This is still a very very rough draft , more to come

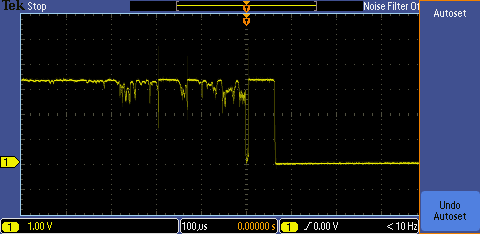
=====================================================================================

Both the Max32 and Uno32 chipKIT boards run at 80 Mhz clock rate, and what this means is that they can complete instructions in the 10ths of picoseconds. This is compared to the average person’s reaction time that ranges in the 10ths of a second. Since sketches always run in constant loops, a sketch that detects if a button is pressed may loop for hundreds to thousands of times (most likely more) in the span between when the button is pressed and when it is released.

If user input devices (such as switches and buttons) were perfectly ideal, then the difference in microcontroller speed to reaction time would not be such a issue. Unfortunately switches operate far from ideal, and have a characteristic know as bounce.

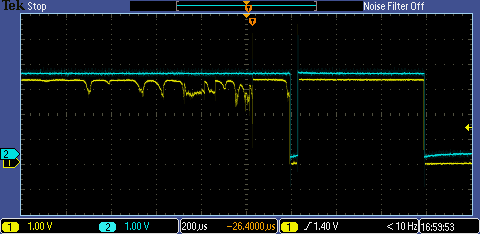
When a user presses a button or throws a switch, we would like to think of it as an ideal transition where at the moment the switch or button is activated it is considered to be in a high voltage state. In reality when a button is pressed or released (or switch thrown), there is a small amount of time (in the microsecond range) where the electrical signal can fluctuate anywhere from 0 volts to the voltage level designated as logic level high (typically 3.3V or 5V). This is caused by the physical material of the switch or button reverberating and finally winding down to a steady state. Think of it as, if you have a metal rod and you strike it against a surface, the rod will vibrate from the force of the impact. Eventually these vibrations will calm down (which is meant by winding down to a steady state). In the same sense buttons and switches do the same thing, but on a very small scale. While the physical material is vibrating, it affects the voltage level of the output.

Overall this causes the transition edges, to not be as clean (ideal) as we would like.



The above figure is a graph of voltage over time of a button releasing (identical circuit configuration as in Project 4: Button Controlled LEDs. You can see that the button goes from a steady voltage state then right before it is released, a small period of turbulence for about 400 microseconds before cutting off.

Because the Max32 and Uno32 (and most microcontrollers in general) run at such a fast rate, they can potentially capture these fluctuations which can have the potential



The above figure is now showing two different signals, (Yellow) the voltage over time signal of the button input, and (BLUE) the voltage over time signal of an external LED output (identical circuit and software configuration to Project 4). The circuit/sketch basicly reads the input from the switch and then if HIGH will drive the LED HIGH. You can see that for most of the turbulence in the button input, it is not low enough voltage level for the chipKIT to considered it as a logic level LOW, but there is a gap were the chipKIT does see a LOW.

You can see that the output now fluctuates because of the bounce. Instead of one fluid ON/OFF signal that correlates to a single press of a button or switch throw, the signal is now choppy on the ends. With only a simple program sketch the microcontroller could interprets this as multiple button presses. It won’t be able to tell the difference from a button being pressed for 1 second and a button being pressed for 100 microseconds (while the latter being physically impossible by human standards).

De-bouncing:

While there are many solutions to de-bouncing, some software and some hardware, for this project we will focus on software approach that implements a rolling input average. Several different hardware solutions to de-bouncing will be presented in forthcoming chipKIT projects.