Carlton Duffett
EC450 HW2
Professor Giles
2/27/2015

EC450 Homework 3

Overall Design:

I designed this program as a state machine with the following 4 modes:

Mode:	Code:	Identification:
Record	`r`	Red light is steady on
Playback	`p`	Red light is off
Transition **to** playback	`t`	Red light flashes 3 times, then remains off
Transition **from** playback	`f`	Red light flashes 3 times, then remains steady on

The main WDT interrupt handler simply checks the current mode and calls a function to implement the mode's behavior.

Record Mode:

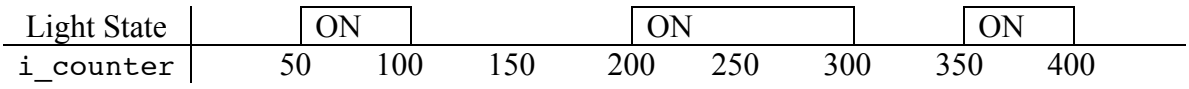
To record the button presses in an efficient way, I record only the *transitions* that the button makes from pressed to not pressed. To identify a transition I maintain a global variable **`last_button`** that holds the state of the button from the last interrupt. I compare this to a local variable **`this_button`**, the state of the button in the current interrupt. For timing, I maintain a global variable **`i_counter`**, the number of WDT interrupts since recording started.

Whenever the button's state changes (e.g. **`last_button != this_button`**) I record the current **`i_counter`** in a global array of transitions, **`t[]`**. I keep track of the position in the array using **`t_counter`**.

For example, the array **`t[]`** may contain:

```
t[50, 100, 200, 300, 350, 400, ...]
```

This corresponds to the button presses and releases (and resulting flashes of the green light):



At interrupt 50 the button is pressed, recording a transition. At interrupt 100 the button is released, recording another transition, etc. Recording continues in this way until a limit is reached (see "Limitations" section).

This implementation assumes that when recording starts the button is initially not pressed.

Playback Mode:

Playback is surprisingly simple due to the way the flashes were recorded. Since `t[]` contains the time of each on/off transition, I simply XOR the value of P1.6 (the green LED) whenever a transition is reached. Since I assume that recording starts with the light off (and the button not pressed), the first transition XORs the light to the ON state. The next transition XORs the light back to the OFF state, and playback continues in this way until the end of the recording.

Transitions:

The transitions between recording and playback are handled in a similar way. In both cases, `setupTransition()` is called to turn off both lights and reset all counters. The `transition()` function then handles the flashing of the red LED before changing the mode to record or playback. To identify what mode comes next, `transition()` takes a variable `nextMode` as an input.

Transition from recording to playback occurs if:

- The 2 minutes allocated for recording are used up
- The 100 elements of `t[]` allocated for recording transitions are used up
- The user waits more than 4 seconds since his/her last button press

Transition from playback to recording occurs if:

- The 2 minutes allocated for playback are used up
- The end of the recording is reached

When playback is finished, the last recording is erased to make room for a new recording.

Limitations:

The main constraint on this design is memory. Since the MSP430 G2553 chip only has 512 bytes of memory, the maximum size of the recording array is roughly 250 integers. Also, since an integer is only 16 bits, a counter can only count up to roughly 65,000. Fortunately, these constraints had little impact on practical use of this program.

Using a 7.4ms interrupt length, the maximum possible record time is approximately 8 minutes. This is unnecessarily long in practice and I restricted my recording period to only 2 minutes. I also chose to restrict the number of recorded transitions to 200. This allows for 50 full button presses per minute, or roughly one per second. These constraints give the user plenty of flexibility when recording while avoid the physical limitations of the processor. Since the program is shielded from the processors physical limits, when a limit is encountered the program handles it by transitioning from one state to the other gracefully.