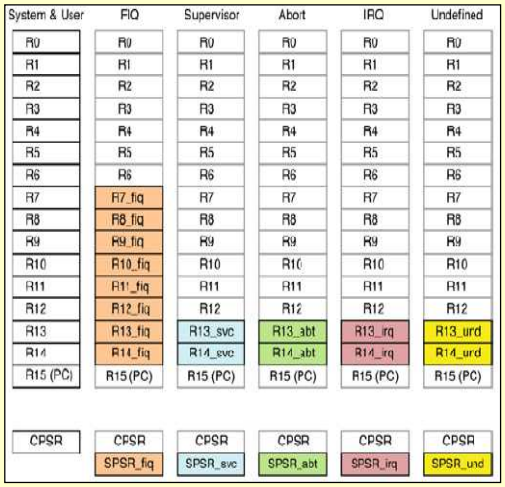
Repaso



La Familia LPC21xx

- Flash 0.18µm
  - Programación IAP/ISP.
  - Alimentación 1.8/3V y alimentación única (3V).
- Core 16/32 bits
  - Real-Time Debug.
  - Emulación y Traza embedded.
  - Controlador de Interrupciones Vectorizadas.
- Periféricos
  - ADC, CAN, Timer, I2C, SPI
  - USB 2.0, Ethernet MAC, Protocolo Wireless.
- Herramientas
  - ARM, Keil, Hitex, Ashling, IAR, Nohau, etc.

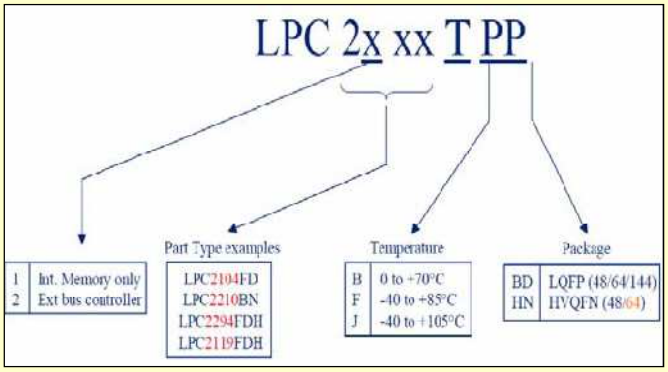
ARM7TDMI-S

- Basado en el core ARM
- **T**- Extensión de la arquitectura Thumb.
  - Set de Instrucciones ARM de 32 bits.
  - Set de Instrucciones Thumb de 16 bis.
  - Selección de modo mediante ejecución de una instrucción.
- **D**- Core con extensiones para depuración (Debug).
- **M**- Core con multiplicador mejorado.
- **I**- Core con Embedded ICE Macrocell.
- **S**- Totalmente sintetizable

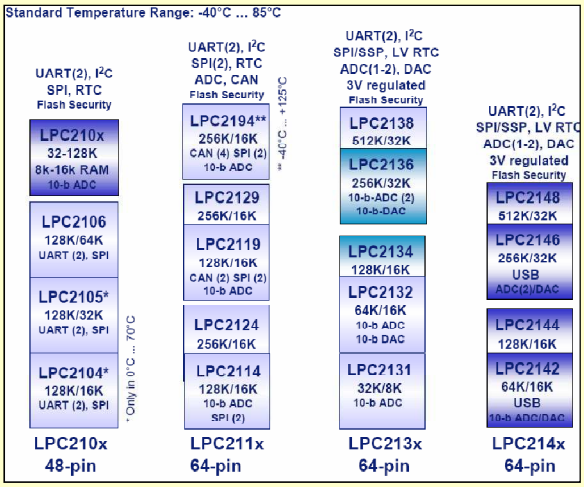
Los diseñadores desean…

- Fácil upgrade con capacidades de memoria de programa suficientes, y tiempos cortos de aprendizaje (ej. mismas herramientas).
- Soluciones en un chip, con numerosos periféricos:
  - On-Chip Flash.
  - Timers, PWM, UART, SPI, I2C, etc.
  - ADC, DAC, POR, etc.
- Prestaciones determinísticas (predecibles):
  - Sin caché.
  - Mismas prestaciones entre el código ejecutado en Flash y en RAM.
- Bajo consumo (<75µA en power down, <1mA/MHz en activo)
- Sistema de prioridad de interrupciones flexible.
- Prestaciones elevadas a precio de 8 bits.
- Buenas Herramientas de desarrollo y depuración (JTAG con traza).

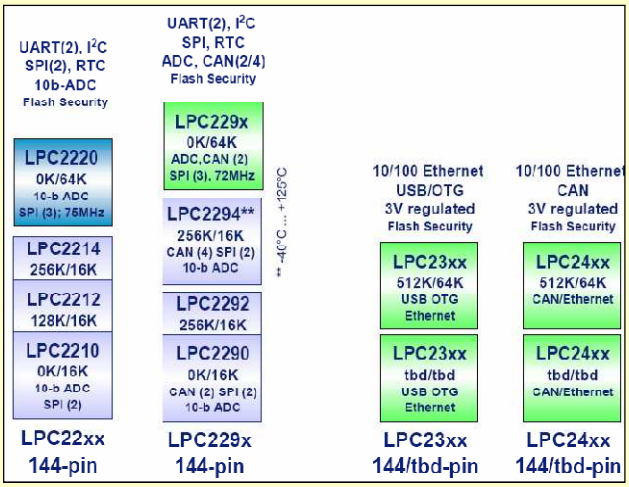
LPC2xxx - Identificación



La familia LPC2xxx



La familia LPC2xxx



La familia LPC2xxx

- ARM7 es un estandar en el diseño de SE de 32 bits.
  - Aumento de usuarios que migran de los 8-16bits a los 32 bits (>20%)
  - Precios muy competitivos.
- Flash
  - Primer AR7TDMI con proceso de fabricación de 0.18µm.
  - Capacidad de memoria de 32Kb a 512Kb.
  - Tiempo de acceso reducido (cercano a cero ciclos espera) debido al hardware interno MAM (Memory Accelerator Module).
- Amplia selección de periféricos internos.
- Versiones con reducido número de pines y pequeños encapsulados.

La familia LPC2xxx

- Capacidades entre 32Kb y 512Kb.
- Frecuencia de trabajo de 60 Mhz debido a los accesos de 128 bits.
- Bajo consumo (< 1mW/ MHz)
- Sistema de corrección de errores en circuito (ECC) para mejorar la fiabilidad.
- Tiempos de acceso de 1 ms por página (256 bytes): 4µs/byte.
- Boot-loader interno para In-System update.
- Hasta 100k ciclos de escritura (retención datos de 20 años) .
- Sistema de protección del código de programa.

Versiones con y sin bus externo

- Ancho de 32 bits beneficia el modo ARM.
  - Mejora de velocidad al trabajar con buses de 32 bits reales.
- Ancho de 16 bits beneficia al modo Thumb.
  - Usando buses de 16 bits los sistema usan menos recursos que con 32 bits, pero son más lentos que en modo ARM.
  - Altas prestaciones en modo Thumb.
  - Mayor número de pines disponible para E/S.
- Reducción de la EMI en la ejecución del programa en memoria interna.

Excepciones y prioridades

Exception	Mode	Address
Reset	Supervisor	0x00000000
Undefined instruction	Undefined	0x00000004
Software interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
IRQ (interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

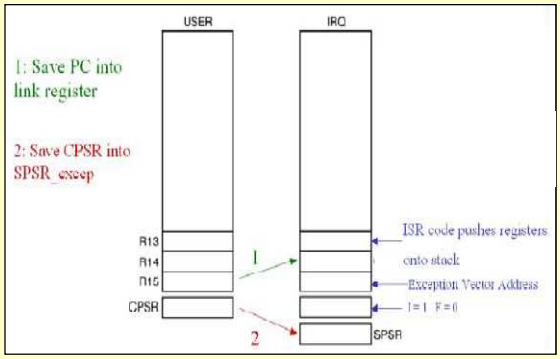
Priority	Exception
Highest 1	Reset
2	Data Abort
3	FIQ
4	IRQ
5	Prefetch Abort
Lowest 6	Undefined instruction SWI

Instrucciones asociadas

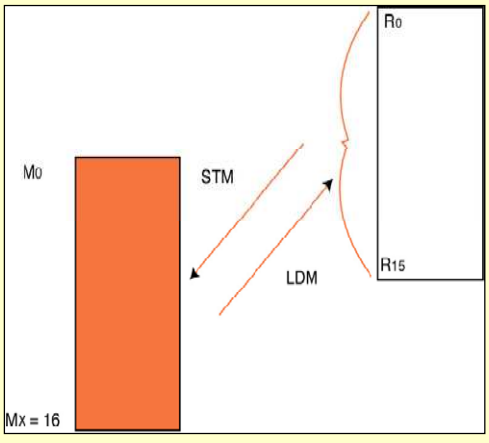
- MOVS R15,R14 ; Retorno y SPSR
- MOVS R15,R14 ; Para FIQ, IRQ, Abort
- SUBS R15, R14,#8 ; Data Abort
- EQMOV R1, #0x00800000 ; mejor que:
- if(x<100)

```
{
    x++;
}
```

Actuación excepciones

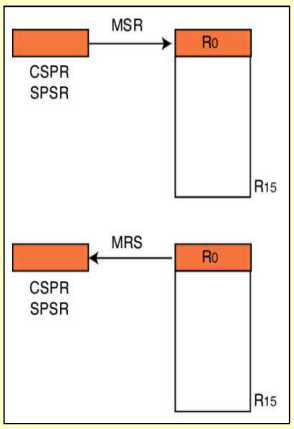


Copia de múltiples registros

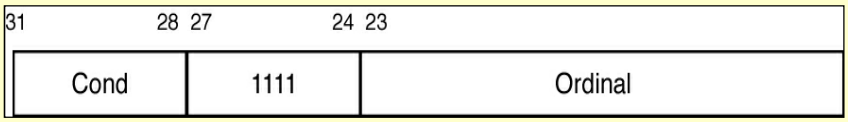


Modificación CPSR

No válido para
Modo Usuario



SWI



SWI #3
switch(*(R14-4) & 0x00FFFFFF)

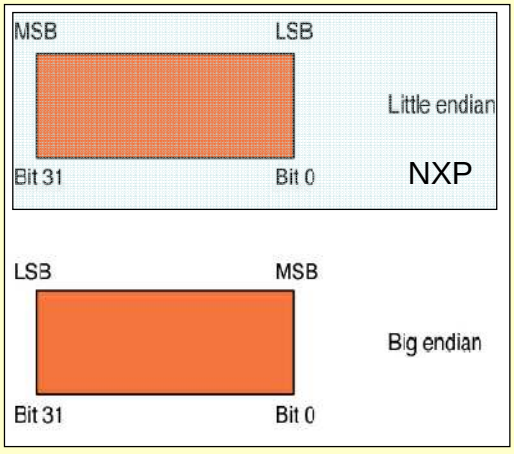
case (SWI-1)

#define SWIcall2 asm { swi #2}

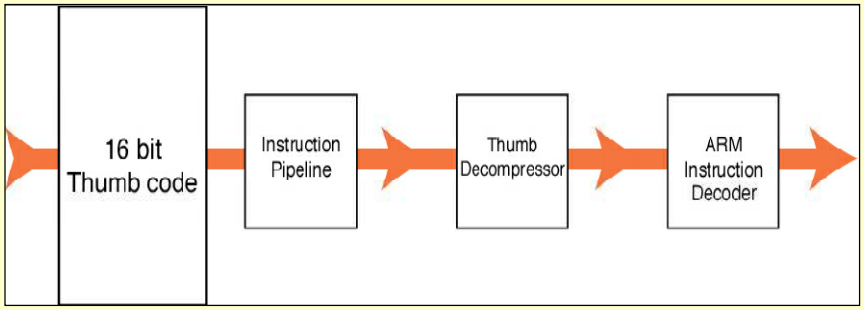
MAC

- MUL Multiplica resultado 32 bit
- MULA Multiplica y acumula result 32 bits
- UMULL Multipl no signado result 64 bits
- UMLAL Multipl y acc no signado result 64 bits
- SMULL Multipl signado resultado 64 bits
- SMLAL Multipl y acc signado result 64 bits

Formatos de datos



Thumb

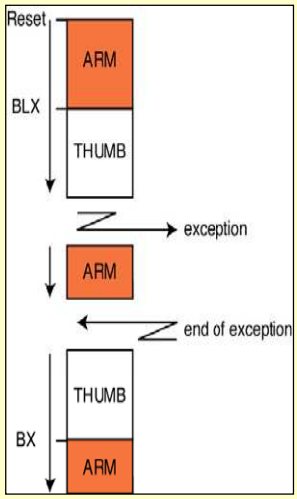


ARM Instruction
ADD R0, R0,R1

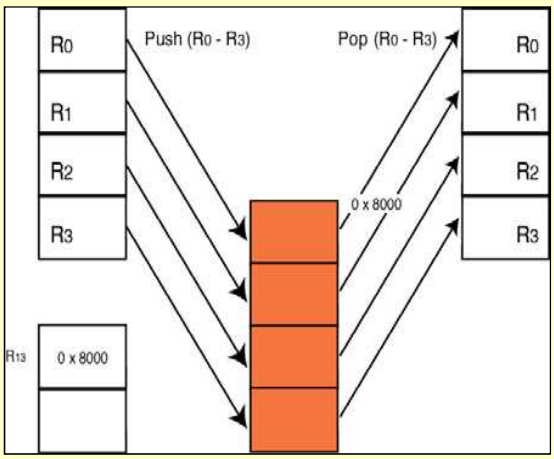
THUMB Instruction
ADD R0,R1

R0 = R0+R1

Thumb



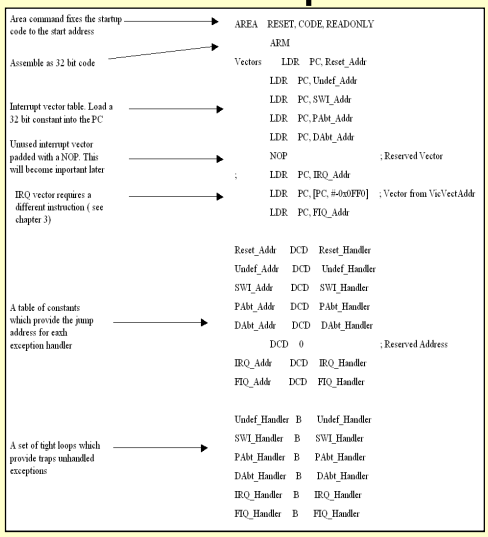
Thumb



Pasaje bidireccional ARM Thumb

```
#pragma ARM // pasar a instrucciones ARM
int main(void)
{
while(1)
{
thumb_function(); //Call THUMB function
}
}
#pragma THUMB //Pasar a instrucciones THUMB
void thumb_function(void)
{
unsigned long i,delay;
for (i = 0x00010000;i < 0x01000000 ;i = i<<1) //LED flasher
{
for (delay = 0;delay<0x000100000;delay++) //simple delay loop
{
;
}
IOSET1 = i; //Set the next LED
}
}
```

Startup



Start up

- First: setup interrupt vectors

```
Vectors:
        b       _start          // reset - _start
        ldr     pc,_undf        // undefined - _undf
        ldr     pc,_swi         // SWI - _swi
        ldr     pc,_pabt        // program abort - _pabt
        ldr     pc,_dabt        // data abort - _dabt
        nop                     // reserved
        ldr     pc,[pc,#-0xFF0] // IRQ - read the VIC
        ldr     pc,_fiq         // FIQ - _fiq
```

- Second: setup stacks for all processor modes

```
ldr   r0,=_stack
msr   CPSR_c,#MODE_UND|I_BIT|F_BIT // Undefined Instruction Mode
mov   sp,r0
sub   r0,r0,#UND_STACK_SIZE
msr   CPSR_c,#MODE_ABT|I_BIT|F_BIT // Abort Mode
mov   sp,r0
sub   r0,r0,#ABT_STACK_SIZE
msr   CPSR_c,#MODE_FIQ|I_BIT|F_BIT // FIQ Mode
mov   sp,r0
sub   r0,r0,#FIQ_STACK_SIZE
msr   CPSR_c,#MODE_IRQ|I_BIT|F_BIT // IRQ Mode
mov   sp,r0
sub   r0,r0,#IRQ_STACK_SIZE
msr   CPSR_c,#MODE_SVC|I_BIT|F_BIT // Supervisor Mode
mov   sp,r0
sub   r0,r0,#SVC_STACK_SIZE
msr   CPSR_c,#MODE_SYS|I_BIT|F_BIT // System Mode
mov   sp,r0
```

- Third: copy initialized data from ROM to RAM

```
  ldr   r1,=_etext    // -> ROM data start
  ldr   r2,=_data     // -> data start
  ldr   r3,=_edata    // -> end of data
1: cmp  r2,r3         // check if data to move
  ldrlo r0,[r1],#4    // copy it
  strlo r0,[r2],#4
  blo   1b            // loop until done
```

- Fourth: clear the BSS segment

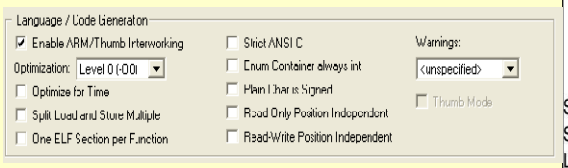
```
  mov   r0,#0           // get a zero
  ldr   r1,=__bss_start__ // -> bss start
  ldr   r2,=__bss_end__   // -> bss end
2: cmp  r1,r2           // check if data to clear
  strlo r0,[r1],#4      // clear 4 bytes
  blo   2b              // loop until done
```

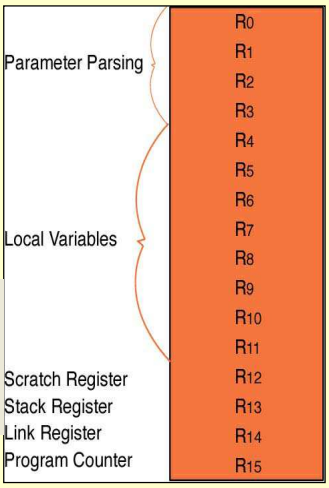
- Fifth: Call main()

```
  mov   r0,#0         // no arguments (argc = 0)
  mov   r1,r0
  mov   r2,r0
  mov   fp,r0         // null frame pointer
  mov   r7,r0         // null frame pointer for thumb
  ldr   r10,=main
  mov   lr,pc
  bx    r10           // enter main()
reset:
  // code to reset the processor goes here
```

ARM Procedure Call Standard

Respetar el APCS, permite
Vincular módulos generados
Con distintos compiladores





STDIO

```
int fputc(int ch, FILE *f) {
   return (sendchar(ch));
   }
int fgetc(FILE *f) {
   return (sendchar(getkey()));
```

STDIO

```
int sendchar (int ch) { /* Enviar al puerto serie */
    if (ch == '\n') {
    while (!(U1LSR & 0x20));
    U1THR = CR; /* output CR */
    }
while (!(U1LSR & 0x20));
    return (U1THR = ch);
    }
int getkey (void) { /* Leer del puerto serie */
    while (!(U1LSR & 0x01));
    return (U1RBR);
```

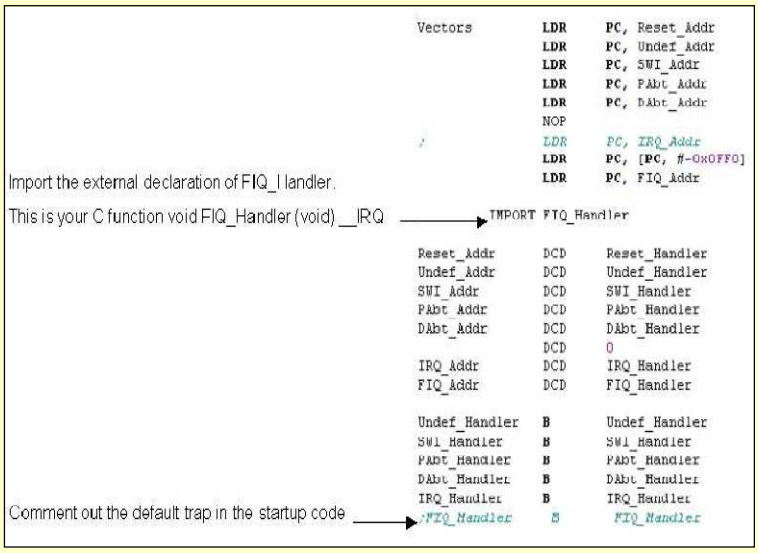

Periféricos

- #define SFR (*((volatile unsigned long *) 0xFFFFF000))
- Se deberán incluir los archivos que contienen las definiciones propias de cada procesador
  - LPC21xx.h
  - LPC22xx.h
  - LPC210x.h

Rutinas de atención de interrupciones

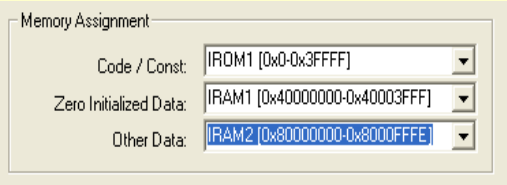
```
void FIQint (void) __irq
  {
  IOSET1 = 0x00FF0000; //Set pines LED
  EXTINT = 0x00000002; //Clear Flag
                       //interrupciones perif
  }
```


Vectores

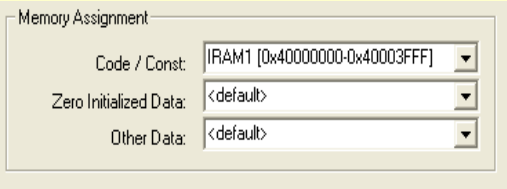


Ubicación del código

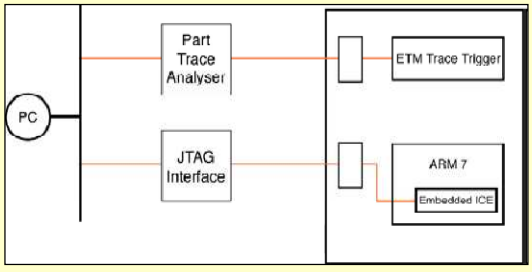
En Flash

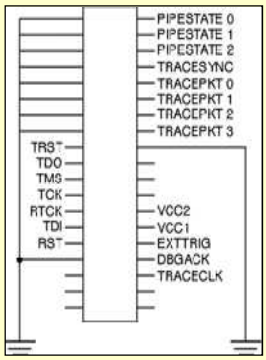


En RAM



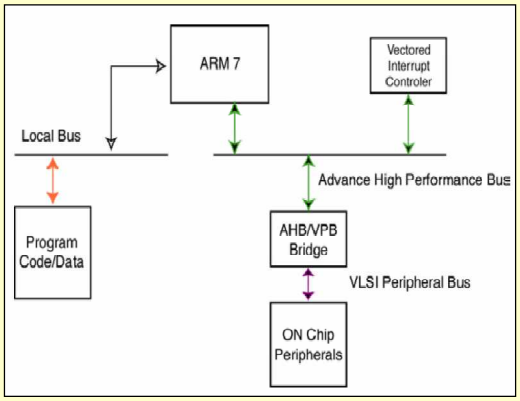
JTAG



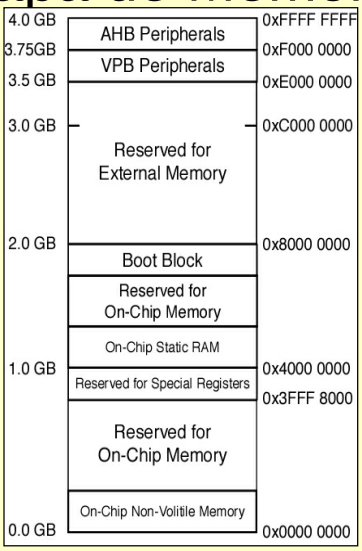


Buses Internos

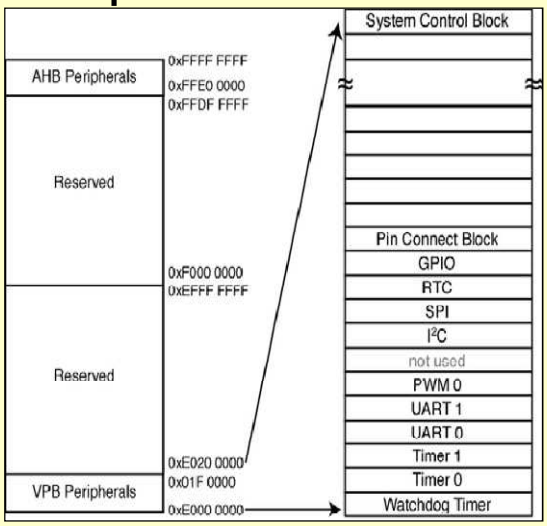
- AHB. Advance High Performance Bus.
- VPB. VLSI Peripherall Bus



Mapa de Memoria



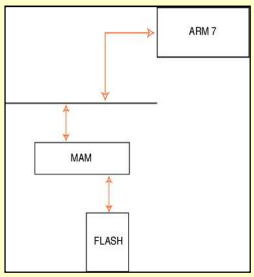
Mapa de Memoria

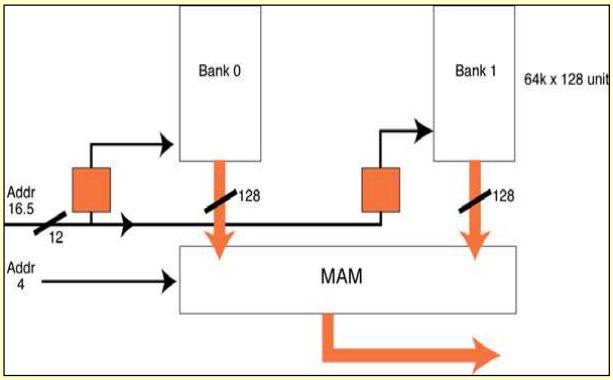


Memory Accelerator Module

- Está presente en el bus local y se sitúa entre la memoria Flash y la CPU (ARM).
- Permite acelerar el tiempo de acceso a la Flash.
- La CPU es capaz de alcanzar una frecuencia de operación de 80 MHz, sin embargo la memoria flash tiene un tiempo de acceso de 50ns (20 MHz).
  - Solución 1. →Ejecutar el programa en RAM.
    - Problema Escasa capacidad disponible!!!
  - Solución 2. →Memoria caché de instrucciones
  - →Almacena la zona de memoria más reciente accedida.
    - Problema. Periférico complejo que consume un número elevado de puertas (silicio).
- MAM es un compromiso entre la complejidad de la caché y la facilidad para acceder la CPU a la Flash.

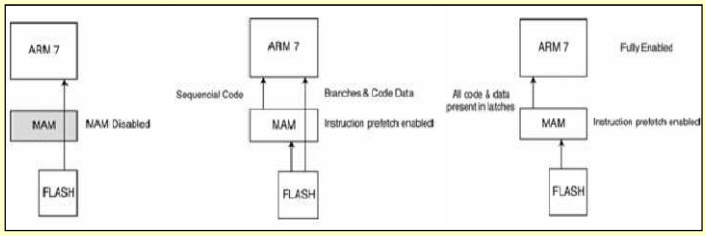
Memory Accelerator Module





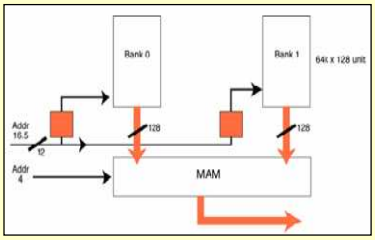
Memory Accelerator Module

- Modos de funcionamiento:
  - Dashabilitado (OFF)
    - Los accesos a memoria se realizan directamente desde la Flash.
  - Parcialmente habilitado
    - Sólo los accesos a datos y los saltos se realizan directamente desde la flash.
  - Habilitación total
    - Todos los accesos quedan latcheados a través del MAM.



MAM

- Trata de tener la próxima instrucción a ejecutar en su memoria local, a tiempo para ser ejecutada por la CPU.
- Un único acceso a la flash puede cargar 4 instrucciones (modo ARM) o hasta 8 (modo Thumb).
- La memoria de programa está interpolada entre 2 bancos de memoria, haciendo que durante la ejecución de un código secuencial, una fase de búsqueda se ejecuta en un banco, mientras los siguientes 128 bits se prepara. Esto asegura que estarán listos una vez que los últimos 128 bits se hayan ejecutado.
- Este mecanismo funciona mejor en modo ARM dónde el uso de los códigos de condición para provocar los saltos, hace que el flujo del programa se mantenga lineal la mayor parte del tiempo. En el caso de bucles o saltos cortos, los buffers intermedios retienen las instrucciones que serán re-ejecutadas si así son requeridas



MAM: Registros de configuración

- MAMCR. Selección del modo de funcionamiento.
- MAMTIM. Control de tiempo.
  - Selecciona el número de ciclos CCLK necesarios para acceder a la flash.
  - Adapta la frecuencia del MAM a la frecuencia de la CPU.
  - Selección de 1 a 7 ciclos de duración de la fase de búsqueda de la instrucción.
  - Recomendación del fabricante:
    - Si Fcpu < 20 Mhz □□MAMTIM=001 (1)
    - Si 20 Mhz < Fcpu < 40 Mhz □□ MAMTIM=010 (2)
    - Si Fcpu > 40 Mhz □□ MAMTIM=011(3)

Name	Description	Access	Reset value[1]	Address
MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 3–8.	R/W	0x0	0xE01F C000
MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for Flash memory fetches (1 to 7 processor clocks).	R/W	0x07	0xE01F C004

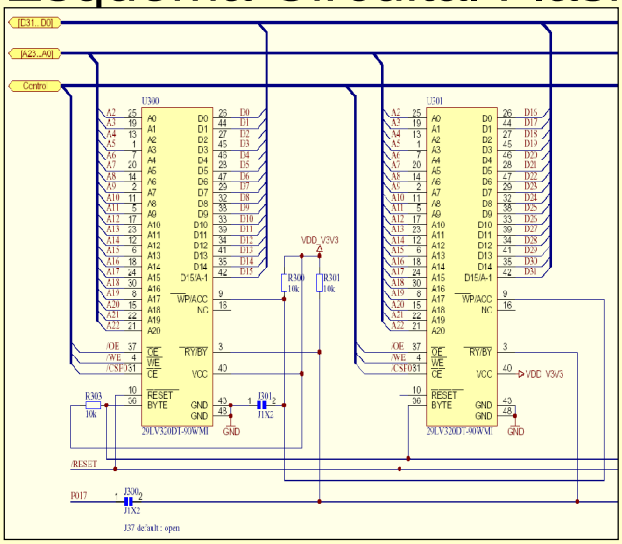
Table 8. MAM Control Register (MAMCR - address 0xE01F C000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAM_mode_control	00	MAM functions disabled	0
		01	MAM functions partially enabled	
		10	MAM functions fully enabled	
		11	Reserved. Not to be used in the application.	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

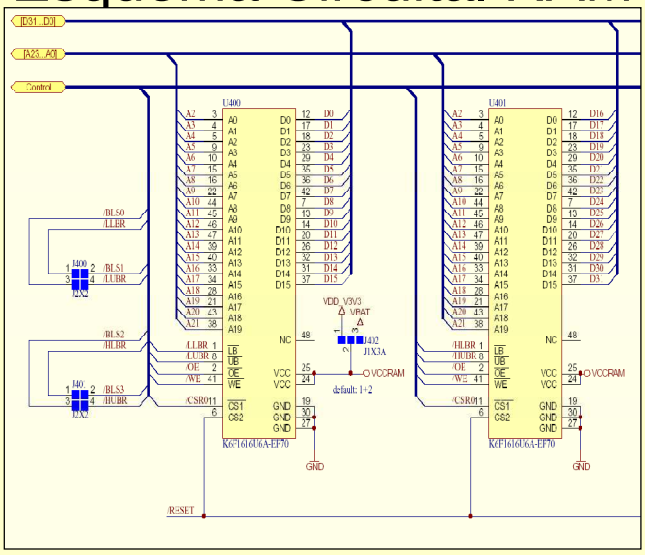
Table 9. MAM Timing register (MAMTIM - address 0xE01F C004) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MAM_fetch_cycle_timing	000	0 - Reserved.	07
		001	1 - MAM fetch cycles are 1 processor clock (CCLK) in duration	
		010	2 - MAM fetch cycles are 2 CCLKs in duration	
		011	3 - MAM fetch cycles are 3 CCLKs in duration	
		100	4 - MAM fetch cycles are 4 CCLKs in duration	
		101	5 - MAM fetch cycles are 5 CCLKs in duration	
		110	6 - MAM fetch cycles are 6 CCLKs in duration	
		111	7 - MAM fetch cycles are 7 CCLKs in duration	
			Warning: These bits set the duration of MAM Flash fetch operations as listed here. Improper setting of this value may result in incorrect operation of the device.	
7:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

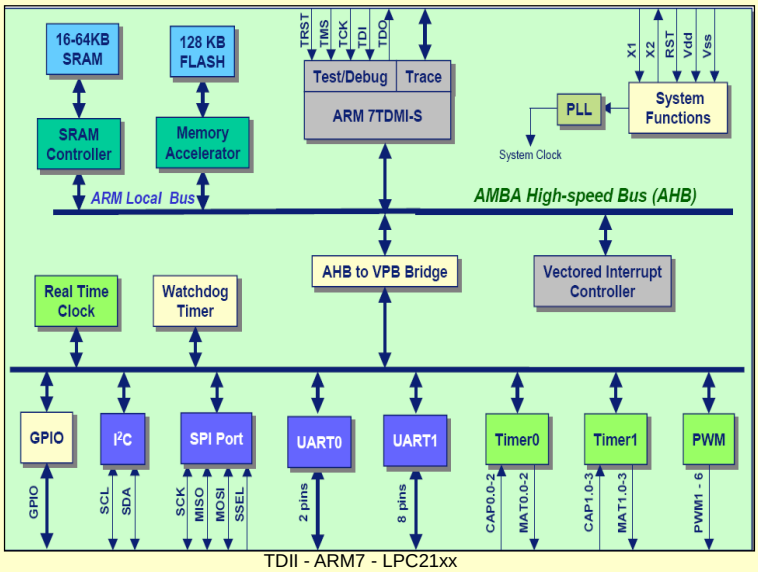
Esquema Circuital Flash



Esquema Circuital RAM

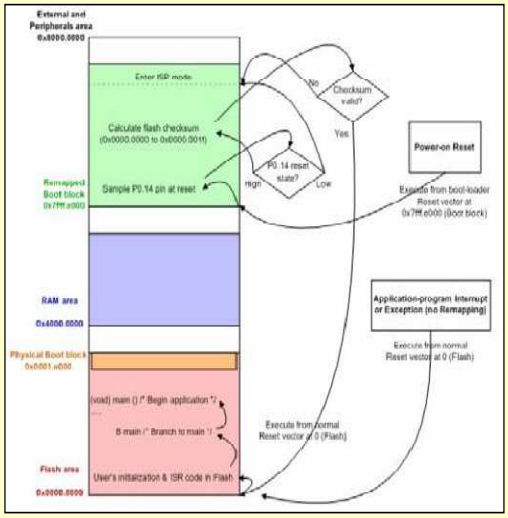


LPC2104/5/6



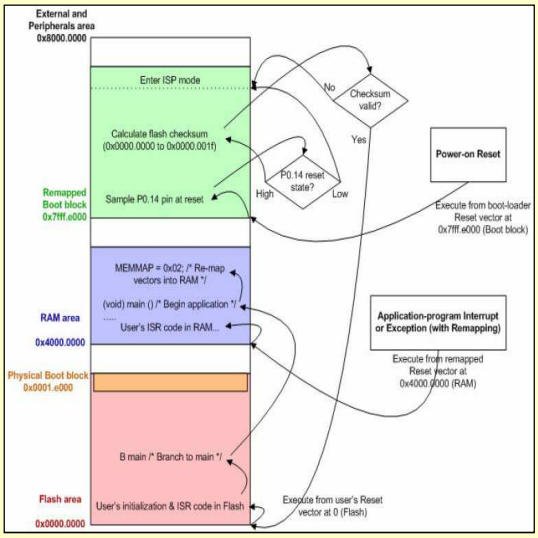
Secuencia de arranque (Vectores en flash)

- Después del Reset siempre se mapea el bloque de memoria Boot (12Kb) (que reside al final de la memoria flash, ej. 0x0001.e000) en la dirección 0x7fff.d000. El vector de reset (0x0000.0000) se remapea a la dir. 0x7fff.d000.
- El código testea el pin P0.14 y comprueba si hay un programa válido en la flash.
  - Comprueba la clave o firma grabada en la posición 0x14 (vector no utilizado) de la memoria flash.
  - Si el resultado es cero, transfiere el control a la dirección indicada por el vector de Reset normal situado en la posición cero de la Flash.
- Antes de comenzar con la ejecución del main() el sistema inicializa variables de usuario, la pila, etc…

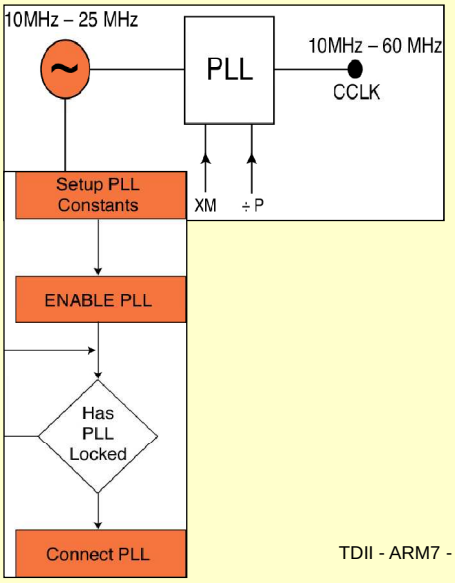


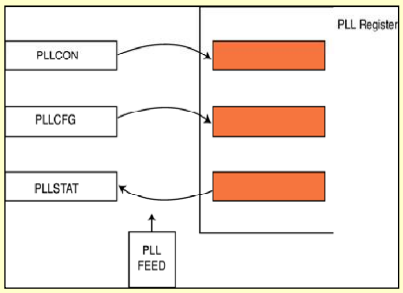
Secuencia de Arranque (Vectores mapeados en RAM)

- Con objeto de facilitar las tareas de depuración cuando se trabaja con interrupciones, es posible mapear la tabla de vectores y el código de usuario que reside en Flash, en la RAM interna.
- Para ello es necesario acceder al registro MEMMAP (Memory Mapping Control Register).

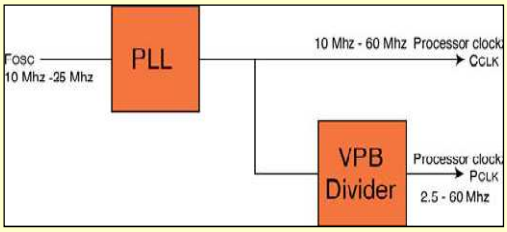


PLL

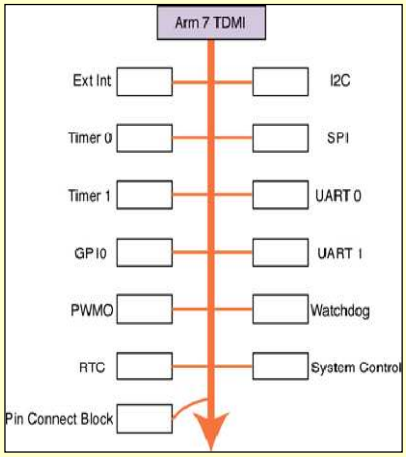


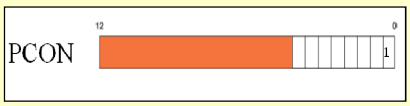


Divisor VPB

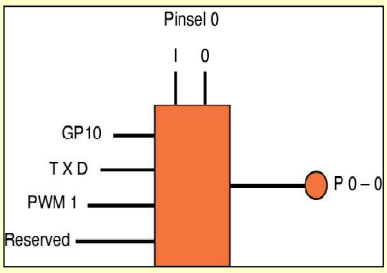


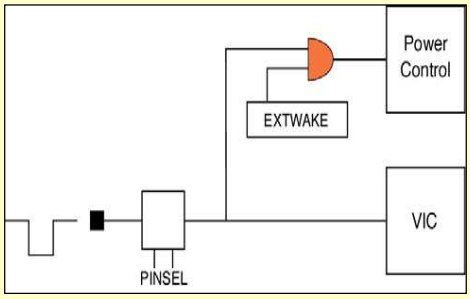
Periféricos Generales





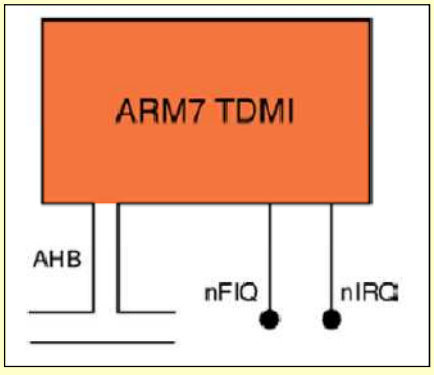
Selección de función de patas





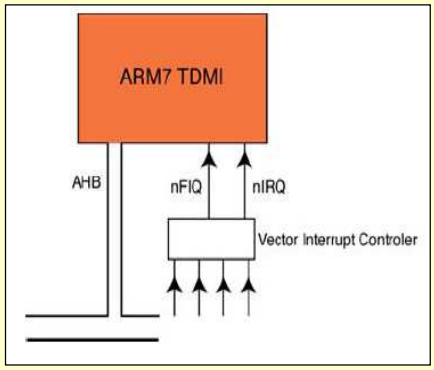
Interrupciones

ARM tiene dos fuentes de interrupción (FIQ, IRQ)



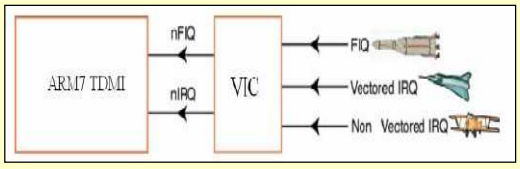
Interrupciones

- En el LPC21xx se añade el VIC, esencial para poder atender más interrupciones rápidamente.



Interrupciones

- 32 entradas de interrupción:
  - Vectorizada IRQ (16 como máximo)
  - No Vectorizada IRQ (las que no sean vectorizadas)
  - FIQ (una o varias, a elección)
- Asignación dinámica de prioridades (16 niveles) para las interrupciones IRQ vectorizadas.
- Interrupción software (SWI)



VECTORED INTERRUPT CONTROLLER (VIC)

Address	Name	Description	Access	Reset Value*
0xFFFF F000	VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ	RO	0
0xFFFF F004	VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ	RO	0
0xFFFF F008	VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0
0xFFFF F00C	VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0
0xFFFF F010	VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ	R/W	0
0xFFFF F014	VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	W	0
0xFFFF F018	VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0
0xFFFF F01C	VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register	W	0
0xFFFF F020	VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode	R/W	0
0xFFFF F030	VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0

VECTORED INTERRUPT CONTROLLER (VIC)

0xFFFF F034	VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.	R/W	0
0xFFFF F100	VICVectAddr0	Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.	R/W	0
0xFFFF F104	VICVectAddr1	Vector address 1 register	R/W	0
0xFFFF F108	VICVectAddr2	Vector address 2 register	R/W	0
0xFFFF F10C	VICVectAddr3	Vector address 3 register	R/W	0
0xFFFF F110	VICVectAddr4	Vector address 4 register	R/W	0
0xFFFF F114	VICVectAddr5	Vector address 5 register	R/W	0
0xFFFF F118	VICVectAddr6	Vector address 6 register	R/W	0
0xFFFF F11C	VICVectAddr7	Vector address 7 register	R/W	0
0xFFFF F120	VICVectAddr8	Vector address 8 register	R/W	0
0xFFFF F124	VICVectAddr9	Vector address 9 register	R/W	0

VECTORED INTERRUPT CONTROLLER (VIC)

Address	Name	Description	Access	Reset Value*
0xFFFF F128	VICVectAddr10	Vector address 10 register	R/W	0
0xFFFF F12C	VICVectAddr11	Vector address 11 register	R/W	0
0xFFFF F130	VICVectAddr12	Vector address 12 register	R/W	0
0xFFFF F134	VICVectAddr13	Vector address 13 register	R/W	0
0xFFFF F138	VICVectAddr14	Vector address 14 register	R/W	0
0xFFFF F13C	VICVectAddr15	Vector address 15 register	R/W	0
0xFFFF F200	VICVectCntl0	Vector control 0 register. Vector Control Registers 0–15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0
0xFFFF F204	VICVectCntl1	Vector control 1 register	R/W	0
0xFFFF F208	VICVectCntl2	Vector control 2 register	R/W	0
0xFFFF F20C	VICVectCntl3	Vector control 3 register	R/W	0
0xFFFF F210	VICVectCntl4	Vector control 4 register	R/W	0
0xFFFF F214	VICVectCntl5	Vector control 5 register	R/W	0
0xFFFF F218	VICVectCntl6	Vector control 6 register	R/W	0
0xFFFF F21C	VICVectCntl7	Vector control 7 register	R/W	0
0xFFFF F220	VICVectCntl8	Vector control 8 register	R/W	0
0xFFFF F224	VICVectCntl9	Vector control 9 register	R/W	0
0xFFFF F228	VICVectCntl10	Vector control 10 register	R/W	0
0xFFFF F22C	VICVectCntl11	Vector control 11 register	R/W	0
0xFFFF F230	VICVectCntl12	Vector control 12 register	R/W	0
0xFFFF F234	VICVectCntl13	Vector control 13 register	R/W	0
0xFFFF F238	VICVectCntl14	Vector control 14 register	R/W	0
0xFFFF F23C	VICVectCntl15	Vector control 15 register	R/W	0

VECTORED INTERRUPT CONTROLLER (VIC)

VICSoftInt	Function	Reset Value
31:0	1:force the interruot request with this bit number. 0: do not force the interrupt request with this bit number. Writing zeroes to bits in VICSoftInt has no effect, see VICSoftIntClear.	0

VICSoftIntClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Software Interrupt register, thus releasing the forcing of this request. 0: writing a 0 leaves the correeponding bit in VICSoftInt unchanged.	0

VICRawIntr	Function	Reset Value
31:0	1: the interrupt request or software interrupt with this bit number is asserted. 0: the interrupt request or software interrupt with this bit number is negated.	0

VICIntEnable	Function	Reset Value
31:0	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See the VICIntEnClear register (Table 46 below), for how to disable interrupts.	0

VICIntEnClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request. 0: writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	0

VECTORED INTERRUPT CONTROLLER (VIC)

VICIntSelect	Function	Reset Value
31:0	1: the interrupt request with this bit number is assigned to the FIQ category. 0: the interrupt request with this bit number is assigned to the IRQ category.	0

VICIRQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as IRQ, and asserted.	0

VICFIQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as FIQ, and asserted.	0

VICVectCntl0-15	Function	Reset Value
5	1: this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
4:0	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower-numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0

VICVectAddr0-15	Function	Reset Value
31:0	When one or more interrupt request or software interrupt is (are) enabled, classified as IRQ, asserted, and assigned to an enabled vectored IRQ slot, the value from this register for the highest priority such slot will be provided when the IRQ service routine reads the Vector Address register (VICVectAddr).	0

VECTORED INTERRUPT CONTROLLER (VIC)

VICDefVectAddr	Function	Reset Value
31:0	When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned.	0

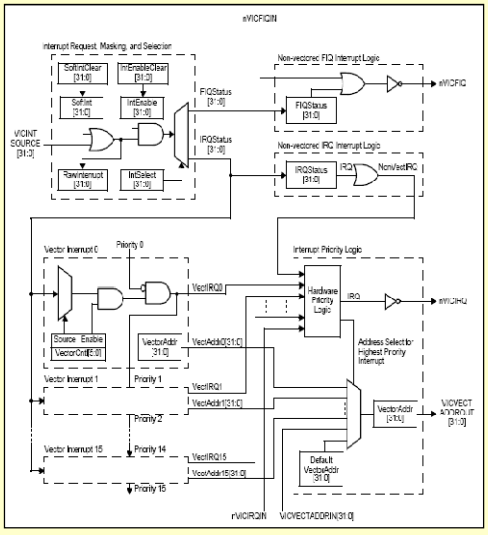
VICVectAddr	Function	Reset Value
31:0	If any of the interrupt requests or software interrupts that are assigned to a vectored IRQ slot is (are) enabled, classified as IRQ, and asserted, reading from this register returns the address in the Vector Address Register for the highest-priority such slot. Otherwise it returns the address in the Default Vector Address Register. Writing to this register does not set the value for future reads from it. Rather, this register should be written near the end of an ISR, to update the priority hardware.	0

VICProtection	Function	Reset Value
0	1: the VIC registers can only be accessed in privileged mode. 0: VIC registers can be accessed in User or privileged mode.	0

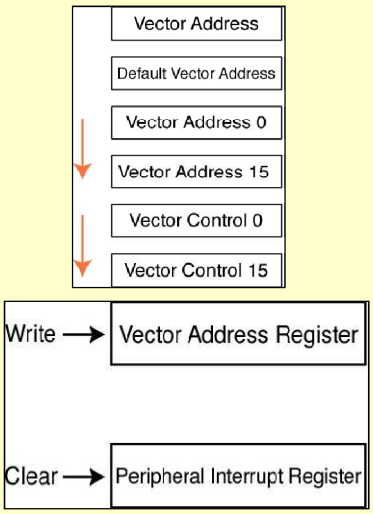
Fuentes de interrupción

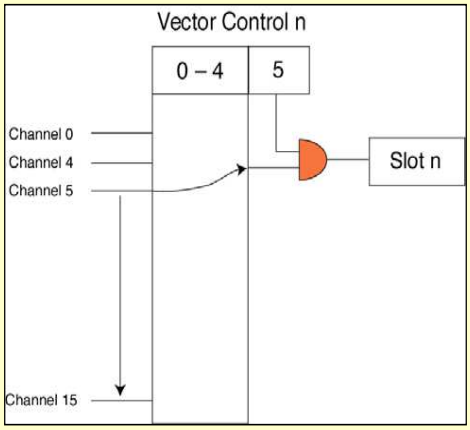
Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
Timer 0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
Timer 1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART 0	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART 1	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6) Capture 0 - 3 (CR0, CR1, CR2, CR3)	8
I2C	SI (state change)	9
SPI	SPIF, MODF	10
-	reserved	11
PLL	PLL Lock (PLOCK)	12
RTC	RTCCIF (Counter Increment), RTCALF (Alarm)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16

Fuentes de interrupción

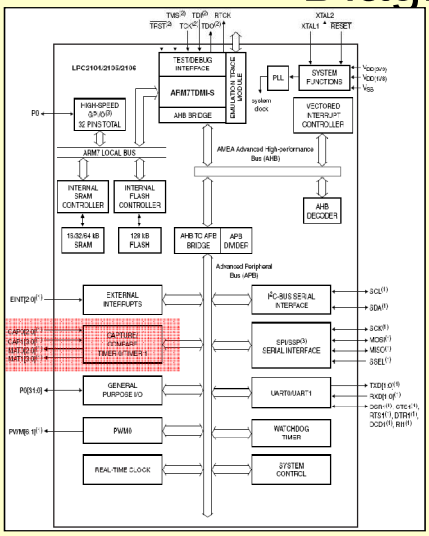


Interrupciones Vectorizadas





Diagrama



APB peripheral	Base address	Peripheral name
0	0xE000 0000	Watchdog timer
1	0xE000 4000	Timer 0
2	0xE000 8000	Timer 1
3	0xE000 C000	UART0

Timers y VIC

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	23	22	21	20	19	18	17	16
Symbol	-	-	-	-	-	-	-	EINT2
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	15	14	13	12	11	10	9	8
Symbol	EINT1	EINT0	RTC	PLL	-	SPI/SSP	I2C	PWM0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	7	6	5	4	3	2	1	0
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Temporizadores LPC2105/6

- Operación como temporizador o contador de eventos
- 32-bit Timer/Counter con prescaler programable de 32-bits
- Hasta 4 (Timer 1) y 3 (Timer 0) canales de captura de 32-bits que pueden tomar una instantánea del timer cuando se produzca una transición de la entrada. Un evento de captura pueden producir interrupciones.

Temporizadores LPC2105/6

- Cuatro registors de coincidencia de 32-bits que permiten:
  - Operación continua con opción de generación de interrupción en la coincidencia.
  - Detener el temporizador en la coincidencia con la generación opcional de interrupción.
  - Reset del timer en la coincidencia con la generación opcional de interrupción.

Temporizadores LPC2105/6

- Hasta cuatro (Timer 1) y tres (Timer 0) salidas externas correspondiente a los registros de coincidencia con las siguientes opciones:
  - Pasar a estado bajo en la coincidencia.
  - Pasar a estado alto en la coincidencia
  - Cambiar de estado en la coincidencia.
  - No realizar acción alguna en la coincidencia.

Patas asociadas

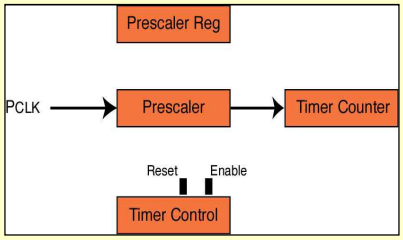
Pin name	Pin direction	Pin Description
CAP0.2..0 CAP1.3..0	Input	**Capture Signals**- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt.
MAT0.0 MAT1.0	Output	**External Match Output 0/1**- When match register 0/1 (MR0/1) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output.
MAT0.1 MAT1.1	Output	**External Match Output 1-** See the MAT0/MAT1 description above.
MAT0.2 MAT1.2	Output	**External Match Output 2-** See the MAT0/MAT1 description above.
MAT1.3	Output	**External Match Output 3-** See the MAT1 description above.

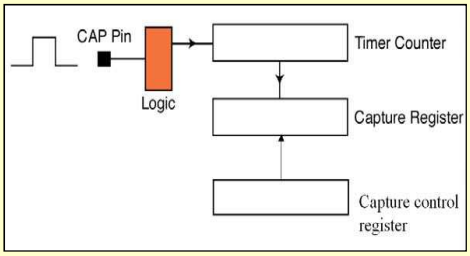
Patas asociadas al timer 0

P0.3/SDA/MAT0.0	21[2]	I/O	**P0.3** — Port 0 bit 3. The output is open-drain.
		I/O	**SDA** — I^2C-bus data input/output. Open-drain output (for I^2C-bus compliance).
		O	**MAT0.0** — Match output for Timer 0, channel 0. The output is open-drain.
P0.4/SCK/CAP0.1	22[1]	I/O	**P0.4** — Port 0 bit 4.
		I/O	**SCK** — Serial clock for SPI/SSP[3]. Clock output from master or input to slave.
		I	**CAP0.1** — Capture input for Timer 0, channel 1.
P0.5/MISO/MAT0.1	23[1]	I/O	**P0.5** — Port 0 bit 5.
		I/O	**MISO** — Master In Slave Out for SPI/SSP[3]. Data input to SPI/SSP master or data output from SPI/SSP slave.
		O	**MAT0.1** — Match output for Timer 0, channel 1.
P0.6/MOSI/CAP0.2	24[1]	I/O	**P0.6** — Port 0 bit 6.
		I/O	**MOSI** — Master Out Slave In for SPI/SSP[3]. Data output from SPI/SSP master or data input to SPI/SSP slave.
		I	**CAP0.2** — Capture input for Timer 0, channel 2.
P0.7/SSEL/PWM2	28[1]	I/O	**P0.7** — Port 0 bit 7.
		I	**SSEL** — Slave Select for SPI/SSP[3]. Selects the SPI/SSP interface as a slave.
		O	**PWM2** — Pulse Width Modulator output 2.
P0.8/TXD1/PWM4	29[1]	I/O	**P0.8** — Port 0 bit 8.
		O	**TXD1** — Transmitter output for UART 1.
		O	**PWM4** — Pulse Width Modulator output 4.

Similar para el timer 1

Temporizadores





Registros asociados

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
IR	0xE0004000 T0IR	0xE0008000 T1IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight (Timer 1) or seven (Timer 0) possible interrupt sources are pending.	R/W	0
TCR	0xE0004004 T0TCR	0xE0008004 T1TCR	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
TC	0xE0004008 T0TC	0xE0008008 T1TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
PR	0xE000400C T0PR	0xE000800C T1PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
PC	0xE0004010 T0PC	0xE0008010 T1PC	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.	R/W	0
MCR	0xE0004014 T0MCR	0xE0008014 T1MCR	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
MR0	0xE0004018 T0MR0	0xE0008018 T1MR0	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

Registros asociados

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
MR1	0xE0004018 T0MR1	0xE0008018 T1MR1	Match Register 1. See MR0 description.	R/W	0
MR2	0xE0004020 T0MR2	0xE0008020 T1MR2	Match Register 2. See MR0 description.	R/W	0
MR3	0xE0004024 T0MR3	0xE0008024 T1MR3	Match Register 3. See MR0 description.	R/W	0
CCR	0xE0004028 T0CCR	0xE0008028 T1CCR	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
CR0	0xE000402C T0CR0	0xE000802C T1CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the capture[0] signal.	RO	0
CR1	0xE0004030 T0CR1	0xE0008030 T1CR1	Capture Register 1. See CR0 description.	RO	0
CR2	0xE0004034 T0CR2	0xE0008034 T1CR2	Capture Register 2. See CR0 description.	RO	0
CR3	0xE0004038 T0CR3	0xE0008038 T1CR3	Capture Register 3. See CR0 description. Not usable on Timer 0.	RO	0
EMR	0xE000403C T0EMR	0xE000803C T1EMR	External Match Register. The EMR controls the external match pins MATn.	R/W	0

Registros asociados

Timer Control Register (TCR: Timer 0 - T0TCR: 0xE0004004; Timer 1 - T1TCR: 0xE0008004)

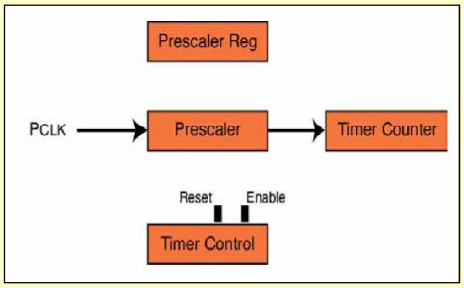
TCR	Function	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0

Timer Counter (TC: Timer 0 - T0TC: 0xE0004008; Timer 1 - T1TC: 0xE0008008)

Prescale Register (PR: Timer 0 - T0PC: 0xE000400C; Timer 1 - T1PC: 0xE000800C)

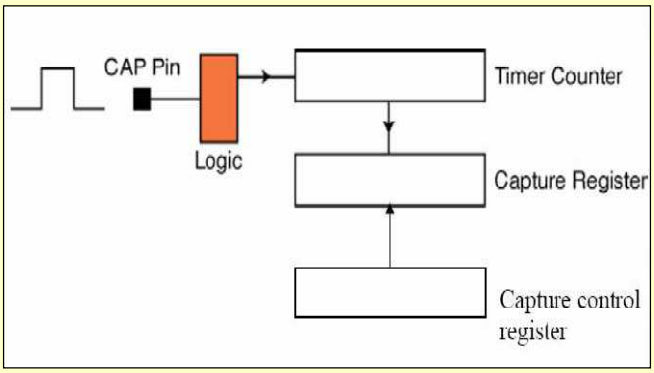
Timers: el contador

- Modo temporizador:
  - Contador de 32 bits y preescaler de 32 bits.
  - Fcia timer counter=PCLK/(valor preescaler+1)
  - Tick=1/PCLK*(preescaler+1)
  - Tickmax=1/PCLKmin*(preescalermax+1)=286,33s (para 60Mhz CCLK)
  - Tickmin=1/PCLKmax*(preescalermin+1)=16ns (para 60Mhz CCLK)



Captura de flancos

- Captura configurable de flanco de subida, bajada o toggle



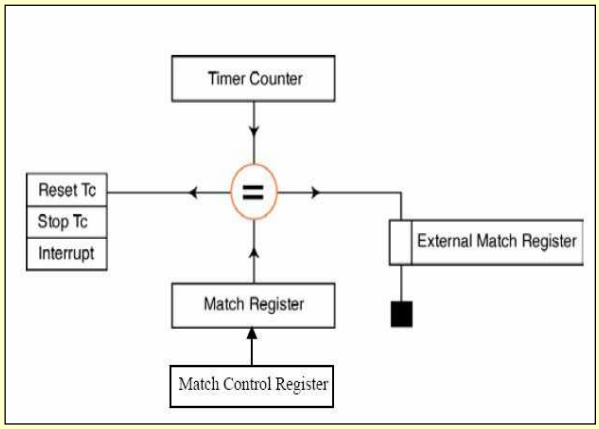
Programación captura

```
int main(void)
{
    VPBDIV = 0x00000002; // Set pclk to 30 MHz
    PINSEL0 = 0x00000020; // Enable pin 0.2 as capture channel0
    T0PR = 0x00007530; // Load prescaler for 1 Msec tick
    T0TCR = 0x00000002; // Reset counter and prescaler
    T0CCR = 0x00000005; // Capture on rising edge of channel0
    T0TCR = 0x00000001; // enable timer
    VICVectAddr4 = (unsigned)T0isr; // Set the timer ISR vector
       address
    VICVectCntl4 = 0x00000024; // Set channel
    VICIntEnable = 0x00000010; // Enable the interrupt
    while(1);
    }
```

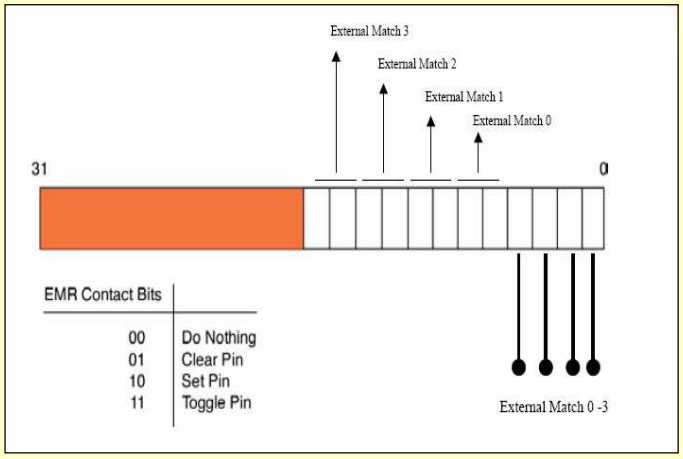

ISR Timer

```
void T0isr (void) __irq
{
    static int value;
    value = T0CR0; // read the capture value
    T0IR |= 0x00000001; // Clear match 0 interrupt
    VICVectAddr = 0x00000000; // Dummy write to
                    // signal end of  interrupt
}
```

Comparación



Comparación: generación de señales



Comparación: ejemplo

- Retardo de varios milisegundos:
  - El preescaler a 0 (valor de reset)
  - PCLK=15MHz.

```
void delay_timer_1(int mseg){
T1TCR=0X02;              // Reset del contador
T1MCR=0x06;              // Stop at match and clear
T1MR0=(15000-1)*mseg;
T1TCR=0x01;              //Start
while(!(T1IR&0x01));     // Espera hasta que pase el tiempo
T1IR=0x01;               //clear flag
}
```


Ejemplo comparación

- Generación de una señal PWM.
  - Reg comp 0 para T y Reg comp 1 para TH
  - Configuración:
  - int main(void)

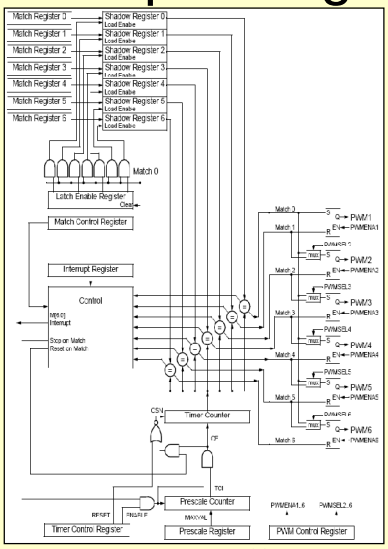
```
{
VPBDIV = 0x00000002;                  // Configure the VPB divi
PINSEL0 |= 0x00000800;                //Match1 as output
T0PR = 0x0000001E;                    //Load prescaler
T0TCR = 0x00000002;                   //Reset counter and prescaler
T0MCR = 0x00000003;                   //On match reset counter and generate an interrupt
T0MR0 = 0x00000101;                   //Set the cycle time
T0MR1 = 0x00000000;                   // Set duty cycle to zero
T0EMR = 0x00000042;                   //On match clear MAT1
T0TCR = 0x00000001;                   //enable timer
VICVectAddr4 = (unsigned)T0isr;       //Set the timer ISR vector address
VICVectCntl4 = 0x00000024;            //Set channel
VICIntEnable |= 0x00000010;           //Enable the interrupt
while(1);
}
```

Ejemplo comparación

- Atención a la interrupción:

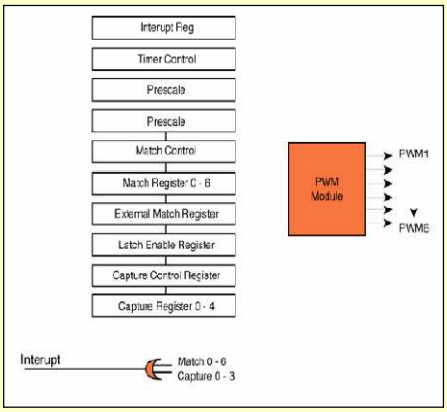
```
void T0isr (void) __irq
{
T0EMR |= 0x00000002;          //Set MAT1 high for begining
                              //of the cycle
T0MR1++;                      //Increment PWM Duty cycle
T0MR1 = T0MR1&0x000000FF;     //Limit duty cycle
T0IR|= 0x00000001;            //Clear match 0 interrupt
VICVectAddr = 0x00000000;     //Dummy write to signal end
                              // of interrupt
}
```


PWM: esquema general



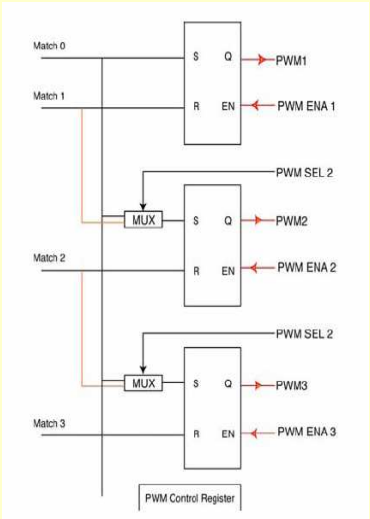
PWM

- Es un módulo timer con algunas funciones añadidas.
  - 6 canales PWM



PWM

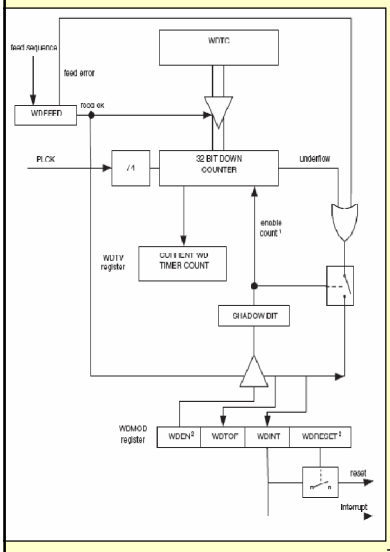
- Salidas de comparación pasadas por biestables



PWM ejemplo

```
void main(void)
{
   PINSEL0 |= 0x00028000;  //Enable pin 0.7   as PWM2
   PWMPR  = 0x00000001;    //Load prescaler

   PWMPCR = 0x0000404;      //PWM channel 2 double edge control, output enabled
   PWMMCR = 0x00000003; //On match with timer reset the counter
   PWMMR0 = 0x00000010; //set cycle rate to sixteen ticks
   PWMMR1 = 0x00000002; //set rising edge of PWM2 to 2 ticks
   PWMMR2 = 0x00000008; //set falling edge of PWM2 to 8 ticks
   PWMLER = 0x00000007; //enable shadow latch for match 0 - 2
   PWMEMR = 0x00000280; //Match 1 and Match 2 outputs set high
   PWMTCR = 0x00000002; //Reset counter and prescaler
   PWMTCR = 0x00000009; //enable counter and PWM, release counter from reset

   while(1)          // main loop
   {
     //........      //Modulate PWMMR1 and PWMMR2
   }
}
```


WATCHDOG



- Periodos configurables de hasta 1145 s (15 MHz de PCLK)

$(T_{PCLK} \times 256 \times 4)$ to $(T_{PCLK} \times 2^{32} \times 4)$

- Modos de funcionamiento:
  - Reset
  - Interrupción.
- Se habilita por software, pero sólo se puede deshabilitar tras un reset o una interrupción por desbordamiento del watchdog.

Watchdog: Ejemplo

```
#define WDEN                0x00000001
#define WDRESET                     0x00000002
#define WDTOF               0x00000004
#define WDINT               0x00000008
#define WDT_FEED_VALUE      0x003FFFFF
void WDTInit( void )
{
      WDTC = WDT_FEED_VALUE;
      WDMOD = WDEN | WDRESET;
      WDFEED = 0xAA;                /* Feeding sequence */
      WDFEED = 0x55;
}
void WDTFeed( void )
{
      WDFEED = 0xAA;                /* Feeding sequence */
      WDFEED = 0x55;
}
```


SCB: Mapeo de la memoria (MEMMAP)

Address	Name	Description	Access	Reset Value*
External Interrupts				
0xE01FC140	EXTINT	External interrupt flag register.	R/W	0
0xE01FC144	EXTWAKE	External interrupt wakeup register.	R/W	0
Memory Mapping Control				
0xE01FC040	MEMMAP	Memory mapping control	R/W	0
Phase Locked Loop				
0xE01FC000	PLLCON	PLL control register.	R/W	0
0xE01FC004	PLLCFG	PLL configuration register.	R/W	0
0xE01FC008	PLLSTAT	PLL status register.	RO	0
0xE01FC00C	PLLFEED	PLL feed register.	WO	NA
Power Control				
0xE01FC0C0	PCON	Power control register.	R/W	0
0xE01FC0C4	PCONP	Power control for peripherals.	R/W	0x3BE
VPB Divider				
0xE01FC100	VPBDIV	VPB divider control.	R/W	0

Address	Name	Description	Access
0xE01FC040	MEMMAP	Memory mapping control. Selects whether the ARM interrupt vectors are read from the Flash Boot Block, User Flash or RAM.	R/W

SCB: Mapeo de la memoria (MEMMAP)

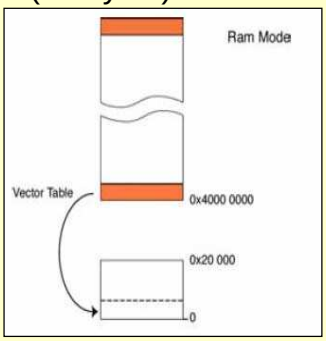
- Permite modificar el mapa de memoria de los vectores de interrupción, para manejar las interrupciones en los diferentes modos de funcionamiento

Table 19. Memory Mapping control register (MEMMAP - address 0xE01F C040) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAP	00	Boot Loader Mode. Interrupt vectors are re-mapped to Boot Block.	00
		01	User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash.	
		10	User RAM Mode. Interrupt vectors are re-mapped to Static RAM.	
		11	Reserved. Do not use this option.	
			Warning: Improper setting of this value may result in incorrect operation of the device.	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

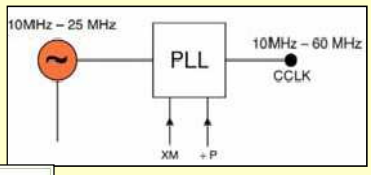
SCB: Mapeo de la memoria (MEMMAP)

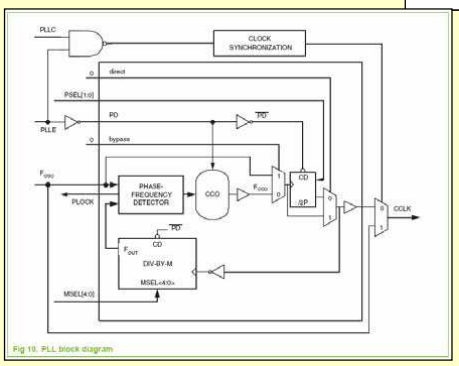
- Con objeto de depurar el código en RAM el sistema permite a través de MEMMAP mapear la tabla de vectores a partir de la dirección 0x4000.0000 siendo una imagen de la posición 0x00 a 0x3f (64 bytes) tras el reset.



PLL: Phase Locked Loop

- Permite por software seleccionar la frecuencia de la CPU.



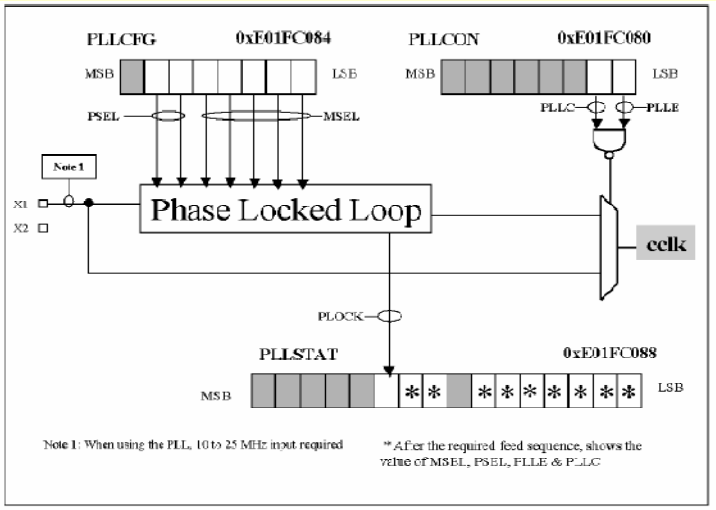


PLL - Registros

Table 20. PLL registers

	Name	Description	Access	Reset value[1]	Address
0xE01FC080	PLLCON	PLL Control Register. Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	0xE01F C080
0xE01FC084	PLLCFG	PLL Configuration Register. Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	0xE01F C084
0xE01FC088	PLLSTAT	PLL Status Register. Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling the PLL, as well as the status of the PLL.	RO	0	0xE01F C088
0xE01FC08C	PLLFEED	PLL Feed Register. This register enables loading of the PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO	NA	0xE01F C08C

PLL: Diagrama de bloques lógico



PLL: Registros de config. y control (PLLCFG y PLLCON)

0xE01FC084

Table 22. PLL Configuration register (PLLCFG - address 0xE01F C084) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. **Note:** For details on selecting the right value for MSEL see Section 4–8.9 "PLL frequency calculation" on page 38.	0
6:5	PSEL	PLL Divider value. Supplies the value "P" in the PLL frequency calculations. **Note:** For details on selecting the right value for PSEL see Section 4–8.9 "PLL frequency calculation" on page 38.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

0xE01FC080

Table 21. PLL Control register (PLLCON - address 0xE01F C080) bit description

Bit	Symbol	Description	Reset value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 4–23.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the microcontroller. Otherwise, the oscillator clock is used directly by the microcontroller. See PLLSTAT register, Table 4–23.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL: Registros de control y estado (PLLFEED y PLLSTAT)

0xE01FC08C

Table 25. PLL Feed register (PLLFEED - address 0xE01F C08C) bit description

Bit	Symbol	Description	Reset value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for PLL configuration and control register changes to take effect.	0x00

0xE01FC088

Table 23. PLL Status register (PLLSTAT - address 0xE01F C088) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL	Read-back for the PLL Multiplier value. This is the value currently used by the PLL.	0
6:5	PSEL	Read-back for the PLL Divider value. This is the value currently used by the PLL.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power-down mode is activated.	0
9	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the microcontroller. When either PLLC or PLLE is zero, the PLL is bypassed and the oscillator clock is used directly by the microcontroller. This bit is automatically cleared when Power-down mode is activated.	0
10	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL: Programación (I)

- El registro PLLCFG contiene las constantes multiplicadora y divisora que permiten obtener el valor de la frecuencia de CPU.
  - M. Factor multiplicador del oscilador (1-31).
  - P. Factor divisor del Oscilador Controlador por Corriente.
    - ***(valores 1, 2, 4, 8).***
  - **Importante:** Los valores en los registros son M-1 y P-1.
- La frecuencia del procesador está dada por:
  - cclk = M·Fosc ó cclk = Fcco/(2·P)
  - Fcco = CCLK·2·P → Fcco = Fosc·M·2·P
    - Fosc. Frecuencia del oscilador (10 MHz a 30 MHz).
    - Fcco. Frecuencia del Oscilador Controlador por Corriente del PLL. Ha de estar comprendida entre 156 MHz y 320 MHz.
    - cclk. Frecuencia de salida del PLL (10 MHz a 60 MHz).

PLL: Programación (II)

Table 1: PLL labels

Label	Name	Min	Max
f_{osc}	oscillator frequency (slave)	1 Mhz	50 Mhz
	oscillator frequency (oscillation mode)	1 Mhz	30 Mhz
	oscillator frequency (PLL Mode)	10 Mhz	25 Mhz
Fcco	current controlled oscillator frequency	156 Mhz	320 Mhz
cclk	CPU clock	1Mhz	60 Mhz
MSEL	PLL multiplier value	0x0	0x5
PSEL	PLL divider	0x0	0x3

Table 2: PLL multiplier values

MSEL hex	Multiplier value decimal
0	1
1	2
2	3
3	4
4	5
5	6

Table 3: PLL divider values

PSEL Hex	Divider Values decimal
0	1
1	2
2	4
3	8

PLL: Programación (III)

Table 4: Calculated operating values for 10 MHz to 13 MHz

f_{osc}	cclk	MSEL (hex)	PSEL (hex)				cclk*2	PSEL (decimal)			
			0	1	2	3		1	2	4	8
10	10	0				x	20	20	40	80	160
10	20	1			x	x	40	40	80	160	320
10	30	2			x		60	60	120	240	480
10	40	3		x	x		80	80	160	320	640
10	50	4		x			100	100	200	400	800
10	60	5		x			120	120	240	480	960
11	11	0				x	22	22	44	88	176
11	22	1			x		44	44	88	176	352
11	33	2			x		66	66	132	264	528
11	44	3		x			88	88	176	352	704
11	55	4		x			110	110	220	440	880
12	12	0				x	24	24	48	96	192
12	24	1			x		48	48	96	192	384
12	36	2			x		72	72	144	288	576
12	48	3		x			96	96	192	384	768
12	60	4		x			120	120	240	480	960
13	13	0				x	26	26	52	104	208
13	26	1			x		52	52	104	208	416
13	39	2		x	x		78	78	156	312	624
13	52	3		x			104	104	208	416	832

Table 5: Calculated operating values for 14 MHz to 20 MHz

f_{osc}	cclk	MSEL (hex)	PSEL (hex)				cclk*2	PSEL (decimal)			
			0	1	2	3		1	2	4	8
14	14	0				x	28	28	56	112	224
14	28	1			x		56	56	112	224	448
14	42	2		x			84	84	168	336	672
14	56	3		x			112	112	224	448	896
15	15	0				x	30	30	60	120	240
15	30	1			x		60	60	120	240	480
15	45	2		x			90	90	180	360	720
15	60	3		x			120	120	240	480	960
16	16	0				x	32	32	64	128	256
16	32	1			x		64	64	128	256	512
16	48	2		x			96	96	192	384	768
17	17	0				x	34	34	68	136	272
17	34	1			x		68	68	136	272	544
17	51	2		x			102	102	204	408	816
18	18	0				x	36	36	72	144	288
18	36	1			x		72	72	144	288	576
18	54	2		x			108	108	216	432	864
19	19	0				x	38	38	76	152	304
19	38	1			x		76	76	152	304	608
19	57	2		x			114	114	228	456	912
20	20	0			x	x	40	40	80	160	320
20	40	1		x	x		80	80	160	320	640
20	60	2		x			120	120	240	480	960

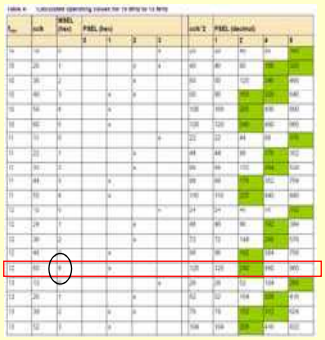
Nota de aplicación: AN10331

PLL: Programación (Ejemplo)

- XTAL = 12 Mhz □□ Fcpu = 60 Mhz
- 1º) M =Fcclk/Fosc = 5
- 2º) Comprobamos límites del CCO:

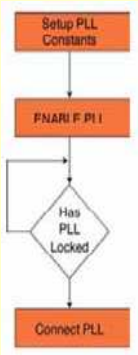
Fcco = Fosc·M·2·P = 120·P

  - Si P=1 □□ Fcco = 120 Mhz < Fmin !!
  - Si P=2 □□ Fcco = 240 Mhz (válido)



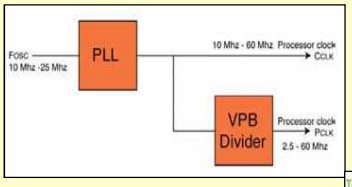
PLL: Programación (Ejemplo)

```
#define PLL_PLLE 1 //PLL enable (1)or disable(0)
#define PLL_PLLC 1 //PLL connect(1) or disconnect(0)
#define PLL_M 4 //PLL Multiplier value
#define PLL_P 1//PLL divider value: p
Void PLL_config(void)
{
PLLCFG=(PLL_M ) | ((PLL_P) << 5);
PLLCON=PLL_PLLE;
PLLFEED = 0xAA;
PLLFEED = 0x55;
while((PLLSTAT & (1 << 10)) == 0); // Wait for PLL lock
PLLCON=PLL_PLLE|PLL_PLLC<<1; //connect PLL
PLLFEED = 0xAA;
PLLFEED = 0x55;
```



SCB: Divisor de frecuencia APB

- Determina la relación entre la frecuencia de la CPU (Cclk) y la frecuencia de bus de los periféricos (Pclk).



Cclk/4 →Reset

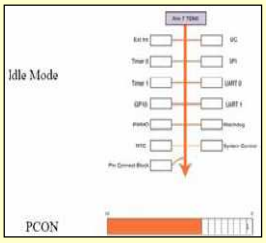
Table 34. APB Divider register (APBDIV - address 0xE01F C100) bit description

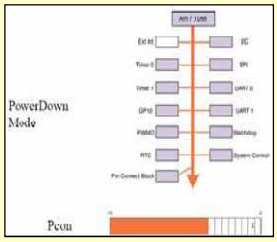
Bit	Symbol	Value	Description	Reset value
1:0	APBDIV	00	APB bus clock is one fourth of the processor clock.	00
		01	APB bus clock is the same as the processor clock.	
		10	APB bus clock is one half of the processor clock.	
		11	Reserved. If this value is written to the APBDIV register, it has no effect (the previous setting is retained).	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Modos de bajo consumo (I)

Soporta 2 modos de bajo consumo:

- **IDLE.**
  - La ejecución queda suspendida (CPU parada) hasta que se produce un Reset o una interrupción externa.
  - Los periféricos siguen funcionando y las rutinas de interrupción también.
  - El consumo se reduce considerablemente y depende de los periféricos que estén activos y de la frecuencia del bus.
- **POWER-DOWN.**
  - El oscilador se desconecta de la CPU y de los periféricos.
  - Los datos se mantienen en la RAM y en los registros internos.
  - Para salir de este modo es necesario un Reset o determinada fuente de interrupción que no haga uso del reloj.
  - El consumo de reduce prácticamente a cero.





Modos de bajo consumo (II)

Table 4: Core power consumption
Typical numbers of LPC2138 core power consumption

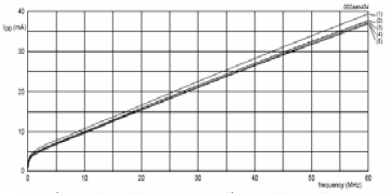
Core mode	Memory - State	Offset @ 1 MHz [mA]	Offset @ 10 MHz [mA]	Additional power consumption [mA/MHz]	Power consumption @ 60 MHz [mA]
Active[1]	Flash - Thumb	~5.6	~10.3	~0.52	~36.3
Active[1]	Flash - ARM	~6.4	~11.6	~0.58	~40.6
Active[1]	SRAM[2]	~5.5	~11.9	~0.71	~47.4
Idle	-	~1.3	-	~0.17	~11.3

Table 5: Peripheral power consumption
Typical numbers of LPC2138 peripheral power consumption

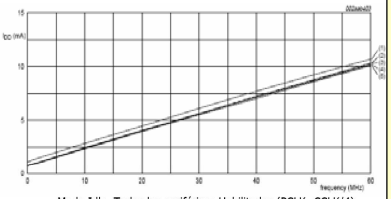
Peripheral	Configuration	Internal PCLK divider	1 MHz < PCLK < 10 MHz [uA/MHz]	10 MHz < PCLK < 20 MHz [uA/MHz]	PCLK > 20 MHz [uA/MHz]
Timer/PWM[1]	Continuously counting	1	24.0	24.0	24.0
ADC[2][3][4]	10 channel burst mode	-	15.8	10.4	8.2
UART	Receiver mode	16	30.0	30.0	30.0
I2C	Master transmitter mode	-	13.0	10.2	6.3
SPI	Slave receiver mode	8	14.9	8.5	5.0
SSP	Slave receiver mode	2	14.9	8.5	5.0

[1] Pulse-width-modulator (PWM): Only timer registers are active
[2] 10 channels ADC with a sample rate of 400 kHz
[3] Value for only PCLK at 4Mhz and 8 MHz
[4] Running on maximum output clock rate (400 kHz)

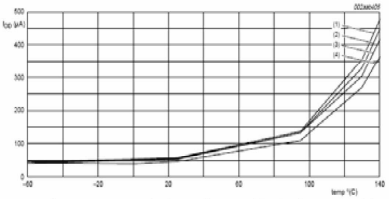
Modos de bajo consumo (I_{DD})



Ejecución normal en Flash. Todos los periféricos Habilitados (PCLK=CCLK/4)



Modo Idle. Todos los periféricos Habilitados (PCLK=CCLK/4)



Modo Power-Down. Todos los periféricos Habilitados (PCLK=CCLK/4)

Power Mgt. Modes	Core	Peripherals	RAM	RTC	Wake-up	Current
Idle	Stopped	Active	Active	Active	Any Interrupts	< 5mA
Power Down	Stopped	Stopped	Inactive	Inactive	Ext. Interr., Rst	< 10µA
Hibernate	Power off	Power off	Power on	Active	Ext. Interr., Rst	< 7µA
Deep Power Down	Power off	Power off	Power off	Stopped (osc. off)	Ext. Interr., Rst	< 5µA

Ejemplo modos de bajo consumo

Modos de bajo consumo (Registros)

Table 29. Power control registers

Name	Description	Access	Reset value[1]	Address
PCON	Power Control Register. This register contains control bits that enable the two reduced power operating modes of the microcontroller. See Table 4–30.	R/W	0x00	0xE01F C0C0
PCONP	Power Control for Peripherals Register. This register contains control bits that enable and disable individual peripheral functions, Allowing elimination of power consumption by peripherals that are not needed.	R/W	0x0008 17BE	0xE01F C0C4

Modos de bajo consumo (PCON y PCONP)

Table 30. Power Control register (PCON - address 0xE01F C0C0) bit description

Bit	Symbol	Value	Description	Reset value
0	IDL		Idle mode control.	0
		0	Idle mode is off.	
		1	The processor clock is stopped, while on-chip peripherals remain active. Any enabled interrupt from a peripheral or an external interrupt source will cause the processor to resume execution.	
1	PD		Power-down mode control.	0
		0	Power-down mode is off.	
		1	The oscillator and all on-chip clocks are stopped. A wakeup condition from an external interrupt can cause the oscillator to restart, the PD bit to be cleared, and the processor to resume execution.	
2	BODPDM		Brown Out Power-down Mode.	0
		0	Brown Out Detection (BOD) remains operative during Power-down mode, and its Reset can release the microcontroller from Power-down mode[1].	
		1	The BOD circuitry will go into power down mode when PD = 1, resulting in a further reduction in power. In this case the BOD can not be used as a wakeup source from Power Down mode.	
3	BOGD[2]		Brown Out Global Disable.	0
		0	The BOD circuitry is enabled.	
		1	The BOD is fully disabled at all times, consuming no power.	
4	BORD[2]		Brown Out Reset Disable.	0
		0	The reset is enabled. The first stage of low voltage detection (2.9 V) Brown Out interrupt is not affected.	
		1	The second stage of low voltage detection (2.6 V) will not cause a chip reset.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 31. Power Control for Peripherals register (PCONP - address 0xE01F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	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	PCPWM0	PWM0 power/clock control bit.	1
6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	PCI2C0	The I2C0 interface power/clock control bit.	1
8	PCSPI0	The SPI0 interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSPI1	The SSP interface power/clock control bit.	1
11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12	PCAD0	A/D converter 0 (ADC0) power/clock control bit. Note: Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	1
18:13	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19	PCI2C1	The I2C1 interface power/clock control bit.	1
20	PCAD1	A/D converter 1 (ADC1) power/clock control bit. Note: Clear the PDN bit in the AD1CR before clearing this bit, and set this bit before setting PDN.	0
31:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

SBC: Reset

- Causas de Reset:
  - Pin /RESET. (nivel bajo).
  - Watchdog.
  - Brown-Out Detect.
- La activación del Reset causa el arranque de Wakeup Timer, esperando a:
  - 1. La entrada externa retorne a nivel alto.
  - 2. El oscilador comience a funcionar.
  - 3. Transcurran un número de ciclos de reloj de espera.
  - 4. El circuito interno se inicialice correctamente.
- Después del POWER-ON el pin de reset necesita mantenerse a nivel bajo al menos durante 10 ms.
  - Si el oscilador ya está funcionando, y la señal es estable en X1, tan sólo es necesario que se mantenga 300 ns.
- Una serie de pines se muestrean durante el reset:
  - P1.20 → TRACESYNC
  - P1.26 → RTCK
  - P0.14 → Modo **I**n-**S**ystem-**P**rogramming

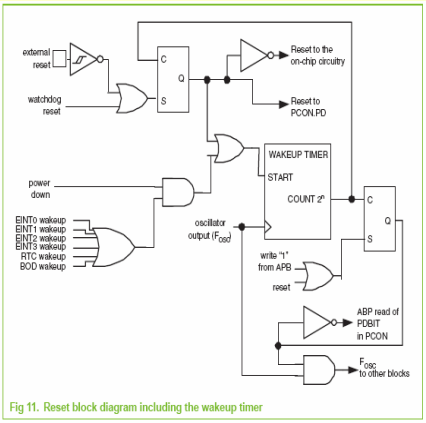


Fig 11. Reset block diagram including the wakeup timer

SBC: Reset (identificación)

- RSIR. Registro de Identificación de la causa del Reset.

Table 32. Reset Source identificator Register (RSIR - address 0xE01F C180) bit description

Bit	Symbol	Description	Reset value
0	POR	Assertion of the POR signal sets this bit, and clears all of the other bits in this register. But if another Reset signal (e.g., External Reset) remains asserted after the POR signal is negated, then its bit is set. This bit is not affected by any of the other sources of Reset.	see text
1	EXTR	Assertion of the $\overline{\text{RESET}}$ signal sets this bit. This bit is cleared by POR, but is not affected by WDT or BOD reset.	see text
2	WDTR	This bit is set when the Watchdog Timer times out and the WDTRESET bit in the Watchdog Mode Register is 1. It is cleared by any of the other sources of Reset.	see text
3	BODR	This bit is set when the 3.3 V power reaches a level below 2.6 V. If the V_{DD} voltage dips from 3.3 V to 2.5 V and backs up, the BODR bit will be set to 1. Also, if the V_{DD} voltage rises continuously from below 1 V to a level above 2.6 V, the BODR will be set to 1, too. This bit is not affected by External Reset nor Watchdog Reset. **Note:** only in case a reset occurs and the POR = 0, the BODR bit indicates if the V_{DD} voltage was below 2.6 V or not.	seet text
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

SBC: Wakeup Timer

- Asegura que el oscilador y otras funciones analógicas del sistema funcionen correctamente antes de que la CPU comience la ejecución del programa.
  - Importante en:
    - Power-on.
    - Todos los tipos de reset.
    - Aquellas funciones que permiten desconectar determinados recursos tras el POWER-DOWN.
- Una vez que es detectado el reloj, espera 4096 ciclos para habilitar los circuitos internos que han de inicializarse

SBC: Brown-Out Detect

- Monitorea la alimentación (VDD) del chip.
- Interrumpe a la CPU si VDD cae por debajo de 2.9 V.
- Resetea a la CPU si VDD cae por debajo de 2.6 V.

Configuración del Sistema: Ej. programación

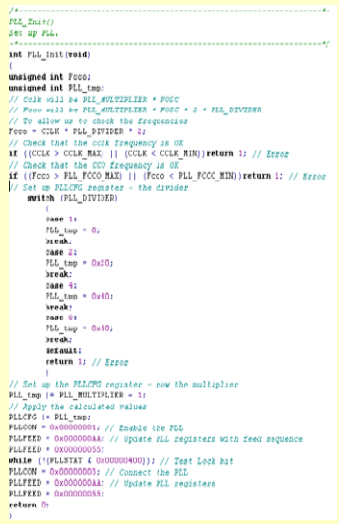
```
/* XTAL=12Mhz ? Fcpu=60Mhz */
#define PLL_PLLE 1 //PLL enable (1)or disable(0)
#define PLL_PLLC 1 //PLL connect(1) or disconnect(0)
#define PLL_M 5 //PLL Multiplier value
#define PLL_P 2 //PLL divider value: p
#define VPB_DIVIDER 0 //the divider of VPB

/* System Initialization */
void InitLPC2000(void) {
WDMOD=0; //disable WDT
VICIntEnClr=0xffffffff; //disable all interrupts
VICVectAddr=0;
VICIntSelect=0;

/* PLL configuration */
if(PLL_PLLE){
PLLCFG=(PLL_M-1) | ((PLL_P-1) << 5);
PLLCON=PLL_PLLE;
PLLFEED = 0xAA;
PLLFEED = 0x55;
while((PLLSTAT & (1 << 10)) == 0); // Wait for PLL lock
PLLCON=PLL_PLLE|PLL_PLLC<<1; //connect PLL
PLLFEED = 0xaa;
PLLFEED = 0x55;
}
VPBDIV=VPB_DIVIDER; //peripheral clock config

/* MemRemap Config */
#ifdef __Ram_Mode
MEMMAP = 0x2; //remap to 0x40000000
#endif
#ifdef __Flash_Mode
MEMMAP = 0x1; //remap to 0x0
#endif
#ifdef __ExtMem_mode
MEMMAP = 0x3; //remap to 0x80000000, only for lpc22xx
#endif
}
```


Configuración del Sistema: Ej. programación

```
// Device header (from Keil)
#include <lpc21xx.h>
// Oscillator / resonator frequency (in Hz)
// e.g. (10000000UL) when using 10 MHz oscillator
#define FOSC (12000000UL)
// Between 1 and 32
#define PLL_MULTIPLIER (5U)
// 1, 2, 4 or 8
#define PLL_DIVIDER (2U)
// 1, 2 or 4
#define VPB_DIVIDER (1U)
// CPU clock
#define CCLK (FOSC * PLL_MULTIPLIER)
// Peripheral clock
#define PCLK (CCLK / VPB_DIVIDER)
#define PLL_FCCO_MIN (156000000UL)
#define PLL_FCCO_MAX (320000000UL)
#define CCLK_MIN (10000000UL)
#define CCLK_MAX (60000000UL)
// Function prototypes
void System_Init(void);
int PLL_Init(void);
int VPB_Init(void);
void MAM_Init(void);
void Set_Interrupt_Mapping(void);
```

```
// ------ Private constants ---------------------------------------
// Interrupt mapping set through the "target" settings in the IDE
#ifndef RAM
#define MAP 0x01
#else
#define MAP 0x02
#endif
/*----------------------------------------------------------------------*-
System_Init()
Configures:
- PLL
- VPB divider
- Memory accelerator module
- Interrupt mapping
-*----------------------------------------------------------------------*/
void System_Init(void)
{
// Set up the PLL
if (PLL_Init() != 0)
{
while(1); // PLL error - stop
}
// Set up the VP bus
if (VPB_Init() != 0)
{
while(1); // VPB divider error - stop
}
// Set up the memory accelerator module
MAM_Init();
// Control interrupt mapping
Set_Interrupt_Mapping();
}
```

Configuración del Sistema: Ej. programación



Configuración del Sistema: Ej. programación

```
/*--------------------------------------------------------------------*-
MAM_Init()
Set up the memory accelerator module.
NOTE: Here we DISABLE the MAM, for maximum predictability.
Adapt as needed for your application.
-*--------------------------------------------------------------------*/
void MAM_Init(void)
{
// Turn off MAM
MAMCR = 0;
}
/*--------------------------------------------------------------------*-
Set_Interrupt_Mapping()
Remaps interrupts to RAM or Flash memory, as required.
For Flash, MAP = 0x01
For RAM, MAP = 0x02
Here, value is set through Keil uVision
(depending on target built).
-*--------------------------------------------------------------------*/
void Set_Interrupt_Mapping(void)
{
MEMMAP = MAP;
}
```

RTC: Real Time Clock

- Reloj-calendario capaz de medir tiempo cuando la CPU está funcionando, y opcionalmente en el modo Power-Down, e incluso con la alimentación V_{DD} desconectada.
- Validez de la fecha hasta el 2099.
  - La frecuencia de oscilación se extrae de:
  - La frecuencia del bus de los periféricos (pclk), es decir, de la salida del divisor de frecuencia o prescaler de los periféricos (APB).
- Un cristal externo de 32.768 KHz.
- La alimentación se suministra a través de un pin dedicado (V_{BAT}) que puede conectarse a la alimentación principal (V_{DD}), o a una batería externa o pila de litio de 3 V.

RTC: Diagrama de bloques

- **Generador de reloj.**
  - Frecuencia de entrada de 32768 Hz (CLK32k) derivada de:
    - Un cristal externo de 32,768 KHz.
    - La frecuencia de bus de los periféricos (Pclk), mediante un prescaler interno configurable mediante los registros PREINT y PREFRAC.
- **Registros contadores de tiempo:**
  - Segundos, minutos, hora, día, día de la semana, mes y año.
- **Registros de alarma.**
- **Sistema de gestión de interrupción:**
  - Periódicas cada 1/32768 seg.
  - Periódicas cada segundo, minuto, hora, …
  - A una hora-fecha determinada (modo alarma).
    - Registro de máscara de la alarma.

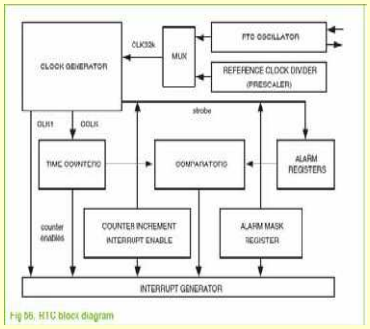
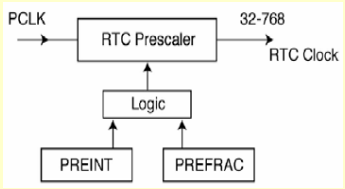


Fig 86. RTC block diagram

RTC: Prescaler interno

- La frecuencia de funcionamiento del RTC es de 32768 Hz.
- Si la fuente de reloj es Pclk, es necesario programar el prescaler interno para obtener con exactitud la frecuencia de trabajo.
- Los registros PREINT y PREFRAC constituyen la parte entera y decimal del divisor (prescaler).
  - PREINT = (int)(pclk/32768)-1
  - PREFRAC = pclk – ((PREINT+1) x 32768)




```
PREINT      = 0x00000392;       //Set RTC prescaler for 30.000 MHz Pclk
PREFRAC = 0x00004380;
CCR    = 0x00000001;       //Start the RTC
```


RTC: Prescaler interno (esquema)

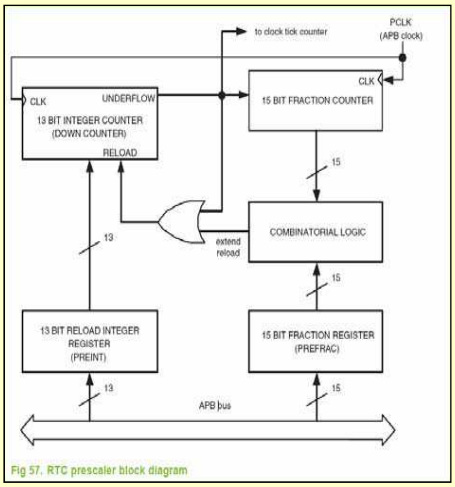
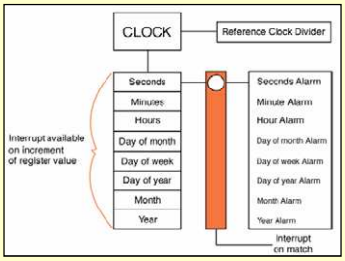


Fig 57. RTC prescaler block diagram

RTC: Alarmas e interrupciones

- **Interrupción por incremento del valor de un registro de tiempo:**
  - Segundos, Minutos, Horas, Día del mes, Día de la semana, Día del año o Año.
  - El registro **CIIR** permite seleccionar (máscara local) qué registros interrumpirán cuando se incrementen.
- **Interrupción por alarma programada:**
  - Coincidencia de la hora-fecha actual con la fijada en los registros de alarma.
  - El registro **AMR** permite seleccionar (máscara local) qué registros se compararán para generar la alarma.



RTC: Registros de control

Table 169. Real Time Clock (RTC) register map

Name	Size	Description	Access	Reset value[1]	Address
ILR	2	Interrupt Location Register	R/W	*	0xE002 4000
CTC	15	Clock Tick Counter	RO	*	0xE002 4004
CCR	4	Clock Control Register	R/W	*	0xE002 4008
CIIR	8	Counter Increment Interrupt Register	R/W	*	0xE002 400C
AMR	8	Alarm Mask Register	R/W	*	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	*	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	*	0xE002 4018
CTIME2	32	Consolidated Time Register 2	RO	*	0xE002 401C
SEC	6	Seconds Counter	R/W	*	0xE002 4020
MIN	6	Minutes Register	R/W	*	0xE002 4024
HOUR	5	Hours Register	R/W	*	0xE002 4028
DOM	5	Day of Month Register	R/W	*	0xE002 402C
DOW	3	Day of Week Register	R/W	*	0xE002 4030
DOY	9	Day of Year Register	R/W	*	0xE002 4034
MONTH	4	Months Register	R/W	*	0xE002 4038
YEAR	12	Years Register	R/W	*	0xE002 403C
ALSEC	6	Alarm value for Seconds	R/W	*	0xE002 4060
ALMIN	6	Alarm value for Minutes	R/W	*	0xE002 4064
ALHOUR	5	Alarm value for Seconds	R/W	*	0xE002 4068
ALDOM	5	Alarm value for Day of Month	R/W	*	0xE002 406C
ALDOW	3	Alarm value for Day of Week	R/W	*	0xE002 4070
ALDOY	9	Alarm value for Day of Year	R/W	*	0xE002 4074
ALMON	4	Alarm value for Months	R/W	*	0xE002 4078
ALYEAR	12	Alarm value for Year	R/W	*	0xE002 407C
PREINT	13	Prescaler value, integer portion	R/W	0	0xE002 4080
PREFRAC	15	Prescaler value, integer portion	R/W	0	0xE002 4084

RTC: Registros de propósito general

Table 200. Miscellaneous registers

Name	Size	Description	Access	Address
IlR	7	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	R/W	0xE002 4000
CTC	15	Clock Tick Counter. Value from the clock divider	RO	0xE002 4004
CCR	4	Clock Control Register. Controls the function of the clock divider.	R/W	0xE002 4008
CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	R/W	0xE002 400C
AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	R/W	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	0xE002 4018
CTIME2	32	Consolidated Time Register 2	RO	0xE002 401C

Permiten **leer** el reloj-calendario con sólo 3 lecturas !!

RTC: Registros de tiempo (sólo lectura)

Table 206: Consolidated Time register 0 (CTIME0 - address 0xE002 4014) bit description

Bit	Symbol	Description	Reset value
5:0	Seconds	Seconds value in the range of 0 to 59	NA
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
13:8	Minutes	Minutes value in the range of 0 to 59	NA
15:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
20:16	Hours	Hours value in the range of 0 to 23	NA
23:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	Day Of Week	Day of week value in the range of 0 to 6	NA

Table 207: Consolidated Time register 1 (CTIME1 - address 0xE002 4018) bit description

Bit	Symbol	Description	Reset value
4:0	Day of Month	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).	NA
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	Month	Month value in the range of 1 to 12.	NA
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
27:16	Year	Year value in the range of 0 to 4095.	NA
31:28	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 208: Consolidated Time register 2 (CTIME2 - address 0xE002 401C) bit description

Bit	Symbol	Description	Reset value
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).	NA
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

RTC: Registros contadores de tiempo (R/W)

Table 209. Time counter relationships and values

Counter	Size	Enabled by	Minimum value	Maximum value
Second	6	Clk1 (see Figure 18–56)	0	59
Minute	6	Second	0	59
Hour	5	Minute	0	23
Day of Month	5	Hour	1	28, 29, 30 or 31
Day of Week	3	Hour	0	6
Day of Year	9	Hour	1	365 or 366 (for leap year)
Month	4	Day of Month	1	12
Year	12	Month or day of Year	0	4095

Table 210. Time counter registers

Name	Size	Description	Access	Address
SEC	6	Seconds value in the range of 0 to 59	R/W	0xE002 4020
MIN	6	Minutes value in the range of 0 to 59	R/W	0xE002 4024
HOUR	5	Hours value in the range of 0 to 23	R/W	0xE002 4028
DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).[1]	R/W	0xE002 402C
DOW	3	Day of week value in the range of 0 to 6[1]	R/W	0xE002 4030
DOY	9	Day of year value in the range of 1 to 365 (366 for leap years)[1]	R/W	0xE002 4034
MONTH	4	Month value in the range of 1 to 12	R/W	0xE002 4038
YEAR	12	Year value in the range of 0 to 4095	R/W	0xE002 403C

RTC: Registros ILR, CTCR, CCR

Table 201: Interrupt Location Register (ILR - address 0xE002 4000) bit description

Bit	Symbol	Description	Reset value
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.	NA
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	NA
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 202: Clock Tick Counter Register (CTCR - address 0xE002 4004) bit description

Bit	Symbol	Description	Reset value
14:0	Clock Tick Counter	Prior to the Seconds counter, the CTC counts 32,768 clocks per second. Due to the RTC Prescaler, these 32,768 time increments may not all be of the same duration. Refer to the Section 18–6 "Reference clock divider (prescaler)" on page 228 for details.	NA
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 203: Clock Control Register (CCR - address 0xE002 4008) bit description

Bit	Symbol	Description	Reset value
0	CLKEN	Clock Enable. When this bit is a one the time counters are enabled. When it is a zero, they are disabled so that they may be initialized.	NA
1	CTCRST	CTC Reset. When one, the elements in the Clock Tick Counter are reset. The elements remain reset until CCR[1] is changed to zero.	NA
3:2	CTTEST	Test Enable. These bits should always be zero during normal operation.	NA
4	CLKSRC	If this bit is 0, the Clock Tick Counter takes its clock from the Prescaler, as on earlier devices in the Philips Embedded ARM family. If this bit is 1, the CTC takes its clock from the 32 kHz oscillator that's connected to the RTCX1 and RTCX2 pins (see Section 18–7 "RTC external 32 kHz oscillator component selection" for hardware details).	NA
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

RTC: Registros CIIR y AMR

Table 204: Counter Increment Interrupt Register (CIIR - address 0xE002 400C) bit description

Bit	Symbol	Description	Reset value
0	IMSEC	When 1, an increment of the Second value generates an interrupt.	NA
1	IMMIN	When 1, an increment of the Minute value generates an interrupt.	NA
2	IMHOUR	When 1, an increment of the Hour value generates an interrupt.	NA
3	IMDOM	When 1, an increment of the Day of Month value generates an interrupt.	NA
4	IMDOW	When 1, an increment of the Day of Week value generates an interrupt.	NA
5	IMDOY	When 1, an increment of the Day of Year value generates an interrupt.	NA
6	IMMON	When 1, an increment of the Month value generates an interrupt.	NA

Table 205: Alarm Mask Register (AMR - address 0xE002 4010) bit description

Bit	Symbol	Description	Reset value
0	AMRSEC	When 1, the Second value is not compared for the alarm.	NA
1	AMRMIN	When 1, the Minutes value is not compared for the alarm.	NA
2	AMRHOUR	When 1, the Hour value is not compared for the alarm.	NA
3	AMRDOM	When 1, the Day of Month value is not compared for the alarm.	NA
4	AMRDOW	When 1, the Day of Week value is not compared for the alarm.	NA
5	AMRDOY	When 1, the Day of Year value is not compared for the alarm.	NA
6	AMRMON	When 1, the Month value is not compared for the alarm.	NA
7	AMRYEAR	When 1, the Year value is not compared for the alarm.	NA

RTC: Oscilador externo

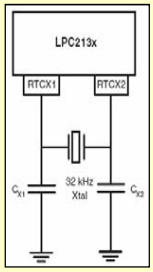


Table 216. Recommended values for the RTC external 32 kHz oscillator $C_{X1/X2}$ components

Crystal load capacitance C_L	Maximum crystal series resistance R_S	External load capacitors C_{X1}, C_{X2}
11 pF	< 100 kΩ	18 pF, 18 pF
13 pF	< 100 kΩ	22 pF, 22 pF
15 pF	< 100 kΩ	27 pF, 27 pF

RTC: Ej. Programación (I)

```
int main(void)
{
  VPBDIV = 0x00000002;
  IODIR1 = 0XU0FF0000; // set LED ports to output
  IOSET1 = 0X00020000;
  PREINT = 0x00000392; // Set RTC prescaler for 30MHz Pclk
  PREFRAC = 0x00004380;
  CIIR = 0x00000001;   // Enable seconds counter interrupt
  ALSEC = 0x00000003;  // Set alarm register for 3 seconds
  AMR = 0x000000FE;    // Enable seconds Alarm
  CCR = UX00000001;    // Start the RTC

  VICVectAddr13 = (unsigned)RTC_isr; //Set the timer ISR vector address
  VICVectCntl13 = 0x00000020;         //set channel
  VICIntEnable     = 0X00002000;      //Enable the interrupt

  while(1);

}

void RTC_isr(void)
{
  unsigned led;

  if(ILR&0x00000001)      //Test for RTC counter interrupt
  {
    led = IOPIN1;         //read the current state of the IO pins
    IOCLR1  =  led&0x00030000; //Clear the illuminated LED
    IOSET1  = ~led&0x00030000; //Set the idle LED
    ILR = 0x00000001;          //Clear the interrupt register
  }

  if(ILR & 0X00000002)
  {
    IOSET1  = 0x00100000;      //Set LED 0.7
    ILR = 0x00000002;          //clear the interrupt register
  }

  VICVectAddr = 0x00000000;          //Dummy write to signal end of interrupt
}
```


RTC: Ej. Programación (II)

```
void ISR_rtc(void) /* every second */
{

  /* Actions */

  ILR |= 1;     // Clear interrupt flag
  VICVectAddr = 0;     // Acknowledge Interrupt
  PCON = 1;     // IDLE mode
}
void init_rtc(void)
{
  ILR = 3;     // Disable 32'768 interrupt
  CCR = 0x11;     // Clock enable + 32'767Hz quartz enable
  CIIR = 0x01;     // Interupt every second
  VICVectAddr1 = (unsigned long)ISR_rtc;     // set interrupt vector in slot 1
  VICVectCntl1 = 0x0000002D;     // use it for RTC Interrupt
  VICIntEnable = 0x00002000;     // Enable RTC Interrupt
}
void set_time(void)
{
  YEAR = 2006;     // Year
  MONTH = 5;     // Month
  DOM = 23;     // Day of month
  DOY = 38;     // Day of year
  DOW = 143;     // Day of week
  HOUR = 23;     // Hours
  MIN = 14;     // Minutes
  SEC = 30;     // Seconds
}
```

RTC: Ej. Programación (III)

```
#ifndef __RTC_H
#define __RTC_H

typedef struct {
    DWORD RTC_Sec;     /* Second value - [0,59] */
    DWORD RTC_Min;     /* Minute value - [0,59] */
    DWORD RTC_Hour;    /* Hour value - [0,23] */
    DWORD RTC_Mday;    /* Day of the month value - [1,31] */
    DWORD RTC_Mon;     /* Month value - [1,12] */
    DWORD RTC_Year;    /* Year value - [0,4095] */
    DWORD RTC_Wday;    /* Day of week value - [0,6] */
    DWORD RTC_Yday;    /* Day of year value - [1,365] */
} RTCTime;

#define IMSEC       0x00000001
#define IMMIN       0x00000002
#define IMHOUR      0x00000004
#define IMDOM       0x00000008
#define IMDOW       0x00000010
#define IMDOY       0x00000020
#define IMMON       0x00000040
#define IMYEAR      0x00000080

#define AMRSEC      0x00000001  /* Alarm mask for Seconds */
#define AMRMIN      0x00000002  /* Alarm mask for Minutes */
#define AMRHOUR     0x00000004  /* Alarm mask for Hours */
#define AMRDOM      0x00000008  /* Alarm mask for Day of Month */
#define AMRDOW      0x00000010  /* Alarm mask for Day of Week */
#define AMRDOY      0x00000020  /* Alarm mask for Day of Year */
#define AMRMON      0x00000040  /* Alarm mask for Month */
#define AMRYEAR     0x00000080  /* Alarm mask for Year */

#define PREINT_RTC  0x000001C8  /* Prescaler value, integer portion,
                                   PCLK = 15Mhz */
#define PREFRAC_RTC 0x000061C0  /* Prescaler value, fraction portion,
                                   PCLK = 15Mhz */
#define ILR_RTCCIF  0x01
#define ILR_RTCALF  0x02

#define CCR_CLKEN   0x01
#define CCR_CTCRST  0x02
#define CCR_CLKSRC  0x10
```

```
extern void RTCHandler (void) __irq;
extern void RTCInit( void );
extern void RTCStart( void );
extern void RTCStop( void );
extern void RTC_CTCReset( void );
extern void RTCSetTime( RTCTime );
extern RTCTime RTCGetTime( void );
extern void RTCSetAlarm( RTCTime );
extern void RTCSetAlarmMask( DWORD AlarmMask );

#endif /* end __RTC_H */
```

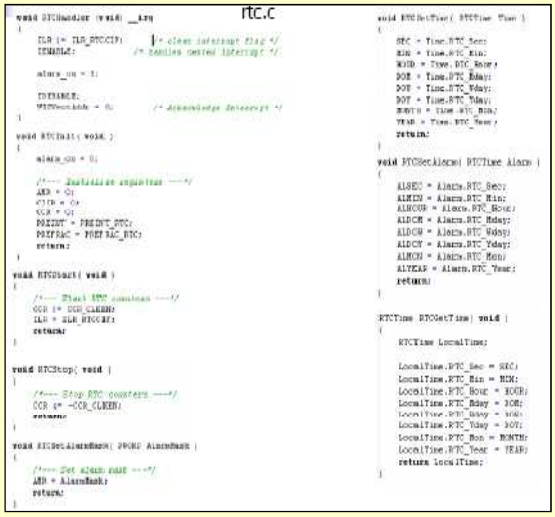

RTC: Ej. Programación (III)

```
#include "LPC214x.H"                        /* LPC21xx definitions */
#include "type.h"
#include "irq.h"
#include "timer.h"
#include "rtc.h"

extern DWORD alarm_on;
RTCTime local_time, alarm_time, current_time;

/*****************************************************************************
**  Main Function  main()
*****************************************************************************/
int main (void)
{

    init_VIC();

    /* Initialize RTC module */
    RTCInit();

    local_time.RTC_Sec = 0;
    local_time.RTC_Min = 0;
    local_time.RTC_Hour = 0;
    local_time.RTC_Mday = 0;
    local_time.RTC_Wday = 3;
    local_time.RTC_Yday = 0;        /* current date 12/08/2005 */
    local_time.RTC_Mon = 12;
    local_time.RTC_Year = 2005;
    RTCSetTime( local_time );       /* Set local time */

    alarm_time.RTC_Sec = 0;
    alarm_time.RTC_Min = 0;
    alarm_time.RTC_Hour = 0;
    alarm_time.RTC_Mday = 1;
    alarm_time.RTC_Wday = 0;
    alarm_time.RTC_Yday = 1;        /* alarm date 01/01/2006 */
    alarm_time.RTC_Mon = 1;
    alarm_time.RTC_Year = 2006;
    RTCSetAlarm( alarm_time );      /* set alarm time */

    RTCSetAlarmMask(AMRSEC|AMRMIN|AMRHOUR|AMRDOM|AMRDOW|AMRDOY|AMRMON|AMRYEAR);
    CIIR = IMMIN | IMYEAR;
    /* 2006/01/01/00:00:00 is the alarm on */

    RTCStart();
```

```
    while (1)
    {                           /* Loop forever */
    if ( alarm_on != 0 )
    {
        alarm_on = 0;
        /* Get current time when alarm is on */
        current_time = RTCGetTime();
    }
    }
    return 0;
}
```

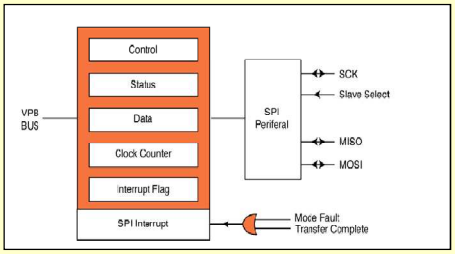
RTC: Ej. Programación (III)

rtc.c



SPI

- Puerto serie síncrono full duplex, sin gestión de bus.
- Velocidad de transmisión máxima: 1/8 PCLK.
- Transferencias de 8 a 16 bits.
- Modo maestro o modo esclavo.



SPI: reloj SCK

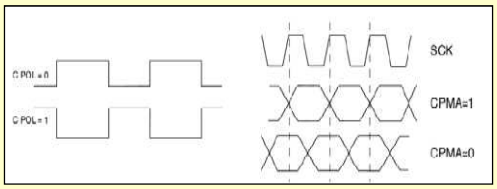
- El reloj puede configurarse en:
  - Velocidad de transmisión:

Table 159: SPI Clock Counter Register (S0SPCCR - address 0xE002 000C) bit description

Bit	Symbol	Description	Reset value
7:0	Counter	SPI0 Clock counter setting.	0x00

The SPI0 rate may be calculated as: PCLK / SPCCR0 value. The PCLK rate is CCLK /APB divider rate as determined by the APBDIV register contents.

    - El valor mínimo de este registro debe ser 8.
  - Polaridad y flanco:



SPI: línea SSEL

SSEL0	Input	**Slave Select.** The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control. On the LPC213x (unlike earlier Philips ARM devices) the SSEL0 pin can be used for a different function when the SPI0 interface is only used in Master mode. For example, pin hosting the SSEL0 function can be configured as an output digital GPIO pin and used to select one of the SPI0 slaves.

- La activación de la línea SSEL debe realizarla el maestro, usando un pin de un puerto, como salida. Los instantes de activación dependerán del tipo de esclavo.

SPI: reloj CLK

- CPOL y CPH deciden:
  - Cuándo se envía el primer bit.
  - Cuándo se envían los demás.
  - Cuándo se muestrean los datos.

Table 153. SPI data to clock phase relationship

CPOL	CPHA	Firsta data driven	Other data driven	Data sampled
0	0	Prior to first SCK rising edge	SCK falling edge	SCK rising edge
0	1	First SCK rising edge	SCK rising edge	SCK falling edge
1	0	Prior to first SCK falling edge	SCK rising edge	SCK falling edge
1	1	First SCK falling edge	SCK falling edge	SCK rising edge

SPI: configuración.

- Modo maestro/esclavo.
- Reloj SCK.
- Envío de bits empezando por LSB o por MSB
- Habilitación general

Table 156: SPI Control Register (S0SPCR - address 0xE002 0000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	BitEnable	0	The SPI controller sends and receives 8 bits of data per transfer.	0
		1	The SPI controllert sends and receives the number of bits selected by bits 11:8.	
3	CPHA		Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending.	0
		0	Data is sampled on the first (leading) clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	
		1	Data is sampled on the second (trailing) clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active.	
4	CPOL		Clock polarity control.	0
		0	SCK is active high.	
		1	SCK is active low.	
5	MSTR		Master mode select.	0
		0	The SPI operates in Slave mode.	
		1	The SPI operates in Master mode.	
6	LSBF		LSB First controls which direction each byte is shifted when transferred.	0
		0	SPI data is transferred MSB (bit 7) first.	
		1	SPI data is transferred LSB (bit 0) first.	

SPI: configuración.

- Número de bits a enviar
- Habilitación de la interrupción

7	SPIE		Serial peripheral interrupt enable.	0
		0	SPI interrupts are inhibited.	
		1	A hardware interrupt is generated each time the SPIF or MODF bits are activated.	
11:8	BITS		When bit 2 of this register is 1, this field controls the number of bits per transfer:	0000
		1000	8 bits per transfer	
		1001	9 bits per transfer	
		1010	10 bits per transfer	
		1011	11 bits per transfer	
		1100	12 bits per transfer	
		1101	13 bits per transfer	
		1110	14 bits per transfer	
		1111	15 bits per transfer	
		0000	16 bits per transfer	

SPI: registros de estado e interrupción

Table 157: SPI Status Register (S0SPSR - address 0xE002 0004) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI control register.	0
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI data register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI data register. **Note:** this is not the SPI interrupt flag. This flag is found in the SPINT register.	0

Table 160: SPI Interrupt register (S0SPINT - address 0xE002 001C) bit description

Bit	Symbol	Description	Reset value
0	SPI Interrupt Flag	SPI interrupt flag. Set by the SPI interface to generate an interrupt. Cleared by writing a 1 to this bit. **Note:** this bit will be set once when SPIE = 1 and at least one of SPIF and MODF bits changes from 0 to 1. However, only when the SPI Interrupt bit is set and SPI interrupt is enabled in the VIC, SPI based interrupt can be processed by interrupt handling software.	0
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

SPI: modo maestro.

- Configurar modo maestro, velocidad, polaridad, flanco, número de bits, bit de comienzo y habilitar SPI.
- Escribir en el registro de datos y, según el esclavo, activar SSEL

Table 158: SPI Data Register (S0SPDR - address 0xE002 0008) bit description

Bit	Symbol	Description	Reset value
7:0	DataLow	SPI Bi-directional data port.	0x00
15:8	DataHigh	If bit 2 of the SPCR is 1 and bits 11:8 are other than 1000, some or all of these bits contain the additional transmit and receive bits. When less than 16 bits are selected, the more significant among these bits read as zeroes.	0x00

- Esperar la activación del bit SPIF (registro S0SPSR).
  - Opcionalmente se puede atender por interrupción.
- Leer los registros de estado y de datos, para:
  - Borrar el flag.
  - Recibir el dato del esclavo (si es el caso)
- Continuar enviando más datos, volviendo al paso 2.

SPI: modo esclavo

- Configurar velocidad, etc.
- Opcional: si hay que enviar un dato al maestro, debe escribirse en el registro de datos del SPI.
  - Analizar cuidadosamente el problema de la sincronización maestroesclavo.
- Esperar la activación del bit SPIF (registro S0SPSR).
  - Opcionalmente se puede atender por interrupción.
- Leer los registros de estado y de datos, para:
  - Borrar el flag.
  - Recibir el dato del esclavo (si es el caso)
- Continuar recibiendo/enviando más datos, volviendo al paso 2.

SPI modo maestro: spi.h ejemplo

```
#define SPI0_SEL                0x00000080
#define MAX_TIMEOUT             0xFF
#define SPI0_ABORT              0x01

/* SPI0 interrupt */
#define SPI0_MODE_FAULT         0x02
#define SPI0_OVERRUN            0x04
#define SPI0_COL                0x08
#define SPI0_TX_DONE            0x10
#define ABRT                    1 << 3

/* SPI0 interrupt status */
#define MODF                    1 << 4
#define ROVR                    1 << 5
#define WCOL                    1 << 6
#define SPIF                    1 << 7

#define RORIC                   0x00000001
#define RTIC                    0x00000002

/* SPI 0 PCR register */
#define SPI0_BE                 0x00000004
#define SPI0_CPHA               0x00000008
#define SPI0_CPOL               0x00000010
#define SPI0_MSTR               0x00000020
#define SPI0_LSBF               0x00000040
#define SPI0_SPIE               0x00000080
```

- Declaración de funciones

```
extern DWORD SPIInit( void );
extern void SPISend( BYTE *Buf, DWORD Length );
extern void SPIReceive( BYTE *Buf, DWORD Length );
extern BYTE SPIReceiveByte( void );
```


SPI: spi.c ejemplo

```
void SPIInit( void )
{
  S0SPCR = 0x00;
  PINSEL0 &= 0xFFFF00FF;
  PINSEL0 |= 0x00001500;
  IODIR0 = SPI0_SEL;
  IOSET0 = SPI0_SEL;
/* ajuste de la velocidad de transmisión*/
  S0SPCCR = 0x8;
/* 8 bt, CPOL=CPHA=0, master mode, MSB first,
interrupt enabled */
  S0SPCR = SPI0_SPIE | SPI0_MSTR;
}
```

```
void SPISend( BYTE *buf, DWORD Length )
{
  DWORD i;
  BYTE Dummy;

  if ( Length == 0 )
        return;
  for ( i = 0; i < Length; ++ )
  {
        S0SPDR = *buf;
        while ( !(S0SPSR & SPIF) );
        Dummy = S0SPDR; /* borrado del flag*/
        buf++;
  }
  return;
}
```

```
void SPIReceive( BYTE *buf, DWORD Length )
{
  DWORD i;

  for ( i = 0; i < Length; i++ )
  {
        *buf = SPIReceiveByte();
        buf++;
  }
  return;
}

BYTE SPIReceiveByte( void )
{
  BYTE data;

/* hay que escribir en S0SPDR para generar el reloj */
  S0SPDR = 0xFF;

  /* Wait for transfer complete, SPIF bit set */
   while ( !(S0SPSR & SPIF) );
   data = S0SPDR;
  return ( data );
}
```