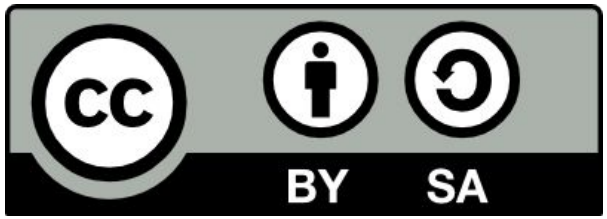
http://www.proyecto-ciaa.com.ar/

Configuración de pines del microcontrolador NXP LPC4337 mediante registros SCU y GPIO

- **Mg. Ing. Eric Pernia.**
- **Ing. Ian Olivieri**

Licencia



“Configuración de pines del microcontrolador NXP LPC4337 mediante registros SCU y GPIO”

Temario

1. Diagrama en bloques de un pin.
2. Reconocer los pines del microcontrolador (MCU).
3. System Control Unit (SCU).
4. Registro SCU de configuración de pin SFSPN_M.
5. Configuración de pines para LED1 y TEC1.
6. SCU - Resumen de configuración para GPIO.
7. Registro de configuración de dirección de GPIO.
8. Configuración de dirección de pines para LED1 y TEC1.
9. Funciones LPCOpen de GPIO para leer y escribir pines.
10. Ejemplo Pulsador ---> LED.

Diagrama en bloques de un pin

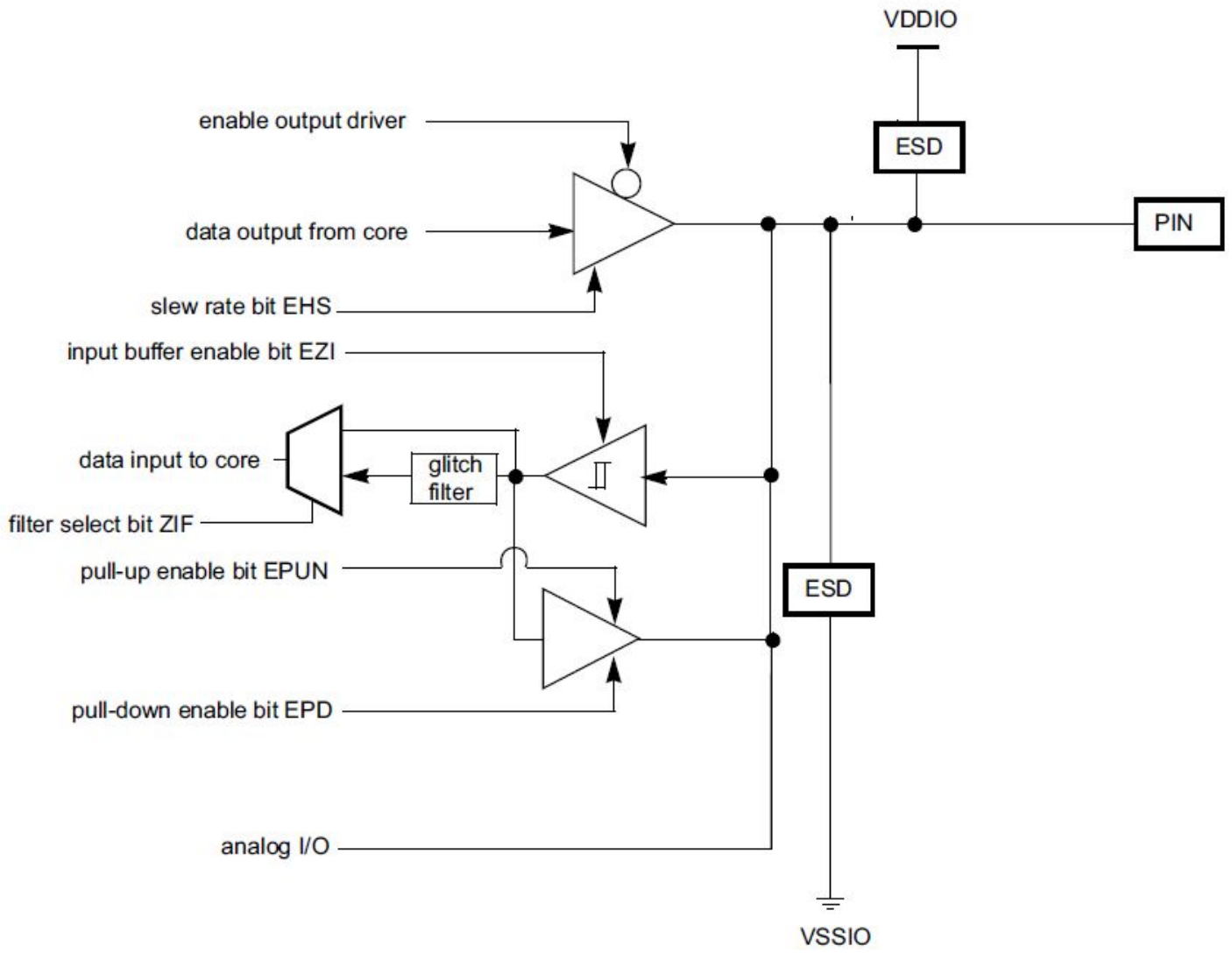
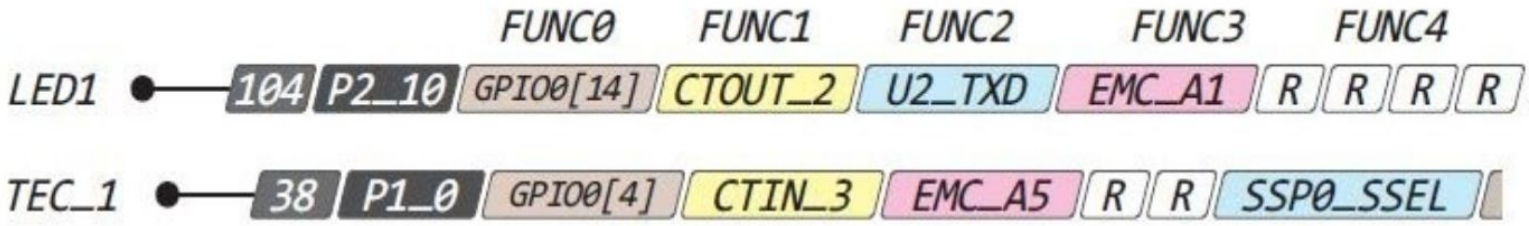


Fig 43. Block diagram of the I/O pad

Reconocer los pines del MCU

Supongamos que queremos configurar en la EDU-CIAA-NXP los pines conectados a **LED1 como salida digital** y **TEC1 como entrada digital**.

Si miramos el archivo de asignación de pines de la EDU-CIAA-NXP veremos que:



- El diodo LED, **LED1**, está conectado al Pin N° **104** del microcontrolador.
- El pulsador, **TEC1**, está conectado al Pin N° **38** del microcontrolador.

System Control Unit (SCU)

Para configurar un pin en el microcontrolador LPC4337 debemos conocer un subsistema encargado de la configuración de los pines el cual se llama **System Control Unit (SCU)**.

Este se encarga de configurar a qué periférico interno se conecta un pin físico del chip (FUNC), además de otras configuraciones eléctricas de los mismos (*pull resistors*, *glich filter*, *input buffer*, etc.).

De esta forma, existe un registro de 32 bits asociado a cada pin donde configuramos al mismo nombrado:

SFSPN_M, con **N** número de *port* y **M** número de *pin*.

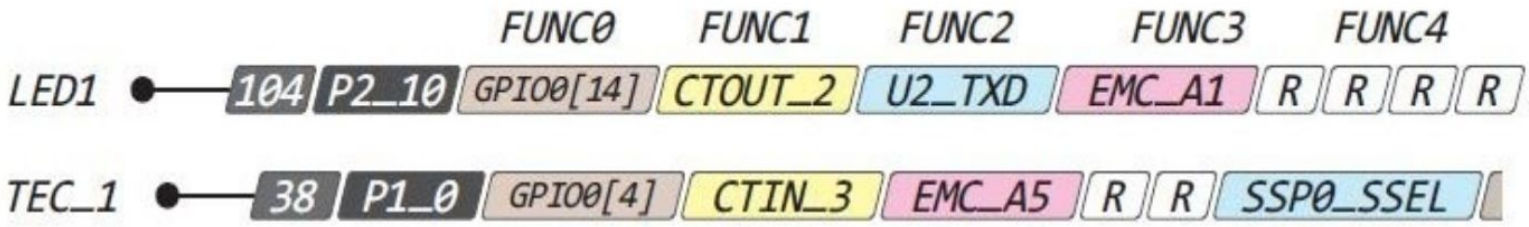
System Control Unit (SCU)

Los parámetros port y pin para los registros SCU pueden ser obtenidos del manual de usuario del LPC4337 (página 242 de 1433). En esta presentación se utiliza como alternativa el documento asignación de pines de la EDU-CIAA-NXP que tiene un resumen de los pines de la plataforma.

Las funciones pueden ser FUNC0, FUNC1,... FUNC7 y su elección depende de como se requiera utilizar el pin.

System Control Unit (SCU)

Si miramos nuevamente el archivo de asignación de pines de la EDU-CIAA-NXP veremos que:



- **LED1** (pin N° **104** del MCU) corresponde a **P2_10** (Port = 2 y Pin = 10). Entonces se configura mediante el registro **SFSP2_10**.
- **TEC1** (pin N° **104** del MCU) corresponde a **P1_0** (Port = 1 y Pin = 0). Entonces se configura mediante el registro **SFSP1_0**.

También se ve que para estos pines físicos, la función que corresponde a **GPIO** es la **FUNC0**, que corresponde a los primeros bits de un cierto registro **SFSPN_M** como veremos a continuación.

Registro SCU de configuración de pin SFSPN_M

Bits **MODE** del registro SFSPN_M para configurar a qué periférico interno se conecta un pin físico del MCU.

Table 192. Pin configuration registers for normal-drive pins (SFS, address 0x4008 6000 (SPSP0_0) to 0x4008 67AC (SFSPF_11)) bit description

Bit	Symbol	Value	Description	Reset value	Access
2:0	MODE		Select pin function.	0	R/W
		0x0	Function 0 (default)		
		0x1	Function 1		
		0x2	Function 2		
		0x3	Function 3		
		0x4	Function 4		
		0x5	Function 5		
		0x6	Function 6		
		0x7	Function 7		

Registro SCU de configuración de pin SFSPN_M

Bits **EPD** y **EPUN** del registro SFSPN_M para configurar resistencias de Pull-down y Pull-Up conectadas a un pin físico del MCU.

3	EPD		Enable pull-down resistor at pad.	0	R/W
		0	Disable pull-down.		
		1	Enable pull-down.Enable both pull-down resistor and pull-up resistor for repeater mode.		
4	EPUN		Disable pull-up resistor at pad. By default, the pull-up resistor is enabled at reset.	0	R/W
		0	Enable pull-up. Enable both pull-down resistor and pull-up resistor for repeater mode.		
		1	Disable pull-up.		

Registro SCU de configuración de pin SFSPN_M

Bit **EHS** (seleccionar *Slew rate*) del registro SFSPN_M para configurar la velocidad de conmutación de un pin físico del MCU.

5	EHS		Select Slew rate.	0	R/W
		0	Slow (low noise with medium speed)		
		1	Fast (medium noise with fast speed)		

Bit **EZI** del registro SFSPN_M para configurar la habilitación de buffer de entrada de un pin físico del MCU.
Para modo INPUT recordar que debe estar activo.

6	EZI		Input buffer enable. The input buffer is disabled by default at reset and must be enabled for receiving.	0	R/W
		0	Disable input buffer		
		1	Enable input buffer		

Registro SCU de configuración de pin SFSPN_M

Bit **ZIF** del registro SFSPN_M para configurar la habilitación de filtro de ruido de entrada de un pin físico del MCU.

7	ZIF		Input glitch filter. Disable the input glitch filter for clocking signals higher than 30 MHz.	0	R/W
		0	Enable input glitch filter		
		1	Disable input glitch filter		
31:8	-		Reserved	-	-

Las funciones de biblioteca que nos provee el fabricante (LPC Open) para realizar la configuración de SCU se encuentran en la biblioteca scu_18xx_43xx.h. En particular, utilizaremos la función:

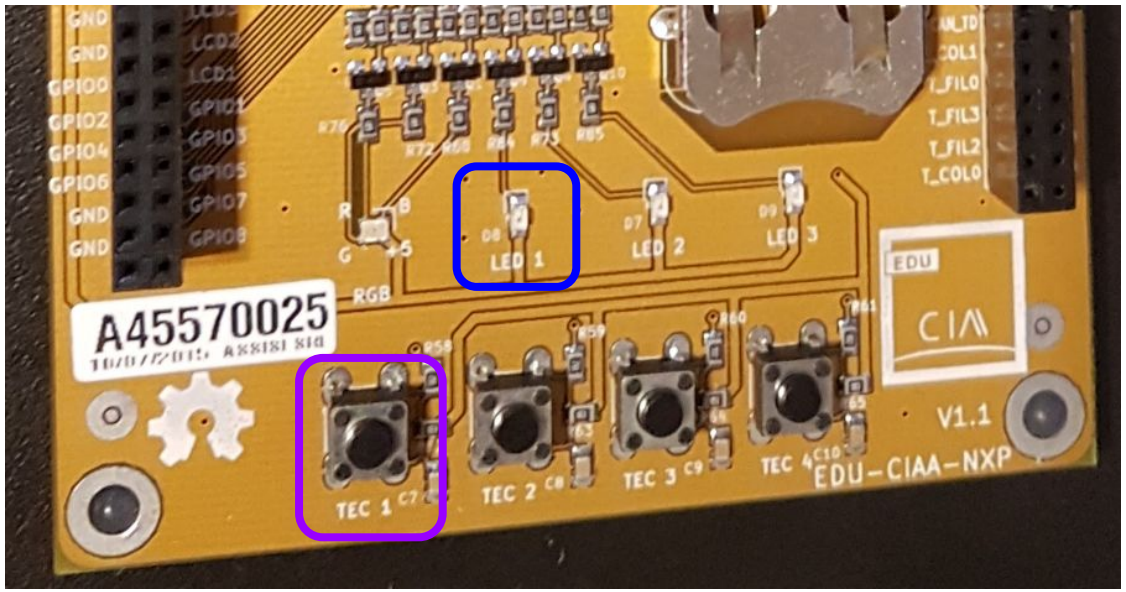
void Chip_SCU_PinMux(uint8_t **port**, uint8_t **pin**, uint16_t **mode**, uint8_t **func**);

Configuración de pines para LED1 y TEC1

A continuación veremos cómo se configuran ambos pines físicos (LED1 y TEC1) para que se conecten a sus respectivas funcionalidades de GPIO.

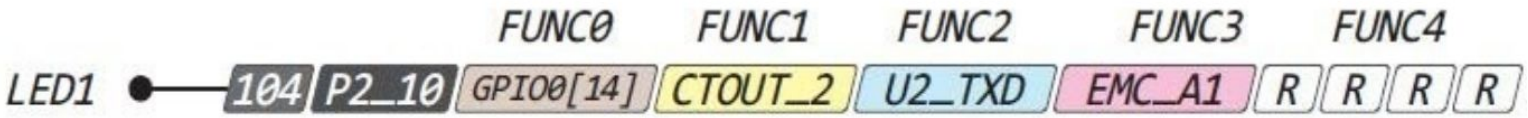
Diodo LED: **LED1** <---> P2_10, GPIO0[14], FUNC0

Pulsador: **TEC1** <---> P1_0, GPIO0[4], FUNC0



Configuración de pines para LED1 y TEC1

Diodo LED: LED1 <---> P2_10, GPIO0[14], FUNC0



Configurar que el pin P2_10 se conecte internamente a GPIO0[14] mediante registro **SFSP2_10**:

Dirección del registro **SFSP2_10** = SCU_BASE + SFSP2_10 (offset)
= 0x40086000 + 0x128 = **0x40086128**

Valor de configuración a guardar en esa dirección =
= SCU_MODE_INACT | SCU_MODE_FUNC0
= (0x2 << 3) | 0x0 = **0x00000010**

Usando LPCopen:

```
Chip_SCU_PinMux( 2,  10,  SCU_MODE_INACT,  SCU_MODE_FUNC0 );
```

Configuración de pines para LED1 y TEC1

Pulsador: TEC1 <---> P1_0, GPIO0[4], FUNC0

Configurar que el pin P1_0 se conecte internamente a GPIO0[4] mediante registro **SFSP1_0**:

Dirección del registro **SFSP1_0** = SCU_BASE + SFSP1_0(offset)
= 0x40086000 + 0x080 = **0x40086080**

Valor de configuración a guardar en esa dirección =
= SCU_MODE_INACT | SCU_MODE_INBUFF_EN | SCU_MODE_FUNC0
= (0x2 << 3) | (0x1 << 6) | 0x0 = **0x00000090**

Usando LPCopen:

```
Chip_SCU_PinMux( 1,  0,  SCU_MODE_INACT | SCU_MODE_INBUFF_EN,
                         SCU_MODE_FUNC0 );
```

SCU - Resumen de configuración para GPIO

En LPC4337 Para la funcionalidad de GPIO vamos a ver que se encuentran (si el pin soporta) en los modos FUNC0 o FUNC4. Además:

- Para el modo OUTPUT: SCU_MODE_INACT
- Para el modo INPUT (sin pull-up ni pull-down):
  SCU_MODE_INACT | SCU_MODE_INBUFF_EN
- Para el modo INPUT_PULLUP (sólo con pull-up):
  SCU_MODE_PULLUP | SCU_MODE_INBUFF_EN
- Para el modo INPUT_PULLDOWN (sólo con pull-down):
  SCU_MODE_PULLDOWN | SCU_MODE_INBUFF_EN
- Para el modo INPUT_REPEATER (con pull-up y pull-down):
  SCU_MODE_REPEATER | SCU_MODE_INBUFF_EN

Opcionalmente se puede agregar a todos: SCU_MODE_ZIF_DIS

Estos datos se encuentran en la página 413 del manual de usuario del LPC4337.
Para más información referirse al capítulo 16 y 17 del manual de usuario del LPC4337

Registro de configuración de dirección de GPIO

Los GPIO (General Purpose Input/Output) son las entradas y salidas digitales del microcontrolador. Los pines con funcionalidad de GPIO se agrupan en puertos y se numeran de la forma **GPIO**N**[**M**]**. Con **N**, número de puerto y **M**, número de pin en el puerto.

NOTA: Aquí, **N** y **M** no son los mismos que **N** y **M** para los registros de SCU.

Existe por cada puerto (puertos GPIO0 a GPIO7) un registro de dirección donde, mediante cada bit que lo compone, se configuran la dirección cada pin, es decir, si será entrada (0) o salida (1).

Table 262. GPIO port direction register (DIR, addresses 0x400F 6000 (DIR0) to 0x400F 601C (DIR7)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIR	Selects pin direction for pin GPIOn[m] (bit 0 = GPIOn[0], bit 1 = GPIOn[1], ..., bit 31 = GPIOn[31]). 0 = input. 1 = output.	0	R/W

Registro de configuración de dirección de GPIO

Las funciones de biblioteca que nos provee el fabricante (LPC Open) para realizar la configuración de GPIO se encuentran en la biblioteca gpio_18xx_43xx.h.

En particular, utilizaremos para configurar un pin como entrada o salida se utiliza la función:

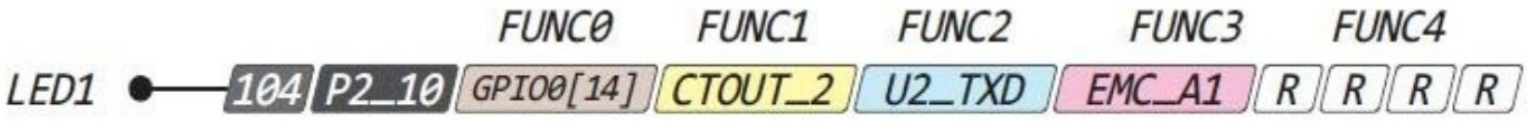
```
void Chip_GPIO_SetDir( LPC_GPIO_T *pGPIO,
                       uint8_t portNum,
                       uint32_t bitValue,
                       uint8_t out );
```

El primer parámetro de esta función (LPC_GPIO_T *pGPIO) equivale a **LPC_GPIO_PORT** para el LPC4337.

Aquí, **portNum** y **bitValue** no son los mismos que **port** y **pin** para el SCU. Son otros parámetros que tienen la forma **GPIOportNum[bitValue]**.

Configuración de dirección de pines para LED1 y TEC1

Diodo LED: LED1 <---> P2_10, GPIO0[14], FUNC0



El LED1 corresponde a GPIO**0**[**14**], se configurará como **salida** (**1**) mediante registro **GPIO_PORT0_DIR**:

Dirección del registro **GPIO_PORT0_DIR** =
= **GPIO_BASE** + **GPIO_PORT0_B_OFFSET** + **GPIO_PORT0_DIR_OFFSET**
= 0x400F4000 + 0x0 + 0x2000

Valor de configuración a guardar en esa dirección = **(1 << 14)** = 0x00004000

Usando LPCopen:

```
Chip_GPIO_SetDir( LPC_GPIO_PORT,  0,  (1<<14),  1 );
```

Configuración de dirección de pines para LED1 y TEC1

Pulsador: TEC1 <---> P1_0, GPIO0[4], FUNC0

TEC1 corresponde a GPIO**0**[**4**], se configurará como **entrada** (**0**) mediante registro **GPIO_PORT0_DIR**:

Dirección del registro **GPIO_PORT0_DIR** =
= **GPIO_BASE** + **GPIO_PORT0_B_OFFSET** + **GPIO_PORT0_DIR_OFFSET**
= 0x400F4000 + 0x0 + 0x2000

Valor de configuración a guardar en esa dirección: **~(1 << 4)**

Usando LPCopen:

```
Chip_GPIO_SetDir( LPC_GPIO_PORT, 0, (1<<4), 0 );
```

Funciones LPCOpen de GPIO para leer y escribir pines

Para leer y escribir valores de un pin se utilizan las funciones:

```
void Chip_GPIO_SetPinState(
                            LPC_GPIO_T *pGPIO,
                            uint8_t port,
                            uint8_t pin,
                            bool setting
                          );

bool Chip_GPIO_GetPinState(
                            LPC_GPIO_T *pGPIO,
                            uint8_t port,
                            uint8_t pin
                           );
```

Para más información referirse al capítulo 19 del manual de usuario del LPC4337

Ejemplo Pulsador ---> LED

Hasta aquí tenemos configurados ambos pines LED1 y TEC1 como salida y entrada, respectivamente, ya sea escribiendo registros, o bien, utilizando la biblioteca LPCopen.

Solo nos resta hacer un pequeño programa que lea el valor de TEC1 y lo refleje en LED1, que usando LPCopen es...

Ejemplo Pulsador ---> LED

```c
#include <chip.h>

#define OUTPUT   1
#define INPUT    0

#define ON       1
#define OFF      0

int main( void )
{
   // Diodo LED: LED1 <---> P2_10, GPIO0[14] ---------------------------------
   Chip_SCU_PinMux( 2, 10, SCU_MODE_INACT, SCU_MODE_FUNC0 );
   Chip_GPIO_SetDir( LPC_GPIO_PORT, 0, (1<<14), OUTPUT );
   Chip_GPIO_SetPinState( LPC_GPIO_PORT, 0, 14, OFF );

   // Pulsador: TEC1 <---> P1_0, GPIO0[4] ------------------------------------
   Chip_SCU_PinMux( 1, 0, SCU_MODE_INACT | SCU_MODE_INBUFF_EN, SCU_MODE_FUNC0 );
   Chip_GPIO_SetDir( LPC_GPIO_PORT, 0, (1<<4), INPUT );

   while( 1 ) {
      // !(TEC1) --> LED1
      // Negado (!) porque el pulsador esta en 1 por defecto y al presionar da 0
      uint8_t value = Chip_GPIO_GetPinState( LPC_GPIO_PORT, 0, 4 );
      Chip_GPIO_SetPinState( LPC_GPIO_PORT, 0, 14, !value );
   }
   return 0;
}
```

Bibliografía

- Manual de usuario del microcontrolador NXP LPC4337.
- Archivos de LPCOpen.
- Web del proyecto CIAA.

¡Muchas gracias!

Consultas:

ericpernia@gmail.com