Pushbutton, LED and Interrupts

ELE 271: Laboratory 5

Introduction

This experiment is about external interrupts. Interrupts allow for more efficient monitoring of external of inputs by using dedicated hardware instead of software via the so-called busy-waiting or polling method.

Read Section 11.8.

Part 1

(a). In the first exercise the goal is to use the blue pushbutton (PC.13) to turn the green LED (PA.5) on and off.

Configure GPIO PA.5 pin as output.

Configure GPIO PC.13 pin as input.

Configure the external interrupt registers to monitor pin PC.13.

Use an interrupt service routine to toggle the LED every time the pushbutton is pressed.

```
#include "stm32l476xx.h"

// PA.5  <--> Green LED
// PC.13 <--> Blue user button

void configure_LED_pin(){
    <your code goes here>
}

void configure_BUTTON_pin(){
    <your code goes here>
}

void configure_EXTI(){
    <enable SYSCFG clock>
    <configure EXTICR to connect to PC.13>
    <configure EXTI registers to enable interrupts>
}

void EXTI15_10_IRQHandler(void) {
    <toggle LED and clear flag>
    }
```

```
int main(void){
    configure_LED_pin();
    configure_BUTTON_pin();
    configure_EXTI();

  // Dead loop
    while(1){;)
}
```

Part 2

Connect PB505 ground to STM32 Nucleo ground. Set the function generator to square wave. Connect the TTL output (!!!) of the function generator on the PB505 to a selected GPIO pin. Configure the pin as input.

Toggle green LED on every falling edge of the input.

Set the square wave frequency to 10KHz. Use the logic analyzer to capture the input square wave and the output signal sent to the GPIO output pin.

Determine the delay resulting from the overhead in servicing the interrupt.

At approximately what frequency does the microcontroller no longer keep up with the input signal?