Tema 8
RESUMEN

En el **tema 8** (***Temporización: TOC, PWM y TIC***) se enseña cómo utilizar el bloque TIMERS del microcontrolador. Este tema va acompañado de unos videos explicativos, los cuales son de una versión anterior de las transparencias y utilizando un entorno de desarrollo distinto (el Keil uVision 5 en lugar del STM32CubeIDE). Sin embargo, las explicaciones siguen siendo totalmente válidas, teniendo (para cualquier tipo de incongruencia) validez las transparencias publicadas. Este tema se compone de los siguientes apartados:

En el primer punto se habla de los conceptos previos del bloque TIMERS, en el que se indica que el subsistema de temporización sirve para generar "unidades de tiempo" tanto para él mismo, como para otros periféricos, pero que sin embargo, un temporizador NO CUENTA TIEMPO, SINO CICLOS DE RELOJ, introduciendo.

- El concepto de reloj del sistema (SYSCLK).
- El concepto de preescalado.
- El subsistema de reloj del curso.
- Mostrando dos ejemplos de temporización (sin y con preescalado).
- Y finalmente mostrando dos ejemplos de las dos funcionalidades básicas de un temporizador (TOC y TIC).

En el segundo punto se describe la **arquitectura** del **BLOQUE TIMERS**, indicando primero sus características básicas y diagrama de bloques, después explicando su modo de funcionamiento y finalmente sus diferentes registros de control, datos y estado.

Los diferentes ***registros de control*** del bloque TIMERS, que sirven para configurar su funcionalidad antes de utilizarlo, son los siguientes:

- **TIMx→CR1 –** Control Register 1, que es un registro de 16 bits con sólo 2 utilizables en el curso:
  - ARPE – Auto-reload preload enable (sólo se usará en PWM)
    - 0 – No se utilizará el TIMx_ARR
    - 1 – Se utilizará el TIMx_ARR
  - CEN – Counter enable
    - 0 – contador deshabilitado
    - 1 – contador arrancado

- **TIMx→CR2** – Control Register 2, que siempre se pone a 0 en el curso.
- **TIMx→SMCR –** Slave Mode Control Register, que siempre se pone a 0 en el curso.
- **TIMx→DIER –** DMA/Interrupt Enable Register, en donde:
    - CCyIE se pondrán a '1' cuando se vaya a utilizar esa fuente de IRQ, y se mantendrán a '0' cuando no.
- **TIMx→EGR** – Event Generation Register, que es un registro de 16 bits que se utiliza para generar eventos de forma manual. El registro no se utilizará de forma habitual. De hacerlo, todos los bits se mantendrán a '0' salvo UG, que se pondrá a '1' para refrescar los valores de los registros internos.
    - UG – Se pone un '1' para generar un evento de update, con lo que se actualizan los registros. El hardware pone el bit automáticamente a '0'. Siempre se activa.
- **TIMx→CCMR1** – Capture/Compare Mode Register 1, que es un registro de 16 bits, para configurar los canales 1 y 2 (existe otro registro equivalente para los canales 3 y 4: TIMx_CCMR2). La configuración inicial se hace escribiendo el bit CCyS – Capture/Compare y Selection:
    - 00 – como TOC
    - 01 – como TIC asociado al su propia entrada TIMx
    - 10 – como TIC asociado a la entrada TIMx del otro canal del registro (no se utiliza este modo en el curso)
    - 11 – (no se utiliza este modo en el curso)
    - **Para la funcionalidad TOC y PWM**
        - OCyCE – Output Compare y Clear Enable (se pondrá siempre a 0)
        - OCyM – Output Compare y Mode
            - 000 – Sin salida
            - 001 – La salida se pone a 1 tras la comparación exitosa
            - 010 – La salida se pone a 0 tras la comparación exitosa
            - 011 – Toogle
            - 100 – Se fuerza la salida a 0
            - 101 – Se fuerza la salida a 1
            - 110 – PWM con el primer semiciclo a 1
            - 111 – PWM con el primer semiciclo a 0
        - OCyPE – Preload Enable,
            - Se habilita con '1' (se toma el valor de CCRy al generar un update event) y se deshabilita con '0' (CCRy coge el valor inmediatamente, según se escribe en él). Se usa para PWM.
        - OCyFE – (se utiliza '0')
    - **Para la funcionalidad TIC**
        - ICyF – Filter (se utiliza 0000 – sin filtrado previo)
        - ICyPSC – Input Capture Prescaler (se utiliza 00 – sin pre-escalado específico)
- **TIMx→CCER** – Capture/Compare Enable Register, en el que para canal:
    - **Para la funcionalidad TOC y PWM**

        - CCyNP – Se deja a '0' en TOC y PWM
        - CCyP – Se deja a '0' en TOC y PWM
        - CCyE – Output Enable
            - Un '0' desactiva la salida hardware, y un '1' la activa.
    - **Para la funcionalidad TIC**
        - CCyNP:CCyP – Polarity:
            - 00 – activo por flanco de subida
            - 01 – activo por flanco de bajada
            - 10 – (reservado)
            - 11 – activo por ambos flancos
        - CCyE – Output Enable:
            - Un '0' deshabilita la captura, y un '1' la habilita

Los diferentes ***registros de datos*** del bloque TIMERS, que sirven para leer/escribir los datos en los pines del bloque, son los siguientes:

- **TIMx→CNT** – Counter, donde se tiene el valor de los pasos del contador.
- **TIMx→PSC** – Prescaler, en donde en función del valor indicado aquí hace que la frecuencia del reloj del CNT sea:
    - PSC = 0 -> fck/1 (sin preescalado)
    - PCS = 1 -> fck/2 (divido por 2)
    - PSC = 2 -> fck/3 (divido por 3)
- **TIMx→ARR** – Auto-Reload Register, que es el valor que se cargará en el registro interno de auto-reload (sólo lo vamos a usar en PWM). Aunque no se use, hay que escribir un valor que no sea 0 para TOC y TIC por lo que se aconseja ponerlo a 0xFFFF si no se usa.
- **TIMx→CCRy** – Capture/Compare Register (Channel y), de manera que:
    - En modo TOC y PWM marca en valor en el que ocurrirá la comparación exitosa
    - En el modo TIC es donde se almacenará el valor del CNT cuando el evento externo ocurra

El único ***registro de estado*** del bloque TIMERS, que sirve para comprobar y consultar la ejecución de las actuaciones sobre los pines del bloque, es el siguiente:

- **TIMx→SR – Status Register**, que es un registro de 16 bits con sólo 8 útiles para el curso:
    - CCyOF: 4 flags que indican con un '1' si ha ocurrido un error de overrun (un flag para cada uno de los canales). Para limpiar el flag hay que escribir un '0'.
    - CCyIF: 4 flags de indican con un '1' que ha ocurrido el evento programado en ese canal. El flag se limpia leyendo el TIMx_CCRy correspondiente, o escribiendo un '0' en ese bit.

Finalmente, se muestran las peculiaridades al usar el bloque TIMERS con el software STM32CubeIDE, en su perspectiva CubeMX.

Una vez vista en su totalidad el periférico, se pasa a particularizar su uso para los 3 modos de funcionamiento básicos:

- **Timer Output Compare (TOC):** utilizado para que se produzca un evento cuando el temporizador llegue a un valor prefijado (al estilo de la alarma de un reloj despertador, donde se configura previamente la hora en la que debe sonar y al llegar el reloj a esa hora, suena). También se puede utilizar para generar una señal cuadrada, con *duty cycle* del 50% y frecuencia seleccionada (y si se quiere, variable).
- **Pulse Width Modulation (PWM):** utilizado para generar señales en las que se queda fijado el periodo de la señal, y se varía el *duty cycle* según necesidades del proyecto.
- **Timer Input Capture (TIC):** utilizado para registrar el valor del temporizador, cuando ocurre un evento externo.

Por lo tanto, en el tercer punto se describe la **funcionalidad TOC** del **BLOQUE TIMERS** (consiste en empezar a contar y cuando llegue a un valor configurado se genera una interrupción o un evento, por ejemplo, una alarma horaria), indicando primero su funcionalidad, a continuación cómo proceder con los registros de configuración para poder utilizarlo y finalmente dos ejemplos prácticos (uno sin interrupciones y otro con interrupciones) para probar por el alumno, con su programa correspondiente (inicialización y funcionamiento continuo para el ejemplo sin interrupciones y variables globales/RAI, inicialización y funcionamiento continuo para el ejemplo con interrupciones) -> El ejemplo genera contador de segundos partiendo de un PCLK1 de 32MHz.

En el siguiente punto se describe la **funcionalidad PWM** del **BLOQUE TIMERS** (consiste en aplicar energía con una forma de onda cuadrada de una determinada frecuencia, y donde se controla la cantidad de tiempo que la señal se encuentra a nivel alto (el llamado “duty cycle” – DC), indicando primero su funcionalidad, a continuación cómo proceder con los registros de configuración para poder utilizarlo y finalmente un ejemplo práctico para probar por el alumno, con su programa correspondiente (inicialización y funcionamiento continuo) -> El ejemplo, partiendo de un reloj de 32MHz, genera una señal PWM de 100Hz, cambiando el DC de 10 en 10%, cada vez que se pulsa PA0. La salida se hace por PB7 y el LED verde cambiará de intensidad según varíe el DC, mientras que el DC se muestra por el LCD.

En el último punto se describe la **funcionalidad TIC** del **BLOQUE TIMERS** (que se utiliza para medir el tiempo que pasa entre eventos externos), indicando primero su funcionalidad, a continuación cómo proceder con los registros de configuración para poder utilizarlo y finalmente dos ejemplos prácticos (uno sin interrupciones y otro con interrupciones) para probar por el

alumno, con su programa correspondiente (inicialización y funcionamiento continuo para el ejemplo sin interrupciones y variables globales/RAI, inicialización y funcionamiento continuo para el ejemplo con interrupciones) -> El ejemplo, partiendo de un PCLK1 de 32MHz, cuenta el tiempo que se tarda entre pulsaciones del botón USER, mostrándolo en ms en el LCD.