Steve Blackbird (TardoFreak)

GENERATORE DI BASE TEMPI

4 October 2010

Non è passato neanche un anno dalla mia iscrizione in questo bellissimo sito ed in questo, seppur breve periodo, frequentando il forum ho avuto modo di leggere tanti interventi e molte richieste di circuiti. Una delle richieste più comuni è quella di un **generatore di base tempi** per gli usi più disparati che vanno dal circuito di timing per lampade stroboscopiche alle frequenze campione a 1KHz passando per l' Hertz per pilotare timers ed orologi. Ho pensato quindi di realizzare un circuito che permetta di generare le frequenza più utilizzate.

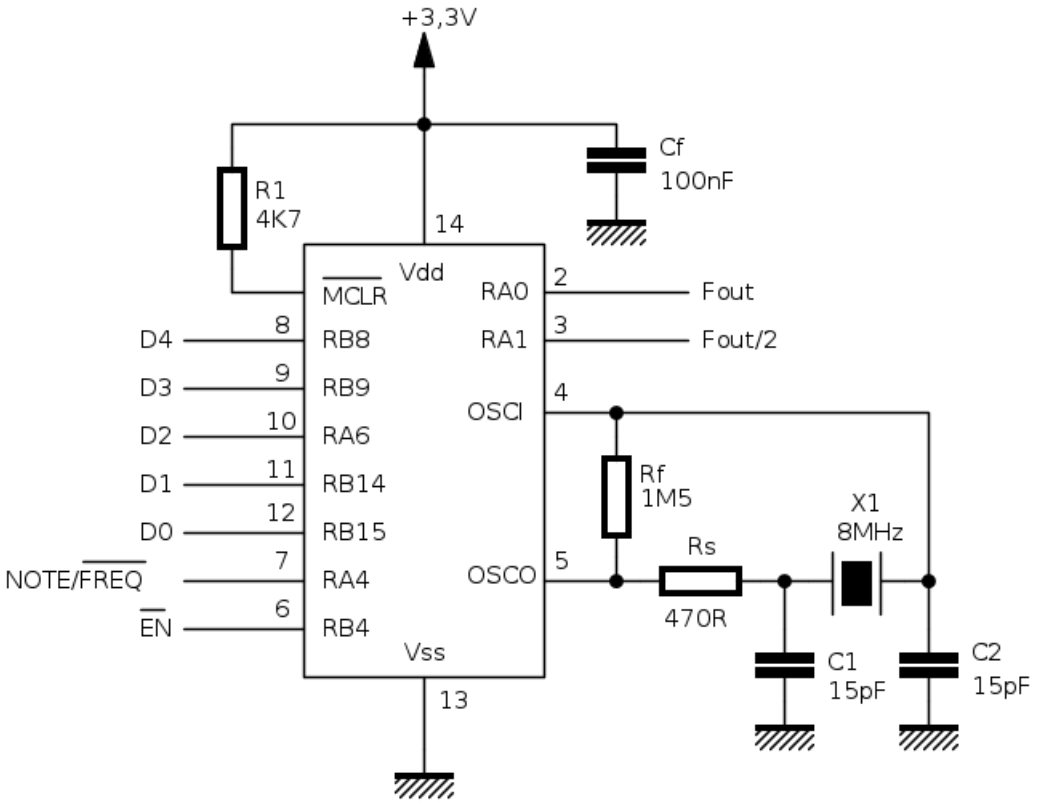
Non si tratta di un semplice oscillatore quarzato ma di un vero e proprio generatore adattabile e programmabile. Per generare una frequenza bassa e precisa ci si affida alla precisione dei **quarzi** che, almeno nelle intenzioni (oggi sono un po' fatti "con la pressa") assicurano precisione e stabilita'. Il problema sta nel fatto che, se realizzare un semplice generatore di clock con un quarzo non è un problema, la **catena di divisione** lo è.

Generare un **clock** ad 1 Hz partendo da un quarzo anche solo da 1 MHz significa utilizzare un circuito integrato per l' oscillatore e ben 6 stadi divisori per dieci. In pratica servono 7 circuiti integrati per ottenere una base tempi precisa. L' idea è quella di utilizzare un solo circuito integrato che permetta di generare una frequenza, i suoi sottomultipli di 2, e che sia programmabile tramite l' impostazione di alcuni pin e che abbia un ingresso di abilitazione che serva a bloccare la frequenza in uscita e farla ripartire correttamente.

Il circuito

In un passato neanche troppo remoto un circuito integrato del genere veniva realizzato su commissione. I costi erano a dir poco mostruosi quindi questo tipo di approccio veniva adottato da grandi industrie per grandi produzioni. Le logiche programmabili risolsero il problema abbattendo drasticamente i costi. In effetti la soluzione migliore sarebbe quella di utilizzare una logica programmabile ma per un hobbista la cosa protrebbe non essere applicabile. Infatti è più facile trovare un hobbista che utilizza microcontrollori piuttosto che uno che utilizza logiche programmabili. Per questa applicazione ho pensato di utilizzare un microcontrollore della microchip (sai che novità), in particolare un **PIC24F04KA200**.

Micro poco costoso, programmabile in C, di facile reperibilità ed in contenitore da 14 pin. Ho scelto un quarzo da 8MHz in modo da avere un clock interno a 16 MHz. (usando il PLL) ed un incremento del timer ogni 62,5ns. Vi sono 2 uscite di frequenza e, variando il quarzo ed i valori nel programma, si possono ottenere praticamente tutte le frequenze che potrebbero interessare. Con questo quarzo e questo programma così com' è le frequenza primarie sono (espresse in Hz.): 200K 100K 80K 50K 20K 10K 8K 5K 2K 1000 800 500 200 100 80 50 20 10 8 5 2 1. Oltre queste ci sono anche delle frequenze utilizzate in elettronica e sono (espresse in Hz.): 153,6K 64K 48K 44,1K 35K 22K 19K 9600 60. Tutte queste, esclusa la 64K, sono approssimate. Sono anche disponbili 24 frequenze musicali. Ma ora analizziamo il circuito.

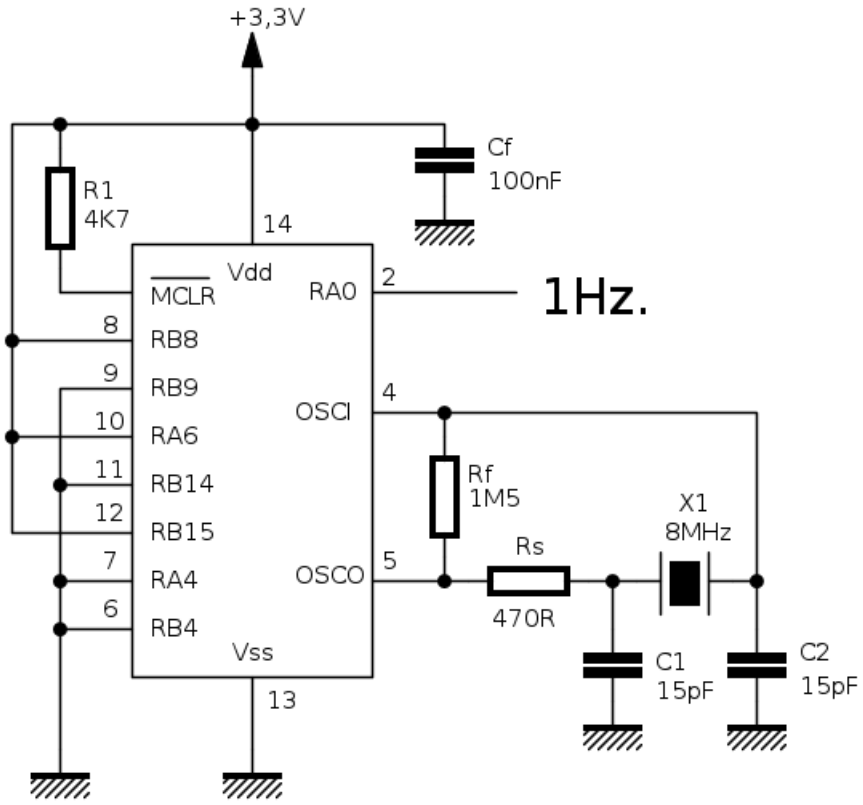


Come si può notare è molto semplice: il PIC, un quarzo, la resistenza che collega l' ingresso MCLR alla Vdd, un condensatore da 100n (ospite fisso) sull' alimentazione. Il circuito del quarzo è più complesso di quello indicato nel datasheet ma è più affidabile per una serie di motivi. La resistenza Rf serve a garantire la reazione che innesca l' oscillatore (con alcuni quarzi è necessaria) e la resistenza Rs evita di caricare il quarzo.
Sulla sinistra ci sono gli ingressi da D0 a D4 che servono a selezionare la frequenza in base alla tabella che vedete di seguito, il segnale NOTE/FREQ se a 0 seleziona le frequenze standard, se a 1 seleziona il banco di frequenza musicali. Il segnale di EN

(attivo basso) abilita o blocca le uscite di frequenze. Nel passaggio da 1 (bloccato) a 0 l' uscita di frequenza "Fout" ripartirà con un fronte di salita ed il primo semiperiodo positivo mentre la "Fout/2" ripartirà con il primo semiperiodo a 0.

D4	D3	D2	D1	D0	NOTE/FREQ = 0	NOTE/FREQ = 1	
0	0	0	0	0	200 KHz	440	La4
0	0	0	0	1	100 KHz	466,16	La#4
0	0	0	1	0	80 KHz	493,84	Si4
0	0	0	1	1	50 KHz	523,2	Do5
0	0	1	0	0	20 KHz	554,64	Do#5
0	0	1	0	1	10 KHz	587,36	Re5
0	0	1	1	0	8 KHz	622,24	Re#5
0	0	1	1	1	5 KHz	659,2	Mi5
0	1	0	0	0	2 KHz	698,4	Fa5
0	1	0	0	1	1000 Hz	740	Fa#5
0	1	0	1	0	800 Hz	784	Sol5
0	1	0	1	1	500 Hz	830,56	Sol#5
0	1	1	0	0	200 Hz	1760	La6
0	1	1	0	1	100 Hz	2864,64	La#6
0	1	1	1	0	80 Hz	1975,36	Si6
0	1	1	1	1	50 Hz	2092,8	Do7
1	0	0	0	0	20 Hz	2218,56	Do#7
1	0	0	0	1	10 Hz	2394,44	Re7
1	0	0	1	0	8 Hz	2488,96	Re#7
1	0	0	1	1	5 Hz	2636,8	Mi7
1	0	1	0	0	2 Hz	2793,6	Fa7
1	0	1	0	1	1 Hz	2960	Fa#7
1	0	1	1	0	0,1 Hz	3136	Sol7
1	0	1	1	1	153,6 KHz	3322,24	Sol#7
1	1	0	0	0	64 KHz		
1	1	0	0	1	48 KHz		
1	1	0	1	0	44100 Hz		
1	1	0	1	1	35 KHz		
1	1	1	0	0	22 KHz		
1	1	1	0	1	19 KHz		
1	1	1	1	0	9600 Hz		
1	1	1	1	1	60 Hz		

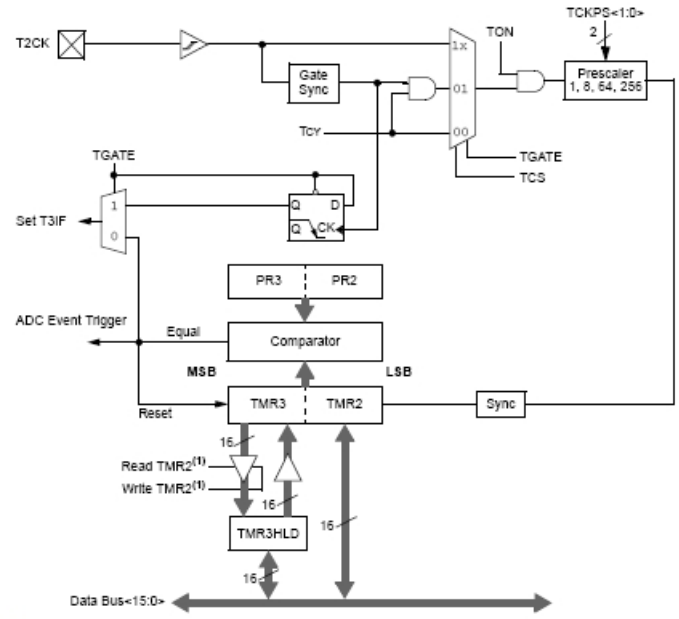
Anche le frequenze delle note musicali sono approssimate. Da notare che le prime 12 frequenze corrispondono all' ottava centrale del pianoforte mentre le 12 successive saltano son 2 ottave più su. Questo perche', usando anche l' uscita "Fout/2", si possono avere in pratica 4 ottave piene, 2 per ogni gruppo di 12 semitoni. Quindi se abbiamo bisogno di una base tempi da 1Hz per pilotare il nostro orologio digitale, o un timer, sarà sufficiente realizzare un circuito come quello indicato nello schema seguente:



Il firmware

Generare un segnale che, più o meno, sia di una data frequenza con un programma scritto in C è banale, meno banale è generare una frequenza precisa. Lavorare in assembler premette, oltre allo sfruttamento totale delle possibilità del micro, la possibilita' di calcolare e verificare con esattezza matematica i tempi di esecuzione. Con il C questo è possibile solo guardando con attenzione il lavoro "dietro le quinte" del compilatore. Non abbiamo quindi la possibilità di sapere a priori quanto tempo (o meglio quanti cicli macchina) dura una linea di programma se non andando ad analizzare il codice disassemblato. Questo vuol dire che non possiamo fare affidamento sulle varie funzioni di ritardo solitamente presenti nelle librerie.
Un sistema per generare una frequenza precisa è quello di utilizzare un timer munito di registro di comparazione che, al raggiungimento del valore impostato, lo faccia

ripartire e commuti un' uscita. Inoltre il timer dovrebbe avere una risoluzione tale da permettere di determinare con esattezza il periodo del segnale di uscita. Inizialmente avevo pensato ad utilizzare un micro a 8 bit ma, purtroppo, i micro a basso pinout non hanno un timer da almeno 16 bit con comparazione e prescaler. Inoltre l' uso del prescaler limita la risoluzione. I micro a 16 bit, anche quelli a basso pinout, hanno un timer a 32 bit con comparatore e, se bastevole non fosse, anche un prescaler. E' per questo motivo che la scelta è caduta su un PIC24. Oddio, sarebbe stato meglio utilizzare un dsPIC a 5V ma ... ne ero sprovvisto e la consegna era a 10 giorni. Il firmware si può comunque trasportare su un micro a 5V se necessario.
Questo è lo schema interno del timer2/3 utilizzato come timer a 32 bit.



TIMER23.png

Come possiamo notare il timer può ricevere il segnale di clock attraverso un ingresso esterno con possibilità di sincronizzazione oppure dal clock macchina indicato con Tcy. Noi usereme questa opzione. Non abbiamo bisogno del prescaler perché il timer pilotato da una frequenza di 16MHz ha la possibilita' di genereare ritardi fino a più di 2900 s. con una risoluzione di 62,5 ns. quindi lo imposteremo a 1. L' uscita del comparatore può inviare il segnale di avvenuta comparazione per l' ADC e/o generare una interrupt. Noi lo useremo per generare l' interrupt.

Il programma

Il firmware è scritto per il compilatore MPLAB C30 nella versione Lite, fornito gratuitamente da Microchip. La prima parte del programma vede la solita dichiarazione dei fuses. La documentazione sui fuses la si trova al fondo del file

incluso <p24f04ka200.h>. Di seguito ci sono le defines per le porte di I/O. Da notare l' assenza della dichiarazione delle uscite "Fout" e "Fout/2". Questo perché devono essere contigue in modo da generare le due frequenze semplicemente incrementando la variabile dato_out, come si vede subito dopo nella funzione di servizio dell' interrupt del timer. Ci sono poi le due tabelle in ROM contenenti i valori da caricare nel registro di comparazione per ottenere le frequenze desiderate.

Per quanto riguarda la routine di interrupt c' è da dire che il tempo impiegato per mandare fuori le uscite è irrilevante per quanto riguarda la precisione delle frequenza. Quando arriva l' interrupt il timer non è fermo nè da ricaricare, lui funziona per i fatti suoi generando sempre lo stesso ritardo quindi non abbiamo bisogno di nessuna compensazione del tempo utilizzato per gestire l' uscita. Nel caso di un timer ad incremento dovremmo invece tenerne conto, infatti il caricamento del timer porta via del tempo che dovrebbe essere tenuto in considerazione. Troviamo poi la funzione di settaggio del comparatore che, quando raggiunto dal timer, lo resetta e genera l' interrupt.
Il main e' classico. Dapprima le inizializzazioni del timer, I/O e variabili. Il ciclo infinito di funzionamento controlla sempre se c'è una variazione sugli ingressi ed imposta la nuova frequenza e, se il generatore non è bloccato, fa ripartire il timer.

Una piccola nota su come calcolare il valore da impostare nel timer per ottenere un frequenza "f". Il clock interno, quello che fa funzionare la MPU e le periferiche è, in questo caso, 16Mhz. Questo perché viene ottenuto dalla frequenza del quarzo elevata dal PLL e poi divisa per 2. L' interrupt del timer deve avvenire per due volte per generare un solo ciclo della frequenza di uscita. La formula per calcolare il valore "n" da impostare nel timer per ottenere la frequenza "f" è:

$$n = \frac{T_{CY}/2}{f} = \frac{8000000}{f}$$

Questa formula può essere utile per implementare funzioni per generare la musica tramite un microcontrollore o comunque per generare le frequenze che più vi aggradano.

Download

TF_GenBT.c

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Tardofreak:generatore-di-base-tempi"