# EE/CPE 3280 Assignment 3 – CountZilla Interrupted

## Program: 100 Points

The Countzilla program is a program to help explore different strategies for programming interrupts. In this program, you will use the 5-position joystick to control a counter which is displayed on the screen.

## Basic Operation

A number is displayed on the screen. The up/down buttons increase/decrease it one unit at a time. The left/right buttons increase/decrease it in auto-repeat mode. The center button resets the counter to zero.

To make things a bit more interesting, you must use interrupts in different modes for each button.

## Specifications

*Definition: “Code that updates the display” means two lines of code: a sprintf() call and a GLCD\_PrintString() call.*

1. Counter. The current value of the counter shall be displayed on the LCD at all times except when the left button is held down.
2. Main program. The main program shall have initialization code that runs at startup. After startup, the main program shall enter an infinite loop that updates the display with the current value of the counter. The main loop shall **not** include any code except code that updates the display.
3. Up button. Pressing the up button shall increment the counter by one. Pressing the up button continuously shall not change the value of the counter, other than the initial increment by one. The up button shall be connected to the MCU through an edge-triggered interrupt. The ISR for the up button shall include code that updates the display.
4. Down button. Pressing the down button shall decrement the counter by one. Pressing the down button continuously shall not change the value of the counter, other than the initial decrement by one. The up button shall be connected to the MCU through an edge-triggered interrupt. The ISR for the down button shall **not** include code that updates the display.
5. Right button. Pressing the right button shall increment the counter by one or more. Pressing the right button continuously shall increment the counter 10 times per second. The right button shall be connected to the MCU through a level-sensitive interrupt. The ISR for the right button shall include code that updates the display.
6. Left button. Pressing the left button shall decrement the counter by one or more. Pressing the left button continuously shall decrement the counter 10 times per second. The left button shall be connected to the MCU through a level-sensitive interrupt. The ISR for the left button shall **not** include code that updates the display. Note: This means that the display should not update when the button is held down. Don’t try to force it to update!
7. Center button. Pressing the center button shall set the counter to the value zero. The center button shall be connected to the MCU through an edge-triggered interrupt. The ISR for the center button shall **not** include code that updates the display.

Summary Function Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Button | Function | Repeat | Interrupt Type | Update Display in ISR |
| Up | Increment | No | Edge | Yes |
| Down | Decrement | No | Edge | No |
| Right | Increment | 10x/second | Level | Yes |
| Left | Decrement | 10x/second | Level | No |
| Center | Reset to zero | No | Edge | No |

## Coding Tips

* The pushbuttons create logic ‘0’ when pressed. You may need to invert these signals in the schematic using a “Not” gate.
* Sprintf format strings can be used to display a number using a fixed number of digits. See this [C reference](http://www.cplusplus.com/reference/cstdio/printf/) for help.
* Any program file (.c) that uses sprintf needs to #include <stdio.h>. Any program file (.c) that uses the GLCD library needs to #include <project.h>.
* Variables and functions used across multiple .c files must be globals (defined outside of main or other functions). To use them in other .c files, declare them as “extern”.
* If you need a time delay, remember CyDelay(), which delays the number of milliseconds specified.
* Bouncy buttons need to be debounced. This can be done in an interrupt with a short time delay.
* Edge-triggered interrupts create “pending” interrupts if they see a second edge while in the ISR. This second edge could be created by a bouncing button. If you need to clear a pending interrupt, use <interrupt\_name>\_ClearPending();
* Is your display flickering when you push buttons? It may be that the GLCD\_PrintStr function is being interrupted when buttons are pressed. Try disabling interrupts before the GLCD call and re-enabling after the GLCD call.

## TO TURN IN Through Canvas

Turn in the following:

1. Submit a single MS-Word document containing the following through Canvas
   1. Top-level schematic (select all of your drawing, copy and paste it into the Word doc)
   2. main.c (Select all text with ctrl-a, copy and paste into the Word doc)
   3. Modified portions of all ISR .c files (usually just the “Includes” portion and the “Interrupt” portion). Make sure that you have a header at the top of each file that identifies its purpose.
   4. Copy in any other files that your created or modified
2. Demonstrate your program to the instructor during class time on the due date. (Online students may send a video demo to the instructor)