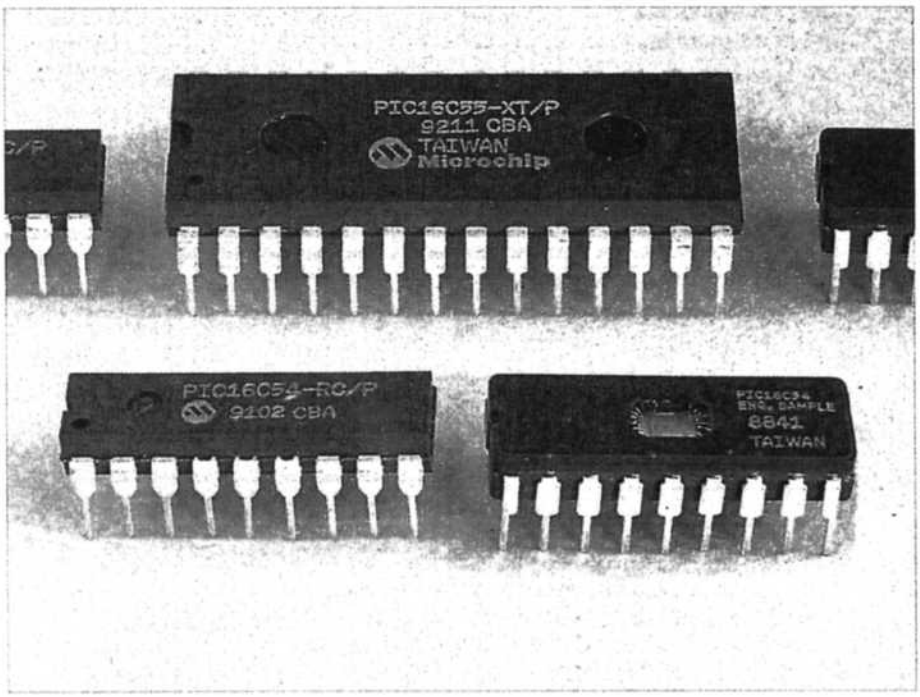
PIC 16C5x: IMPARIAMO A PROGRAMMARLO

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

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!

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 "piantata" e, eventualmente, forza con un reset interno la riesecuzione del programma, mentre il secondo serve per

dividere o contare il clock interno oppure un segnale digitale che giunge dall'esterno.

Ci sono quattro diversi tipi di oscillatore previsti per i PIC, anche se poi in commercio ne troviamo essenzialmente due: quello quarzato, che offre le migliori prestazioni e quello a rete RC.

In Figura 1 troviamo la piedinatura dei chip PIC16C54,55,56 e 57.

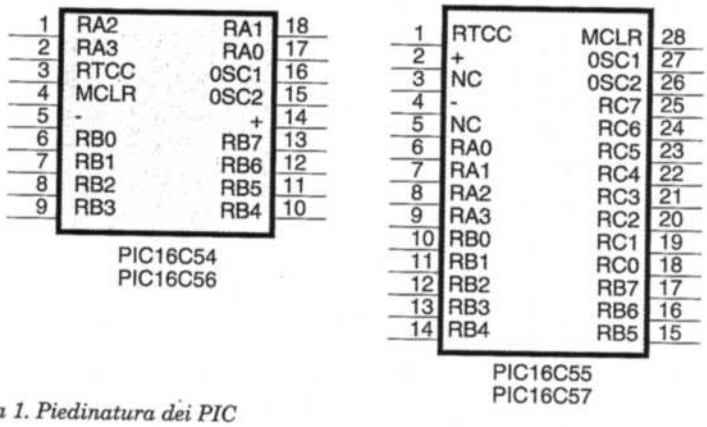


Figura 1. Piedinatura dei PIC

Applicazioni dei microcontroller

Prima di passare alla descrizione implementativa del chip, vogliamo soffermarci su quelle che sono le reali applicazioni di un controller di questo tipo.

Non possiamo, infatti, dichiarare che con il PIC si possa fare tutto quello che si gestisce con altri microprocessori come quelli della serie 80xx e nemmeno con personal computer.

Il PIC si inseriscono nella fascia dei microcontroller che, per la loro versatilità, per il loro ottimo rapporto prezzo/prestazioni e per la loro semplicità di programmazione consentono di pilotare dodici o venti linee digitali come, ad esempio, un display alfanumerico simile a quello utilizzato nel progetto del visualizzatore per campanelli presentato nel novembre '92, oppure un altro integrato, come l'UM91260A impiegato nel combinatore telefonico automatico presentato in febbraio '93, oppure senza altri componenti possono gestire tastiere e relè

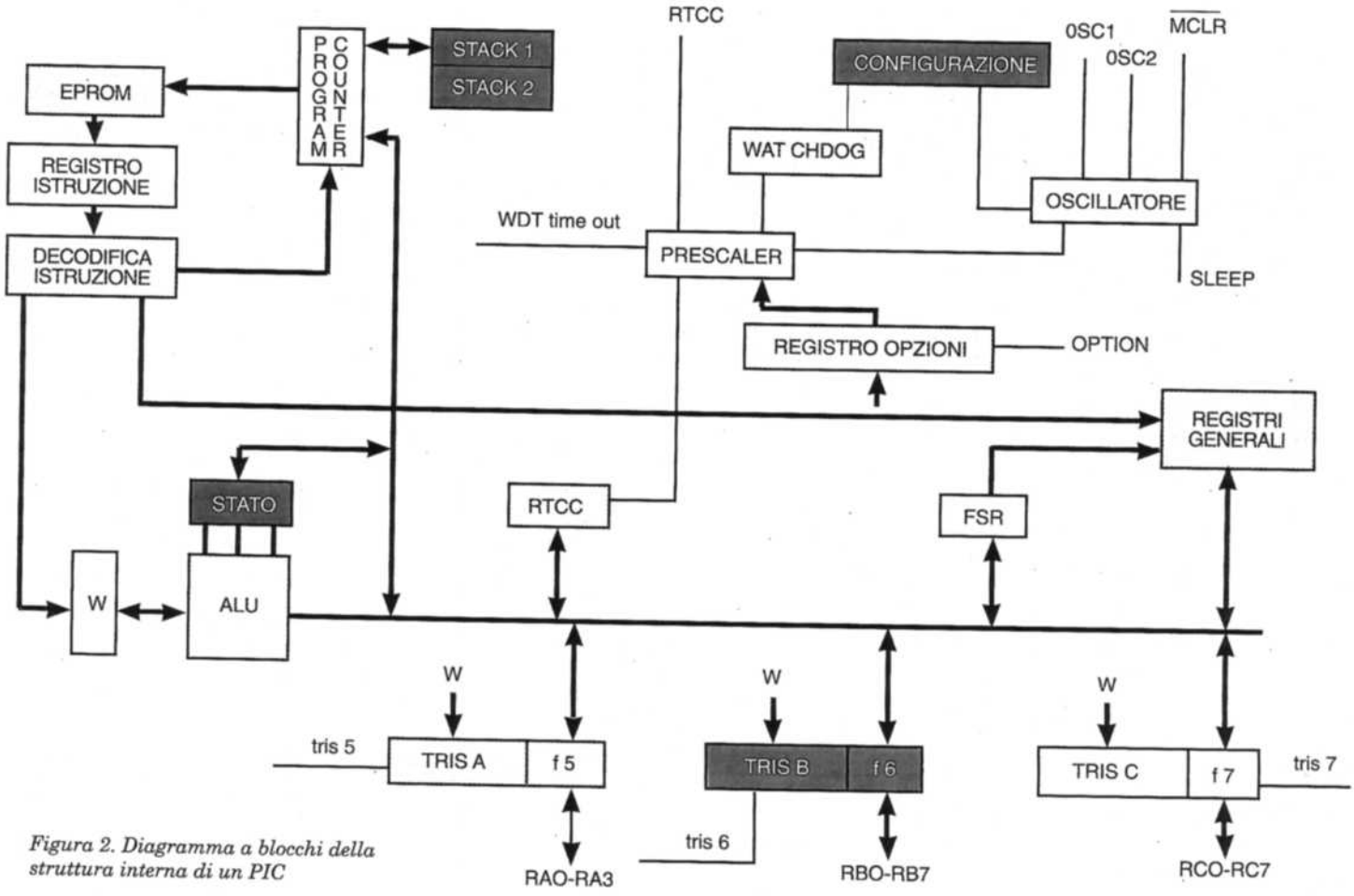


Figura 2. Diagramma a blocchi della struttura interna di un PIC

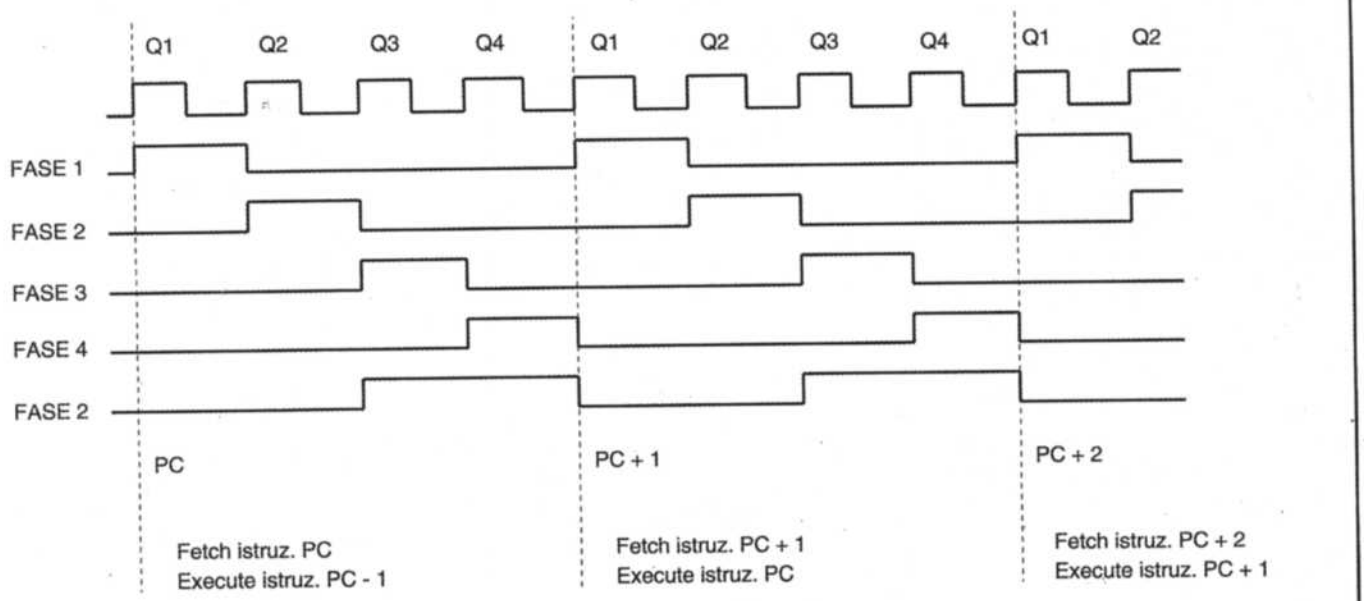


Figura 3. Diagramma temporale del ciclo di fetch

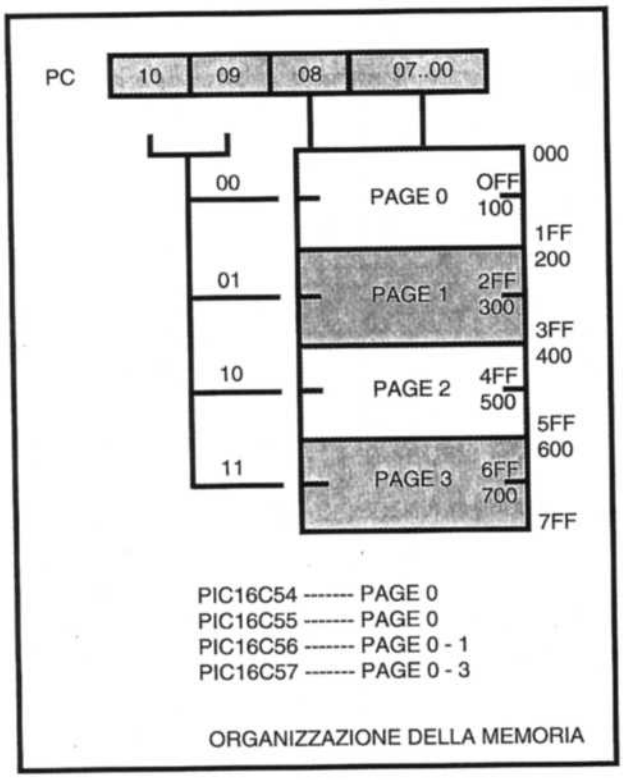


Figura 4. Organizzazione della memoria SRAM

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. Posiamo 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 del 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 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

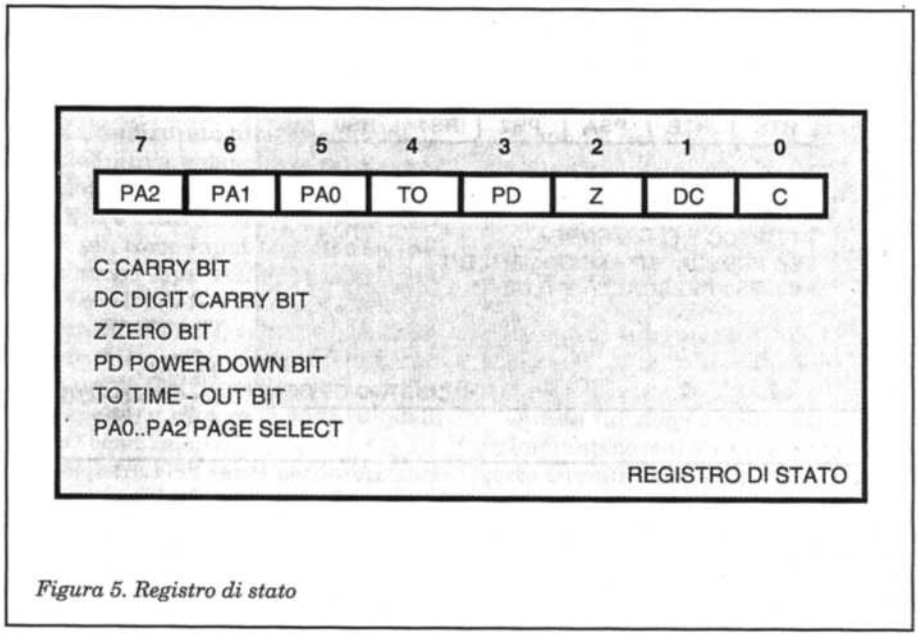


Figura 5. Registro di stato

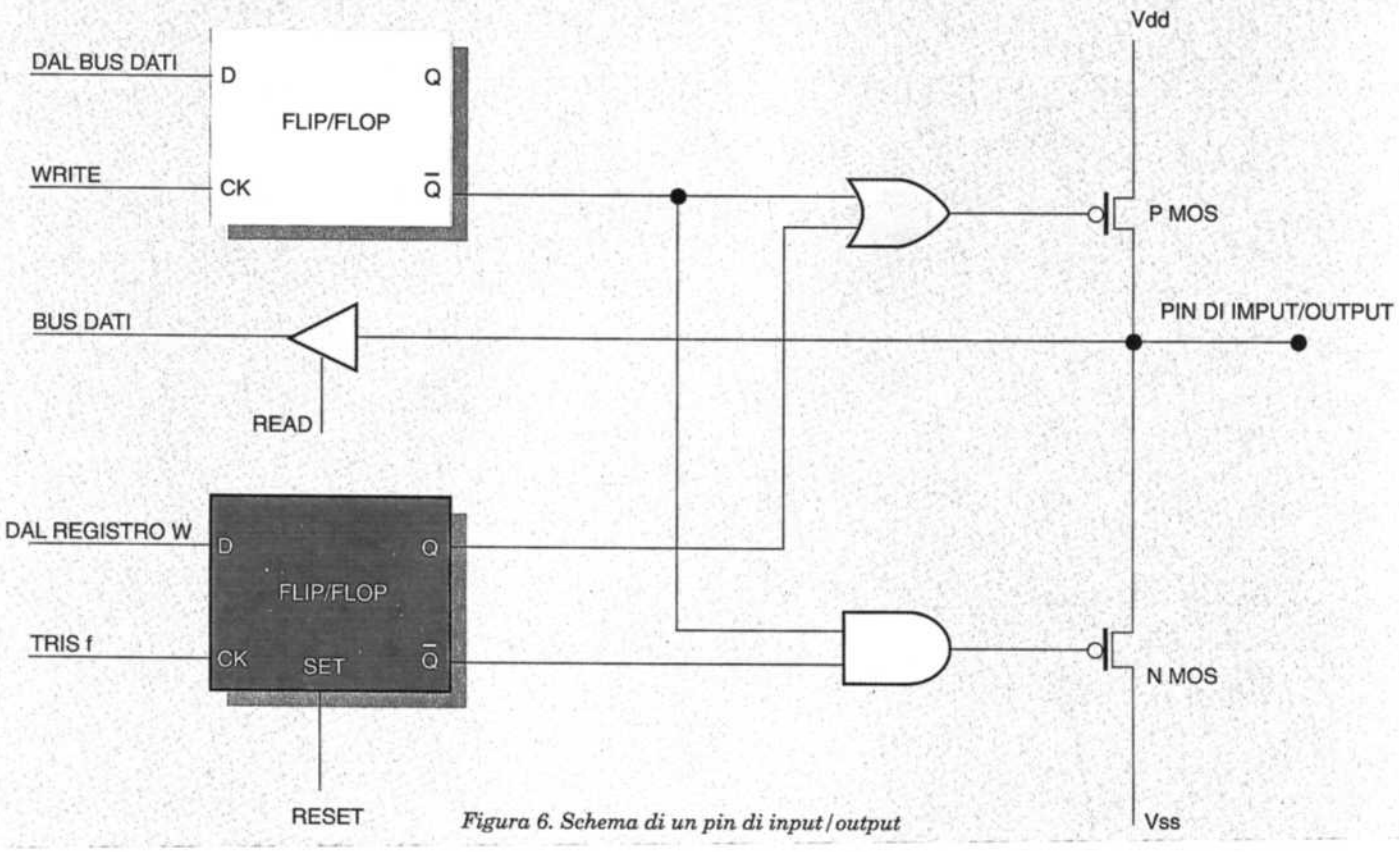


Figura 6. Schema di un pin di input/output

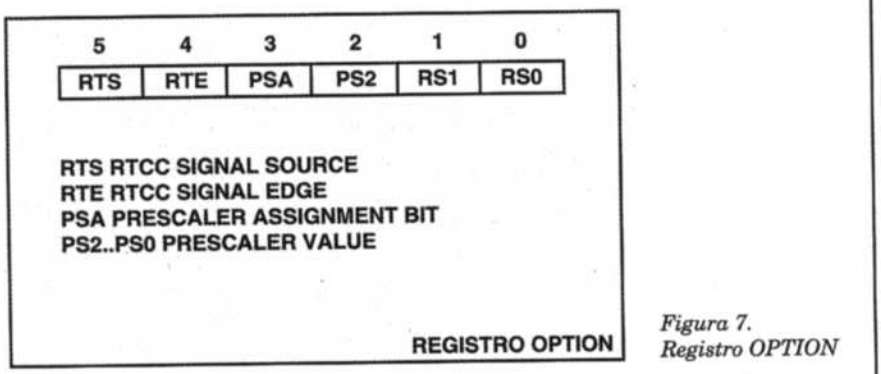


Figura 7. Registro OPTION

è possibile vedere nella mappa della memoria di Figura 4: per indirizzare tutti e quattro i banchi del PIC16C57 sono necessari due bit in più rispetto agli altri device. Tornando alla Figura 2, possiamo notare che dopo la decodifica è possibile lavorare con i registri generali oppure con il registro W (Word). Quest'ultimo rappresenta uno dei registri più frequentemente usati dal programmatore poiché ogni "valore" deve passare da lì.Ad esempio, per fare una somma o una divisione attraverso la ALU (Aritmetic Logic Unit) è necessario passare per W.

La ALU è la componente che esegue le operazioni logiche.

Lo stato successivo a ogni operazione del PIC viene memorizzato nel registro di STATO, visibile in Figura 5.

I tre bit PA0-PA2, servono per selezionare i banchi possibili dei registri dei vari PIC (lo vedremo in seguito); il bit di time-out (TO) viene settato a 1 durante il reset di alimentazione e dalle istruzioni CLRWDT e SLEEP, mentre viene resettato da un time-out del registro WATCHDOG.

Il bit di POWER DOWN (PD), invece, è settato a 1 dalla sola istruzione CLRWDT e resettato dall'istruzione SLEEP. Il bit ZERO Z, è settato a 1 solo se il risultato dell'ultima operazione eseguita è zero. I due bit DIGIT CARRY (DC) e CARRY (C) sono, invece, settati se sopravviene un overflow. Sempre in Figura 2, alla ALU sono connessi i registri F5, F6, F7 e i rispettivi TRIS_A, TRIS_B e TRIS_C.

Questi registri controllano lo stato delle porte di ingresso/uscita e in Figura 6 possiamo vedere come sono strutturati i pin del PIC.

Il flip-flop in alto corrisponde a un bit del registro relativo a una porta (ad esempio la A), mentre quello in basso corrisponde a un bit del registro TRIS_A (sempre in riferimento alla porta A).

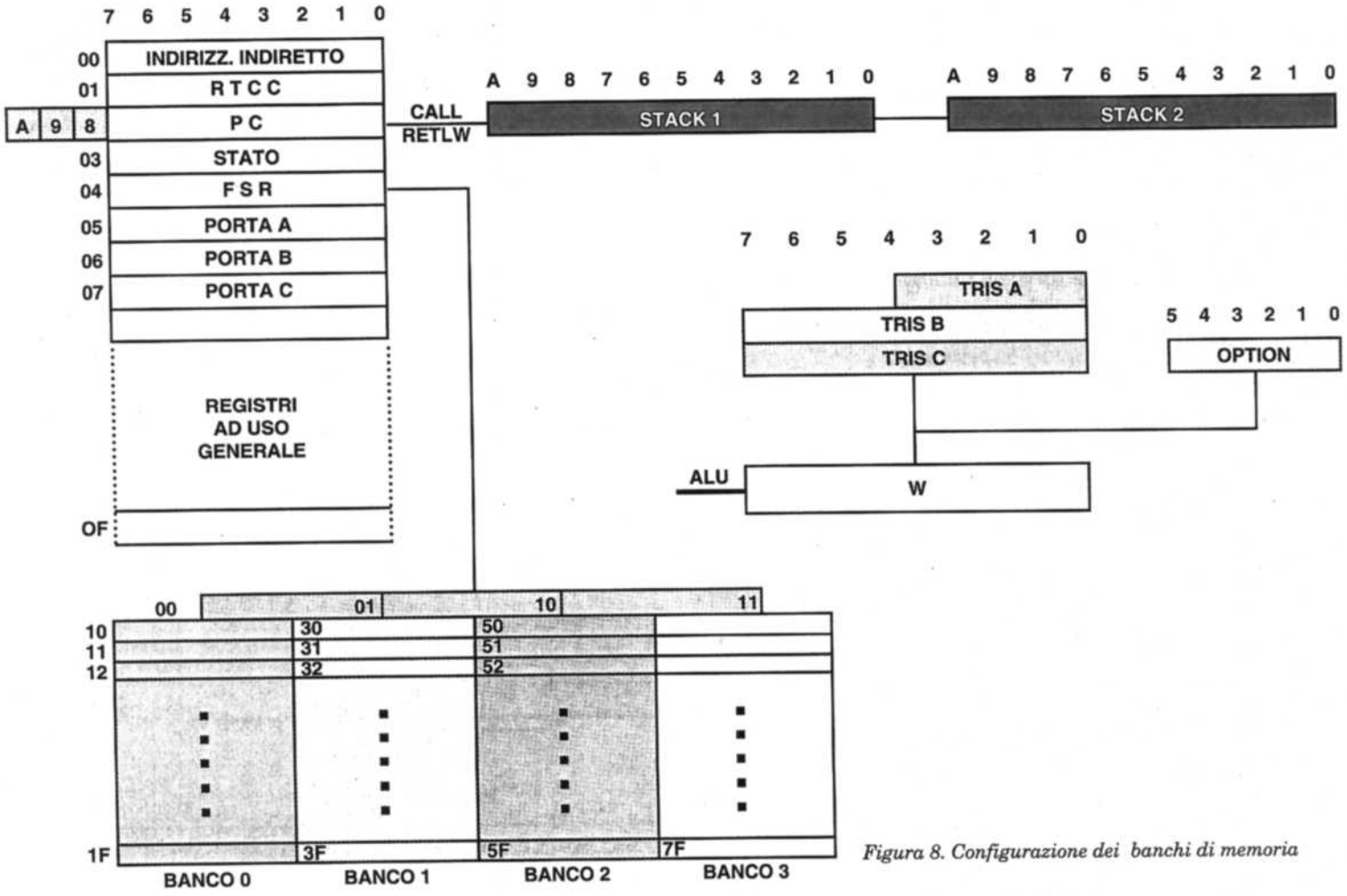


Figura 8. Configurazione dei banchi di memoria