Lab 09: LEDs and buttons



R. Ferrero, A. C. Marceddu, M. Russo

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

This work is licensed under the Creative Commons (CC BY-SA) License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/



- Add to led.h file the prototype:
 void led4and11 On (void);
- Add to 'led' group the file funct_led.c
- Implement in funct_led.c the function led4and11_On(void), powering on the LEDs 4 and 11 acting on the FIOSET register.
- Note: the state (on/off) of the other LEDs must not be modified.
- Test the function calling it from the main.

- Add to led.h file the prototype:
 - void led4 Off(void);
- Implement in funct_led.c the function led4_Off(void), switching off LED 4 acting on FIOCLR register.
- Note: the state (on/off) of the other LEDs must not be modified.
- Test the function calling it from the main.

- Add to led.h file the prototype:
 void ledEvenOn OddOff(void);
- Implement in funct_led.c the function ledEvenOn_OddOff(void), powering on the LEDs with even index number and powering off odd ones, acting on FIOPIN register.
- Test the function calling it from the main.

- Add to led.h file the prototype:
 void LED On (unsigned int num);
- Implement in funct_led.c the function void LED_On (unsigned int num) powering on the LED corresponding to the parameter passed:
 - num = 0 -> LED 4
 - num = 1 -> LED 5
 - num = 7 -> LED 11
- Test the function calling it from the main.

- Add to led.h file the prototype:
 void LED Off (unsigned int num);
- Implement in funct_led.c the function void LED_Off (unsigned int num) powering off the LED corresponding to the parameter passed: num = 0 -> LED 4
 - num = 1 -> LED 5
 - num = 7 -> LED 11
- Test the function calling it from the main.

- In the main, before entering the endless loop, power on LED 8 using LED On.
- By pressing button KEY1, power off the current LED and power on the LED on the left (when arrived to LED 4, jump to LED 11).
- By pressing button KEY2, power off the current LED and power on the LED on the right (when arrived to LED 11, jump to LED 4).
- By pressing button INT0, get back to original configuration, with LED 8 on.

What LED is on?

- To know which LED is on you can:
 - Read content of LPC GPIO2->FIOPIN
 - Read content of LPC GPIO2->FIOSET
 - define a global variable in sample.c or funct_led.c:

```
unsigned int led_value;
```

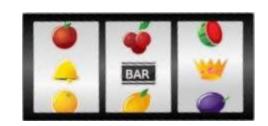
led_value stores the only powered LED.

In the other files you can access the variable declaring:

```
extern unsigned int led_value;
```

- A single button press can trigger more than one interrupt service request.
- In exercise 6, as a result, pressing a button once might cause the lighting of a LED located further than 1 position with respect to the current one.
- A software anti-bounce mechanism must be implemented to serve just the first interrupt request and ignore the immediately following request. Hint: the problem is about too much speed.

• The objective is to implement a slot machine with 3 reels.



Each reel shows one of two symbols:

Reel	Symbol 1	Symbol 2
Reel 1	led 4 on	led 5 on
Reel 2	led 6 on	led 7 on
Reel 3	led 8 on	led 9 on

Exercise 8: implementation

- The KEY1 button starts a new game and controls the first reel:
 - It powers off all the LEDs
 - It randomly powers on either LED 4 or LED 5.
- The KEY2 button controls the second reel:
 - It randomly powers on either LED 6 or LED 7.
- The INTO button controls the third reel and determines if victory has happened:
 - It randomly powers on either LED 8 or LED 9.
 - Based on the result, it powers on LED 10 or 11.

Exercise 8: result of the game

- The player wins if the 3 symbols are the same:
 - all the reels show symbol 1, or
 - all the reels show symbol 2.
- At the end of the game (after pressing INT0), victory is indicated lighting LED 11.
- If the symbols are not the same, instead, the player loses and LED 10 is lit.

Random number generation

- LEDs 4-9 must be randomly lit.
- If the game is not finished, the re-pressing of KEY2 must not be considered by the program.
- A simple way to generate a random number is to always increment an index in the idle loop (the infinite cycle) in sample.c and to use modular arithmetic. Mind the maximum possible value!