

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 istruzioni 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 micro-

controller, 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 inseriti 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.

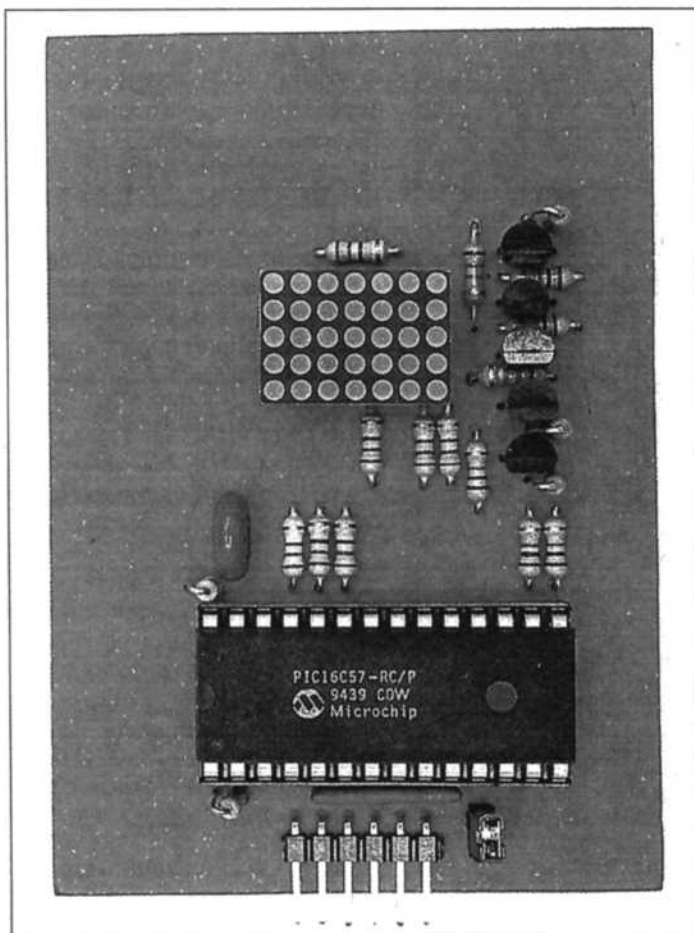
In generale, possiamo dire che le famiglie 16C5X, 16C55X e 12C50X, hanno soltanto la periferica del Timer 0, la famiglia 12C67X in più hanno anche un convertitore A/D ad 8 bit.

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 stesse, 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 selezionare una 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

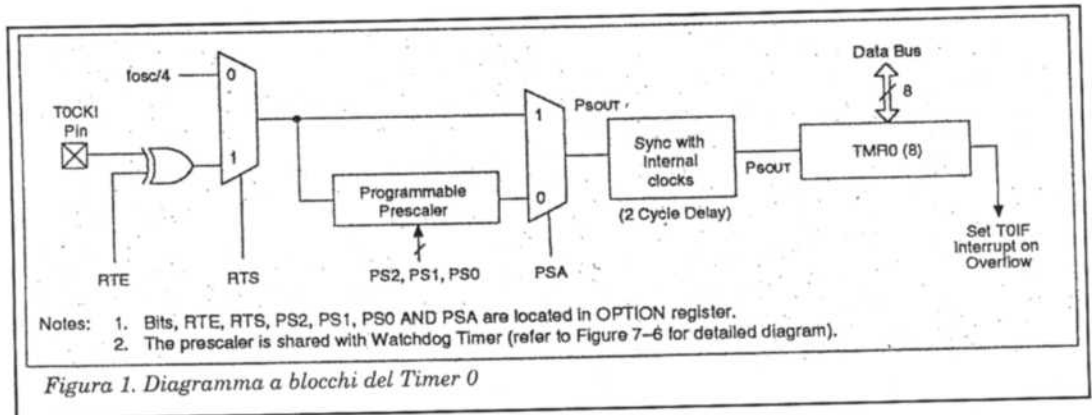


Figura 1. Diagramma a blocchi del Timer 0

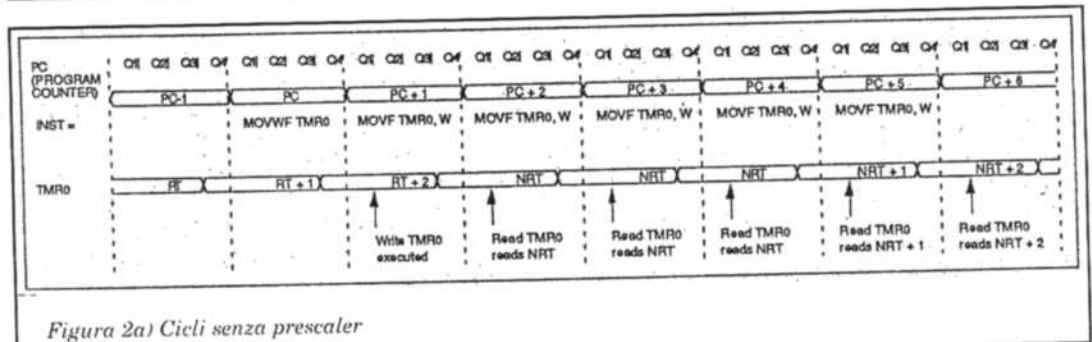


Figura 2a) Cicli senza prescaler

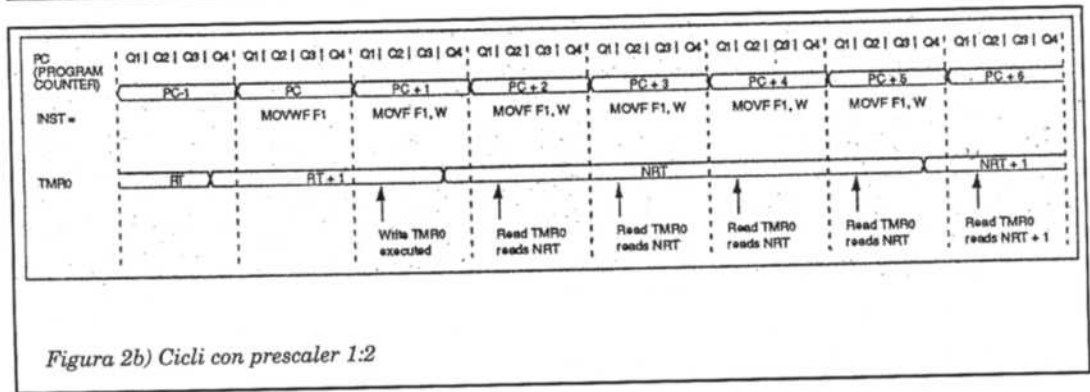


Figura 2b) Cicli con prescaler 1:2

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

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	TOIE	INTE	RBIE	TOIF	INTF	RBF
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.

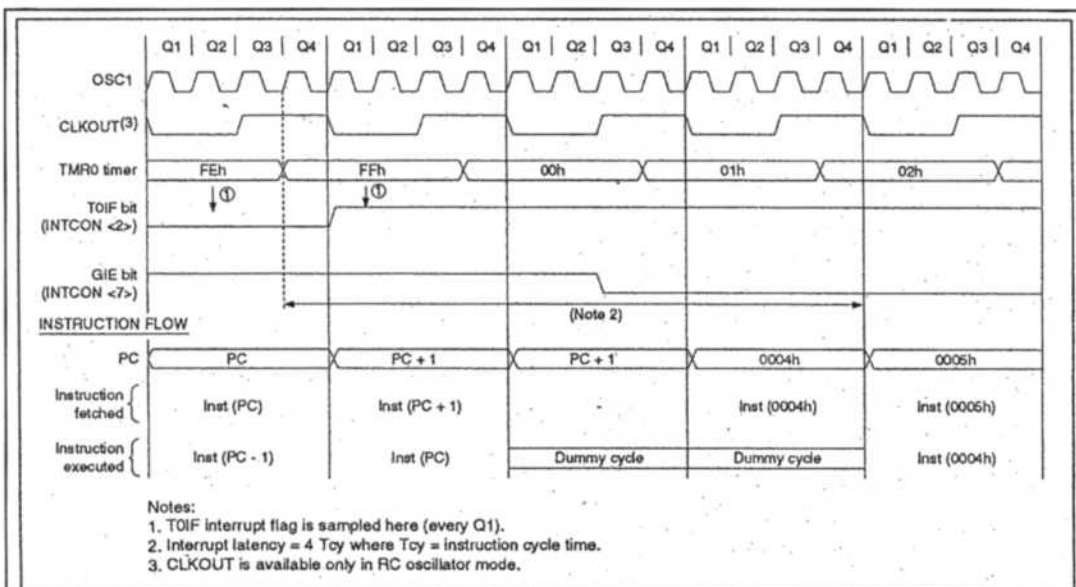


Figura 3. Diagramma dei tempi dopo interrupt del Timer0, la più comune e utilizzata periferica

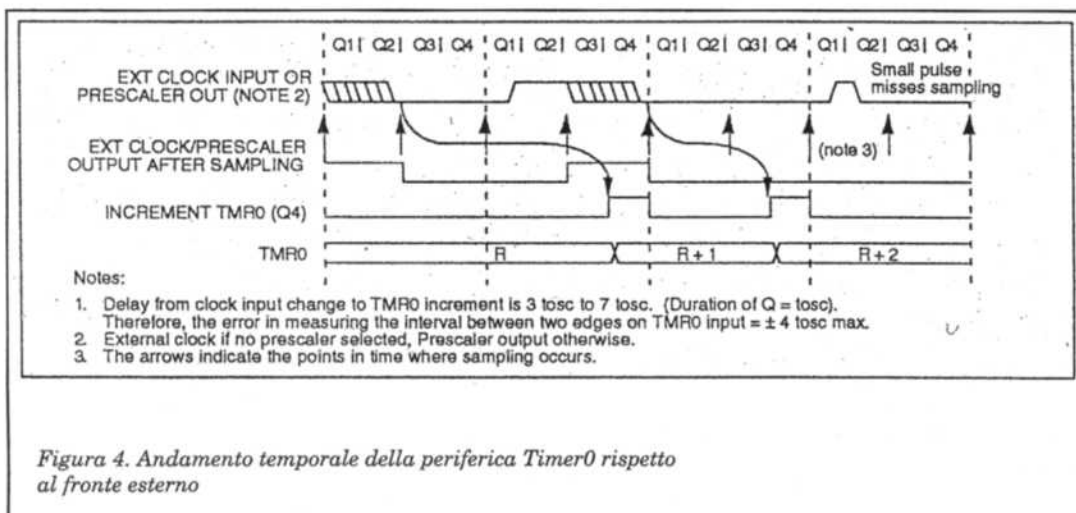


Figura 4. Andamento temporale della periferica Timer0 rispetto al fronte esterno

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 TOIF) 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 azzerava 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 milisecondi 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

MICROCONTROLLER

PROGRAMMA 1

```
bcf STATUS,RP0 ;Banco 0
clrf TMR0 ;Clear TMR0
bsf STATUS,RP0 ;Banco 1
clrwdt ;Azzera watchdog e prescaler
movlw b'xxx1xxx' ;Selezione 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'xxx0xxx' ;Selezione 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
btsc 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

NORTH STAR TECHNOLOGY

RICERCHE ELETTRONICHE - SVILUPPO NUOVI PRODOTTI

STEPPING MOTORS

Cercasi rappresentanti per zone libere

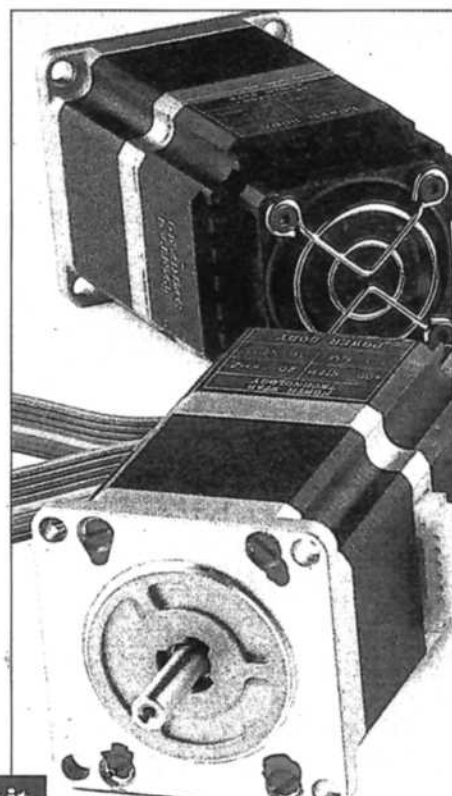
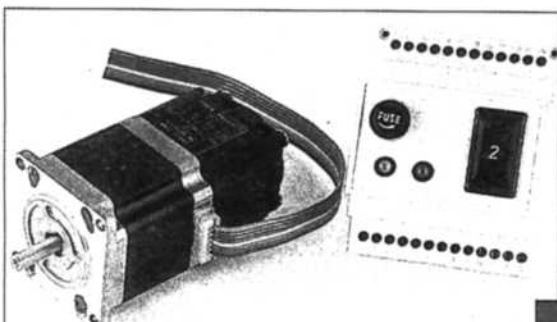
La nuova linea di motori "Power Body" è la soluzione ottimale per ogni tipo di applicazione ove sia richiesto un motore passo passo. La sua praticità d'uso è data dal suo azionamento (driver) incluso nel corpo motore, grazie ad esso si possono ottenere velocità mai raggiunte prima (20 kHz di clock, dipende dal modello), e con caratteristiche di coppia eccezionali. Il corpo che contiene il driver è costruito in alluminio lavorato dal pieno per ottenere maggiori prestazioni nella dissipazione termica, il motore è di qualità garantita con caratteristiche tecniche costruttive che ne fanno un prodotto ad uso professionale.

L'azionamento può funzionare con segnali di ingresso tipo standard TTL e tutti i segnali sono fotocoppiati per eliminare eventuali disturbi, le funzioni sono le seguenti: clock, direzione, attivazione, limitazione della corrente (half/full per alcuni modelli).

I vantaggi offerti dai motori passo passo "Power Body" sono enormi, basti pensare che i collegamenti fra azionamento e motore sono del tutto eliminati in quanto sono interni, le temperature nei vani di controllo non sono più critiche perché non è più presente la parte di azionamento, la scomodità e la perdita di tempo nel costruire o acquistare il driver che comunque è sempre un prodotto non studiato per il tipo di motore che intendete utilizzare.

Questa linea di motori è composta da vari modelli che si differenziano per coppia e velocità.

North Star Technology
Via Venezia, 13
32040 Domegge di Cadore (BL)
Tel. 0435/520177
Fax 0435/520265



register, è possibile impostare il valore della divisione, che nel caso dell'abbinamento al Timer0 è così elencata:

PS2	PS1	PS0	DIVISIONE
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Una cosa molto importante da sapere è che qualsiasi istruzione che si riferisce al registro TMR0 (registro di indirizzo 0x1) come ad esempio una bsf 1,x oppure movwf 1, ecc. azzerà il prescaler.

Per questo motivo, si deve prestare attenzione quando, nel corso del programma, si vuole assegnare il prescaler

Tabella 2. Selezione condensatori per l'oscillatore TMR1

Osc Type	Freq	C1	C2
LP	32 kHz§	15 pF	15 pF
	100 kHz	15 pF	15 pF
	200 kHz	0 - 15 pF	0 - 15 pF

Higher capacitance increases the stability of oscillator but also increases the start-up time. These values are for design guidance only.

§ For $V_{DD} > 4.5 V$, $C1 = C2 \sim 30 pf$ is recommended

da una all'altra periferica e viceversa.

Per passare il prescaler dal Timer0 al Watchdog, si devono scrivere le istruzioni del Programma 1.

Abbiamo così concluso la descrizione della periferica Timer0. Per utilizzarla, basterà che vengano settati come descritto precedentemente i flag relativi nell'OPTION register e nell'INTCON register e che troviamo riassunti in Tabella 1-a e 1-b.

Il Timer1

La seconda periferica che troviamo nella famiglia 16C7X è il cosiddetto Timer1. Dal nome si capisce che anch'esso è un timer come il precedente e che quindi servirà per "contare" qualche cosa. Tale modulo, diversamente dal precedente, è a 16 bit come si vede nella Figura 6, e quindi formato dall'unione di due singoli registri ad 8 bit: il TMR1H ed il TMR1L.

Entrambi i registri sono leggibili e riscrivibili in qualsiasi momento. Questa volta l'interrupt verrà generato al raggiungimento del valore 0x0000 incrementando di uno il valore 0xffff, sempre che il bit TMR1E sia stato abilitato. Il funzionamento dei bit TMR1E e TMR1F è identico a quello visto per i corrispondenti TOIE e TOIF. Questa volta, però, il Timer1 ha un registro tutto suo, visibile in Figura 7.

Il bit TMR1ON abilita o meno il Timer1. Il bit TMR1CS seleziona l'incremento sul pin esterno TCK1 oppure sincronizzato col clock del controller (esattamente come nel caso del Timer0). Il bit T1INSYNC dice se sincronizzare il segnale esterno con il clock del controller oppure no (diversamente dal Timer0 che non lo prevedeva).

Il bit T1OSCEN abilita l'oscillatore del Timer1.

I bit successivi consentono di impostare il valore del prescaler per questo timer, ma questa volta le divisioni possibili sono in numero minore:

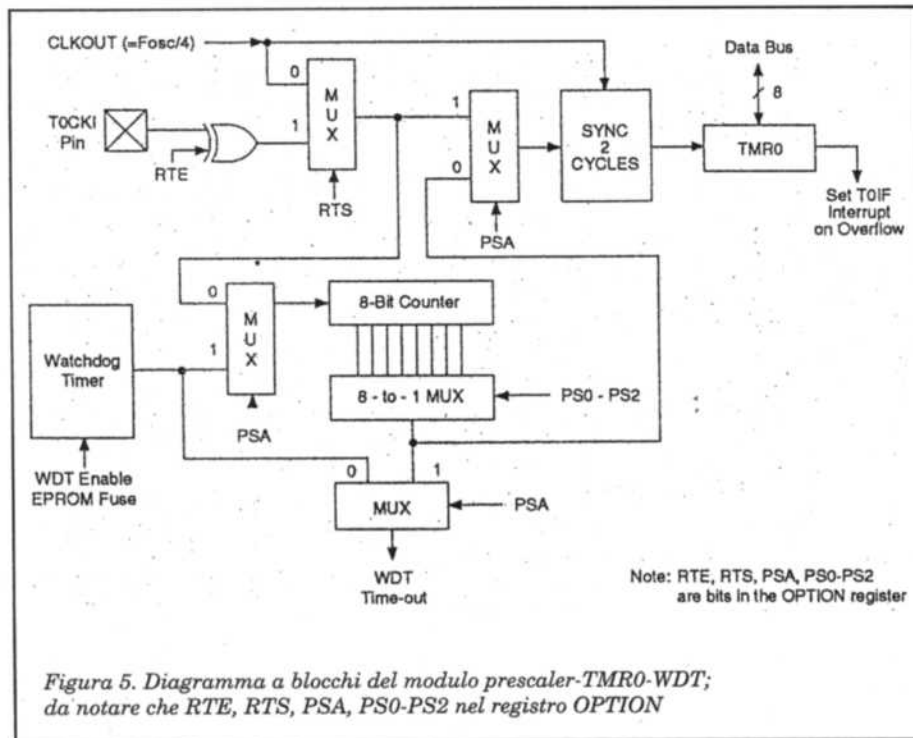


Figura 5. Diagramma a blocchi del modulo prescaler-TMR0-WDT; da notare che RTE, RTS, PSA, PS0-PS2 nel registro OPTION

Tabella 3. Registri associati al TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
0C	PIR1	PSPIF	-	-	-	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	-	-	-	SSPIE	CCP1IE	TMR2IE	TMR1IE
0E	TMR1L	Timer1 Least Significant Byte							
0F	TMR1H	Timer1 Most Significant Byte							
10	T1CON	-	-	T1CKPS1	T1CKPS0	T1OSCEN	T1INSYNC	TMR1CS	TMR1ON

Legend - = Unimplemented locations, Read as '0'

Note: Shaded boxes are not used by Timer1 module.

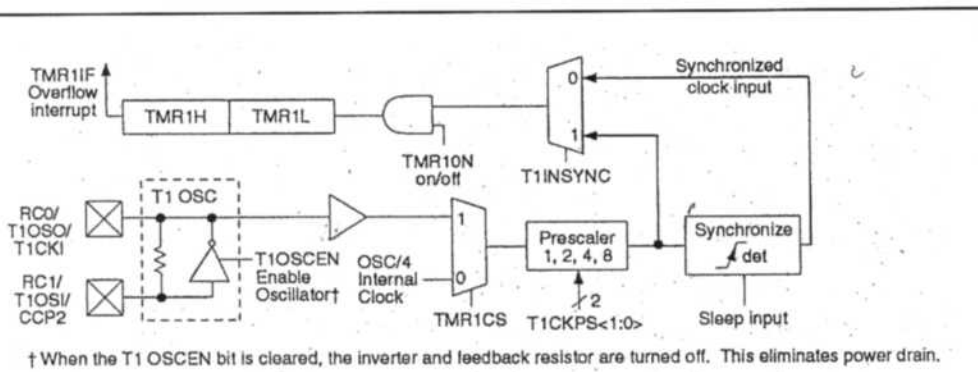


Figura 6. Diagramma a blocchi del modulo Timer1: uno dei più usati nella gestione delle interfacce

circuito ed uno a 32.768 Hz per il mantenimento di un orologio. Il PIC resta in sleep fino all'interrupt provocato dal Timer1 che lavora con una frequenza più bassa e che quindi consuma circa 100 volte di meno rispetto all'oscillatore a 4 MHz, favorendo la durata delle batterie.

In questo modo si avrebbe un risveglio ogni N millisecondi.

Nella Tabella 2 vediamo le capacità da inserire nel circuito per i vari oscillatori impiegati.

Poiché il Timer1 può lavorare quindi in asincrono con il clock del microcontroller, per leggerne il contenuto si dovranno usare delle precauzioni.

Nel Programma 2 riportiamo la sequenza corretta delle istruzioni necessarie.

In Tabella 3 troviamo i registri associati al Timer1. Nella prossima puntata analizzeremo ulteriori esempi di gestione I/O.

continua

T1CKPS1 T1CKPS0 DIVISIONE

1	1	1:8
1	0	1:4
0	1	1:2
0	0	1:1

Il prescaler, per questo timer, è sempre inserito (si può disabilitare impostando la divisione per 1:1).

Uno dei vantaggi del Timer1 è quello di poter lavorare anche separatamente dal clock del microcontroller stesso: ad esempio è possibile avere un quarzo a 4 MHz per il normale funzionamento del

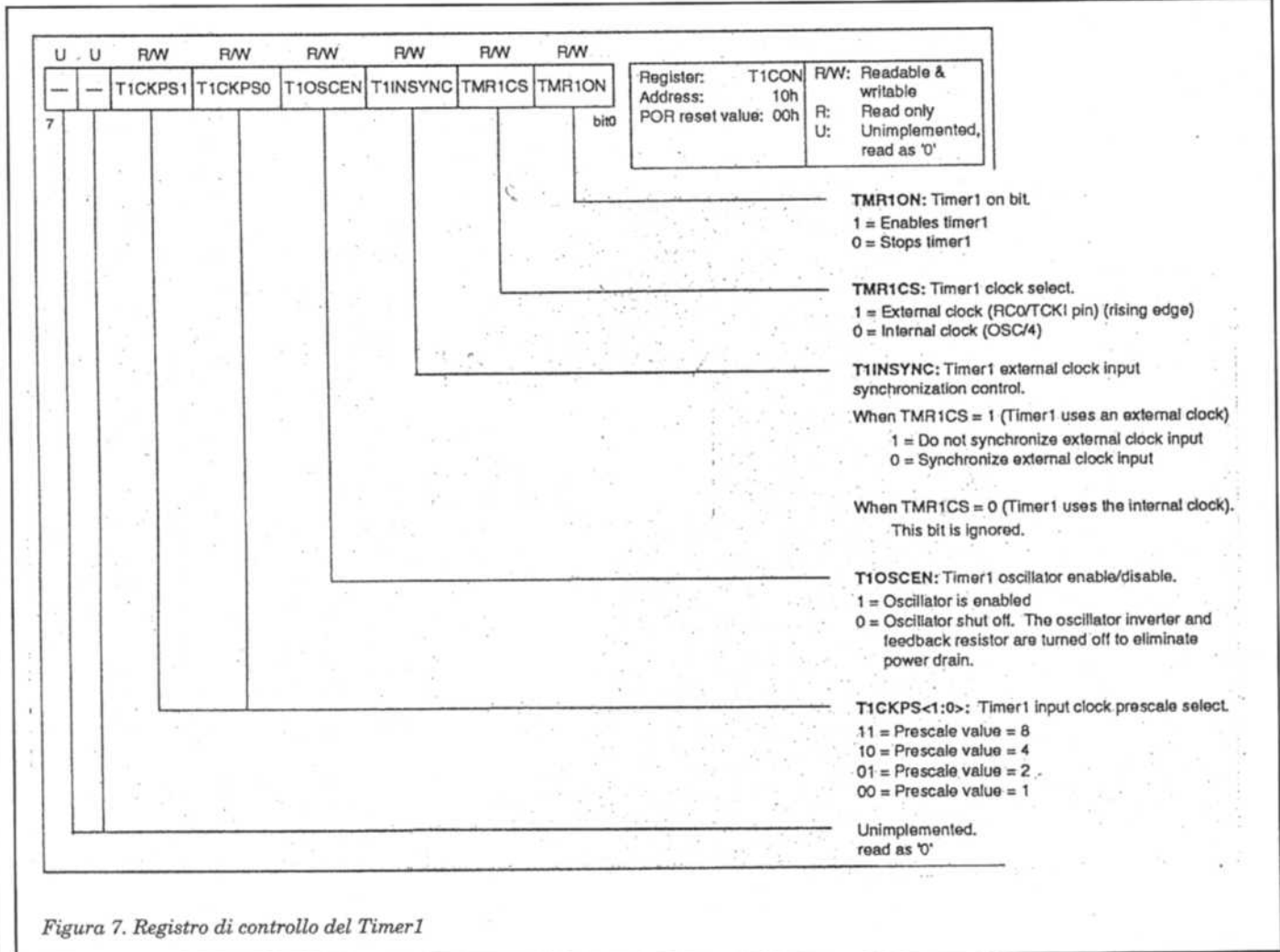


Figura 7. Registro di controllo del Timer1

LE PERIFERICHE DEI PIC: FACCIAMONE CONOSCENZA

Nel tempo siamo entrati nel dettaglio del funzionamento dei PIC, sino a presentare un intero corso di programmazione. Non basta però conoscere questi importanti processor, ma è necessario saperli connettere con il mondo esterno: ecco quindi le istruzioni su come fare!

Paolo Sbrana - 2ª parte

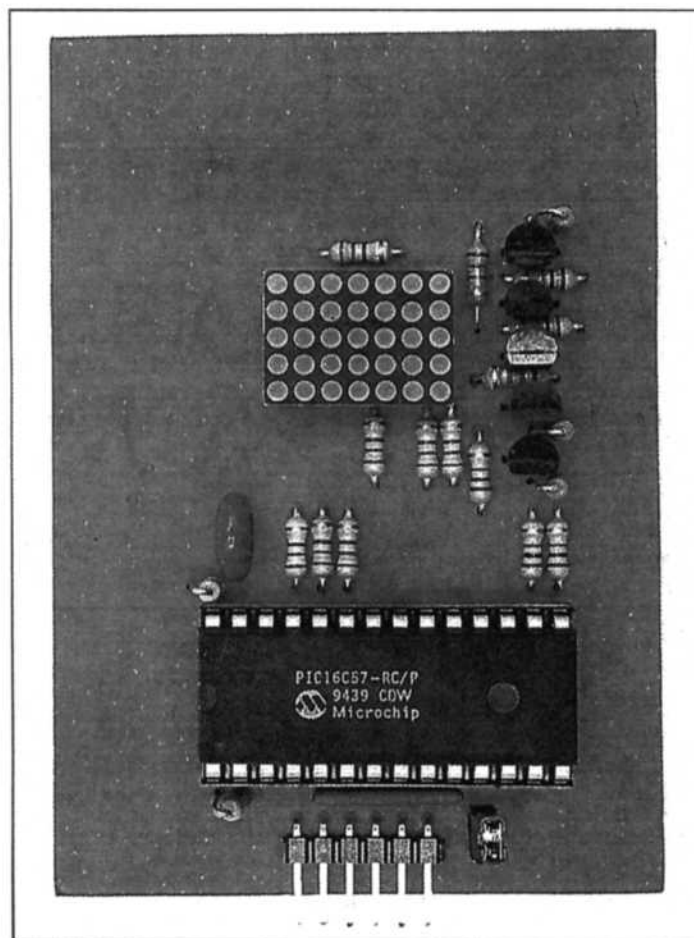
Come abbiamo visto lo scorso mese, a seconda del tipo di microcontroller impiegato, otteniamo delle funzioni aggiuntive che non richiedono ulteriore programmazione software per la loro esecuzione, come per esempio l'arrivo di un interrupt allo scadere di un tempo prefissato. Tutte queste "feature" vengono chiamate in gergo periferiche, poiché risiedono fisicamente all'interno della periferia della CPU stessa (ovviamente a livello di silicio!).

Assumiamo allora che una periferica, come già abbiamo accennato, sia una vera e propria struttura hardware che dialoga costantemente con il chip mentre questo sta lavorando, ma che non ne rallenta le normali azioni (non avrebbe senso).

Per fare un esempio semplice, supponiamo di avere una periferica che dialoga con lo standard della seriale RS232 (in realtà il PIC ne è provvisto e prossimamente la vedremo).

Ciò significa che non ci dovremo più preoccupare di scrivere del software che testi costantemente l'ingresso per vedere se giungono

dati, ma ci sarà un hardware dedicato che si incaricherà di farlo e di avvisarci tramite interrupt quando il byte sarà giunto completamente.



Capite quindi che in questo modo snelliamo il programma di gestione del micro e non avremo più la preoccupazione di sincronizzarci con il bit di start, dato che un circuito interno al chip lo farà per noi.

Passiamo allora a valutare le periferiche che proporremo tra poco: un terzo timer, detto Timer2, ed un modulo detto Capture Compare, che sarà però in grado di svolgere o l'una o l'altra funzione.

Il Timer2

Dopo il Timer0 ad otto bit ed il Timer1 a sedici bit, ne troviamo un terzo ad otto bit, il cui schema a blocchi è visibile in Figura 8.

A questo punto ne possiamo vedere le caratteristiche principali: otto bit, prescaler selezionabile tra 1:1, 1:4 e 1:16, postscaler selezionabile tra 1:1 e 1:16 in passi da uno.

A differenza degli altri due timer, questo però può essere incrementato solamente dal clock del chip principali, ovviamente tramite il prescaler appena citato.

Al registro di conteggio vero e proprio TMR2 è associato il registro PR2, detto "di periodo".

Durante il normale funzionamento, questi due registri vengono continuamente confrontati e si ottiene un interrupt quando i valori di questi due registri sono uguali.

Tale interrupt è comunque soggetto al postscaler: questo significa quindi che l'interrupt potrebbe anche essere volutamente ritardato rispetto al momento del confronto stesso.

Per attivare il Timer2, che altrimenti resterebbe in riposo, si devono settare correttamente i bit del registro T2CON (Timer2 CONTROL register) visibili in Figura 9. I primi due bit consentono l'impostazione del prescaler:

Tabella 4. Registri associati al Timer2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
0C	PIR1	PSPIF	-	-	-	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	-	-	-	SSPIE	CCP1IE	TMR2IE	TMR1IE
11	TMR2	Timer2							
12	T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
92	PR2	Timer2 period Register							

Legend - = Unimplemented locations, Read as '0'
 Note: Shaded boxes are not used by Timer2 module.

Tabella 5. Risoluzione del PWM ad 8, 1 e 10 bit

Max Resolution (High Resolution Mode)	Frequency		
	TMR2 Prescale = 1	TMR2 Prescale = 4	TMR2 Prescale = 16
10 bit	19.53 kHz	4.88 kHz	1.22 kHz
9 bit	39.06 kHz	9.77 kHz	2.44 kHz
8 bit	78.13 kHz	19.53 kHz	4.88 kHz

mune, non possono coesistere simultaneamente, quindi dovremo decidere necessitiamo di un modulo capture o un modulo compare, ma non solo: notiamo che in comune c'è anche la periferica già vista del Timer1.

Quindi, per sfruttare ad esempio il modulo capture, dovremo rinunciare sia al modulo compare, sia alla periferica Timer1.

Tabella 6. Risoluzione del PWM a 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.13 kHz	157.5 kHz	210.53 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFF	0xFF	0xFF	0xFF	0x7F	0x5F
Resolution (High-resolution mode†)	10-bit	10-bit	10-bit	8-bit	7-bit	6.5-bit
Resolution (Standard-resolution mode†)	8-bit	8-bit	8-bit	6-bit	5-bit	4.5-bit

† Standard resolution mode has the CCPIX: CCPIY bit constant (or'0'), and only compares the TMR2 against the PR2. The Q-cycles are not used.

T2CKPS1	T2CKPS0	PRESCALER
0	0	1:1
0	1	1:4
1	X	1:16

Il bit TMR2ON ordina al chip di attivare (se a 1) o meno (se a 0) la periferica Timer2.

Con i restanti bit TOUTPS si imposta il valore di divisione del postscaler, che, come visto, passa da 1:1 a 1:16 in passi da 1.

Ciò è molto importante perché significa che possiamo dividere anche per

un numero primo come 7 oppure 11 o 13. Nella Tabella 4 sono disponibili tutti i registri associati al Timer2.

Visto così, il Timer2 potrebbe far pensare ad un comune timer aggiunto al chip in ausilio dei primi due, ma vedremo nel corso delle puntate che servirà anche più frequentemente di quanto ci si aspetti.

Il modulo Capture/Compare

Il PIC16C74, come altri, è dotato di due periferiche dette moduli "capture e compare".

Che cosa sono ed a che cosa servono? Per rispondere a questa domanda dobbiamo prima capire quale sia la loro struttura e come questi agiscano.

In Figura 10a e 10b troviamo rispettivamente il diagramma a blocchi del modulo capture e del modulo compare.

La prima notazione da fare è che, avendo elementi in co-

rica Timer1. Soffermiamoci adesso sul modulo capture. Si vede che il Timer viene incrementato da un segnale presente sul pin RCy/CCPx, dopo aver attraversato un prescaler ed un discriminatore di fronti.

Nel modo Capture, i due registri dedicati CCPRxH e CCPRxL vengono caricati con lo stesso valore presente rispettivamente nei due registri TMR1H e TMR1L quando accade un certo evento sul pin RC2/CCP1.

Per evento intendiamo le quattro possibili situazioni: un fronte di discesa, un fronte di salita, ogni 4 fronti di salita, ogni 16 fronti di salita.

In pratica, abbiamo capito che possiamo ottenere un interrupt, ad esempio, tra due fronti consecutivi di salita relativi ad un certo segnale e poi leggere il valore del tempo trascorso tra questi due eventi direttamente nei registri prima citati.

Visto così potrebbe non sembrare di molta utilità, ma supponiamo di volere realizzare una centrale antifurto con il decoder del ricevitore radio direttamente nel microcontroller.

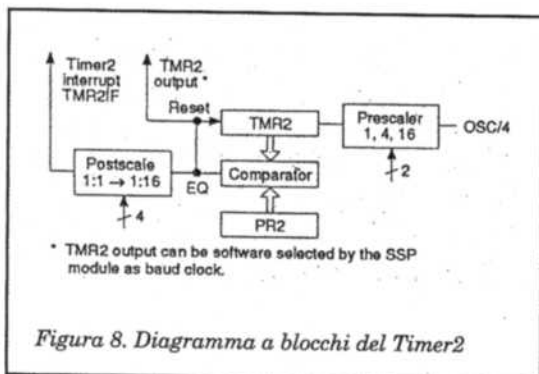


Figura 8. Diagramma a blocchi del Timer2

