

STM32 COURSE



Activar Windows
Ve a Configuración para activar Windows

FEBRUARY 23 2021

003 GPIO External Interrupt

Creado por: Ing. Christian Salazar

SECTION 3

003 GPIO External Interrupt



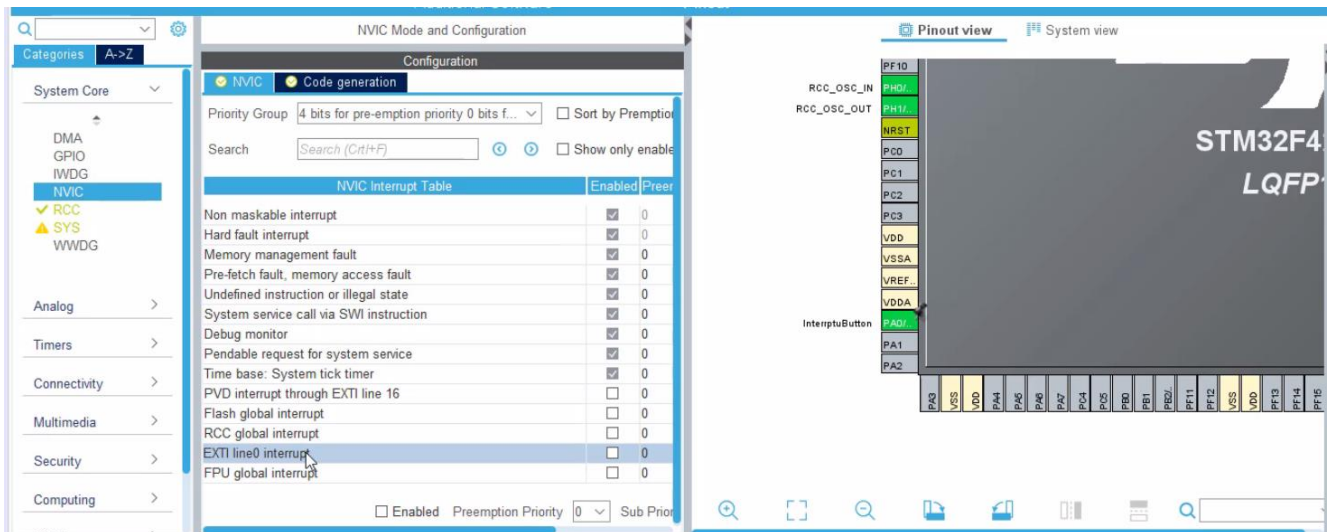
What will we learn?

In this video we will program the PA0 input of the STM32F429 Discovery card as an Interrupt input, which has a Blue Button (User Button) to be able to generate in mathematical terms a unit step that is not more than a pulse, we must take into account the above clarified in the previous video about the noise or rebound of the mechanical action elements, we will also manipulate the NVIC priority vector to which CMSIS responds where we will change the priority of our interruption.

“We will use HAL Drivers, which will greatly help us to port and recycle code routines from one processor in one Family to another in another Family ”.

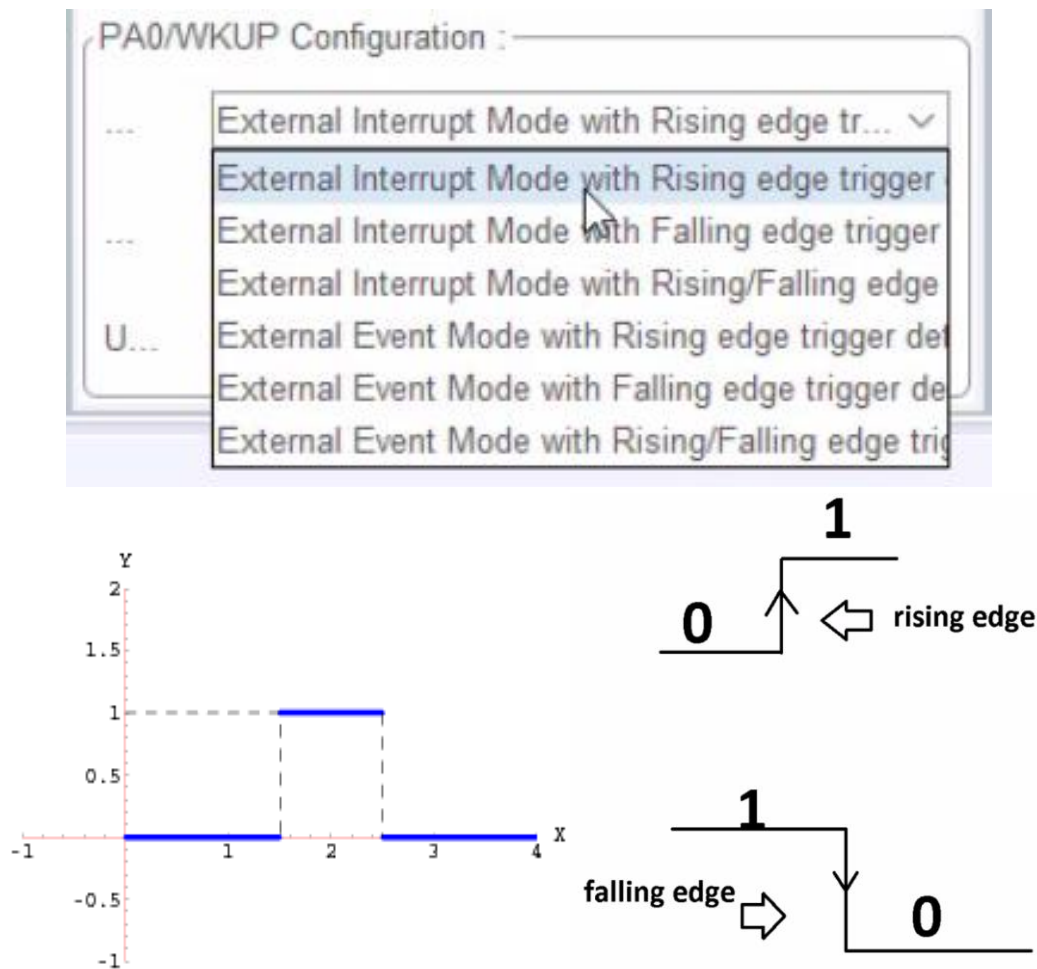
Key points

NVIC



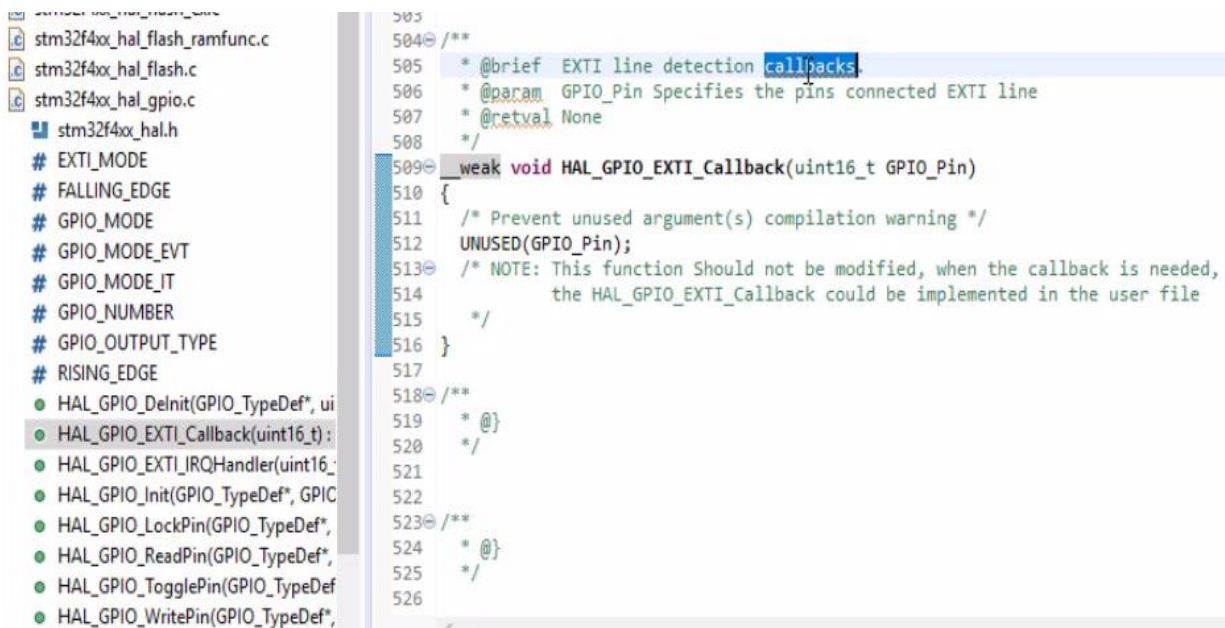
The NVIC is a vector of priorities to which CMSIS refers to treat the enabled interruptions, these can be from a USART peripheral, I2C, SPI, TIMER, etc., in our case the EXTI interrupt, that is, an external interruption, this NVIC also It is affected and used when using RTOS to manage our interruptions, keep in mind that a low number that is to say 0 (Zero) is the highest priority and to keep adding is to keep lowering the priority, this vector does not help when at a given moment it is present two or more interruptions at the same time, then the interruption will be attended from the highest priority to the lowest, if we have two interruptions of the same priority, we have the option of sub-priority or if in turn two or more interruptions coincide with the same priority and sub-priority what CMSIS does is attend to the interruption that has first in its list of interruptions, these priority criteria can become deterministic if it were to We are going to use several interrupts in some project.

LEAD RISING, DOWN, OR BOTH



The action of pressing and releasing a button can be considered as a unit step in ideal conditions, but in real conditions we must consider the noise or rebound that exists at the moment of connection and disconnection, as we can see in the image we have the option of generating an interruption of the main program (while infinite) in three ways with the EXTIs, these are with the rising edge which is the change from 0 to 1 which is when the interruption will occur, also with the falling edge which is the change from 1 to 0, remember that normally we will have both states in a complete cycle and we can also generate the interruption with both edges at the same time, that is, it interrupts with the change of state of the input.

CALLBACK INTERRUPTION FUNCTION



```
stm32f4xx_hal_flash_ramfunc.c
stm32f4xx_hal_flash.c
stm32f4xx_hal_gpio.c
stm32f4xx_hal.h
# EXTI_MODE
# FALLING_EDGE
# GPIO_MODE
# GPIO_MODE_EVT
# GPIO_MODE_IT
# GPIO_NUMBER
# GPIO_OUTPUT_TYPE
# RISING_EDGE
● HAL_GPIO_DeInit(GPIO_TypeDef*, uint16_t)
● HAL_GPIO_EXTI_Callback(uint16_t):
● HAL_GPIO_EXTI_IRQHandler(uint16_t)
● HAL_GPIO_Init(GPIO_TypeDef*, GPIO_Pin_t)
● HAL_GPIO_LockPin(GPIO_TypeDef*, uint16_t)
● HAL_GPIO_ReadPin(GPIO_TypeDef*, uint16_t)
● HAL_GPIO_TogglePin(GPIO_TypeDef*, uint16_t)
● HAL_GPIO_WritePin(GPIO_TypeDef*, uint16_t, GPIO_PinState)

503
504 /**
505  * @brief EXTI line detection callbacks.
506  * @param GPIO_Pin Specifies the pins connected EXTI line
507  * @retval None
508  */
509 __weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
510 {
511     /* Prevent unused argument(s) compilation warning */
512     UNUSED(GPIO_Pin);
513     /* NOTE: This function Should not be modified, when the callback is needed,
514        the HAL_GPIO_EXTI_Callback could be implemented in the user file
515     */
516 }
517
518 /**
519  * @}
520  */
521
522
523 /**
524  * @}
525  */
526
```

Once the interruption occurs and CMSIS has determined to attend to it, the sequence of doing so consists firstly of modifying the appropriate registers (EXTI interrupt registers) to restart the interrupt triggered signal, this is done in our case in the stm32f4xx_it file .c that is in the / src folder where main.c is located, once the interruption is attended by the CMSIS we can add a function or code to be executed when the interruption occurs, this is done with the function shown in the previous image which is the one that is executed or raised immediately after attending the aforementioned registers and flags, it should be clarified that this void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin) function will be called by all enabled EXTIs (EXTI0 - EXTI15) with an argument (uint16_t GPIO_Pin) the which can be from GPIO_PIN_0 to GPIO_PIN_15 to determine who triggered the function, that is, we can have up to 16 very different interruptions many of this type because the PA0, PB0, PC0, etc. pins are multiplexed for the EXTI0 interruption, so we can trigger EXTI0 with any of these pins as well for PA1, PB1, PC1 for EXTI1 and so on, in the

video we can see how can we separate codes for each EXTI with a simple IF () loop.