Programación en C – ED3

Importar Librerias/Bibliotecas

#ifdef __USE_CMSIS

#include "LPC17xx.h"

#endif

Nos aseguramos que usemos CMSIS e incluir la librería para la placa LPC1769 que usamos, esto nos permite usar LPC_GPIO0 por ejemplo.

Utilizar pines de los puertos

PINCON

PINSEL

Con PINSEL se selecciona la función del pin, estas pueden ser pin de entrada/salida (IO), pin de entrada al ADC, pin para comunicación UART, etc. Se usan 2 bits por pin. VER TABLA PINSEL

LPC_PINCON->PINSEL1 &= ~(3<<12); //Pone pin 22 de puerto 0 como GPIO

LPC_PINCON->PINSEL4 |= (1<<20);

LPC_PINCON->PINSEL4 &= ~(1<<21); //Pin 10 de puerto 2 como pin de interrupción externa.

Primero pongo el 1 en 20 y después el 0 en 21, dejo el resto de los pines como estaban antes.

VER TABLA!

PINMODE

Con PINMODE se modifican las resistencias conectadas al pin. Pull up, pull down, repeater mode o nada. Esta configuración es independiente de PINSEL. Aunque no se usen los pines como IO las resistencias se cambian. VER TABLA PINMODE

LPC_PINCON->PINMODE0 |= (3<<4); //Pin 2 de puerto 0 con pulldown

LPC_PINCON->PINMODE1 &= ~(1<<16);

LPC_PINCON->PINMODE1 |= (1<<17); //Pin 24 de puerto 0 sin pull up o pull down

FIO FAST INPUT OUTPUT

FIODIR

Con FIODIR se configuran los pines como entrada o salida, 1 = salida, 0 = entrada

Ej:

LPC_GPIO0 -> FIODIR |= (1<<10); //Pin 10 de puerto 0 como salida;

LPC_GPIO2 -> FIODIR &= ~(7<<3); //Pines 3,4 y 5 del puerto 2 como entrada;

Cuando ponemos un pin en salida hacemos un |= (or) para que solamente se modifiquen en el FIODIR los bits que pusimos en 1 y todos los demás permanezcan iguales.

X OR 0 = X, X OR 1 = 1

Cuando ponemos un pin en entrada hacemos &= ~ (and y negamos el valor que ingresamos) para que solo los pines donde ponemos 1 (que se niegan y quedan en 0) sean modificados y los demás permanezcan iguales.

X AND 1 = X, X AND 0 = 0; por lo que: X AND ~1 = 0.

FIOMASK

Con FIOMASK se puede configurar si los pines se pueden modificar con FIOPIN,FIOSEL, FIOCLR, o no. Un 0 en la posición del pin deja que se pueda modificar, un 1 no.

Ej:

LPC_GPIO0->FIOMASK &= ~(3<<10); //Solo pines 10 y 11 pueden modificarse con FIO

FIOSET

Con FIOSET se pone un pin de salida en alto, poniendo un 1 en la posición de ese pin en FIOSET.

Ej:

LPC_GPIO0 -> FIOSET |= (1<<22); //Pone en alto el pin 22 del puerto 0

LPC_GPIO2 -> FIOSET |=(3<<6); //Pone en alto los pines 6 y 7 del puerto 2 (d'3'=b'11')

FIOCLR

Con FIOCLR se pone un pin de salida en bajo, poniendo un 1 en la posición de ese pin en FIOCLR.

Ej:

LPC_GPIO0 -> FIOCLR |= (1<<15); //Pone en bajo el pin 15 del puerto 0

LPC_GPIO1 -> FIOCLR |=(7<<10); //Baja los pines 10,11 y 12 del puerto 1 (d'7'=b'111')

FIOPIN

Leyendo FIOPIN se obtiene el estado de los pines del puerto. Escribiendo en FIOPIN se ponen los pines del puerto como se desean, sin necesidad de hacer FIOSET y FIOCLR.

```
LPC_GPIO0 -> FIOPIN |= (1<<22);          //Pone en alto el pin 22 del puerto 0

LPC_GPIO0 -> FIOPIN = (1<<22);           //Alto el pin 22 de p0 y pone en bajo todos los demas

LPC_GPIO0 -> FIOPIN &= ~ (1<<5);         //Pone en bajo el pin 5 del puerto 0

LPC_GPIO0 -> FIOPIN = LPC_GPIO1 -> FIOPIN; //Copia el puerto 1 en puerto 0

If (LPC_GPIO2 -> FIOPIN & (1<<8)) {…}    //Si pin 8 del puerto 2 está en alto, entonces hace {...}
```

SysTick Timer

Es un timer de 24 bits, de muy fácil uso.

Con la siguiente instrucción, se lo configura para que interrumpa cada una cantidad de ciclos de reloj deseada.

```
SysTick_Config( ciclos );            //Configurar SysTick para que interrumpa cada 'ciclos'
```

SystemCoreClock contiene la frecuencia del sistema.

Podemos configurar el SysTick para que interrumpa cada 1 milisegundo así:

```
SysTick_Config( SystemCoreClock / 1000);        //SysTick que interrumpa cada 1ms
```

Para atender la interrupción del SysTick es necesario que creemos una función llamada SysTick_Handler, dentro de ella incluimos el código que queremos que se ejecute.

```
void SysTick_Handler(void)  {

  ….

}
```

Interrupciones

En ARM las interrupciones se manejan con una tabla con las direcciones de las funciones que atienden a cada interrupción.

Con el controlador NVIC se pueden atender interrupciones anidadas, es decir interrupciones adentro de otras interrupciones.

Las interrupciones tienen prioridades, eso quiere decir que una interrupción solo es interrumpida por una de mayor prioridad, quedando en pausa y reanudando cuando la otra termine.

Las funciones para configurar interrupciones son (IRQn_Type IRQn es el tipo de interrupción, ver tabla):

Habilitacion:

void NVIC_EnableIRQ(IRQn_Type IRQn) //Habilita la interrupción.

void NVIC_DisableIRQ(IRQn_Type IRQn) //Deshabilita la interrupción

Pendiente:

uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn) //Devuelve 1 si interrup esta pendiente

void NVIC_SetPendingIRQ(IRQn_Type IRQn) //Setea la interrupción como pendiente

void NVIC_ClearPendingIRQ(IRQn_Type IRQn) //Saca de pendiente la interrupción

Una interrupción puede estar marcada como pendiente y no estar activa.

uint32_t NVIC_GetActive(IRQn_Type IRQn) //Devuelve 1 si la interrup esta activa

Si está activa pero pendiente también devuelve un 1. Una interrupción se marca como activa automáticamente cuando se comienza a atender, si se interrumpe por otra interrupción permanece activa. Se desmarca como activa cuando se termina de atender la interrupcion.

void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)

//Setea el valor de prioridad para la interrupción

uint32_t NVIC_GetPriority(IRQn_Type IRQn)

//Devuelve el valor de prioridad de la interrupcion

Los valores más bajos son de mayor importancia.

Las interrupciones son atendidas por handlers, son funciones que se ejecutan cuando ocurren estas interrupciones. Los nombres de las funciones de los handlers se encuentran en la TABLA!

Interrupción Externa

Las interrupciones externas usan los pines EINT0, EINT1, EINT2 y EINT3. Se deben seleccionar esas funciones con PINSEL.

EXTMODE configura si se interrumpe por nivel (0) o por flanco (1)

EXTPOLAR configura si se activa por bajo o flanco descendente (0) o por alto o flanco ascendente (1)

Si bien son registros de 32 bits se usan solo los 4 menos significativos (para las 4 interrupciones externas)

Se acceden con LPC_SC

Ej:

```
LPC_SC -> EXTMODE |= (1<<0);

LPC_SC -> EXTPOLAR &= ~(1<<0);

//Interrupcion EINT0 configurada para interrumpir por flanco descendente
```

Cuando se recibe una interrupción externa se setea una flag que debemos bajar cuando atendemos la interrupción en el handler para poder reconocer futuros eventos. Debemos escribir un 1 en EXTINT en la posición de la interrupcion deseada 0,1,2 o 3 para bajar la bandera.

Ej:

```
LPC_SC -> EXTINT |= (1<<0);              //Bajamos bandera de EINT0

LPC_SC -> EXTINT |= (15<<0);    //Bajamos bandera de las 4 interrupciones ext.
```

Ejemplo de Interrupción Externa:

1. Buscamos en tabla que pin es Interrupción externa:

EINT0 Pin 10 de puerto 2, PINSEL4 bits 20 y 21, PINMODE4 bits 20 y 21

2. Configuramos en PINSEL esa función del pin:

```
LPC_PINCON -> PINSEL4 |= (1<<20);

LPC_PINCON -> PINSEL4 &= ~(1<<21);
```

3. Configuramos las resistencias de pull up con PINMODE:

```
LPC_PINCON -> PINMODE4 &= ~(3<<20);
```

4. Configuramos la interrupción por nivel bajo:

```
LPC_SC -> EXTMODE &= ~(1<<0);

LPC_SC -> EXTPOLAR &= ~(1<<0);
```

5. Seteamos prioridad 0 (máxima de interrupción):

```
NVIC_SetPriority(EINT0_IRQn, 0);
```

6. Habilitamos la interrupcion:

```
NVIC_EnableIRQ(EINT0_IRQn);
```

7. Escribimos un Handler para que haga el código que queremos:

```
void EINT0_IRQHandler(void)
{
……
LPC_SC -> EXTINT |= (1<<0);     //bajamos la bandera
}
```

Código completo:

```
#ifdef __USE_CMSIS                                  //Incluimos librerias
#include "LPC17xx.h"
#endif
void EINT0_IRQHandler(void);                        //Declaramos función de handler

int main(void)
{
LPC_PINCON -> PINSEL4 |= (1<<20);                   //Configuramos pin como EINT0
LPC_PINCON -> PINSEL4 &= ~(1<<21);
LPC_PINCON -> PINMODE4 &= ~(3<<20);                 //Pull up
LPC_SC -> EXTMODE &= ~(1<<0);                       //Int por nivel
LPC_SC -> EXTPOLAR &= ~(1<<0);                      //Int por nivel bajo
NVIC_SetPriority(EINT0_IRQn, 0);                    //Máxima prioridad para la int
NVIC_EnableIRQ(EINT0_IRQn);                         //Habilita interrupción

while (1)
{                                                   //Esperamos interrupciones
}
return 0;
}

void EINT0_IRQHandler(void)                         //Atendemos interrupcion externa
{
……
LPC_SC -> EXTINT |= (1<<0);     //bajamos la bandera
}
```

TIMER

Tenemos 4 TIMERs 0,1,2 y 3. De 32 bits con prescaler de 32 bits cada uno. Son timers externos, es decir son periféricos. Se permite comparar su valor con registros match, 4 match para cada timer y guardar su valor con registros capture.

Se puede configurar para que cuando los match registers MR0/1/2/3 coincidan con el valor del contador: genere una interrupción y/o reinicie el timer.

El clock por default de todos los periféricos es ¼ de la frecuencia del sistema. Si el sistema trabaja a 100MHz, el timer utiliza un clock de 25MHz.

Los registros que se usan con el timer son:

- PCONP (se accede con LPC_SC -> PCONP), este registro indica a que periféricos se energiza. Los bits 1,2,22 y 23 corresponden a los timer 0,1,2 y 3 respectivamente.
- CTCR (se accede con LPC_TIM0 -> CTCR), este registro configura el timer como timer o como contador de eventos, para utilizarlo como timer se coloca 00 en sus 2 bits menos significativos
- TCR (se accede con LPC_TIM0 -> TCR), este registro habilita (1) o deshabilita (0) el conteo del timer, con un 1 o un 0 respectivamente en el bit menos significativo.
- TC (se accede con LPC_TIM0 -> TC), es el contador, se puede escribir un 0 para resetearlo “manualmente”.
- PR (se accede con LPC_TIM0 -> PR), es el valor en el cual el prescaler resetea y aumenta el contador TC en 1.
- PC (se accede con LPC_TIM0 -> PC), es el contador del prescaler, se puede escribir un 0 para resetearlo.
- CR0,CR1 (se accede con LPC_TIM0 -> CR0), son registros de captura, cuando un evento de captura ocurre, configurado en el CCR, se guarda el valor de TC ahí.
- MR0,MR1,MR2,MR3 (se accede con LPC_TIM0 -> MR0), son los registros de match, comparan su valor con el de TC y cuando coinciden pueden interrumpir o reiniciar el timer según se configure el MCR.
- MCR (se accede con LPC_TIM0 -> MCR) para el MR0 en el bit 0 indica si interrumpe (1) o no (0), en el bit 1 si reinicia el TC (1) o no (0), para el bit 2 si reinicia y detiene el contador (1) o no (0) estos 3 bits se repiten en 3,5 y 6 para MR1; 7,8 y 9 para MR2 y 10,11 y 12 para MR3.

- IR (se accede con LPC_TIM0 -> IR) este registro muestra las banderas de interrupción del timer, del bit 0 al 3 se encuentran las banderas de los Match register del 0 al 3 respectivamente, en 4 y 5 de los capture channels 0 y 1 respectivamente.
  Las banderas se limpian escribiendo un 1 en el mismo bit (por más que ya estén en 1 "acknowledgement")
- CCR (se accede con LPC_TIM0 -> CCR) este registro que flancos en las entradas de capturas se usan para cargar una captura del timer y si interrumpe o no cuando se realiza una captura. Para CR0, el bit 0 indica si se realiza una captura en un flanco ascendente (1) o no se realiza en ese flanco (0). El bit 1 indica si se realiza una captura en flanco descendente (1) o no se realiza en ese flanco (0). El bit 2 indica si se produce una interrupción cada vez que se produce un evento de captura (1) o no (0). Se destaca que se puede configurar para interrumpir por ambos flancos. La configuración se repite en bits 3, 4 y 5 para CR1.

En las capturas se usan Pin 26 y 27 de puerto 1 para CR0 y CR1 en TIMER0, buscar el resto en las tablas.

Se recuerda de Interrupciones que para habilitar las interrupciones por Timer, por mas que se configuren los registros del timer, se deberá usar la función:

NVIC_EnableIRQ(TIMER0_IRQn);

Y se deberá utilizar un handler:

void TIMER0_IRQHandler (void)

En el cual se pregunte porque interrumpio (ver en las banderas si interrumpio por MR0, MR1, etc..) y limpiar la bandera.

Ejemplo de Timer0 para realizar en el pin 22 de P0 (LED) una onda PWM de 15ms de Alto y un período total de 50ms. Para eso hago MR0 el período total y MR1 el tiempo en alto. MR0 interrumpe y reinicia el contador, MR1 solo interrumpe.

1º) Configuro un pin como salida usando PINSEL y FIODIR:

```
LPC_PINCON->PINSEL1 &= ~(3<<12);

LPC_GPIO0-> FIODIR |= (1<<22);
```

2º) Configuro Timer0 como timer y lo detengo:

```
LPC_TIM0 -> CTCR = 0;

LPC_TIM0->TCR = 0;
```

3º)Configuro los Match Registers:

```
LPC_TIM0 -> MR0 = 50;

LPC_TIM0 -> MR1 = 15;

LPC_TIM0 -> MCR = 11;
```

4º)Seteo el prescaler y reinicio contadores (hace tick cada 1/25MHz = 40ns):

```
LPC_TIM0 -> PR = 25000;

LPC_TIM0 -> PC = 0;

LPC_TIM0 -> TC = 0;
```

5º) Habilito interrupción por Timer y habilito el conteo:

```
NVIC_EnableIRQ(TIMER0_IRQn);

LPC_TIM0 -> TCR = 1;
```

6º) Escribo un handler de la interrupción que pregunte por que match register interrumpió, limpie la bandera y cambie la salida PWM:

```
 void TIMER0_IRQHandler (void)

{

if (LPC_TIM0 -> IR &(1<<0))

        {

        LPC_GPIO0 -> FIOSET |= (1<<22);

        LPC_TIM0 ->IR |=(1<<0);

        }

If (LPC_TIM0 -> IR &(1<<1))

        {

        LPC_GPIO0 -> FIOCLR |= (1<<22);

        LPC_TIM0 ->IR |=(1<<1);

        }

}
```

Código completo probado en placa:

```
#ifdef __USE_CMSIS                          //Incluyo librerias
#include "LPC17xx.h"
#endif
void TIMER0_IRQHandler (void);              //Declaro funcion handler del timer
int main(void)
{
LPC_PINCON->PINSEL1 &= ~(3<<12);            //Pin 22 de puerto 0 como IO
LPC_GPIO0-> FIODIR |= (1<<22);              //Pin 22 de puerto 0 como salida
LPC_TIM0 -> CTCR = 0;                       //Timer0 como timer
LPC_TIM0->TCR = 0;                          //Timer0 deshabilitado
LPC_TIM0 -> MR0 = 50;                       //Cargo valores en MatchReg0 y 1
LPC_TIM0 -> MR1 = 15;
LPC_TIM0 -> MCR = 11;                       // b'1011' en MCR, interrumpe (MR0,MR1) y reset (MR0)
LPC_TIM0 -> PR = 25000;                     //Prescaler en 25000, el TC aumenta cada 1ms
LPC_TIM0 -> PC = 0;                         //Pongo en 0 contador de prescaler.
LPC_TIM0 -> TC = 0;                         //Pongo en 0 contador de timer.
NVIC_EnableIRQ(TIMER0_IRQn);                //Habilito interrupción por timer
LPC_TIM0 -> TCR = 1;                        //Timer0 habilitado
while (1)                                   //Esperamos interrupcion
{
}
}
void TIMER0_IRQHandler (void)               //Función que atiende la interrupción por timer
{
if (LPC_TIM0 -> IR &(1<<0))                 //Interrumpio por MR0?
        {
        LPC_GPIO0 -> FIOSET |= (1<<22);     //Pongo salida en Alto
        LPC_TIM0 ->IR |=(1<<0);             //Bajo bandera de MR0
        }
if (LPC_TIM0 -> IR &(1<<1))                 //Interrumpio por MR1?
        {
        LPC_GPIO0 -> FIOCLR |= (1<<22);     //Pongo salida en Bajo
        LPC_TIM0 ->IR |=(1<<1);             //Bajo bandera de MR1
        }
}
```

ADC para Boris

La placa LPC cuenta con un ADC de 12 bits.

Para utilizar el ADC lo primero que hay que hacer es energizarlo. El bit 12 de PCONP se encarga de eso. En reset el ADC queda desenergizado:

```
LPC_SC -> PCONP |= (1<<12);              //energiza ADC
```

Los registros para usar el ADC son: ADCR (control register), ADGDR (contiene el bit DONE y el ultimo resultado), ADINTEN (configura interrupción), ADDR0/1/2/3/4/5/6/7 (contienen los resultados de la ultima conversión del canal 0,1,2,3,4,5,6 y 7 respectivamente. ADSTAT (donde esta la bandera de interrupt).

Se utilizan con LPC_ADC.

El registro ADCR es el más importante:

Los bits de 0 a 7 indican que canal se usa para hacer la conversión, solo uno de estos bits debe estar en 1.

Los bits de 8 a 15 contienen un numero, el clock de periféricos (o el seleccionado para el ADC en caso de usar otro), dividido por este numero más uno dan el clock de conversión, que deberá ser 13Mhz o menos, ya que es la velocidad máxima a la que el ADC puede trabajar.

El bit 16 enciende el modo BURST en el cual el ADC realiza varias conversiones hasta 200kHz.

Bits 17 a 20 no se usan.

Bit 21 habilita o no el ADC (no es el que inicia conversión)

Bits 22 y 23 no se usan.

Bits 24 a 26 inician conversión sino esta en el modo BURST. (000 no inicia, 001 inicia conversión, 010 Inicia conversión cuando se produce una int externa en EINT0, 011 incia conversión cuando se realiza una captura en el timer0, 100 inicia conversión cuando timer0 coincide con match register 1, 101 inicia conversión cuando timer0 coincide con match register 3, 110 inicia conversión cuando timer1 coincide con match register 0, 111 inicia conversión cuando timer1 coindice con match register 1).

Bit 27 indica si la conversión se inicia en el flanco de subida de las interrupciones (0) o en el de bajada (1).

Bits 28 a 31 no se usan.

Los bits 4 a 15 de ADGDR contienen el ultimo resultado. Y el bit 31 indica si terminó la conversión (con 1) .

Los bits de 0 a 7 de ADINTEN indican si la conversión del canal 0,1,2,3,4,5,6 o 7 respectivamente interrumpen (1) o no (0).

Los bits 4 a 15 de ADDR0/1/2/3/4/5/6/7 contienen el resultado de la última conversión en el respectivo canal. El bit 31 el DONE para ese canal.

Para utilizar las interrupciones no hay que olvidarse de habilitarlas en el NVIC:

```
NVIC_EnableIRQ(ADC_IRQn);
```

Su respectivo handler:

void ADC_IRQHandler (void);

Con leer el resultado de la conversión es suficiente para que se baje la bandera de la interrupción.

Como el resultado de conversión se encuentra desde el bit 4, para utilizarlo hay que correrlo 4 posiciones y enmascarar solo esos 12 bits:

Ejemplo:

lectura = (LPC_ADC->ADDR0 >> 4) & 0xfff; //Siendo lectura un int.

No hay que olvidar configurar pines como entrada de ADC en PINSEL

Ejemplo de ADC con Timer cada 250ms, en el handler de la interrupción por Timer se inicia la conversión:

```
#ifdef __USE_CMSIS                          //Librerias
#include "LPC17xx.h"
#endif

int lectura = 0;                            //defino variables y funciones

void TIMER1_IRQHandler (void);
void ADC_IRQHandler (void);
int main(void) {

        LPC_PINCON -> PINSEL1 |= (1<<14);    //pin como canal 0 del ADC
        LPC_PINCON -> PINSEL1 &= ~(1<<15);
        LPC_SC -> PCONP |= (1<<12);   //Energizo ADC
        LPC_ADC -> ADCR |= (1<<0);    //Elijo canal 0
        LPC_ADC -> ADCR |= (1<<9);    //Div 2 (clk perif = 25Mhz, clk ADC = 12,5MHz
        LPC_ADC -> ADCR |= (1 << 21); //Habilito ADC
        LPC_ADC -> ADINTEN = 1;                     //ADC interrumpe
        LPC_TIM1 -> CTCR = 0;               //Timer1 como timer
        LPC_TIM1->TCR = 0;                          //Timer1 deshabilitado
        LPC_TIM1 -> MR0 = 250;              //Cargo valores en MatchReg0
        LPC_TIM1 -> MCR = 3;          // b'011' en MCR, interrumpe (MR0) y reset (MR0)
        LPC_TIM1 -> PR = 25000;             //Prescaler en 25000, el TC aumenta cada 1ms
        LPC_TIM1 -> PC = 0;                         //Pongo en 0 contador de prescaler.
        LPC_TIM1 -> TC = 0;                         //Pongo en 0 contador de timer.
        NVIC_EnableIRQ(TIMER1_IRQn);  //Habilito interrupción por timer
        NVIC_EnableIRQ(ADC_IRQn);     //Habilito interrupción de ADC
        LPC_TIM1 -> TCR = 1;                //Timer1 habilitado

        while (1);
        return 0 ;

}

void TIMER1_IRQHandler (void) //Interrumpe timer1
        {
                LPC_ADC -> ADCR |=(1<<24);          //Comienzo de conversion
                LPC_TIM1 ->IR |=(1<<0);                     //Bajo bandera timer1
        }

void ADC_IRQHandler (void)
{
lectura = (LPC_ADC->ADDR0 >> 4) & 0xfff;            //Paso los bits leidos a lectura.
}
```

Tablas PINSEL

Table 79. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

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

Table 80. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL	00
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO	00
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI	00
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1	00
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1	00
11:10	P0.21[1]	GPIO Port 0.21	RI1	Reserved	RD1	00
13:12	P0.22	GPIO Port 0.22	RTS1	Reserved	TD1	00
15:14	P0.23[1]	GPIO Port 0.23	AD0.0	I2SRX_CLK	CAP3.0	00
17:16	P0.24[1]	GPIO Port 0.24	AD0.1	I2SRX_WS	CAP3.1	00
19:18	P0.25	GPIO Port 0.25	AD0.2	I2SRX_SDA	TXD3	00
21:20	P0.26	GPIO Port 0.26	AD0.3	AOUT	RXD3	00
23:22	P0.27[1][2]	GPIO Port 0.27	SDA0	USB_SDA	Reserved	00
25:24	P0.28[1][2]	GPIO Port 0.28	SCL0	USB_SCL	Reserved	00
27:26	P0.29	GPIO Port 0.29	USB_D+	Reserved	Reserved	00
29:28	P0.30	GPIO Port 0.30	USB_D–	Reserved	Reserved	00
31:30	-	Reserved	Reserved	Reserved	Reserved	00

Table 81. Pin function select register 2 (PINSEL2 - address 0x4002 C008) bit description

PINSEL2	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1.0	GPIO Port 1.0	ENET_TXD0	Reserved	Reserved	00
3:2	P1.1	GPIO Port 1.1	ENET_TXD1	Reserved	Reserved	00
7:4	-	Reserved	Reserved	Reserved	Reserved	0
9:8	P1.4	GPIO Port 1.4	ENET_TX_EN	Reserved	Reserved	00
15:10	-	Reserved	Reserved	Reserved	Reserved	0
17:16	P1.8	GPIO Port 1.8	ENET_CRS	Reserved	Reserved	00
19:18	P1.9	GPIO Port 1.9	ENET_RXD0	Reserved	Reserved	00
21:20	P1.10	GPIO Port 1.10	ENET_RXD1	Reserved	Reserved	00
27:22	-	Reserved	Reserved	Reserved	Reserved	0
29:28	P1.14	GPIO Port 1.14	ENET_RX_ER	Reserved	Reserved	00
31:30	P1.15	GPIO Port 1.15	ENET_REF_CLK	Reserved	Reserved	00

Table 82. Pin function select register 3 (PINSEL3 - address 0x4002 C00C) bit description

PINSEL3	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1.16[1]	GPIO Port 1.16	ENET_MDC	Reserved	Reserved	00
3:2	P1.17[1]	GPIO Port 1.17	ENET_MDIO	Reserved	Reserved	00
5:4	P1.18	GPIO Port 1.18	USB_UP_LED	PWM1.1	CAP1.0	00
7:6	P1.19	GPIO Port 1.19	MCOA0	$\overline{\text{USB\_PPWR}}$	CAP1.1	00
9:8	P1.20	GPIO Port 1.20	MCI0	PWM1.2	SCK0	00
11:10	P1.21[1]	GPIO Port 1.21	$\overline{\text{MCABORT}}$	PWM1.3	SSEL0	00
13:12	P1.22	GPIO Port 1.22	MCOB0	USB_PWRD	MAT1.0	00
15:14	P1.23	GPIO Port 1.23	MCI1	PWM1.4	MISO0	00
17:16	P1.24	GPIO Port 1.24	MCI2	PWM1.5	MOSI0	00
19:18	P1.25	GPIO Port 1.25	MCOA1	Reserved	MAT1.1	00
21:20	P1.26	GPIO Port 1.26	MCOB1	PWM1.6	CAP0.0	00
23:22	P1.27[1]	GPIO Port 1.27	CLKOUT	$\overline{\text{USB\_OVRCR}}$	CAP0.1	00
25:24	P1.28	GPIO Port 1.28	MCOA2	PCAP1.0	MAT0.0	00
27:26	P1.29	GPIO Port 1.29	MCOB2	PCAP1.1	MAT0.1	00
29:28	P1.30	GPIO Port 1.30	Reserved	V_{BUS}	AD0.4	00
31:30	P1.31	GPIO Port 1.31	Reserved	SCK1	AD0.5	00

Table 83. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

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

Table 84. Pin function select register 7 (PINSEL7 - address 0x4002 C01C) bit description

PINSEL7	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
17:0	-	Reserved	Reserved	Reserved	Reserved	0
19:18	P3.25[1]	GPIO Port 3.25	Reserved	MAT0.0	PWM1.2	00
21:20	P3.26[1]	GPIO Port 3.26	STCLK	MAT0.1	PWM1.3	00
31:22	-	Reserved	Reserved	Reserved	Reserved	0

Table 85. Pin function select register 9 (PINSEL9 - address 0x4002 C024) bit description

PINSEL9	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
23:0	-	Reserved	Reserved	Reserved	Reserved	00
25:24	P4.28	GPIO Port 4.28	RX_MCLK	MAT2.0	TXD3	00
27:26	P4.29	GPIO Port 4.29	TX_MCLK	MAT2.1	RXD3	00
31:28	-	Reserved	Reserved	Reserved	Reserved	00

Table 86. Pin function select register 10 (PINSEL10 - address 0x4002 C028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved. Software should not write 1 to these bits.	NA
3	GPIO/TRACE		TPIU interface pins control.	0
		0	TPIU interface is disabled.	
		1	TPIU interface is enabled. TPIU signals are available on the pins hosting them regardless of the PINSEL4 content.	
31:4	-	-	Reserved. Software should not write 1 to these bits.	NA

Tablas PINMODE

Table 87. Pin Mode select register 0 (PINMODE0 - address 0x4002 C040) bit description

PINMODE0	Symbol	Value	Description	Reset value
1:0	P0.00MODE		Port 0 pin 0 on-chip pull-up/down resistor control.	00
		00	P0.0 pin has a pull-up resistor enabled.	
		01	P0.0 pin has repeater mode enabled.	
		10	P0.0 pin has neither pull-up nor pull-down.	
		11	P0.0 has a pull-down resistor enabled.	
3:2	P0.01MODE		Port 0 pin 1 control, see P0.00MODE.	00
5:4	P0.02MODE		Port 0 pin 2 control, see P0.00MODE.	00
7:6	P0.03MODE		Port 0 pin 3 control, see P0.00MODE.	00
9:8	P0.04MODE[1]		Port 0 pin 4 control, see P0.00MODE.	00
11:10	P0.05MODE[1]		Port 0 pin 5 control, see P0.00MODE.	00
13:12	P0.06MODE		Port 0 pin 6 control, see P0.00MODE.	00
15:14	P0.07MODE		Port 0 pin 7 control, see P0.00MODE.	00
17:16	P0.08MODE		Port 0 pin 8 control, see P0.00MODE.	00
19:18	P0.09MODE		Port 0 pin 9control, see P0.00MODE.	00
21:20	P0.10MODE		Port 0 pin 10 control, see P0.00MODE.	00
23:22	P0.11MODE		Port 0 pin 11 control, see P0.00MODE.	00
29:24	-		Reserved.	NA
31:30	P0.15MODE		Port 0 pin 15 control, see P0.00MODE.	00

Table 88. Pin Mode select register 1 (PINMODE1 - address 0x4002 C044) bit description

PINMODE1	Symbol	Description	Reset value
1:0	P0.16MODE	pin 16 control, see P0.00MODE.	00
3:2	P0.17MODE	pin 17 control, see P0.00MODE.	00
5:4	P0.18MODE	pin 18 control, see P0.00MODE.	00
7:6	P0.19MODE[1]	pin 19 control, see P0.00MODE.	00
9:8	P0.20MODE[1]	pin 20control, see P0.00MODE.	00
11:10	P0.21MODE[1]	pin 21 control, see P0.00MODE.	00
13:12	P0.22MODE	pin 22 control, see P0.00MODE.	00
15:14	P0.23MODE[1]	pin 23 control, see P0.00MODE.	00
17:16	P0.24MODE[1]	pin 24 control, see P0.00MODE.	00
19:18	P0.25MODE	pin 25 control, see P0.00MODE.	00
21:20	P0.26MODE	pin 26 control, see P0.00MODE.	00
29:22	-	Reserved. [2]	NA
31:30	-	Reserved.	NA

PORT0

Table 89. Pin Mode select register 2 (PINMODE2 - address 0x4002 C048) bit description

PINMODE2	Symbol	Description	Reset value
1:0	P1.00MODE	Port 1 pin 0 control, see P0.00MODE.	00
3:2	P1.01MODE	Port 1 pin 1 control, see P0.00MODE.	00
7:4	-	Reserved.	NA
9:8	P1.04MODE	Port 1 pin 4 control, see P0.00MODE.	00
15:10	-	Reserved.	NA
17:16	P1.08MODE	Port 1 pin 8 control, see P0.00MODE.	00
19:18	P1.09MODE	Port 1 pin 9 control, see P0.00MODE.	00
21:20	P1.10MODE	Port 1 pin 10 control, see P0.00MODE.	00
27:22	-	Reserved.	NA
29:28	P1.14MODE	Port 1 pin 14 control, see P0.00MODE.	00
31:30	P1.15MODE	Port 1 pin 15 control, see P0.00MODE.	00

Table 90. Pin Mode select register 3 (PINMODE3 - address 0x4002 C04C) bit description

PINMODE3	Symbol	Description	Reset value
1:0	P1.16MODE[1]	Port 1 pin 16 control, see P0.00MODE.	00
3:2	P1.17MODE[1]	Port 1 pin 17 control, see P0.00MODE.	00
5:4	P1.18MODE	Port 1 pin 18 control, see P0.00MODE.	00
7:6	P1.19MODE	Port 1 pin 19 control, see P0.00MODE.	00
9:8	P1.20MODE	Port 1 pin 20 control, see P0.00MODE.	00
11:10	P1.21MODE[1]	Port 1 pin 21 control, see P0.00MODE.	00
13:12	P1.22MODE	Port 1 pin 22 control, see P0.00MODE.	00
15:14	P1.23MODE	Port 1 pin 23 control, see P0.00MODE.	00
17:16	P1.24MODE	Port 1 pin 24 control, see P0.00MODE.	00
19:18	P1.25MODE	Port 1 pin 25 control, see P0.00MODE.	00
21:20	P1.26MODE	Port 1 pin 26 control, see P0.00MODE.	00
23:22	P1.27MODE[1]	Port 1 pin 27 control, see P0.00MODE.	00
25:24	P1.28MODE	Port 1 pin 28 control, see P0.00MODE.	00
27:26	P1.29MODE	Port 1 pin 29 control, see P0.00MODE.	00
29:28	P1.30MODE	Port 1 pin 30 control, see P0.00MODE.	00
31:30	P1.31MODE	Port 1 pin 31 control, see P0.00MODE.	00

Table 91. Pin Mode select register 4 (PINMODE4 - address 0x4002 C050) bit description

PINMODE4	Symbol	Description	Reset value
1:0	P2.00MODE	Port 2 pin 0 control, see P0.00MODE.	00
3:2	P2.01MODE	Port 2 pin 1 control, see P0.00MODE.	00
5:4	P2.02MODE	Port 2 pin 2 control, see P0.00MODE.	00
7:6	P2.03MODE	Port 2 pin 3 control, see P0.00MODE.	00
9:8	P2.04MODE	Port 2 pin 4 control, see P0.00MODE.	00
11:10	P2.05MODE	Port 2 pin 5 control, see P0.00MODE.	00
13:12	P2.06MODE	Port 2 pin 6 control, see P0.00MODE.	00
15:14	P2.07MODE	Port 2 pin 7 control, see P0.00MODE.	00
17:16	P2.08MODE	Port 2 pin 8 control, see P0.00MODE.	00
19:18	P2.09MODE	Port 2 pin 9 control, see P0.00MODE.	00
21:20	P2.10MODE	Port 2 pin 10 control, see P0.00MODE.	00
23:22	P2.11MODE[1]	Port 2 pin 11 control, see P0.00MODE.	00
25:24	P2.12MODE[1]	Port 2 pin 12 control, see P0.00MODE.	00
27:26	P2.13MODE[1]	Port 2 pin 13 control, see P0.00MODE.	00
31:28	-	Reserved.	NA

Table 92. Pin Mode select register 7 (PINMODE7 - address 0x4002 C05C) bit description

PINMODE7	Symbol	Description	Reset value
17:0	-	Reserved	NA
19:18	P3.25MODE[1]	Port 3 pin 25 control, see P0.00MODE.	00
21:20	P3.26MODE[1]	Port 3 pin 26 control, see P0.00MODE.	00
31:22	-	Reserved.	NA

Table 93. Pin Mode select register 9 (PINMODE9 - address 0x4002 C064) bit description

PINMODE9	Symbol	Description	Reset value
23:0	-	Reserved.	NA
25:24	P4.28MODE	Port 4 pin 28 control, see P0.00MODE.	00
27:26	P4.29MODE	Port 4 pin 29 control, see P0.00MODE.	00
31:28	-	Reserved.	NA

TABLAS ADC

Table 532: A/D Control Register (AD0CR - address 0x4003 4000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the AD0.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK_ADC0) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 13 MHz. Typically, software should program the smallest value in this field that yields a clock of 13 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	1	The AD converter does repeated conversions at up to 200 kHz, scanning (if necessary) through the pins selected by bits set to ones in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Remark:** START bits must be 000 when BURST = 1 or conversions will not start.	0
		0	Conversions are software controlled and require 65 clocks.	
20:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The A/D converter is operational.	0
		0	The A/D converter is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on the P2.10 / EINT0 / NMI pin.	
		011	Start conversion when the edge selected by bit 27 occurs on the P1.27 / CLKOUT / USB_OVRCRn / CAP0.1 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 535: A/D Data Registers (AD0DR0 to AD0DR7 - 0x4003 4010 to 0x4003 402C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0x3FF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	NA
29:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits.This bit is cleared by reading this register.	
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	NA

Table 534: A/D Status register (AD0INTEN - address 0x4003 400C) bit description

Bit	Symbol	Value	Description	Reset value
0	ADINTEN0	0	Completion of a conversion on ADC channel 0 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 0 will generate an interrupt.	
1	ADINTEN1	0	Completion of a conversion on ADC channel 1 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 1 will generate an interrupt.	
2	ADINTEN2	0	Completion of a conversion on ADC channel 2 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 2 will generate an interrupt.	
3	ADINTEN3	0	Completion of a conversion on ADC channel 3 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 3 will generate an interrupt.	
4	ADINTEN4	0	Completion of a conversion on ADC channel 4 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 4 will generate an interrupt.	
5	ADINTEN5	0	Completion of a conversion on ADC channel 5 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 5 will generate an interrupt.	
6	ADINTEN6	0	Completion of a conversion on ADC channel 6 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 6 will generate an interrupt.	
7	ADINTEN7	0	Completion of a conversion on ADC channel 7 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 7 will generate an interrupt.	
8	ADGINTEN	0	Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts.	1
		1	Only the global DONE flag in ADDR is enabled to generate an interrupt.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

TABLAS TIMER

Table 426. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4000 4004 T1TCR - 0x4000 8004 T2TCR - 0x4009 0004 T3TCR - 0x4009 4004
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4000 4008 T1TC - 0x4000 8008 T2TC - 0x4009 0008 T3TC - 0x4009 4008
PR	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4000 400C T1PR - 0x4000 800C T2PR - 0x4009 000C T3PR - 0x4009 400C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	T0PC - 0x4000 4010 T1PC - 0x4000 8010 T2PC - 0x4009 0010 T3PC - 0x4009 4010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	T0MCR - 0x4000 4014 T1MCR - 0x4000 8014 T2MCR - 0x4009 0014 T3MCR - 0x4009 4014
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	T0MR0 - 0x4000 4018 T1MR0 - 0x4000 8018 T2MR0 - 0x4009 0018 T3MR0 - 0x4009 4018
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0x4000 401C T1MR1 - 0x4000 801C T2MR1 - 0x4009 001C T3MR1 - 0x4009 401C
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0x4000 4020 T1MR2 - 0x4000 8020 T2MR2 - 0x4009 0020 T3MR2 - 0x4009 4020
MR3	Match Register 3. See MR0 description.	R/W	0	T0MR3 - 0x4000 4024 T1MR3 - 0x4000 8024 T2MR3 - 0x4009 0024 T3MR3 - 0x4009 4024
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	T0CCR - 0x4000 4028 T1CCR - 0x4000 8028 T2CCR - 0x4009 0028

Table 428. Timer Control Register (TCR, TIMERn: TnTCR - addresses 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When 1, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 430. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		0	Feature disabled.	
6	MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		0	This interrupt is disabled	
7	MR2R	1	Reset on MR2: the TC will be reset if MR2 matches it.	0
		0	Feature disabled.	
8	MR2S	1	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		0	Feature disabled.	
9	MR3I	1	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		0	This interrupt is disabled	
10	MR3R	1	Reset on MR3: the TC will be reset if MR3 matches it.	0
		0	Feature disabled.	
11	MR3S	1	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		0	Feature disabled.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 429. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment the Timer's Prescale Counter (PC), or clear the PC and increment the Timer Counter (TC).	00
		00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. The Prescale Counter is incremented on every rising PCLK edge.	
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	

TABLAS INTERRUPCION

Table 10. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description

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

Table 11. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description

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

Table 12. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description

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

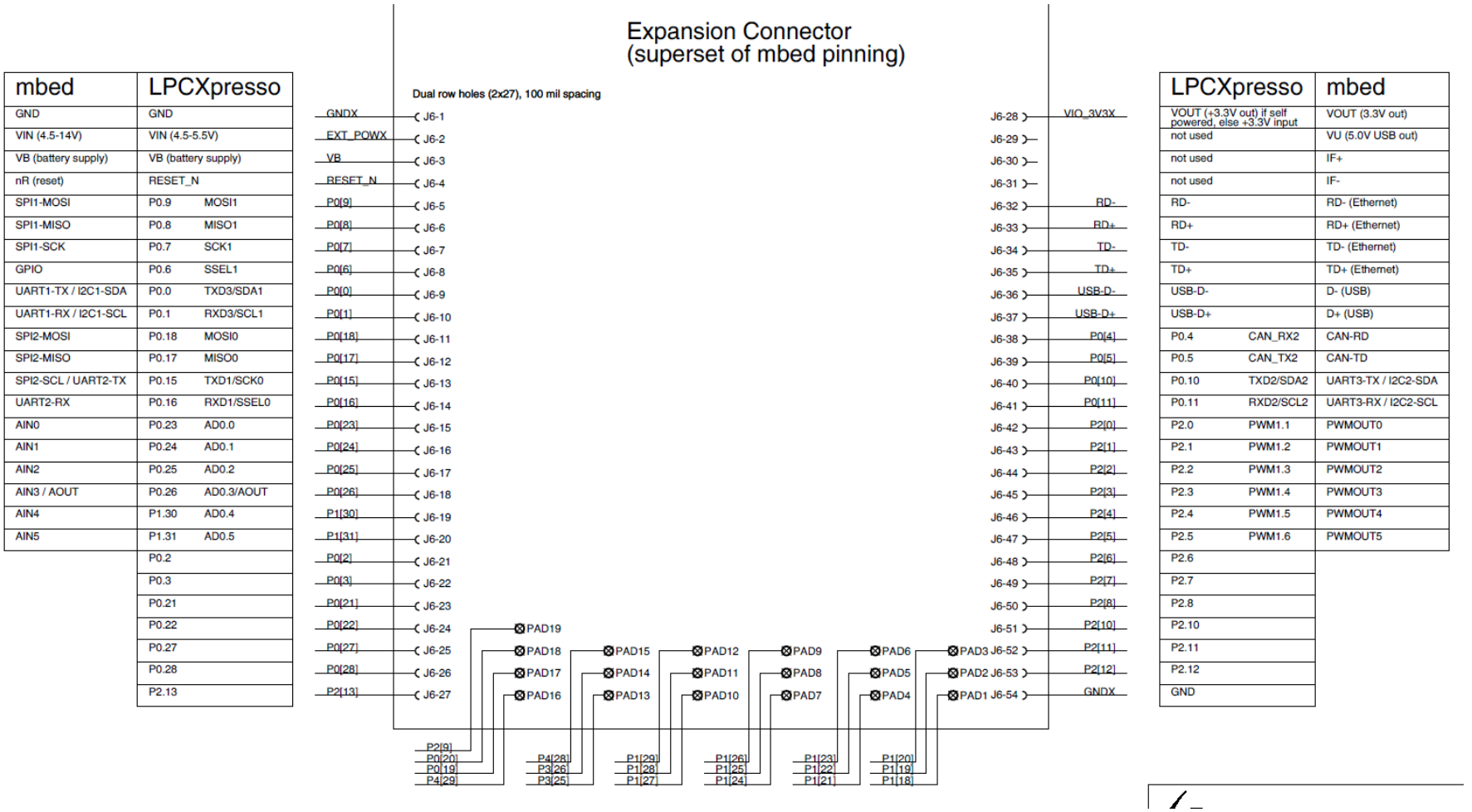
TABLAS PCONP

Table 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved.	NA
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	-	Reserved.	NA
6	PCPWM1	PWM1 power/clock control bit.	1
7	PCI2C0	The I^2C0 interface power/clock control bit.	1
8	PCSPI	The SPI interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSSP1	The SSP 1 interface power/clock control bit.	1
11	-	Reserved.	NA
12	PCADC	A/D converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	0
13	PCCAN1	CAN Controller 1 power/clock control bit.	0
14	PCCAN2	CAN Controller 2 power/clock control bit.	0
15	-	Reserved.	NA
16	PCRIT	Repetitive Interrupt Timer power/clock control bit.	0
17	PCMCPWM	Motor Control PWM	0
18	PCQEI	Quadrature Encoder Interface power/clock control bit.	0
19	PCI2C1	The I^2C1 interface power/clock control bit.	1
20	-	Reserved.	NA
21	PCSSP0	The SSP0 interface power/clock control bit.	1
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I^2C interface 2 power/clock control bit.	1
27	PCI2S	I^2S interface power/clock control bit.	0
28	-	Reserved.	NA
29	PCGPDMA	GPDMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

LPC1769 PINOUT

mbed	LPCXpresso	
GND	GND	
VIN (4.5-14V)	VIN (4.5-5.5V)	
VB (battery supply)	VB (battery supply)	
nR (reset)	RESET_N	
SPI1-MOSI	P0.9	MOSI1
SPI1-MISO	P0.8	MISO1
SPI1-SCK	P0.7	SCK1
GPIO	P0.6	SSEL1
UART1-TX / I2C1-SDA	P0.0	TXD3/SDA1
UART1-RX / I2C1-SCL	P0.1	RXD3/SCL1
SPI2-MOSI	P0.18	MOSI0
SPI2-MISO	P0.17	MISO0
SPI2-SCL / UART2-TX	P0.15	TXD1/SCK0
UART2-RX	P0.16	RXD1/SSEL0
AIN0	P0.23	AD0.0
AIN1	P0.24	AD0.1
AIN2	P0.25	AD0.2
AIN3 / AOUT	P0.26	AD0.3/AOUT
AIN4	P1.30	AD0.4
AIN5	P1.31	AD0.5
	P0.2	
	P0.3	
	P0.21	
	P0.22	
	P0.27	
	P0.28	
	P2.13	



LPCXpresso		mbed
VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
not used		VU (5.0V USB out)
not used		IF+
not used		IF-
RD-		RD- (Ethernet)
RD+		RD+ (Ethernet)
TD-		TD- (Ethernet)
TD+		TD+ (Ethernet)
USB-D-		D- (USB)
USB-D+		D+ (USB)
P0.4	CAN_RX2	CAN-RD
P0.5	CAN_TX2	CAN-TD
P0.10	TXD2/SDA2	UART3-TX / I2C2-SDA
P0.11	RXD2/SCL2	UART3-RX / I2C2-SCL
P2.0	PWM1.1	PWMOUT0
P2.1	PWM1.2	PWMOUT1
P2.2	PWM1.3	PWMOUT2
P2.3	PWM1.4	PWMOUT3
P2.4	PWM1.5	PWMOUT4
P2.5	PWM1.6	PWMOUT5
P2.6		
P2.7		
P2.8		
P2.10		
P2.11		
P2.12		
GND		

B-LABS

Lista de interrupciones para: IRQn_Type IRQn

```
/******  LPC17xx Specific Interrupt Numbers *******************************************************/
  WDT_IRQn                      = 0,        /*!< Watchdog Timer Interrupt                          */
  TIMER0_IRQn                   = 1,        /*!< Timer0 Interrupt                                  */
  TIMER1_IRQn                   = 2,        /*!< Timer1 Interrupt                                  */
  TIMER2_IRQn                   = 3,        /*!< Timer2 Interrupt                                  */
  TIMER3_IRQn                   = 4,        /*!< Timer3 Interrupt                                  */
  UART0_IRQn                    = 5,        /*!< UART0 Interrupt                                   */
  UART1_IRQn                    = 6,        /*!< UART1 Interrupt                                   */
  UART2_IRQn                    = 7,        /*!< UART2 Interrupt                                   */
  UART3_IRQn                    = 8,        /*!< UART3 Interrupt                                   */
  PWM1_IRQn                     = 9,        /*!< PWM1 Interrupt                                    */
  I2C0_IRQn                     = 10,       /*!< I2C0 Interrupt                                    */
  I2C1_IRQn                     = 11,       /*!< I2C1 Interrupt                                    */
  I2C2_IRQn                     = 12,       /*!< I2C2 Interrupt                                    */
  SPI_IRQn                      = 13,       /*!< SPI Interrupt                                     */
  SSP0_IRQn                     = 14,       /*!< SSP0 Interrupt                                    */
  SSP1_IRQn                     = 15,       /*!< SSP1 Interrupt                                    */
  PLL0_IRQn                     = 16,       /*!< PLL0 Lock (Main PLL) Interrupt                    */
  RTC_IRQn                      = 17,       /*!< Real Time Clock Interrupt                         */
  EINT0_IRQn                    = 18,       /*!< External Interrupt 0 Interrupt                    */
  EINT1_IRQn                    = 19,       /*!< External Interrupt 1 Interrupt                    */
  EINT2_IRQn                    = 20,       /*!< External Interrupt 2 Interrupt                    */
  EINT3_IRQn                    = 21,       /*!< External Interrupt 3 Interrupt                    */
  ADC_IRQn                      = 22,       /*!< A/D Converter Interrupt                           */
  BOD_IRQn                      = 23,       /*!< Brown-Out Detect Interrupt                        */
  USB_IRQn                      = 24,       /*!< USB Interrupt                                     */
  CAN_IRQn                      = 25,       /*!< CAN Interrupt                                     */
  DMA_IRQn                      = 26,       /*!< General Purpose DMA Interrupt                     */
  I2S_IRQn                      = 27,       /*!< I2S Interrupt                                     */
  ENET_IRQn                     = 28,       /*!< Ethernet Interrupt                                */
  RIT_IRQn                      = 29,       /*!< Repetitive Interrupt Timer Interrupt              */
  MCPWM_IRQn                    = 30,       /*!< Motor Control PWM Interrupt                       */
  QEI_IRQn                      = 31,       /*!< Quadrature Encoder Interface Interrupt            */
  PLL1_IRQn                     = 32,       /*!< PLL1 Lock (USB PLL) Interrupt                     */
  USBActivity_IRQn              = 33,       /* USB Activity interrupt                           */
  CANActivity_IRQn              = 34,       /* CAN Activity interrupt                           */
```

Lista de handlers para interrupciones:

```
//*****************************************************************************
//
// 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);
void BOD_IRQHandler(void) ALIAS(IntDefaultHandler);
void USB_IRQHandler(void) ALIAS(IntDefaultHandler);
void CAN_IRQHandler(void) ALIAS(IntDefaultHandler);
void DMA_IRQHandler(void) ALIAS(IntDefaultHandler);
void I2S_IRQHandler(void) ALIAS(IntDefaultHandler);
void ENET_IRQHandler(void) ALIAS(IntDefaultHandler);
void RIT_IRQHandler(void) ALIAS(IntDefaultHandler);
void MCPWM_IRQHandler(void) ALIAS(IntDefaultHandler);
void QEI_IRQHandler(void) ALIAS(IntDefaultHandler);
void PLL1_IRQHandler(void) ALIAS(IntDefaultHandler);
void USBActivity_IRQHandler(void) ALIAS(IntDefaultHandler);
void CANActivity_IRQHandler(void) ALIAS(IntDefaultHandler);
```