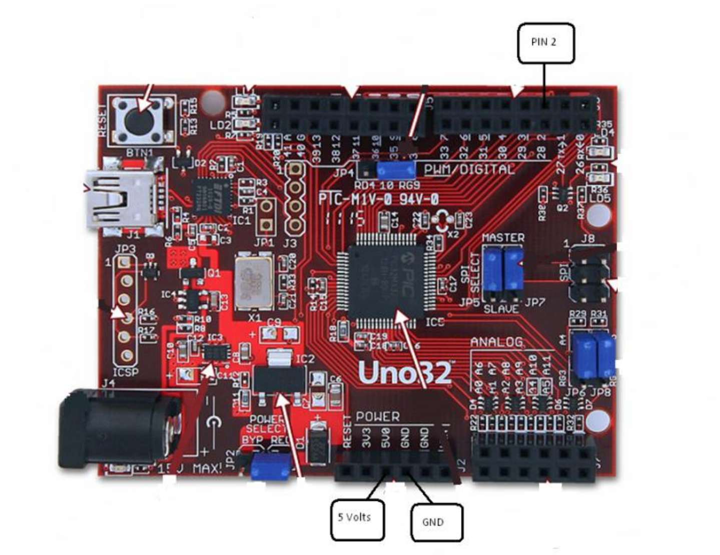
Author: Umashankar Shetty C

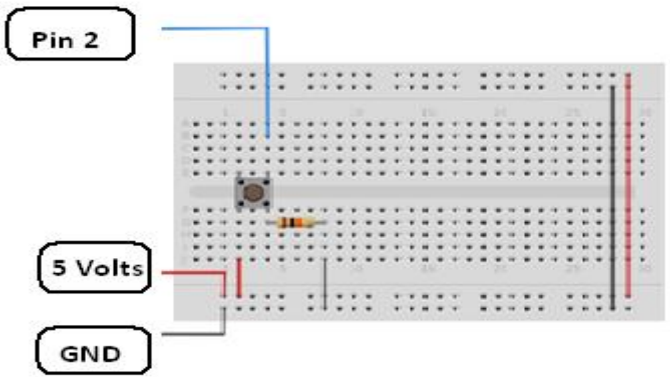
Once you've got a pushbutton working, you often want to do some action based on how many times the button is pushed. To do this, you need to know when the button changes state from off to on, and count how many times this change of state happens. This is called **state change detection** or **edge detection.**

Hardware Required:

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

Hardware Connection:





Connect three wires to the Chipkit UNO32 board. The first goes from one leg of the pushbutton through a pull-down resistor (here 10K Ohms) to ground. The second goes from the corresponding leg of the pushbutton to the 5 volt supply. The third connects to a digital i/o pin (here pin 2) which reads the button's state.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to voltage, so that we read a HIGH. (The pin is still connected to ground, but the resistor resists the flow of current, so the path of least resistance is to +5V.)

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, not connected to either voltage or ground. It will more or less randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

The sketch below continually reads the button's state. It then compares the button's state to its state the last time through the main loop. If the current button state is different from the last button state and the current button state is high, then the button changed from off to on. The sketch then increments a button push counter.

The sketch also checks the button push counter's value, and if it's an even multiple of four, it turns the LED on pin 13 ON. Otherwise, it turns it off.

Code:

```
/*
  State change detection (edge detection)

 Often, you don't need to know the state of a digital input all the time, but you just need to know
when the input changes from one state to another. For example, you want to know when a
button goes from OFF to ON.  This is called state change detection, or edge detection.

 This example shows how to detect when a button or button changes from off to on and on to
off.

 The circuit:
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * LED attached from pin 13 to ground (or use the built-in LED on
   most Chipkit UNO32 boards)
  */

// this constant won't change:
const int  buttonPin = 2;    // the pin that the pushbutton is attached to
const int ledPin = 13;       // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

void setup() {
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communication:
  Serial.begin(9600);
}


void loop() {
  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);

  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button
      // wend from off to on:
      buttonPushCounter++;
      Serial.println("on");
      Serial.print("number of button pushes:  ");
      Serial.println(buttonPushCounter);
    }
    else {
      // if the current state is LOW then the button
      // wend from on to off:
      Serial.println("off");
    }
  }
  // save the current state as the last state,
  //for next time through the loop
  lastButtonState = buttonState;


  // turns on the LED every four button pushes by
  // checking the modulo of the button push counter.
  // the modulo function gives you the remainder of
  // the division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
   digitalWrite(ledPin, LOW);
  }

}
```