LE PERIFERICHE DEI PIC

Per sfruttare appieno le potenzialità di un microcontroller, è necessario conoscere innanzi tutto la sua logica di funzionamento, ma è anche fondamentale capire come la cpu dialoga con il mondo esterno. Progetto vi spiega come interfacciare un Pic con i componenti di un circuito

Paolo Sbrana - 1ª parte

Da quando abbiamo pubblicato il corso sulla programmazione dei microcontroller, molti lettori hanno deciso di passare all'elettronica digitale "computerizzata", piuttosto che rimanere su quella cosiddetta "cablata", e il motivo è banale: mentre con la prima si può modificare il funzionamento di una macchina solamente con la variazione di instruzioni software, con la seconda si deve necessariamente riprogettare l'hardware dedicato, con conseguente riprogettazione del relativo circuito stampato.

Se poi si hanno dei problemi di malfunzionamento, si deve ripartire da capo.

La migliore delle scelte, quindi, è passare ad una logica "programmabile", ovvero ad un hardware il meno specifico possibile abbinato ad un software totalmente dedicato.

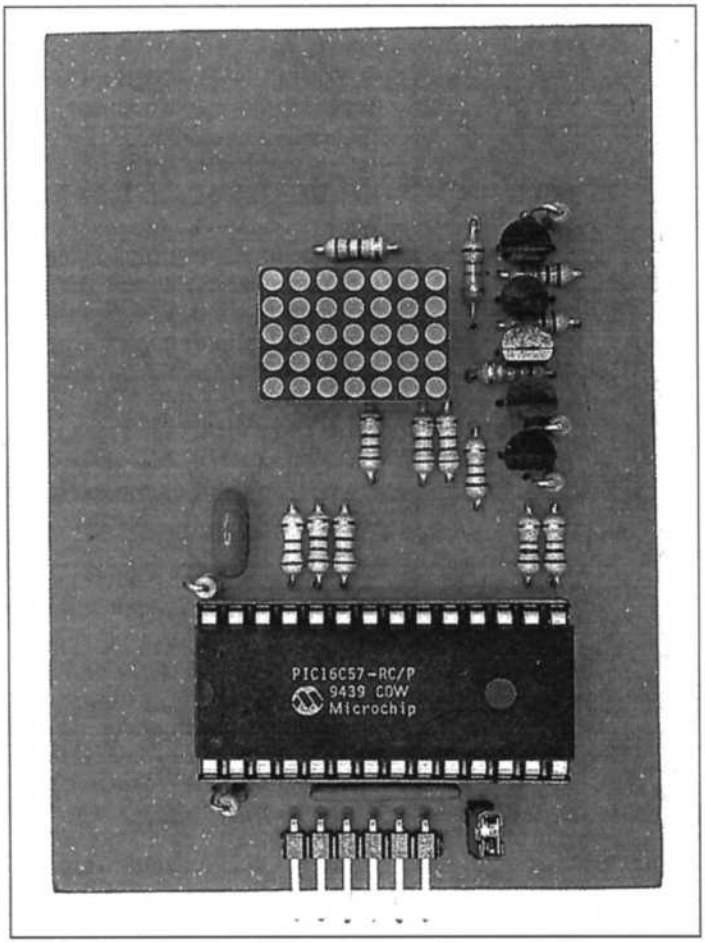
Chi ci ha seguito per tutto il corso presentato nello scorso anno comincia adesso a realizzare autonomamente i primi circuiti funzionanti, ovviamente non complicatissimi, ma ideali per fare esperienza di microprogrammazione.

In una delle tante lettere giunte in redazione per esempio, un lettore ci comunica che è riuscito a realizzare un impianto di allarme con le caratteristiche degne delle migliori centrali commerciali.

Per sfruttare pienamente le capacità di un microcontroller, però, è indispensabile conoscere a fondo anche le "periferiche" che questo ha a bordo.

Le periferiche dei PIC

Ma che cosa sono queste periferiche? In linea di massima sono quella parte interna al chip stesso che non si trovano nei microprocessori.



Vediamo un esempio: se prendiamo un processore puro, non avrà watchdog, convertitori A/D, comparatori, moduli Capture, moduli PWM, USART, UART, ecc.

Nei microcontroller invece, per velocizzarne il funzionamento e per far risparmiare spazio e denaro, vengono di solito inserito uno o più circuiti appena visti, e vengono detti "periferiche" del microcontroller.

In particolare, nei PIC abbiamo a disposizione una vasta gamma di periferiche, alcune delle quali sono in comune a tutti i modelli, altre specifiche per famiglia.

In generale, possiamo dire che le famiglie 16C5X, 16C55X e 12C50X, hanno soltanto la periferica del Timer 0, la famiglia 12C67X in più hanno anche un convertitore A/D ad 8 bit.

La famiglia 16C62X ha in più due comparatori analogici ed il brown-out-detection.

La famiglia 16C6X possiede una seriale IIC-bus o SPI selezionabile, generatore PWM, brown-out-detection, 3 timer e qualcuno anche una USART completa. La famiglia 16C7X invece è simile alla 16C6X con in più un convertitore A/D ad 8 bit ed un altro generatore PWM, oltre ad una sezione detta Capture e Compare. La particolarità di queste periferiche è che sono sempre le stesrse, ovvero in un PIC16C65, troveremo la stessa USART che ha il PIC16C74 e nel PIC16C71, troveremo lo stesso convertitore che rileviamo nel PIC16C73.

Da ciò, si capisce che sarà sufficiente analizzare attentamente la singola periferica per sfruttarla bene in qualsiasi modello di microcontroller la troviamo.

Poiché la maggior parte delle periferiche viste le possiamo trovare raccolte nel PIC16C74, durante il corso faremo sempre riferimento a questo controller, eccetto che per la periferica relativa ai comparatori, che analizzeremo in un PIC16C62X.

Il Timer0

La periferica che in assoluto è comune a tutti è il Timer0, oppure come lo avevamo chiamato nel PIC16C5X il RTCC (Real Time Clock Counter). In Figura 1 possiamo vedere il diagramma a blocchi del Timer0.

Per prima cosa notiamo che è interfacciato con il bus dei normali registri, quindi è leggibile e riscrivibile come un comune registro (quindi ad 8 bit). In più, dove il microcontroller lo consente, è presente anche una segnalazione di overflow tramite un interrupt dedicato.

Ma capiamo come funziona questo timer: il registro TMR0 viene incrementato di una unità a seconda di eventi che tra poco vedremo. Quando questo registro raggiunge lo 0 (ovvero passa da 255 a 255 + 1 = 0), si ha un interrupt dedicato (oppure si va a testare in continuazione il suo valore).

Quali sono gli eventi che fanno incrementare il registro TMR0?.

È possibile selezionarne uno a scelta tra: fronte (o di salita o di discesa) sul pin TOCK del PIC o ad ogni ciclo di clock (frequenza del quarzo divisa per 4).

Inoltre, è possibile inserire un prescaler tra questi due eventi ed il conteggio vero e proprio, in modo tale da dividere il numero degli eventi trascorsi.

Il Timer0 ha una particolarità: quando viene scritto (quindi anche in fase di incremento) l'incremento avviene dopo

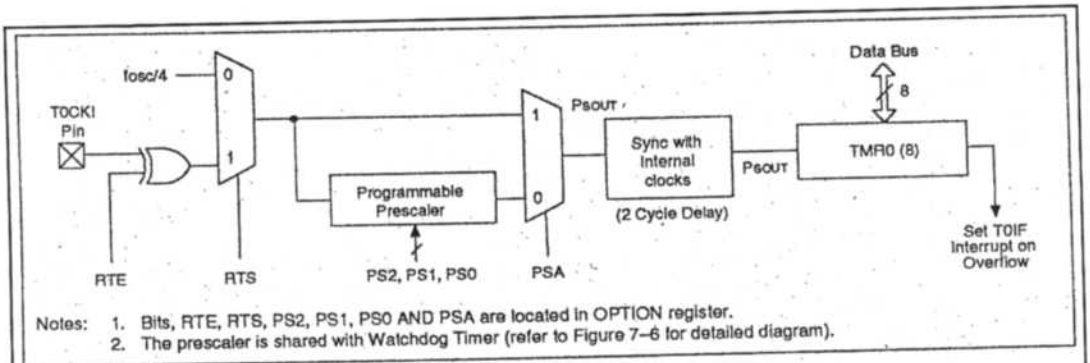


Figura 1. Diagramma a blocchi del Timer 0

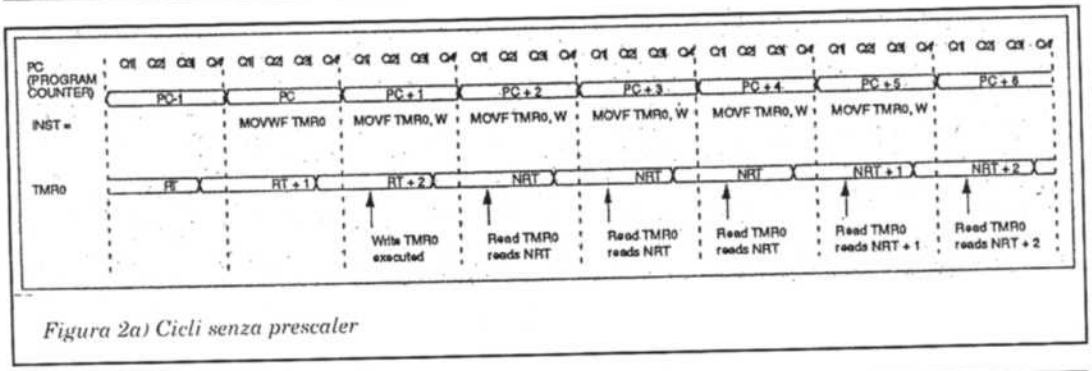


Figura 2a) Cicli senza prescaler

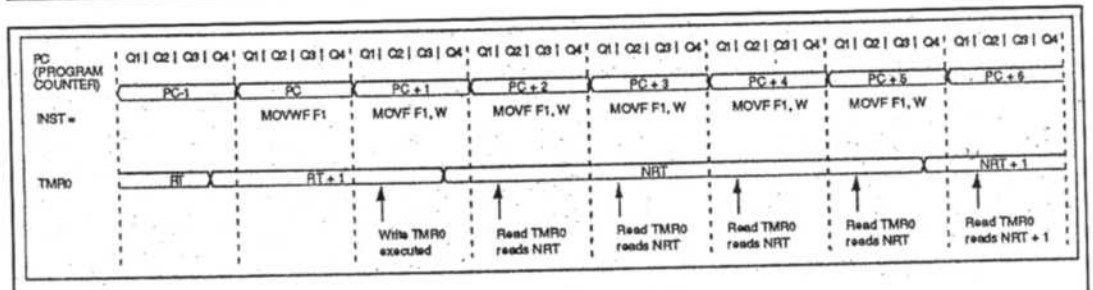


Figura 2b) Cicli con prescaler 1:2

Tabella 1a. Elenco dei registri correlati al TMR0

Register Name	Function	Address	Power-on Reset Value
TMR0	Timer/counter register	01h	xxxx xxxx
OPTION	Configuration and prescaler assignment bits for TMR0	81h	1111 1111
INTCON	TMR0 overflow interrupt flag and mask bits	0Bh	0000 000x

Tabella 1b. Elenco dei bit da settare per il TMR0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01	TMR0	TIMERO							
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81	OPTION	RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0
95	TRISA	–	–	TRISA 5	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0

Legend – = Unimplemented locations, Read as '0'
Shaded boxes are not used by TMR0 module.

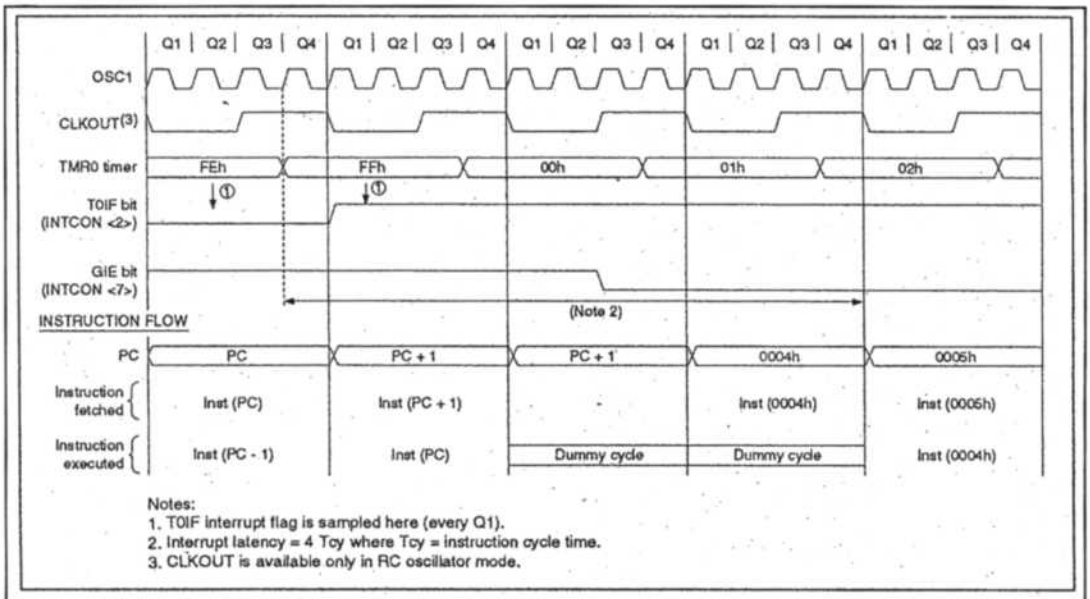


Figura 3. Diagramma dei tempi dopo interrupt del Timer0, la più comune e utilizzata periferica

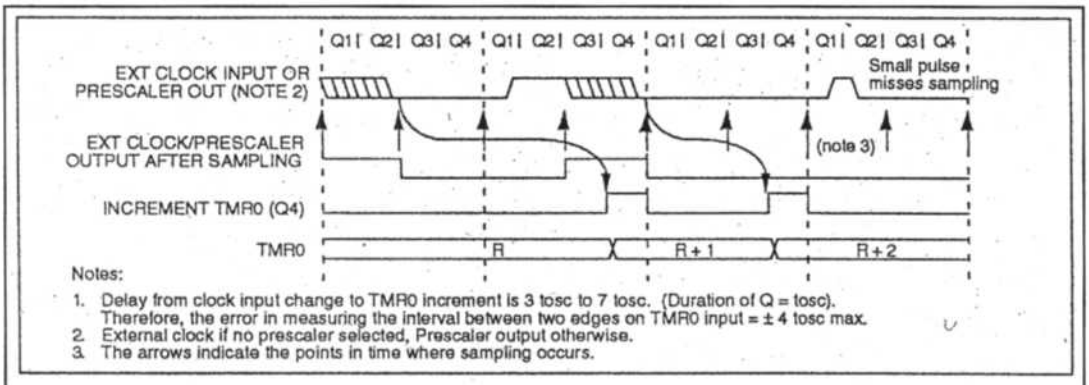


Figura 4. Andamento temporale della periferica Timer0 rispetto al fronte esterno

due cicli di clock, e di questo si deve tener conto durante il calcolo di eventuali tempistiche. In Figura 2a e 2b si vede la differenza di funzionamento tra l'incremento rispettivamente senza e con prescaler.

Abbiamo detto che quando il registro TMR0 passa da 0xff a 0x00, viene generato un interrupt (se abilitato nel registro INTCON il bit TOIE) e viene settato il bit TOIF del registro INTCON. Tale flag rimane settato fino a quando non verrà cancellato con una istruzione software.

Questo ci assicura che non perderemo mai la segnalazione così memorizzata. Allo stesso tempo, non possiamo riabilitare tale interrupt fino a quando non resettiamo il bit TOIF.

Le tempistiche di reazione all'interrupt generato dal Timer0 sono visibili in Figura 3.

L'istruzione che abilita allora l'interrupt del Timer0 dovrà essere così scritta:

```
bcf TOIF  ; reset flag
bsf TOIE  ; abilita interrupt Timer0
```

ovvero, prima si azzera il flag relativo all'interrupt, poi si abilita l'interrupt. In caso contrario, l'interrupt da TMR0 potrebbe non arrivare mai perché il flag TOIF era già settato in precedenza.

Per vedere, in fase di salto alla routine di interrupt, se il Timer0 è andato in overflow, basterà testare il bit TOIF e, se settato, riportarlo a zero.

Abbiamo detto che l'incremento del registro TMR0 può avvenire a causa di uno di due eventi selezionabili da software.

Il bit RTS dell'OPTION register indica se l'incremento deve avvenire a ogni ciclo di clock oppure alla presenza di un fronte sul pin TOCK.

Nel secondo caso è possibile decidere, impostando correttamente il bit RTE sempre dell'OPTION register, se l'incremento deve avere luogo sul fronte di salita o di discesa del segnale.

I due tipi di funzionamento sono legati alle applicazioni che vogliamo supportare con il microcontroller.

Se, per esempio, desideriamo contare degli eventi (passaggio di oggetti, pressioni di un pulsante, segnali di allarme, ecc.) il modulo del Timer0 configurato come contatore esterno è l'ideale, perché pensa a tutto lui: per sapere il numero dei conteggi, basterà andarsi a leggere, quando vogliamo, il contenuto del registro TMR0.

Se, invece, desideriamo avere una base dei tempi per cui ogni N millisecondi dobbiamo fare una certa cosa, allora il modulo Timer0 va selezionato per l'incremento automatico legato al ciclo di clock del microcontroller.

Nel caso della lettura del fronte però, ci sono delle limitazioni legate alla sincronizzazione del segnale esterno con la fase del clock del controller: riferendoci alla Figura 4, la sincronizzazione viene fatta dopo il modulo del prescaler.

L'uscita di tale modulo viene campionata due volte ad ogni ciclo di istruzione (4 di clock) per la determinazione del fronte di salita o di discesa.

Per questo motivo, è necessario che il tempo di uscita del modulo prescaler rimanga alto per almeno 2*tosc basso per almeno altri 2*tosc dove tosc=periodo di oscillazione.

Questo ci fa capire che abbiamo una limitazione sulla frequenza del segnale

PROGRAMMA 1

```
bcf     STATUS,RP0      ;Banco 0
clrf    TMR0            ;Clear TMR0
bsf     STATUS,RP0      ;Banco 1
clrwdt                  ;Azzera watchdog e prescaler
movlw   b'xxxx1xxx'     ;Seleziona nuovo valore
                        prescaler
movwf   OPTION          ;Valore
bcf     STATUS,RP0      ;Banco 0

Per passare invece dall'Watchdog al Timer0:

clrwdt                  ;Clear watchdog e prescaler
bsf     STATUS,RP0      ;
movlw   b'xxxx0xxx'     ;Seleziona TMR0, valore presc.
                        e source
movwf   OPTION          ;
bcf     STATUS,RP0      ;Banco 0
```

PROGRAMMA 2

```
movf    TMR1H,W         ;Leggi byte alto
movwf   REG1            ;
movf    TMR1L,W         ;Leggi byte basso
movwf   REG2            ;
movf    TMR1H,W         ;Leggi byte alto
subwf   REG1,W          ;Sottrai lettura 1 con lettura 2
btfsc   STATUS,Z        ;Risultato = 0?
goto    CONTINUA        ;SI
movf    TMR1H,W         ;NO —>Leggi nuovamente byte
                        alto
movwf   REG1            ;
movf    TMR1L,W         ;Leggi byte alto
movwf   REG2            ;
;Riabilita interrupt se richiesto
CONTINUA
```

da controllare legata alla frequenza del clock del PIC.

Parliamo allora del prescaler disponibile sul Timer0.

Per prima cosa, non è detto che sia necessariamente dedicato al Timer0, ma potrebbe anche essere destinato alla periferica Watchdog, poiché essendo in comune tra le due, dovremo scegliere noi la sua assegnazione settando il bit PSA dell'OPTION register.

In Figura 5, troviamo il diagramma a blocchi del modulo prescaler-TMR0-WDT. Anche il prescaler è comunque un registro ad otto bit, non leggibile direttamente ma solo con una particolare tecnica che vedremo più avanti.

Con i bit PS2, PS1, PS0 dell'OPTION