Commodore 64

MicroComputer

Manuale d'uso

commodore COMPUTER

AVVERTENZE

Questa apparecchiatura genera ed usa energia a radiofrequenza e se non è installata ed usata correttamente e cioè in stretta conformità con le istruzioni del fabbricante, può provocare interferenze alla ricezione radio e televisiva. Essa è stata omologata ed è risultata conforme ai limiti per i dispositivi di calcolo Classe B e conforme alle specifiche nella Subpart J della Part 15 delle regole FCC che sono intese a fornire ragionevole protezione a fronte di tale interferenza nelle installazioni residenziali. In ogni caso non c'è garanzia che non si verifichi interferenze in una particolare installazione. Se questa apparecchiatura provoca interferenza alla ricezione radio o televisiva, che può essere determinata spegnendo e riaccendendo l'apparecchiatura stessa, l'utente è invitato a cercare di correggere l'interferenza mediante una o più delle seguenti misure:

- riorientare l'antenna ricevente
- riposizionare il computer rispetto al ricevitore
- spostare il computer allontanandolo dal ricevitore
- collegare il computer a prese diverse in modo che computer e ricevitore siano alimentati da circuiti derivati diversi

Se necessario l'utente dovrà consultare il rivenditore o un tecnico radiotelevisivo esperto per ulteriori suggerimenti. L'utente potrà trovare utile il seguente opuscolo preparato dalla Federal Communications Commission: «How to Identify and Resolve Radio-TV Interference Problems», che può essere richiesto all'U.S. Government Printing Office, Washington, D.C. 20402, N. codice 004-000-00345-4.

COMMODORE 64
GUIDA PER L'USO

Pubblicato dalla
Commodore Italiana S.r.l.

TAVOLA DEI CONTENUTI

INTRODUZIONE

Siete ora l'orgoglioso proprietario del COMMODORE 64, per cui vi facciamo le nostre più vive congratulazioni per aver acquistato uno dei migliori computer del mondo. La COMMODORE è nota come la società del *computer amico* ed essere amici significa fornire manuali di istruzione facili da leggere, da comprendere e da usare. La GUIDA PER L'USO DEL COMMODORE 64 contiene tutte le informazioni necessarie per disporre opportunamente l'apparecchiatura, prendere conoscenza con il COMMODORE 64 ed avviarvi in maniera facile e divertente ad imparare a compilare i propri programmi.

Per coloro che non intendono imparare la programmazione, abbiamo inserito tutte le informazioni che vi occorrono per usare i programmi COMMODORE o altri programmi pronti e/o cassette per giochi (software di terzi) nella parte iniziale. Ciò significa che non occorre effettuare ricerche in tutto il manuale per cominciare ad usare il computer.

Diamo ora uno sguardo ad alcune delle interessanti caratteristiche che vi stanno aspettando all'interno del vostro COMMODORE 64. Innanzitutto se dovete costruire dei grafici, avete a disposizione il «costruttore di immagini» più avanzato di tutta l'industria dei microcomputer, che abbiamo chiamato GRAFICI DI ANIMAZIONE e che vi consente di disegnare vostre immagini in quattro colori diversi, esattamente come quelle che vedete nei videogiochi delle sale specializzate. Non solo, ma l'EDITOR DI ANIMAZIONE vi consente di animare fino a 8 diversi livelli di immagini alla volta. Potete cioè spostare le vostre creazioni ovunque sullo schermo, addirittura far passare un'immagine davanti o dietro ad un'altra. Il vostro COMMODORE 64 consente inoltre il rilevamento automatico di collisione che istruisce il computer ad intraprendere l'azione appropriata quando le immagini animate si urtano l'una con l'altra.

Il COMMODORE 64 dispone inoltre di effetti musicali sonori incorporati le cui capacità sfidano quelle di molti sintetizzatori musicali. Potete così disporre di tre voci o timbri indipendenti, ciascuna con un campo di 9 ottave intere «tipo pianoforte». Inoltre avete la possibilità di lavorare con quattro diverse forme d'onda (a dente di sega, a triangolo, ad impulso variabile e di rumore), un generatore ADSR programmabile (salita, discesa, piano, smorzo), un generatore di inviluppi, filtri programmabili alti, bassi e passabanda per ciascuna voce nonchè regolazioni di volume e di risonanza variabili. E se volete che la vostra musica venga riprodotta con attrezzature sonore professionali, COMMODORE 64 vi consente di collegare la vostra uscita audio a pressochè qualsiasi sistema di amplificazione di alta qualità.

E giacchè stiamo parlando di collegare il COMMODORE 64 ad altre apparecchiature . . . cogliamo l'occasione per dire che il vostro sistema può essere ampliato aggiungendogli accessori, noti come periferiche, man mano che le vostre esigenze di calcolo crescono. Alcune delle opzioni previste comprendono un registratore DATASSETTE* o fino a cinque unità di memoria a dischi VIC 1541 per i programmi da voi realizzati e/o che volete eseguire. Se disponete già di un'unità disco VIC 1540 il rivenditore potrà aggiornarla per l'impiego con COMMODORE 64.

Potete aggiungere una stampante a matrice VIC per ottenere copie stampate o tabulati di programmi, lettere, fatture, ecc. . . Se volte collegarvi con computer più potenti e con le relative massicce data base vi basta inserire ad innesto un caricatore VICMODEM per avere a disposizione i servizi di centinaia di specialisti e per accedere a numerose reti di informazioni utilizzando il telefono d'ufficio o di casa. Infine, se siete interessanti agli infiniti programmi applicativi disponibili in CP/M**, potete munire COMMODORE 64 di un microprocessore a innnesto Z-80.

Altrettanto importante di tutto l'hardware disponibile è il fatto che questa GUIDA PER L'USO sia in grado di aiutarvi a sviluppare la vostra conoscenza del computer. Essa non vi dirà tutto ciò che dovete sapere sui computer ma vi rinvierà a numerose pubblicazioni dalle quali potrete attingere informazioni più dettagliate sugli argomenti di vostro interesse. Noi vogliamo che vi godiate veramente il vostro nuovo COMMODORE 64. E per divertirvi, ricordate: la programmazione non è il tipo di cosa che potete imparare in un giorno. Siate pazienti e leggete a fondo la GUIDA PER L'USO. Ma prima di iniziare, dedicate qualche minuto a compilare ed a spedire la cartolina di registrazione fornita insieme al computer. Ciò vi assicura che il vostro COMMODORE 64 sia correttamente registrato presso la sede della COMMODORE in modo da poter ricevere le informazioni più aggiornate riguardanti i miglioramenti futuri che lo riguardano. Buon divertimento!

NOTA: Molti programmí sono ancora in fase di sviluppo nel momento in cui questo manuale viene stampato. Vi consigliamo quindi di rimanere in contatto con il rivenditore locale COMMODORE e con i vari clubs e riviste di utenti Commodore che vi terranno aggiornati sulla enorme quantità di programmi applicativi che vengono scritti in tutto il mondo per COMMODORE 64.

---

* DATASSETTE è un marchio di fabbrica registrato della COMMODORE Business Machines, Inc.

** CP/M è un marchio di fabbrica registrato della Digital Research Inc. Le specifiche possono essere variate senza preavviso.

CAPITOLO 1

PRIMI APPROCCI CON IL COMMODORE 64

CONTENUTO DELL'IMBALLO E COLLEGAMENTO DEL COMMODORE 64

Le segueti istruzioni mostrano come collegare il COMMODORE 64 ad un comune televisore, ad un sitema sonoro o ad un monitor. Prima di collegare qualsiasi cosa al computer controllare il contenuto dell'imballo del COMMODORE 64: Insieme a questo munuale, si dovrebbe trovare quanto segue:

1. COMMODORE 64
2. Alimentatore (piccola scatola con una spina di alimentazione AC e un cavo con la presa per il COMMODORE 64)
3. Cavo coassiale per antenna

Se uno qualsiasi di questi componenti e mancante rivolgiti immediatamente al tuo rivenditore.

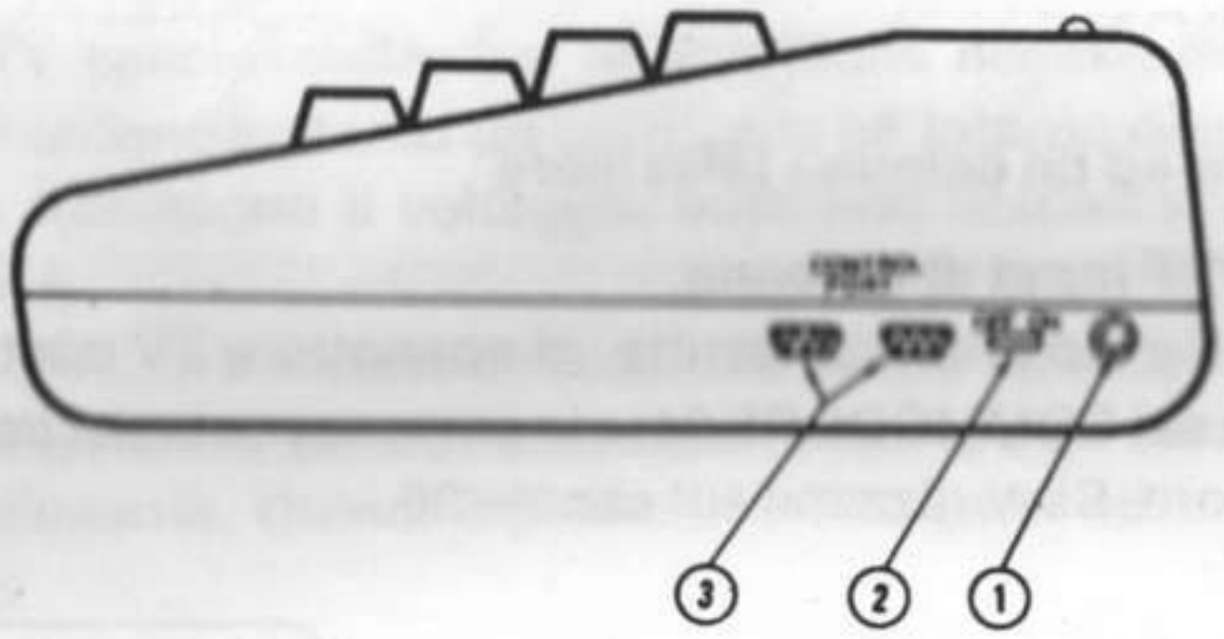
Vediamo innanzitutto dove sono sistemate e come funzionano le varie porte di questo computer.

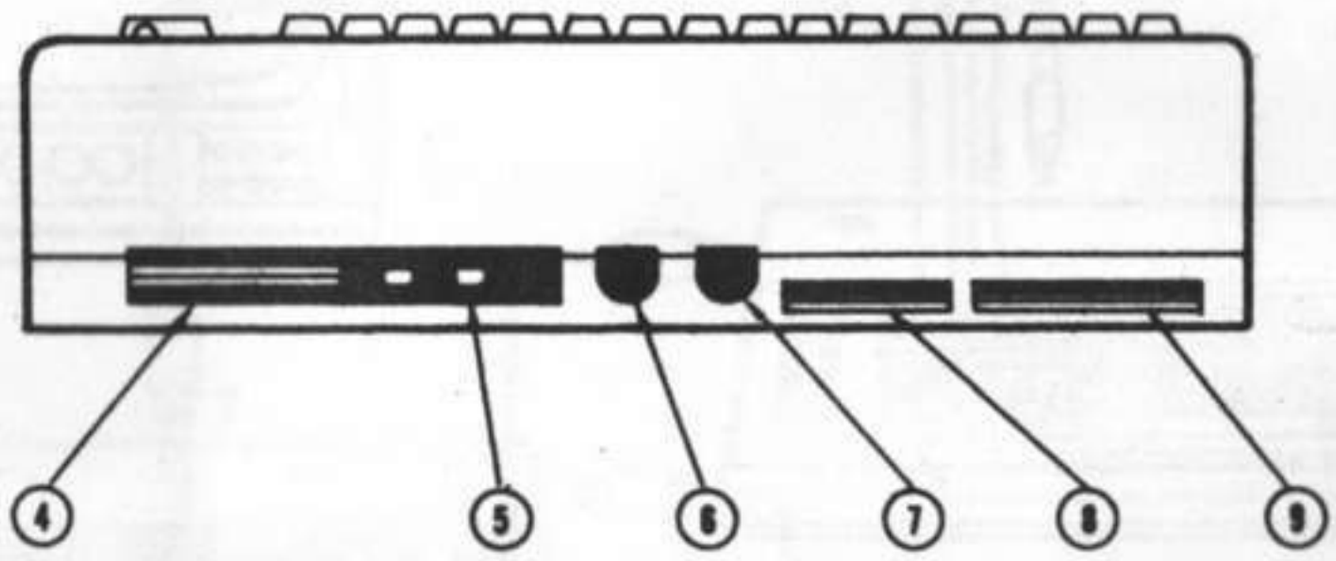
CONNETTORI SITUATI SULLA PARTE DESTRA DEL COMMODORE 64

1. **Presa di alimentazione,** la presa del cavo di alimentazione va collegata qui.
2. **Interruttore di alimentazione**
3. **Porta per il comando giochi,** ad ognuno di questi connettori può essere collegato un joystick, paddle, o una penna ottica.

CONNETTORI POSTERIORI

4. **Connettore per cartuccia,** questa porta rettangolare accetta programmi o giochi memorizzati su cartucce.
5. **Connettore-TV,** questo connettore fornisce sia il segnale video che quello audio all'input antenna (75 Ω) del televisore.
6. **Audio e video output,** questa porta fornisce direttamente il segnale audio che può essere collegato ad un sistema Hi-Fi. Mette anche a disposizione un segnale video che può pilotare un TV-monitor.



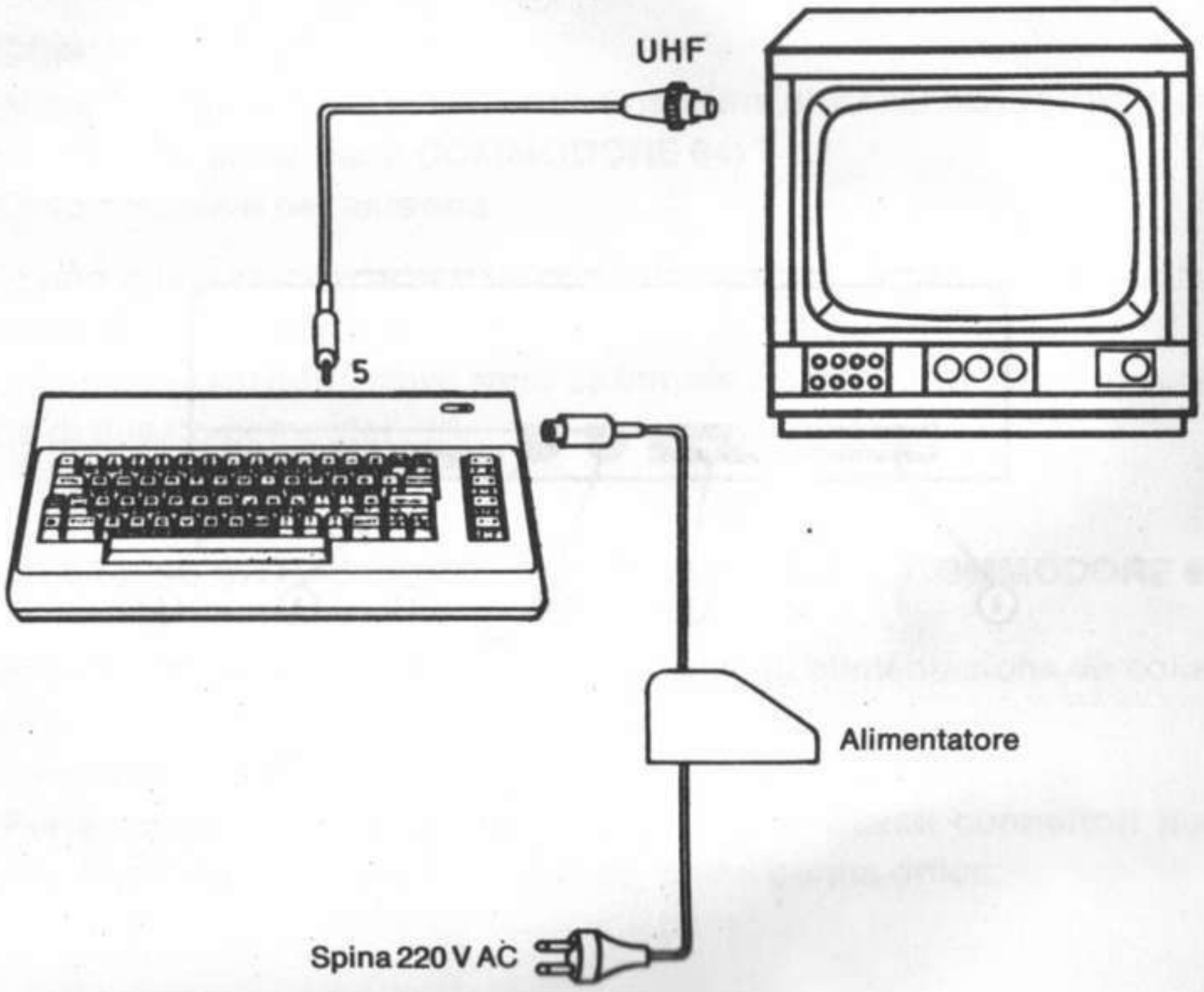


7. **Porta seriale,** alla quale si può collegare una stampante e un floppy disk singolo.
8. **Interfaccia per registratore,** un registratore a cassette può essere collegato al computer per poter salvare e poi caricare programmi e dati in memoria.
9. **User port,** porta libera e programmabile di input/output e allo stesso tempo connettore per cartucce ad inserimento come l'interfaccia RS232.

ISTALLAZIONE

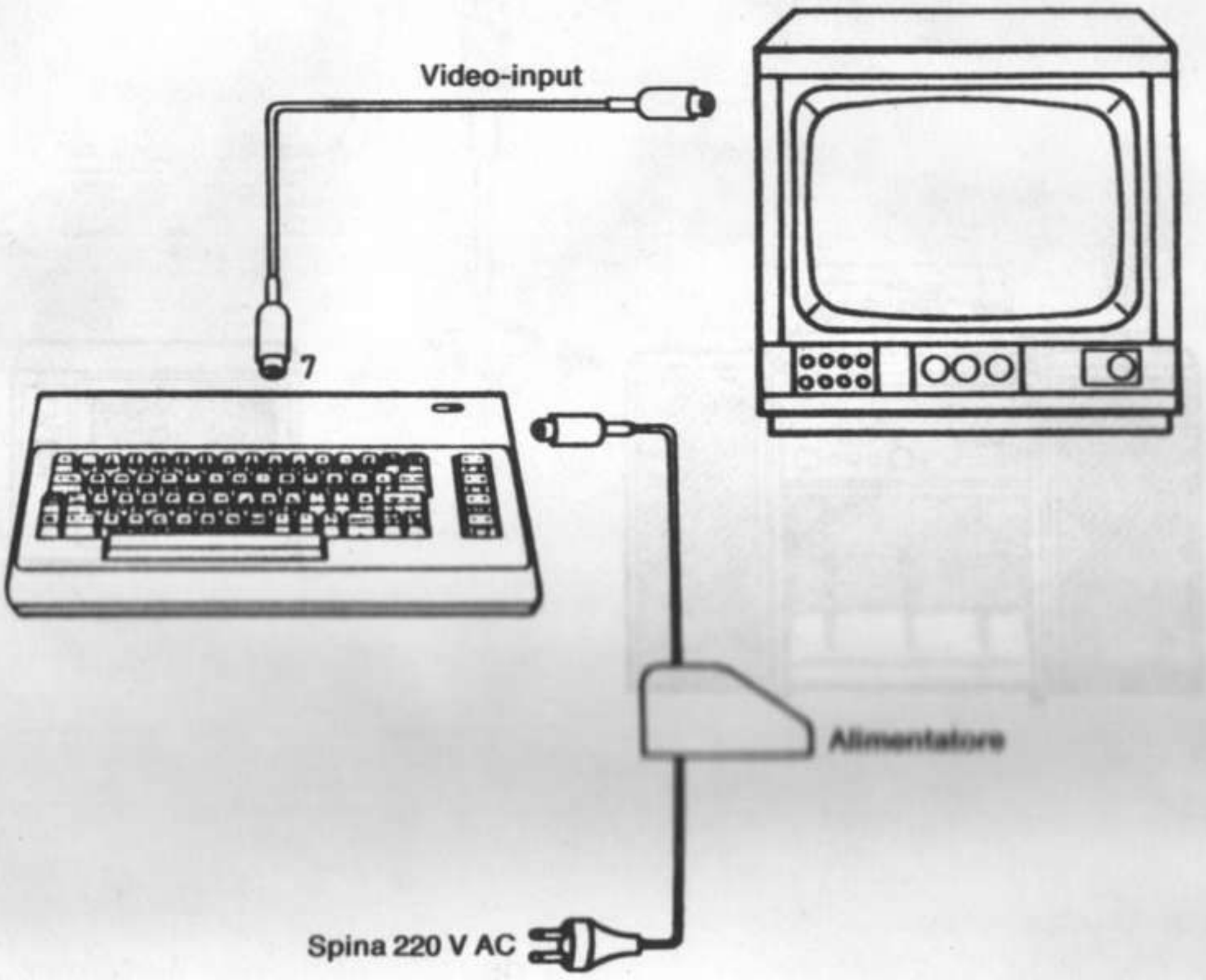
Collegamento ad un comune televisore

1. **Uso dell'UHF input di antenna.**
   Collegare il cavo antenna fornito, al connettore TV posto sulla parte posteriore del COMMODORE 64, e la parte opposta del cavo alla presa del televisore. Sintonizzarsi sul canale 36.



2. **Uscita video**
   Usando un TV-monitor, il miglior risultato si ottiene se questo è a colori. Occore un cavo coassiale (non fornito) con una spina a 5 poli (DIN 41524) che va inserita nel connettore 6 del COMMODORE 64. Sul lato opposto del cavo, occore un connettore video (DIN 45322) da inserire nel monitor. Se il televisore e già predisposto con un con una presa video, lo si può usare anche come monitor. Per fare questo bisogna assicurarsi che il connettore video del televisore sia usato come «input».
   Per commutare questo connettore come input, viene fornito un voltaggio ausiliario di 12 V, al polo 1 del medesimo; ma il COMMODORE 64 non fornisce questo voltaggio. Si prega di consultare un

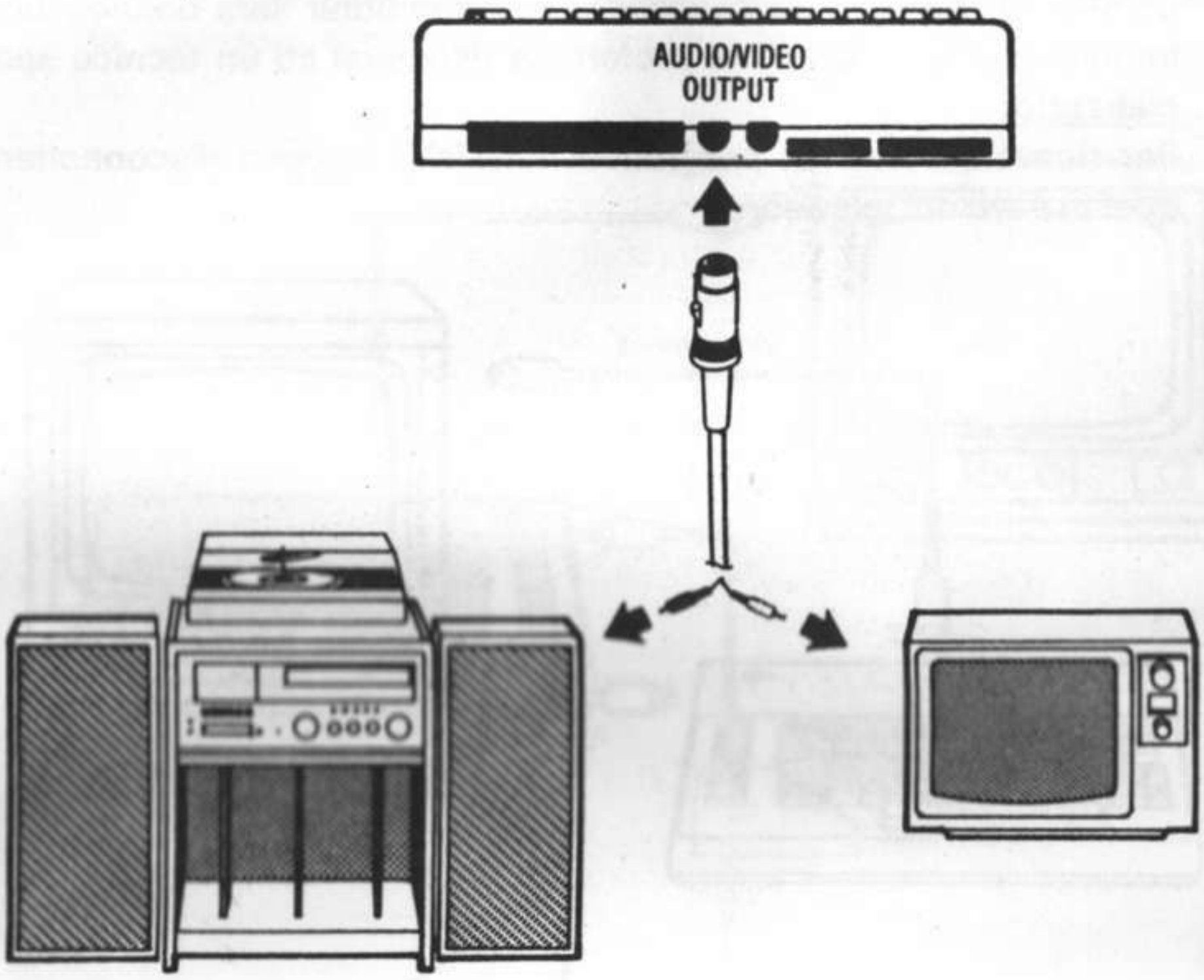
tecnico-TV specializzato, per le modifiche necessarie. Alcuni televisori richiedono soltanto un ponticello all'interno della presa video, inquanto forniscono il voltaggio ausiliario richiesto, sul polo 5 del connettore.
ATTENZIONE, in nessuna circostanza questi 12 V devono arrivare al COMMODORE 64, se ciò accadesse il computer sarà danneggiato immediatamente. Quindi è preferibile rivolgersi ad un tecnico specializzato.
Per ricevere i normali programmi televisivi bisogna disconnettere questo cavo del televisore.



3. **Connessione alimentazione**
   Collegare l'alimentatore al COMMODORE 64 (connettore 1) ed inserire la spina-AC alla presa a muro.

4. Collegamenti opzionali

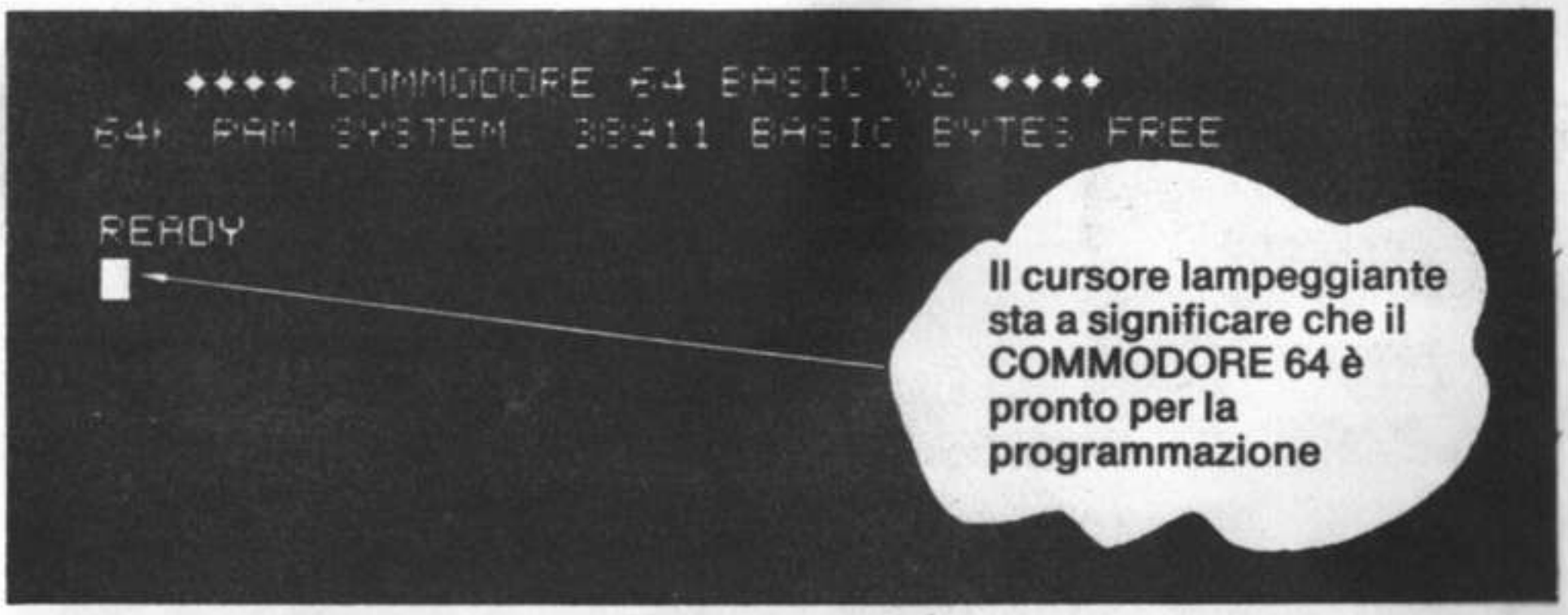
Il COMMODORE 64 fornisce un canale audio in alta fedeltà, quindi si ha la possibilità, se desiderato, di collegarlo con un amplificatore di qualità. Il segnale sonoro si trova sul polo 3 del connettore 6 (vedi figura in basso).



Il COMMODORE 64 ha la possibilità di usare come unità periferiche il floppy disk singolo VIC 1541 e la stampante VIC 1525.
Questo sistema completo viene mostrato nella figura in basso.

COLLAUDO

1. **Accendere il computer usando l'interruttore situato sullo parte destra della tastiera.**
2. **Dopo alcuni secondi il seguente messaggio comparirà sullo schermo.**



3. **Se il televisore ha il pomello per la sintonia manuale, muoverlo fino a quando si ottiene una immagine limpida. Se invece il televisore ha un AFC, si sintonizzerà automaticamente.**
4. **Si può anche aggiustare il contrasto ed il colore così da ottenere la migliore visualizzazione sul televisore.**

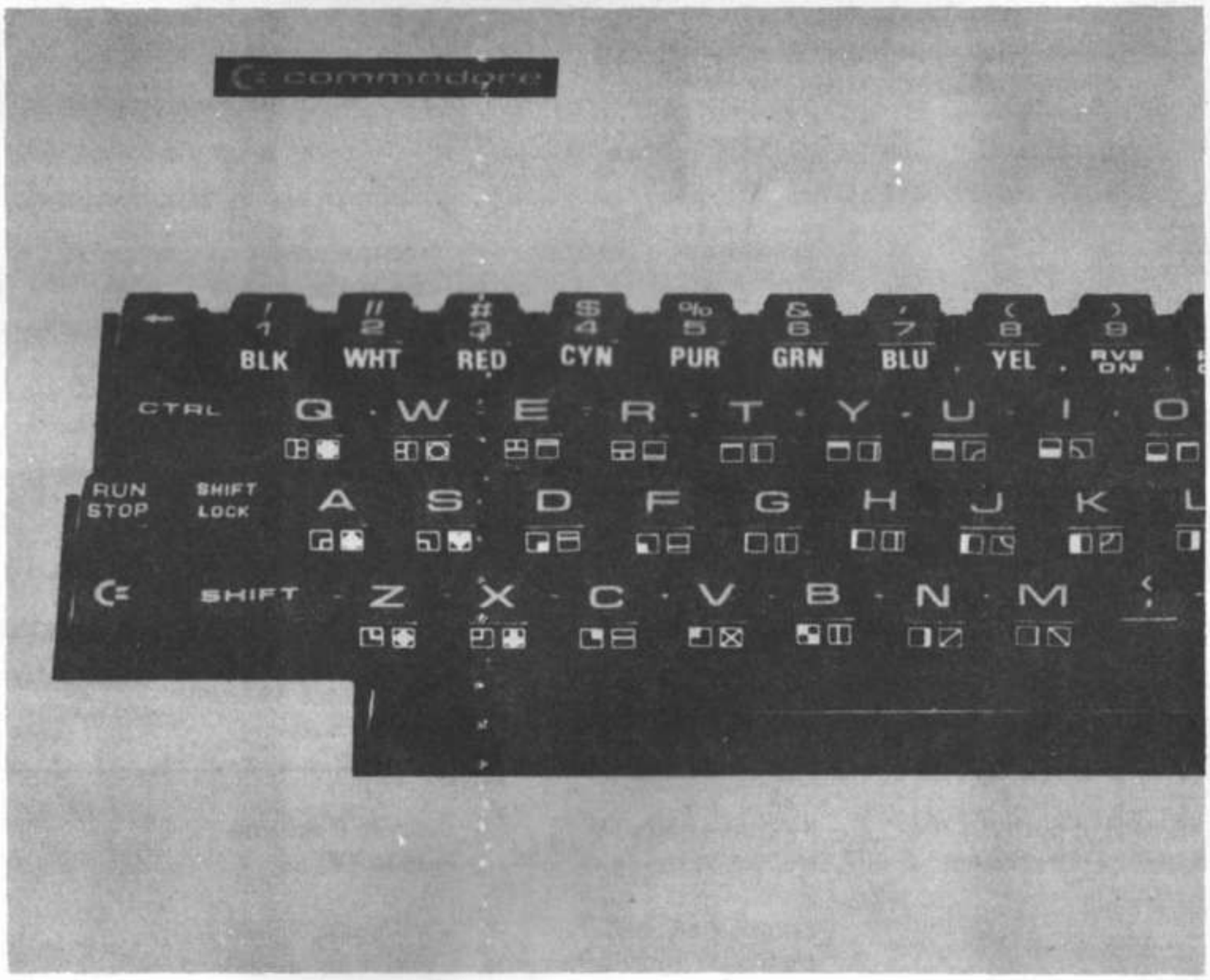
Se non si ottengono i risultati previsti controllare ancora una volta i collegamenti e i cavi. La tavola seguente sarà di aiuto per isolare qualsiasi problema.

SINTOMO	CAUSA	RIMEDI
La spia di accenzione è spenta	computer «spento»	assicurarsi che l'interruttore sia nella posizione «ON»
	cavo alimentazione non collegato	controllare se la presa di alimentazione ha falsi contatti
	il fusibile è soltato	rivolgersi ad un centro autorizzato per la sostituzione del fusibile
L'immagine video non compare sullo schermo	TV sintonizzato sul canale sbagliato	controllare su altri canali per ottenere l'immagine
	collegamento incorretto	il computer si collega al terminale antenna UHF
	schermo non collegato	controllare il collegamento dell'output video
A cartuccia inserita il segnale video è disturbato	cartuccia inserita incorrettamente	reinserire la cartuccia dopo aver spento il computer
Schermo in bianco/nero o con colori sbiaditi	TV non sintonizzato perfettamente	sintonizzare la TV
Immagine con eccessivi rumori di fondo	il volume della TV è troppo alto	abbassare il volume della TV
Immagine perfetta, ma assenza del sonoro	il volume della TV troppo basso	alzare il volume della TV
	output ausiliario collegato impropriamente	collegare il jack audio dell'input ausiliario selezionato dell'amplificatore

REGOLAZIONE DEI COLORI

C'è un modo semplice per ottenere un profilo di colori sul televisore in modo da poter facilmente regolare l'apparecchio. Quantunque per il momento non si abbia ancora familiarità con il funzionamento del computer, basta procedere per vedere come è facile usare il COMMODORE 64.

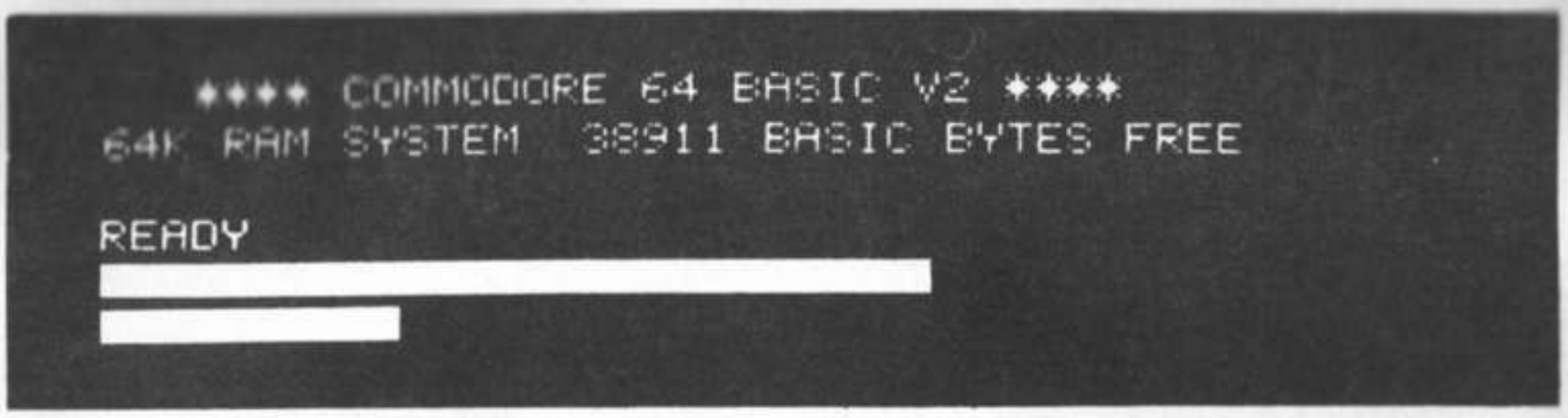
Osservare innanzitutto il lato sinistro della tastiera ed individuare il tasto contrassegnato CTRL. La scritta è l'abbreviazione di ConTRoL e questo tasto viene usato unitamente ad altri per istruire il computer a svolgere un compito specifico.



Per usare una funzione di controllo occorre tenere abbassato il tasto CTRL ed abbassare contemporaneamente un secondo tasto.

Provare in questo modo: tenere abbassato il tasto CTRL mentre si preme contemporaneamente il tasto 9 Sollevare quindi entrambi i tasti. Non dovrebbe essere successo nulla di ovvio ma se ora si preme qualsiasi altro tasto, lo schermo mostra il carattere visualizzato in negativo anzichè nel carattere normale – come il messaggio di apertura o qualsiasi cosa che è stata battuta precedentemente.

Tenere abbassato il tasto SPACE BAR. Cosa succede? Se è stata eseguita la suddetta procedura correttamente, fintantoche rimane abbassata la barra di spazio SPACE BAR si dovrebbe vedere una barra azzurra spostarsi attraverso lo schermo e quindi abbassarsi alla riga successiva.

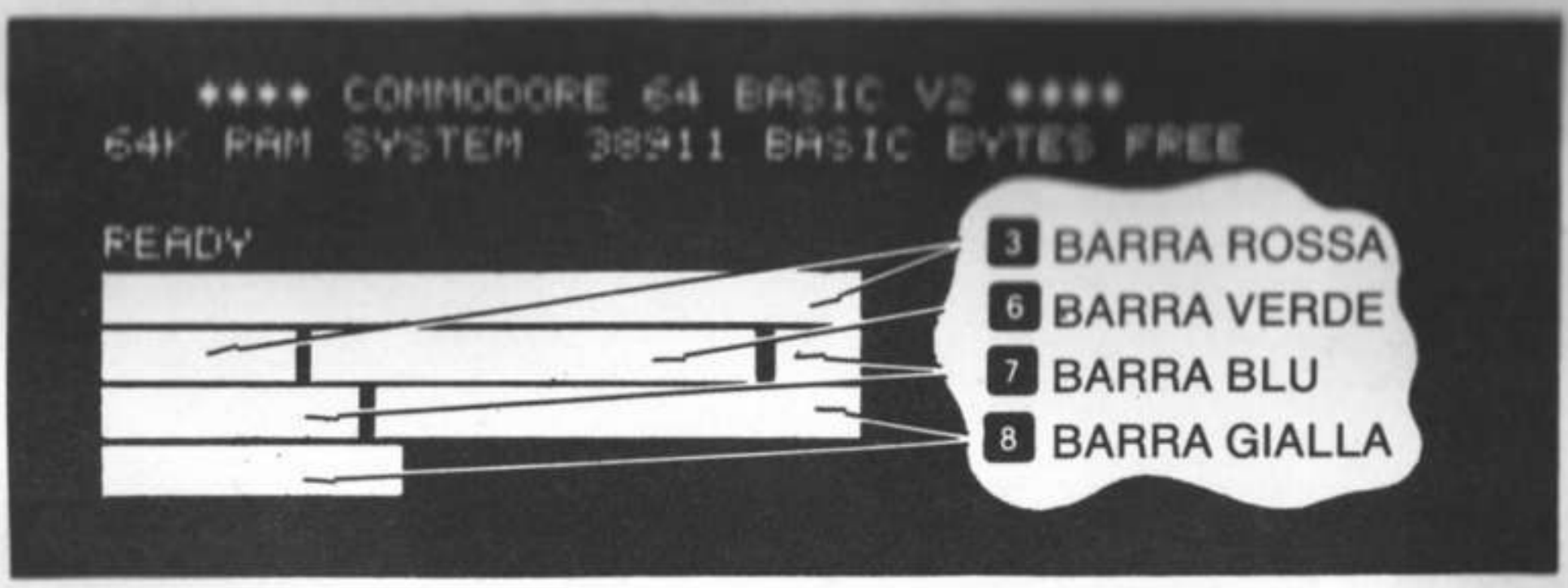


Ora, tenere abbassato CTRL mentre si preme uno qualsiasi degli altri tasti numerici, ciascuno dei quali è contrassegnato nella parte anteriore da un colore. Qualsiasi cosa verrà visualizzata da questo punto in avanti sarà in quel colore. Per esempio, tenere abbassato CTRL e il tasto 8 e quindi rilasciarli entrambi. Premere ora il tasto SPACE BAR.

Osservare lo schermo. La barra è ora di color giallo! In maniera analoga è possibile cambiare il colore della barra scegliendolo fra quelli indicati sui tasti numerici premendo CTRL ed il tasto appropriato.

Provare a cambiare per due o tre volte il colore della barra, quindi regolare luminosità e contrasto del televisore in modo che ciò che si vede sullo schermo corrisponda ai colori scelti.

Lo schermo dovrebbe apparire più o meno come segue:



A questo punto tutto è correttamente regolato e funziona perfettamente. I capitoli seguenti presenteranno il linguaggio BASIC. In ogni caso è possibile iniziare immediatamente ad usare alcune delle applicazioni «pronte» e i giochi disponibili per il COMMODORE 64 senza conoscere nulla sulla programmazione di computer.

Ciascuno di questi «packages» contiene informazioni dettagliate sul modo in cui usare il programma. Si suggerisce comunque di legger prima i capitoli di questo manuale per prendere maggior familiarità con il funzionamento base del nuovo sistema.

CAPITOLO 2

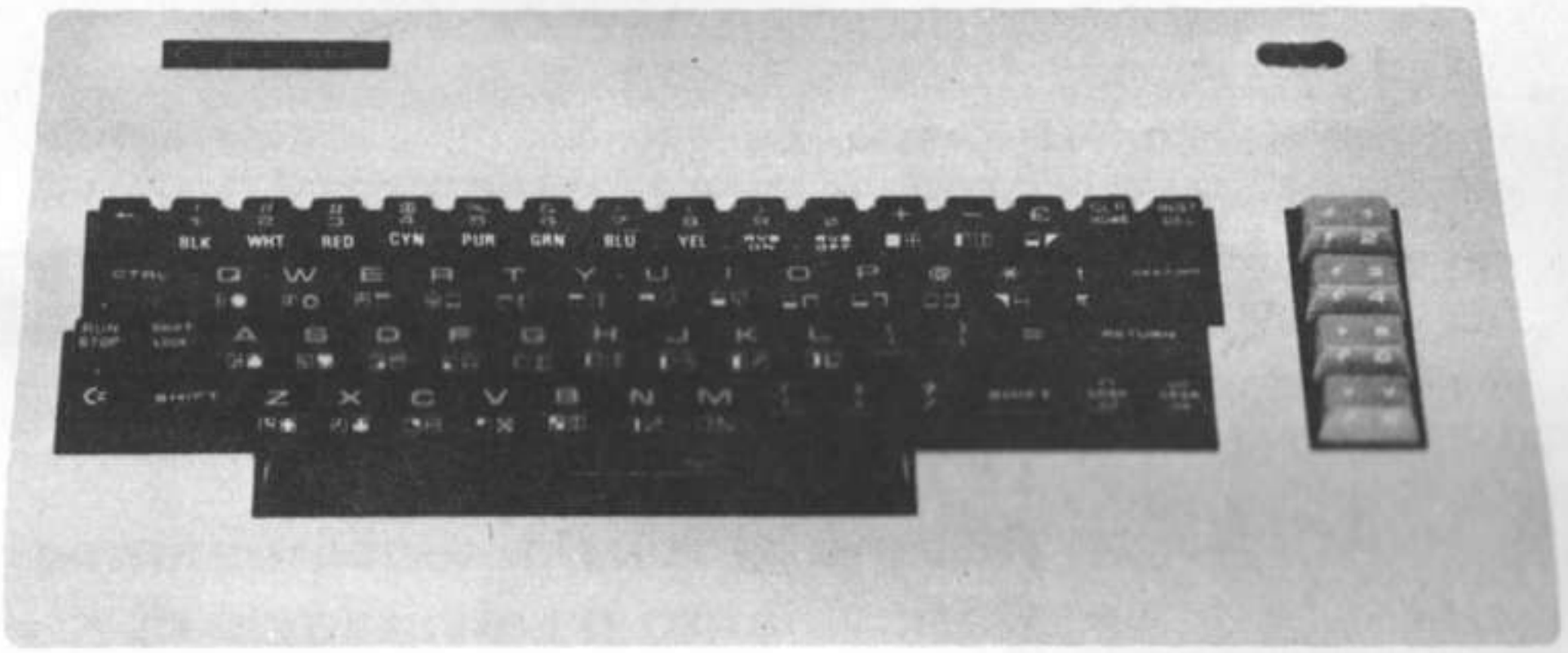
PER COMINCIARE

- **La tastiera**
- **Ritorno al funzionamento normale**
- **Caricamento e salvataggio di programmi**
- **PRINT e calcoli**
- **Precedenza**
- **Come combinare le cose**

LA TASTIERA

Ora che tutto è messo a punto e regolato, occorre dedicare qualche instante e prendere familiarità con la tastiera che è il più importante mezzo di comunicazione con il COMMODORE 64.

Sotto molti aspetti la tastiera è simile di una normale macchina da scrivere. Ci sono comunque numerosi nuovi tasti che controllano funzioni specializzate. Segue ora una breve descrizione dei vari tasti e delle rispettive funzioni. Il funzionamento dettagliato di ciascun tasto sarà illustrato in successivi capitoli.



RETURN

Il tasto RETURN segnala al computer di osservare l'informazione battuta ed immette quell'informazione in memoria.

SHIFT

Il tasto SHIFT funziona come quello di una normale macchina da scrivere. Molti tasti sono in grado di visualizzare due lettere o simboli e due caratteri grafici. Nel modo «maiuscolo/minuscolo» il tasto SHIFT fornisce i caratteri standard maiuscoli. Nel modo «maiuscolo/grafici» il tasto SHIFT visualizza il carattere grafico riportato sul lato destro del tasto.

Nel caso dei tasti speciali di funzione, il tasto SHIFT dà la funzione contrassegnata sulla parte superiore del tasto stesso.

LE CORREZIONI

Nessuno è perfetto e COMMODORE 64 lo sa. Un certo numero di tasti di correzione consente di rimediare agli errori di battitura e di spostare le informazioni sullo schermo.

CRSR

Ci sono dei tasti contrassegnati CRSR (CuRSoR-cursore), uno con freccia verso l'alto e verso il basso ⇑ CRSR ⇓, l'altro con frecce verso destra e verso sinistra ⇐ CRSR ⇒. E' possibile usare questi tasti per spostare il cursore verso l'alto e verso il basso o verso sinistra e verso destra. Nel modo non preceduto da shift, i tasti CRSR consentono di spostare il cursore verso il basso e verso destra. L'uso contemporaneo dei tasti SHIFT e CRSR consente di spostare il cursore verso l'alto o verso sinistra. I tasti di spostamento del cursore dispongono di una speciale funzione di ripetizione che mantiene il cursore in movimento fino a che non si libera il tasto.

INST/DEL

Se si batte il tasto INST/DEL, il cursore si sposta di uno spazio, cancellando (DELeting-cancellazione) il carattere precedentemente battuto. Se ci si trova nel mezzo di una riga, il carattere alla sinistra viene cancellato ed i caratteri alla destra si spostano automaticamente per occuparne lo spazio.

Premendo il tasto INST/DEL preceduto dal tasto SHIFT è possibile inserire (INSerT-inserimento) informazioni su una riga. Per esempio, se ci si accorge di un errore di un battuta all'inizio di una riga – probabilmente si è saltato una parte di un nome – si può usare il tasto ⇐ CRSR ⇒ per ritornare sull'errore e quindi battere INST/DEL per inserire uno spazio. Basta quindi battere la lettera mancante.

CLR/HOME

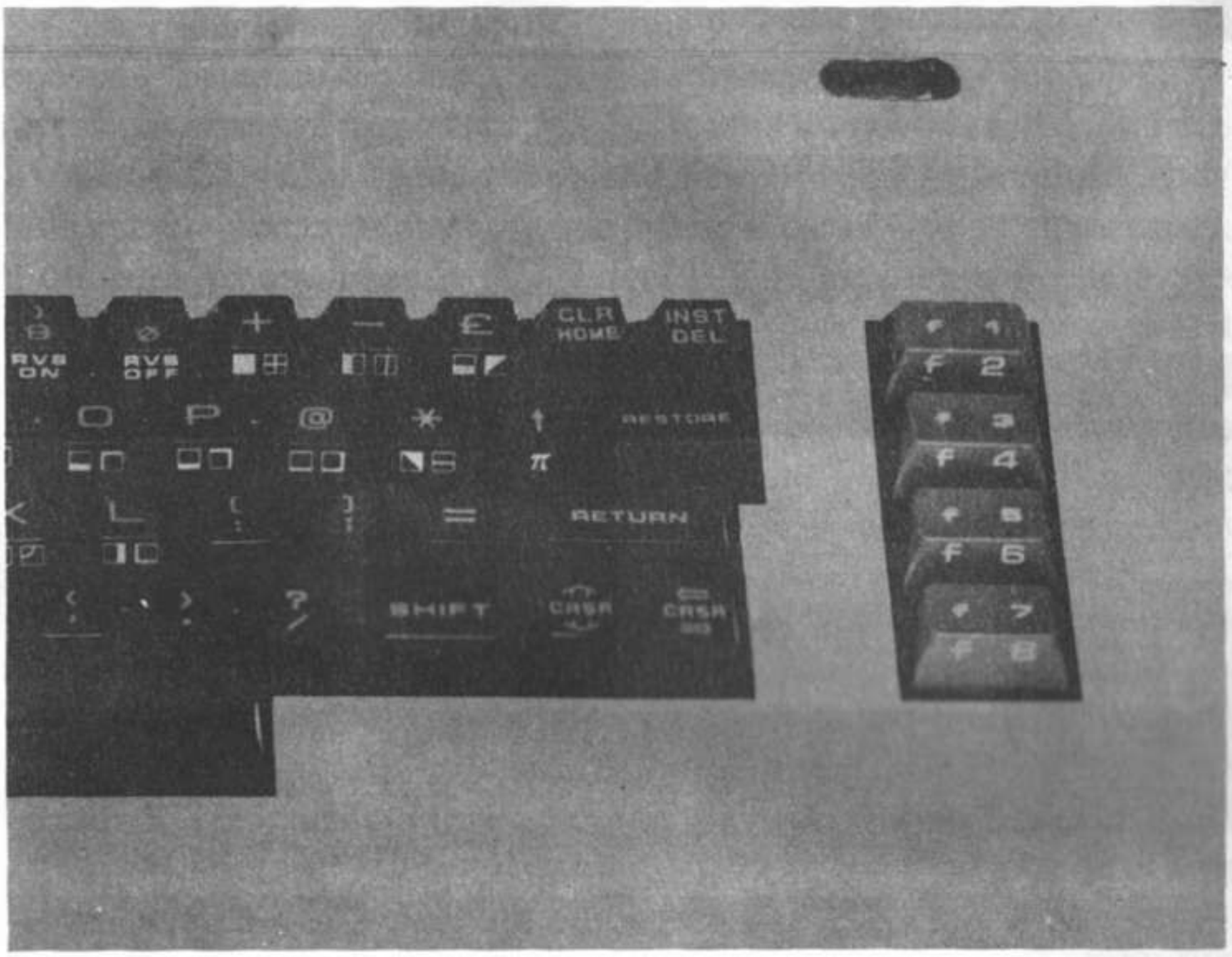
Il tasto CLR/HOME porta il cursore sulla posizione «HOME» ossia sulla posizione di partenza, che si trova nell'angolo superiore sinistro dello schermo. La pressione del tasto CLR/HOME preceduta dal tasto SHIFT cancella lo schermo e riporta il cursore in posizione di partenza.

RESTORE

Il tasto RESTORE (ripristino) funziona come implica lo stesso nome e cioè ripristina il computer alla condizione normale in cui si trovava prima di modificarla con un programma o qualche comando. Di questo tasto si parlerà a lungo nei prossimi capitoli.

I TASTI DI FUNZIONI

I quattro tasti posti sul lato destro della tastiera possono essere «programmati» per svolgere numerose funzioni. Essi possono cioè essere definiti in molti modi per svolgere i rispettivi compiti.



CTRL

Il tasto CTRL che significa ConTRoL (controllo) consente di definire i colori e di eseguire altre funzioni specializzate. Per accedere ad una funzione di controllo basta tener abbassato il tasto CTRL mentre si preme il tasto corrispondente alla funzione desiderata. Si è già avuto la possibilità di provare il funzionamento del tasto CTRL quando si sono cambiati i colori del testo per creare diverse barre colorate durante la procedura di messa a punto.

RUN/STOP

Normalmente la pressione del tasto RUN/STOP interrompe l'esecuzione di un programma BASIC. Esso segnala cioà al computer di interrompere (STOP) l'esecuzione di qualche cosa. L'uso del tasto RUN/STOP preceduto dal tasto SHIFT consente di caricare automaticamente un programma da nastro.

IL TASTO COMMODORE C=

Il tasto COMMODORE C= esegue numerose funzioni. Prima di tutto consente di passare sullo schermo dal modo testo al modo grafici.

All'accensione, il computer è predisposto nel modo maiuscole/grafici per cui tutto ciò che viene battuto comparirà in lettere maiuscole. Come si è già detto, l'uso del tasto SHIFT in questo modo visualizza il carattere grafico posto sul lato destro dei tasti.

Se si tiene abbassato il tasto C= e si preme contemporaneamente il tasto SHIFT, lo schermo passa ai caratteri maiuscoli e minuscoli. Ora, se si tiene abbassato il tasto C= e si preme qualsiasi altro tasto con un simbolo grafico, compare il simbolo grafico posto sul lato sinistro del tasto.

Per ritornare al modo maiuscole/grafici, tenere abbassato il tasto C= e premere di nuovo SHIFT.

La seconda funzione del tasto C= è di consentire l'accesso ad una seconda serie di otto colori di testo. Tenendo abbassato il tasto C= ed uno qualsiasi dei tasti numerici, qualsiasi testo sarà battuto nel colore alternativo disponibile in relazione al tasto abbassato. Il Capitolo 5 elenca i colori di testo disponibili ciascun tasto.

RITORNO AL FUNZIONAMENTO NORMALE

Ora che si è avuta la possibilità di osservare la tastiera, è possibile esplorare una delle molte capacità del COMMODORE 64.

Se sullo schermo figurano ancora le barre a colori create durante la regolazione del televisore, premere i tasti SHIFT e CLR/HOME. Lo schermo dovrebbe cancellarsi ed il cursore disporsi in posizione di partenza (angolo superiore sinistro dello schermo).

Ora, premere simultaneamente C= ed il tasto 7. La manovra riporta in azzurro il colore del testo. C'è però un'altra operazione richiesta per riportare tutto in condizioni normali. Occorre cioè tenere abbassato il tasto CTRL ed il tasto 0 (Attenzione: Zero e non la O maiuscola) per riportare lo schermo al funzionamento normale. Ci si ricorderà che era stato predisposto lo schermo in negativo con i tasti CTRL 9 per creare le barre colorate (le barre colorate erano in effetti degli spazi in negativo). Se lo schermo fosse stato predisposto nel modo normale durante la prova dei colori, il cursore si sarebbe spostato ma avrebbe lasciato spazi vuoti.

SUGGERIMENTO:

Dopo aver imparato a fare le cose nel modo difficile, ecco un modo semplice per riportare il sistema al modo di funzionamento normale. Premere simultaneamente

RUN/STOP RESTORE

Ciò cancella lo schermo e riporta tutto al funzionamento normale. Se nel computer c'è un programma, questo viene lasciato inalterato. Vale la pena di ricordare questa sequenza, particolarmente quando si esegue molta programmazione.

Se vi vuole ripristinare la macchina come avviene quando la si spegne e la si riaccende, battere: SYS 64738 e premere RETURN . Fare attenzione ad usare questo comando! Esso cancella qualsiasi programma o informazione che si trova correntemente nel computer.

CARICAMENTO E SALVATAGGIO DI PROGRAMMI

Una delle caratteristiche più importanti del COMMODORE 64 è la sua capacità di salvare e caricare programmi su e da cassetta di nastro o disco.

Questa capacità consente di salvare i programmi per utilizzarli in un tempo successivo o di acquistare programmi pronti da usare con il COMMODORE 64.

Assicurarsi che l'unità disco o l'unità datasette siano collegate correttamente.

Caricamento di programmi pronti

Per coloro che sono interessati ad usare soltanto programmi pronti diponibili su cartucce, cassette o disco ecco come procedere:

1. **CARTUCCE:** Il computer COMMODORE 64 dispone di una serie di programmi e di giochi su cartuccia. I programmi offrono una grande varietà di applicazioni personali e gestionali ed i giochi sono quelli che si trovano normalmente nelle sale specializzate e non imitazioni. Per caricare questi giochi, accendere per prima cosa il televisore. Quindi SPEGNERE il COMMODORE 64. **OCCORRE SEMPRE SPEGNERE IL COMMODORE 64 PRIMA DI INSERIRE O RIMUOVERE LA CARTUCCIA ALTRIMENTI LA SI DISTRUGGE!** Successivamente inserire la cartuccia. Accendere ora il COMMODORE 64. Infine battere l'appropriato tasto START come indicato sul foglio di istruzione che viene fornito con ciascun gioco.

2. **CASSETTE:** Usare il registratore DATASETTE e le normali cassette audio che vengono fornite come parte del programma. Assicurarsi che il nastro sia completamente riavvolto all'inizio del primo lato. Quindi basta battere LOAD. Il computer risponde con PRESS PLAY ON TAPE cui occorre rispondere premendo il tasto PLAY sul datasette. A questo punto lo schermo del computer si cancella fino a che non viene trovato il programma. A questo punto sullo schermo comparirà FOUND (NOME DEL PROGRAMMA). Premere ora il tasto C=. L'operazione provoca il caricamento del programma nel computer. Se si vuole interrompere il caricamento basta premere il tasto RUN/STOP.

3. **DISCO:** Usando l'unità disco, inserire delicatamente il disco pre-programmato in modo che la sua etichetta sia rivolta verso l'alto e verso l'operatore. Individuare la piccola tacca sul disco (potrebbe essere coperta con un piccolo pezzo di nastro). Se si inserisce il disco correttamente, la tacca si dovrebbe trovare sul lato sinistro. Una volta che il disco à all'interno chiudere lo sportello di protezione premendo sulla leva. Battere ora LOAD «NOME PROGRAMMA», 8 e battere quindi il tasto RETURN. Il disco frullerà leggermente e sullo schermo comparirà:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

Quando compare READY ed il cursore, battere RUN. A questo punto il programma è pronto per l'uso.

Caricamento di programmi da nastro

Il caricamento di un programma da nastro o da disco è altrettanto semplice. Per caricare da nastro occorre riavvolgere il nastro all'inizio e battere:

```
LOAD "PROGRAM NAME"
```

Se non si ricorda il nome del programma, basta battere LOAD per caricare in memoria il primo programma sul nastro.

Dopo aver premuto RETURN il computer risponde con:

```
PRESS PLAY ON TAPE
```

Dopo aver premuto il tasto di riproduzione, lo schermo si cancella, lasciando soltanto il colore di fondo mentre il computer cerca il programma.

Una volta trovato, sullo schermo compare:

```
FOUND PROGRAM NAME
```

Per caricare (LOAD) il programma, premere il tasto C=. Per uscire dalla procedura caricamento, premere il tasto RUN/STOP. Se si batte il tasto COMMODORE, lo schermo assume il colore del bordo mentre il programma viene caricato.

Al termine della procedura di caricamento, lo schermo ritorna alla condizione normale e compare la richiesta READY.

Caricamento di programmi da disco

Il caricamento di un programma da disco segue la stessa procedura. Battere:

```
LOAD "PROGRAM NAME",8
```

Dopo aver battuto RETURN, il disco inizia a frullare e sullo schermo compare:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

NOTA:

Quando si carica un nuovo programma nella memoria del computer, qualsiasi istruzione che si trovava precedentemente nel computer viene cancellata. Assicurarsi si salvare un programma sul quale si sta lavorando prima di caricarne uno nuovo. Una volta che un programma è stato caricato, è possibile eseguirlo (RUN), listarlo (LIST) o effettuare modifiche e salvare la nuova versione.

Salvataggio di programmi su nastro

Dopo aver immesso un programma, se lo si vuole salvare su nastro, battere:

```
SAVE "PROGRAM NAME"
```

«PROGRAM NAME» (Nome programma) può essere una combinazione di un massimo di 16 caratteri. Dopo aver battuto RETURN, il computer risponde con:

```
PRESS PLAY AND RECORD ON TAPE
```

Premere contemporaneamente i tasti di registrazione e di riproduzione sul datasette. Lo schermo si cancella, facendo comparire il colore del bordo.

Una volta che il programma è salvato su nastro, ricompare la richiesta READY indicando che è possibile iniziare a lavorare su un altro programma o spegnere il computer.

Salvataggio di programmi su disco

Il salvataggio di un programma su disco è ancora più semplice. Battere:

```
SAVE "PROGRAM NAME",8
```

L'8 è il codice del disco ed in questo modo si fa sapere al computer che si vuole salvare il programma sul disco.

Dopo aver premuto RETURN, il disco inizia a frullare ed il computer risponde con:

```
SAVING "PROGRAM  NAME"
OK
READY
■
```

STAMPA E CALCOLI

Dopo aver imparato un paio delle operazioni più difficili necessarie per conservare i programmi di interesse, è possibile creare qualche programma da salvare e usare successivamente.

Cercare di battere quanto segue, esattamente come scritto:

Se si fa un errore di battitura, usare il tasto INST/DEL per cancellare il carattere immediatamente alla sinistra del cursore. E' possibile cancellare quanti caratteri è necessario.

Vediamo ora cosa è successo nell'esempio. Innanzitutto si è istruito (comandato) il computer a PRINT (battere ossia scrivere) qualsiasi cosa si trovava all'interno delle virgolette. Battendo RETURN si è detto al computer di fare ciò che si era indicato e sullo schermo è comparsa la scritta COMMODORE 64.

Quando si usa l'istruzione PRINT in questa forma, tutto ciò che è racchiuso tra virgolette viene stampato ossia scritto esattamente come lo si è battuto.

Se il computer risponde con:

?SYNTAX ERROR

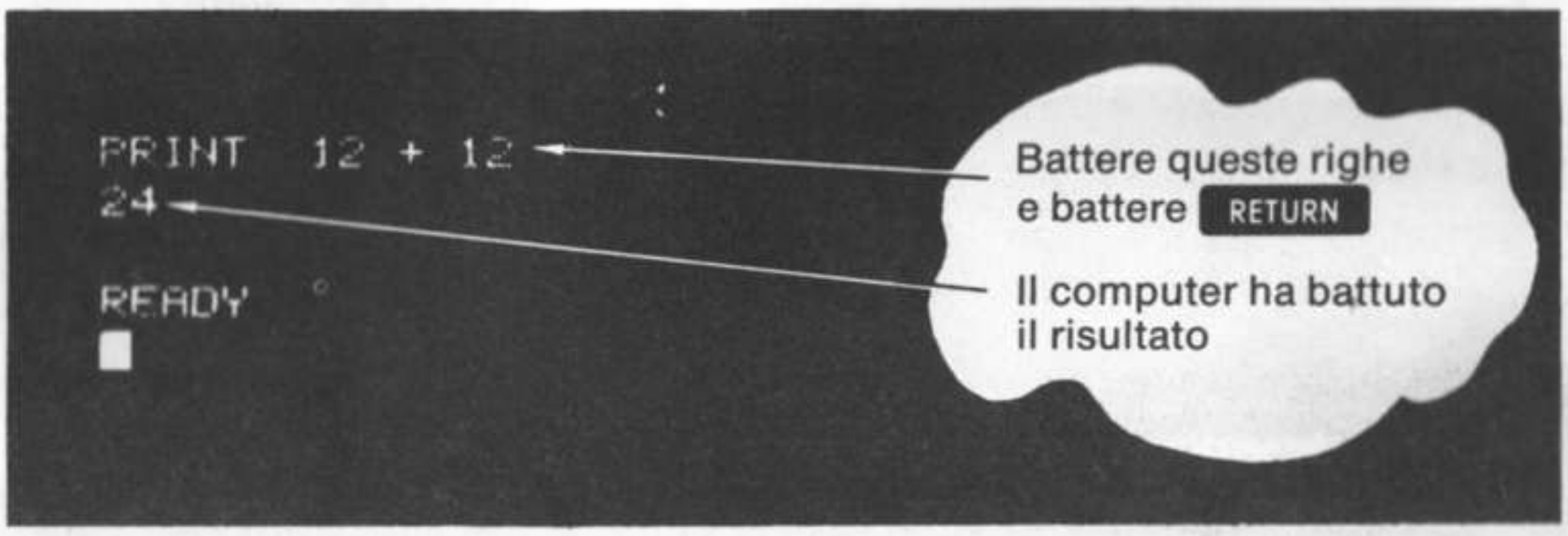
chiedersi se non si è per caso fatto un errore di battitura o si sono dimenti-

cate le virgolette. Il computer è preciso e si aspetta che le istruzioni vengano date in una forma specifica altrettanto precisa.

Ma non è il caso di preoccuparsi: basta ricordarsi di immettere le cose come vengono presentate negli esempi. Ricordarsi che non è possibile danneggiare il computer qualunque cosa si batta su di esso e che il modo migliore per imparare il BASIC è di provare varie e vedere cosa succede.

PRINT è uno dei comandi più utili e più potenti nel linguaggio BASIC. Con esso è possibile visualizzare pressochè qualsiasi cosa si desideri, compresi i grafici ed i risultati dei calcoli.

Per esempio, provare quanto segue. Cancellare lo schermo tenendo abbassato il tasto SHIFT ed il tasto CLR/HOME e battere (assicurarsi di usare il tasto «1» e non la lettera «1»):



Si è così scoperto che il COMMODORE 64, nella sua forma base, è un calcolatore. Il risultato «24» è stato calcolato e stampato automaticamente. In effetti è inoltre possibile eseguire sottrazioni, moltiplicazioni, divisioni, elevamento ad esponente e funzioni matematiche avanzate tipo calcolo di radici quadrate, ecc. E non si è limitati ad un singolo calcolo su una riga, ma di questo parlerà più avanti.

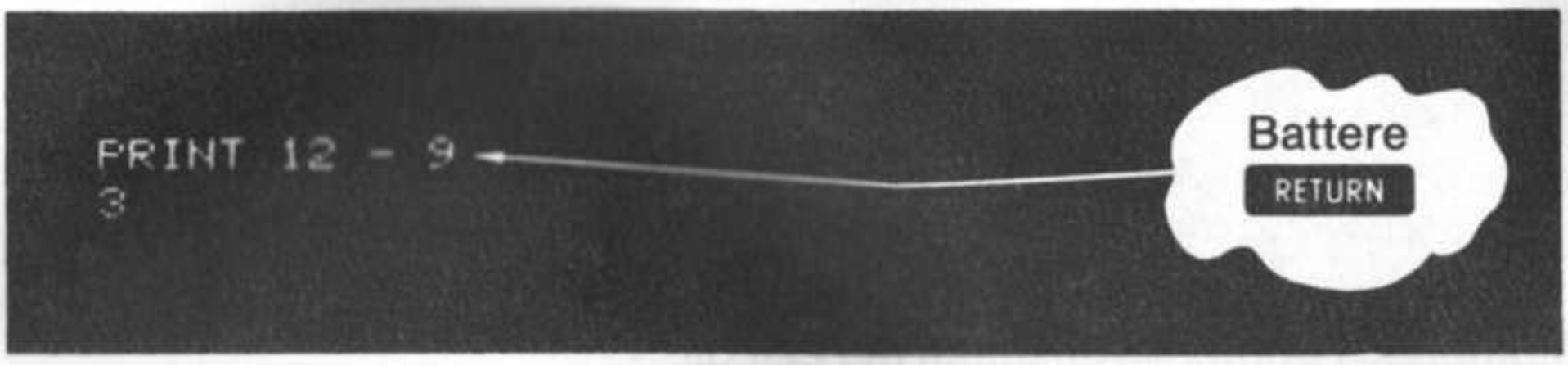
Notare che nella forma suddetta, PRINT si è comportato in maniera diversa dal primo esempio. In questo caso, viene battuto un valore o un risultato di un calcolo anzichè il messaggio esatto che era stato immesso, in quanto sono state omesse le virgolette.

Somma

Il segno più (+) indica l'addizione: si istruisce cioè il computer a stampare il risultato di 12 sommato a 12. Altre operazioni aritmetiche assumono una forma simile all'addizione. Ricordarsi di battere sempre RETURN dopo aver battuto PRINT ed il calcolo da eseguire.

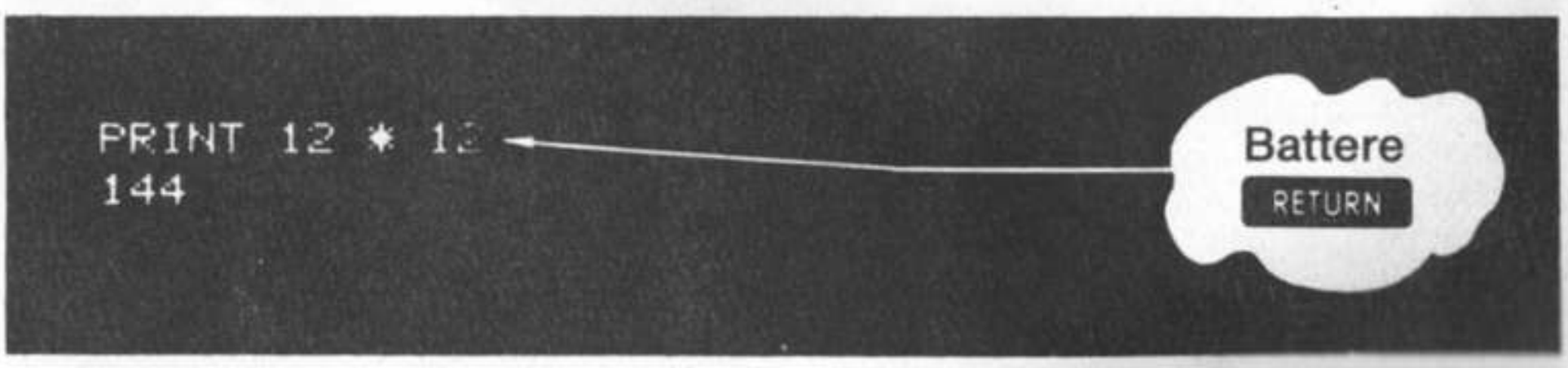
Sottrazione

Per sottrarre, usare il segno meno (−) convenzionale. Battere:



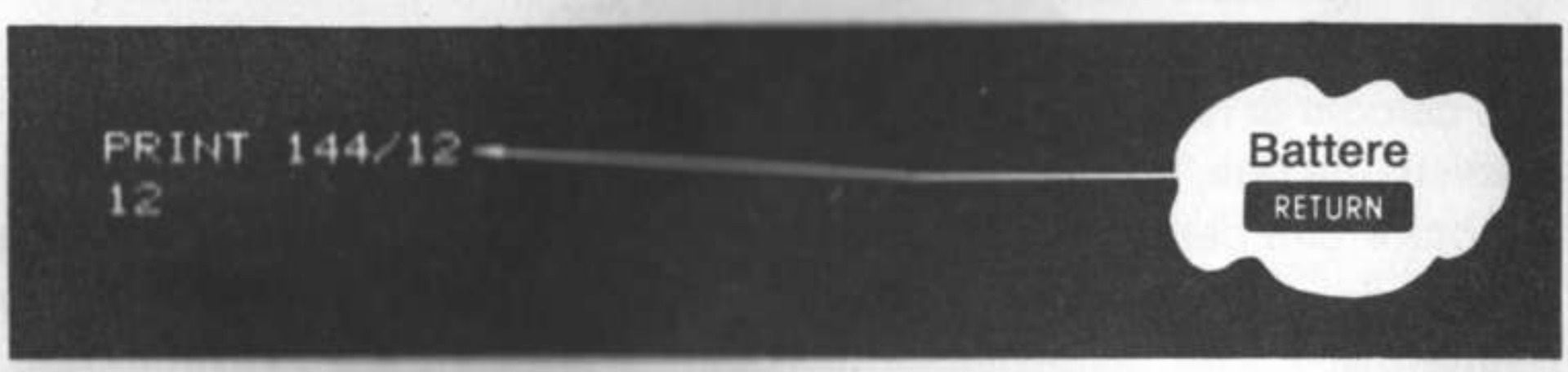
Moltiplicazione

Il segno di moltiplicazione è l'asterisco (*). Se si vuole moltiplicare 12 volte 12, si batterà:



Divisione

La divisione è indicata dalla familiare barretta «/». Ad esempio per dividere 144 per 12, battere:



Elevamento ad esponente

In maniera analoge è possibile facilmente elevare un numero ad una potenza (il che equivale a moltiplicare un numero per se stesso per un numero specificato di volte). La freccia verso l'alto «↑» significa elevamento ad esponente.

```
PRINT 12 ↑ 5
248832
```

Ciò equivale a battere:

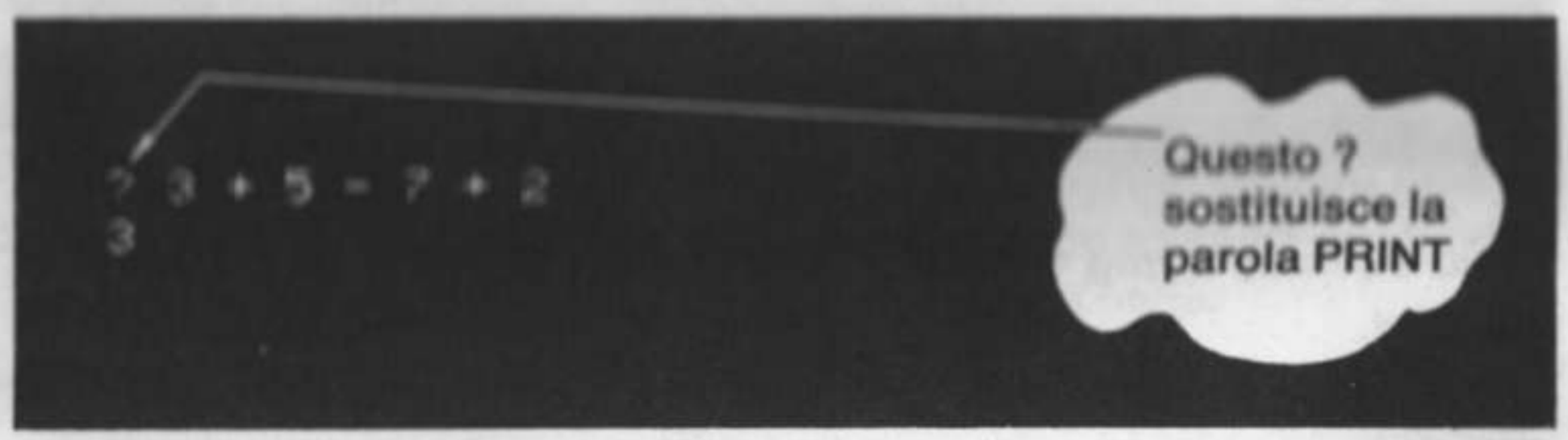
```
PRINT 12 * 12 * 12 * 12 * 12
248832
```

SUGGERIMENTO:

Il BASIC dispone di un certo numero di scorciatoie per fare determinate cose. Una di esse è l'abbreviazione di comandi (o parole chiave) BASIC. Ad esempio, può essere usato un ? in luogo di PRINT. Man mano si procede verranno presentati molti comandi; L'Appendice D mostra l'abbreviazione per ciascuno di essi e ciò che compare sullo schermo quando si batte la forma abbreviata.

L'ultimo esempio richiama un punto molto importante: è possibile eseguire molti calcoli sulla stessa riga e questi calcoli possono essere di tipo misto.

E' possibile calcolare questo problema:



Fino a questo punto si sono usati numeri piccoli ed esempi semplici, ma COMMODORE 64 è in grado di eseguire calcoli più complessi.

Si potrebbero ad esempio sommare numerose grandi cifre. Provare con questo esempio, senza però usare le virgole, altrimenti si ottiene un errore:

```
? 123.45 + 345.78 + 7895.687
8364.917
```

Provare ora questo:

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Se si prende il gusto di sommare manualmente, si trova un diverso risultato.

Cosa è successo? Quantunque il computer abbia molta potenza, ha tuttavia un limite ai numeri che può gestire. Il COMMODORE 64 può lavorare con numeri che contengono 10 cifre, ma quando un numero viene stampato, vengono visualizzate soltanto 9 cifre.

Così nell'esempio, il risultato è stato «arrotondato» per inserirsi nel campo appropriato. COMMODORE 64 arrotonda per eccesso quando la successiva cifra è cinque o più, arrotonda per difetto quando la successiva cifra è quattro o meno.

I numeri compresi fra 0.01 e 999,999,999 sono stampati usando la notazione standard. I numeri al di fuori di questo campo vengono stampati usando la notazione scientifica.

La notazione scientifica non è che un altro modo per esprimere un numero molto grande o molto piccolo sotto forma di potenza di 10.

Se si batte:

```
? 123000000000000000000
1.23E+17
```

Ciò equivale a dire 1.23 • 10↑17 e si usa per rendere le cose molto semplici.

C'è però un limite ai numeri che il computer può gestire anche nella notazione scientifica. Questi limiti sono:

Numero più grande: ± 1.70141183E+38
Numero più piccolo: ± 2.93873588E−39

PRECEDENZA

Se si cercasse di eseguire qualche calcolo misto diverso dagli esempi mostrati precedentemente, si potrebbero non trovare i risultati previsti. Il motivo è che il computer esegue i calcoli in un certo ordine.
In questo calcolo:

20 + 8/2

non è possibile dire se la risposta deve essere 24 o 14 fino a che non si sa in quale ordine si eseguono i calcoli. Se si somma 20 a 8 diviso per 2 (ossia 4), il risultato è 24. Ma se si somma 20 a 8 e si divide quindi per 2 il risultato è 14. Provare con l'esempio e vedere cosa si ottiene come risultato.

Il motivo per aver ottenuto 24 è che il COMMODORE 64 esegue i calcoli da sinistra a destra secondo quanto segue:

Primo:	–	segno meno che indica i numeri negativi
Secondo:	↑	elevamento ad esponente da sinistra a destra
Terzo:	*/	moltiplicazione e divisione, da sinistra a destra
Quarto:	+–	addizione e sottrazione, da sinistra a destra

Seguendo l'ordine di precedenza indicato nel suddetto esempio la divisione viene eseguita per prima e quindi viene eseguita l'addizione per dare un risultato di 24.

E' bene a questo punto esercitarsi con alcuni problemi propri e prevedere i risultati secondo le regole sopra indicate.

C'è anche un modo facile per variare la precedenza, usando cioè le parentesi per separare le operazioni che si vogliono eseguire per prime.

Per esempio, se si vuole dividere 35 per 5 più 2 si batte:

```
? 35 / 5 + 2
 9
```

e si ottiene 35 diviso 5 con 2 aggiunto al risultato, il che non è ciò che si intendeva. Provare invece in questo modo:

```
? 35 / (5 + 2)
 5
```

E' successo che il computer valuta per primo ciò che è contenuto nelle parentesi. Se ci sono parentesi all'interno di altre parentesi, vengono calcolate per prime le parentesi più interne.

Dove ci sono numerose parentesi su una riga, ad esempio:

```
? (12 + 9) * (6 + 1)
147
```

il computer le valuta da sinistra a destra. Qui 21 verrebbe moltiplicato per 7 per avere il risultato di 147.

COME COMBINARE LE COSE

Quantunque sia stato dedicato un po' di tempo ad argomenti che potrebbero sembrare non importanti, i particolari qui presentati avranno tuttavia più senso una volta iniziata la programmazione e si dimostreranno preziosi.

Per dare un'idea di come le cose vadano al loro posto, si consideri quanto segue: come è possibile combinare tutti e due i tipi di istruzioni di stampa finora esaminate per stampare qualcosa che abbia più significato sullo schermo?

Si sa che racchiudendo qualcosa tra le virgolette si stampano quelle informazioni sullo schermo esattamente come sono state immesse e che usando gli operatori matematici è possibile eseguire calcoli. Così perchè non combinare i due tipi di istruzione PRINT come segue:

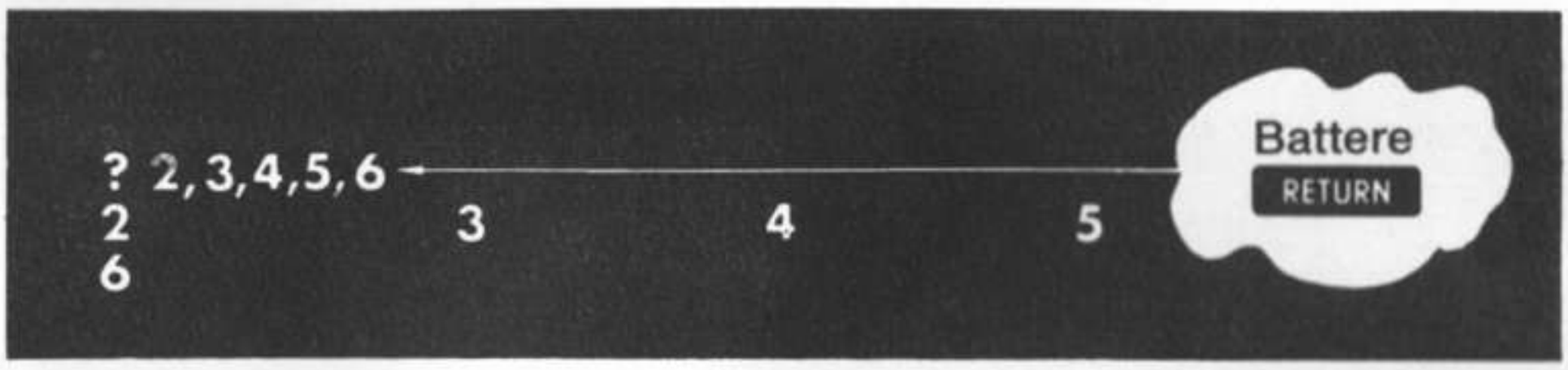
Il punto e virgola significa assenza di spazio

```
? "5 * 9 = "; 5 * 9
5 * 9 = 45
```

Quantunque ciò possa sembrare in un certo modo ridondante, non si è fatto altro che usare insieme entrambi i tipi di istruzione PRINT. La prima parte stampa «5 * 9 =» esattamente come è stato battuto. La seconda parte esegue il lavoro effettivo e stampa il risultato, con il punto e virgola che separa la parte messaggio dell'istruzione dal calcolo effettivo.

Occorre sempre separare le parti di un'istruzione di stampa mista con qualche segno di punteggiatura perchè le cose funzionino correttamente. Provare con una virgola in luogo del punto e virgola e vedere cosa succede.

Per il curioso, il punto e virgola può far si che la successiva parte dell'istruzione venga stampata immediatamente dopo la parte precedente, senza spazi. La virgola fa qualcosa di diverso. Quantunque sia un separatore accettabile, distanzia le cose ulteriormente. Se si batte:



i numeri vengono stampati occupando tutto lo schermo ed anche la riga successiva.

Lo schermo del COMMODORE 64 è suddiviso in quattro aree di 10 colonne ciascuna. La virgola esegue la tabulazione di ciascun risultato nella successiva area disponibile. Dato che è stato chiesto di stampare più informazioni di quante ne possa accogliere la riga, (si è cioè tentato di inserire cinque aree di 10 colonne su una riga), l'ultima voce è stata spostata alla riga successiva.

La differenza base tra la virgola ed il punto e virgola nella formattazione delle istruzioni PRINT può essere usata vantaggiosamente creando schermi più complessi: essa consente di creare molto facilmente alcuni risultati sofisticati.

CAPITOLO 3

INIZIO DELLA PROGRAMMAZIONE IN BASIC

- La fase successiva GOTO
- Suggerimenti per la correzione
- Variabili
- IF . . . THEN
- Le iterazioni FOR . . . NEXT

LA FASE SUCCESSIVA

Finora sono state eseguite semplici operazioni immettendo nel computer una singola riga di istruzioni. Una volta abbassato il tasto RETURN, l'operazione specificata viene eseguita immediatamente. Questo modo di procedere è detto modo CALCOLATORE o IMMEDIATO.

Ma per effettuare qualche cosa di importante, occorre far sì che il computer operi con più di un'istruzione su una sola riga. Un certo numero di istruzioni combinate fra di loro è detto PROGRAMMA e consente di usare la piena potenza del COMMODORE 64.

Per rendersi come sia facile scrivere il primo programma COMMODORE 64, provare come segue:

Cancellare lo schermo tenendo abbassato il tasto SHIFT e quindi abbassando il tasto CLR/HOME.

Battere NEW e premere RETURN. (Ciò cancella qualsiasi numero che potrebbe essere rimasto nel computer dalle precedenti sperimentazioni).

Battere ora quanto segue esattamente come indicato (ricordarsi di battere RETURN dopo ciascuna riga).

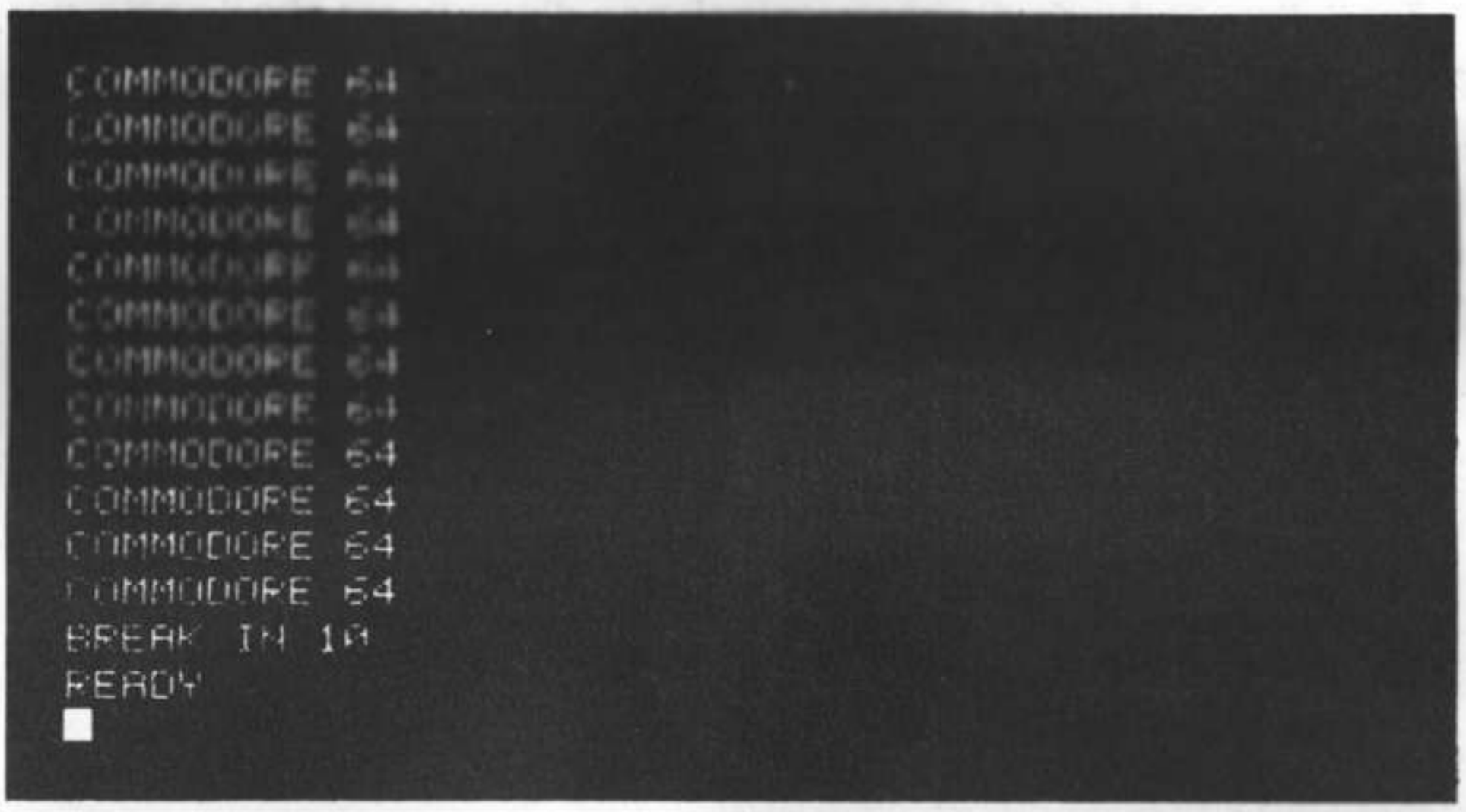
```
10 ?"COMMODORE 64"
20 GOTO 10
```

Battere ora RUN e premere ancora RETURN – ed osservare ciò che succede. Lo schermo si anima facendo comparire la scritta COMMODORE 64. Dopo aver osservato lo schermo, premere RUN/STOP per interrompere il programma.

In questo breve programma sono stati introdotti numerosi ed importanti concetti che formano la base di tutta la programmazione.

Notare che si è fatta precedere ciascuna istruzione da un numero, detto numero di RIGA, numero che dice al computer in quale ordine lavorare per ciascuna istruzione. Questi numeri rappresentano anche un punto di riferimento in caso in cui un programma debba tornare indietro ad una particolare riga. I numeri di riga possono essere rappresentati da qualsiasi valore intero compreso tra 0 e 63.999.

```
10 PRINT "COMMODORE 64"
↑      ↑
└──────┴── ISTRUZIONE
└─────────── NUMERO DI LINEA
```



E' buona prassi di programmazione numerare le righe ad incrementi di 10 nel caso in occorresse inserire successivamente altre istruzioni.

Oltre a PRINT, il programma ha usato anche un altro comando BASIC, e cioè GOTO. Questo comando istruisce il computer a portarsi direttamente su una particolare riga, ad eseguirla e quindi a continuare da quel punto.

```
10 PRINT "COMMODORE 64"
20 GOTO 10
```

Nell'esempio, il programma stampa il messaggio nella riga 10, va alla riga successiva (20), che dà istruzioni di ritornare alla riga 10 e stampa il messaggio di nuovo. Quindi il ciclo si ripete. Dato che non si è dato al computer un modo per uscire da questo circolo vizioso, il programma continuerà all'infinito o fino a che non lo si interrompe fisicamente con il tasto RUN/STOP.

Una volta interrotto il programma battere: LIST. Il programma sarà visualizzato intatto, in quanto è ancora nella memoria del computer. Notare anche che il computer ha convertito automaticamente il punto di domanda in PRINT. Il programma può essere ora cambiato, salvato o eseguito di nuovo.

Un'altra importante differenza tra battere qualche cosa nel modo immediato e scrivere un programma è che una volta eseguita l'istruzione e cancellato lo schermo, l'istruzione immediata va persa. Per contro è sempre possibile far ricomparire un programma semplicemente battendo LIST.

Fra l'altro, parlando di abbreviazioni, non dimenticare che il computer può esaurire lo spazio su una riga se se ne usano troppe.

SUGGERIMENTI PER LE CORREZIONI

Se si compie un errore sulla riga, ci sono numerose possibilità di correzione.

1. E' possibile ribattere una riga in qualsiasi momento: il computer la sostituirà automaticamente alla vecchia.
2. Una riga indesiderata può essere cancellata semplicemente battendo il numero di riga e RETURN.
3. E' possibile inoltre correggere facilmente una riga esistente usando i tasti del cursore ed i tasti di editing.

Si supponga di aver compiuto un errore di battitura in una riga dell' esempio. Per correggerlo senza ribattere l'intera riga procedere come segue:

Battere LIST, quindi usando insieme i tasti SHIFT e ↑↓ CRSR spostare il cursore verso l'alto fino a che non è posizionato sulla riga che deve essere modificata.

Ora usare il tasto del cursore con freccia verso destra per spostare il cursore sul carattere che si vuole modificare, ribattendo la modifica sul vecchio carattere. Premere infine RETURN per far sì che la riga corretta sostituisca la vecchia.

Se occorre più spazio sulla riga, posizionare il cursore nel punto in cui occorre spazio e premere contemporaneamente i tasti SHIFT e INST/DEL. Con questa manovra si apre uno spazio. Battere ora le ulteriori informazioni e premere RETURN. Analogamente è possibile cancellare caratteri indesiderati disponendo il cursore alla destra del carattere indesiderato e premendo il tasto INST/DEL.

Per verificare che le modifiche siano state immesse, battere di nuovo LIST per far ricomparire il programma corretto. Le righe non devono essere necessariamente immesse in ordine numerico: pensa il computer ad inserirle automaticamente nell'appropriata sequenza.

Provare a corregare il programma esemplificativo cambiando la riga 10 ed aggiungendo un punto e virgola alla fine della riga come indicato a pagina 35. Quindi eseguire (RUN) di nuovo il programma.

```
10 PRINT "COMMODORE";
```

Non dimenticare di spostare il cursore oltre la riga 20 prima di eseguire il programma

VARIABILI

Le variabili sono alcune delle caratteristiche più usate di qualsiasi linguaggio di programmazione in quanto possono rappresentare nel computer un numero molto maggiore di informazioni. La conoscenza del modo di funzionamento delle variabili rende il calcolo più facile e consente di eseguire operazioni non altrimenti possibili.

```
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
COMMODORE   COMMODORE   COMMODORE   COMMODORE
BREAK IN 10
READY
```

Si immagini di avere un certo numero di scatole all'interno del computer, ciascuna delle quali contiene un numero o una stringa di caratteri di testo. Ciascuna di queste scatole deve essere contrassegnata con un nome a scelta. Questo nome è detto variabile e rappresenta le informazioni contenute nella rispettiva scatola.

Per esempio dicendo:

```
10 X% = 15
20 X = 23.5
30 X$ = "LA SOMMA DI X%+X = "
```

Il computer può rappresentare le variabili in questo modo:

X%	**15**
X	**23.5**
X$	**LA SOMMA DI X%+X =**

Una nuova variabile rappresenta la scatola o la locazione di memoria in cui viene caricato il valore corrente della variabile. Come è possibile

vedere, ad una variabile si può assegnare un numero intero o un numero in virgola mobile o una stringa di testo.

Il simbolo % che segue un nome variabile indica che la variabile rappresenterà un numero intero. Quelli che seguono sono nomi di variabili valide intere:

A%
X%
A1%
NM%

Il «$» che segue il nome variabile indica che la variabile rappresenta una stringa di testo. Quelli che seguono sono esempi di variabili stringa:

A$
X$
MI$

Le variabili in virgola mobile seguono lo stesso formato, con l'indicatore di tipo:

A1
X
Y
MI

Nell'assegnare un nome ad una variabile ci sono alcune cose da tener presenti. Innanzitutto una variabile può avere uno o due caratteri. Il primo carattere deve essere un carattere alfabetico da A a Z; il secondo carattere può essere un carattere alfabetico o numerico (nel campo da 0 a 9). Può essere incluso un terzo carattere per indicare il tipo di variabile (intera o stringa di testo), % o $.

E' possibile usare nomi variabili con più di due caratteri alfabetici ma solo i primi due vengono riconosciuti dal computer. Così PA e PARTNO sono identici e fanno riferimento alla stessa variabile.

L'ultima regola per i nomi variabili è semplice: essi non possono contenere qualsiasi parola chiave BASIC (parole riservate) tipo GOTO, RUN, ecc. Fare riferimento all'Appendice D per un elenco completo delle parole riservate BASIC.

Per vedere come si possono utilizzare le variabili, battere il programma completo introdotto precedentemente ed eseguirlo (RUN) Ricordarsi di premere RETURN dopo ciascuna riga del programma.

```
NEW
10 X% = 15
20 X = 23.5
30 X$ = "THE SUM OF X% + X = "
40 PRINT "X% = "; X%, "X = "; X
50 PRINT X$; X% + X
```

Se tutto è stato eseguito come indicato, si dovrebbe ottenere sullo schermo il seguente risultato.

```
RUN
X% = 15       X = 23.5
THE SUM OF X% + X = 38.5
READY
```

Sono stati riuniti tutti i trucchi finora imparati per formattare uno schermo così come lo si vede e per stampare la somma delle due variabili.

Nelle righe 10 e 20 è stato assegnato un valore intero a X% ed assegnato un valore in virgola mobile a X. Ciò inserisce il numero associato con la variabile nella rispettiva «scatola». Nella riga 30 è stata assegnata una stringa di testo a X$. La riga 40 riunisce i due tipi di istruzione PRINT per battere un messaggio ed il valore effettivo di X% e di X. La riga 50 stampa la stringa di testo assegnata a X$ e la somma di X% e X.

Notare che quantunque venga usato X come parte di ciascun nome variabile, gli identificatori % e $ rendono X%, X e X$ unici, e cioè fanno in modo che essi rappresentino tre variabili distinte.

Ma le variabili sono molte più potenti. Se se ne cambia il valore, il nuovo valore sostituisce quello originale nella stessa «scatola». Ciò consente di scrivere un'istruzione del tipo:

$$X = X + 1$$

L'istruzione di questo genere non verrebbe mai accettata nell'algebra normale ma rappresenta per contro uno dei concetti più usati nella programmazione e significa «prendere il valore corrente di X, aggiungere uno ed iserire la nuova somma nella «scatola» che rappresenta X».

IF . . . THEN

Armati della capacità di aggiornare facilmente il valore delle variabili, è possibile ora provare un programma di questo tipo:

```
NEW
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Non si è fatto altro che introdurre due nuovi comandi BASIC e disporre qualche controllo sul breve programma di stampa introdotto all'inizio del capitolo.

IF . . . THEN aggiunge una certa logica al programma. Esso dice che se (IF) una condizione è vera, allora (THEN) occorre fare qualche cosa. Se (IF) la condizione non è più vera, allora (THEN) occorre passare alla successiva riga nel programma.

E' possibile fissare un certo numero di condizioni usando l'istruzione IF . . . THEN:

SYMBOLO	*SIGNIFICATO*
<	Minore di
>	Maggiore di
=	Uguale a
<>	Diverso da
>=	Maggiore di o uguale a
<=	Minore di o uguale a

L'uso di una qualsiasi di queste condizioni è facile ma sorprendentemente efficace.

```
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
```

Nel programma campione è stato creato un «loop» o «iterazione» sul quale sono stati posti alcuni vincoli dicendo: Se (IF) un valore è minore di un certo numero allora (THEN) occorre fare qualche cosa.

La riga 10 definisce CT (CounT-Conteggio) uguale a 0. La riga 20 stampa il messaggio. La riga 30 aggiunge uno alla variabile CT. Questa riga conta quante volte occorre eseguire il loop o iterazione. Ogni volta che l'iterazione viene eseguita, CT aumenta di uno.

La riga 40 è la riga di controllo. Se CT è meno di 5, indicando che è stata eseguita l'iterazione cinque volte, il programma ritorna alla riga 20 e stampa di nuovo. Quando CT è uguale a 5 – indicando che è stato stampato cinque volte COMMODORE 64 – il programma va alla riga 50, che segnala la fine (END) del programma.

Provare praticamente il programma. Cambiando il limite CT nella riga 40 si può far stampare qualsiasi numero di righe.

IF . . . THEN ha molti altri usi, che verranno esaminati negli esempi successivi.

ITERAZIONI FOR . . . NEXT

C'è un modo più semplice e preferibile per eseguire ciò che è stato fatto nell'esempio precedente usando un'iterazione FOR . . . NEXT. Si consideri quanto segue:

```
NEW

10 FOR CT = 1 TO 5
20 PRINT "COMMODORE 64"
30 NEXT CT

RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Come si può vedere, il programma se è ridotto notevolmente ed è più diretto.

CT inizia in 1 alla riga 10. Quindi la riga 20 esegue la stampa. Alla riga 30 CT è incrementato di 1. L'istruzione NEXT nella riga 30 rimanda automaticamente il programma alla riga 10 dove è 10 disposta la parte FOR dell'istruzione FOR . . . NEXT. Questo processo continua fino a che CT non raggiunge il limite impostato.

La variabile usata in un'iterazione FOR . . . NEXT può essere incrementata di quantitativi più piccoli di 1, se necessario.

Provare con questo esempio:

```
NEW

10 FOR NB = 1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB

RUN
1             1.5           2             2.5
3             3.5           4             4.5
5             5.5           6             6.5
7             7.5           8             8.5
9             9.5           10
```

Se si immette e si esegue questo programma, si vedono comparire sullo schermo i numeri da 1 a 10 incrementi di 0,5.

Non si è fatto altro che stampare i valori che NB assume man mano che procede con le iterazioni.

E' possibile anche specificare se la variabile debba aumentare o diminuire. Si sostituisca la seguente alla riga 10:

10 FOR NB = 10 to 1 STEP – .5

ed osservare che si verifica il contrario, dato che NB va da 10 a 1 in ordine discendente.

CAPITOLO 4

BASIC AVANZATO

- **Introduzione**
- **Semplice esempio di animazione**
  — Iterazione nidificata
- **INPUT**
- **GET**
- **Numeri casuali ed altre funzioni**
- **Gioco degli indovinelli**
- **Il lancio di dadi**
- **Grafici casuali**
  – Funzioni CHR$ e ASC

INTRODUZIONE

I successivi capitoli sono stati scritti per coloro che hanno già una certa familiarità con il linguaggio di programmazione BASIC e con i concetti necessari per scrivere i programmi più avanzati.

Coloro invece che affrontano ora per la prima volta la programmazione, potranno trovare alcune informazioni leggermente troppo tecniche per comprendere completamente. Ma coraggio . . . poichè per i successivi Capitoli 6 e 7 (rispettivamente **GRAFICI ANIMATI** e **CREAZIONE DEL SUONO**) sono stati creati sette esempi scritti appositamente per i principianti. Gli esempi daranno una buona idea del modo in cui usare le sofisticate capacità sonore e grafiche disponibili sul COMMODORE 64.

Se si vuole saperne di più sulla compilazione di programmi in BASIC, consultare l'Appendice N che contiene una bibliografica adatta allo scopo.

Per chi ha già familiarità con la programmazione BASIC, questi capitoli aiuteranno ad affrontare le tecniche di programmazione avanzata BASIC. Informazioni più dettagliate si possono trovare nel **COMMODORE 64 PROGRAMMER'S REFERENCE MANUAL**, disponibile presso il Rivenditore COMMODORE.

SEMPLICI ESEMPIO DI ANIMAZIONE

E' possibile provare qualcuna delle capacità grafiche di COMMODORE 64 riunendo ciò che è stato visto finora unitamente a qualche nuovo concetto. Chi è ambizioso può battere il seguente programma e vedere cosa succede. Si noterà che all'interno dell'istruzione di stampa è possibile includere comandi del cursore e comandi dello schermo. Quando in un listato di programma si vede comparire qualcosa del tipo (CRSR LEFT), tenere abbassato il tasto SHIFT e battere il tasto che sposta il cursore verso destra o verso sinistra. Lo schermo mostrerà la rappresentazione grafica di un cursore con freccia verso sinistra (due barre verticali invertite). Allo stesso modo, premendo SHIFT e CLR/HOME compare un cuore in negativo.

```
NEW

10  REM BOUNCING BALL
20  PRINT "{CLR/HOME}"
25  FOR X = 1 TO 10 : PRINT "{CRSR/DOWN}": NEXT
30  FOR BL = 1 TO 40
40  PRINT"  ●{CRSR LEFT}";:REM <● is a SHIFT-Q>
50  FOR TM = 1 TO 5
60  NEXT TM
70  NEXT BL
75  REM MOVE BALL RIGHT TO LEFT
80  FOR BL = 40 TO 1 STEP -1
90  PRINT"  {CRSR LEFT}{CRSR LEFT}●{CRSR LEFT} ";
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

: Indica un nuovo comando

Questi spazi sono intenzionali

SUGGERIMENTO:

Tutte le parole in queste testo saranno completate su una riga. In ogni caso fino a che non si preme RETURN il COMMODORE 64 si sposterà automaticamente alla riga successiva anche nel mezzo di una parola.

Il programma presenterà una pallina rimbalzante che si muove sullo schermo da sinistra verso destra e da destra verso sinistra.

Se si osserva il programma da vicino (indicato a pagina 44) è possibile vedere come questa azione viene ottenuta.

La riga 10 è un REMark (nota) che spiega ciò che fa il programma; essa non ha però alcun effetto sul programma stesso.

```
 10  REM BOUNCING BALL
 20  PRINT "{CLR/HOME}"
 25  FOR X = 1 TO 10 : PRINT "{CRSR/DOWN}": NEXT
 30  FOR BL = 1 TO 40
 40  PRINT" ●{CRSR LEFT}";:REM (● is a SHIFT-Q)
 50  FOR TM = 1 TO 5
 60  NEXT TM
 70  NEXT BL
 75  REM MOVE BALL RIGHT TO LEFT
 80  FOR BL = 40 TO 1 STEP -1
 90  PRINT" {CRSR LEFT}{CRSR LEFT}●{CRSR LEFT}";
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

La riga 20 cancella qualsiasi informazione dallo schermo.

La riga 25 PRINT (stampa) 10 comandi di movimento verso il basso del cursore. Ciò non fa che posizionare la pallina al centro dello schermo. Se la riga 25 fosse eliminata, la pallina si sposterebbe alla riga superiore dello schermo.

La riga 30 crea un'iterazione per spostare la pallina di 40 colonne da sinistra verso destra.

La riga 40 fa un mucchio di cose. Per prima cosa stampa uno spazio per cancellare le precedenti posizioni della pallina quindi stampa la pallina ed infine esegue un movimento verso sinistra del cursore per predisporre tutto cancellare di nuovo la posizione corrente della pallina.

L'iterazione creata nelle righe 50 e 60 rallenta leggermente la pallina ossia ritarda il programma. Senza di essa, la pallina si muoverebbe troppo velocemente per poterla vedere.

La riga 70 completa l'iterazione che stampa le palline sullo schermo, impostata nella riga 30. Ogni volta che l'iterazione viene eseguita, la pallina si muove di un altro spazio verso destra. Come notato dall'illustrazione, è stata inserita un'iterazione nell'ambito di un'altra.

Ciò è perfettamente accettabile. Si hanno dei problemi soltanto quando le iterazioni si incrociano una con l'altra. Nel compilare programmi è utile controllare come illustrato in questo caso, per assicurarsi che la logica di un'iterazione sia corretta.

Per vedere ciò che succederebbe incrociando un'iterazione, basta invertire le istruzioni delle righe 60 e 70. Si ottiene in tal caso un errore in quanto il computer si confonde e non può rendersi conto di quanto sta succedendo.

Le righe da 80 a 120 non fanno altro che invertire le fasi nella prima parte del programma e spostare la pallina da sinistra a destra verso

sinistra. La riga 90 è leggermente diversa dalla riga 40 in quanto la pallina si muove in direzione opposta (occorre cancellare la pallina verso destra e spostarla verso sinistra).

E quando ciò è stato fatto il programma ritorna alla riga 20 per iniziare da capo l'intero processo. Interessante, non è vero?

Per una variazione sul programma corregare la riga 40 come segue:

```
40 PRINT "●";
```

← Per creare lo 0, tenere abbassato il tasto SHIFT e battere la lettera «Q»

Eseguire il programma e vedere cosa succede ora. Poichè è stato escluso il controllo del cursore, ciascuna pallina rimane sullo schermo fino a che non viene cancellata dal movimente della pallina da destra verso sinistra nella seconda parte del programma.

INPUT

Finora, tutto ciò che compariva in un programma era stato impostato prima della sua esecuzione. Una volta iniziato il programma nulla poteva essere cambiato. INPUT consente invece di trasmettere nuove informazioni ad un programma mentre è in esecuzione e farlo operare sulle nuove informazioni.

Per avere un'idea di come funziona INPUT, battere NEW RETURN ed immettere questo breve programma:

```
10 INPUT A$
20 PRINT "YOU TYPED: ";A$
30 PRINT
40 GOTO 10
RUN
? COMMODORE 64
YOU TYPED: COMMODORE 64
```

Ciò che e' stato battuto

Il computer ha così risposto

Ciò che succede quando si esegue questo programma è molto semplice. Compare un punto di domanda per indicare che il computer è in attessa che si batta qualche cosa. Immettere qualsiasi carattere o gruppo di caratteri da tastiera e premere RETURN. Il computer risponde con «YOU TYPED:» seguito dalle informazioni immesse.

Ciò può sembrare molto elementare ma si immagini cosa è possibile far fare al computer con le informazioni immesse.
E' possibile INPUT (immettere) variabili numeriche o stringhe ed addirittura fare in modo che l'istruzione INPUT informi l'utente con un messaggio. Il formato di INPUT è:

INPUT "MESSAGGIO RICHIESTA";Variabile

La richiesta deve contenere meno di 40 caratteri

Oppure semplicemente:

INPUT VARIABILE

NOTA: Per uscire da questo programma tenere abbassati i tasti RUN/STOP e RESTORE.

Il programma seguente non è soltanto utile ma dimostra molto di ciò che è stato finora presentato, compresa la nuova istruzione di input.

```
NEW

1 REM TEMPERATURE CONVERSION PROGRAM
5 PRINT "{CLR/HOME}"
10 PRINT "CONVERT FROM FAHRENHEIT OR CELSIUS
   (F/C)": INPUT A$
20 IF A$ = "" THEN 10
30 IF A$ = "F" THEN 100
40 IF A$ = "C" THEN 50
50 INPUT "ENTER DEGREES CELSIUS: ";C
60 F = (C*9)/5+32
70 PRINT C;" DEG. CELSIUS = "; F;" DEG.
   FAHRENHEIT"
80 PRINT
90 GOTO 10
100 INPUT "ENTER DEGREES FAHRENHEIT: ";F
110 C = (F-32)*5/9
120 PRINT F;" DEG. FAHRENHEIT = ";C;" DEG.
    CELSIUS"
130 PRINT
140 GOTO 10
```

Qui non c'è spazio

Non dimenticare di premere return

Se si immette e si esegue questo programma, si vede INPUT in azione.

La riga 10 usa l'istruzione di input non solo per raccogliere informazioni ma anche per stampare la richiesta. Notare inoltre che è possibile chiedere un numero o una stringa (usando uno variabile numerica o stringa).

Le righe 20, 30 e 40 eseguono alcuni controlli su ciò che è stato battuto. Nella riga 20, se non è stato immesso nulla (è stato semplicemente premuto RETURN), il programma torna alla riga 10 e chiede di nuovo l'input.

Alla riga 30, se è stato battuto F, si sa che l'utente desidera convertire in gradi Celsius una temperatura espressa in gradi Fahrenheit cosicchè il programma salta alla parte che esegue quella conversione.

La riga 40 esegue un altro controllo. Sappiamo che ci sono soltanto due scelte valide che l'utente può immettere. Per accedere alla riga 40 l'utente deve battere qualche carattere diverso da F. Ora viene eseguito un controllo per accertarsi se quel carattere è una C; in caso contrario il programma richiede di nuovo l'input.

Tutto ciò può sembrare eccessivamente complicato ma in realtà si tratta di una buona prassi di programmazione.

Un utente che non ha familiarità con il programma può sentirsi molto frustrato se questo fa qualcosa di strano in quanto è stato compiuto un errore nell'immettere le informazioni.

Una volta determinato quale tipo di conversione eseguire, il programma esegue il calcolo e stampa la temperatura immessa e la temperatura convertita.

Il calcolo è semplicemente matematico con l'uso della formula apposita per le conversioni di temperatura. Al termine del calcolo e dopo la stampa del risultato, il programma torna all'inizio e riprende da capo.

Dopo l'esecuzione lo schermo dovrebbe apparire come quello qui di seguito riportato.

```
CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ?F
ENTER DEGREES FAHRENHEIT: 32
 32 DEG. FAHRENHEIT = 0 DEG. CELSIUS

CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ?
```

Dopo aver eseguito il programma assicurarsi di salvarlo su disco o su nastro. Questo programma nonchè gli altri presentati nel corso del manuale possono formare una buona base della propria libreria di programmi.

GET

L'istruzione GET consente di immettere un carattere alla volta da tastiera senza premere RETURN. Ciò accelera notevolmente l'immissione di dati in molte applicazioni. Qualunque sia il tasto che si preme, viene assegnato alla variabile specificata con GET.

La seguente routine illustra come funziona GET:

```
NEW

1 PRINT "{CLR/HOME}"
10 GET A$: IF A$ = "" THEN 10
20 PRINT A$;
30 GOTO 10
```

Qui non c'è spazio

Se si esegue (RUN) il programma, lo schermo si cancella ed ogni volta che si batte un tasto la riga 20 lo stampa sullo schermo e quindi accede (GET) ad un altro carattere. E' importante notare che il carattere immesso non verrà visualizzato a meno che non lo si stampi (PRINT) specificatamente sullo schermo, come è stato fatto in questo caso.

Anche la seconda istruzione sulla riga 10 è molto importante. GET funziona in continuazione anche se non viene premuto alcun tasto (a differenza di INPUT che attende una risposta), cosicchè la seconda parte di questa riga controlla in continuazione la tastiera fino a che non viene premuto un tasto.

Vedere cosa succede se la seconda parte della riga 10 viene eliminata.

Per interrompere questo programma è possibile premere i tasti RUN/STOP e RESTORE.

La prima parte del programma di conversione della temperatura dovrebbe facilmente essere riscritta per usare GET. Caricare (LOAD) il programma di conversione della temperatura e modificare le righe 10, 20 e 40 come indicato:

```
10 PRINT "CONVERT FROM FAHRENHEIT OR CELSIUS
   (F/C)"
20 GET A$: IF A$ = "" THEN 20
40 IF A$ <> "C" THEN 20
```

Questa modifica fa sì che il programma funzioni in maniera più uniforme, dato che nulla succede fino a che l'utente non batte una delle risposte desiderate per scegliere il tipo di conversione.

Una volta effettuata questa modifica è bene salvare su nastro o su disco la nuova versione del programma.

NUMERI CASUALI ED ALTRE FUNZIONI

Il COMMODORE 64 contiene numerose funzioni che vengono usate per eseguire operazioni speciali. Le funzioni possono essere viste come programmi incorporati inclusi nel BASIC. Ma anzichè battere un certo numero di istruzioni ogni volta che occorre eseguire un calcolo specializzato, basta battere il comando per la funzione desiderata: il computer farà il resto.

Molte volte nel creare un gioco o programma educativo, occorre generare un numero casuale per simulare ad esempio il lancio di dadi. E' possibile certamente scrivere un programma che generi questi numeri ma un modo più facile consiste nell'utilizzare la funzione dei numeri RaNDom (casuali).

Per vedere come funziona effettivamente RND, provare con questo breve programma:

```
NEW

10 FOR X = 1 TO 10
20 PRINT RND(1),
30 NEXT
```

SE SI TRALASCIA LA VIRGOLA, LA LISTA DI NUMERI COMPARE COME UNA FILA

Dopo aver eseguito il programma, lo schermo si presenta come indicato nella figura:

```
 .789280697        .664673958
 .256373663        .012344228?
 .682952381        3.90587279E-04
 .402343724        .879306926
 .158209863        .245596701
```

I numeri non corrispondono? Se corrispondessero sarebbe un bel problema, in quanto essi devono essere completamente casuali!

Cercare di eseguire il programma altre volte per verificare che i risultati siano sempre diversi. Anche se i numeri non seguono alcun profilo, si può però notare che alcune cose rimangono inalterate ogni volta che il programma viene eseguito.

Innanzitutto i risultati sono sempre compresi tra 0 e 1 ma mai uguali a 0 o a 1. Ciò non accadrà mai se si vuole simulare il lancio casuale di un dado dato che ci si aspetta numeri compresi tra 1 e 6.

L'altra caratteristica importante da osservare è che si tratta di numeri reali (con cifra decimale). Ciò potrebbe a sua volta rappresentare un problema dato che occorrono spesso numeri interi.

Esistono numerosi modi semplici per produrre i numeri dalla funzione RND nel campo desiderato.

Sostituire la riga 20 con la seguente ed eseguire di nuovo il programma:

```
20 PRINT 6*RND(1),

RUN


3.60563664          4.53660853
5.47238963          8.40850227
3.19265054          4.39547668
3.16331095          5.50620749
9.32527884          4.17090293
```

Ciò ha risolto il problema di ottenere risultati non superiori a 1 ma esiste sempre la parte decimale del risultato. A questo punto può essere richiamata un'altra funzione.

La funzione INTeger (intero) converte i numeri reali in valori interi.

Ancora una volta sostituire la riga 20 con la seguente ed eseguire il programma per vedere l'effetto della modifica:

```
20 PRINT INT(6*RND(1)),

RUN


2 2                 3                   1                   0
2                   4                   5                   5
0                   1
```

Il programma ha fatto molto per avvicinare all'obiettivo originale di generare numeri casuali compresi tra 1 e 6. Se si esamina da vicino ciò che è stato generato in questa occasione, si trova che i risultati sono compresi soltanto nel campo da 0 a 5.

Con l'ultima fase, aggiungere 1 all'istruzione come segue:

```
20 PRINT INT(6*RND(1))+1,
```

Ora, si è ottenuto il risultato desiderato.

In generale è possibile inserire un numero, una variabile o qualsiasi

espressione BASIC tra le parentesi della funzione INT. In relazione al campo desiderato basta moltiplicare il limite superiore per la funzione RND. Ad esempio, per generare numeri casuali tra 1 e 25 si potrebbe battere:

```
20 PRINT INT(25*RND(1))+1
```

La forma generale per creare una serie di numeri casuali in un certo campo è la seguente:

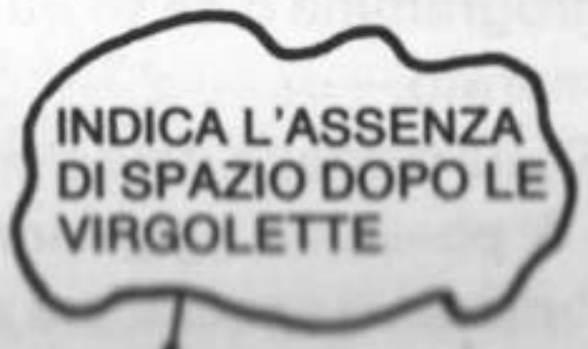
NUMBER = INT (UPPER LIMIT*RND(1)) + LOWER LIMIT

GIOCO DEGLI INDOVINELLI

Dato che è stato speso un pò di tempo per conoscere i numeri casuali perchè non utilizzare le informazioni apprese?

Il gioco che segue non solo illustra un buon uso dei numeri casuali ma introduce anche ulteriori teorie di programmazione.

Nell'eseguire questo programma verrà generato un numero casuale, NM.




```
NEW

1 REM NUMBER GUESSING GAME
2 PRINT "{CLR/HOME}"
5 INPUT "ENTER UPPER LIMIT FOR GUESS ";LI
10 NM = INT(LI*RND(1))+1
15 CN = 0
20 PRINT "I'VE GOT THE NUMBER."
30 INPUT "WHAT'S YOUR GUESS"; GU
35 CN = CN + 1
40 IF GU > NM THEN PRINT "MY NUMBER IS
   LOWER": PRINT : GOTO 30
50 IF GU < NM THEN PRINT "MY NUMBER IS
   HIGHER": PRINT : GOTO 30
60 IF GU = NM THEN PRINT "GREAT! YOU GOT MY
   NUMBER"
65 PRINT "IN ONLY "; CN ;"GUESSES.":PRINT
70 PRINT "DO YOU WANT TO TRY ANOTHER (Y/N)";
80 GET AN$: IF AN$="" THEN 80
90 IF AN$ = "Y" THEN 2
100 IF AN$ <> "N" THEN 80
110 END
```

E' possibile specificare la dimensione del numero all'inizio del programma. A questo punto occorre indovinare qual è il numero.

Segue ora un'esecuzione esemplificativa unitamente alla spiegazione.

```
ENTER UPPER LIMIT FOR GUESS ? 25
I'VE GOT THE NUMBER.

WHAT'S YOUR GUESS ? 15
MY NUMBER IS HIGHER.

WHAT'S YOUR GUESS ? 20
MY NUMBER IS LOWER.

WHAT'S YOUR GUESS ? 19
GREAT! YOU GOT MY NUMBER
IN ONLY 3 GUESSES.

DO YOU WANT TO TRY ANOTHER (Y/N) ?
```

Le istruzioni IF/THEN confrontano i tentativi di individuare il numero con il numero effettivamente generato. In relazione al numero ipotizzato il programma dice se questo era più alto o più basso del numero casuale generato.

Dalla formula data per determinare il campo di numeri casuali vedere se è possibile aggiungere alcune righe al programma che consenta all'utente di specificare anche il campo inferiore di numeri generati.

Ogniqualvolta si fa un'ipotesi, CN è incrementato di 1 per tener nota del numero dei tentativi. Usando il programma, accertarsi se è possibile usare un buon ragionamento per indovinare un numero nel minor numero di tentativi.

Quando si ottiene la risposta esatta, il programma stampa il messaggio «GREAT! YOU GOT MY NUMBER» unitamente al numero di tentativi che sono stati necessari.

E' possibile quindi riprendere da capo il procedimento.

Ricordarsi che il programma genera ogni volta un nuovo numero casuale.

SUGGERIMENTI DI PROGRAMMAZIONE:

Nelle righe 40 e 50 viene usato un due punti per separare istruzioni multiple su una sola riga.
Ciò non solo risparmia lavoro di battitura ma nei lunghi programmi consente di risparmiare spazio di memoria.

Notare inoltre nelle istruzioni IF/THEN sulle stesse due righe che è stato istruito il computer a stampare (PRINT) qualche cosa anzichè saltare immediatamente a qualche altro punto nel programma.

L'ultimo punto illustra il motivo per l'uso dei numeri di riga a incrementi di 10. Dopo che il programma è stato scritto si è deciso di aggiungere la parte di conteggio. Semplicemente aggiungendo queste nuove righe al termine del programma numerate in modo da farle cadere tra le appropriate righe esistenti, il programma è stato facilmente modificato.

IL LANCIO DI DADI

Il programma che segue simula il lancio di due dadi. E' possibile utilizzarlo così come è oppure usarlo come parte di un gioco più complesso.

```
 5 PRINT "Care to try your luck?"
10 PRINT "RED DICE   = ";INT(6*RND(1))+1
20 PRINT "WHITE DICE = ";INT(6*RND(1))+1
30 PRINT "HIT SPACE BAR FOR ANOTHER ROLL":PRINT
40 GET A$: IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
```

Si vuole tentare la fortuna?

Da ciò che si è finora imparato sui numeri casuali e sul BASIC, si dovrebbe poter seguire e comprendere ciò che succede.

GRAFICI CASUALI

Come nota finale sui numeri casuali e come introduzione al disegno di grafici, è bene dedicare un istante ad immettere e ad eseguire questo semplice programma:

```
10 PRINT "{CLR/HOME}"
20 PRINT CHR$(205.5 + RND(1));
40 GOTO 20
```

Come ci si potrebbe aspettare, la riga chiave è la numero 20. Un'altra funzione, CHR$ (stringa di caratteri) fornisce un carattere basato su un numero di codici standard da 0 a 255. Ogni carattere che il COMMODORE 64 può stampare è codificato in questo modo (vedere Appendice F).

Per trovare rapidamente il codice di qualsiasi carattere battere:

PRINT ASC("X")

dove X è il carattere che si sta controllando (può trattarsi di qualsiasi carattere stampabile, compresi i segni grafici). La risposta è il codice per il carattere battuto. Come probabilmente ci si immagina, «ASC» è un'altra funzione che dà il codice standard «ASCII» per il carattere battuto.

E' ora possibile stampare quel carattere battendo:

PRINT CHR$(X)

Se si prova a battere:

PRINT CHR$ (205); CHR$ (206)

si vedranno comparire i due caratteri grafici di destra sui tasti M e N. Questi sono i due caratteri che il programma usa per il labirinto.

Usando la formula 205.5 + RND(1) il computer sceglie un numero casuale compreso tra 205.5 e 206.5. C'è una probabilità del 50% che il numero risulti al disopra o al disotto di 206. CHR$ ignora qualsiasi valore frazionario cosicchè metà delle volte viene stampato il carattere con il codice 205 mentre le rimanenti volte biene visualizzato il codice 206.

Se si vuole fare qualche esperimento con questo programma, si può ad esempio cercare di cambiare 205.5 aggiungendo o sottraendo un paio di decine. Ciò conferisce ad entrambi i caratteri una maggior possibilità di venir scelti.

CAPITOLO 5

COMANDI AVANZATI PER COLORI E GRAFICI

- **Colore e grafici**
- **Stampa (PRINT) dei colori**
- **Codici CHR$ dei colori**
- **PEEK e POKE**
- **Grafici sullo schermo**
- **Altro sulle palline rimbalzanti**

COLORE E GRAFICI

Finora sono state esplorate alcune delle sofisticate capacità di calcolo del COMMODORE 64. Ma una delle sue caratteristiche più affascinanti è la sua eccezionale abilità di produrre grafici e colori.

Si è visto un rapido esempio dei grafici nei programmi della «pallina rimbalzante» e del «labirinto». Ma questi hanno soltanto sfiorato la potenza a disposizione.

In questo capitolo verranno introdotti numerosi nuovi concetti per spiegare la programmazione dei grafici e dei colori e per mostrare come è possibile creare propri giochi ed esempi di animazione avanzata.

Poichè finora ci si è concentrati sulle capacità di calcolo della macchina, tutti gli schermi generati erano in un solo colore (testo azzurro su fondo blu con un bordo azzurro).

In questo capitolo si vedrà come aggiungere il colore ai programmi e come controllare tutti quegli strani simboli grafici che figurano sulla tastiera.

LA STAMPA DEI COLORI

Come si sarà scoperto nella prova di messa a punto del televisore del Capitolo 1, è possibile cambiare i colori del testo semplicemente tenendo abbassato il tasto CTRL ed uno dei tasti dei colori. Ciò funziona molto bene nel modo immediato ma cosa succede se si vuole incorporare cambiamenti di colore nei programmi?

Quando è stato presentato il programma della «pallina rimbalzante» si è visto come è possibile incorporare comandi da tastiera tipo movimento del cursore all'interno delle istruzioni PRINT. In modo analogo è possibile aggiungere modifiche al colore del testo ai programmi.

E' possibile lavorare con un'intera serie di 16 colori di testo. Usando il tasto CTRL ed un tasto numerico, sono disponibili i seguenti colori:

1	2	3	4	5	6	7	8
Nero	Bianco	Rosso	Blu-verde	Porpora	Verde	Blu	Giallo

Se si tiene abbassato il tasto C= unitamente al tasto di numero appropriato, possono essere usati questi ulteriori otto colori:

1	2	3	4	5	6	7	8
Arancio	Marrone	Rosso chiaro	Grigio 1	Grigio 2	Verde chiaro	Azzurro	Grigio 3

Battere NEW e provare con quanto segue. Tenere abbassato il tasto CTRL e contemporaneamente battere il tasto 1. Successivamente, battere il tasto R senza però tenere abbassato il tasto CTRL. Ora premendo di nuovo il tasto CTRL e contemporaneamente premere il tasto 2. Sollevare il tasto CTRL e premere il tasto A. Provare con i vari numeri alternandoli con le lettere e battere la parola RAINBOW come segue:

```
10 PRINT"↑R↑A↑I↑N↑B↑O↑W"
         ↑ ↑ ↑ ↑ ↑ ↑ ↑
   CTRL 1 2 3 4 5 6 7
RUN
RAINBOW
```

Nello stesso modo in cui i comandi del cursore compaiono come caratteri grafici all'interno dei segni di virgolette delle istruzioni di stampa, così anche i comandi dei colori sono rappresentati sotto forma di caratteri grafici.

Nel precedente esempio quando si è tenuto abbassato CTRL e si è battuto 3 è comparso il carattere "←". CTRL 7 hanno fatto comparire "£". Ciascun comando dei colori, usato in questo modo, visualizzerà proprio codice grafico esclusivo. La tabella mostra la rappresentazioni grafiche di ciascun comando di colore stampabile.

TASTIERA	COLORE	SCHERMO	TASTIERA	COLORE	SCHERMO
CTRL 1	NERO		C= 1	ARANCIO	
CTRL 2	BIANCO		C= 2	MARRONE	
CTRL 3	ROSSO		C= 3	ROSSO CHIA.	
CTRL 4	BLU VERDE		C= 4	GRIGIO 1	
CTRL 5	PORPORA		C= 5	GRIGIO 2	
CTRL 6	VERDE		C= 6	VERDE CHIA.	
CTRL 7	BLU		C= 7	AZZURRO	
CTRL 8	GIALLO		C= 8	GRIGIO 3	

Quantunque l'istruzione PRINT possa apparire un poco strana sullo schermo, quando si esegue (RUN) il programma verrà visualizzato soltanto il testo. E questo cambierà automaticamente colore secondo i comandi inseriti nell'istruzione di stampa.

Provare con altri esempi a propria scelta mescolando qualsiasi numero di colori all'interno di un'istruzione PRINT. Ricordarsi anche che è possibile usare la seconda serie di colori di testo servendosi dei tasti COMMODORE e dei tasti numerici.

SUGGERIMENTO:

Si noterà dopo aver eseguito un programma con modifiche di colore o di modo (negativo) che la richiesta "READY" e qualsiasi ulteriore tasto battuto compare nello stesso colore o nello stesso modo. Per ritornare alla presentazione normale, ricordarsi di premere:

RUN/STOP e RESTORE

CODICI CHR$ DEI COLORI

Prima di procedere con questo capitolo, occorre dare un breve sguardo all'Appendice F.

Si è notato nell'osservare la lista dei codici CHR$ nell'Appendice F che ciascun colore (nonchè la maggior parte degli altri comandi di tastiera, ad esempio i movimenti del cursore) hanno un codice esclusivo. Questi codici possono essere stampati direttamente per ottenere gli stessi risultati della pressione del tasto CTRL e dell'appropriato tasto all'interno dell'istruzione PRINT.

Provare con questo esempio:

```
NEW
10 PRINT CHR$(147) : REM (CLR/HOME)
10 PRINT CHR$(30);"CHR$(30) CHANGES ME TO?"
RUN
CHR$(30) CHANGES ME TO?
```

Il testo dovrebbe ora risultare verde. In molti casi l'uso della funzione CHR$ sarà molto facile, particolarmente se si vuole provare a cambiare i colori. Alla pagina seguente c'è un modo diverso per ottenere un arcobaleno di colori. Dato che c'è un numero di righe simili (40-110) usare i tasti di editing per risparmiare molta battitura. Vedere le note dopo la lista per rinfrescare la memoria sulle procedure di correzione.

NEW

```
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      " ;:REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
50 PRINT CHR$(28);: GOTO 10
```

```
60 PRINT CHR$(30);: GOTO 10
70 PRINT CHR$(31);: GOTO 10
80 PRINT CHR$(144);: GOTO 10
90 PRINT CHR$(156);: GOTO 10
100 PRINT CHR$(158);: GOTO 10
110 PRINT CHR$(159);: GOTO 10
```

Battere normalmente le righe da 5 a 40. Lo schermo dovrebbe apparire come segue:

```
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      ":REM  REVERSE BARS
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5);: GOTO 10
```

NOTE DI CORREZIONE

Usare il tasto di movimento del cursore verso l'alto per posizionare il cursore alla riga 40. Quindi battere 5 sopra il 4 di 40. Successivamente usare il tasto per lo spostamento a destra del cursore per portarsi sul 5 nelle parentesi CHR$. Premere SHIFT INST/DEL per creare uno spazio e battere «28». Ora premere semplicemente RETURN con il cursore disposto ovunque sulla riga.

Lo schermo dovrebbe presentarsi come segue:

```
NEW

1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      " :REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
50 PRINT CHR$(28);: GOTO 10
```

Non è il caso di preoccuparsi. La riga 40 c'è ancora. Basta listare (LIST) il programma e vedere. Usando la stessa procedura continuare a modificare l'ultima riga con un nuovo numero di riga ed il codice CHR$ fino a che non sono state immesse tutte le righe rimanenti. Come controllo

finale, listare l'intero programma prima di eseguirlo per assicurarsi che tutte le righe siano state immesse correttamente (RUN).

Ecco una breve spiegazione di cosa sta succedendo.

E' stata finora probabilmente rappresentata la maggior parte del programma delle barre a colori salvo qualche nuova strana istruzione alla riga 30. Ma si può rapidamente osservare cosa in effetti fa l'intero programma.

La riga 5 stampa il codice CHR$ per CLR/HOME.

La riga 10 attiva le scritte in negativo e stampa 5 spazi, che si trasformano in una barra dato che sono in negativo. La prima volta nel corso del programma la barra sarà azzurra, il normale colore del testo.

La riga 20 usa la funzione del numero casuale per scegliere un colore a caso tra 1 e 8.

La riga 30 continua una variazione dell'istruzione IF . . . THEN che è detta ON . . . GOTO e che consente al programma di scegliere da una lista di numeri ai quali andare. Se la variabile (in questo CL) ha un valore di 1, il primo numero di riga è quello scelto (in questo caso 40). Se il valore è 2, viene usato il secondo numero della lista, ecc.

Le righe da 40 a 110 convertono semplicemente i colori casuali dei tasti nell'appropriato codice CHR$ per quel colore e riportano il programma alla riga 10 per stampare (PRINT) una sezione della barra in quel colore. Quindi l'intero processo riparte da capo.

Provare a produrre 16 numeri casuali, a espandere ON . . . GOTO per manipolarli e ad aggiungere i rimanenti codici CHR$ per visualizzare i rimanenti otto colori.

PEEK e POKE

Non ci proponiamo ora di sezionare il computer ma semplicemente di dare uno sguardo all'interno della macchina per «inserire» determinate cose in qualche punto.

Esattamente come le variabili possono essere viste come una rappresentazione di «scatola» all'interno della macchina nelle quali si inseriscono informazioni, così è possibile pensare che alcune «scatole» particolarmente definite all'interno del computer rappresentino locazioni specifiche di memoria.

Il COMMODORE 64 esamina queste locazioni di memoria per rendersi conto di quale deve essere il colore dello sfondo e del bordo, quali caratteri devono essere visualizzati sullo schermo – e dove – e per numerosi altri motivi.

Inserendo («POKE») un valore diverso nell'appropriata locazione di memoria è possibile cambiare colore, definire e spostare oggetti ed addirittura creare musica.

Queste locazioni di memoria possono essere rappresentate come segue:

53280 X	53281 Y	53282	53283
COLORE DEL BORDO	COLORE DELLO SFONDO		

A pagina 60 sono state indicate soltanto quattro locazioni, due delle quali controllano i colori dello sfondo e dello schermo. Provare a battere istruzione che segue:

POKE 53281,7 RETURN

I colore dello sfondo dello schermo cambierà in giallo in quanto è stato inserito il valore 7 – corrispondente al giallo – nella locazione che controlla il colore dello sfondo.

Provare ad inserire (POKE) diversi valori nella locazione del colore dello sfondo ed osservare i relativi risultati. E' possibile inserire (POKE) qualsiasi valore compreso fra 0 e 255 ma di essi soltanto quelli da 0 a 15 funzioneranno.

I valori effettivi per inserire (POKE) ciascun colore sono:

0	NERO	8	ARANCIO
1	BIANCO	9	MARRONE
2	ROSSO	10	ROSSO CHIARO
3	CELESTE	11	GRIGIO 1
4	PORPORA	12	GRIGIO 2
5	VERDE	13	VERDE CHIARO
6	BLU	14	AZZURRO
7	GIALLO	15	GRIGIO 3

Si può pensare ad un modo per visualizzare le varie combinazioni di colore dello sfondo e del bordo? Il programma che segue può essere di qualche aiuto:

```
NEW

10 FOR BA = 0 TO 15
20 FOR BO  = 0 TO 15
30 POKE 53280, BA
40 POKE 53281, BO
50 FOR X = 1 TO 2000: NEXT X
60 NEXT BO: NEXT BA

RUN
```

Sono stati in questo caso create due semplici iterazioni per inserire (POKE) i vari valori in modo da cambiare i colori del fondo e del bordo. L'iterazione DELAY (ritardo) alla riga 50 serve soltanto a rallentare il movimento.

Chi è curioso può provare anche il programma che segue:

? PEEK (53280) AND 15

Si dovrebbe ottenere un valore di 15. Questo è l'ultimo valore attribuito al bordo ed ha un senso in quanto sia il colore dello sfondo che del bordo è il grigio (valore 15) dopo che il programma è eseguito.

Immettendo AND 15 si eliminano tutti gli altri valori salvo 1 - 15 a causa del modo in cui i codici dei colori sono memorizzati nel computer. Normalmente ci si aspetterà di trovare lo stesso valore che è stato inserito (POKE) per ultimo nella locazione.

In generale PEEK consente di esaminare una locazione specifica e vedere quale valore è in essa presente. E' possibile pensare all'aggiunta di una riga al programma che visualizza il valore dello sfondo e del bordo mentre il programma viene eseguito? Eccola:

25 PRINT CHR$(147); "BORDER = ";PEEK(53280) AND 15, "BACK-GROUND = "; PEEK (53281) AND 15

GRAFICI SULLO SCHERMO

In tutta la stampa delle informazioni che è stata finora eseguita, il computer ha trattato le informazioni in modo sequenziale, stampando cioè un carattere dopo l'altro, iniziando dalla posizione corrente del cursore (salvo nel caso in cui si è chiesta una nuova riga o è stato usato «,» nella formattazione di PRINT).

Per stampare (PRINT) i dati in un particolare punto è possibile iniziare da un punto noto sullo schermo e PRINT il numero appropriato di comandi del cursore per formattare lo schermo. Ma ciò richiede passi di programma e molto tempo.

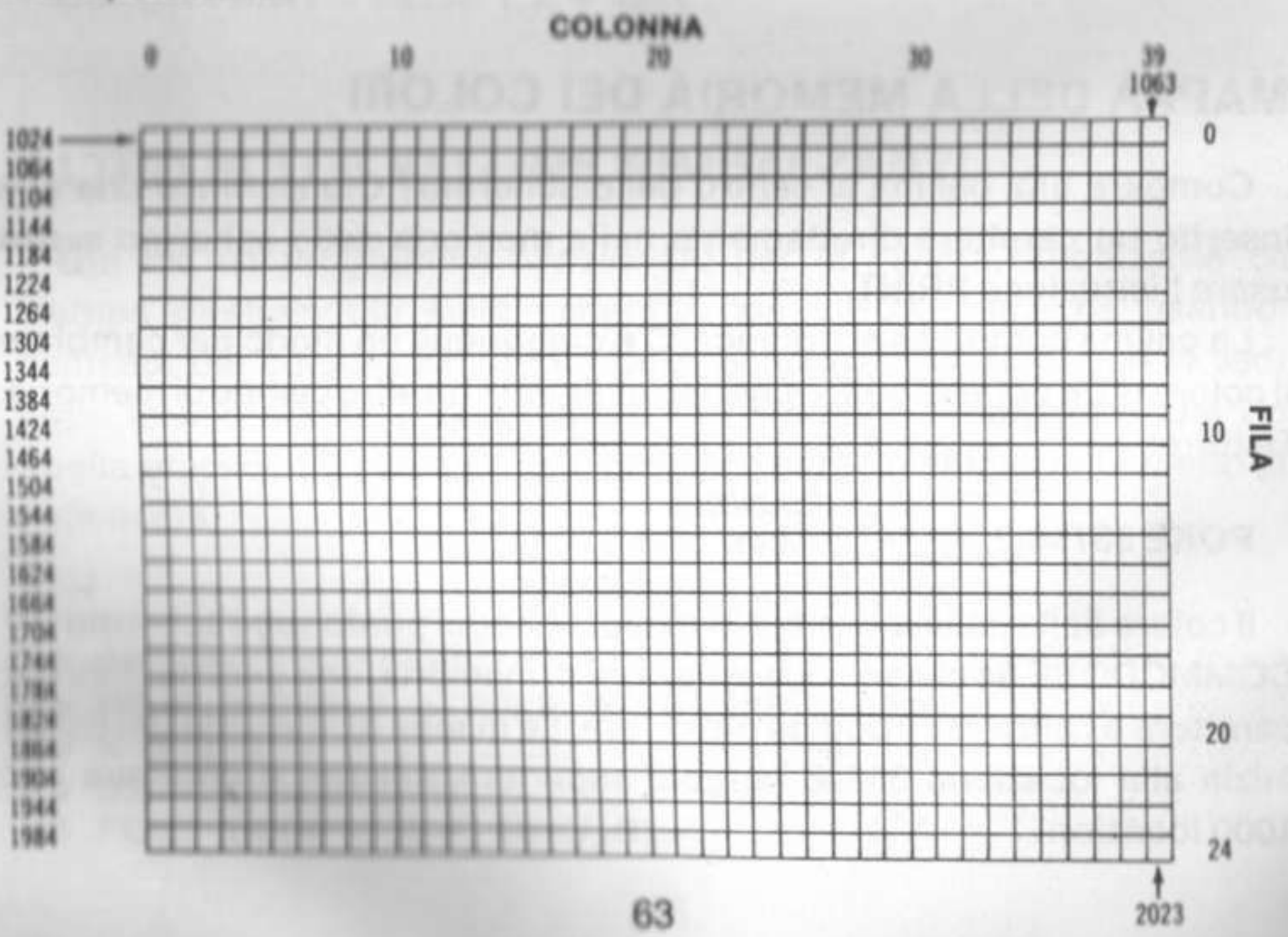
Esattamente come ci sono taluni punti nella memoria del COMMODORE 64 per controllare i colori così ce ne sono anche alcuni che è possibile usare per controllare direttamente ciascuna locazione sullo schermo.

LA MAPPA DI MEMORIA DELLO SCHERMO

Dato che lo schermo del computer è in grado di contenere 1000 caratteri (40 colonne per 25 righe), significa che si sono 1000 locazioni di memoria accantonate per gestire ciò che viene posto sullo schermo stesso. La struttura dello schermo può essere vista come una griglia, in cui ciascun quadrato rappresenta una locazione di memoria.

E dato che ciascuna locazione della memoria può contenere un numero da 0 a 255, ci sono 256 possibili valori per ciascuna locazione di memoria. Questi valori rappresentano i diversi caratteri che COMMODORE 64 può visualizzare (vedere Appendice E).

Inserendo (POKE) il valore di un carattere nell'appropriata locazione di memoria dello schermo, quel carattere viene visualizzato nella posizione corrispondente.



La memoria dello schermo nel COMMODORE 64 inizia normalmente alla locazione di memoria 1024 e termina alla locazione 2023. La locazione 1024 si trova nell'angolo superiore sinistro dello schermo. La locazione 1025 è la posizione del successivo carattere alla sua destra e così via lungo la fila. La locazione 1063 è la posizione più a destra della prima fila. La successiva locazione che segue l'ultimo carattere sulla fila è il primo carattere della fila successiva.

Si supponga di voler controllare una pallina che rimbalza sullo schermo. La pallina si trova nel mezzo dello schermo, sulla colonna 20 fila 12. La formula per il calcolo della locazione di memoria sullo schermo è la seguente:

```
POINT = 1024 + X +40*Y
        ←COLONNA
        ←FILA
```

dove X è la colonna e Y è la fila.

Pertanto la locazione di memoria della pallina è:

```
1024 + 20 + 480 oppure 1524
       ←COLONNA
       ←FILA (40*12)
```

Cancellare ora lo schermo con SHIFT e CLR/HOME e battere:

```
POKE 1524,81
     ←CODICE DEL CARATTERE
     ←LOCAZIONE
```

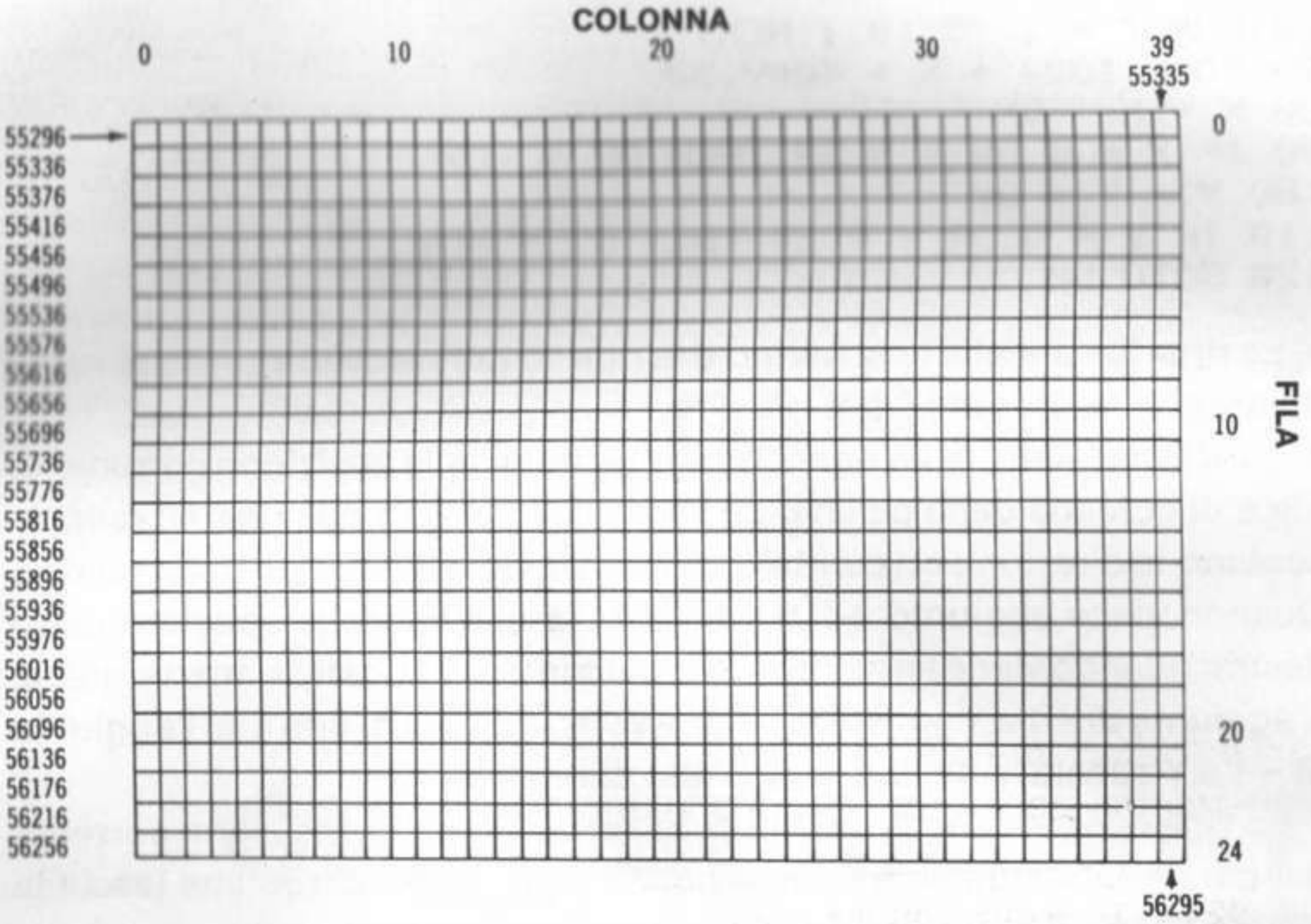
MAPPA DELLA MEMORIA DEI COLORI

Compare una pallina al centro dello schermo? Ciò significa che si è inserito un carattere direttamente nella memoria dello schermo senza usare l'istruzione PRINT.

La pallina comparsa era bianca. C'è comunque un modo per cambiare il colore di un oggetto sullo schermo alterando un altro campo di memoria. Battere:

```
POKE 55796,2
     ←LOCAZIONE
     ←COLORE
```

Il colore della pallina cambia in rosso. Per ogni punto sullo schermo del COMMODORE 64 ci sono due locazioni di memoria, una per il codice del carattere e l'altra per il codice del colore. La mappa di memoria dei colori inizia alla locazione 55296 (angolo superiore sinistro) e continua per 1000 locazioni.



Gli stessi codici dei colori da 0 a 15 che sono stati usati per cambiare i colori del bordo e dello sfondo qui possono essere usati per cambiare direttamente i colori dei caratteri.

La formula usata per calcolare le locazioni di memoria dello schermo può essere modificata per fornire le locazioni in cui inserire (POKE) i codici dei colori. La nuova formula è:

COLOR PRINT = 55296 + X + 40·Y

ALTRO SULLE PALLINE RIMBALZANTI

Qui c'è un programma revisionato per la pallina rimbalzante che stampa direttamente sullo schermo con i POKE anzichè usando i comandi del cursore all'interno delle istruzioni PRINT. Come si vedrà dopo aver eseguito il programma, la flessibilità è maggiore che non quella precedente e consente di programmare un'animazione molto più sofisticata.

```
NEW

10 PRINT "{CLR/HOME}"
20 POKE 53280,7 : POKE 53281,13
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
```

```
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X = 0 OR X = 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
120 GOTO 50
```

La riga 10 cancella lo schermo e la riga 20 definisce lo sfondo al verde chiaro con un bordo giallo.

Le variabili X e Y nella riga 30 tengono nota della posizione corrente di fila e di colonna della pallina. Le variabili DX e DY nella riga 40 rappresentano la direzione orizzontale e verticale del movimento della pallina. Quando viene aggiunto +1 al valore X, la pallina viene spostata verso destra; quando viene aggiunto −1 la pallina viene spostata verso sinistra. L'aggiunta di +1 a Y sposta la pallina verso il basso di una fila; l'aggiunta di −1 a Y sposta la pallina verso l'alto di una fila.

La riga 50 inserisce la pallina sullo schermo nella posizione corrente del cursore. La riga 60 è l'ormai nota iterazione di ritardo che lascia la pallina sullo schermo quanto basta per poterla osservare.

La riga 70 cancella la pallina inserendo uno spazio (codice 32) nel punto in cui essa si trovava precedentemente sullo schermo.

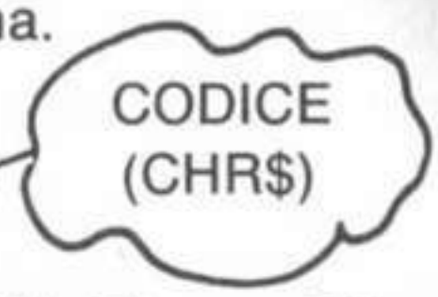
La riga 80 raggiunge il fattore di direzione a X. La riga 90 si accerta che la pallina abbia raggiunto una delle pareti laterali, invertendo la direzione se c'è un rimbalzo. La riga 100 e 110 esegue la stessa cosa per le pareti superiore ed inferiore.

La riga 120 riporta il programma allo schermo e sposta di nuova la pallina.

Cambiando il codice nella riga 50 da 81 ad un altro codice di carattere, è possibile trasformare la pallina in qualsiasi altro carattere. Se si cambia DX o DY in zero, la pallina rimbalza in maniera diritta anzichè in diagonale.

E' inoltre possibile aggiungere qualche altra informazione. Finora le sole cose controllate erano i valori X e Y che uscivano dai limiti dello schermo. Aggiungere le seguenti righe al programma.

```
21 FOR L = 1 TO 10
25 POKE 1024 + INT(RND(1)*1000), 166
27 NEXT L
115 IF PEEK(1024 + X + 40*Y) = 166 THEN DX = -DX:
    GOTO 80
```



Le righe da 21 a 27 inseriscono 10 blocchi sullo schermo in posizioni casuali. La riga 115 controlla (PEEK) che la pallina sia sul punto da rimbalzare in un blocco ed in questo caso ne cambia la direzione.

CAPITOLO 6

GRAFICI ANIMATI (SPRITES)

- **Introduzione ai grafici animati**
- **Creazione di effetti di animazione**
- **Ulteriori note sugli effetti di animazione**
- **Aritmetica binaria**

INTRODUZIONE AI GRAFICI ANIMATI

Nei precedenti capitoli che si occupavano dei grafici si è visto che i simboli grafici possono essere usati nelle istruzioni PRINT per creare effetti di animazione ed aggiungere altre caratteristiche agli schermi.

E' stato inoltre presentato un modo inserire (POKE) codici di caratteri in specifiche locazioni di memoria dello schermo, in modo da far comparire direttamente gli appropriati caratteri nel punto voluto sullo schermo.

La creazione di effetti di animazione in entrambi questi casi richiede una massa notevole di lavoro in quanto si devono creare gli oggetti dagli esistenti simboli grafici. Lo spostamento dell'oggetto richiede un certo numero di istruzioni di programma per seguire l'oggetto e portarlo in una nuova posizione. Ed a causa delle limitazioni nell'uso dei simboli grafici, il profilo e la risoluzione dell'oggetto potrebbero non essere quelli ideali.

Usando i grafici animati si elimina una buona parte di questi problemi. Con il termine «grafico animato» s'intende un oggetto programmabile ad alta risoluzione che può essere creato attribuendogli pressochè qualsiasi profilo – attraverso i comandi BASIC. L'oggetto può essere facilmente spostato sullo schermo semplicemente indicando al computer la posizione che l'oggetto deve occupare. Il computer si occupa di tutto il resto.

Ma i grafici animati hanno anche altre capacità: il loro colore può essere cambiato, è possibile dire se un oggetto entra in collisione con un altro, li si può disporre in modo che uno si porti davanti o dietro un altro e le loro dimensioni possono venir facilmente ampliate.

L'impegno per far tutto ciò è minimo ma per poter ottenere effetti di animazione occorre conoscere qualche altro dettaglio sul modo in cui il COMMODORE 64 opera ed sul modo in cui i numeri sono manipolati all'interno del computer. Non è tuttavia difficile quanto sembra: basta seguire gli esempi per poter creare propri effetti di animazione o far compiere agli oggetti le cose più strane.

CREAZIONE DI EFFETTI ANIMAZIONE

Gli effetti di animazione sono controllati da uno speciale dispositivo per la creazione di immagini incorporato nel COMMODORE 64. Questo dispositivo gestisce lo schermo video svolgendo tutto il lavoro di creare e di seguire i caratteri ed i grafici, creando colori e spostandoli sullo schermo.

Il circuito dello schermo dispone di 46 diverse locazioni «ON/OFF» che agiscono come locazioni interne di memoria. Ciascuna di queste locazioni si suddivide in una serie di otto blocchi. E ciascun blocco può essere a sua volta «on» o «off» (rispettivamente «attivato» o «disattivato»). Ma di ciò si parlerà più a lungo in seguito. Inserendo (POKE) l'appropriato valore decimale nella corretta locazione di memoria è possibile controllare gli effetti di animazione.

Oltre ad accedere a molte delle locazioni che creano immagini si userà anche una parte della memoria principale di COMMODORE 64 per memorizzare le informazioni (dati) che definiscono i grafici animati. Infine, verranno usate otto locazioni di memoria immediatamente dopo la memoria dello schermo per dire al computer esattamente da quale area di memoria ciascun disegno animato ricaverà i suoi dati.

Man mano si procederà con gli esempi il processo apparirà abbastanza lineare e se ne comprenderà agevolmente il funzionamento.

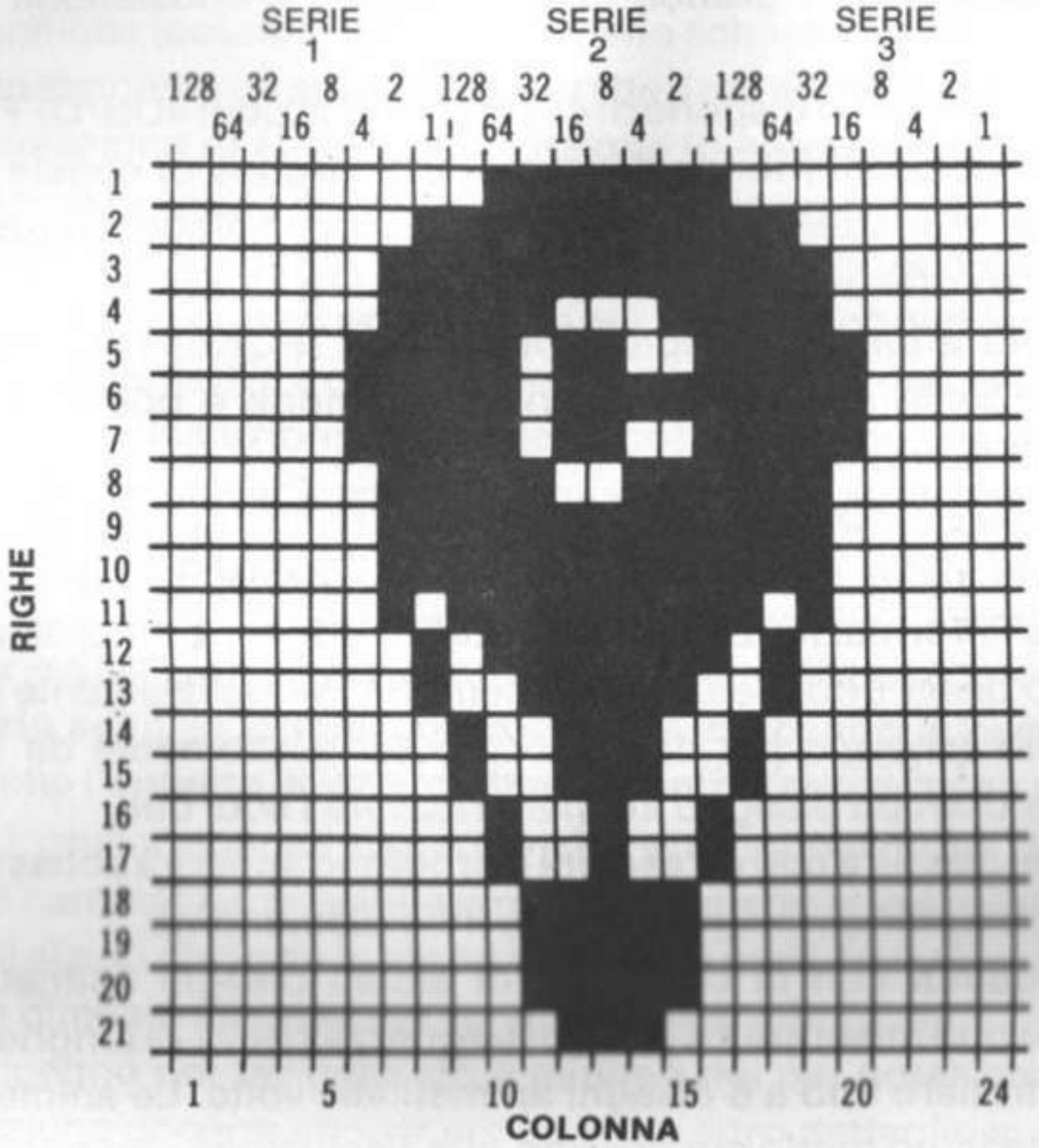
Procediamo quindi con la creazione di alcuni disegni animati. Un oggetto animato può misurare 24 punti di larghezza per 21 di lunghezza e si possono controllare fino a 8 disegni animati alla volta. Le animazioni sono visualizzate in un modo speciale ad alta risoluzione che trasforma lo schermo in un'area larga 328 punti ed alta 200 punti.

Si voglia ad esempio creare un pallone e farlo svolazzare nel cielo. Il pallone può essere disegnato nella griglia 24 x 21 (pagina 70).

La fase successiva consiste nel convertire il disegno grafico in dati che il computer può usare. Occorre a questo punto procurarsi un blocco per appunti o carta per grafici e disegnare una semplice griglia che misuri 21 spazi in altezza e 24 in larghezza. Nella parte superiore scrivere 128, 64, 32, 16, 8, 4, 2, 1 per tre volte (come indicato) per ciascuno dei quadrati larghezza 24. Numerare il lato sinistro della griglia da 1 e 21. Scrivere la parola DATA al termine di ciascuna riga. Riempire ora la griglia con un disegno a piacere oppure usare il pallone riportato in figura. E' più facile disegnare per prima cosa il profilo esterno e quindi lavorare all'interno riempiendo la griglia.

Ora considerando «on» tutti i quadratini riempiti, sostituire un 1 a ciascuno di essi. I quadrati vuoti sono quelli considerati «off» e ad essi corrisponde uno zero.

Iniziando sulla prima riga, occorre convertire i punti in tre elementi di dati separati che il computer può leggere. Ciascuna serie di otto quadrati nel pallone corrisponde ad un elemento di dati detto byte. Procedendo da sinistra, i primi otto quadrati sono vuoti, ossia 0, così il valore per quella serie di numeri è 0.



La serie intermedia assomiglia a quella che segue (di nuovo 1 indica un quadratino pieno, 0 un quadratino vuoto):

128	64	32	16	8	4	2	1
0	1	1	1	1	1	1	1

0 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127

La terza serie sulla prima riga contiene a sua volta degli spazi vuoti cosicchè anch'essa è uguale a zero. Pertanto i dati per la prima riga sono:

DATA 0, 127, 0

Le serie che costituiscono la riga due sono calcolate come seque:

Serie 1:	0	0	0	0	0	0	0	1

1 = 1

Serie 2:	1	1	1	1	1	1	1	1

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Serie 3:	1	1	0	0	0	0	0	0

128 + 64 = 192

Per la riga 2, i dati sarebbero:

DATA 1, 255, 192

Allo stesso modo le tre serie che costituiscono ciascuna riga rimanente vengono convertite nel rispettivo valore decimale. Procedere al resto della conversione di questo esempio.

Ora che si dispone dei dati per l'oggetto, come è possibile usarli? Battere il seguente programma e vedere cosa succede:

```
1 REM UP, UP, AND AWAY!
5 PRINT "{CLR/HOME}"
10 V=53248  : REM START OF  DISPLAY CHIP
11 POKE V+21,4 : REM ENABLE SPRITE 2
12 POKE 2042,13 : REM SPRITE 2 DATA FROM 13TH BLK
20 FOR N = 0 TO 62: READ Q : POKE 832+N,Q: NEXT
30 FOR X = 0 TO 200
40 POKE V+4,X: REM UPDATE X COORDINATES
50 POKE V+5,X: REM UPDATE Y COORDINATES
60 NEXT X
70 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,15,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Grafico Animato

Ricava le informazioni da DATA*

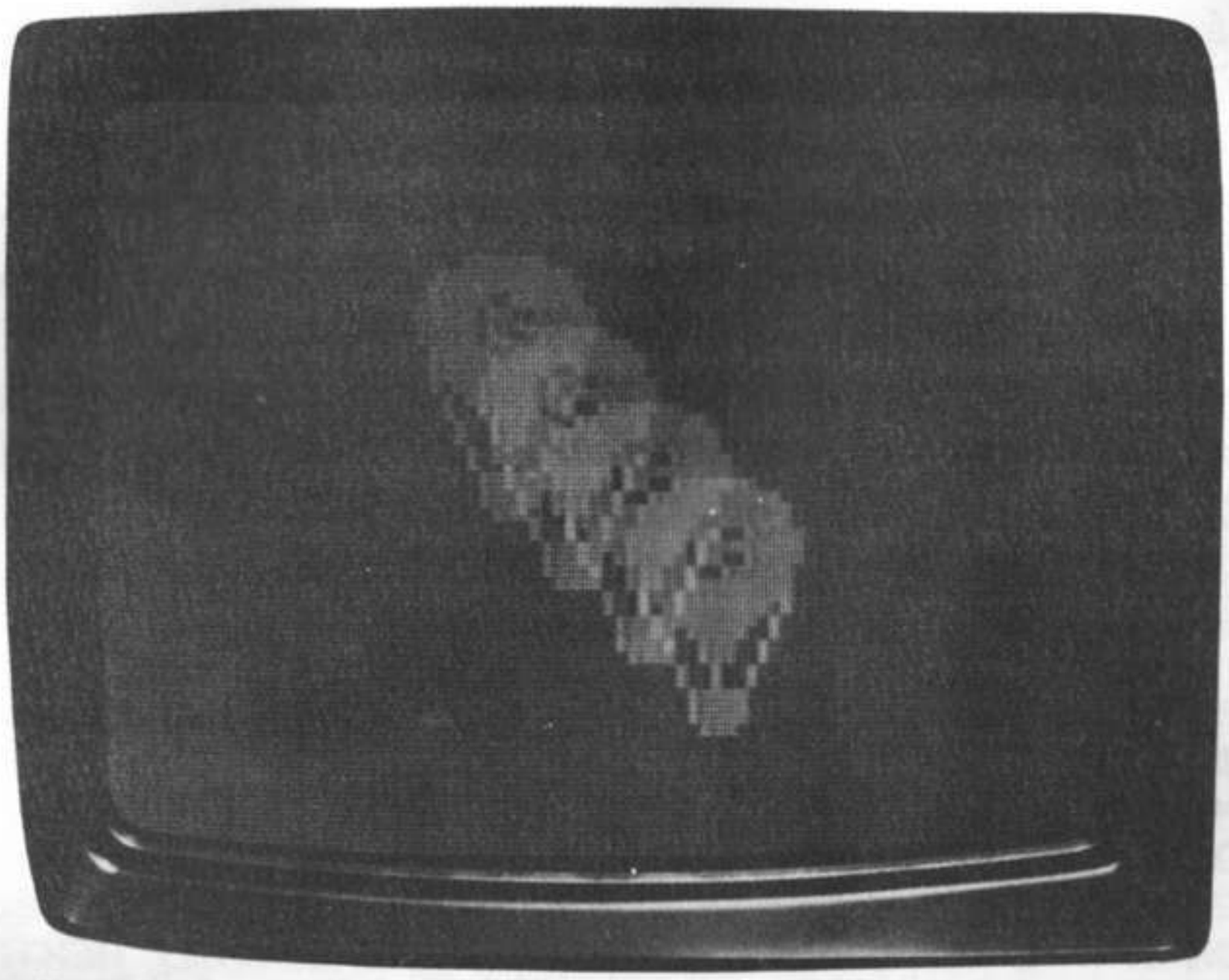
Informazioni lette da Q*

* Per ulteriori particolari su READ e DATA vedere il Capitolo 8.

Se tutto è stato battuto correttamente, il pallone veleggia dolcemente nel cielo (pagina 72).

Per comprendere cosa è successo, occorre per prima cosa sapere quali sono le locazioni che creano le immagini che controllano le funzioni necesarie allo scopo. Queste locazioni, dette registri, possono essere illustrate in questo modo:

Registro(i)	*Descrizione*
0	Coordinata X del disegno animato 0
1	Coordinata Y del disegno animato 0
2-15	Abbinato come 0 e 1 per i disegni animati 1-7
16	Bit più significativo – Coordinata X
21	Compare il disegno animato: 1 = appare, 0 = scompare
29	Espande il disegno animato in direzione «X»
23	Espande il disegno animato in direzione «Y»
39-46	Disegno animato colore 0-7



Fotografia effettiva dello schermo

Oltre a queste informazioni occorre sapere da quale sezione di 64 blocchi ciascuna serie di 8 blocchi di memoria otterrà i disegni animati nei rispettivi dati (1 serie non è usata).

Questi dati sono gestiti dalle 8 locazioni che seguono la memoria dello schermo:

2040	41	42	43	44	45	46	2047
↑	↑	↑	↑	↑	↑	↑	↑
SPRITE 0	1	2	3	4	5	6	7

Verrà ora descritta la procedura esatta per far muovere i disegni ed infine verrà scritto il programma relativo.

Sono poche le cose necessarie per creare e spostare un oggetto.

1. Far comparire il disegno o i disegni appropriati sullo schermo inserendoli (POKE) nella locazione 21 che lo attiva.
2. Definire il puntatore del disegno (locazioni 2040-7) nel punto in cui devono essere letti i dati del disegno.
3. Inserire (POKE) i dati effettivi in memoria.
4. Attraverso un'iterazione, aggiornare le coordinate X e Y per far muovere il disegno.
5. Facoltativamente, è possibile espandere l'oggetto, cambiarne i colori oppure eseguire numerose funzioni speciali. Usare la locazione 29 per espandere il disegno nella direzione «X» e la locazione 23 per espanderlo nella direzione «Y».

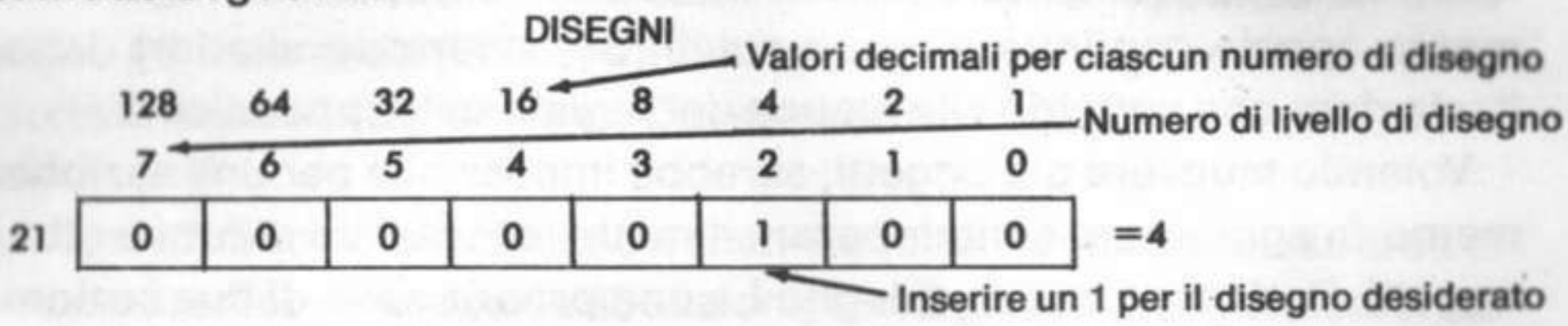
Nel programma ci sono soltanto poche cose che potrebbero non essere note dali punti finora discuesi.

Nella riga 10;
V = 53248

definisce V alla locazione di partenza della memoria del chip del video. In questo modo si aumenta semplicemente V del numero necessario per ottenere la locazione effettiva di memoria. I numeri di registro sono quelli indicati nella mappa dei registri.

Nella riga 11,
POKE V+21,4

fa sì che compaia il disegno 2 inserendo un 4 in quello che viene chiamato registro di abilitazione (21) per attivare il disegno 2. Si può pensare a ciò nel modo seguente:



Ciascun livello di disegno è rappresentato nella sezione 21 della memoria dei disegni e 4 risulta essere il livello di disegno 2. Se si usasse il livello 3 occorrerebbe inserire un 1 nel disegno 3 che ha un valore di 8. In effetti se si usassero entrambi i disegni 2 e 3 si inserirebbe un 1 sia in 4 che in 8. Occorrerebbe quindi sommare i numeri esattamente come si è fatto con i DATA sulla carta per grafici. Così l'attivazione dei disegni 2 e 3 verrebbe rappresentata come V+21,12.

Nella riga 12;
POKE 2042,13

istruisce il computer ad ottenere i dati per il disegno 2 (locazione 2042) dalla tredicesima area della memoria. Si sa dalla costruzione del disegno che esso occupa 63 sezioni di memoria. La cosa potrebbe essere passata inosservata ma i numeri che sono stati posti nella parte superiore della griglia corrispondono a 3 byte el computer. In altre parole ciascuna serie dei seguenti numeri – 128, 64, 32, 16, 8, 4, 2, 1 equivale ad 1 byte di memoria del computer.

Pertanto con le 21 file della griglia moltiplicate per 3 byte per ciascuna fila, ciascun disegno occupa 63 byte di memoria.

1 DISEGNO INTERO

```
20 FOR N = 0 to 62: READ Q: POKE 832+N,Q: NEXT
```

Questa riga si occupa della creazione effettiva del disegno. I 63 byte di dati che rappresentano il disegno creato vengono letti (READ) nell'iterazione ed inseriti (POKE) nel tredicesimo blocco di memoria. Questo inizia alla locazione 832.

```
30 FOR X – 0 TO 200
40 POKE V+4,X
50 POKE V+5,X
```

COORDINATA X DEL DISEGNO 2

COORDINATA Y DEL DISEGNO 2

Si ricorderà che la coordinata X rappresenta il movimento orizzontale di un oggetto sullo schermo e la coordinata Y ne rappresenta il movimento verticale. Pertanto quando i valori di X cambiano nella riga 30 da 0 a 200 (un numero alla volta), il disegno si muove sullo schermo verso il basso e verso destra di uno spazio per ogni numero. I numeri vengono letti (READ) dal computer abbastanza velocemente per far sì che il movimento appaia continuo e non a scatti. Se occorrono ulteriori dettagli basta dare uno sguardo alla mappa dei registri nell'Appendice O.

Volendo muovere più oggetti, sarebbe impossibile per una sezione di memoria aggiornare contemporaneamente le locazioni di tutti e otto gli oggetti. Pertanto ciascun disegno ha una propria serie di due sezioni di memoria che gli consentono di muoversi sullo schermo.

La riga 70 inizia il ciclo da capo dopo una passata sullo schermo. Il resto del programma è costituito dai dati per il pallone. Naturalmente appaiono ben diversi sullo schermo, non è vero?

Provare ora ad aggiungere la riga seguente:

25 POKE V+23,4: POKE V+29,4: REM EXPAND

ed eseguire (RUN) il programma di nuovo. Il pallone si è allargato raddoppiando la sua misura originale e tutto è avvenuto in modo molto semplice. Inserendo (POKE) 4 (di nuovo per indicare il disegno 2) nelle sezione di memoria 23 e 29, il disegno 2 è stato ampliato in direzione X e Y.

E' importante notare che il movimento inizierà nell'angolo superiore sinistro dell'oggetto. Espandendo l'oggetto in entrambe le direzione il punto di partenza rimane lo stesso. Per ulteriore divertimento, apportare le seguenti modifiche:

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X = 1 to 190
45 POKE V+6,X
55 POKE V+7,190-X
```

E' stato creato un secondo disegno (numero 3) inserendo (POKE) 12 nella locazione di memoria che fa sì che compaia il disegno (V+21). Il 12 attiva i disegni 3 e 2 (00001100 = 12).

Le righe aggiunte 45 e 55 spostano il disegno 3 inserendo (POKE) valori nelle locazioni delle coordinate X e Y del disegno (V+6 e V+7).

Si vuole riempire il cielo con ulteriore azione? Basta apportare queste aggiunte:

```
11 POKE V+21,28
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

28 E' IN EFFETTI 4 (DISEGNO 2) + 8 (DISEGNO 3) + 16 (DISEGNO 4)

Nella riga 11 stavolta è stato fatto in modo di far apparire un altro disegno (4) inserendo (POKE) 28 nell'appropriata locazione «on» della sezione di memoria del disegno. Ora sono attivati i disegni 2-4 (00011100 = 28).

La riga 12 indica che il disegno 4 ricaverà i suoi dati dalla stessa area di memoria (tredicesima area da 63 sezioni) come gli altri disegni inserendo (POKE) 2044,13.

Nella riga 25 i disegni 2 e 3 sono ampliati inserendo (POKE) 12 (disegno 2 e 3 attivati) nelle locazioni di memoria ampliate in direzione X e Y (V+23 e V+29).

La riga 48 sposta il disegno 3 lungo l'asse X. La riga 58 posiziona il disegno 3 a metà strada sullo schermo alla locazione 100. Poichè questo valore non cambia, come aveva fatto in precedenza con X da 0 a 200, il disegno 3 si muove soltanto orizzontalmente.

NOTE ULTERIORI SUI DISEGNI ANIMATI

Dopo tutto questo divertimento con i disegni e le loro possibilità di animazione, sono necessarie alcune altre spiegazioni. Innanzitutto è possibile cambiare il colore del disegno in uno qualsiasi dei 16 colori standard (da 0 a 15) che sono stati usati per cambiare il colore dei caratteri. Vedere il Capitolo 5 oppure l'Appendice G.

Per esempio, per cambiare in verde chiaro il disegno 1, battere: POKE V+40,13 (assicurarsi di porre V=53248).

Si sarà notato, usando i programmi esemplificativi, che l'oggetto non si è mai spostato verso il margine destro dello schermo e ciò in quanto lo

schermo è largo 320 punti ed il registrato della direzione X può soltanto contenere un valore di 255. Come può dunque un oggetto spostarsi sull'intero schermo?

Nella mappa di memoria c'è una locazione che non è stata finora citata. Si tratta della locazione 16 (della mappa) che controlla qualcosa chiamato «bit più significativo» (MSB) della locazione di direzione X del disegno. In effetti ciò consente di spostare il disegno in un punto orizzontale compreso tra 256 e 320.

Il MSB del registro X funziona in questo modo: dopo che il disegno è stato spostato alla locazione X 255, inserire un valore nella locazione di memoria 16 che rappresenta il disegno che si vuole spostare. Per esempio per far spostare il disegno 2 alle locazioni orizzontali 256-320, inserire (POKE) il valore del disegno 2 che è (4) nella locazione di memoria 16:

POKE V+16,4.

Iniziare ora a spostare nella direzione abituale X il registro per il disegno 2 (che è nella locazione 4 della mappa) iniziando di nuovo da 1. Dato che ci si sta spostando soltanto di altri 64 spazi, le locazioni X dovrebbero stavolta essere comprese fra 0 e 63.

L'intero concetto è meglio illustrato con una versione del programma originale del disegno 1:

```
10 V= 53248 : POKE V+21,4 : POKE 2042,13
20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q : NEXT
25 POKE V+5, 100
30 FOR X = 0 TO 255
40 POKE V+4,X
50 NEXT
60 POKE V+16,4
70 FOR X = 0 TO 63
80 POKE V+4, X
90 NEXT
100 POKE V+16,0
110 GOTO 30
```

La riga 60 definisce il bit più significativo per il disegno 2. La riga 70 inizia lo spostamento della locazione nella direzione standard X, spostando il disegno 2 del resto del percorso sullo schermo.

La riga 100 è importante in quanto «esclude» il MSB in modo che il disegno può iniziare di nuovo a spostarsi dal margine sinistro dello schermo.

ARITMETICA BINARIA

Va oltre gli scopi di questo manuale introduttivo entrare nei particolari del modo in cui il computer tratta i numeri. Forniremo comunque una buona base per la conoscenza del processo per consentire di realizzare sofisticati esempi di animazione.

Prima di procedere occorre definire alcuni termini:

BIT – Si tratta della più piccola quantità di informazione che un computer può memorizzare.
Si pensi ad un bit come ad un interruttore che può essere «acceso» o «spento» (rispettivamente on e off). Quando un BIT è «on» ha un valore di 1; quando un BIT è «off» ha un valore di 0.

Dopo il BIT il successivo livello è il BYTE.

BYTE – Il BYTE è definito come una serie di BIT. Dato che un BYTE è costituito da una serie di 8 BIT, è possibile avere in effetti un totale di 255 diverse combinazioni di BIT. In altre parole, è possibile avere tutti i BIT «off» in modo che il BYTE appaia come segue:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

Il suo valore sarà in questo caso 0. Tutti i BIT «on» appariranno come segue:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

il che equivale a dire 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.

La fase sucessiva è detta REGISTRO.

REGISTRO-E' definito come un blocco di BYTE riuniti. Ma in questo caso ciascun REGISTRO è realmente lungo soltanto 1 BYTE. Una serie di REGISTRI costituisce una MAPPA DEI REGSTRI. Le mappe dei registri sono tabelle tipo quelle osservate nella creazione del disegno del pallone. Ciascun REGISTRO controlla una diversa funzione: ad esempio quello che attiva il disegno è detto registro di abilitazione. Per rendere il disegno più lungo occorre espandere il registro X mentre per renderlo più largo occorre espandere il registro Y. Tener presente che un REGISTRO è un BYTE che esegue un compito specifico.

Passiamo ora al resto dell'aritmetica binaria.

CONVERSIONE DA BINARIO A DECIMALE

Valore decimale								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2↑0
0	0	0	0	0	0	1	0	2↑1
0	0	0	0	0	1	0	0	2↑2
0	0	0	0	1	0	0	0	2↑3
0	0	0	1	0	0	0	0	2↑4
0	0	1	0	0	0	0	0	2↑5
0	1	0	0	0	0	0	0	2↑6
1	0	0	0	0	0	0	0	2↑7

Usando la combinazione di tutti gli otto bit, è possibile ottenere qualsiasi valore decimale da 0 a 255. Si comincia ora a vedere perchè quando si inseriscono (POKE) valori di carattere o di colore nelle locazioni di memoria i valori devono essere nel campo da 0 a 255? Ciascuna locazione di memoria può contenere un byte di informazione.

Qualsiasi combinazione possibile di otto 0 e 1 si trasforma in un valore decimale unico compreso fra 0 e 255. Se tutte le posizioni contengono un 1, il valore del byte è uguale a 255. Se tutte contengono uno zero, il valore del byte è zero; «00000011» equivale a 3 e così via. Ciò costituisce la base per la creazione dei dati che rappresentano i disegni e per la loro manipolazione. A titolo di esempio, se questo raggruppamento di bit rappresentasse parte di un disegno (0 è uno spazio, 1 è un'area colorata):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	1	1	1	
128 +	64 +	32 +	16 +	8 +	4 +	2 +	1 +	= 255

Occorrerebbe inserire (POKE) 255 nell'appropriata locazione di memoria per rappresentare quella parte dell'oggetto.

SUGGERIMENTO:

Per risparmiare il fastidio di convertire numeri binari in valori decimali – e occorrerà farlo spesso – basterà usare il seguente programma. E' una buona idea immettere e salvare il programma per uso futuro.

```
5 REM BINARY TO DECIMAL CONVERTER
10 INPUT "ENTER 8-BIT BINARY NUMBER :";A$
12 IF LEN (A$) <> 8 THEN PRINT "8 BITS PLEASE...":
   GOTO 10
15 TL = 0 : C = 0
20 FOR X = 8 to 1 STEP -1 : C = C + 1
30 TL = TL + VAL(MID$(A$,C,1))*2↑(X-1)
40 NEXT X
50 PRINT A$;" BINARY ";" = ";TL;" DECIMAL"
60 GOTO 10
```

Questo programma prende il numero binario che è stato immesso come stringa e ne osserva ciascun carattere da sinistra a destra (la funzione MID$). La variabile C indica su quale carattere lavorare man mano che il programma esegue le iterazioni.

La funzione VAL nella riga 30 dà il valore effettivo del carattere. Dato che si stanno trattando caratteri numerici, il valore è lo stesso del carattere. Per esempio, se il primo carattere di A$ è 1, il valore sarà a sua volta 1.

La parte finale della riga 30 moltiplica il valore del carattere corrente per l'appropriata potenza di 2. Dato che nell'esempio il primo valore è nella posizione 2↑7, TL sarebbe la prima volta pari a 1 moltiplicato 128, ossia 128. Se il bit è o il valore per quella posizione sarebbe pure zero.

Questo processo viene ripetuto per tutti gli otto caratteri dato che TL tiene nota del valore decimale corrente del numero binario.

CAPITOLO 7

CREAZIONE DELLA MUSICA CON IL COMMODORE 64

- **Struttura di un programma sonoro**
- **Melodie con il COMMODORE 64**
- **Definizione del suono**
- **Programmi dimostrativi**

La maggior parte dei programmatori usa il suono del computer per due ragioni:

Fare della musica e generare effetti sonori. Prima di entrare nei dettagli della programmazione sonora, vediamo velocemente come è strutturato un programma sonoro . . . e includiamo anche un piccolo programma con il quale fare degli esperimenti.

STRUTTURA DI UN PROGRAMMA SONORO

Innanzitutto bisogna conoscere le 5 basi su cui generare il suono sul COMMODORE 64. Volume, controllo della forma d'onda, attack/decay, sustain/release, (ADSR) e alta/bassa frequenza. La prima (attack/decay) è responsabile per la tonalità del suono con la quale è possibile distinguere i diversi strumenti musicali. Queste importanti caratteristiche si possono modificare anche da programma.

Per questo proposito il COMMODORE 64 è equipaggiato con un integrato sonoro per l'interfacciamento (SID). Il SID contiene due registri per ottenere i parametri per sintetizzare un suono specifico. Infatti il COMMODORE 64 è capace di emettere tre suoni (voci) contemporaneamente. Vediamo la prima di queste voci.

L'indirizzo base del SID è 54272, abbreviamolo con un variabile, SI:

SI = 54272

La nota di una tonalità, o l'altezza, è data dalla sua frequenza che è determinata da un parametro registrato del SID. Questo parametro può avere un valore che và da quasi 0 a 65000. Nel capitolo precedente abbiamo visto, che numeri cosî grandi non possono essere immagazzinati su un singolo byte di memoria. Così dobbiamo separare il parametro di frequenza in due; ordine di byte basso, ordine di byte alto. Questa coppia di byte occupa i primi due registri del SID.

FL = SI **(frequenza, ordine di byte basso «low»)**
FH = SI + 1 **(frequenza, ordine di byte alto «high»)**

Il **volume** può essere settato in 16 scale diverse in un raggio compreso tra 0 (spento) a 15 (volume massimo). I parametri corrispondenti sono memorizzati nel registro 24.

L = **SI + 24** **(volume)**

Per selezionare una delle forme d'onda menzionate precedentemente «poke» uno di questi parametri: 17, 33, 65 o 129, dentro questo registro. Se si sceglie il 65 (impulso rettangolare) si deve definire un parametro addizionale per determinare il «duty cycle», che è la relazione tra «ON» e «OFF», della onda rettangolare. Entrambi i parametri sono memorizzati nei registri 2 e 3.

TL = SI + 2 (duty cycle, ordine di byte basso «low»)
TH = SI + 3 (duty cycle, ordine di byte alto «high»)

Vediamo insieme la più affascinante possibilità: La modulazione di una tonalità. Ci sono 4 diverse fasi che formano la struttura dell'impulso di un tono. Il primo è il tempo di attack. Se non si ha familiarità con questi concetti si può pensare per «attack» come la media di incremento con cui una nota raggiunge il suo volume massimo. La fase seguente è il «decay», che è la media con la quale la nota cade dal suo livello di volume più alto, al livello che noi chiamiamo «sustain» che è il terzo parametro che abbiamo spacificato. Dopo la fase sustain che è il tempo nel quale si tiene premuto il tasto, la nota torna a 0, con una media di incremento che chiamiamo «release». Così abbiamo descritto i 4 parametri che caratterizzano la struttura di una nota. Ricapitolando i parametri sono ATTACK/DECAY/SUSTAIN/RELEASE or «ADSR». Ognuno di questi parametri può essere settato in 16 scale diverse. Ogni parametro ha bisogno di 4 bits per essere specificato. Questo significa che abbiamo bisogno di 2 registri diversi per ottenere tutti e 4 i parametri. Così definiamo:

A = SI + 5 (ATTACK/DECAY)
H = SI + 6 (SUSTAIN/RELEASE)

Ognuno di questi registri è diviso in due nybbles (vedi paragrafo precedente per l'aritmetica binaria). Il parametro attack è determinato dai 4 bit più significativi del registro 5, mentre per determinare il decay sono usati i 4 bit meno significativi, dello stesso registro.
Il livello di sustain è tenuto nei 4 bit più significativi del registro 6, mentre il release è memorizzato nei meno significativi del medesimo.

Piccoli valori nella parte di attack del registro 5, causa un suono molto duro; mentre un valore grande (fino a 15) produce un suono debole. Esaminiamo tutti i parametri fin qui discussi, usando un piccolo programma dimostrativo.

PROGRAMMA DIMOSTRATIVO

Prima di tutto bisogna decidere quale voce vogliamo usare. Per ogniuna di queste dobbiamo abilitare i 4 parametri prima mensionati (volume, forma d'onda ecc.). Si possono usare fino a tre voci simultaneamente; nel nostro caso useremo soltanto la voce numero 1.

```
10 SI=54272: FL=SI:
   FH=SI+1: W=SI+4:
   A=SI+5: H=SI+6:
   L=SI+24
```
1. Definizione dei registri

```
20 POKE L,15
```
2. Volume massimo

```
30 POKE A,16+9
```
3. Attack

```
40 POKE H,4*16+4
```
4. Sustain – Release

```
50 POKE FH,29:POKE FL,69
```
5. Frequenza High e Low, per la definizione della nota a (la). Vedere l'appendice P. Per la definizione delle diverse note.

```
60 POKE W,17
```
6. Generatore di forma d'onda in posizione ON.

```
70 FORT=1T0500:NEXT
```
7. Iterazione che determina la durata della nota.

```
80 POKE W,0:POKE A,0:
   POKE H,0
```
8. Azzeramento della forma d'onda.

Dopo aver battuto «run» si può udire la nota generata da questo programma.

MELODIE CON IL COMMODORE 64

Non c'è bisogno di essere un musicista per comporre delle melodie con il COMMODORE 64. Il prossimo programma è un esempio di come si può fare. Useremo sempre una voce delle tre a disposizione.

Cancellare il programma precedente con NEW e memorizzare il seguente:

```
10 REM TONALITÀ
20 SI=54272:FL=SI:FH=SI+1:
   E=SI+4:A=SI+5:H=SI+6:
   L=SI+24
```
Nome del programma

Definizione dei registri

```
30 POKE L,15
40 POKE A,9
50 READ X:READ Y

60 IFY=-1THEN POKE W,0:END
70 POKE FH,X:POKE FL,Y

80 POKE W,17
90 FORT=1TO100:NEXT
100 POKE W,0
110 FORT=1T050:NEXT
120 GOTO40
130 DATA17,103,19,137,21,237,
    23,59,26,20,29,69,32,219,34,207
140 DATA-1,-1
```

Volume massimo

Attack

Frequenza dei Low-bit e High-bit in relazione ai dati delle linee 130 e 140

Se il programma trova -1; termina

Poke nei registri di frequenza Low-bit High-bit

Generatore di forma d'onda in posizione ON

Iterazione di ritardo

Generatore in posizione OFF

Iterazione diritardo per RELEASE

TONO seguente

Questi numeri determinano le note della scala DO-DURO

Questi dati, che non hanno significato come frequenza, segnalano alla linea 60 che il programma è finito

Se vogliamo generare un suono come quello del cembalo, dobbiamo cambiare la linea 80 nel modo seguente:

```
80 POKE W,33
```

Questo poke selezione un dente di sega come forma d'onda, che contiene molte armoniche per generare un suono tagliente; ma la sceltà della forma d'onda è soltanto un modo per cambiare un suono. Con un parametro speciale nei registri ADSR possiamo cambiare il cembalo in un banjo. Questo si ottiene con il seguente comando POKE nella linea 40:

```
40 POKE A,3
```

Possiamo anche imitare il suono di diversi strumenti musicali come un vero sintetizzatore. Come fare? Meglio dire in che modo cambiare il contenuto dei registri; cosa che vedremo adesso.

IMPORTANTE PER LA DEFINIZIONE DEL SUONO

1. VOLUME – la scelta del volume è valida per tutti e tre i generatori di suono. Il corrispondente registro si troverà all'indirizzo 54296. Il volume massimo si otterrà con una poke pari a 15

POKE L,15 oppure POKE 54296,15

Per annullare il generatore di suono basta una poke pari a 0 in questo registro.

POKE L,0 oppure POKE 54296,0

Di solito si abilita il volume all'inizio di un programma musicale; ma attraverso l'alterazione programmata di questi parametri, si possono creare effetti interessanti.

2. Forma d'onda – come abbiamo visto nell'esempio precedente, la forma d'onda influenza il carattere sonoro di una tonalità. Per ognuna di queste voci si può scegliere una forma d'onda diversa. Questa può essere triangolare, a dente di sega, rettangolare e rumore bianco. La tavola in basso contiene i diversi indirizzi dei registri e i loro contenuti. Se si vuole scegliere per la prima voce, la forma d'onda triangolare, eseguire i seguenti comandi:

POKE W,17 oppure POKE 54276,17

Il primo numero (indirizzo) indica il registro, mentre il secondo numero (contenuti dell'indirizzo del registro) controlla la forma d'onda.

DEFINIZIONE DELLA FORMA D'ONDA

REGISTRI				CONTENUTI			
VOCE	1	2	3	RUMORE	RETTANGOLARE	DENTE DI SEGA	TRIANGOLARE
	4	11	18	129	65	33	17

Questa è la tavola usata nella linea 30 del programma «scala musicale». Con POKE SI+4,17 si è scelta la forma d'onda triangolare, che si è trasformata in dente di sega cambiando il 17 con il 33. Vediamo adesso come può cambiare la struttura che determina la media di incremento del volume, in una tonalità. Da ricordare che questa nota musicale si ottiene soltanto avendo predisposto sia il volume che la forma d'onda.

3. ADSR – il valore per ATTACK e DECAY, come per la forma d'onda, può essere scelto separatamente per ogni voce ed è rappresentato da un numero. ATTACK è la media di incremento con cui la nota raggiunge il suo volume massimo. DECAY determina il valore del tempo con cui la nota cade dal suo volume massimo al livello di SUSTAIN programmato. Se si seleziona 0 come livello di SUSTAIN, il parametro DECAY determina il tempo del livello di volume 0; cosichè è identico alla durata della nota. Il valore degli indirizzi per la differenti voci e la predisposizione di AD può essere vista nella seguente tavola. I valori per ATTACK e DECAY sono semplicemente addizionati e poked nei registri prestabiliti.

ATTACK-DECAY

REGISTRI				CONTENUTI	
VOCE	1	2	3	ATTACK	DECAY
	5	12	19	15•16 (Debole) ... 0•16 (Duro)	15 (Debole) ... 0 (Duro)

Se si seleziona soltanto un tempo di ATTACK con una poke 54277,64, il tempo di DECAY si setta automaticamente a 0, e viceversa. Con poke 54277,66 si setta il valore medio di ATTACK (64=4•16) e DECAY ad un valore piccolo (2); quindi 66 è il risultato della somma 64+2. Il miglior modo per riconoscere un parametro così composito è di scrivere POKE A,4•16+2, invece di POKE 54277,66 (l'indirizzo del registro A deve essere specificato prima). Abbiamo quindi raggiunto il punto dove possiamo provare le cose fin qui descritte con un piccolo programma. Come al solito battiamo NEW, spingiamo return, e inseriamo il seguente programma:

```
10 REM PROGRAMMA SPERIMENTALE
20 SI=54272: FL=SI: FH=SI+1: TL=SI+2:
   TH= SI+3: W=SI+4: A=SI+5: H=SI+6:
   L= SI+24
30 PRINT"PREMI UN TASTO"                  Commento video
40 GETZ$:IFZ$=""THEN40                    Chiave premuta?
50 POKE L,15                              Volume
60 POKE A,1•16+5                          Attack-Decay
70 POKE H,0•16+0                          Sustain-Release
80 POKE TH,8: POKE TL,0                   Duty cicle
90 POKE FH,14: POKE FL,162                Frequenza
100 POKE W,17                             Generatore, forma d'onda ON
110 FORT=1TO200:NEXT                      Iterazione
120 POKE W,0                              Generatore OFF
130 GOTO40                                Premi un tasto
```

Abbiamo usato la voce 1 per creare il tono con un corto ATTACK e un corto DECAY, dopo aver raggiunto il massimo volume (linea 60). L'effetto sonoro ricavato è un suono metallico. Per creare un suono diverso è sufficente cambiare questa riga. Per far questo premere il tasto RUN/STOP e listare il programma. Dopo averlo visualizzato sullo schermo cambiare la linea 60 nel seguente modo:

```
60 POKE A,11•16+14
```

Premendo return il computer memorizza la nuova riga nel programma in memoria.

Il tono, che si otterrà con questa predisposizione, è come quelle di un oboe o di un flauto. Facciamo alcune prove e cambiamenti alla forma d'onda e all'ADRS, per renderci conto di come le diverse predisposizioni di questi parametri influenzano il carattere di un tono.

Usando il parametro SUSTAIN si può determinare il volume di un tono, dopo la fase di ATTACK/DECAY. Di solito la durata di un tono è determinata usando il loop FOR . . . NEXT. Come nel registro precedente, SUSTAIN e RELEASE sono determinati dallo stesso registro; quindi non si deve fare altro che addizionare il valore di entrambi i parametri e poke la somma di questi, nel registro corrispondente. Vedi tabella in basso.

SUSTAIN – RELEASE

REGISTRI				CONTENUTI	
VOCE	1	2	3	SUSTAIN	RELEASE
	6	13	20	15•16 (alto) ... 0•16 (basso)	15 (lento) ... 0 (veloce)

Cambiando il valore degli 0 nelle righe di programma precedente, tra i valori 0 - 15, si può udire la differenza.

4. Selezione di voci e note – come visto precedentemente si devono specificare i 2 valori, per determinare la frequenza o la nota di un singolo tono. Chiameremo questi valori, ordine alto di byte e ordine basso di byte, della frequenza. L'assegnazione del nome della nota per il valore della frequenza è riportata nella tabella dell'appendice P.

Siccome le voci sono relegate a indirizzi diversi, con il COMMODORE 64 si possono programmare le 3 voci indipendentemente; in questo modo si possono creare cansoni a tre voci o corde.

INDIRIZZI DEL GENERATORE DI NOTA E POKE PER HIGH BYTE – LOW BYTE PER NOTE IN 5. OTTAVA

REGISTRI				CONTENUTI PER LE NOTE IN 5. OTTAVA											
VOCE	1	2	3	C	C#	D	D#	E	F	F#	G	G#	A#	H#	H	C
HI-BYTE	1	8	15	35	37	39	41	44	46	49	52	55	58	62	66	70
LO-BYTE	0	7	14	3	24	77	163	29	188	132	117	148	226	98	24	6

Per generare una C (DO) con la VOCE 1 POKE le seguenti istruzioni:

POKE 54273,35: POKE 54272,3 oppure **POKE SI+1,35: POKE SI,3**

Ugualmente la VOCE 2 ai ottiene con:

POKE 54280,35: POKE 54279,3 oppure **POKE SI+8,35: POKE SI+7,3**

ESECUZIONE DI UN MOTIVO SUL COMMODORE 64

Il seguente programma può essere usato per comporre o suonare un motivo (usando VOCE 1). Ci sono due importanti lezioni in questo programma: innanzitutto come abbreviare tutti i lunghi numeri di controllo nella prima riga del programma . . . dopodichè è possibile usare la lettera W per «forma d'onda» invece del numero 54276.

La seconda lezione riguarda il modo in cui si usano i dati. Questo programma è scritto in modo da consentire di immettere tre numeri per ciascuna nota: il VALORE DELLA NOTA IN HIGH FREQUENCY, il VALORE DELLA NOTA IN LOW FREQUENCY e la DURATA DELLA NOTA CHE VERRA' SUONATA.

Per questo motivo si userà una durata di 125 per una croma, 150 per una semiminima, 375 per una semiminima puntata, 500 per una minima e 1000 per una semibreve. Questi valori possono essere aumentati o diminuiti per adeguarli ad un particolare tempo o al proprio gusto musicale.

Per vedere come viene immesso un motivo, osservare la riga 100. Sono stati immessi 34 e 75 come regolazioni di HIGH e LOW FREQUENCY per suonare un «Do» (dalla scala campione indicata precedentemente) e quindi il numero 250 per la semiminima. Così la prima nota del nostro motivo è una nota Do semiminima. Anche la seconda nota è una semiminima, ma stavolta è un Mi . . . e così via fino alla fine del motivo. E' possibile immettere qualsiasi altro motivo in questo modo, aggiungendo tante righe di istruzioni DATA quante ne occorrono. E' possibile continuare i numeri delle note e di durata da una riga alla successiva ma ciascuna riga deve cominciare con la parola DATA. DATA–1,–1,–1 deve essere l'ultima riga nel programma. Questa riga «termina» il motivo.

Battere la parola NEW per cancellare il precedente programma e battere il programma seguente, quindi battere RUN per ascoltare il motivo.

MICHAEL ROW THE BOAT ASHORE–1 MISURA

```
5 V=54296:W=54276:A=54277:H=54273:LF=54272:S=54278:PH
  =54275:PL=54274
10 POKEV,15:POKEW,65:POKEA,190:POKEH,15:POKEL,15
20 READH
30 READL
40 READD
50 IFH=–1THENEND
60 POKEHF,H:POKELF,L
70 FORX=D–50TOD–20:POKES,136:NEXT
80 FORT=ITOD:NEXT:POKEHF,0:POKELF,0:POKEW,0
```

```
 90 GOTO10
100 DATA34,75,250,43,52,250,51,97,375,43,52,125,51,97
105 DATA250,57,172,250
110 DATA51,97,500,0,0,125,43,52,250,51,97,250,57,172
115 DATA1000,51,97,500
120 DATA-1,-1,-1
```

CREAZIONE DI EFFETTI SONORI

A differenza della musica, gli effetti sonori sono spesso legati ad una specifica «azione» di programmazione ad esempio l'esplosione creata da un combattente spaziale quando penetra attraverso una barriera nel gioco «Guerre spaziali» . . . o la cicalina in un programma gestionale che dice all'utente che sta per cancellare per errore il suo disco.

Sono disponibili numerose opzioni se si vogliono creare diversi effetti sonori. Ecco 10 idee di programmazione che potrebbero aiutare ad iniziare la sperimentazione con gli effetti sonori:

1. Cambio del volume mentre viene eseguita la nota, ad esempio per creare un effetto d'eco.
2. Alternanza rapida tra due note per creare un effetto «tremolo».
3. Forma d'onda . . . provare diverse regolazioni per ciascuna voce.
4. Attack/decay . . . per alterare la velocità di salita e di caduta di un suono.
5. Sustain/release . . . per prolungare o smorzare il volume di un effetto sonoro oppure per combinare una serie di suoni. Occorre provare diverse regolazioni.
6. Effetti a più voci . . . cioè l'esecuzione di più di una voce contemporaneamente, con ciascuna voce voce controllata indipendentemente o cun una voce che suona più o meno a lungo di un'altra o che serve da «eco» o risposta ad una prima nota.
7. Cambio delle note sulla scala o cambio dell'ottava usando i valori nella tabella VALORI DELLE NOTE MUSICALI.
8. Uso della forma d'onda quadra e di diverse regolazioni di impulsi per creare effetti diversi.
9. Uso della forma d'onda di rumore per generare «rumore bianco» per accentuare gli effetti sonori tonali o per creare esplosioni, colpi di cannone o rumore di passi. Le stesse note musicali che creano la musica possono anche essere usate con la forma d'onda di rumore per creare diversi tipi di rumore bianco.