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.

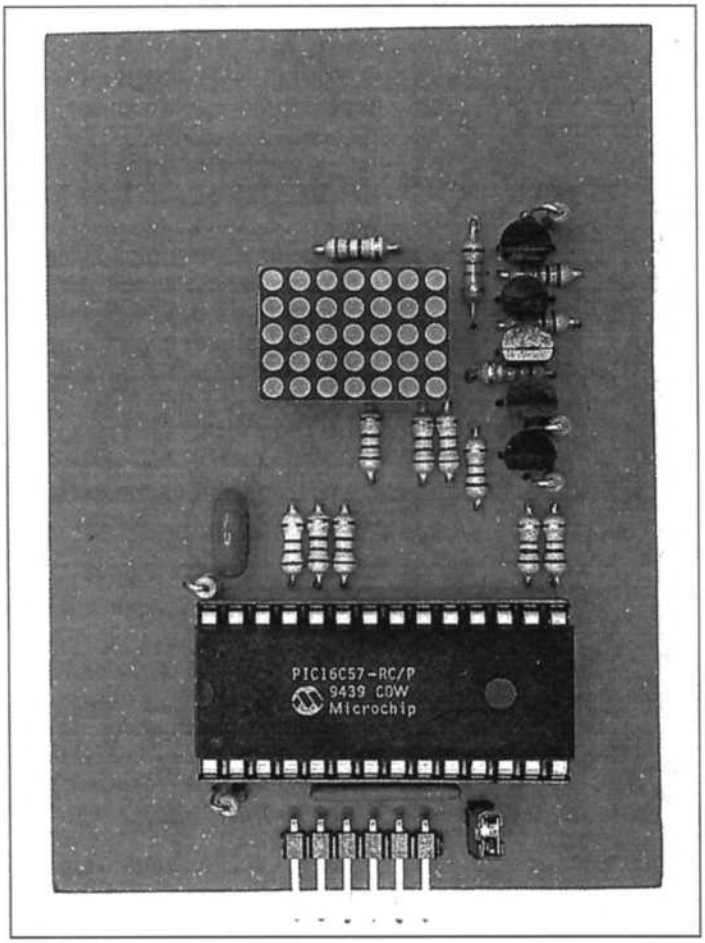
In generale, possiamo dire che le famiglie 16C5X, 16C55X e 12C50X, hanno soltanto la periferica del Timer 0, la famiglia 12C67X in più hanno anche un convertitore A/D ad 8 bit.

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.

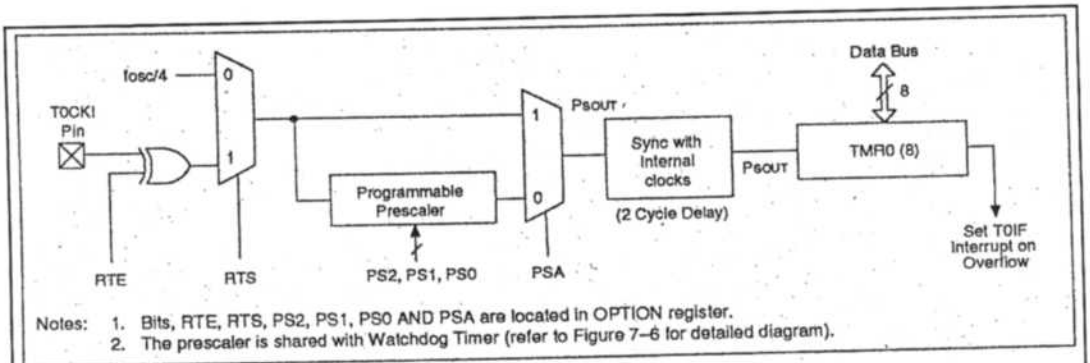
Ma capiamo come funziona questo timer: il registro TMR0 viene incrementato di una unità a seconda di eventi che tra poco vedremo. Quando questo registro raggiunge lo 0 (ovvero passa da 255 a 255 + 1 = 0), si ha un interrupt dedicato (oppure si va a testare in continuazione il suo valore).

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



Notes: 1. Bits, RTE, RTS, PS2, PS1, PS0 AND PSA are located in OPTION register.
2. The prescaler is shared with Watchdog Timer (refer to Figure 7–6 for detailed diagram).

Figura 1. Diagramma a blocchi del Timer 0

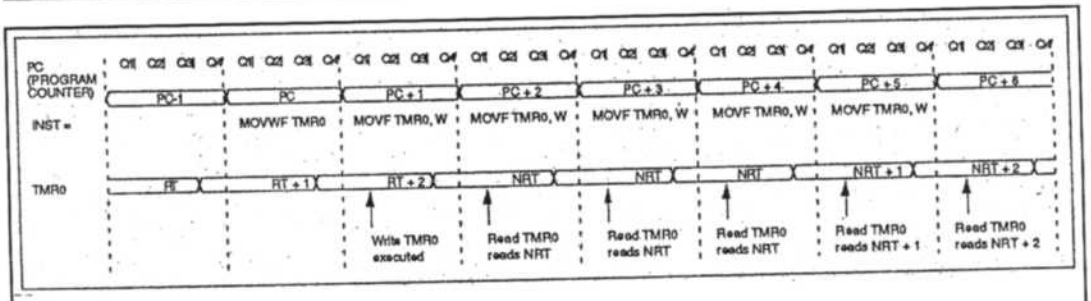


Figura 2a) Cicli senza prescaler

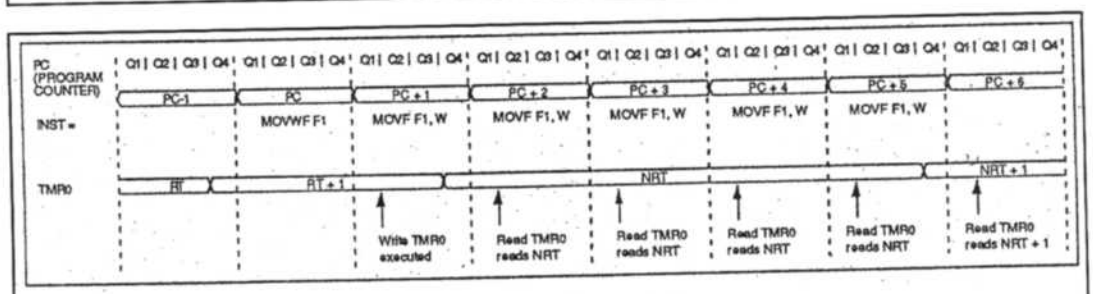


Figura 2b) Cicli con prescaler 1:2

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.

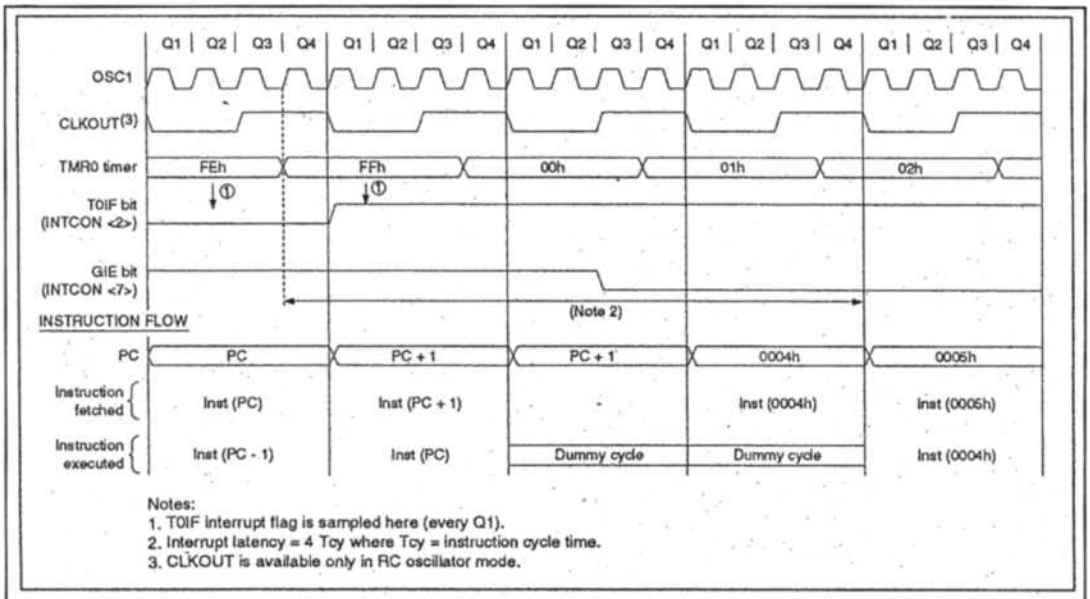


Figura 3. Diagramma dei tempi dopo interrupt del Timer0, la più comune e utilizzata periferica

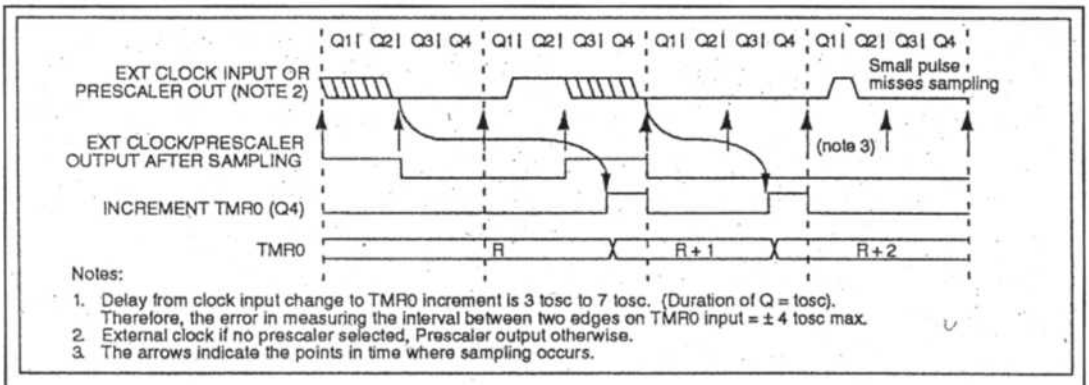


Figura 4. Andamento temporale della periferica Timer0 rispetto al fronte esterno

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

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. azzera il prescaler.

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

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 ~ 30 pf is recommended

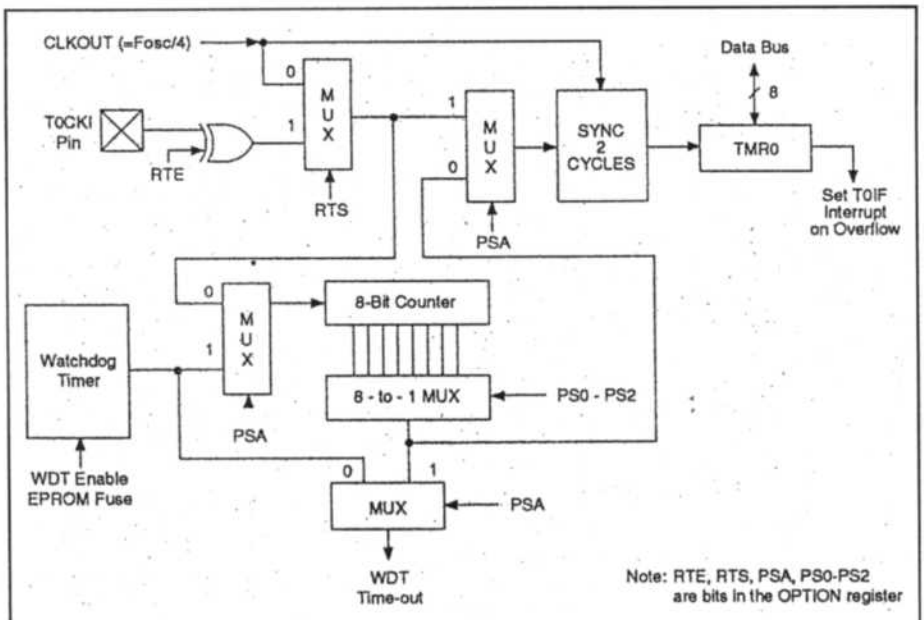


Figura 5. Diagramma a blocchi del modulo prescaler-TMR0-WDT; da notare che RTE, RTS, PSA, PS0-PS2 nel registro OPTION

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:

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	T0IE	INTE	RBIE	T0IF	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	T1OSCENT	T1INSYNC	TMR1CS	TMR1ON

Legend – = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used by Timer1 module.

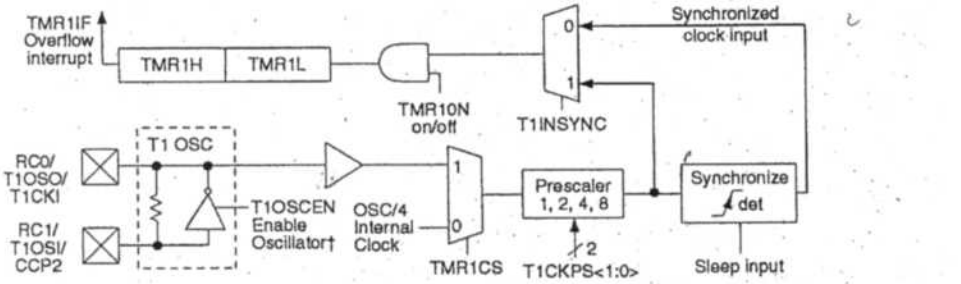


Figura 6. Diagramma a blocchi del modulo Timer1: uno dei più usati nella gestione delle interfacce

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 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

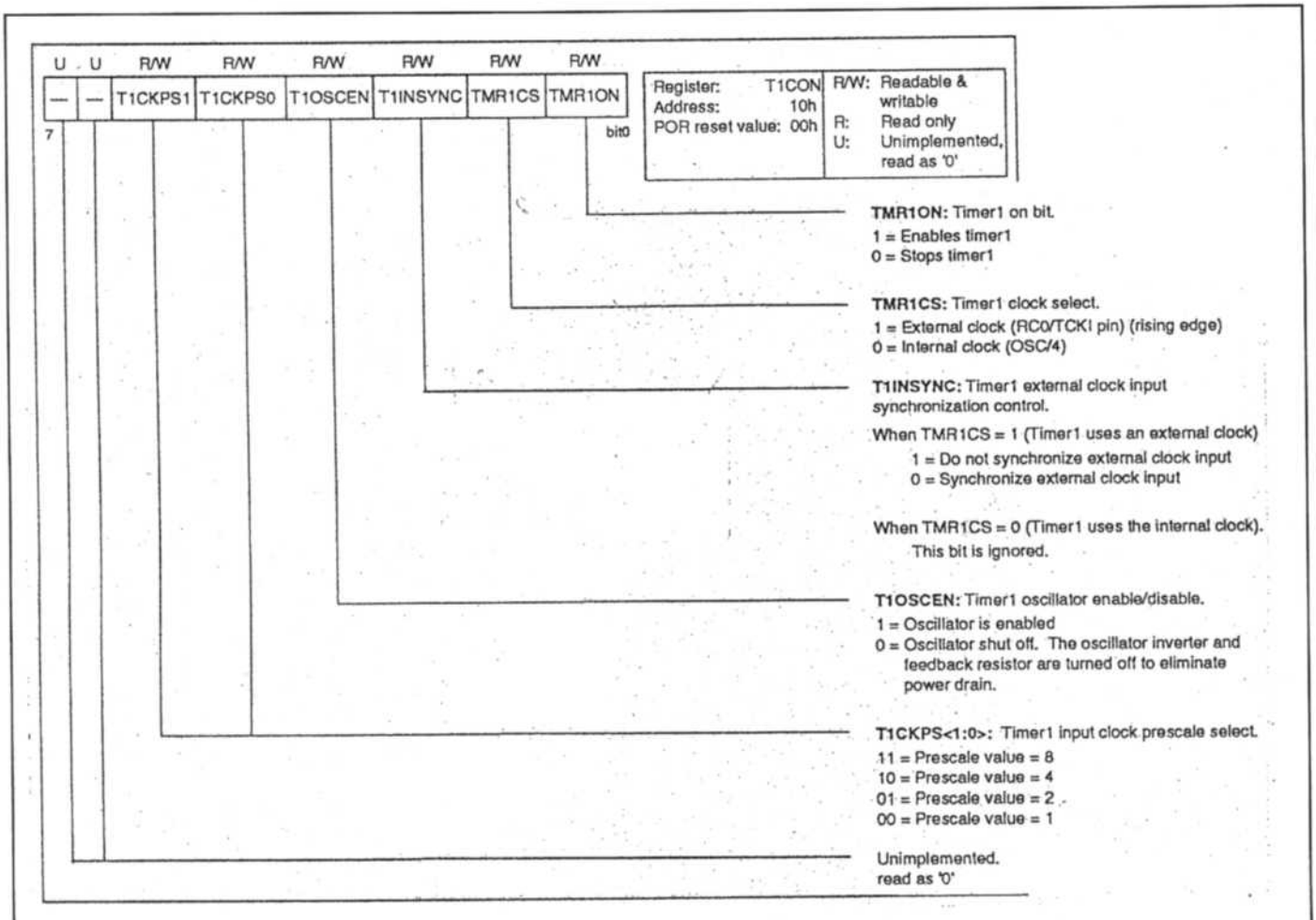


Figura 7. Registro di controllo del Timer1

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 processori, ma è necessario saperli connettere con il mondo esterno: ecco quindi le istruzioni su come fare!

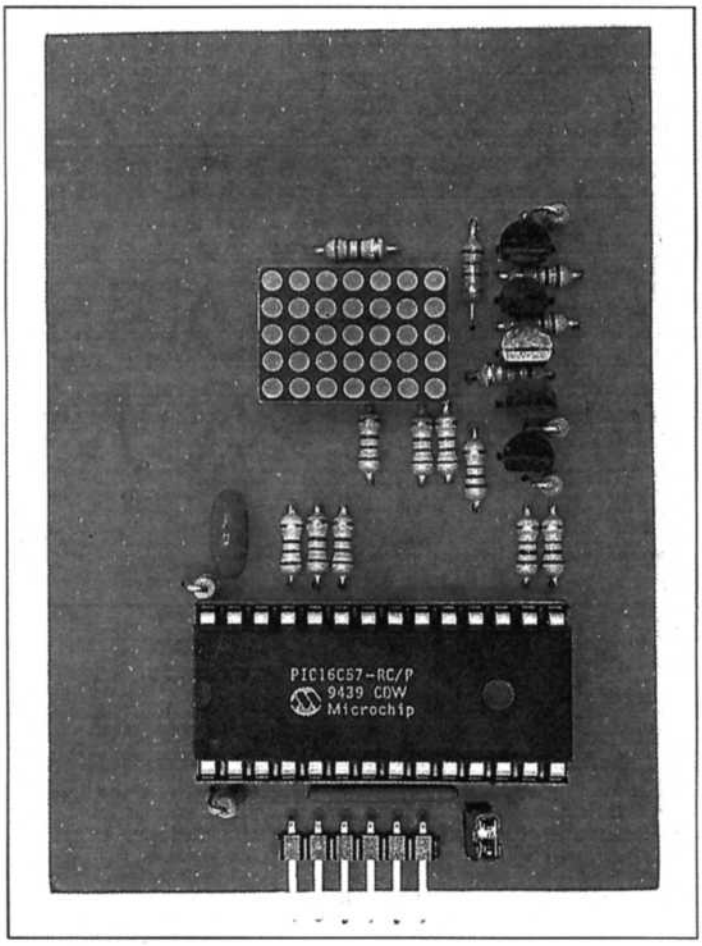
Paolo Sbrana - 2ª parte

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	T0IE	INTE	RBIE	T0IF	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	T2CKPS(
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 simult neamente, quindi dovremo decidere necessitiamo di un modulo capture o un modulo compare, ma non solo: n tiamo che in comune c'è anche la per ferica già vista del Timer1.

Quindi, per sfruttare ad esempio u modulo capture, dovremo rinuncia sia al modulo compare, sia alla perif

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 agains 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.

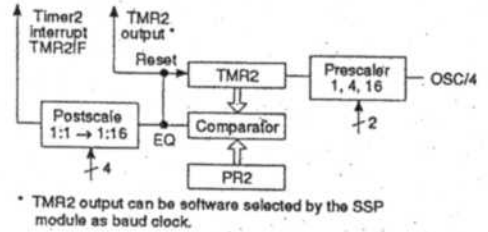


Figura 8. Diagramma a blocchi del Timer2

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 su modulo capture. Si vede che il Timer viene incrementato da un segnale pre sente sul pin RCy/CCPx, dopo aver at traversato un prescaler ed un discrim natore di fronti.

Nel modo Capture, i due registri de dicati CCPRxH e CCPRxL vengono ca ricati con lo stesso valore presente ri spettivamente nei due registri TMR1H e TMR1L quando accade un certo even to sul pin RC2/CCP1.

Per evento intendiamo le quattr possibili situazioni: un fronte di disce sa, un fronte di salita, ogni 4 fronti d salita, ogni 16 fronti di salita.

In pratica, abbiamo capito che pos siamo ottenere un interrupt, ad esem pio, tra due fronti consecutivi di salita relativi ad un certo segnale e poi legge re il valore del tempo trascorso tra questi due eventi direttamente nei re gistri prima citati.

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

Tabella 7a. Registri associati a capture e Timer1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	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	T1XKPS0	T1OSCENT	T1INSYNC	TMR1CS	TMR1ON
15	CCPR1L	Timer1 Capture Register (LSB)							
16	CCPR1H	Timer1 Capture Register (MSB)							
17	CCP1CON	–	–	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
1A	CCPR2L	Timer1 Capture Register (LSB)							
1B	CCPR2H	Timer1 Capture Register (MSB)							
1C	CCP2CON	–	–	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0

Legend – = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used in this mode.

Tabella 7b. Registri associati a compare e Timer1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	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	T1XKPS0	T1OSCENT	T1INSYNC	TMR1CS	TMR1ON
15	CCPR1L	Timer1 Compare Register (LSB)							
16	CCPR1H	Timer1 Compare Register (MSB)							
17	CCP1CON	–	–	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
1A	CCPR2L	Timer1 Compare Register (LSB)							
1B	CCPR2H	Timer1 Compare Register (MSB)							
1C	CCP2CON	–	–	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0

Legend – = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used in this mode.

Tabella 7c. Registri associati a PWM e Timer2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	–	–	–	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	–	–	–	SSPIE	CCP1IE	TMR2IE	TMR1IE
11	TMR2	Timer2							
92	PR2	Timer2 period Register							
12	T2CON	–	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CPS0
15	CCPR1L	Timer2 Duty Cycle Register							
16	CCPR1H	Timer2 Duty Cycle Register (Slave)							
17	CCP1CON	–	–	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
1A	CCPR2L	Timer2 Duty Cycle Register							
1B	CCPR2H	Timer2 Duty Cycle Register (Slave)							
1C	CCP2CON	–	–	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0

Legend – = Unimplemented locations, Read as '0'
Note: Shaded boxes are not used in this mode.

Allora, calcolando correttamente le tempistiche di incremento del Timer1 e chiedendo un interrupt ad ogni fronte di salita del segnale ricevuto, sarà possibile ricevere bit per bit tutto il codice del radiocomando, con il vantaggio di lavorare sotto interrupt e, quindi, di dedicarvi pochissimo tempo di CPU.

Gli allarmi auto delle nuove generazioni, sfruttano nella maggioranza dei casi un metodo analogo per ridurre il circuito ad un solo integrato che svolga

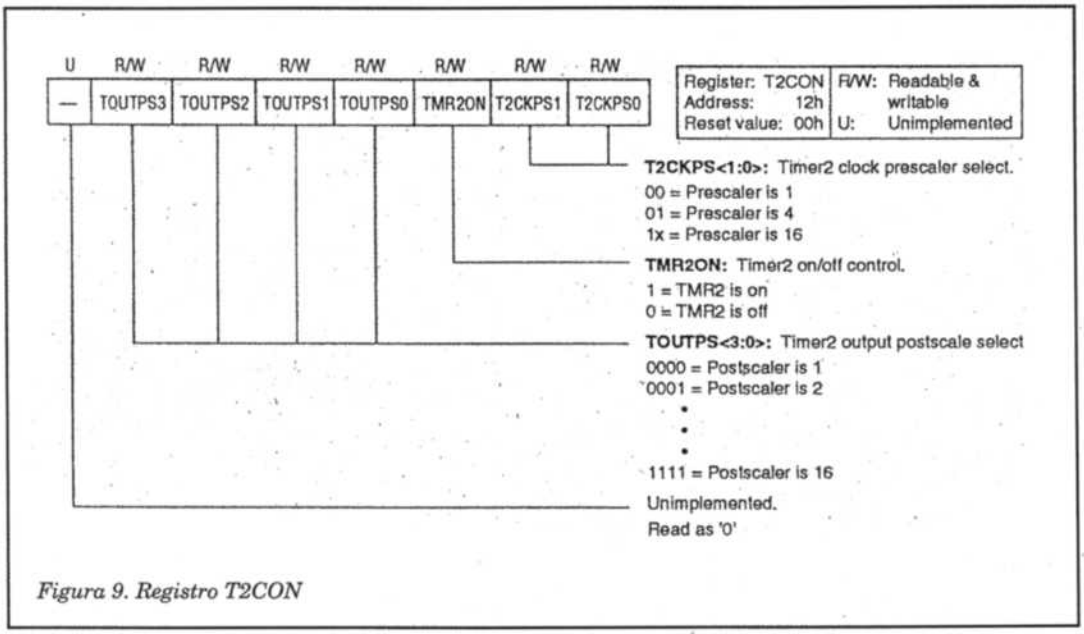


Figura 9. Registro T2CON

però tutti i compiti richiesti. Ovviamente quella citata è soltanto una delle molteplici applicazioni del modulo Capture, ma risulta essere la più ricercata dalla maggior parte dei progettisti firmware.

Per quanto riguarda questo modulo, dobbiamo fare una notazione relativa al prescaler che può essere importante in fase di sviluppo di nuovi propri progetti.

Quando il modulo non è attivo, il prescaler è azzerato.

Il passare da un valore ad un altro del prescaler potrebbe generare un interrupt.

Quindi, il contatore del prescaler non sarà azzerato e la prima lettura potrebbe essere falsata. Per ovviare a questo si suggeriscono le seguenti righe di software:

```
clrf   CCP1CON   ;Spegni
                 modulo CCP
movlw  NEWVAL    ;Carica nuovo
                 valore prescaler
movwf  CCP1CON   ;in CCP1CON
```

Ovviamente, per poter sfruttare il modulo capture, come quello compare, il Timer1 deve essere in funzione in modalità sincrona e non asincrona.

Il modulo Compare, invece, come dice il nome stesso, compara il valore dei due registri CCPRxH e CCPRxL con quelli corrispondenti del Timer1 e dà un interrupt quando il risultato è identico.

Anche per questo modulo, le applicazioni potrebbero sembrare poche, ma vedremo che così non è, magari in abbinamento al modulo Capture. In definitiva, possiamo definire un tempo X ed ottenere un interrupt passato tale tempo.

Sempre per rimanere nell'importante campo della sicurezza, questo modulo viene sfruttato per generare la rampa di frequenze idonea a far oscillare la sirena di un antifurto.

L'ultima periferica che andiamo a valutare questo mese è il generatore di PWM, il cui diagramma a blocchi è ben visibile in Figura 11: come si nota, questa volta il timer associato è il Timer2.

Per chi ancora non lo sapesse, diciamo che PWM è l'acronimo di Pulse Width Modulation, ovvero modulazione a larghezza di impulso.

Ciò significa che, ad esempio, per pilotare un motore in corrente continua si scelga una frequenza fissa e poi se ne vari il duty-cycle per far variare la

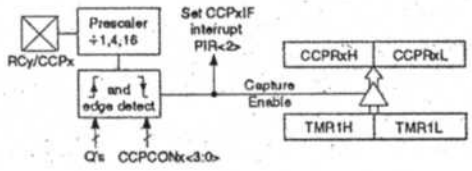
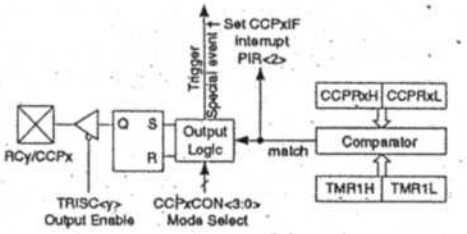
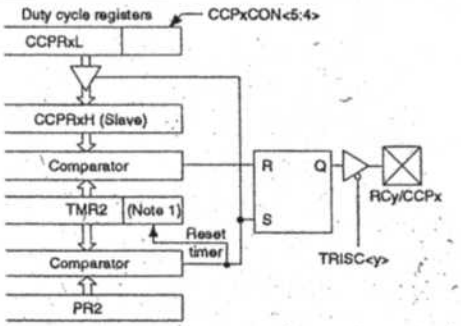


Figura 10a. Diagramma a blocchi del modulo Capture



† For CCP1 (if enabled), reset Timer1.
For CCP2 (if enabled), resetTimer1 one set GO bit (ADCON0<2>).

Figura 10b. Diagramma a blocchi del modulo Compare



te: 8-bit timer is concantenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

Figura 11. Diagramma a blocchi del modulo PWM

velocità del motore stesso. Molto importante è che la frequenza, una volta impostata, non vari, perché la tensione di uscita diventa funzione anche della frequenza.

Con il modulo PWM inserito nei PIC, abbiamo a disposizione, come si vede nelle Tabelle 5 e 6 molte frequenze e un controllo che può arrivare fino ai 10 bit.

Se per esempio lavoriamo con 20 MHz, è possibile pilotare un PWM a 19,53 kHz con la risoluzione di 10 bit, ovvero di 1.024 passi.

Uno degli svantaggi di questo modulo, è che non consente di definire frequenze a piacere, ma soltanto calcolate in un certo modo.

Allora, se desideriamo ottenere una ben precisa frequenza di PWM, dovremmo calcolarci un quarzo appositamente, tenendo presente che il periodo del PWM è (PR2 +1)*4tosc* (valore del prescaler del Timer2).

Se allora impieghiamo un quarzo da 20 MHz, settando a 255 il registro PR2 ed il prescaler a 1:1, la frequenza del PWM sarà di circa 19.530 Hz gestibile con una risoluzione di massimo 10 bit. In Figura 12, abbiamo il registro CCP1CON: i primi 4 bit servono per impostare la modalità di funzionamento del modulo, mentre i restanti 2 servono per indicare alla periferica, soltanto in modalità PWM, se si desidera avere una risoluzione ad 8 oppure a 10 bit.

Le modalità prima citate sono:

1 - Modulo Capture/Compare/PWM off (tutto spento)
2 - Modo Capture ad ogni fronte di discesa
3 - Modo Capture ad ogni fronte di salita
4 - Modo Capture ogni 4 fronti di salita
5 - Modo Capture ogni 16 fronti di salita
6 - Modo Compare: setta l'uscita CCPx se risultato identico
7 - Modo Compare: azzera l'uscita CCPx se risultato identico
8 - Modo Compare: genera interrupt se risultato identico
9 - Modo Compare: controllo trigger
10 - Modo PWM

Per avere la conoscenza dei registri impiegati da queste periferiche, nelle Tabelle 7a, 7b e 7c ne troviamo l'elenco completo utile anche da conservare.

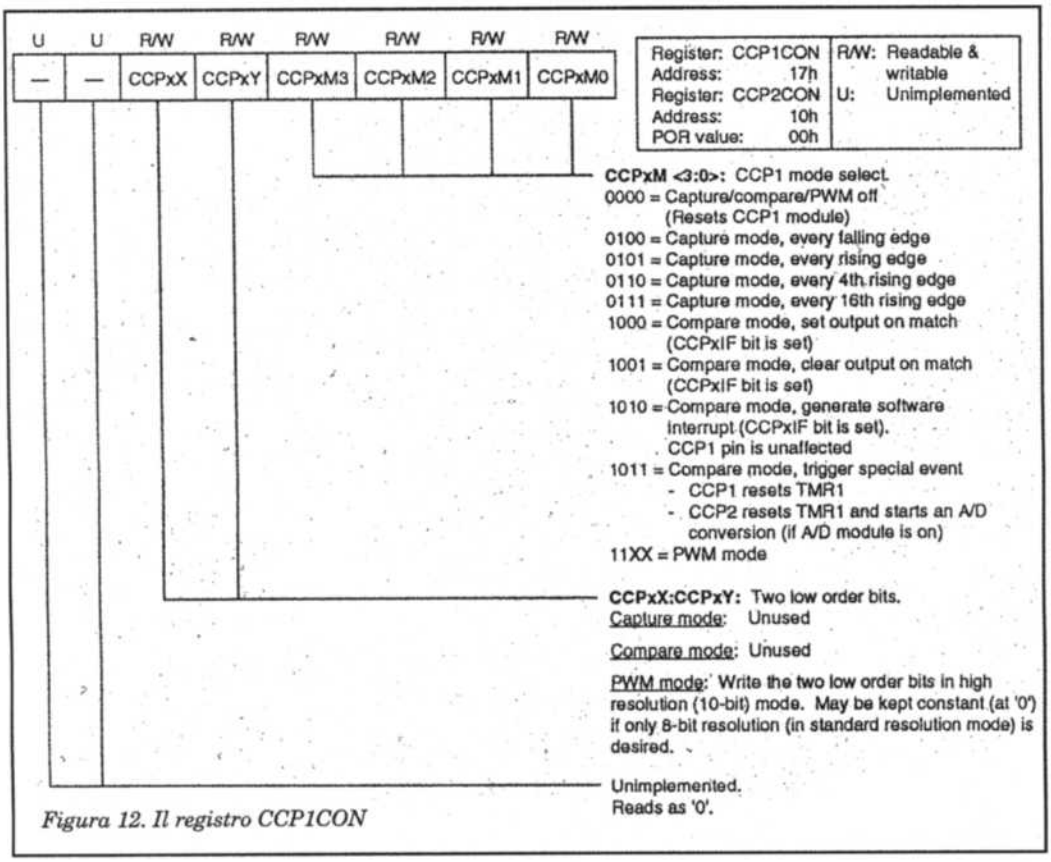


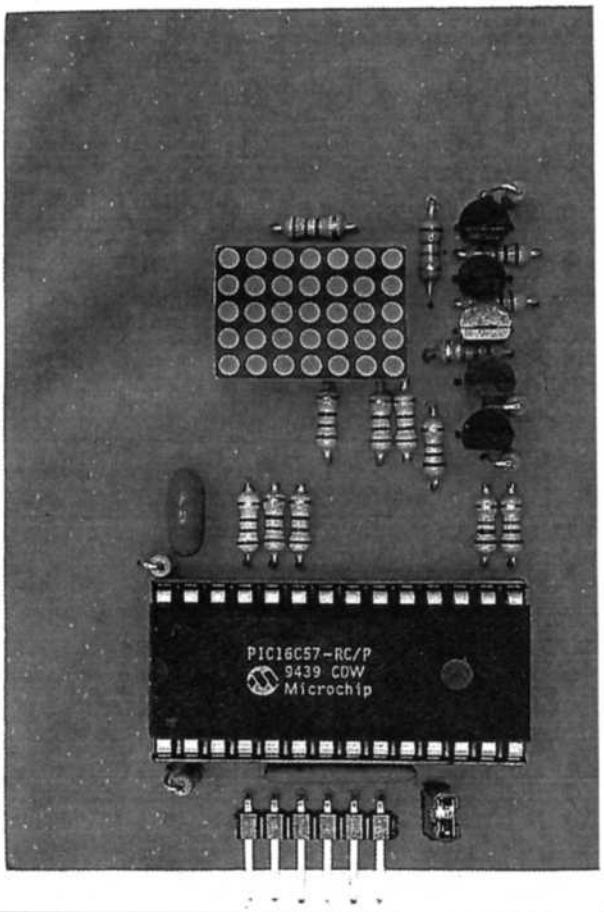
Figura 12. Il registro CCP1CON

Ma non finisce qui

Il prossimo mese inizieremo a trattare il modulo della seriale, che addirittura è previsto in sincrono ed in asincrono.

continua

LE PERIFERICHE DEI PIC: FACCIAMONE CONOSCENZA



Continuiamo nella nostra trattazione delle periferiche dei Pic, il necessario complemento a ogni progetto basato sui processori in tecnologia Risc

Paolo Sbrana - 3ª parte

Fino ad adesso, abbiamo preso in considerazione i timer dei PIC, anche associati ad altri moduli come il Capture oppure il Compare, ma questo mese tratteremo una delle periferiche più sfruttate da chi impiega i microcontroller nei propri lavori: la connessione seriale.

Nei PIC ci sono due tipi di porta seriale. La prima, che vedremo in questo articolo, è detta SSP (Synchronous Serial Port), mentre la seconda, che tratteremo il mese prossimo, è detta SCI (Serial Communication Interface).

La SSP è soltanto di tipo sincrono, mentre la seconda può essere sia di tipo sincrono che di tipo asincrono.

Con la SSP, quindi, ci potremo interfacciare con bus e periferiche che dialogano con un segnale di clock, come per esempio il bus IIC oppure il Microwire, mentre con la SCI possiamo dialogare con un computer attraverso una seriale standard RS232.

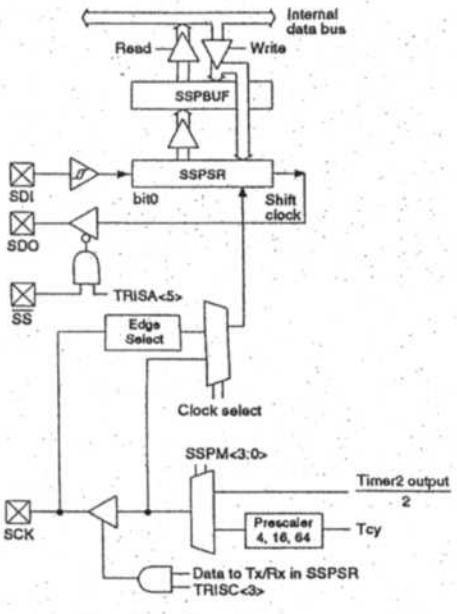


Figura 13. Diagramma a blocchi della SSP in modo SPI

Il modulo SSP

Il modulo della seriale SSP è una periferica che, come abbiamo accennato, consente il dialogo con altre periferiche o microcontroller, che ad esempio possono essere memorie seriali eeprom, shift register, driver per display, convertitori A/D e/o D/A, ecc.

Questo modulo può operare in due modi distinti: o in SPI (Serial Peripheral Interface) oppure in IIC (Inter-Integrated Circuit).

Vediamo per primo il modo di funzionamento in SPI: in Figura 13 ne troviamo il diagramma a blocchi. Dal chip verso l'esterno sono disponibili quattro segnali: SDI (Serial Data Input), SDO (Serial Data Output), SS (Slave Select) e SCK (Serial ClocK).

Generalmente, lo Slave Select viene impiegato però solo quando la periferica viene utilizzata come SLAVE. Al momento di inizializzazione della porta SPI, dobbiamo specificare diverse opzioni andando a settare correttamente i bit del registro SSPCON, il cui significato è visibile in Figura 14.

Con i primi quattro bit si decide se

PROGRAMMA 3

```
LOOP    bsf     STATUS,RP0      ;Banco 1
        btfss   SSPSTAT,BF      ;Operazione terminata?
        goto    LOOP            ;No
        bcf     STATUS,RP0      ;Banco 0
        movf    SSPBUF,W        ;Copia SSPBUF in W
        movwf   RXDATA          ;Copia W in RXDATA
        movf    TXDATA,W        ;Copia TXDATA in W
        movwf   SSPBUF          ;Copia W in SSPBUF
```

configurare la periferica come SPI MASTER (con clock diviso per 4, 16 o 64), come SPI SLAVE (con o senza gestione del controllo SS), come IIC MASTER oppure come IIC SLAVE (con indirizzo a 7 o 10 bit).

Il bit CKP (ClocK Polarity) consente di definire in modo SPI se la trasmissione deve avvenire sul fronte di salita e la ricezione sul fronte di discesa o viceversa, mentre in modo IIC se lasciare il controllo del clock.

Il bit SSPEN (Sync Serial Port ENable) abilita o meno la perifica al funzionamento seriale oppure configura i pin interessati come normali linee di I/O. Al reset, la periferica è disabilitata.

Il bit SSPOV (Sync Serial Port OVerflow) indica in entrambe le modalità che un byte è stato ricevuto correttamente.

Il bit WCOL (Write COLlision) indica se ci sono state collisioni sul bus seriale (sempre possibili ad esempio in un sistema multimaster). Lo stato della periferica è, invece, localizzato nel registro SSPSTAT mostrato in Figura 15.

Il bit BF (Buffer Full) indica se il registro SSPBUF è pieno o non ancora riempito.

Il bit UA (Update Address) serve solamente in modalità IIC e indica che l'indirizzo nel registro SSPADD deve essere aggiornato. Il bit R/W (Read/Write) serve anch'esso in modalità IIC e indica la scrittura o la lettura sul bus. Il bit S (Start bit) è molto importante perché ci indica quando è stata riconosciuta una condizione di Start (ovviamente in modalità IIC).

Allo stesso modo, il bit P (stoP bit) ci indica quando è stata riconosciuta una condizione di Stop.

Anche il bit D/A (Data Address) serve soltanto in modalità IIC e ci dice se l'ultimo byte ricevuto è un byte di dato oppure un indirizzo.

Tornando alla Figura 13, il modulo SSP consiste di un registro di ricezione/trasmissione (SSPSR) e di un buffer (SSPBUF).

Il registro SSPSR shifta i dati in ingresso e in uscita dalla periferica, mentre il registro SSPBUF mantiene memorizzato l'ultimo byte completo scritto nel SSPSR fino a quando il dato ricevuto non è valido.

Quando ciò accade, ovvero sono giunti gli 8 bit del dato, tale informazione viene copiata in SSPBUF e i bit BF e SSPIF visti precedentemente vengono settati. Questo sistema del doppio buffer, ci consente di avviare la ricezione del byte successivo prima ancora di aver letto il dato precedente, con ovvio risparmio di tempo. Ogni tentativo di scrittura nel registro SSPBUF durante una ricezione o una trasmissione verrà ignorata, ma verrà settato il bit delle collisioni WCOL. Sarà compito del software di resettare tale bit.

Quando il software è in attesa di un nuovo dato, il registro SSPBUF do-

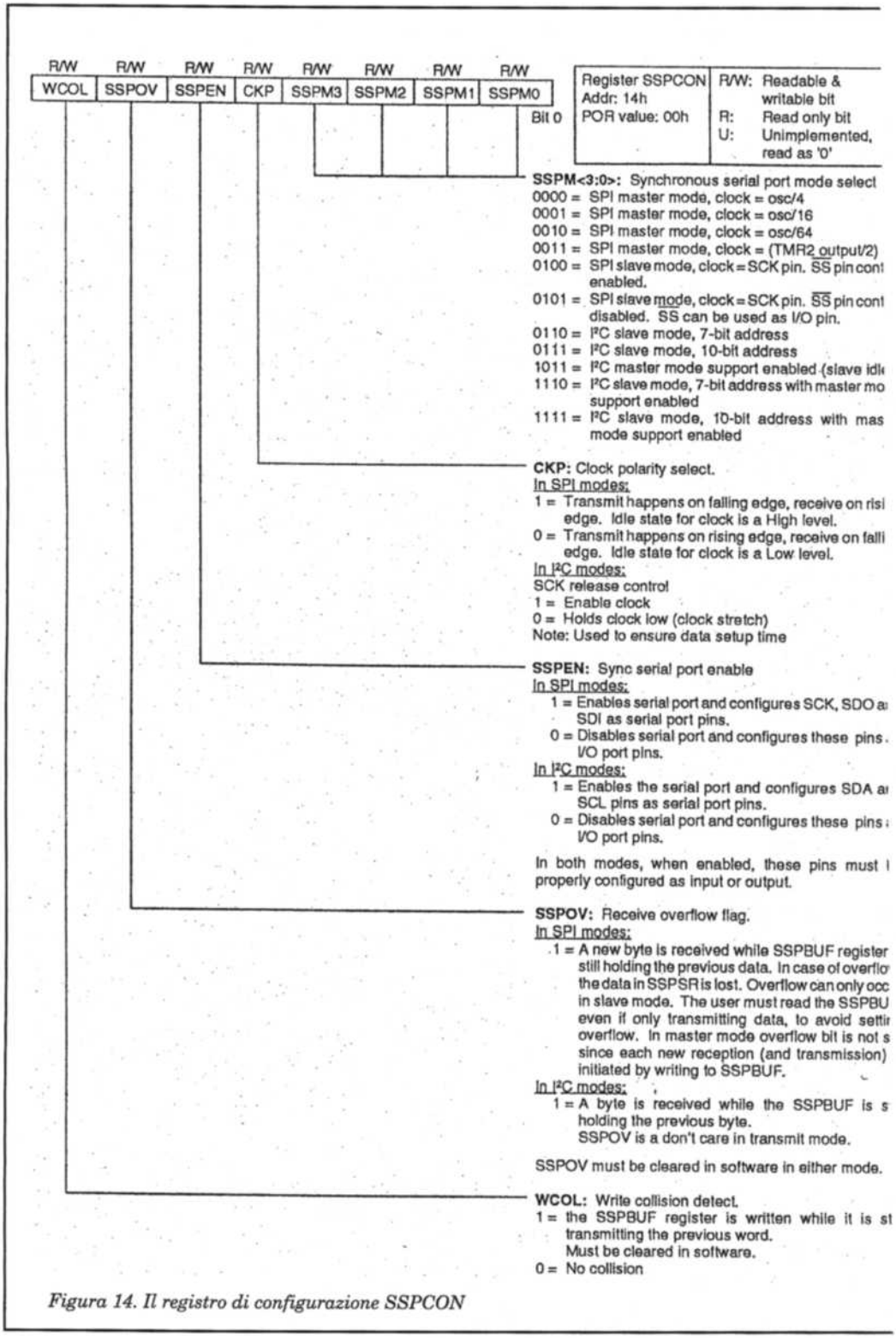


Figura 14. Il registro di configurazione SSPCON

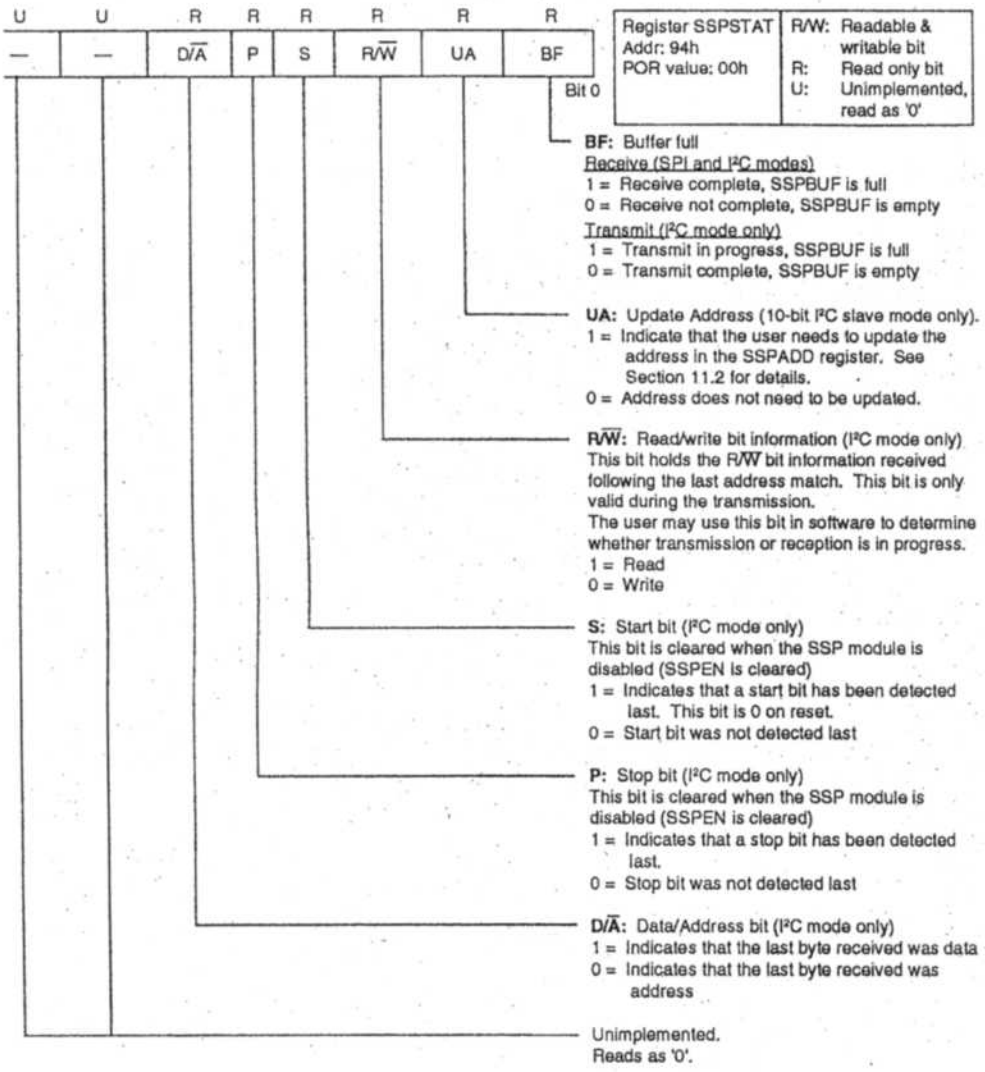


Figura 15. Il registro di stato SSPSTAT

vrebbe essere letto prima che ci venga scritto dal nuovo dato. Il bit BF ci aiuta indicandoci quando l'SSPBUF è stato caricato correttamente (trasmissione completata). Quando il registro SSPBUF viene letto, automaticamente il bit BF viene azzerato. Ovviamente questo è irrilevante nel caso in cui si utilizzi la periferica SPI solo in trasmissione.

Normalmente, si sfrutta l'interrupt dell'SSP per determinare se la trasmissione o la ricezione sono terminate. Se però non viene abilitato l'interrupt dell'SSP, il programmatore dovrà testare a polling il bit BF per sapere quando la comunicazione è avvenuta.

Vediamo allora un esempio di come caricare il registro SSPBUF e il registro SSPSR per la trasmissione di un dato (Programma 3).

Per abilitare la periferica SSP, dobbiamo innanzitutto settare il bit SSPEN. Questo configura i pin SDI, SDO, SS e SCK come pin seriali e non come generici pin di I/O.

In ogni caso, vanno egualmente settati i bit del registro TRISC in funzione della direzione dei pin stessi.

Il pin SDI sarà un input, il pin SDO sarà un output come pure il pin SS. Il pin SCK, invece, dovrà essere di input se in MASTER mode, di output se in SLAVE mode.

La Figura 16 mostra la tipica connessione tra due microcontrollers. Il controller MASTER (processor 1) inizia il trasferimento dei dati inviando il segnale di clock.

I dati sono passati in seriale. Entrambi i processori dovranno necessariamente essere programmati con le stesse opzioni ed in particolar modo con la stessa ClocK Polarity (CKP).

Il MASTER può iniziare il trasferimento dei dati in ciascun momento, perché controlla il segnale di Clock.

Inoltre, il protocollo software indica al MASTER quando lo SLAVE voglia iniziare un dialogo.

In modo MASTER, il dato viene trasmesso o ricevuto fino a quando non viene scritto il registro SSPBUF. Se l'SPI è settata solo in ricezione, l'output dell'SCK dovrebbe essere disabilitato programmando quel pin come input. Il registro SSPSR continuerà a ricevere i bit dal pin SDI alla velocità di clock programmata.

In modalità SLAVE, la velocità del clock è determinata dalla velocità del clock ricevuto dal MASTER.

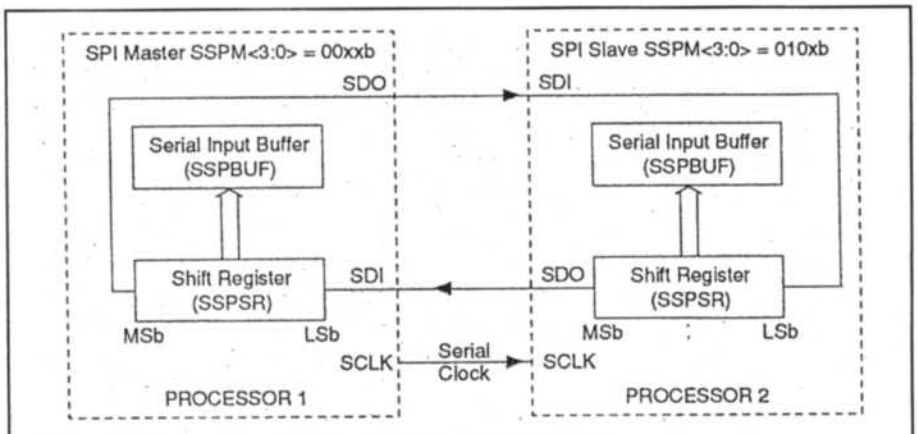


Figura 16. Connessione SPI tipo MASTER/SLAVE

Tabella 8. Registri associati alle operazioni della SPI

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	–	–	–	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	–	–	–	SSPIE	CCP1IE	TMR2IE	TMR1IE
13	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
14	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
94	SSPSTAT	–	–	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

Legend – = Unimplemented locations, Read as '0' - Note: Shaded boxes are not used by SSP module in SPI mode.

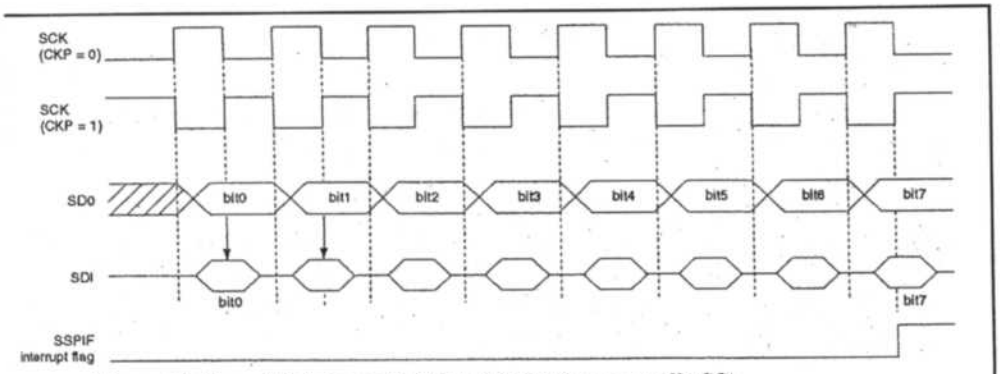


Figura 17. Tempistiche in modo SPI MASTER (con controllo SS)

Per la particolare configurazione della periferica, è possibile, lavorando con il clock del controller a 20 MHz, avere un clock sulla seriale fino a 5 MHz.

Nelle Figure 17 e 18 troviamo le tempistiche relative al dialogo della SPI rispettivamente in modalità MASTER e SLAVE.

Il controllo SS consente la modalità SLAVE sincrona. L'SPI deve essere in modalità SLAVE ed il pin 5 della porta A deve essere configurato come input.

Quando il pin SS è basso, la trasmissione e la ricezione sono abilitate e il

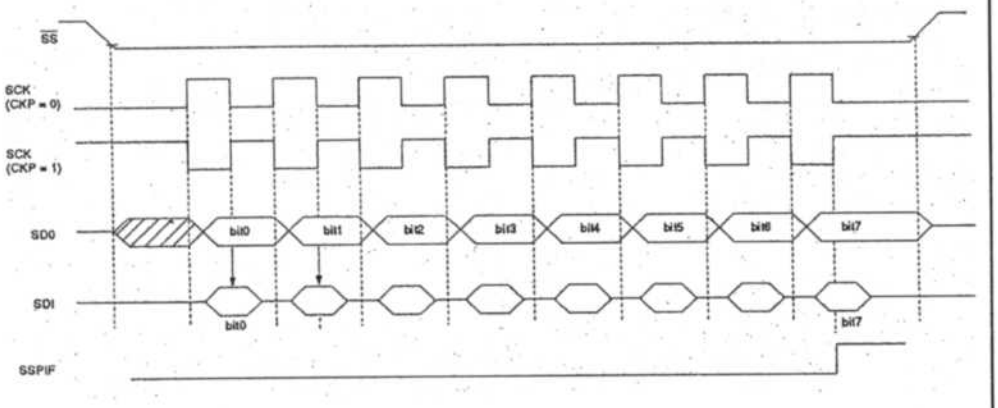


Figura 18. Tempistiche in modo SPI SLAVE (con controllo SS)

Quando la SPI deve operare come ricevitore, il pin SDO può anche essere configurato come input.

Questo disabilita, chiaramente, la trasmissione da quel pin.

Il pin SDI può sempre essere lasciato come input per evitare conflitti sul bus.

Nella Tabella 8 vediamo tutti i registri associati alle operazioni della SPI.

Il protocollo IIC-bus

Prima di spiegare come funziona il modulo relativo alla periferica IIC, diamo le informazioni di base sul funzionamento del bus IIC, che abbiamo trattato più volte sulle pagine di Progetto.

Il bus IIC è nato in casa Philips/Signetics per far dialogare più device su un numero ristretto di fili (2). Uno di questi due fili è il clock, l'altro è il dato.

Ovviamente, i dati vengono trasferiti in modo seriale ed alla frequenza di 100 kHz o 400 kHz a seconda delle device collegate.

Fondamentalmente, sul bus IIC possono risiedere diversi MASTER e diversi SLAVE, sfruttando un protocollo ormai collaudato che non dà adito a errori.

Concettualmente, un MASTER inizia un dialogo con uno SLAVE indicandolo con un indirizzo.

I successivi trasferimenti di dati sono in funzione delle esigenze dei due dialoganti.

In Figura 19 vediamo le condizioni di START e di STOP sul bus: si ha uno START quando il dato passa da alto a basso livello mentre la linea del clock è ad alto livello. Si ha invece una condizione di STOP quando il dato passa da basso ad alto livello con il clock ad alto livello.

Per indirizzare le device, sono disponibili indirizzi a 7 oppure a 10 bit: in Figura 20a e 20b se ne vede il formato. Con l'indirizzo viene inoltre passato allo slave se necessitiamo di un invio dati o se invece stiamo solo inviando un comando.

In Figura 21, vediamo come la device SLAVE risponde al MASTER una volta che è stata indirizzata (con l'ACK).

In Figura 22 invece, troviamo lo scambio dei dati sul bus.

continua

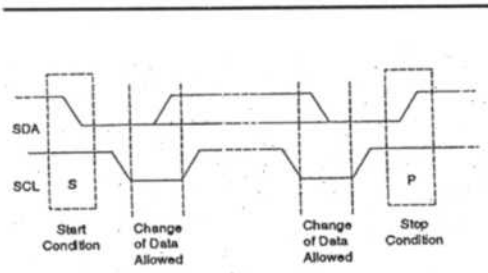


Figura 19. Condizione di START e di STOP nel bus IIC

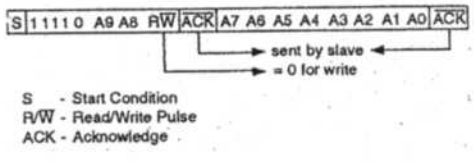


Figura 20a. Indirizzamento a 7 bit

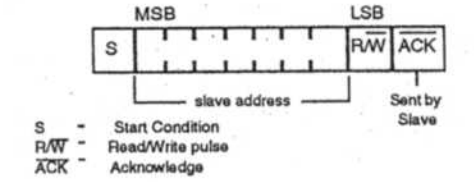


Figura 20b. Indirizzamento a 10 bit

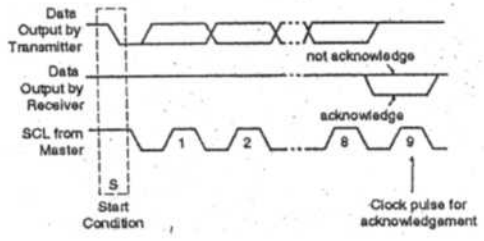


Figura 21. Rilevamento dell'ACKNOWLEDGE nel bus IIC

pin SDO è pilotato. Quando il pin SS va alto, il pin SDO diventa ad alta impedenza. Per questo motivo, in funzione delle applicazioni, è necessario inserire un resistore o di pull-up o di pull-down.

Per emulare una comunicazione a due fili, il pin SDO può essere connesso al pin SDI.

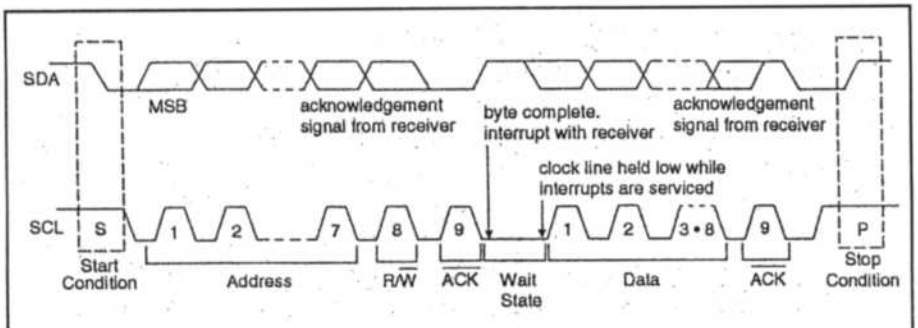


Figura 22. Trasferimento dati sul bus IIC

LE PERIFERICHE DEI PIC: FACCIAMONE CONOSCENZA

Per sfruttare appieno le potenzialità di un microcontroller, è fondamentale non solo conoscerne il principio di funzionamento, ma serve capire anche quali unità collegare al chip

Paolo Sbrana - 4ª parte

Riprendiamo la trattazione del protocollo IIC-bus riferendoci alle Figure 21 e 22 dello scorso mese, che mostravano rispettivamente il metodo di acknowledge ed il protocollo di trasferimento dati sulla linea.

Tutti i dati devono necessariamente essere trasmessi con blocchi di 8 bit, ovvero di un byte e non esiste un limite al numero di byte che è possibile trasmettere durante un dialogo.

Dopo ogni byte ricevuto, il modulo slave ricevente genera un segnale di acknowledge (ACK) cioè di avvenuta ricezione portando bassa la linea dei dati.

Se ciò non avviene, allora il master deve subito interrompere il dialogo in corso con tale modulo.

Questo meccanismo viene sfruttato quando lo slave non ha più informazioni da inviare oppure ha altri compiti da svolgere più urgenti.

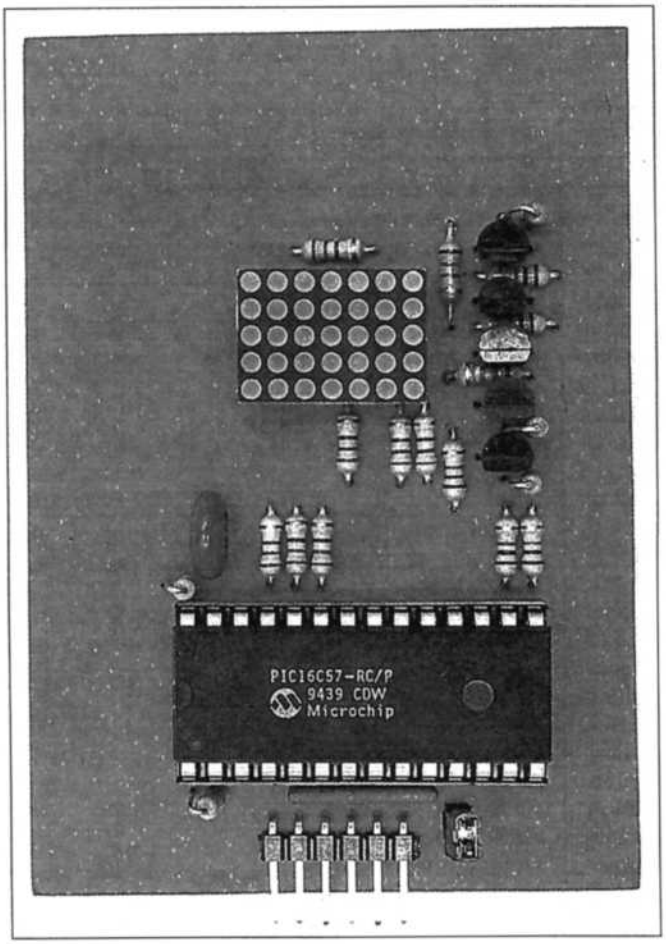
In questo caso, lo slave deve liberare la linea SDA portandola ad alta impedenza per permettere al master di generare la condizione di STOP che abbiamo già descritto.

Al master, inoltre, è consentito generare la condizione di STOP anche durante un impulso di acknowledge da parte dello slave per terminare rapidamente un dialogo. Se, durante un colloquio, lo slave ha necessità di ritardare la trasmissione di un byte, ad esempio perché si deve calcolare un certo risultato oppure perché attende a sua volta un segnale da un'altra periferica, è sufficiente che forzi a basso livello la linea del clock (SCL), portando così il master in uno stato di attesa (wait) indeterminata.

Il dialogo riprende quando lo slave decide di liberare la linea SCL.

Per mezzo di questo protocollo, che permette la messa in attesa del master anche durante il passaggio di un singolo bit, possiamo dire che in determinati momenti il master "reale" diventa slave dello slave "reale", ovvero si invertono le funzioni tra i due dialoganti.

Tutto questo però porta al problema che può sorgere quando uno slave si guasta e, per coincidenza, setta bassa la sua linea SCL: in questi casi, si bloccherà tutto il circuito fino a quando non si provvederà alla riparazione fisica.



In Figura 23 e 24, vediamo le sequenze master-transmitter e master-receiver.

Da notare che, poiché il dato deve essere obbligatoriamente di otto bit, se si devono indirizzare più di sette bit (uno sta per Read/Write) è necessario inviare due byte, dove nel primo ci saranno i primi 7 bit dell'indirizzo e nel secondo i rimanenti.

Ovviamente, sta allo slave saper interpretare il significato dei due byte di indirizzo.

Quando un master non vuole rilasciare il bus (generando una condizione di STOP) deve essere generata da uno slave un'altra condizione di START.

Poiché quest'ultima avviene non durante una condizione di bus libero, ma dopo l'acknowledge di un trasferimento dati, permette al master di inviare un comando allo slave per ricevere ugualmente la sua richiesta.

Questa sequenza è rappresentata in Figura 25.

Tabella 9. Possibili azioni dopo la ricezione di un byte

Status Bits as Data Transfer is Received				
BF	SSPOV	SSPSR - SPBUF	Generate $\overline{ACK}$ Pulse	Set SSPIF bit (SSP Interrupt if Enable)
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	No	No	Yes

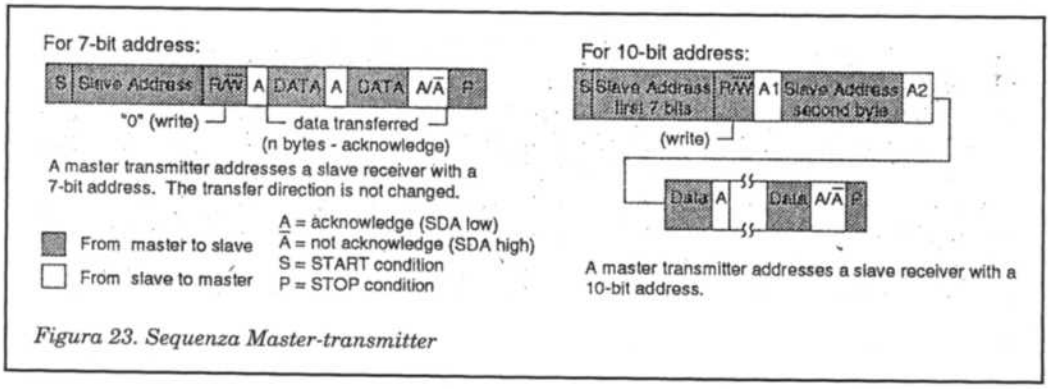


Figura 23. Sequenza Master-transmitter

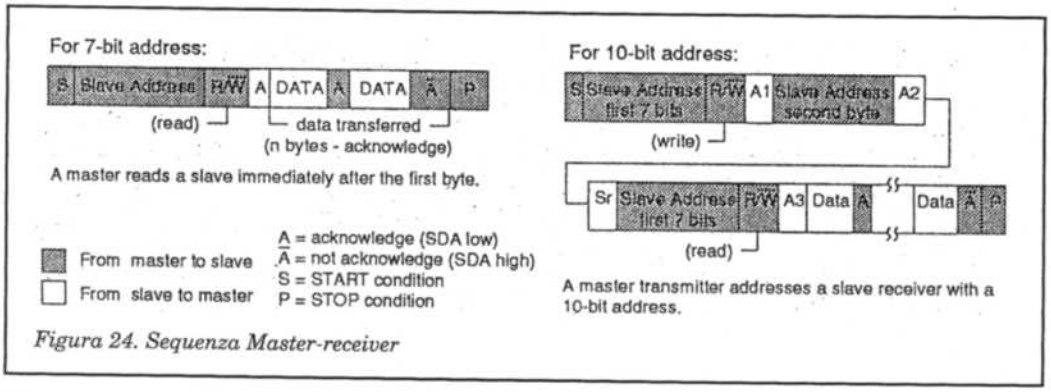


Figura 24. Sequenza Master-receiver

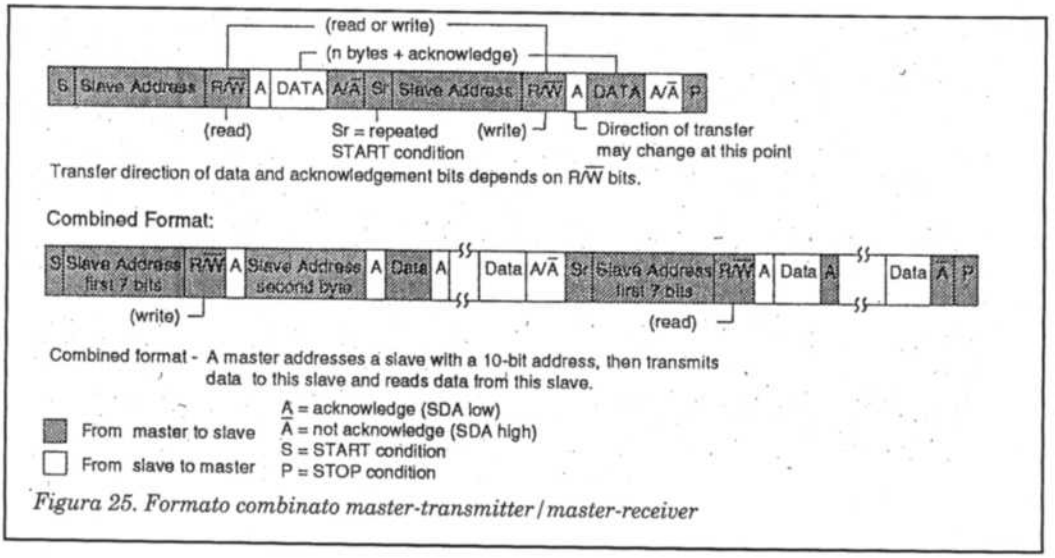


Figura 25. Formato combinato master-transmitter/master-receiver

Multimaster

Il protocollo IIC-bus permette, come abbiamo già detto, di connettere più unità tra di loro, chiaramente comunicanti con lo stesso protocollo. Oltre a un certo numero di slave, è possibile connettere anche più master, ovviamente attivandone uno per volta con tale funzione. Tale implementazione viene definita multi-master. Quando due o più master tentano di entrare in possesso del bus allo stesso tempo, necessita una fase di arbitrazione e una di sincronizzazione. La prima fase, avviene sulla linea dei dati SDA, mentre la linea del clock è alta.

Tra i due master in antagonismo, vince questa fase quello che tiene bassa la linea SDA, mentre l'altro porta in three-state il proprio stadio di uscita.

Un master che perde l'arbitrazione, può generare degli impulsi di clock fino al termine del byte di dato dove ha perso l'arbitrazione stessa. Quando i due master indirizzano la stessa device, la fase di arbitrazione prosegue sul byte dei dati.

I master che hanno implementata anche la funzione di slave e che hanno perso l'arbitrazione, devono immediatamente porsi nello stato receiver-slave affinché il master vincitore possa eventualmente indirizzarli come slave. In alcuni casi, la fase di arbitrazione non è consentita, e cioè quando sta tra condizioni di START ripetuto, oppure tra condizioni di STOP, oppure tra una condizione di START ripetuta ed una condizione di stop.

La sincronizzazione del clock, invece, avviene dopo che la fase di arbitrazione è iniziata. In Figura 27 si può vedere tale fase.

La periferica SSP-IIC

E, finalmente, vediamo come funziona la periferica interna al PIC che implementa la comunicazione con il bus IIC. Per prima cosa diciamo che tale

Tabella 10. Registri associati alle operazioni IIC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0C	PIR1	PSPIF	–	–	–	SSPIF	CCP1IF	TMR2IF	TMR1IF
8C	PIE1	PSPIE	–	–	–	SSPIE	CCP1IE	TMR2IE	TMR1IE
13	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
93	SSPADD	Synchronous Serial Port (I^2C mode) Address Register							
14	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
94	SSPSTAT	–	–	D/A	P	S	R/W	UA	BF

Legend – = Unimplemented locations, Read as '0' - Note: Shaded boxes are not used by the SSP module in I^2C mode.

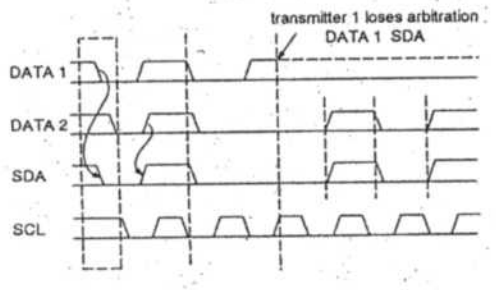


Figura 26. Arbitrazione multi-master con due master

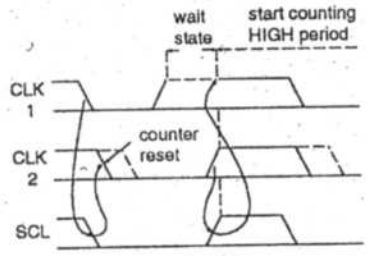


Figura 27. Sincronizzazione del clock

periferica è stata realizzata tenendo conto di tutte le caratteristiche necessarie ad una device di tipo slave, mentre per ovvie ragioni (diversità di impiego da programma a programma) nella funzionalità master offre supporto hardware per lo sviluppo di rapidi comandi software per una veloce gestione del modulo.

La periferica SSP-IIC offre sia lo standard a 100 kHz sia quello a 400 kHz (fast mode) come pure sia il formato d'indirizzamento a 7 ed a 10 bit.

In Figura 28 ne vediamo il diagramma a blocchi. Due pin sono impiegati per il trasferimento dei dati (SCL e SDA) e devono essere configurati in input o output dal programmatore attraverso i bit corrispondenti nel registro di configurazione della porta dedicata. I registri utilizzati in questo modulo sono 5: SSPCON (SSP CONtrol register), SSPSTAT (SSP STATus register), SSPBUF (Serial receive transmit BUFfer register), SSPSR (SSP Shift Register) e SSPADD (SSP ADDress register).Il registro SSPCON consente il controllo delle operazioni permettendo cinque tipi di funzionamento: IIC Slave mode con 7 bit address, IIC Slave mode con 10 bit address, IIC Slave mode con 7 bit address + supporto per Master mode, IIC Slave mode con 10 bit address + supporto per Master mode e IIC Master mode. Il registro SSPSTAT mostra lo stato del trasferimento dei dati. Sono comprese informazioni tipo il riconoscimento di una condizione di START o di STOP, la discriminazione se un byte è di indirizzamento o di dato, se il bit successivo è l'ultimo dei 10 bit di indirizzo (solo nel caso di indirizzamento a 10 bit) e se il dato successivo andrà letto o scritto. SSPSTAT è di sola lettura.

Il registro SSPBUF è il registro di scambio tra il modulo della periferica ed il bus dati del PIC. Il registro SSPSR invece è quello di scambio tra SSPBUF e la linea IIC-bus. Da notare che tali registri sono connessi in modo tale che, mentre uno sta dialogando con il bus IIC, l'altro può benissimo dialogare con il bus dati del PIC, permettendo ad esempio la ricezione di un secondo byte quando ancora si sta manipolando il primo. Infine il registro SSPADD mantiene l'indirizzo del bus IIC.

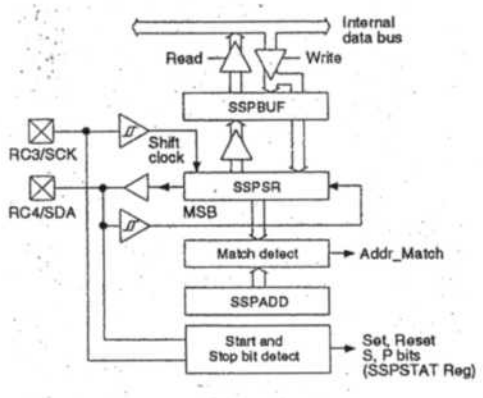


Figura 28. Diagramma a blocchi del modulo SSP-IIC

Modalità SLAVE

Nel funzionamento come slave, quando il modulo viene indirizzato correttamente, automaticamente viene generato un impulso di acknowledge e contemporaneamente il registro SSPBUF viene caricato con il contenuto del registro SSPSR (dato appena giunto).

Ciò avviene se tutto va per il meglio, ma ci sono alcune condizioni che portano ad altri risultati: se ad esempio o il bit BF (Buffer Full) o il bit SSPOV (SSP OVerflow) erano settati prima dell'arrivo del nuovo byte, allora l'ACK non sarà generato ed il nuovo valore non sarà trasferito nell'SSPBUF, ma verrà settato il bit SSPIF (SSP Interrupt Flag). Nella Tabella 9, vediamo le azioni possibili a seconda delle condizioni di lavoro.

Una volta che il modulo SSP è stato abilitato in modo slave, aspetta una condizione di START.

Successivamente, riceve il primo byte in SSPSR. I sette bit più significativi vengono comparati con il contenuto del registro SSPADD e se viene riscontrata l'uguaglianza SSPSR viene copiato in SSPBUF (se i bit BF e SSPOV lo permettono), viene settato il bit BF (Buffer Full), viene generato l'im-

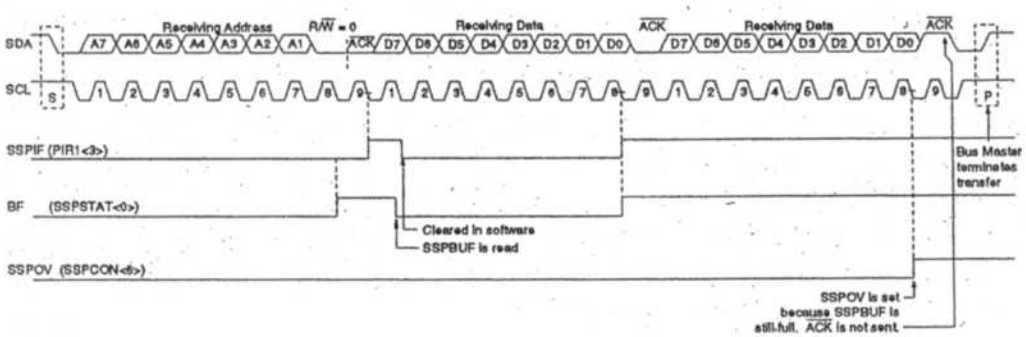


Figura 29. Ricezione IIC-bus a 7 bit

pulso di acknowledge ed il flag SSPIF viene settato. Se poi l'interrupt di tale periferica è stato abilitato, si ha anche una chiamata al vettore degli interrupt.

Il riconoscimento d'indirizzo è un po' più lungo se si lavora con 10 bit: il modulo riceve dapprima il byte con la parte alta dell'indirizzo e, se la comparazione con SSPADD dà esito positivo, in SSPADD deve essere inserita la parte bassa dell'indirizzo che giungerà con il secondo byte per poi procedere come nel caso dei 7 bit. Quando la periferica viene correttamente indirizzata, il bit R/W indica se è richiesta una ricezione oppure una trasmissione. Se tale bit vale 0, allora è richiesta una ricezione (lettura) e l'indirizzo viene passato in SSPBUF. Quando avviene una condizione di overflow dell'indirizzo, non viene generato alcun acnowledge e tale condizione viene evidenziata settando o il bit BF (Buffer Full) o il bit SSPOV (SSP OVerflow). Un interrupt viene generato a ogni trasferimento di byte settando il bit SSPIF, che dovrà poi essere azzerato dal software. In Figura 29 troviamo la sequenza di una ricezione a 7bit. In Figura 30, invece, abbiamo la trasmissione a 7 bit, che avviene quando il bit R/W vale 1.

Il byte ricevuto viene depositato in SSPBUF mentre l'impulso di acknowledge viene inviato sul nono bit e la linea SCL viene mantenuta bassa (blocco del master per attesa risposta).

Il dato da trasmettere deve essere collocato in SSPBUF, poi si deve settare il bit CKP di SSPCON in modo da liberare l'attesa sul master.

Anche in questo caso, un interrupt viene generato ad ogni trasferimento di byte settando il bit SSPIF. Egualmente andrà azzerato via software.

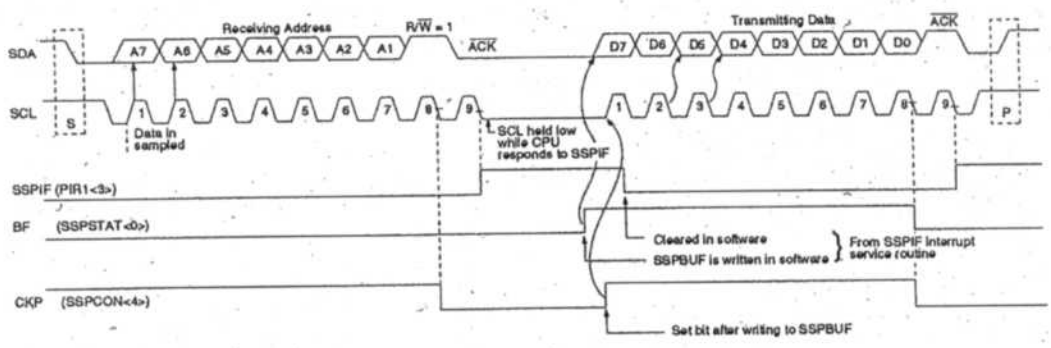


Figura 30. Trasmissione IIC-bus a 7 bit

```
IDLE_MODE (7-bit):
if (Addr_match)              {   Set interrupt;
                                 if (R/W = 1)  {Send ACK = 0;
                                                set XMIT_MODE;
                                                }
                                 else if (R/W = 0) set RCV_MODE;
                              }

RCV_MODE:
if ((SSPBUF=Full) OR (SSPOV = 1))
      {Set SSPOV;
      Do not acknowledge;
      }
else  { transfer SSPSR → SSPBUF;
      send ACK = 0;
      }
Receive 8-bits in SSPSR;
Set interrupt;

XMIT_MODE:
While ((SSPBUF = Empty) AND (CKP=0)) Hold SCL Low;
Send byte;
Set interrupt;
if (ACK Received = 1)        {End of transmission;
                              Go back to IDLE_MODE;
                              }
else if (ACK Received = 0)   Go back to XMIT_MODE;

IDLE_MODE (10-Bit):
If (High_byte_addr_match AND (R/W = 0))
    { PRIOR_ADDR_MATCH = FALSE;
      Set interrupt;
      if ((SSPBUF = Full) OR ((SSPOV = 1))
            { Set SSPOV;
              Do not acknowledge;
            }
      else  { Set UA = 1;
              Send ACK = 0;
              While (SSPADD not updated) Hold SCL low;
              Clear UA = 0;
              Receive Low_addr_byte;
              Set interrupt;
              Set UA = 1;
              If (Low_byte_addr_match)
                    { PRIOR_ADDR_MATCH = TRUE;
                      Send ACK = 0;
                      while (SSPADD not updated) Hold SCL low;
                      Clear UA = 0;
                      Set RCV_MODE;
                    }
            }
    }
else if (High_byte_addr_match AND (R/W = 1)
            { if (PRIOR_ADDR_MATCH)
                    { send ACK = 0;
                      set XMIT_MODE;
                    }
              else PRIOR_ADDR_MATCH = FALSE;
```

Figura 31. Operazioni del modulo IIC

Modalità MASTER

La modalità master viene supportata dalla generazione di un interrupt ad ogni condizione di START o STOP. Il bit di START (S) e il bit di STOP (P) sono azzerati da un reset o quando il modulo SSP viene disabilitato. Il possesso del bus può essere preso quando il bit P è settato oppure quando il bus è in idle e sia il bit P che il bit S sono azzerati. Le linee SCL e SDA, in modalità master, i corrispondenti pin di uscita sono sempre a basso livello come valore sulla porta e per modificarli in 1 si deve impostare a 1 (ricezione) il bit corrispondente alla direzione di quel pin. È stato realizzato un sistema che, quando vale 1, impedisce di far scorrere corrente attraverso un'eventuale linea tenuta a basso livello. L'interrupt viene generato dalle seguenti condizioni: START, STOP, byte trasferito.

Modalità MULTI-MASTER

Il riconoscimento di una condizione di START e di STOP ci consente di determinare se il bus è libero oppure no. I bit S e P vengono gestiti come nella modalità master. Quando il bus non è libero, abilitare il bit SSPIF genererà un interrupt alla prima condizione di STOP rilevata. La linea SDA deve essere monitorata per vedere se il livello del segnale è quello che ci aspettiamo ed in caso contrario il bus deve essere rilasciato. In Figura 31 troviamo le operazioni del modulo IIC. Quando è attivata la modalità slave, il modulo continua a ricevere. Se l'arbitrazione è stata persa durante il trasferimento dell'indirizzo, il modulo potrebbe essere indirizzato. Se ciò accade, un impulso di acknowledge verrà generato. Se, invece, l'arbitrazione è stata persa durante la fase di trasferimento di un dato, la device dovrà ritrasmettere il dato più tardi.

continua

LE PERIFERICHE DEI PIC: FACCIAMONE CONOSCENZA

È ormai diventato un appuntamento fisso con i microcontrollori della famiglia dei Pic: questa serie sulle periferiche è senza dubbio interessante e aiuta ad entrare in questo mondo dei microcontrollori

Paolo Sbrana - 5ª parte

Dopo aver ampiamente trattato il modulo SSP, andremo adesso a scoprire l'altro modulo seriale presente in alcuni tipi di PIC e cioè il modulo SCI (Serial Communication Interface).

Sappiamo che con il modulo visto precedentemente è possibile comunicare con altre device che supportano protocolli seriali sincroni standard IIC e SPI, ma non è permesso dialogare ad esempio con la seriale RS232, poiché tale standard richiede un protocollo asincrono.

Il modulo SCI è stato implementato nei PIC proprio per consentire anche questo tipo di collegamento.

Le modalità di funzionamento che tale periferica supporta sono la asincrona full-duplex, la sincrona-master half-duplex e la sincrona-slave half-duplex.

La possibilità di funzionamento full-duplex deriva dal fatto che sono contemporaneamente presenti due sezioni, una per la trasmissione, una per la ricezione con un unico componente in comune: il generatore di baud rate.

In Figura 32 e 33 troviamo i rispettivi diagrammi a blocch che permettono di comprenderne il principio di funzionamento.

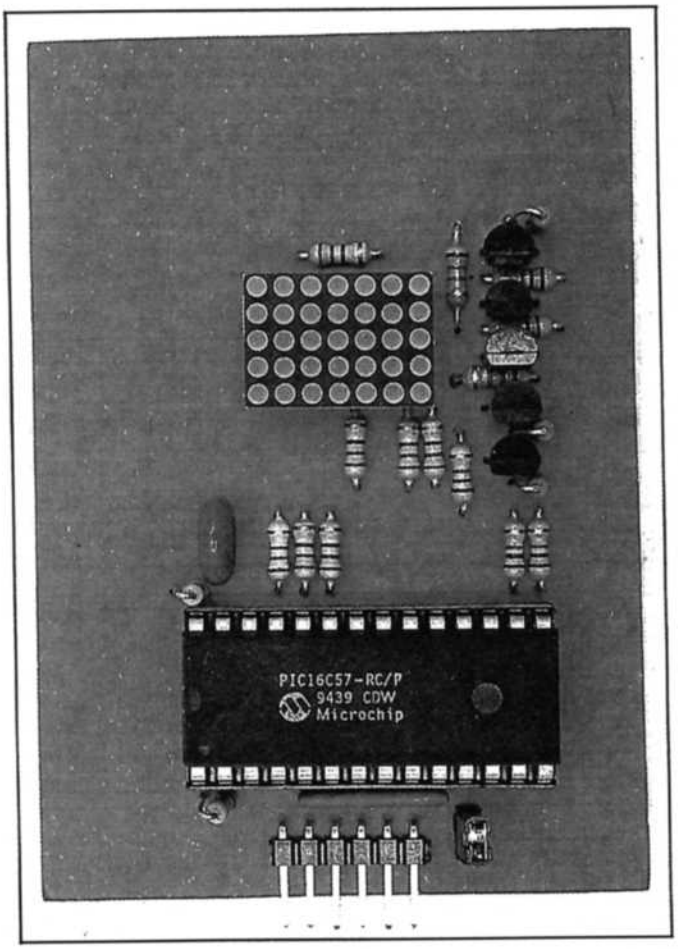
La sezione trasmittente

Dalla Figura 32 vediamo che il registro che si occupa di bufferizzare il dato da inviare è detto TXREG e si interfaccia direttamente con il bus dati del PIC. Il byte inserito in TXREG viene copiato in un altro registro detto TSR (Transmit Shift Register) che si occuperà poi di posizionare un bit per volta sul pin di uscita. Notiamo però che il TSR è lungo 9 bit, uno cioè in più del registro TXREG.

Questo perché è possibile selezionare via software se vogliamo dialogare anche con un bit di parità oltre gli otto dei dati.

Poiché però questo bit non è acquisibile direttamente da TXREG (che è di soli 8 bit), dovrà essere settato con una istruzione software.

Il registro associato alla sezione trasmittente è il TXSTA (TX STAtus) il cui significato si trova in Figura 34.



Il bit 0 (TXD8) è proprio il bit da settare nel caso in cui si sia scelto di inviare anche la parità.

Attenzione però che la parità deve essere calcolata via software e poi il risultato deve essere qui inserito.

Da non confondere con il bit 6 (TX8/9) che vedremo in seguito.

Il bit 1 (TRMT) è di sola lettura e avvisa se il registro TSR (Transmit Shift Register) è ancora pieno o no, ovvero se la trasmissione del dato è terminata oppure no.

Il bit 2 (BRGH) serve per selezionare due modi di velocità del baud rate solo però nella modalità asincrona.

Il bit 3 non è stato ancora impiegato.

Il bit 4 (SYNC), invece, indica la scelta tra la modalità sincrona e quella asincrona.

Il bit 5 (TXEN) abilita o meno la trasmissione di un dato.

Il bit 6 (TX8/9) serve per dire al modulo se vogliamo che venga spedito anche il

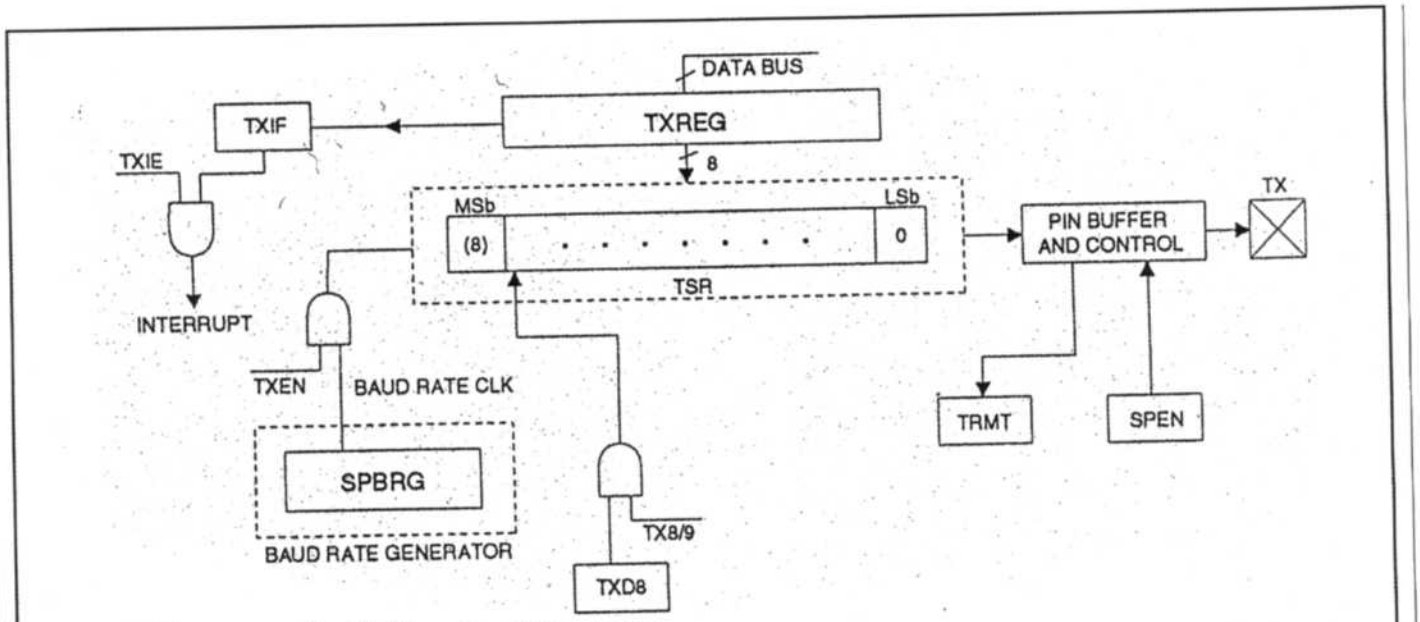


Figura 32. Diagramma a blocchi della sezione di trasmissione

bit di parità oppure no. Attenzione, perché in questo bit non deve essere inserito il valore di parità, ma serve soltanto per la selezione del modo operativo.

Infine, il bit 7 (CSRC) indica se si è scelta la modalità master oppure la slave, ma soltanto quando si è in modo sincrono.

La sezione ricevente

In Figura 33 troviamo il diagramma a blocchi della sezione ricevente. Appare leggermente più complesso dell'altro, ma fondamentalmente sono specularmente simili.

Questa volta, il dato non deve essere fornito al modulo, ma prelevato dal modulo stesso.

Per questa importante operazione, sul bus dati si interfaccia il registro RCREG.

Una particolarità di questo registro, è che il programmatore lo vedrà sempre come un unico registro, mentre in realtà sono due connessi come una memoria FIFO (First In First Out).

Questo ci consente di avviare una seconda ricezione, anche se il dato relativo alla prima non è ancora stato prelevato.

Inoltre, RCREG è a nove bit, quindi restituisce anche l'eventuale bit di parità.

Il registro associato alla sezione ricevente è il RXSTA (RX STAtus) il cui significato si trova in Figura 35.

- Il bit 0 (RCD8) è proprio il bit da leggere nel caso in cui si sia scelto di sfruttare anche la parità.
- Il bit 1 (OERR) ci dice se è avvenuto un errore di overrun.
- Il bit 2 (FERR) invece ci informa se si è verificato un errore di framing.
- Il bit 3 non è ancora stato impiegato.
- Il bit 4 (CREN) è quello che serve ad abilitare o meno il modulo alla ricezione continua.
- Il bit 5 (SREN) invece abilita o meno il modulo alla ricezione ma solo in modalità sincrona.
- Il bit 6 (RC8/9) inoltre serve per segnalare al modulo se vogliamo che venga ricevuto anche il bit di parità oppure no.
- Il bit 7, infine, è realmente molto importante perché permette di configurare i pin relativi a questa periferica come pin di trasmissione e pin di ricezione.

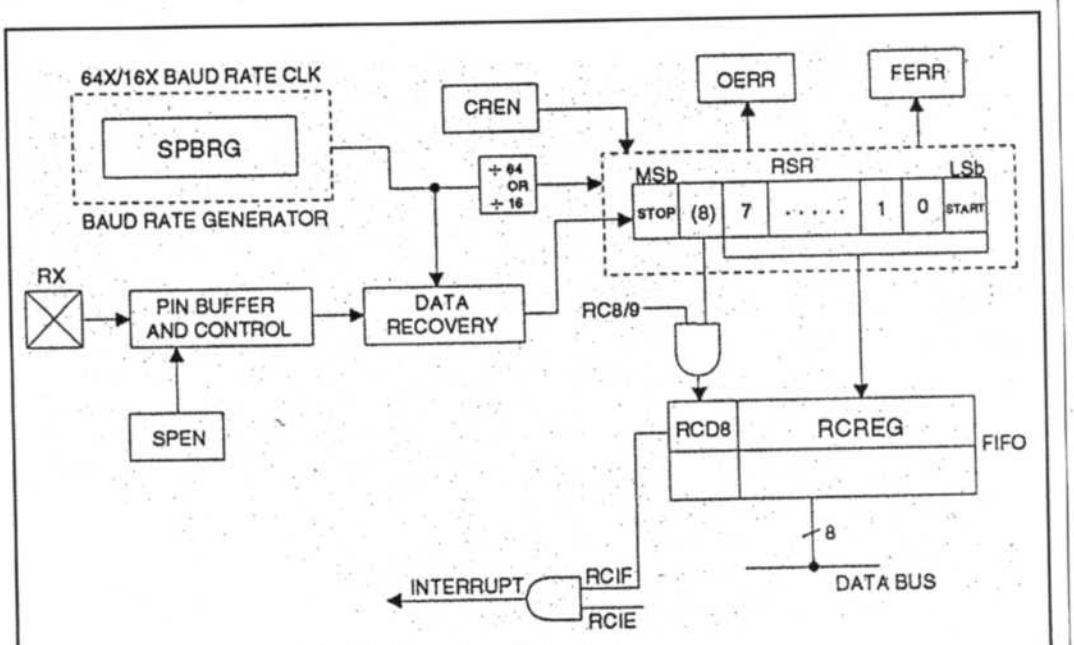


Figura 33. Diagramma a blocchi della sezione di ricezione

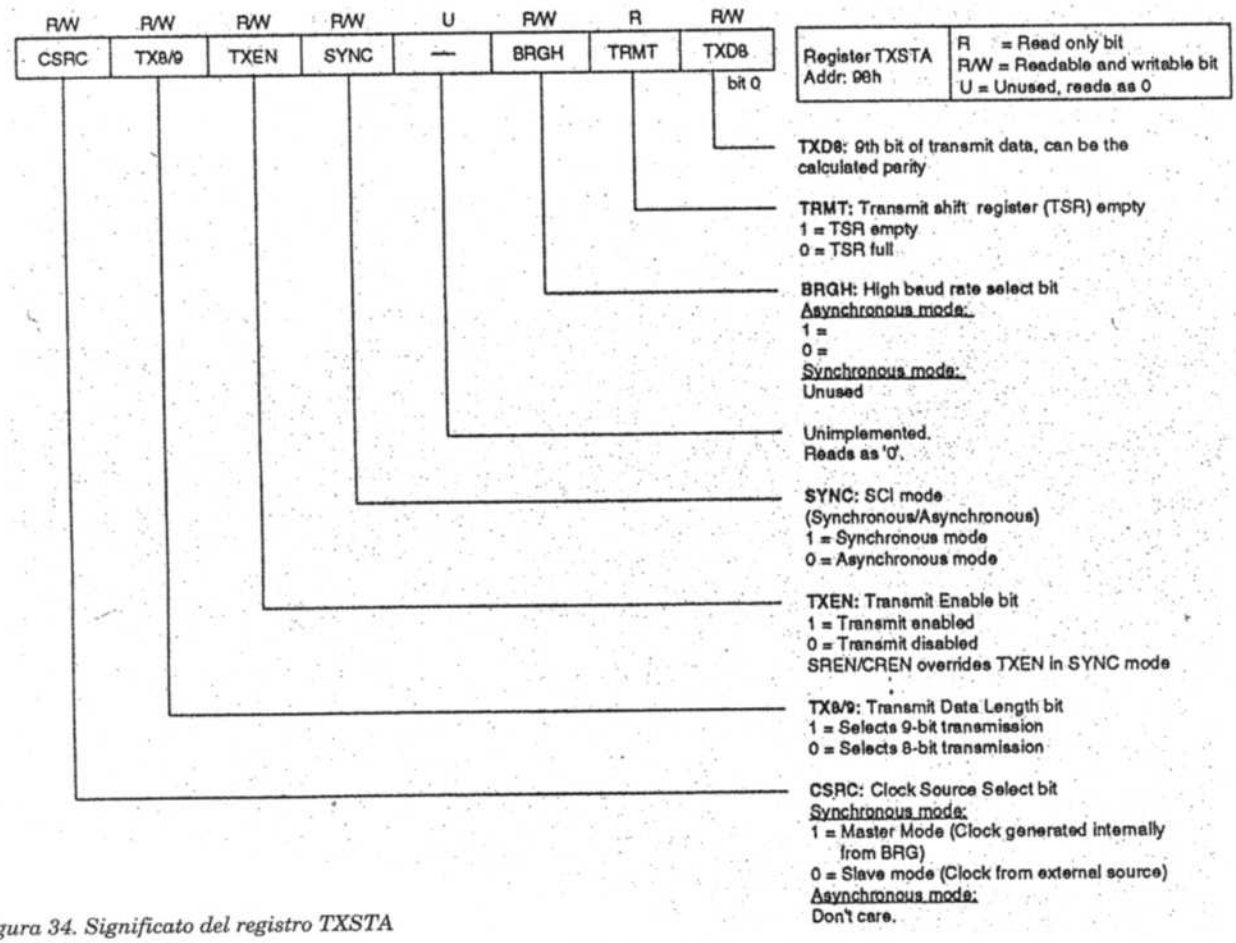


Figura 34. Significato del registro TXSTA

Il baud generator

Il baud rate generator è l'unico componente in comune alle due sezioni e serve sia l'operatività sincrona che quella asincrona.

È costituito da un registro ad 8 bit (SPBRG) e da un divisore. Per calcolare il corretto valore da inserire nel registro SPREG e i settaggi dei bit SYNC e BRGH del registro TXSTA ci sono tre semplici formule.

La prima e la seconda, implicano la modalità di funzionamento asincrona e quindi impongono di porre a zero il bit SYNC di TXSTA. Poi, se si sceglie di porre a zero il bit BRGH, la formula è data da:

Baud rate = Fosc / 64 x (X+1)

mentre se si sceglie di settare il bit BRGH a 1, la formula sarà:

Baud rate = Fosc / 16 x (X+1).

Nel caso in cui sia necessario lavorare in modalità sincrona e quindi settando a 1 il bit SYNC, la formula sarà:

Baud rate = Fosc / 4 x (X+1).

La X è il numero compreso tra 0 e 255 da inserire nel registro SPBRG.

Per facilitare i calcoli, la Microchip offre delle tabelle già pronte per le frequenze di clock più comuni e per i baud rate più impiegati.

Nelle Tabelle 11, 12a e 12b sono rispettivamente visibili i buad rate per la modalità sincrona, per la modalità asincrona con il bit BRGH=0 e per la modalità asincrona con il bit BRGH=1.

Il dato sul piedino di ricezione RX è campionato tre volte per avere un ottimo margine di sicurezza. Se il bit BRGH vale 1, tale margine è superiore, come si vede dalle Figure 36, 37a e 37b.

Modalità asincrona

La modalità asincrona viene selezionata azzerando il bit SYNC del registro TXSTA.

Vediamo la trasmissione: il cuore della trasmissione è il registro TSR, che si carica con il dato trovato in TXREG e invia uno ad uno i bit sul pin di uscita. Il registro TXREG deve essere caricato via software dal programmatore.

Il TSR non può essere ricaricato fino a quando non ha finito di inviare l'ultimo bit relativo alla trasmissione precedente.

Quando ciò avviene, il TXREG si svuota caricando il TSR con il nuovo dato e viene settato un bit di interrupt (TXIF).

Tale interrupt può essere abilitato o meno, ma il bit TXIF non è resettabile via software, ma solo quando il registro TXREG è nuovamente vuoto.

Per verificare, invece, quando è vuoto il TSR, si deve testare il bit TMRT poiché il registro TSR non è mappato in memoria e quindi non può essere né letto né scritto.

La trasmissione viene abilitata portando a 1 il bit TXEN del registro TXSTA.

Azzerare il bit TXEN durante una trasmissione ne causerà l'interruzione.

Per abilitare l'invio del nono bit per la parità, si deve settare il bit TX8/9.

Il nono bit poi deve essere scritto in TXD8 prima che venga inserito il dato da inviare in TXREG.

Questo perché, a seconda dello stato, la scrittura di un byte in TXREG potrebbe significare subito l'inizio della trasmissione, ad esempio se TSR è vuoto e TXEN è settato. I passi da seguire per eseguire una corretta trasmissione sono i seguenti:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale asincrona configurando i bit SYNC=0 e SPEN=1;
- Azzerare i bit CREN e SREN;
- Se si desidera abilitare l'interrupt, settare il bit TXIE;
- Se si desidera il bit di parità, settare il bit TX8/9;
- Abilitare la trasmissione settando il bit TXEN;
- Se in modo 9 bit, caricare il bit TXD8;
- Caricare il dato da inviare in TXREG.

In Figura 38a e 38b vediamo i timing diagram, rispettivamente della trasmissione di un byte e di due byte consecutivi.

Passiamo adesso alla ricezione asincrona

Il cuore di questa sezione è il registro RSR (Receive Shift Register). Dopo il campionamento di uno stop bit, il dato ricevuto viene trasferito nel registro RCREG (se è vuoto) e viene settato il bit RCIF relativo all'interrupt della ricezione.

Se poi il bit RCIE era stato settato, si avrà la generazione di un interrupt.

Il bit RCIF è resettabile solo dall'hardware nel momento in cui il dato viene letto dal registro RCREG.

Come abbiamo già accennato, tale registro è composto in realtà da due registri a 9 bit.

È possibile, quindi, ricevere due dati consecutivi in RCREG mentre un terzo è in arrivo in RSR.

All'arrivo dello stop bit del terzo dato, se ancora i due dati precedenti non sono stati letti verrà settato il bit OERR di overrun ed il terzo dato verrà perso irrimediabilmente.

Allora il bit OERR deve essere azzerato dal programmatore resettando il bit CREN.

Se durante una ricezione, il bit di stop viene letto a livello basso, allora viene settato il bit FERR.

Poiché leggere il registro RCREG significa leggere anche i bit RCD8 e FERR con i nuovi valori, è fondamentale per il programmatore leggere dapprima il registro RCSTA, poi il registro

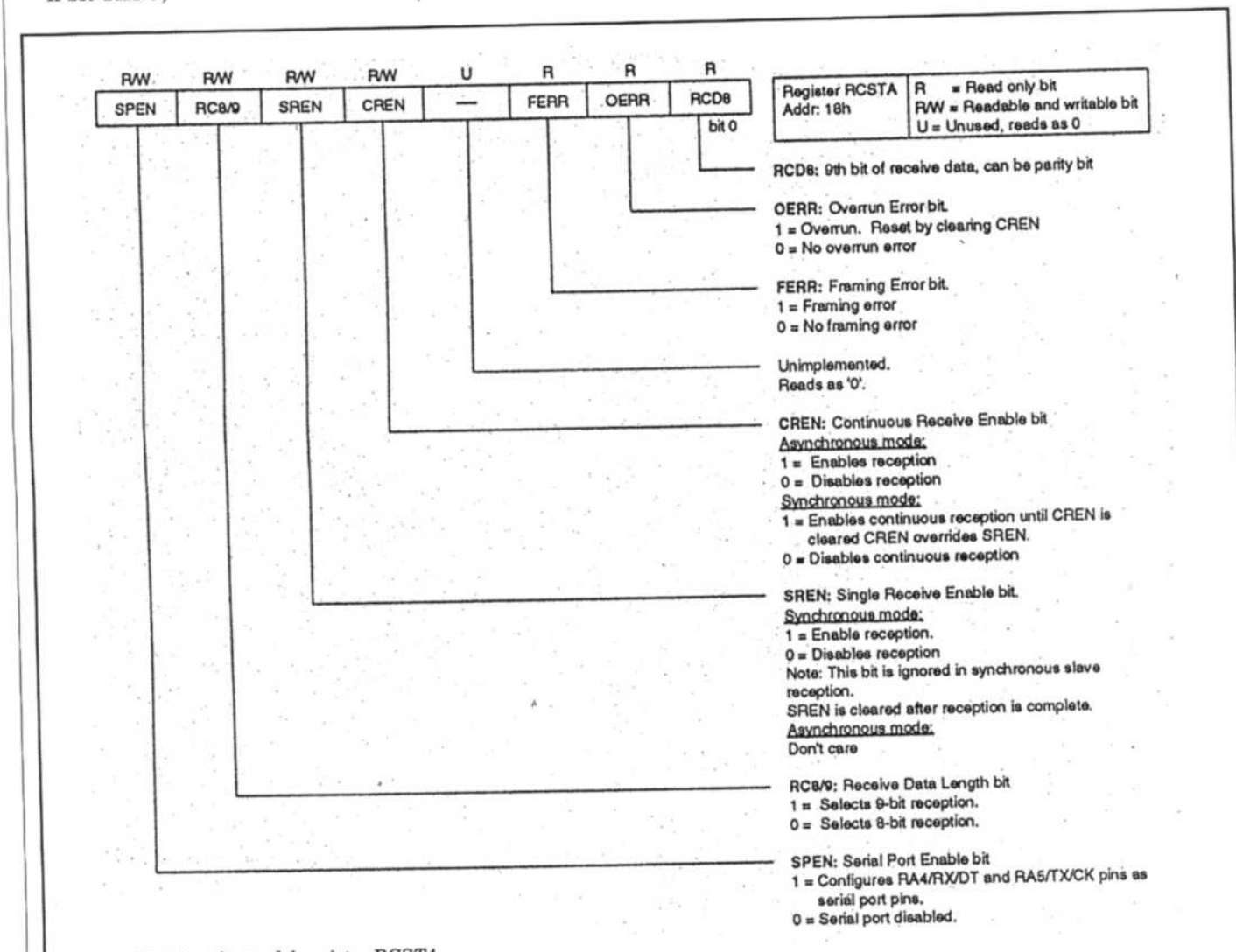


Figura 35. Significato del registro RCSTA

Tabella 11. Baud rate in modalità sincrona

BAUD RATE (K)	fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)
	KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR	
0.3	NA	–	–	NA	–	–	NA	–	–	NA	–	–
1.2	NA	–	–	NA	–	–	NA	–	–	NA	–	–
2.4	NA	–	–	NA	–	–	NA	–	–	NA	–	–
9.6	NA	–	–	NA	–	–	9.766	+1.73	255	9.622	+0.23	185
19.2	19.53	+1.73	255	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92
76.8	76.92	+0.16	64	76.92	+0.16	51	75.76	–1.36	32	77.82	+1.32	22
96	96.15	+0.16	51	95.24	–0.79	41	96.15	+0.16	25	94.20	–1.88	18
300	294.1	–1.96	16	307.69	+2.56	12	312.5	+4.17	7	298.3	–0.57	5
500	500	0	9	500	0	7	500	0	4	NA	–	–
HIGH	5000	–	0	4000	–	0	2500	–	0	1789.8	–	0
LOW	19.53	–	255	15.625	–	255	9.766	–	255	6.991	–	255

BAUD RATE (K)	fosc = 5.0688 MHz		SPBRG value (decimal)	3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR	
0.3	NA	–	–	NA	–	–	NA	–	–	0.303	+1.14	26
1.2	NA	–	–	NA	–	–	1.202	+0.16	207	1.170	–2.46	6
2.4	NA	–	–	NA	–	–	2.404	+0.16	103	NA	–	–
9.6	9.6	0	131	9.622	+0.23	92	9.615	+0.16	25	NA	–	–
19.2	19.2	0	65	19.04	–0.83	46	19.24	+0.16	12	NA	–	–
76.8	79.2	+3.13	15	74.57	–2.90	11	83.34	+8.51	2	NA	–	–
96	97.48	+1.54	12	99.43	+3.57	8	NA	–	–	NA	–	–
300	316.8	+5.60	3	298.3	–0.57	2	NA	–	–	NA	–	–
500	NA	–	–	NA	–	–	NA	–	–	NA	–	–
HIGH	1267	–	0	894.9	–	0	250	–	0	8.192	–	0
LOW	4.950	–	255	3.496	–	255	0.9766	–	255	0.032	–	255

RCREG, per non perdere quelle due informazioni.

I passi da seguire per una corretta ricezione sono i seguenti. Chiariamo che la loro sequenza è rigorosa e va seguita attentamente:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale asincrona configurando i bit SYNC=0 e SPEN=1;
- Se si desidera abilitare l'interrupt, settare il bit RXIE;
- Se si desidera il bit di parità, settare il bit RX8/9;
- Abilitare la ricezione settando il bit CREN;
- Quando la ricezione è terminata, il bit RCIF viene settato e, se abilitato l'interrupt di ricezione, viene generato un interrupt;
- Leggere il registro RCSTA per l'eventuale non bit e gli eventuali errori;
- Leggere il dato da presente in RCREG;
- Se ci sono stati errori, azzerarli azzerando il bit CREN per poi riabilitarlo.

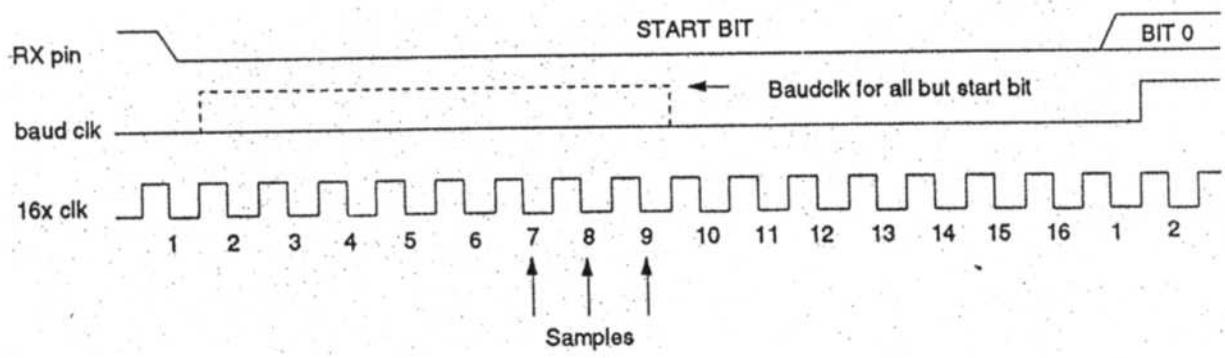


Figura 36. Campionamento sul pin RX con BRGH = 0

Tabella 12a. Buad rate in modalità asincrona con BRGH=0

BAUD RATE (K)	fosc = 20 MHz KBAUD	%ERROR	SPBRG value (decimal)	16 MHz KBAUD	%ERROR	SPBRG value (decimal)	10 MHz KBAUD	%ERROR	SPBRG value (decimal)	7.15909 MHz KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	–	–	NA	–	–	NA	–	–	NA	–	–
1.2	1.221	+1.73	255	1.202	+0.16	207	1.202	+0.16	129	1.203	+0.23	92
2.4	2.404	+0.16	129	2.404	+0.16	103	2.404	+0.16	64	2.380	–0.83	46
9.6	9.469	–1.36	32	9.615	+0.16	25	9.766	+1.73	15	9.322	–2.90	11
19.2	19.53	+1.73	15	19.23	+0.16	12	19.53	+1.73	7	18.64	–2.90	5
76.8	78.13	+1.73	3	83.33	+8.51	2	78.13	+1.73	1	NA	–	–
96	104.2	+8.51	2	NA	–	–	NA	–	–	NA	–	–
300	312.5	+4.17	0	NA	–	–	NA	–	–	NA	–	–
500	NA	–	–	NA	–	–	NA	–	–	NA	–	–
HIGH	312.5	–	0	250	–	0	156.3	–	0	111.9	–	0
LOW	1.221	–	255	0.977	–	255	0.6104	–	255	0.437	–	255

BAUD RATE (K)	fosc = 5.0688 MHz KBAUD	%ERROR	SPBRG value (decimal)	3.579545 MHz KBAUD	%ERROR	SPBRG value (decimal)	1 MHz KBAUD	%ERROR	SPBRG value (decimal)	32.768 kHz KBAUD	%ERROR	SPBRG value (decimal)
0.3	0.31	+3.13	255	0.301	+0.23	185	0.300	+0.16	51	0.256	–14.67	1
1.2	1.2	0	65	1.190	–0.83	46	1.202	+0.16	12	NA	–	–
2.4	2.4	0	32	2.432	+1.32	22	2.232	–6.99	6	NA	–	–
9.6	9.9	+3.13	7	9.322	–2.90	5	NA	–	–	NA	–	–
19.2	19.8	+3.13	3	18.64	–2.90	2	NA	–	–	NA	–	–
76.8	79.2	+3.13	0	NA	–	–	NA	–	–	NA	–	–
96	NA	–	–	NA	–	–	NA	–	–	NA	–	–
300	NA	–	–	NA	–	–	NA	–	–	NA	–	–
500	NA	–	–	NA	–	–	NA	–	–	NA	–	–
HIGH	79.2	–	0	55.93	–	0	15.63	–	0	0.512	–	0
LOW	0.3094	–	255	0.2185	–	255	0.0610	–	255	0.0020	–	255

Modalità sincrona (master)

In modalità sincrona, non è più possibile lavorare in full-duplex, ma soltanto in half-duplex, ovvero nel momento in cui stiamo ricevendo un dato non possiamo trasmetterlo e vice versa.

Per entrare in questa modalità si devono settare i bit SYNC e SPEN del registro TXSTA.

La modalità master indica che il PIC trasmette il clock alle altre unità (slave).

Per selezionare questa opzione, si deve settare il bit CSRC.

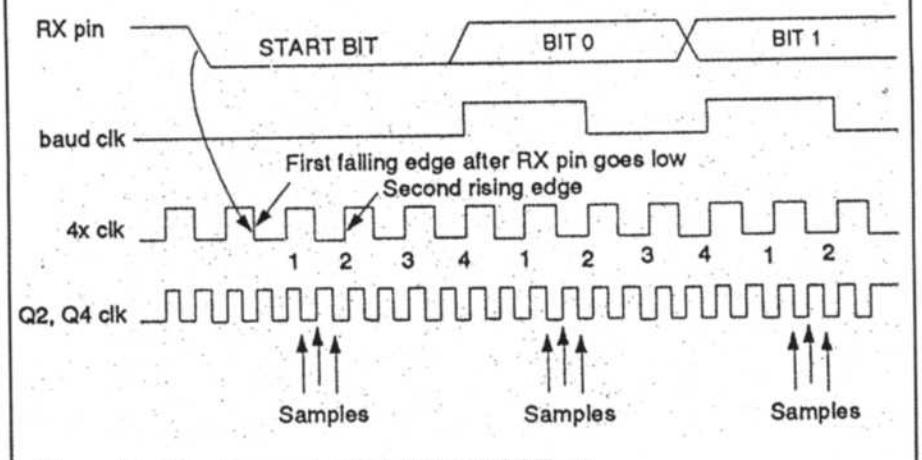


Figura 37a. Campionamento sul pin RX con BRGH = 1

La trasmissione sincrona come master

Poiché la periferica impiegata è la stessa vista per la trasmissione asincrona, il funzionamento è pressoché identico.

La differenza sostanziale è che dove prima avevamo il pin di trasmissione, adesso troviamo il pin del clock e dove prima avevamo il pin di ricezione, adesso troviamo il pin dei dati.

Per questo motivo, non sarà mai possibile dialogare nei due versi contemporaneamente.

In Figura 40a e 40b si trovano due sequenze di trasmissione sincrona.

I passi da seguire per eseguire una corretta trasmissione sincrona (master) sono i seguenti:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale sincrona configurando i bit SYNC=1, SPEN=1

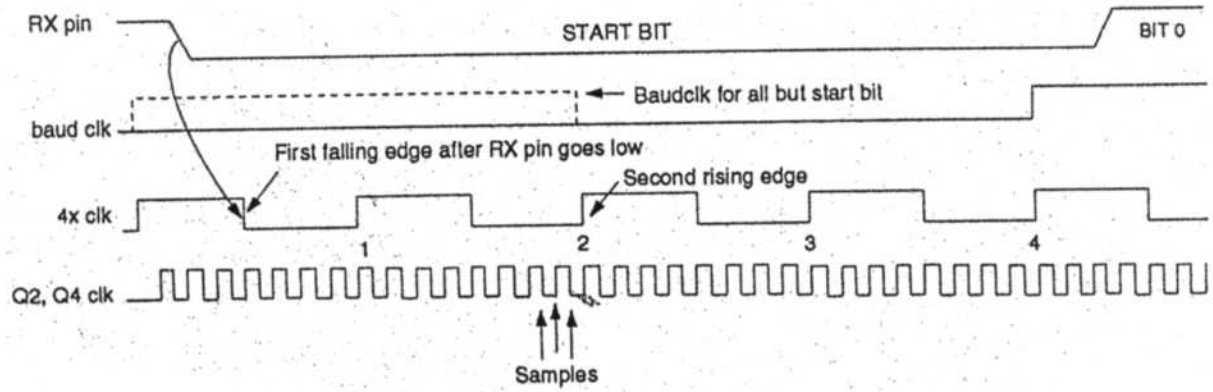


Figura 37b. Particolare del campionamento sullo start bit

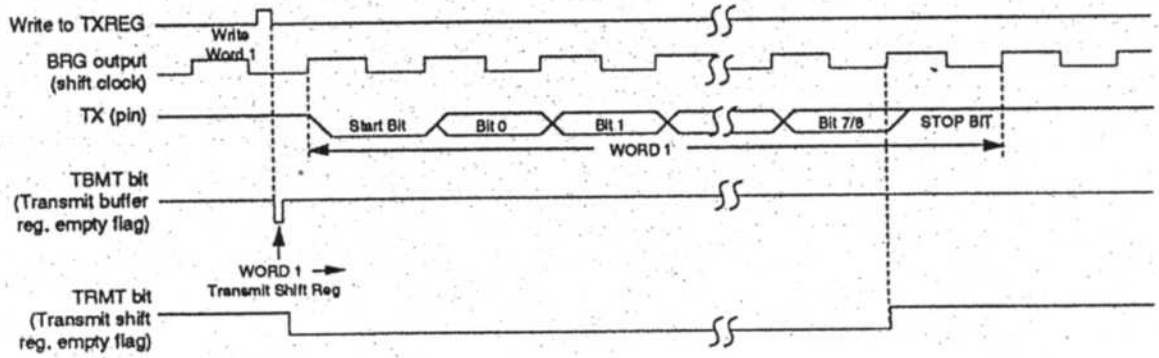


Figura 38a. Trasmissione asincrona

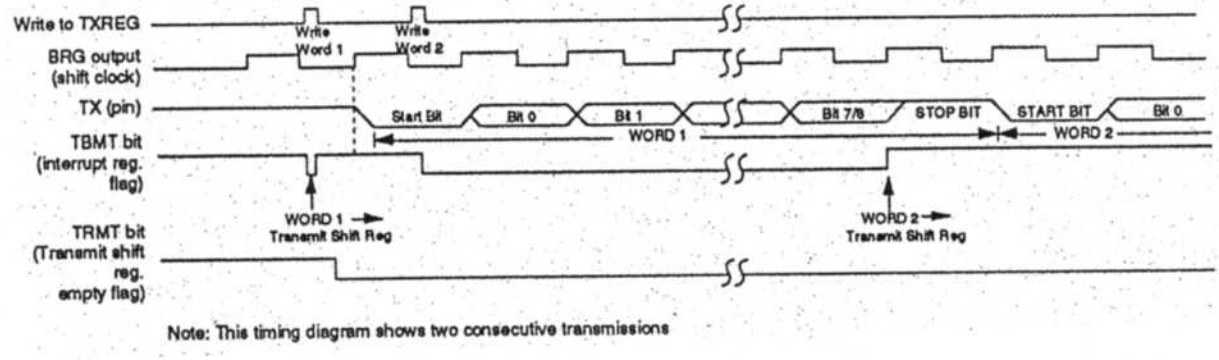


Figura 38b. Trasmissione asincrona back to back

e CSRC=1;
- Se si desidera abilitare l'interrupt, settare il bit TXIE;
- Se si desidera il bit di parità, settare il bit TX8/9;
- Abilitare la trasmissione settando il bit TXEN;
- Se in modo 9 bit, caricare il bit TXD8;
- Caricare il dato da inviare in TXREG.

La ricezione sincrona come master

La ricezione sincrona come master viene abilitata settando o il bit SREN oppure il bit CREN.

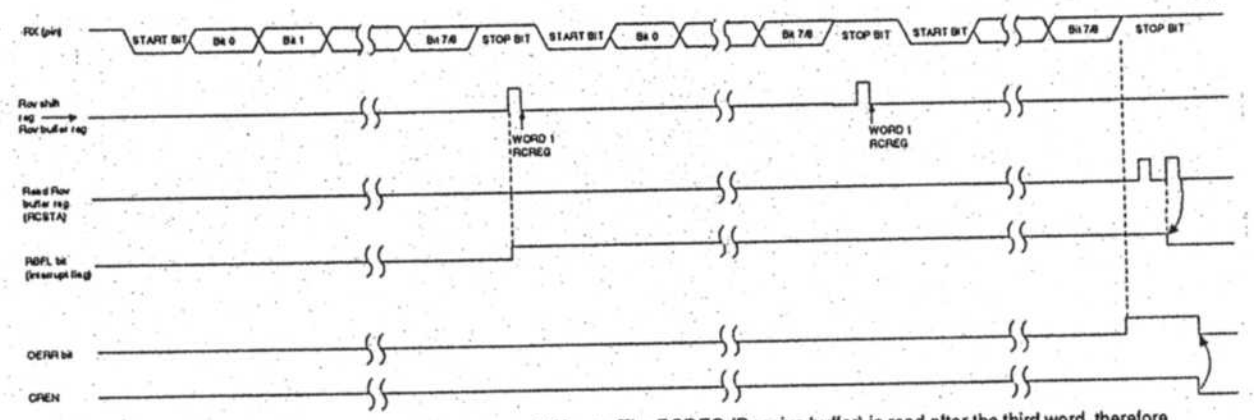
Il dato viene letto sul pin DT sul fronte di discesa del clock.

Se SREN=1, allora viene ricevuto un singolo dato, se invece CREN=1 si ha ricezione continua fino a quando CREN non venga resettato.

Se entrambi i bit sono a 1, CREN ha la priorità (SREN non viene considerato). Dopo aver ricevuto l'ultimo bit, il dato viene posto in RCREG (se vuoto).

Da questo momento in poi la sequenza delle operazioni è la stessa trovata nella ricezione asincrona.

I passi da seguire per una corretta



Note: This timing diagram shows three words appear on RX input. The RCREG (Receive buffer) is read after the third word, therefore causing the OERR (overrun) bit to set.

Figura 39. Ricezione asincrona

ricezione sincrona (master) sono i seguenti:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale asincrona configurando i bit SYNC=1, SPEN=1 e CSRC=1;
- Se si desidera abilitare l'interrupt, settare il bit RXIE;
- Se si desidera il bit di parità, settare il bit RX8/9;
- Abilitare la ricezione settando il bit CREN o il bit SREN a seconda del funzionamento prescelto;
- Quando la ricezione è terminata, il bit RCIF viene settato e, se abilitato l'interrupt di ricezione, viene generato un interrupt;
- Leggere il registro RCSTA per l'eventuale non bit e gli eventuali errori;
- Leggere il dato da RCREG;
- Se ci sono stati errori, azzerarli azzerando il bit CREN. In Figura 41 è visibile la sequenza di una ricezione.

Modalità sincrona (slave)

In questa tipologia di funzionamento, sia la trasmissione che la ricezione sono identiche alla modalità sincrona

Tabella 12b. Buad rate in modalità asincrona con BRGH=1

BAUD RATE (K)	fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.16 MHz		SPBRG value (decimal)
	KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR	
9600	9615.34	+0.16	129	9615.34	+0.16	103	9615.39	+0.16	64	9520.07	–0.83	46
19200	19230.77	+0.16	64	19230.77	+0.16	51	18939.39	–1.36	32	19454.05	+1.32	22
38400	37878.79	–1.36	32	38461.54	+0.16	25	39062.5	+1.7	15	37286.93	–2.90	11
57600	56818.18	–1.36	21	58823.53	+2.12	16	56818.18	–1.36	10	55930.39	–2.90	7
115200	113636.4	–1.36	10	111111.1	–3.55	8	125000	+8.51	4	111860.8	–2.90	3
250000	250000	0	4	250000	0	3	NA	–	–	NA	–	–
625000	625000	0	1	NA	–	–	625000	0	0	NA	–	–
1250000	1250000	0	0	NA	–	–	NA	–	–	NA	–	–

BAUD RATE (K)	fosc = 5.068 MHz		SPBRG value (decimal)	3.579 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR		KBAUD	%ERROR	
9600	9600	0	32	9727.02	+1.32	22	8928.57	–6.99	6	NA	–	–
19200	18645.29	–2.94	16	18643.46	–2.90	11	20833.33	+8.51	2	NA	–	–
38400	39600	+3.12	7	37286.93	–2.90	5	31250	–18.61	1	NA	–	–
57600	52800	–8.33	5	55930.39	–2.90	3	62500	+8.51	0	NA	–	–
115200	105600	–8.33	2	111860.8	–2.90	1	NA	–	–	NA	–	–
250000	NA	–	–	223721.6	–10.51	0	NA	–	–	NA	–	–
625000	NA	–	–	NA	–	–	NA	–	–	NA	–	–
1250000	NA	–	–	NA	–	–	NA	–	–	NA	–	–

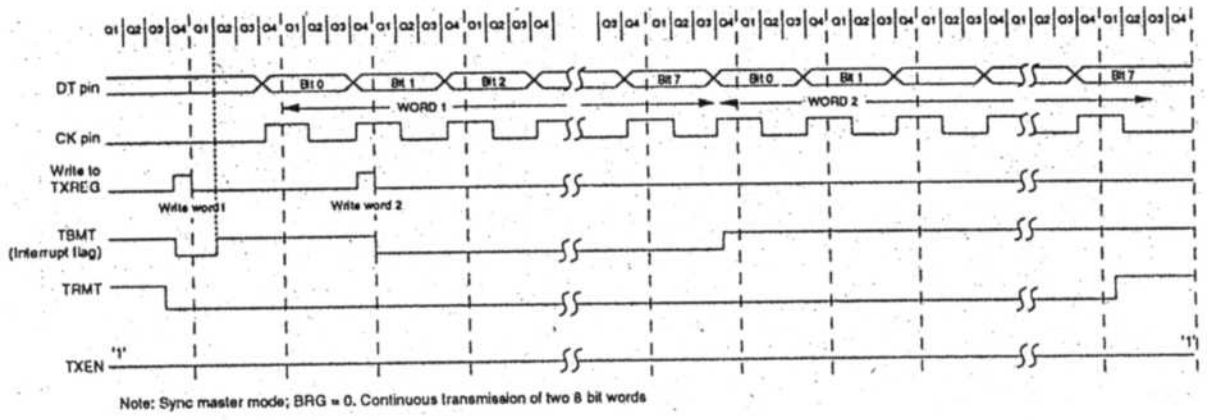


Figura 40a. Trasmissione sincrona

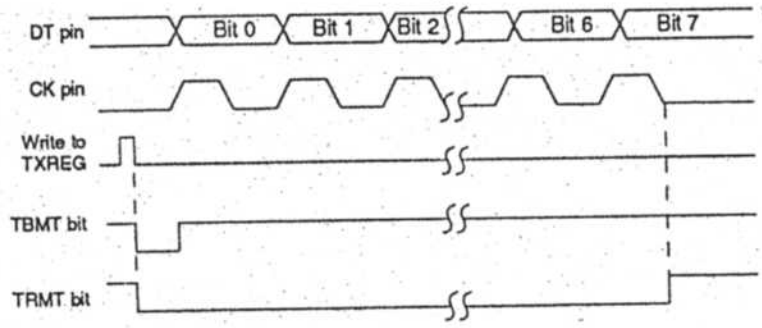


Figura 40b. Trasmissione sincrona attraverso TXEN

(master) eccetto nel caso dello sleep mode. Cerchiamo allora di capire questa importante differenza partendo dalla trasmissione. Se due dati sono stati scritti nel registro TXREG e successivamente viene eseguita l'istruzione di entrata in sleep, che cosa succede? Il primo dato viene immediatamente passato al registro TSR e trasmesso, mentre il secondo rimarrà in TXREG e il bit TXIF non sarà settato.

Quando il primo dato è stato del tutto inviato, il TXREG trasferirà il secondo dato al TSR ed il TXIF sarà così settato.

Se il bit TXIE è stato abilitato, l'interrupt sveglierà il PIC dallo stato di SLEEP e, se il GIE (Global Interrupt Enable) era abilitato, il program counter salterà all'indirizzo del vettore di interrupt, ovvero 0x004.

I passi da seguire per eseguire una corretta trasmissione sincrona (slave) sono i seguenti:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale sincrona configurando i bit SYNC=1, SPEN=1 e CSRC=0;
- Azzerare CREN;

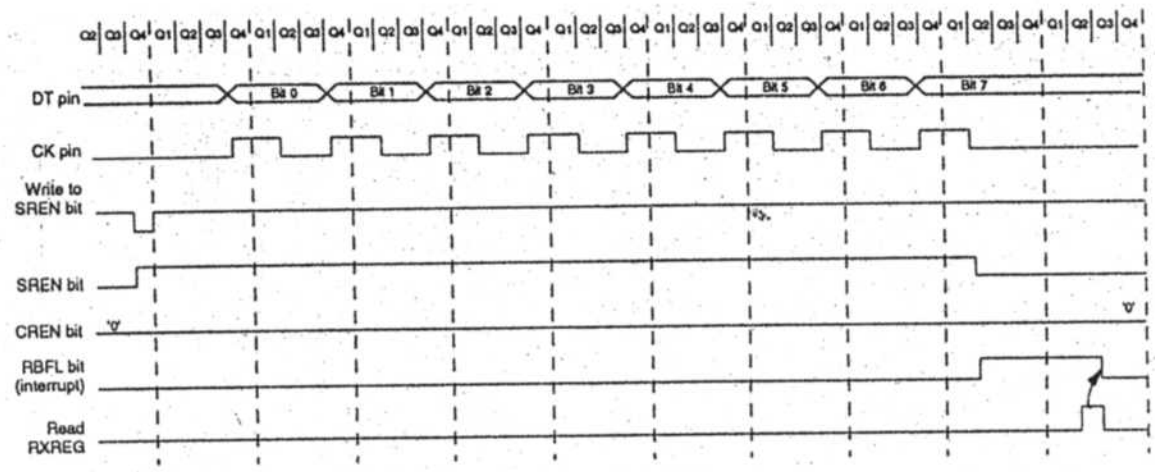


Figura 41. Ricezione sincrona come master

- Se si desidera abilitare l'interrupt, settare il bit TXIE;
- Se si desidera il bit di parità, settare il bit TX8/9;
- Abilitare la trasmissione settando il bit TXEN;
- Se in modo 9 bit, caricare il bit TXD8;
- Caricare il dato da inviare in TXREG.

Se, invece, viene abilitata la ricezione (CREN) prima di entrare in sleep, allora un dato può essere egualmente ricevuto anche in SLEEP.

All'arrivo dell'ultimo bit, il registro RSR trasferirà il dato in RCREG e, se il bit RXCIE è settato, si genererà un interrupt che sveglierà il PIC dallo stato di SLEEP e, come prima, se il GIE era abilitato, il program counter salterà all'indirizzo del vettore di interrupt.

I passi da seguire per una corretta ricezione sincrona (slave) sono i seguenti:

- Inizializzare il registro SPBRG per il baud rate desiderato, compreso il bit BRGH;
- Abilitare la porta seriale asincrona configurando i bit SYNC=1, SPEN=1 e CSRC=0;
- Se si desidera abilitare l'interrupt, settare il bit RCIE;
- Se si desidera il bit di parità, settare il bit RX8/9;
- Abilitare la ricezione settando il bit CREN=1;
- Quando la ricezione è terminata, il bit RCIF viene settato e, se abilitato l'interrupt di ricezione, viene generato un interrupt;
- Leggere il registro RCSTA per l'eventuale non bit e gli eventuali errori;
- Leggere il dato da RCREG;
- Se ci sono stati errori, azzerarli azzerando il bit CREN.

A questo punto, ci fermiamo qui: è importante infatti riuscire a non porre il carro davanti ai buoi e seguire gli argomenti con logica lineare e senza affrettare eccessivamente le problematiche che si possono incontrare lavorando con i microprocessori.

Nella prossima puntata continueremo a scoprire come grazie alle periferiche è possibile ottenere il massimo dai microprocessori e come proprio i Pic siano molto flessibili da questo lato. Ecco quindi un'altra buona ragione per continuare a seguirci tutti i mesi. E non è certamente l'unica.

continua

Le periferiche dei pic

I Pic sono dei processori in tecnologia Risc che permettono di realizzare circuiti complessi in spazi ridotti. Ne stiamo analizzando le periferiche

Paolo Sbrana - 7ª parte

La periferica che tratteremo in questa puntata si trova esclusivamente nella famiglia PIC16C62x e, nel momento in cui scriviamo, sono disponibili tre tipi di tali chip: il PIC16C620, il PIC16C621 ed il PIC16C622. Tutti e tre i PIC citati sono a 18 pin, ma usciranno presto anche chip a 28 pin siglati PIC16C642 e PIC16C662 con a bordo la stessa periferica. Il modulo in questione è formato da due comparatori, che possono essere connessi fino a otto modi diversi.

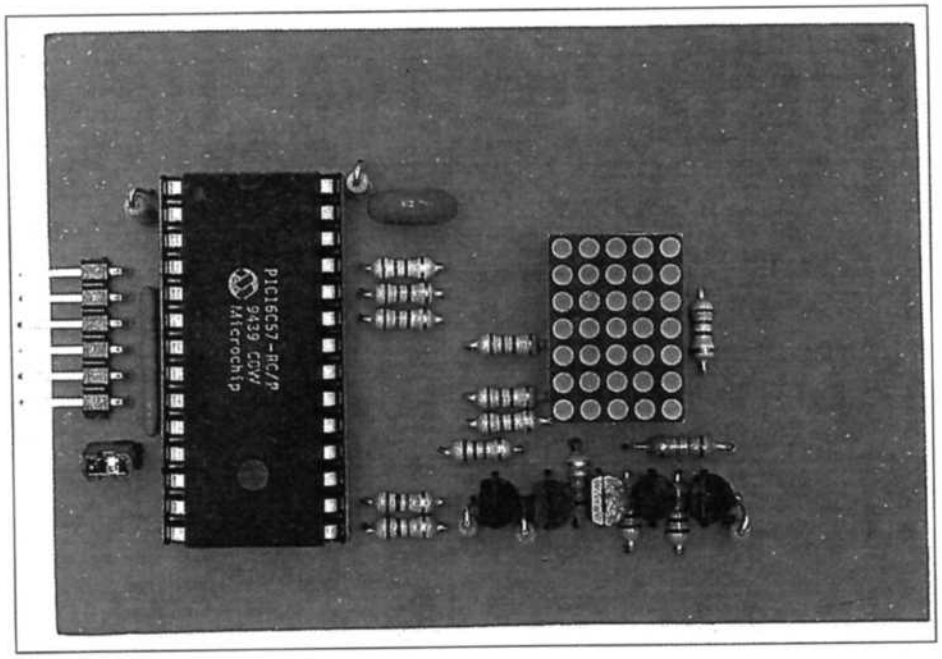
A che cosa servono dei comparatori in un microcontroller? Nel caso più semplice a fornire il risultato di un confronto tra due segnali analogici. Ad esempio, per implementare un interruttore crepuscolare, si potrebbero impiegare dei PIC con il convertitore A/D a bordo, per valutare l'intensità della luce e per agire di conseguenza, ma è anche possibile sfruttare un comparatore fissando preventivamente la soglia di attivazione desiderata.

Lo stesso dicasi per un termostato che deve comandare un raffreddatore oppure un riscaldatore.

La scelta allora cade sui comparatori per la semplicità di impiego e, fatto molto importante, sul minor costo del componente rispetto alla versione con A/D a bordo.

Allo stesso modo, se dobbiamo realizzare un circuito che necessiti di comparatori, è più conveniente inserire un chip della famiglia 62x piuttosto che preferire un comparatore esterno.

Ma oltre a questi due banali esempi, con i due comparatori a bordo del chip è possibile anche implementare dei circuiti che, per la loro precisione, danno addirittura risultati migliori del convertitore a 8 bit.



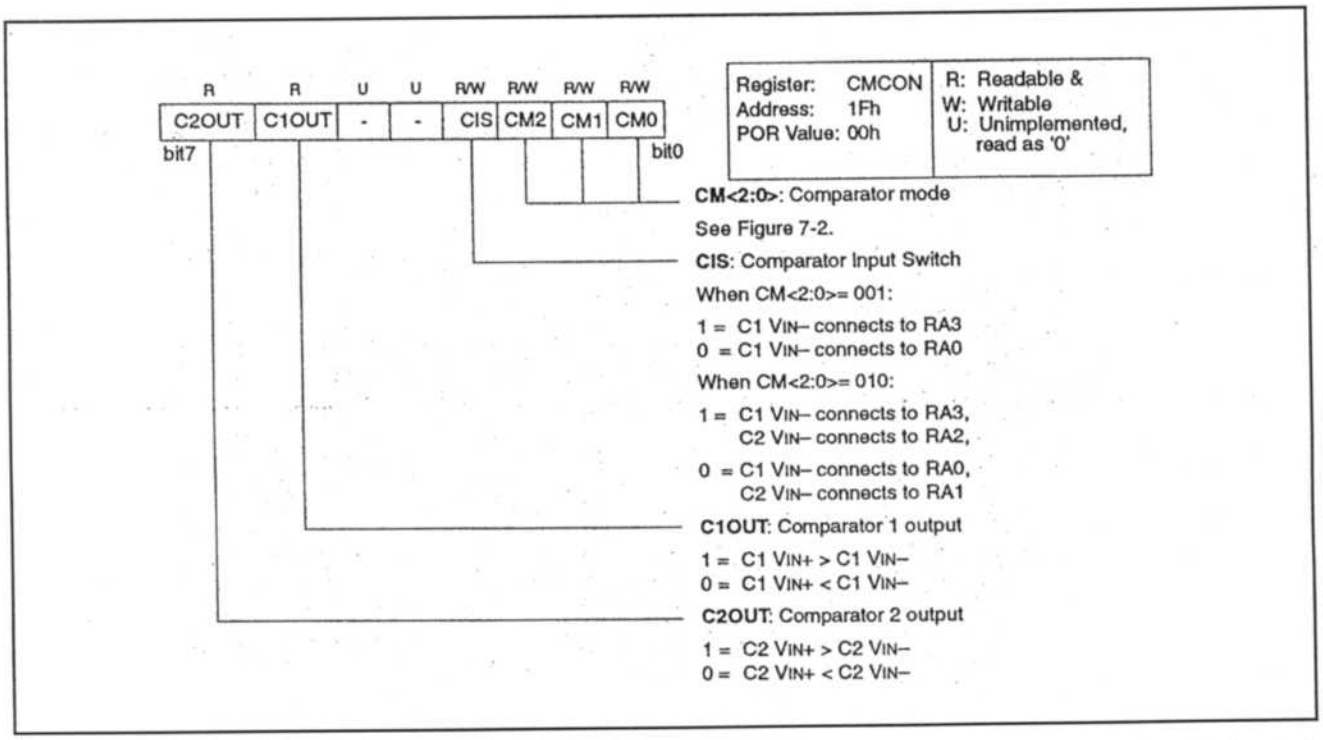


Figura 48. Il registro di controllo CMCON

Nelle applicazioni che Microchip offre ai progettisti, ne troviamo una in cui si sfrutta un PIC16C622 per realizzare un preciso Ohmmetro/Capacimetro le cui prestazioni sono paragonabili a tester commerciali di grande marca.

La periferica in dettaglio

Cerchiamo di capire quale sia il funzionamento del modulo comparatore partendo dall'analisi del registro che ne controlla le azioni e visibile in Figura 48 ovvero il registro CMCON.

I tre bit CM0, CM1 e CM2 servono per impostare la modalità di connessione che vedremo successivamente.

Il bit 3 (CIS = Comparator Input Switch) serve per commutare le porte RA0 e RA3.

I bit 4 e 5 non sono stati implementati.

I due bit 6 e 7 sono rispettivamente le uscite del comparatore C1 e del comparatore C2. Tali uscite valgono 1 quando Vin+ è maggiore di Vin- e valgono 0 diversamente. Vediamo allora le possibili configurazioni dei due comparatori aiutandoci con la Figura 49.

Quando i tre bit di configurazione valgono 000, i comparatori sono in uno stato di reset in cui le due uscite sono a livello 0.

Con i tre bit a 100 otteniamo due comparatori che lavorano in modo indipendente.

Con la configurazione 011, invece, i due comparatori lavorano sempre in modo indipendente, ma hanno in comune la Vin+.

Con i bit CM0, CM1 e CM2 a 101, si attiva solo il comparatore C2. Per disattivare entrambi i comparatori, si dovrà impostare 111.

Con la configurazione 010, si attivano i due comparatori con la Vin+ in comune e quattro ingressi multiplexati. Una soluzione simile (Vin+ in comune e tre input mutiplexati) si ha settando i bit a 001. L'unica configurazione che consente di avere su due pin le corrispondenti uscite dei due comparatori, è la 110.

Per utilizzare il modulo correttamente, è necessario settare opportunamente il registro TRISA, che gestisce la direzione della porta A.

Si deve fare attenzione se si attivano gli interrupt relativi a questo modulo: devono essere disabilitati durante un eventuale cambio di configurazione, in quanto potrebbero essere generati dei falsi interrupt.

Analisi di un singolo comparatore

Per definizione, un comparatore deve avere in ingresso due segnali da poter valutare, un Vin+ e un Vin-.

A seconda di come viene configurato il modulo, uno dei due segnali viene generato internamente oppure letto dall'esterno.

Nel secondo caso, tale segnale non potrà essere minore di 0 volt o maggiore di 5 volt.

Tabella 14. Registri associati al modulo comparatore

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1Fh	CMCON	C2OUT	C1OUT	–	–	CIS	CM2	CM1	CM0
9Fh	VRCON	VREN	VROE	VRR	–	VR3	VR2	VR1	VR0
0Bh	INCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0Ch	PIR1	–	CMIF	–	–	–	–	–	–
8Ch	PIE1	–	CMIE	–	–	–	–	–	–
85h	TRISA	–	–	–	TRIS4	TRISA3	TRISA2	TRISA1	TRISA0

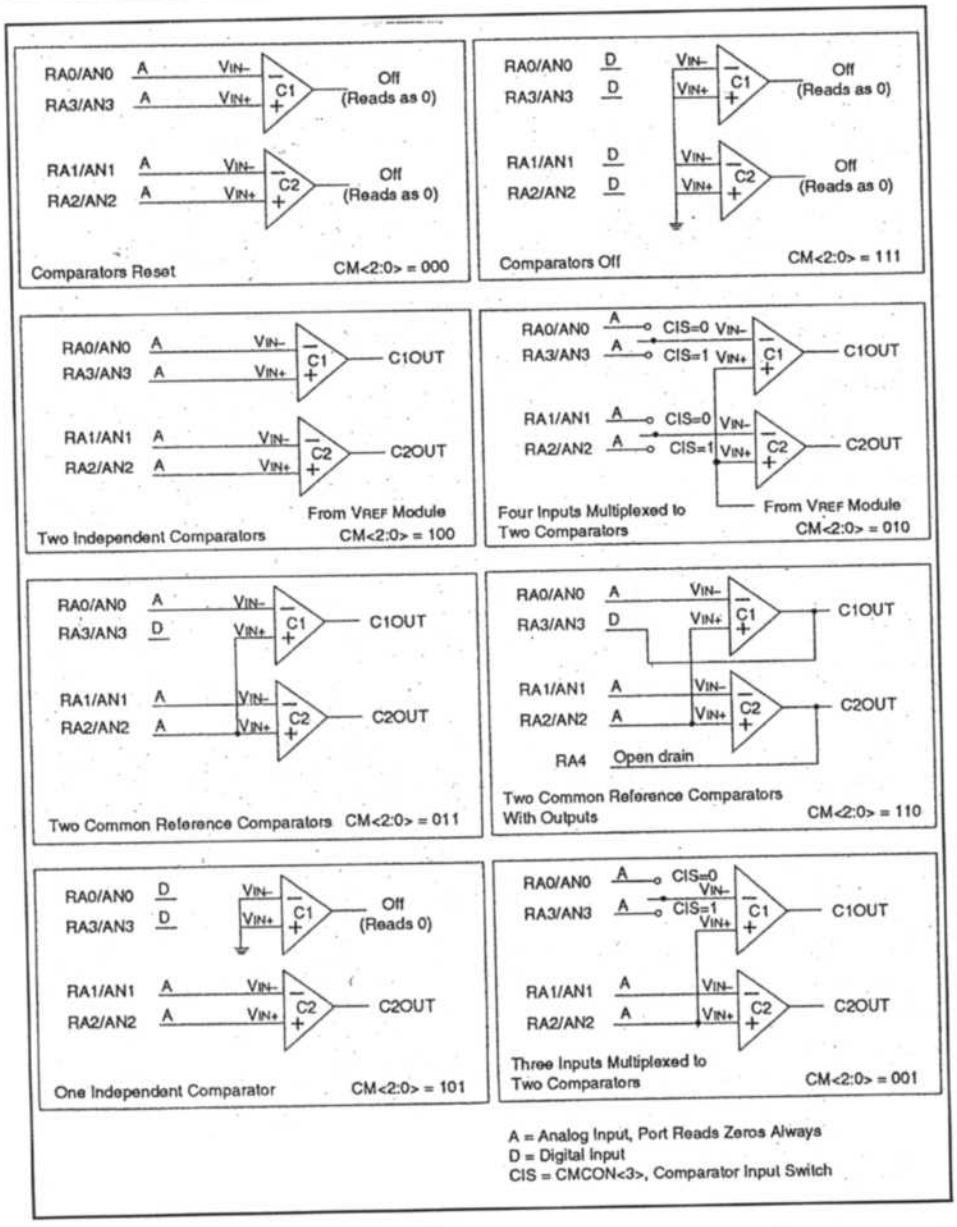


Figura 49. Possibili configurazioni dei comparatori

In Figura 50 troviamo la forma d'onda in uscita a un comparatore in funzione a un ingresso di tipo simil-sinusoidale. Si nota la soglia d'intervento e l'isteresi del componente.

In Figura 51 troviamo il diagramma a blocchi del modulo comparatore. Tali uscite possono essere lette attraverso il registro CMCOM o sulla porta A per un tipo particolare di configurazione e sono di sola lettura.

Si devono, a questo punto, precisare due cose.

Quando viene letta la porta A, tutti i pin configurati come ingressi analogici saranno sempre letti 0. I pin, invece, configurati come ingressi digitali varranno 1 o 0 in funzione del valore del trigger di Schmitt interno.

Inoltre, inviare un segnale analogico su di un pin configurato come input digitale può far assorbire al buffer di input più corrente di quella specificata.

Nel caso in cui si decida di impiegare l'interrupt relativo al modulo comparatore, si dovrà settare il bit CMIE del registro PIE1 ed il bit PEIE del registro INTCON.

Inoltre, dovrà essere settato il bit GIE. Poiché il flag di interrupt è unico, si avrà generazione di interrupt anche nel caso in cui un solo comparatore modifichi il suo stato di uscita (condizione per la generazione di interrupt).

Come nel caso del converter, anche in questo caso è possibile far eseguire una comparazione anche in sleep ed essere svegliati dall'interrupt relativo.

Al contrario, poiché il modulo comparatore consuma una certa corrente, se si decide di entrare in sleep senza dover eseguire comparazioni, si devono disabilitare i due comparatori. Un reset del chip forza il registro CMCON in uno stato di reset in cui i comparatori sono spenti (000). Ciò assicura che tutti gli input siano analogici, limitando la corrente in fase di inizializzazione.

In Figura 52 abbiamo un modello di input analogico semplificato.

Si notino i due diodi di protezione tra positivo e negativo inseriti nel caso in cui si impieghi tale pin come digitale.

Per questo motivo, però, i segnali analogici non possono scendere al di sotto di 0 volt e non possono superare i 5 volt, poiché in tal caso in uno dei due diodi scorrerebbe una discreta corrente.

L'impedenza di ingresso massima raccomandata è di 10 kΩ. I registri associati al modulo comparatore sono elencati nella Tabella 14.

Il riferimento di tensione

Il riferimento di tensione interno è formato da un partitore resistivo con 16 resistenze. Tale comparatore è segmentato per offrire due range di Vref e ha un controllo di on/off per preservare corrente quando non viene attivato.

Il registro che gestisce il riferimento di tensione si chiama VRCON ed in Figura 53 ne vediamo il contenuto.

I bit Vr0, Vr1, Vr2 e Vr3 servono per selezionare la tensione di riferimento nei due possibili range.

Il bit Vrr imposta invece uno dei due possibili range.

Il bit Vroe (Vref Output Enable) se viene posto a zero disconnette la tensione di riferimento dal pin RA2, viceversa se viene posto a 1 permette di ottenere sul pin RA2 la stessa tensione di riferimento interna. Infine, il bit Vren (Vref Enable) abilita o disabilita il modulo generatore di tensione. In Figura 54 vediamo il diagramma a blocchi del modulo riferimento di tensione. Per configurare correttamente tale modulo, la Vref viene così calcolata:

se Vrr=1 —>
Vref=(Vr<3:0>/24)*Vdd
se Vrr=0 —>
Vref=(Vdd*0,25) +(Vr<3:0>/32)*Vdd

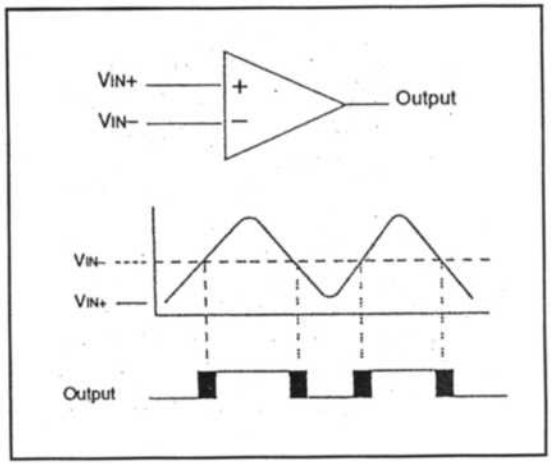


Figura 50. Singolo comparatore

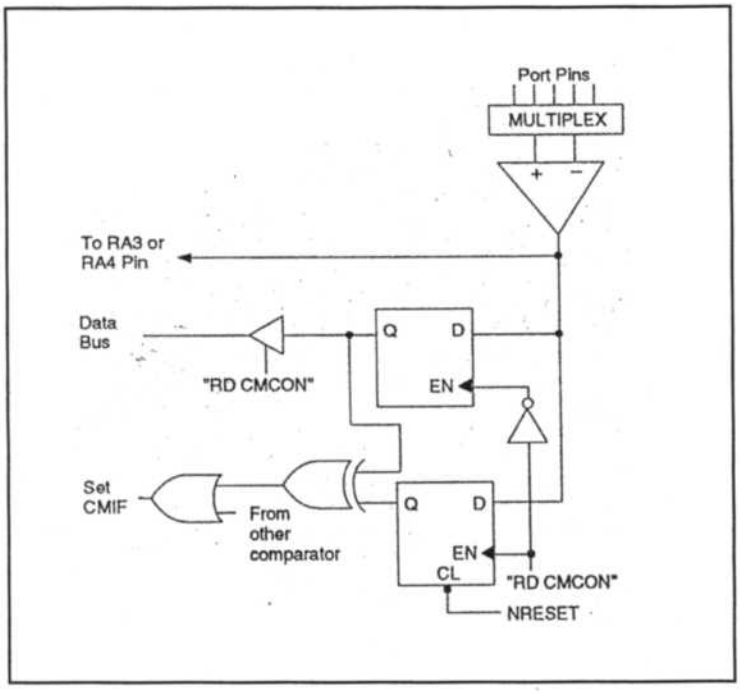


Figura 51. Diagramma a blocchi degli output del comparatore

Tabella 15. Registri associati al riferimento di tensione

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
9Fh	VRCON	VREN	VROE	VRR	–	VR3	VR2	VR1	VR0
1Fh	CMCON	C2OUT	C1OUT	–	–	CIS	CM2	CM1	CM0
8Ch	TRISA	–	–	–	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0

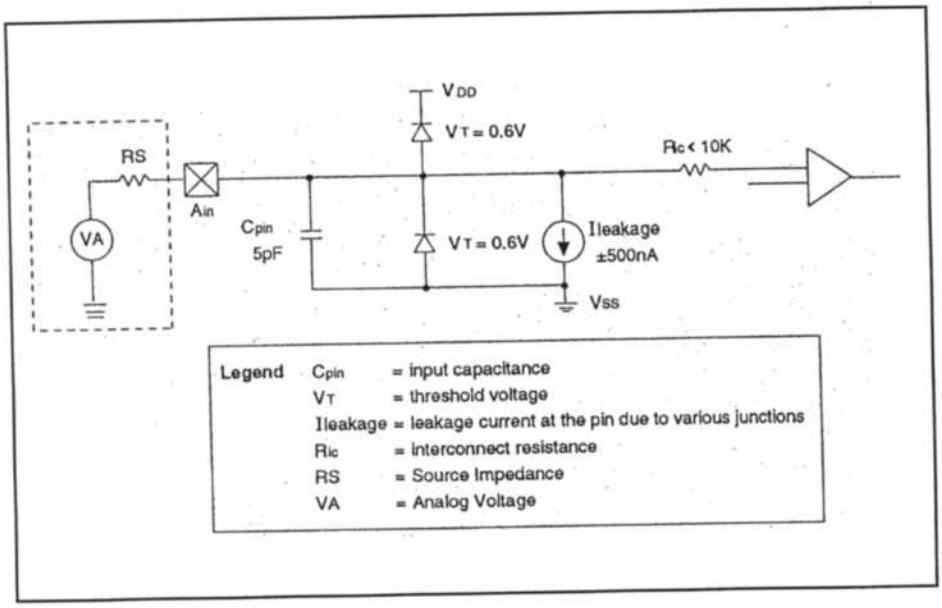


Figura 52. Modello di input analogico

Data la costituzione del modulo, non sarà possibile implementare la Vref con range da 0 a 5 volt compresi.

Inoltre, si dovrà curare molto la tensione di alimentazione Vdd, in quanto la Vref deriva da questa e, variazioni della Vdd introdurranno variazioni proporzionali nella Vref.

Al contrario di altri registri, quando il chip si risveglia da uno sleep, il VRCON non subisce modifiche.

Quando invece si presenta un reset della device, il modulo viene disabilitato e la connessione tra RA2 e l'uscita di

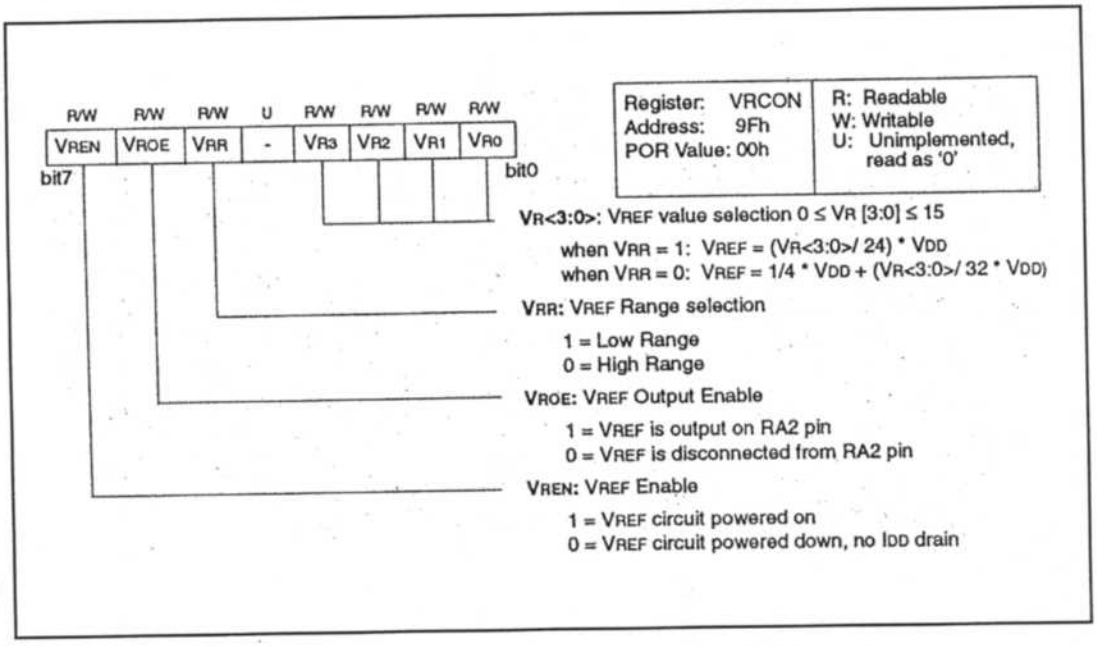


Figura 53. Il registro VRCON

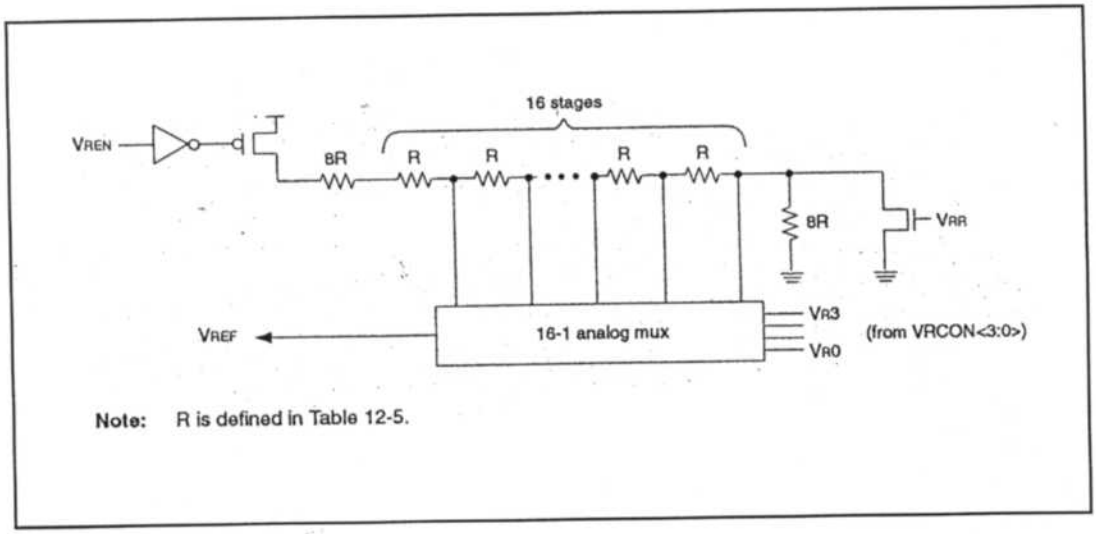
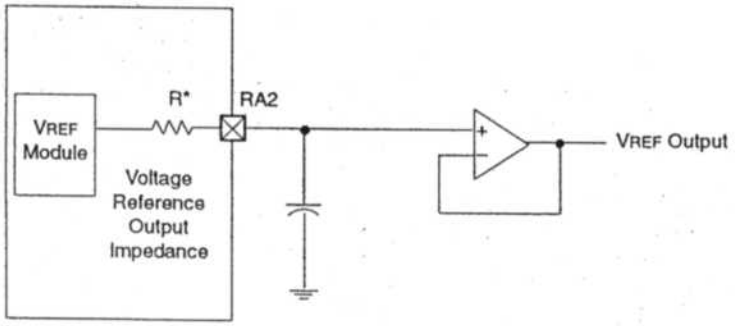


Figura 54. Diagramma a blocchi del riferimento di tensione



*R is dependent upon the Voltage Reference Configuration VRCON<3:0> and VRCON<5>.

tale modulo viene troncata. Viene, inoltre, selezionato il range alto di lavoro e i tre bit di selezione tensione vengono azzerati.

Il modulo generatore di tensione di riferimento, lavora in modo indipendente dal modulo comparatore, anche se noi lo abbiamo inserito nella stessa sezione descrittiva. L'output di questo modulo potrà essere connesso al pin RA2 se il bit 2 del registro TRISA sarà settato come pure il bit Vroe del registro VRCON. Abilitare la connessione del modulo al pin RA2 implica che se su quel pin si trova un segnale analogico, si avrà un maggior consumo di corrente. Lo stesso dicasi nel caso di configurazione del pin 2 come output digitale. Il pin RA2 potrebbe essere usato come semplice convertitore D/A con limitata capacità di carico. Per aumentare la capacità di carico, si deve applicare un amplificatore operazionale come mostrato in Figura 55.

Figura 55. Esempio del buffer di uscita del riferimento di tensione

Nella Tabella 15, infine, troviamo l'elenco dei registri associati con il modulo di generazione di tensione di riferimento.

- continua -