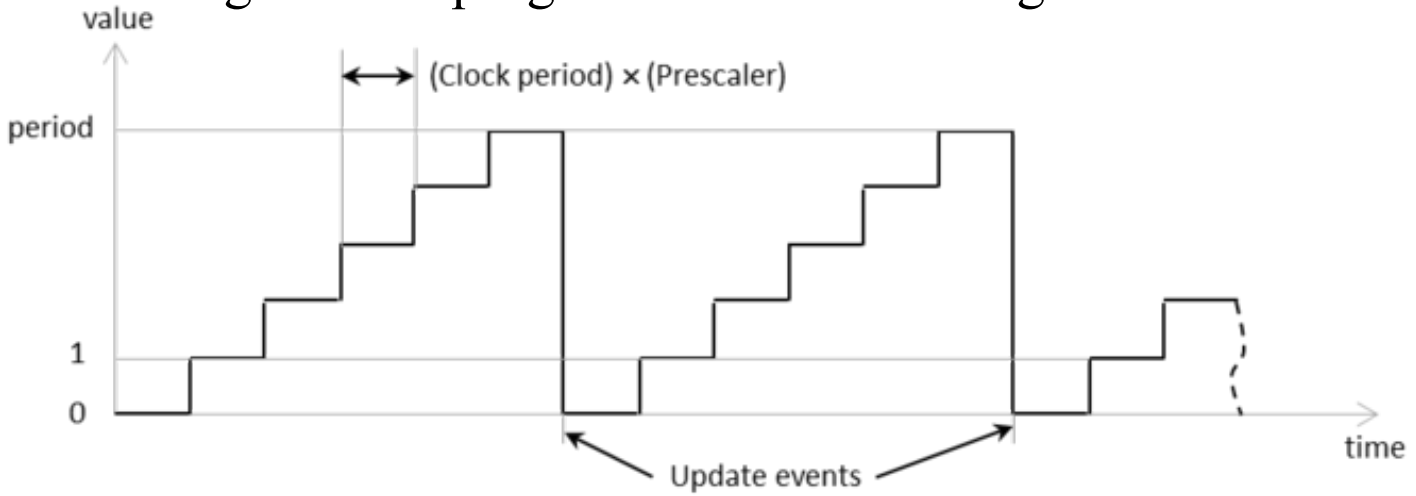
Timers

Luigi Coppolino, Giovanni Mazzeo

Timer Hardware

- I timer hardware sono usati per:
  - Generare segnali a varie frequenze
  - Generare pulse-width-modulated output
  - Misurare il tempo trascorso tra due eventi di interesse
- Un timer hardware è un contatore che conta ad una certa velocità, definita dall'utente, da zero sino ad uno specifico valore di periodo preimpostato, generando *eventi* quando tale valore di periodo è raggiunto.
- Esegue in background al programma che si sta eseguendo.



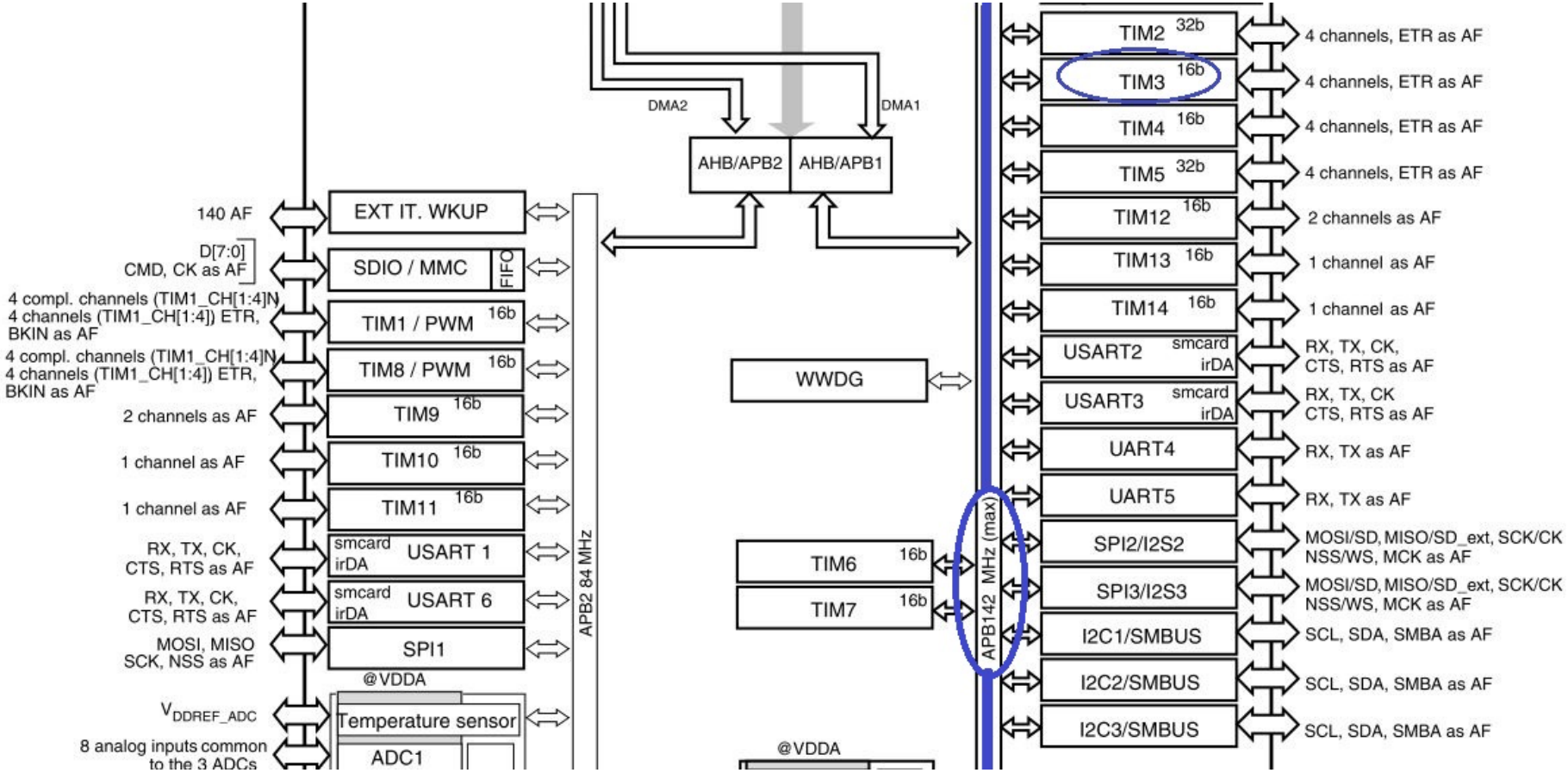
STM32F3 Timers #1

- Il microcontrollore della STM32F3-Discovery board è provvisto di dieci timer, di cui:
  - 6 sono *General-Purpose Timers* (TIM2 to TIM4 e TIM15 to TIM17) a 16 o a 32 bit
  - 2 sono *Advanced Timers* (TIM1 e TIM8) per, e.g., motor control
  - 2 sono *Basic Timers* (TIM6 and TIM7) usati ad esempio per fornire una base dei tempi per fare il ”trigger” dei convertitori DAC o ADC

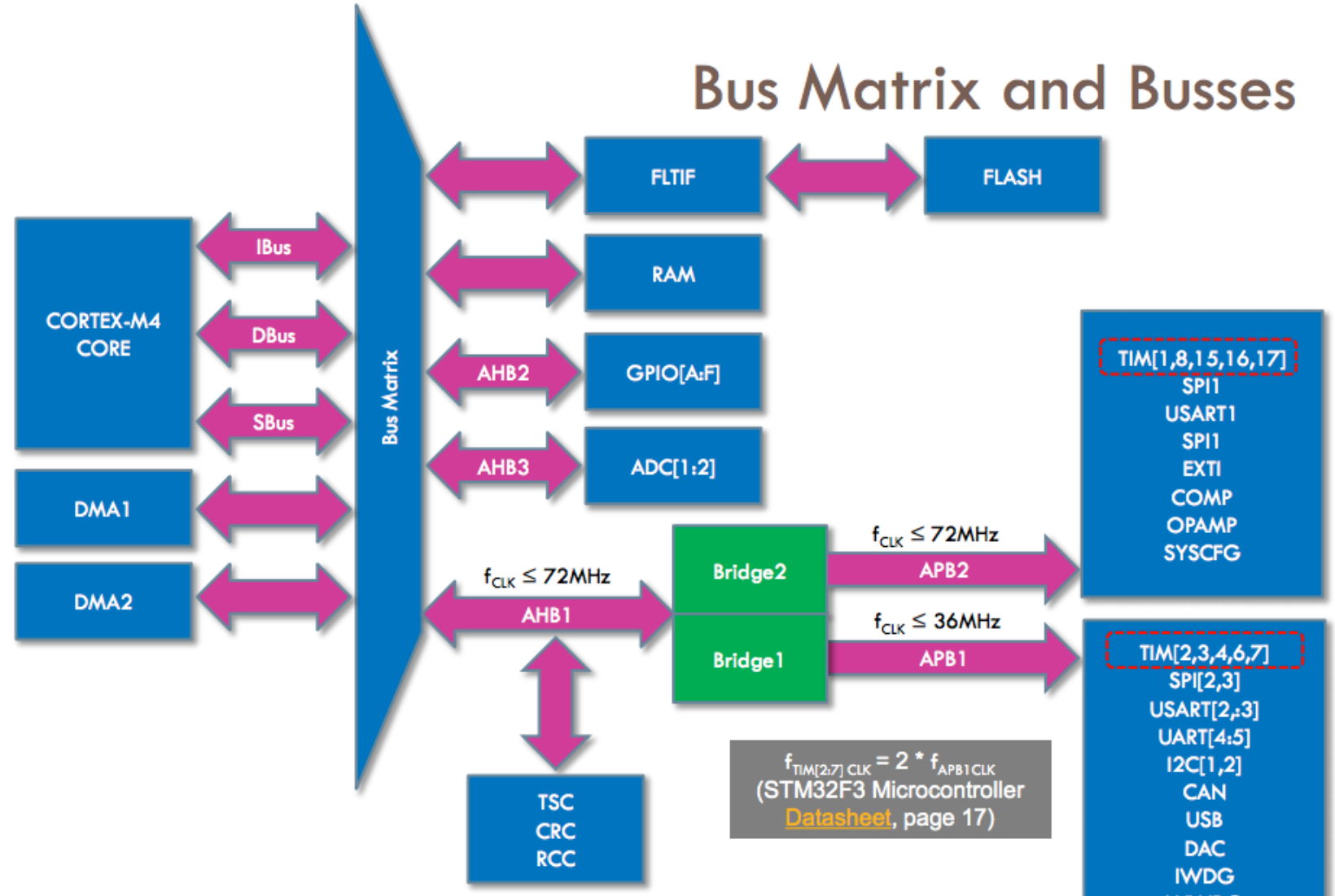
STM32F3 Timers #2

- Tutti i timer chiaramente dipendono dalla frequenza del clock del processore, nella nostra board i timer:
  - Ricevono in input un clock a 72 o a 36MHz
  - Fanno uso di un *Prescaler* (a 16 bit) per dividere la frequenza del clock in input e gestire quindi frequenze di proprio interesse
- Possono contare sia verso l'alto (Upcounting), sia verso il basso (Downcounting)
- Raggiunto il valore di "*Auto-Reload*" il timer *setta* un *flag* nel suo status register per comunicare l'avvenuto raggiungimento del valore di auto-reload. Se configurato, il timer può generare un evento (un **interrupt**) per il microprocessore.

Il TIM nel SoC STM32F3



Bus Matrix and Busses



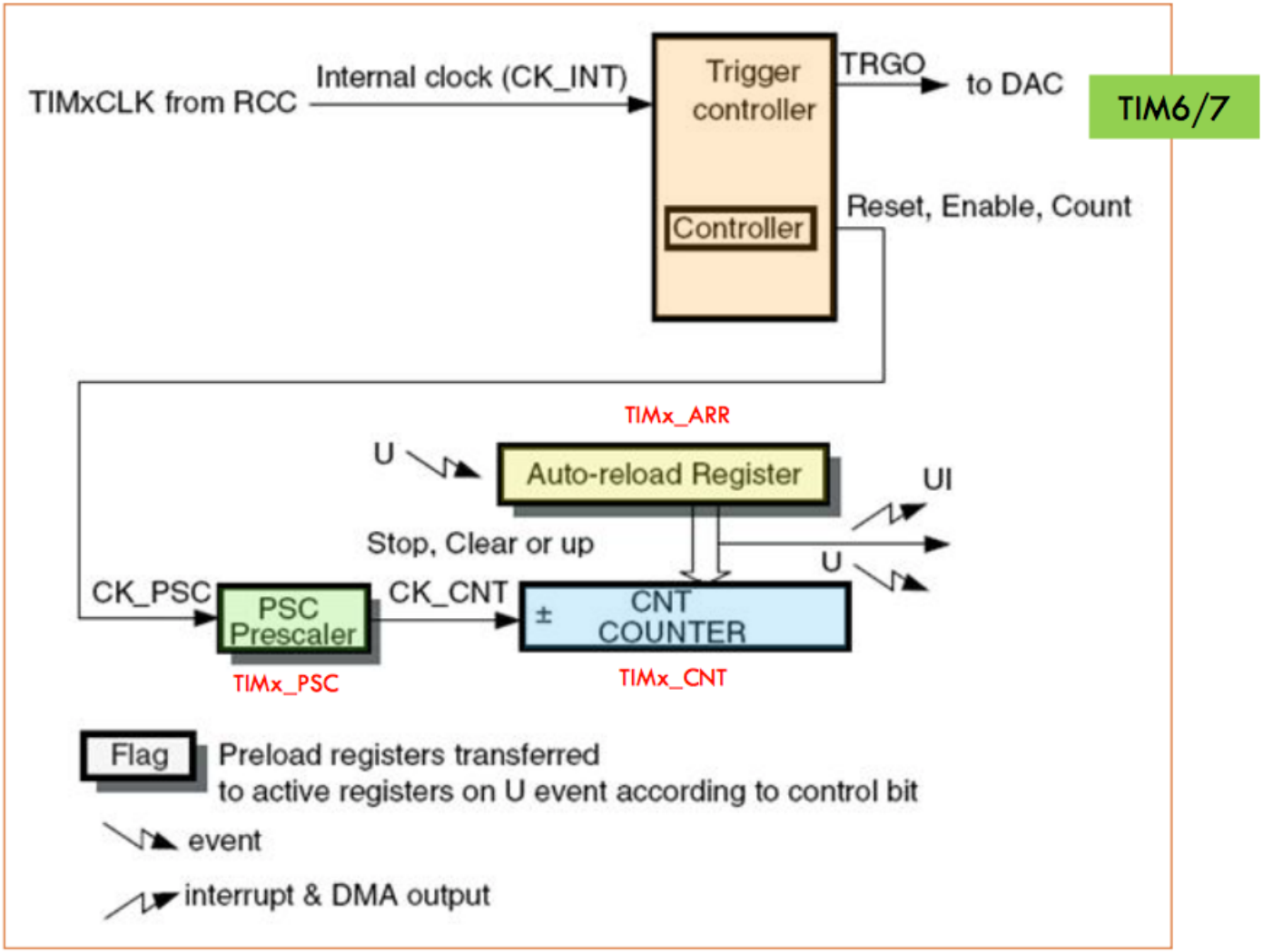
AHB[1:3]: Advanced High-performance Bus
APB: Advanced Peripheral Bus
RCC: Reset and Clock Control

STM32F3 Microcontroller Reference Manual, pages 41-44

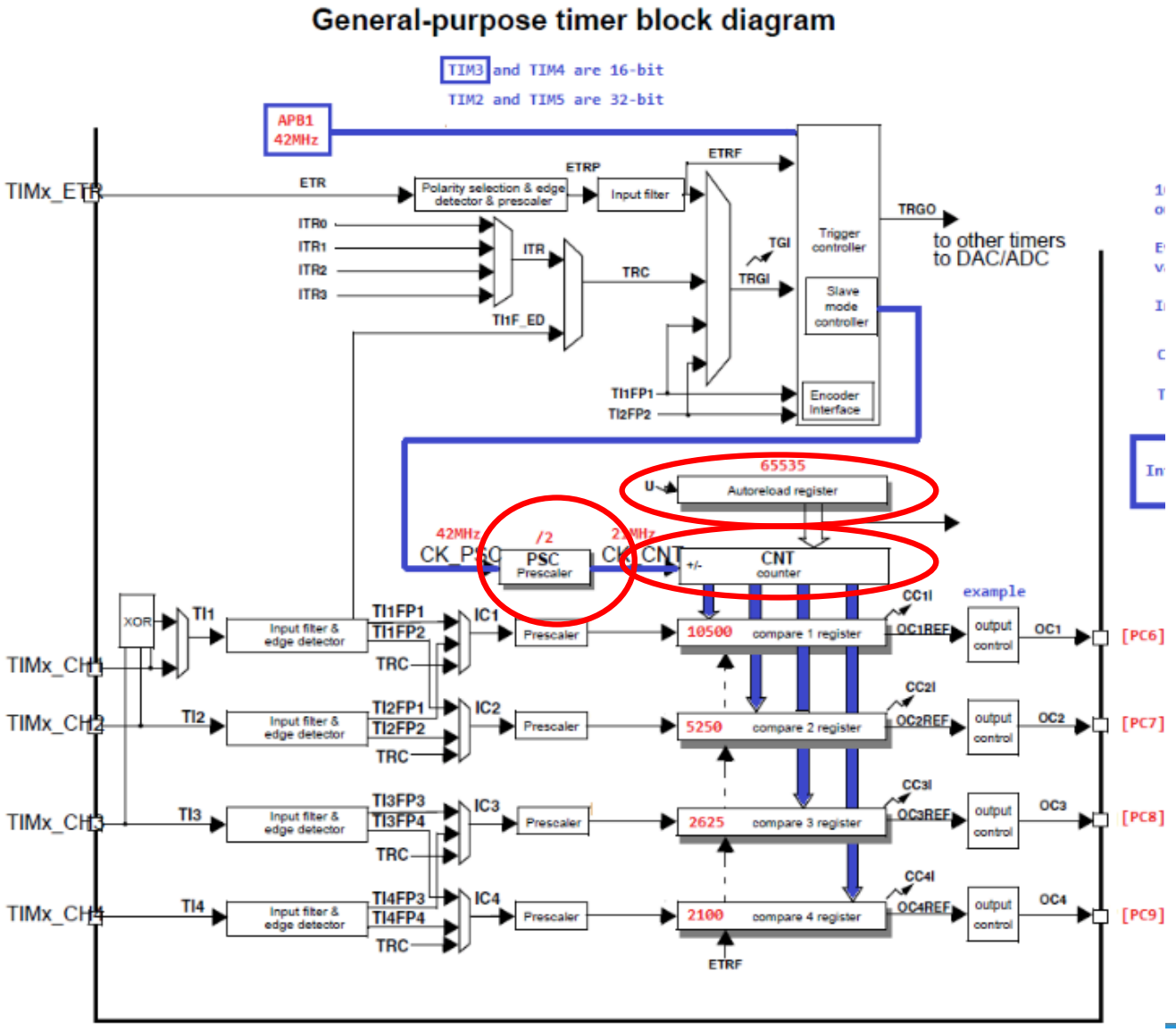
Principali Unità di un TIM

- I tre registri fondamentali nel counter sono:
  - Counter Register – contenente il valore corrente del contatore
  - Prescaler Register – registro al cui interno l'utente scrive (via software) il valore di divisione del clock
  - Auto-Reload Register – registro che contiene il valore finale di conteggio
- Dal Manuale:
  - "In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event."
  - "The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It can be changed on the fly. The new prescaler ratio is taken into account at the next update event"

Basic Timer Scheme

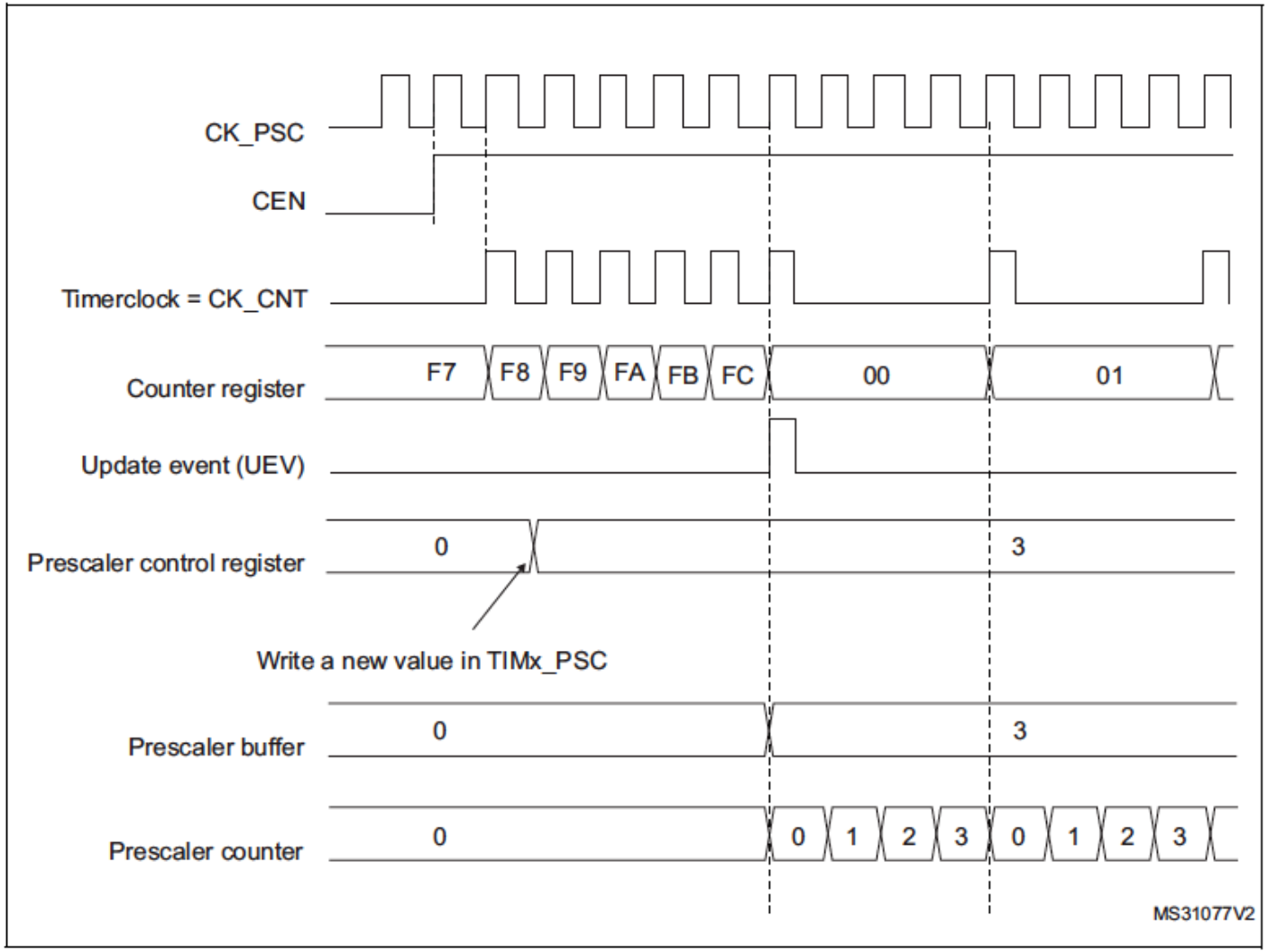


Extended Block Diagram TIM



Timing Diagram

Figure 199. Counter timing diagram with prescaler division change from 1 to 4



Modello di Programmazione

- Come ogni periferica, prima di essere utilizzata deve essere inizializzata impostando l'identificativo del TIM che si vuole usare (TIM1, TIM2, …), valore del prescaler, del periodo, ecc:

  (Vedere il manuale per informazioni dettagliate sui registri)

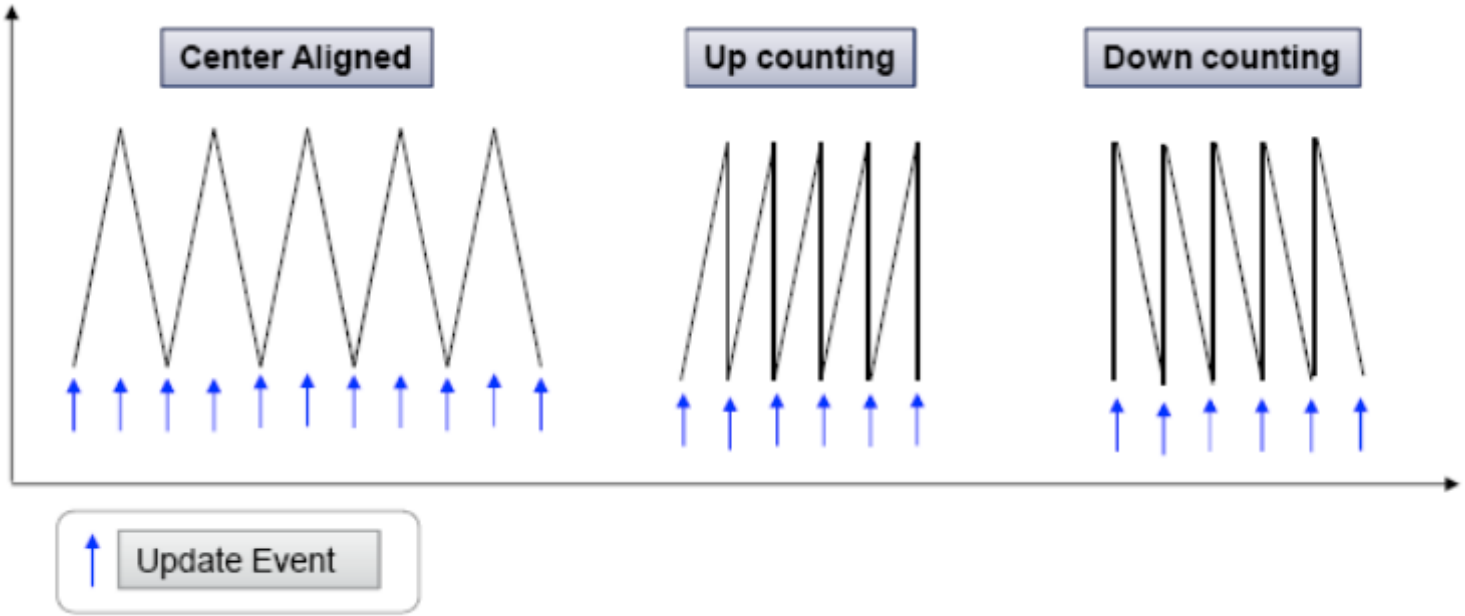
```
TIM_HandleTypeDef TimHandle;
TimHandle.Instance = TIMx;
TimHandle.Init.Period = PeriodValue;
TimHandle.Init.Prescaler = PrescalerValue;
TimHandle.Init.ClockDivision = 0;
TimHandle.Init.CounterMode = TIM_COUNTERMODE_UP;
HAL_TIM_Base_Init(&TimHandle)
```

Esempio di Definizione del Periodo del Timer

- Supponiamo di volere che il nostro Timer TIM2 invii un interrupt ogni secondo, come definire il valore del prescaler e del periodo?
  - Sappiamo che la frequenza del clock in ingresso al TIM2 è
    - 36MHz
  - Il prescaler è rappresentato su 16 bit, ciò significa che può essere impostato max a:
    - 2^16=65536.
  - Impostando il valore a 18000 avremo che la frequenza del Timer scenderà a:
    - 36MHz/18000=2KHz
  - Impostando infine il periodo a 2000 avremo proprio che ogni 2000 conteggi verrà generato un evento, ovvero:
    - 2KHz/2000=1Hz e quindi ogni secondo

Counting Modes #1

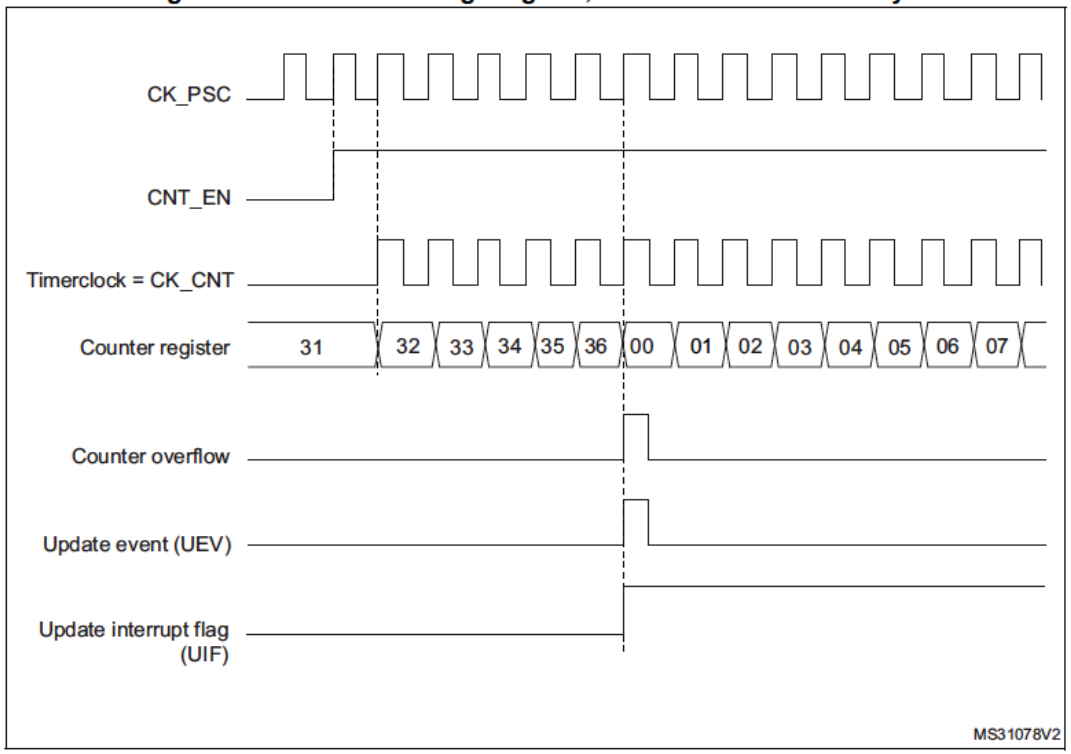
- Il counter può essere programmato per contare in tre modi differenti:
  - Up counting mode
  - Down counting mode
  - Center-aligned mode



Counter Modes #2

- In upcounting mode, e.g., the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.
- In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

Figure 200. Counter timing diagram, internal clock divided by 1



Modi di Utilizzo del TIM

- Il timer TIM può essere utilizzato in due possibili modi:
  - In Polling: una volta impostata la frequenza desiderata si controlla di continuo il valore del contatore leggendo un registro specifico(all'interno di un ciclo while).
  - Con Interrupts: si imposta il timer affinché generi un interrupt (o un evento) quando il contatore raggiunge il valore di periodo impostato.

Esempio di Applicazione

- Solitamente il TIM lo si utilizza con le interruzioni. A cosa può servire il timer TIM con Interrupts?
  - Un esempio di applicazione di un timer potrebbe essere quello di un controllore di temperatura esterna su un automobile per far si che il guidatore riceva una notifica quando la temperatura è inferiore allo zero.
  - La temperatura di certo non può cambiare drasticamente ogni millisecondo. Dunque invece di controllare in *polling* (quindi in un ciclo *while*) il contenuto del registro sovraccaricando inutilmente il microcontrollore impostiamo un timer che ogni 30 secondi, ad esempio, verifiche il contenuto del registro dove è memorizzato il valore di temperatura misurato dal sensore della macchina

Dietro le Quinte

- Quando lo sviluppatore vuole effettuare un conteggio con interrupt cosa succede realmente?

Durante la fase di inizializzazione l'utente deve configurare il timer

bit

l

18.4.1 TIMx control register 1 (TIMx_CR1)

18.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow (for TIM2 to TIM5) and if UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

l valore

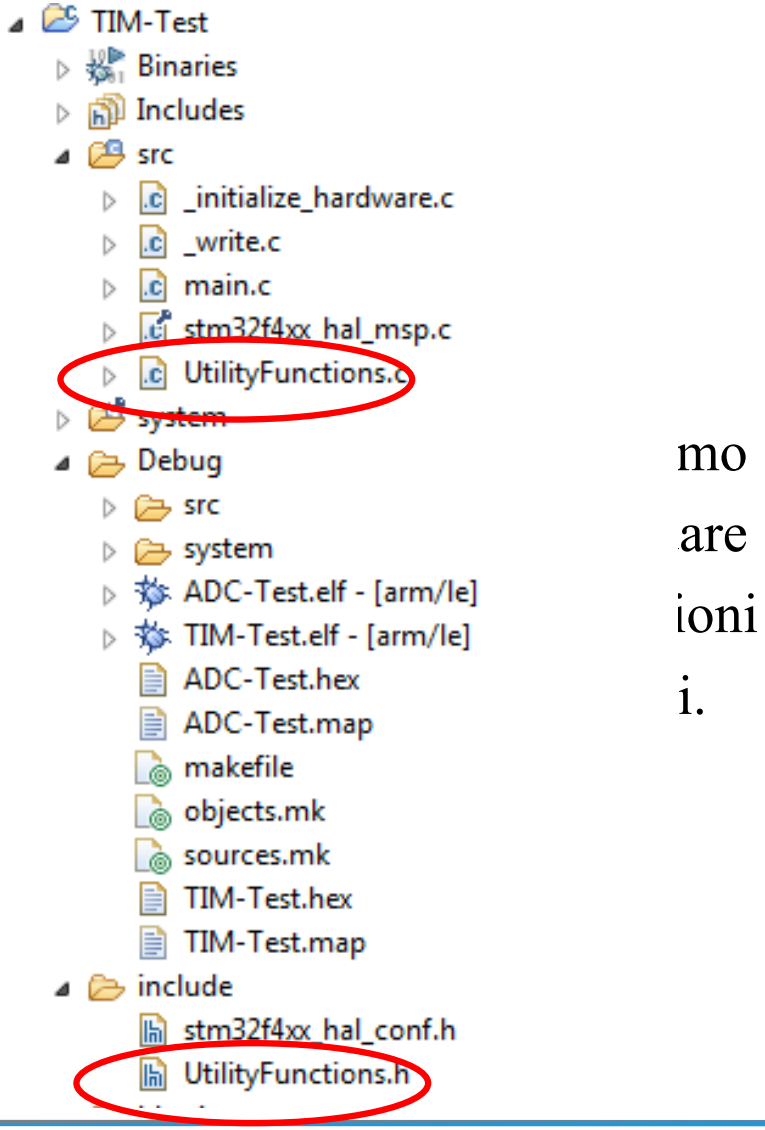
reload

egister

re

Esercizio 1/

- Usare un timer per far lampeggiare o
  scheda STM32F3-Discovery
- Proviamo entrambe le soluzioni: polli
  - Prima di cominciare dobbiamo capire co
    programma. A differenza della scorsa lez
    tutto il contenuto del nostro programma
    bene solo per esercizi semplici. Creiamo
    che ci serviranno e il corrispondente .h c

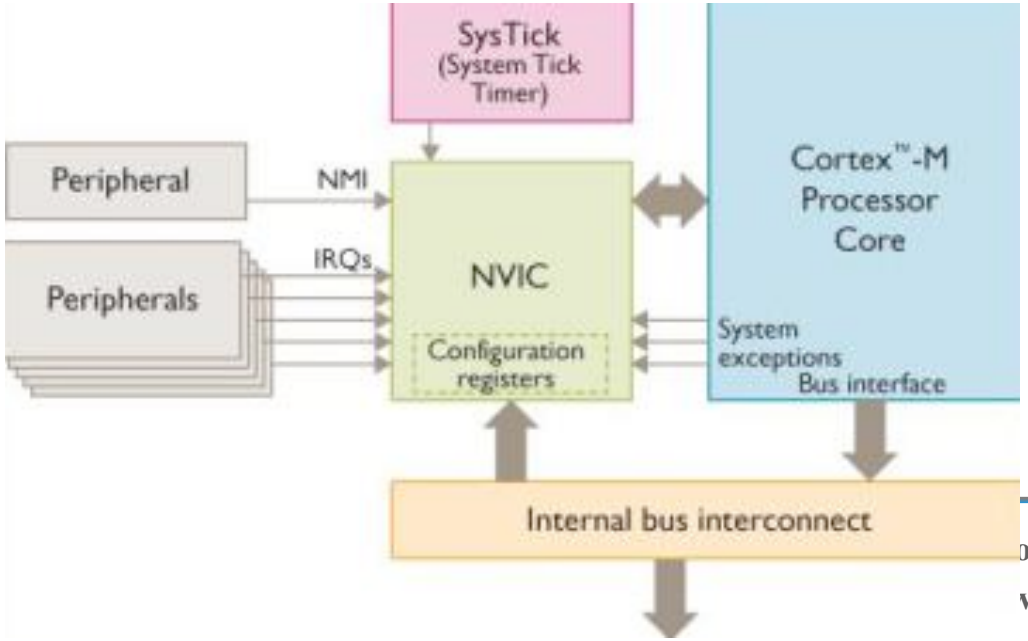


Esercizio Timer in Polling

- Come fatto nella scorsa lezione inizializziamo i Led, creiamo quindi una funzione init_led() nel file UtilityFunctions.c all'interno della quale mettere la parte di codice utile all'inizializzazione
- In seguito dobbiamo inizializzare il Timer TIM.
  - Vediamo dunque quale timer può fare al caso nostro dal manuale
  - Attiviamo il clock su quel timer
  - Definiamo il suo modello di programmazione configurando la tempificazione opportunamente.
  - Avviamo il contatore
  - Nel ciclo "while" controlliamo il contenuto del registro del contatore. Se uguale ad un secondo attiviamo/disattiviamo il led

Esercizio Timer con Interrupts #1

- Al solito dobbiamo inizializzare i LEDs.
- Prima di inizializzare il TIM dobbiamo configurare le interrupt relative al counter TIM inizializzando opportunamente il **Nested Vector Interrupt Controller (NVIC)**
- Per fare ciò dobbiamo opportunamente abilitare la **linea di interrupt** relativa al timer, e.g., TIM2, e la **priorità** di tale linea di interrupt




```
/*Configure the TIM2 IRQ priority */
HAL_NVIC_SetPriority(TIM2_IRQn, 0 ,0U);

 /* Enable the TIM2 global Interrupt */
HAL_NVIC_EnableIRQ(TIM2_IRQn);
```

Esercizio Timer con Interrupts #2

- Configurato il NVIC bisogna configurare il timer TIM2 **avviando il conteggio in modalità interrupt**

```
__HAL_RCC_TIM2_CLK_ENABLE();

TimHandle.Instance = TIM2;
TimHandle.Init.Prescaler=18000-1;
TimHandle.Init.CounterMode=TIM_COUNTERMODE_UP;
TimHandle.Init.ClockDivision=TIM_CLOCKDIVISION_DIV1;
TimHandle.Init.Period=OUR_PERIOD;

HAL_TIM_Base_Init(&TimHandle);

HAL_TIM_Base_Start_IT(&TimHandle);
```

Esercizio Timer con Interrupts #3

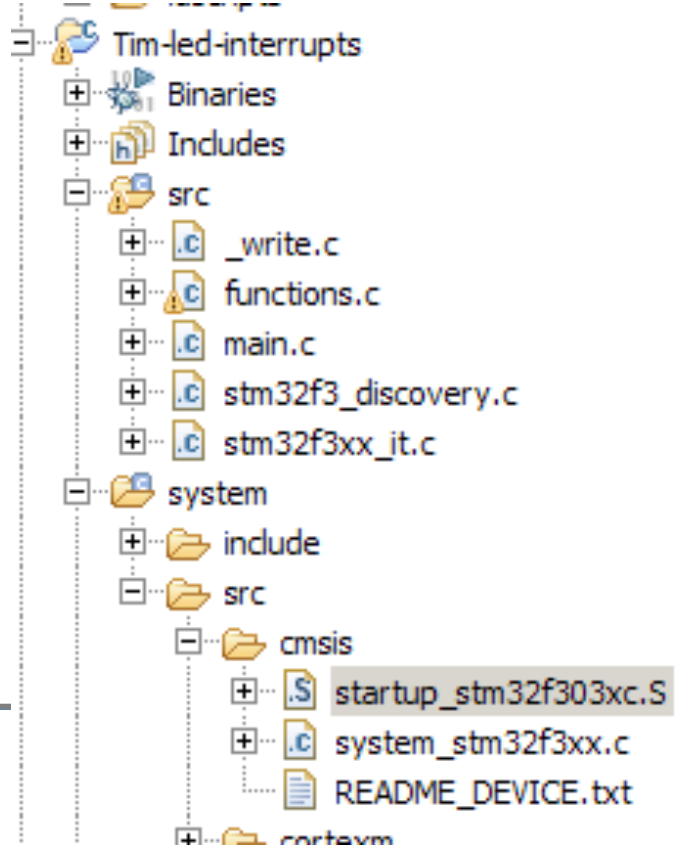
- Infine dobbiamo sovrascrivere le Interrupt Service Routine (ISR), i.e., la funzione che sarà invocato ogni qual volta il timer termina il conteggio. Tale funzione è anche detta di *callback.* Al suo interno potremo accendere e spegnere il led.
- E nel ciclo *while* del main?

```
void TIM2_IRQHandler(void)
{
  HAL_TIM_IRQHandler(&TimHandle);
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_15);
    trace_printf("Into the Callback!!! \n");
}
```

Esercizio Timer con Interrupts #4

- Come fanno ad essere chiamate tali funzioni?
  - *TIM2_IRQHandler* è l'handler di default per i processori Cortex-M. I nomi degli handler sono definiti all'interno del **file di startup** utile per la configurazione iniziale della board all'avvio.




```
.word   TIM1_UP_TIM16_IRQHandler
.word   TIM1_TRG_COM_TIM17_IRQHandler
.word   TIM1_CC_IRQHandler
.word   TIM2_IRQHandler
.word   TIM3_IRQHandler
.word   TIM4_IRQHandler
.word   I2C1_EV_IRQHandler
```

Esercizio Timer con Interrupts #4

- Come fanno ad essere chiamate tali funzioni?
  - *TIM2_IRQHandler* è l'handler di default per i processori Cortex-M. I nomi degli handler sono definiti all'interno del **file di startup** utile per la configurazione iniziale della board all'avvio. E' invocato automaticamente dal processore ARM Cortex
  - *HAL_TIM_IRQ_Handler* è la funzione dell' HAL che decidiamo di chiamare ogni qual volta sarà automaticamente invocata dal processore *TIM2_IRQHandler*. Tale funzione valuterà il motivo per cui è stato invocata l'interruzione (e.g. Timer scaduto) ed invocherà la corrispondente funzione, e.g., *HAL_TIM_PeriodElapsedCallback*
  - *HAL_TIM_PeriodElapsedCallback* è una delle funzioni di callback. Noi dobbiamo fare l'*override* di tale funzione per definire cosa fare (e.g. accendere il led)

Esercizio TIM-Button con Interrupts #1

- Svolgere un esercizio in cui il periodo del timer TIM è cambiato **on-fly** attraverso l'uso del pulsante *"User Button"* che dovrà funzionare anch'esso con interruzioni.
- L'utente deve dunque configurare il timer TIM2 come fatto in precedenza ed in seguito configurare lo User Button affinchè funzioni con interruzioni.
- In seguito all'interno dell'handler relativa allo User Button si dovrà modificare il periodo del timer in modo tale da aumentare la frequenza di lampeggio del led.

Esercizio TIM-Button con Interrupts #2

- Come configurare le interruzioni sullo User Button?
  - Similmente a quanto fatto per il timer TIM è necessario inizializzare la linea di interrupt del NVIC relativa al button unitamente alla priorità
  - Tuttavia, è importante notare che il button non è un'unità interna al SoC, dunque non configureremo una linea associata alla periferica stessa come fatto per il timer ma una **linea esterna di interrupt EXTI**.
  - In pratica, configureremo la EXTI associata alla linea di GPIO relativa al pulsante (Quella trovata nello user manual nelle prime lezioni).
  - EXTI-GPIO è semplice:

Esercizio TIM-Button con Interrupts #2

- Come configurare le interruzioni sullo User Button?

EXTI0_IRQn	EXTI0_IRQHandler	Handler for pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for pins connected to line 4

  - EXTI-GPIO è semplice.

Esercizio TIM-Button con Interrupts #3

- All'interno dell'handler EXTI si dovrà infine richiamare la funzione di inizializzazione del timer con un valore di periodo diverso
- Come fare?