EXTERNAL INTERRUPT USING PUSH BUTTON

STEPS TO PROGRAM A PUSH BUTTON INTERRUPT

1. Open the STM32CUBE Ide and Launch the STMCUBE Ide Create a New STM32 Project.

File→New→STM32 Project

2. Project Selection Window will POP-UP, Select Board Selector Type Nucleo F446RE in "Commercial Number", In the STM32 Project window. Give project Name ->Select "Project Targeted Type "—Empty and click on Finish.

3. To program External interrupt using the button we are required to include the necessary Libraries in the program.

Systick and PLL Header files are included with the required variables, directives, and frequency calculations.

4. Define Directives and global variables in a sequence in accordance with the availability of the registers, bitwise shift operations are defined by referring to the reference manual rm3090.

5. Configure the Systick timer to generate an interrupt for every second(Refer to 10.1.2 in rm3090 manual)

6. Configure the system clock using PLL to achieve the desired frequency, bitwise shift operations are defined and a pointer to that structure is defined below.

i. RCC_CR
ii. RCC->PLLCFGR
iii. RCC_CFGR
iv. RCC_CR
v. RCC_APB1ENR
vi. PWR_CR
vii. PWR_CSR
viii. FLASH_ACR
ix. RCC_CFGR

It enables and configures the HSI clock, PLL, power control interface, and flash memory for optimal performance. Finally, it switches the system clock to the PLL output. This detailed explanation aligns with the information provided in the STM32F4 Reference Manual with the same syntax.

7. Coming to the main program First Initialize the PLL and Systick to set up the system clock and generate an interrupt for the Second.

Secondly, Enable clocks for the GPIOA, GPIOC, and System configuration controller(SYSCFG)

RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_GPIOCEN;//0x40023830

RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN; /// 0x40023844

i.Port A is connected to the AHB1 Bus. So clock to port A must be enabled on AHB1 for PORTA the RCC_AHB1ENR register. (check sections 6.3.10 and 6.3.13 to enable clock for PORTA& PORTC)

ii. Now coming to registers related to PORTA, firstly the PORTA should be configured as either input or output, using (GPIOx_MODER register on a relevant bit on section 7.4.1 on rm3090 manual). // 0x40020000

iii. System configure the register EXTI to be connected with PC13

SYSCFG->EXTICR[3] |= EXTI_C13; // 0x40013814

EXTI->IMR |= BUTTON_PIN;/// 0x40013C00

EXTI->FTSR |= BUTTON_PIN;// 0x40013C0C

Unmask the interrupt and configure EXTI to trigger on the falling edge by 1 << 13 for IMR and FTSR.

iv. Enable EXTI interrupt in NVIC and also enable the global interrupt after configuring EXTI. Then the main work is on the Interrupt handler.

8. In the EXTI interrupt handler, if the EXTI_PR register is (1<<13), The “PR” register allows the software to clear pending interrupt flags by writing 1 to the corresponding bit.

```
if(EXTI->PR & BUTTON_PIN){
    EXTI->PR |= BUTTON_PIN;


GPIOA->ODR ^= LED_PIN; }
```

ODR register controls the output state of GPIOA to toggle PA5 by performing XOR operation.

9. The operation External push button for the STM32F446xx microcontroller is to initialize the system clock using PLL, use SysTick for millisecond timing and delays, configure PA5 as an output for the LED, set PC13 as an input with an external interrupt to detect button presses, and toggle the LED on PA5 whenever the button on PC13 is pressed.

10. Then complete the compile(build steps) and debug steps to see the behavior on the board LED blinking on PA5 with the onboard External interrupt push button.