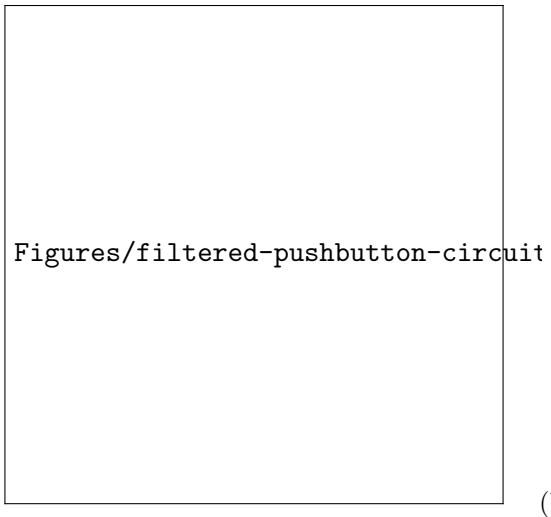
Switch Bounce & Catch the Clown Game
Embedded System Design, Lab 6

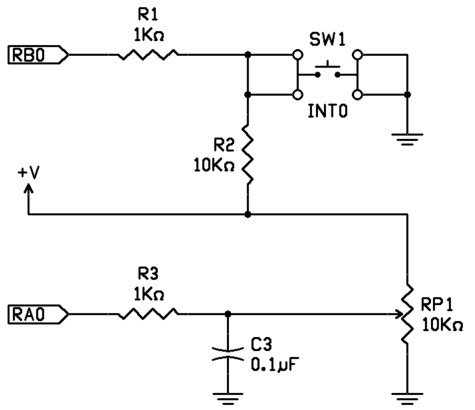
Ben Lorenzetti

November 5, 2015

Contents



(a) A Hardware Debounced Pushbutton: *R_3 and C form a low pass filter.*



(b) PIC16F887 Demo Board Schematic: *pushbutton SW1 is unfiltered and requires software debouncing.*

Figure 1: Hardware vs Software Debouncing for Mechanical Switches

1 Objectives and Problem Description

1.1 Does the Switch Bounce?

1.2 Catch the Clown!

Build a game for testing reaction times with an 8 LED rotating display, a pushbutton trigger, and a knob for adjusting the speed/difficulty. If the player presses the trigger in sync with the LED display, then the display stops rotating to indicate victory. The specifications can be summarized in the four points below:

1. For an 8 LED display, one LED should be illuminated at a time and the illuminated position should rotate right one digit every period.
2. The period should be adjustable on the fly with the rotatry potentiometer knob.
3. If the user triggers the switch while the topmost (most significant bit) LED is illuminated, then the LED display should stop rotating until the switch is released. The LED rotation loop should also continue–including through the topmost state–if the switch is active but was triggered during the wrong state.
4. The pushbutton switch should be debounced based on the results from part 1.

2 Procedure

2.1 Switch Bounce

3 Expected Results

4 Experiment and Design Revisions

4.1 PORTB Input

On the 44–pin demo board, the pushbutton is connected to `RB0` on `PORTB`, in an active low configuration with a $1k\Omega$, $10k\Omega$ voltage divider, as shown in figure 1.

When I first wrote my implementation for part 1 of the lab, I could not get the μController to respond to pressing the pushbutton switch (SW1). After inspecting the hardware with a ohmmeter, I knew the problem had to be in software. I created a knew 'hello world' program for debugging the pushbutton switch.

It turns out the bug was a configuration problem: `RB0` is connected to one of the 16 analog inputs, `AD12`. By default, the pin configured use with the ADC and the digital input amplifiers are turned off. Here is a nugget that was buried on page 49 of the PIC16F887 datasheet:

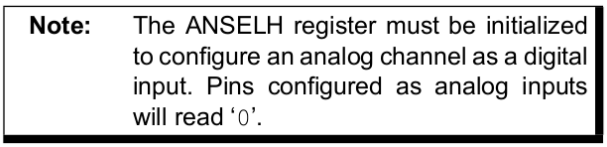


Figure 2: Port B Configuration Note

With this nugget, the working 'pushbutton hello world' program was:

```
; pushbutton-test.asm
; Test active-low pushbutton on RB0 with active-high LED on RD0

#include <p16f887.inc>
        __CONFIG        _CONFIG1, _LVP_OFF & _FCMEN_OFF & _IESO_OFF & _BOR_OFF
            & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF &
            _INTRC_OSC_NOCLKOUT
        __CONFIG        _CONFIG2, _WRT_OFF & _BOR21V

Initialize_Data_and_IO
        CLRF    PORTB
        CLRF    PORTD
        BSF     STATUS, RP0     ; Switch from Bank 0 to Bank 1
        BSF     PORTB, RB0      ; configure RB0 as input
        BCF     PORTD, RD0      ; configure RD0 as output
        BSF     STATUS, RP1     ; Switch from Bank 1 to Bank 3
        BCF     ANSELH, ANS12   ; by default RB0/AN12 is configured as analog
                                ;   input. Set to '0' to enable digital input
        BANKSEL 0x00            ; Switch to Bank 0
Main_Loop
        MOVF    PORTB, W        ; copy pushbutton input into W
        XORLW   1 << RB0        ; invert active-low pushbutton for active-high
                                ;   output
        MOVWF   PORTD           ; update LED display
        GOTO    Main_Loop
        END
```

4.2 Command Line Assembly

My `.asm` source files were assembled on the command line so please do this if they don't compile nicely in the IDE. On Ubuntu, with the default MPLAB installation location, from the directory containting `catch-the-clown.asm`, the commands are:

```
$ cp /opt/microchip/mplabx/v3.10/mpasmx/p16f887.inc ./p16f887.inc
$ /opt/microchip/mplabx/v3.10/mpasmx/mpasmx -p16f887 catch-the-clown.asm
$ more catch-the-clown.ERR
```

5 Observations

6 Discussion

7 Implementation Code

7.1 Does the Switch Bounce?

```
; debounce-time.asm
; Ben Lorenzetti
; Embedded Systems Design, Fall 2015

#include <p16f887.inc>
        __CONFIG        _CONFIG1, _LVP_OFF & _FCMEN_OFF & _IESO_OFF & _BOR_OFF
            & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF &
            _INTRC_OSC_NOCLKOUT
        __CONFIG        _CONFIG2, _WRT_OFF & _BOR21V

;---------------------- Declare Global Variables ----------------------------------;
#define active_state    0x20

Main
        ;----------------- Initialize Variables ---------------------------------;
        CLRF    PORTB           ; clear input port, just for good measure
        CLRF    active_state    ; initial active state for SW1 is active low
        CLRF    PORTD   ; PORTD serves as counter for # of bounce events
        ;----------------- Initialize I/O ---------------------------------------;
        BSF     STATUS, RP0     ; switch from Bank 0 to Bank 1
        CLRF    TRISD           ; configure port D to output for LEDs
        BSF     TRISB, RB0      ; configure pushbutton on RB0 for input
        BANKSEL ANSELH          ; by default, RB0/AN12 is configured as analog
        BCF     ANSELH, ANS12   ;   input. Reconfigure to digital.
        BANKSEL 0x00            ; return to Bank 0

Bounce_Event
        MOVF    PORTB, W        ; store pushbutton input into W
        XORWF   active_state, W ; compare pushbutton to new active level
        ANDLW   1 << RB0        ; keep only pushbutton bit
        BTFSS   STATUS, Z       ; if (pushbutton != active_state),
        GOTO    Bounce_Event    ;   then continue checking for bounce event
        INCF    PORTD, F        ; else increment the bounce counter and
        COMF    active_state    ;   invert the active state next event
        GOTO    Bounce_Event
;----------------------------- End of File ------------------------------------;
        END
```

7.2 Catch the Clown!