HOME COMPUTER AMICO 2000

a cura della A.S.E.L. s.r.l. - parte ottava

HARDWARE

Uso della porta utente

Nell'AMICO 2000A c'è un integrato che gestisce le operazioni di input e output: si tratta del dispositivo 8255 il cui schema appare in Fig. 1.

Questo integrato dispone di tre *porte* ovvero di tre mezzi per comunicare col mondo esterno costituiti da 8 bit ciascuno Di queste tre porte A, B e C, le prime due sono utilizzate per la gestione del display e della tastiera e sono controllate dal programma di monitor, la C invece è detta "PORTA UTENTE" perché utilizzabile a nostro piacimento. Si tratta solo di sapere come.

Ognuna di queste tre porte ha un indirizzo della locazione di memoria nella quale il monitor (per la A e B) o l'utilizzatore (per la C) scrive una ben determinata parola (ovvero 8 bit) che fa funzionare queste porte in un altrettanto ben determinato modo; vediamo quali sono gli indirizzi di memoria interessati dall'8255:

Indirizzo della Porta A = FD00
Indirizzo della Porta B = FD01
Indirizzo della PORTA C (PORTA UTENTE) = FD02

La porta utente dell'8255 può essere utilizzata previa programmazione del modo di funzionamento: per far ciò esiste un *registro di definizione del modo di funzionamento* che nel nostro caso ha indirizzo FD03.

Fisicamente la porta utente C sono nell'AMICO 2000A quegli otto capicorda posti in alto e in mezzo nella piastra nel quale sono presenti quegli otto bit paralleli (nel senso che si leggono o si scrivono allo stesso tempo) di cui abbiamo parlato.

Prima di passare ad esaminare come si usa il registro di definizione premettiamo che la porta C è divisa in due parti: la porta CH (H sta per High = alto) ovvero i bit 7, 6, 5, 4 e la porta CL (L sta per Low = basso) ovvero i bit 3, 2, 1, 0 come mostra la Fig. 2.

Queste due parti della porta C hanno la particolarità di poter funzionare una in uscita e una in entrata (indifferentemente CH o CL) oppure tutte e due in uscita o in entrata; questo naturalmente dipende da ciò che abbiamo programmato cioè da cosa viene scritto nel registro che definisce il modo di funzionamento.

Detto questo vediamo subito che scrivere una parola nel registro di definizione del modo di funzionamento, ovvero all'indirizzo FD03, non significa altro se non determinare la funzione delle varie porte e cioé se la CPU in queste porte deve leggere un dato che le viene presentato (condizione della porta di input) o se essa stessa deve presentare un dato sulle porte (condizione di output).

La Fig. 3 rappresenta il registro del modo di funzionamento; notiamo che i bit sulle posizioni 2, 5, 6 e 7 sono "fissi" ovvero sono stati scritti così dal nostro programma di monitor perché l'8255 potesse lavorare nel progetto dell'AMICO 2000A in un certo modo.

Vediamo come usare (programmare) gli altri bit:

B4 corrisponde alla porta A: si scrive 0 in Output e 1 in Input
B1 corrisponde alla porta B: si scrive 0 in Output e 1 in Input
B3 corrisponde alla porta CH: si scrive 0 in Output e 1 in Input
B0 corrisponde alla porta CL: si scrive 0 in Output e 1 in Input

Come esempio nella Fig. 4 sono riportate le configurazioni utilizzate dal monitor che sono:

89 per eseguire la routine del display
99 per eseguire la routine della tastiera.

Per lavorare con la porta C si dovrà agire allora nel seguente modo:

1) Decido come usare la porta C scrivendo una parola di definizione alla locazione FD03 (decido cioé se utilizzarla come uscita o ingresso dati);

2) Scrivo (se la porta C è stata posta in Output) o leggo (se la porta C è stata posta in input) il dato nella locazione FD02 (che è appunto l'indirizzo relativo alla porta C).

È possibile usare la porta C indirizzando uno per uno, ovvero singolarmente, i bit della porta utente. Per far questo ci si serve soltanto del registro di definizione (non della locazione relativa alla porta C) e si procede come segue.

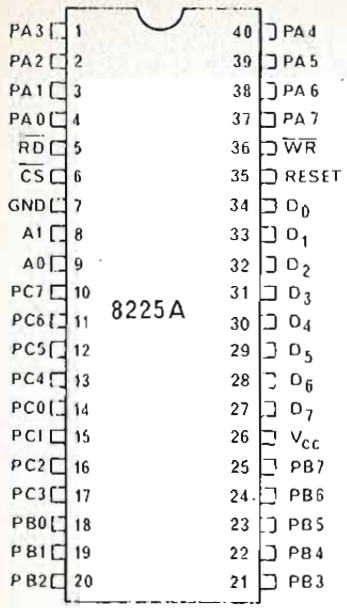
Prima di tutto si definisce il modo di funzionamento delle varie porte scrivendo una opportuna parola nella locazione FD03.

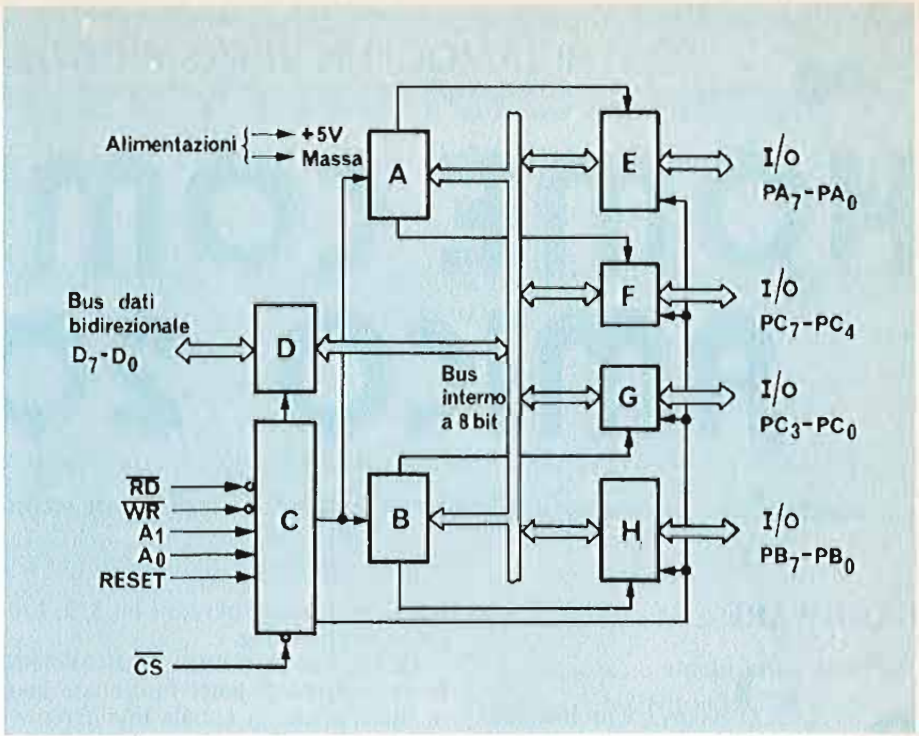
Poi e questa è una particolarità dell'integrato 8255, scrivo ancora nella locazione FD03 una configurazione (una parola) tale da permettermi l'operazione desiderata secondo la tabella che segue e con riferimento alla fig. 5.

B7 = 0 con ciò seleziono questo particolare modo di funzionamento che mi permette di usare la stessa locazione FD03.

B6 - B5 - B4 = 0 o 1 è indifferente nulla cambia.

CONFIGURAZIONE
PIEDINI 8255A





A = Controllo gruppo A; B = Controllo gruppo B; C = Logica di controllo lettura-scrittura; D = Buffer dati; E = Porta A (8 bit); F = Porta CH (parte alta 4 bit); G = Porta CL (parte bassa 4 bit); H = Porta B (8 bit).

Fig. 1 - Schema a blocchi dell'integrato 8255A.

FUNZIONE DEI PIEDINI

D_7-D_0	Bus dati bidirezionale
RESET	Ingresso di azzeramento
$\overline{CS}$	Selezione
$\overline{RD}$	Ingresso impulso di lettura
$\overline{WR}$	Ingresso impulso di scrittura
A0,A1	Indirizzi delle porte
PA7-PA0	Linee di uscita della Porta A
PB7-PB0	Linee di uscita della Porta B
PC7-PC0	Linee di uscita della Porta C

Ora scriverò nella stessa locazione FDØ3, secondo quanto abbiamo spiegato poc'anzi, prima una parola che metta a 1 il bit 6, poi (e qui il tempo di durata dell'impulso dipende dal tempo di esecuzione della istruzione) una parola che metta a Ø lo stesso bit 6 della porta C: si faccia riferimento alla Fig. 7.

Compreso tutto ciò posso scrivere il programma come segue:

Da notare che in questo programma non si effettua alcuna operazione di "store" all'indirizzo della porta C (FDØ2), infatti in questo modo di funzionamento, come abbiamo detto, si opera esclusivamente con il registro di definizione del modo di funzionamento alla locazione FDØ3.

Con questo programma si ottiene un impulso di circa 10 μS (cioé il tempo di

B3 - B2 - B1 = ØØØ = Uscita sul bit Ø
ØØ1 = Uscita sul bit 1
ØlØ = Uscita sul bit 2
Øll = Uscita sul bit 3
lØØ = Uscita sul bit 4
lØl = Uscita sul bit 5
llØ = Uscita sul bit 6
lll = Uscita sul bit 7

BØ = 1 se si vuole scrivere un 1 sul bit scelto.

BØ = Ø se si vuole scrivere uno Ø sul bit scelto.

Facciamo, per passare alla pratica, il seguente esempio. Si voglia realizzare un programma che generi un impulso al bit 6 della porta C.

Innanzitutto programmeremo il modo di funzionamento in maniera che la porta CH (perché il 6° bit è in questa parte della porta C) si trovi in condizione di Output. Scriveremo allora, ad esempio il dato 81 nella locazione di memoria FDØ3 ponendoci nelle condizioni illustrate dalla Fig. 6.

Selezione del bit della porta C su cui si intende operare

ØØØØ	A9	LDA #$81	
1	81		
2	8D	STA $FDØ3	Definizione del modo di funzionamento
3	Ø3		
4	FD		
5	A9	LDA #$ØD	
6	ØD		
7	8D	STA $FDØ3	Metto a 1 il bit 6 (Set del bit 6)
8	Ø3		
9	FD		
A	A9	LDA #$ØC	
B	ØC		
C	8D	STA $FDØ3	Metto a Ø il bit 6 (Clear del bit 6)
D	Ø3		
E	FD		
F	4C	JMP Monitor	
Ø21Ø	22		
1	FE		

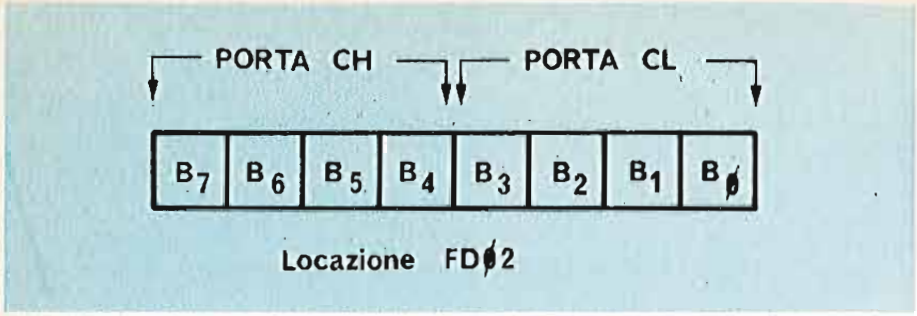


Fig. 2 - Configurazione della porta utente C all'indirizzo FDØ2.

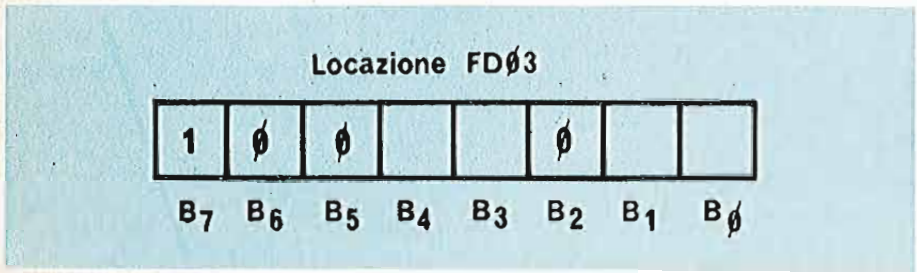


Fig. 3 - locazione FDØ3 del registro del modo di funzionamento.

durata di esecuzione delle istruzioni) al bit 6 della porta C ogni volta che facciamo partire il programma.

Se si desidera ottenere un tempo di durata dell'impulso più lungo è necessario inserire un loop di ritardo tra il *Set* (portare a 1) e il *Clear* (portare a Ø) del bit. Se si vuole poi che l'impulso sia ripetitivo si utilizza l'istruzione di JMP; vediamo di seguito il programma arricchito di queste altre prestazioni:

```
0200        LDA #$81
  1         81
  2         8D   STA FDØ3     } Definizione del modo di funzionamento
  3         Ø3
  4         FD
  5         A2   LDX #$ØØ     Preparo il loop di ritardo caricando ØØ nel
  6         ØØ                registro X
  7 Onda    A9   LDA #$ØD     }
  8         ØD                }
  9         8D   STA FDØ3     } Set del bit 6
  A         Ø3                }
  B         FD                }
  C Loop1   CA   DEX          Decremento X(1)
  D         DØ   BNE Loop1    Salto al Loop 1 quando X ≠ Ø(2)
  E         FD
  F         A9   LDA #$ØC     }
0210        ØC                }
  1         8D   STA FDØ3     } Clear del bit 6
  2         Ø3                }
  3         FD                }
  4 Loop2   CA   DEX          } Ripeto la routine di ritardo(3)
  5         DØ   BNE Loop2    }
  6         FD                }
  7         4C   JMP Onda     } Ritorno all'inizio del set bit 6 per generare
  8         Ø7                } un'onda quadra continua
  9         Ø2                }
```

Note: 1) Dopo il primo decremento il contenuto di X è = FF (infatti Ø - 1 = -1 = = FF); 2) Continuo a decrementare fino a che X = Ø. Questo avviene dopo 256 decrementi. Durante questo periodo di tempo il bit 6 è a 1; 3) Nota bene: entriamo in questo Loop con X = Ø (risultato del loop precedente).

La durata di ogni semionda si calcola dal tempo impiegato dal microprocessore ad eseguire le istruzioni DEX e BNE (4μS in totale) moltiplicato per il numero dei loop (256); allora 4 x 256 = 1,024 mS, cioè una frequenza di circa 500 Hz.

Si può naturalmente diminuire il tempo di ciclo caricando un valore minore da decrementare o aumentarlo servendosi di due loop concatenati che fanno uso dei due registri X e Y:

```
      LDY #$yy   (yy = numero che
      LDX #$ØØ        determina il
Loop  DEX              ritardo)
      BNE Loop
      DEY
      BNE Loop
```

Il ritardo generato da questa routine è uguale a 1,024 msec moltiplicato per il numero caricato nel registro Y all'inizio della routine; il tempo di ritardo massimo è quindi di circa 262 millisecondi (1,024 x 256).

Facendo girare il programma appena descritto sull'AMICO 2000A è possibile vedere l'onda quadra generata se si è in possesso di un oscilloscopio collegando il probe al piedino 6 della porta utente.

Se viene inserito un ritardo maggiore nel programma possiamo osservare con un semplice voltmetro (tester commutato sui 10 Vcc fondo scala) una tensione che varia periodicamente fra il piedino 6 e massa.

Una volta che si è in possesso delle tecniche per lavorare sulla porta utente, non esistono più problemi a collegarsi all'esterno a qualsiasi altro dispositivo elettronico in grado di presentare alla sua uscita livelli alti e bassi di tensione o di saperli leggere.

Le subroutine del monitor

Nel programma di Monitor, quello che permette al microelaboratore di funzionare, sono impiegate alcune routine che possono essere convenientemente utilizzate nei vostri programmi.

Vediamo ora quali sono e come si usano.

Routine di TASTO ATTIVO

Routine di IDENTIFICAZIONE DEL TASTO

Routine 1 di RINFRESCO DISPLAY

Routine 2 di RINFRESCO DISPLAY

A) *Routine di tasto attivo* - Ha come indirizzo di partenza la locazione FEEB. Si esce dalla subrotine con ØØ in Accumulatore se non vi è alcun tasto premuto e con Ø1 in Accumulatore se c'é qualche tasto premuto.

Questa subroutine si usa per far succedere qualcosa quando si preme un tasto qualunque. Alla fine di questa subroutine si fa un test sul contenuto dell' accumulatore utilizzando una istruzione di BEQ o BNE.

Normalmente però è più utilizzata la routine che segue, di identificazione del

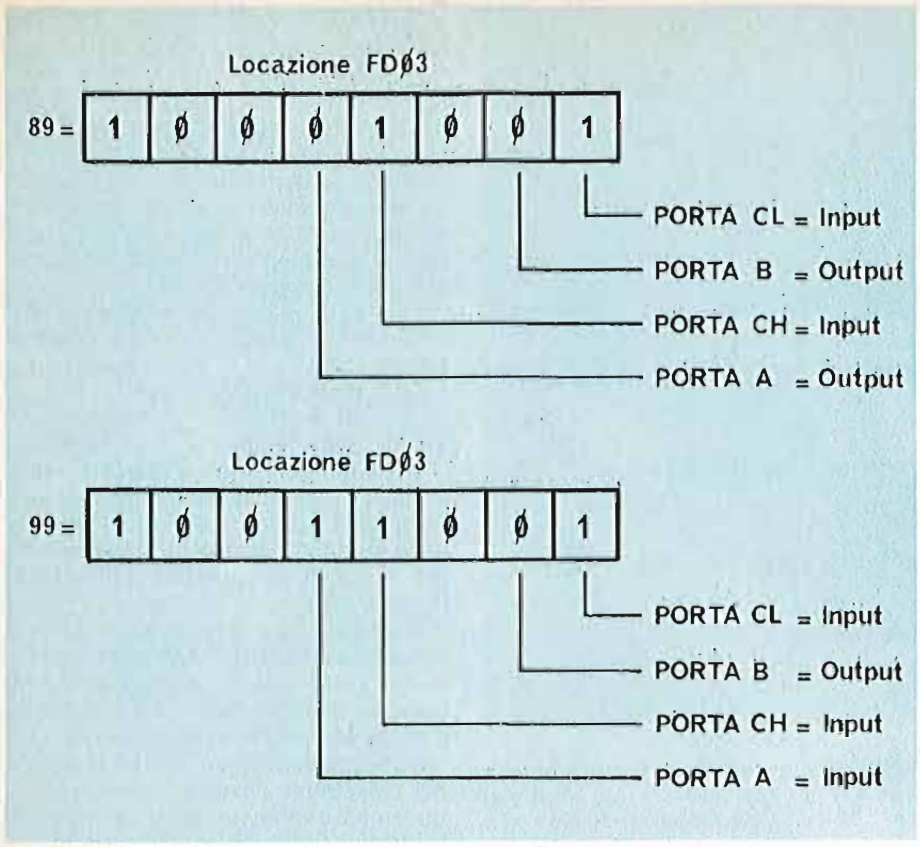


Fig. 4 - Due configurazioni del registro del modo di funzionamento utilizzate dal programma di monitor per eseguire la routine del display e quella della tastiera.

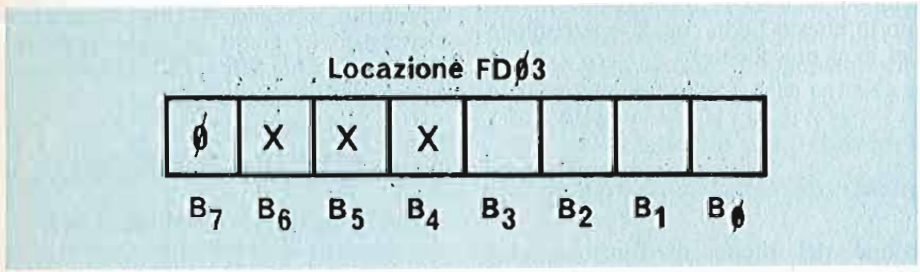


Fig. 5 - Uso del registro del modo di funzionamento per usare la porta C indirizzando i suoi bit uno per uno.

tasto, perché permette di assegnare ad ogni tasto una determinata funzione da eseguire.

B) *Routine di identificazione del tasto* - Ha come indirizzo di partenza la locazione FF57. Prima di entrare in questa routine è bene inizializzare l'integrato 8255 per essere certi che esso sia in grado di trattare i segnali così come la routine di identificazione del tasto richiede; per far questo basta caricare il numero 99 nel registro di definizione del modo di funzionamento.

Si possono usare le seguenti istruzioni:

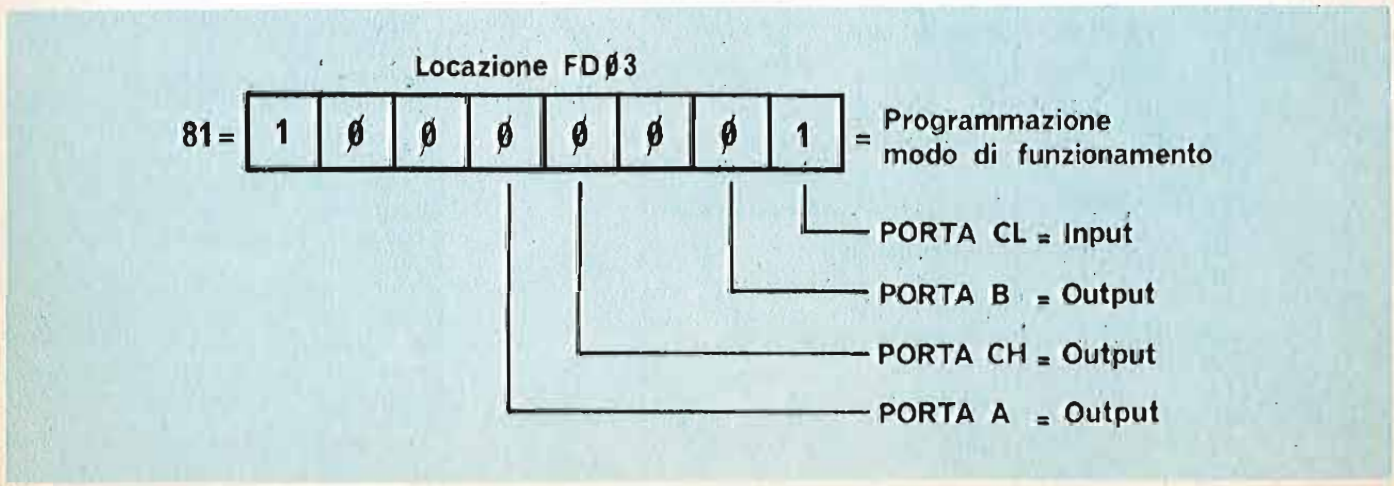
```
A9      LDA     #$99
99
8D      STA     $ FDØ3
Ø3
FD
```

Si esce dalla routine con il valore del tasto contenuto in Accumulatore secondo quanto riportato nella Tabella 1. Nella tabella sono riportate due serie di valori diversi per ogni tasto che dipendono dal modo di funzionamento in cui è stato posto il processor: decimale (mediante l'istruzione SED) o esadecimale (mediante l'istruzione CLD). Se non vi è alcun tasto premuto o più di un tasto premuto il contenuto dell'accumulatore vale 21 se in funzionamento decimale o 15 se in funzionamento esadecimale. Per usare questa routine si fa una comparazione in uscita con il numero corrispondente al tasto che interessa seguita da una istruzione di Branch.

C) *Routine I rinfresco display* - Ha come indirizzo di partenza la locazione FFØ6.

Durante l'esecuzione la routine preleva il contenuto esadecimale della locazione di memoria indirizzata dalle loca-

Fig. 6 - Programmazione del modo di funzionamento per porre la porta CH in condizione di Output.



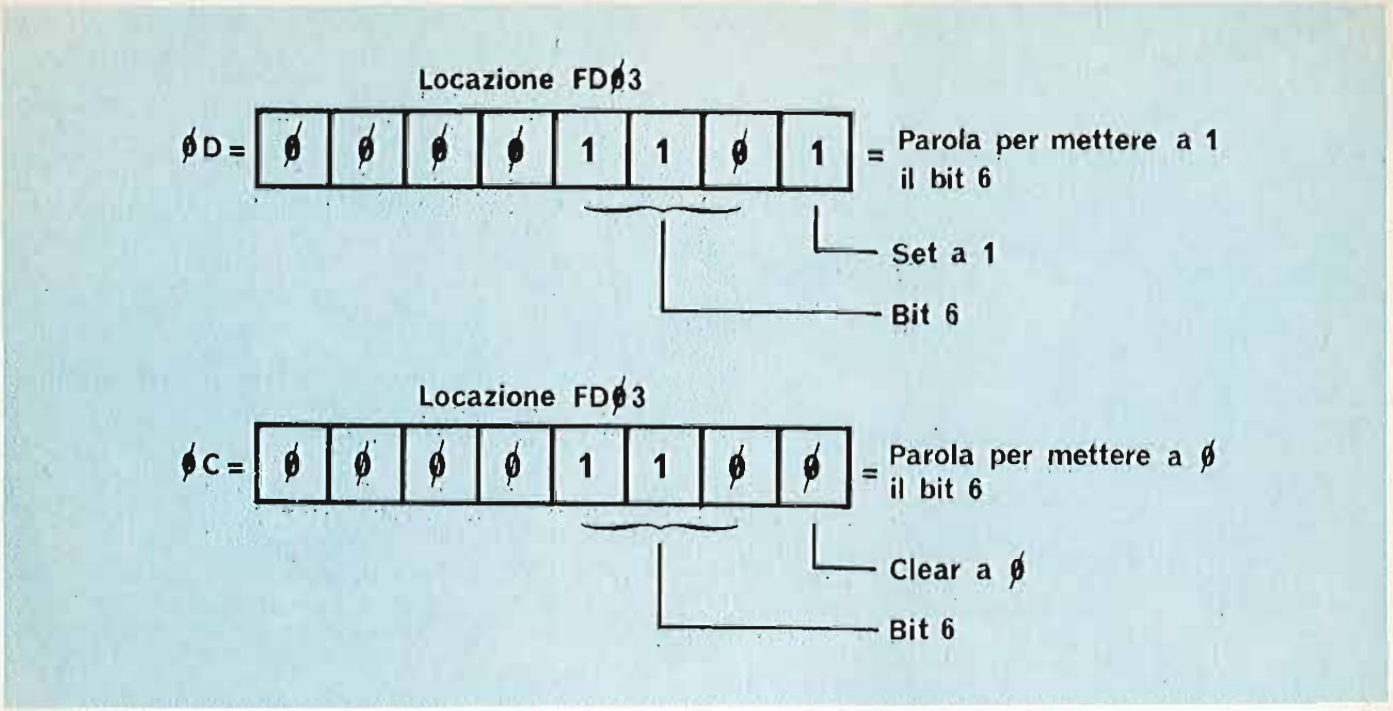


Fig. 7 - Le due parole ØD e ØC sscritte alla locazione FDØ3 servono a portare a 1 o a Ø il bit 6 della porta C.

zioni di pagina base ØØFA (parte bassa) e ØØFB (parte alta). In altre parole il contenuto delle locazioni ØØFA e ØØFB diventa l'indirizzo di una locazione di cui si vuole vedere il contenuto.

Per utilizzare convenientemente questa routine è evidente che è necessario caricare in precedenza queste locazioni di memoria.

Per utilizzare in pratica questa routine facciamo l'esempio di un programma che carica sul display indirizzi la locazione Ø2Ø2. È chiaro che sul display apparirà il contenuto di memoria relativo all'indirizzo Ø2Ø2; vogliamo poi che questo contenuto sia Ø4 (vedi Listing).

Attenzione! A questo punto il programma sembrerebbe terminato, ma se si facesse partire l'esecuzione vedremmo un lampo di luce nel display e poi più niente.

Infatti per mantenere la configurazione di cifre fissa sul display è necessario "rinfrescare" continuamente il display stesso: si completa perciò il programma con una istruzione di JMP che fa ripetere in continuazione la subroutine:

```
Ø2ØE      4C  JMP Ø2ØB
   F      ØB
Ø21Ø      Ø2
```

Ora si può far partire il programma.

Vediamo ora in particolare come funziona questa subroutine: viene prelevato il contenuto delle locazioni di memoria puntate da ØØFA e ØØFB che viene trasportato nella locazione ØØF9. Successivamente il contenuto delle locazioni di memoria ØØF9, ØØFA e ØØFB viene trasferito sul display come mostra la Fig. 8.

Listing

```
Ø2ØØ      A9  LDA #$Ø2      Carico Ø2 nell'accumulatore
   1      Ø2
   2      85  STA $FA       Memorizzo Ø2 alla locazione ØØFA
   3      FA
   4      85  STA $FB       Memorizzo Ø2 alla locazione ØØFB
   5      FB
   6      A9  LDA #$Ø4      Carico Ø4 in ACC
   7      Ø4
   8      8D  STA $Ø2Ø2      Memorizzo Ø4 nella locazione Ø2Ø2
   9      Ø2
   A      Ø2
   B      2Ø  JSR FFØ6      Salto alla subroutine 1 di rinfresco del display
   C      Ø6
   D      FF
```

Tabella 1 - Dati presenti in Accumulatore al ritorno della subroutine di identificazione dei tasti secondo il tasto premuto.

TASTO	DATO[1]	TASTO	DATO[2]	DATO[3]
0	ØØ	A	1Ø	A
1	Ø1	B	11	B
2	Ø2	C	12	C
3	Ø3	D	13	D
4	Ø4	E	14	E
5	Ø5	F	15	F
6	Ø6	AD	16	1Ø
7	Ø7	DA	17	11
8	Ø8	↑	18	12
9	Ø9	RUN	19	13
		REG	2Ø	14
		NESSUN TASTO	21	15

[1] = Funzionamento decimale e esadecimale
[2] = Funzionamento decimale
[3] = Funzionamento esadecimale

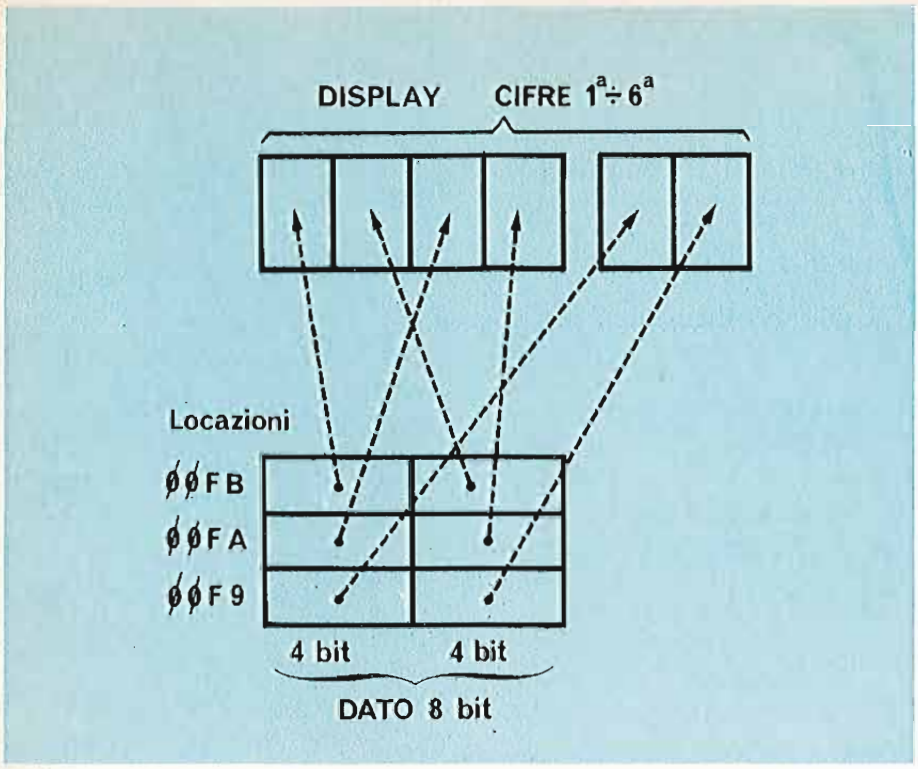


Fig. 8 - Corrispondenza fra contenuto delle locazioni in pagina base F9, FA e FB e le sei cifre del display.

In questa stessa routine di rinfresco display si può entrare alla locazione FF0C invece che alla FF06, saltando in questo modo il prelevamento del dato e la modifica del contenuto di 00F9. In questo modo possiamo visualizzare ciò che vogliamo direttamente sui sei digit del display.

D) *Routine 2 di rinfresco display* - Ha come indirizzo di partenza la locazione FF7E.

Questa routine porta sul display il contenuto delle sei successive locazioni di memoria così come segue:

008F	1ª cifra a sinistra del display
0090	2ª cifra
0091	3ª cifra
0092	4ª cifra
0093	5ª cifra
0094	6ª cifra

Il caricamento di queste locazioni di memoria deve essere fatto tenendo conto della corrispondenza esistenze tra i singoli bit e i segmenti delle cifre del display secondo quanto riportato nella Fig. 9. Il bit a 1 corrisponde a segmento acceso.

Nella tabella 2 inoltre è riportato un esempio di alfabeto che può essere visualizzato dall'AMICO 2000A. È chiaro che ognuno poi potrà scrivere i segni più strani scrivendo delle parole opportune che mettano a 1 i bit corrispondenti segmenti che si vuole far accendere. Si fa notare infine che il bit 7 non viene utilizzato.

Vediamo subito un programma esemplificativo che presenti sul display: ASEL -.

Facciamo riferimento alla Tabella 2 e carichiamo i dati:

Locazione	Dato	Rappresentazione
008F	C0	-
0090	F7	A
1	ED	S
2	F9	E
3	B8	L
4	C0	-

Scriviamo ora il seguente semplice programma di uso generale che permette di visualizzare ciò che abbiamo scritto nelle diverse locazioni:

0200	20	JSR FF7E
		Salto alla subroutine di rinfresco
1	7E	
2	FF	
3	4C	JMP 0200
		Eseguo in continuazione la subroutine precedente
4	00	
5	02	

Tabella 2 - Esempio di alfabeto per display a sette segmenti e dati relativi per generare quel carattere.

Numeri	Lettere	
1 = 86	A = F7	n = 54
2 = DB	b = 7C	o = BF
3 = CF	C = B9	P = F3
4 = E6	d = 5E	q = 67
5 = ED	E = F9	r = 50
6 = FD	F = F1	S = ED
7 = 87	G = BD	t = 78
8 = FF	H = F6	U = BE
9 = EF	I = 86	Y = EE
0 = BF	J = 9E	Cifra
- = C0	L = B8	Spenta = 00

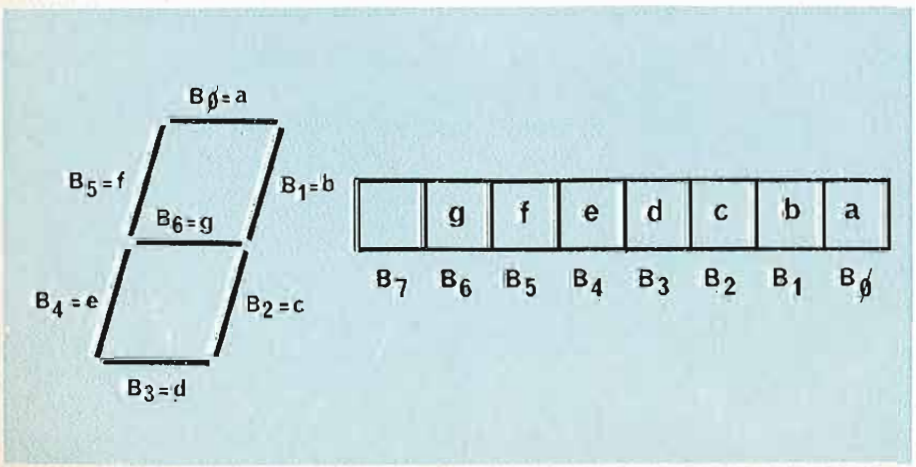


Fig. 9 - Corrispondenza fra i vari segmenti di ogni cifra e i bit della locazione di memoria ad essa relativa.

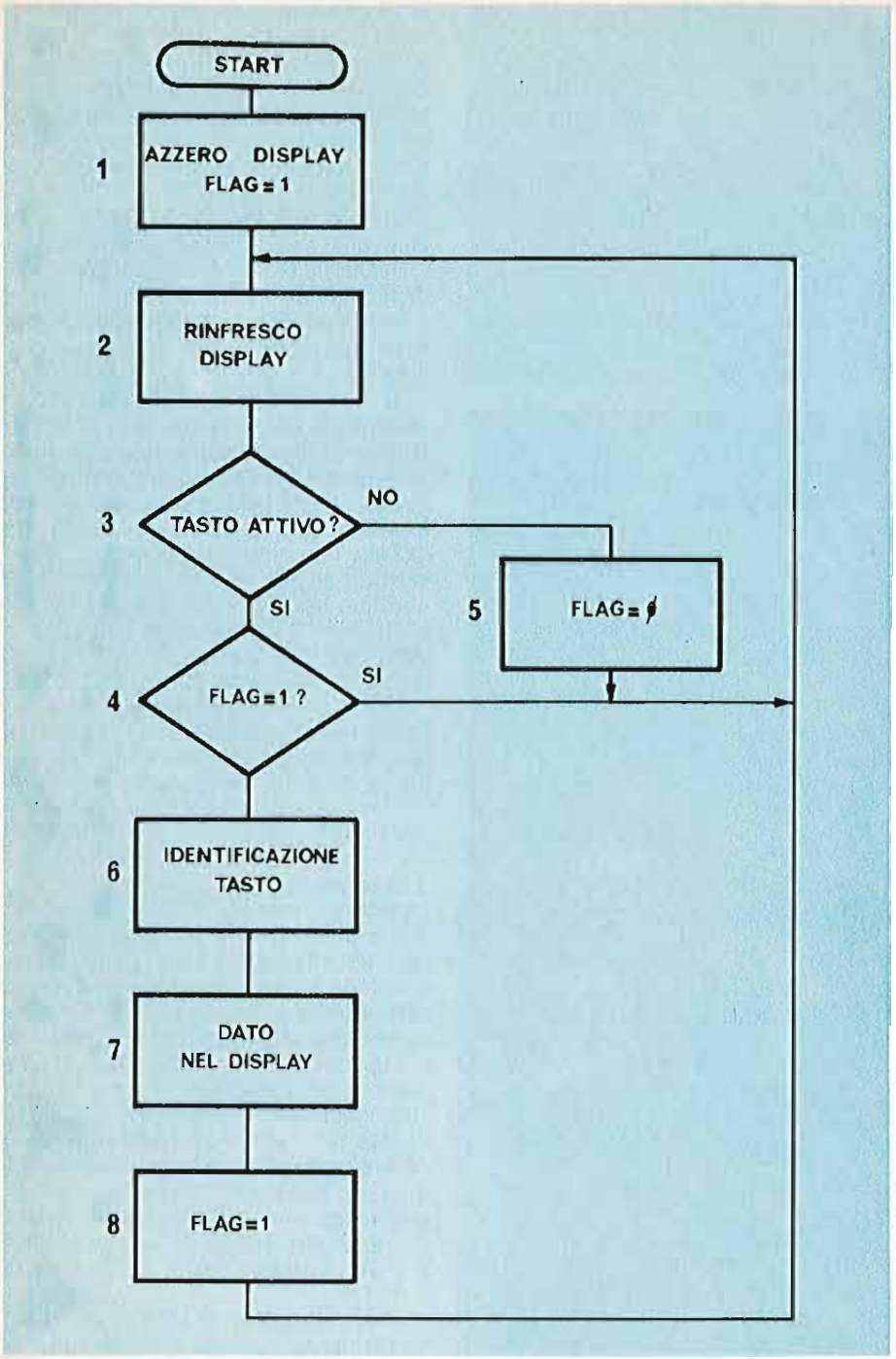


Fig. 10 - Flow chart programma per l'uso delle routine di identificazione del tasto e di tasto attivo.

Vediamo ora un programma che mostri l'uso della routine di identificazione del tasto.

Ø2ØØ		F8	SED	Metto la macchina in modo decimale
1	Loop	2Ø	JSR FF57	Salto alla routine di identificazione del tasto
2		57		
3		FF		
4		C9	CMP = $Ø5	Confronto il contenuto dell'ACC con Ø5
5		Ø5		
6		DØ	BNE Loop	Se non è uguale a Ø5 salto al Loop
7		F9		
8		4C	JMP Monitor	Se è uguale a Ø5 passo il controllo del micro al
9		22		monitor (il display si accende)
A		FE		

Facendo partire questo programma il display rimarrà spento finché non si preme il tasto 5, solo in questo caso si salta il Monitor che riprende il controllo del display facendolo accendere.

Ora esamineremo un programma più complesso e completo che mette in evidenza l'uso delle due routine di tasto attivo e di identificazione del tasto. Con questo programma vogliamo che il display mostri tutte le sue cifre a zero e che sul display compaia il valore dei tasti da Ø a F quando questi vengano premuti. Prima di dare la lista delle istruzioni è opportuno esaminare il flow chart per meglio comprenderne la costruzione (Fig. 10).

Blocco 1: Si azzera il display e si pone un FLAG a 1, con questa operazione in pratica si memorizza il dato Ø1 in una certa locazione di memoria: lo scopo, come vediamo più avanti, è quello di controllare se in questa locazione di memoria c'è 1 o Ø, cioè se è stata modificata in qualche modo.

Blocco 2: Si salta alla routine di rinfresco del display.

Blocco 3: Da questo blocco e fino al 5 il programma che è stato scritto ha la funzione di permettere da una parte che l'uso del tasto RUN che fa partire il programma non venga identificato, (cioè nel pur brevissimo tempo durante il quale questo tasto rimane premuto il programma continua a rinfrescare il display azzerato), dall'altra che ogni tasto premuto venga identificato una sola volta come spieghiamo più avanti.

Quando allora entriamo nel blocco 3 con il tasto RUN premuto alla domanda "TASTO ATTIVO?" si esce con SI e si entra nel blocco 4.

Blocco 4: Qui ci si chiede se il FLAG è a 1: tutte le volte che passiamo al blocco 3 mentre RUN rimane premuto il FLAG sarà ancora a 1 in quanto niente lo ha modificato. In questo caso si esce dal SI e si torna alla routine di rinfresco del display che mostra ancora tutti zeri.

Blocco 5: Passando attraverso il blocco 3 dopo aver rilasciato il tasto RUN nessun tasto sarà attivo e quindi si uscirà dal NO entrando nel blocco 5 che pone il FLAG a Ø e torna a rinfrescare il display.

A questo punto, posto il FLAG a Ø, quando andiamo a premere uno dei tasti della tastiera esadecimale entriamo nel blocco 3, usciamo con SI entrando nel blocco 4 e da questo usciamo con NO entrando nel blocco successivo.

Blocco 6: Si entra nella routine di identificazione del tasto.

Blocco 7: Il dato (cioè il valore dello stesso tasto) viene visualizzato nel display e posto nella prima cifra a destra.

Blocco 8: Il FLAG viene ripristinato a 1 per far in modo che il tasto identificato e che abbiamo premuto (ad esempio il 5) venga riconosciuto una sola volta. In pratica infatti anche nel pur

breve tempo durante il quale il tasto rimane premuto il programma gira numerosissime volte, se non mettessimo ancora il FLAG a 1 il numero 5 corrispondente al tasto premuto riempirebbe tutte le sei cifre del display invece dell'ultima a destra. Quando però il tasto 5 viene rilasciato ecco che il FLAG va nuovamente a zero e il programma è pronto ad accettare un nuovo tasto.

In pratica le cifre che inseriremo, ovvero i tasti che premeremo entreranno dall'ultima cifra a destra e scorreranno fino alla prima a sinistra fino a sostituire tutti gli zeri del display.

Cercate di comprendere bene l'uso del FLAG perché questo artificio viene usato molto di frequente nella compilazione dei programmi.

Ora possiamo scrivere il programma; badate che questo programma contiene diversi passaggi interessanti dal punto di vista ottimizzazione del numero delle istruzioni impiegate e va quindi studiato con attenzione.

Per meglio comprendere tutti i passaggi di questo programma esaminiamo di seguito queste note:

1) Uscendo da questa routine senza alcun tasto premuto nell'accumulatore si ha 0, uso questo stesso 0 per azzerare il FLAG. Se invece c'è qualche tasto premuto continua l'esecuzione del programma e si va a controllare il FLAG.

2) Carico il registro X con il valore contenuto nella locazione 0000 (il FLAG).

3) Questo blocco di istruzioni permette di far entrare nell'ultima cifra a destra il tasto premuto: viene eseguito quattro volte perché ogni cifra è composta da 4 bit (es.: 5 = 0101).

4) In questo modo carico nell'Accumulatore la nuova parola da inserire nella locazione di pagina base F9 (cioè nelle ultime due cifre a destra del display). Per una miglior comprensione si veda la Fig. 11.

5) Le due istruzioni CLC e BCC equivalgono ad un salto incondizionato, le abbiamo usate al posto del JMP perché diversamente il programma non sarebbe stato rilocabile (ovvero sarebbe stato legato alla locazione di partenza 0200).

Facciamo notare che saltando ad IND2 (locazione 0206) realizziamo un notevole risparmio di istruzioni ripassando su passi di programma già scritti.

```
0200          A9  LDA  #$00      Carico 00 in Acc.
   1          00
   2          85  STA  $FB  ┐
   3          FB             │
   4          85  STA  $FA  ├   Azzero il display
   5          FA             │
   6  IND 2   85  STA  $F9  ┘
   7          F9
   8          A9  LDA  #$01  ┐
   9          01             ├  FLAG = 1
   A  IND 1   85  STA  $00   ┘
   B          00
   C  LOOP    20  JSR  SCANS     Rinfresco display
   D          0C
   E          FF
   F          20  JSR  TESTAS    Salto alla routine di tasto attivo
0210          EB                 (MONITOR)
   1          FE
   2          F0  BEQ  IND 1     Se ACC = 0 non vado avanti, ma
   3          F6                 azzero il FLAG (1)
   4          A6  LDX  $00       Vedo se il FLAG = 0 (2)
   5          00
   6          D0  BNE  LOOP      Se non è uguale a 0 torno in Loop
   7          F4                 diversamente continuo il program.
   8          D8  CLD            Funzionamento in esadecimale
   9          A9  LDA #$99
   A          99
   B          8D  STA  CONTR     Inizializzazione dell'8255
   C          03
   D          FD
   E          20  JSR  TASTO      Salto alla subroutine di identifica-
   F          57                 zione del tasto e torno col valore del
0220          FF                 tasto premuto in ACC.
   1          A0  LDY  #04       Uso Y come contatore
   2          04
   3  LOOP 1  06  ASL  $F9  ┐
   4          F9             │
   5          26  ROL  $FA   ├
   6          FA             │
   7          26  ROL  $FB   ├  Eseguo 4 volte lo shift per far entrare
   8          FB             │  il tasto premuto nel display (3)
   9          88             │
   A          D0  BNE  LOOP 1┘
   B          F7
   C          05  ORA  $F9     Eseguo l'operazione di OR fra il
   D          F9                contenuto di F9 e l'accumulatore (4)
   E          18  CLC           (5)
   F          90  BCC  IND 2     Rimetto a posto il digit meno signi-
0230          D5                 ficativo e vado a porre il FLAG = 1
```

L'Interrupt

Per completare il set delle istruzioni del 6502 rimangono ancora pochi particolari da esaminare; vediamo ora una caratteristica hardware che ha ripercussioni anche nel software: l'INTERRUPT.

Supponiamo che il nostro calcolatore stia percorrendo un suo programma principale, che stia per esempio contando quante volte si apre e chiude un interruttore collegato ad una sua linea di ingresso. Supponiamo che, mentre sta eseguendo questo lavoro debba, *contemporaneamente*, tenere acceso un display a 6 cifre aggiornando il dato che vi è scritto.

Il vero problema è dato appunto dal "contemporaneamente" in quanto la macchina in effetti può eseguire un solo calcolo per volta: deve perciò passare in continuazione dall'esecuzione di un programma all'esecuzione di un altro per farli avanzare di pari passo tutti e due.

Questo passaggio è possibile se si interrompe periodicamente il lavoro della CPU tramite un temporizzatore esterno, che fornisce ad un piedino della CPU un segnale ad onda quadra di interruzione.

Il processor quando riceve questo segnale di interruzione, abbandona l'esecuzione del programma principale per andare a servire le necessità del programma secondario.

Un altro esempio di interruzione può essere quello di un interruttore di fine corsa che in chiusura deve interrompere

qualsiasi calcolo stia eseguendo la CPU perché diversamente c'è un pistone che va a sbattere rovinando la macchina controllata. Alla chiusura del fine corsa allora viene generato il segnale di interruzione e la macchina va immediatamente ad eseguire il programma che contiene i vari controlli per l'arresto del pistone.

Da tutto ciò ci si può rendere conto dell'importanza fondamentale di questo segnale che permette di controllare fenomeni fra loro asincroni e completamente indipendenti l'uno dall'altro.

Tipi di Interrupt

Il microprocessore 6502 ha due diversi tipi di ingresso di interrupt tramite piedini marcati con le sigle $\overline{IRQ}$ e $\overline{NMI}$.

La differenza sostanziale fra questi due piedini è che $\overline{IRQ}$ è sensibile al livello di ingresso, vale a dire che si genera una interruzione se il piedino $\overline{IRQ}$ va a Ø Volt mentre $\overline{NMI}$ è sensibile ai fronti vale a dire che si genera una interruzione se su questo piedino si presenta un fronte di discesa, cioè se la sua tensione passa da 5 V a Ø Volt e solo a questo passaggio (vedi Fig. 12).

In pratica il controllo dello stato di questi due piedini viene fatto in continuazione dalla CPU e in particolare prima della esecuzione di ciascuna istruzione. Nota Bene: per il piedino $\overline{NMI}$ il fronte di discesa viene memorizzato da un circuito interno che permette il funzionamento sul solo fronte.

Un'altra differenza fra i due tipi di interrupt è che mentre l'interruzione generata su $\overline{IRQ}$ può essere mascherata (ignorata dalla CPU), la CPU serve sempre quella che nasce su $\overline{NMI}$.

Il 3° bit dello Status (bit I) serve proprio a bloccare la interruzione generata su $\overline{IRQ}$. Se viene posto a 1 da programma, l'interruzione non passa, se viene posto a Ø l'interrupt è abilitato.

Normalmente all'accensione della macchina di ha I = 1.

Per il bit I esistono due istruzioni specifiche:

CLI (Clear Interrupt) codice op. 58, che pone I = Ø.

SEI (Set Interrupt) codice op. 78, che pone I = 1.

Il bit I viene anche automaticamente messo a 1 (cioè viene disabilitato l'interrupt) dalla CPU, quando essa va a servire una interruzione; questo perché diversamente la CPU continuerebbe a servire la stessa interruzione (ricordiamoci che siamo sensibili al livello) (vedi Fig. 13).

La messa a Ø di I avviene invece tramite l'istruzione RTI (ritorno da Interrupt) codice op. 40 o tramite la CLI.

Come già detto invece l'interruzione su $\overline{NMI}$ nasce solo su un fronte di discesa su questo piedino e non può venire bloccata dalla CPU; perché avvenga una successiva richiesta di interruzione il piedino $\overline{NMI}$ deve tornare alto, quindi ancora basso.

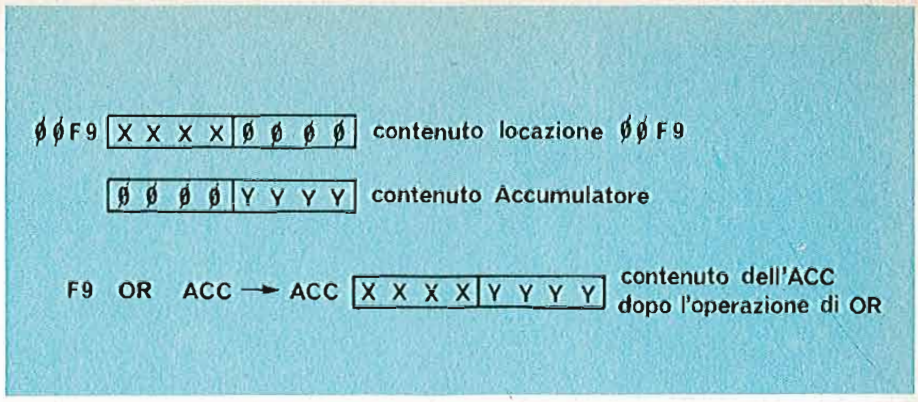


Fig. 11 - Operazione di OR fra il dato contenuto nella locazione ØØF9 e quello contenuto in accumulatore.

Il funzionamento della interruzione

Come risponde la CPU ad una interruzione?

Essa interrompe l'esecuzione del programma che sta eseguendo, salva nello Stack lo Status e il PC che vi era in macchina al momento dell'interruzione, questo per poter poi tornare al programma principale al punto dove era stato arrestato. La CPU quindi preleva da locazione fisse della memoria il punto da cui parte la routine di gestione dell'interruzione (ovvero il nuovo programma da eseguire) che finisce sempre con una istruzione di RTI. Con RTI viene eseguita la procedura esattamente inversa ripristinando i valori (presenti nello Stack)

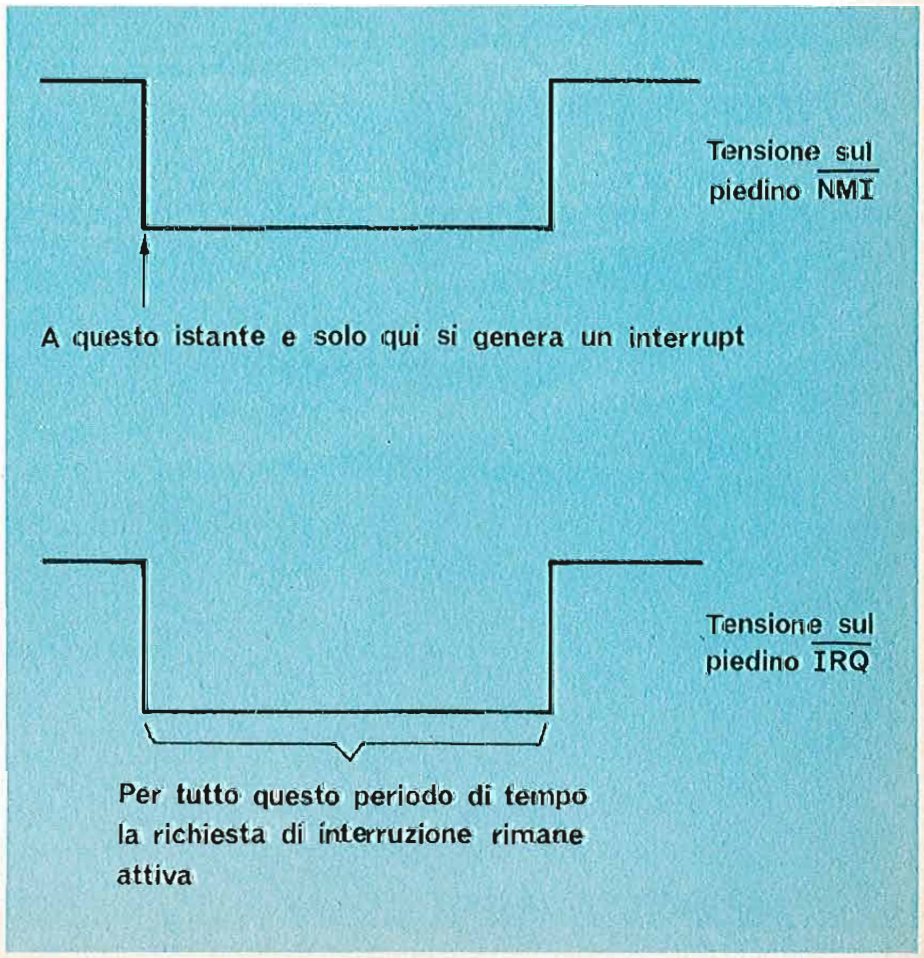


Fig. 12 - Differenza fra le richieste di interruzione nel 6502.

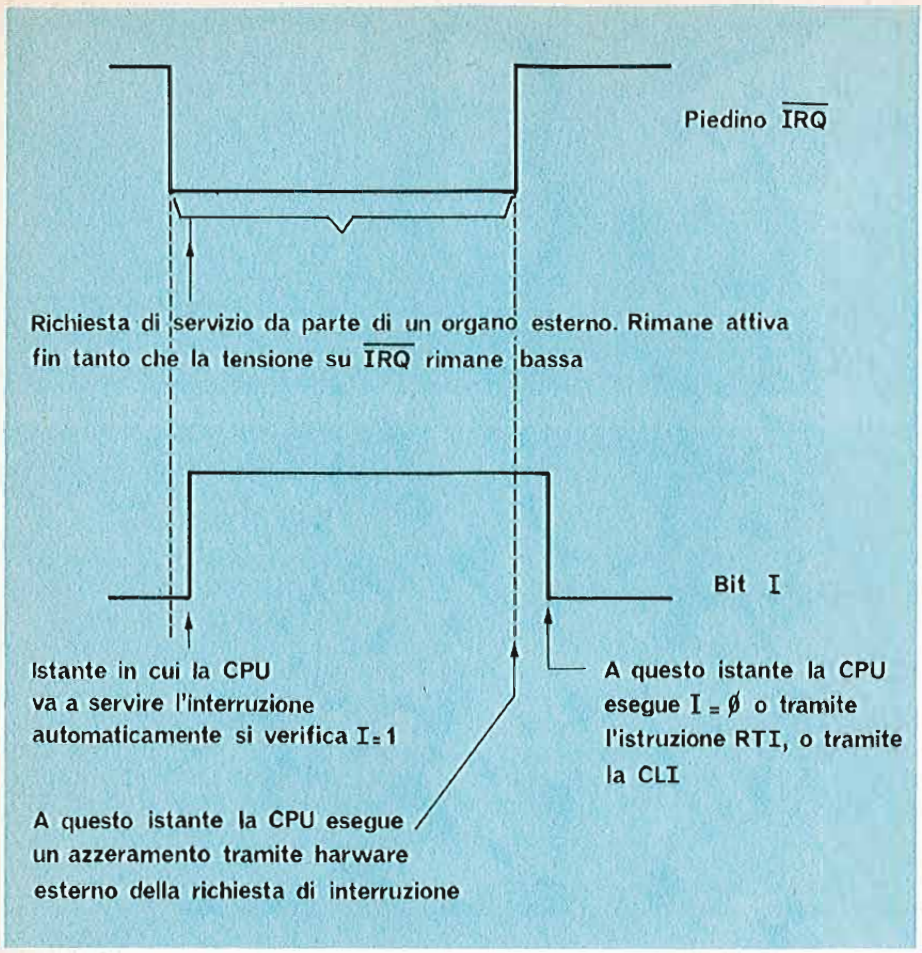


Fig. 13 - Funzionamento dell'Interrupt.

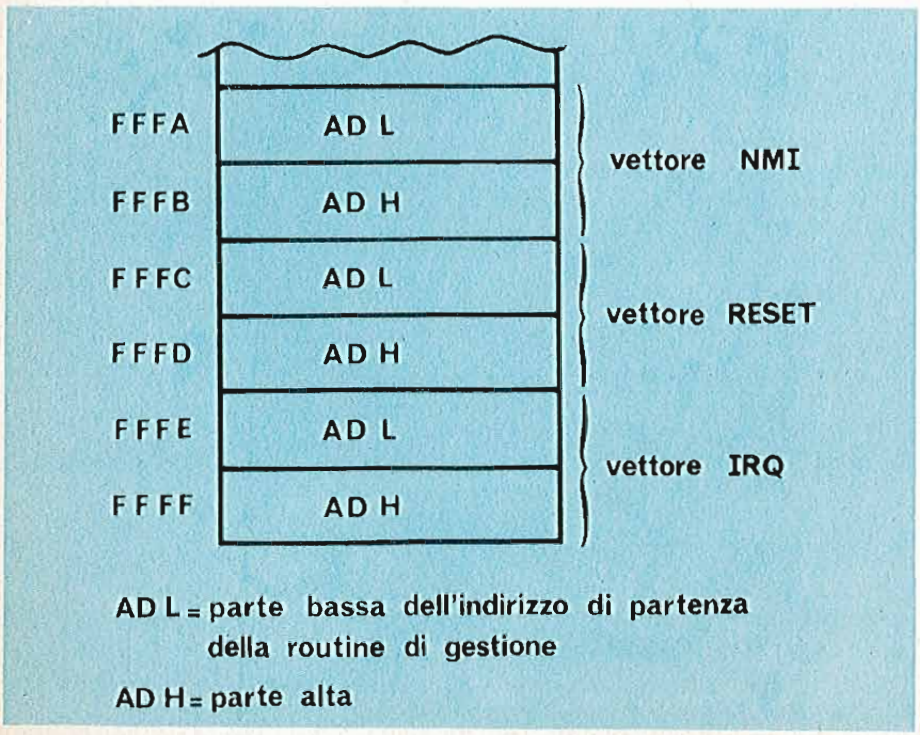


Fig. 14 - Posizionamento in memoria dei vettori di partenza della routine di Interrupt.

dello Status e del PC di partenza.

Si noti che questa procedura automatica non prevede il salvataggio dell'ACC., di X e di Y, salvataggi che, se necessario, devono essere eseguiti dallo stesso programma di gestione dell'interrupt.

I vettori di partenza delle routine di Interrupt sono posizionati in memoria secono quanto riportato nella Fig. 14. Questi cosiddetti vettori non sono altro che due locazioni di memoria successive che contengono l'indirizzo di memoria in cui inizia il programma di gestione dell'interrupt; nel caso dell'AMICO 2000 sono già scritti nella PROM del Monitor (vedi Fig. 15).

Il vettore di Reset verrà esaminato in seguito.

Notiamo subito che gli indirizzi in cui sono memorizzati i vettori di restart sono fissi e posizionati in fondo alla memoria (le ultime 6 parole dei 64k indirizzabili). È perciò che generalmente l'ultimo k della memoria di un 6502 è di memoria ROM.

Esempio dell'uso di Interrupt

Per usare l'Interupt nell'AMICO 2000, che ha i vettori di partenza fissi sulla PROM del Monitor, si ricorre ad un artificio di programmazione presente nel monitor stesso.

Le routine di $\overline{NMI}$ e di $\overline{IRQ}$ si riducono ad un salto indiretto sul contenuto delle locazioni di memoria RAM (quindi modificabili dall'utente) che sono:

Ø3FC = ADL di $\overline{NMI}$

Ø3FD = ADH di $\overline{NMI}$

Ø3FE = ADL di $\overline{IRQ}$

Ø3FF = ADH di $\overline{IRQ}$

Quindi nei nostri programmi utente dobbiamo inserire, prima di usare l'interrupt, i nostri specifici vettori di restart in queste locazioni di RAM (ovvero l'indirizzo dal quale deve partire la routine dell'interrupt).

Per fare un esempio dell'uso di interrupt dobbiamo avere a disposizione un segnale hardware che generi l'interrupt stesso. Osservando lo schema elettrico dell'AMICO 2000 (pubblicato sul numero 3/79 di Sperimentare a pag. 289) si può vedere che il tasto di HALT (HLT) quando è premuto genera su $\overline{NMI}$ (piedino 6) un fronte di discesa di tensione. Possiamo perciò sfruttare questo tasto per generare una interruzione.

Facendo girare il programma che segue vedremo il display spegnersi per accendere per circa 16 secondi tutti 8 sul display ogni volta che viene premuto il tasto di HLT.

```
0200          A9  LDA  #$00
   1          00
   2          8D  STA  $03FC
   3          FC
   4          03                  Caricamento del vettore di restart di
   5          A9  LDA  #$03       NMI (locazione 0300)
   6          03
   7          8D  STA  $03FD
   8          FD
   9          03
   A  Loop    18  CLC
   B          90  BCC  Loop       Il processor continua a girare in que-
   C          FD                  sto punto (il display si spegne)
```

Scriviamo la routine che serve l'interruzione:

```
0300          A9  LDA  #$88
   1          88
   2          85  STA  $FB
   3          FB
   4          85  STA  $FA       Carico 88 sul display
   5          FA
   6          85  STA  $F9
   7          F9
   8          A9  LDA  #$10
   9          10
   A          85  STA  $00
   B          00
   C          A9  LDA  #$00       Parametri della routine di ritardo
   D          00                  (durata accensione display)
   E          85  STA  $01
   F          01
0310  Loop 1  20  JSR  SCANS      Routine rinfresco display
   1          0C
   2          FF
   3          C6  DEC  $01
   4          01
   5          D0  BNE  Loop 1    Routine di ritardo
   6          F9
   7          C6  DEC  $00
   8          00
   9          D0  BNE  Loop 1
   A          F5
   B          40  RTI            Ritorno al programma principale
```

Analizziamo il programma:

1) Si caricano i vettori di restart, ovvero la locazione dalla quale deve partire la routine che serve l'interruzione.

2) In questo Loop il programma gira su se stesso tenendo spente tutte le cifre del display.

A questo punto l'unico sistema che consente di intervenire sulla macchina è il ricorso all'interrupt. Useremo l'$\overline{\text{NMI}}$ per via della presenza del tasto HLT.

3) Alla locazione 0300 comincia la routine di interrupt.

Si noti che per il loop di ritardo non si usano i registri indice X e Y poiché la routine di scansione del display li modifica. Durante il loop di ritardo il display rimane acceso mostrando tutti 8.

Come ultima nota avvertiamo che se si preme ancora il tasto HLT mentre il display è acceso esso continuerà a rimanere acceso perché non si esce più dalla routine di interrupt.

Un esercizio e un premio ai migliori

Ormai chi ci ha seguito fin qui ha in mano quasi tutti gli strumenti per cominciare a realizzare programmi di una certa complessità; vogliamo proporvi le specifiche di un programma da realizzare. Invitiamo tutti coloro che riusciranno a scriverlo a inviarlo alla ASEL, Via Cortina d'Ampezzo 17, Milano: il programma più intelligente e realizzato col minor numero di istruzioni sarà premiato con un bellissimo dono.

Specifiche del programma: premendo RUN sul display devono apparire tutti zeri. Premendo ora un tasto con numero dispari sulla 1ª, 3ª e 5ª cifra del display (a cominciare da sinistra) dovranno accendersi alternativamente (a una cadenza di circa 1/2 secondo) l'anello superiore e quello inferiore della cifra 8 mentre le altre cifre sono spente.

Questa alternanza durerà finché premendo un qualsiasi tasto con cifra pari la stessa alternanza passerà sulle cifre 2ª, 4ª e 6ª del display, mentre le altre si spegneranno. Dovrà essere possibile passare in qualsiasi momento e per quante volte si vuole dall'accensione delle cifre pari a quelle dispari e viceversa.

Si è sempre in tempo per imparare

Informiamo i lettori che per la prima volta avessero preso visione di questa serie di articoli che essa è cominciata con il numero 12 del 1978. Richiedendo gli arretrati ci si potrà rendere conto della semplicità con cui un argomento così moderno e fondamentale nella formazione tecnica di un professionista come di uno sperimentatore, sia stato esposto. Lo ha dimostrato nel corso di questi mesi il consenso unanime di tutti i lettori che si sono appassionati all'argomento sul microprocessore fino a ieri ritenuto di difficile apprendimento.

Funzionamento in SINGLE STEP

Sull'AMICO 2000A esiste un interruttore per il funzionamento della macchina in "Single Step" o passo singolo che fino ad ora non abbiamo utilizzato.

Con questo tipo di funzionamento è possibile esaminare istruzione per istruzione lo svolgersi di un programma fermandosi ad esaminare per ogni passo *(step)* il contenuto dell'accumulatore e degli altri registri di macchina. Ciò è molto utile quando, ad esempio, non si riesce a comprendere perché un certo programma non va come vorremmo e scoprire quindi dove è stato fatto l'errore.

Per poter utilizzare la macchina in single step prima di tutto è necessario caricare i dati 00 e FF rispettivamente alle locazioni 03FC e 03FD. Si provvederà quindi a scrivere il programma sul quale si intende operare in single step badando che il programma stesso non vada ad interessare le locazioni 03FC e 03FD.

Si porta ora sul display indirizzi la locazione di partenza del programma quindi si sposta a destra l'interruttore del single step.

A questo punto premendo il tasto RUN vediamo che sul display appare l'indirizzo della seconda istruzione del programma e, sui due digit dei dati, il

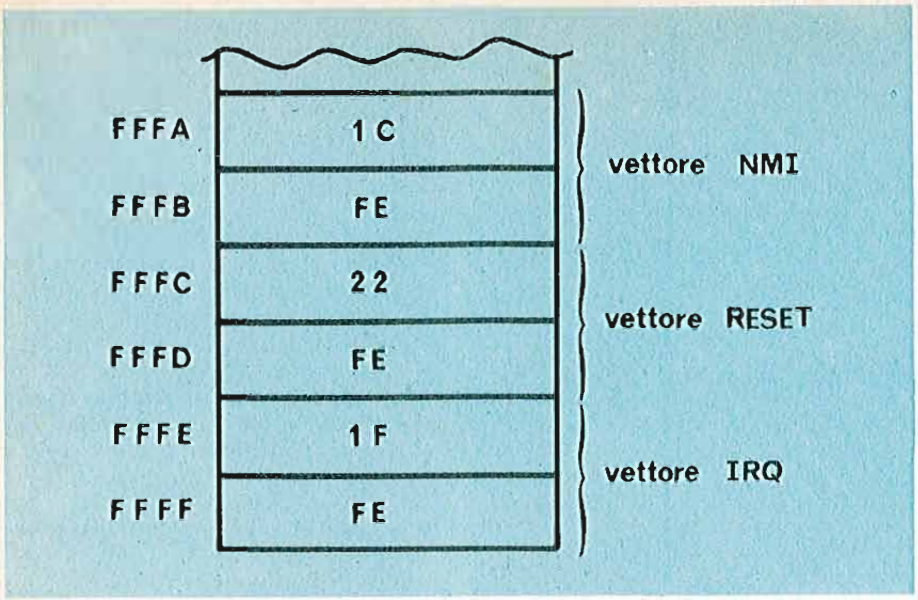


Fig. 15 - Vettori di restart dell'AMICO 2000/A.

contenuto esadecimale della locazione di memoria corrispondente.

Procedendo in questo modo (premendo ad ogni istruzione RUN) si può esaminare lo sviluppo del programma, i vari salti o condizionamenti.

Se si desidera esaminare, dopo l'esecuzione di una determinata istruzione, il contenuto di un qualsiasi registro o locazione di memoria della macchina sarà sufficiente portarsi all'indirizzo corrispondente al registro o alla locazione desiderata secondo quanto indicato di seguito:

00F3 = Accumulatore
00F4 = Registro Y
00F5 = Registro X
00F6 = Program Counter (parte bassa)
00F7 = Program Counter (parte alta)
00FD = Status
00FE = Stack Pointer

Nel caso si desideri modificare uno qualsiasi di questi registri si procede al solito modo.

Per riprendere l'esecuzione in single step del programma è sufficiente richiamare il Program Counter premendo il tasto REG e di seguito il tasto RUN passando così alla prossima istruzione.

Si badi bene che *nessuna* delle operazioni di analisi dei vari registri o locazioni modifica di per se stessa i dati del programma.

Errata corrige: *Nel programma "Corsa dei Cavalli" pubblicato a pag. 738 del n° 9/1979 di Sperimentare, il byte all'indirizzo 021A deve essere cambiato da 36 a 3B.*