

2.1.10 Gestione di un Real Time Clock – Software di controllo in tempo reale

L'acquisizione di un segnale analogico da parte di un calcolatore digitale richiede un processo di campionamento del segnale a intervalli di tempo prestabiliti. Detti intervalli, indicati con T_c , sono detti **intervalli di campionamento**. Il numero di intervalli di campionamento nell'unità di tempo è detto **frequenza di campionamento** ($f_c = 1/T_c$). La durata minima di questo intervallo di tempo deve essere tale da garantire le seguenti operazioni:

- 1) acquisizione corretta del campione attuale
- 2) elaborazione del campione attuale

La prima operazione richiede un tempo che è la somma del tempo di conversione dell'A/D più il tempo generato dal S/H per inseguire il segnale analogico. La seconda operazione richiede un tempo variabile in funzione della complessità delle elaborazioni da effettuare sul campione.

Nei sistemi di controllo, in base al risultato delle elaborazioni si aggiunge una ulteriore operazione consistente nell'invio, agli organi di uscita, di un adeguato segnale di comando, solitamente mediante un convertitore digitale analogico (D/A).

Per questo motivo è necessario prevedere l'inserimento nel sistema di un timer che consenta la misura del tempo in una prefissata unità. Generalmente si tratta di una periferica formata da un contatore di tipo down inizializzato da software e decrementato da hardware, secondo lo schema di principio di Fig.2.1.10.1 (per semplicità di schematizzazione l'A.B., il D.B. e il C.B. sono riuniti in un unico canale di comunicazione tra CPU, Memoria e I/O; si parla globalmente di **bus del sistema**).

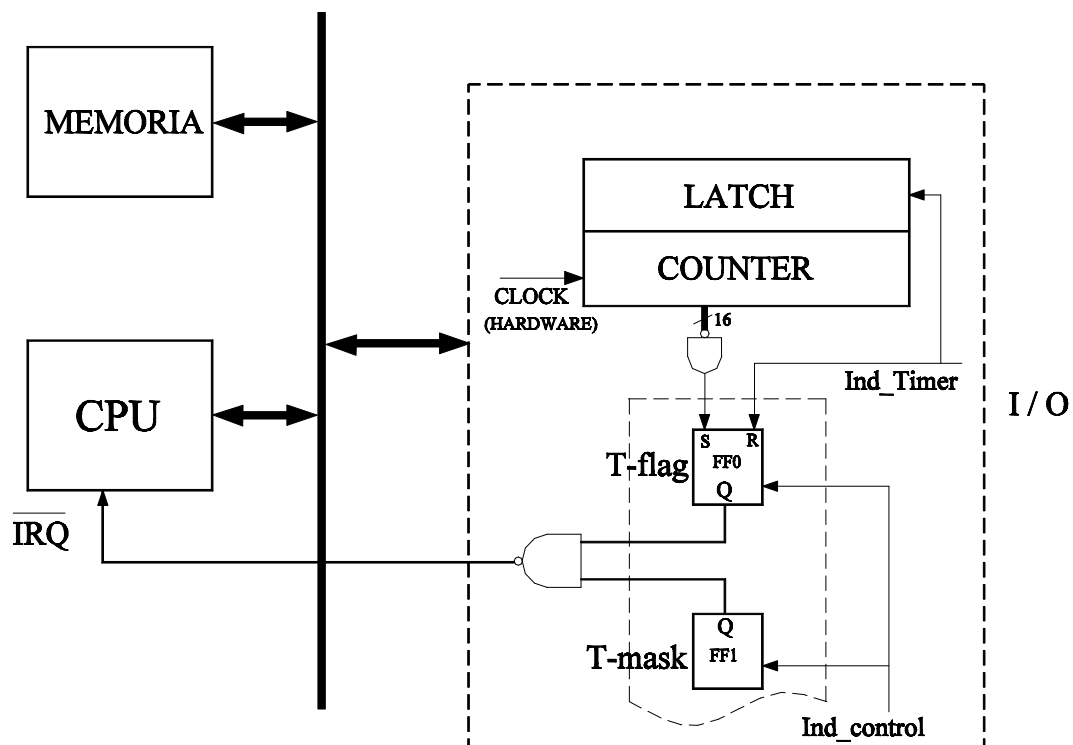


Fig.2.1.10.1 Unità periferica costituita da un Timer per la realizzazione di un dispositivo adatto a misurare il tempo.

Il valore scritto nel registro **LATCH** viene, al termine dell'istruzione che ha eseguito la scrittura, ricopiato nel **COUNTER**. L'indirizzamento del **LATCH** produce inoltre il reset del **FF T-flag (bit di stato)**. Il valore scritto nel **COUNTER** è successivamente decrementato di una unità ad ogni impulso del segnale generato da un oscillatore locale, non mostrato in figura (**CLOCK HARDWARE**). L'azzeramento del **COUNTER** pone a uno **T-flag** e, se anche **T-mask** è a uno, si genera un **richiesta di interrupt**. Con una logica adeguata, non mostrata in figura, l'azzeramento del **COUNTER** produce un nuovo trasferimento del dato scritto nel **LATCH** al contatore, talché si riproduce ancora il ciclo precedente (funzionamento **free running**). In questo modo si ha la ripetizione indefinita di intervalli di tempo tutti uguali, al termine di ognuno dei quali vi è una segnalazione di interrupt alla **CPU**. La durata di ogni intervallo è determinata dal valore scritto inizialmente nel **LATCH** e dal periodo del segnale di clock generato dall'oscillatore locale. La routine di risposta a questa richiesta di interrupt, ogni volta che viene avviata, dovrà tener conto che è trascorsa un'altra unità di tempo e agire in conseguenza all'interno del sistema.

Per rendere più concreta questa descrizione, si fa riferimento ad un esempio in cui si suppone che l'oscillatore locale produca un **CLOCK** alla frequenza di un MHz, a cui corrisponde un periodo di Clock di $1\mu\text{sec}$. Si fa inoltre l'ipotesi di voler produrre un segnale di interrupt ogni 10msec al termine dei quali avviare l'acquisizione di un campione da parte di un A/D, secondo le modalità descritte nel paragrafo precedente. Ne consegue che il valore **N** da caricare sul **LATCH** per inizializzare questa temporizzazione è (vedi Fig.2.1.10.2):

$$N = \frac{T_C}{T_{\text{Clock}}} = \frac{10000\mu\text{s}}{1\mu\text{s}} = 10000 \quad (2.1.10.1)$$

La Fig.2.1.10.2 illustra la cadenza prodotta da questa inizializzazione.

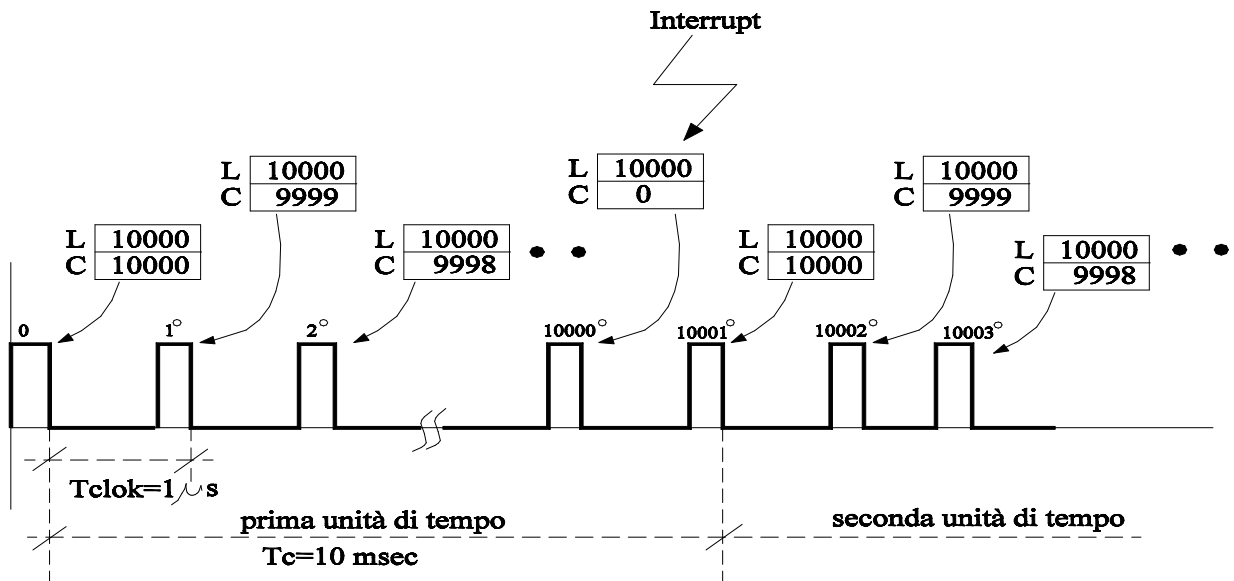


Fig.2.1.10.2. Grafico che mostra le temporizzazioni della periferica Timer che produce intervalli di campionamento di 10 msec.

Il programma che gestisce il Timer e il dispositivo periferico descritto in Fig.2.1.10.1 costituiscono un sistema per la misura del tempo, detto **Real Time Clock (RTC)**.

Relativamente all'esempio di acquisizione fatto nel paragrafo precedente, la risposta agli interrupt del Timer devono semplicemente dare il "consenso" al convertitore **A/D** di avviare una nuova acquisizione. Per questo scopo si può utilizzare il contenuto di una prefissata locazione di memoria, di indirizzo **FGRTC**, come variabile booleana (a due soli valori), ed associare al valore uno, di questa variabile, il consenso ad una nuova acquisizione ed al valore zero la inibizione a eseguire questa operazione. Questa variabile è detta **semaforo**, il valore uno ha significato di **verde**, lo zero ha il significato di **rosso**. La routine di risposta all'interrupt ponendo il semaforo a "verde" dà il consenso al programma principale di avviare una nuova acquisizione. Il programma principale avviata la nuova acquisizione riporta il semaforo a "rosso" (**FGRTC=0**) e si pone in attesa che la successiva risposta all'interrupt lo rimetta a verde consentendogli una nuova acquisizione.

La logica di questa sincronizzazione può essere rappresentata dal flow-chart di Fig.2.1.10.3.

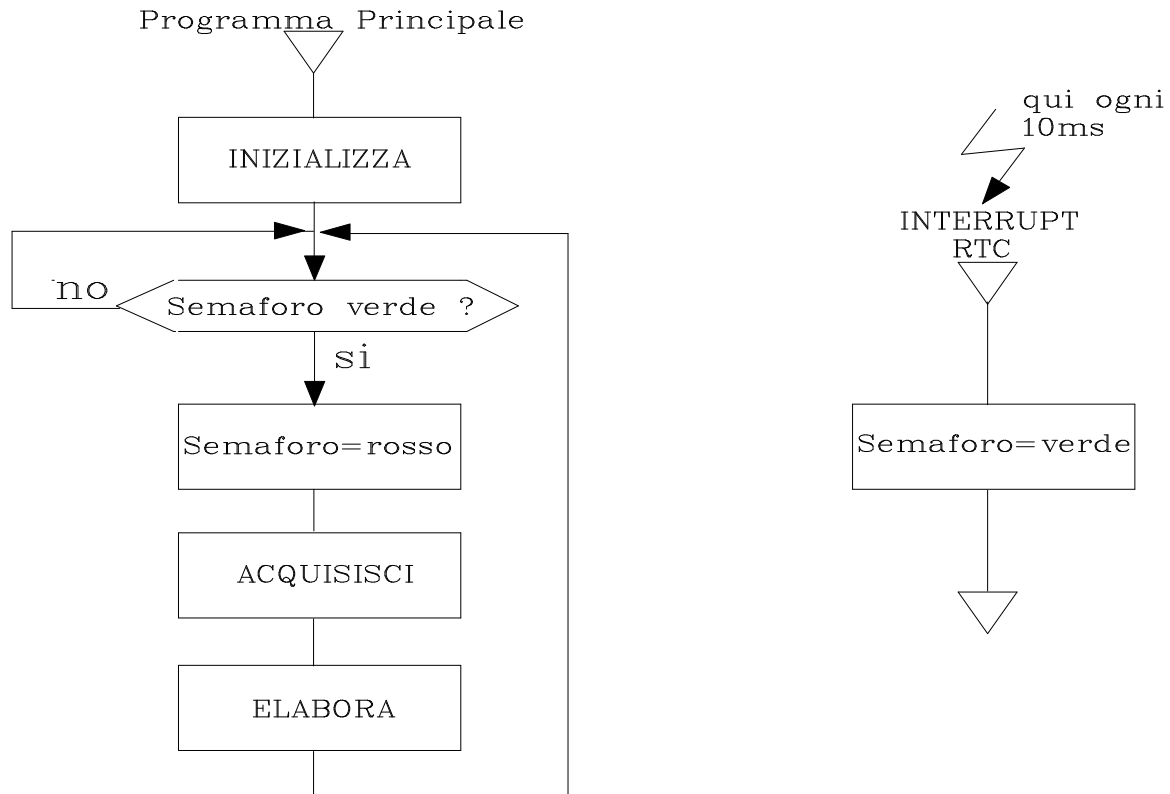


Fig.2.1.10.3. Logica di programma che produce acquisizioni in sincronismo con interrupt ogni 10 msec.

Con riferimento alla CPU fin qui utilizzata come esempio i programmi corrispondenti ai flow chart di Fig.2.1.10.3 sono:

;Programma principale

```

MAIN: JSR INIZIALIZZA
WAIT: MOVE FGRTC, D0      ;Aspetta il verde
      BEQ WAIT
      MOVE #0, D0         ;Ripristina rosso
      MOVE D0, FGRTC
      JSR ACQUISISCI      ;Nuovo campione
      JSR ELABORA        ;Elaboralo
      JMP WAIT
  
```

;Risposta interrupt RTC (Lanciata ogni 10 msec)

```

IRQ_RTC: MOVE D0, -(SP)    ;Salva
          MOVE IND_TIMER, D0 ;Reset T1-flag
          MOVE #1, D0      ;Verde.
          MOVE D0, FGRTC
          MOVE (SP)+, D0    ;Ripristina
          RTI
  
```

Il programma principale con la subroutine **INIZIALIZZA** (non scritta) pone inizialmente il semaforo a rosso, scrive il valore **N** sul **LATCH** del Timer (10000) e abilita le interruzioni sull'interfaccia ponendo a uno il bit T-mask.

Le successive due istruzioni fermano il programma principale in attesa che la routine di risposta all'interrupt ponga il semaforo verde. Quando si verifica questo evento il programma ripone a rosso il semaforo e passa alla subroutine di acquisizione del campione. In questo caso si pensa di acquisire il campione con la tecnica del *busy waiting*, dato che, in questo semplice sistema, non ci sono periferiche da controllare oltre le due descritte (Timer e A/D). La subroutine **ACQUISISCI** è, quindi, del tipo di quella descritta nel paragrafo precedente nella quale vengono rimosse tutte le istruzioni necessarie a contare i campioni indicati dalla variabile **COUNT**. Al ritorno da questa procedura il campione è acquisito e si passa alla routine **ELABORA** che può usare questo campione per tutte le elaborazioni del caso. Nel caso più semplice l'elaborazione può essere banalmente la corretta memorizzazione in un adeguato buffer di memoria.

La routine di risposta all'interrupt, a parte il salvataggio ed il ripristino del contesto, produce due operazioni importanti:

- 1) annulla il bit T-flag con una *inutile* operazione di lettura. Si osserva infatti che la logica di Fig.2.1.10.1 mostra che è sufficiente indirizzare il Timer per fare il reset di T-flag;
- 2) pone a verde la variabile semaforo, consentendo il programma principale di lanciare una nuova acquisizione.