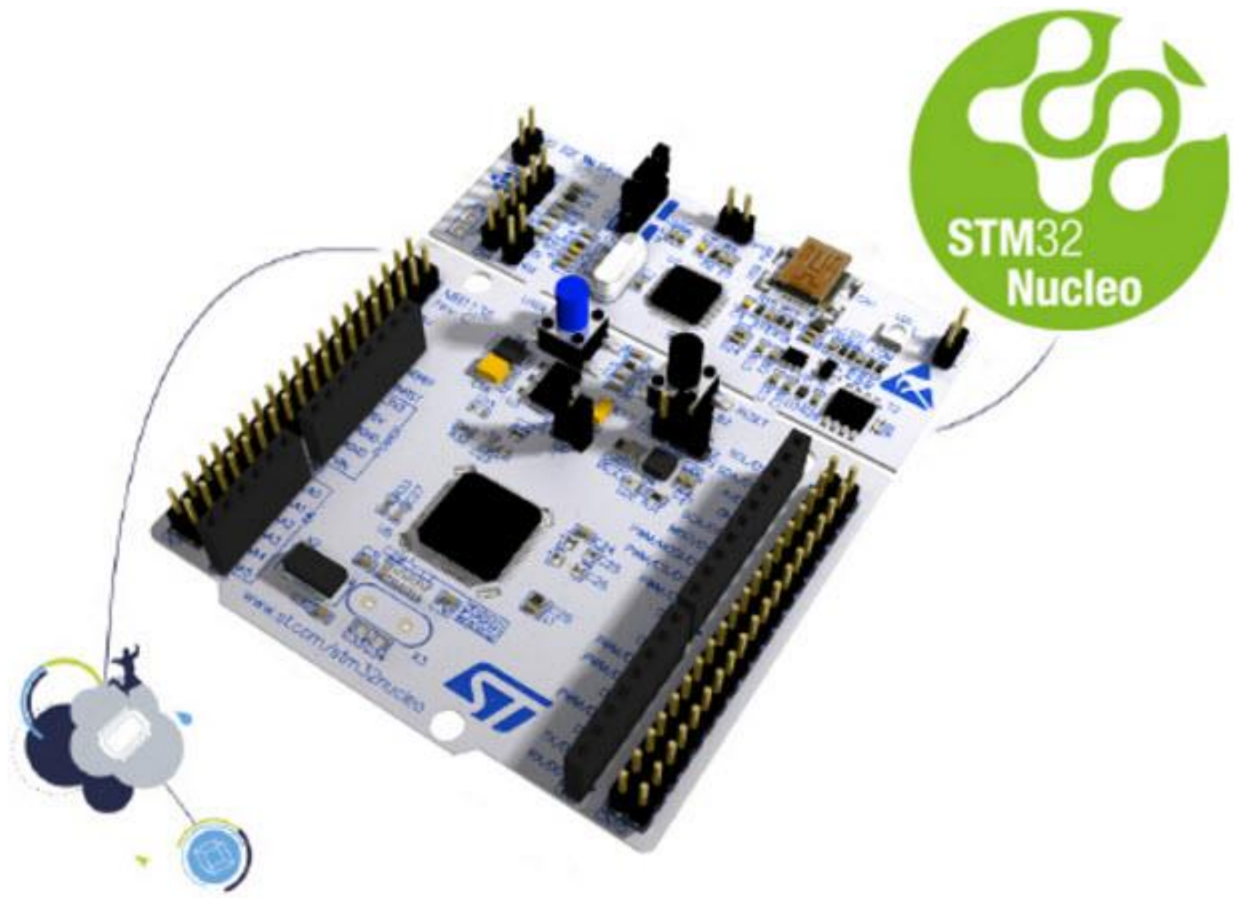
Technical report

Assignment 2:Interrupts

By Veselin Atanasov



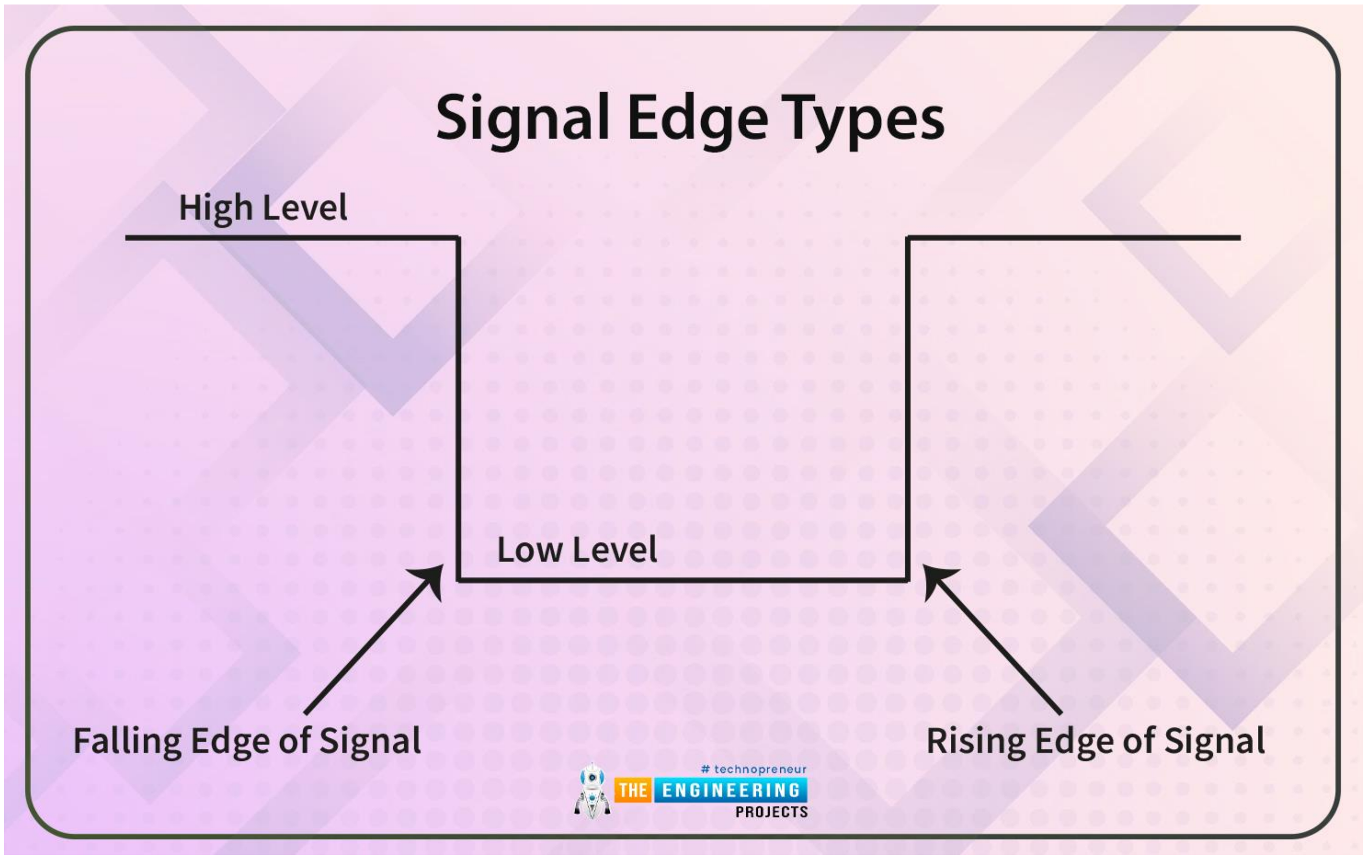
Teachers: Rene Bakx, Robert Verhijen

Date:18-09-2024

Contents

Introduction

In this assignment the goal is to understand external interrupts and how to use them. For this assignment we need to create a program that flips a led based on falling and rising edge interrupts one button and a that toggles a led using falling edge trigger on another button.



Description

The task is to enable pin-change interrupts for two buttons in order to change states of two leds. One of the buttons should control the led so it is turned on while the button is pressed and its turned off when the button is released. To achieve that, there is need for two interrupt triggers- falling and rising edge(or rising and falling, depending on the resistor configuration). The other button should toggle a different led. To achieve that either falling or rising edge is needed.

The very first step to make this program is to set up the GPIO pins that are going to be needed for the buttons and the leds(in this case PC13(B1), PA8(B2), PA5(LED1) and PA10(LED2)). After that is really important to enable SYSCFG(system configuration clock) which is in the RCC->APB2ENR register. Then the EXTICR(external interrupt configuration) register should be set up. The EXTICR register that needs to be addressed depends on the pin that causes the interrupt. In this case PC13 falls in the EXTICR4 group.
Note!:It is important to keep in mind that when coding using CMSIS the groups are selected using array indexing(0, 1, 2 correspond to 1, 2, 3) because those groups are represented as structs. After the group is selected the bits in this group should be reset to avoid unexpected behaviour. Afterwards, the exact pin in this group should be specified. After the pin is selected, the FTSR or RTSR register(or both as it is for B1) should be set. After this the line to the NVIC register should be opened with the IMR register.
Note!: Sometimes pins may be from different groups but still use the same interrupt lines! The final step is to add the interrupt to the vector table. That is done using the CMSIS function NVIC_EnableIRQ() which takes as argument the lines that can signal to NVIC that interrupt event occurred. After all of this is done it is only left to program the interrupt handler. The interrupt handler should be as simple as possible. In this program it just flips a bit for the led in the ODR register. It is a good practice to name the interrupt handler using the following manner EXTIN_IRQHandler(N stands for the interrupt lines that can signal NVIC for an interrupt to occur). It is important to note that if code is written in C++ the "extern C" line should be written before the declaration of the interrupt handler.

Conclusion

Interrupts are a really good tools to handle a lot of scenarios. Whether it is an emergency stop of a machine or some timed event. This assignment was a good exercise on how they actually work while also further exercising our documentation-analysis skills and expanding our knowledge on the STM32 board.