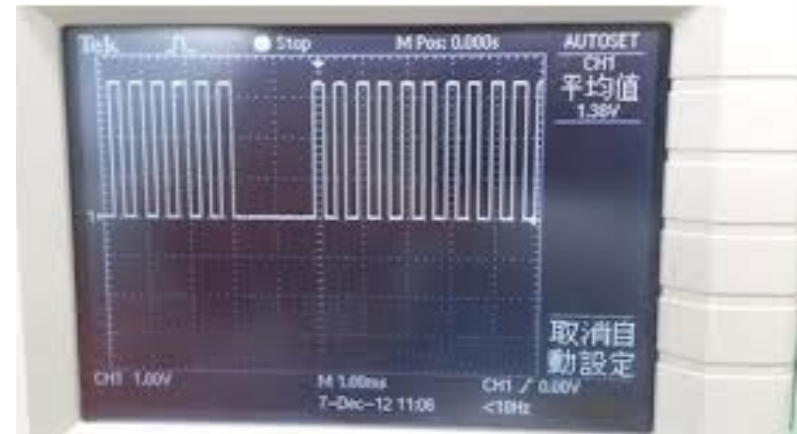
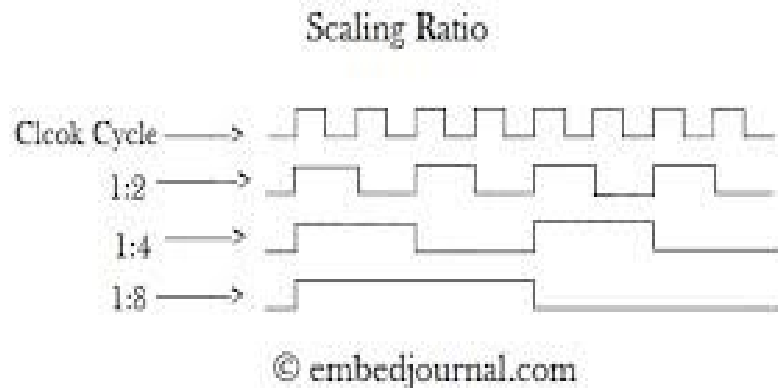

Timer y SysTick Timer



¿Qué es un Timer?

Podemos definir un timer simple como un contador de pulsos de clock

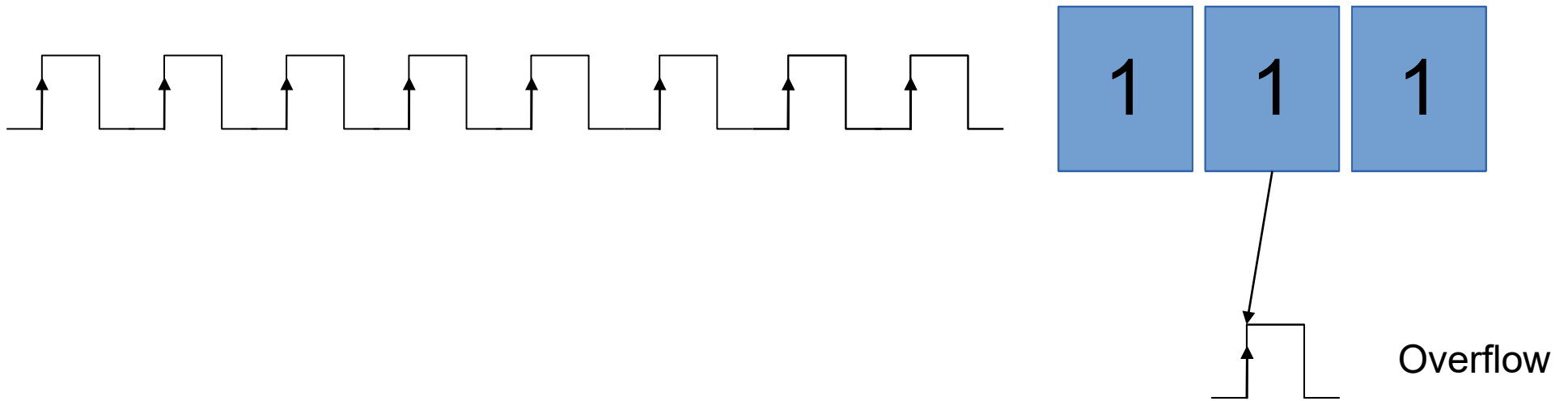


Conceptos

Clock -> marcar el paso

Preescaler -> división de frecuencia

Funcionamiento de Timer Elemental



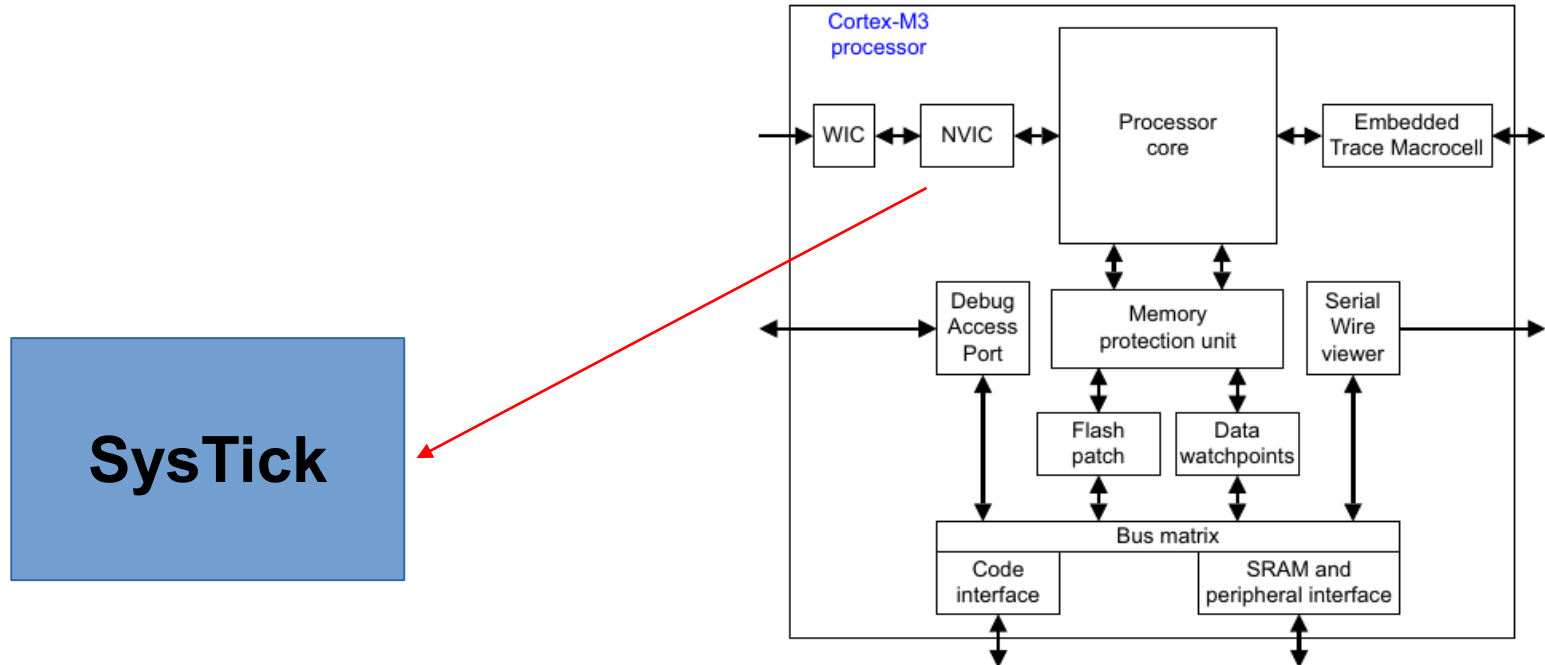
¿A qué frecuencia trabajamos?

```
void InicPLL(void){  
...  
...  
...  
}
```

Para nosotros es una caja negra que deja el clock de la MCU configurado para trabajar en 100MHz.

A partir de su ejecución la frecuencia del reloj del sistema es 100Mhz.

¿Qué es el SysTick?

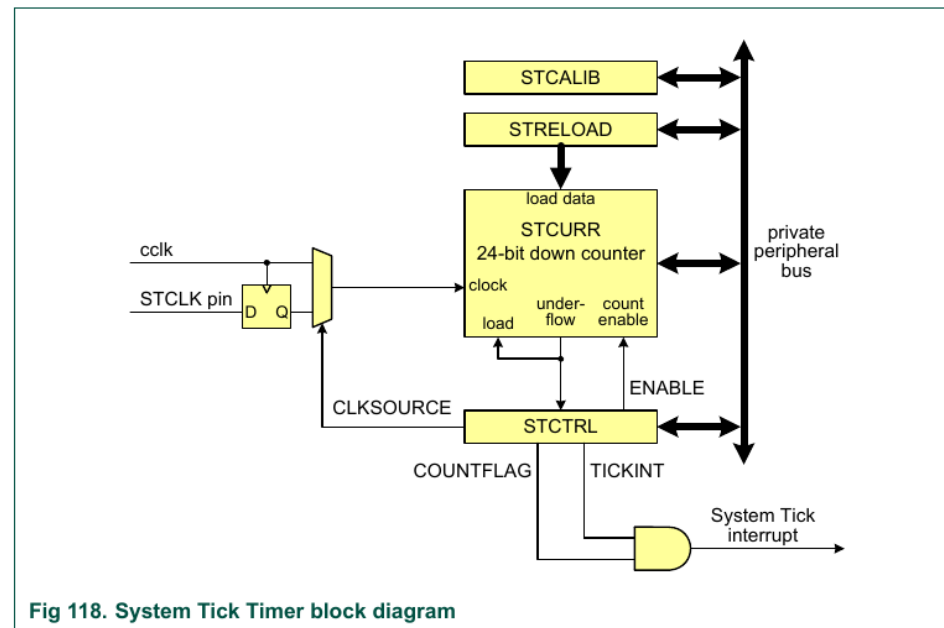


- Como timer simple, el SysTick es un contador de pulsos de clock.
- Se encuentra en el bloque del NVIC dentro del core ARM Cortex M3.

SysTick

Características

- Timer de 24-bits.
- Cuenta en forma descendente.
- Cuando llega a cero, genera una interrupción.
- Su objetivo es generar interrupciones de un período fijo de 10ms para un clock del sistema de 100MHz.



Un Clock para el SysTick

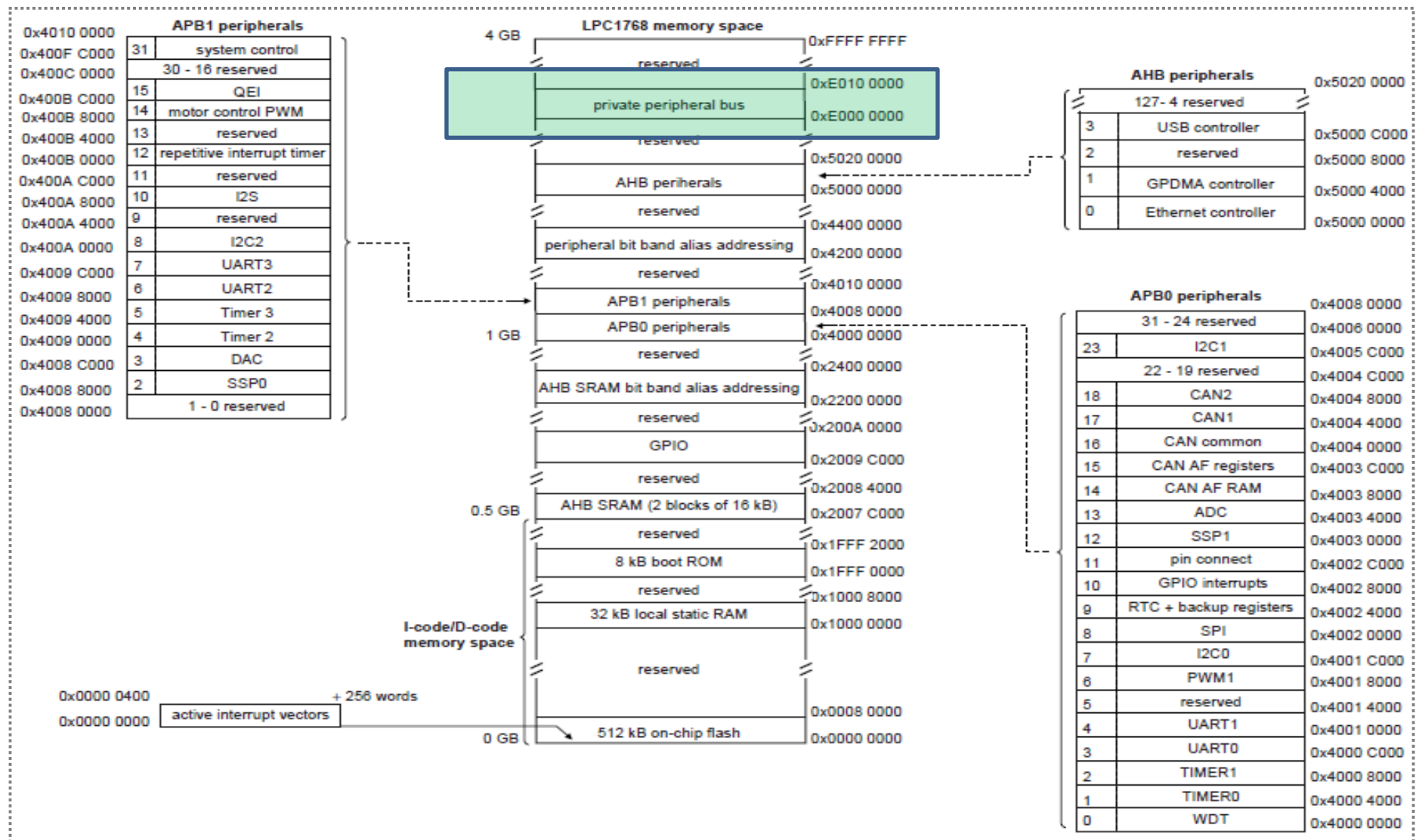
- El System Tick Timer puede recibir señal de reloj de diferentes fuentes: puede ser del BUS o de un pin (P3.26) donde comparte otras funciones. En nuestro caso utilizaremos STCLK.

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

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

[1] Not available on 80-pin package.

Mapa de memoria - LPC1769



Registros que lo controlan

Table 438. System Tick Timer register map

Name	Description	Access	Reset value ^[1]	Address
STCTRL	System Timer Control and status register	R/W	0x4	0xE000 E010
STRELOAD	System Timer Reload value register	R/W	0	0xE000 E014
STCURRE	System Timer Current value register	R/W	0	0xE000 E018
STCALIB	System Timer Calibration value register	R/W	0x000F 423F	0xE000 E01C

Systick tiene cuatro registros que lo controlan:

- STCTRL: controla funcionamiento
- STRELOAD: lo utilizaremos en la inicialización
- STCURRE: devuelve la cuenta actual del contador
- STCALIB: cuenta por defecto

STCTRL

System Timer Control and status register (STCTRL - 0xE000 E010)

- Controla el funcionamiento de SysTick y tiene el flag de interrupciones.

Table 439. System Timer Control and status register (STCTRL - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0
2	CLKSOURCE	System Tick clock source selection. When 1, the CPU clock is selected. When 0, the external clock pin (STCLK) is selected.	1
Bit	Symbol	Description	Reset value
15:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	COUNTFLAG	System Tick counter flag. This flag is set when the System Tick counter counts down to 0, and is cleared by reading this register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Importante:

- La habilitación de interrupciones no requiere del NVIC
- El flag de interrupciones se limpia leyendo este registro, pero en la práctica se limpia solo

STRELOAD

System Timer Reload (STRELOAD - 0xE000 E014)

- Su valor se cargará en el System Tick Timer cuando el valor de cuenta llegue a cero.
- Este registro se carga por software como parte de la inicialización del timer.
- El registro STCALIB puede usarse como valor para cargar STRELOAD si el clock está configurado a la frecuencia adecuada (100Mhz).

Table 440. System Timer Reload value register (STRELOAD - 0xE000 E014) bit description

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

STCURRE

System Timer Current (STCURRE - 0xE000 E018)

- Al ser leído por el software, el registro STCURRE devuelve el valor de la cuenta actual de contador del System Tick.

Table 441. System Timer Current value register (STCURRE - 0xE000 E018) bit description

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in STCTRL.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

STCALIB

System Timer Calibration (STCALIB - 0xE000 E01C)

- Contiene un valor inicializado en fábrica para que permita al SysTick generar interrupciones cada 10ms si el clock que lo alimenta es de 100MHz. Este es el uso para el que fue diseñado por ARM.

Table 442. System Timer Calibration value register (STCALIB - 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		Reload value to get a 10 millisecond System Tick underflow rate when running at 100 MHz. This value initialized at reset with a factory supplied value selected for the LPC17xx. The provided values of TENMS, SKEW, and NOREF are applicable only when using a CPU clock or external STCLK source of 100 MHz.	0x0F 423F
29:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	SKEW		Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is initialized at reset with a factory supplied value selected for the LPC17xx. See the description of TENMS above. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0
31	NOREF		Indicates whether an external reference clock is available. This bit is initialized at reset with a factory supplied value selected for the LPC17xx. See the description of TENMS above. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

SysTick - Funcionamiento

- Para generar interrupciones recurrentes a intervalos de tiempo específicos, se debe configurar el registro STRELOAD.
- Un valor default se provee en el registro STCALIB y puede cambiarse por software.
- Este valor default permite generar interrupciones cada 10 ms si el reloj de la CPU está configurado a 100 Mhz
- El cálculo del valor a configurar en STRELOAD se obtiene del modo siguiente:

$$RELOAD = (SysTick\ clk/100) - 1$$

Representación de Registros - SysTick

Bit	Symbol
0	ENABLE
1	TICKINT
2	CLKSOURCE
15:3	-
16	COUNTFLAG
31:17	-

```

//!< ////////////////////////////////// SYSTICK //////////////////////////////////
//!< Tipo de dato específico para manejar el SYSTICK
typedef struct
{
    union{
        __RW uint32_t _STCTRL;
        struct{
            __RW uint32_t _ENABLE:1;
            __RW uint32_t _TICKINT:1;
            __RW uint32_t _CLKSOURCE:1;
            __RW uint32_t _RESERVED0:12;
            __RW uint32_t _COUNTFLAG:1;
            __RW uint32_t _RESERVED1:16;
        }bits;
    };
    __RW uint32_t _STRELOAD;
    __RW uint32_t _STCURRE;
    __R uint32_t _STCALIB;
}systick_t;

```

```

//!< 0xE000E010UL: __RW uint32_t de control del SysTick:
#define DIR_SYSTICK ( (systick_t *) 0xE000E010UL )

```

```

#define STCTRL DIR_SYSTICK->_STCTRL
#define ENABLE DIR_SYSTICK->bits._ENABLE
#define TICKINT DIR_SYSTICK->bits._TICKINT
#define CLKSOURCE DIR_SYSTICK->bits._CLKSOURCE
#define COUNTFLAG DIR_SYSTICK->bits._COUNTFLAG
#define STRELOAD DIR_SYSTICK->_STRELOAD
#define STCURRE DIR_SYSTICK->_STCURRE
#define STCALIB DIR_SYSTICK->_STCALIB

```

Name	Reset value ^[1]	Address
STCTRL	0x4	0xE000 E010
STRELOAD	0	0xE000 E014
STCURRE	0	0xE000 E018
STCALIB	0x000F 423F	0xE000 E01C

System Timer

Inicialización

```
void SysTickInic( void )  
{  
    STRELOAD = ( SysTick clk / N ) - 1 ;  
    STCURR = 0;  
  
    ENABLE = 1;  
    TICKINT = 1;  
    CLKSOURCE = 1;  
    return;  
}
```

¿y este nombre?

El valor de STRELOAD
dependerá de la aplicación

Función de interrupción

```
void SysTick_Handler ( void )  
{  
  
    ... Aquí va el código que deba  
    ejecutarse a cada vencimiento del  
    timer...  
  
}
```

Si **N**=1 → tick cada 10ms. (si clock=100Mhz)
El valor de clock ya está dividido 100

Prototipo en *cr_startup_lpc176x.c*


```
62//*****
63// Forward declaration of the default handlers. These are aliased.
64// When the application defines a handler (with the same name), this will
65// automatically take precedence over these weak definitions
66//*****
67    void ResetISR(void);
68WEAK void NMI_Handler(void);
69WEAK void HardFault_Handler(void);
70WEAK void MemManage_Handler(void);
71WEAK void BusFault_Handler(void);
72WEAK void UsageFault_Handler(void);
73WEAK void SVCcall_Handler(void);
74WEAK void DebugMon_Handler(void);
75WEAK void PendSV_Handler(void);
76WEAK void SysTick_Handler(void);
77WEAK void IntDefaultHandler(void);
78
```



débil

Vector en *cr_startup_lpc176x.c*

```
148//*****
149// The vector table.
150// This relies on the linker script to place at correct location in memory.
151//*****
152extern void (* const g_pfnVectors[])(void);
153__attribute__((section(".isr_vector")))
154void (* const g_pfnVectors[])(void) = {
155    // Core Level - CM3
156    &vStackTop, // The initial stack pointer
157    ResetISR,    // The reset handler
158    NMI_Handler, // The NMI handler
159    HardFault_Handler, // The hard fault handler
160    MemManage_Handler, // The MPU fault handler
161    BusFault_Handler, // The bus fault handler
162    UsageFault_Handler, // The usage fault handler
163    0, // Reserved
164    0, // Reserved
165    0, // Reserved
166    0, // Reserved
167    SVC_Handler, // SVC_Handler
168    DebugMon_Handler, // Debug monitor handler
169    0, // Reserved
170    PendSV_Handler, // The PendSV handler
171    SysTick_Handler, // The SysTick handler
172
173    // Chip Level - LPC17
174    WDT_IRQHandler, // 16, 0x40 - WDT
175    TIMER0_IRQHandler, // 17, 0x44 - TIMER0
176    TIMER1_IRQHandler, // 18, 0x48 - TIMER1
177    TIMER2_IRQHandler, // 19, 0x4c - TIMER2
178    TIMER3_IRQHandler, // 20, 0x50 - TIMER3
179    UART0_IRQHandler, // 21, 0x54 - UART0
180    UART1_IRQHandler, // 22, 0x58 - UART1
```



ISR del SysTick

```
//-----  
// Rutina de servicio de la interrupción de SysTick  
//-----  
void SysTick_Handler(void)  
{  
    __RW uint32_t dummy;  
  
    dummy = STCTRL;           // Limpia flag de int  
  
    if(contTimeout)  
    {  
        contTimeout--;  
    }  
  
}
```

Ejemplo – SysTick

- Uso de SysTick para temporizar un led parpadeante

```
//-----  
// Rutina de servicio de la interrupción de SysTick  
//-----  
void SysTick_Handler(void)  
{  
    static __RW uint8_t contTimeout = 100;  
    __RW uint32_t dummy;  
    dummy = STCTRL;  
  
    if(flagLED)  
    {  
        contTimeout--;  
        if (!contTimeout)  
        {  
            contTimeout = 100;  
            if(GetPIN (LEDXpresso,0))  
                SetPIN (LEDXpresso,1); // Pone pin LED del stick en 1  
            else  
                SetPIN (LEDXpresso,0); // Pone pin LED del stick en 0  
        }  
    }  
}
```