GPIO Programming Lab

Overview

For this lab you will create a device which makes sounds through a speaker based on how the user presses several switches.

Hardware Details

Use a switches SW1 to control the speaker. Drive a speaker SP1 from a GPIO output (labeled Audio) using capacitor C1 to block DC current. Resistor R1 is optional and reduces the volume of the sound. Use the circuit diagram shown in the last slide of the GPIO lecture.

Software Details

Write code in C to do the following

The tone should sound while SW1 is pressed down and stopped when the SW1 is pressed down again.

The following software design is suggested:

- Create an initialization function which configures GPIO inputs and outputs based on which pins to which you've wired your switches and speaker.
- Use the delay-loop function delay_us which has been provided in the demo project to cycle at the correct frequency.
- Create a function play_tone(unsigned int duration_ms) which generates a square wave with the given period (specified in microseconds) and duration (milliseconds). This can be done by toggling the audio output pin, waiting for a time delay, and repeating this process. Calculate the necessary value to pass to delay_us based on period (inverse of frequency). Calculate the number of times to toggle the output based on period and duration. This demo code is given in the last lecture slide.
- Create a function update_buttons(void) to repeatedly check to see if the switch SW1 is pressed and respond accordingly.
- If SW1 is pressed and the play_tone has not been called call play_tone(unsigned int duration_ms) with a duration of your choice.
- If SW1 is pressed and the play_tone was already called, stop the play_tone by driving the GPIO output of the speaker accordingly.

Interrupt Programming Lab

Overview

For this lab you will create a device which measures how quickly a person can press a switch in response to an LED being lit. This will give you an idea of how much work the processor can do in the time it takes you to react to an event.

Hardware Details

Use a switch to control the system and use the RGB LEDs as output devices.

Software Details

The main code should perform the following:

- Initialize peripherals
- Repeat the following
    - Turn off all LEDs
    - Clear counter
    - Wait a random amount of time (e.g. within 1-3 seconds)
    - Turn on one LED
    - Repeat until ISR has been triggered, as indicated by the flag being set
        - § increment counter
    - Save counter value in memory
    - Wait for approximately 5 seconds

The ISR should perform the following:

- Set a flag indicating the ISR has executed

You will also need some support functions:

- Use the **leds.c** module to initialize and control the RGB LEDs.
- Use the C standard library function **rand()** to generate a random integer.
- Use the **delay_ms** function provided by **delay.c** to wait for a number of milliseconds.

Testing

In order to see the number of iterations counted, set a breakpoint in your main function after the switch press has been detected and examine the counter variable using the watch window.

What happens if user pressed switch without any LED being on ? (check for this error condition and light the red LED.