Interrupciones en las CPU Cortex-M y en el LPC4088

Diseño Basado en Microprocesadores

Víctor Manuel Sánchez Corbacho

Dpto. de Automática, Electrónica, Arquitectura y Redes de Computadores

2016

Contenido

Métodos de respuesta a eventos. Sondeo.

- Las aplicaciones con μC deben responder a eventos que se producen en el mundo exterior y generar respuestas adecuadas.
- Una solución es comprobar regularmente la presencia de eventos.
  - $\Rightarrow$ Método de **sondeo (*polling*)**.
- Pero en algunos casos el retardo puede ser inaceptable.

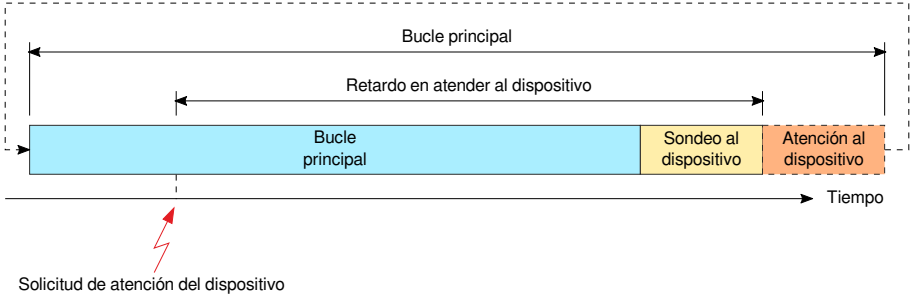


Diagrama flujo sondeo. Un dispositivo.

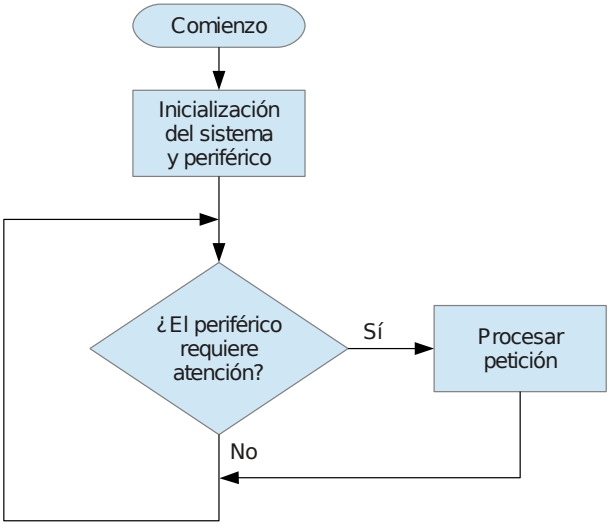
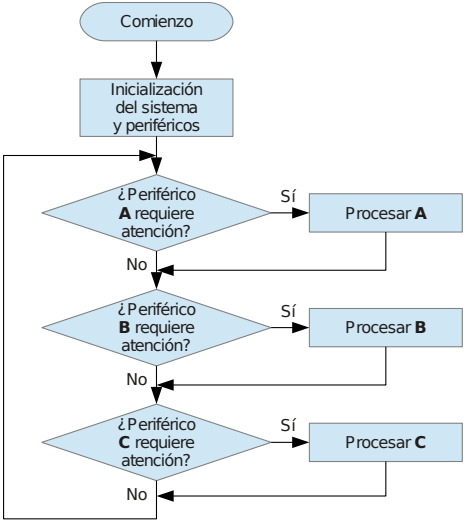
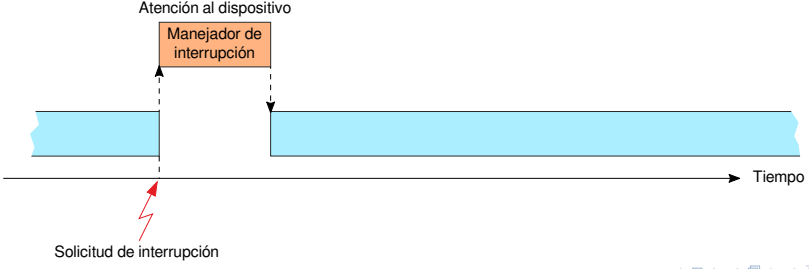


Diagrama flujo sondeo. Varios dispositivos.



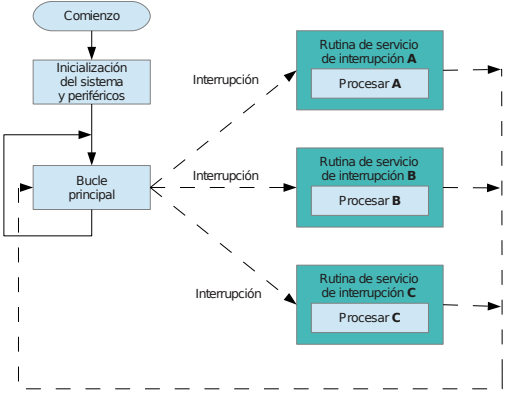
Métodos de respuesta a eventos. Interrupciones.

- Las interrupciones son **eventos** generados por un **dispositivo** hardware que **cambian el flujo del programa**.
- Cuando un dispositivo hardware necesita atención de la CPU:
  1. El **dispositivo activa** una señal de petición de **interrupción**.
  2. **La CPU suspende la tarea que esté ejecutando** en ese momento.
  3. La CPU ejecuta una **rutina de servicio de interrupción (ISR)** o **manejador de interrupción** para atender al dispositivo.
  4. Después de ejecutar la ISR, la CPU **retoma la tarea suspendida por el mismo punto donde se interrumpió**.



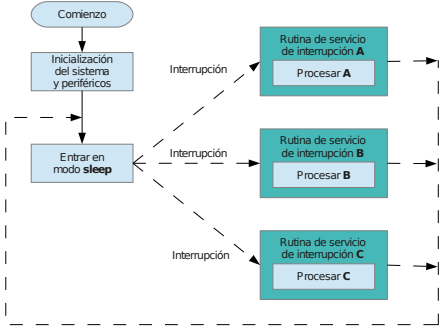
Organización en bucle principal e interrupciones

- Las tareas que no conllevan respuestas rápidas se ejecutan en el bucle principal de la aplicación.
- Los dispositivos que requieren una respuesta rápida se atienden mediante interrupciones.



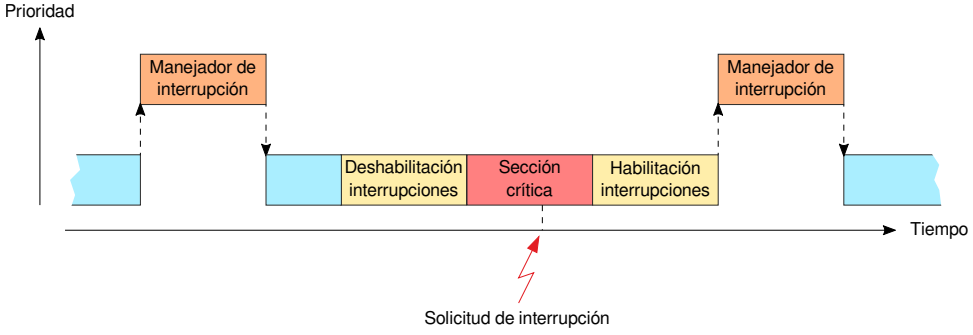
Ahorro de energía con modo *sleep* e interrupciones

- El microcontrolador pasa la mayor parte del tiempo *durmiendo* en un modo de ahorro de energía (modos *sleep* o *power-down*).
- El microcontrolador *despierta* cuando un dispositivo que necesita atención genera una interrupción.
- Después de atender al dispositivo, el microcontrolador vuelve a *dormir*.

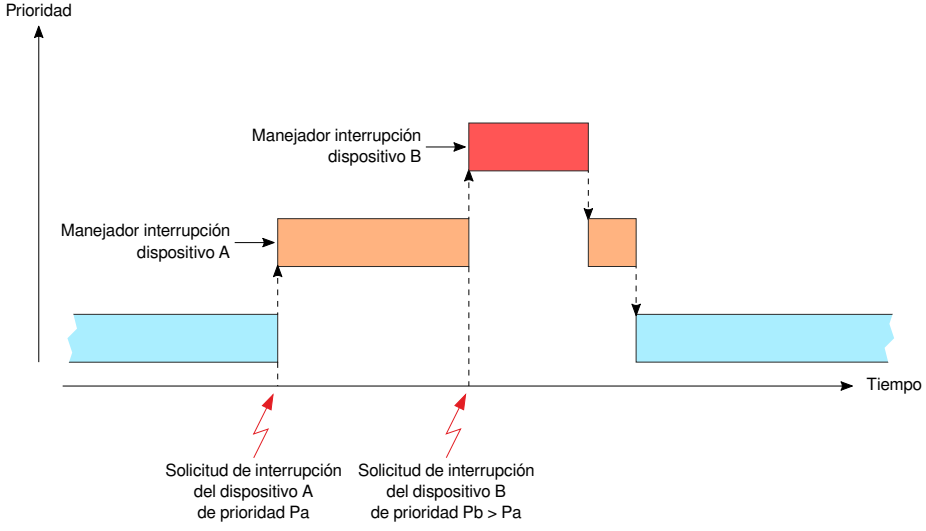


Habilitación/deshabilitación de interrupciones

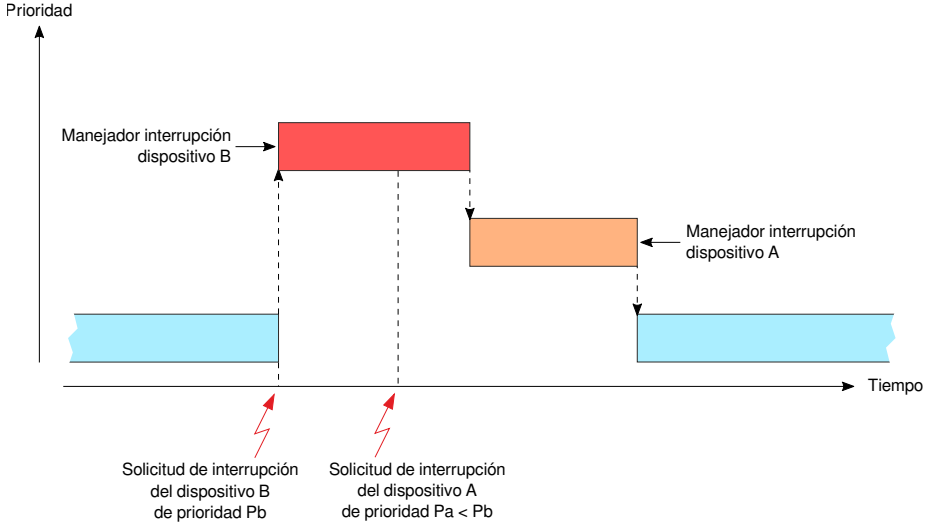
- Las interrupciones pueden deshabilitarse temporalmente para llevar a cabo una operación que no debe ser interrumpida: **sección crítica**.
- Las peticiones de interrupción recibidas pueden ser atendidas posteriormente cuando las interrupciones se habiliten de nuevo.



Jerarquía de interrupciones



Jerarquía de interrupciones



Excepciones

- Una excepción es una circunstancia anómala que surge durante la ejecución del programa.
- Son causas internas al programa, no externas como las interrupciones.
- Ejemplos de causas de excepciones:
  - Intento de ejecución de una instrucción no válida.
  - Acceso a una zona de memoria protegida.
  - División por 0.
- Cuando se produce una excepción, el programa actual se suspende y la CPU ejecuta una función manejadora de excepción.
- Después de resolver el problema (si es posible) el manejador de excepción continua el programa suspendido.

Excepciones e interrupciones en las CPU ARM Cortex-M

- **Sólo estudiaremos los detalles básicos**.
  - Para ampliar: The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, Joseph Yiu.
- Las CPU Cortex-M pueden procesar:
  - Excepciones del sistema.
  - Interrupciones.
- Las interrupciones son consideradas un tipo de excepción.
- Cada excepción tiene asignado un número entre 1 y 255.
  - Los números 1 a 15 están reservados para las excepciones del sistema.
  - Los números 16 a 255 se usan para interrupciones.
- Cada excepción/interrupción tiene asociado un número de prioridad (fijo o programable).
  - Cuanto más bajo es número de prioridad, mayor es el nivel de prioridad.

Lista de excepciones de las CPU ARM Cortex-M3/M4

Nº excep.	Tipo excepción	Prioridad	Descripción
1	Reset	–3 (la más alta)	Reset
2	NMI	–2	Interrupción no enmascarable
3	Hard Fault	–1	Manejador de fallas por defecto
4	MemManage Fault	Programable	Falla del sistema de memoria
5	Bus Fault	Programable	Falla del sistema de buses
6	Usage Fault	Programable	Instrucción no válida
7–10	Reservados	–	–
11	SVC	Programable	Llamada al supervisor
12	Debug monitor	Programable	Evento de depuración
13	Reservado	–	–
14	PendSV	Programable	Llamada de servicio pendiente
15	SYSTICK	Programable	Temporizador System Tick
16	Interrupción 0	Programable	Int. de periférico o exterior
17	Interrupción 1	Programable	Int. de periférico o exterior
...	...	...	...
255	Interrupción 239	Programable	Int. de periférico o exterior

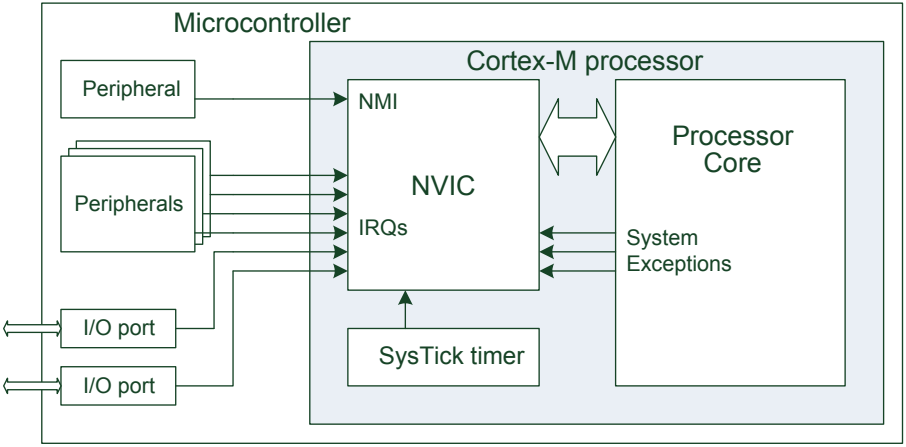
La interrupción no enmascarable NMI

- Interrupción no enmascarable (*Non-Maskable Interrupt*).
- Es la interrupción/excepción de más alta prioridad salvo el Reset.
- No enmascarable significa que no se puede deshabilitar.
- Se usa para atender eventos críticos, por ejemplo un fallo en la alimentación del sistema.

El controlador de interrupciones NVIC

- En las CPU Cortex-M las interrupciones y excepciones son gestionadas por el NVIC (*Nested Vectored Interrupt Controller*).
- Las funciones del NVIC son:
  - Permite habilitar/deshabilitar selectivamente cada interrupción.
  - Permite definir la prioridad de cada interrupción.
  - Recibe las señales de interrupción y excepciones de sistema y las gestiona en función de su estado de habilitación y prioridad.
- La habilitación/deshabilitación y prioridad de las excepciones de sistema no dependen del NVIC sino del SCB (*System Control Block*).

Diagrama del controlador de interrupciones NVIC



Registros del NVIC del LPC4088

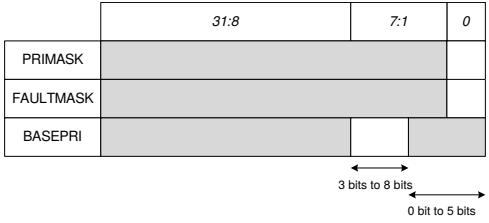
Table 51. NVIC register map

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	52
				ISER1 - 0xE000 E104	53
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	54
				ICER1 - 0xE000 E184	55
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	56
				ISPR1 - 0xE000 E204	57
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	58
				ICPR1 - 0xE000 E284	59
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	60
				IABR1 - 0xE000 E304	61
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	62
				IPR1 - 0xE000 E404	63
				IPR2 - 0xE000 E408	64
				IPR3 - 0xE000 E40C	65
				IPR4 - 0xE000 E410	66
				IPR5 - 0xE000 E414	67
				IPR6 - 0xE000 E418	68
				IPR7 - 0xE000 E41C	69
				IPR8 - 0xE000 E420	70
				IPR9 - 0xE000 E424	71
				IPR10 - 0xE000 E428	72
STIR	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	WO	-	STIR - 0xE000 EF00	73

Registros de selección de prioridad en el NVIC

- La prioridad de cada interrupción se ajusta mediante los registros IPR del NVIC (cada uno con 4 campos de 8 bits para 4 interrupciones).
- La prioridad es más alta cuanto más bajo es el valor prioridad.
- La prioridad más alta es la 0.
- El número de bits de los registros de prioridad lo determina el fabricante del microcontrolador.
- En el LPC4088 los campos de prioridad tienen implementados 5 bits:
  - Existen 32 niveles de prioridad diferentes.
  - Prioridad más alta: 0
  - Prioridad más baja: 31 (valor en el registro $>$ 248).

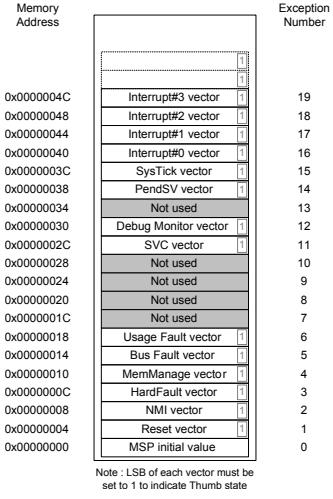
Registros PRIMASK, FAULTMASK y BASEPRI



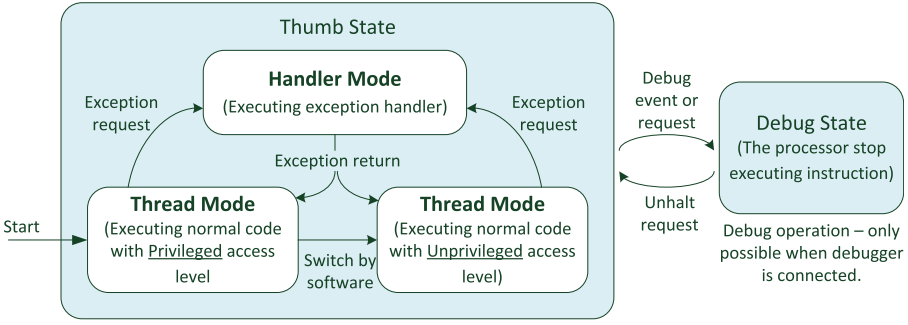
- **PRIMASK**: si está a 1 deshabilita todas las interrupciones y excepciones, salvo Reset, NMI y HardFault. A 1 tras el reset.
- **FAULTMASK**: si está a 1, deshabilita todas las interrupciones y excepciones, salvo Reset y NMI. A 0 tras el reset.
- **BASEPRI**: si $\neq 0$, deshabilita interrupciones con número de prioridad mayor o igual que su valor. A 0 tras el reset.

Tabla de vectores de excepción

- Cuando acepta una interrupción/excepción, **la CPU necesita conocer dónde está el manejador de interrupción/excepción apropiado**.
- **La tabla de vectores de excepción guarda la dirección de comienzo de cada manejador de interrupción/excepción**.
- Por defecto, la tabla de vectores de excepción está situada en la dirección 0 (reubicable mediante el registro VTOR).
- La herramienta de desarrollo define automáticamente su contenido.



Modos de operación del procesador



Secuencia simplificada de atención a una excepción

- **Condiciones de aceptación**:
  - El procesador está ejecutando (no detenido o en reset).
  - La excepción/interrupción está habilitada.
  - La excepción/interrupción tiene mayor prioridad que el nivel actual.
  - La excepción/interrupción no está enmascarada por PRIMASK, FAULTMASK o BASEPRI.
- **Secuencia de entrada (por hardware)**:
  1. Se guardan en la pila varios registros de la CPU.
  2. Se obtiene la dirección del vector de interrupción.
  3. Actualización de los registros del NVIC, PSR, LR, PC y SP.
  4. Comienzan a ejecutarse las instrucciones del manejador.
- **Secuencia de salida (por hardware)**:
  1. Se recuperan de la pila los registros de la CPU guardados.
  2. Actualización de los registros del NVIC, PSR, LR, PC y SP.
  3. Se prosigue la ejecución del programa interrumpido.

El estándar CMSIS

- CMSIS: *Cortex Microcontroller Software Interface Standard*
- Conjunto de funciones desarrolladas por ARM para estandarizar la forma de acceder a los recursos del procesador y los periféricos de los microcontroladores con CPU Cortex-M.
- Incluye funciones para facilitar la programación del NVIC.
  - Usaremos estas funciones en lugar de acceder directamente a los registros del NVIC.

Funciones CMSIS para acceder al NVIC

- `void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)`
- `uint32_t NVIC_GetPriorityGrouping(void)`
- `void NVIC_EnableIRQ(IRQn_Type IRQn)`
- `void NVIC_DisableIRQ(IRQn_Type IRQn)`
- `uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)`
- `void NVIC_SetPendingIRQ(IRQn_Type IRQn)`
- `void NVIC_ClearPendingIRQ(IRQn_Type IRQn)`
- `uint32_t NVIC_GetActive(IRQn_Type IRQn)`
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)`
- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)`
- `uint32_t NVIC_EncodePriority(uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)`
- `void NVIC_DecodePriority(uint32_t Priority, uint32_t PriorityGroup, uint32_t *pPreemptPriority, uint32_t *pSubPriority)`
- `void NVIC_SystemReset(void)`

Funciones CMSIS para acceder a PRIMASK, FAULTMASK y BASEPRI

- BASEPRI
  - `void __enable_irq(void)`
  - `void __disable_irq(void)`
  - `void __set_PRIMASK(uint32_t priMask)`
  - `uint32_t __get_PRIMASK(void)`
- FAULTMASK
  - `void __set_FAULTMASK(uint32_t faultMask)`
  - `unint32_t __get_FAULTMASK(void)`
- BASEPRI
  - `void __set_BASEPRI(uint32_t basePri)`
  - `unint32_t __get_BASEPRI(void)`

Funciones que usaremos más

- **`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.
- **`void __enable_irq(void)`**
  Habilitar interrupciones.
- **`void __disable_irq(void)`**
  Deshabilitar interrupciones.

Fuentes de interrupción en el LPC4088 (1/2)

Nº int.	IRQn_Type	Nº excep.	Offset vector	Fuente
0	WDT_IRQn	16	0x40	Watchdog
1	TIMER0_IRQn	17	0x44	Timer 0
2	TIMER1_IRQn	18	0x48	Timer 1
3	TIMER2_IRQn	19	0x4C	Timer 2
4	TIMER3_IRQn	20	0x50	Timer 3
5	UART0_IRQn	21	0x54	UART 0
6	UART1_IRQn	22	0x58	UART 1
7	UART2_IRQn	23	0x5C	UART 2
8	UART3_IRQn	24	0x60	UART 3
9	PWM1_IRQn	25	0x64	PWM 1
10	I2C0_IRQn	26	0x68	I2C0
11	I2C1_IRQn	27	0x6C	I2C1
12	I2C2_IRQn	28	0x70	I2C2
13	–	29	0x74	No usado
14	SSP0_IRQn	30	0x78	SSP0
15	SSP1_IRQn	31	0x7C	SSP1
16	PLL0_IRQn	32	0x80	PLL0
17	RTC_IRQn	33	0x84	RTC y monitor de eventos
18	EINT0_IRQn	34	0x88	Int. externa EINT0
19	EINT1_IRQn	35	0x8C	Int. externa EINT1
20	EINT2_IRQn	36	0x90	Int. externa EINT2

Fuentes de interrupción en el LPC4088 (2/2)

Nº int.	IRQn_Type	Nº excep.	Offset vector	Fuente
21	EINT3_IRQn	37	0x94	Int. externa EINT3
22	ADC_IRQn	38	0x98	ADC
23	BOD_IRQn	39	0x9C	BOD
24	USB_IRQn	40	0xA0	USB
25	CAN_IRQn	41	0xA4	CAN
26	DMA_IRQn	42	0xA8	Controlador DMA
27	I2S_IRQn	43	0xAC	I2S
28	ENET_IRQn	44	0xB0	Ethernet
29	MCI_IRQn	45	0xB4	Tarjeta SD
30	MCPWM_IRQn	46	0xB8	Controlador motor
31	QEI_IRQn	47	0xBC	Encoder cuadratura
32	PLL1_IRQn	48	0xC0	PLL1
33	USBActivity_IRQn	49	0xC4	Actividad USB
34	CANActivity_IRQn	50	0xC8	Actividad CAN
35	UART4_IRQn	51	0xCC	UART4
36	SSP2_IRQn	52	0xD0	SSP2
37	LCD_IRQn	53	0xD4	Controlador LCD
38	GPIO_IRQn	54	0xD8	GPIO
39	PWM0_IRQn	55	0xDC	PWM0
40	EEPROM_IRQn	56	0xE0	EEPROM
41	CMP0_IRQn	57	0xE4	Comp. analógico 0
42	CMP1_IRQn	58	0xE8	Comp. analógico 1

Manejadores de excepciones e interrupciones en C

- En la mayoría de microcontroladores las funciones manejadoras de excepción e interrupción necesitan características especiales.
- En cambio, el mecanismo de excepción e interrupciones de las CPU ARM Cortex permite que las funciones manejadoras de interrupciones sean funciones normales en C.
- Los nombres de las funciones de excepción e interrupción están predefinidos en los ficheros de inicialización CMSIS proporcionados por el fabricante del microcontrolador.
- Debemos definir nuestras funciones manejadoras de excepción e interrupción usando estos nombres.

Nombres de las funciones de excepción e interrupción del LPC4088

NMI_Handler	HardFault_Handler	MemManage_Handler
BusFault_Handler	UsageFault_Handler	SVC_Handler
DebugMon_Handler	PendSV_Handler	SysTick_Handler
WDT_IRQHandler	TIMER0_IRQHandler	TIMER1_IRQHandler
TIMER2_IRQHandler	TIMER3_IRQHandler	UART0_IRQHandler
UART1_IRQHandler	UART2_IRQHandler	UART3_IRQHandler
PWM1_IRQHandler	I2C0_IRQHandler	I2C1_IRQHandler
I2C2_IRQHandler	SSP0_IRQHandler	SSP1_IRQHandler
PLL0_IRQHandler	RTC_IRQHandler	EINT0_IRQHandler
EINT1_IRQHandler	EINT2_IRQHandler	EINT3_IRQHandler
ADC_IRQHandler	BOD_IRQHandler	USB_IRQHandler
CAN_IRQHandler	DMA_IRQHandler	I2S_IRQHandler
ENET_IRQHandler	MCI_IRQHandler	MCPWM_IRQHandler
QEI_IRQHandler	PLL1_IRQHandler	USBActivity_IRQHandler
CANActivity_IRQHandler	UART4_IRQHandler	SSP2_IRQHandler
LCD_IRQHandler	GPIO_IRQHandler	PWM0_IRQHandler
EEPROM_IRQHandler		

Interrupciones de los timers