LaurTec

ATmega 2560

Corso di programmazione per microcontrollori Atmel

Autore : *Marcello Pinna*

ID: UT0005-IT

Copyright © 2020 Mauro Laurenti

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.
Tutti i diritti di questa opera sono riservati. Ogni riproduzione ed ogni altra forma di diffusione al pubblico dell'opera, o parte di essa, senza un'autorizzazione scritta dell'autore, rappresenta una violazione della legge che tutela il diritto d'autore, in particolare non ne è consentito un utilizzo per trarne profitto.
La mancata osservanza della legge 22 Aprile del 1941 n. 633 è perseguibile con la reclusione o sanzione pecuniaria, come descritto al Titolo III, Capo III, Sezione II.
A norma dell'art. 70 è comunque consentito, per scopi di critica o discussione, il riassunto e la citazione, accompagnati dalla menzione del titolo dell'opera e dal nome dell'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.
L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Tutti i marchi citati in quest'opera sono dei rispettivi proprietari.

Indice

Capitolo VII....4
Introduzione ai Timer dell'Atmega 2560....4
Timer0 - Counter0....7
Unità Timer0 e Counter 0....8
Unità Output Compare 0 (A e B)....8
Modalità operative per la generazione di una frequenza....11
Registri di configurazione....16
History....23

Capitolo VII

Introduzione ai Timer dell'Atmega 2560

Il Timer di un microcontrollore è una periferica hardware interna impiegata per generare eventi e misurare tempi. Lavora separatamente dall'esecuzione del programma e consente di ottenere una notevole efficienza e precisione rispetto alle tecniche software basate su cicli di ritardo. Praticamente, si tratta di un registro, generalmente ad 8 o 16 bit, il cui valore viene incrementato di una unità ad ogni impulso di clock fino a raggiungere il suo valore massimo, per poi ricominciare da zero. Il passaggio dal suo valore massimo allo zero viene definito *overflow*. Il valore massimo di conteggio che può raggiungere un timer viene calcolato con la formula 2^n-1, dove *n* rappresenta il numero dei bit. Quindi, nel caso di un timer ad 8 bit il suo valore massimo è uguale a $2^8 - 1 = 255$, nel caso di un timer a 16 bit è uguale a $2^{16} - 1 = 65535$. Definendo come T_{clock} (periodo) la durata di un impulso di clock (ovvero $1/f_{clock}$), il tempo totale T impiegato dal timer per raggiungere il suo massimo valore è $T = (N + 1)\cdot T_{clock}$, dove N è il valore massimo che può assumere il registro. Il valore +1 è necessario perché il valore 0 è il valore di partenza. Gli impulsi di clock per incrementare un Timer possono essere di origine interna o esterna. Nel primo caso il Timer può essere usato per misurare il tempo, generare onde quadre, accendere un LED, realizzare un orologio (funzione *counter*). Il Timer può essere anche usato per generare un evento d'interrupt alla scadenza del tempo prefissato (funzione timer). Per un Timer si parla spesso di risoluzione, questa rappresenta la quantità di tempo più piccola che il Timer è in grado di distinguere per poter aumentare di un'unità il suo registro. Risulta evidente che il tempo totale T sarà molto piccolo, in quanto è legato al numero dei bit del Timer e alla frequenza di clock. Per rallentare il tutto si fa uso del *prescaler* per "ingannare" il microcontrollore ed ottenere tempi più lunghi. Il *prescaler* è dunque un divisore di frequenza prima del Timer.

Quanti timer/counter possiede l'ATmega2560?

L'ATmega 2560 possiede all'interno ben 6 Timer, nominati: Timer0, Timer1, Timer2, Timer3, Timer4, Timer5. I Timer 0 e 2 sono timer a 8 bit, i restanti sono Timer a 16 bit. I Timer 1,3,4 e 5 sono identici concettualmente, infatti cambiano solo i nomi dei registri di configurazione ma non il modo di impostarli. Come detto in precedenza, la sorgente di clock può essere di origine interna o esterna ed è selezionata da una logica controllata dai bit *Clock Select* (CS) di cui si parlerà in seguito. La sorgente di clock esterna, per il Timer0, 1, 3, 4 e 5 è identificata rispettivamente con i pin T0, T1, T3, T4, T5 mentre per il Timer2 e identificata con i pin TOSC1 e TOSC2.

Definizioni di Frequenza, Periodo, Duty Cycle e Fase

La frequenza *f* è una grandezza che caratterizza eventi periodici, definendo il numero di eventi al secondo. Questa grandezza la si esprime in Hertz (Hz) e la formula che la definisce è la seguente:

$f = E / T = Hz$

dove:

E = numero degli eventi
T = 1 secondo

Quindi, se un'onda rettangolare viene generata con un cambio di livello ogni secondo (E=1) la frequenza *f* è uguale ad 1/1 = 1Hz, se vengono generati due cambi di livello ogni secondo (E=2) la frequenza sarà uguale a 2/1 = 2Hz, se ne vengono generati cinque ogni secondo (E=5) la frequenza sarà uguale a 5/1 = 5Hz e così via. Dalla formula di cui sopra risulta ovvio che aumentando il numero degli eventi nell'unità di tempo la frequenza aumenterà in modo proporzionale.

Il periodo T è esattamente l'inverso della frequenza e rappresenta il tempo in cui un'onda compie una sola oscillazione e poi torna alla condizione iniziale. La formula che definisce il periodo T è la seguente:

$T = 1 / f$

dove *f* è uguale alla frequenza dell'evento periodico. Quindi, per una frequenza di 1Hz il periodo T è uguale ad 1/1 = 1 secondo, per una frequenza di 2Hz il periodo è uguale a 1/2 = 0.5s, per una frequenza di 5Hz il periodo è uguale a 1/5 = 0.2s e così via.

Un'altra grandezza utile che si definisce tipicamente per le onde quadre è il *Duty Cycle* D.C., ovvero ciclo di lavoro utile, è un rapporto espresso in percentuale tra il tempo in cui un'onda rettangolare resta a valore alto (T_{ON}) diviso per il suo periodo T. Il periodo è anche equivalente alla somma dei tempi in cui il segnale resta a valore alto (T_{ON}) e poi a valore basso (T_{OFF}). La formula per calcolare il DC in % è:

$$DC\% = \frac{T_{ON}}{T} \cdot 100$$

In Figura 1, è mostrato un dettaglio dell'onda quadra e relativo DC.

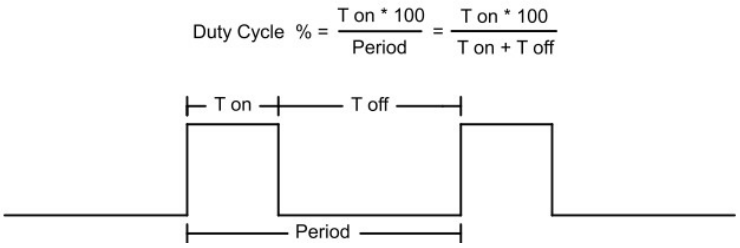


Figura 1: *Visualizzazione grafica di un'onda quadra e relativo Duty Cycle.*

Prendendo come esempio un'onda rettangolare, con una frequenza di 5Hz, applicando la formula sopra riportata, il periodo T è uguale a 1/5 = 0.20s, se il tempo T_{ON} in cui il segnale resta alto è 0,1s, il *Duty Cycle* è uguale a 0,1·100/0,20 = 50%. Quando un'onda presenta un livello alto T_{ON} uguale a quello basso T_{OFF} si parla di un'onda rettangolare simmetrica, ovvero onda quadra, con un *Duty Cycle* pari al 50%.

Un'altra grandezza importante che caratterizza un segnale periodico, oltre al periodo e la frequenza, è la fase. La fase di un'onda rettangolare, detto in modo semplice, si può definire come lo spostamento del suo centro rispetto allo zero iniziale. Si osservi la Figura 2, che confronta la modalità *Fast* PWM con la modalità *Phase Correct* PWM, in modo non invertente:

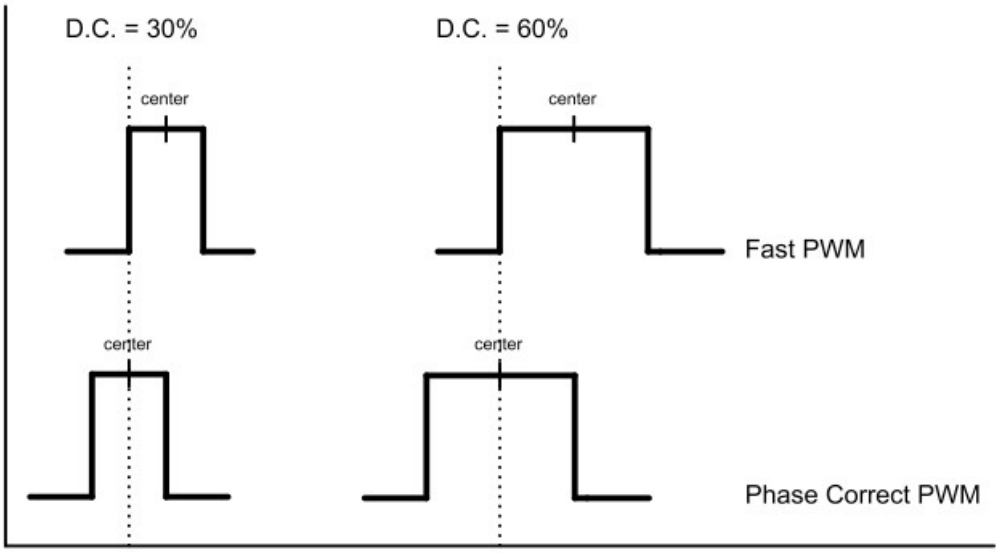


Figura 2: *Visualizzazione grafica della fase di un'onda rettangolare.*

La linea a punti rappresenta per entrambe le modalità lo zero iniziale; quello che si nota subito è che in modalità *Fast* il centro del segnale si espande solo verso il lato destro, sfasando il PWM, mentre in *Phase Correct* PWM si espande in entrambi i lati. Questo succede perché, nel primo caso, l'onda viene generata immediatamente all'accensione del sistema, ovvero quando il Timer si trova a 0; nel secondo caso essa viene generata quando il Timer, durante il suo conteggio, raggiunge un valore stabilito in precedenza, ovvero dopo l'accensione del sistema, mantenendo in fase costante il PWM. Questo comportamento è molto importante nella scelta dell'una o dell'altra modalità secondo la destinazione d'uso.

Timer0 - Counter0

Il Timer0 è un contatore generico a 8 bit, il che significa che può contare solo da 0 a 255 in modo sincrono. Possiede due unità indipendenti *Output Compare* A e B, con relativi registri chiamati rispettivamente OCR0A e OCR0B che condividono la stessa configurazione del Timer e che possono essere usati per la generazione di due segnali con frequenza uguale, ma con *Duty Cycle* diverso. Il Timer0 utilizza il divisore di frequenza (*prescaler*), che permette di rallentare la frequenza principale di clock e di ottenere tempi più lunghi, secondo i seguenti fattori: 1, 8, 64, 256, 1024. Il Timer0 è in grado di generare, utilizzando le tre sorgenti di interrupt di cui dispone, (TOV0, OCF0A, e OCF0B), segnali PWM con un periodo variabile sui pin OC0A e OC0B. In base all'impostazione dei suoi registri può lavorare in tre modi diversi: *Normal*, CTC (*Clear Timer on Compare Match*) e PWM (*Pulse Width Modulation*). Il modo PWM, ovvero la generazione di una forma d'onda con *Duty Cycle* regolabile, può a sua volta essere impostato in modo veloce (*Fast*) oppure a fase corretta (*Phase Correct*). In Figura 3 è mostrato il diagramma a blocchi del Timer0.

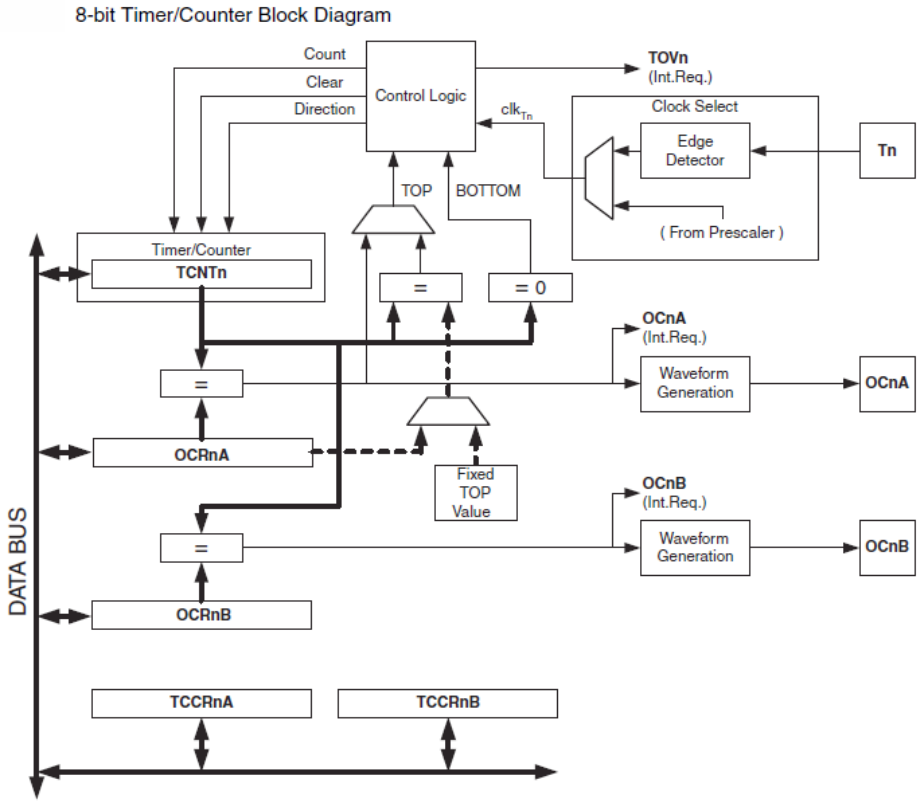


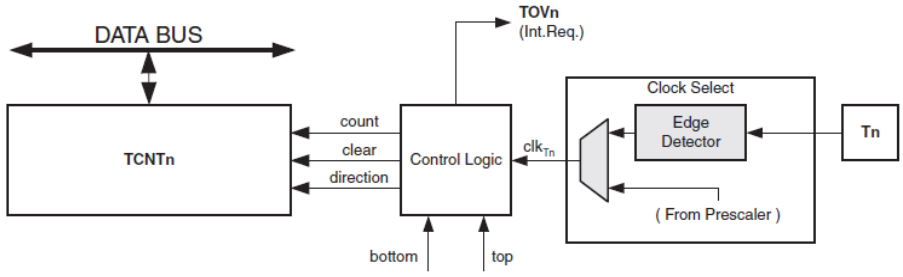
Figura 3: *Diagramma a blocchi del Timer0.*

Come si può vedere dallo schema a blocchi, nel caso specifico del Timer0 - Counter0, la periferica è formata essenzialmente da una sorgente di clock proveniente dal *prescaler* oppure da una sorgente esterna T0 (in ingresso), un selettore di clock, una logica di controllo che gestisce il conteggio del tempo e la sua direzione, un'unità Timer/Counter TCNT0, due unità *Output Compare* OCR0A e OCR0B, e da due registri TCCR0A e TCCR0B che contengono il valore con cui confrontare il Timer.

Unità Timer0 e Counter 0

Il registro TCNTn è un contatore programmabile bidirezionale ad 8 bit e rappresenta il cuore del Timer0. Secondo il modo in cui viene utilizzato può essere incrementato, decrementato o azzerato ad ogni ciclo di clock. La sorgente di clock può avere origine interna o esterna ed è selezionata dai bit *Clock Select* (CS02-CS01-CS00). Impostando a 0 il bit CS02 il contatore viene fermato. In Figura 4 sono mostrati i dettagli del diagramma a blocchi.

Figure 16-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk$_{Tn}$	Timer/Counter clock, referred to as clk$_{T0}$ in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Figura 4: *Dettagli del diagramma a blocchi del Timer0.*

Il selettore di clock gestisce la base del tempo, utilizzando il clock di sistema in arrivo dal *prescaler* oppure utilizzando un clock esterno proveniente dal pin T0. La logica di controllo si occupa di:

- Effettuare il conteggio di un'unità ad ogni ciclo di clock.
- Controllo della direzione (*direction*) ovvero incremento e decremento.
- Scrittura del registro TCNT0.
- Segnalare quando il contatore raggiunge il valore BOTTOM, TOP o MAX (secondo la modalità scelta).
- Impostare l'*Overflow* avvenuto sul bit TOV0 ed effettuare successivamente l'azzeramento del registro stesso (*clear*)

Unità Output Compare 0 (A e B)

Le unità output compare ad 8 bit A e B, verificano continuamente l'uguaglianza tra il valore contenuto nel timer TCNT0, valore che viene costantemente aggiornato durante la fase di conteggio e quello impostato inizialmente nei rispettivi registri OCR0A e OCR0B. Quando il valore di TCNT0 è uguale a quello di OCR0A o OCR0B le unità segnalano l'avvenuta corrispondenza alla CPU per mezzo dei Flag OCF0A e OCF0B posti ad 1. Il contatore viene anche azzerato, qualora la modalità operativa scelta preveda quest'ultima opzione. Nel caso siano abilitati gli interrupt, i Flag sono azzerati automaticamente dall'hardware o, se necessario, manualmente tramite software. Entrambe le unità

possiedono un doppio buffer che viene abilitato solo in modalità PWM e che serve a sincronizzare i registri OCR0A e OCR0B, impedendo la presenza di segnali incompleti o non simmetrici; in modalità NORMAL e CTC questo buffer viene disabilitato. Quando il doppio buffer è abilitato, la CPU accede solo al buffer dei due registri mentre, se disabilitato, accede direttamente ai due registri. In Figura 5 viene mostrato il diagramma a blocchi:

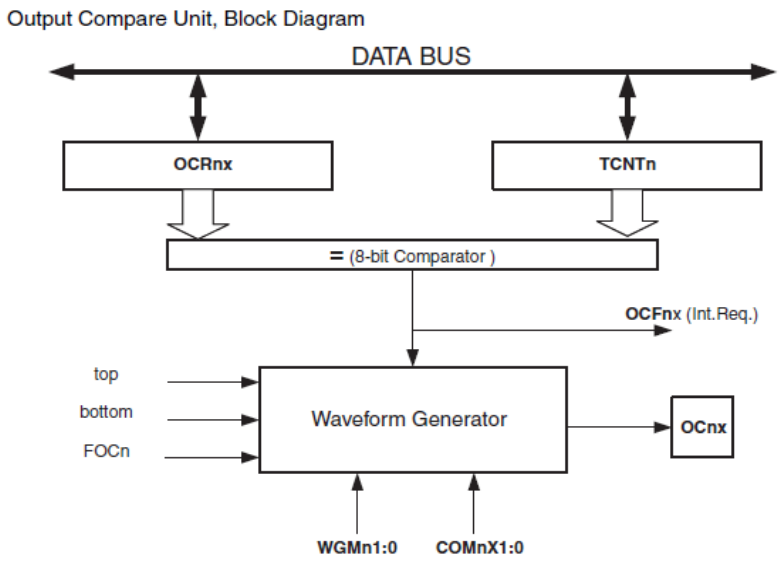


Figura 5: *Dettagli del diagramma a blocchi delle unità Capture e Compare.*

Come si evince dal diagramma, è presente un'unità che confronta il valore del Timer TCNT0 con il valore impostato precedentemente nel registro OCR0A (OCR0B). Nel caso di uguaglianza, viene sempre settato il Flag OCF0A (OCF0B) indipendentemente dal fatto che le interruzioni siano o meno abilitate. In base alla modalità scelta tramite i bit WGM e COM, che compongono il *waveform generator*, la forma d'onda ottenuta viene inviata in uscita al pin OC0A (OC0B).

Definizioni usate in questo capitolo

BOTTOM	valore minimo del conteggio del Timer0 = 0 = 0x00
MAX	valore massimo del conteggio del Timer0 = 255 = 0xFF
TOP	valore di confronto maggiore di 0, impostato nel registro OCR0A o nel registro OCR0B, usato per limitare la risoluzione in bit del contatore (può essere anche uguale a MAX)
TCNT0	Timer/Counter Register 0, ovvero il Timer0/Counter0
TIMSK0	Timer/Counter Interrupt Mask Register, contenente i bit OCIE0B, OCIE0A e TOIE0
OCIE0B	Output Compare Interrupt Enable canale B (bit di abilitazione dell'interrupt)
OCIE0A	Output Compare Interrupt Enable canale A (bit di abilitazione dell'interrupt)

TOIE0	Timer/Counter Overflow Interrupt Enabled (bit di abilitazione dell'interrupt)
TIFR0	Timer/Counter 0 Interrupt Flag Register, contenente i bit OCF0B, OCF0A e TOV0
OCF0B	Output Compare Flag canale B (flag di avvenuta uguaglianza, impostato ad 1 quando essa avviene)
OCF0A	Output Compare Flag canale A (flag di avvenuta uguaglianza, impostato ad 1 quando essa avviene)
TOV0	Timer/Counter Overflow Flag (flag di controllo dell'*overflow*, impostato ad 1 quando esso avviene)
TCCR0A	Timer/Counter Control Register A
TCCR0B	Timer/Counter Control Register B
OCR0A	Output Compare Register A (il suo valore viene confrontato con quello contenuto nel registro TCNT0 durante il conteggio)
OCR0B	Output Compare Register B (il suo valore viene confrontato con quello contenuto nel registro TCNT0 durante il conteggio)
WGM00	Wave Generation Mode (bit per la selezione della modalità operativa, ovvero per la generazione della forma d'onda desiderata)
WGM01	Wave Generation Mode (bit per la selezione della modalità operativa, ovvero per la generazione della forma d'onda desiderata)
WGM02	Wave Generation Mode (bit per la selezione della modalità operativa, ovvero per la generazione della forma d'onda desiderata)
COMA00	Compare Output Mode (bit di controllo del pin output compare OC0A)
COMA01	Compare Output Mode (bit di controllo del pin output compare OC0A)
CS00	Clock Select (bit per la selezione della sorgente di clock e del prescaler)
CS01	Clock Select (bit per la selezione della sorgente di clock e del prescaler)
CS02	Clock Select (bit per la selezione della sorgente di clock e del prescaler)
OC0A	pin di uscita Output Compare A, corrispondente a PB7
OC0B	pin di uscita Output Compare B , corrispondente a PG5
I-bit	bit 7 GIE (Global Interrupt Enable) del registro Status SREG

Modalità operative per la generazione di una frequenza

Poiché le unità *Output Capture* A e B sono identiche, nelle descrizioni che seguiranno prenderò come riferimento solamente l'unità A.

Le modalità operative di lavoro previste per la generazione di una frequenza sono quattro:

1. Normal
2. CTC
3. Fast PWM (PWM mode)
4. Phase Correct PWM (PWM mode)

La scelta di una modalità di lavoro è funzione della combinazione dei bit WGM00, WGM01 e WGM02, mentre il modo con cui la forma d'onda viene generata è funzione dei bit COM0A0 e COM0A1. Le configurazione e relativi bit sono discussi nel paragrafo "Registri". In ultimo ricordo che, nelle modalità PWM, alla frequenza generata, è sempre associato un *Duty Cycle*, rappresentato dal valore assegnato al registro OCR0A. Se la frequenza scelta appartiene a quelle fisse, generate solo dal prescaler (TOP = MAX), si può modificare il *Duty Cycle*, aumentando o diminuendo il valore di OCR0A da 0 a 255. Se la frequenza scelta è variabile, (TOP = OCR0A), si ottiene come conseguenza un *Duty Cycle* fisso uguale al 50%. Vediamo in dettaglio le varie modalità:

NORMAL

Il Timer TCNT0 conta sempre per incremento da BOTTOM a MAX e quando questo valore viene superato (*Overflow*) riparte da BOTTOM. Il timer non viene mai ripulito dai suoi valori precedenti, di conseguenza i dati già contenuti vengono semplicemente sovrascritti. Dopo l'*Overflow*, nello stesso ciclo di clock in cui TCNT0 torna a BOTTOM, viene impostato ad 1 il Flag TOV0. Questo Flag viene continuamente impostato e mai azzerato con un Clear, in quanto si comporta come un nono bit, a meno che non lo si azzeri manualmente oppure lo si usi in un evento interrupt sull'*Overflow* che esegue questa operazione automaticamente via hardware. La modalità NORMAL non è molto usata poiché non garantisce molta precisione nel tempo. È sconsigliata per la generazione di forme d'onda, a meno che non si utilizzi l'evento interrupt che garantisce una prestazione superiore.

CTC (Clear Timer on Compare match)

Questa modalità viene usata per modificare, ovvero abbassare, la risoluzione del contatore allo scopo di generare un'onda quadra simmetrica non invertente, con frequenza variabile ma con un *Duty Cycle* costante uguale al 50%. In questo caso TOP è uguale al valore di OCR0A. Il Timer conta sempre per incremento da BOTTOM a TOP e poi riparte da BOTTOM. Ad ogni corrispondenza tra il valore del contatore ed il valore TOP viene impostato ad 1 il Flag OCF0A, viene aggiornato il registro OCR0A e contemporaneamente viene azzerato il Timer stesso. Questo Flag non è mai azzerato dal

Timer e può essere usato per generare un interrupt, dove viene azzerato automaticamente dall'hardware o manualmente tramite software. L'andamento grafico del conteggio assume una forma di triangolo rettangolo continua nel tempo, ovvero viene generata una forma d'onda a dente di sega. Usare questa modalità permette un grande controllo della frequenza in uscita. Modalità utile, ad esempio, per far lampeggiare un LED. La frequenza della forma d'onda viene calcolata con la seguente formula:

$$f_{OCnx} = \frac{f_{CLK-IO}}{2 \cdot N \cdot (1 + OCRnx)}$$

dove N rappresenta il fattore di prescaler.

Da questa formula si evince che la frequenza in uscita sul pin OC0A, per un valore di OCR0A = 0, è sempre uguale alla metà di quella in ingresso; quella più alta si ottiene solo con un prescaler N = 1. In Figura 6 è riportato il relativo diagramma temporale:

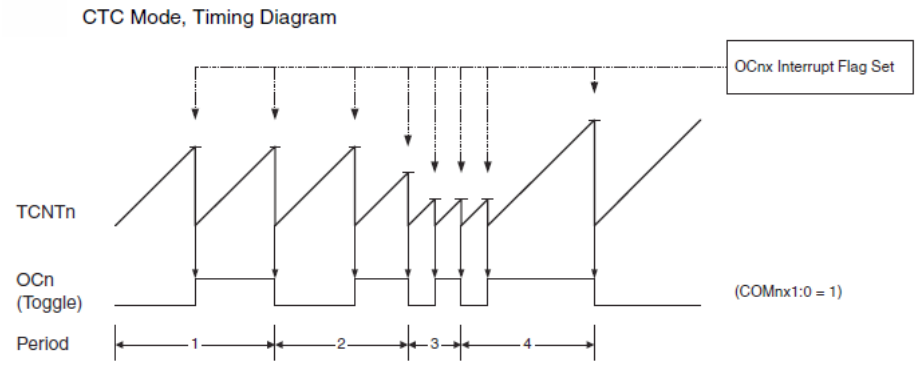


Figura 6: *Diagramma temporale della Modalità CTC.*

Da questo diagramma si può facilmente vedere come viene generato un segnale (onda quadra) con un *Duty Cycle* fisso del 50% senza utilizzare il PWM. Tramite l'impostazione dei bit COM0A1 e COM0A0 in modo "*Toggle OC0A on Compare Match*", ogni volta che avviene la corrispondenza tra il valore conteggiato dal timer e quello impostato nel registro OCR0A, si ottiene un'inversione (*toggle*) dello stato logico del pin OC0A e l'azzeramento del timer stesso. La durata di quest'onda, o meglio dire il suo periodo, è in funzione del valore TOP, più è alto questo valore, maggiore sarà il periodo del segnale e minore è la frequenza generata.

FAST PWM

Questa, si può definire una modalità ibrida tra la NORMAL e la CTC. Dalla Figura 7 si evince che esistono due modi per generare un Fast PWM: il modo 3, dove il valore TOP è uguale al valore MAX ed il modo 7, dove il valore TOP è uguale al valore OCR0A. In entrambi i casi, il timer TCNT0 conta sempre da BOTTOM a MAX per intercettare TOP e poi riparte da BOTTOM. Se TOP è uguale a MAX, al verificarsi dell'uguaglianza tra il valore del Timer e OCR0A, viene impostato ad 1 il Flag TOV0 e viene aggiornato OCR0A. Se TOP è uguale a OCR0A viene impostato ad 1 solo il Flag OCF0A. Entrambi i Flag non sono mai azzerati dal Timer ma possono essere usati per generare un interrupt. In questo caso vengono azzerati automaticamente dall'hardware o manualmente tramite software. L'andamento grafico del conteggio assume una forma di triangolo rettangolo continua nel tempo, ovvero viene generata una forma d'onda a dente di sega. Se TOP è uguale a MAX questo metodo prevede, per ogni valore del prescaler scelto, una frequenza fissa ed un *Duty Cycle* proporzionale al valore di OCR0A. Se TOP è uguale a OCR0A si ottiene una frequenza variabile ed un *Duty Cycle* fisso uguale al 50%. In base alle rispettive formule di calcolo, questa frequenza risulta doppia rispetto a quella ottenuta dalla *phase correct* ed è preferibile per applicazioni ad alta frequenza, in cui non sia importante la fase del segnale PWM, in quanto quest'ultima cambia al variare di OCR0A, ovvero del *Duty Cycle* stesso. Può essere quindi usata, ad esempio, per realizzare un DAC (*Digital Analog Converter*) o controllare l'intensità di un LED. La frequenza del PWM viene calcolata con la seguente formula:

$$f_{OCnxPWM} = \frac{f_{clk-IO}}{N \cdot 256}$$

dove N rappresenta il fattore di prescaler. In Figura 7 è riportato il relativo diagramma temporale:

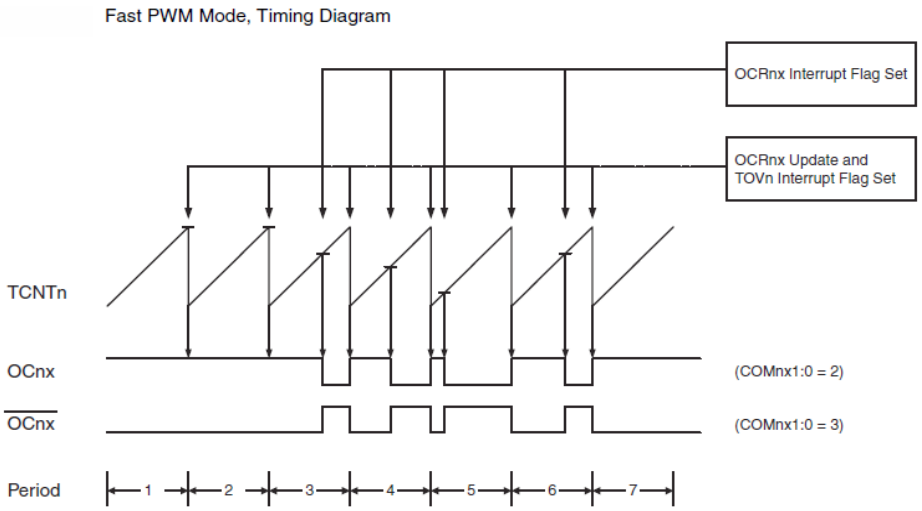


Figura 7: *Diagramma temporale della modalità FAST PWM.*

Tramite l'impostazione dei bit COM0A1 e COM0A0 in modo "*Clear OC0A on Compare Match - Set OC0A at Bottom*", l'onda quadra viene generata al raggiungimento, da parte di TCNT0, dei valori BOTTOM e TOP, causando allo stesso tempo un'inversione dello

stato logico del pin OC0A. In modalità *Set OC0A on Compare Match - Clear OC0A at Bottom* l'onda quadra viene generata dal raggiungimento, da parte di TCNT0, dei valori TOP e BOTTOM, causando la stessa inversione dello stato logico del pin OC0A.

Phase Correct

Esistono due modi per generare un *Phase Correct PWM*: il modo 1 (TOP = MAX) ed il modo 5 (TOP = OCR0A). In entrambi i casi, il timer TCNT0 conta ciclicamente da BOTTOM a TOP e da TOP a BOTTOM senza mai fermarsi. L'andamento grafico del conteggio assume in questo caso una forma di triangolo isoscele continua nel tempo.
Se TOP è uguale a MAX viene aggiornato il registro OCR0A ed impostato ad 1 il Flag OCF0A. Se TOP è uguale al valore impostato nel registro OCR0A, viene impostato ad 1 solo il Flag OCF0A. Ogni volta che TCNT0 raggiunge BOTTOM viene impostato ad 1 il Flag TOV0. Entrambi i Flag non sono mai resettati, ma possono essere usati per generare un interrupt, in qual caso vengono azzerati automaticamente dall'hardware o manualmente tramite software. Se TOP è uguale a MAX, questo metodo prevede, per ogni valore di prescaler scelto, una frequenza fissa ed un *Duty Cycle* proporzionale al valore di OCR0A. Se TOP è uguale a OCR0A si ottiene una frequenza variabile ed un *Duty Cycle* fisso uguale al 50%. In base alle rispettive formule di calcolo, questa frequenza risulta dimezzata rispetto a quella ottenuta dalla Fast PWM ed è richiesta per applicazioni dove è importante mantenere anche la fase costante, ovvero che essa non cambi al variare del *Duty Cycle* (valore OCR0A), ad esempio la regolazione PWM in motori a corrente continua. La frequenza del PWM viene calcolata con la seguente formula:

$$f_{OCnxPCPPWM} = \frac{f_{clk-IO}}{N \cdot 510}$$

dove N rappresenta il fattore di prescaler. In Figura 8 è riportato il relativo diagramma temporale:

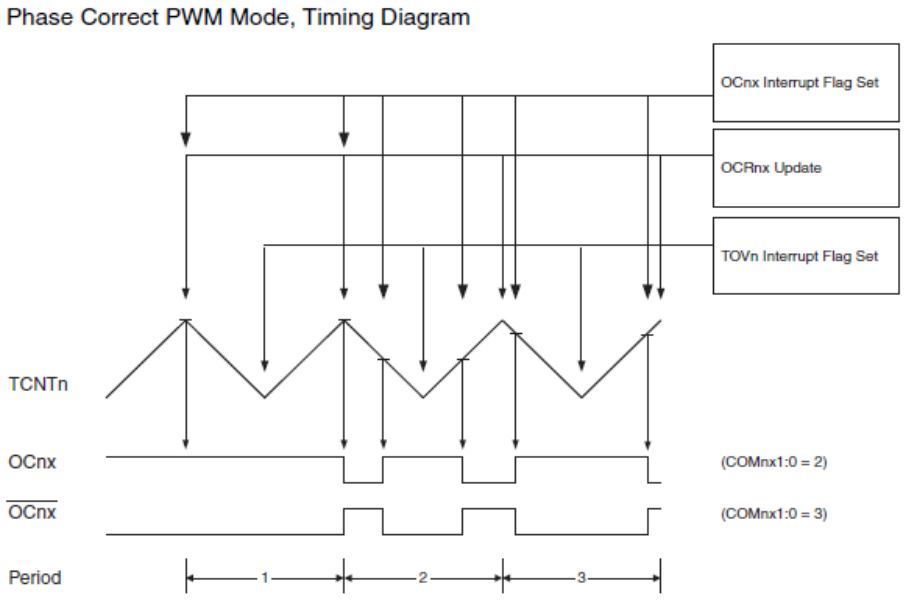


Figura 8: *Diagramma temporale della modalità Phase Correct PWM.*

Tramite l'impostazione dei bit COM0A1 e COM0A0 in modo "*Clear OC0A on Compare Match when up-counting - Set OC0A on Compare Match when down-counting*", l'onda quadra viene generata al raggiungimento, da parte di TCNT0, dei due valori TOP, generando per ognuno di essi, un'inversione dello stato logico del pin OC0A. In questa modalità il livello di uscita è alto (VCC) quando il timer conta da TOP a BOTTOM e da BOTTOM a TOP. In modalità "*Set OC0A on Compare Match when up-counting - Clear OC0A on Compare Match when down-counting*" il livello di uscita è alto (VCC) quando il timer conta da BOTTOM a TOP e da TOP a BOTTOM.

Sia il modo Fast PWM che il modo *Phase Correct PWM* possono essere usati per generare frequenze diverse da quelle fisse, come nel modo CTC.

Per semplificare i calcoli in tutte le modalità operative, ho creato un foglio di lavoro in OpenOffice, chiamato Timer-Counter0, che prevede tutte queste possibilità. Il file può essere scaricato assieme agli esempi del corso.

Registri di configurazione

Nei paragrafi seguenti sono descritti i dettagli dei registri precedentemente introdotti.

TCCR0A - TCCR0B: Timer Counter Control Register A - B

Questi sono i due registri di controllo del Timer/Counter0 che permettono, in base alla combinazione dei bit presenti, la generazione della forma d'onda scelta. In Figura 9, 10, 12 sono mostrati i dettagli dei rispettivi bit e relativa descrizione.

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 9: *Timer/Counter Control Register A.*

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 10: *Timer/Counter Control Register B.*

Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note: 1. MAX = 0xFF
2. BOTTOM = 0x00

Figura 11: *Descrizione dei bit Waveform Generation Mode.*

La prima cosa che si nota è che i bit 0 e 1 (WGM00 e WGM01) fanno parte del registro TCCR0A mentre il bit 2 (WGM02) fa parte del registro TCCR0B (il primo numero dopo WGM identifica il timer utilizzato, il numero successivo il bit considerato). Ciò implica una particolare attenzione in fase di scrittura di un programma, al fine di evitare errori di impostazione nella generazione della forma d'onda. Si tenga inoltre presente che la forma d'onda può essere generata quando il contatore raggiunge il valore BOTTOM, MAX oppure TOP (TOP può essere anche uguale a MAX). A titolo di esempio, consideriamo

le modalità 3 e 7 della Figura 12. Entrambe generano un Fast PWM ma nel primo caso l'impostazione del bit Flag TOV0 avviene al valore MAX, nel secondo caso al valore TOP. Mentre nel primo caso basta portare ad 1 solo i bit 0 e 1 del registro TCCR0A (WGM00 e WGM01), nel secondo caso si deve portare ad 1 anche il bit 3 del registro TCCR0B (WGM02). In definitiva, la combinazione dei bit WGM00, WGM01 e WGM02 determinano la scelta della modalità operativa. Quindi, per impostare la modalità 3 basta portare ad 1 i bit relativi al solo registro TCCR0A con la seguente istruzione:

```
TCCR0A |= (1<< WGM00) | (1<< WGM01);
```

Per impostare la modalità 7 si deve portare ad 1 anche il bit WGM02 del registro TCCR0B aggiungendo nel programma un ulteriore istruzione:

```
TCCR0A |= (1<< WGM00) | (1<< WGM01);
TCCR0B |= (1<< WGM02);
```

Ovviamente, se un bit si trova inizialmente a 0 si può tralasciare di scrivere la relativa istruzione di impostazione nel main del programma, ma per chiarezza si consiglia di impostare ogni bit.

I bit 7 e 6 del registro TCCR0A (COM0A1 e COM0A0, *Compare Output Mode A*) controllano il pin in uscita OC0A (PB7), mentre i bit 5 e 4 dello stesso registro (COM0B1 e COM0B0, *Compare Output Mode B*) controllano il pin in uscita OC0B (PG5). Poiché questi pin svolgono le stesse funzioni, in quanto cambia solo il loro nome, descriverò solo la parte riguardante i bit 7 e 6 relativi all'unità A, che da questo momento definirò canale A. Lo stesso ragionamento si applica al canale B. Nelle Figure 12, 13 e 14 mostrati i dettagli relativi al canale A in modalità non PWM (Normal e CTC), Fast PWM e Phase Correct PWM:

Compare Output Mode, non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Toggle OC0B on Compare Match
1	0	Clear OC0B on Compare Match
1	1	Set OC0B on Compare Match

Figura 12: *Compare Output Mode – non PWM mode.*

Compare Output Mode, Fast PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on Compare Match, set OC0B at BOTTOM (non-inverting mode)
1	1	Set OC0B on Compare Match, clear OC0B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 121 for more details.

Figura 13: *Compare Output Mode – Fast PWM mode.*

Compare Output Mode, Phase Correct PWM Mode[1]

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on Compare Match when up-counting. Set OC0B on Compare Match when down-counting
1	1	Set OC0B on Compare Match when up-counting. Clear OC0B on Compare Match when down-counting

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 122 for more details.

Figura 14: *Compare Output Mode – Phase Correct PWM mode.*

Da queste combinazioni, si evince che se uno dei due bit COM0A1-COM0A0, o entrambi, sono impostati a 1 verrà disabilitato il modo NORMAL. Allo stesso tempo si evince il peso di questi bit nella scelta di una modalità non invertente o invertente, con *Duty Cycle* fisso oppure variabile. Se ad esempio si vuole accendere e spegnere un LED, con intervallo di un secondo utilizzando il pin OC0A, è sufficiente scegliere la modalità CTC o Fast PWM ed impostare ad 1 il solo bit COM0A0 (*Toggle OC0A on compare match*). Se invece si vuole variare la sua luminosità si può scegliere la modalità Fast PWM ed impostare ad 1 il solo bit COM0A1 (*Clear OC0A on Compare Match- Set OC0A at Bottom, duty cycle variabile*). Ora facciamo un passo indietro verso il registro TCCR0B. Prenderò in esame solo i primi tre bit meno significativi di questo registro, ovvero il bit 2, 1 e 0 (CS02, CS01, CS00). La loro combinazione definisce il fattore di *prescaler* (divisore di frequenza) secondo la Figura 15.

Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Figura 15: *Descrizione dei bit Clock Select.*

Quindi, se voglio impostare il prescaler a 1/1024 devo portare ad 1 i bit CS00 e CS02 scrivendo. Poiché CS01 si trova già a 0 scrivo semplicemente:

```
TCCR0B |= (1<<CS00) | (1<<CS02);
```

Se voglio fermare il timer basta azzerare TCCR0B scrivendo:

```
TCCR0B = 0;
```

TCNT0 : Timer/Counter Register

Registro ad 8 bit del Timer0 in cui viene aggiornato continuamente il valore del suo conteggio in modo automatico, sia in ordine crescente che decrescente. I dati possono essere sia letti che scritti.

OCR0A – OCR0B: Output Compare Register A - B

Questi sono due registri di controllo relativi alle unità Output Compare A e B, che contengono un valore ad 8 bit che viene continuamente confrontato con il valore contenuto in TCNT0. Quando questi due valori combaciano tra loro, si può generare un interrupt oppure una forma d'onda in uscita ai pin OC0A - OC0B.

TIMSK0: Timer/Counter Interrupt Mask Register

Quando i bit OCIE0B, OCIE0A e TOIE0, rispettivamente:

- Output Compare Match B Interrupt Enable
- Output Compare Match A Interrupt Enable
- Timer/Counter Overflow Interrupt Enabled

sono posti ad 1, insieme a I-bit del registro SREG, vengono abilitati gli interrupt sul *Compare Match e sull'Overflow*. Per i bit 2 e 1 l'interrupt viene generato quando il valore conteggiato da TCNT0 è uguale a quello contenuto nei rispettivi registri OCR0B-OCR0A. Per il bit 0 l'interrupt viene generato quando TCNT0 raggiunge il valore MAX, TOP o BOTTOM. In Figura 17 è mostrato il dettaglio dei bit del registro.

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

figura 17 - TIMSK0 – Timer/Counter Interrupt Mask Register

Figura 16: *Bit del registro TIMSK0 – Timer/Counter Interrupt Mask Register.*

TIFR0: Timer/Counter 0 Interrupt Flag Register

Quando subentra un evento interrupt, l'hardware lo segnala scrivendo 1 nei rispettivi Flag OCF0B, OCF0A e TOV0 di questo registro. I bit sono rispettivamente:

- Output Compare Match B Flag
- Output Compare Match A Flag
- Timer/Counter0 Overflow Flag

I relativi Flag post ad 1, permettendo l'esecuzione della ISR corrispondente. Successivamente essi vengono ripuliti automaticamente (Clear) dall'hardware stesso oppure tramite software. In Figura 18 è mostrato il dettaglio dei bit del registro.

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 17: *Flag del registro TIFR0 – Timer/Counter 0 Interrupt.*

Tempi e frequenze massime fisse ottenibili in modalità NORMAL

In modalità NORMAL, con un quarzo (frequenza di clock) da 16 MHz, quindi con un Tc (periodo) uguale a 1/16000000 = 0,0000000625s ed una risoluzione massima R di (2 ^ 8) – 1 = 255bit, il tempo massimo fisso Tmax ottenibile per ogni divisore di frequenza è dato dalla formula: Tmax = ((R + 1) * Tc * N), dove N rappresenta il valore del prescaler. La frequenza massima fissa Fmax sarà uguale a Fmax = 1/ Tmax. Ricordando che i prescaler del timer 0 sono 1, 8, 64, 256, 1024 si otterrà dunque la seguente Tabella.

N (prescaler)	Tmax (tempo max ottenibile)	Fmax (Frequenza max ottenibile)
	[s]	[Hz]
1	0,000016	62500,0
8	0,000128	7812,5
64	0,001024	976,6
256	0,004096	244,1
1024	0,016384	61,0

Calcolo di un tempo richiesto non superiore a 0,016384 secondi (N = 1024)

Se si vuole ottenere un certo ritardo che non superi il tempo massimo previsto dalla Tabella appena calcolata, per esempio 16,384ms con prescaler 1024, basta applicare questa formula:

$$TIMER0\,counts = \left[\frac{T}{N * Tc}\right] - 1$$

dove:

TIMER0 counts : numero dei conteggi da assegnare a TCNT0
T : tempo richiesto
Tc : 1/*f*c (frequenza di clock)
N : fattore di prescaler

Per fare un esempio, se si volesse ottenere un ritardo di 16ms con un quarzo (frequenza di clock) uguale a 16 MHz, si deve calcolare il numero dei conteggi necessari per ottenere questo ritardo. Dalla Tabella, risulta evidente che si deve usare il prescaler con un valore N = 1024, infatti gli altri valori N vanno scartati in quanto Tmax è inferiore al tempo richiesto. Quindi si ha:

Tc = 1/16000000 = 0,0000000625s

TIMER0 count = (0,016s / (0,0000000625s * 1024)) -1 = 249

E fin qui va bene, perché il numero dei conteggi calcolato risulta alla fine un numero intero (ricordo che tutti i registri lavorano solo con numeri interi), non è superiore a 255 ed il ritardo richiesto sta al di sotto del valore limite. Nel caso in cui il numero dei conteggi risultasse un numero decimale, accettando un errore si può troncare o approssimare per eccesso il numero, ottenendo un numero intero.
Nel caso in cui il valore fosse maggiore di 255, si dovrà necessariamente utilizzare l'evento interrupt sull'overflow o usare un timer a 16 bit piuttosto che 8. Gli esempi che saranno mostrati proporranno una soluzione a questo tipo di problema.

Indice Alfabetico

B
BOTTOM....8
C
clear....8
Clock Select....4
COM....9
COM0A0....17
COM0A1....17
counter....4
CS....4
CS00....8
CS01....8
CS02....8
CTC....7, 9, 11
D
DAC....13
Digital Analog Converter....13
direction....8
Duty Cycle....5 e segg., 13
F
Fast PWM....6, 11
M
MAX....8
N
Normal....7, 11
NORMAL....9
O
OC0A....7, 9, 12, 17
OC0B....7, 9, 17
OCF0A....19
OCF0B....19
OCIE0A....19
OCIE0B....19
OCR0A....7 e seg.
OCR0B....7 e seg.
Output Compare....7
overflow....4
Overflow....8
P
PB7....17
Phase Correct....6
Phase Correct PWM....11
prescaler....4, 7, 18
S
SREG....10
T
T0....8
TCCR0A....7
TCCR0B....7
TCNT0....7 e seg., 13, 19
TCNTn....8
toggle....12
TOIE0....19
TOP....8
TOSC1....4
TOSC2....4
TOV0....8, 19
W
waveform generator....9
WGM....9

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
14/06/20	1.0	Marcello Pinna	Mauro Laurenti	Versione Originale.