

Maquinaria de timers - primitivas y drivers en base al systick

Informática II
R2004 - 2021

Primer ejercicio de timers

Generar una función de inicialización y la ISR del SysTick, configurando la interrupción cada 1 milisegundo, de manera de poder hacer titilar un led cada 1 segundo, de la siguiente manera:

```
uint8_t estado = 0;


while ( 1 ) {

    if ( segundo ){

        segundo = 0;
        LedRojo ( estado );
        estado ^=1;

    }

}
```



¿Qué debe hacer una interrupción de SysTick?

```
//Para un SysTick configurado a 1mseg
uint16_t timer = 1000;

void SysTick_Handler (void)
{
    timer --;
    If ( ! timer )
    {
        timer = 1000;
        segundo = 1;
    }
}
```

Cuento 1000 msecs = 1 seg

```
while ( 1 ) {

    uint8_t estado = 0;
    if ( segundo ){

        segundo = 0;
        LedRojo ( estado );
        estado ^=1;
    }
}
```

En este ejemplo, uso una variable timer que cuenta las interrupciones de systick (cada 1ms), y una variable segundo que actua como un flag entre la interrupción y el programa principal

En este esquema, ¿como puedo prender/apagar el timer?

```
//Para un SysTick configurado a 1mseg  
uint16_t timer = 0;
```

```
void SysTick_Handler (void)  
{
```

```
    if (timer)  
    {  
        timer --;  
        If ( ! timer )  
        {  
            timer = 1000;  
            segundo = 1;  
        }  
    }
```

Prendo/Apago el timer

Cuento el tiempo

```
while ( 1 ) {
```

```
    uint8_t estado = 0;  
    If ( GetKey() == TECLA0 ){  
        if ( timer ) timer = 0;  
        else timer = 1000;  
    }
```

```
    if ( segundo ){  
        segundo = 0;  
        LedRojo ( estado );  
        estado ^=1;  
    }
```

```
}
```

¿Qué tareas puedo hacer con un timer?

Comienzo del timer - ***TimerStart(uint32_t tiempo);***

Debería poner la variable timer en el valor tiempo (expresado en milisegundos, para el ejemplo realizado anteriormente)

Detener el timer - ***TimerStop();***

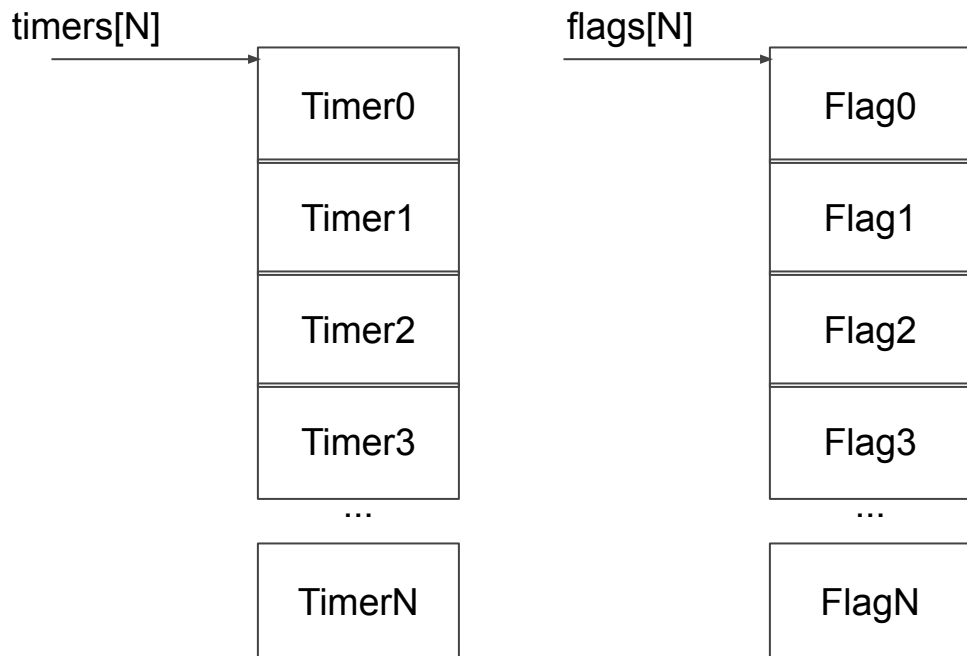
Debería poner la variable timer en 0, para evitar que se siga contando el tiempo (y por las dudas poner la variable flags en 0 también)

Chequear si el timer se venció - ***uint8_t TimerEvent(void)***

Chequear si el flag del timer se puso en 1, y devolver esta variable, o realizar la acción correspondiente. En cualquier caso, volver la variable flag a 0

¿Qué pasa si tengo N timers?

Para ampliar el ejemplo anterior y manejar al mismo tiempo N timers, puedo generar un vector de tiempos y de flags, y mantener la misma estructura de código:



```
void SysTick_Handler (void)
{
    for ( i = 0 ; i < N ; i++ ){
        if ( timers[i] )
        {
            timers[i] --;
            if ( ! timers[i] )
                flags[i] = 1;
        }
    }
}
```

¿Como lo amplio para N timers?

Comienzo del timer n -ésimo - **`TimerStart(uint8_t ntimer , uint32_t tiempo);`**

Debería poner la variable timer en la posición ntimer del vector en el valor tiempo (expresado en milisegundos, para el ejemplo realizado anteriormente)

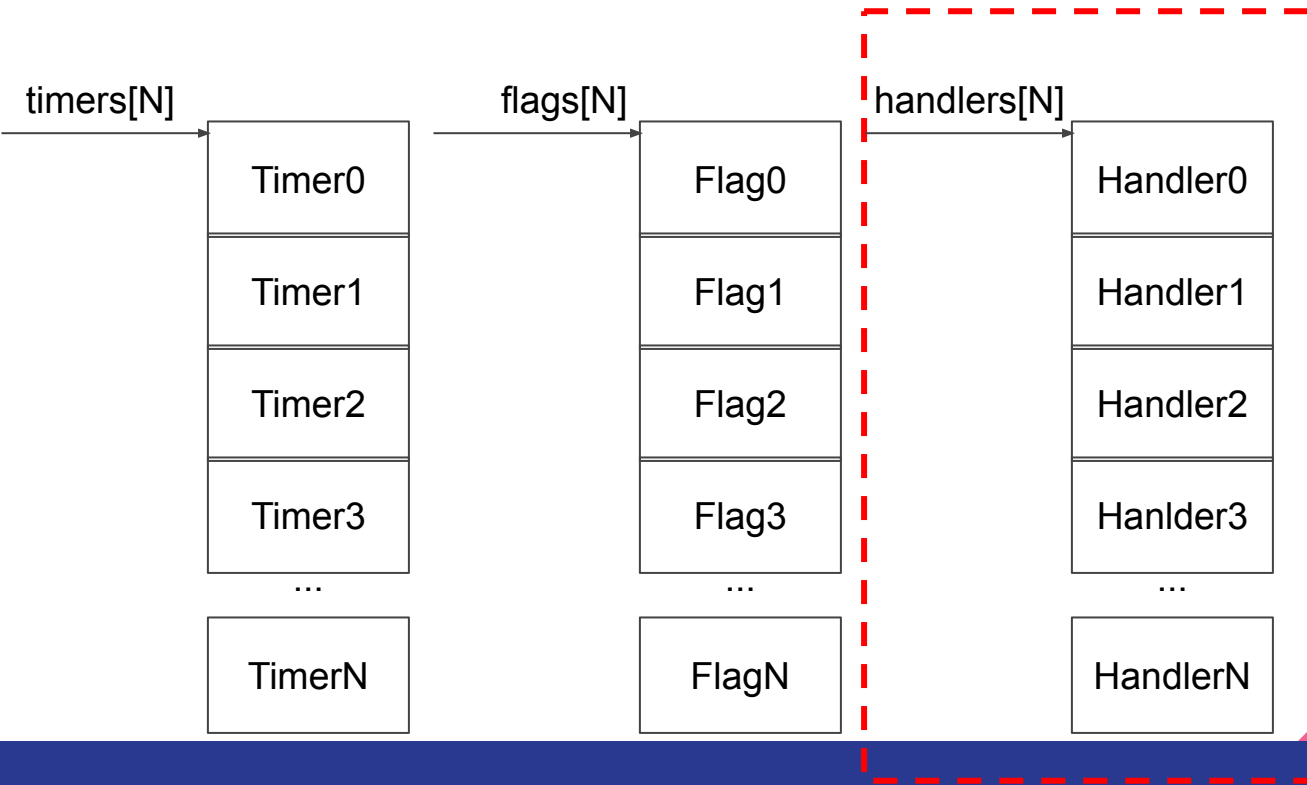
Detener el timer n -ésimo - **`TimerStop(uint8_t ntimer);`**

Debería poner la variable timer en 0 en la posición ntimer, para evitar que se siga contando el tiempo del n -ésimo timer (y por las dudas poner la variable flags en 0 también)

Chequear si el timer n -ésimo se venció - **`uint8_t TimerEvent(uint8_t ntimer)`**

Chequear si el flag del timer n -ésimo se puso en 1, y devolver esta variable, o realizar la acción correspondiente. En cualquier caso, volver la variable flag a 0

Ampliando - ¿Cómo desarrollamos las primitivas que usamos en el 1er cuatrimestre?



Agregamos un vector de **PUNTEROS A FUNCIÓN** que deberá invocarse cuando se vence el timer n-esimo.

La función encargada de CARGAR este vector es ***TimerStart***, y la función encargada de invocar al Handler es ***TimerEvent***

Ejercicio - Desarrollamos las primitivas

Comienzo del timer n-ésimo:

```
TimerStart ( uint8_t ntimer , uint32_t tiempo ,  
             ( void ) *Handler ( void ) , uint8_t base );
```

Detener el timer n - ésimo:

```
TimerStop ( uint8_t ntimer );
```

Chequear si algún timer se venció, e invocar al Handler correspondiente:

```
void TimerEvent ( void )
```



Ejercicio 2 - Usando los timers

1. Realizar un programa que haga titilar un led. El mismo deberá encenderse y apagarse a una frecuencia de 0,25Hz, luego 0,5Hz, luego a 1Hz, luego a 2Hz. Las transiciones de una frecuencia a la siguiente se darán cada 10 segundos.
2. Modificar el programa anterior de manera de ir cambiando el led que titila con la opresión del interruptor del stick.
3. Modificar el programa anterior de manera de que el sistema se apague o se prenda cuando el interruptor del stick (el mismo que se utiliza para cambiar de color) se mantenga presionado por 5 segundos.

Las posiciones de los leds y el interruptor son:

| | | | |
|------------|------|-----------|------|
| Led Rojo: | P1.0 | Led Azul: | P1.2 |
| Led Verde: | P1.1 | Pulsador: | P0.4 |

