Ulysses Butler
CS 484: Embedded System

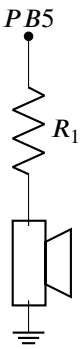
Final Project Write Up

1 Electronics

1.1 Sound

In order to get the sound working with this project, we used PWM mode 7 with Timer0. Since we're using output compare mode, we hook the speaker up to `OC0B` (which is located on the same physical pin as `PD5`.

In our program, we want to play a different tone for each light. In our project, we decided that the green lights will play an F, red will play a G, blue will play an A, and yellow will play a B tone. These correspond to roughly 349 Hz, 392 Hz, 440 Hz, and 493 Hz, respectively. To acheive this, we can use a prescalar of 256 for all of these tones. Depending on what tone we're trying to play, we simply set the approprite values for the output compare registers, `OCR0A` and `OBR0B`.



Once the timer is set up, we hook `OC0B` up to a potentiometer, which is then attached to the transducer. The transducer is connected to ground, completing the circuit. Once this is done, the potentiometer can essentially be used to control the volume of the tone. To stop the tone, we can just turn off the timer.

1.2 Lights

The LEDs can be connected to any of the remaining ports. For this project, we've decided to hook them up to `PB0`, `PB1`, `PB2`, and `PB3`. That being said, these can easily be modifed in software. The IO registers are initialized using the `lcdInit()` function, which takes the numbers of four physical pins as parameters. These circuit for these lights then passes through 330 Ω resistors to ground. An LED is then turned off or on by controlling its respective pin.

When we display the current sequence to the user, we want to wait about a second before we turn off the light so the user has a chance to see what it is. Once the light is off, we want to wait for another moment for the user to see that the light is changing. This way, if the same light is shows multiple times in a row, the user will be able to easily disguish between each flash. This is accomplished by waiting between the commands to turn the light on or off. In order to accurately time the duration of the wait, we use the 16-bit timer. Since nothing needs to be done during the wait, the function can simply block and return after the desired time. This is done by starting the timer with approprite scalar and output compare value, then watching the approprite flag. Once the flag is set, we return from the function.

1.3 Buttons

Similarly, the four buttons are connected to four pins indicated by parameters to the `buttonInit()` function. For our project, we decided to connect then to `PD0`, `PD1`, `PD2`, and `PD3`. These pins are set out input pins, then the internal pull-up resistors are enabled. The buttons are then connected to ground so we can monitor their values in real time. Here, we again use a timer to compensate for button bounce. Once the button is lifted, we wait for a little over 50 ms and make sure the button is still lifted at the end.

1.4 Random Number Generation

In order to generate a random sequence of values, we use the ADC as it reads values from the internal thermometer. In order to generate a random value, we generate all 10 bits of the ADC. This is far too precise to be accurate, so the last few bits are extremely volatile. We then return the value of the last two-bits. This gives us a random number from 0 to 3 (inclusive), one value for each light. You could say we built a... two-bit random number generator.