**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PINSEL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\***

\* @file lpc17xx\_pinsel.h

\* @brief Contiene todas las definiciones de macros y prototipos de funciones

\* compatibles con la librería de firmware del bloque de conexión de pines en el LPC17xx.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* El software descrito aquí es solo para fines ilustrativos y proporciona

\* a los clientes información de programación sobre los productos.

\* Este software se suministra "TAL CUAL" sin garantías de ningún tipo.

\* NXP Semiconductors no asume ninguna responsabilidad ni obligación

\* por el uso del software, y no transfiere ninguna licencia ni título

\* bajo ninguna patente, derecho de autor, o derecho sobre máscaras

\* del producto. NXP Semiconductors se reserva el derecho de realizar

\* cambios en el software sin notificación previa. NXP Semiconductors

\* tampoco hace ninguna representación o garantía de que dicha aplicación

\* sea adecuada para el uso especificado sin más pruebas o modificaciones.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Inclusiones ---------------------------------------------------------------- \*/

#include "lpc17xx.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\*!< Macros para la selección de PUERTO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PINSEL\_PORT\_0 ((0)) /\*\*< PUERTO 0 \*/

#define PINSEL\_PORT\_1 ((1)) /\*\*< PUERTO 1 \*/

#define PINSEL\_PORT\_2 ((2)) /\*\*< PUERTO 2 \*/

#define PINSEL\_PORT\_3 ((3)) /\*\*< PUERTO 3 \*/

#define PINSEL\_PORT\_4 ((4)) /\*\*< PUERTO 4 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Macros para la selección de función de pin

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PINSEL\_FUNC\_0 ((0)) /\*\*< función predeterminada \*/

#define PINSEL\_FUNC\_1 ((1)) /\*\*< primera función alternativa \*/

#define PINSEL\_FUNC\_2 ((2)) /\*\*< segunda función alternativa \*/

#define PINSEL\_FUNC\_3 ((3)) /\*\*< tercera o función alternativa reservada \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Macros para definir el número de pin del puerto

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PINSEL\_PIN\_0 ((0)) /\*\*< Pin 0 \*/

#define PINSEL\_PIN\_1 ((1)) /\*\*< Pin 1 \*/

#define PINSEL\_PIN\_2 ((2)) /\*\*< Pin 2 \*/

.

.

.

#define PINSEL\_PIN\_29 ((29)) /\*\*< Pin 29 \*/

#define PINSEL\_PIN\_30 ((30)) /\*\*< Pin 30 \*/

#define PINSEL\_PIN\_31 ((31)) /\*\*< Pin 31 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Macros para definir el modo del pin

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PINSEL\_PINMODE\_PULLUP ((0)) /\*\*< Resistencia de pull-up interna \*/

#define PINSEL\_PINMODE\_TRISTATE ((2)) /\*\*< Triestado \*/

#define PINSEL\_PINMODE\_PULLDOWN ((3)) /\*\*< Resistencia de pull-down interna \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Macros para definir el modo del pin (normal/drenador abierto)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PINSEL\_PINMODE\_NORMAL ((0)) /\*\*< El pin está en modo normal (no drenador abierto). \*/

#define PINSEL\_PINMODE\_OPENDRAIN ((1)) /\*\*< El pin está en modo drenador abierto. \*/

/\* Tipos públicos ------------------------------------------------------------- \*/

/\*\* @defgroup PINSEL\_Public\_Types Tipos Públicos de PINSEL

\* @{

\*/

/\*\* @brief Estructura de configuración de pines \*/

typedef struct

{

uint8\_t Portnum; /\*\*< Número de puerto, debe ser PINSEL\_PORT\_x,

donde x debe estar en el rango de 0 a 4 \*/

uint8\_t Pinnum; /\*\*< Número de pin, debe ser PINSEL\_PIN\_x,

donde x debe estar en el rango de 0 a 31 \*/

uint8\_t Funcnum; /\*\*< Número de función, debe ser PINSEL\_FUNC\_x,

donde x debe estar en el rango de 0 a 3 \*/

uint8\_t Pinmode; /\*\*< Modo del pin, debe ser:

- PINSEL\_PINMODE\_PULLUP: Resistencia de pull-up interna

- PINSEL\_PINMODE\_TRISTATE: Triestado

- PINSEL\_PINMODE\_PULLDOWN: Resistencia de pull-down interna \*/

uint8\_t OpenDrain; /\*\*< Modo de drenador abierto, debe ser:

- PINSEL\_PINMODE\_NORMAL: El pin está en modo normal (no drenador abierto)

- PINSEL\_PINMODE\_OPENDRAIN: El pin está en modo drenador abierto \*/

} PINSEL\_CFG\_Type;

/\* Funciones públicas --------------------------------------------------------- \*/

/\*\* @defgroup PINSEL\_Public\_Functions Funciones Públicas de PINSEL

\* @{

\*/

PINSEL\_ConfigPin(&confPin);

**/\* --------------------------------- Fin del archivo -------------------------- \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GPIO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\***

/\* Inclusiones ------------------------------------------------------------------- \*/

#include "lpc17xx\_gpio.h"

/\* Funciones Públicas ----------------------------------------------------------- \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Configurar la dirección para el puerto GPIO.

\* @param[in] portNum Valor del Número de Puerto, debe estar en el rango de 0 a 4

\* @param[in] bitValue Valor que contiene todos los bits para configurar la dirección,

\* en el rango de 0 a 0xFFFFFFFF.

\* ejemplo: valor 0x5 para configurar la dirección de los bits 0 y 1.

\* @param[in] dir Valor de la dirección, debe ser:

\* - 0: Entrada.

\* - 1: Salida.

\* @return Ninguno

\*

\* Nota: Todos los bits restantes que no están activados en bitValue (valor '0')

\* no se verán afectados por esta función.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void **GPIO\_SetDir**(uint8\_t portNum, uint32\_t bitValue, uint8\_t dir)

{

LPC\_GPIO\_TypeDef \*pGPIO = GPIO\_GetPointer(portNum);

if (pGPIO != NULL) {

// Habilitar Salida

if (dir) {

pGPIO->FIODIR |= bitValue;

}

// Habilitar Entrada

else {

pGPIO->FIODIR &= ~bitValue;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Configurar el Valor para los bits que tienen dirección de salida en el puerto GPIO.

\* @param[in] portNum Valor del número de puerto, debe estar en el rango de 0 a 4

\* @param[in] bitValue Valor que contiene todos los bits en GPIO para configurar,

\* en el rango de 0 a 0xFFFFFFFF.

\* ejemplo: valor 0x5 para configurar los bits 0 y 1.

\* @return Ninguno

\*

\* Nota:

\* - Para todos los bits que se han configurado como dirección de entrada, esta función no

\* tendrá efecto.

\* - Todos los bits restantes que no están activados en bitValue (valor '0')

\* no se verán afectados por esta función.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GPIO\_SetValue(uint8\_t portNum, uint32\_t bitValue)

{

LPC\_GPIO\_TypeDef \*pGPIO = GPIO\_GetPointer(portNum);

if (pGPIO != NULL) {

pGPIO->FIOSET = bitValue;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Limpiar el Valor para los bits que tienen dirección de salida en el puerto GPIO.

\* @param[in] portNum Valor del número de puerto, debe estar en el rango de 0 a 4

\* @param[in] bitValue Valor que contiene todos los bits en GPIO para limpiar,

\* en el rango de 0 a 0xFFFFFFFF.

\* ejemplo: valor 0x5 para limpiar los bits 0 y 1.

\* @return Ninguno

\*

\* Nota:

\* - Para todos los bits que se han configurado como dirección de entrada, esta función no

\* tendrá efecto.

\* - Todos los bits restantes que no están activados en bitValue (valor '0')

\* no se verán afectados por esta función.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GPIO\_ClearValue(uint8\_t portNum, uint32\_t bitValue)

{

LPC\_GPIO\_TypeDef \*pGPIO = GPIO\_GetPointer(portNum);

if (pGPIO != NULL) {

pGPIO->FIOCLR = bitValue;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Leer el estado actual en el pin del puerto que tiene dirección de entrada en GPIO

\* @param[in] portNum Número de puerto para leer el valor, en el rango de 0 a 4

\* @return Valor actual del puerto GPIO.

\*

\* Nota: El valor de retorno contiene el estado de cada pin del puerto (bit) en ese GPIO, independientemente

\* de si su dirección es entrada o salida.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint32\_t **GPIO\_ReadValue**(uint8\_t portNum)

{

LPC\_GPIO\_TypeDef \*pGPIO = GPIO\_GetPointer(portNum);

if (pGPIO != NULL) {

return pGPIO->FIOPIN;

}

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Habilitar la interrupción GPIO (solo se usa para P0.0-P0.30, P2.0-P2.13)

\* @param[in] portNum Número de puerto para habilitar la interrupción, debe ser: 0 o 2

\* @param[in] bitValue Valor que contiene todos los bits en GPIO para habilitar,

\* en el rango de 0 a 0xFFFFFFFF.

\* @param[in] edgeState Estado del borde, debe ser:

\* - 0: Borde ascendente

\* - 1: Borde descendente

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void **GPIO\_IntCmd**(uint8\_t portNum, uint32\_t bitValue, uint8\_t edgeState)

{

if((portNum == 0)&&(edgeState == 0))

LPC\_GPIOINT->IO0IntEnR = bitValue;

else if ((portNum == 2)&&(edgeState == 0))

LPC\_GPIOINT->IO2IntEnR = bitValue;

else if ((portNum == 0)&&(edgeState == 1))

LPC\_GPIOINT->IO0IntEnF = bitValue;

else if ((portNum == 2)&&(edgeState == 1))

LPC\_GPIOINT->IO2IntEnF = bitValue;

else

//Error

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Obtener el estado de la interrupción GPIO (solo se usa para P0.0-P0.30, P2.0-P2.13)

\* @param[in] portNum Número de puerto para leer el valor, debe ser: 0 o 2

\* @param[in] pinNum Número de pin, debe ser: 0..30 (con puerto 0) y 0..13

\* (con puerto 2)

\* @param[in] edgeState Estado del borde, debe ser:

\* - 0: Borde ascendente

\* - 1: Borde descendente

\* @return EstadoFuncional puede ser:

\* - ENABLE: La interrupción ha sido generada debido a un borde ascendente

\* en P0.0

\* - DISABLE: No se ha detectado un borde ascendente en P0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FunctionalState **GPIO\_GetIntStatus**(uint8\_t portNum, uint32\_t pinNum, uint8\_t edgeState)

{

if((portNum == 0) && (edgeState == 0))//Borde Ascendente

return (FunctionalState)(((LPC\_GPIOINT->IO0IntStatR)>>pinNum)& 0x1);

else if ((portNum == 2) && (edgeState == 0))

return (FunctionalState)(((LPC\_GPIOINT->IO2IntStatR)>>pinNum)& 0x1);

else if ((portNum == 0) && (edgeState == 1))//Borde Descendente

return (FunctionalState)(((LPC\_GPIOINT->IO0IntStatF)>>pinNum)& 0x1);

else if ((portNum == 2) && (edgeState == 1))

return (FunctionalState)(((LPC\_GPIOINT->IO2IntStatF)>>pinNum)& 0x1);

else

//Error

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Limpiar la interrupción GPIO (solo se usa para P0.0-P0.30, P2.0-P2.13)

\* @param[in] portNum Número de puerto para leer el valor, debe ser: 0 o 2

\* @param[in] bitValue Valor que contiene todos los bits en GPIO para habilitar,

\* en el rango de 0 a 0xFFFFFFFF.

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GPIO\_ClearInt(uint8\_t portNum, uint32\_t bitValue)

{

if(portNum == 0)

LPC\_GPIOINT->IO0IntClr = bitValue;

else if (portNum == 2)

LPC\_GPIOINT->IO2IntClr = bitValue;

else

//Número de puerto inválido

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Establecer valor de máscara para bits en el puerto FIO

\* @param[in] portNum Número de puerto, en el rango de 0 a 4

\* @param[in] bitValue Valor que contiene todos los bits a establecer,

\* en el rango de 0 a 0xFFFFFFFF.

\* @param[in] maskValue Valor de máscara que contiene el estado para cada bit:

\* - 0: no enmascarar.

\* - 1: enmascarar.

\* @return Ninguno

\*

\* Nota:

\* - Todos los bits restantes que no están activados en bitValue (valor '0')

\* no se verán afectados por esta función.

\* - Después de ejecutar esta función, en el registro de máscara, el valor '0' en cada bit

\* permite el acceso al pin físico correspondiente a través de una lectura o escritura,

\* mientras que el valor '1' en el bit (enmascarado) significa que el pin correspondiente no se

\* cambiará con acceso de escritura y, si se lee, no se reflejará en el pin actualizado.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void FIO\_SetMask(uint8\_t portNum, uint32\_t bitValue, uint8\_t maskValue)

{

LPC\_GPIO\_TypeDef \*pFIO = GPIO\_GetPointer(portNum);

if(pFIO != NULL) {

// Enmascarar

if (maskValue){

pFIO->FIOMASK |= bitValue;

}

// Desenmascarar

else {

pFIO->FIOMASK &= ~bitValue;

}

}

}

**/\* --------------------------------- Fin del archivo -------------------------- \*/**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INTERRUPCIONES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void EINT0\_IRQHandler(void)

void EINT1\_IRQHandler(void)

void EINT2\_IRQHandler(void)

void EINT3\_IRQHandler(void)

void SysTick\_Handler(void);

**/\* --------------------------------- Fin del archivo -------------------------- \*/**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* EXTERNAS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Incluye ------------------------------------------------------------------- \*/

#include "lpc17xx\_exti.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Inicialización de EXT

\* - Establece los registros EXTINT, EXTMODE, EXTPOLAR a su valor predeterminado

\* @param[in] **Ninguno**

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EXTI\_Init(void)

{

LPC\_SC->EXTINT = 0xF;

LPC\_SC->EXTMODE = 0x0;

LPC\_SC->EXTPOLAR = 0x0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Configuración para EXT

\* - Establece los registros EXTINT, EXTMODE, EXTPOLAR

\* @param[in] EXTICfg Puntero a una estructura EXTI\_InitTypeDef

\* que contiene la información de configuración para la

\* interrupción externa especificada

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EXTI\_Config(EXTI\_InitTypeDef \*EXTICfg)

{

LPC\_SC->EXTINT = 0x0;

EXTI\_SetMode(EXTICfg->EXTI\_Line, EXTICfg->EXTI\_Mode);

EXTI\_SetPolarity(EXTICfg->EXTI\_Line, EXTICfg->EXTI\_polarity);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Establecer el modo para el pin EXTI

\* @param[in] EXTILine línea de interrupción externa, debe ser:

\* - EXTI\_EINT0: línea de interrupción externa 0

\* - EXTI\_EINT1: línea de interrupción externa 1

\* - EXTI\_EINT2: línea de interrupción externa 2

\* - EXTI\_EINT3: línea de interrupción externa 3

\* @param[in] mode modo externo, debe ser:

\* - EXTI\_MODE\_LEVEL\_SENSITIVE

\* - EXTI\_MODE\_EDGE\_SENSITIVE

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EXTI\_SetMode(EXTI\_LINE\_ENUM EXTILine, EXTI\_MODE\_ENUM mode)

{

if (mode == EXTI\_MODE\_EDGE\_SENSITIVE)

{

LPC\_SC->EXTMODE |= (1 << EXTILine);

}

else if (mode == EXTI\_MODE\_LEVEL\_SENSITIVE)

{

LPC\_SC->EXTMODE &= ~(1 << EXTILine);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Establecer la polaridad para el pin EXTI

\* @param[in] EXTILine línea de interrupción externa, debe ser:

\* - EXTI\_EINT0: línea de interrupción externa 0

\* - EXTI\_EINT1: línea de interrupción externa 1

\* - EXTI\_EINT2: línea de interrupción externa 2

\* - EXTI\_EINT3: línea de interrupción externa 3

\* @param[in] polarity valor de polaridad externa, debe ser:

\* - EXTI\_POLARITY\_HIGH\_ACTIVE\_OR\_RISING\_EDGE

\* - EXTI\_POLARITY\_LOW\_ACTIVE\_OR\_FALLING\_EDGE

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EXTI\_SetPolarity(EXTI\_LINE\_ENUM EXTILine, EXTI\_POLARITY\_ENUM polarity)

{

if (polarity == EXTI\_POLARITY\_HIGH\_ACTIVE\_OR\_RISING\_EDGE)

{

LPC\_SC->EXTPOLAR |= (1 << EXTILine);

}

else if (polarity == EXTI\_POLARITY\_LOW\_ACTIVE\_OR\_FALLING\_EDGE)

{

LPC\_SC->EXTPOLAR &= ~(1 << EXTILine);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Limpiar la bandera de interrupción externa

\* @param[in] EXTILine línea de interrupción externa, debe ser:

\* - EXTI\_EINT0: línea de interrupción externa 0

\* - EXTI\_EINT1: línea de interrupción externa 1

\* - EXTI\_EINT2: línea de interrupción externa 2

\* - EXTI\_EINT3: línea de interrupción externa 3

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EXTI\_ClearEXTIFlag(EXTI\_LINE\_ENUM EXTILine)

{

LPC\_SC->EXTINT |= (1 << EXTILine);

}

/\*\*

\* @}

\*/

**/\* --------------------------------- Fin del archivo ------------------------------ \*/**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Systick \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Incluye ------------------------------------------------------------------- \*/

#include "lpc17xx\_systick.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Inicializa el System Tick utilizando la fuente de reloj interna del CPU

\* @param[in] time intervalo de tiempo (ms)

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SYSTICK\_InternalInit(uint32\_t time)

{

uint32\_t cclk;

float maxtime;

cclk = SystemCoreClock;

/\* Con la frecuencia de reloj interno para LPC17xx siendo 'SystemCoreClock'

\* Y el límite de 24 bits para el valor LOAD

\* Por lo tanto, el tiempo máximo que se puede establecer es:

\* 1/SystemCoreClock \* (2^24) \* 1000 (ms)

\*/

// Verifica si el valor del tiempo es válido o no

maxtime = (1<<24)/(SystemCoreClock / 1000) ;

if(time > maxtime)

// Bucle de error

while(1);

else

{

// Selecciona el reloj del CPU como la fuente de reloj del System Tick

SysTick->CTRL |= ST\_CTRL\_CLKSOURCE;

/\* Establece el valor RELOAD

\* RELOAD = (SystemCoreClock/1000) \* time - 1

\* con base de tiempo en milisegundos

\*/

SysTick->LOAD = (cclk/1000)\*time - 1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Inicializa el System Tick utilizando una fuente de reloj externa

\* @param[in] freq frecuencia del reloj externo (Hz)

\* @param[in] time intervalo de tiempo (ms)

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SYSTICK\_ExternalInit(uint32\_t freq, uint32\_t time)

{

float maxtime;

/\* Con la frecuencia del reloj externo para LPC17xx siendo 'freq'

\* Y el límite de 24 bits para el valor RELOAD

\* Por lo tanto, el tiempo máximo que se puede establecer es:

\* 1/freq \* (2^24) \* 1000 (ms)

\*/

// Verifica si el valor del tiempo es válido o no

maxtime = (1<<24)/(freq / 1000) ;

if (time > maxtime)

// Bucle de error

while(1);

else

{

// Selecciona el reloj externo como la fuente de reloj del System Tick

SysTick->CTRL &= ~ ST\_CTRL\_CLKSOURCE;

/\* Establece el valor RELOAD

\* RELOAD = (freq/1000) \* time - 1

\* con base de tiempo en milisegundos

\*/

maxtime = (freq/1000)\*time - 1;

SysTick->LOAD = (freq/1000)\*time - 1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Habilita/Deshabilita el contador System Tick

\* @param[in] NewState Estado del contador System Tick, debe ser:

\* - ENABLE

\* - DISABLE

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SYSTICK\_Cmd(FunctionalState NewState)

{

CHECK\_PARAM(PARAM\_FUNCTIONALSTATE(NewState));

if(NewState == ENABLE)

// Habilita el contador System Tick

SysTick->CTRL |= ST\_CTRL\_ENABLE;

else

// Deshabilita el contador System Tick

SysTick->CTRL &= ~ST\_CTRL\_ENABLE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Habilita/Deshabilita la interrupción del System Tick

\* @param[in] NewState Estado de la interrupción del System Tick, debe ser:

\* - ENABLE

\* - DISABLE

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SYSTICK\_IntCmd(FunctionalState NewState)

{

CHECK\_PARAM(PARAM\_FUNCTIONALSTATE(NewState));

if(NewState == ENABLE)

// Habilita la interrupción del System Tick

SysTick->CTRL |= ST\_CTRL\_TICKINT;

else

// Deshabilita la interrupción del System Tick

SysTick->CTRL &= ~ST\_CTRL\_TICKINT;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Obtiene el valor actual del contador System Tick

\* @param[in] Ninguno

\* @return valor actual del contador System Tick

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint32\_t SYSTICK\_GetCurrentValue(void)

{

return (SysTick->VAL);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* @brief Limpia la bandera del contador

\* @param[in] Ninguno

\* @return Ninguno

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SYSTICK\_ClearCounterFlag(void)

{

SysTick->CTRL &= ~ST\_CTRL\_COUNTFLAG;

}

**/\* --------------------------------- Fin del archivo ------------------------------ \*/**