INTERRUPCIONES

Consideraciones prácticas

Interrupciones – Cortex M3

Hardware:

» Controlador de interrupciones

» Periférico que genera la interrupción

» Pin, si es necesario, por el que esta se va a manifestar.

Software:

» Inicialización de vectores

» Inicialización del controlador de interrupciones

» Inicialización del periférico que generará la interrupción.

» Inicialización de los pines asociados.

¿Como se configuran?

1. Inicializar la tabla de vectores.
2. Configurar el NVIC.
3. Configurar el periférico correspondiente y los pines asociados.
4. Escribir la rutina de servicio de interrupción (ISR).

Tabla de vectores

Es la tabla que contiene la dirección de las rutinas que serán ejecutadas ante la ocurrencia y procesamiento de cada interrupción. Está armada con array de punteros a funciones.

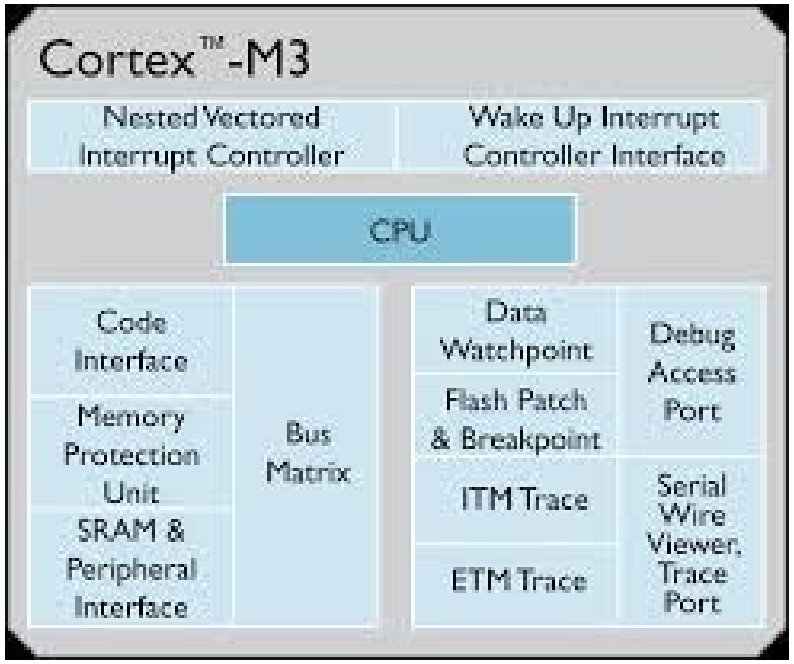
```
//*****************************************************************************
// The vector table.
// This relies on the linker script to place at correct location in memory.
//*****************************************************************************
extern void (* const g_pfnVectors[])(void);
__attribute__ ((section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
// Core Level - CM3
&_vStackTop,       // The initial stack pointer
ResetISR,          // The reset handler
NMI_Handler,       // The NMI handler
HardFault_Handler, // The hard fault handler
MemManage_Handler, // The MPU fault handler
BusFault_Handler,  // The bus fault handler
UsageFault_Handler,// The usage fault handler
0,           // Reserved
0,           // Reserved
0,           // Reserved
0,           // Reserved
SVC_Handler,       // SVCall handler
DebugMon_Handler,  // Debug monitor handler
0,           // Reserved
PendSV_Handler,    // The PendSV handler
SysTick_Handler,   // The SysTick handler
```

Tabla de vectores

```
// Chip Level - LPC17
WDT_IRQHandler,      // 16, 0x40 - WDT
TIMER0_IRQHandler,   // 17, 0x44 - TIMER0
TIMER1_IRQHandler,   // 18, 0x48 - TIMER1
TIMER2_IRQHandler,   // 19, 0x4c - TIMER2
TIMER3_IRQHandler,   // 20, 0x50 - TIMER3
UART0_IRQHandler,    // 21, 0x54 - UART0
UART1_IRQHandler,    // 22, 0x58 - UART1
UART2_IRQHandler,    // 23, 0x5c - UART2
UART3_IRQHandler,    // 24, 0x60 - UART3
PWM1_IRQHandler,     // 25, 0x64 - PWM1
I2C0_IRQHandler,     // 26, 0x68 - I2C0
I2C1_IRQHandler,     // 27, 0x6c - I2C1
I2C2_IRQHandler,     // 28, 0x70 - I2C2
SPI_IRQHandler,      // 29, 0x74 - SPI
SSP0_IRQHandler,     // 30, 0x78 - SSP0
SSP1_IRQHandler,     // 31, 0x7c - SSP1
PLL0_IRQHandler,     // 32, 0x80 - PLL0 (Main PLL)
RTC_IRQHandler,      // 33, 0x84 - RTC
EINT0_IRQHandler,    // 34, 0x88 - EINT0
EINT1_IRQHandler,    // 35, 0x8c - EINT1
EINT2_IRQHandler,    // 36, 0x90 - EINT2
EINT3_IRQHandler,    // 37, 0x94 - EINT3
ADC_IRQHandler,      // 38, 0x98 - ADC
BOD_IRQHandler,      // 39, 0x9c - BOD
USB_IRQHandler,      // 40, 0xA0 - USB
CAN_IRQHandler,      // 41, 0xa4 - CAN
DMA_IRQHandler,      // 42, 0xa8 - GP DMA
I2S_IRQHandler,      // 43, 0xac - I2S
ENET_IRQHandler,     // 44, 0xb0 - Ethernet
RIT_IRQHandler,      // 45, 0xb4 - RITINT
MCPWM_IRQHandler,    // 46, 0xb8 - Motor Control PWM
QEI_IRQHandler,      // 47, 0xbc - Quadrature Encoder
PLL1_IRQHandler,     // 48, 0xc0 - PLL1 (USB PLL)
USBActivity_IRQHandler,      // 49, 0xc4 - USB Activity interrupt to wakeup
CANActivity_IRQHandler,      // 50, 0xc8 - CAN Activity interrupt to wakeup
};
```

Controlador de Interrupciones

- El **NVIC** (Nested Vectorized Interrupt Controller) es un controlador de interrupciones que está en el CORE, es parte de lo que los fabricantes compran a ARM al licenciar el Cortex M3.



Interrupciones - NVIC

- El NVIC es parte del core Cortex M3. Todos sus registros de control están mapeados en memoria y permiten el control de interrupciones, systick y debugger.

- El NVIC soporta 240 entradas externas de interrupción. El número de entradas usadas lo determina el fabricante del chip, para el LCP1769 son 35. El NVIC tiene también una entrada NMI.

- Su implementación también la decide el fabricante. Está implementada en el LPC1769 (Pin 53, P2.10).

Interrupciones - NVIC

- Las entradas de excepción del Cortex M3 de 0 a 15 son internas del core y a partir de la excepción 16 comienzan las entradas de interrupción externa.

- El NVIC puede trabajar con interrupciones anidadas y con interrupciones encadenadas. Lleva la cuenta de las que están pendientes.

- Además puede desplazar de la ejecución a una interrupción para dar paso a otra más prioritaria.

Interrupciones y su configuración

- Configuración del controlador de interrupciones NVIC.
- Configuración de las interrupciones deseadas en el periférico correspondiente.
- Configuraciones de la funcionalidad de los pines para que todo funcione.

NVIC - Configuración

Table 51. NVIC register map

Name	Description	Access	Reset value	Address
ISER0 to ISER1	Interrupt Set-Enable Registers. These 2 registers allow enabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ISER0 - 0xE000 E100 ISER1 - 0xE000 E104
ICER0 to ICER1	Interrupt Clear-Enable Registers. These 2 registers allow disabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ICER0 - 0xE000 E180 ICER1 - 0xE000 E184
ISPR0 to ISPR1	Interrupt Set-Pending Registers. These 2 registers allow changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ISPR0 - 0xE000 E200 ISPR1 - 0xE000 E204
ICPR0 to ICPR1	Interrupt Clear-Pending Registers. These 2 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 ICPR1 - 0xE000 E284
IABR0 to IABR1	Interrupt Active Bit Registers. These 2 registers allow reading the current interrupt active state for specific peripheral functions.	RO	0	IABR0 - 0xE000 E300 IABR1 - 0xE000 E304
IPR0 to IPR8	Interrupt Priority Registers. These 9 registers allow assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	RW	0	IPR0 - 0xE000 E400 IPR1 - 0xE000 E404 IPR2 - 0xE000 E408 IPR3 - 0xE000 E40C IPR4 - 0xE000 E410 IPR5 - 0xE000 E414 IPR6 - 0xE000 E418 IPR7 - 0xE000 E41C IPR8 - 0xE000 E420
STIR	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	WO	0	STIR - 0xE000 EF00

Interrupciones - Registros

En el NVIC hay 5 grupos de registros que controlan su funcionamiento:

- **ISER e ICER 0 y 1:** sirven para habilitar y deshabilitar cada fuente de interrupción.
- **ISPR e ICPR 0 y 1:** sirven para setear y resetear los flags de interrupciones pendientes.
- **IABR 0 y 1:** sirven para indicar las interrupciones activas.
- **IPR 0 al 8:** sirven para configurar el nivel de prioridad de cada interrupción.
- **STIR:** permite generar interrupciones de software. Para cualquier vector o fuente externa.

ISER0: Habilitar interrupciones

Table 52. Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100)

Bit	Name	Function
0	ISE_WDT	Watchdog Timer Interrupt Enable. Write: writing 0 has no effect, writing 1 enables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ISE_TIMER0	Timer 0 Interrupt Enable. See functional description for bit 0.
2	ISE_TIMER1	Timer 1. Interrupt Enable. See functional description for bit 0.
3	ISE_TIMER2	Timer 2 Interrupt Enable. See functional description for bit 0.
4	ISE_TIMER3	Timer 3 Interrupt Enable. See functional description for bit 0.
18	ISE_EINT0	External Interrupt 0 Interrupt Enable. See functional description for bit 0.
19	ISE_EINT1	External Interrupt 1 Interrupt Enable. See functional description for bit 0.
20	ISE_EINT2	External Interrupt 2 Interrupt Enable. See functional description for bit 0.
21	ISE_EINT3	External Interrupt 3 Interrupt Enable. See functional description for bit 0.
22	ISE_ADC	ADC Interrupt Enable. See functional description for bit 0.
23	ISE_BOD	BOD Interrupt Enable. See functional description for bit 0.
24	ISE_USB	USB Interrupt Enable. See functional description for bit 0.
25	ISE_CAN	CAN Interrupt Enable. See functional description for bit 0.
26	ISE_DMA	GPDMA Interrupt Enable. See functional description for bit 0.
27	ISE_I2S	I^2S Interrupt Enable. See functional description for bit 0.
28	ISE_ENET	Ethernet Interrupt Enable. See functional description for bit 0.
29	ISE_RIT	Repetitive Interrupt Timer Interrupt Enable. See functional description for bit 0.
30	ISE_MCPWM	Motor Control PWM Interrupt Enable. See functional description for bit 0.
31	ISE_QEI	Quadrature Encoder Interface Interrupt Enable. See functional description for bit 0.

ISR (Interrupt Service Rutine)

Las ISR son las rutinas a donde va a la ejecución del programa cuando el CPU atiende una interrupción

```
//*****************************************************************************
*
//
// Forward declaration of the default handlers. These are aliased.
// When the application defines a handler (with the same name), this will
// automatically take precedence over these weak definitions
//
//*****************************************************************************
*
     void ResetISR(void);
WEAK void NMI_Handler(void);
WEAK void HardFault_Handler(void);
WEAK void MemManage_Handler(void);
WEAK void BusFault_Handler(void);
WEAK void UsageFault_Handler(void);
WEAK void SVCall_Handler(void);
WEAK void DebugMon_Handler(void);
WEAK void PendSV_Handler(void);
WEAK void SysTick_Handler(void);
WEAK void IntDefaultHandler(void);
```

ISR - declaraciones

```
//*****************************************************************************
// Forward declaration of the specific IRQ handlers. These are aliased
// to the IntDefaultHandler, which is a 'forever' loop. When the application
// defines a handler (with the same name), this will automatically take
// precedence over these weak definitions
//*****************************************************************************
void WDT_IRQHandler(void) ALIAS(IntDefaultHandler);
void TIMER0_IRQHandler(void) ALIAS(IntDefaultHandler);
void TIMER1_IRQHandler(void) ALIAS(IntDefaultHandler);
void TIMER2_IRQHandler(void) ALIAS(IntDefaultHandler);
void TIMER3_IRQHandler(void) ALIAS(IntDefaultHandler);
void UART0_IRQHandler(void) ALIAS(IntDefaultHandler);
void UART1_IRQHandler(void) ALIAS(IntDefaultHandler);
void UART2_IRQHandler(void) ALIAS(IntDefaultHandler);
void UART3_IRQHandler(void) ALIAS(IntDefaultHandler);
void PWM1_IRQHandler(void) ALIAS(IntDefaultHandler);
void I2C0_IRQHandler(void) ALIAS(IntDefaultHandler);
void I2C1_IRQHandler(void) ALIAS(IntDefaultHandler);
void I2C2_IRQHandler(void) ALIAS(IntDefaultHandler);
void SPI_IRQHandler(void) ALIAS(IntDefaultHandler);
void SSP0_IRQHandler(void) ALIAS(IntDefaultHandler);
void SSP1_IRQHandler(void) ALIAS(IntDefaultHandler);
void PLL0_IRQHandler(void) ALIAS(IntDefaultHandler);
void RTC_IRQHandler(void) ALIAS(IntDefaultHandler);
void EINT0_IRQHandler(void) ALIAS(IntDefaultHandler);
void EINT1_IRQHandler(void) ALIAS(IntDefaultHandler);
void EINT2_IRQHandler(void) ALIAS(IntDefaultHandler);
void EINT3_IRQHandler(void) ALIAS(IntDefaultHandler);
void ADC_IRQHandler(void) ALIAS(IntDefaultHandler);
```

ISR

```
//*****************************************************************************
// Default exception handlers. Override the ones here by defining your own
// handler routines in your application code.
//*****************************************************************************
__attribute__ ((section(".after_vectors")))
void NMI_Handler(void)
{
    while(1)
    {
    }
}
                __attribute__ ((section(".after_vectors")))
                void SysTick_Handler(void)
                {
                    while(1)
                    {
                    }
                }
                                //*****************************************************************************
                                // Processor ends up here if an unexpected interrupt occurs or a
                                specific
                                // handler is not present in the application code.
                                //*****************************************************************************
                                __attribute__ ((section(".after_vectors")))
                                void IntDefaultHandler(void)
                                {
                                    while(1)
                                    {
                                    }
                                }
```

Modificador WEAK

```
#define WEAK __attribute__ ((weak))
```

- WEAK significa que esa función será pisada por otra con el mismo nombre.
- Si no hay otra con el mismo nombre, entonces es válida.
- Se usa esto para definir funciones default.

ALIAS

```
#define ALIAS(f) __attribute__ ((weak, alias (#f)))
```

- ALIAS significa que el nombre de la función definida refiere a otra función.
- Se usa para hacer que muchas funciones se refieran a un misma función. Es una función default que debe estar definida si o si por seguridad.
- El modificador ALIAS está definido para que incluya también al atributo weak, por lo tanto las funciones ISR (Rutinas de servicio de interrupciones) están declaradas por default como ALIAS lo que implica que todas se refieren a otra común y que son válidas siempre y cuando no se defina otra con el mismo nombre que no sea WEAK.

Ejemplo declaración de ISR

cr_startup_lpc176x.c

```
 void EINT3_IRQHandler(void) ALIAS(IntDefaultHandler);
//*****************************************************************************
// Processor ends up here if an unexpected interrupt occurs or a specific
// handler is not present in the application code.
//*****************************************************************************
__attribute__ ((section(".after_vectors")))
void IntDefaultHandler(void)
{
    while(1)
    {
    }
}
```

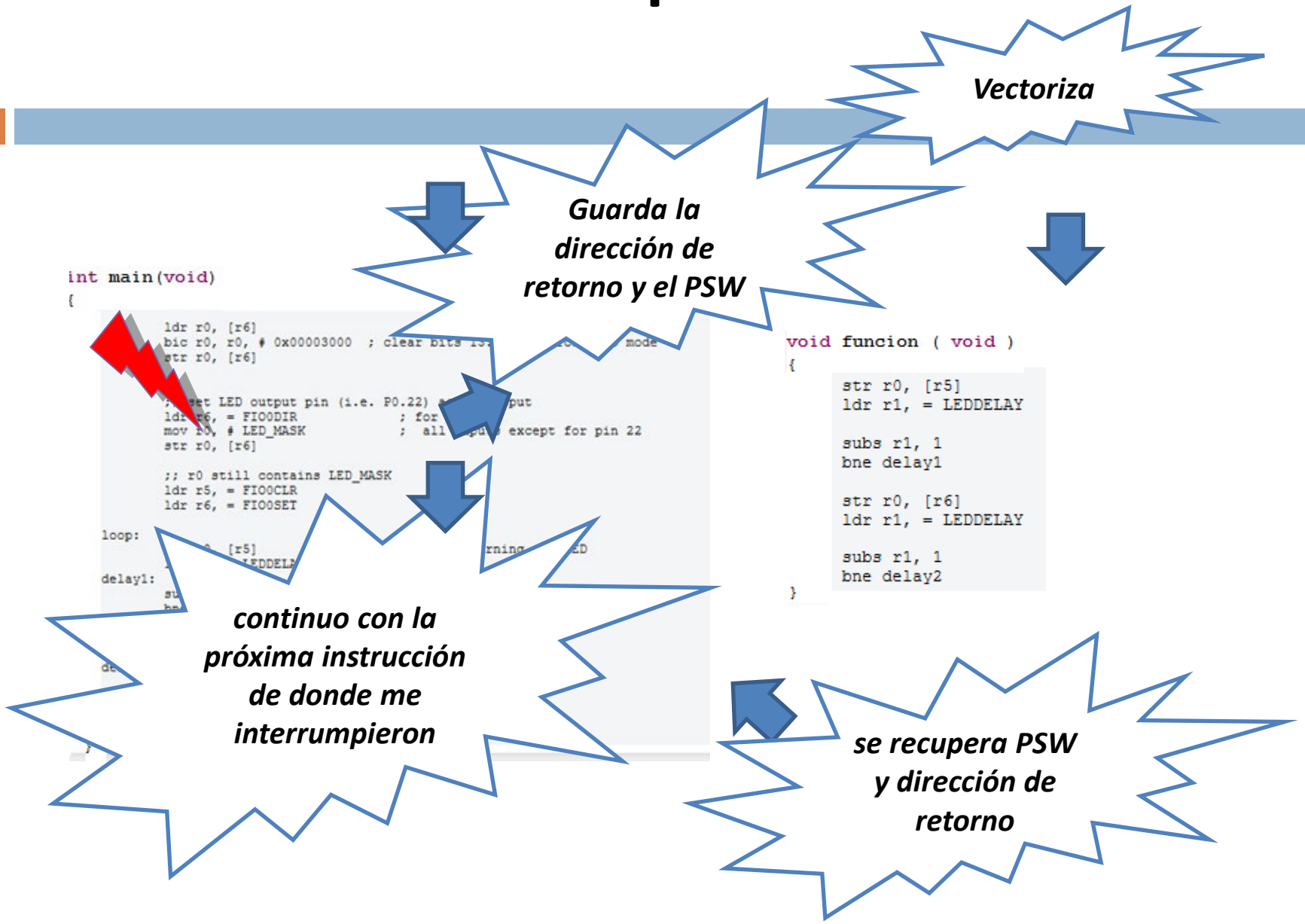
main.c

```
 void EINT3_IRQHandler(void){

      IO2IntClr |= (0x01 << 13);                  // Limpio flag

      if(flagLED){
            SetPIN (LEDXpresso,0);                // Pone pin LED del stick en 1
            flagLED = 0;
      }
      else{
            SetPIN (LEDXpresso,1);                // Pone pin LED del stick en 0
            flagLED = 1;
      }
 }
```

Interrupciones

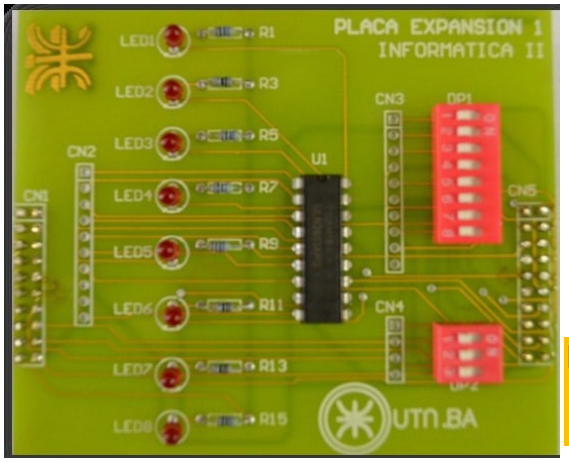


Interrupciones Externas

Implementación

Interrupción EXTERNA	Puerto	Infotronic	Función
EINT0	P210	SW1	KEY0
EINT1	P211	ENT.DIG2	BORNERA
EINT2	P212	EXPANSION 17	DIP2_2
EINT3	P213	SW10	KEY3

KEY4 P126 | **KEY3 P213 EINT3** | **KEY2 P011** | **KEY1 P018** | **KEY0 P210 EINT0**



DIP2-2 P212 EINT0

Interrupciones – Externas y GPIO

- El LPC1769 tiene 4 fuentes de interrupción externa. Estas fuentes se identifican de 0 a 3.
- Las interrupciones externas se encuentran disponibles en los pines del LPC1769 como parte de las funciones alternativas de los pines.

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	$\overline{\text{EINT0}}$	NMI	Reserved	00
23:22	P2.11[1]	GPIO Port 2.11	$\overline{\text{EINT1}}$	Reserved	I2STX_CLK	00
25:24	P2.12[1]	GPIO Port 2.12	$\overline{\text{EINT2}}$	Reserved	I2STX_WS	00
27:26	P2.13[1]	GPIO Port 2.13	$\overline{\text{EINT3}}$	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

```
EINT0_IRQHandler,// 34, 0x88 - EINT0
EINT1_IRQHandler,// 35, 0x8c - EINT1
EINT2_IRQHandler,// 36, 0x90 - EINT2
EINT3_IRQHandler,// 37, 0x94 - EINT3
```

Interrupciones Externas - Configuración

Pin name	Pin direction	Pin description
EINT0	Input	**External Interrupt Input 0** - An active low/high level or falling/rising edge general purpose interrupt input. This pin may be used to wake up the processor from Sleep, Deep-sleep, or Power-down modes.
EINT1	Input	**External Interrupt Input 1** - See the EINT0 description above.
EINT2	Input	**External Interrupt Input 2** - See the EINT0 description above.
EINT3	Input	**External Interrupt Input 3** - See the EINT0 description above.
$\overline{\text{RESET}}$	Input	**External Reset input** - A LOW on this pin resets the chip, causing I/O ports and peripherals to take on their default states, and the processor to begin execution at address 0x0000 0000.

Registros

Name	Description	Access	Reset value	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 10.	R/W	0x00	0x400F C140
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive. See Table 11.	R/W	0x00	0x400F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. See Table 12.	R/W	0x00	0x400F C14C

EXTINT

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

EXTMODE

Indica si la interrupción se activará por nivel o flanco.

Si el bit correspondiente vale 0, la interrupción será activa por nivel y si vale 1 por flanco.

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for $\overline{\text{EINT0}}$.	0
		1	$\overline{\text{EINT0}}$ is edge sensitive.	
1	EXTMODE1	0	Level-sensitivity is selected for $\overline{\text{EINT1}}$.	0
		1	$\overline{\text{EINT1}}$ is edge sensitive.	
2	EXTMODE2	0	Level-sensitivity is selected for $\overline{\text{EINT2}}$.	0
		1	$\overline{\text{EINT2}}$ is edge sensitive.	
3	EXTMODE3	0	Level-sensitivity is selected for $\overline{\text{EINT3}}$.	0
		1	$\overline{\text{EINT3}}$ is edge sensitive.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

EXTPOLAR

Indica si la interrupción correspondiente será activa por nivel bajo/alto o por flanco descendente o ascendente. Esto dependerá de la elección hecha en EXTMODE

Bit	Symbol	Value	Description	Reset value
0	EXTPOLAR0	0	$\overline{\text{EINT0}}$ is low-active or falling-edge sensitive (depending on EXTMODE0).	0
		1	$\overline{\text{EINT0}}$ is high-active or rising-edge sensitive (depending on EXTMODE0).	
1	EXTPOLAR1	0	$\overline{\text{EINT1}}$ is low-active or falling-edge sensitive (depending on EXTMODE1).	0
		1	$\overline{\text{EINT1}}$ is high-active or rising-edge sensitive (depending on EXTMODE1).	
2	EXTPOLAR2	0	$\overline{\text{EINT2}}$ is low-active or falling-edge sensitive (depending on EXTMODE2).	0
		1	$\overline{\text{EINT2}}$ is high-active or rising-edge sensitive (depending on EXTMODE2).	
3	EXTPOLAR3	0	EINT3 is low-active or falling-edge sensitive (depending on EXTMODE3).	0
		1	$\overline{\text{EINT3}}$ is high-active or rising-edge sensitive (depending on EXTMODE3).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Receta de configuración

Interrupción EXTERNA	Puerto	Infotronic	Función
EINT0	P210	SW1	KEY0
EINT1	P211	ENT.DIG2	BORNERA
EINT2	P212	EXPANSION 17	DIP2_2
EINT3	P213	SW10	KEY3

```
//-----------------------------------------------------------------------------
// Configuración de interrupciones externas
//-----------------------------------------------------------------------------
void InicExtInt(void){

// Configuracion interrupcion externa EINT3, EINT2, EINT1, EINT0
SetPINSEL(KEY0, PINSEL_FUNC1);              // Configuro el P210 como interrup Ext EINT 0
SetPINSEL(KEY3, PINSEL_FUNC1);              // Configuro el P213 como interrup Ext EINT 3
EXTMODE=(0x0F);                             // Todas x Flanco
EXTPOLAR=(0x00);                                 // Todas Polaridad activo bajo
ISER0 |=(0x01 <<18);                        // Habilito Interrupcion externa 0
ISER0 |=(0x01 <<21);                        // Habilito Interrupcion externa 3

}
```

Agradecimientos

Ing. Marcelo Giura

Ing. Gabriel Soccodato