UNIVERSIDAD NACIONAL DE COLOMBIA

Segundo proyecto
Interrupción externa

Alejandro Ramírez Jaramillo

Universidad Nacional de Colombia sede Manizales Email: alramirezja@unal.edu.co

Introducción

La interrupción externa es una herramienta del microcontrolador que le permite reaccionar a eventos externos. En muchas aplicaciones es necesario que el microcontrolador ejecute ciertas acciones (código) en respuesta a un cambio en una de sus entradas en el mismo instante en que sucede, para lo cual se usan las interrupciones externas. Estas interrupciones detienen la ejecución del código actual para ejecutar una sección específica de código como respuesta a una entrada predeterminada, que para este caso va a ser un filo de subida generado por un botón. Para este proyecto se hará uso de el botón 1 de la tarjeta STM32L476 que esta conectado al pin 13 del puerto C y el led 2 conectado al pin 5 del puerto A. Este ejemplo fue diseñado para una tarjeta STM32L476.

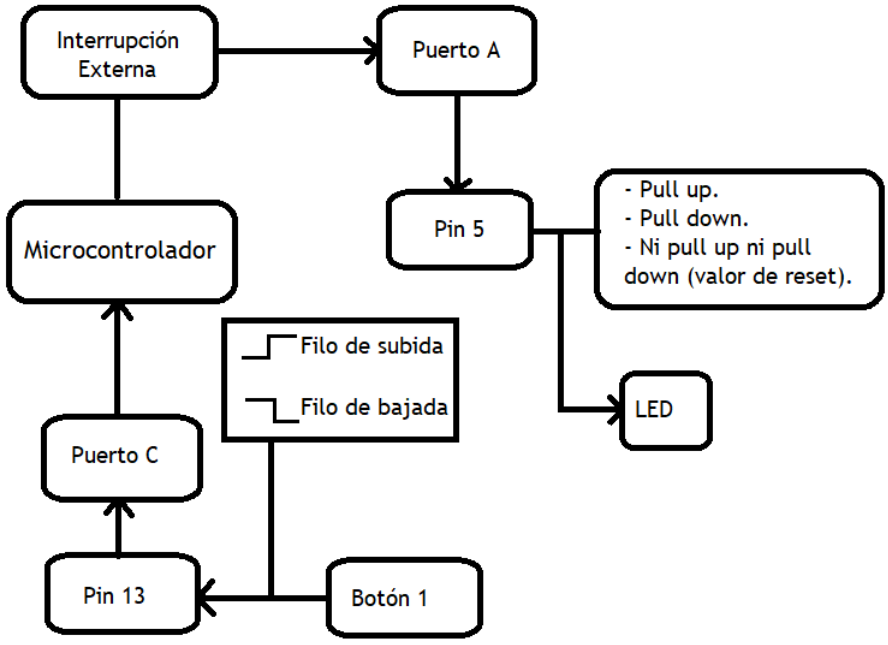


Figure 1. Diagrama de bloques interrupción externa

Contenido

1 Registros

+ RCC address: 0x4002 1000
+ GPIOA address: 0x4800 0000
+ GPIOC address: 0x4800 0800
+ SYSCFG address: 0x4001 0000
+ EXTI address: 0x4001 0400
+ NVIC address:

Address	Name	Type	Required privilege	Reset value	Description
0xE000E100-0xE000E11F	NVIC_ISER0-NVIC_ISER7	RW	Privileged	0x00000000	*Table 4.3.2: Interrupt set-enable register x (NVIC_ISERx) on page 210*
0XE000E180-0xE000E19F	NVIC_ICER0-NVIC_ICER7	RW	Privileged	0x00000000	*Table 4.3.3: Interrupt clear-enable register x (NVIC_ICERx) on page 211*
0XE000E200-0xE000E21F	NVIC_ISPR0-NVIC_ISPR7	RW	Privileged	0x00000000	*Table 4.3.4: Interrupt set-pending register x (NVIC_ISPRx) on page 212*
0XE000E280-0xE000E29F	NVIC_ICPR0-NVIC_ICPR7	RW	Privileged	0x00000000	*Table 4.3.5: Interrupt clear-pending register x (NVIC_ICPRx) on page 213*
0xE000E300-0xE000E31F	NVIC_IABR0-NVIC_IABR7	RW	Privileged	0x00000000	*Table 4.3.6: Interrupt active bit register x (NVIC_IABRx) on page 214*
0xE000E400-0xE000E4EF	NVIC_IPR0-NVIC_IPR59	RW	Privileged	0x00000000	*Table 4.3.7: Interrupt priority register x (NVIC_IPRx) on page 215*

Figure 2. Direcciones NVIC, PM0214 Programming Manual, Pag. 208

La inicialización se hace sumando a la dirección del primer registro (RCC, GPIOA, GPIOC, etc.) el valor de offset del segundo (ODR,MODER, AHB2ENR, etc.).

1.1 Inicialización de los relojes de los periféricos

1.1.1 RCC_AHB2ENR

+ Adress offset: 0x4C
+ 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.	RNGEN	HASHEN	AESEN (1)
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 3. Distribución bits AHB2ENR, RM0351 Reference Manual, Pag. 251

Para el ejercicio el usuario usará el led 1 que se encuentra en la tarjeta, que está conectado al pin 5 del puerto A y el interruptor conectado al pin 13 del puerto C. Teniendo en cuenta esto, cargue al registro el valor correspondiente para activar el reloj de los dos puertos.

1.1.2 RCC_APB2ENR

Registro usado para activar el reloj de periféricos distintos a los puertos. En este ejercicio se hará uso de SYSCFG por lo cual se debe activar el reloj para este con el bit SYSCFGEN.

+ Adress offset: 0x60
+ 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.	DFSDM1EN	Res.	SAI2EN	SAI1EN(1)	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
							rw		rw	rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	SDMMC1EN	Res.	Res.	FWEN	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	rw	rw	rw	rw	rw			rs							rw

1. Not available on STM32L41xxx/42xxx devices.

Figure 4. Distribución bits APB2ENR, RM0351 Reference Manual, Pag. 258

1.2 Configuración de entradas y salidas

1.2.1 GPIOx_MODER

Con este registro establezca el pin 5 del puerto A como una salida y el pin 13 del puerto C como entrada.

+ Adress offset: 0x00
+ Reset value puerto A: 0xABFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	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
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5. Distribución bits MODER, RM0351 Reference Manual, Pag. 303

Modos:
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode

1.2.2 GPIOx_ODR

+ Adress offset: 0x14
+ Reset value puerto A: 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
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 6. Distribución bits ODR, RM0351 Reference Manual, Pag. 305

Para poner en HIGH un pin específico basta con escribir un uno en el bit cuya posición corresponda con el número del pin.

1.2.3 GPIOx_IDR

Este registro contiene el valor de los pines de entrada del microcontrolador, con el se leen las entradas. Los valores se encuentran en los primeros 16 bits (0-15) los cuales pueden ser leídos pero no escritos, la posición de cada bit corresponde con el número del pin.

+ Adress offset: 0x10
+ Reset value: 0x0000 XXXX

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
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Figure 7. Distribución bits IDR, RM0351 Reference Manual, Pag. 305

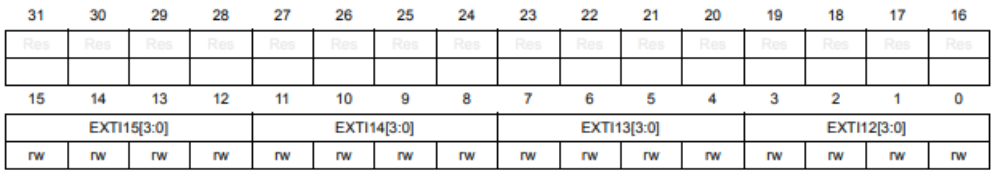
Para hacer la lectura cargue el valor del registro IDR en otro registro.

1.3 Configuración de la interrupción externa

1.3.1 SYSCFG_EXTICR4

External interrupt configuration register 4, con este registro se escoge la fuente (entrada) que se usará para la interrupción. Existen CR1, CR2, CR3 y CR4, cada uno corresponde a un grupo de pines distintos y tienen offsets diferentes. El registro EXTICR4 corresponde a los pines: 12, 13, 14 y 15 de todos los puertos. Los bits se agrupan de a 4 y dependiendo del valor del grupo de bits se elige el puerto al cual pertenece el pin de entrada.

+ Adress offset: 0x14
+ Reset value: 0x0000 XXXX



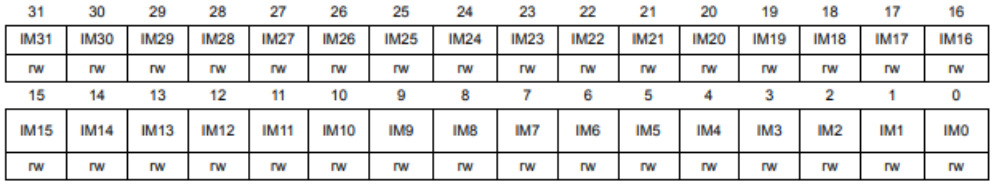
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 8. Distribución bits SYSCFG_EXTICR4, RM0351 Reference Manual, Pag. 321

EXTIY (Y=15,14,13,12):
+0000: PA[Y] pin.
+0001: PB[Y] pin.
+0010: PC[Y] pin.
+0011: PD[Y] pin.
+0100: PE[Y] pin.
+0101: PF[Y] pin.
+0110: PG[Y] pin.

1.3.2 EXTI_IMR1

Interrupt mask register, este registro determina si se hará enmascaramiento a la solicitud de interrupción proveniente de un pin específico, el pin depende de la ubicación del bit en el registro (0-15), por ejemplo para enmascarar la solicitud de interrupción del pin 13 se escribe un cero en el bit 13.
+ Adress offset: 0x00
+ Reset value: 0xFF82 0000



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	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
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

Figure 9. Distribución bits EXTI_IMR1, RM0351 Reference Manual, Pag. 405

+ 0: Enmascarar la solicitud de interrupción.
+ 1: No enmascarar la solicitud de interrupción.

1.3.3 EXTI_RTSR1

Rising trigger selection register 1. La solicitud de interrupción se da cuando en una entrada se detecta un cambio, sin embargo este cambio puede ser un flanco de subida o un flanco de bajada, por lo tanto se hace necesario escoger si uno o ambos de ellos indican una solicitud de interrupción. Con este registro se escoge que pines hacen una solicitud de interrupción cuando se identifica un flanco de subida.
+ Adress 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.	RT22	RT21	RT20	RT19	RT18	Res.	RT16
									rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 10. Distribución bits EXTI_RTSR1, RM0351 Reference Manual, Pag. 405

+ Bits 31:23 y bit 17 reservados, no deben ser modificados.
+ Bits 22:18 y 16:0 pueden ser sobrescritos.
- 0: Solicitud de interrupción por flanco de subida deshabilitado.
- 1: Solicitud de interrupción por flanco de subida habilitado.

1.3.4 EXTI_PR1

Pending register 1. Este registro funciona como una bandera que indica que esta pendiente hacer la interrupción, si ese es el caso, se comienza a ejecutar la interrupción. Este registro debe limpiarse mientras se ejecuta la interrupción escribiendo un uno en el bit correspondiente al pin de entrada.
+ Adress offset: 0x14
+ Reset value: Indefinido.

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.	PIF22	PIF21	PIF20	PIF19	PIF18	Res.	PIF16
									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
PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
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

Figure 11. Distribución bits EXTI_PR1, RM0351 Reference Manual, Pag. 408

+ Bits 31:23 y bit 17 reservados, no deben ser modificados.
+ Bits 22:18 y 16:0 pueden ser sobrescritos.
- 0: No ha habido una solicitud de interrupción.
- 1: Hay una solicitud de interrupción.

1.3.5 NVIC_IPR10

Interrupt priority register. Las interrupciones dependiendo de su tipo y entrada tienen una posición y prioridad determinada, las cuales se usarán para dar prioridad a esa interrupción con el registro IPRx.
+ Adress offset: 0x400 + 0x04*x
+ Reset value: 0x0000 0000

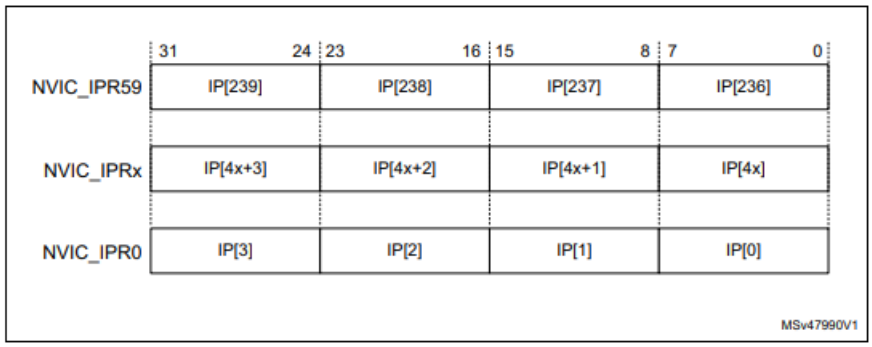


Figure 12. Distribución bits NVIC_IPR1, PM0214 Programming Manual, Pag. 215

38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_ALARM	RTC alarms through EXTI line 18 interrupts	0x0000 00E4
42	49	settable	DFSDM1_FLT3	DFSDM1_FLT3 global interrupt	0x0000 00E8
43	50	settable	TIM8_BRK	TIM8 Break interrupt	0x0000 00EC

Figure 13. Prioridad interrupciones, RM0351 Reference Manual, Pag. 397

Para dar prioridad a una interrupción, se carga en el registro IPRx un valor de x tal que al hacer la operación del bit IP el resultado sea la posición de la interrupción mostrada en la tabla anterior.

1.3.6 NVIC_ISERx

Interrupt set-enable register x, registro encargado de habilitar las interrupciones, dado el número de interrupciones disponibles se dividen en 8 grupos (x=0,1,2,...7).

+ Adress offset: 0x100 + 0x04*x

+ Reset value: 0x0000 0000

NVIC_ISER0 bits 0 to 31 are for interrupt 0 to 31, respectively

NVIC_ISER1 bits 0 to 31 are for interrupt 32 to 63, respectively

....

NVIC_ISER6 bits 0 to 31 are for interrupt 192 to 223, respectively

NVIC_ISER7 bits 0 to 15 are for interrupt 224 to 239, respectively

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETENA[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Figure 14. Interrupciones habilitadas, PM0214 Programming Manual, Pag. 210

+0: Interrupción deshabilitada.

+1: Interrupción habilitada.

2 Vectores

Para hacer uso de las interrupciones es necesario preparar vectores que contienen las direcciones de distintas funciones del microcontrolador, estas ya se encuentran en el código del ejercicio. A continuación se mostrará una parte de estos vectores (linea 67-77) como ejemplo, en el código los vectores van desde la linea 67-168. La interrupción que se usará en el ejercicio tendrá como entrada el pin 13 de puerto C por lo tanto necesita del manejador correspondiente (señalado en la figura 12) EXTI15_10_IRQHandler, que contendrá el código a ejecutar durante la interrupción.

```
////////////////////////////////////////////////////////////////////////////////
// Vectors
////////////////////////////////////////////////////////////////////////////////
// Add all other processor specific exceptions/interrupts in order here
        .word     __StackTop                  // Top of the stack. from linker script
        .word     _main +1                   // reset location, +1 for thumb mode
        .word     NMI_Handler + 1
    .word     HardFault_Handler  + 1
    .word     MemManage_Handler + 1
    .word     BusFault_Handler + 1
    .word     UsageFault_Handler + 1
```

Los manejadores se encuentran al final del código de la interrupción externa, la mayoría de estos están vacíos lo que implica que no se ejecutara nada para esas interrupciones, exceptuando EXTI15_10_IRQHandler (linea 284-302).

```
EXTI15_10_IRQHandler:                   /* EXTI Lines 10 to 15 interrupts
*/
        ldr r6, = EXTI_PR1
        ldr r5, [r6]
        cmp r5, 0x2000
        beq     int
        bx lr

int:                                                        // Set GPIOA Pin5 to 1 (bit 5 in ODR register)
        ldr r6, = GPIOA_ODR             // Load GPIOA output data register
        ldr r5, [r6]                    // Read its content to r5
        eor r5, 0x0020                  // write 1 to pin 5
        str r5, [r6]                    // Store result in GPIOA output data register

        ldr r7, = EXTI_PR1
        ldr r8, [r7]
        orr r8, 0x2000
        str r8, [r7]                                    //Clear any pending event on EXTI

        bx lr
```

3 Código External-interrupt

El código que se estudia encenderá y apagará el led que se esta conectado al pin 5 del puerto A cada vez que se presione el interruptor del pin 13 del puerto C haciendo uso de interrupciones externas.

3.1 Inicialización

Se inicializan los registros usando los valores de offset y direcciones que se encuentran en la sección Registros.

```
// Register Addresses
// You can find the base addresses for all peripherals from Memory Map section
// RM0351 on page 78. Then the offsets can be found on their relevant sections.

//       RCC   base address is 0x40021000
//   AHB2ENR register offset is 0x4C
.equ     RCC_AHB2ENR,   0x4002104C // RCC AHB2 peripheral clock reg (page 251)

//       RCC   base address is 0x40021000
//   APB2ENR register offset is 0x60
.equ     RCC_APB2ENR,   0x40021060 // RCC APB2 peripheral clock reg (page 258)

//       GPIOA base address is 0x48000000
//   MODER register offset is 0x00
//   ODR   register offset is 0x14
.equ     GPIOA_MODER,   0x48000000 // GPIOA port mode register (page 303)
.equ     GPIOA_ODR,     0x48000014 // GPIOA output data register (page 305)

//       GPIOC base address is 0x48000800
//   MODER register offset is 0x00
//   IDR   register offset is 0x10
.equ     GPIOC_MODER,   0x48000800 // GPIOC port mode register (page 303)
.equ     GPIOC_IDR,     0x48000810 // GPIOC output data register (page 305)

//       SYSCFG base address is 0x40010000
//   EXTICR4 register offset is 0x14
.equ     SYSCFG_EXTICR4,   0x40010014 // SYSCFG port mode register (page 321)

//       EXTI base address is 0x40010400
//   IMR1 register offset is 0x00
//   RTSR1 register offset is 0x08
//   PR1 register offset is 0x14
.equ     EXTI_IMR1,    0x40010400  // IMR port mode register (page 405)
.equ     EXTI_RTSR1,   0x40010408  // RTSR port mode register (page 405)
.equ     EXTI_PR1,     0x40010414  // PR1 port mode register (page 408)

.equ     NVIC_IPR10,   0xE000E428
.equ     NVIC_ISER1,   0xE000E104
```

3.2 Vectores

Se crean los vectores, con la dirección del vector +1 (el +1 es por el modo thumb).

```
        .word     __StackTop                  // Top of the stack. from linker script
        .word     _main +1                   // reset location, +1 for thumb mode
        .word     NMI_Handler + 1
    .word     HardFault_Handler  + 1
    .word     MemManage_Handler + 1
    .word     BusFault_Handler + 1
    .word     UsageFault_Handler + 1
        .word    0
        .word    0
        .word    0
        .word    0
        .word    SVC_Handler +1
        .word    DebugMon_Handler +1
        .word    0
```

Debido a la longitud no se incluyeron todas las lineas de vectores, la cuales se encuentran desde la linea 71 hasta la linea 168.

3.3 Configuración

Usando el registro AHB2ENR active el reloj para el puerto A y el puerto C:

```
ldr r6, = RCC_AHB2ENR       // Load peripheral clock reg address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x5                         // Set bit 0 to enable GPIOA and GPIOC clock
str r5, [r6]                // Store result in peripheral clock register
```

Configure el pin 5 del puerto A como salida y el pin 13 del puerto C como entrada:

```
ldr r6, = GPIOC_MODER       // Load GPIOA MODER register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0xF3FFFFFF          // Write 00 to bits 27, 26 for P13
str r5, [r6]                // Store result in GPIOA MODER register

ldr r6, = GPIOA_MODER       // Load GPIOA MODER register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0xFFFFF7FF          // Write 01 to bits 11, 10 for P5
str r5, [r6]                // Store result in GPIOA MODER register
```

Active el reloj de control de SYSCFG:

```
ldr r6, = RCC_APB2ENR       // Load peripheral clock reg address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                 // Set bit 0 to enable SYSCFG clock
str r5, [r6]                // Store result in peripheral clock register
```

Usando EXTICR4 seleccione el pin 13 del puerto C como la entrada de la interrupción:

```
ldr r6, = SYSCFG_EXTICR4    // Load SYSCFG EXTICR4 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x20                    // Write 0010 to bits 7, 6, 5, 4 for P13
str r5, [r6]                // Store result in SYSCFG EXTICR4 register
```

No se va a enmascarar la solicitud de interrupción, configure el registro IMR1 para que esto se cumpla:

```
ldr r6, = EXTI_IMR1             // Load EXTI IMR1 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x2000                          // Set 1 to bits 13 for P13
str r5, [r6]                // Store result in EXTI IMR1 register
```

Use el registro RTSR1 para que la interrupción se active cuando en la entrada se detecte un flanco de subida:

```
ldr r6, = EXTI_RTSR1            // Load EXTI RTSR1 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x2000                  // Write 1 to bits 13 for P13
str r5, [r6]                // Store result in EXTI RTSR1 register
```

Habilite la interrupcion 40 usando el registro ISERx, identifique a cual de los ocho grupos pertenece y el bit que le corresponde:

```
ldr r6, = NVIC_ISER1
ldr r5, [r6]
orr r5, 0x100
str r5, [r6]
```

De prioridad a la interrupción que se esta usando, a traves su posición entre las interrupciones y el registro IPRx:

```
        ldr r6, = NVIC_IPR10
        ldr r5, [r6]
        orr r5, 0x10
        str r5, [r6]
```

Use CPSIE i para habilitar las interrupciones globales.

3.4 Interrupción

Después de configurar la interrupción, se programa el código que se ejecutará cuando se active la interrupción, con el nombre EXTI15_10_IRQHandler (ya establecido en la sección de vectores).
En primer lugar se lee el registro PR1 para revisar si hay una solicitud de interrupción, comparando el valor en el registro (bit 13) con un valor predeterminado, que en el caso de ser iguales, procede con la interrupción.

```
EXTI15_10_IRQHandler:                   /* EXTI Lines 10 to 15 interrupts
*/
        ldr r6, = EXTI_PR1
        ldr r5, [r6]
        cmp r5, 0x2000
        beq     int
        bx lr
```

Entonces se lee el registro ODR para saber que valor se encuentra en los pines de salida, se aplica XOR a ese valor con un valor de 0x20 (0b00100000), un XOR entre cualquier valor x y un 1, va a revertir el valor de x. Si en el bit 5 de ODR se encontraba un 1, al aplicar XOR el valor del bit 5 se volverá 0 y viceversa. Esta sección de código hace que cada vez que se ejecuta la interrupción el led se encienda o apague.

```
int:                                                    // Set GPIOA Pin5 to 1 (bit 5 in ODR register)
        ldr r6, = GPIOA_ODR             // Load GPIOA output data register
        ldr r5, [r6]                    // Read its content to r5
        eor r5, 0x0020                  // write 1 to pin 5
        str r5, [r6]                    // Store result in GPIOA output data register
```

Para finalizar la interrupción, limpie el bit 13 del registro de solicitud de interrupción (PR1) poniendo un 1 en esa posición.

```
int:                                                    // Set GPIOA Pin5 to 1 (bit 5 in ODR register)
        ldr r6, = GPIOA_ODR             // Load GPIOA output data register
        ldr r5, [r6]                    // Read its content to r5
        eor r5, 0x0020                  // write 1 to pin 5
        str r5, [r6]                    // Store result in GPIOA output data register
```

4 Ejercicios prácticos

1. Modifique el código anterior para que la interrupción externa solo se active con el filo de bajada producido por el botón al ser presionado, para esto use el registro EXTI_FTSR1 (Página 406, RM0351 Reference manual).
2. Cree dos secuencias de luces con leds y utilice la interrupción externa para cambiar entre una secuencia y otra cada vez que se oprima el botón B1.

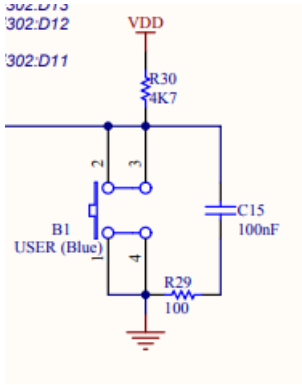


Figure 15. Conexiones internas botón 1, MB1136 Schematic board, Pag. 64

5 Referencias

+ PM0214 Programming manual, STM32 Cortex-M4 MCUs and MPUs programming manual.
+ RM0351 Reference manual, STM32L4x5 and STM32L4x6 advanced Arm-based 32-bit MCUs.
+ fMB1136 Schematic board.