Gestione delle interruzioni

di Andrea de Prisco

Grazie alla gestione delle interruzioni da parte del video interface chip 6567, questo mese realizzeremo per il nostro Commodore 64 finestre testo/grafica hgr e la visualizzazione contemporanea di 16 sprite.

Seconda parte

Gli Interrupt del VIC 6567

Dicevamo lo scorso mese che ogni 60-esimo di secondo l'integrato di Input/Output Cia 6526, manda un segnale di interruzione al microprocessore 6510. Se l'interruzione è accolta, questi molla tutto ed esegue il programma di manipolazione delle interruzioni. Con le applicazioni Orologio, Sveglia e Caratteri Flashing abbiamo anche visto come sia possibile sfruttare gli interrupt per gestire una mini programmazione parallela.

Dando uno sguardo un po' più attento ai collegamenti tra i vari chip del 64 ci si può rendere conto che le interruzioni possono giungere anche da altrove. Innanzitutto dall'esterno della macchina: il contatto IRQ del microprocessore è riportato anche sulla porta espansioni del 64. In questo modo è possibile che periferiche esterne diano comandi di interruzione al microprocessore, per eseguire particolari operazioni.

Non affronteremo da questo punto di vista il problema: ci occuperemo delle interruzioni da parte del video interface chip 6567. Anche quest'ultimo ha per così dire, un filo diretto col 6510; quando "lo ritiene opportuno", può mandare la sua brava interruzione a papà. Tanto per cambiare, per attivare gli interrupt del 6567 è necessario agire sui registri interni, listati bit per bit a pagina 124.

Il Raster Register

Prima di entrare nel vivo del discorso, ci sarebbe da parlare un po' del Raster Register o registro di quadro. È formato da 9 bit e quindi è locato in due registri del video interface chip. Gli 8 bit di ordine più basso si trovano alla locazione 53266, il bit più significativo è locato nel registro 53265, sempre nella posizione più alta (la 7). Svolge una duplice funzione, a seconda che sia usato in lettura (PEEK) o in scrittura (POKE). Nel primo caso restituisce l'attuale posizione del "pennello elettronico" del vostro TV: come noto, l'immagine video è formata da un insieme di linee orizzontali che , opportunamente costruite da un fa-

Listato 1

```
100 FORI=52768TO52895:READII:POKEI,II:NEXT
110 REM ********************************
111 REM *                              *
120 REM * T I T O L A T R I C E  6567 *
150 REM *                              *
160 REM *  ESEMPIO DI SCROLLING FINE   *
161 REM *                              *
162 REM *   ----------------------     *
163 REM *   (C) 1985  ADP SOFTWARE     *
164 REM *   ----------------------     *
165 REM *                              *
170 REM ********************************
180 PRINT"            T I T O L A T R I C E
   "
190 DIMA$(100):I=0
200 A$="":INPUTA$:A=LEN(A$):IFA=0THEN230
210 IFA<39THENA$(I)=LEFT$("                                      "
,19-A/2)+A$:I=I+1:GOTO200
220 A$(I)=LEFT$(A$,38):A$=MID$(A$,39):I=I+1:A=LEN(A
$):GOTO210
230 POKE53265,PEEK(53265)AND247:K=0
240 PRINT"";
250 POKE53265,(PEEK(53265)AND248)OR7
260 PRINT"□"A$(K):K=K+1:IFK>ITHENK=0
270 FORP=6TO0STEP-1
280 POKE255,P:SYS52768
290 FORT=1TO50:NEXT
300 NEXT:SYS52791:GOTO260
310 DATA173,17,208,41,128,208,249,173,18,208,208,24
4,173,17,208,41,248,5,255
320 DATA141,17,208,96,120,173,17,208,41,128,208,249
,173,18,208,201,251,208,242
330 DATA173,17,208,9,7,141,17,208,162,0,189,40,4,15
7,0,4,232,208,247,189,40,5
340 DATA157,0,5,232,208,247,189,40,6,157,0,6,232,20
8,247,189,40,7,157,0,7,232
350 DATA224,192,208,245,162,0,189,40,216,157,0,216,
232,208,247,189,40,217,157
360 DATA0,217,232,208,247,189,40,218,157,0,218,232,
208,247,189,40,219,157,0,219
370 DATA232,224,192,208,245,88,96
```

Listato 2

```
1 REM **********************************
2 REM *                                *
3 REM *     S   C   R   O   L   L      *
4 REM *                                *
5 REM *   ESEMPIO DI SCROLLING FINE    *
6 REM * SENZA USO DEL RASTER REGISTER *
7 REM *                                *
8 REM **********************************
10 POKE53265,PEEK(53265)AND247
20 PRINT""
30 POKE53265,(PEEK(53265)AND248)OR7
40 PRINT" **** SFARFALLAMENTO DI SCHERMO **** "
50 FORI=6TO0STEP-1
60 POKE53265,(PEEK(53265)AND248)ORI
70 REM FORT=1TO50:NEXT
80 NEXT:GOTO30
```

scio di elettroni all'interno del cinescopio, formano l'intera schermata. Indipendentemente da applicazioni computerecce, questo avviene anche mentre si segue il telegiornale. È l'occhio che con la sua inerzia non si accorge di quanto effettivamente accade illudendosi di avere davanti una immagine intera. Apparentemente, interrogare il Raster Register non sembrerebbe molto utile: specialmente da Basic, che con la sua esasperante lentezza, prendere una qualsiasi decisione a un determinato valore del quadro, diventa praticamente impossibile. Da linguaggio macchina la cosa diventa più interessante: è possibile realizzare un ciclo di attesa finché il registro di quadro non assuma un particolare valore. La domanda però è sempre la stessa: per quale motivo si dovrebbe sfruttare l'informazione data dal Raster Register?

Non è difficile rispondere. Per cominciare, tra tutti i valori che può assumere il registro di quadro (con 9 bit da 0 a 511), solo tra 51 e 250 il pennello sta effettivamente mostrando qualcosa sul video. Per tutti gli altri valori il video interface chip o visualizza il bordo o aspetta semplicemente di iniziare un nuovo quadro. Interrogando continuamente il Raster Register possiamo ad esempio modificare qualcosa sul video mentre il pennello è fuori campo (parlando a rallenty, mentre non visualizza nulla). Per capire meglio, sfruttiamo un'altra delle possibilità offerte dal 6567: lo scrolling fine (di un solo pixel alla volta) nelle quattro direzioni.

Viene usato per far entrare "in campo" lentamente nuove informazioni mentre lentamente vecchie informazioni spariscono dalla parte opposta. Il programma Titolatrice listato in queste pagine già visto, ma non commentato, sul n. 30 di MC, ne è un esempio.

Dando Run vengono richieste le linee da mostrare (max 100). Si dà il via battendo Return all'ultima richiesta di input. Il video interface chip svolge gran parte del lavoro, ma non tutto. Per implementare lo scrolling fine bisogna scrivere un programma opportuno, preferibilmente in linguaggio macchina. La prima cosa da fare, è passare al modo 38 colonne per lo scrolling orizzontale o al modo 24 righe se si desidera quello verticale. Ciò per far posto alle nuove informazioni prima dello scrolling vero e proprio. Per il movimento verticale verso l'alto, i passi sono:

1) passare al modo 24 righe;

2) impostare il registro di scrolling verticale al valore massimo (tutto lo schermo si abbassa di 8 pixel, nascondendo sotto il bordo inferiore la 25-esima riga);

3) riempire la riga 25 con le informazioni da mostrare;

4) variare lentamente il registro di scrolling in modo da far apparire la riga 25 e far scomparire la prima;

5) con una routine in linguaggio macchina per muovere il contenuto dello schermo di una posizione verso l'alto;

6) ritornare al passo 2;

Per il movimento orizzontale l'algoritmo è sostanzialmente lo stesso: uniche ovvie differenze sono:

a) il modo da usare è quello a 38 colonne.

b) Si agisce sul registro di scrolling orizzontale.

c) I nuovi dati andranno posizionati sull'estrema colonna di destra o di sinistra a seconda della direzione dello scroll.

Le POKE da usare sono:

```
POKE 53270,PEEK(53270)AND 247
```

seleziona il modo 38 colonne.

```
POKE 53270,PEEK(53270) OR 8
```

ritorna al modo standard 40 colonne.

```
+-----------------------------------+
49152 !                                   !
      !                                   !
      !                                   !
      !     17 registri coordinate        !
      !                                   !
      !                                   !
      !          sprite  0-7              !
      !                                   !
      !                                   !
49168 !                                   !
      +-----------------------------------+
49173 !        attiva sprite 0-7          !
      +-----------------------------------+
49175 !       exp. v.  sprite 0-7         !
      +-----------------------------------+
49179 !   priorita' sprite 0-7 testo      !
      +-----------------------------------+
49180 ! selezione multicolor sprite 0-7   !
      +-----------------------------------+
49181 !       exp. h.  sprite 0-7         !
      +-----------------------------------+
49182 ! collisione sprite 0-7 sprite 0-7 !
      +-----------------------------------+
49183 !   collisione sprite 0-7  testo    !
      +-----------------------------------+
49189 !      multicolor sprite #1         !
      +-----------------------------------+
49190 !      multicolor sprite #2         !
      +-----------------------------------+
49191 !                                   !
      !                                   !
      !     colore degli sprite 0-7       !
      !                                   !
      !           (8 byte)                !
      !                                   !
49198 !                                   !
      +-----------------------------------+
49200 !                                   !
      !                                   !
      !                                   !
      !    16 puntatori ai 16 sprite      !
      !                                   !
      !                                   !
      !                                   !
49215 !                                   !
      +-----------------------------------+
49216 !                                   !
      !                                   !
      !                                   !
      !     17 registri coordinate        !
      !                                   !
      !                                   !
      !          sprite  8-15             !
      !                                   !
      !                                   !
49232 !                                   !
      +-----------------------------------+
49237 !        attiva sprite 8-15         !
      +-----------------------------------+
49239 !       exp. v.  sprite 8-15        !
      +-----------------------------------+
49243 !   priorita' sprite 8-15 testo     !
      +-----------------------------------+
49244 ! selezione multicolor sprite 8-15 !
      +-----------------------------------+
49245 !       exp. h.  sprite 8-15        !
      +-----------------------------------+
49246 !collisione sprite 8-15 sprite 8-15!
      +-----------------------------------+
49247 !   collisione sprite 8-15 testo    !
      +-----------------------------------+
49253 !      multicolor sprite #1         !
      +-----------------------------------+
49254 !      multicolor sprite #2         !
      +-----------------------------------+
46255 !                                   !
      !                                   !
      !     colore degli sprite 8-15      !
      !                                   !
      !           (8 byte)                !
      !                                   !
49262 !                                   !
      +-----------------------------------+
```

```
POKE 53265,PEEK(53265)AND 247
```

seleziona il modo 24 righe.

```
POKE 53265,PEEK(53265) OR 8
```

ritorna al modo 25 righe

Con X e Y compresi tra 0 e 7 tramite le due seguenti poke, si importano i registri di scrolling orizzontale e verticale:

```
POKE 53270,(PEEK(53270) AND 248) OR X - orizzontale
POKE 53265,(PEEK(53265) AND 248) OR Y - verticale
```

Fin qui sembrerebbe tutto normale. Il problema più grave è dato dal fatto che, nell'algoritmo sopra visto, il passo 5 e il passo 2 (ad ogni iterazione) dovrebbero avvenire contemporaneamente, pena un fastidioso sfarfallio del quadro durante la costruzione dell'immagine da parte del 6567. Considerato poi che "contemporaneamente" (questa volta nel vero senso della parola) per il 64 è impossibile, l'unica cosa da fare è eseguire sequenzialmente le due operazioni quando il 6567 non visualizza nulla, in altre parole, quando il Raster Register ha superato il valore di 250. Tanto per avere un'idea di cosa voglia dire non aspettare il momento opportuno per eseguire i due punti di cui sopra, digitate il programma Scroll.

Vedrete il quadro sfarfallare ad ogni iterazione.

Il programma Titolatrice effettua la trasposizione di tutto lo schermo e resetta il registro di scroll quando il registro di quadro ha superato 250. La realizzazione di questo sincronismo è affidata a una routine in linguaggio macchina, come si può notare dalle linee di DATA presenti nel listato.

Le interruzioni

Per generare veri e propri interrupt da parte del 6567 si agisce sui registri 26 e 27, locati rispettivamente a 53273 e 53274. Il video interface chip può mandare segnali di interruzione al verificarsi di uno dei seguenti 4 eventi: è avvenuta una collisione tra due sprite, tra uno sprite e il testo, è stata appoggiata la penna ottica sullo schermo TV, il registro di quadro ha raggiunto un determinato valore. Per selezionare tra questi, al verificarsi di quale evento bisogna avvertire il 6510 con una interruzione, si agisce sul registro locato a 53274, detto appunto di abilitazione dell'interruzione. Settando il bit 0 avremo un interrupt sincronizzato con una data posizione del quadro, settando il bit 1 l'interruzione si verificherà a ogni collisione sprite-dati, col bit 2 ad ogni collisione sprite-sprite, setteremo il bit 3 se interessati a interrupt da penna ottica. Sarà opportuno disabilitare le interruzioni del 6526 se vogliamo abilitare quelle del 6567. La poke che ci permette questo è:

```
POKE 56334, PEEK(56334) AND 254
```

La possibilità degli interrupt causati da collisioni da sprite, può essere usata per i giochi in linguaggio macchina, ad esempio incrementare il punteggio ad ogni navicella spaziale abbattuta. Si potrebbe inserire la routine che conteggia i punti a capo del

programma di manipolazione delle interruzioni, come già visto per i programmi "Orologio", "Sveglia" e "Caratteri Flashing". Se la navicella è abbattuta, il microprocessore molla tutto (come al solito) e incrementa il punteggio.

Per provocare una interruzione ad un dato valore del quadro, comunicheremo al 6567 tale quantità. Per fare questo si usa il Raster Register in scrittura: una POKE in tale registro, viene memorizzata all'interno del video interface chip per provocare l'interruzione ogni volta che il quadro eguaglia tale valore. Anche in questo caso, occorre ricordare che il Raster Register è formato di 9 bit e di conseguenza, locato in due registri del 6567. Si tratterà sempre di eseguire due PEEK o due POKE.

Un'applicazione abbastanza semplice di tale procedimento potrebbe essere la visualizzazione di mezza pagina video di un colore e la rimanente di un altro. Colore di sfondo, si intende. L'informazione del colore di sfondo, com'è noto, è mantenuta nella locazione 53281. Se cambiamo tale valore, lo schermo cambia tinta. L'idea è quella di passare da un colore ad un altro, ripetutamente, a determinati valori del quadro. Diciamo che si parte col colore di sfondo blu. Quando il 6567 ha disegnato metà schermo, cambiamo colore di sfondo, (agendo in 53281) in rosso. Terminato il quadro, si ripassa al colore blu, per poi riattivare il rosso a metà schermo. Tutto questo venticinque volte al secondo, la frequenza con cui il video interface chip disegna il quadro. I passi veri e propri sono:

1) disabilitare le interruzioni del 6522.

2) Cambiare il contenuto di $0314 e $0315 (puntatore alla routine di manipolazione dell'interruzione).

3) Scrivere in Ram la nuova routine di manipolazione delle interruzioni.

4) Impostare in registro di quadro al valore di 150 (approx.mezzo schermo).

5) Abilitare le interruzioni del 6567, settando il bit 0 del registro 53274.

La nuova routine di manipolazione delle interruzioni, sarà chiamata, per la prima volta, appena il video interface chip ha completato il primo semi schermo (che ha colorato in blu). Questa routine, sarà approssimativamente fatta così:

1) se il colore di schermo è rosso, vai a 5 (è stata già completata la schermata);

2) si passa al colore di schermo rosso;

3) si imposta il Raster Register al valore di 251, per provocare una nuova interruzione al termine del quadro;

4) salta a $EA31 per completare il programma di manipolazione delle interruzioni (scansione della tastiera, incremento del TI$, etc.);

5) si passa al colore di schermo blu;

6) si imposta il Raster Register al valore di 150, per provocare una nuova interruzione a metà quadro;

7) salta a $EA 31 per completare il programma di manipolazione delle interruzioni (scansione della tastiera, incremento di TI$, etc.).

I registri del 6567, bit per bit.

Decimale	Hex	Bit	Descrizione
53248	$D000	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 0
53249	$D001	0-7	coordinata verticale dello sprite 0
53250	$D002	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 1
53251	$D003	0-7	coordinata verticale dello sprite 1
53252	$D004	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 2
53253	$D005	0-7	coordinata verticale dello sprite 2
53254	$D006	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 3
53255	$D007	0-7	coordinata verticale dello sprite 3
53256	$D008	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 4
53257	$D009	0-7	coordinata verticale dello sprite 4
53258	$D00A	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 5
53259	$D00B	0-7	coordinata verticale dello sprite 5
53260	$D00C	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 6
53261	$D00D	0-7	coordinata verticale dello sprite 6
53262	$D00E	0-7	8 bit meno significativi della coordinata orizzontale dello sprite 7
53263	$D00F	0-7	coordinata verticale dello sprite 7
53264	$D010	0	bit 9 della coordinata orizzontale dello sprite 0
		1	bit 9 della coordinata orizzontale dello sprite 1
		2	bit 9 della coordinata orizzontale dello sprite 2
		3	bit 9 della coordinata orizzontale dello sprite 3
		4	bit 9 della coordinata orizzontale dello sprite 4
		5	bit 9 della coordinata orizzontale dello sprite 5
		6	bit 9 della coordinata orizzontale dello sprite 6
		7	bit 9 della coordinata orizzontale dello sprite 7
53265	$D011	0-2	registro di scroll verticale
		3	seleziona il modo 24 righe
		4	sgancia l'immagine video
		5	seleziona il modo bit-map
		6	seleziona l'extended background color mode
		7	bit 8 del raster register
53266	$D012	0-7	raster register (registro di quadro)
53267	$D013	0-7	coordinata verticale della penna ottica
53268	$D014	0-7	coordinata orizzontale della penna ottica
53269	$D015	0	attiva lo sprite 0
		1	attiva lo sprite 1
		2	attiva lo sprite 2
		3	attiva lo sprite 3
		4	attiva lo sprite 4
		5	attiva lo sprite 5
		6	attiva lo sprite 6
		7	attiva lo sprite 7
53270	$D016	0-2	registro di scroll orizzontale
		3	seleziona il modo 38 colonne
		4	attiva il modo multicolor
		5	reset del 6567 (dato al momento dell'accensione)
		6-7	non connessi
53271	$D017	0	espansione verticale dello sprite 0
		1	espansione verticale dello sprite 1
		2	espansione verticale dello sprite 2
		3	espansione verticale dello sprite 3
		4	espansione verticale dello sprite 4
		5	espansione verticale dello sprite 5
		6	espansione verticale dello sprite 6
		7	espansione verticale dello sprite 7
53272	$D018	0	non connesso
		1-3	puntatore generatore caratteri
		4-7	puntatore memoria video
53273	$D019	0	resetta interrupt del raster register
		1	resetta interrupt da collisione sprite-testo
		2	resetta interrupt da collisione sprite-sprite
		3	resetta interrupt da penna ottica

			4-6	non connessi
			7	segnala avvenuta IRQ
53274	$D01A		0	abilit. interrupt del raster register
			1	abilit. interrupt da collisione sprite-testo
			2	abilit. interrupt da collisione sprite-sprite
			3	abilit. interrupt da penna ottica
			4-7	non connessi
53275	$D01B		0	priorita' col testo sprite 0
			1	priorita' col testo sprite 1
			2	priorita' col testo sprite 2
			3	priorita' col testo sprite 3
			4	priorita' col testo sprite 4
			5	priorita' col testo sprite 5
			6	priorita' col testo sprite 6
			7	priorita' col testo sprite 7
53276	$D01C		0	selezione multicolor per lo sprite 0
			1	selezione multicolor per lo sprite 1
			2	selezione multicolor per lo sprite 2
			3	selezione multicolor per lo sprite 3
			4	selezione multicolor per lo sprite 4
			5	selezione multicolor per lo sprite 5
			6	selezione multicolor per lo sprite 6
			7	selezione multicolor per lo sprite 7
53277	$D01D		0	espansione orizzont. dello sprite 0
			1	espansione orizzont. dello sprite 1
			2	espansione orizzont. dello sprite 2
			3	espansione orizzont. dello sprite 3
			4	espansione orizzont. dello sprite 4
			5	espansione orizzont. dello sprite 5
			6	espansione orizzont. dello sprite 6
			7	espansione orizzont. dello sprite 7
53278	$D01E		0	collisione sprite 0 con altro sprite
			1	collisione sprite 1 con altro sprite
			2	collisione sprite 2 con altro sprite
			3	collisione sprite 3 con altro sprite
			4	collisione sprite 4 con altro sprite
			5	collisione sprite 5 con altro sprite
			6	collisione sprite 6 con altro sprite
			7	collisione sprite 7 con altro sprite
53279	$D01F		0	collisione sprite 0 con il testo
			1	collisione sprite 1 con il testo
			2	collisione sprite 2 con il testo
			3	collisione sprite 3 con il testo
			4	collisione sprite 4 con il testo
			5	collisione sprite 5 con il testo
			6	collisione sprite 6 con il testo
			7	collisione sprite 7 con il testo
53280	$D020		0-3	colore del bordo
			4-7	non connessi
53281	$D021		0-3	colore di fondo #0
			4-7	non connessi
53282	$D022		0-3	colore di fondo #1
			4-7	non connessi
53283	$D023		0-3	colore di fondo #2
			4-7	non connessi
53284	$D024		0-3	colore di fondo #3
			4-7	non connessi
53285	$D025		0-3	colore sprite multicolor #0
			4-7	non connessi
53286	$D026		0-3	colore sprite multicolor #1
			4-7	non connessi
53287	$D027		0-3	colore sprite 0
			4-7	non connessi
53288	$D028		0-3	colore sprite 1
			4-7	non connessi
53289	$D029		0-3	colore sprite 2
			4-7	non connessi
53290	$D02A		0-3	colore sprite 3
			4-7	non connessi
53291	$D02B		0-3	colore sprite 4
			4-7	non connessi
53292	$D02C		0-3	colore sprite 5
			4-7	non connessi
53293	$D02D		0-3	colore sprite 6
			4-7	non connessi
53294	$D02E		0-3	colore sprite 7
			4-7	non connessi

Le interruzioni del 6567 funzionano in un modo un po' strano: ogni volta che il video interface chip manda un interrupt, vuole conferma di avvenuta ricezione da parte del 6510. Se ciò non accade, non verranno mandati altri segnali di interrupt. La routine sopra schematizzata deve essere arricchita dal punto 0, da eseguire prima di ogni altro test o operazione. Il passo zero, per l'appunto, manda conferma al 6567, settando opportunamente un bit nel registro 53273. Il bit da settare deve essere lo stesso dell'interruzione scelta in 53274. Bit 0 nel caso nostro:
0) setta il bit 0 di 53273, per confermare l'avvenuta interruzione.

È il passo da aggiungere all'algoritmo sopra mostrato.

Non useremo gli interrupt da 66567 per visualizzare (inutilmente) schermate bicolore: ci occuperemo di qualcosa di più costruttivo: miscelare i modi testo e bit-map, e mostrare contemporaneamente 16 sprite.

L'algoritmo per visualizzare mezza pagina grafica e mezza pagina testo è sostanzialmente lo stesso visto per bi-colorare lo schermo. L'unica differenza sta nel passaggio di "modo" in luogo del semplice cambiamento di colore dei punti 2 e 5. Se modifichiamo i suddetti passi nel seguente modo:

2) si passa al modo bit-map
5) si passa al modo testo

avremo mezza schermata in alta risoluzione e mezza schermata di testo: molto utile per mostrare sia output grafici che normalissimi caratteri (es. didascalie).

La routine Hgr/Irq funziona come la grafica hgr, vista sul n. 31 di MC, ma visualizza contemporaneamente nei due modi. La prima linea serve per specificare l'altezza del "Taglio". H = 0.5 mostrerà mezzo schermo testo e mezzo bit-map. H = 0, tutto bit-map; H = 0.3, il 30% di testo e 70% di grafica. Non ha molto senso H = 1, che come è facile intuire, mostra tutto testo. Dopo il Run, per plottare punti sullo schermo, si assegnano le coordinate alle variabili X e Y e si esegue un GOSUB 30.

16 Sprite

Sfrutteremo ora gli interrupt del 6567 per visualizzare contemporaneamente 16 sprite. Il programma è listato a pagina 126 e non è altro che una routine di manipolazione delle interruzioni che gestisce tale funzione. Per poter sfruttare 16 sprite, è opportuno avere le idee ben chiare sull'argomento sprite-standard, ampiamente descritto sul manuale della macchina. Come sappiamo, il video interface chip può visualizzarne, così com'è, solo 8. L'idea è quella di far riprodurre al 6567 una schermata con una prima serie di 8 sprite, la successiva visualizzandone altri 8, poi nuovamente gli 8 di partenza e così via, ripetendo sempre lo stesso ciclo.

Useremo l'area di memoria compresa tra $D000 (decimale 49152) e $D070 (deci-

male 49264) per costruire due serie di pseudo-registri del 6567. Nella prima serie di pseudo-registri (da 49152 a 49198) inseriremo i dati relativi alla prima serie di 8 sprite (posizione, espansione, colore, ecc.); nella seconda serie di registri (da 49216 a 49263) i dati relativi alla seconda serie di sprite. Proprio come se avessimo a che fare con i veri e propri registri del 6567, mappati a partire da 53248: le prime due locazioni contengono le coordinate (x e y) del primo sprite, le seconde, del secondo sprite ecc. Potremo anche interrogare gli pseudo-registri per le collisioni, tenendo però presente che le due serie sono distinte: saranno segnalate le collisioni solo tra sprite della stessa serie. Nelle locazioni comprese tra 49200 e 49215, inseriremo i puntatori alla descrizione dei nostri 16 sprite, allo stesso modo dei byte 2040-2047, usati nella gestione normale.

Facciamo un esempio: vogliamo far apparire lo sprite 14 nelle coordinate (100,50) dello schermo. Lo sprite 14 corrisponde allo sprite 6 della seconda serie (14-8=6). Dopo aver inserito in memoria la forma dello sprite, e nel byte 49213 (il 14-esimo puntatore) la posizione della sua descrizione (contando come sempre a blocchi di 64 celle), inseriremo le sue coordinate nei registri 12 e 13 della seconda serie di pseudo registri.

Quindi l'istruzione sarà:

```
POKE 49216 + 12, 100
POKE 49216 + 13, 50
```

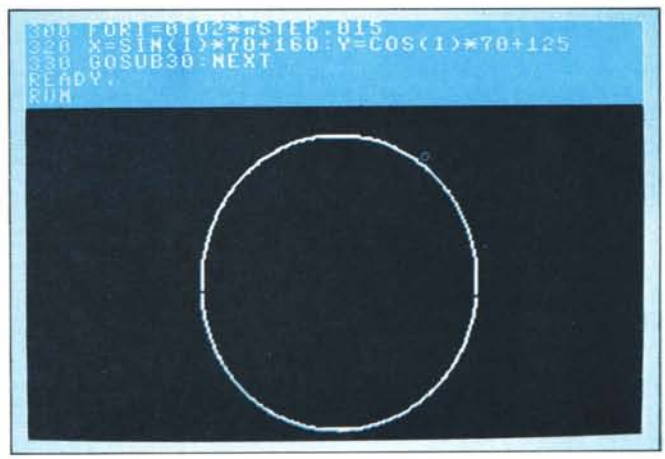
Per abilitare la visualizzazione di uno sprite si usa (come sempre) il registro 21:

```
POKE 49216 + 21, PEEK(49216 + 21) OR 2↑6
```

Se avessimo visualizzato uno sprite di ordine inferiore a 8, avremmo usato la serie di pseudo-registri mappati a partire da 49152.

Il programma che visualizza 16 sprite è assai semplice. La prima operazione che compie è di abilitare le interruzioni da parte del 6567 ogni volta che il Raster Register raggiunge il valore di 251, ovvero ad ogni completamento di schermo. Ad ogni interruzione, il microprocessore non fa altro che riversare, alternativamente, la serie di pseudo-registri nei registri veri e propri del video interface chip. Completa questa operazione prima che il 6567 inizi una nuova schermata. L'effetto finale è appunto quello di visualizzare una schermata con i primi 8 sprite, la successiva con gli altri 8, poi di nuovo gli 8 iniziali, ciclicamente. Il risultato è più che soddisfacente: l'occhio umano, sempre per la sua inerzia, vede i 16 sprite simultaneamente, sebbene leggermente "sfarfallandosi", ma non 8 alla volta come effettivamente accade. Come dire: il trucco c'è, ma (quasi) non si vede! MC

Esempio di finestra testo-grafica.



Listato 3

```
0 H=0.5
1 REM ********************************
2 REM *                              *
3 REM *       GRAFICA  HGR/IRQ       *
4 REM *                              *
5 REM *    ----------------------    *
6 REM *    (C) 1985  ADP SOFTWARE    *
7 REM *    ----------------------    *
8 REM *                              *
9 REM ********************************
10 HH=H*200+51:POKE251,INT(HH/8)*8+2
15 FORI=52192TO52432:READII:POKEI,II:NEXT
20 BA=57344:SYS52192:GOTO150
30 IFX<0ORX>319ORY<0ORY>199THENRETURN
40 A=(XAND504):AA=7-(XAND7)
50 B=(YAND248)/8:BB=YAND7
60 P=BA+B*320+A+BB:POKE253,2↑AA
70 POKE255,P/256:POKE254,(P/256-PEEK(255))*256+.5
80 SYS52414:RETURN
90 DATA173,2,221,9,3,141,2,221,173,0,221,41,252,141
,0,221,173,24,208,41,1,9,8
95 DATA141,24,208,173,17,208,9,32,141,17,208,169,16
,162,0,157,0,192,157,0,193
100 DATA157,0,194,157,0,195,202,208,241,169,224,133
,255,169,0,133,254,168,145
105 DATA254,200,208,251,230,255,208,247,120,169,204
,141,21,3,169,82,141,20,3
110 DATA173,14,220,41,254,141,14,220,169,1,141,26,2
08,169,119,133,252,169,0,141
115 DATA18,208,173,17,208,41,127,141,17,208,88,96,1
65,252,208,50,173,2,221,9
120 DATA3,141,2,221,173,0,221,41,252,9,3,141,0,221,
169,20,141,24,208,173,17,208
125 DATA41,223,141,17,208,165,251,141,18,208,173,17
,208,41,127,141,17,208,169
130 DATA119,133,252,208,46,173,2,221,9,3,141,2,221,
173,0,221,41,252,141,0,221
135 DATA169,8,141,24,208,173,17,208,9,32,141,17,208
,169,0,141,18,208,173,17,208
140 DATA41,127,141,17,208,169,0,133,252,169,1,141,2
5,208,76,49,234,120,169,5
145 DATA133,1,160,0,177,254,5,253,145,254,169,7,133
,1,88,96
150 REM ******************************
160 REM * TUTTE LE RIMANENTI LINEE *
170 REM * OSPITERANNO IL PROGRAMMA *
180 REM * CHE SFRUTTA TALE GRAFICA *
190 REM * 'X' FRA 0 E 319          *
200 REM * 'Y' FRA 0 E 199          *
210 REM * GOSUB 30                 *
220 REM ******************************
```

Listato 4

```
10 FORI=49408TO49617:READII:POKEI,II:NEXT
20 SYS49408
30 REM ********************************
40 REM *                              *
50 REM *        16   S P R I T E      *
60 REM *                              *
61 REM *    ----------------------    *
62 REM *    (C) 1985  ADP SOFTWARE    *
63 REM *    ----------------------    *
64 REM *                              *
70 REM ********************************
1000 DATA120,169,193,141,21,3,169,51,141,20,3,173,1
7,208,41,127,141,17,208,169
1010 DATA251,141,18,208,169,1,133,254,141,26,208,17
3,14,220,41,254,141,14,220
1020 DATA169,0,162,112,157,0,192,202,16,250,88,96,1
69,1,141,25,208,165,254,208
1030 DATA75,169,1,133,254,173,30,208,141,94,192,173
,31,208,141,95,192,162,16
1040 DATA189,0,192,157,0,208,202,16,247,162,9,189,3
7,192,157,37,208,202,16,247
1050 DATA173,21,192,141,21,208,173,23,192,141,23,20
8,162,2,189,27,192,157,27
1060 DATA208,202,16,247,162,7,189,48,192,157,248,7,
202,16,247,76,49,234,169,0
1070 DATA133,254,173,30,208,141,30,192,173,31,208,1
41,31,192,162,16,189,64,192
1080 DATA157,0,208,202,16,247,162,9,189,101,192,157
,37,208,202,16,247,173,85
1090 DATA192,141,21,208,173,87,192,141,23,208,162,2
,189,91,192,157,27,208,202
1100 DATA16,247,162,7,189,56,192,157,248,7,202,16,2
47,76,49,234
```