Diseño Basado en Microprocesadores

Tema 2. Microcontroladores

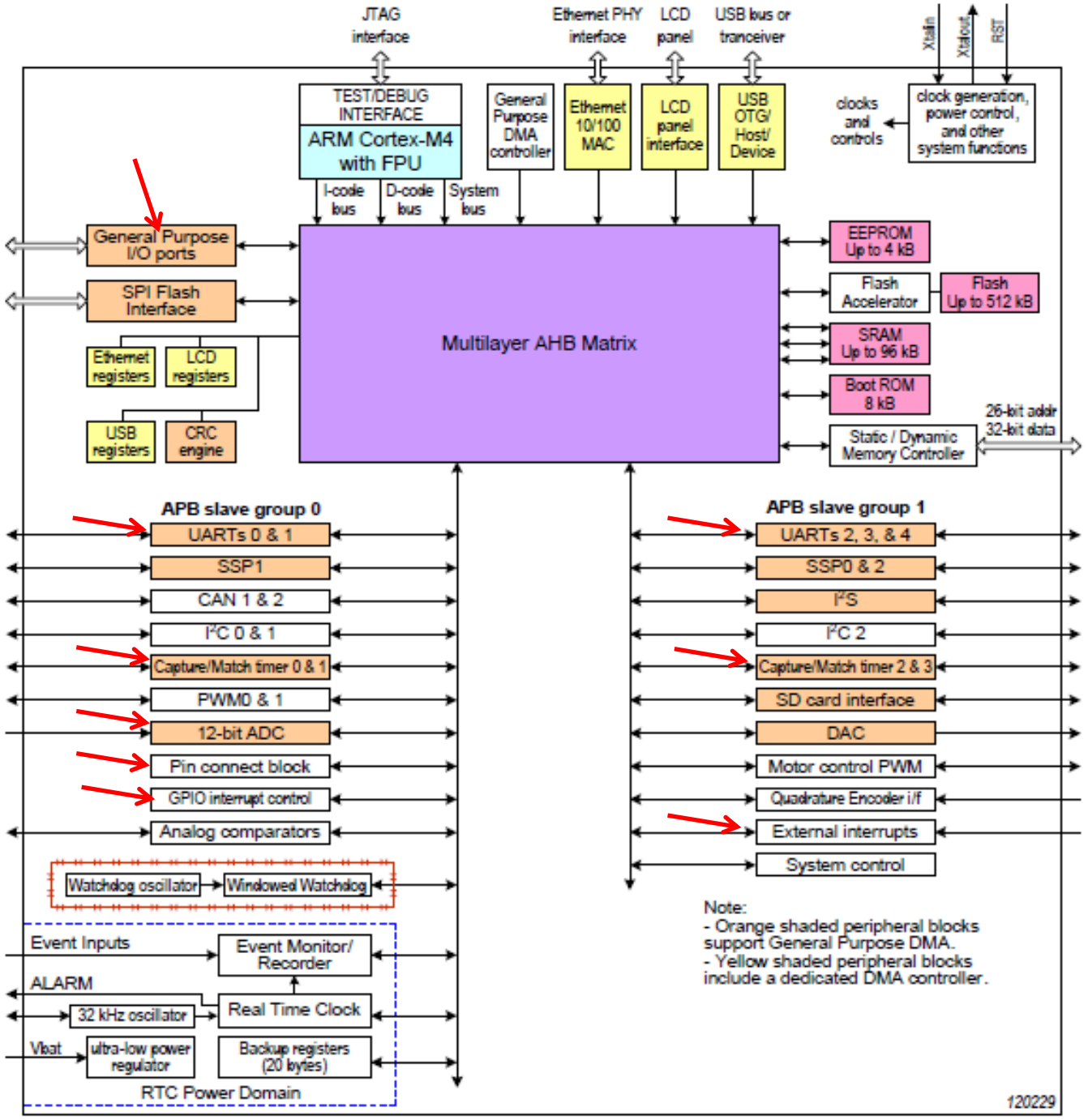
- 2.1. Introducción a los microcontroladores
- 2.2. Entradas/Salidas Digitales
- 2.3. Temporizadores
- 2.4. Excepciones
- 2.5. Conversión Analógica/Digital
- 2.6. Comunicación serie RS232C
- 2.7. Teclado, conversión D/A y sonido
- 2.8. Interfaz I2C

2.2. Entradas/salidas digitales

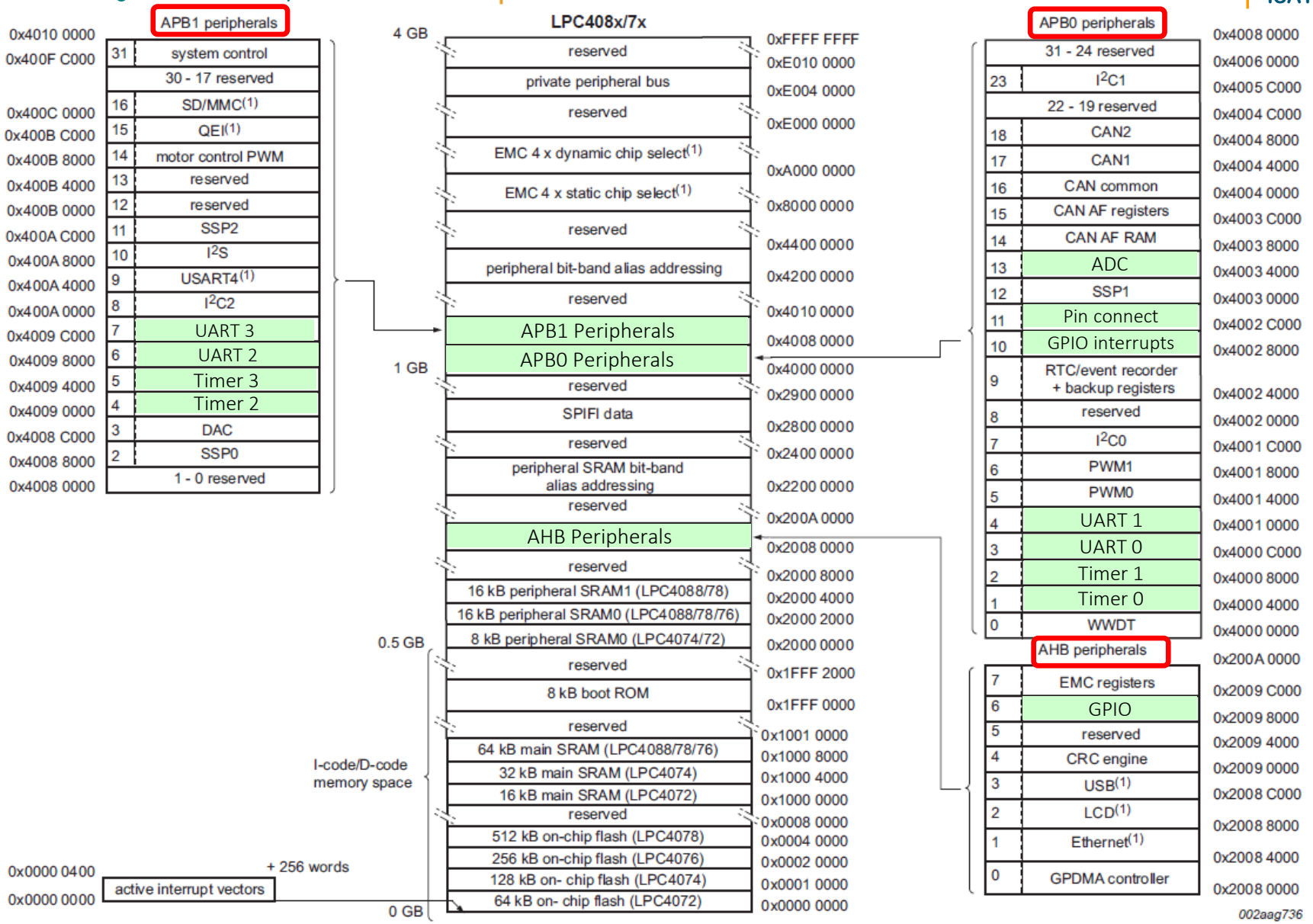
- 2.2.1. Introducción periféricos
- 2.2.2. Mapa de registros de periféricos
- 2.2.3. Pines de entrada/salida del LPC4088
- 2.2.4. Funciones de los pines de E/S
- 2.2.5. Registros de configuración de pines
- 2.2.6. Acceso a los registros de configuración
- 2.2.7. Características GPIO
- 2.2.8. Aplicaciones. Conexión de E/S
- 2.2.9. Tabla de registros GPIO
- 2.2.10. Información en los pines GPIO
- 2.2.11. Acceso a los registros GPIO
- 2.2.12. Biblioteca gpio_lpcxx.h
- 2.2.13. Ejemplos

UM10562 (Rev. 3, 12-Marzo-2014)
Cap. 6: LPC408x/LPC407x Pin configuration
Cap. 7: LPC408x/LPC407x I/O configuration
Cap. 8: LPC408x/LPC407x GPIO

Introducción. Periféricos



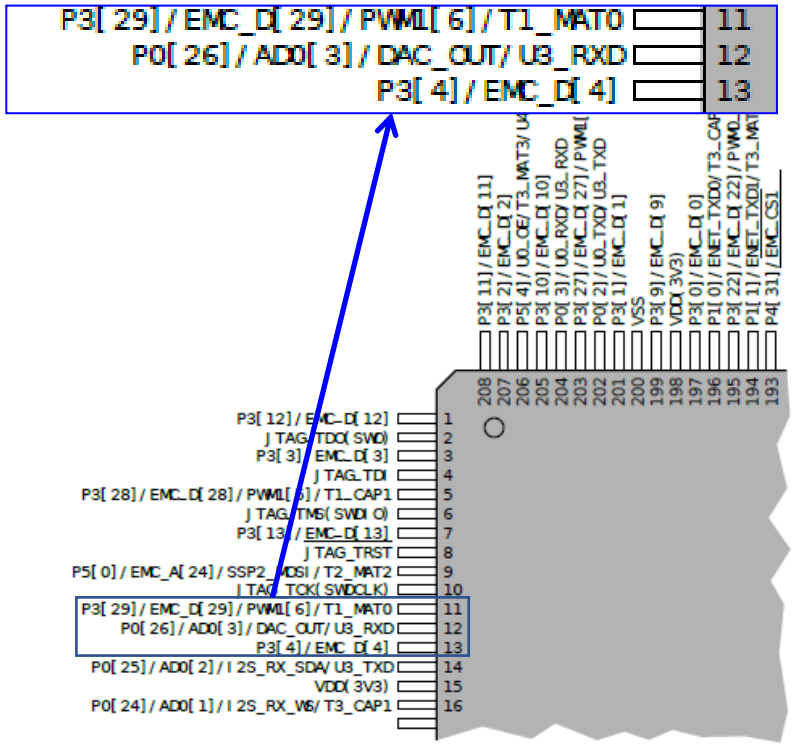
Mapa de memoria de periféricos



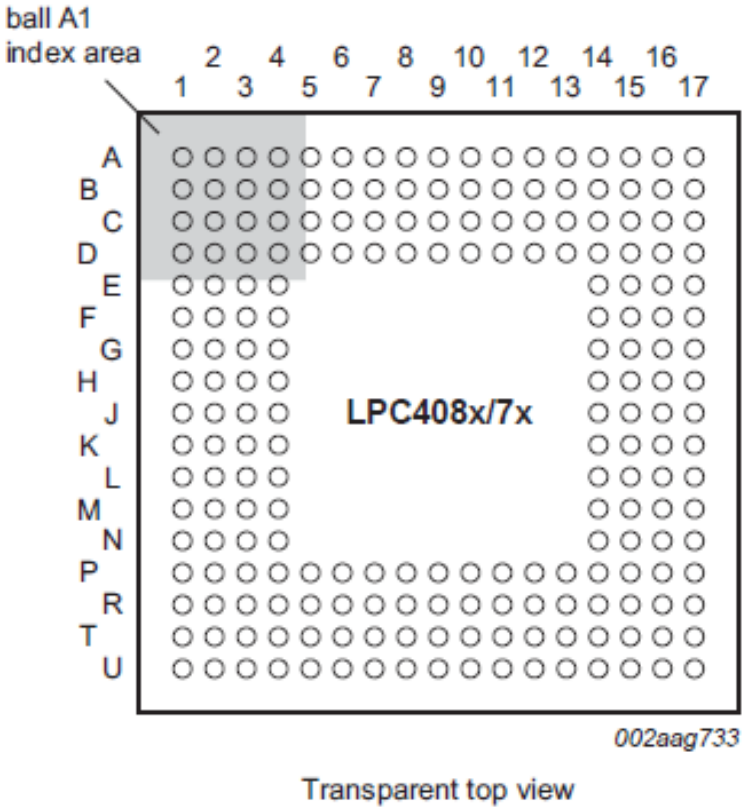
(1) Not available on all parts. See Table 2 and Table 4.

Pines del LPC4088

Tabla 75, Cap. 6

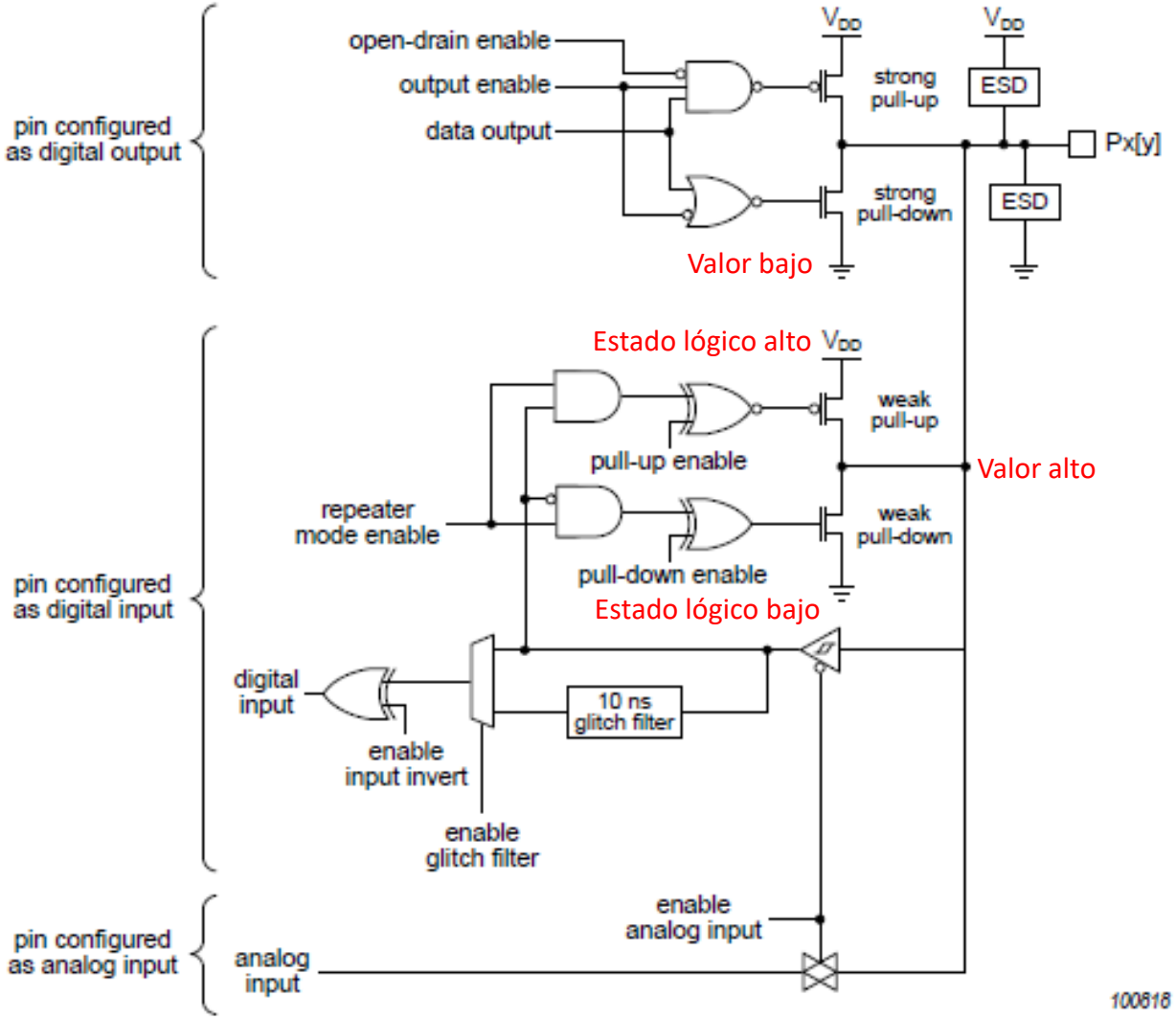


Encapsulado LQFP208



Encapsulado TFBGA208

Configuración I/O de pines (UM10562 Cap 7)



Pines de entrada/salida

Puerto	Nº líneas	Líneas disponibles
P0	[0 .. 31]	(5) 11,15,16,17,18
P1	[0 .. 31]	(10) 0,1,4,8,9,10,14,15,16,17
P2	[0 .. 31]	(8) 7,16,17,18,20,24,28,29
P3	[0 .. 31]	(16) 16 a 31
P4	[0 .. 31]	(3) 28,29,31
P5	[0 .. 4]	(0)
Total	165	(42)

Consultar LPC4088 Developer's Kit Use's Guide pág.. 42

Registros de configuración de pines I/O

Tablas 77 a 82

Table 77. I/O Control registers for port 0

Port pin	Register	Access	Reset Value[1]	Address	IOCON type[2]	208-pin	180-pin	144-pin	80-pin
P0[0]	IOCON_P0_0	R/W	0x030	0x4002 C000	D (tables 83, 84)	x	x	x	x
P0[7]	IOCON_P0_7	R/W	0x0A0	0x4002 C01C	W (tables 91, 92)	x	x	x	x
P0[12]	IOCON_P0_12	R/W	0x1B0	0x4002 C030	A (tables 85, 86)	x	x	x	-
P0[28]	IOCON_P0_28	R/W	0	0x4002 C070	I (tables 89, 90)	x	x	x	-
P0[29]	IOCON_P0_29	R/W	0	0x4002 C074	U (tables 87, 88)	x	x	x	x

Registros IOCON_Px_nn (0x4002 C000 a 0x4002 C290)

Tipo IOCON	Se aplica a pines
D	GPIO
A	con función analógica
U	con función USB D+ o D-
I	Con función I2C ocupado por la LCD
W	Igual que D pero con filtro glitch de entrada

Funciones del pin de E/S

Tablas 84, 86, 88, 90, 92

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P1_15	P1[15]	ENET_RX_CLK		I2C2_SDA				
IOCON_P1_18	P1[18]	USB_UP_LED1	PWM1[1]	T1_CAP0		SSP1_MISO		
IOCON_P1_19	P1[19]	USB_TX_E1	USB_PPWR1	T1_CAP1	MC_0A	SSP1_SCK	U2_OE	
IOCON_P1_20	P1[20]	USB_TX_DP1	PWM1[2]	QEI_PHA	MC_FB0	SSP0_SCK	LCD_VD[6]	LCD_VD[10]
IOCON_P1_21	P1[21]	USB_TX_DM1	PWM1[3]	SSP0_SSEL	MC_ABORT		LCD_VD[7]	LCD_VD[11]
IOCON_P1_22	P1[22]	USB_RCV1	USB_PWRD1	T1_MAT0	MC_0B	SSP1_MOSI	LCD_VD[8]	LCD_VD[12]

Table 86. Type A I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_12	P0[12]	USB_PPWR2	SSP1_MISO	ADC0[6]				
IOCON_P0_13	P0[13]	USB_UP_LED2	SSP1_MOSI	ADC0[7]				
IOCON_P0_23	P0[23]	ADC0[0]	I2S_RX_SCK	T3_CAP0				
IOCON_P0_24	P0[24]	ADC0[1]	I2S_RX_WS	T3_CAP1				
IOCON_P0_25	P0[25]	ADC0[2]	I2S_RX_SDA	U3_TXD				
IOCON_P0_26	P0[26]	ADC0[3]	DAC_OUT	U3_RXD				
IOCON_P1_30	P1[30]	USB_PWRD2	USB_VBUS	ADC[4]	I2C0_SDA	U3_OE		
IOCON_P1_31	P1[31]	USB_OVRCR2	SSP1_SCK	ADC[5]	I2C0_SCL			

Tablas 83, 85, 87, 89, 91

Table 83. Type D IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 84 for specific values.	000
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control). See Section 7.3.2 "Pin mode".	10
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	HYS		Hysteresis. See Section 7.3.3 "Hysteresis".	1
		0	Disable.	
		1	Enable.	
6	INV		Input polarity. See Section 7.3.4 "Input Inversion".	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
8:7	-		Reserved. Read value is undefined, only zero should be written.	NA
9	SLEW		Driver slew rate. See Section 7.3.7 "Output slew rate".	0
		0	Standard mode. Output slew rate control is enabled. More outputs can be switched simultaneously.	
		1	Fast mode. Slew rate control is disabled. This mode reduces the output delay by 1 ns compared to the standard mode. Fast mode is recommended for pins used with the EMC, LCD, and SPIFI interfaces.	
10	OD		Controls open-drain mode. See Section 7.3.9 "Open-Drain Mode".	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
31:11	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 85. Type A IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 86 for specific values.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control). See Section 7.3.2 "Pin mode".	10
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	-		Reserved. Read value is undefined, only zero should be written.	NA
6	INV		Input polarity. See Section 7.3.4 "Input Inversion".	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	ADMODE		Select Analog/Digital mode. See Section 7.3.5 "Analog/digital mode".	1
		0	Analog mode.	
		1	Digital mode.	
8	FILTER		Controls glitch filter. See Section 7.3.6 "Input filter".	1
		0	Noise pulses below approximately 10 ns are filtered out	
		1	No input filtering is done	
9	-		Reserved. Read value is undefined, only zero should be written.	NA
10	OD		Controls open-drain mode. See Section 7.3.9 "Open-Drain Mode".	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
14:11	-		Reserved. Read value is undefined, only zero should be written.	NA
16	DACEN		DAC enable control. This bit applies only to P0[26], which includes the DAC output function DAC_OUT. See Section 7.3.10 "DAC enable".	0
		0	DAC is disabled	
		1	DAC is enabled	
31:17	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 87. Type U IOCON registers bit description

Bit	Symbol	Description	Reset value
2:0	FUNC	Selects pin function. See Table 88 for specific values.	000
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 89. Type I IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 90 for specific values.	0
5:3	-		Reserved. Read value is undefined, only zero should be written.	NA
6	INV		Input polarity. See Section 7.3.4 "Input Inversion".	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	-		Reserved. Read value is undefined, only zero should be written.	NA
8	HS		Configures I^2C features for standard mode, fast mode, and Fast Mode Plus operation. See Section 7.3.8 "I^2C modes".	0
		0	I^2C 50ns glitch filter and slew rate control enabled.	
		1	I^2C 50ns glitch filter and slew rate control disabled.	
9	HIDRIVE		Controls sink current capability of the pin, only for P5[2] and P5[3]. See Section 7.3.8 "I^2C modes".	0
		0	Output drive sink is 4 mA. This is sufficient for standard and fast mode I^2C.	
		1	Output drive sink is 20 mA. This is needed for Fast Mode Plus I^2C. Refer to the appropriate specific device data sheet for details.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 91. Type W IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 92 for specific values.	000
4:3	MODE		Selects the output functional mode for the pin (on-chip pull-up/pull-down resistor control). See Section 7.3.2 "Pin mode".	[1]
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	HYS		Hysteresis. See Section 7.3.3 "Hysteresis".	1
		0	Disable.	
		1	Enable.	
6	INV		Input polarity. See Section 7.3.4 "Input Inversion".	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	ADMODE		Select Analog/Digital mode. See Section 7.3.5 "Analog/digital mode".	1
		0	Analog mode.	
		1	Digital mode.	
8	FILTER		Controls glitch filter. See Section 7.3.6 "Input filter".	0
		0	Noise pulses below approximately 10 ns are filtered out	
		1	No input filtering is done	
9	SLEW		Driver slew rate. See Section 7.3.7 "Output slew rate".	0
		0	Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously.	
		1	Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details.	
10	OD		Controls open-drain mode. See Section 7.3.9 "Open-Drain Mode".	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
31:11	-		Reserved. Read value is undefined, only zero should be written.	NA

Acceso a los registros de configuración

Para los IOCON se definen las siguientes etiquetas:

```
#define LPC_APB0_BASE     (0x40000000UL)
#define LPC_IOCON_BASE  (LPC_APB0_BASE + 0x2C000)
#define LPC_IOCON   ((LPC_IOCON_TypeDef     *) LPC_IOCON_BASE)

typedef struct{
__IO uint32_t P0_0;             /* 0x000 */
__IO uint32_t P0_1;
      ……..……
__IO uint32_t P5_3;
__IO uint32_t P5_4;             /* 0x290 */
} LPC_IOCON_TypeDef;
```

Para configurar cada PIN de cada puerto, escribir en:

LPC_IOCON -> Pn_x = valor de config. (con n=0..4 y x=0..31; n=5 y x=0..4)

Configuración I/O para un pin

1. Buscar el pin en las tablas 77 a 82 del manual.
2. Obtener de qué tipo es el pin.
3. Consultar las funciones del pin en una de las tablas 84, 86, 88, 90, 92 .
4. Obtener la combinación del campo FUNC para el registro IOCON correspondiente.
5. Consultar la estructura del registro IOCON correspondiente en una de las tablas 83, 85, 87, 89, 91.
6. Programar la función deseada en los bits 2:0 del registro IOCON correspondiente. Si la función es analógica, poner el bit ADMODE a 0.

Ejemplo: Seleccionar la función ADC0[4] para el pin P1[30]:

P1[30]/USB_PWRD2/USB_VBUS/ ADC0[4]/I2C0_SDA/U3_OE
000 /001 /010 /011 /100 /101

1. Localizamos el pin P1[30] en la tabla 78.
2. En la tabla 78 vemos que el pin es de tipo A.
3. Los pines tipo A se recogen en la tabla 86.
4. El valor para el campo FUNC es 011.
5. La estructura de registros IOCON de los pines tipo A se indica en la tabla 85.
6. LPC_IOCON->P1_30 = 3; /* FUNC = 11, ADMODE = 0, sin pull-up/pull-down */

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DAC EN	-	-	-	-	OD	-	FILT ER	AD MO DE	INV	-	MODE		FUNC		

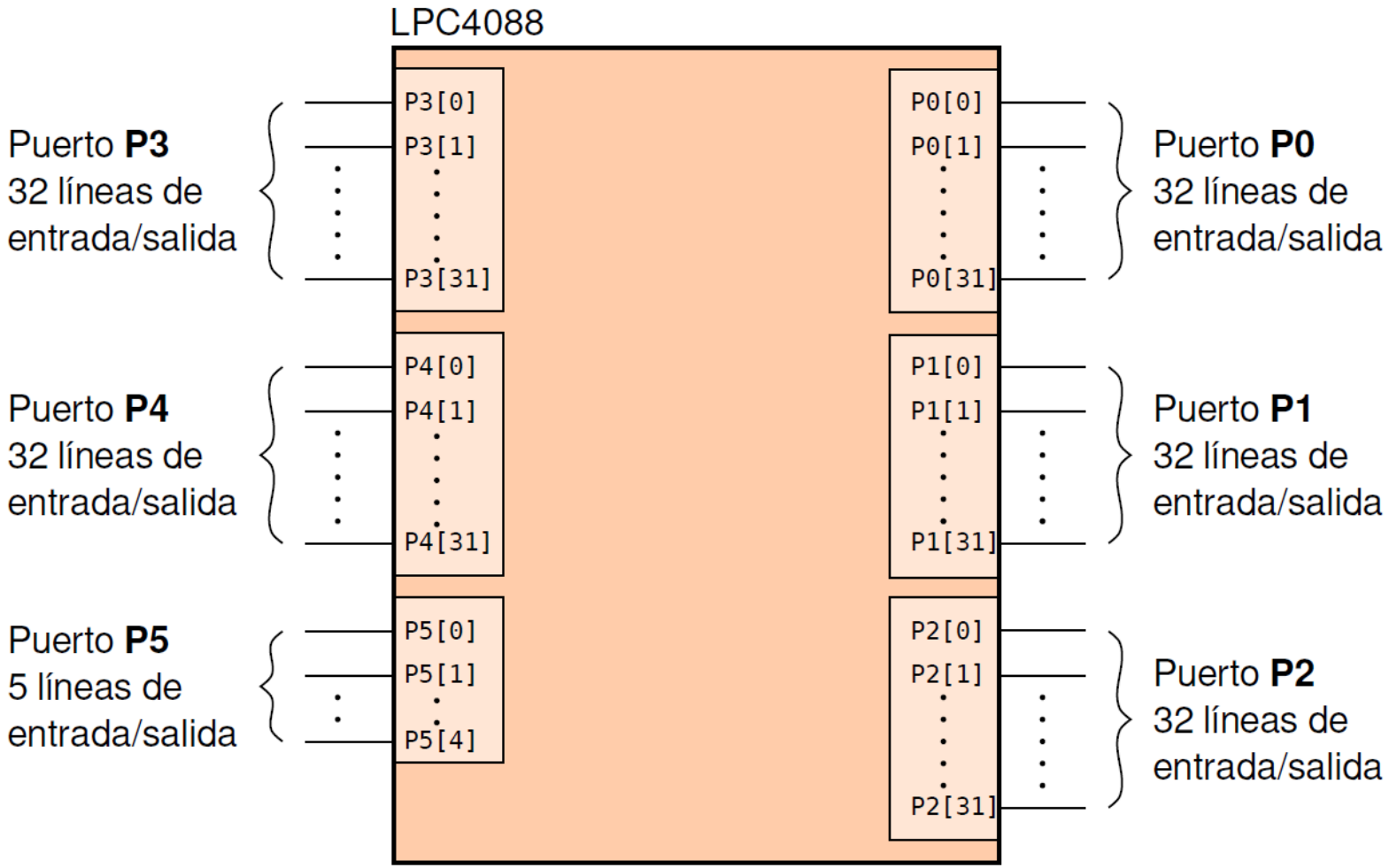
Características GPIO (1)

GPIO ➡ ***General Purpose Input/Output***

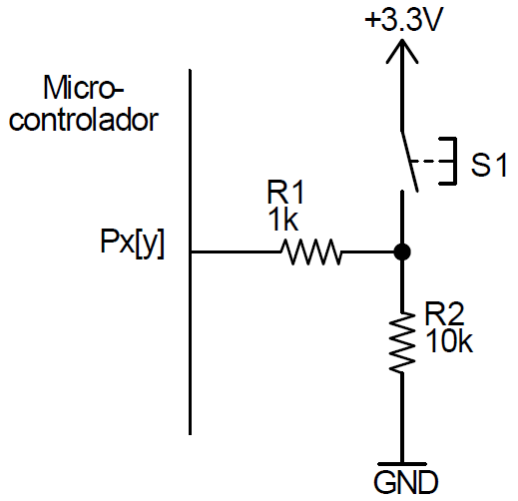
1)Todos los puertos usan el bus AHB (Advanced High-performance Bus) más rápido que el bus APB
2)Tiene registro de máscara para seleccionar pins activos
3)Todos los registros GPIO son accesibles por DMA
4)Todos los registros son direccionables por bytes, halfword y word
5)Todos los GPIO después del reset son por defecto entrada pull-up
6)Puertos P0 y P2 permiten interrupciones
7)Aplicaciones:
- Manejo de LEDs y otros indicadores
- Lectura del estado de interruptores y pulsadores
- Intercambio de datos con otros dispositivos
- Generación de pulsos digitales

Características GPIO (2)

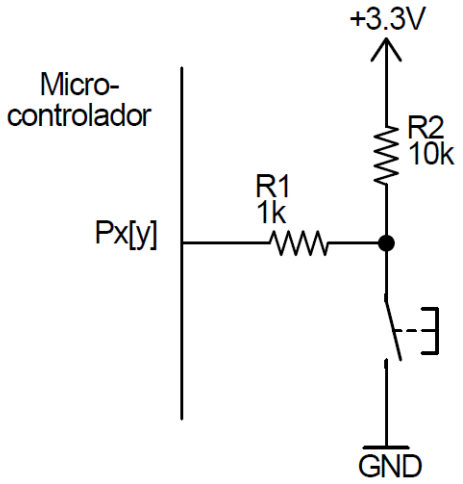
- Los pines de un puerto pueden usarse de forma conjunta o individual
- Cada pin individual se nombra Pnum_puerto[num_pin] Ejemplo: P3[5]
- Hasta 165 pines GPIO



Aplicaciones: Conexión de pulsadores

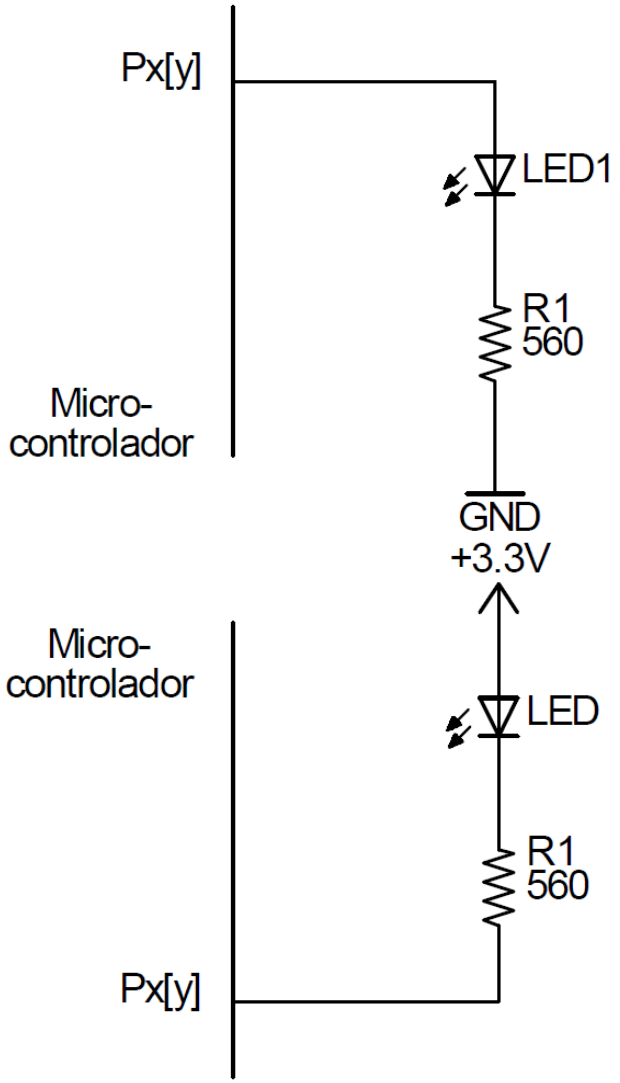


- Con esta conexión, cuando en el programa leamos el pin:
  - Encontraremos el pin 1 mientras el pulsador esté pulsado.
  - Encontraremos el pin 0 mientras el pulsador no esté pulsado.
- R2 establece un nivel eléctrico bajo en el pin mientras el pulsador no está pulsado. Por ello se llama resistencia de pull-down.
- R1 protege al pin si por error se configura como salida y se escribe a 0.



- Con esta conexión, cuando en el programa leamos el pin:
  - Encontraremos el pin 0 mientras el pulsador esté pulsado.
  - Encontraremos el pin 1 mientras el pulsador no esté pulsado.
- R2 establece un nivel eléctrico alto en el pin mientras el pulsador no está pulsado. Por ello se llama resistencia de pull-up.
- R1 protege al pin si por error se configura como salida y se escribe a 1.

Aplicaciones: Conexión de LEDs



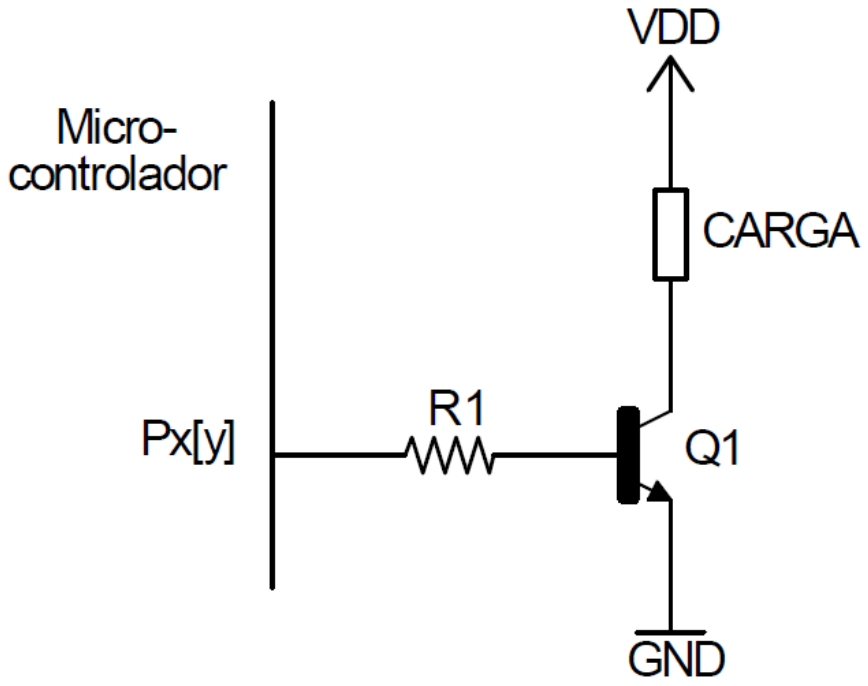
Con esta conexión, cuando en el programa escribamos en el pin:

- El LED se encenderá al escribir un 1
- El LED se apagará al escribir un 0

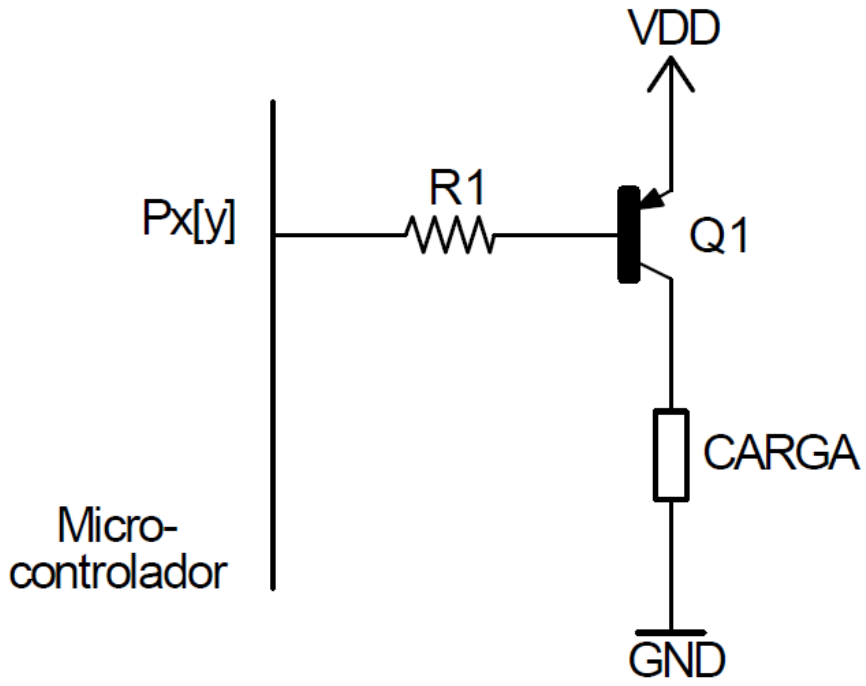
Con esta conexión, cuando en el programa escribamos en el pin:

- El LED se encenderá al escribir un 0
- El LED se apagará al escribir un 1

Aplicaciones: Conexión de cargas mediante transistor

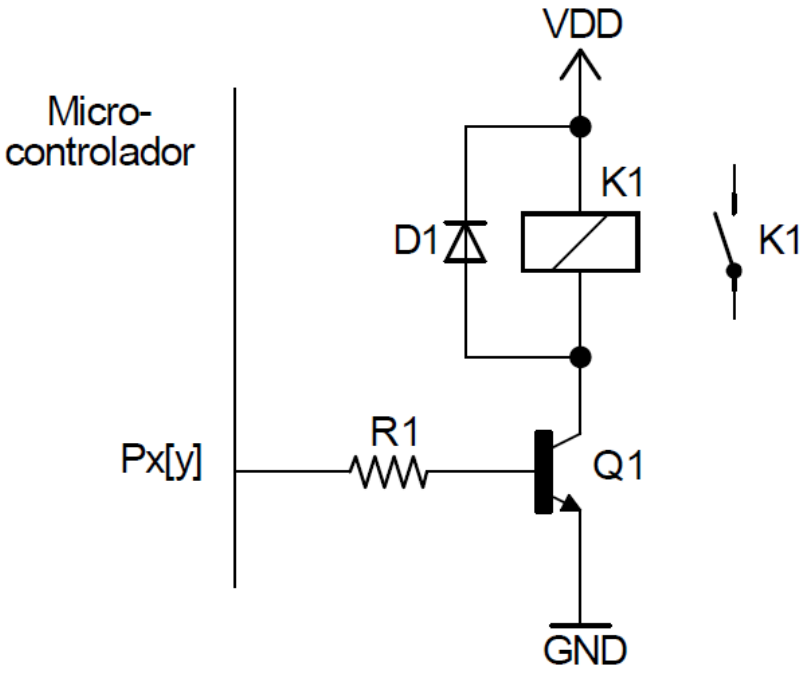


Pin a 1 enciende. Pin a 0 apaga.

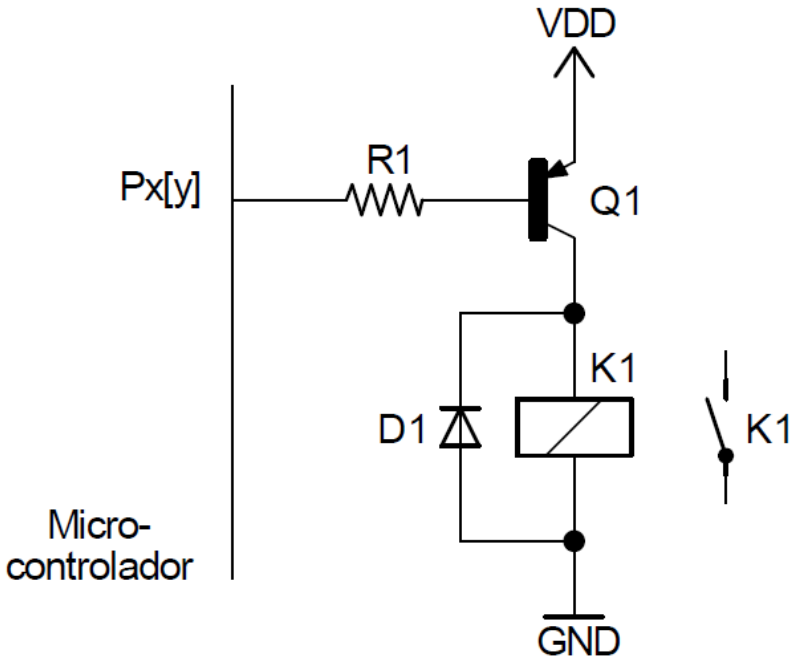


Pin a 0 enciende. Pin a 1 apaga.

Aplicaciones: Conexión de cargas mediante relé



Pin a 1 activa el relé. Pin a 0 lo desactiva.



Pin a 0 activa el relé. Pin a 1 lo desactiva.

Mapa de registros GPIO

Table 94. Register overview: GPIO (base address 0x2009 8000)

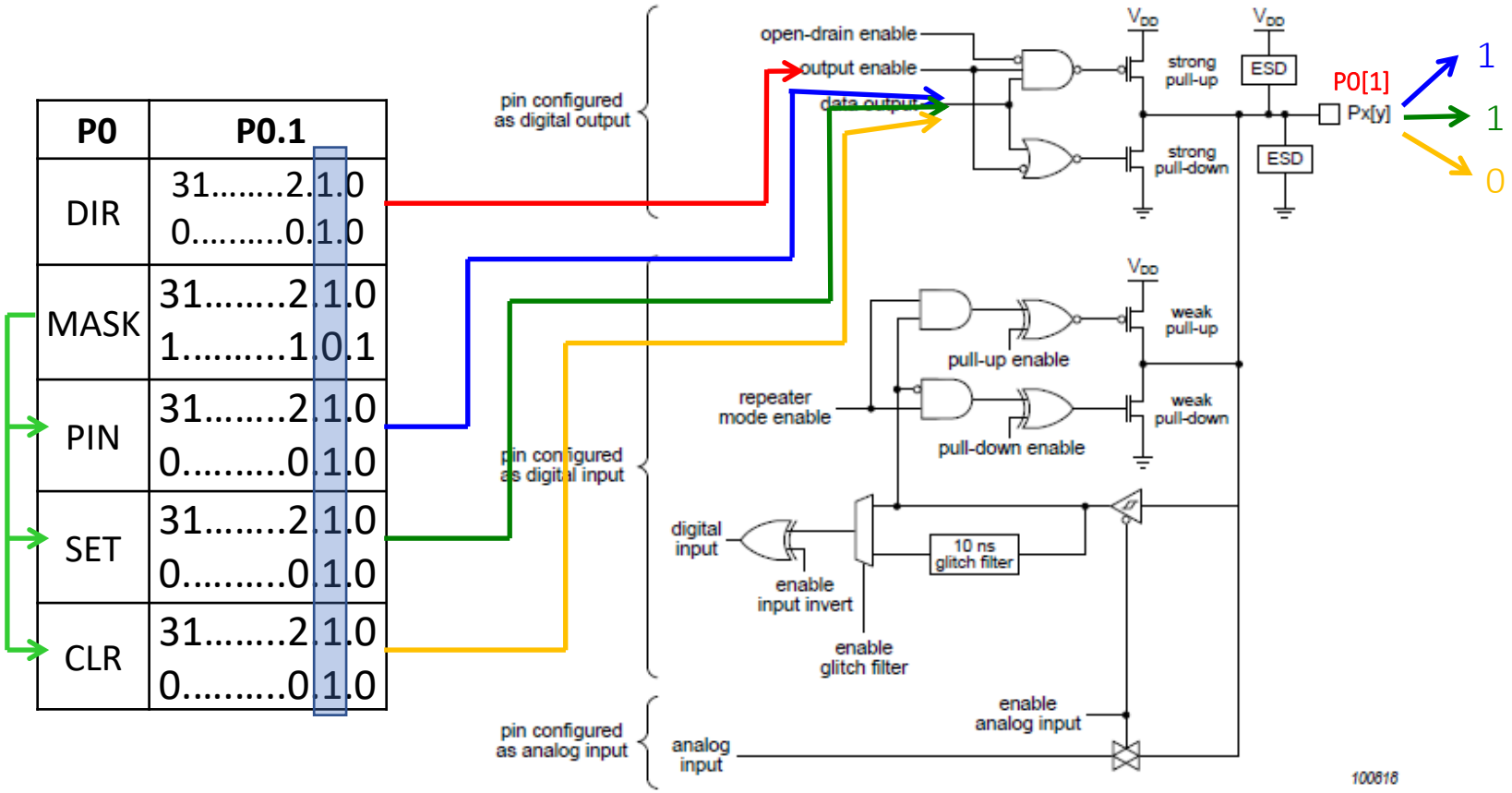
Name	Access	Address offset	Description	Reset value	Table
DIR0	R/W	0x000	GPIO Port0 Direction control register.	0	96
MASK0	R/W	0x010	Mask register for Port0.	0	97
PIN0	R/W	0x014	Port0 Pin value register using FIOMASK.	0	98
SET0	R/W	0x018	Port0 Output Set register using FIOMASK.	0	99
CLR0	WO	0x01C	Port0 Output Clear register using FIOMASK.	-	100
DIR1	R/W	0x020	GPIO Port1 Direction control register.	0	96
MASK1	R/W	0x030	Mask register for Port1.	0	97
PIN1	R/W	0x034	Port1 Pin value register using FIOMASK.	0	98
SET1	R/W	0x038	Port1 Output Set register using FIOMASK.	0	99
CLR1	WO	0x03C	Port1 Output Clear register using FIOMASK.	-	100
DIR2	R/W	0x040	GPIO Port2 Direction control register.	0	96
MASK2	R/W	0x050	Mask register for Port2.	0	97
PIN2	R/W	0x054	Port2 Pin value register using FIOMASK.	0	98
SET2	R/W	0x058	Port2 Output Set register using FIOMASK.	0	99
CLR2	WO	0x05C	Port2 Output Clear register using FIOMASK.	-	100
DIR3	R/W	0x060	GPIO Port3 Direction control register.	0	96
MASK3	R/W	0x070	Mask register for Port3.	0	97
PIN3	R/W	0x074	Port3 Pin value register using FIOMASK.	0	98
SET3	R/W	0x078	Port3 Output Set register using FIOMASK.	0	99
CLR3	WO	0x07C	Port3 Output Clear register using FIOMASK.	-	100
DIR4	R/W	0x080	GPIO Port4 Direction control register.	0	96
MASK4	R/W	0x090	Mask register for Port4.	0	97
PIN4	R/W	0x094	Port4 Pin value register using FIOMASK.	0	98
SET4	R/W	0x098	Port4 Output Set register using FIOMASK.	0	99
CLR4	WO	0x09C	Port4 Output Clear register using FIOMASK.	-	100
DIR5	R/W	0x0A0	GPIO Port5 Direction control register.	0	96
MASK5	R/W	0x0B0	Mask register for Port5.	0	97
PIN5	R/W	0x0B4	Port5 Pin value register using FIOMASK.	0	98
SET5	R/W	0x0B8	Port5 Output Set register using FIOMASK.	0	99
CLR5	WO	0x0BC	Port5 Output Clear register using FIOMASK.	-	100

Mapa de registros

Nombre	Descripción	Reset
DIR n n=0..5	Registro de control de las direcciones del puerto GPIO de cada pin. (Los 32 pins de cada puerto están configurados por defecto como entradas. Si se desea configurarlos como salidas deben ponerse a 1)	0x0
MASKn n=0..5	Registro máscara de los bits del puerto. Los bits a 0 están habilitados. Permiten escribir vía PIN, SET, y CLR, y leer con PIN.	0x0
PINn n=0..5	El estado de los pins del puerto son leídos y enmascarados con el inverso de MASK (Al leer el puerto los bits no habilitados son puestos a 0). Si se escribe en el registro los valores son colocados en los bits habilitados por 0 de MASK.	0x0
SETn n=0..5	Este registro controla el estado de salida de los pins usando el enmascaramiento con el inverso de MASK. Escribiendo 1s se produce nivel alto en los pins del puerto. Si se lee el registro se obtiene el contenido de salida del puerto.	0x0
CLRn n=0..5	Este registro controla el estado de salida de los pins usando el enmascaramiento con el inverso de MASK. Escribiendo 1s se produce nivel bajo en los pins del puerto.	0x0

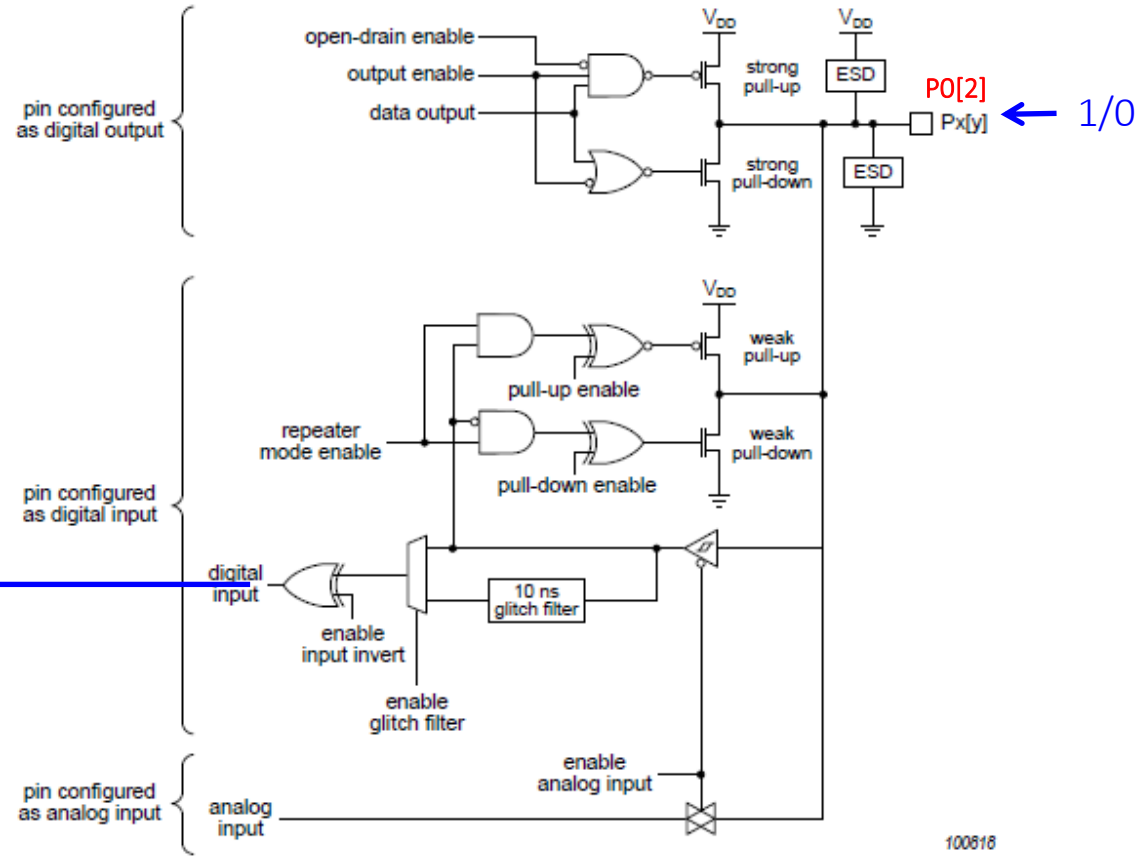
Pin P0[1] como salida

P0	P0.1
DIR	31.....2.1.0 0.....0.1.0
MASK	31.....2.1.0 1.....1.0.1
PIN	31.....2.1.0 0.....0.1.0
SET	31.....2.1.0 0.....0.1.0
CLR	31.....2.1.0 0.....0.1.0



Pin P0[2] como entrada

P0	P0.2
DIR	31.....2.1.0 0.....0.0.0
MASK	31.....2.1.0 1.....0.1.1
PIN	31.....2.1.0 0.....1/0.0.0
SET	31.....2.1.0 0.....1.0.0
CLR	31.....2.1.0 0.....0.0.0



Acceso a los registros (1)

En el fichero LPC407x_8x_177x_8x.h se definen las direcciones correspondientes a todos los registros de los periféricos internos del LPC4088. Para los GPIO se definen las siguientes etiquetas:

```
#define LPC_AHB_BASE          (0x20080000UL)

#define LPC_GPIO0_BASE        (LPC_AHB_BASE  + 0x18000)
#define LPC_GPIO1_BASE        (LPC_AHB_BASE  + 0x18020)
#define LPC_GPIO2_BASE        (LPC_AHB_BASE  + 0x18040)
#define LPC_GPIO3_BASE        (LPC_AHB_BASE  + 0x18060)
#define LPC_GPIO4_BASE        (LPC_AHB_BASE  + 0x18080)
#define LPC_GPIO5_BASE        (LPC_AHB_BASE  + 0x180A0)

#define LPC_GPIO0             ((LPC_GPIO_TypeDef*) LPC_GPIO0_BASE)
#define LPC_GPIO1             ((LPC_GPIO_TypeDef*) LPC_GPIO1_BASE)
#define LPC_GPIO2             ((LPC_GPIO_TypeDef*) LPC_GPIO2_BASE)
#define LPC_GPIO3             ((LPC_GPIO_TypeDef*) LPC_GPIO3_BASE)
#define LPC_GPIO4             ((LPC_GPIO_TypeDef*) LPC_GPIO4_BASE)
#define LPC_GPIO5             ((LPC_GPIO_TypeDef*) LPC_GPIO5_BASE)
```

Acceso a los registros (2)

```
typedef struct
{ __IO uint32_t DIR;
       uint32_t RESERVED0[3];
  __IO uint32_t MASK;
  __IO uint32_t PIN;
  __IO uint32_t SET;
  __O  uint32_t CLR;
} LPC_GPIO_TypeDef;
```

1) Para escribir en un registro de un puerto:

LPC_GPIOn->DIR = valor (n=0..5)

2) Para leer un registro de un puerto:

Variable= LPC_GPIOn -> PIN (n=0..5)

Biblioteca gpio_lpcxx.h (1)

*gpio_regs:	PUERTO0..PUERTO5 puntero a cada puerto
mascara_pin:	PIN0..PIN31 máscara de selección del pin o pines.
dirección:	DIR_ENTRADA o DIR_SALIDA
valor:	TRUE, FALSE

1) Configurar la dirección de uno o más pines.

void **gpio_ajustar_dir**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin, uint32_t direccion)

2) Obtener la dirección de uno o mas pines.

uint32_t **gpio_obtener_dir**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin);

3) Leer el estado de un pin.

static inline bool_t **gpio_leer_pin**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin)

4) Leer el estado de un puerto completo.

static inline uint32_t **gpio_leer_puerto**(LPC_GPIO_TypeDef *gpio_regs)

Biblioteca gpio_lpcxx.h (2)

5) Establecer el estado de uno o más pines de salida al mismo estado.

static inline void **gpio_escribir_pin**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin, bool_t valor)

6) Establecer el estado de los pines de salida de un puerto.

static inline void **gpio_escribir_puerto**(LPC_GPIO_TypeDef *gpio_regs, uint32_t valor)

7) Poner a 1 uno o más pines de salida.

static inline void **gpio_pin_a_1**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin)

8) Poner a 0 uno o más pines de salida.

static inline void **gpio_pin_a_0**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin)

9) Invertir el estado de uno o más pines de salida.

static inline void **gpio_invertir_pin**(LPC_GPIO_TypeDef *gpio_regs, uint32_t mascara_pin)

Ejemplo (1)

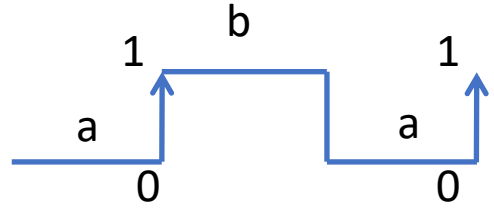
```
/*============================================================
Ejemplo de programación de los puertos de entrada/salida  desde C
Entrada1: P0 (0..7)
Entrada2: P0 (8..15)
Salida: P0 (16..23)   suma de entrada 1 y 2.
Versión: lectura del puerto P0 completo y separación de los datos.
 ============================================================*/
#include "LPC407x_8x_177x_8x.h"
int main(void)
{ uint8_t entrada1, entrada2, suma;

   LPC_GPIO0 -> DIR = 0xFF0000;
    while (1)
    {  entrada1 = LPC_GPIO0 -> PIN & 0xFF;
        entrada2 = (LPC_GPIO0 -> PIN >> 8) & 0xFF;
        suma = entrada1 + entrada2;
        LPC_GPIO0 -> PIN = (suma << 16);
    }
}
```

Ejemplo (2)

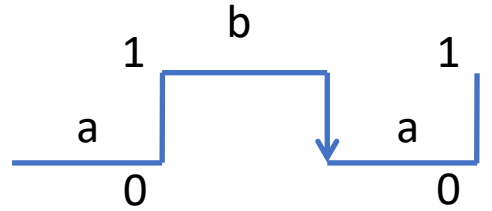
```
/*=====================================================
Ejemplo de programación de los puertos de entrada/salida  desde C
Entrada1: P0 (0..7)
Entrada2: P0 (8..15)
Salida: P0 (16..23)   suma de entrada 1 y 2.
Versión: acceso de forma separada a cada byte de los puertos .
 =====================================================*/
#include "LPC407x_8x_177x_8x.h"
int main(void)
{ uint8_t entrada1, entrada2, suma;

    LPC_GPIO0->DIR[2] = 0xFF;
    while (1)
    {  entrada1 = LPC_GPIO0->PIN[0];
        entrada2 = LPC_GPIO0-> PIN[1];
        suma = entrada1 + entrada2;
        LPC_GPIO0-> PIN[2] = suma;
        /* O simplemente:
        LPC_GPIO0-> PIN[2] = LPC_GPIO0->PIN[0] + LPC_GPIO0-> PIN[1]; */
    }
}
```

Detección de flancos

Flanco de subida:	Casos
while (valor_pin);	- a(0) -> falso -> siguiente línea - b(1) -> cierto -> espero
while (NO valor_pin);	- a(0) -> no falso (cierto) -> espero el 1 - b(0) -> no falso (cierto) -> espero el 1

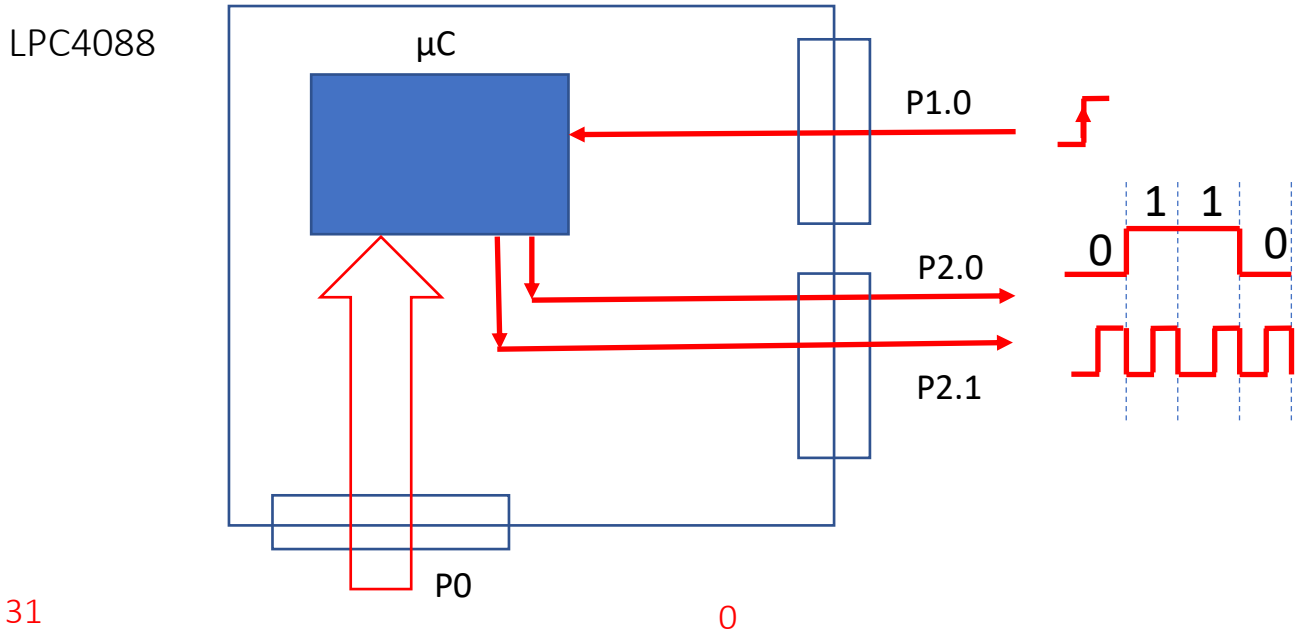


Flanco de bajada:	Casos
while (NO valor_pin);	- a(0) -> **no falso (cierto) -> espero** - b(1) -> no cierto (falso) -> **siguiente línea**
while (valor_pin);	- a(1) -> cierto -> espero el 0 - b(1) -> cierto -> espero el 0



Ejemplo (3)

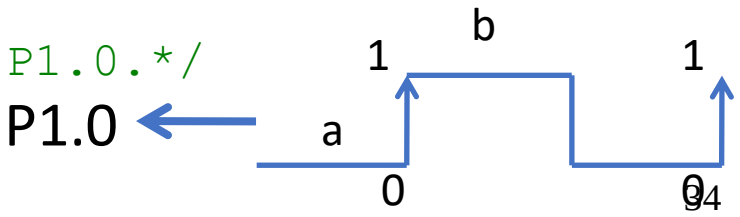
Realizar un programa que lea una entrada digital de 32 bits por el puerto P0 cuando se produzca un flanco positivo en P1.0. Una vez leído enviarlo en serie por P2.0 desde el pin P0.0 al P0.31, validando cada bit con un pulso de reloj en P2.1 con semiperiodos iguales de tiempo no exacto (bucle for).



0000.0101.0000.1001.0000.1111.0011.1010

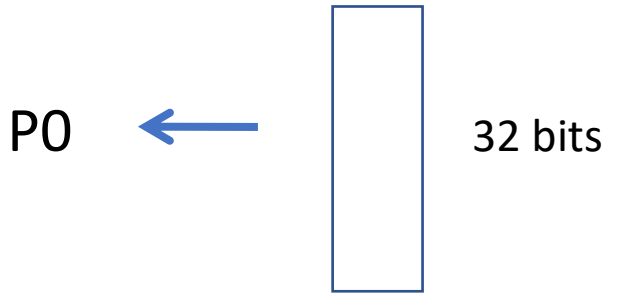
Ejemplo (3)

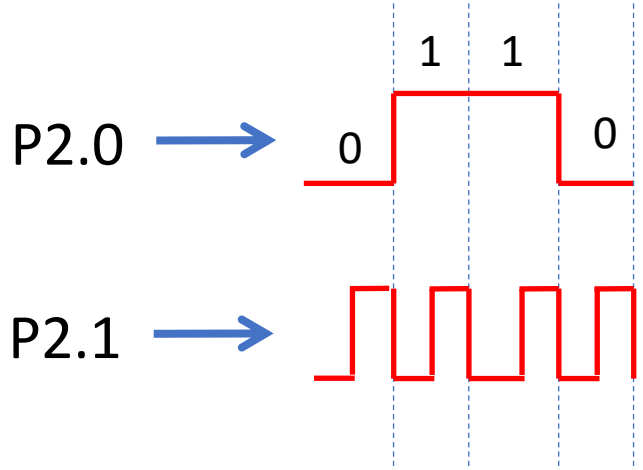
```
/*=================================================
Ejemplo de programación de los puertos de entrada/salida
desde C
El programa espera un flanco positivo en P1.0. Cuando se
produce lee P0 y lo envía en serie P0.0 a P0.31 por P2.0
validando con pulso de reloj en P2.1
=================================================*/
#include "LPC407x_8x_177x_8x.h"
int main(void)
{   uint32_t dato_entrada;
    uint16_t t;
    uint8_t i;
    /* Configurar los pines P2.0 y P2.1 como salidas. */
    LPC_GPIO2->DIR = 0x03;
    while (1)
    {  /* Esperar un flanco positivo en el pin P1.0.*/
       while (LPC_GPIO1->PIN & 0x01);
       while (!(LPC_GPIO1->PIN & 0x01));
```



Ejemplo (3)

```
        /* Leer el dato de 32 bits que llega por P0. */
          dato_entrada = LPC_GPIO0->PIN;

        for (i= 0; i < 32; i++)
        {  /* Enviar por el pin P2.0 el bit menos significativo de dato_entrada.*/
           if (dato_entrada & 0x01) LPC_GPIO2->SET = 0x01;
           else LPC_GPIO2->CLR = 0x01;

           /* Generar semiciclo a 0 de la señal de reloj.
           La duración no está calibrada.*/
           LPC_GPIO2->CLR = 0x02;
           for (t = 0; t < 10000; t++);
           /* Generar semiciclo a 1 de la señal de reloj. */
           LPC_GPIO2->SET = 0x02;
           for (t = 0; t < 10000; t++);
           /* Desplazar dato_entrada un bit a la derecha. */
           dato_entrada >>= 1;
          }
    }
}
```





Ejemplo (4)

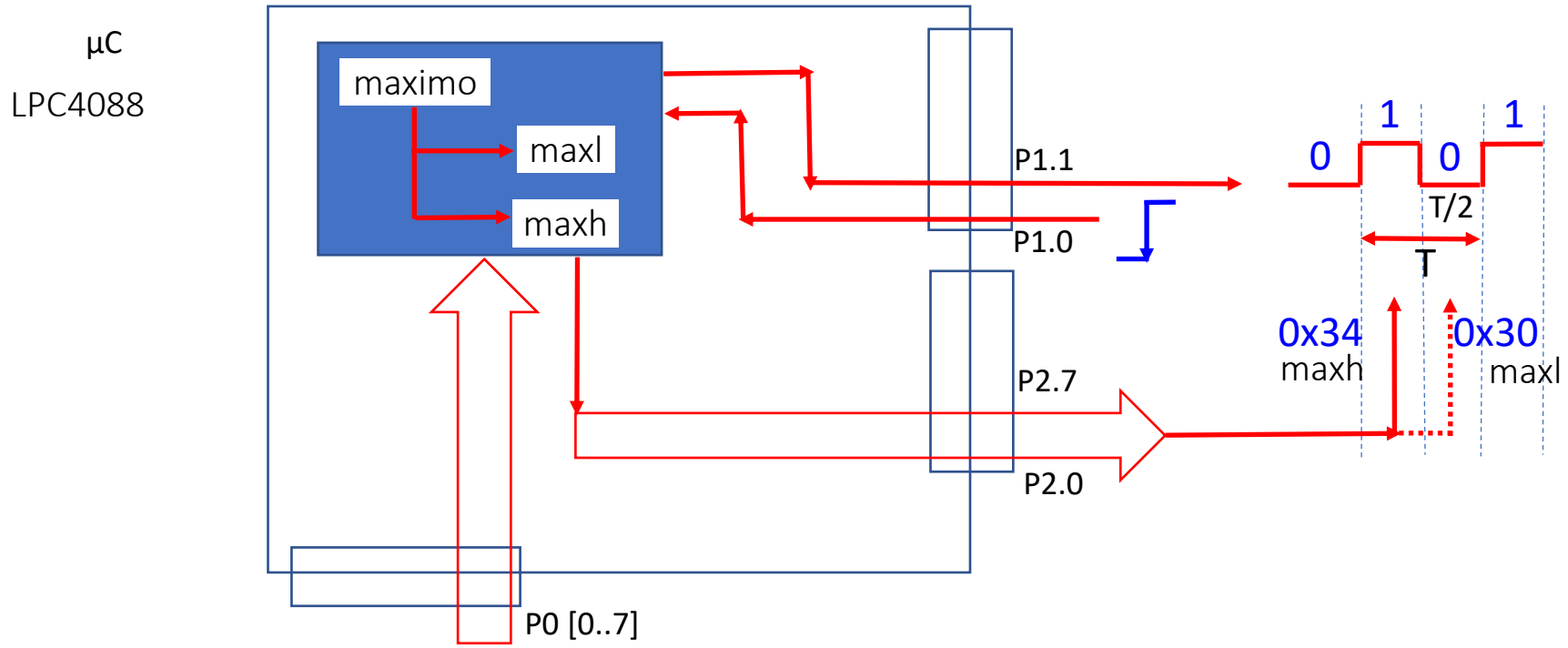
Realizar un programa en C, que calcule el máximo valor de una entrada digital tamaño byte por el puerto P0. Para leer el puerto P0 se debe esperar por P1.0 un flanco de bajada. El valor máximo disponible (tras ir comparando todas las entradas que vayan llegando) se debe enviar por el puerto P2 en ASCII-Hexadecimal (dos caracteres consecutivos por cada byte de entrada). Por cada valor máximo enviado hay que poner un pulso de reloj en el pin P1.1 de 50000 pasos el período. Primero se envía el ASCII del nible alto en el semiperíodo de nivel alto y después el ASCII del nible bajo en el semiperíodo de nivel bajo.

1.b.binatural.2[0100.0000] -> 2.b.ascii.16[0x34][0x30]

nºdatos.tamaño.codificacion.basenum

nºdatos:	1..n
tamaño:	bit, b(byte), hw, w(word), d(dobleword)
codificacion:	binat, ascii, asciibcd, asciihex, bcdemp, bcddesemp
basenum:	2,10,16

Ejemplo (4)



xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.0100.0000

31 0

1.b.binatural.2[0100,0000] -> 2.b.ascii.16[0x34][0x30]

Ejemplo (4)

```
main()
{ uint8_t máximo=0, entrada, maxh, maxl;
  LPC_GPIO2 -> DIR |= 0XFF;
  LPC_GPIO1 -> DIR |= 1<<1;
  while (1)
  {while (!(LPC_GPIO1 -> PIN & 0X01));
   while (LPC_GPIO1 -> PIN & 0X01);
   entrada= LPC_GPIO0 -> PIN;
   if (entrada>maximo) maximo=entrada;
   maxh=(máximo & 0xF0)>>4;
   if (maxh>9) maxh += 'A'-10; else maxh += '0';
   maxl=(maximo & 0x0F);
   if (maxl>9) maxl += 'A'-10; else maxl += '0';
   LPC_GIOP2 ->PIN = maxh;
   LPC_GPIO1 -> SET |= 1<<1;
   for (t=0; t<25000; t++);
   LPC_GIOP2 ->PIN = maxl;
   LPC_GPIO1 -> CLR |= 1<<1;
   for (t=0; t<25000; t++);
  }
}
```