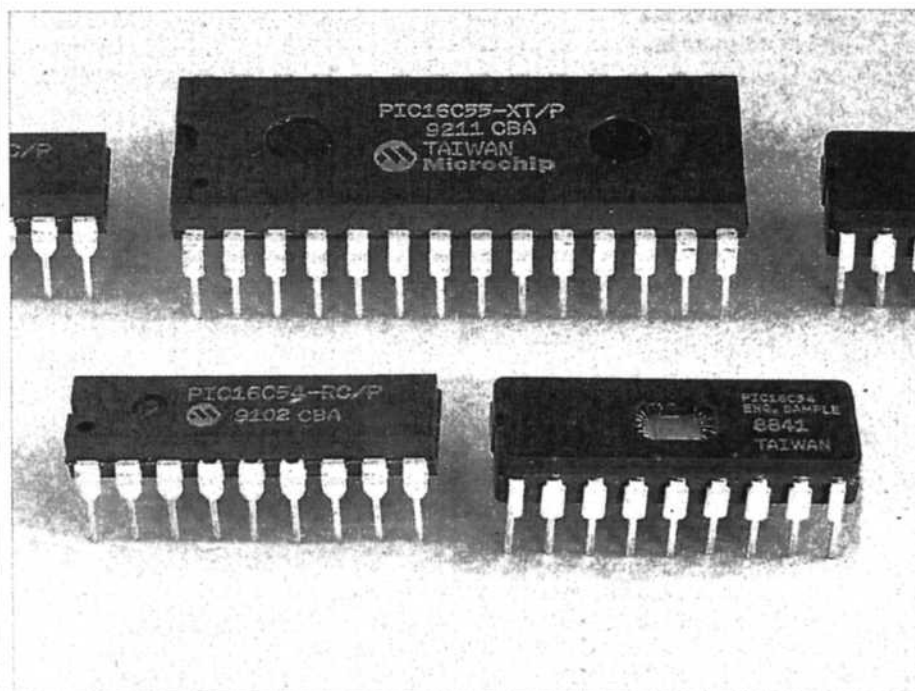


PIC 16C5x: IMPARIAMO A PROGRAMMARLO

Visto il grande interesse suscitato dai progetti realizzati con il controller PIC16C5x, abbiamo deciso di dedicare una serie di puntate alla sua programmazione; una volta avviati a un buon livello di preparazione, inoltre, presenteremo il progetto di un programmatore di PIC.

di Andrea Sbrana IW5CBO - 1ª parte



Caratteristiche tecniche

Set di istruzioni limitato (33)
 Massima velocità operativa
 20 MHz
 Eprom interna da 512 byte a
 2 K x12 bit
 SRAM interna costituita da 25
 a 72 registri a 8 bit
 7 registri dedicati
 2 livelli di stack
 Da 12 a 20 porte bidirezionali
 programmabili
 Prescaler interno a 8 bit
 Oscillatore interno
 Watchdog
 Eprom mascherabile
 alla lettura esterna
 SLEEP per risparmio di corrente
 Range di funzionamento 3-5,5 V
 Consumo <2 mA a 5 V 4 MHz
 15 µA a 3 V 32 kHz
 <3 µA in stand-by

La RAM interna, a differenza di quanto ci si possa aspettare, è formata da un certo numero di registri a otto bit, indirizzabili sia direttamente che indirettamente.

Sono a disposizione anche altri sette registri dedicati, il cui utilizzo verrà analizzato in seguito.

I livelli di stack, cioè il numero di chiamate a subroutine annidate, è fissato in due, secondo il tipo di device utilizzato.

Sempre a seconda del tipo scelto, le porte di ingresso/uscita possono essere 12 oppure 20, tutte programmabili via software e con continuità: anche durante il normale funzionamento, per ogni singolo bit sia in input che output.

Tutti i tipi di PIC possiedono sia il watchdog che un prescaler: il primo, letteralmente tradotto come "cane da guardia", controlla che la CPU (termine con cui chiameremo alcune volte il PIC) non si sia in qualche modo "pian-tata" e, eventualmente, forza con un reset interno la riesecuzione del programma, mentre il secondo serve per

Nel corso degli ultimi mesi, abbiamo pubblicato diversi progetti che si basavano su un nuovo tipo di controller: il PIC. Siamo fortemente convinti sia della validità di questi componenti che della futura diffusione e per questo motivo abbiamo deciso di incominciare questo corso di programmazione. Una volta arrivati a comprendere tutte le funzioni, inoltre, pubblicheremo un semplice progetto di programmatore di PIC che vi permetterà di mettere in pratica quanto "studiato". Iniziamo con l'analisi delle caratteristiche peculiari di questi controller.

PIC 16C5x: le caratteristiche

Possiamo notare che hanno un set di istruzioni molto ridotto che, se da un lato limita l'operatività, dall'altro facilita la velocità esecutiva. La EPROM interna può variare da 512 byte a 2 K x12 bit, a seconda del modello di PIC scelto.

Chi ha già un minimo di pratica sui controller, sa che 512 byte di EPROM nella maggior parte dei casi non vengono mai completamente riempiti.

Esiste, inoltre, la possibilità di mascherare il programma memorizzato per non farlo poi rileggere in alcun modo da eventuali "copiatori" di professione!

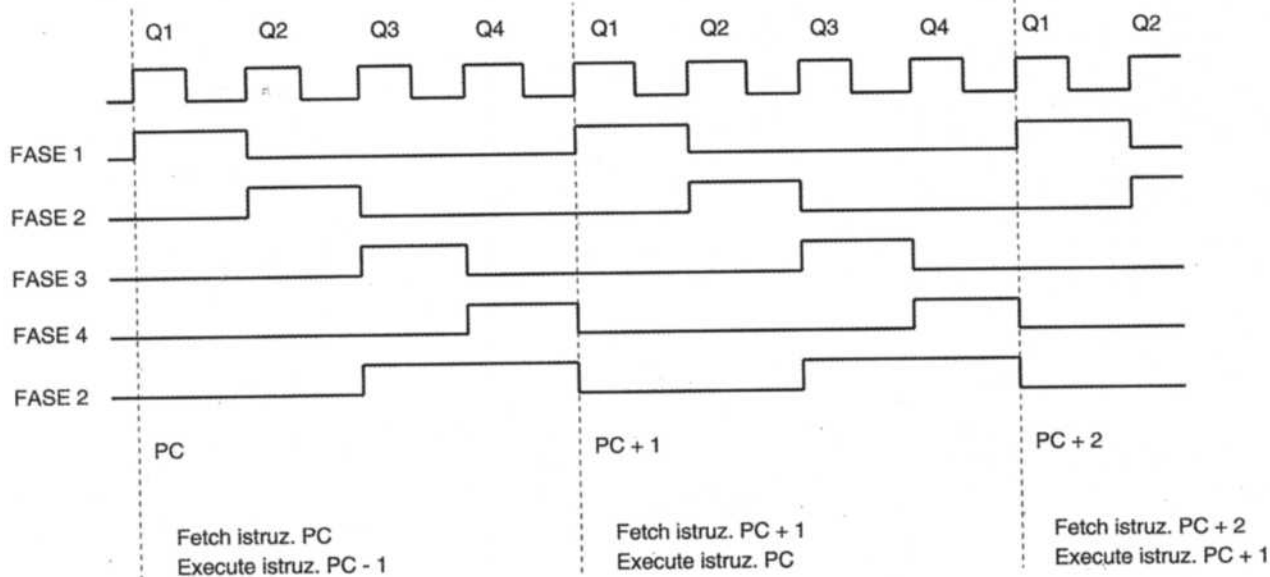


Figura 3. Diagramma temporale del ciclo di fetch

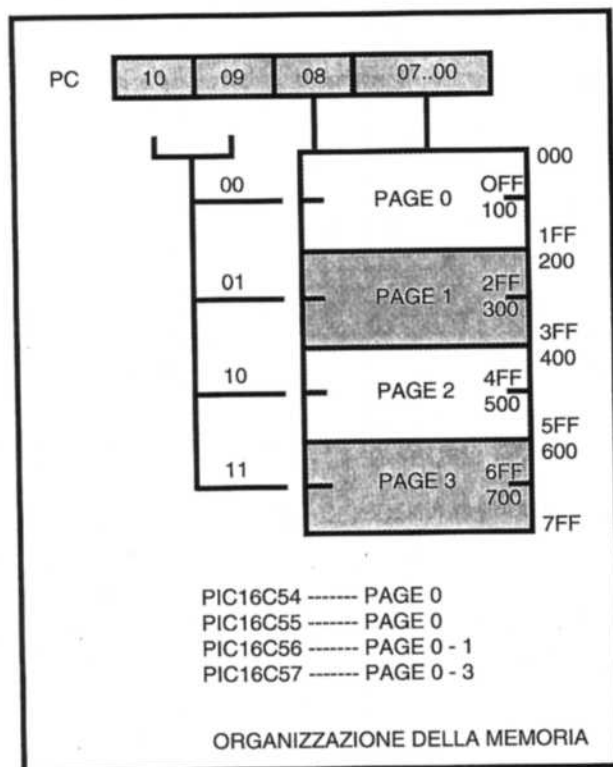


Figura 4. Organizzazione della memoria SRAM

come visto nel progetto della tastiera per antifurti pubblicata nel mese di gennaio '93. Ovviamente, sono possibili molte altre applicazioni, limitate solo dalla struttura interna del chip, dalla fantasia del progettista e dalle reali esigenze tecniche. Possiamo solamente citare alcune delle applicazioni più comuni in cui abbiamo utilizzato un PIC: trasmissione di dati e/o codici DTMF, controllo velocità motori, convertitori per seriale RS232, antifurti, controllo impianto elettrico di edifici, contatori, visualizzatori, telecomandi intelligenti.

Descrizione della struttura interna

In Figura 1 vediamo la piedinatura corrispondente ai vari device attualmente in commercio, mentre in Figura 2 troviamo lo schema a blocchi.

La prima componente da vedere è la EPROM: in tale memoria il PIC immagazzina il programma che noi abbiamo scritto.

Questa memoria potrà essere di 512 byte, di 1 K oppure di 2 K, a seconda della device scelto. Ogni cella dell'EPROM è di 12 bit.

L'indirizzamento della istruzione viene garantito da un registro detto PROGRAM COUNTER (PC) che ha il compito di indirizzare una cella della EPROM

e di tenere conto nel far ciò di alcuni elementi fondamentali.

Il primo di questi è la cella da puntare dopo un reset (ad esempio quando viene data l'alimentazione): inizialmente il PC punta alla cella di indirizzo più alto, quindi nel caso del PIC16C54 punta alla 1FFh, dove per h intendiamo il numero espresso in esadecimale.

Altro elemento fondamentale è il livello di stack in cui il PIC si trova: in pratica ci sono due registri, STACK1 e STACK2, che memorizzano un indirizzo in base a ogni istruzione CALL (che vedremo in seguito) e con la sequenza di una memoria LIFO (Last In First Out) o, per meglio dire, di una "pila".

Se, quindi, nel programma il PIC trova una istruzione CALL, inserisce in STACK1 l'indirizzo attuale e nel PC viene memorizzato l'indirizzo a cui saltare. Se successivamente viene trovata un'altra istruzione CALL, il PIC memorizza il valore registrato in STACK1 dentro STACK2 e in STACK1 inserisce il nuovo valore attuale.

Viceversa, a ogni istruzione RETLW, viene riportato nel PC il valore presente nello STACK1 e, dopo, in STACK1 viene trasferito il valore di STACK2.

Ultimo elemento da tener presente è il "ciclo di fetch". Ogni CPU, infatti, ha il seguente modo di procedere nel suo

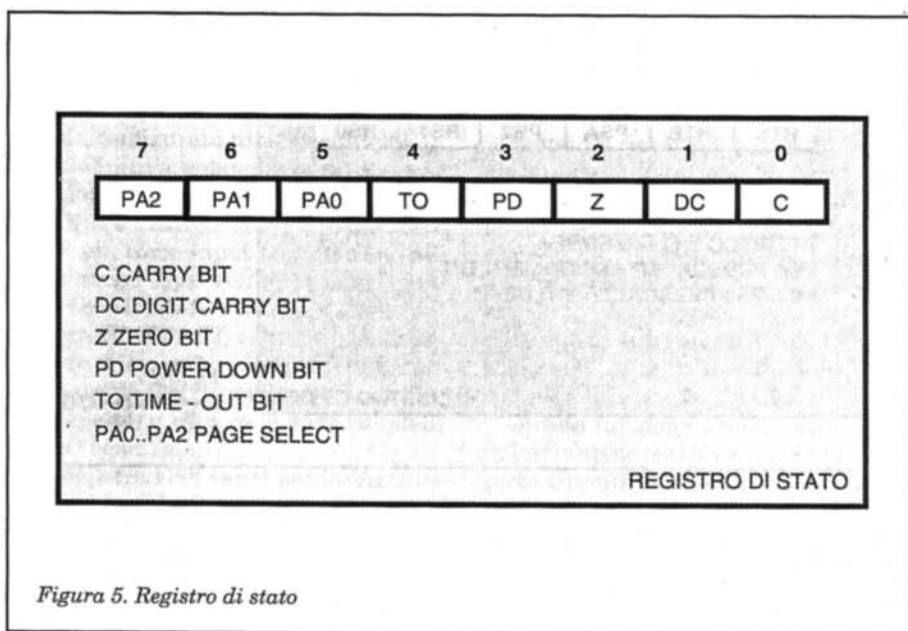


Figura 5. Registro di stato

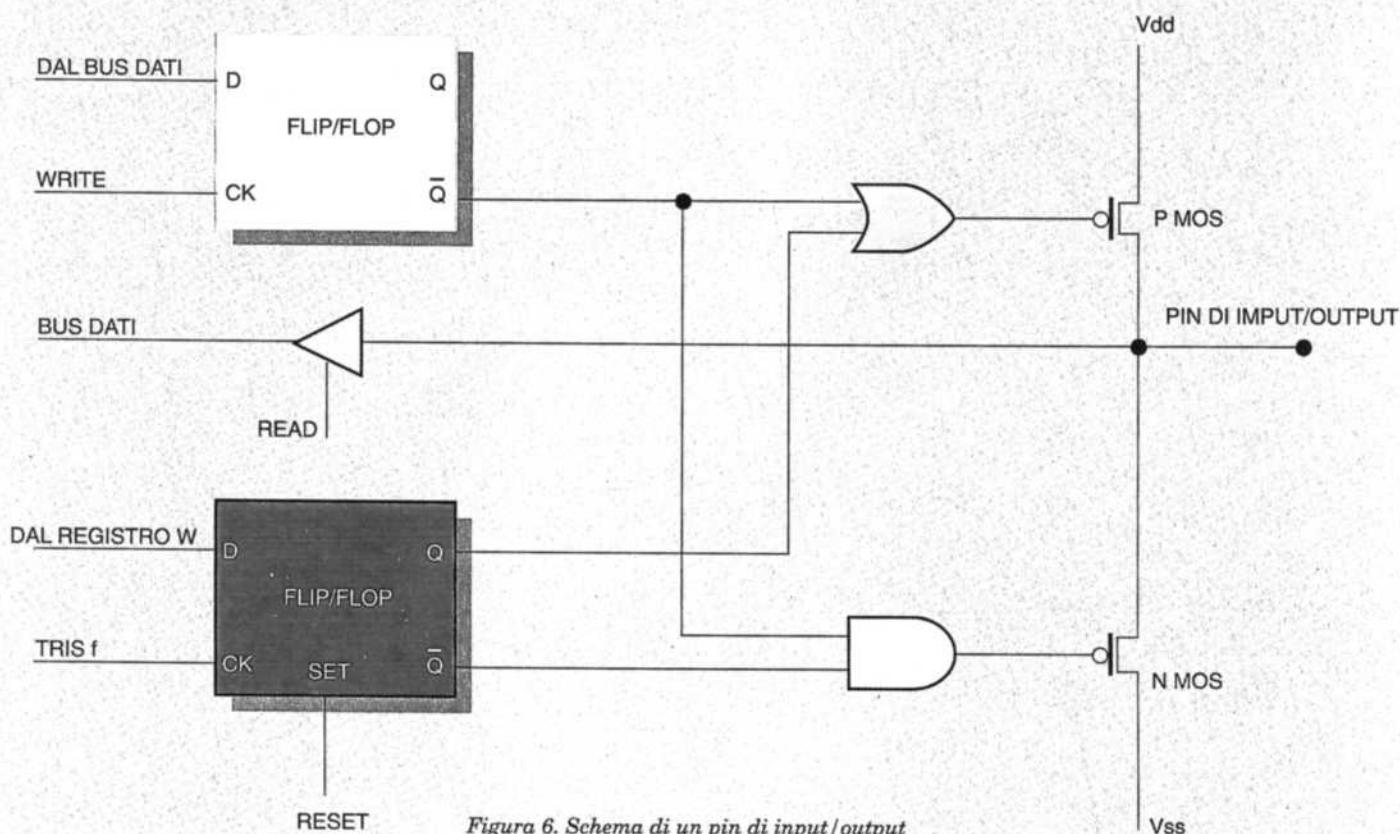
lavoro: il PC punta a una cella di memoria che contiene l'istruzione da eseguire successivamente, istruzione che viene immagazzinata nel REGISTRO ISTRUZIONE. Da qui passa alla zona attiva di DECODIFICA ISTRUZIONE e viene eseguita, incrementando così di nuovo il PC.

In pratica, il ciclo di fetch consiste nel

far puntare al PC la cella dell'istruzione successiva a quella corrispondente all'attuale operazione in svolgimento in modo tale da risparmiare tempo.

In Figura 3, potete vedere il diagramma degli stati di questi passaggi.

Infine, dato che i vari tipi di PIC hanno memoria di diversa lunghezza, anche i bit del PC variano di conseguenza, come



È semplice intuire che quando il segnale sul clock del flip-flop in basso ha un fronte ascendente, il flip-flop memorizza sull'uscita Q il valore letto sull'ingresso D. Supponiamo che questo sia 1 e, quindi, Q si porta a 1 e il P-MOS viene interdetto e isola il pin di input dal positivo. Inoltre, essendo Q negato uguale a zero, anche il transistor N-MOS viene interdetto, isolando il pin di ingresso anche dal negativo.

A questo punto il pin in questione è stato programmato per funzionare da INPUT e qualsiasi valore tenti di scrivere via software viene ignorato.

Il dato di input potrà essere letto con un'istruzione che abiliti il three-state READ sul buffer di ingresso al BUS DATI. Supponiamo, invece, che il dato letto sia 0; allora i due transistor verranno abilitati a funzionare tramite le due porte OR e AND nel seguente modo: la porta AND darà un 1 in uscita quando il dato letto dal primo flip-flop è zero, mentre darà 1 nel caso contrario.

Quindi, il transistor N-MOS porterà il pin di input/output a massa se il dato

immagazzinato nel primo flip-flop è 0 (pin configurato come uscita bassa). Viceversa, si attiverà il transistor P-MOS, portando il pin di input/output al positivo (pin configurato come uscita alta).

In definitiva, abbiamo visto, tornando alla Figura 2, che il registro TRIS_A (TRIS_B e TRIS_C) serve a configurare i vari pin come input oppure come output, anche separatamente (corrispondenza con il flip-flop in basso), mentre il registro F5 (F6 e F7) viene utilizzato per settare a 1 o a 0 i pin già configurati come uscita (corrispondenza con il flip-flop in alto) e leggere lo stato di quelli settati come input.

Il registro FSR serve per indirizzare indirettamente un altro registro e lo vedremo prossimamente con le istruzioni idonee. Il registro OPTION viene impiegato per attivare o disattivare il prescaler, per variarne la scala di riduzione e per decidere se il segnale di RTCC deve essere valutato sul fronte di discesa oppure su quello di salita.

In Figura 7 possiamo vedere questo registro bit per bit. Il registro PRESCA-

LER è un registro dedicato al conteggio di impulsi che possono arrivarvi sia dal clock interno che da un pin apposito, il RTCC (Real Time Clock Counter).

Scopo del registro è quello di dividere gli impulsi in multipli di 2, fatto molto interessante se si pensa, ad esempio, di sfruttare il PIC come frequenzimetro o come lettore di rapporti impulso/pausa.

Il registro WATCHDOG, invece, se attivato, poiché gestito via software, consente di determinare se la CPU si è "bloccata" e, in tale situazione, ha il compito di resettare il tutto.

Questa funzione è molto utile specie in impianti dove il funzionamento di un certo circuito deve essere garantito in ogni momento. Infine, è presente un registro detto di CONFIGURAZIONE che non può essere scritto, ma solamente letto e che dà informazioni quali il device, il tipo di oscillatore e altre.

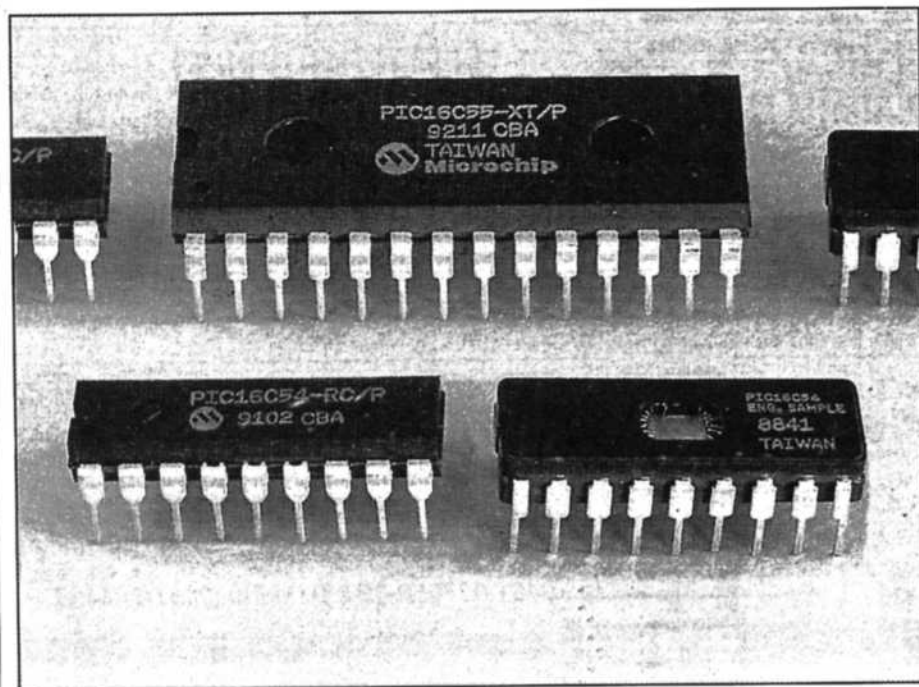
Per concludere la spiegazione sui PIC, ne riportiamo in Figura 8 la struttura interna in relazione ai vari banchi di memoria possibili.

- continua -

CORSO DI PROGRAMMAZIONE PER IL CONTROLLER PIC16C5X

Nella prima parte della descrizione di questi controller abbiamo iniziato l'analisi della struttura interna e dei registri principali di questo interessante famiglia di controller; questo mese vedremo le istruzioni del PIC, come vengono codificate e quali output producono.

di Andrea Sbrana IW5CBO - 2ª Parte



Le prime nozioni da tenere presente quando ci si accinge a scoprire un nuovo controller sono la sua struttura e le sue istruzioni, per poter estrarre da esso il migliore dei risultati.

Per prima cosa notiamo che anche se il PIC è un controller ad 8 bit, le istru-

zioni vengono codificate, insieme agli operandi, in un'area di dodici bit.

Le istruzioni

Ogni istruzione ha una codifica, un codice mnemonico, l'eventuale operando e può o meno modificare lo stato del

registro di STATO visto nella puntata precedente.

La prima operazione che prendiamo in esame è la NOP, ovvero la "no operation", cioè quella operazione che non modifica in alcun modo lo stato del PIC.

La domanda che molti già si porranno è questa: perché inserire una istruzione che non fa niente? Vedremo in seguito che per attese e per ogni tipo di timer tale istruzione deve essere obbligatoriamente usata.

La sua codifica vede tutti i bit a zero (000h), non ha operandi e non produce cambi di stato, ma semplicemente fa passare un ciclo in più durante l'esecuzione di un programma.

Anche l'istruzione MOVWF non altera il registro di stato del PIC, ma modifica il registro f (cioè il suo operando) immagazzinandovi ciò che trova nel registro W. Il numero di f potrà variare, a seconda del device, da zero a un massimo 1Fh.

Questa istruzione può essere utile ad esempio per settare un registro a un ben definito valore iniziale prima di un determinato conto.

Ci sono poi due istruzioni che resettano il contenuto di W e di un registro f dato come operando e sono rispettivamente CLRW (Clear W) e CLRf (Clear f). La chiamata a entrambe fa in modo che venga modificato il bit numero 2 del registro di STATO.

Ricordiamo che questo bit viene settato a 1 se il risultato di una istruzione è zero.

La codifica per CLRW è 040h, mentre per CLRf è 06fh.

Per valutare le successive istruzioni, fissiamo ora alcune notazioni: f è sempre considerato il numero di un registro e potrà variare tra 0 e 1fh, d invece indica la destinazione del risultato dell'operazione, che potrà essere W se d=0, oppure f se d=1.

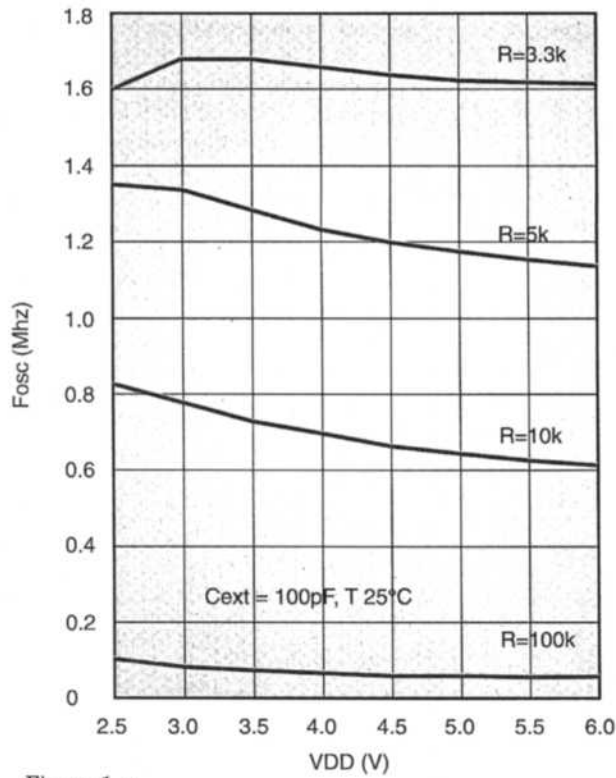


Figura 1. a

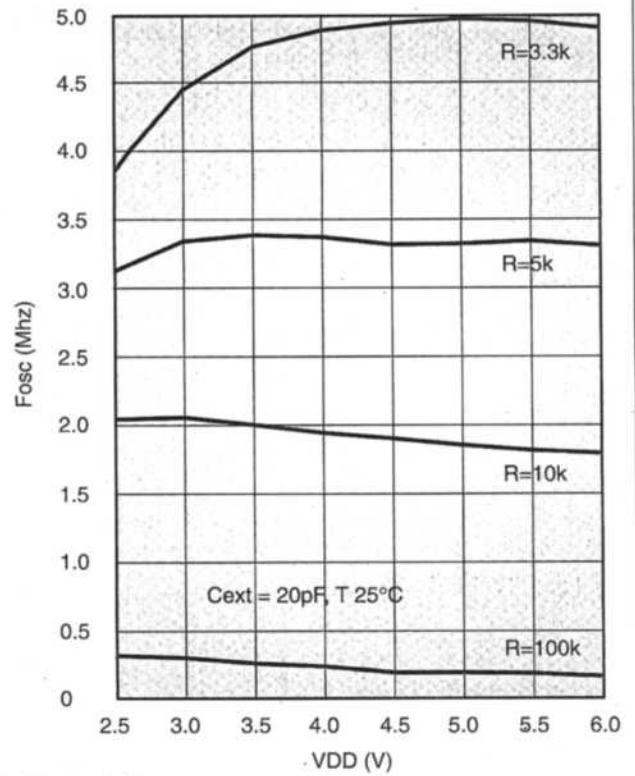


Figura 1. b

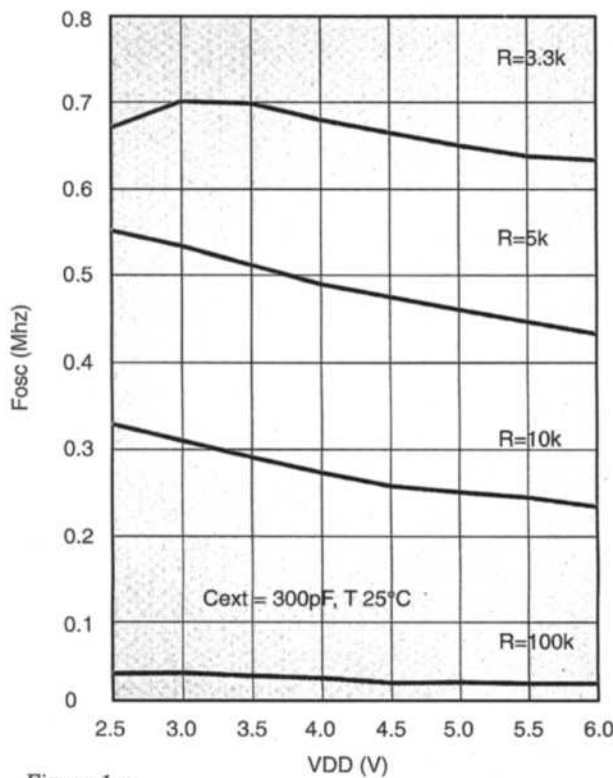


Figura 1. c

Cext	Rext	Average Fosc @ 5V, 25° C	
20 pF	3.3k	4.71 MHz	± 28%
	5k	3.31 MHz	± 25%
	10k	1.91 MHz	± 24%
	100k	207.76 KHz	± 39%
100pF	3.3k	1.65 MHz	± 18%
	5k	1.23 MHz	± 21%
	10k	711.54KHz	± 18%
	100k	75.62 KHz	± 28%
300pF	3.3k	672.78 KHz	±14%
	5k	489.49 KHz	± 13%
	10k	275.73 KHz	± 13%
	100k	28.12 KHz	± 23%

Figura 2

Figura 1: a),b),c) slittamento in frequenza dei circuiti oscillanti RC

Figura 2: Valori di R e C per ottenere frequenze standard

La prima istruzione a due operandi è la SUBWF (SUBtract W From).

Il primo operando indica il registro da cui sottrarre W e il secondo la destinazione: è possibile, infatti, sottrarre W ad esempio dal registro 09h e decidere se riscrivere il risultato in 09h oppure in W.

La codifica di SUBWF è 08Fh e a istruzione avvenuta vengono modificati i bit C, DC e Z del registro di STATO.

La DECF (DECrement f), invece, decrementa il valore che trova nel registro f e pone il risultato in f o in W.

Se il valore iniziale era zero, dopo questa operazione nel registro troveremo FFh.

Il codice della DECF è 0CFh e anche questa modifica il registro di STATO nel bit 2: può essere usata per introdurre

dei cicli andando poi a testare il bit 2 di questo registro per vedere se si è arrivati a zero.

Le istruzioni "booleane"

Esistono due tipiche istruzioni da matematica booleana e cioè la IORWF (Inclusive OR W and f) e la ANDWF (AND W and f) che, rispettivamente eseguono l'operazione di or e di and fra W e i registri indicati come operandi.

Il bit 2 del registro di STATO viene modificato in conseguenza.

Le codifiche per le due istruzioni sono, rispettivamente, 10Fh e 14Fh.

Come le precedenti, anche la XORWF (eXclusive OR W and f) compie un'operazione booleana e cioè l'or esclusivo tra W e il registro f.

Il suo codice operativo è 18Fh e anch'essa modifica il registro di STATO.

La ADDWF (ADD W and f) somma il valore di W a quello contenuto nel registro f e lo deposita in d.

Poiché possono intervenire problemi di overflow, vengono interessati i bit C, DC e Z del registro di STATO.

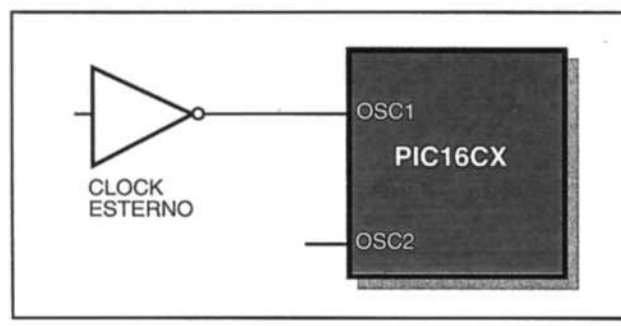
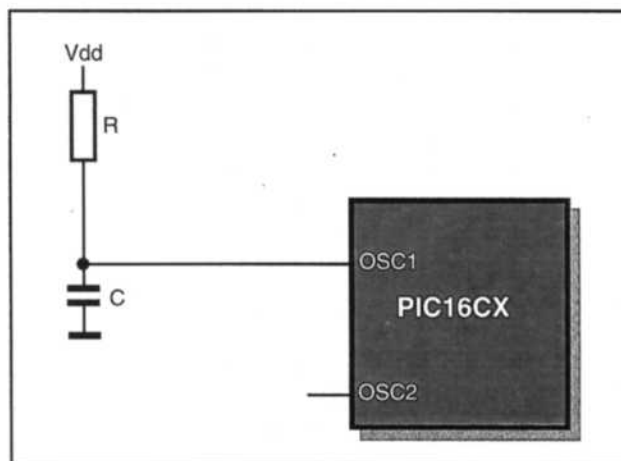
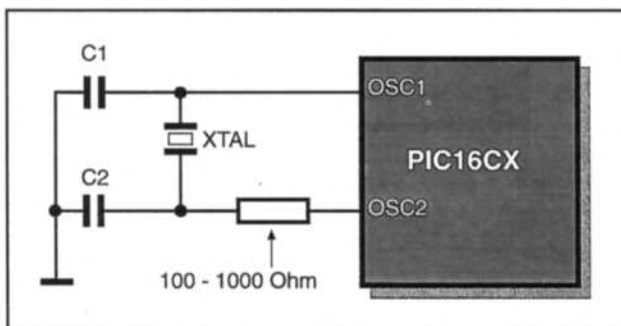
Il codice della ADDWF è 1CFh.

Analizziamo ora un'altra istruzione che "sposta" i contenuti dei registri, e cioè la MOVF (MOVE f). Con essa, il valore contenuto nel registro f viene immagazzinato nella destinazione d.

Poiché la destinazione può essere esclusivamente o il registro stesso o W, nel primo caso si ha l'effetto di una NOP (f viene riscritto con il suo valore).

Il codice di questa istruzione è 20Fh.

Con la COMF (COMPLEMENT f) viene



OSCILLATORI AL CRISTALLO

Osc Type	Freq	C1	C2
LP XT	32 KHz	15pF	15pF
	100 KHZ	15 - 30 pF	200 - 300 pF
	200 KHz	15 - 20 pF	100 - 200 pF
	455 KHz	15 - 20 pF	15 - 100 pF
	1 MHz	15 - 30 pF	15 - 30 pF
	2 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 - 30 pF	15 - 30 pF
	8 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF

RISONATORI CERAMICI

Oscillatori Type	Risonatori Frequenza	Capacità C1 = C2
XT	455 KHz	150 - 330 pF
	2.0 MHz	20 - 330 pF
	4.0 MHz	20 - 330 pF
HS	8.0 MHz	20 - 200 pF

Figura 3. Valori dei componenti in realazione alla frequenza e al tipo di oscillatore

eseguito il complemento del registro f, cioè vengono sostituiti gli 1 con gli 0 e viceversa.

Anche per questa istruzione viene influenzato il bit2 del registro di STATO. Il codice operativo della COMF è 24Fh.

L'istruzione complementare alla DECF vista precedentemente è la INCF (INCRement f): rispetta tutte le caratteristiche viste per la DECF, ma invece di decrementare il valore di f, lo incrementa di uno.

Il suo codice operativo è 28Fh.

Una istruzione molto importante e usata sempre per la creazione di cicli è la DECFSZ (DECRement f Sip se Zero). In pratica esegue un decremento unitario come la DEC, ma al termine controlla se il risultato è zero.

Se ciò avviene, l'istruzione seguente nel programma viene saltata (skipped) e si passa a quella successiva.

È intuibile che in un ciclo si debba sempre testare se il conteggio è giunto alla fine o meno e questa istruzione esegue proprio questa operazione.

Il suo codice è 2CFh.

"I Pic, in virtù del fatto che sono processori "risc" dispongono di poche istruzioni, comunque sufficienti a svolgere operazioni anche complesse"

La manipolazione dei byte

Ci sono poi tre istruzioni che manipolano il byte del registro f e cioè la RRF (Rotate Right f), la RLF (Rotate Left f) e la SWAPF (SWAP halves f).

La prima ruota i bit del registro f verso destra, cioè in pratica li "shifta" di una posizione: il bit ottavo va al posto del settimo, il settimo al posto del sesto e così via fino al primo che va nel bit C del registro di STATO.

La RLF shifta anch'essa, ma verso sinistra con le stesse modalità operative della RRF.

La SWAPF, invece, scambia di posto i due nibble del byte da trattare.

I codici operativi delle tre istruzioni sono rispettivamente 30Fh, 34Fh e 38Fh.

Infine, la INCFSZ (INCRement f Skip se Zero) ha le stesse caratteristiche viste per la DECFSZ, ma questa volta

Le istruzioni orientate al byte

0000 0000 0000	NOP	-	-	None
0000 001F FFFF	MOVWF	f	f=W	None
0000 0100 0000	CLRW	-	W=0	Z
0000 011F FFFF	CLRf	f	f=0	Z
0000 10DF FFFF	SUBWF	f,d	d=f-W	C,DC,Z
0000 11DF FFFF	DECf	f,d	d=f-1	Z
0001 00DF FFFF	IORWF	f,d	d=W or f	Z
0001 01DF FFFF	ANDWF	f,d	d=W & f	Z
0001 10DF FFFF	XORWF	f,d	d=W or-ex f	Z
0001 11DF FFFF	ADDWF	f,d	d=W+f	C,DC,Z
0010 00DF FFFF	MOVf	f,d	d=f	Z
0010 01DF FFFF	COMf	f,d	d=f comp.	Z
0010 10DF FFFF	INCF	f,d	d=f+1	Z
0010 11DF FFFF	DECFSZ	f,d	d=f-1, salta se zero	None
0011 00DF FFFF	RRF	f,d	d(n-1)=f(n),d(7)=C,C=f(0)	C
0011 01DF FFFF	RLF	f,d	d(n+1)=f(n),d(0)=C,C=f(7)	C
0011 10DF FFFF	SWAPF	f,d	d=f(4-7)-f(03)	None
0011 11DF FFFF	INCFSZ	f,d	d=f+1, salta se zero	None

Tabella 1

l'operazione sul valore del registro è di incremento e non di decremento, utile per cicli con conteggio positivo.

Il suo codice operativo è 3CFh.

Abbiamo visto così tutte le istruzioni orientate al byte che sono presenti in Tabella 1.

In Tabella 2, invece, troviamo le sole quattro istruzioni orientate al bit.

Precisiamo che f indica il solito registro, mentre b il numero del bit su cui operare. b potrà valere da 0 a 7 essendo il registro f un 8 bit.

La prima istruzione che valutiamo è la BCF (Bit Clear f) che pone a zero il bit b del registro f.

Questa istruzione è molto usata per disattivare un'uscita se, ad esempio, il registro è il 5, il 6 o il 7.

Il suo codice operativo è 4BFh.

La complementare della BCF è la BSF (Bit Set f), che pone a 1 il bit b del registro f. Anche questa è molto utile per attivare le uscite del PIC.

Il suo codice operativo è 5BFh.

Ci sono ora due istruzioni che testano il valore di un singolo bit e saltano l'istruzione successiva se tale bit è a zero (BTFSC cioè Bit Test f Skip se Clear) oppure a uno (BTFSS

cioè Bit Test f Skip se Set).

I loro codici operativi sono rispettivamente 6BFh e 7BFh.

Al contrario delle due precedenti, queste istruzioni di test consentono di verificare il livello logico che ho su una porta di input.

Le ultime operazioni che prendiamo in esame sono quelle visibili in Tabella 3 e cioè quelle relative all'input di dati da programma (K indica il valore espresso in un byte) e al controllo del PIC.



