UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

Time intervals measurement

A.Carini – Microcontrollers

Time interval measurement

- Measuring time intervals is often required in the implementation of control systems or digital supervision systems.
- Many DSPs and practically all µCs provide to the user one or more peripherals dedicated to time interval measurement, called *timers.*
- Timers can be:
  - *General purpose (GPT)*
  - Dedicated to specific functions (e.g., PWM modulators).
  - *Real time clocks (RTC)*
  - *Watchdog timer (WDT).*
- Measuring time is required to establish accurate *pauses* or *delays* in the code, or for the generation of periodic command signals to peripherals.

Timing of applications

- In principle, the timing of an application can be implemented with code blocks, in the form of cycles (e.g., *for, while, repeat*), inside the main program, that have the required time length. The approach is *inefficient.*
- A much more efficient approach is based on *interrupts.*
- All timers can generate an interrupt request after a *programmed* time interval.
- Any *delay, wait, cadence* function can be managed by associating the timing activities to the ISRs of these IRQs.
- Nevertheless, particular care must be taken to the other sources of interrupts, not associated with the timer, which could interfere with the stability of the timing.
- In the next example, if the perturbation of IRQ1 is large, IRQ2 could also not be serviced. If it is small, most likely it will produce just some *jitter* in the events associated with the timer.

Example

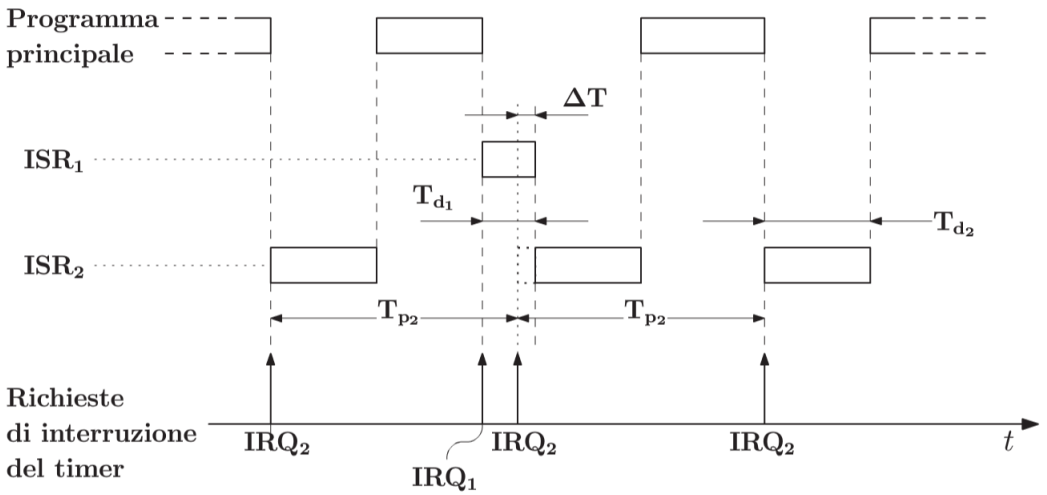

Figura 11.1: Esempio di interferenza tra una routine temporizzata tramite interruzioni e una ulteriore sorgente di interruzione. Come si vede, la periodicità di ISR_2 viene leggermente perturbata; un effetto noto come *"jitter"*.

Timing of applications

- It is possible to attenuate the interference problems with the following strategies:
  1. *Associate to timer* the IRQ with highest priority (effective only if ISR nesting is allowed).
  2. *Stop* interrupt generation at priorities higher than timer by masking (applicable only for short period of times).
  3. *Resort* to different timing mechanisms, based on timer working in *match* or *compare* mode (to be discussed later).

The timer circuit

- The timer is just a *binary counter* with n bits.
- It counts the *clock periods* or *multiples* of this quantity. Any timer has a configuration register to set the *rate* of counting, using a clock divider.
- Often, it is possible to set the continuing mode (up, down, with stop at overflow, with restart at overflow).
- Some timers allow to set the starting value in a *preload* register.
- When the value set in the *match* (or *compare*) register is reached, the timer can generate an interrupt request, or can set the status of an output bit without any CPU intervention.
- The counting frequency can be set to a fraction of the clock, with a circuit called *prescaler.*

The timer circuit

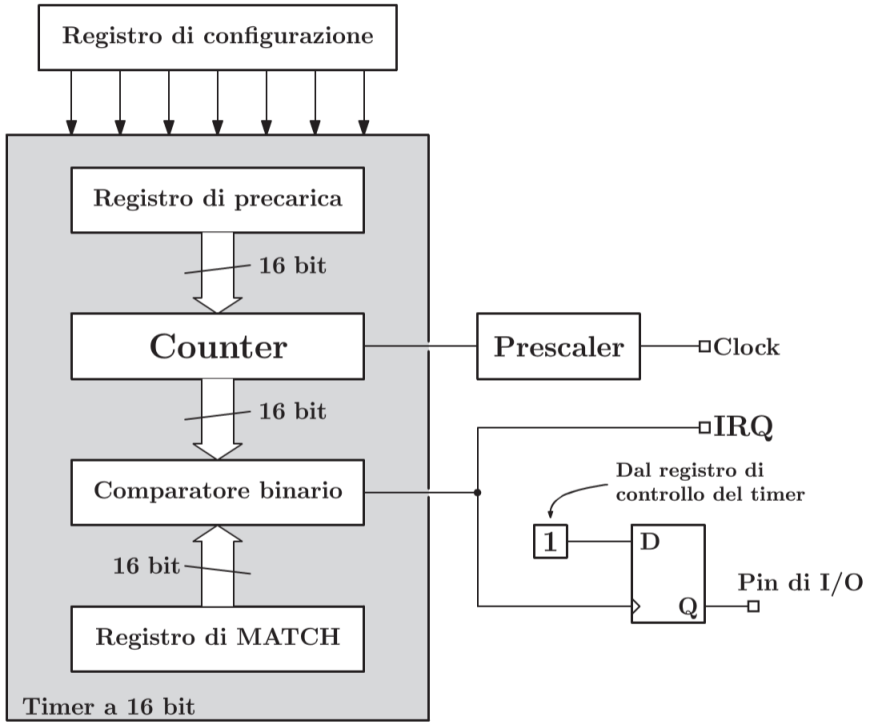

Figura 11.2: Esempio della tipica struttura circuitale di un timer a 16 bit.

Timer operation modes

- In the timer configuration register, it is possible to configure the counting mode of the timer, not only deciding the direction of counting, but also setting the restart mode when the compare value is reached.
- It is possible to impose the timer to stop till a novel start is decided by the CPU, a mode called *program controlled.*
  - This mode can be used to introduce a predetermined delay between two actions of the CPU.
- Alternatively, it is possible to automatically reload the starting value and restart counting without any CPU intervention, obtaining periodic IRQs: *free-run mode*.
  - Can be used to periodically repeat the action without any jitter.
  - Pulses of different width can be originated with the *match* register. The *match* register is loaded with the width of the pulse. At the timer start, the output bit is set, and it is reset when the *match* conditions is reached.

Example: program controlled

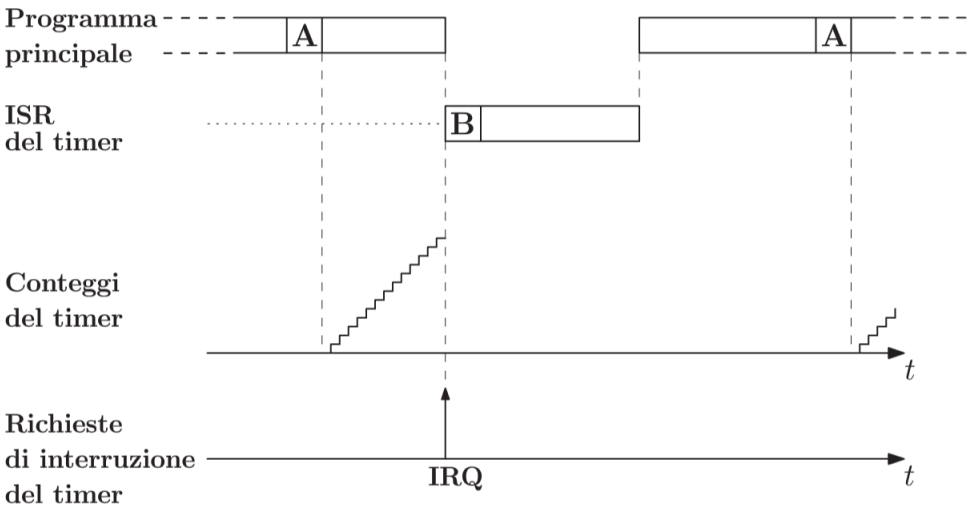

Figura 11.3: Esempio di uso del timer per introdurre un ritardo *noto e costante* tra due attività del processore, A e B. Si noti come il processore possa continuare a svolgere altri compiti prima che intervenga la IRQ.

Example: free-run

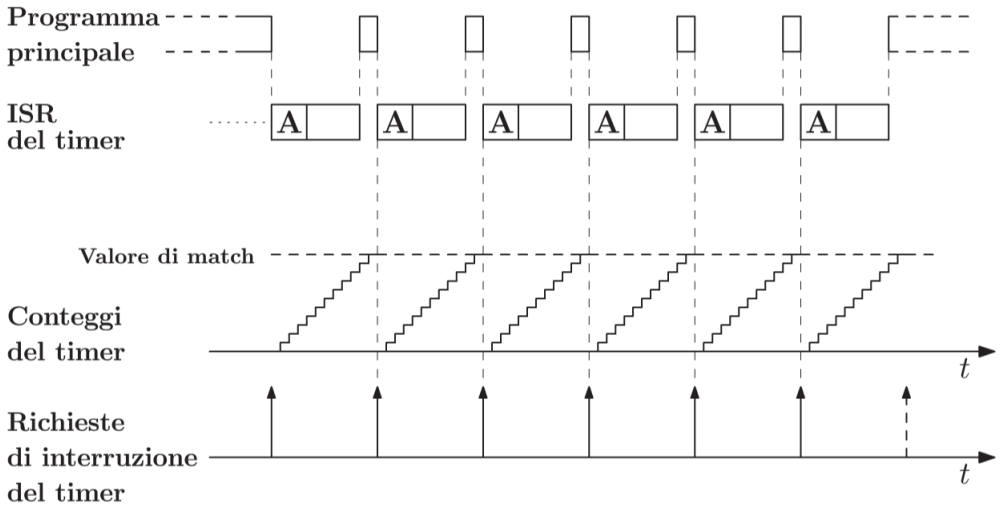

Figura 11.4: Esempio di uso del timer in modalità *"free-run"* per avviare una ISR oppure agire su un pin di I/O del microcontrollore (o DSP) con una frequenza nota. Se si agisce su un pin, poiché questo avviene *senza* intervento della CPU, si è certi che non si verificherà mai un problema di interferenza con altre ISR.

Pulse width modulation (PWM) mode

- Differently from the previous cases, the pulses have variable width but constant period.
- Thus, the value stored in the *match register* is periodically modified, e.g., by the interrupt service routine, to modify the width of the pulses.
- At the same time, the *overflow* value is kept constant, to keep constant the pulse period.
- In many cases this operation mode is facilitated by the presence of specific timers, called *PWM modulators*.

Example: PWM modulation

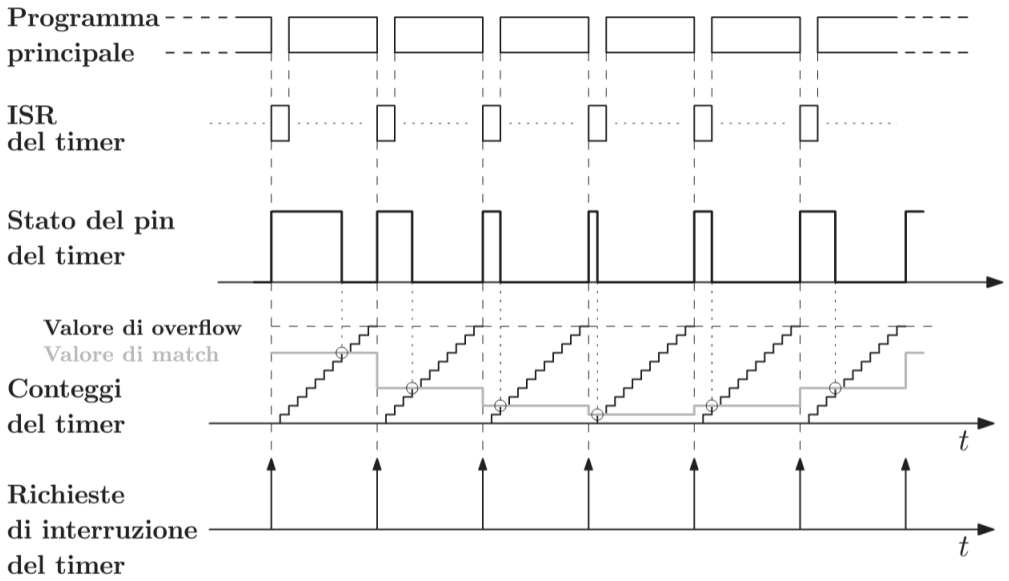

Figura 11.5: Uso del timer in modalità *"match"*, o *"compare"*, per la generazione di impulsi con durata variabile e periodo costante.

Timer in capture mode

- In *capture* mode, the timer start counting in *response* to an *external event,* a rising or falling edge on one of the external pins of processor.
- This mode allows to *measure the time width* of external events, the pulse width on an input pin measured in clock periods.
- Can be used to realize automatic synchronization systems between two signals.
- The counter normally works in *free-run*. His values are saved in appropriate registers, one for each event.
- The temporal width of the two events can be obtained by the subtraction of the two saved values.
- Provided there is at most an overflow between the two events, the subtraction in 2 complement will provide the right value of the time width.
- An interrupt can be generated at each event (A and B), but it is much more convenient to have it only at the concluding event.

Example: capture mode

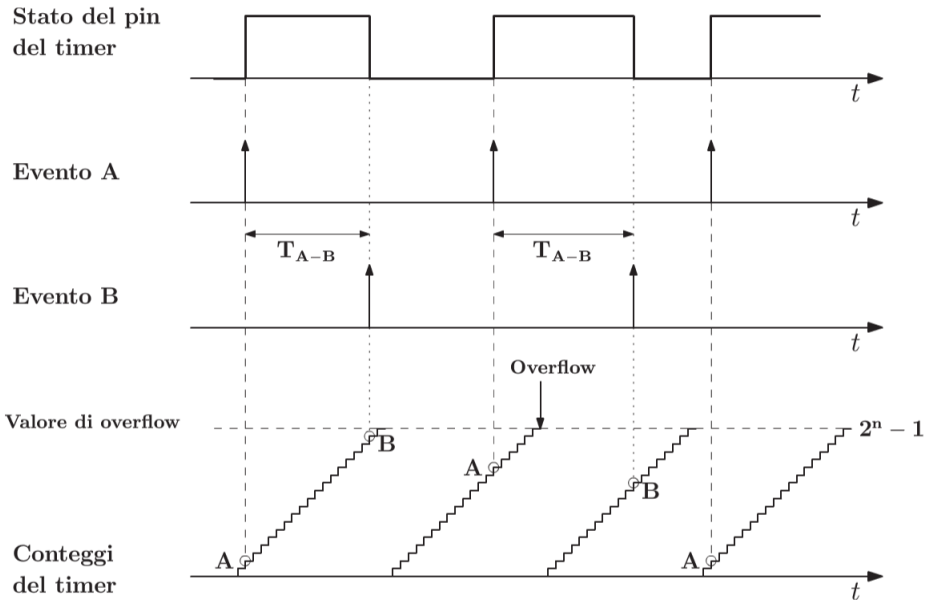

Figura 11.6: Uso del timer in modalità *"capture"* per la misura della durata dell'intervallo temporale compreso tra un evento, A, e un secondo evento, B.

Timers with specific functions

- Some timers are optimized for specific functions, and cannot be used for different tasks:
  - *Real time clock, RTC*
  - *Watchdog timer, WDT*
  - *PWM modulators.*

- The *real time clock* is a programmable timer that can generate interrupts with a predetermined period without external interactions.
- Very often it is organized with many counting registers connected, that together allow to represent intervals in seconds, minutes, hours, days.

Watchdog timer WTD

- Operates in free-run.
- Once activated, it can generate a non-maskable interrupt typically at overflow.
- The interrupt can be used for different ends.
- Quite commonly, it is used to detect and correct *deadlock conditions.*
- For this purpose, the last action of algorithm in execution is assume to be the WDT reset.
- It the algorithm is stalled, the WTD will never be reset and a non-maskable interrupt (NMI) will be generated at overflow.
- The ISR of this NMI will detect the malfunction and will typically reset the system to restart from known conditions.

Example: watchdog

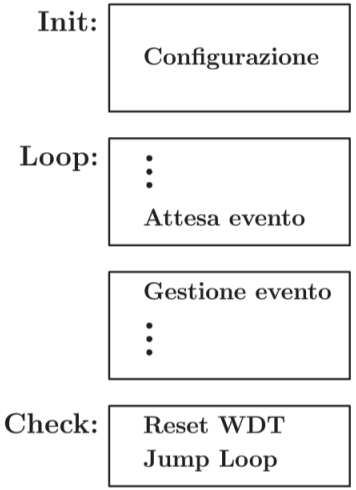

Figura 11.7: Struttura tipo di un codice che sfrutta il WatchDog Timer, WDT, per gestire l'eventuale *stallo* determinato da un evento che non si verifichi nei tempi previsti.

Counting frequency

- The first choice in configuring a timer is the *counting frequency.*
- The maximum frequency is the processor clock frequency, but it is always possible to reduce it by setting the *prescale register,* a clock divider.
- The two parameters we have to choose are the *maximum width* and the counting *resolution*.
- The ratio between the maximum measurable interval and the resolution is constant and equal to 2^n, with *n* the counter bits.
- A low value of the prescaler allows a high resolutions in separating events in time, but reduces the maximum interval. A high value allows to enlarge the interval but reduces resolution.
- Example: a 16 bit prescaler with clock of 10 MHz. The maximum interval is 429s. The highest resolution is 100ns, but the corresponding maximum interval is just 6.5ms.

Operation modes

- We shall also decide the counter operation mode.
- The first choice is between *program controlled,* i.e., *single shot* and *free-run.*
- Then we shall define if it is necessary the generation of an interrupt at *match* or at *overflow*, or if we want a *compare* mode, maybe without interrupts because we have just to change the status of an I/O pin.
- In case we need the *capture* mode we shall define the sequence of events that cause the timer value read.

Typical applications

- Command pulse generation with a programmable delay referred to specific events.
- Time measurements, frequency measurements, or other physical quantity measurements, e.g., using transducers with PWM output.
- Synchronization of signals or processes.
- PWM modulation for controlling static converters as those of electrical drivers, battery chargers, photovoltaic systems, uninterruptable power supply (UPS).

See:

- Simone Buso, «Introduzione alle applicazioni industriali di Microcontrollori e DSP» Società editrice Esculapio, 2018
  - Chapter 11