Diseño Basado en Microprocesadores

Tema 2. Microcontroladores

- 2.1. Introducción a los microcontroladores
- 2.2. Entradas/Salidas Digitales
- 2.3. Temporizadores
- 2.4. Excepciones
- 2.5. Conversión Analógica/Digital
- 2.6. Comunicación serie RS232C
- 2.7. Teclado, conversión D/A y sonido
- 2.8. Interfaz I2C

2.3. Excepciones

2.3.1. Conceptos sobre excepciones
2.3.2. Manejo de las excepciones
2.3.3. Tipos de excepciones
2.3.4. Tabla de vectores de excepción
2.3.5. Excepciones de interrupción
2.3.6. Jerarquía de excepciones
2.3.7. Atención a dispositivos
2.3.8. Controlador de interrupciones vectorizado anidado
2.3.9. Diagrama del controlador NVIC
2.3.10. Fuentes de interrupción
2.3.11. Registros de interrupción NVIC
2.3.12. Estructura de programa con interrupciones
2.3.13. Manejadores (handlers) de interrupciones
2.3.14. Funciones CMSIS
2.3.15. Interrupciones en los GPIO
2.3.16. Ejemplo manejador IRQ para P0
2.3.17. Interrupciones externas específicas
2.3.18. Ejemplo manejador IRQ para EINT0/EINT1
2.3.19. Interrupciones en los timers
2.3.20. Ejemplo manejador IRQ para T0

2.3.1. Conceptos sobre excepciones

Excepción

- Llamada a la CPU para que atienda a un evento extraordinario de forma inmediata. La CPU termina la ejecución de la instrucción en curso, atiende a la excepción y vuelve a la siguiente instrucción por donde iba en el momento de producirse

Vector de excepción

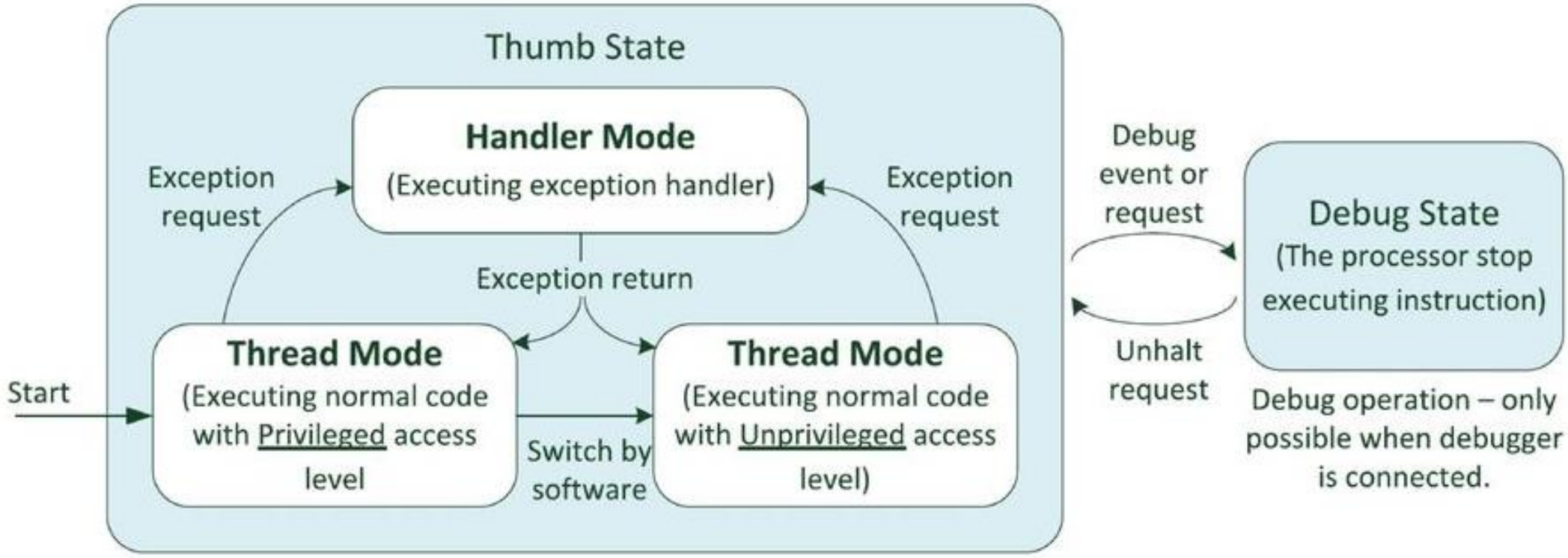
- Dirección de memoria donde comienza la rutina del servicio de excepción

Tabla de vectores de excepción

- Zona de memoria (suele ser la zona baja de memoria), donde se sitúan los vectores de excepción. Cada dispositivo tiene asociada una posición en la tabla de vectores de excepción

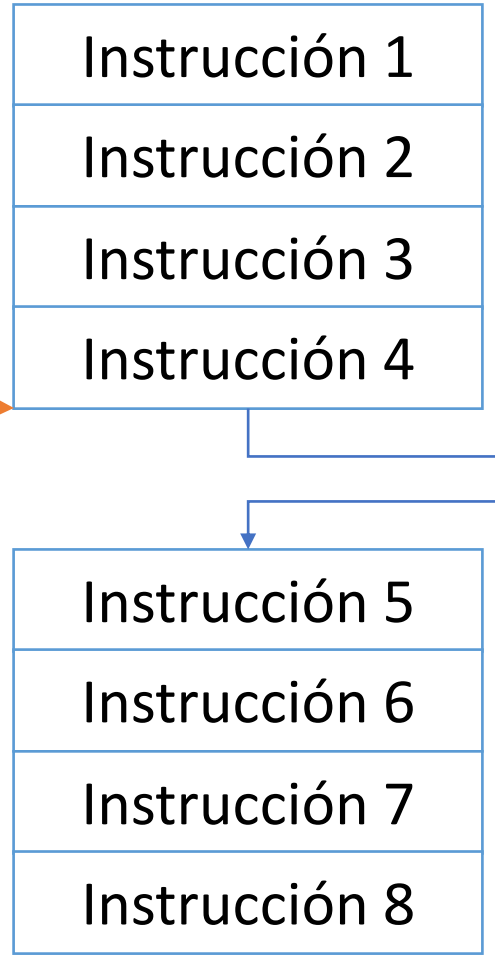
2.3.2. Manejo de las excepciones (1)

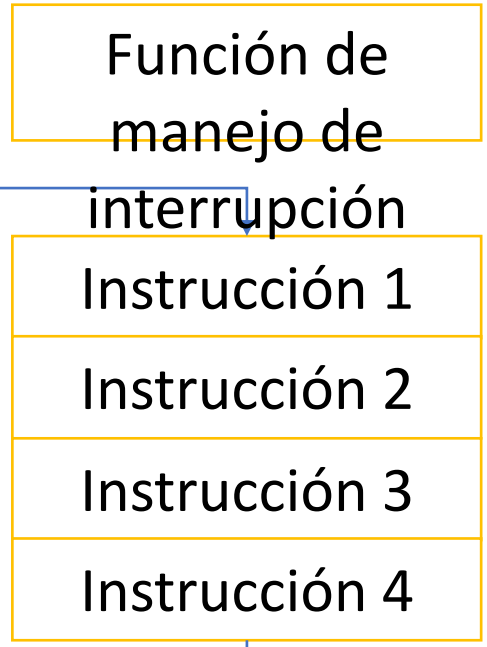
Las excepciones son circunstancias extraordinarias internas al programa que se producen en la ejecución de los mismos. Esto obliga a una variación en el flujo del programa.



2.3.2. Manejo de las excepciones (3)

1) Petición excepción
2) Excepción aceptada





Generado por Hardware:
- La CPU detiene la tarea que está ejecutando
- Se guarda dirección de retorno
- La CPU ejecuta la rutina de excepción o manejador correspondiente
- Una vez terminada la rutina de excepción, la CPU continua por donde iba.

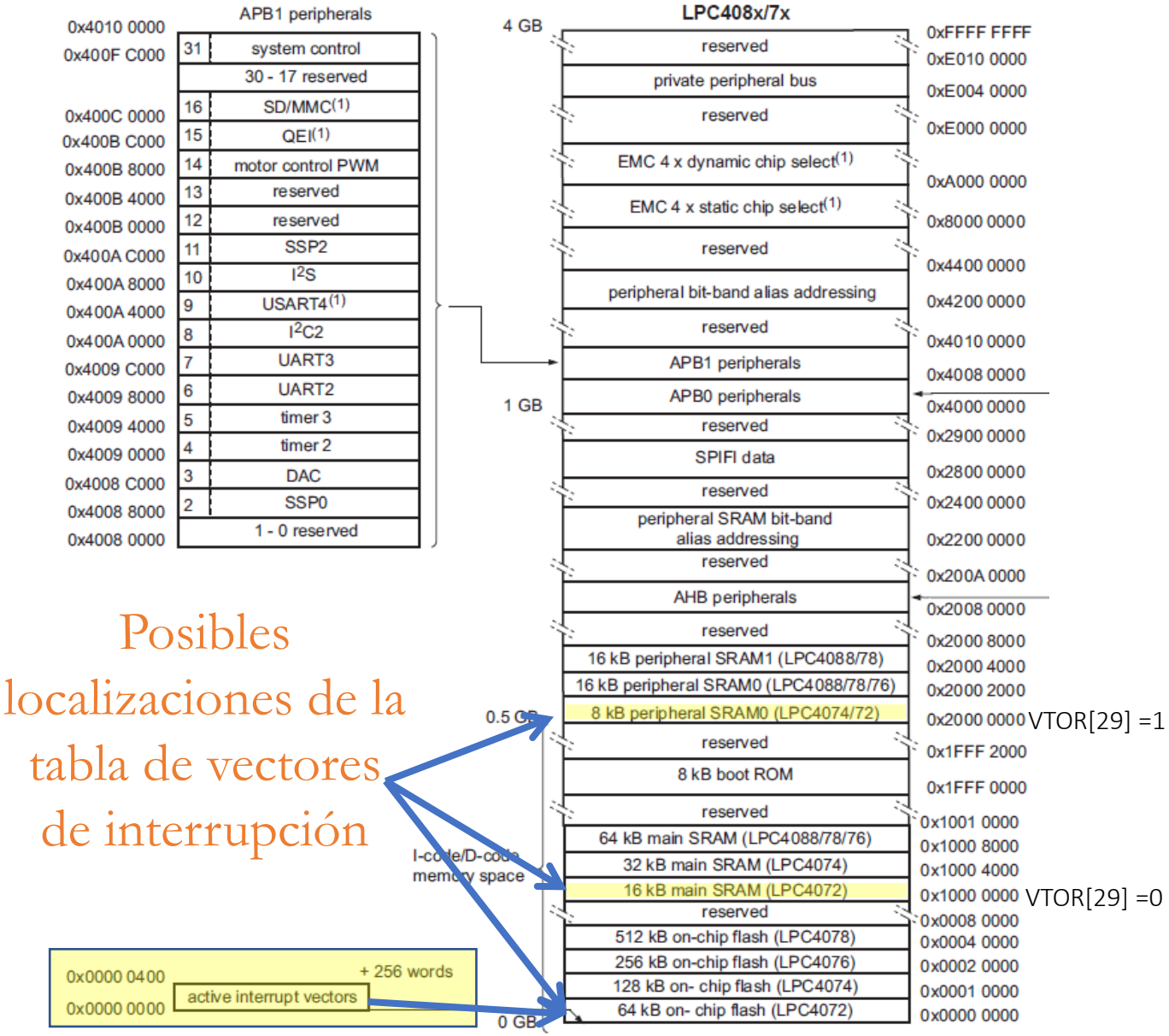
2.3.3. Tipos de excepciones

```
                AREA    RESET, DATA, READONLY
                EXPORT      __Vectors
__Vectors   DCD     __initial_s ; Top of Stack / Dirección extremo superior de pila
                DCD         Reset_Handler           ; 0: Reset Handler / Manejador de reset
            DCD     NMI_Handler             ; 1: NMI Handler / Manejador de interrup. no enmascarables
            DCD     HardFault_Handler       ; 2: Hard Fault Handler / Manejador de fallos de hardware
            DCD     MemManage_Handler ; 3: MPU Fault Handler / Manejador de fallos de memoria
            DCD     BusFault_Handler        ; 4: Bus Fault Handler / Manejador de fallos de buses
            DCD     UsageFault_Handler      ; 5: Usage Fault Handler / Manejador de fallos
            DCD     0xEFFFF39E              ; 6: Reserved- vector sum / Valor de chequeo
            DCD     0                       ; 7: Reserved
            DCD     0                       ; 8: Reserved
            DCD     0                       ; 9: Reserved
            DCD     SVC_Handler             ; 10: SVCall Handler / Manejador de llamada al supervisor
            DCD     DebugMon_Handler                ; 11: Debug Monitor Handler / Manejador de eventos debug
            DCD     0                       ; 12: Reserved
            DCD     PendSV_Handler          ; 13: PendSV Handler / Manejador de eventos pendientes
            DCD     SysTick_Handler         ; 14: SysTick Handler / Manejador del sistema de pulsos
```

2.3.4. Tabla de vectores de excepción

- 256 excepciones: 15 internas y 239 externas
- Por defecto a partir de la dirección 0x00000000, ocupa los 256 primeros words de la memoria, hasta la dirección 0x00000400
- Cuando se produce una excepción, el procesador salta a la posición correspondiente dentro de la tabla de vectores de excepción para usar el vector de dirección de la ISR

18	Vector Interr 2	0x00000048
17	Vector Interr 1	0x00000044
16	Vector Interr 0	0x00000040
15	Vector SysTick	0x0000003C
14	Vector PendSV	0x00000038
13	No usado	0x00000034
12	Vector Debug Monit.	0x00000030
11	Vector SVC	0x0000002C
10	No usado	0x00000028
9	No usado	0x00000024
8	No usado	0x00000020
7	No usado	0x0000001C
6	Vector Usage Fault	0x00000018
5	Vector Bus Fault	0x00000014
4	Vector MenManage	0x00000010
3	Vector HardFault	0x0000000C
2	Vector NMI	0x00000008
1	Vector Reset	0x00000004
0	Valor inicial pila	0x00000000



Posibles localizaciones de la tabla de vectores de interrupción

(1) Not available on all parts. See Table 2 and Table 4.

2.3.5. Excepciones de interrupción

- Utilizadas por dispositivos/ periféricos externos que interrumpen a la CPU
- Se denominan IRQ Interrupt ReQuest (solicitud interrup.)
- Los sistemas embebidos deben responder a eventos externos
- Dos modos de atender a los eventos externos: por muestreo y por interrupciones

```
DCD     WDT_IRQHandler          ; 16: Watchdog Timer
DCD     TIMER0_IRQHandler       ; 17: Timer0
DCD     TIMER1_IRQHandler       ; 18: Timer1
DCD     TIMER2_IRQHandler       ; 19: Timer2
DCD     TIMER3_IRQHandler       ; 20: Timer3
DCD     UART0_IRQHandler        ; 21: UART0
DCD     UART1_IRQHandler        ; 22: UART1
DCD     UART2_IRQHandler        ; 23: UART2
DCD     UART3_IRQHandler        ; 24: UART3
DCD     PWM1_IRQHandler         ; 25: PWM1
DCD     I2C0_IRQHandler         ; 26: I2C0
DCD     I2C1_IRQHandler         ; 27: I2C1
DCD     I2C2_IRQHandler         ; 28: I2C2
DCD     0                       ; 29: reserved, not for SPIFI anymore
DCD     SSP0_IRQHandler         ; 30: SSP0
DCD     SSP1_IRQHandler         ; 31: SSP1
DCD     PLL0_IRQHandler         ; 32: PLL0 Lock (Main PLL)
DCD     RTC_IRQHandler          ; 33: Real Time Clock
DCD     EINT0_IRQHandler        ; 34: External Interrupt 0
DCD     EINT1_IRQHandler        ; 35: External Interrupt 1
DCD     EINT2_IRQHandler        ; 36: External Interrupt 2
DCD     EINT3_IRQHandler        ; 37: External Interrupt 3
DCD     ADC_IRQHandler          ; 38: A/D Converter
DCD     BOD_IRQHandler          ; 39: Brown-Out Detect
DCD     USB_IRQHandler          ; 40: USB
DCD     CAN_IRQHandler          ; 41: CAN
DCD     DMA_IRQHandler          ; 42: General Purpose DMA
DCD     I2S_IRQHandler          ; 43: I2S
DCD     ENET_IRQHandler         ; 44: Ethernet
DCD     MCI_IRQHandler          ; 45: SD/MMC card I/F
DCD     MCPWM_IRQHandler        ; 46: Motor Control PWM
DCD     QEI_IRQHandler          ; 47: Quadrature Encoder Interface
DCD     PLL1_IRQHandler         ; 48: PLL1 Lock (USB PLL)
DCD     USBActivity_IRQHandler  ; 49: USB Activity interrupt to wakeup
DCD     CANActivity_IRQHandler  ; 50: CAN Activity interrupt to wakeup
DCD     UART4_IRQHandler        ; 51: UART4
DCD     SSP2_IRQHandler         ; 52: SSP2
DCD     LCD_IRQHandler          ; 53: LCD
DCD     GPIO_IRQHandler         ; 54: GPIO
DCD     PWM0_IRQHandler         ; 55: PWM0
DCD     EEPROM_IRQHandler       ; 56: EEPROM
```

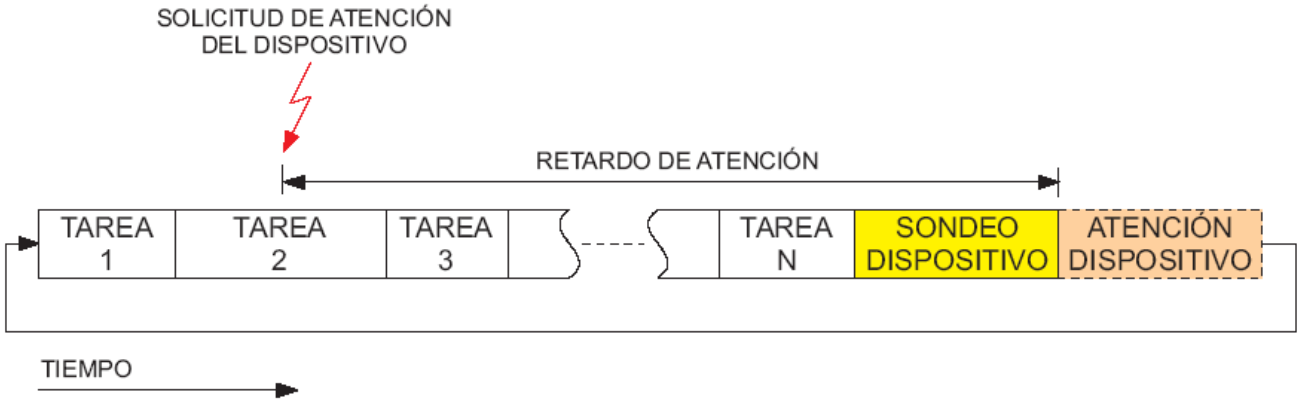
2.3.6. Jerarquía de excepciones

Nº excep.	Tipo excepción	Prioridad
1	Reset	–3 (la más alta)
2	NMI	–2
3	Hard Fault	–1
4	MemManage Fault	Programable (0..31)
5	Bus Fault	Programable (0..31)
6	Usage Fault	Programable (0..31)
7-10	Reservados	-
11	SVC	Programable (0..31)
12	Debug monitor	Programable (0..31)
13	Reservado	-
14	PendSV	Programable (0..31)
15	SYSTICK	Programable (0..31)
16	Interrupción 0	Programable (0..31)
17	Interrupción 1	Programable (0..31)
...	...	...
255	Interrupción 239	Programable (0..31)

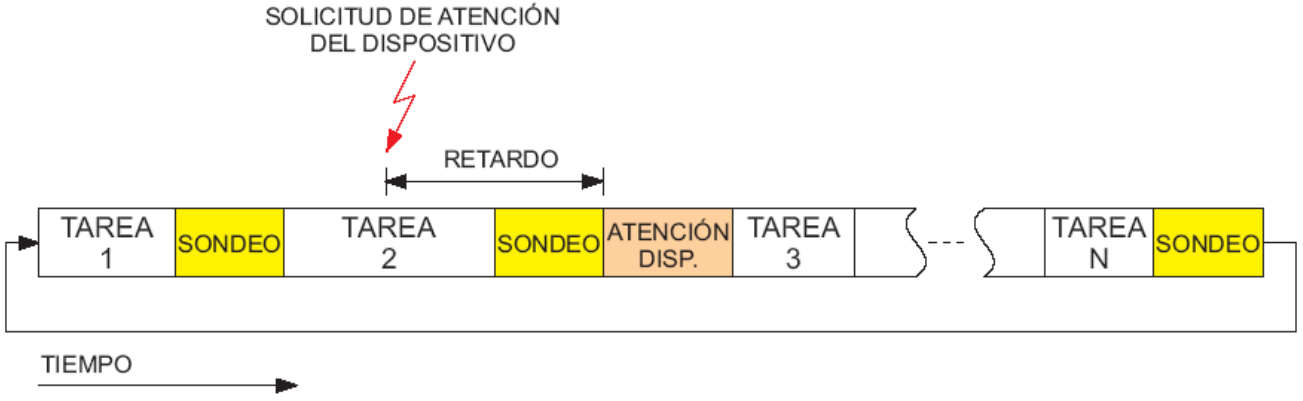
2.3.7. Atención a dispositivos (1)

MEDIANTE SONDEO (POLLING)

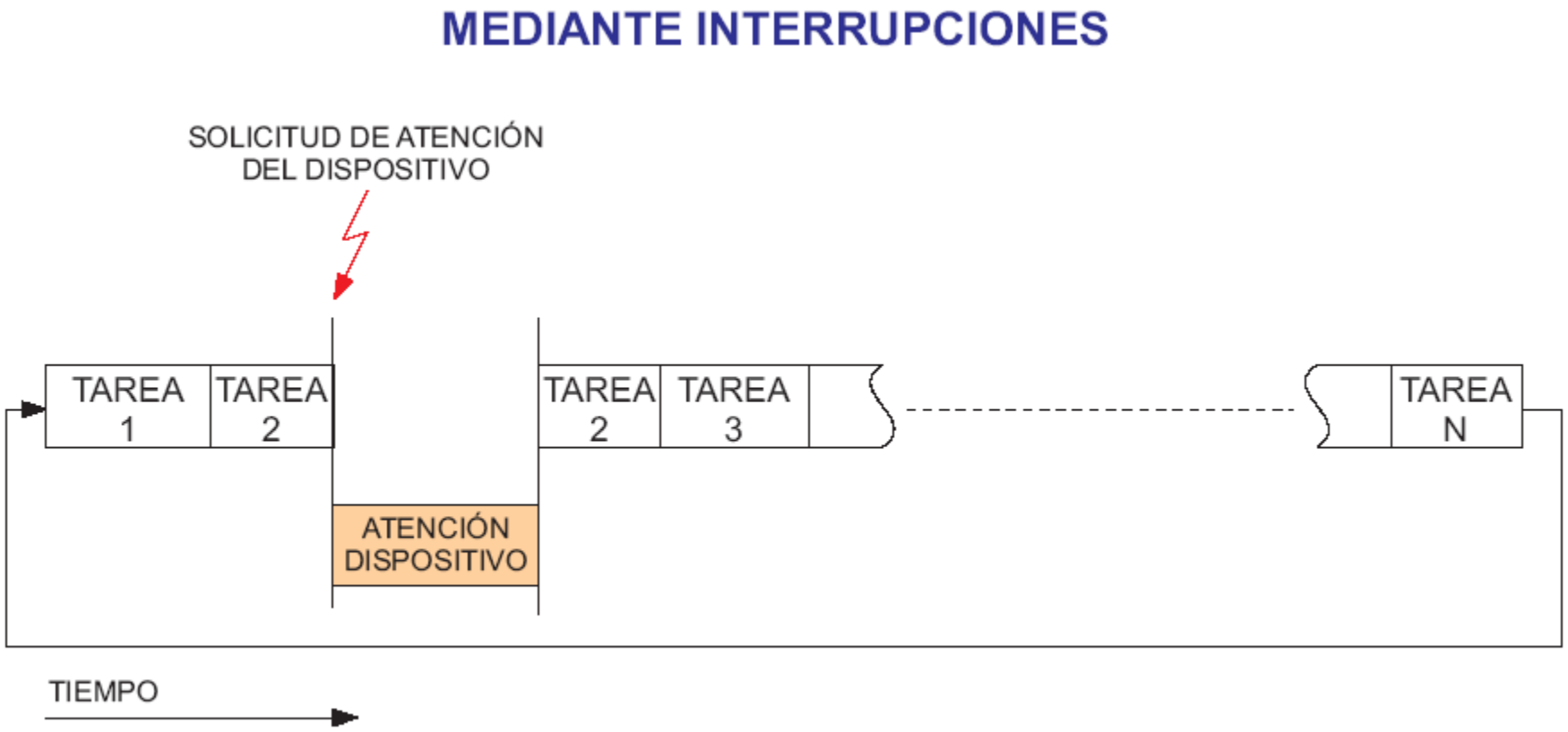
UN SOLO SONDEO EN EL BUCLE PRINCIPAL



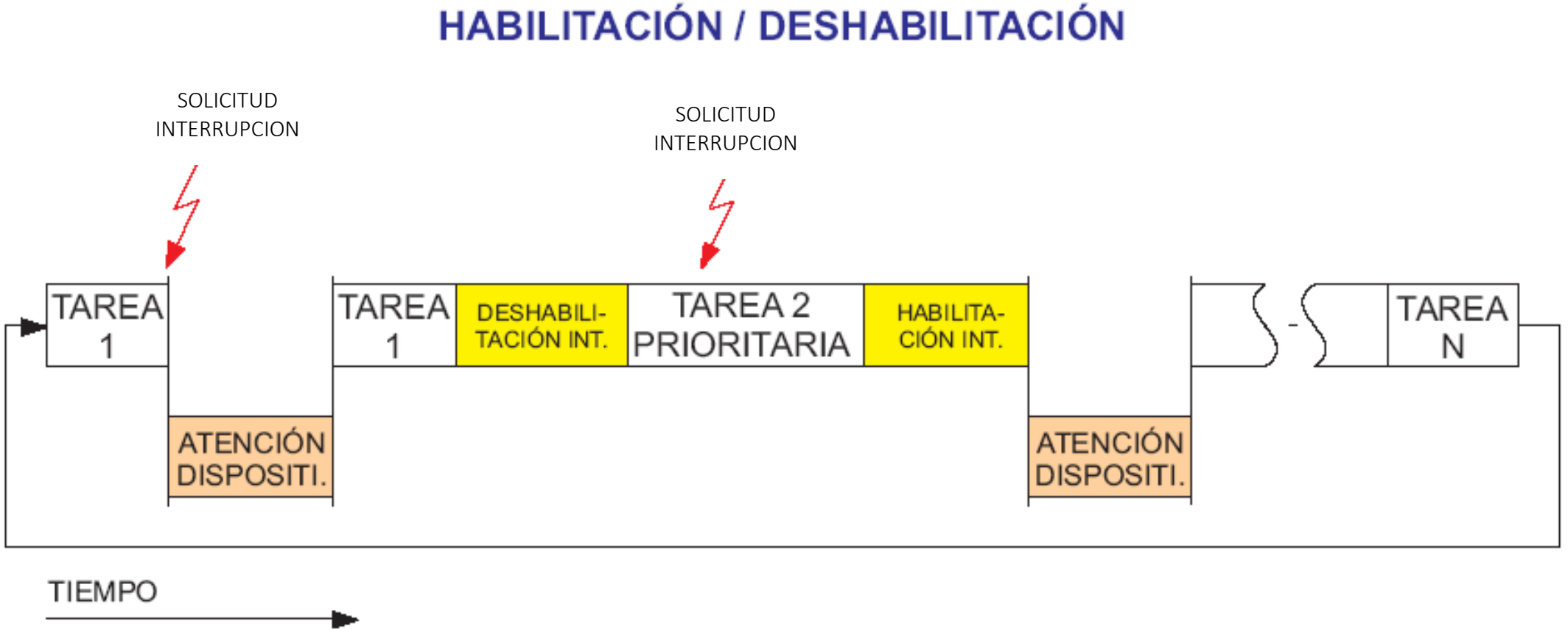
MÚLTIPLES SONDEOS EN EL BUCLE PRINCIPAL



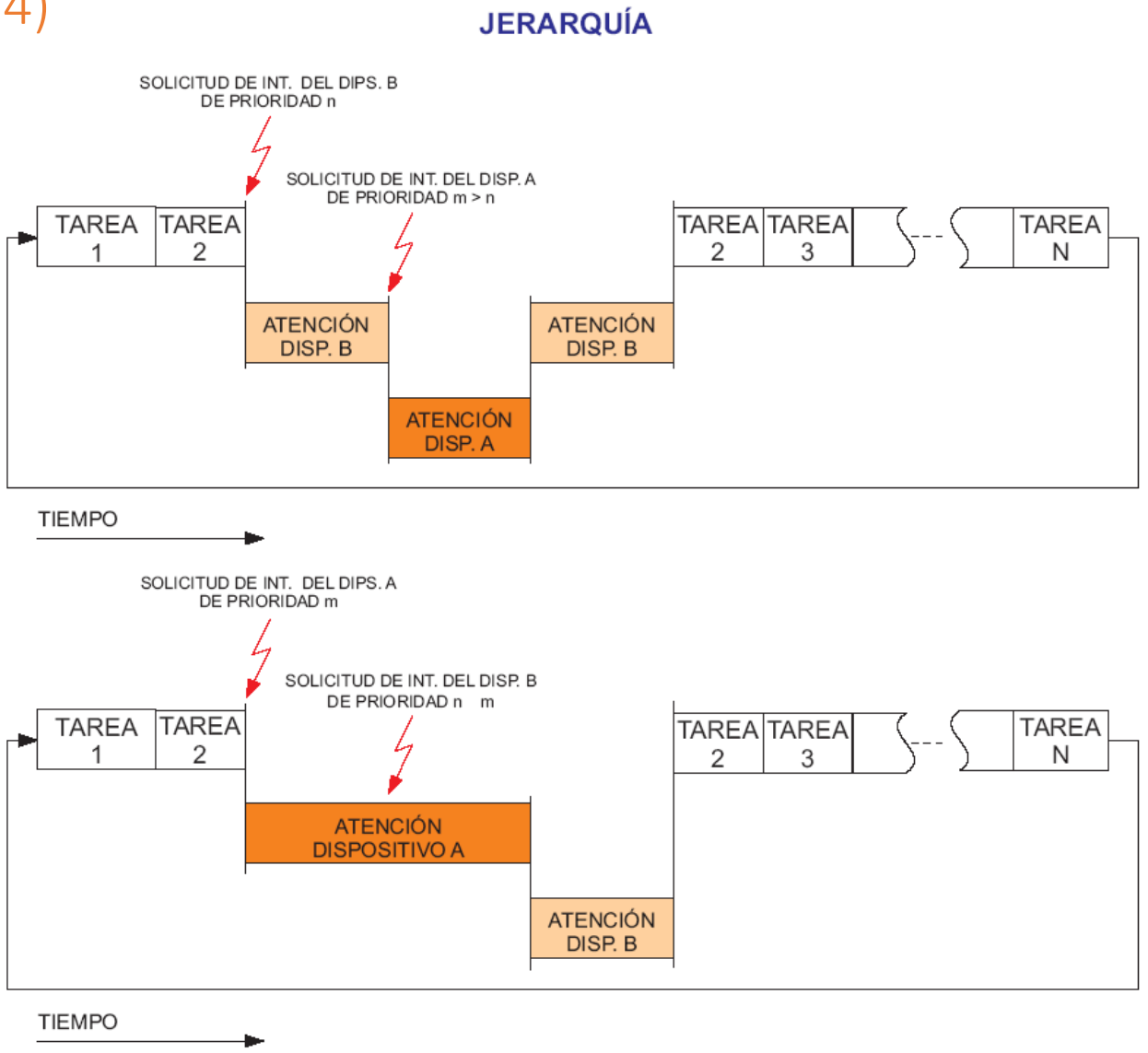
Atención a dispositivos (2)



Atención a dispositivos (3)



Atención a dispositivos (4)

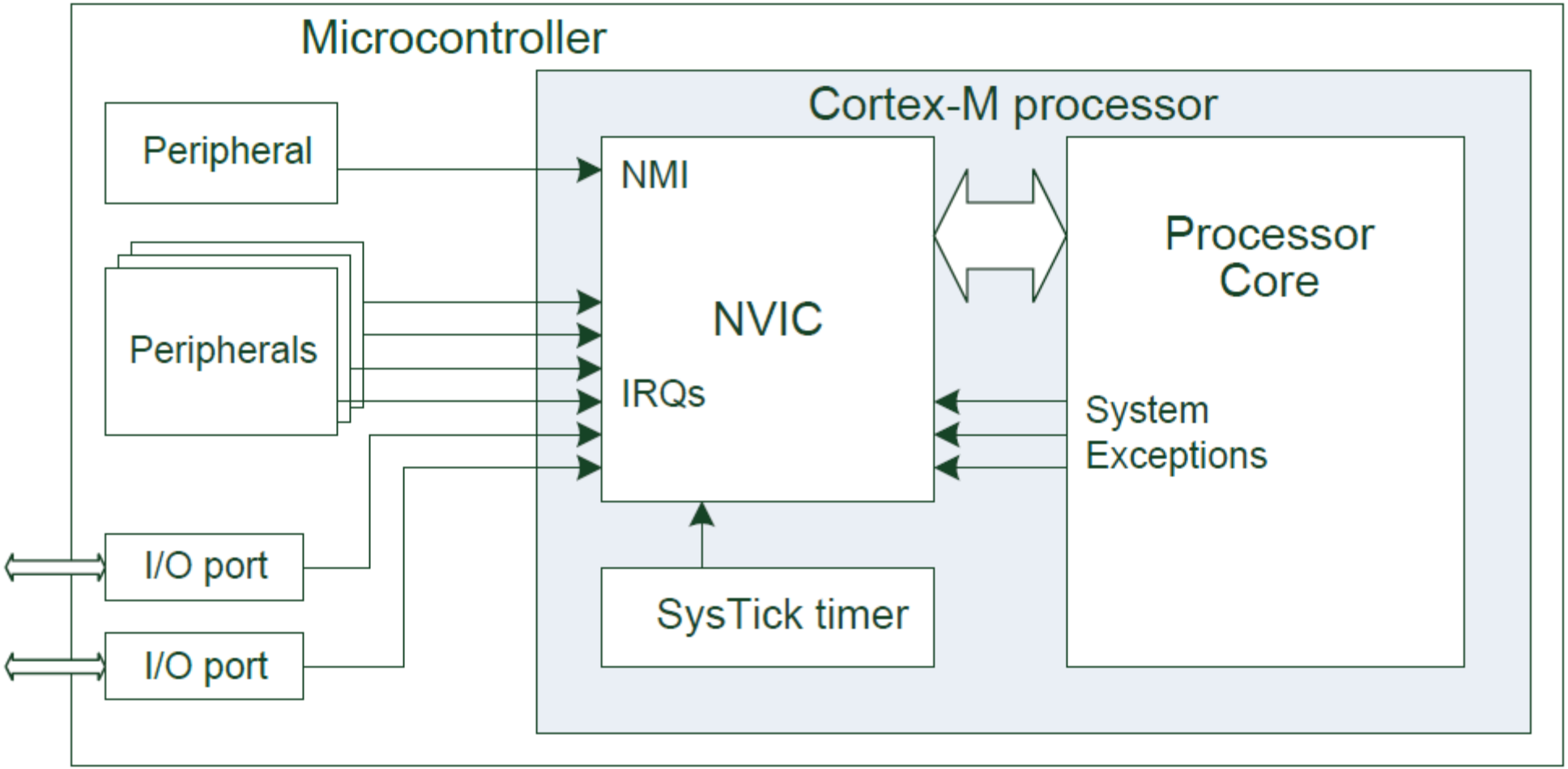


2.3.8. Controlador de interrupciones vectorizado anidado NVIC

- Es una parte integral del ARM CORTEX-M4
- Firmemente acoplado presenta una baja latencia de interrupción
- Sistema de control de excepciones e interrupciones:
- Permite habilitar/deshabilitar cada una de ellas (Se hace en el SBC System Control Block, no en el NVIC)
- Permite definir las prioridades de cada una de ellas (Idem)
- Recibe las señales de excepción e interrupción y las gestiona en función de su configuración.
- 32 niveles programables de prioridad de interrupciones con prioridad hardware a nivel de enmascaramiento
- Tabla de vectores relocalizable
- Interrupciones no enmascarables NMI: no se puede deshabilitar máxima prioridad después del reset, para eventos críticos (fallo en la alimentación)
- Generación de interrupción software

NVIC (Cap. 5 UM10562 Nested Vectored Interrup Controller)

2.3.9. Diagrama del Controlador NVIC



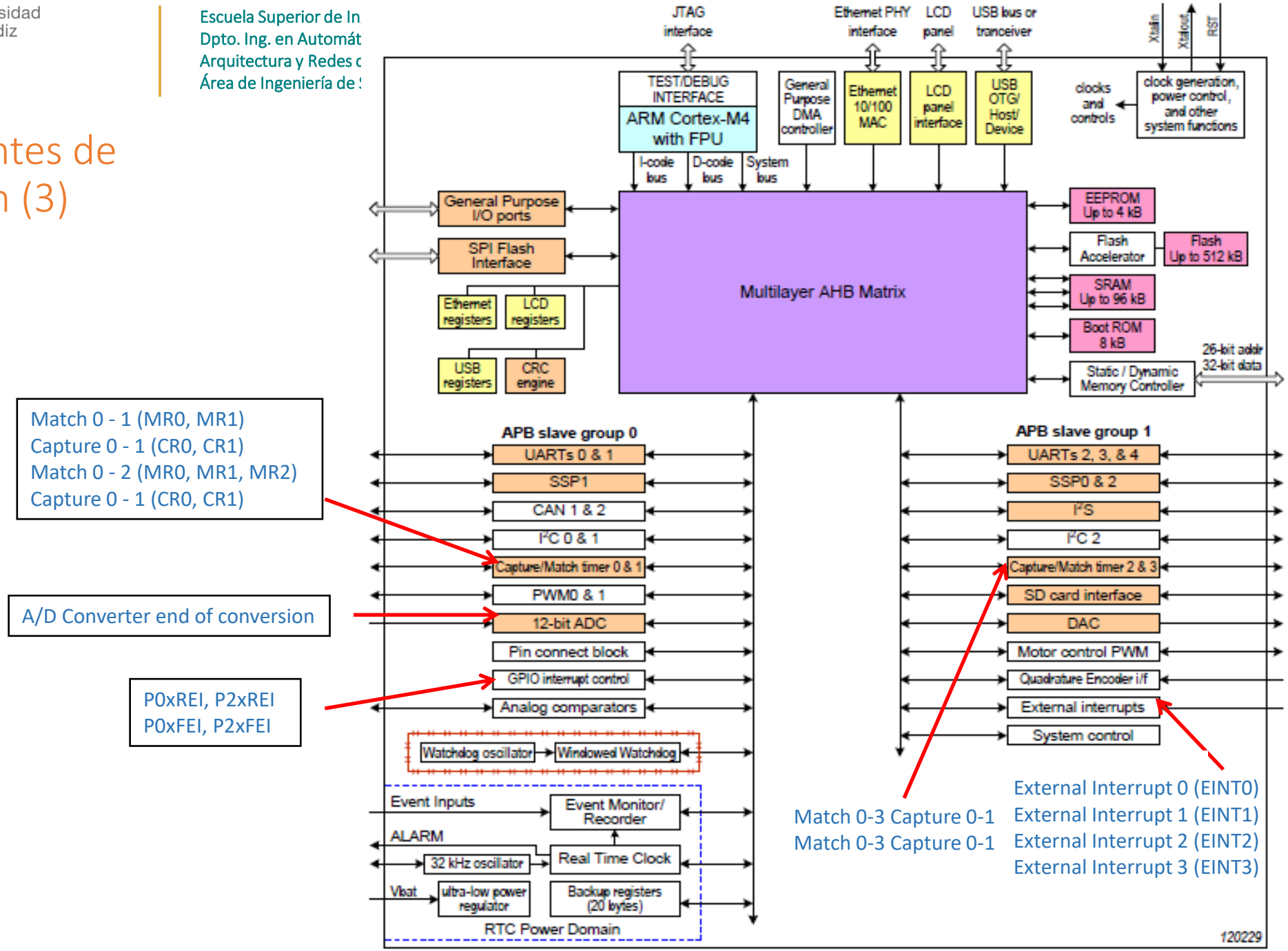
2.3.10. Fuentes de interrupción (1)

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
0	16	0x40	WDT	Watchdog Interrupt (WDINT)
1	17	0x44	Timer 0	Match 0 - 1 (MR0, MR1), Capture 0 - 1 (CR0, CR1)
2	18	0x48	Timer 1	Match 0 - 2 (MR0, MR1, MR2), Capture 0 - 1 (CR0, CR1)
3	19	0x4C	Timer 2	Match 0-3 Capture 0-1
4	20	0x50	Timer 3	Match 0-3 Capture 0-1
5	21	0x54	UART0	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
6	22	0x58	UART1	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), Modem Control Change End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
7	23	0x5C	UART 2	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
8	24	0x60	UART 3	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
9	25	0x64	PWM1	Match 0 - 6 of PWM1, Capture 0-1 of PWM1
10	26	0x68	I2C0	SI (state change)
11	27	0x6C	I2C1	SI (state change)
12	28	0x70	I2C2	SI (state change)
13	29	0x74	(unused)	-
14	30	0x78	SSP0	Tx FIFO half empty of SSP0, Rx FIFO half full of SSP0, Rx Timeout of SSP0, Rx Overrun of SSP0
15	31	0x7C	SSP 1	Tx FIFO half empty, Rx FIFO half full, Rx Timeout, Rx Overrun
16	32	0x80	PLL0 (Main PLL)	PLL0 Lock (PLOCK0)
17	33	0x84	RTC and Event Monitor/Recorder	Counter Increment (RTCCIF), Alarm (RTCALF) EV0, EV1, EV2
18	34	0x88	External Interrupt	External Interrupt 0 (EINT0)
19	35	0x8C	External Interrupt	External Interrupt 1 (EINT1)
20	36	0x90	External Interrupt	External Interrupt 2 (EINT2)
21	37	0x94	External Interrupt	External Interrupt 3 (EINT3)

2.3.10. Fuentes de interrupción (2)

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
22	38	0x98	ADC	A/D Converter end of conversion
23	39	0x9C	BOD	Brown Out detect
24	40	0xA0	USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA, USB_HOST_INT, USB_ATX_INT, USB_OTG_INT, USB_I2C_INT
25	41	0xA4	CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx
26	42	0xA8	DMA Controller	Interrupt status of all DMA channels
27	43	0xAC	I2S	irq, dmareq1, dmareq2
28	44	0xB0	Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.
29	45	0xB4	SD Card Interface	RxDataAvlbl, TxDataAvlbl, RxFifoEmpty, TxFifoEmpty, RxFifoFull, TxFifoFull, RxFifoHalfFull, TxFifoHalfEmpty, RxActive, TxActive, CmdActive, DataBlockEnd, StartBitErr, DataEnd, CmdSent, CmdRespEnd, RxOverrun, TxUnderrun, DataTimeOut, CmdTimeOut, DataCrcFail, CmdCrcFail
30	46	0xB8	Motor Control PWM	IPER[2:0], IPW[2:0], ICAP[2:0], FES
31	47	0xBC	Quadrature Encoder	INX_Int, TIM_Int, VELC_Int, DIR_Int, ERR_Int, ENCLK_Int, POS0_Int, POS1_Int, POS2_Int, REV_Int, POS0REV_Int, POS1REV_Int, POS2REV_Int
32	48	0xC0	PLL1 (Alt PLL)	PLL1 Lock (PLOCK1)
33	49	0xC4	USB Activity Interrupt	USB_NEED_CLK
34	50	0xC8	CAN Activity Interrupt	CAN1WAKE, CAN2WAKE
35	51	0xCC	UART4	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
36	52	0xD0	SSP2	Tx FIFO half empty of SSP2 Rx FIFO half full of SSP2, Rx Timeout of SSP2, Rx Overrun of SSP2
37	53	0xD4	LCD controller	BER, VCompI, LNBUI, FUFI, CrsrI
38	54	0xD8	GPIO interrupts	P0xREI, P2xREI, P0xFEI, P2xFEI
39	55	0xDC	PWM0	Match 0 - 6 of PWM0 Capture 0-1 of PWM0
40	56	0xE0	EEPROM	EE_PROG_DONE, EE_RW_DONE

2.3.10. Fuentes de Interrupción (3)



Match 0 - 1 (MR0, MR1)
Capture 0 - 1 (CR0, CR1)
Match 0 - 2 (MR0, MR1, MR2)
Capture 0 - 1 (CR0, CR1)

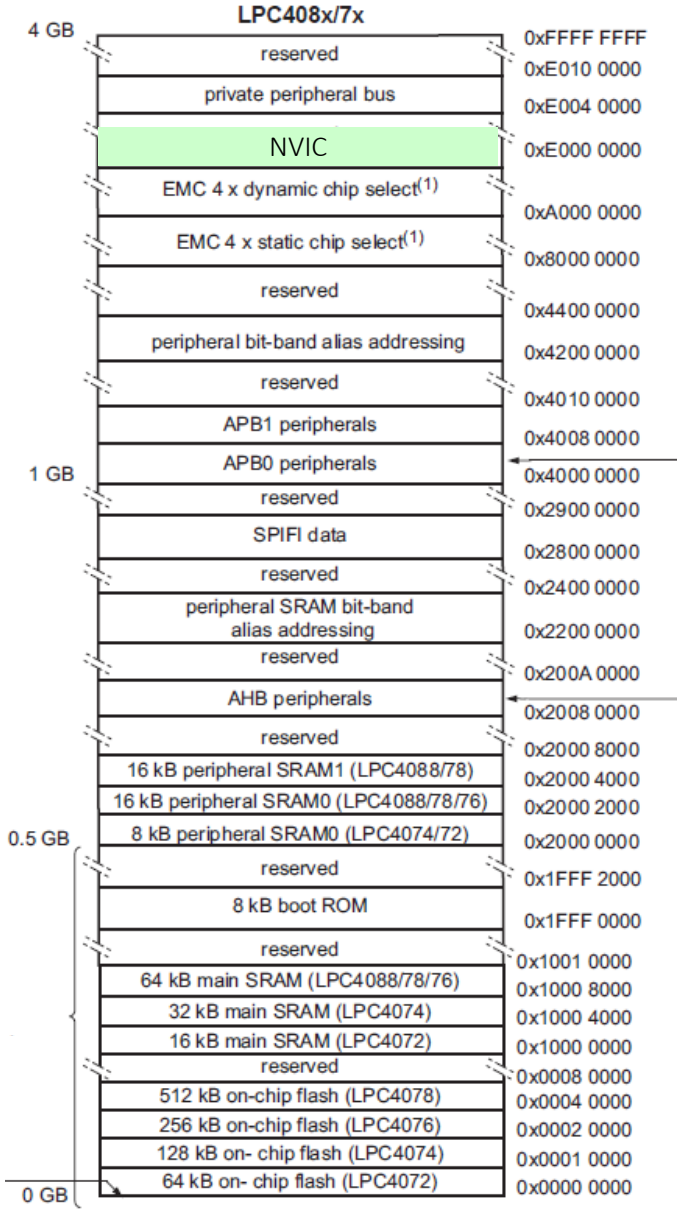
A/D Converter end of conversion

P0xREI, P2xREI
P0xFEI, P2xFEI

Match 0-3 Capture 0-1
Match 0-3 Capture 0-1

External Interrupt 0 (EINT0)
External Interrupt 1 (EINT1)
External Interrupt 2 (EINT2)
External Interrupt 3 (EINT3)

2.3.11. Registros de interrupción NVIC (1)



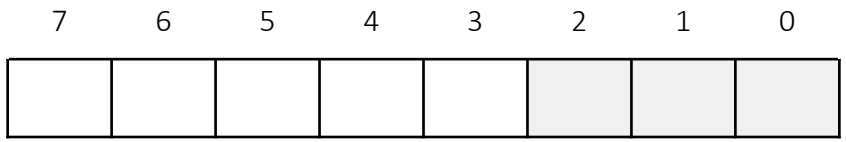
2.3.11. Registros de interrupción NVIC (2)

Name	Description	Access	Reset value	Address	Table
ISER0 to ISER1	Interrupt Set-Enable Registers. These registers allow enabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ISER0 - 0xE000 E100	53
				ISER1 - 0xE000 E104	54
ICER0 to ICER1	Interrupt Clear-Enable Registers. These registers allow disabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ICER0 - 0xE000 E180	55
				ICER1 - 0xE000 E184	56
ISPR0 to ISPR1	Interrupt Set-Pending Registers. These registers allow changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ISPR0 - 0xE000 E200	57
				ISPR1 - 0xE000 E204	58
ICPR0 to ICPR1	Interrupt Clear-Pending Registers. These registers allow changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ICPR0 - 0xE000 E280	59
				ICPR1 - 0xE000 E284	60
IABR0 to IABR1	Interrupt Active Bit Registers. These registers allow reading the current interrupt active state for specific peripheral functions.	RO	0	IABR0 - 0xE000 E300	61
				IABR1 - 0xE000 E304	62
IPR0 to IPR10	Interrupt Priority Registers. These registers allow assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	RW	0	IPR0 - 0xE000 E400	63
				IPR1 - 0xE000 E404	64
				IPR2 - 0xE000 E408	65
				IPR3 - 0xE000 E40C	66
				IPR4 - 0xE000 E410	67
				IPR5 - 0xE000 E414	68
				IPR6 - 0xE000 E418	69
				IPR7 - 0xE000 E41C	70
				IPR8 - 0xE000 E420	71
				IPR9 - 0xE000 E424	72
				IPR10 - 0xE000 E428	73
STIR	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	WO	-	STIR - 0xE000 EF00	74

2.3.11. Registros de interrupción NVIC (3)

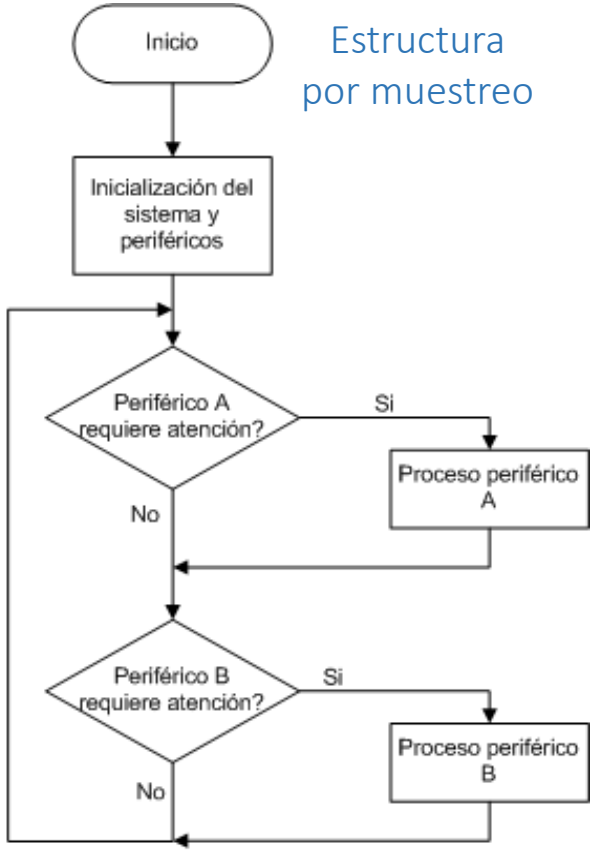
Registro de prioridad IPR0

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_WDT	Watchdog Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_TIMER0	Timer 0 interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_TIMER1	Timer 1 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_TIMER2	Timer 2 interrupt priority. See functional description for bits 7-3.

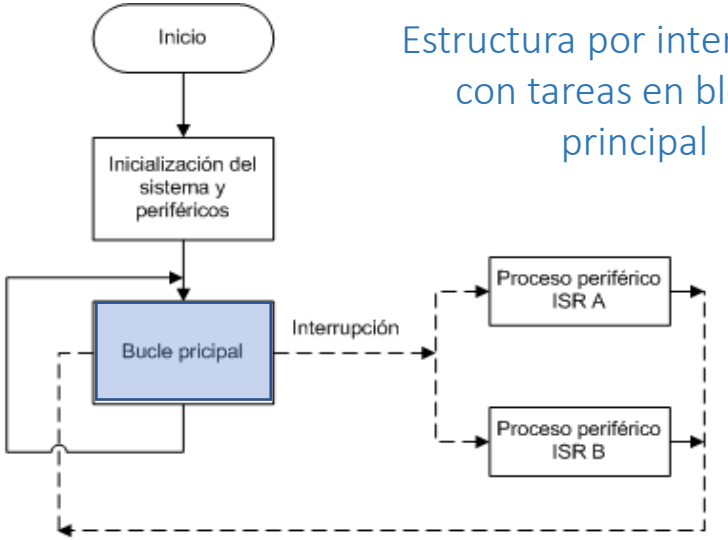


2.3.12. Estructuras de programa con interrupciones (1)

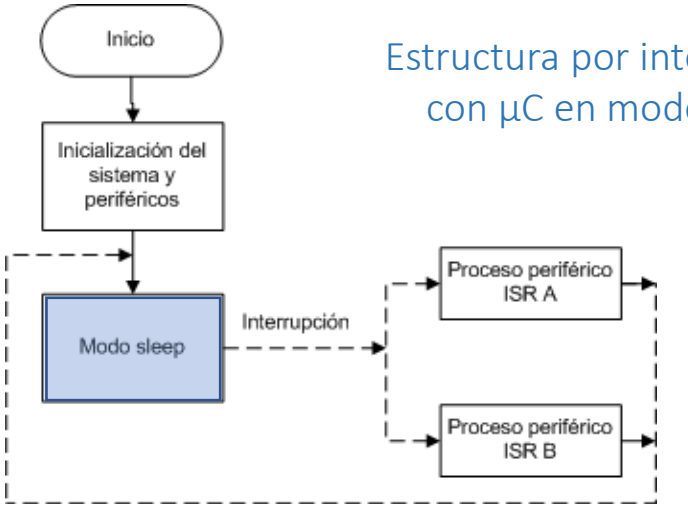
Estructura por muestreo



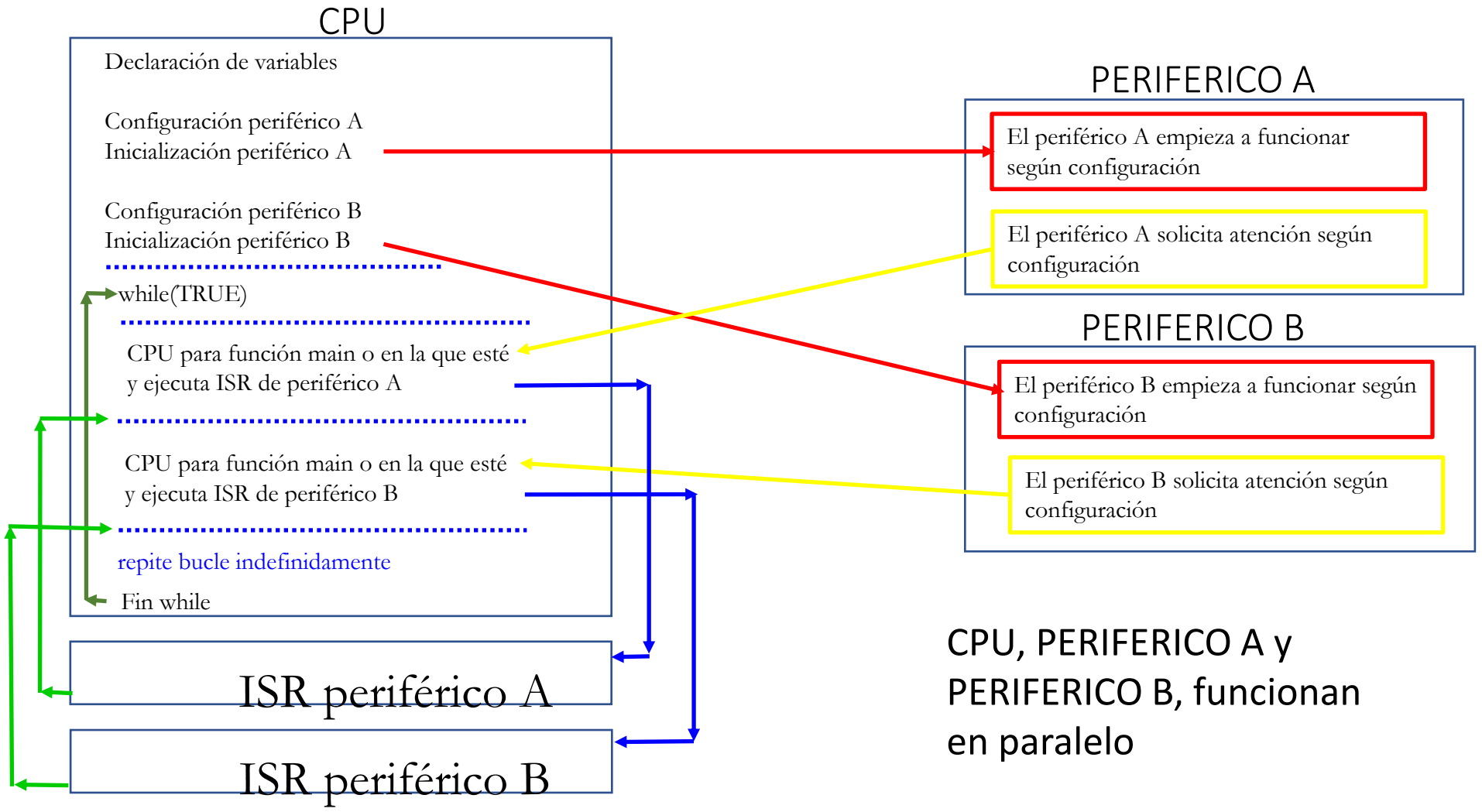
Estructura por interrupción con tareas en bloque principal



Estructura por interrupción con µC en modo Sleep



2.3.12. Estructuras de programa con interrupciones (2)



CPU, PERIFERICO A y PERIFERICO B, funcionan en paralelo

2.3.13. Manejadores (handlers) de interrupciones

Para llamar a las funciones de interrupción (ISR) se deben usar unos nombres determinados y se programan como funciones normales de C. Para otros m

```
DCD    WDT_IRQHandler          ; 16: Watchdog Timer
DCD    TIMER0_IRQHandler       ; 17: Timer0
DCD    TIMER1_IRQHandler       ; 18: Timer1
DCD    TIMER2_IRQHandler       ; 19: Timer2
DCD    TIMER3_IRQHandler       ; 20: Timer3
DCD    UART0_IRQHandler        ; 21: UART0
DCD    UART1_IRQHandler        ; 22: UART1
DCD    UART2_IRQHandler        ; 23: UART2
DCD    UART3_IRQHandler        ; 24: UART3
DCD    PWM1_IRQHandler         ; 25: PWM1
DCD    I2C0_IRQHandler         ; 26: I2C0
DCD    I2C1_IRQHandler         ; 27: I2C1
DCD    I2C2_IRQHandler         ; 28: I2C2
DCD    0                       ; 29: reserved
DCD    SSP0_IRQHandler         ; 30: SSP0
DCD    SSP1_IRQHandler         ; 31: SSP1
DCD    PLL0_IRQHandler         ; 32: PLL0 Lock (Main PLL)
DCD    RTC_IRQHandler          ; 33: Real Time Clock
DCD    EINT0_IRQHandler        ; 34: External Interrupt 0
DCD    EINT1_IRQHandler        ; 35: External Interrupt 1
DCD    EINT2_IRQHandler        ; 36: External Interrupt 2
DCD    EINT3_IRQHandler        ; 37: External Interrupt 3
DCD    ADC_IRQHandler          ; 38: A/D Converter
DCD    BOD_IRQHandler          ; 39: Brown-Out Detect
DCD    USB_IRQHandler          ; 40: USB
DCD    CAN_IRQHandler          ; 41: CAN
DCD    DMA_IRQHandler          ; 42: General Purpose DMA
DCD    I2S_IRQHandler          ; 43: I2S
DCD    ENET_IRQHandler         ; 44: Ethernet
DCD    MCI_IRQHandler          ; 45: SD/MMC card I/F
DCD    MCPWM_IRQHandler        ; 46: Motor Control PWM
DCD    QEI_IRQHandler          ; 47: Quadrature Encoder Interface
DCD    PLL1_IRQHandler         ; 48: PLL1 Lock (USB PLL)
DCD    USBActivity_IRQHandler  ; 49: USB Activity interrupt to wakeup
DCD    CANActivity_IRQHandler  ; 50: CAN Activity interrupt to wakeup
DCD    UART4_IRQHandler        ; 51: UART4
DCD    SSP2_IRQHandler         ; 52: SSP2
DCD    LCD_IRQHandler          ; 53: LCD
DCD    GPIO_IRQHandler         ; 54: GPIO
DCD    PWM0_IRQHandler         ; 55: PWM0
DCD    EEPROM_IRQHandler       ; 56: EEPROM
```

2.3.14. Funciones CMSIS Cortex Microcontroller Software Interface Standard

- Funciones desarrolladas por ARM para hacer estándar la forma de acceder a los periféricos de los microcontroladores con CPU Cortex-M
- En particular usaremos algunas funciones del grupo que maneja el NVIC en vez de acceder directamente a los registros de este

void NVIC_EnableIRQ (IRQn Type IRQn)

- Habilitar la interrupción indicada por IRQn

void NVIC_DisableIRQ (IRQn Type IRQn)

- Deshabilitar la interrupción indicada por IRQn

void NVIC_SetPriority (IRQn Type IRQn, uint32 t priority)

- Ajustar la prioridad de la interrupción indicada por IRQn al valor indicado por priority

void NVIC_ClearPendingIRQ (IRQn Type IRQn)

- Borrar una solicitud de interrupción pendiente

Para habilitar/deshabilitar las interrupciones IRQ en el PRIMASK las funciones:

- void __enable_irq(void);
- void __disable_irq(void);

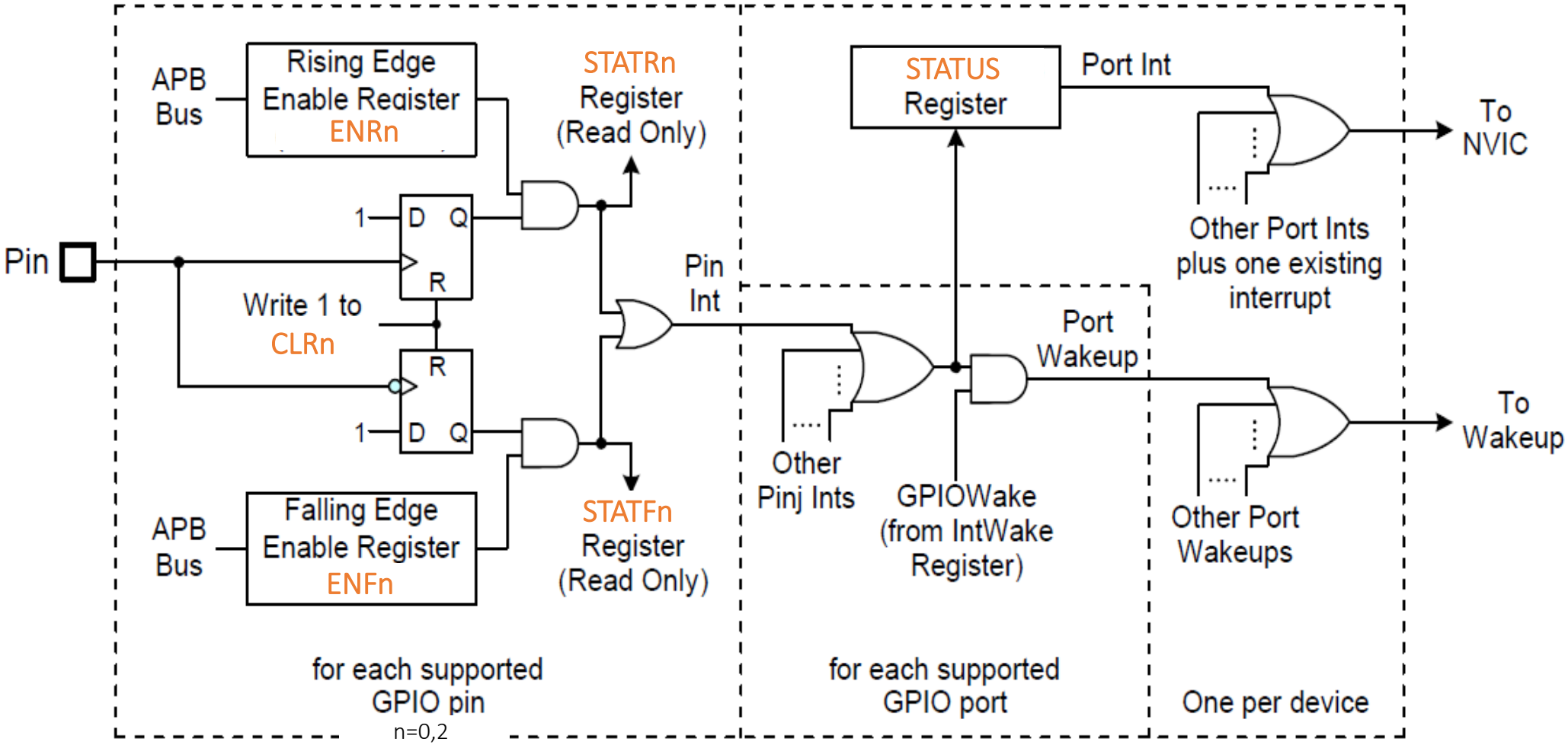
Etiquetas para los números de interrupción

```
WDT_IRQn        = 0, Watchdog Timer Interrupt
TIMER0_IRQn     = 1, Timer0 Interrupt
TIMER1_IRQn     = 2, Timer1 Interrupt
TIMER2_IRQn     = 3, Timer2 Interrupt
TIMER3_IRQn     = 4, Timer3 Interrupt
UART0_IRQn      = 5, UART0 Interrupt
UART1_IRQn      = 6, UART1 Interrupt
UART2_IRQn      = 7, UART2 Interrupt
UART3_IRQn      = 8, UART3 Interrupt
PWM1_IRQn       = 9, PWM1 Interrupt
I2C0_IRQn       = 10, I2C0 Interrupt
I2C1_IRQn       = 11, I2C1 Interrupt
I2C2_IRQn       = 12, I2C2 Interrupt
Reserved0_IRQn  = 13, Reserved
SSP0_IRQn       = 14, SSP0 Interrupt
SSP1_IRQn       = 15, SSP1 Interrupt
PLL0_IRQn       = 16, PLL0 Lock (Main PLL) Interrupt
RTC_IRQn        = 17, Real Time Clock Interrupt
EINT0_IRQn      = 18, External Interrupt 0 Interrupt
EINT1_IRQn      = 19, External Interrupt 1 Interrupt
EINT2_IRQn      = 20, External Interrupt 2 Interrupt
EINT3_IRQn      = 21, External Interrupt 3 Interrupt
ADC_IRQn        = 22, A/D Converter Interrupt
BOD_IRQn        = 23, Brown-Out Detect Interrupt
USB_IRQn        = 24, USB Interrupt
CAN_IRQn        = 25, CAN Interrupt
DMA_IRQn        = 26, General Purpose DMA Interrupt
I2S_IRQn        = 27, I2S Interrupt
ENET_IRQn       = 28, Ethernet Interrupt
MCI_IRQn        = 29, SD/MMC card I/F Interrupt
MCPWM_IRQn      = 30, Motor Control PWM Interrupt
QEI_IRQn        = 31, Quadrature Encoder Interface Interrupt
PLL1_IRQn       = 32, PLL1 Lock (USB PLL) Interrupt
USBActivity_IRQn  = 33, USB Activity interrupt
CANActivity_IRQn  = 34, CAN Activity interrupt
UART4_IRQn      = 35, UART4 Interrupt
SSP2_IRQn       = 36, SSP2 Interrupt
LCD_IRQn        = 37, LCD Interrupt
GPIO_IRQn       = 38, GPIO Interrupt
PWM0_IRQn       =  39, PWM0
EEPROM_IRQn     =  40, EEPROM
CMP0_IRQn       =  41, CMP0
CMP1_IRQn       =  42, CMP1
} IRQn_Type;
```

2.3.15. Interrupciones en el GPIO (1)

- Todos los pines del puerto P0 y P2 permiten interrupciones.
- Cada pin del Puerto puede ser programado para generar una interrupción con un flanco de subida, un flanco de bajada o ambos
- La detección del flanco es asíncrono, puede operar cuando el reloj no está presente, en modo bajo consumo.
- Cada interrupción habilitada contribuye a una señal que puede ser usada para salir del modo bajo consumo.
- Los registros proporcionan una vista del software de interrupciones pendientes por flanco de subida, pendientes por flancos de bajada y todas las GPIO pendientes.
- La función de interrupción GPIO no requiere que el pin sea configurado para GPIO.

2.3.15. Diagrama interrupciones en GPIO



2.3.15. Interrupciones en el GPIO

Table 95. Register overview: GPIO interrupt (base address 0x4002 8000)

Name	Access	Address offset	Description	Reset value[1]	Table
STATUS	RO	0x080	GPIO overall Interrupt Status.	0	101
STATR0	RO	0x084	GPIO Interrupt Status for Rising edge for Port 0.	0	102
STATF0	RO	0x088	GPIO Interrupt Status for Falling edge for Port 0.	0	103
CLR0	WO	0x08C	GPIO Interrupt Clear.	-	104
ENR0	R/W	0x090	GPIO Interrupt Enable for Rising edge for Port 0.	0	105
ENF0	R/W	0x094	GPIO Interrupt Enable for Falling edge for Port 0.	0	106
STATR2	RO	0x0A4	GPIO Interrupt Status for Rising edge for Port 0.	0	107
STATF2	RO	0x0A8	GPIO Interrupt Status for Falling edge for Port 0.	0	108
CLR2	WO	0x0AC	GPIO Interrupt Clear.	-	109
ENR2	R/W	0x0B0	GPIO Interrupt Enable for Rising edge for Port 0.	0	110
ENF2	R/W	0x0B4	GPIO Interrupt Enable for Falling edge for Port 0.	0	111

2.3.15. Interrupciones en el GPIO (2)

```
typedef struct
{__I  uint32_t STATUS;
  __I  uint32_t STATR0;
  __I  uint32_t STATF0;
  __O  uint32_t CLR0;
  __IO uint32_t ENR0;
  __IO uint32_t ENF0;
       uint32_t RESERVED0[3];
  __I  uint32_t STATR2;
  __I  uint32_t STATF2;
  __O  uint32_t CLR2;
  __IO uint32_t ENR2;
  __IO uint32_t ENF2;
} LPC_GPIOINT_TypeDef;
```

LPC_GPIOINT -> miembro estructura

2.3.15. Interrupciones en el GPIO (3)

Nombre función ISR: GPIO_IRQHandler

Numero IRQ: GPIO_IRQn

void NVIC_EnableIRQ (IRQn_Type IRQn)

void NVIC_DisableIRQ (IRQn_Type IRQn)

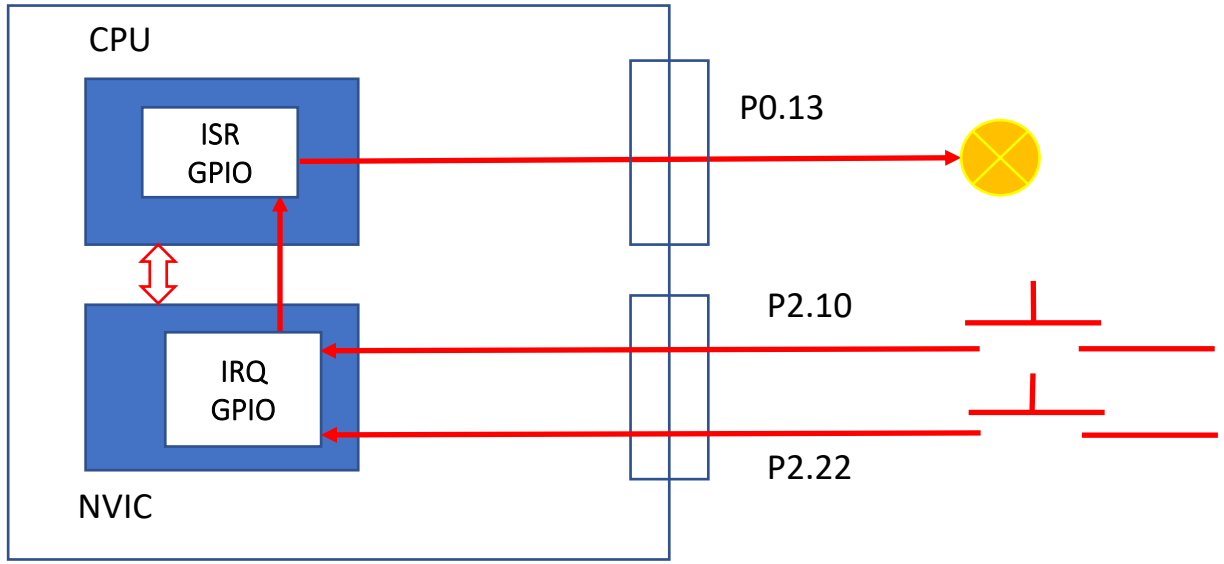
void NVIC_SetPriority (IRQn_Type IRQn, uint32 t priority)

void NVIC_ClearPendingIRQ (IRQn_Type IRQn)

2.3.16. Ejemplo ISR para GPIO (1)

Ejemplo de interrupciones de GPIO

Programar dos interrupciones GPIO para actuar sobre el led conectado al pin13 de P0. Configurar el pin 22 del P2 para generar una interrupción por flanco de bajada para encender el led y configurar el pin 10 de P2 para generar una interrupción por flanco de bajada para apagar el led.



2.3.16. Ejemplo ISR para GPIO (2)

```
#include “LPC407x_8x_177x_8x.h”
void GPIO_IRQHandler(void);

int main(void)
{/* Configurar el pin P0[13] como salida para manejar un LED*/
   LPC_GPIO0 -> DIR |= (1<<13);
   /* Apagar el LED.*/
   LPC_GPIO0-> SET = (1<<13);
   /* Configurar los pines P2[22] y P2[10] (botones de la tarjeta)
    * para generar interrupciones por flanco de bajada.*/
   LPC_GPIOINT ->ENF2 |= (1<<10) | (1<<22);
   NVIC_ClearPendingIRQ (GPIO_IRQn);
   NVIC_EnableIRQ (GPIO_IRQn);
   NVIC_SetPriority (GPIO_IRQn, 0);
   __enable_irq();
   while (1);
}
```

2.3.16. Ejemplo ISR para GPIO (3)

```
void GPIO_IRQHandler(void)
{  /* Verificar si hay interrupción pendiente del pin P2[22]*/

   if (LPC_GPIOINT -> STATF2 & (1<<22))
   {  NVIC_ClearPendingIRQ (GPIO_IRQn); /* Borrar el flag de petición. */
      LPC_GPIO0 -> CLR = (1<<13);     /* Encender el LED. */
      LPC_GPIOINT -> CLR2 = (1<<22));
   }

   /* Verificar si hay interrupción pendiente del pin P2[10]*/

   if (LPC_GPIOINT -> STATF2 & (1<<10))
   {  NVIC_ClearPendingIRQ (GPIO_IRQn); /* Borrar el flag de petición. */
      LPC_GPIO0 -> SET = (1<<13);     /* Apagar el LED. */
      LPC_GPIOINT -> CLR2 = (1<<10));
   }
}
```

2.3.17. Interrupciones externas específicas

Entrada Externa de interrup.	Funciones	Valor func.
EINT0/NMI	**P0[29]**/USB_D+1/**EINT0**	**2**
	P2[10]/**EINT0/NMI**	1/2
EINT1	P0[30]/USB_D-1/**EINT1**	2
	P2[11]/**EINT1**/SD_DAT[1]/I2S_TX_SCK/LCD_CLKIN	1
EINT2	P2[12]/**EINT2**/SD_DAT[2]/I2S_TX_WS/LCD_VD[4]/LCD_VD[3]/LCD_VD[8]/LCD_VD[18]	1
EINT3	P2[13]/**EINT3**/SD_DAT[3]/I2S_TX_SDA/LCD_VD[5]/LCD_VD[9]/LCD_VD[19]	1

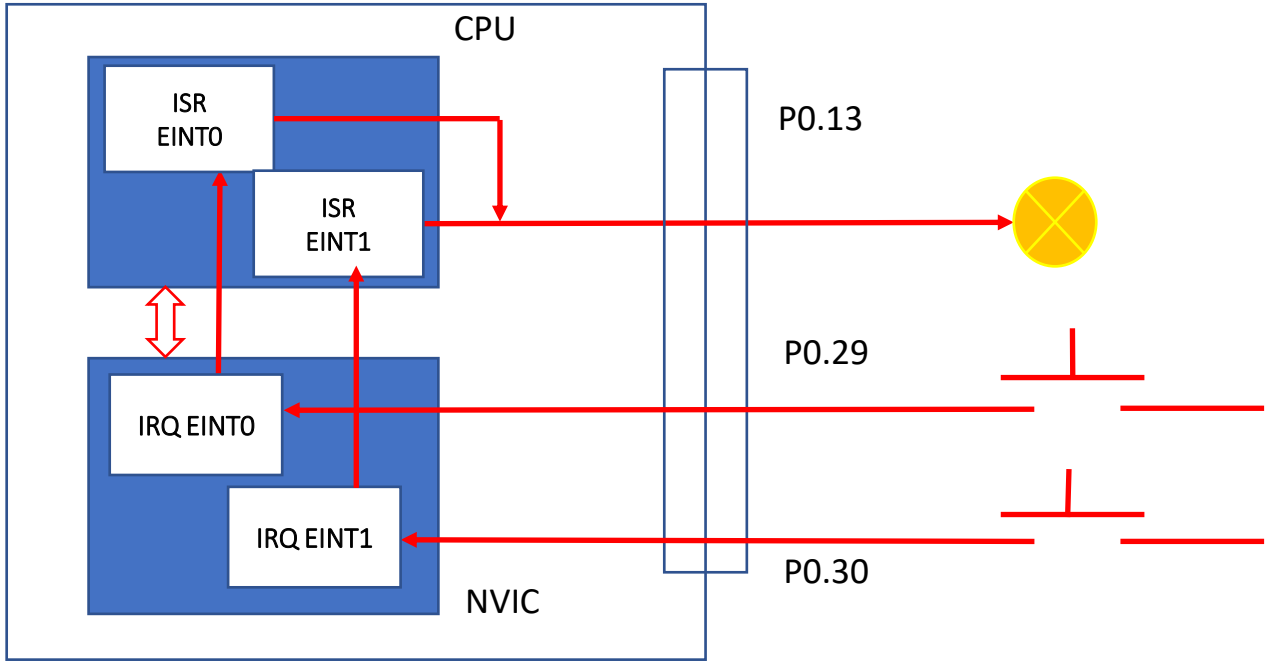
LPC_IOCON->P0_29 = 2

Etiquetas de ISR y numero interrupción

Nombre de la ISR	Numero de la interrupción
EINT0_IRQHandler	EINT0_IRQn
EINT1_IRQHandler	EINT1_IRQn
EINT2_IRQHandler	EINT2_IRQn
EINT3_IRQHandler	EINT3_IRQn
NMI_Handler	NonMaskableInt_IRQn

2.3.18. Ejemplo ISR para EINTn (1)

Programar la entrada externa de interrupciones 0 para encender el led conectado a P0[13] y la entrada externa de interrupciones 1 para apagar dicho led



2.3.18. Ejemplo ISR para EINTn (2)

```
#include “LPC407x_8x_177x_8x.h”
void EINT0_IRQHandler(void);
void EINT1_IRQHandler(void);

int main(void)
{  /* Configurar el pin P0[13] como salida para manejar un LED. */
   LPC_GPIO0 -> DIR |= (1<<13);
   /* Apagar el LED.*/
   LPC_GPIO0 -> SET = (1<<13);

   /* Seleccionar la función EINT0 del pin P0[29] */
   LPC_IOCON->P0_29 = 2;
   /* Seleccionar la función EINT1 del pin P0[30] */
   LPC_IOCON->P0_30 = 2;
```

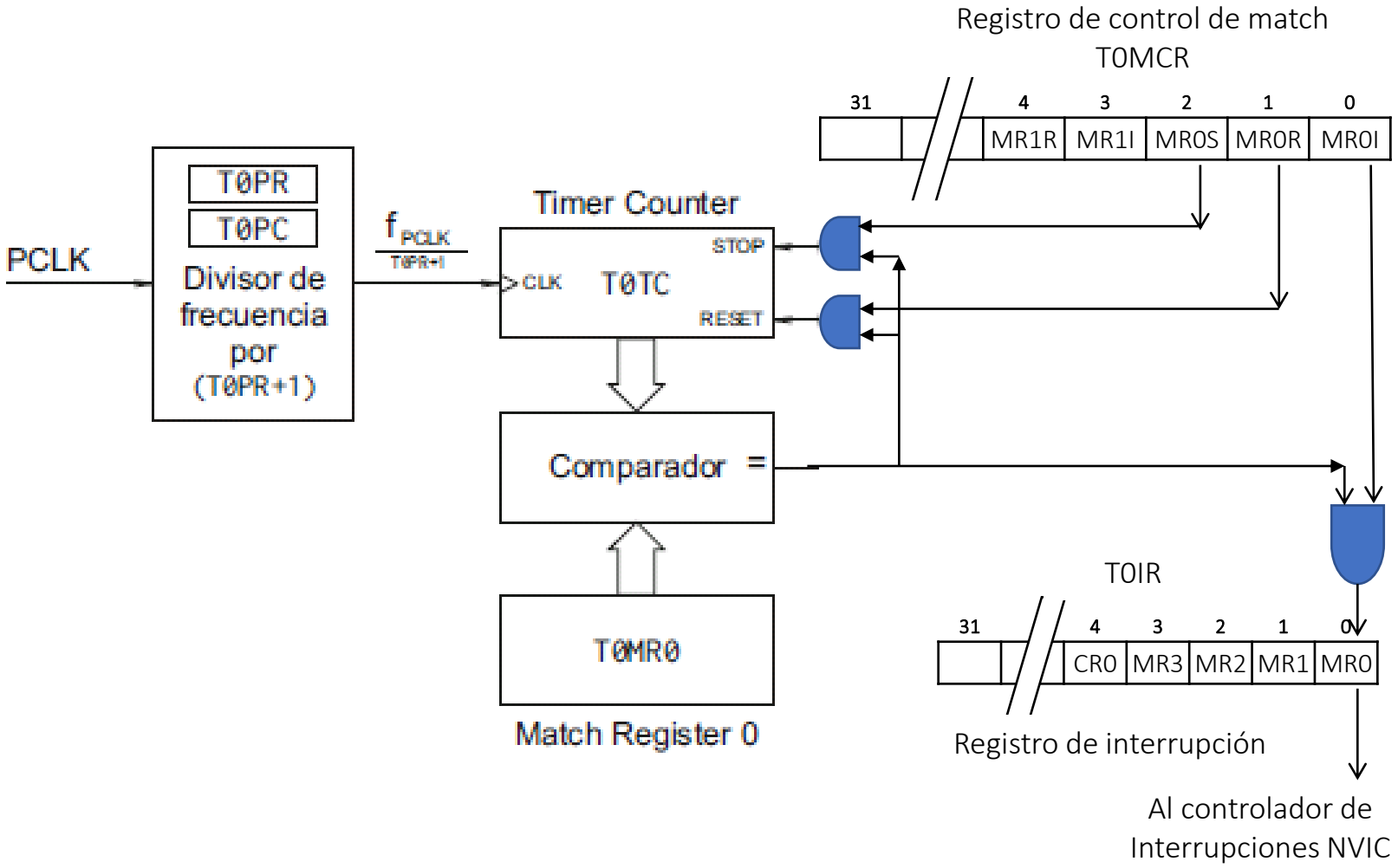
2.3.17. Ejemplo ISR para EINTn (3)

```
    /* Flanco de subida en P0 */
    LPC_GPIOINT ->ENF0 |= (1<<29) | (1<<30);

    /* La interrupción EINT0 */
    NVIC_ClearPendingIRQ (EINT0_IRQn);
    NVIC_EnableIRQ (EINT0_IRQn);
    NVIC_SetPriority (EINT0_IRQn, 0);

    /* La interrupción EINT1*/
    NVIC_ClearPendingIRQ (EINT1_IRQn);
    NVIC_EnableIRQ (EINT1_IRQn);
    NVIC_SetPriority (EINT1_IRQn, 0);

    _enable_irq();

    while (1);
}
```

2.3.18. Ejemplo ISR para EINTn (4)

```
void EINT0_IRQHandler (void)
{
   LPCGPIOINT-> CLR0 =1<<29;    /* Borrar el flag de petición de interrupción. */
   LPCGPIO0 -> CLR = (1<<13);    /* Encender el LED. */
   NVIC_ClearPendingIRQ (EINT0_IRQn);  /* Reconocer la interrupción en el NVIC*/
}

void EINT1_IRQHandler (void)
{
   LPCGPIOINT-> CLR0 =1<<30;    /* Borrar el flag de petición de interrupción. */
   LPCGPIO0 -> SET = (1<<13);    /* Apagar el LED. */
   NVIC_ClearPendingIRQ (EINT1_IRQn);  /* Reconocer la interrupción en el NVIC*/
}
```

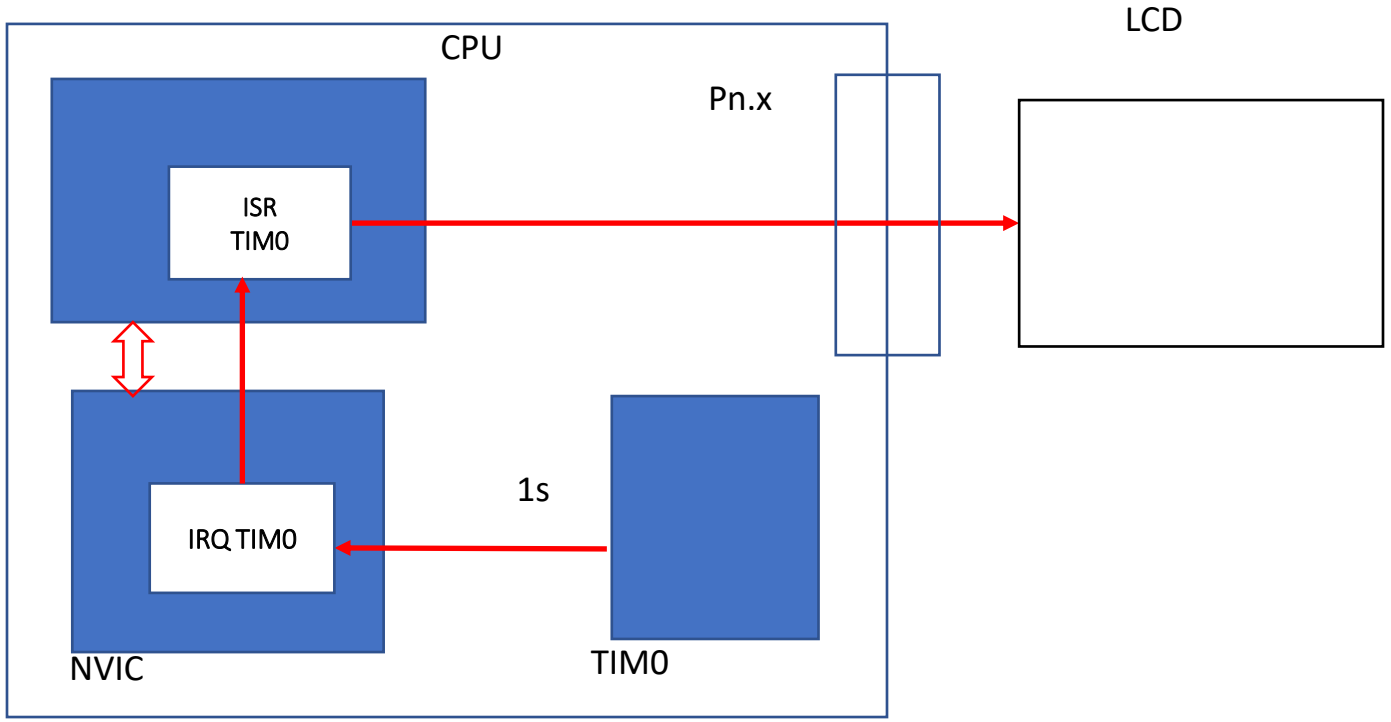
2.3.19. Interrupciones en los timers. Diagrama (1)



2.3.20. Ejemplo ISR para TIMERn (1)

Ejemplo de función ISR en C usando manejador de interrupción IRQ del Timer

Programar el timer 0 para que provoque un evento "match" cada segundo con incrementos de TC de 1 µs y se lleve una cuenta de los segundos.



2.3.20. Ejemplo ISR para TIMERn (2)

```
#include “LPC407x_8x_177x_8x.h”

void TIMER0_IRQHandler(void);
uint8_t contador_segundos = 0;

int main(void)
{  LPC_TIM0 -> TCR = 0;
   LPC_TIM0 -> TC = 0;
   LPC_TIM0 -> PC = 0;
   LPC_TIM0 -> MCR = 3; /* Bit interrupción y Reset */
   LPC_TIM0 -> PR = PeripheralClock/1000000 – 1;
   LPC_TIM0 -> MR0 = 1000 * 1000 -1; /* retardo 1s */
```

2.3.20. Ejemplo ISR para TIMERn (3)

```
    NVIC_ClearPendingIRQ (TIMER0_IRQn);
    NVIC_EnableIRQ (TIMER0_IRQn);
    NVIC_SetPriority (TIMER0_IRQn, 0);
    __enable_irq();

    LPC_TIM0 -> TCR = 1;
    while (1);
}

void TIMER0_IRQHandler (void)
{  LPC_TIM0 -> IR = 1;
   contador_segundos++;
   NVIC_ClearPendingIRQ (TIMER0_IRQn);
}
```