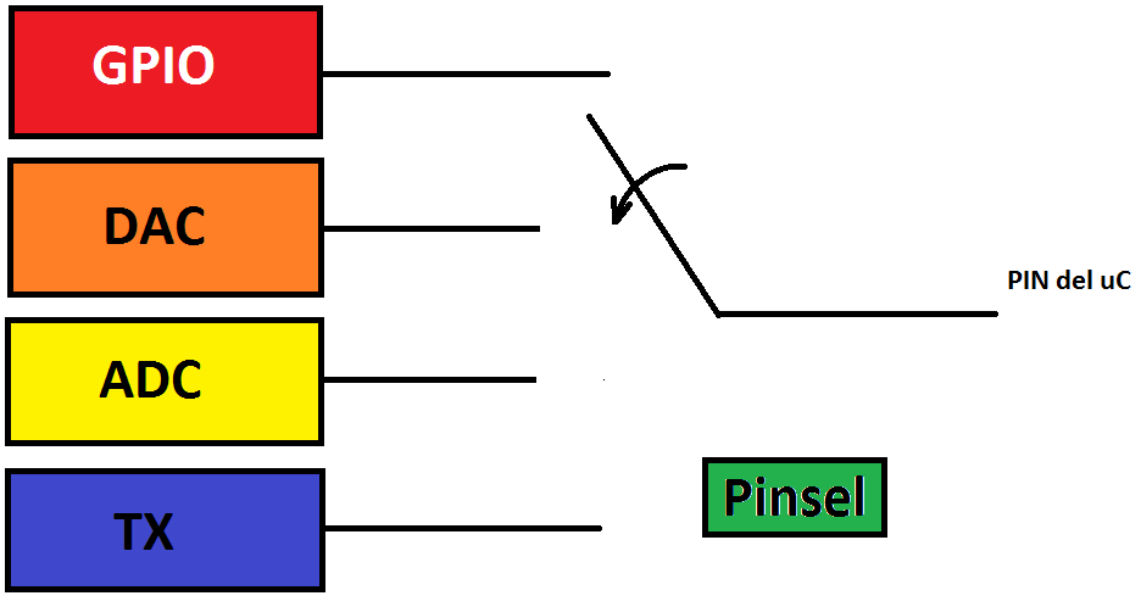
Temas

- Pin Connect Block - GPIO
- Registros del Pin connect Block
- Características eléctricas de los pines de E/S
- Registros del GPIO
- Ubicación en memoria de los registros GPIO
- Archivo LPC17xx.h - Explicación
- Ejemplo prender y apagar un led
- Actividad práctica

•Pin Connect Block - GPIO

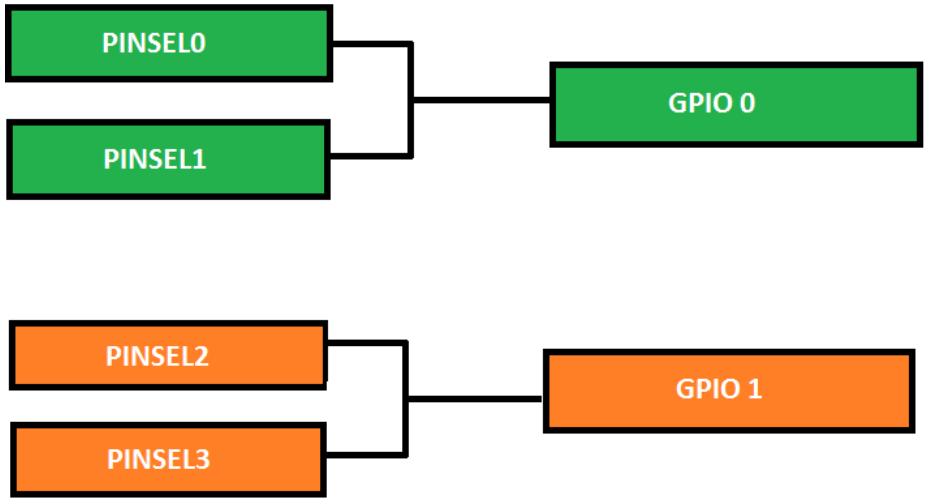
El primer bloque es el Pin Connect Block, este permite conectar la mayoría de los pines del procesador para tener más de una función. Permite conectar los pines con salida al exterior del procesador con los diferentes periféricos.



Por cada pin tenemos 4 funciones posibles por lo tanto para seleccionar alguna de las funciones disponibles necesitamos 2 bits.
Luego necesitamos 2 bits por pin, por lo tanto para seleccionar las funciones posibles del puerto cero (P0), de 32 bits necesitamos 2 registros de 32 bits.

•Registros del Pin connect Block

Como ejemplo, el registro PINSEL0(**0x4002 C000**) controla las funciones de los 16 pines menos significativos del puerto cero (0-15). Por su parte el registro PINSEL1 (**0x4002 C004**) controla la función de los 16 pines más significativos del puerto cero. Esto puede verse más claro en el manual de usuario UM10360



Siguiendo esta lógica, el registro PINSEL2 y PINSEL3 se encargarán de seleccionar los periféricos a los que se conectarán los pines del Puerto 1

Nuestro procesador tiene sus pines de salida organizados en Puertos de 32 pines cada uno. Hay que tener en cuenta que no siempre podemos utilizarlos a todos, esto es similar a PIC en el sentido de que algunos puertos no tenían disponibles todos los pines del puerto, según el encapsulado.

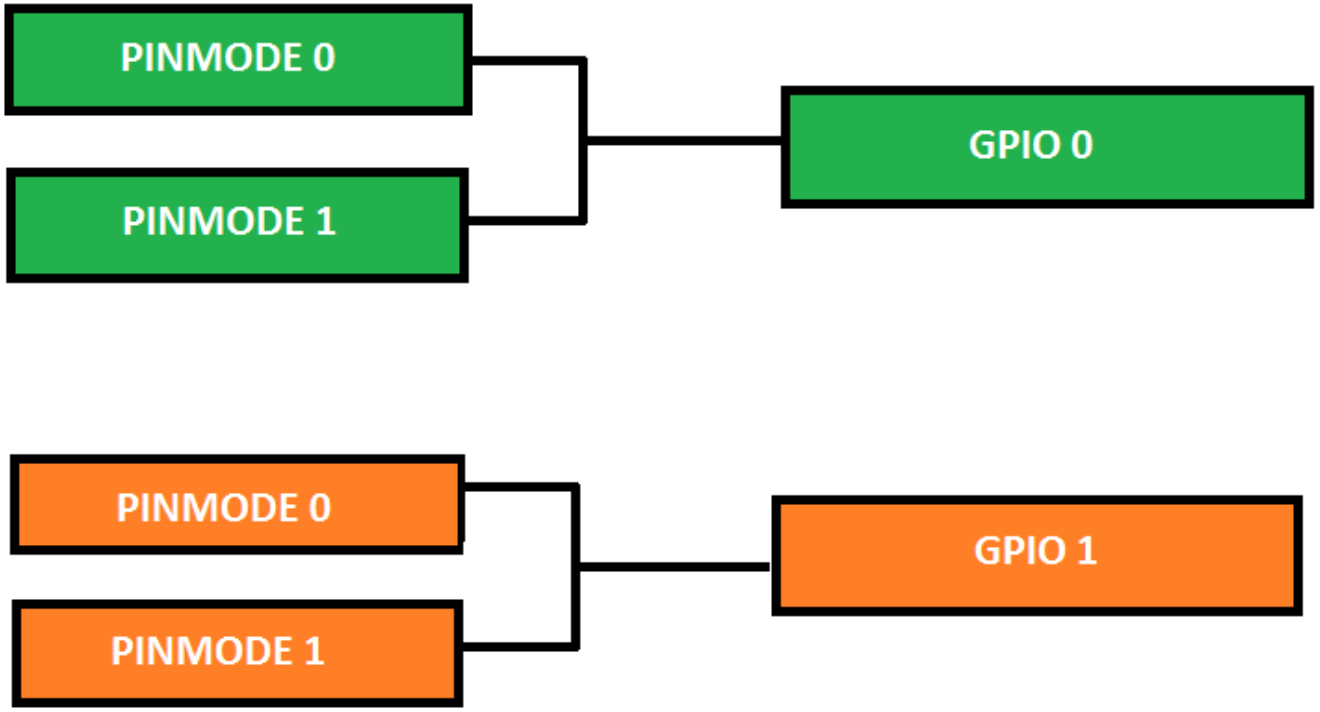
•Registros del Pin connect Block

A continuación se muestran las funciones de los pines 0-15 del puerto GPIO 0 para las diferentes combinaciones de valores para el registro PINSEL 0 y el valor del registro después de un reset

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

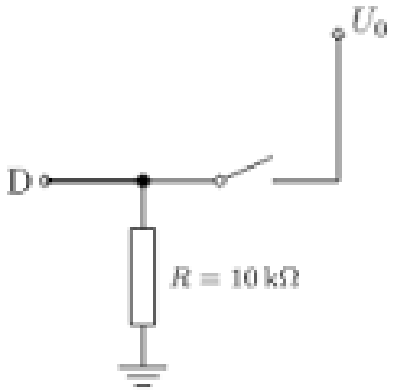
•Registros del Pin connect Block

Dentro del Pin Connect Block tenemos los registros PIN MODE los cuales nos permiten configurar las resistencias de pull up/pull down. Tenemos cuatro posibles conexiones, por lo tanto estamos en el mismo caso que con los registros PINSEL, necesitamos 2 registros de PINMODE por cada puerto GPIO

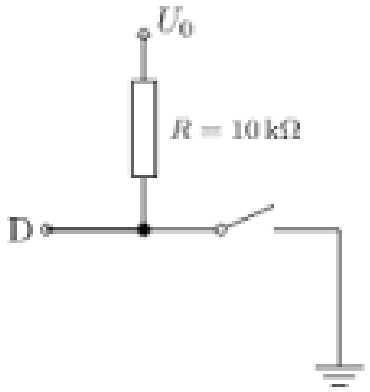


•Registros del Pin connect Block

Las 4 opciones posibles son Pull Up, Repeater mode, ninguno, Pull Down



Pull Down

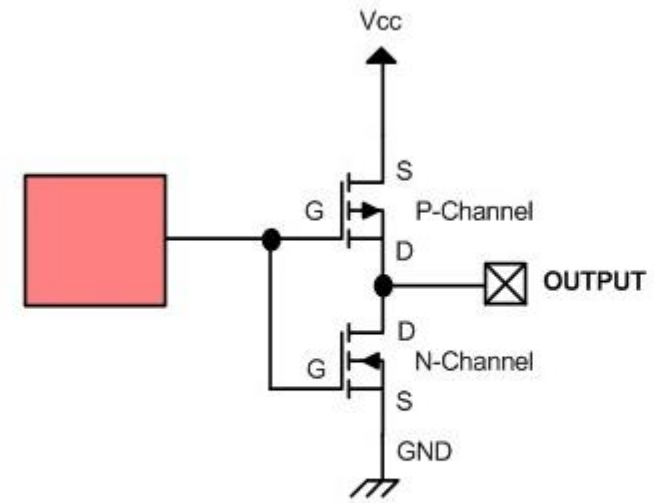


Pull Up

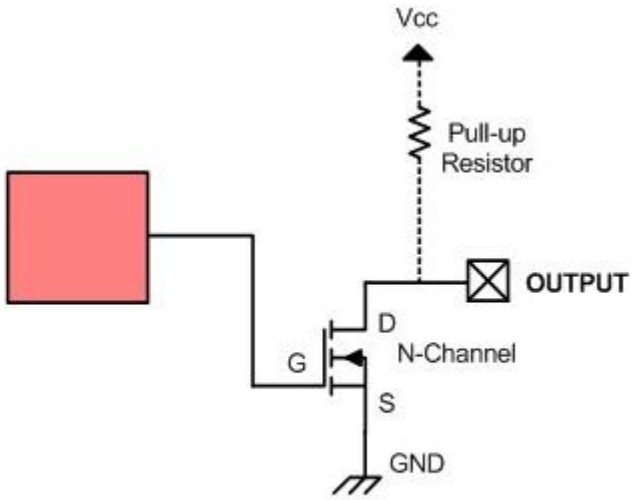
En modo Repeater conecta la resistencia de pull Up cuando la salida lógica está en estado alto y conecta la de pull down cuando está en estado lógico bajo

•Registros del Pin connect Block

Contamos con los registros PINMODE_OD que controlan si el pin funcionará en modo OPEN DRAIN. Necesitamos solo un bit para configurar esta opción.



A. Normal CMOS I/O Port



B. Open Drain CMOS I/O Port

Luego de un reset tenemos PINSEL =0 (configurado como GPIO), PINMODE=0 (Pull Up), PINMODE_OD=0 (salida en modo normal, no open drain) por esto es que no lo tuvimos en cuenta en el primer código, sin embargo en un examen se agrega la configuración incluso aunque esté por defecto después de un reset, o bien se agregan los comentarios en el código que den cuenta de que se está teniendo en cuenta aún lo que no está explícito

- Cacterísticas eléctricas de los pines de E/S

Los aspectos principales que debemos tener en cuenta a la hora de comenzar a conectar nuestra placa con otro harware son las características eléctricas de la misma, las características eléctricas de los pines de la misma.

- Salida
  - Tensión en estado alto/bajo
  - Corriente que puede entregar o recibir en estado alto/bajo
- Entrada
  - Tensiones que puede tolerar 3.3, 5.5

Para encontrar esta información debemos remitirnos a la Hoja de datos del procesador.

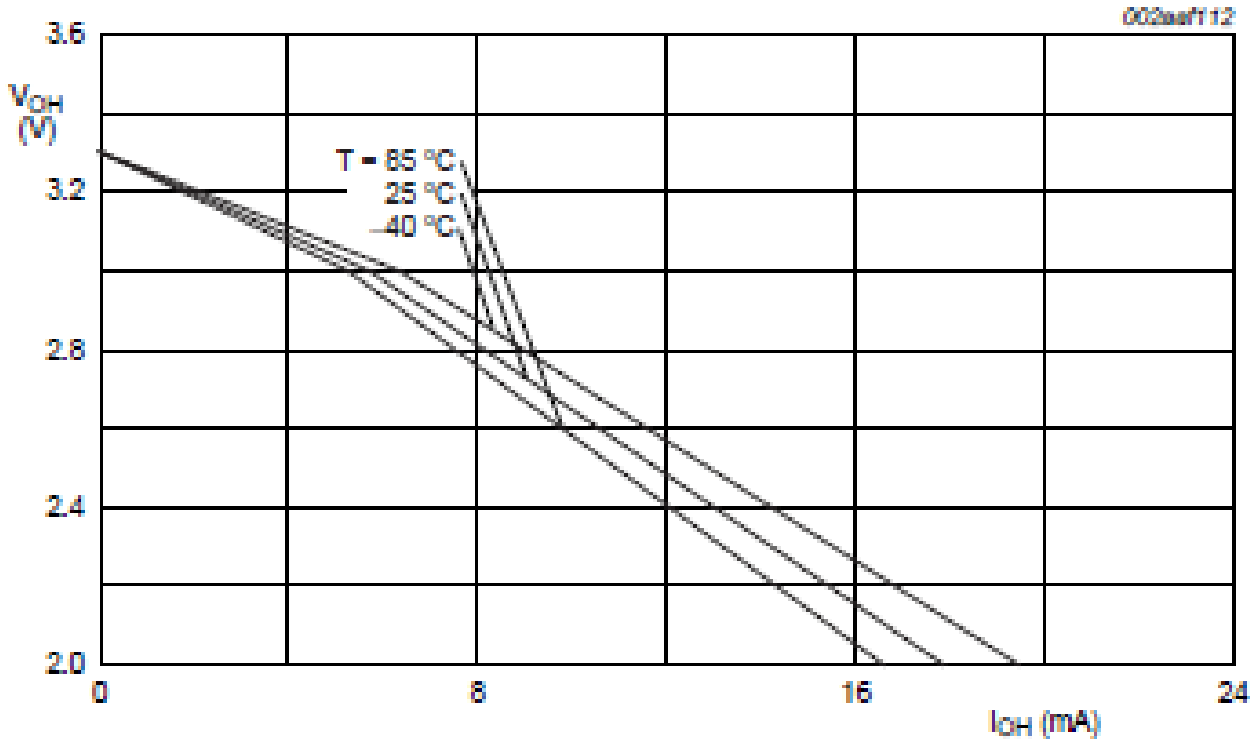
•Cacterísticas eléctricas de los pines de E/S

Características eléctricas

Symbol	Parameter	Conditions		Min	Typ[1]	Max	Unit
V_{OH}	HIGH-level output voltage	I_{OH} = –4 mA		$V_{DD(3V3)}$ – 0.4	-	-	V
V_{OL}	LOW-level output voltage	I_{OL} = 4 mA		-	-	0.4	V
I_{OH}	HIGH-level output current	$V_{OH} = V_{DD(3V3)} – 0.4$ V		–4	-	-	mA
I_{OL}	LOW-level output current	V_{OL} = 0.4 V		4	-	-	mA
V_{IH}	HIGH-level input voltage			$0.7V_{DD(3V3)}$	-	-	V
V_{IL}	LOW-level input voltage			-	-	$0.3V_{DD(3V3)}$	V

•Cacterísticas eléctricas de los pines de E/S

Características eléctricas

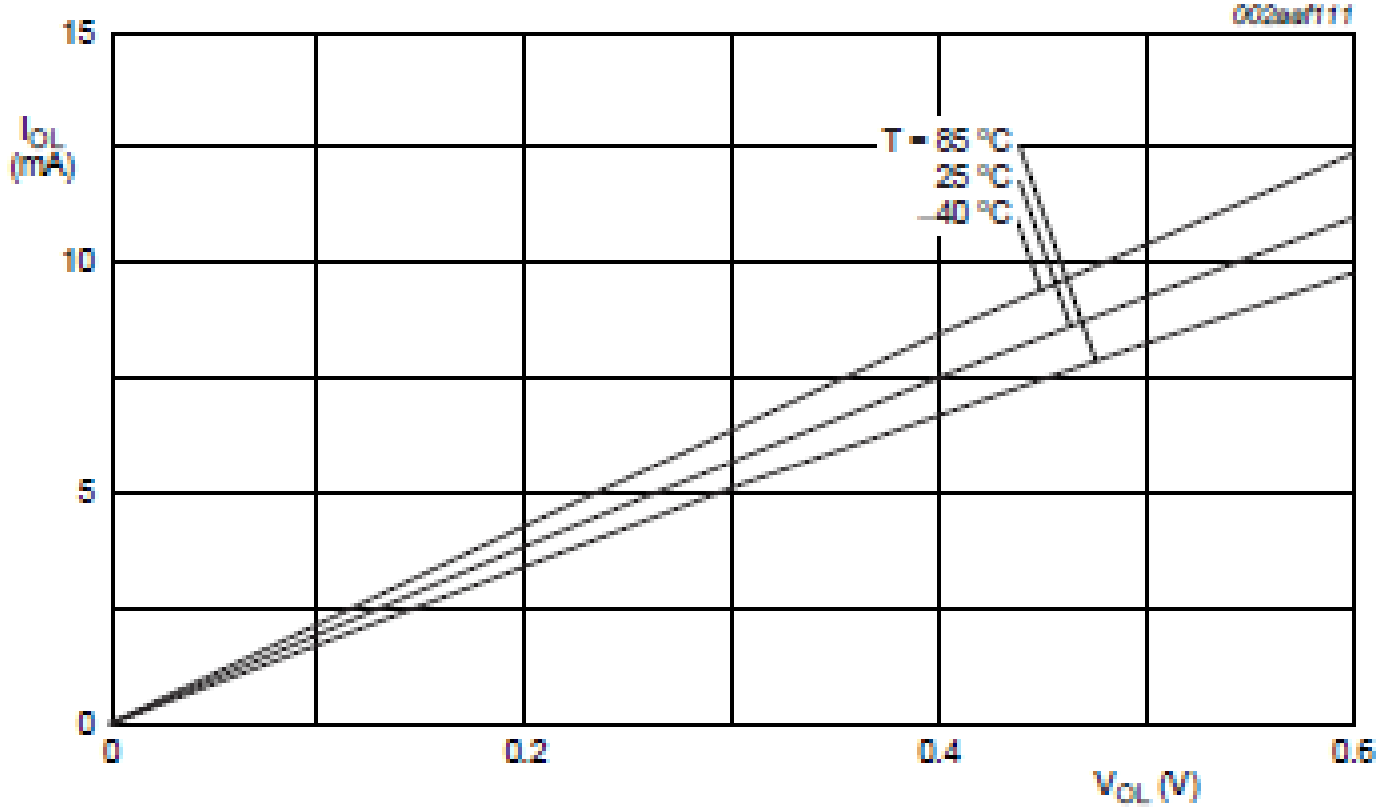


Conditions: $V_{DD(REG)(3V3)} = V_{DD(3V3)} = 3.3$ V; standard port pins.

Fig 12. Typical HIGH-level output voltage V_{OH} versus HIGH-level output source current I_{OH}

•Cacterísticas eléctricas de los pines de E/S

Características eléctricas

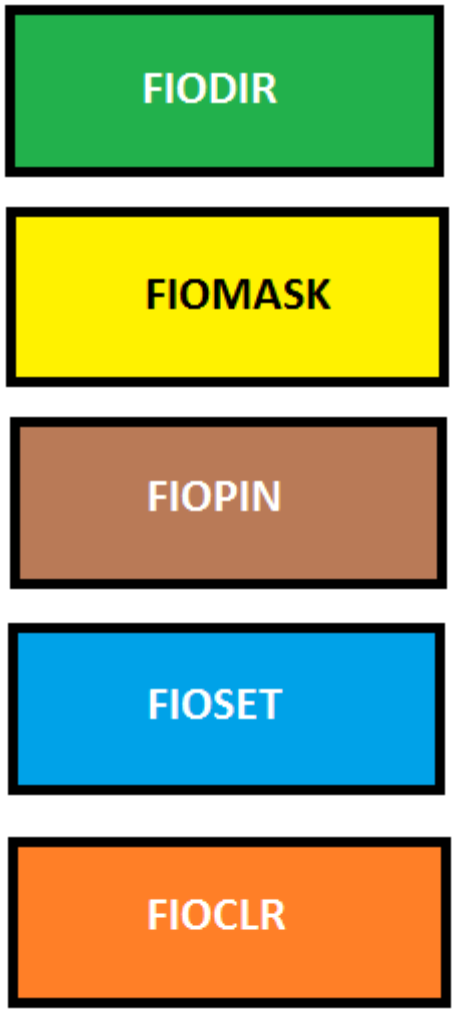


Conditions: $V_{DD(REG)(3V3)} = V_{DD(3V3)} = 3.3$ V; standard port pins.

Fig 13. Typical LOW-level output current I_{OL} versus LOW-level output voltage V_{OL}

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO



FIO es por Fast Input Output (Entrada Salida rápida)

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO

FIODIR

Contamos con FIO0DIR-FIO4DIR, Se encarga de controlar la dirección de los pines del puerto, el bit menos significativo del registro FIO0DIR configura la dirección del pin menos significativo del puerto

El Bit 1 del FIO0DIR, configura el P0.1 (pin 1 del puerto cero GPIO 0)

Un ‘0’ configura el pin como entrada

Un ‘1’ configura el pin como salida

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO

FIOMASK

Los registros FIOxMASK Se encargan de generar una máscara en los pines del puerto de modo que si el bit del registro FIOMASK está en ‘0’ el pin del puerto puede controlarse desde los registros FIOPIN, FIOSET, FIOCLR, si está en ‘1’ no es afectado por la escritura de los registros ya mencionados.

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO

FIOPIN

Los registros FIO0PIN-FIO4PIN Contienen el estado del pin. Si escribimos (Salida) un ‘0’ el pin se pone en estado bajo, si leemos (entrada) un ‘0’ el pin está en estado bajo

Si escribimos (salida) un ‘1’ el pin se pone en estado alto
Si leemos (entrada) un ‘1’ el pin está en estado alto

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO

FIOSET

Los registros FIO0SET-FIO4SET se encargan de poner en estado de ‘1’ los pines cuyos correspondientes bits del registro FIOxSET se pusieron en ‘1’.

Si colocamos un ‘1’ en el bit menos significativo del registro FIO0SET, se pone en ‘1’ el pin P0.0 (pin cero del GPIO 0)

Escribir ceros en el registro FIOxSET no afecta el estado de los pines de puerto

•Registros del GPIO

Nos introducimos ahora al manejo de los registros asociados al módulo GPIO

Los registros FIO0CLR-FIO4CLR se encargan de poner en estado de ‘0’ los pines cuyos correspondientes bits del registro FIOxCLR se pusieron en ‘1’.

Si colocamos un ‘1’ en el bit menos significativo del registro FIO0CLR, se pone en ‘0’ el pin P0.0 (pin cero del GPIO 0)

Escribir ceros en el registro FIOxCLR no afecta el estado de los pines de puerto

FIOCLR

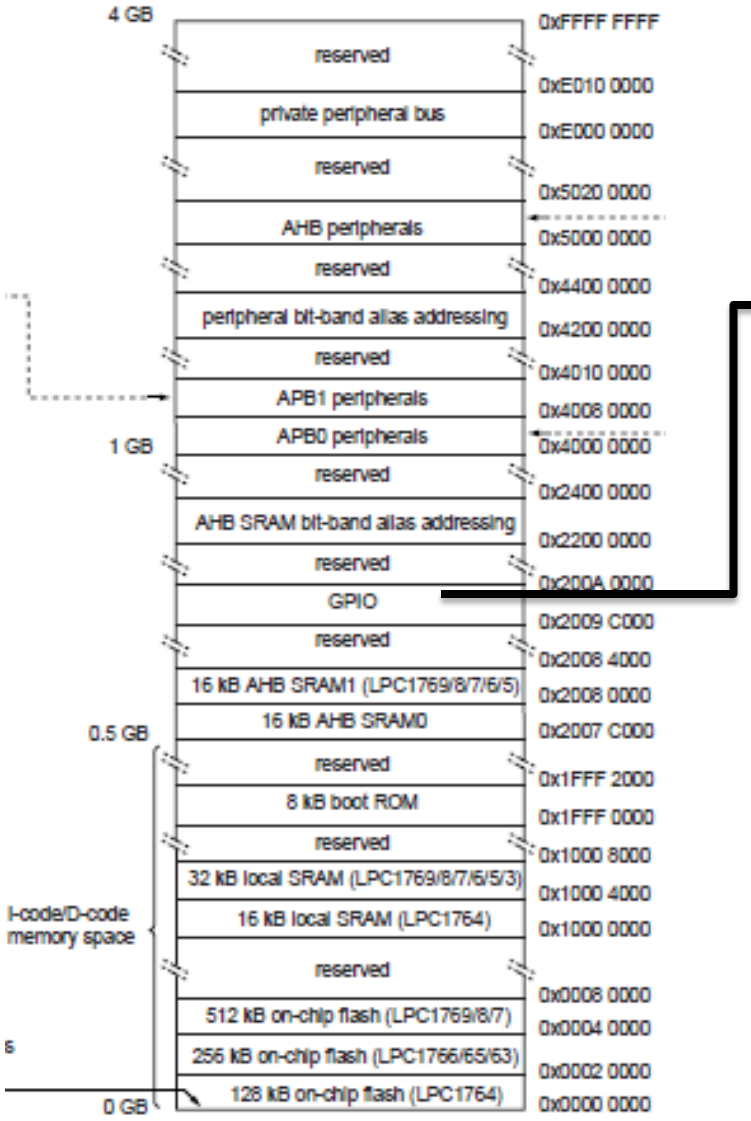
•Registros del GPIO

FIOxSET/FIO0CLR permiten setear y resetear pines de forma simple y rápida, por otro lado FIOxPIN permite cargar todo un valor deseado de 32 bits en el puerto.

- Registros del GPIO

Falta mencionar los registros asociados a las interrupciones de GPIO. Estos permiten generar interrupciones a partir de flancos en los pines conectados al módulo GPIO. No es el objetivo mencionarlos aquí, más que para saber que están disponibles.

•Ubicación en memoria de los registros GPIO



PORTn Register Name & Address
FIO0DIR - 0x2009 C000
FIO1DIR - 0x2009 C020
FIO2DIR - 0x2009 C040
FIO3DIR - 0x2009 C060
FIO4DIR - 0x2009 C080
FIO0MASK - 0x2009 C010
FIO1MASK - 0x2009 C030
FIO2MASK - 0x2009 C050
FIO3MASK - 0x2009 C070
FIO4MASK - 0x2009 C090
FIO0PIN - 0x2009 C014
FIO1PIN - 0x2009 C034
FIO2PIN - 0x2009 C054
FIO3PIN - 0x2009 C074
FIO4PIN - 0x2009 C094
FIO0SET - 0x2009 C018
FIO1SET - 0x2009 C038
FIO2SET - 0x2009 C058
FIO3SET - 0x2009 C078
FIO4SET - 0x2009 C098
FIO0CLR - 0x2009 C01C
FIO1CLR - 0x2009 C03C
FIO2CLR - 0x2009 C05C
FIO3CLR - 0x2009 C07C
FIO4CLR - 0x2009 C09C

•Ubicación en memoria de los registros GPIO

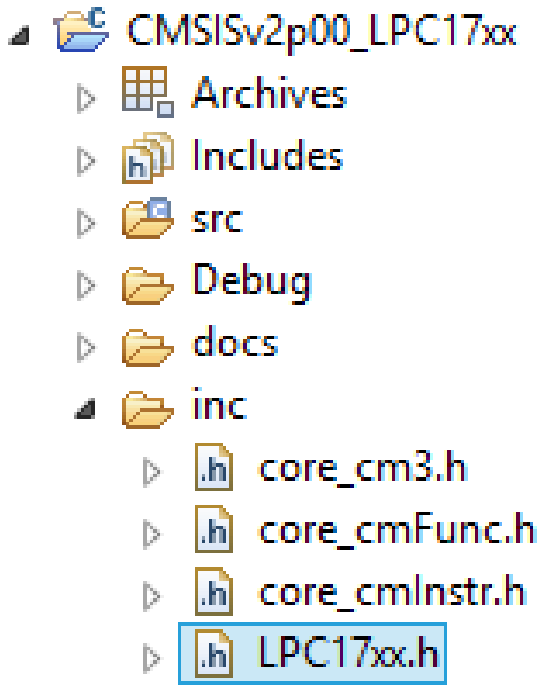
PORTn Register Name & Address
FIO0DIR - 0x2009 C000
FIO1DIR - 0x2009 C020
FIO2DIR - 0x2009 C040
FIO3DIR - 0x2009 C060
FIO4DIR - 0x2009 C080
FIO0MASK - 0x2009 C010
FIO1MASK - 0x2009 C030
FIO2MASK - 0x2009 C050
FIO3MASK - 0x2009 C070
FIO4MASK - 0x2009 C090
FIO0PIN - 0x2009 C014
FIO1PIN - 0x2009 C034
FIO2PIN - 0x2009 C054
FIO3PIN - 0x2009 C074
FIO4PIN - 0x2009 C094
FIO0SET - 0x2009 C018
FIO1SET - 0x2009 C038
FIO2SET - 0x2009 C058
FIO3SET - 0x2009 C078
FIO4SET - 0x2009 C098
FIO0CLR - 0x2009 C01C
FIO1CLR - 0x2009 C03C
FIO2CLR - 0x2009 C05C
FIO3CLR - 0x2009 C07C
FIO4CLR - 0x2009 C09C

GPIO0		
FIO0DIR		0x2009 C000
RESERVED	desde	0x2009 C004
	hasta	0x2009 C00F
FIO0MASK		0x2009 C010
FIO0PIN		0x2009 C014
FIO0SET		0x2009 C018
FIO0CLR		0x2009 C01C

GPIO1		
FIO1DIR		0x2009 C020
RESERVED	desde	0x2009 C024
	hasta	0x2009 C02F
FIO1MASK		0x2009 C030
FIO1PIN		0x2009 C034
FIO1SET		0x2009 C038
FIO1CLR		0x2009 C03C

•Archivo LPC17xx.h - Explicación

Entre todos los archivos que nos proporciona el fabricante tenemos el archivo LPC17xx.h el cual no da definiciones de estructuras.



•Archivo LPC17xx.h - Explicación

Consideraremos la instrucción que utilizamos en nuestro código para poner en alto el pin 22 del puerto cero (P0.22)

```
LPC_GPIO0->FIOSET = (1 << LED2);     //se enciende el led
```

Recordemos LED2=22

LPC_GPIO0 es un puntero que apunta a la base de una estructura de datos de tipo

“LPC_GPIO” (0x2009C000), que se puede ver en la siguiente filmina

•Archivo LPC17xx.h - Explicación

```
185 /*------------- General Purpose Input/Output (GPIO)
186 typedef struct
187 {
188   union {
189     __IO uint32_t FIODIR;
190     struct {
191       __IO uint16_t FIODIRL;
192       __IO uint16_t FIODIRH;
193     };
194     struct {
195       __IO uint8_t  FIODIR0;
196       __IO uint8_t  FIODIR1;
197       __IO uint8_t  FIODIR2;
198       __IO uint8_t  FIODIR3;
199     };
200   };
201   uint32_t RESERVED0[3];
202   union {
203     __IO uint32_t FIOMASK;
204     struct {
205       __IO uint16_t FIOMASKL;
206       __IO uint16_t FIOMASKH;
207     };
208     struct {
209       __IO uint8_t  FIOMASK0;
210       __IO uint8_t  FIOMASK1;
211       __IO uint8_t  FIOMASK2;
212       __IO uint8_t  FIOMASK3;
213     };
214   };
```

```
215   union {
216     __IO uint32_t FIOPIN;
217     struct {
218       __IO uint16_t FIOPINL;
219       __IO uint16_t FIOPINH;
220     };
221     struct {
222       __IO uint8_t  FIOPIN0;
223       __IO uint8_t  FIOPIN1;
224       __IO uint8_t  FIOPIN2;
225       __IO uint8_t  FIOPIN3;
226     };
227   };
228   union {
229     __IO uint32_t FIOSET;
230     struct {
231       __IO uint16_t FIOSETL;
232       __IO uint16_t FIOSETH;
233     };
234     struct {
235       __IO uint8_t  FIOSET0;
236       __IO uint8_t  FIOSET1;
237       __IO uint8_t  FIOSET2;
238       __IO uint8_t  FIOSET3;
239     };
240   };
```

•Archivo LPC17xx.h - Explicación

```
241   union {
242     __O  uint32_t FIOCLR;
243     struct {
244       __O  uint16_t FIOCLRL;
245       __O  uint16_t FIOCLRH;
246     };
247     struct {
248       __O  uint8_t  FIOCLR0;
249       __O  uint8_t  FIOCLR1;
250       __O  uint8_t  FIOCLR2;
251       __O  uint8_t  FIOCLR3;
252     };
253   };
254 } LPC_GPIO_TypeDef;
```

La estructura podría armarse simplemente de la siguiente forma

typedef struct
{
__IO uint32_t FIODIR;
uint32_t RESERVED0[3];
__IO uint32_t FIOMASK;
__IO uint32_t FIOPIN;
__IO uint32_t FIOSET;
__O uint32_t FIOCLR;
} LPC_GPIO_TypeDef;

Sin embargo la estructura unión nos da la ventaja de acceder a partes de los registros como 2 registros de 16bits o 4 bytes.

•Archivo LPC17xx.h - Explicación

```
LPC_GPIO0->FIOSET = (1 << LED2);     //se enciende el led
```

Mediante el operador “->” accede al elemento FIOSET de la estructura LPC_GPIO y le carga el valor (1<<LED2)

De igual manera con el mismo operador podemos acceder a diferentes elementos de la misma estructura. Como pudimos ver, la estructura contiene todos los registros del Puerto 0.

Para operar sobre el puerto 1 utilizamos “LPC_GPIO1” que es un puntero a una estructura del mismo tipo (LPC_GPIO), pero este puntero apunta a otra dirección, la dirección a la que apunta LPC_GPIO1 es la misma que LPC_GPIO0 pero desplazada 0x00020 lugares de memoria, justamente para saltar a la dirección de memoria correspondiente a la estructura del puerto 1

•Actividad práctica

Escribir un código que encienda y apague un led en función del estado lógico de un pin de la placa LPCXpresso.

•Si un pin está en estado alto el led parpadeará
•Si el led está en estado bajo el led no parpaderá

Remítase a la documentación de la placa

LPCXpressoLPC1769revB

•Actividad práctica

Nota:
Se deberá tener en cuenta:
•El pin de entrada no puede estar al aire (sin referencia de tensión)
•El pin de entrada no deberá estar configurado como drenador abierto (open drain)

Todo lo que se deje configurado por defecto, dado que es el estado después de un reset, se configura nuevamente o se hace un comentario para dejar explicito que se está teniendo en cuenta ese punto.

•Actividad práctica