MICROS 32 BITS
STM – Interrupciones 2

Realizada por: ROBINSON JIMENEZ MORENO

Profesora: LUISA FERNANDA GARCÍA

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

10 Nested vectored interrupt controller (NVIC)

10.1 NVIC features

The nested vector interrupt controller NVIC includes the following features:

- up to 98 maskable interrupt channels for STM32F75xxx and STM32F74xxx (not including the 16 interrupt lines of Cortex$^{\circledR}$-M7 with FPU)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- low-latency exception and interrupt handling
- power management control
- implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to programming manual PMxxxx.

7.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

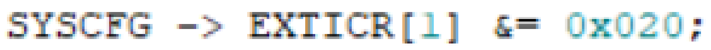
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001:PJ[x] pin
1010:PK[x] pin

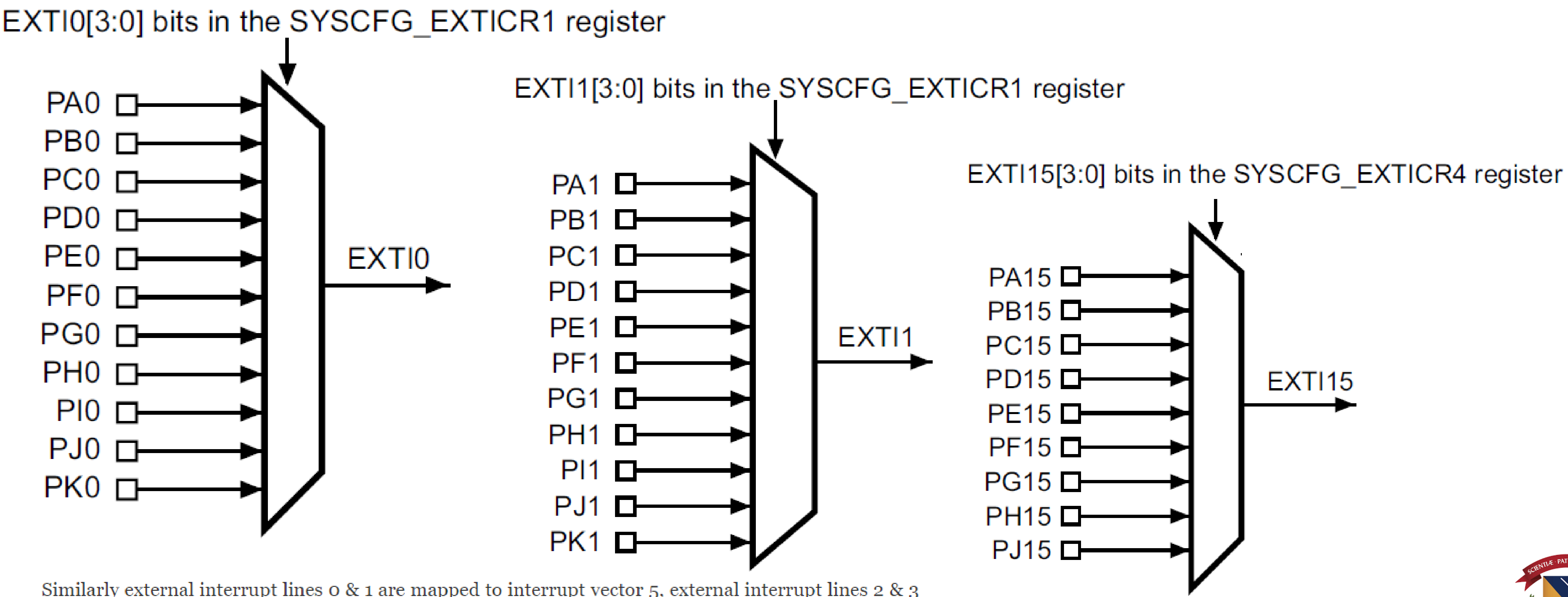


```
SYSCFG -> EXTICR[1] &= 0x020;
```

pin PC5, interrupción EXTI5

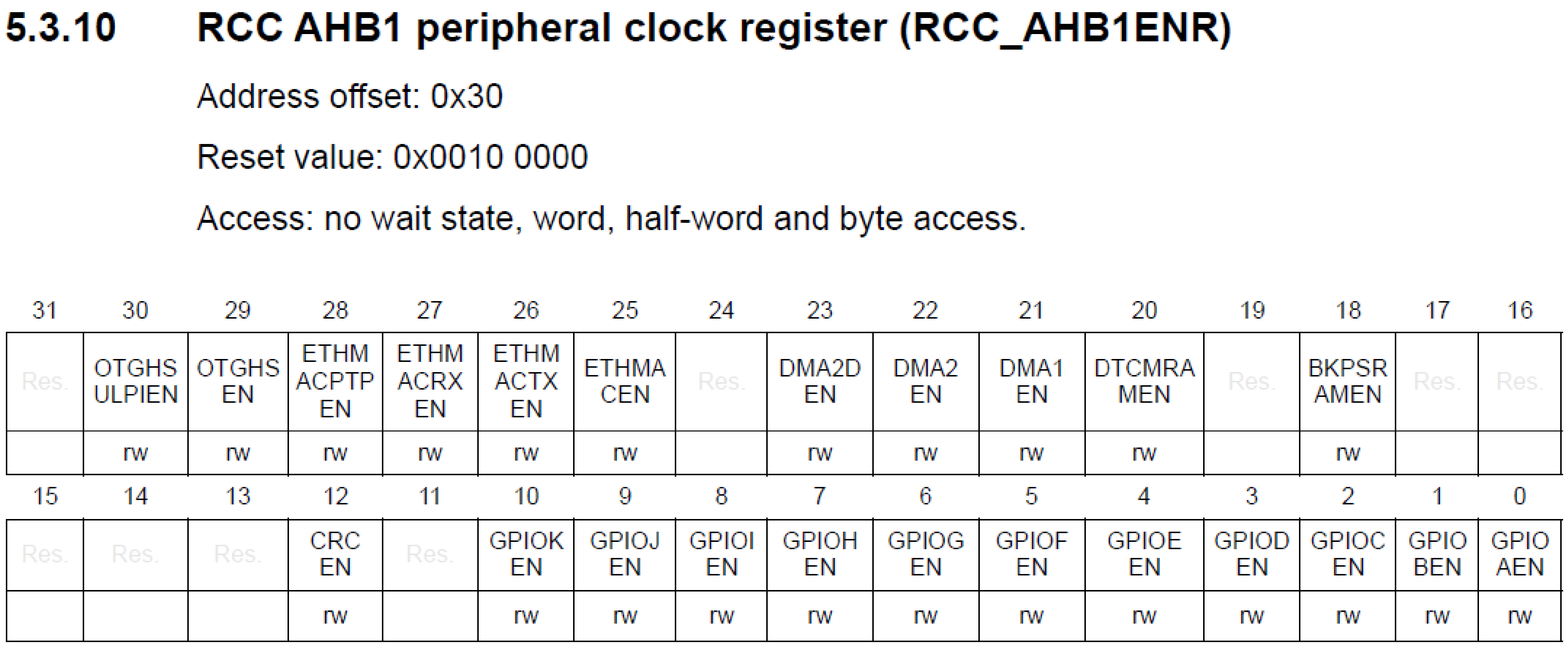
External interrupt/event line mapping

Up to 168 GPIOs are connected to the 16 external interrupt/event lines in the following manner:



EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
PH0
PI0
PJ0
PK0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
PH1
PI1
PJ1
PK1
EXTI1
EXTI15[3:0] bits in the SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
PH15
PJ15
EXTI15

Similarly external interrupt lines 0 & 1 are mapped to interrupt vector 5, external interrupt lines 2 & 3 are mapped to interrupt vector 6 and external interrupt lines 4 through 15 are mapped to interrupt vector 7.



5.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

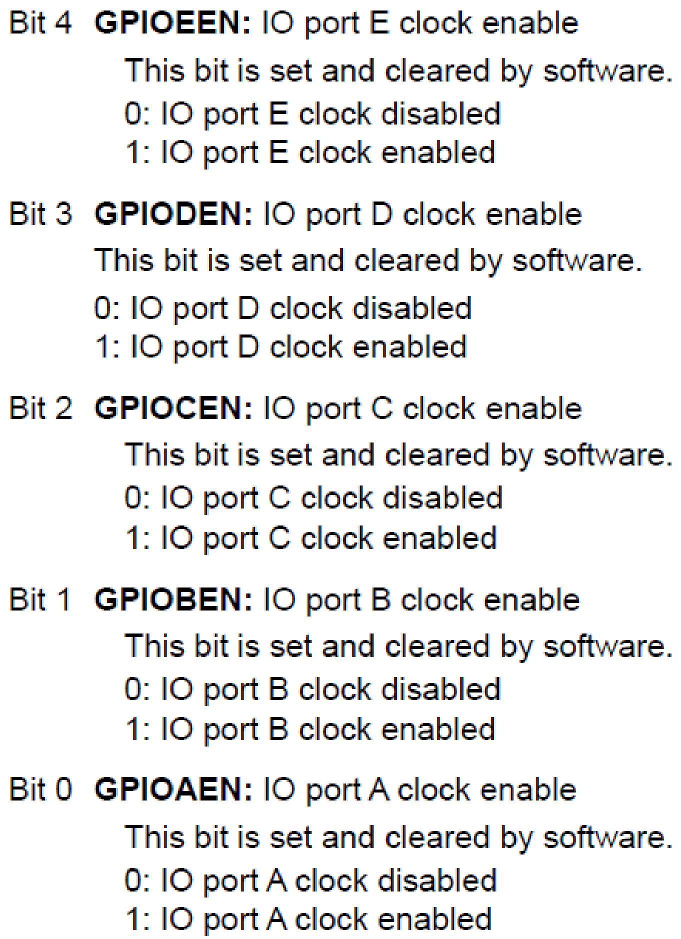
Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	OTGHS ULPIEN	OTGHS EN	ETHM ACPTP EN	ETHM ACRX EN	ETHM ACTX EN	ETHMA CEN	Res.	DMA2D EN	DMA2 EN	DMA1 EN	DTCMRA MEN	Res.	BKPSR AMEN	Res.	Res.
	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC EN	Res.	GPIOK EN	GPIOJ EN	GPIOI EN	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIO BEN	GPIO AEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



Bit 4 **GPIOEEN:** IO port E clock enable

This bit is set and cleared by software.

0: IO port E clock disabled

1: IO port E clock enabled

Bit 3 **GPIODEN:** IO port D clock enable

This bit is set and cleared by software.

0: IO port D clock disabled

1: IO port D clock enabled

Bit 2 **GPIOCEN:** IO port C clock enable

This bit is set and cleared by software.

0: IO port C clock disabled

1: IO port C clock enabled

Bit 1 **GPIOBEN:** IO port B clock enable

This bit is set and cleared by software.

0: IO port B clock disabled

1: IO port B clock enabled

Bit 0 **GPIOAEN:** IO port A clock enable

This bit is set and cleared by software.

0: IO port A clock disabled

1: IO port A clock enabled

5.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	LTDC EN	Res.	Res.	SAI2EN	SAI1EN	SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
					rw			rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSCFG EN	SPI4 EN	SPI1 EN	SDMMC1 EN	ADC3 EN	ADC2 EN	ADC1 EN	Res.	Res.	USART6 EN	USART1 EN	Res.	Res.	TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Bit 16 **TIM9EN:** TIM9 clock enable

This bit is set and cleared by software.
0: TIM9 clock disabled
1: TIM9 clock enabled

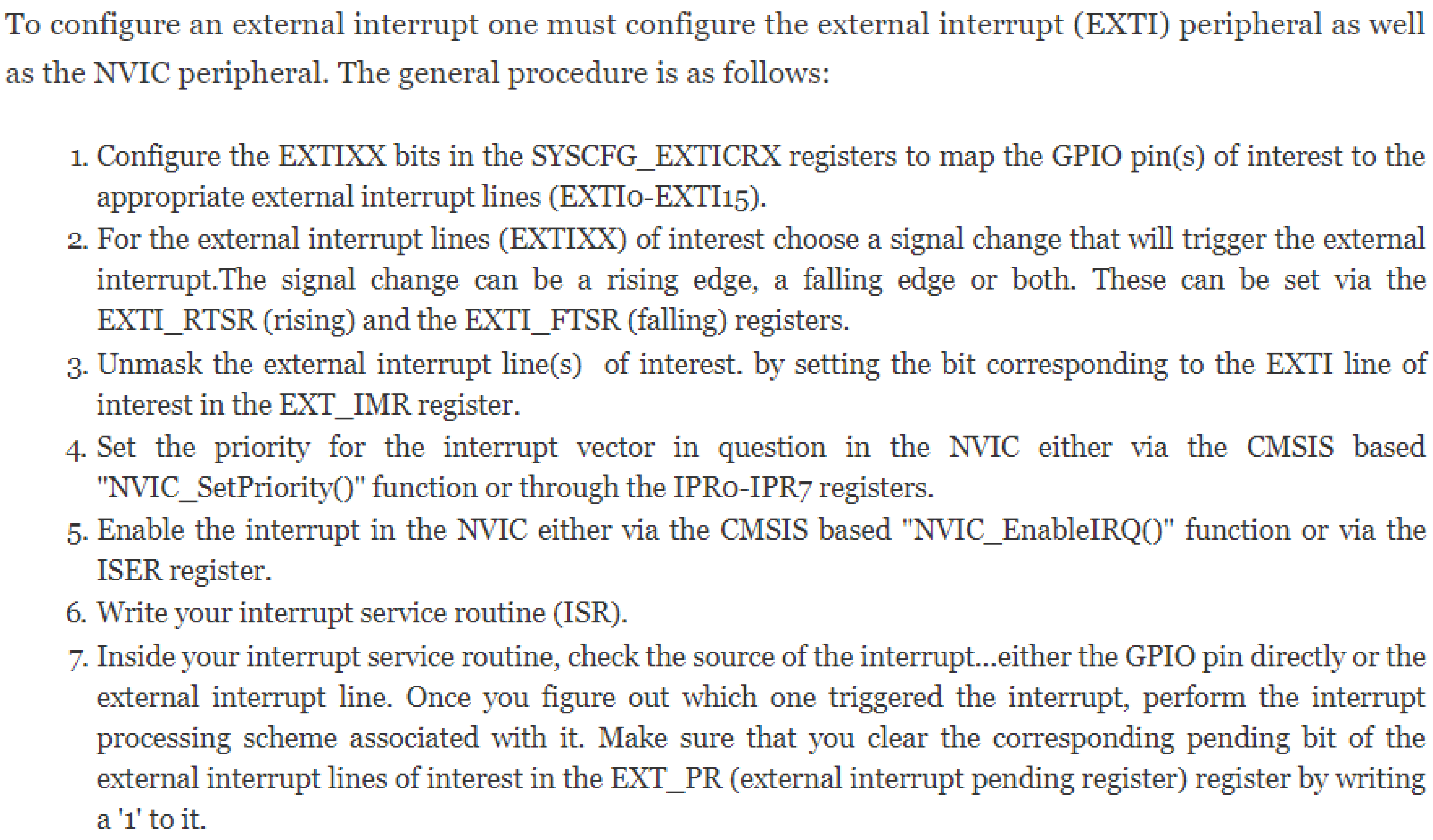
Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGEN:** System configuration controller clock enable

This bit is set and cleared by software.
0: System configuration controller clock disabled
1: System configuration controller clock enabled

Bit 13 **SPI4EN:** SPI4 clock enable

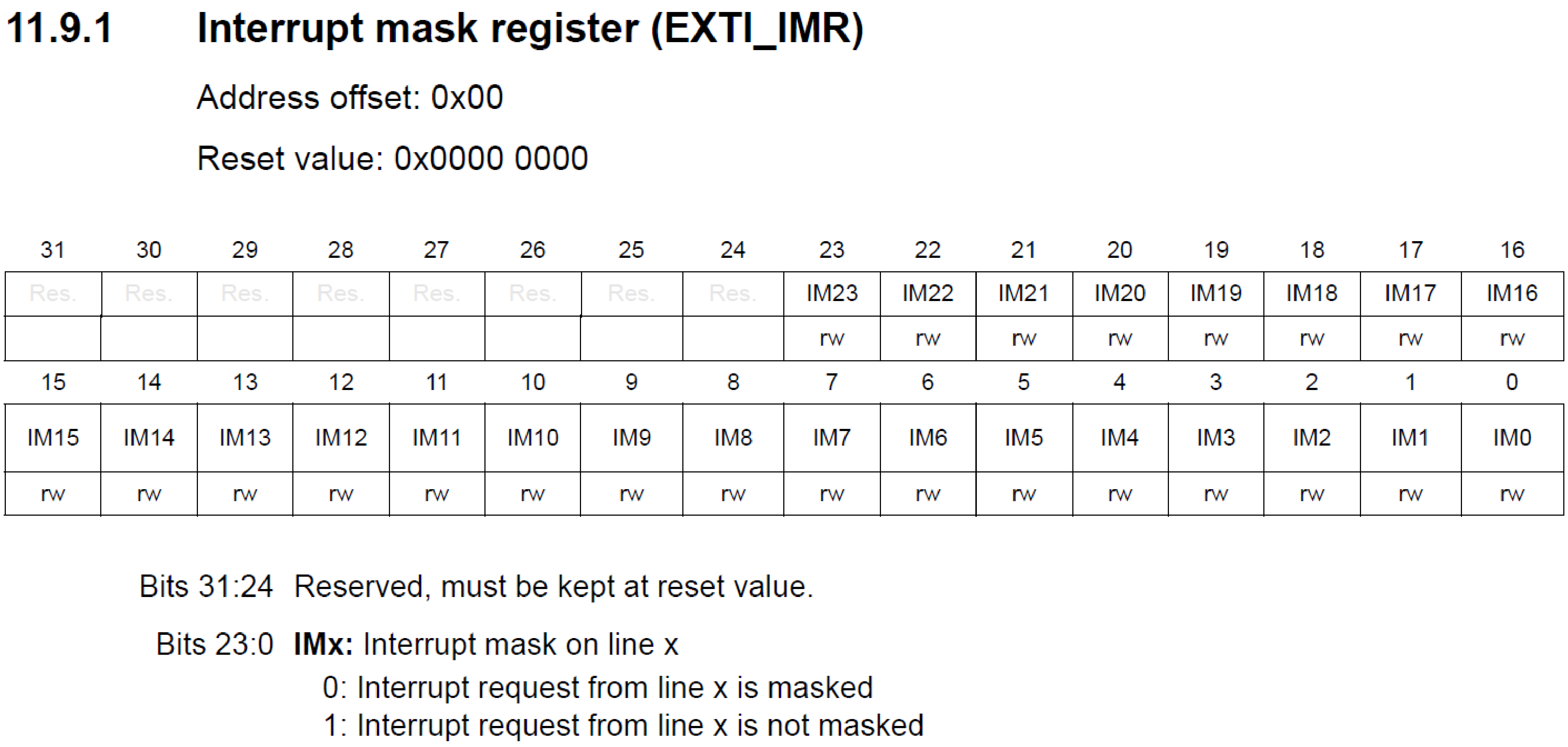
This bit is set and cleared by software.
0: SPI4 clock disabled
1: SPI4 clock enabled



To configure an external interrupt one must configure the external interrupt (EXTI) peripheral as well as the NVIC peripheral. The general procedure is as follows:

1. Configure the EXTIXX bits in the SYSCFG_EXTICRX registers to map the GPIO pin(s) of interest to the appropriate external interrupt lines (EXTI0-EXTI15).
2. For the external interrupt lines (EXTIXX) of interest choose a signal change that will trigger the external interrupt.The signal change can be a rising edge, a falling edge or both. These can be set via the EXTI_RTSR (rising) and the EXTI_FTSR (falling) registers.
3. Unmask the external interrupt line(s) of interest. by setting the bit corresponding to the EXTI line of interest in the EXT_IMR register.
4. Set the priority for the interrupt vector in question in the NVIC either via the CMSIS based "NVIC_SetPriority()" function or through the IPR0-IPR7 registers.
5. Enable the interrupt in the NVIC either via the CMSIS based "NVIC_EnableIRQ()" function or via the ISER register.
6. Write your interrupt service routine (ISR).
7. Inside your interrupt service routine, check the source of the interrupt...either the GPIO pin directly or the external interrupt line. Once you figure out which one triggered the interrupt, perform the interrupt processing scheme associated with it. Make sure that you clear the corresponding pending bit of the external interrupt lines of interest in the EXT_PR (external interrupt pending register) register by writing a '1' to it.

http://hertaville.com/external-interrupts-on-the-stm32f0.html



11.9.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **IMx:** Interrupt mask on line x

0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

11.9.2 Event mask register (EXTI_EMR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EMx:** Event mask on line x

0: Event request from line x is masked
1: Event request from line x is not masked

11.9.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR23	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **TRx:** Rising trigger event configuration bit of line x

0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

11.9.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR23	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **TRx:** Falling trigger event configuration bit of line x

0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

11.9.6 Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

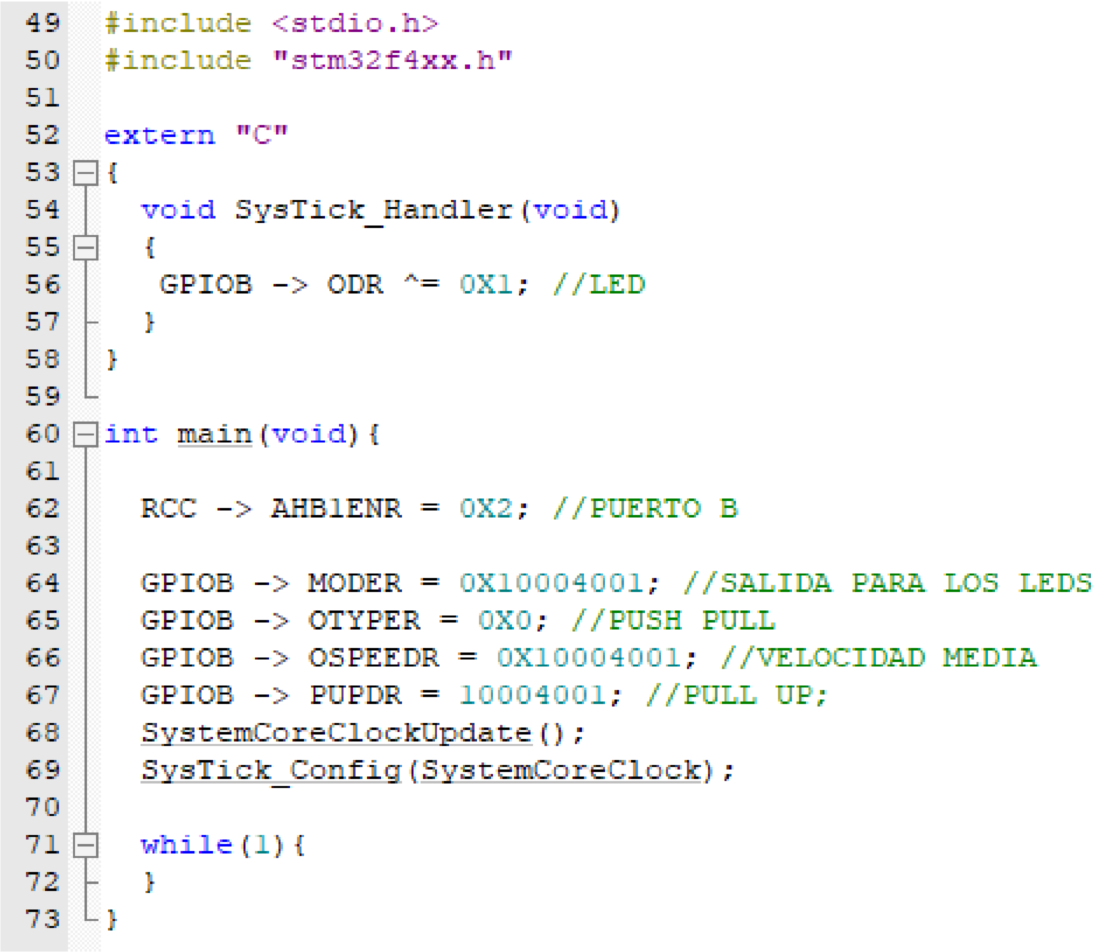
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR23	PR22	PR21	PR20	PR19	PR18	PR17	PR16
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PRx:** Pending bit

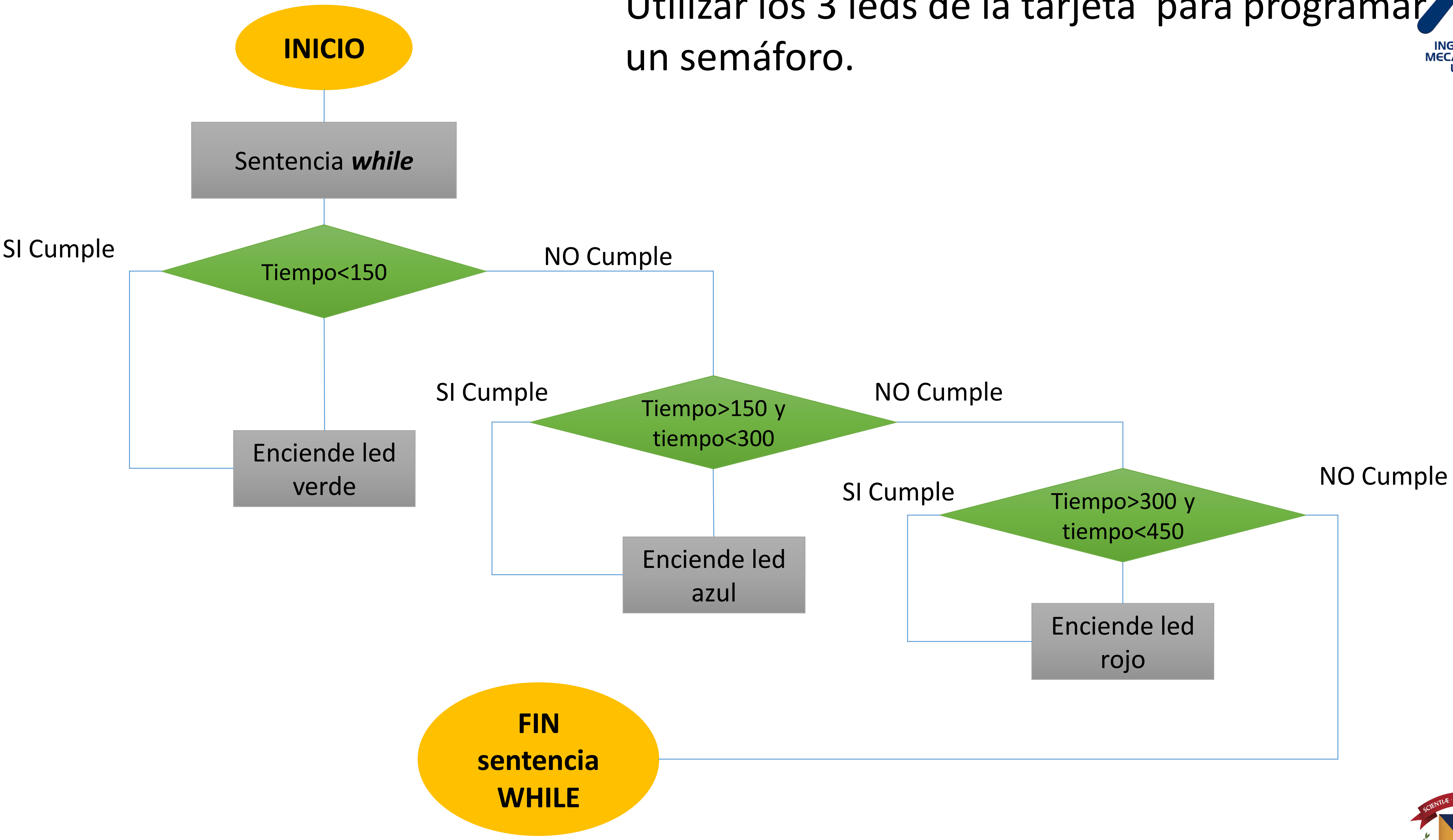
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line.
This bit is cleared by programming it to ‘1’.



```
#include <stdio.h>
#include "stm32f4xx.h"

extern "C"
{
  void SysTick_Handler(void)
  {
   GPIOB -> ODR ^= 0X1; //LED
  }
}

int main(void){

  RCC -> AHB1ENR = 0X2; //PUERTO B

  GPIOB -> MODER = 0X10004001; //SALIDA PARA LOS LEDS
  GPIOB -> OTYPER = 0X0; //PUSH PULL
  GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
  GPIOB -> PUPDR = 10004001; //PULL UP;
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);

  while(1){
  }
}
```

Utilizar los 3 leds de la tarjeta para programar un semáforo.

INGENIERÍA MECATRÓNICA UMNG



INICIO

Sentencia ***while***

Tiempo<150

SI Cumple

NO Cumple

Enciende led verde

Tiempo>150 y tiempo<300

SI Cumple

NO Cumple

Enciende led azul

Tiempo>300 y tiempo<450

SI Cumple

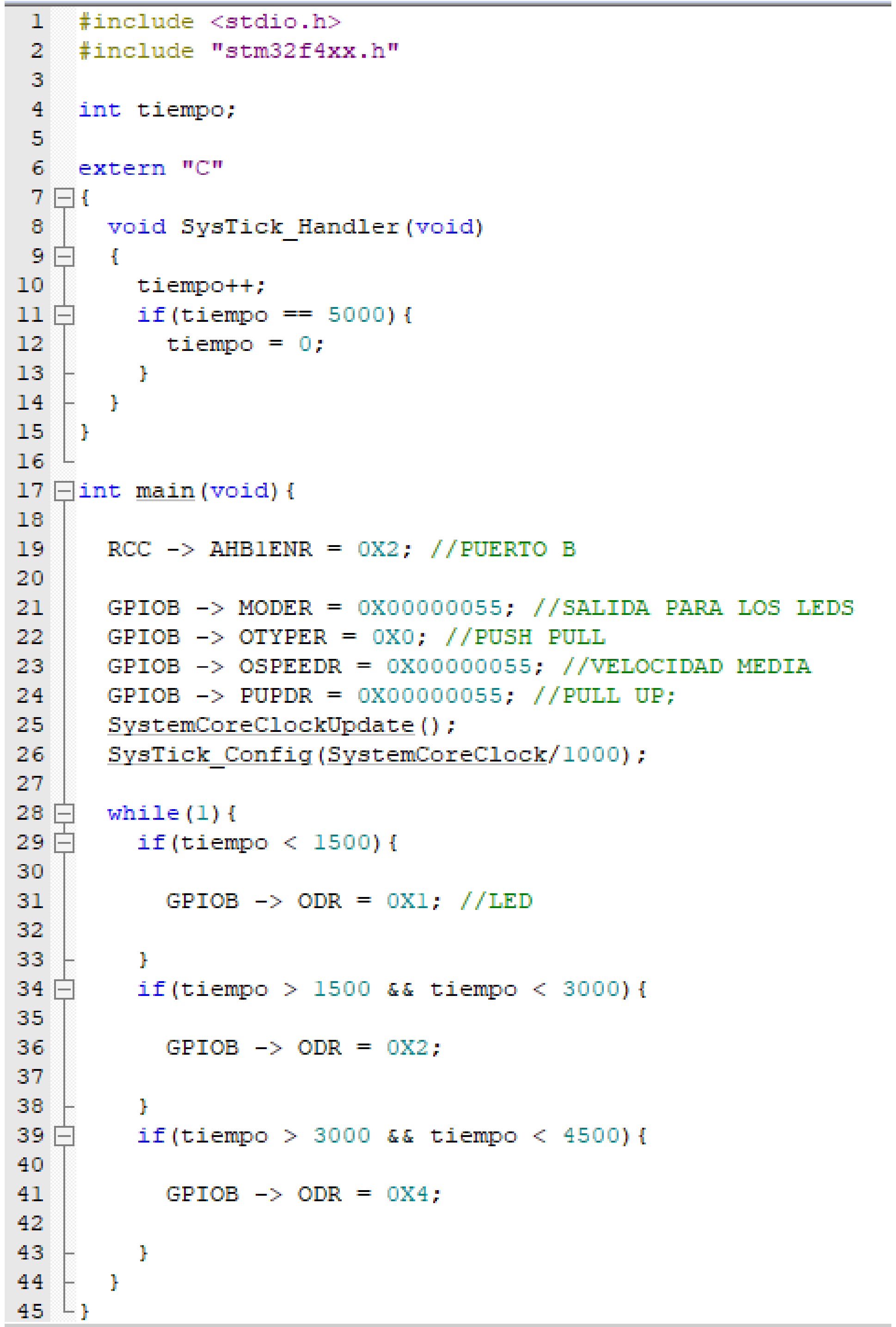
NO Cumple

Enciende led rojo

FIN sentencia WHILE

SCIENTIÆ · PATRIÆ · FAMILIÆ

UNIVERSIDAD MILITAR NUEVA GRANADA



```
#include <stdio.h>
#include "stm32f4xx.h"

int tiempo;

extern "C"
{
  void SysTick_Handler(void)
  {
    tiempo++;
    if(tiempo == 5000){
      tiempo = 0;
    }
  }
}

int main(void){

  RCC -> AHB1ENR = 0X2; //PUERTO B

  GPIOB -> MODER = 0X00000055; //SALIDA PARA LOS LEDS
  GPIOB -> OTYPER = 0X0; //PUSH PULL
  GPIOB -> OSPEEDR = 0X00000055; //VELOCIDAD MEDIA
  GPIOB -> PUPDR = 0X00000055; //PULL UP;
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock/1000);

  while(1){
    if(tiempo < 1500){

      GPIOB -> ODR = 0X1; //LED

    }
    if(tiempo > 1500 && tiempo < 3000){

      GPIOB -> ODR = 0X2;

    }
    if(tiempo > 3000 && tiempo < 4500){

      GPIOB -> ODR = 0X4;

    }
  }
}
```

Utilizar los 3 leds de la tarjeta para programar un semáforo. ¿Cuál es el tiempo de encendido para cada led para que parezca un semáforo?