DAI

MANUALE DEL MICROCOMPUTER

di Rita Bonelli e Claudio Fiorentini

GRUPPO EDITORIALE JACKSON

DAI

MANUALE DEL MICROCOMPUTER

Traduzione dall'originale inglese
"DAI PERSONAL COMPUTER REFERENCE MANUAL"
a cura di:
Rita Bonelli e Claudio Fiorentini
per gentile concessione della:
Data Application International - Bruxelles

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura del volume le signore Francesca di Fiore, Rosi Bozzolo e l'ing. Roberto Pancaldi.

Prima edizione: 1981

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

P R E F A Z I O N E

Questo manuale e' diviso in due parti. Lo scopo della prima parte e ' quello di mettervi in grado di usare il vostro nuovo calcolatore, subito dopo averlo installato, seguendo le istruzioni contenute nelle pagine che seguono.

Questa parte del manuale e' stata preparata per i principianti, cioe' per coloro che non conoscono il linguaggio di programmazione BASIC (Beginners All-purpose Symbolic Instruction Code, cioe': Linguaggio di programmazione per principianti di uso generalizzato), e serve sia per guidare i loro primi passi nella programmazione BASIC che per spiegare loro le caratteristiche peculiari del Personal DAI. Tali caratteristiche, essendo disponibili solo sul DAI, devono essere studiate anche da un programmatore gia' esperto.

Dovete pero' tener presente che lo scopo di questo manuale non e' quello di fornire un corso completo di programmazione in BASIC.

Gli autori sperano che, dopo aver lavorato servendovi di questo manuale e dopo esservi fatta una prima idea di quello che puo' fare il vostro calcolatore, se ben programmato, siate stimolati ad uno studio approfondito dell'argomento.

Esistono alcuni libri dedicati allo studio completo dell'argomento e di essi questo manuale non vuole assolutamente essere considerato un possibile sostituto.

La seconda parte di questo manuale contiene le specifiche sull'implementazione del linguaggio BASIC sul calcolatore DAI. Durante il vostro lavoro di programmazione vi sara' utile ricorrere spesso a questa seconda parte.

Scrivere un manuale adatto ad una gamma molto vasta di utenti non e' compito facile. Si corre il rischio di scontentare tutti, cercando di piacere a ciascuno.

Vi preghiamo pertanto di volerci scusare se in taluni casi vi sembreremo troppo pedanti ed in altri troppo superficiali. Vi preghiamo inoltre di volerci gentilmente segnalare tutte le modifiche che pensate possano servire a migliorare il nostro manuale.

S O M M A R I O

CAPITOLO 0

I N T R O D U Z I O N E A L M I C R O C O M P U T E R D A I

0.1. IL MOMENTO DELICATO DELL'INSTALLAZIONE DEL VOSTRO DAI

La prima cosa da fare, quando siete arrivati a casa con il vostro nuovo calcolatore, e' di trovare un luogo tranquillo vicino ad una presa di corrente e, possibilmente, con un tavolo disponibile. Anche se il vostro DAI puo' lavorare ugualmente bene dopo averlo installato sul pavimento della cucina, questa soluzione puo' essere poco confortevole per voi!

Nella stessa scatola dove avete gia' trovato questo manuale, troverete anche il calcolatore (questa specie di scatola bianca cosi' interessante che assomiglia ad una macchina da scrivere), e tre cavi equipaggiati con spinotti.

Collegate il cavo coassiale all'uscita VIDEO, posta sul pannello posteriore del DAI, e all'entrata dell'antenna della televisione, sempre che voi desideriate usare una unita' di uscita VIDEO (VDU). Quest'ultima puo' essere un qualunque modello, in bianco e nero o a colori di TV, in grado di ricevere UHF. Vi raccomandiamo, comunque, di fare uso di una televisione a colori, onde poter utilizzare una delle piu' interessanti risorse del DAI, e cioe' la GRAFICA A COLORI.

Collegate ora il cavo nero al foro marcato 220 (avendo prima controllato che il selettore di voltaggio sia posizionato su 220) che si trova sul pannello posteriore del DAI ed alla presa di corrente.

Il terzo cavo e' l'interfaccia per il registratore a nastro (cavo di collegamento per calcolatori). Dovete semplicemente collegare questo cavo al foro marcato CASS-1 del calcolatore e rispettivamente ai fori marcati MICRO e EAR del registratore a nastro. Se non avete in casa un registratore a nastro non vi preoccupate, potete ugualmente procedere nello studio dell'intero manuale ed imparare ad usare il vostro calcolatore. Il registratore serve se volete memorizzare dei programmi su nastro (SAVE), e se volete ricaricare in memoria dei programmi gia' memorizzati su nastro (LOAD). Potrete in seguito acquistare un registratore, anche di basso costo, ma cercatene, possibilmente, uno che abbia il contagiri. Il contagiri e' infatti molto utile per poter localizzare i diversi contenuti di un nastro.

0.2. ACCENSIONE DEL CALCOLATORE

A questo punto supponiamo che voi abbiate eseguito correttamente tutti i collegamenti e siate seduti di fronte al vostro DAI.

La cosa da fare ora e' quella di accendere il televisore ed aspettare che si scaldi. Ora accendete il DAI (interrutore rosso sul pennello posto sul retro) e verificate che l'interrutore luminoso posto sul retro e la piccola lampadina verde posta a destra sulla tastiera siano ambedue accese. Infine dovete sintonizzare il televisore in UHF sul canale 36 onde poter ricevere i segnali dal DAI.

Quando quest'ultima operazione e' terminata, voi vedrete, su uno sfondo verde,in lettere maiuscole grandi bianche centrate nella meta' superiore dello schermo la scritta:

```
DAI PERSONAL
COMPUTER
```

Se , invece, vedete comparire su sfondo grigio, nell'angolo in alto a sinistra la scritta:

```
BASIC  V1.1
*-
```

questo significa che avete inavvertitamente premuto un qualunque tasto del DAI dopo averlo acceso e prima di aver sintonizzato il televisore. In questo caso se desiderate vedere comparire la scritta bianca grande in campo verde e volete seguire le istruzioni passo passo, dovete spegnere il calcolatore e riaccenderlo nuovamente.

Ecco, ora lo schermo e' verde e mostra la scritta bianca!

Se ora premete un qualunque tasto, vedrete apparire:

```
BASIC V1.1
*-
```

In alcuni televisori sara' necessario aggiustare la sintonizzazione orizzontale e verticale fino ad avere una buona visione.

0.3. IL CALCOLATORE E' ORA PRONTO

Il vostro calcolatore e' ora pronto per accettare comandi in linguaggio BASIC.

Chiamiamo l'asterisco, che compare sotto il B di BASIC, il carattere di PROMPT, dal momento che il calcolatore lo fa apparire all'inizio di una nuova linea sullo schermo ogni volta che vuole avvisare l'utente che e' pronto ad accettare comandi. Notate che subito dopo l'asterisco (PROMPT) c'e' una lineetta lampeggiante. Questo simbolo viene chiamato CURSOR e mostra dove apparira' il prossimo carattere che voi

scriverete sulla tastiera.
Ogni qualvolta voi vediate apparire sullo schermo dopo tutte le altre linee il carattere PROMPT seguito dal CURSOR (non necessariamente alla sommita' dello schermo), questo significa che il calcolatore e' nello stato CONTROL, cioe' non sta eseguendo un programma, ma e' in attesa che voi gli diate un comando.

Prima di procedere vi preghiamo di prendere nota di quanto segue; e' una cosa MOLTO IMPORTANTE.
Voi non potete danneggiare in alcun modo il vostro calcolatore scrivendo con la tastiera a meno che non usiate un martello e tentiate di asportare i tasti; in tale modo danneggereste in modo permanente il vostro calcolatore!
Pertanto voi potete fare ogni altro tentativo e vedere come il calcolatore risponde. In molti casi, se scrivete caratteri a caso e poi usate il tasto RETURN, vedrete apparire la scritta:

SYNTAX ERROR

questo significa che il calcolatore e' disposto a parlare con voi solo in BASIC e non in altri linguaggi.
Questo manuale ha lo scopo di insegnarvi i primi elementi del linguaggio che parla il vostro calcolatore. Come nello studio di ogni altro linguaggio, umano o non umano, quello che conta di piu' e' la pratica.
Noi vi suggeriamo di provare tutti gli esempi del manuale e di inventarne voi altri. Non preoccupatevi se otterete dei messaggi di errore: il programmatore che non commette errori non e' ancora nato!
Per rendere piu' evidenti gli esempi da provare, essi sono sempre scritti all'inizio di una linea. Voi dovete scrivere gli esempi esattamente come sono riportati sul manuale, mantenendo anche gli spazi indicati. Molto spesso gli SPAZI hanno un ruolo molto importante nella sintassi del linguaggio BASIC. Se vedete apparire SYNTAX ERROR, per prima cosa controllate con il manuale quello che avete scritto sul video.

0.4. INTRODUZIONE ALLA TASTIERA DEL DAI

Per poter comunicare con il vostro DAI, dovete prendere confidenza con la tastiera. Essa e' molto simile alle tastiere delle macchine da scrivere, ma dispone anche di alcuni tasti che sono completamente nuovi, anche per chi ha molta esperienza di dattilografia, e di alcuni tasti che non servono allo scopo che un esperto si aspetta.
Se voi non avete mai usato una macchina da scrivere il vostro svantaggio e' molto minore di quello che pensate. Innanzitutto la programmazione non e' un qualche cosa che

puo' essere realizzato scrivendo molto velocemente. Inoltre una persona esperta di macchine da scrivere potra' trovarsi a disagio nei primi contatti con una tastiera di calcolatore a causa di alcune abitudini che ha acquisito, come usare la lettera "l" minuscola al posto della cifra "1", oppure la lettera "O" maiuscola invece della cifra "zero", che causano errori, lavorando con il calcolatore.

Segue, per comodita' di tutti, un elenco dei principali tasti che, di norma, non si trovano su una macchina da scrivere:

1) I 4 tasti grigi per il controllo del CURSORE situati nell'angolo in alto a sinistra della tastiera;
2) Il tasto CTRL situato sopra il tasto SHIFT di sinistra;
3) Il tasto RETURN situato a destra sopra il CHAR DEL;
4) Il tasto CHAR DEL situato sopra il tasto SHIFT di destra;
5) Il tasto BREAK situato nell'angolo in alto a destra della tastiera;
6) Il tasto REPT situato a destra del tasto CHAR DEL.

Inoltre i tasti sotto elencati servono a svolgere una specifica funzione, oltre che a stampare il carattere evidenziato:

* e' il simbolo della moltiplicazione
/ e' il simbolo della divisione
< ha il significato di MINORE DI
> ha il significato di MAGGIORE DI
(freccia verso l'alto) significa ELEVATO ALLA POTENZA DI.

Inoltre dovete tener presente che, per distinguere lo ZERO dalla lettera O maiuscola, lo ZERO viene rappresentato tagliandolo con una barra.

Vediamo ora a cosa servono i diversi tasti.

1) Tasti di controllo del CURSORE. Le funzioni principali di questi tasti sono inerenti al modo EDIT, che verra' spiegato piu' avanti. Il tasto FRECCIA A SINISTRA e' anche usato dai programmi UTILITY. Essi di norma non hanno altre funzioni, a meno che voi non desideriate che essi svolgano, nel vostro programma, una specifica funzione e provvediate ad assegnarla (per es.: in un gioco desiderate comandare la direzione di movimento di un oggetto).

2) Tasto CTRL. Vi permette di selezionare lo stato di maiuscolo o minuscolo per i caratteri alfabetici che premete dopo di esso. Quando voi accendete il DAI, o dopo la pressione del tasto RESET (punto 5, piu' avanti), tutto quello che viene scritto appare in caratteri maiuscoli, dal momento che i programmi BASIC devono essere scritti in caratteri maiuscoli.

Pero', quando scrivete un programma, voi potete desiderare

che alcune linee o alcune pagine del testo, che servono come spiegazioni, siano scritte in caratteri minuscoli. Inoltre voi potete desiderare che i vostri messaggi appaiano in caratteri minuscoli, e il DAI ve lo consente.

Ci sono due procedure per ottenere caratteri minuscoli sullo schermo:

a) premere i tasti desiderati tenendo premuto uno dei due tasti SHIFT;
b) premere una volta il tasto CTRL e poi usare i tasti desiderati.

Il modo piu' consigliabile e' operare cosi': usare l'opzione b) che, dopo avere premuto una volta il tasto CTRL, fa operare la tastiera come quella di una normale macchina da scrivere e poi usare uno dei due tasti SHIFT per ottenere le maiuscole desiderate. Per ritornare allo stato normale BASIC basta premere ancora una sola volta il tasto CTRL.

Tenete presente che la pressione del tasto CTRL non ha effetto sui tasti non alfabetici; per essi valgono le normali regole, e cioe' che per ottenere il carattere rappresentato nella parte superiore del tasto si deve tenere premuto lo SHIFT. Fate qualche esercizio per rendervi ben conto di quanto spiegato sopra.

Quando avrete riempito lo schermo, usando tutte le 24 linee che esso puo' contenere, vedrete che il DAI vi consente di aggiungere altre linee, spostando verso l'alto il contenuto dello schermo (SCROLLING) di una linea per volta.

Se desiderate ripulire completamente lo schermo operate cosi':

-premete il tasto RETURN per andare a capo
-scrivete ?CHR$(12)
-premete il tasto RETURN.

Vedrete che lo schermo si ripulisce, e, per il momento, accontentatevi di saper ripulire lo schermo, senza sapere perche' e come questo avviene. Cercate di fare pratica anche su questa operazione.

3) Tasto RETURN. Abbiamo spesso pensato che, per ottenere una migliore descrizione di questo tasto, sarebbe meglio chiamarlo TASTO DEL NON RITORNO. Infatti, una volta premuto il RETURN non esiste un modo per annullare il comando appena trasmesso al calcolatore. Vedrete presto che vi troverete spesso nella condizione di desiderare di non aver premuto questo RETURN!

Quando voi scrivete qualcosa nel calcolatore per mezzo della tastiera e' necessario poter segnalare che l'operazione di scrittura e' terminata. Questo si ottiene

usando il tasto RETURN. Prima di premere RETURN potete modificare quanto avete scritto, ma dopo no.

Vi occorrera' un po' di tempo prima di essere sicuri di usare correttamente il tasto RETURN per fare accettare un comando al calcolatore o per terminare una linea di programma. Noi nel manuale vi ricorderemo di farlo.

4) Tasto CHAR DEL. Se vi accorgete di avere scritto dei caratteri errati potete cancellarli uno per uno premendo il tasto CHAR DEL. Se usate ripetutamente questo tasto potete cancellare anche tutta la linea sulla quale vi trovate. Naturalmente non dovete avere premuto gia' il tasto RETURN, perche' in tale caso avete abbandonato la linea sulla quale vi trovavate.

Il modo EDIT, che vedremo piu' avanti, vi consente di correggere in modo molto piu' efficiente e veloce quanto compare sullo schermo.

5) Tasto BREAK. Quando voi accendete il calcolatore ed esso non sta eseguendo un programma (salvo, naturalmente, il programma contenuto in ROM (memoria a sola lettura - Read Only Memory) che permette al microprocessore del sistema di accettare quello che voi scrivete, di interpretarlo e di mettervi in grado di programmare in BASIC), il sistema si trova in uno stato che chiamiamo CONTROL. Quando voi introducete un programma e date al calcolatore il comando di eseguirlo (RUN), cioe' di eseguire le istruzioni contenute nel programma, lo stato del sistema cambia ed esso si trova nello stato PROGRAMMA. Quando il sistema e' nello stato PROGRAMMA voi non potete piu' intervenire dalla tastiera con comandi estemporanei fino a quando l'esecuzione del programma non e' terminata ed il sistema e' ritornato nello stato CONTROL. Esiste, pero', un modo per far tornare in qualsiasi momento il sistema nello stato CONTROL: basta premere il tasto BREAK.

In alcuni casi, pero', il calcolatore e' cosi' occupato che non si accorge che voi avete premuto il tasto BREAK e quindi non si ferma. Quando questo capita, ed ora e' troppo presto per comprenderne la ragione, le uniche cose che si possono fare per far tornare il sistema nello stato CONTROL sono:

-premere il tasto RESET, che si trova alla sinistra della tastiera
-spegnere il calcolatore e riaccenderlo.

Tenete ben presente, che, se ricorrete a queste procedure di emergenza, perdete tutto il contenuto della memoria, e magari avevate impiegato alcune ore per scrivere in memoria un programma! Piu' avanti imparerete come comportarvi in questi casi.

Comunque e' meglio darvi subito questi preziosi consigli:

-quando scrivete un programma in memoria, provvedete, ad intervalli regolari, a memorizzarlo su nastro, anche se non e' ancora completo;
-quando avete finito di scrivere il programma, memorizzatelo tutto di nuovo su nastro prima di provarlo;
-ogni volta, prima di memorizzare, provvedete a riavvolgere il nastro, cosi' avrete sempre all'inizio del nastro l'ultima versione del programma.

Se seguirete queste regole, anche in caso di caduta di tensione, non sarete costretti a rifare un lungo lavoro. Basta semplicemente ricaricare in memoria (LOAD) il contenuto del nastro e continuare.

L'operazione di salvataggio su nastro si chiama SAVE. Per il momento non possiamo entrare nel dettaglio delle operazioni SAVE e LOAD, che riguardano il salvataggio ed il richiamo dei programmi, dato che voi ancora non sapete veramente che cosa e' un programma.

6) Tasto REPT. Esso serve a ripetere il simbolo o la funzione di un altro tasto se viene tenuto premuto contemporaneamente. Provate ad usare questo tasto.

0.5. INCOMINCIATE A CONOSCERE IL VOSTRO CALCOLATORE

Ora voi avete una idea approssimativa di come usare la tastiera e potete cominciare ad usare il vostro calcolatore.

Voi avete comprato un calcolatore che puo' generare disegni a e testi a colori usando il vostro televisore di casa. Per ora voi vi trovate davanti ad uno schermo grigio (e questo e' tutto quello che voi potete ottenere se non disponete di un televisore a colori).

Qui viene spiegato come ottenere i colori sullo schermo. Scrivete i seguenti comandi BASIC (ricordate che tutti i comandi devono essere scritti usando le lettere maiuscole):

COLORT 5 0 0 0 e poi premete RETURN.

Vedrete che il testo appare bianco su fondo verde. Questo succede se non appare la scritta SYNTAX ERROR, a significare che avete fatto un errore di scrittura nel comando come, per esempio, non lasciare lo spazio tra i numeri. in tale caso riprovate.

Se preferite un'altra combinazione testo/sfondo , potete provare a selezionare le varie possibilita' fino ad ottenere la vostra combinazione preferita. Ricordate che il primo numero che segue la parola COLORT (5 nell'esempio) seleziona uno dei 16 colori disponibili per lo sfondo, mentre la seconda cifra (0 nell'esempio), che deve essere separata da uno spazio dal primo numero, determina il colore del testo. Ricordate che il calcolatore considera lo zero (0) come un

normale numero e le 16 possibilita' per lo sfondo si ottengono numerandole da 0 a 15.

I due ultimi numeri, sempre separati da uno spazio, (0 0 nell'esempio) devono sempre essere presenti, anche se apparentemente non corrispondono ad una funzione.Questo e' necessario per rendere la sintassi del comando COLORT uguale a quella del comando COLORG, che verra' spiegato piu' avanti.

Vi chiederete se e' semplice o complicato cambiare i colori sul vostro televisore. Noi pensiamo che, dal momento che avete comprato un calcolatore, voi desideriate che esso svolga per voi anche i lavori piu' semplici, specialmente se sono noiosi, come provare tutte le possibili combinazioni testo/sfondo, che sono 256, scrivendo ogni volta:

```
COLORT N1 N2 0 0   RETURN
```

e facendo assumere rispettivamente ad N1 e ad N2 tutti i valori possibili.

Non e' meglio far fare questo lavoro al calcolatore?

Dal momento che questo e' possibile, esso sara' il vostro primo esercizio di programmazione. Ora cercate di resistere alla tentazione di saltare subito alla pagina dove questo programma e' listato e seguite passo passo la nascita del programma. In tale modo comincerete a conoscere alcune tra le caratteristiche del linguaggio BASIC e del calcolatore DAI messe in evidenza anche da un programma cosi' semplice.

0.6. DOVE FINALMENTE INCOMINCIATE A PROGRAMMARE IL VOSTRO CALCOLATORE

La prima cosa che dovete sapere riguardo alla stesura di un programma in BASIC e' che, quando scrivete RUN e premete il tasto RETURN, il calcolatore esegue una serie di comandi che voi (o un altro programmatore) avete memorizzato nella sua memoria. L'ordine di esecuzione di ciascun comando e' determinato dal numero che precede il comando stesso. Questo numero viene chiamato NUMERO DI LINEA.

Ogni volta che voi scrivete un comando sulla tastiera, senza farlo precedere dal numero di linea, esso viene eseguito non appena azionate il tasto RETURN, e di questo comando non rimane traccia in memoria (si dice che si lavora in modo immediato). Se voi, invece, fate precedere il comando dal numero di linea (il NUMERO DI LINEA puo' essere un qualunque numero intero compreso tra 1 e 65535), esso viene memorizzato quando premete il tasto RETURN, e verra' eseguito solo quando scriverete RUN e poi RETURN.

Ora per iniziare il vostro lavoro dovete scrivere:

NEW e poi RETURN

e questo serve per dire al calcolatore che state incominciando a scrivere un nuovo programma. Poi scrivete sulla prossima linea:

100 COLORT 10 5 0 0 e poi RETURN

Vi sembrera' che non sia successo alcunche'. Sbagliate; voi avete scritto il vostro primo programma sul DAI.
Ora scrivete:

LIST e poi RETURN

e il calcolatore listera' il programma che si trova in memoria. In questo caso:

100 COLORT 10 5 0 0

che e' l'unica linea di programma scritta da voi. E' sempre utile chiedere la lista di un programma con il comando LIST, prima di eseguirlo.
Ora scrivete:

RUN e poi RETURN

e fate attenzione a quello che succede. Vedrete che il testo diventa verde su sfondo arancione: il vostro DAI ha eseguito il comando COLORT contenuto nella linea 100 del programma.
Incoraggiati da questo primo successo, procediamo per scrivere un programma che ci mostri in sequenza tutte le possibili combinazioni testo/sfondo.
Avrete notato che abbiamo attribuito il numero di linea 100 alla prima linea del vostro programma. Perche' non 1? Una delle piu' piacevoli caratteristiche del linguaggio BASIC e' che non tiene conto dell'ordine con il quale voi introducete le linee di programma, ma le memorizza in ordine di numero di linea crescente. Cosi', se dopo aver scritto la linea 100 sopra vista, noi decidiamo che il DAI deve compiere un'altra operazione prima di cambiare i colori dello schermo, bastera' scrivere questa o queste istruzioni attribuendo loro numeri di linea minori di 100. Se noi avessimo scritto la precedente istruzione usando il numero di linea 1 invece di 100, dato che non si puo' usare il numero di linea 0, saremmo stati costretti a riscrivere tutto di nuovo.
E' quindi una buona abitudine, programmando in BASIC, iniziare la numerazione delle linee da un numero ragionevolmente alto, come 100, ed incrementare tale numero di 10 in 10 per le linee seguenti.
Fino ad ora il calcolatore non ci ha risparmiato del lavoro, e, per ottenere sullo schermo i colori verde e arancio, abbiamo lavorato di piu' di quanto non avremmo fatto dando il singolo comando COLORT senza attribuirgli un

numero di linea. Questo e' vero, ma abbiate fiducia e continuate a leggere.

0.7.VARIABILI

Quello che ora dovete imparare, per poter continuare a scrivere il nostro programma, e' l'uso delle variabili. Anche se non e' un' idea molto originale, vi invitiamo a immaginare le variabili come dei nomi per delle caselline (contenitori) nelle quali voi potete conservare dei valori che servono in diversi punti al vostro programma. Per creare una di queste caselline basta inventare un nome che la distingua dalle altre, ed assegnare un valore alla variabile, cioe' riempire la casellina con un valore.

Nel linguaggio della programmazione questi valori si chiamano dati; quindi possiamo dire che le variabili servono a conservare dei dati (contengono dati).

Per esempio, la frase BASIC:

```
LET A = 1
```

serve a memorizzare (conservare) il numero 1 nella casellina di nome A. La frase appena citata va interpretata cosi': "fai in modo che A contenga il numero 1" per chiarire che siete voi che prendete la decisione che nel vostro programma cominci ad esistere una variabile di nome A e che volete contenga il numero 1.

Il nome di una variabile puo' essere formato usando uno o piu' caratteri, ma non deve superare in lunghezza una linea di programma; il primo carattere del nome deve essere alfabetico, mentre i seguenti possono essere sia alfabetici che numerici. Il BASIC del DAI riconosce solo i primi 14 caratteri del nome di una variabile, quindi e'necessario non scrivere mai due variabili diverse con i primi 14 caratteri uguali. E' comodo per il programmatore usare nomi di variabili che abbiamo implicitamente un significato nella lettura del programma, cioe' che siano mnemonici, comunque e' bene non esagerare nella scelta della lunghezza dei nomi delle variabili.Per esempio, nel nostro programma, noi avremo bisogno di variabili per memorizzare i numeri che identificano i colori dello sfondo e del testo dello schermo. Tali variabili devono essere usate nella frase comando COLORT. Noi possiamo decidere di chiamare queste variabili :

```
S per lo sfondo,
T per il testo;
```

ma possiamo, con maggiore evidente chiarezza usare i nomi:

SFONDO e TESTO.

Se voi scrivete:

LET TESTO = 5 e premete il RETURN

succedono due cose:

1) In qualche posto della memoria del DAI viene riservata una casellina di nome TESTO;

2) In questa casellina viene conservato il valore 5.

Come sempre, non ci aspettiamo di essere creduti sulla parola. Controllate voi stessi se quanto detto e' vero, dopo aver letto il prossimo paragrafo.

0.8. IL COMANDO PRINT

Come fate se volete rivolgere una domanda? Naturalmente usate il punto interrogativo. Lo dovete fare anche ora per chiedere qualcosa al calcolatore, solo che dovete scrivere il punto interrogativo per prima cosa, cosi':

? TESTO e premere il RETURN

Il punto interrogativo e' equivalente alla parola PRINT, ne e' una comoda abbreviazione. Voi volete che il calcolatore stampi (PRINT) sullo schermo il valore di una variabile che sta nella sua memoria. Voi vedrete apparire sullo schermo:

5.0

e questo e' proprio il valore che avevate conservato nella variabile TESTO.

Se volete verificare che usando la parola PRINT invece del ? si ottiene lo stesso risultato, provate.

Volete sapere perche' avete ottenuto 5.0 invece di 5? E'presto detto: tutte le variabili, se non specificato altrimenti, contengono numeri decimali (reali o floating-point). Se noi desideriamo che una variabile contenga numeri interi, dobbiamo aggiungere alla fine del suo nome il simbolo %.

Provate, cioe' scrivete:

```
LET TESTO% = 5    e premete RETURN
PRINT TESTO%      e premete RETURN
```

vedrete apparire solo 5.

Avete imparato una cosa nuova: esistono due tipi di variabili. Essi sono:

-le variabili intere, il cui nome termina con %,
-le variabili reali, il cui nome non termina con %.

Il vantaggio che si ha nell'usare variabili intere, invece che decimali, e' che i calcoli vengono svolti piu' velocemente. Per evitarvi il fastidio di scrivere i nomi delle variabili intere con il carattere finale %, il BASIC del DAI vi permette di dichiarare, all'inizio del programma, che un gruppo di nomi di variabili si riferisce a numeri interi. Se, per esempio, volete che tutte le variabili aventi come primo carattere una lettera compresa tra A ed I siano considerate intere, dovete scrivere:

IMP INT A-I e premere RETURN

IMP e' una abbreviazione di IMPLY e INT e' una abbreviazione di INTEGER.
Per completare l'argomento "variabili" bisognerebbe aggiungere molte altre cose, ma, per il momento, basta quanto detto.

Ora scrivete:

NEW e premete RETURN

avete cosi' cancellato la linea di programma gia' scritta.
Scrivete ora:

80 LET TESTO = 1 e poi RETURN
90 LET SFONDO = 0 e poi RETURN
100 COLORT SFONDO TESTO 0 0 e poi RETURN

Accertatevi di avere scritto correttamente e poi scrivete:

LIST e poi RETURN

Appena premuto RETURN lo schermo viene ripulito e il programma viene listato sullo schermo. Ora provate a immaginare cosa accadrebbe se eseguiste questo programma. Se volete verificarlo, dovete scrivere:

RUN e poi RETURN

Questa parte del programma e' stata scritta per dimostrare che usando le variabili, invece dei numeri (costanti), si ottengono gli stessi risultati. In questo modo avete ottenuto un testo blu su sfondo nero. Per il momento, se volete modificare la combinazione di colori sfondo/testo, dovete cambiare i numeri contenuti nelle variabili nelle linee 80 e/o 90, senza modificare la linea 100 e dare nuovamente il RUN al programma.
Vi chiederete come fare a modificare i valori alle linee

80 e 90. Per il momento lo potete fare solo "in modo brutale", e cioe' riscrivendo le linee per intero. Vi preghiamo di provare; vi renderete conto che quando voi scrivete di nuovo una linea di programma che esiste gia', cioe' scrivete una linea di programma con lo stesso numero di linea di un'altra, l'ultima scritta si sostituisce alla precedente. Per la stessa ragione, se desiderate cancellare completamente una linea di programma dovete solo scrivere il numero di linea e poi premere RETURN. Infatti, operando cosi', avete scritto una linea di programma tutta vuota ed il sistema non la considera.

Fate un po' di prove con i colori e poi procedete nella lettura del manuale.

Noi stiamo scrivendo un programma con lo scopo di mostrare tutte le possibili combinazioni di colore sfondo/schermo in sequenza. Per ottenere questo dobbiamo assegnare due valori iniziali alle variabili TESTO e SFONDO, come realmente facciamo nelle linee 80 e 90, poi cambiare il colore allo schermo, come facciamo alla linea 100. Dobbiamo poi ottenere che sia il calcolatore a modificare i valori contenuti in SFONDO e TESTO, invece di farlo noi manualmente. Dal momento che vogliamo vedere tutte le combinazioni possibili, possiamo cominciare con aggiungere 1 alla variabile SFONDO, in modo che contenga il numero corrispondente al colore successivo.

Quando voi desiderate che il calcolatore aggiunga un numero ad una variabile gia' esistente, gli chiedete (LET) che la variabile venga ad essere uguale al suo valore precedente aumentato del valore che voi desiderate, cioe':

```
LET SFONDO = SFONDO + 1.
```

Prima di procedere, dovete essere ben sicuri di aver compreso cosa succede in un programma operando come detto sopra. Potete fare qualche esercizio di questo tipo; scrivete:

```
LET CONTO = 35 e RETURN
? CONTO e RETURN
```

vedrete apparire 35.0; poi scrivete:

```
LET CONTO = CONTO + 12 e RETURN
? CONTO e RETURN
```

vedrete apparire 47.0.

Provate e riprovate fino a quando sarete sicuri che questo difficile concetto del BASIC vi e' divenuto familiare e poi procedete.

Ora cercheremo di applicare quanto appreso sulla possibilita' di aggiungere un numero ad una variabile gia'

esistente. L'uso della parola chiave LET e' opzionale in questo tipo di frasi BASIC; se non si usa LET si risparmia un po' sulla lunghezza del programma. Tuttavia, noi pensiamo che, all'inizio della vostra carriera di programmatori, sia meglio per voi usare la parola LET nei vostri programmi. Questo servira' a farvi meglio ricordare il suo significato.

Ora potete aggiungere questa linea al nostro programma:

```
110 LET SFONDO = SFONDO + 1   e poi RETURN
```

Quando la frase 110 viene eseguita, il valore dell'espressione alla destra del segno = viene calcolato e memorizzato nella variabile posta a sinistra del segno stesso. E' chiaro che il valore di SFONDO, nell'espressione a destra del segno =, e' quello che vi era stato memorizzato alla linea 90. Questo valore precedente di SFONDO aggiunto a 1 viene memorizzato nella variabile a sinistra del segno = che e' ancora SFONDO e quindi SFONDO modifica il suo valore. Dopo l'esecuzione della linea 110 SFONDO contiene 1.

Prima di dare il RUN al programma e' meglio verificare cosa c'e' in memoria e quindi chiedere la lista del programma con LIST e poi RETURN; voi dovreste vedere:

```
80 LET TESTO = 1
90 LET SFONDO = 0
100 COLORT SFONDO TESTO 0 0
110 LET SFONDO = SFONDO + 1
```

abbiamo detto "dovreste" perche', se avete fatto qualche prova di colore, puo' darsi che le linee 80 e 90 non si trovino esattamente come desiderato. In tale caso riscrivetele e richiedete nuovamente la lista. Desideriamo ora pero' mostrarvi come si possono correggere le linee di programma senza doverle riscrivere completamente.

0.9. FUNZIONAMENTO DEL COMANDO EDIT

Scrivete ora:

```
EDIT e poi RETURN
```

Vedrete listato il vostro programma sullo schermo in modo un po' diverso da prima:

-a sinistra vedrete una linea verticale che parte sotto la vostra ultima linea del programma;
-alla fine di ogni linea vedrete uno strano simbolo che assomiglia ad una piccola falce e che segna il punto dove avete premuto il tasto RETURN in fase di scrittura del programma;
-il cursore lampeggiante lo troverete nell'angolo in alto a

sinistra sulla prima cifra del primo numero di linea del programma. Esso puo' essere spostato sullo schermo usando uno dei quattro tasti di controllo del cursore, gia' descritti, muovendolo verso l'alto o il basso, a destra o a sinistra;
-se premete un tasto il carattere corrispondente verra' inserito alla sinistra del cursore lampeggiante;
-il tasto CHAR DEL serve per cancellare il carattere sul quale si trova il cursore lampeggiante e per spostare a sinistra di un posto tutto quello che viene dopo il carattere cancellato.

Tutto questo sembra molto complicato quando lo si legge, ma in pratica troverete che il modo EDIT e' molto versatile e potente, facile da usare e lo troverete indispensabile.
Provate a cambiare il valore di SFONDO e TESTO operando cosi':

-posizionate il cursore alla destra del segno = della linea 90, in modo che lampeggi sulla prima cifra del valore di SFONDO. Se dovete fare un grande spostamento potete tenere premuto insieme al tasto/freccia desiderato anche il tasto REPT per ottenere un movimento piu' veloce.
-premete il tasto CHAR DEL tante volte quante sono le cifre da cancellare.
-scrivete il nuovo valore.
-spostate il cursore dove vi serve di nuovo.

Quando avete terminato di correggere il programma dovete uscire dal modo EDIT. Ci sono due modi per farlo:

-Se siete sicuri di avere bene operato e non desiderate mantenere la vecchia versione del programma, premete il tasto BREAK una volta e dopo premete la barra dello SPAZIO. Se i programmi sono lunghi, dopo pochi secondi appare l'asterisco (*) ma non premete alcun tasto durante questa attesa. Se i programmi sono corti l'asterisco apparira' quasi subito.
-Se volete mantenere la vecchia versione del programma, senza le correzioni apportate, annullando tutte le modifiche operate nel modo EDIT, premete il tasto BREAK due volte.

Ora e' conveniente provare e riprovare queste procedure per diventare sicuri nell'operare correzioni nel modo EDIT. Potete anche dare il RUN al vostro programma ed osservare i diversi colori dello schermo, ma alla fine dei vostri esercizi, dovete chiedere la lista del programma ed assicurarvi che corrisponda all'ultima stesura e cioe':

```
80 LET TESTO = 1
90 LET SFONDO = 0
100 COLORT SFONDO TESTO 0 0
```

110 LET SFONDO = SFONDO + 1

Ora dobbiamo procedere nello studio del BASIC, per imparare come fare a modificare i colori senza il nostro intervento manuale.

0.10. I CICLI DI PROGRAMMA

L'ultima linea di programma introdotta, la 110, modifica il numero contenuto in TESTO al valore 1. Per farla agire dovremo proseguire con la linea 120, cosi':

120 COLORT SFONDO TESTO 0 0 e poi RETURN

Se diamo il RUN al programma vediamo il testo in blu sullo sfondo blu, dato che si ha per lo sfondo il numero 1 e per il colore il numero 1, e quindi tutto blu . Se vogliamo vedere il testo grigio su uno sfondo rosso petunia, dobbiamo aggiungere queste due linee di programma:

130 LET SFONDO = SFONDO + 1 e poi RETURN
140 COLORT SFONDO TESTO 0 0 e poi RETURN

e dare il RUN al programma.

Vi potremmo suggerire di continuare ad aggiungere una linea 150 identica alle linee 110 e 130, ed una linea 160 identica alle linee 120 e 140, e cosi' via per poter ottenere tutti i possibili 16 colori dello sfondo. In realta' sarebbe molto noioso procedere in tale modo, forso piu' noioso che operare come all'inizio, quando andavamo a modificare le linee 80 e 90.

Invece, cercheremo di insegnarvi una delle tecniche di programmazione piu' comuni, e cioe' l'uso dei CICLI (LOOP) per eseguire operazioni ripetitive.

Uno dei modi per ottenere un CICLO in BASIC e' quello di usare l'istruzione GOTO, la quale fa proseguire l'esecuzione del programma dal numero di linea che segue il GOTO.

Scrivete:

120 GOTO 100 e poi RETURN

Prima di chiedervi di far girare il programma con la frase comando RUN, vi preghiamo di esaminare le frasi del programma per rendervi conto di quello che fanno.Le linee 80 e 90 assegnano rispettivamente i valori iniziali 1 e 0 alle variabili TESTO e SFONDO, cosi' la prima combinazione di colori che appare e': scritta blu in campo nero. La linea 100 opera il cambio dei colori a seconda del contenuto di SFONDO e TESTO. La linea 110 aggiunge 1 alla variabile SFONDO, la quale assumera', cosi', successivamente i valori

1, 2, 3, ecc. La linea 120 rimanda il programma ad eseguire la linea 100 e quindi appare una nuova combinazione di colori. Dal momento che non c'e' una istruzione che modifica il valore di TESTO, la scritta rimane sempre blu. Per effetto della istruzione 110 la variabile SFONDO assume valori crescenti a partire da 0, e quindi assume tutti i possibili valori per lo sfondo, che sono 16.

Ora, dopo aver tentato di immaginare cosa succedera', scriviamo RUN e poi RETURN ed osserviamo lo schermo.

Cosa vi aspettavate? I colori cambiano cosi' rapidamente che voi vedete come risultato una scritta blu su uno sfondo bianco ed un messaggio:

NUMBER OUT OF RANGE IN LINE 100

Cosa e' accaduto? Se voi non cercate di abbassare la velocita' di funzionamento del calcolatore, i colori cambiano cosi' rapidamente che l'occhio umano non li puo' percepire. Fortunatamente esiste una istruzione apposita nel BASIC del DAI che puo' essere usata per inserire una pausa nella esecuzione di un programma. Allora scrivete:

105 WAIT TIME 100 e poi RETURN

Il numero 100 e' il numero di intervalli, ciascuno della durata di 20 millisecondi, durante i quali noi vogliamo fermare il calcolatore. In questo caso 100 volte 20 equivale a 2000 ms, quindi a 2 secondi di pausa.

Per quanto riguarda il messaggio di errore, cioe' l'avviso che la variabile SFONDO ha superato il valore 15 valore massimo consentito nella operazione COLORT, esso significa che il nostro programma non e', per il momento, corretto.

Per vedere come correggere il nostro errore dobbiamo procedere nello studio.

0.11. LA FRASE IF

Una delle cose che fa piu' impressione alle persone che non sanno programmare e' che il calcolatore puo' prendere delle decisioni a seconda che si verifichino o meno alcune condizioni. Un modo per ottenere questo e' usare in BASIC la frase IF.....THEN.

Proviamo anche noi ed aggiungiamo nel nostro programma la linea:

```
120 IF SFONDO < 16 THEN GOTO 100 e poi RETURN
```

come potete vedere, invece di avere alla linea 120 una istruzione che fa eseguire al programma un SALTO INCONDIZIONATO alla linea 100 con la frase GOTO 100, in questo caso si ha un salto alla linea 100 solo se (IF) la variabile SFONDO contiene un numero minore di 16. Con questa frase evitiamo di avere il messaggio di errore NUMBER OUT OF RANGE IN LINE 100. Quando la variabile SFONDO raggiunge il valore 16, e quindi la condizione non e' piu' soddisfatta, il programma non fa il salto alla linea 100 ma prosegue con la linea che segue la 120. Nel nostro caso, dato che non abbiamo altre linee, il programma si ferma. Non ha piu' senso mantenere le linee 130 e 140 che avevamo scritto prima di studiare i cicli, e quindi le dobbiamo cancellare. Se ora chidete la lista del programma, prima di dare RUN, vedrete:

```
80 LET TESTO  = 1
90 LET SFONDO = 0
100 COLORT SFONDO TESTO 0 0
105 WAIT TIME 100
110 LET SFONDO = SFONDO + 1
120 IF SFONDO < 16 THEN GOTO 100
```

Abbiamo gia' ottenuto dei buoni risultati. Ma perche' non chiedere al calcolatore di dirci quali numeri identificativi usa per i colori? La linea di programma che fa questo e' la seguente:

```
101 PRINT "SFONDO = ";SFONDO,"TESTO = ";TESTO e poi RETURN
```

Come potrete vedere facendo girare il programma, il calcolatore scrivera' all'inizio:

```
SFONDO = 0          TESTO = 1
```

e questo avviene perche' "SFONDO = " e "TESTO = " sono racchiusi tra apici, inoltre dopo " SFONDO = " appare subito il contenuto della variabile SFONDO perche' essa e' preceduta dal punto e virgola, mentre " TESTO = " viene spostato all'inizio del prossimo campo sullo schermo perche' e' preceduto dalla virgola. Avete sentito per la

prima volta la parola "campo" a proposito dello schermo; ogni riga dello schermo e' divisa in 5 campi di 12 caratteri ciascuno. Se i dati, costanti o variabili, che si vogliono stampare sullo schermo sono separati dal punto e virgola non si hanno spostamenti verso destra, mentre se si usa la virgola si ha lo spostamento al prossimo campo. Tutto questo e' molto piu' complicato da descrivere che da provare. Per questo vi preghiamo di fare qualche prova, per esempio, scambiando tra loro le virgole con i punti e virgola e vedere cosa succede.

Avrete constatato che scrivere un programma e nello stesso tempo spiegare le piu' elementari nozioni del BASIC non e' un lavoro veloce. Ora il nostro programma non e' ancora completo. Infatti siamo riusciti a far cambiare i colori dello SFONDO, a vedere quali numeri corrispondono ai diversi sfondi, ma non abbiamo ancora visto variare i colori del testo. Per completare il nostro programma e vedere tutte le possibili combinazioni sfondo/testo, noi dobbiamo cambiare il colore del testo ogni volta che e' stato percorso tutto il ciclo di cambiamento dei colori dello sfondo. Dobbiamo, quindi, ora aggiungere queste tre linee di programma:

```
130 LET TESTO = TESTO + 1 e poi RETURN
140 IF TESTO < 16 THEN GOTO 90 e poi RETURN
150 COLORT 15 0 0 0 : REM PER OTTENERE TESTO NERO SU  SFONDO
BIANCO
```

e dobbiamo modificare la linea 80 cosi':

```
80 LET TESTO = 0 e poi RETURN
```

La linea 130 viene eseguita solo se non si e' verificata la condizione che fa tornare il programma alla linea 100, cioe' quando SFONDO e' uguale a 16 e sono quindi passati tutti i possibili colori dello sfondo per il testo in essere. In quel momento viene cambiato il colore del testo dalla linea 130. La linea 140 verifica che la variabile TESTO non diventi 16, cosa che provocherebbe un'altra volta l'errore NUMBER OUT OF RANGE IN LINE 100 , e fa tornare il programma alla linea 90, dove SFONDO viene rimesso al valore zero e puo' ripercorrere il ciclo da 0 a 15. Quando TESTO raggiunge il valore 16 il programma terminerebbe lasciandovi davanti ad uno schermo completamente bianco, che corrisponde ai valori SFONDO = 15 e TESTO = 15. Con la linea 150 si ottiene il testo in nero sullo sfondo bianco (SFONDO = 15 e TESTO = 0). Questo viene spiegato nella stessa linea 150 dopo la parola chiave REM.

REM e' l'abbreviazione di REMARK (annotazioni). Qualunque cosa si scriva dopo REM serve solo per il programmatore e non modifica l'azione del programma. Il calcolatore ignora le frasi REM quando esegue un programma ma le stampa quando

chiedete la lista del programma. Esse sono di grande utilita' per il programmatore che puo' scriverne quante vuole nei punti delicati del programma per ricordarsi quanto interessa in una particolare circostanza. Vi preghiamo di notare anche che nella linea 150 prima della parola chiave REM, compaiono i due punti (:). Essi servono a separare due istruzioni sulla medesima linea di programma. Non vi consigliamo di fare uso all'inizio della vostra esperienza di programmatori di molte istruzioni multiple (istruzioni separate da :) perche' questo puo'diminuire la chiarezza del programma. Sara' meglio limitarsi ad aggiungere REM, preceduto dai due punti, dopo una frase che necessiti di qualche spiegazione. L'uso delle istruzioni multiple serve a risparmiare memoria nella stesura di un programma ed a rendere piu' veloce l'esecuzione di un programma. Sara' meglio rimandare ai paragrafi successivi questi perfezionamenti nella stesura di un programma.

Scrivete ora:

```
999 END e RETURN
```

Ora finalmente abbiamo un programma che fa tutto quello che noi volevamo, richiediamone la lista, con LIST, e vedremo:

```
80 LET TESTO = 0
90 LET SFONDO = 0
100 COLORT SFONDO TESTO 0 0
101 PRINT "SFONDO = ";SFONDO,"TESTO = ";TESTO
105 WAIT TIME 100
110 LET SFONDO = SFONDO + 1
120 IF SFONDO < 16 THEN GOTO 100
130 LET TESTO = TESTO + 1
140 IF TESTO < 16 THEN GOTO 90
150 COLORT 15 0 0 0 : REM TESTO NERO SU SFONDO BIANCO
999 END
```

La frase 999 END serve per chiudere il programma e non e' necessaria ma rende piu' evidente la fine del programma. Ora potete far girare il programma e decidere quale e' la combinazione di colori che vi piace di piu' e, dal momento che vengono evidenziati i valori di SFONDO e TESTO, prenderne nota.

Se vi sentite sicuri e pensate di avere ben compreso tutto quello che fa il programma appena ultimato, potete cominciare a pensare di scrivere un programma tutto vostro per il vostro DAI.

0.12. COME SI MEMORIZZANO E RICHIAMANO I PROGRAMMI

Prima di procedere ed imparare altre cose interessanti , e' il momento di imparare come si memorizza un programma su nastro per poterlo richiamare piu' tardi.
Non c'e' nulla di piu' semplice ma procediamo insieme passo passo come sempre:

1) Voi dovete, naturalmente, avere un programma in memoria che e', per esempio, quello dedicato ai colori;
2) Mettete la cassetta sulla quale desiderate registrare il programma nel registratore e siate pronti a far partire la registrazione quando ve lo chiede il calcolatore;
3) Inventate un nome per il vostro programma; questo e' necessario se userete la cassetta per registrare anche altri programmi e vorrete distinguerli tra loro in base ad un nome. Potreste chiamare il programma: COLORPROVA.
4) Scrivete:

SAVE "COLORPROVA" e poi RETURN

il calcolatore vi rispondera':

SET RECORD, START TAPE, TYPE SPACE

e questo e' quello che dovete fare (POSIZIONATE IL NASTRO, FATE PARTIRE IL NASTRO, PREMETE IL TASTO SPAZIO);
5) Quando appare (*) asterisco voi siete avvisati che il calcolatore ha finito di registrare il programma su nastro.

Quello che dovete fare ora e' di ripetere i punti 4) e 5) una seconda volta e cosi' otterrete una seconda registrazione del vostro programma. Si dice di aver fatto una registrazione ridondante ma questo vi da' una maggior sicurezza di non perdere il vostro programma. Infine, se volete essere veramente sicuri che il programma e' stato registrato bene, dovete fare il CHECK (controllo) della registrazione. Per eseguire il controllo riavvolgete il nastro all'inizio del programma registrato e poi scrivete cosi':

CHECK e poi RETURN

subito dopo avviate il registratore premendo il tasto PLAY. Se tutto e' a posto dopo poco vedrete:

COLORPROVA OK

se non lo e', vedrete:

COLORPROVA BAD

Se appare questo ultimo messaggio puo' essere successa una delle seguenti cose:

-la testina del registratore deve essere pulita;
-la qualita' della cassetta non e' abbastanza buona;
-il volume usato durante la registrazione e/o la lettura non andava bene.

Cosa fare nei due primi casi, e' ovvio. Nell'ultimo caso si devono rifare i punti 4) e 5) ed il CHECK usando diverse regolazioni per il volume fino a quando va tutto bene. Conviene allora prendere nota della regolazione di volume che ha dato un buon risultato e non si avranno piu' problemi.

Quando siete sicuri di avere effettuato correttamente l'operazione di SAVE vi conviene provare a spegnere il vostro DAI; poi riaccendetelo e caricate il programma dal nastro nella memoria. Dovete procedere cosi':

1) Riavvolgere il nastro riportandolo all'inizio della registrazione;
2) Scrivere:

LOAD COLORPROVA

3) Premere il RETURN e mettere il registratore su PLAY.

Quando appare l'asterisco (PROMPT), potete richiedere la lista del vostro programma e lo vedrete identico a prima.

Ora avete acquisito un numero sufficiente di nozioni riguardo al vostro DAI ed al suo linguaggio; potete quindi cominciare a fare conoscenza con una delle caratteristiche del vostro personal che lo distingue da tutti gli altri: la GRAFICA A COLORI.

0.13. UN APPROCCIO RISOLUTO

Un disegno sullo schermo del televisore, come una fotografia su carta, e' formato da un gran numero di punti. Quanto piu' piccoli sono i punti e quanto piu' grande e' il loro numero in una data area tanto migliore e' la qualita' della immagine prodotta. In termini tecnici si dice che una pellicola fotografica ha un' alta risoluzione quando il numero dei punti diversi che formano un'area e' cosi' alto, e le loro dimensioni sono cosi' piccole, che il disegno non appare a "punti".
Il DAI vi consente di disegnare sullo schermo, scegliendo tra 3 livelli di risoluzione; essi sono:
-72 punti orizzontali per 65 verticali;

-160 punti orizzontali per 130 verticali;
-336 punti orizzontali per 256 verticali.

A ciascuno di questi 3 livelli voi potete scegliere di usare qualunque quartina dei 16 colori disponibili o tutti i 16 colori. Voi potete anche decidere se usare l'intero schermo per la grafica, o se lasciare spazio nella parte bassa dello schermo per linee di testo (che possono arrivare fino a 4). Il numero totale di opzioni possibili (chiameremo queste opzioni MODE) e' 13, se contiamo il MODO 0 che e' il MODO ALL-TEST. Segue, per comodita', la tabella dei MODI.

MODO	RISOLUZIONE	DIMENSIONI TESTO	COLORI
0	-	24x60	qualunque coppia
1	72x65	-	16
1A	72x65	4 x60	16
2	72x65	-	qualunque quartina
2A	72x65	4 x60	qualunque quartina
3	160x130	-	16
3A	160x130	4 x60	16
4	160x130	-	qualunque quartina
4A	160x130	4 x60	qualunque quartina
5	336x256	-	16
5A	336x256	4 x60	16
6	336x256	-	qualunque quartina
6A	336x256	4 x60	qualunque quartina

La procedura per selezionare un MODO grafico, puo' non sembrare semplice; per esempio se scrivete:

MODE 1 e poi RETURN

il DAI prepara lo schermo per mostrare punti colorati invece di caratteri alfanumerici. Notate pero' che la parte bassa dello schermo e' ancora a disposizione per mostrare fino a 4 linee di testo. Secondo la tabella, appena vista, questo dovrebbe accadere con il MODO 1A e non con il MODO 1. Evidentemente non potete ottenere completamente le possibilita' grafiche sullo schermo quando la macchina si trova nello stato CONTROL. E questo e' un bene perche' altrimenti non potreste controllare quello che avete scritto sullo schermo. Se ne conclude che la scelta dei modi funziona completamente solo quando la macchina si trova nello stato PROGRAMMA, cioe' quando sta eseguendo un programma e voi non potete battere comandi sulla tastiera.

Provate cosi':

```
10 MODE 1 e poi RETURN
20 GOTO 20 e poi RETURN
RUN e poi RETURN
```

Vedete? Ora l'intero schermo viene usato per la grafica. Questo programma non terminera' mai, perche la linea 20 lo fa ritornare su se stesso continuamente.

Per fermare il programma dovete premere il tasto BREAK, vedrete che l'area dedicata alla grafica sullo schermo si restringe e crea in basso il posto per 4 linee di testo. Se il programma che avete interrotto con il tasto BREAK fosse stato un programma che produceva un meraviglioso disegno sullo schermo vi chiederete dove e' andata quella parte di figura che e' sparita per lasciare posto alle 4 linee di testo. Bene, quella parte di figura e' stata conservata in memoria, in una parte di memoria non in diretta comunicazione con lo schermo, ma puo' facilmente essere recuperata, usando un artificio, che vi insegneremo piu' avanti.

Ora potete provare tutti i MODI scrivendo la parola MODE seguita dai numeri da 2 a 6.

0.14. DUE BITS O NON DUE BITS

Vi chiederete cosa e' questa faccenda a proposito della quartina di colori o dei 16 colori. Prima di pensare di avere perso 12 colori per strada permetteteci di spiegarvi di cosa si tratta. Non vi aspettate di trovare del tutto chiaro e comprensibile quello che cercheremo di spiegarvi. Voi potrete usare ugualmente bene le possibilita' grafiche del vostro DAI ma e' importante che leggiate queste pagine per capire perche' ci sono alcune limitazioni nell'uso della grafica a colori. Una piu' completa e dettagliata descrizione del sistema grafico viene data nella seconda parte di questo manuale.

Le restrizioni, alle quali abbiamo accennato dipendono principalmente dal fatto che il sistema che abbiamo adottato ci permette di lavorare con 16 colori in MODO ad alta risoluzione usando la meta' della memoria che sarebbe necessaria per un sistema senza alcuna restrizione.

0.15. DISCUSSIONE

Per poter mostrare i punti colorati sullo schermo uno speciale circuito elettronico, situato all'interno del DAI e che noi chiamiamo "circuito video", scandisce continuamente un'area della memoria RAM per sapere quale colore deve avere ogni punto.

Questa parte della RAM (chiamata SCREEN REFRESH MEMORY o MEMORIA DI SCHERMO) memorizza lo stato dei punti dello schermo (una specie di fotografia dello schermo). Parecchie volte durante ogni secondo i circuiti del video del DAI riproducono il disegno sullo schermo usando lo schema della memoria di schermo come modello.

Iniziando da un caso semplice, supponiamo di voler produrre un disegno in due colori. Voi probabilmente sapete che la piu' piccola unita' di memoria nel calcolatore e' il BIT che e' una cifra che puo' assumere solo i valori 0 e 1. I circuiti del video vedono la memoria di schermo come una griglia dove ogni punto dello schermo corrisponde ad un bit. Per convenzione i circuiti del video riconoscono di dover produrre un punto di un colore (diciamo il COLORE DELLO SFONDO che viene anche chiamato BACKGROUND COLOR) se il corrispondente bit e' zero. Ogni volta che questi "pittori elettronici" incontrano un bit al valore 1 essi producono il corrispondente punto nel secondo colore disponibile (che chiamiamo FOREGROUND COLOR o COLORE DI PRIMO PIANO). Questo avviene qualunque siano i due colori scelti. In questo modo funzionano la maggior parte di sistemi televisivi grafici in bianco e nero.

Come voi potete immaginare, il numero dei BITS di memoria corrispondono al numero dei punti dello schermo. In questo caso, per poter proiettare 86.016 punti (336 x 256), abbiamo bisogno di 86.016 bits di memoria RAM. Si usa parlare di memoria in termini di BYTES (questa infatti e' l'unita' di misura per la capacita' della memoria nei calcolatori e 1 BYTE = 8 BITS) e quindi servono 10.752 bytes di memoria RAM per conservare le informazioni necessarie ai circuiti del video nel MODO ad alta risoluzione. Questa non e' poi una grossa quantita' di memoria. Infatti il DAI e' dotato di 49152 bytes di memoria RAM che, adottando come moltiplicatore il K che e' uguale a 1024, corrispondono a 48K bytes.

Ora noi desideriamo avere i punti multicolori. Dobbiamo pero' ricordare che con un singolo bit si possono solo rappresentare i numeri 0 e 1. Con 2 bits si possono rappresentare i numeri da 0 a 3 (consentendo la scelta tra 4 colori diversi) mentre con 4 bits si possono rappresentare i numeri da 0 a 15 (consentendo la scelta tra 16 colori diversi).

Allora per poter specificare il colore di ogni punto dello schermo sono necessari 4 bits. Nel modo ad alta risoluzione questo significherebbe l'impegno di un blocco di 344.064 bits di memoria per memorizzare lo stato degli 86.016 punti (336 x 256). Questo significherebbe che sarebbero impegnati per il video 43.008 bytes della memoria RAM del DAI. Poiche' il DAI ha 48K (49.152 bytes) RAM resterebbe troppo poco spazio di memoria per i programmi.

Per poter ridurre le dimensioni della memoria di schermo si potrebbe ridurre o il numero di colori disponibili o il potere risolutivo. Dato che e' stato deciso di mantenere l'alto potere risolutivo si sono dovute imporre delle limitazioni nella scelta dei colori. Il nostro sistema dimezza l'occupazione di memoria e ci permette di lavorare

con 16 colori con qualche piccola limitazione.
Attualmente noi possiamo scegliere tra due possibilita' per risparmiare memoria; sta a voi decidere quale sia la decisione giusta in ogni singolo caso. Le due possibilita' sono:
-il MODO a 16 colori;
-il MODO a 4 colori.
Ci occuperemo per prima cosa della seconda possibilita' anche se molte delle cose che diremo sono valide anche per la prima.

In questo caso si ha risparmio di memoria limitando a 4 i colori che possono apparire contemporaneamente sullo schermo.
Vengono associati a ciascun punto solo 2 bits di memoria (riducendo a 21.504 bytes l'occupazione di memoria per l'alta risoluzione) e da questi i circuiti del video traggono le loro informazioni. Questo pero' non significa che voi dovete usare sempre gli stessi 4 colori. Dovete sceglierne 4 per volta tra i 16 possibili e posizionare i loro 4 numeri nei 4 registri (speciali locazioni di memoria) chiamati REGISTRI COLORE (COLOUR REGISTERS). Cosi' in ogni istante sono visibili sullo schermo solo 4 colori differenti. Se voi cambiate uno dei numeri indicativi dei colori in uno dei 4 registri subito tutti i punti dello schermo che corrispondevano al vecchio colore vengono modificati. Questa possibilita' permette di ottenere effetti molto interessanti incluso quello dell'animazione delle figure; infatti potete ottenere piccoli movimenti delle figure non mostrando le medesime nella nuova posizione fino a quando non sono completamente disegnate (troverete piu' ampie spiegazioni al riguardo nella seconda parte del manuale). Quanto detto puo' sembrare un po' confuso ma provando praticamente vi sembrera' piu' chiaro.

Tutti gli esempi che seguono si basano sul MODO a bassa risoluzione. Se desiderate provare in alta risoluzione, dovete solo cambiare il comendo MODE.
Provate a scrivere:

MODE 2 e poi RETURN

lo schermo diventera' nero ad eccezione della parte bassa, dove possono essere contenute fino a 4 linee di testo. Il colore della parte dedicata al testo e' indipendente dai valori dei numeri contenuti nei REGISTRI COLORE. Dal momento che voi non avete modificato il contenuto dei REGISTRI COLORE dopo aver acceso il calcolatore essi contengono zero. Proprio per questa ragione lo schermo appare nero. Se voi modificate il contenuto del primo REGISTRO COLORE subito lo schermo cambia colore in accordo con il numero voluto. Se volete avere lo schermo blu, scrivete:

COLORG 1 0 0 0 e poi RETURN

0.16. IL COMANDO DOT

Ora scrivete:

DOT 5,5 14 e poi RETURN

ed il calcolatore vi rispondera':

COLOR NOT AVAILABLE

Questo succede perche' con il comando DOT, voi chiedete al calcolatore di mostrare un punto giallo (colore numero 14) sullo schermo. Ma il colore giallo, cioe' il numero 14, non e' stato caricato in uno dei registri colore e quindi non e' disponibile. Se voi provate con:

DOT 5,5 0 e poi RETURN

vedrete un punto (o meglio un piccolo quadrato dato che siamo in bassa risoluzione) nero sullo schermo. La posizione del punto dipende dai due numeri separati dalla virgola (5,5) che compaiono dopo la parola chiave DOT nel comando. Se ricordate un po' di geometria, piu' precisamente le coordinate cartesiane, vi renderete conto che 5,5 sono rispettivamente la x e la y del punto in un sistema di coordinate avente l'origine (0,0) nell'angolo in basso a sinistra dello schermo in modo che il quadrante positivo e' quello mostrato.

Potete anche pensare lo schermo come diviso in un certo numero di righe e colonne e che DOT 5,5 selezioni la quinta colonna a partire dall'angolo inferiore sinistro, e la quinta riga. Il colore del punto e' determinato dall'ultimo numero separato da uno spazio dai primi due nel comando DOT. Pero', per poter essere felicemente usato, il numero del colore deve prima essere stato caricato in uno dei registri colore.

0.17. IL COMANDO COLORG

Vediamo come si fa a caricare i registri colore. E' semplice; basta scrivere questa frase:

COLORG 14 1 5 15 e poi RETURN

per caricare rispettivamente la quartina di colori:
-giallo = 14
-blu = 1

-verde = 5
-bianco = 15

Con questo comando, lo schermo diventa giallo perche' il primo registro contiene 14. Vedrete anche un punto blu invece di quello nero di prima perche' il secondo registro contiene 1.

Se volete un punto verde nell'angolo in basso a sinistra potete scrivere:

DOT 0,0 5 e poi RETURN

e, se volete un punto bianco sotto quello blu potete scrivere:

DOT 5,6 15 e poi RETURN

Ora divertitevi a disegnare sullo schermo e ricordate che nessun manuale puo' insegnarvi quello che imparerete facendo la vostra esperienza per riuscire a realizzare un vostro progetto.

0.18. XMAX E YMAX

Come si fa a far apparire un punto nell'angolo in basso a destra? Si puo' andare a vedere nella tabellina dei modi e ricordare il numero massimo di colonne con il quale state lavorando. Dal momento che ora state lavorando nel MODE 2 troverete che il numero massimo di colonne e' 72 e ricordando che per il calcolatore si deve cominciare a contare da 0 il numero da usare e' 71. Allora potete mettere un punto blu nell'angolo in basso a destra scrivendo:

DOT 71,0 1 e poi RETURN

Esiste comunque un modo piu' semplice per riferirsi al numero massimo di riga e di colonna; invece di mettere il numero nel comando DOT si puo' scrivere XMAX per il piu' alto numero di colonna e YMAX per il piu' alto numero di riga. Tenete presente che la riga 0 e' quella piu' in basso mentre la colonna 0 e' quella piu' a sinistra. Questo e' importante perche' anche se passate ad un modo con piu' alta risoluzione non dovete modificare il comando DOT ma esso e' automaticamente valido nel nuovo modo. Facciamo un piccolo esempio:

DOT XMAX,0 14 e poi RETURN

fa sparire il punto blu nell'angolo in basso a destra perche' lo ricopre con un punto giallo e quindi lo rende uguale allo sfondo.

Per posizionare un punto al centro dello schermo,

qualunque sia il modo di risoluzione scelto, potete scrivere:

DOT XMAX/2,YMAX/2 5 e poi RETURN

Il punto verde che avete cosi' ottenuto non sembra proprio nel centro dello schermo. Questo dipende dal fatto che state lavorando in modo diretto ed avete a disposizione le quattro linee inferiori dello schermo riducendo di conseguenza l'area per la grafica. Se aveste scritto i comandi ora visti in un programma l'effetto sarebbe stato diverso. Vi insegnamo ora un piccolo trucco per fare sparire le linee inferiori, a vantaggio della grafica. Scrivete:

60000 MODE 2 e poi RETURN
60010 GOTO 60010 e poi RETURN
RUN 60000 e poi RETURN

Avete imparato un'altra cosa nuova: potete dare il comando di RUN a partire da una determinata linea di programma, in questo caso 60000. Ora voi avrete lo schermo totalmente dedicato alla grafica e vedrete il punto verde esattamente al centro. Il trucco consiste nello scrivere e far girare un piccolo programma che riseleziona lo stesso modo che avevate gia' scelto in modo diretto e poi crea un ciclo senza fine (linea 60010) solo per non lasciarvi il controllo del calcolatore e quindi non perdere le 4 linee inferiori dello schermo. E' evidente che questo programmino potrebbe essere scritto iniziando con qualunque numero di linea. La scelta di 60000 e' stata fatta per non disturbare eventuali altri programmi gia' presenti in memoria. Sapete gia' che se volete interrompere il ciclo dovete usare il tasto BREAK.
Oltre al comando DOT potete usare due altri comandi per disegnare sullo schermo.

0.19. IL COMANDO DRAW

Se volete ottenere una linea orizzontale blu da sinistra a destra dello schermo (da colonna 0 a colonna 71) a distanza di 10 righe dall'alto, invece di scrivere tutti i punti necessari uno dopo l'altro, potete scrivere:

DRAW 0,9 XMAX,9 1 e poi RETURN

Se volete tagliare lo schermo in diagonale, potete scrivere:

DRAW 0,0 XMAX,YMAX 5 e poi RETURN

e vedrete una diagonale verde.
Quindi per tracciare una linea, dopo la parola chiave

DRAW, seguita da uno spazio, dovete scrivere la posizione del punto di inizio della linea poi un'altro spazio e la posizione del punto di arrivo della linea e, dopo un'altro spazio, il colore della linea.

0.20. IL COMANDO FILL

Per ottenere una zona quadrata o rettangolare dello schermo o, al limite, l'intero schermo di un determinato colore potete usare il comando FILL.
Per esempio, per ottenere un rettangolo verde che abbia l'angolo in basso a sinistra nel punto 7,7 e l'angolo in alto a destra nel punto 20,20, possiamo scrivere:

FILL 7,7 20,20 5 e poi RETURN

0.21. PER CONTO VOSTRO

E' venuto il momento di controllare cosa avete imparato! Provate ad ottenere un disegno che desiderate, provate a farlo in modo diretto, cioe' scrivendo i comandi uno per volta senza numero di linea in modo che il comando viene subito eseguito quando si preme il tasto RETURN. Provate a programmare il disegno e fate girare il programma. Se sbagliate qualcosa otterete il messaggio SYNTAX ERROR , allora cercate di capire dove avete sbagliato, correggete e riprovate. Se otterrete un messaggio cosi': OFF SCREEN significa che avete tentato di scrivere fuori dell'area grafica. Provate anche a cambiare il MODE e quando vi sentirete sicuri procedete nella lettura del manuale.

0.22. IL MODE 16-COLORI

In questo modo voi potete disporre contemporaneamente di 16 colori. Pero' c'e questa limitazione: in una linea orizzontale di punti non potete avere 16 punti uno dopo l'altro in 16 colori diversi.
Vediamo come lavora il sistema:

-ciascuna linea orizzontale e' divisa in un numero di segmenti ciascuno di 8 punti. A seconda del grado di risoluzione scelto ci sono 9, 20 o 42 di questi segmenti, o campi come si suole chiamarli, in ogni linea orizzontale;
-all'interno di ognuno di questi campi si possono usare solo due colori diversi nello stesso momento.

Questi campi, di 8 punti ciascuno, lavorano come abbiamo gia' visto per la grafica a due colori. Ciascuno degli 8 bits di memoria, che corrispondono alle posizioni dei punti nel campo, puo' avere o il valore 0 o il valore 1 ed informa

i circuiti del video su quale dei 2 possibili colori disponibili per quel campo deve essere usato. Quali colori allora? La risposta e' QUALUNQUE COPPIA DI DUE COLORI scelta tra i 16 disponibili.

Invece di avere i 4 registri per i colori, come nel modo a 4 colori, nel modo a 16 colori ciascun campo dispone di una coppia di registri-colore, indipendenti da quelli degli altri campi. Cosi' in memoria devono essere riservati 2 bytes per ciascun campo di otto punti. Nel primo di questi bytes gli 8 bits sono messi in corrispondenza con gli 8 punti del campo ed informano i circuiti del video se mostrare il primo o il secondo colore. Il secondo byte e' diviso in due parti di 4 bits ciascuna; come ricorderete in 4 bits puo' essere rappresentato un numero da 0 a 15. Cosi' nella prima parte di questo secondo byte sta il numero corrispondente al primo colore (che si ottiene con il bit a 0 per il relativo punto), e nella seconda parte sta il numero corrispondente al secondo colore (che si ottiene con il bit a 1 per il relativo punto).

Questa volta non si devono, come nel modo a 4 colori, preselezionare i 4 colori scelti caricando prima i registri-colore: la selezione dei colori avviene in modo dinamico quando si disegnano i punti.

Quando voi passate a uno dei modi a 16 colori succede questo:

-il colore memorizzato nella prima meta' dei diversi registri colore dei campi e' quello che si trovava nel primo dei 4 registri-colore del modo precedente (colore dello sfondo).

Voi potete ora mettere un primo punto di qualsiasi colore in qualunque posizione dello schermo; potete poi mettere un punto di qualsiasi colore proprio sopra o sotto un punto gia' messo. Quello che voi non potete fare e' posizionare un punto di un terzo colore in un campo dove, oltre allo sfondo (primo colore), c'e' gia' un altro punto (secondo colore).

Indubbiamente questa e' una limitazione in senso teorico, ma da un punto di vista operativo, non lo e' poi tanto e, specialmente nei modi ad alta risoluzione, non ci sono difficolta' ad ottenere dei risultati di grafica a colori veramente pregevoli.

In questo modo voi potete usare tutti i comandi gia' studiati per il modo a 4 colori. Dovete pero' ricordare che il comando COLORG ha solo questo effetto: il numero che si trovava nel primo registro colore quando voi siete passati in uno dei modi a 16 colori determina il colore dello sfondo. Ogni ulteriore cambiamento cioe' uso del comando COLORG mentre e' attivo uno dei modi a 16 colori, non ha alcun effetto sullo schermo. Diciamo dunque che il comando COLORG deve essere anticipato rispetto al passaggio in uno dei modi a 16 colori.

Provate ora ad usare i vari modi a 16 colori.
A questo punto avete gia' acquisito una discreta conoscenza del vostro DAI, ma vi restano ancora molte cose da scoprire.

La seconda parte di questo manuale vi consentira' di conoscere a fondo il vostro DAI, le sue caratteristiche e le sue possibilita' affinche' possiate usarlo con la massima soddisfazione.

CAPITOLO 1

C A R A T T E R I S T I C H E D E L D A I

1.0. DESCRIZIONE GENERALE

Il Calcolatore Personal DAI e' stato progettato per fornire le migliori prestazioni, pur mantenendosi economicamente competitivo. Esso e' stato costruito in modo che i programmi possano essere caricati da registratori a cassetta di tipo audio o da floppy-disk. I risultati delle elaborazioni dei programmi vengono inviati per mezzo di un connettore per antenna ad un ricevitore televisivo standard di tipo PAL. Il generatore di suoni fornisce in uscita due segnali separati per il collegamento stereofonico sinistro e destro e per il canale sonoro della televisione.
Le risorse del DAI possono essere divise in 4 parti:

-il microcalcolatore;
-la sezione grafica;
-la sezione sonora;
-la sezione per la gestione dell'Input e dell'Output.

Il sistema e' fortemente ottimizzato e si sfruttano le componenti logiche per usi molteplici.

Il Software residente e' composto da 6 moduli:

-Interprete BASIC;
-Routines matematiche;
-Routines per la gestione del video;
-Routines per scansione tastiera e codifica dei dati;
-Programmi di utilita' in linguaggio macchina;
-Programma per l'inizializzazione e la partenza a freddo.

L'Interprete BASIC oltre a molte delle caratteristiche proprie dei Personal comprende anche istruzioni speciali per controllare la grafica e la generazione di suoni, per collegarsi ai programmi di utilita' in linguaggio macchina, per facilitare la stesura e la correzione dei programmi.
Inoltre per rendere il piu' veloce possibile l'esecuzione dei programmi BASIC il sistema trasforma durante la fase di scrittura le istruzioni in un meta-linguaggio (chiamato codice RUN-TIME) in modo che le routine interpretative risultino piu' veloci in fase escutiva.
Le Routine matematiche sono divise in due moduli:

-Modulo per i calcoli con i numeri interi;

-Modulo per i calcoli con i numeri floating-point.

Il primo modulo serve per le operazioni aritmetiche di base mentre il secondo serve anche per le funzioni trascendenti. I calcoli con i numeri interi danno la precisione di 9 cifre, mentre quelli in floating-point danno la precisione di 6 cifre. Il modulo matematico consente di trattare numeri floating-point nell'intervallo + o - 10 elevato a -18 e + o -10 elevato a +18. Quando si inserisce la scheda per il calcolo scientifico le routine matematiche la usano automaticamente per i calcoli.

Le Routine per la gestione del video provvedono a manipolare i dati in memoria nel modo richiesto da ciascun Mode scelto. Esse inoltre servono per il cambio dei colori, per le istruzioni grafiche (DOT, FILL, DRAW) e per tutte le operazioni inerenti al video.
La tastiera del DAI e' formata da 56 tasti collegati ordinatamente con una matrice 8 x 7. Le Routine per la scansione della tastiera e la codifica dei dati scandiscono la tastiera ad intervalli prefissati, individuano i tasti che sono stati premuti e codificano ognuno di essi in accordo con una tabella di corrispondenze. Sfruttando queste caratteristiche di costruzione del DAI e' possibile sostituire la tastiera con un'altra, di diversa configurazione usando la corrispondente tabella dei codici.

I Programmi di Utilita' in linguaggio macchina sono un insieme di sottoprogrammi per la tastiera e per specifiche funzioni, che rendono piu' facile generare programmi in linguaggio macchina, caricarli, ricercarne gli errori ed eseguirli. I sottoprogrammi di controllo e di inizializzazione di questo modulo consentono di interfacciare direttamente programmi in BASIC e programmi in linguaggio macchina. Un programma BASIC puo' chiamare il numero desiderato di sottoprogrammi in linguaggio macchina, senza alcuna limitazione.
Il Programma per l'inizializzazione e la partenza a freddo e' formato da un gruppo di sottoprogrammi, che vengono usati anche dagli altri moduli, e che provvedono, tra l'altro, al controllo dell'utilizzo dei blocchi di memoria. Questa caratteristica del sistema consente al microprocessore 8080A di lavorare indirizzando 72K bytes di memoria invece degli usuali 64 K.

1.1. ELENCO DELLE CARATTERISTICHE GENERALI

1.1.1. MICROCALCOLATORE

-Microprocessore 8080A a 2MHz.
-48K di memoria RAM

-24K di memoria PROM/ROM (gestita a blocchi dal software)
-Memoria per Input/Output
-Chip matematico logico AMD 9511 (opzionale)
-Generatore hardware dei numeri casuali
-Gestione automatica dello Stack.

1.1.2. DISPOSITIVI DI INPUT/OUTPUT

-Tastiera ASCII
-Collegamento tramite ingresso d' antenna ad un televisore a colori (PAL) o bianco/nero
-Canale sonoro audio modulato col segnale TV
-2 interfacce per registratori a cassette Audio di basso costo, con controllo di start e stop
-Canali di output stereo hi-fi
-2 interfacce per paddles
-Bus di interfaccia (DAI DCE-BUS) per collegare:
  -Controller per floppy disk
  -Controller per stampante
  -Scheda di interfaccia standard (DAI-RWC) per:
    -adattatore per bus IEEE
    -collegamenti per comunicazioni
    -collegamenti per controlli
    -programmazione PROM
    -interfaccie speciali
    -input e output analogico.
-Interfaccia RS232 con:
  -Baud rates programmabili
  -Funzioni per terminali o modem.

1.1.3. CARATTERISTICHE GRAFICHE

-Formattazione dello schermo (normalmente 66 caratteri,dei quali 3 di margine a destra e 3 a sinistra, per 24 linee, oppure 11/22/44/66 caratteri per da 13 a 32 linee)
-16 colori o toni di grigio
-3 possibilita' di risoluzione grafica (selezionabile via software):
- 65 x 88 punti
- 130 x 176 "
- 260 x 352 "
(E' possibile anche avere contemporaneamente 4 linee di testo e la restante parte dello schermo dedicata alla grafica).

1.1.4. CARATTERISTICHE MUSICALI

-3 generatori di suono programmabili indipendenti
-1 generatore di rumore bianco

-Ampiezza e frequenza selezionabili via software:
        -smooth music
        -frequenze casuali
        -inviluppo del suono
        -generazione di suoni vocali.

.

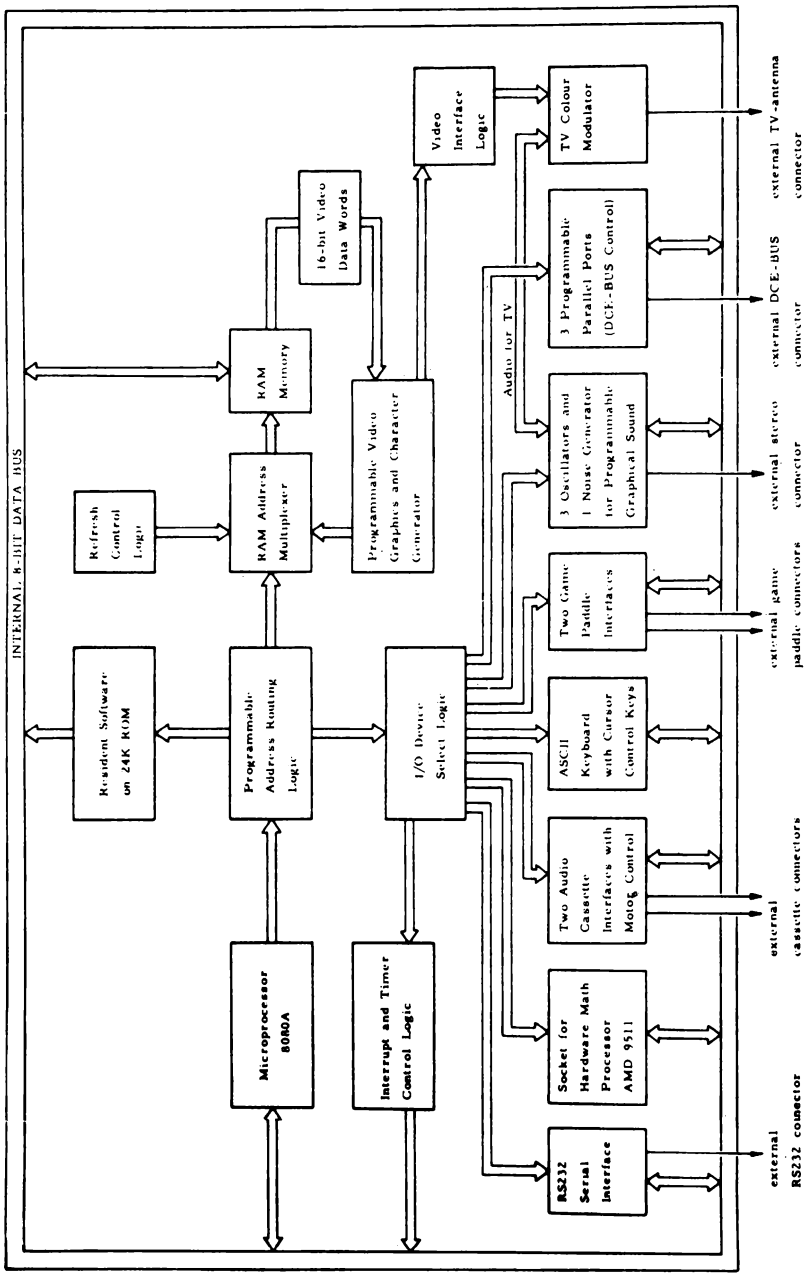
1.1.5. SOFTWARE RESIDENTE

-Interprete BASIC esteso ad alta velocita'
-Modulo matematico completo per operazioni floating-point
-Funzionamento automatico del modulo scientifico se presente
-Comandi grafici per il video:
            -capacita' di plotting complete
            -indirizzamento di singoli punti
            -riempimento di rettangoli ovunque.
-Comandi sonori:
            -specifiche predeminate di volume e inviluppo
            -specifiche individuali di frequenza
            -specifiche individuali di volume
            -specifiche individuali di tremolo
            -specifiche individuali di glissando.
-Programmi di utilita' in linguaggio macchina.

1.1.6. SOFTWARE DI SISTEMA COMPATIBILE

-Assembler DAI
-Supporto Software standard 8080A
-Compilatore FORTRAN
-Supporto software MDS/Intellec non-disc.

1.1.7. DIAGRAMMA FUNZIONALE A BLOCCHI



CAPITOLO 2

I L M I C R O C O M P U T E R

2.1. INTRODUZIONE

Il processore del Personal DAI e' stato progettato sfruttando le caratteristiche del Microprocessore 8080A. Alla base del progetto sta l'architettura del microcalcolatore DCE che fornisce alte prestazioni a basso costo. Il modulo del microcalcolatore e' formato dal microprocessore e dai circuiti di temporizzazione dalla ROM e dalla memoria statica RAM dai Controlli Logici degli Interrupt e dei Timer dalla memoria principale RAM. La memoria principale RAM e' di tipo dinamico ed e' di 48K bytes.

2.2. USO DELLA MEMORIA

La memoria del DAI e' organizzata in modo che per le operazioni di Input e Output vengono usati normali indirizzi di memoria RAM e ROM, come per tutte le altre operazioni del sistema. Nel seguito gli indirizzi vengono riferiti come numeri esadecimali (base 16), in modo che per i 64 kilobytes indirizzabili gli indirizzi variano da 0000 a FFFF. Per i blocchi intercambiabili si ha una ripetizione degli indirizzi.

MAPPA DELLA MEMORIA

0000 - 003F	VETTORE INTERRUZIONI AUTOMATICHE
0040	IMMAGINE CONTROLLO OUTPUT
0041 - 0061	AREA LAVORO PROGRAMMI UTILITA'
0062 - 0071	VETTORE INTERRUZIONI PROGRAMMI UTILITA'
0077 - 00CF	VARIABILI SCHERMO
00D0 - 00FF	AREA LAVORO MODULO MATEMATICO
0100 - 02EB	VARIABILI BASIC
02EC - FINE RAM	STRINGHE E VARIABILI CON INDICE (AREA HEAP)
	PROGRAMMA
	TABELLA DEI SIMBOLI
	MEMORIA RAM NON USATA
	MEMORIA SCHERMO
F800 - F8FF	AREA STACK

Il sistema si serve inoltre delle seguenti variabili a due bytes (gli indizizzi sono memorizzati byte basso-byte alto secondo la logica 8080A):

IND.(esad.)	VARIABILE
029B	INIZIO ZONA STRINGHE E VAR. CON INDICE
029D	DIMENSIONI ZONA PRECEDENTE
029F	INIZIO BUFFER PROGRAMMA
02A1	FINE BUFFER PROGR. E INIZIO TABELLA SIMBOLI
02A3	FINE TABELLA SIMBOLI
02A5	LIMITE INFERIORE MEMORIA RAM VIDEO

2.3. TIMERS E CONTROLLO DELLE INTERRUZIONI

Il DAI dispone di 5 timers programmabili da 64 microsecondi a 16 millisecondi, 2 segnali di interruzione esterni e 2 segnali di interruzioni per I/O seriale. Le priorita' sono state codificate con un sistema di maschere e questo permette di adottare un sistema di interruzioni automatiche o a scansione.

2.3.1. CONTROLLO INTERRUZIONI

Agli 8 indirizzi del vettore delle interruzioni gestito dal microprocessore 8080 sono assegnate le seguenti funzioni:

INDIRIZZO (esad.)	FUNZIONE
00	Timer 1
08	Timer 2
10	Interruzione esterna
18	Timer 3
20	Buffer ricevente pieno
28	Buffer trasmittente vuoto
30	Timer 4
38	Timer 5 / Interruzione ausiliaria

Il segnale di interruzione esterno e' collegato ad un segnale che indica che e' stato selezionato un indirizzo compreso tra F000 e F7FF. Questo normalmente significa che si e' superata la capacita' dello STACK.

Il segnale di interruzione ausiliaria e' collegato ad un segnale di pagina da parte della logica dello schermo. Questo fornisce un comodo orologio con un intervallo di 20 millisecondi. Piu' complete informazioni sulla logica delle interruzioni escono dagli scopi di questo manuale; chi fosse interessato all'argomento puo' leggere il manuale DAI "DCE MICROCOMPUTER SYSTEMS DESIGNER'S HAND BOOK".

2.4. MEMORIA PRINCIPALE RAM

La memoria principale RAM e' divisa in 3 blocchi, chiamati: A, B, C. Ogni blocco comprende 16 K di memoria RAM dinamica.

L'indirizzamento alla RAM e' controllato da una PROM programmata. La memoria R AM e' vista dal programma come un unico blocco che parte dall'indirizzo 0000 ed arriva a BFFF.

Il primo blocco di RAM parte dall'indirizzo 0000 e puo' essere usato solo per i programmi, cioe' non puo' contenere dati per lo schermo. Gli altri due blocchi di RAM sono accessibili al controller dello schermo che li usa in questo modo: preleva dal blocco B gli 8 bits meno significativi (low byte) e dal blocco C gli 8 bits piu' significativi (high byte) per formare le parole di 16 bits (2 bytes) di cui ha bisogno. Per il processore gli indirizzi pari dei bytes sono nel blocco B e gli indirizzi dispari nel blocco C. Per esempio l'indirizzo 4000 sta nel blocco B e l'indirizzo 4001 sta nel blocco C, e cosi' via fino alla fine della RAM.

2.4.1. ROM PROGRAMMABILE (PROM)

La PROM programmata seleziona i seguenti indirizzi :

RAM	Ind. blocchi B + C	Ind. blocco A
48K	4000 - BFFF	0000 - 3FFF

2.4.2. CONFIGURAZIONE RAM PRINCIPALE PER USO GRAFICO

Memoria RAM	Risoluzione grafica	Modes colori	Memoria schermo	Memoria disponibile
48K	65 x 88	4 16	1.5K	46.5K
	130 x 176	4 16	5.8K	42.2K
	260 x 352	4 16	22.8K	25.2K

2.5. ROM E MEMORIA STATICA RAM

Il software di sistema risiede in ROM programmate dall'indirizzo C000 all'indirizzo EFFF. Gli indirizzi da C000 a DFFF si riferiscono a spazi contigui di memoria, mentre per gli indirizzi da E000 a EFFF si hanno 4 blocchi di spazio di memoria scambiabili. Gli indirizzi da F800 a F8FF sono usati per un banco di memoria RAM statica, usata dallo STACK dello 8080A, e per un vettore usato per le istruzioni di salto che serve a emulare un sistema MDS.

2.5.1. MAPPA SEMPLIFICATA DELLA MEMORIA (48K RAM)

INDIRIZZI	FUNZIONI
0000	
029B	Indirizzo inizio stringhe e variabili con indice
029D	Dimensioni zona precedente
029F	Indirizzo inizio Buffer testo
02A1	Indirizzo inizio tabella simboli (fine Buffer testo)
02A3	Indirizzo fine tabella simboli
02A5	Indirizzo termine area schermo
0400	
B350-BFFF	Parte di RAM riservata al video in mode 0
C000-DFFF ROM	Blocchi non scambiabili
E000-EFFF ROM	4 blocchi scambiabili
F000	
F800-F8FF	STACK
FC00-FFFF	Mappa indirizzi I/O

CAPITOLO 3

G R A F I C A

P R O G R A M M A B I L E

3.1. INTRODUZIONE

Il sistema programmabile dei caratteri e della grafica per il video usa dati di lunghezza variabile per sfruttare nel modo piu' efficiente possibile la memoria. E' necessario, prima di procedere, dare alcune definizioni.

SCANSIONE (SCAN): scansione dello schermo da parte del pennello elettronico che sta tracciando il disegno;

LINE: un gruppo di SCAN controllati dalle stesse informazioni nella RAM;

MODE: uno dei diversi modi nei quali le informazioni possono essere visualizzate sullo schermo. Per esempio, in "MODE CARATTERE" i bytes della memoria appaiono come caratteri sullo schermo; nel "MODE 4 COLORI" i bytes descrivono i colori dei BLOBS dello schermo;

BLOB: la piu' piccola area dello schermo che puo' essere gestita con un solo colore (cambia di dimensione nei diversi modi)

CAMPO (FIELD): un gruppo di 8 BLOBS, i cui colori sono controllati da una coppia di bytes in memoria.

Il disegno nasce una linea dopo l'altra sullo schermo; ciascuna linea e' indipendente dalle altre e puo' trovarsi in uno dei Mode possibili. All'inizio di ciascuna linea vengono memorizzati due bytes, che contengono le informazioni circa il Mode della linea e servono per aggiornare i due bytes colore nella RAM. Questi due bytes sono chiamati: CONTROLLO e CONTROLLO COLORE. La parte restante di ogni linea contiene informazioni circa il colore o il carattere, ed il numero di bytes usati dipende dal Mode nel quale si lavora. Lo schermo puo' essere formato da un diverso numero di linee orizzontali. Nel Mode a piu' alta risoluzione sono visibili 352 BLOBS lungo lo schermo. Le due piu' risoluzioni piu' basse hanno rispettivamente 1/2 e 1/4 di questo numero di BLOBS. In una televisione a "625 linee" ci sono circa 520 SCAN visibili, a causa dell'interlacciamento, l'hardware dello schermo puo' solo disegnare due SCAN per ogni linea. Questo consente una risoluzione di 260 x 352 che e' pari al rapporto 3 a 4 delle

dimensioni del video. Cosi' i cerchi risultano rotondi!

I caratteri sono posizionati su questa griglia usando 8 colonne di BLOB per colore, dato che le posizioni dei punti sono definite per ciascun carattere dalla ROM. Questo consente un massimo di 44 caratteri per linea (22/11 nei mode a bassa risoluzione).

Una quarta definizione orizzontale consente un Mode con caratteri ad alta densita' e cioe' 66 caratteri per linea.

Il sistema fornisce in totale 16 colori diversi, compreso il bianco ed il nero. Quando si usa un codice a 4 bit per selezionare il colore, si hanno tutte le possibilita'. In alcuni dei Mode possibili, vengono caricati nei 4 Registri Colore, 4 colori, non necessariamente diversi tra loro. Qualunque codice a 2 bit, che descriva un colore, seleziona uno dei colori caricati in questi registri.

La definizione verticale dipende da un campo di 4 bits nel byte di controllo. Nel modo grafico, questo consente la ripetizione delle informazioni ogni numero pari di scan da 2 a 32. Nel modo carattere, esso definisce il numero di scan necessario per ogni carattere; cosi' la spaziatura verticale sullo schermo puo' variare da una matrice 8 x 7 ad una 8 x 16 consentendo di passare da 35 a 15 linee di caratteri.

Il primo byte della memoria di schermo si trova nel byte piu' alto di un blocco di 8K o di 32K di memoria. Gli altri bytes seguono con indirizzi decrescenti. Sullo schermo appare tutto il contenuto della memoria di schermo, scandendola dal byte piu' alto al byte piu' basso ciclicamente.

3.2. FORMATO DEI DATI NELLA MEMORIA DI SCHERMO

All'inizio di ciascuna linea due bytes rappresentano la parola di controllo della linea stessa. La parola di controllo definisce le modalita' di scansione, la risoluzione grafica orizzontale ed il Mode della linea. Dopo la parola di controllo sono memorizzate un certo numero di parole di dati che rappresentano i colori dei punti o la definizione ed il colore dei caratteri in accordo con il Mode selezionato.

3.2.1. FORMATO DELLA PAROLA DI CONTROLLO

3.2.1.1. BYTE DI INDIRIZZO ALTO (MODE BYTE)

- Bits 7 e 6: Controllo del Display Mode
- Bits 5 e 4: Controllo della risoluzione
- Bits 3, 2, 1 e 0: Contatore della ripetizione delle linee

CONTATORE RIPETIZIONE LINEE

Questo contatore controlla il numero delle scansioni orizzontali durante le quali devono essere mostrati gli stessi dati. Dato che l'interlacciamento della scansione TV non e' considerato, un minimo di due scansioni corrisponde al valore zero di questo contatore. Ogni incremento di questo contatore aggiunge 2 scansioni alla linea. Il massimo spessore programmabile per ogni segmento orizzontale e' di 32 scansioni

CONTROLLO DELLA RISOLUZIONE

I bits per il controllo della risoluzione permettono di selezionare per ogni linea una tra quattro definizioni orizzontali per il display dei dati sullo schermo.

CODICE	SIGNIFICATO (punti per larghezza schermo)
00	88 grafica a bassa definizione (11 chr per linea)
01	176 grafica a media definizione (22 chr per linea)
10	352 grafica ad alta definizione (44 chr per linea)
11	528 testo con 66 caratteri per linea

CONTROLLO DEL DISPLAY MODE

I bits del controllo del Mode determinano come devono essere usati i dati per generare il disegno di un particolare segmento.

CODICE	DISPLAY MODE
00	Grafica a 4 colori
01	Caratteri a 4 colori
10	Grafica a 16 colori
11	Caratteri a 16 colori

3.2.1.2. BYTE DI INDIRIZZO BASSO (TIPO COLORE)

Questo byte viene usato per memorizzare i colori nei 4 Registri Colore per il Mode a 4 colori. Qualunque colore, tra i 4, puo' essere modificato all'inizio di ogni linea. Il Mode puo' usare solamente i colori che sono memorizzati in questi registri. La scelta dei colori e' completamente libera tra i 16 possibili.

Significato dei diversi bits:

7	se e' in ON consente di operare il cambio del colore, se in OFF non vengono considerati i bits da 0 a 5
6	se in OFF forza il Mode Unit Color (vedere 3.2.2.4.)
5 e 4	selezionano uno dei 4 registri colore
3, 2, 1 e 0	selezionano uno dei 16 colori

TABELLA DEI COLORI

CODICE	COLORE CORRISPONDENTE
0	nero
1	blu scuro
2	rosso porpora
3	rosso
4	marron porpora
5	verde smeraldo
6	marron kaki
7	marrone mostarda
8	grigio
9	blu medio
10	arancione
11	rosa
12	blu brillante
13	verde brillante
14	giallo brillante
15	bianco

3.2.2. MODE DATI

3.2.2.1. MODE 4 COLORI

In questo mode sono necessari solo due bits dei dati per definire il colore di un punto. Questi bits si ottengono in parallelo usando il bit di ordine piu' alto del byte alto e del byte basso di ogni parola di dati. I due bytes di un campo sono considerati come 8 coppie di bits. Ciascuna coppia definisce il colore di un punto, selezionandolo tra i 4 colori disponibili nei Registri Colori gia' sistemati per quella linea. Nella linea seguente i colori possono essere modificati.

```
BYTE ALTO
****************************************
* B7 *    *    *    *    *    * B0 *    A1
****************************************

BYTE BASSO
****************************************
* B7 *    *    *    *    *    * B0 *    A0
****************************************
```

Nello schema A1 e A0 sono la coppia di byte usati per indirizzare i colori nella memoria RAM.

3.2.2.2. MODE 16 COLORI

Abbiamo gia' visto che questo Mode e' organizzato in modo da usare la meta' della memoria che sarebbe necessaria, e che e' usata in altri sistemi.

L'organizzazione di base e' che il byte basso seleziona due fra i 16 possibili colori:

- i bits da 0 a 3 danno il colore dello sfondo (Background)
- i bits da 4 a 7 danno il colore del testo (Foreground)

Il byte alto seleziona poi con ogni bits se il colore del BLOB corrispondente deve essere l'uno o l'altro: 0 corrisponde allo sfondo, 1 corrisponde al testo.

```
BYTE ALTO
****************************************
* B7 *    *    *    *    *    * B0 *
****************************************

blob di    bit 1       bit 0      blob di
sinistra  foreground  background   destra

BYTE BASSO
****************************************
* B7 *    *    *    *    *    * B0 *
****************************************

colore foreground   / colore background
```

I colori scelti per ogni campo sono indipendenti da quelli scelti per il campo precedente; cosi', pur avendo una limitazione all'interno dei campi, in una linea si possono avere tutti i colori. In questo Mode non ha significato il contenuto dei registri colore.

3.2.2.3. MODE CARATTERI

In questo Mode i caratteri sono generati tramite un generatore di caratteri situato in ROM e il loro colore e' definito mediante i 4 registri colore o i colori a coppie del Mode a sedici colori.

All'accensione la matrice dei caratteri e' di 6x9 bit, ma e' possibile arrivare a 8x16 bit. Pertanto il contatore della ripetizione delle linee dovrebbe essere almeno a 10 per garantire una buona visione dei caratteri tenendo conto anche delle linee di spaziatura.

In memoria ogni carattere occupa 2 byte: nel byte alto e' contenuto il codice ASCII del carattere stesso mentre il byte basso viene utilizzato per la gestione del colore, in modo diverso, a seconda che ci si trovi in modo a 4 o a 16 colori. In modo a 4 colori ogni bit seleziona una delle due coppie sfondo/testo definite nella COLORT: 0 indica la prima coppia, 1 indica la seconda, cosi', in un segmento orizzontale un carattere puo' essere in ognuna delle due possibili combinazioni carattere/sfondo, ed essere controllato lungo una verticale dal byte basso. Nel modo a 16 colori i primi 4 bit definiscono il colore del carattere, mentre gli ultimi 4 definiscono il colore dello sfondo per quello stesso carattere.

Nella tabella che segue sono riportati gli indirizzi e i contenuti dei byte usati come registri colore:

COLORE (COLORT S T SA TA)	IND. BYTE	CONTENUTO
S	BFFE	9X
T	BFFA	9X
SA	B35E	8X
TA	B35A	8X

dove X e' un qualsiasi numero compreso tra 0 e F e serve per individuare il colore.

In questo modo l'altezza del carattere dipende da un certo numero di scan orizzontali. La larghezza del carattere dipende dalla definizione selezionata nel byte di controllo. La definizione a 352 consente 44 caratteri per linea, mentre quella a 528 consente i normali 66 caratteri per linea. Altre definizioni possono consentire una maggior larghezza dei caratteri, come la scritta che appare al momento dell'accensione, ma non sono gestibili dal BASIC in modo diretto.

Vediamo alcuni esempi:

POKE #BB39,#6A cambia il numero dei caratteri da 66 a 44 nella decima linea del video.

POKE #BB39,#5A cambia il numero dei caratteri da 44 a 22

POKE #BB39,#4A cambia il numero dei caratteri da 22 a 11

E' possibile eliminare l'effetto della griglia verticale mettendo nel terzo e quarto registro della COLORT lo stesso colore dello sfondo.

Caratteri speciali:

CR Termina una linea di caratteri e posiziona il cursore all'inizio della prossima linea. Se necessario produce lo "scrolling" del video.

FF Riempie l'area video di spazi e posiziona il cursore in alto a sinistra.

BS Cancella il carattere scritto per ultimo e sposta il cursore all'indietro.

Una linea di caratteri puo' continuare per 4 righe dello schermo, le 3 linee di continuazione hanno pochi caratteri in meno ed un C sulla prima posizione. Se la terza linea di continuazione e' piena vengono accettati solo i caratteri: CR, FF, BS.
Se si tenta di usare il BS all'inizio di una linea esso viene ignorato.

CAMBIO DEL COLORE DELLO SFONDO O DI UNA LETTERA IN UNA LINEA

Il byte di controllo della linea 1 si trova all'indirizzo BFEF e il byte di controllo del colore della linea 1 si trova all'indirizzo BFEE. Il byte del primo carattere della linea 1 si trove all'indirizzo BFEF-2 ed il byte del controllo del colore del primo carattere all'indirizzo BFEF-3. Ciascuna delle 66 posizioni della linea dello schermo corrisponde ai seguenti indirizzi:
il byte del carattere BFEF - (2 x posizione carattere nella linea)
il byte di controllo del colore per il carattere BFEF - (3 x posizione carattere nella linea).
Bisogna tener presente che anche se ci sono 66 caratteri nella linea, in realta' essi sono 60 perche' i primi 3 e gli ultimi 3 sono lasciati in bianco per il margine. Pertanto il byte di controllo della linea successiva si trova 134 bytes (esadecimale #86) piu' indietro del precedente. Per la

seconda linea tale indirizzo e' #BFEF - #86 = #BF69.

Esempi:

Byte di controllo linea 1: ind. #BFEF

Byte di controllo linea 5: ind. #BFEF - (#86*5) = #BDD7

Byte controllo colore linea 5: ind. #BDD6

Sesto carattere della linea 5: ind. #BDD7 - 6*2 = #BDCC

Byte di controllo colore del sesto carattere della linea 5: ind. #BDCB

Nei vostri programmi potete usare il comando POKE per cambiare il colore dello sfondo o dei caratteri ed il programma UTILITY 3 per trovare la locazione su cui lavorare con POKE. Quando tornate al BASIC le modifiche di colore operate sotto UTILITY vengono annullate se digitate: MODE 1, RETURN, MODE 0.

Esempio 1:

```
COLORT & 0 5 10
POKE #BA2D,#DA
POKE #BA2D,#C3
```

il primo comando POKE cambia il colore dei caratteri dal nero al colore 10 nella linea 12; il secondo comando POKE cambia lo sfondo da & a 3. Le locazioni da #B350 a #B35F controllano il colore dello sfondo e le locazioni da #BFF0 a #BFFF controllano il colore del testo.

Esempio 2:

```
COLORT 0 15 7 &
POKE #735A,#90:POKE #7FFA,#90
POKE #735E,#80:POKE #7FFE,#80
```

e vedrete lo schermo nero e i caratteri neri. I numeri #80 e #90 possono essere sostituiti da qualunque numero tra #80 e #8F e tra #90 e #9F.

Esempio 3:

Cambio del colore dello sfondo e del testo.

```
10 MODE 0
15 REM PARTE DA #BEE2
20 COLORT 3 0 5 15
25 FOR A% = 1 TO 23:PRINT A%,
26 FOR B = 0.0 TO 40.0:PRINT "+";
27 NEXT:PRINT:NEXT
30 REM TROVERETE LINEA 1-2 TESTO 0 SFONDO 8
35 POKE#BEE2,#CF:REM   3-7       0        15
40 POKE#BC44,#DF:REM   8-9      15        15
45 POKE#BB38,#D8:REM  10         8        15
50 POKE#BAB2,#D0:REM  11-12      0        15
60 POKE#B9A6,#DF:REM  13-14     15        15
70 POKE#B89A,#D5:REM  15         5        15
90 POKE#B814,#D0:REM  16         0        15
92 POKE#B78E,#DF:REM  17-18     15        15
93 POKE#B682,#C6:REM  19-21     15         6
94 POKE#B4F0,#C8:REM  22-24     15         8
95 GOTO 95
```

Esempio 4:

Cambio del colore dello sfondo e del testo

```
10 E%=#FF
20 COLORT 8 0 0 8
30 B%=#BFEF
40 FOR A%=1 TO 23
50 D%=B%-3
60 FOR C%=0 TO 65
70 POKE D%,E%
80 D%=D%-2:NEXT
90 B%=B%-#86:NEXT
93 E%=INOT E% IAND #FF
95 GOTO 30
```

TABELLA INDIRIZZI RAM SCHERMO

Num.Linea	Primo byte	Byte controllo colore
1	BFEF	BFEE
2	BF69	BF68
3	BEE3	BEE2
4	BE5D	BE5C
5	BDD7	BDD6
6	BD51	BD50
7	BCCB	BCC4
8	BC45	BC44
9	BBBF	BBBE
10	BB39	BB38
11	BAB3	BAB2
12	BA2D	BA2C
13	B9A7	B9A6
14	B921	B920
15	B89B	B89A
16	B815	B814
17	B78F	B78E
18	B709	B708
19	B683	B682
20	B5FD	B5FC
21	B577	B576
22	B4F1	B4F0
23	B46B	B46A
24	B3E5	B3E4

3.2.2.4. Modo a colore unico

Questo modo puo' essere usato per risparmiare spazio durante la scansione orizzontale del disegno. Si puo' disegnare una banda colorata sempre dello stesso colore usando solo una parola per il controllo ed una parola per i dati.

Per un' occupazione totale dello schermo bastano 40 bytes di memoria RAM.

3.3. INTERFACCIA VIDEO

L'interfaccia per il televisore e' realizzata in modo che un modulo adattatore per il PAL si innesta nella normale logica usata per realizzare il bianco e nero. Altre interfaccie video sono facilmente realizzabili usando un adattatore.

CAPITOLO 4

P O G R A M M A Z I O N E
D E L G E N E R A T O R E
D I S U O N I

4.1. INTRODUZIONE

Il generatore di suoni del DAI e' molto versatile, infatti qualsiasi frequenza e' generata da oscillatori digitali che consentono risultati precisi. Esiste inoltre un generatore random di rumore e si puo' controllare il volume con segnali digitali.

4.2. OSCILLATORI PROGRAMMABILI

La generazione dei suoni si ottiene per mezzo di 3 oscillatori indipendenti programmabili e di un generatore random di rumore. Ciascun oscillatore e' collegato al microprocessore come un qualunque dispositivo di I/O ed e' programmabile a qualunque frequenza compresa tra 30 HZ e 1 MHZ. Ovviamente le alte frequenze non sono molto interessanti per l'audio, ma dal momento che i segnali dei 3 oscillatori vengono sommati insieme prima della modulazione al canale audio della TV si possono ottenere effetti interessanti mescolando insieme varie possibilita'. Gli oscillatori programmabili sono usati per la generazione dei suoni e come interfaccia per i giochi.

4.2.1. SELEZIONE DELLE FREQUENZE

Per programmare una frequenza per un canale si deve posizionare un numero di 16 bits in uno dei seguenti indirizzi:

Canale Oscillatore	Indirizzo
1	FC00 o F001
2	FC02 o F003
3	FC04 o F005

Prima di inviare una frequenza ad un canale il byte di indirizzo FC06 deve contenere uno dei seguenti numeri di 8 bits:

1	36 esadecimale
2	76 esadecimale
3	B6 esadecimale

Il numero di 16 bits necessario per determinare la frequenza e' inviato con due trasferimenti di dati di 8 bits ciascuno, inviando prima i bits meno significativi.

4.2.2. CONTROLLO DEL VOLUME

L'ampiezza in uscita di ogni oscillatore e quella del generatore di rumore sono controllabili in modo digitale scrivendo una parola di controllo all'indirizzo specificato nel paragrafo che tratta l'allocazione di memoria per i dispositivi di I/O.

4.3. GENERATORE RANDOM DI RUMORE

Fra i circuiti generatori di suoni e' incluso un circuito generatore di rumore. Lo scopo e' quello di generare rumori molto simili ai rumori bianchi per poter ottenere suoni compositi complessi e per fornire un generatore temporizzato di sequenze di numeri casuali. Gli eventi generati a casualmente da questo circuito danno la base di informazioni ad una porta di I/O per produrre un vero numero casuale

4.4. MESCOLANZA DI FREQUENZE

Le uscite dei canali sonori e del generatore di rumore vengono mescolate prima della modulazione al canale audio. I canali 1 e 2 e i canali 2 e 3 sono mescolati per le uscite stereo sinistra e destra. Nella configurazione stereo il rumore e' inserito nei canali 1 e 3.

4.5. FORMULA PER IL CALCOLO DELLA FREQUENZA

Per ottenere una frequenza di nHZ da un oscillatore, lo si deve programmare con un numero uguale a 2 moltiplicato 10 elevato alla sesta e poi dividere il risultato per n. Esiste la funzione BASIC FREQ per fare questo calcolo.

CAPITOLO 5

S E Z I O N E

I N P U T / O U T P U T

5.1. INTRODUZIONE

Le operazioni di I/O del DAI si ottengono con la tecnica del posizionamento in indirizzi fissi di memoria (Memory Mapped). In tale modo tutto l' I/O e' direttamente accessibile al BASIC. Naturalmente si devono programmare con cura i comandi POKE per non creare conflitti con l'interprete BASIC.

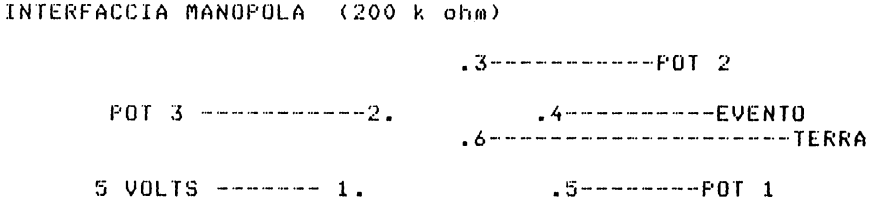
5.2. INTERFACCIA PER I GIOCHI

Il DAI e' fornito dei circuiti necessari per poter collegare due manopole come apparecchiature di input per i giochi. Ogni manopola contiene tre resistori variabili la cui posizione viene letta come valore e come stato di ON o di OFF.

La posizione di ogni resistore si trova ponendo il suo indirizzo binario in 3 bit nella porta FD06. Il canale 0 del generatore di suoni opera allora come un contatore. La lettura della posizione si ottiene nella locazione FD01. Il valore letto viene reso disponibile come un dato di 8 bits.

SCHEMA DI CONNESSIONE (PRESA DIN A 6 POLI/ 240 gradi-Schema visto dall'interno della presa)

INTERFACCIA MANOPOLA (200 k ohm)



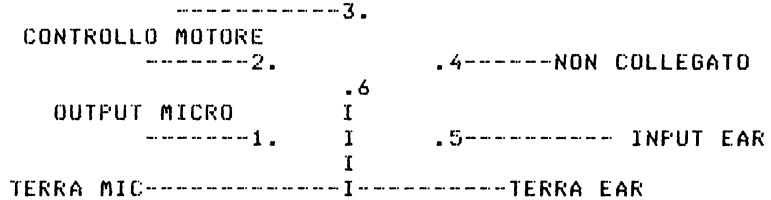
5.3. INTERFACCIA CASSETTE

Il DAI contiene tutti i circuiti necessari per collegare un normale registratore audio di basso costo per l'ingresso e l'uscita di dati e/o programmi su nastro. L' input si ottiene collegando l'uscita EAR oppure i fili

dell'altoparlante all'ingresso del DAI.

SCHEMA DI COLLEGAMENTO (PRESA DIN A 6 POLI/ 240 gradi-Schema visto dall'interno della presa)

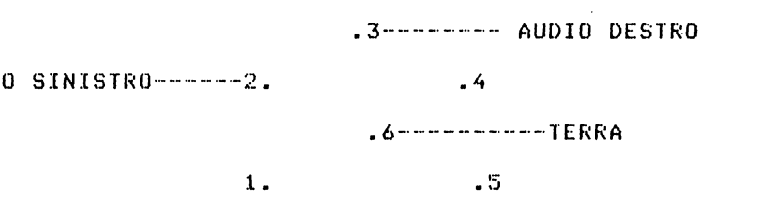
INTERFACCIA REGISTRATORE



5.4. USCITA STEREO

Il generatore di suoni del DAI e' collegabile con i canali stereo destro e sinistro. I canali 0 e 1 ed i canali 2 e 3 sono rispettivamente sommati per ottenere i canali sinistro e destro.

SCHEMA USCITA STEREO



5.5. SCHEDA MATEMATICA

La logica del DAI sopporta come opzione la scheda matematica (Micro 9511). Essa consente calcoli piu' veloci e con maggior precisione di cifre.

La scheda matematica e' considerata una periferica ed ha come indirizzi:

FB00 per i dati
FB02 per i comandi e lo stato.

Il segnale "PAUSE" e' usato correttamente per mettere la

CPU nello stato di attesa di dati. Se si usa la scheda matematica non si possono usare i comandi: SHLD e LHLD.

5.6. TASTIERA ASCII

La tastiera viene scandita come se fosse una matrice di interruttori. La codifica e la gestione dei tasti e' realizzata via software.

5.6.1. SCHEMA DELLA TASTIERA

Schema della assegnazione dei tasti:

	0	1	2	3	4	5	6
0	0	8	CR	H	P	X	freccia su
1	1	9	A	I	Q	Y	freccia giu'
2	2	:	B	J	R	Z	freccia sinistra
3	3	;	C	K	S	[	freccia destra
4	4	,	D	L	T		TAB
5	5	-	E	M	U	SP	CTRL
6	6	.	F	N	V	REPT	BREAK
7	7	/	G	O	W	CHDL	SHIFT

La matrice ha 8 righe, linee di output di indirizzo FF07 e 7 colonne, linee di input di indirizzo FF01.
Nella posizione 4,5 leggasi il simbolo accento circonflesso.

5.6.2. LOGICA DI SCANSIONE DELLA TASTIERA

La scansione e la codifica sono ottenute via software. Queste routine possono anche essere usate per altri scopi e puo' anche essere sostituita la tabella di codifica. Tutte le posizioni vengono scandite ciclicamente e inizia una azione se viene individuata la pressione di un tasto. Se si usa il tasto REPEAT la scansione passa da periodica a alternativa. La velocita' di ripetizione e' fissa.
I codici vengono ottenuti da una tabella. La pressione del tasto SHIFT fa passare da una prima tabella di codifica ad

una seconda tabella di codifica. Posizionando opportunamente un byte di Flag si puo' ottenere che la scansione venga fatta solo per recepire il tasto BREAK, ottenendo ovviamente una maggiore velocita' di risposta.

Al momento dell'accensione del DAI le lettere da A a Z sono maiuscole e se si usa il tasto SHIFT si ottengono le minuscole. Se si preme il tasto CTRL il funzionamento viene invertito rendendo la tastiera simile a quella di una normale macchina da scrivere. L'uso successivo di CTRL inverte il modo maiuscolo-minuscolo.

Consultate la tabella dei codici in 7.3.4..

5.7. DCE-BUS

Il DAI e' collegabile tramite un normale cavo ad un DCE-BUS. La logica del calcolatore consente di trattare questo BUS come un processore DCE con le stesse caratteristiche per quanto riguarda l'indirizzamento, gli interrupt e la gestione. Il BUS-DCE puo' essere collegato direttamente ad apparecchiature esterne.

Nel DAI sono incluse le routine per comunicare con la REAL-WORLD-CARD.

Segue un esempio di routine per gestire una stampante parallela tramite il BUS-DCE.

```
10  CLEAR 1000:REM DOVETE SCEGLIERE VOI
20  DIM PRI(10)
30  INPUT"VOLETE USARE LA STAMPANTE";A$:PRINT
40  IF A$ <>"S"GOTO 100
50  FOR X = #400 TO #419
55  READ C
60  POKE X,C
65  NEXT X
70  POKE #FE03,#AC
75  POKE #2DD,#C3
80  POKE #2DE,#00
85  POKE #2DF,#4
90  DATA 229,213,197,17,2,254,6,16,33,1,254
95  DATA 119,43,54,0,54,1,26,160,194,11,4,193,209,225,201
100 PRINT CHR$(12)
110 IF A$ <>"S" GOTO 200
120 IF A$ = "S" THEN POKE #131,3:REM USCITA AL BUS-DCE
200 ............
```

5.7.1. USCITE DCE-BUS

NOME	DESCRIZIONE	PIN ON REAL WORLD CARD	PIN ON DAI CARD
P0B0	Port 0 Bit 0	24	16
P0B1	1	26	14
P0B2	2		
P0B3	3	28	12
P0B4	4	29	9
P0B5	5	27	11
P0B6	6	25	13
P0B7	7	23	15
P1B0	Port 1 Bit 0	12	30
P1B1	1	10	31
P1B2	2	8	32
P1B3	3	7	25
P1B4	4	9	24
P1B5	5	11	23
P1B6	6	13	22
P1B7	7	15	21
P2B0	Port 2 Bit 0	18	26
P2B1	1	17	27
P2B2	2	16	28
P2B3	3	14	29
P2B4	4	19	20
P2B5	5	20	19
P2B6	6	21	18
P2B7	7	22	17
EXINTR+	Int. Esterno	4	6
IN7+	Inp.Parall. Bit 7(aus.)	3	5
EXRESET	Reset ester.	5	7
+12V	+12V DC	2	2
+5V	+5V DC	1	3
-5V	-5V DC	6	4
INTR	Int.PIN 14 CPU 8080	-	33
IN7+		-	34
Non collegato		-	8

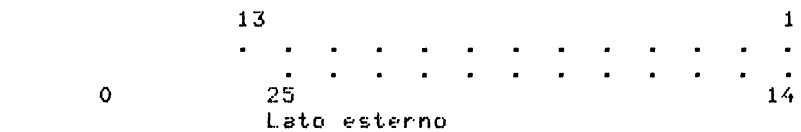
5.8. INTERFACCIA RS232

Il DAI ha una interfaccia RS232 compatibile che fornisce una linea di input seriale, una linea di output seriale ed una linea di stato che serve per fermare l'output (DTR). Queste linee sono disponibili ad un connettore CCITT standard sul retro del calcolatore.

Il segnale DTR consente la sincronizzazione con una stampante. Le locazioni 20 e 28 devono essere messe in ON

per consentire di trasmettere o ricevere i dati. L'interprete BASIC fa uso anche di queste locazioni.

CONNETTORE RS-232 DEL CALCOLATORE



PIN	Funzione
1	Terra
2	Uscita seriale
3	Ingresso seriale
4	Data Terminal Ready
5	+12V
6	+12V
7	Terra
8	+12V
da 9 a 25	non collegati

5.9. INDIRIZZI APPARECCHIATURE I/O

5.9.1. INDIRIZZI APPARECCHIATURE MASTER DI CONTROLLO

Indirizzi	Funzioni
F900-F9FF	disponibili
FA00-FAFF	disponibili
FB00/1	dati scheda matematica
FB02/3	comandi scheda matematica
FC00/1	canale 0 generatore suoni
FC02/3	canale 1
FC04/5	canale 2
FC06/7	comandi generatore suoni
FDXX	vedere 5.9.2.
FE00/1/2	Porte I/O 0/1/2 DCE-BUS
FE03	Comandi porte
FFXX	vedere 5.9.3.

5.9.2. INDIRIZZI APPARECCHIATURE I/O

Indirizzi	Note	IN/OUT	Uso dei bits
FD00	1	IN	0 - 1 - 2 Segnale pagina 3 Output seriale pronto 4 Manopola destra (1=chiuso) 5 Manopola sinistra (1=chiuso) 6 Dati random 7 Input da cassetta
FD01	3	IN	Impulso singolo per temporizzazione manopole
FD04	2	OUT	0 Volume oscillatore 1 1 " 2 " 3 " 4 Volume oscillatore 2 5 " 6 " 7 "
FD05	2	OUT	0 Volume oscillatore 3 1 " 2 " 3 " 4 Volume generatore di rumore 5 " 6 " 7 "
FD06	3	OUT	0 Output cassetta 0 Canale selettore codice manopola 1 " 2 " 3 Bit abilitazione manopola 4 Motore cassetta 1 (0=run) 5 Motore cassetta 2 (0=run) 6 Scambio banchi ROM 7 "

Note:
1) Si puo' leggere o scrivere su ognuna di queste locazioni
2) La lettura non ha alcun effetto sullo stato. La scrittura modifica il volume,ma l'effetto puo' essere modificato dall'interprete BASIC se accede a queste locazioni
3) Non si dovrebbe mai scrivere in queste locazioni

5.9.3. I/O SERIALE, TIMER E INTERRUZIONI

I dispositivi seriali di I/O si basano su un componente LSI. E' bene usare alcune cautele nel trattare questi dispositivi per evitare interferenze con l'interprete BASIC

che usa normalmente queste locazioni.

Indirizzo	Note	Funzione
FF00	1	Buffer input seriale, contiene l'ultimo carattere ricevuto da RS232
FF01	1	Porte input tastiera. I 7 bits piu' alti sono per i dati, il bit 7 e' la linea IN7 dal DCE-BUS ed e' collegato al segnale di pulizia video
FF02	2	Indirizzo registro interruzioni
FF03	1	Registro di stato. Uso dei bits: 7,6 e 5 non usati; 4 buffer trasmissione vuoto, posizionato a 1 se l'RS232 e' pronto ad accettare un carattere di output 3 buffer di ricezione pieno posizionato a 1 se e' stato ricevuto un carattere 2 disfunzione, posizionato a 1 se un carattere e' stato ricevuto fuori tempo 1 errore, posizionato a 1 da un BREAK in input RS232
FF04	2	Registro comandi
FF05	3	Registro comunicazione velocita' RS232: 1/81 110 baud 2/1 stop bit 2/82 150 baud " 4/84 300 baud " 8/88 1200 baud " 10/90 2400 baud " 20/A0 4800 baud " 40/C0 9600 baud "
FF06	3	Output seriale,si puo' scrivere in questa locazione per mandare un carattere in output su RS232. Usare solo quanda il bit 4 dello indirizzo FF03 e' alto
FF07	4	Porta output tastiera, non usato dall'utente
FF08	2	Registro per mascherare interruzioni
FF09	2	Indirizzi Timer
FF0A	2	"
FF0B	2	"
FF0C	2	"
FF0D	2	"

Note:

1) Puo' essere letto ma non scritto dall'utente
2) L'utente non dovrebbe usarlo.
3) Puo' essere solo scritto dall'utente.
4) Non puo' essere letto ed e' pericoloso scriverlo.

CAPITOLO 6

S O F T W A R E
R E S I D E N T E

6.1. INTRODUZIONE

Il software residente e' formato dai seguenti moduli:

- Interprete BASIC;
- Programmi Utilita' in Linguaggio Macchina;
- Modulo Generale di Inizializzazione e Partenza.

Di norma questi Moduli lavorano insieme e consentono di usare il Basic e le cassette. Per i programmi in linguaggio macchina si usano anche altri moduli, sotto forma di sottoprogrammi.

6.2. BASIC DAI RESIDENTE

6.2.1. INDICE ALFABETICO FRASI DAI BASIC

6.2.1.1. COMANDI BASIC

LIST	6.2.5.3.	WAIT	6.2.6.11.
LOAD	6.2.9.2.	UT	6.2.7.7.

6.2.1.2. FUNZIONI BASIC

ABS	6.2.14.1.	LOG	6.2.14.15.
ACOS	6.2.14.2.	LOGT	6.2.14.16.
ALOG	6.2.14.3.	MID$	6.2.14.17.
ASC	6.2.14.4.	PDL	6.2.7.4.
ASIN	6.2.14.5.	PEEK	6.2.7.5.
ATN	6.2.14.6.	PI	6.2.14.18.
CHR$	6.2.14.7.	RIGHT$	6.2.14.19.
COS	6.2.14.8.	RND	6.2.14.20.
CURX	6.2.12.10.	SCRN	6.2.12.8.
CURY	6.2.12.11.	SGN	6.2.14.21.
EXP	6.2.14.9.	SIN	6.2.14.22.
FRAC	6.2.14.10.	SPC	6.2.14.23.
FRE	6.2.11.3.	SQR	6.2.14.24.
FREQ	6.2.13.5.	STR$	6.2.14.25.
GETC	6.2.8.2.	TAB	6.2.14.26.
HEX$	6.2.14.11.	TAN	6.2.14.27.
INP	6.2.7.2.	VAL	6.2.14.28.
INT	6.2.14.12.	VARPTR	6.2.11.5.
LEFT$	6.2.14.13.	XMAX	6.2.12.6.
LEN	6.2.14.14.	YMAX	6.2.12.7.

6.2.1.3. OPERATORI LOGICI E ARITMETICI

Essi sono:

+	-	*	/	MOD	freccia su	=	<	>	<=
IOR	IAND	IXOR	INOT	SHL	SHR	AND	OR		

6.2.2. REGOLE DI FORMATO E LIMITAZIONI

6.2.2.1. VARIABILI E COSTANTI NUMERICHE

Il BASIC DAI consente due tipi di valori numerici: interi e reali (floating-point). I numeri interi sono compresi tra (-2 elevato a 32) - 1 e (+2 elevato a 32) -1; si hanno cioe' circa 9 cifre di precisione. I numeri interi sono esatti e non sono arrotondati. I numeri reali sono compresi tra 10 elevato a -18 e 10 elevato a +18, con 6 cifre stampabili (formato floating-point a 32 bits).
I diversi comandi BASIC possono lavorare o con numeri interi o con numeri reali. Per esempio:

a) DRAW A,B C,D X aspetta di ricevere A,B,C,D e X come

numeri interi;
b) LET A = SQRT(B) aspetta B come un numero positivo reale.

Il BASIC adotta le seguenti regole:
1) Se trova un numero reale, dove aspettava un numero intero, tronca i decimali;
2) Se trova un numero intero, dove aspettava un numero reale, lo trasforma automaticamente in reale;
3) Se si scrive un numero intero (ad esempio 3 invece di 3.0) esso viene codificato come intero o reale a seconda del contesto in cui si trova o se non c'e' una possibilita' di riconoscimento (come nel comando PRINT) come se si fosse usato il comando IMP.

I nomi delle variabili possono essere formati da 1 a 14 caratteri, il primo dei quali deve essere alfabetico, e gli altri o alfabetici o numerici. Eventuali caratteri dopo il quattordicesimo sono ignorati. Se non si aggiunge il suffisso ($, %, !) esso viene aggiunto se il tipo dipende dal comando IMP. Inizialmente tutte queste variabili sono floating-point.
Le variabili numeriche possono riferirsi a numeri interi o reali. I nomi delle variabili intere terminano con il carattere %; quelli delle variabili floating-point con il carattere !. Le variabili stringa terminano con il carattere $. Bisogna considerare gli esempi per capire come agisce il comando IMP.

Esempi:
Inizialmente si ha:

```
I, A, S       sono variabili reali, perche' sono
              l'abbreviazione rispettivamente di I!, A!, S!;

I%, A%, S%    sono variabili intere e sono distinte da
              I, A, S;

I$, A$, S$    sono variabili stringa.
```

Dato che non si e' usato il comando IMP, nelle variabili reali puo' essere omesso il suffisso !, mentre si deve usare per le altre il suffisso % o $. Se si scrive:

```
IMP INT I-N
IMP STR S-S
```

```
allora I rappresenta una abbreviazione di I%;
       A rappresenta una abbreviazione di A!
       S rappresenta una abbreviazione di S$.
```

Pero' una qualunque variabile recante il suffisso, non viene influenzata dal comando IMP.

Quando viene listato un programma, il sistema usa la forma abbreviata per i nomi delle variabili. Cioe', dopo l'ultimo esempio, si vedrebbe stampato: I, A, S.

Se si usa un comando del tipo: IMP INT oppure IMP FPT, senza specificare alcuna lettera, vengono considerate di quel tipo tutte le variabili. Inoltre i numeri interi, come per esempio 3, sono interpretati di conseguenza.

Comando	Corrisponde a	Numero	Variabile
IMP INT	IMP INT A - Z	intero 3	A%
IMP FPT	IMP FPT A - Z	reale 3.0	A!
IMP STR	non consentito	----	----

All'accensione il sistema lavora come se si fosse dato il comando IMP FPT.

6.2.2.2. STRINGHE

1) Una stringa puo' contenere da 0 a 255 caratteri.

2) Le matrici di stringhe possono essere dimensionate esattamente come le matrici numeriche. Per esempio, DIM A$(10,10) crea una matrice di stringhe di 121 elementi, e, precisamente, di 11 righe e di 11 colonne (gli indici cominciano dal valore 0). Ciascuna stringa della matrice puo' avere da 0 a 255 caratteri e gli elementi possono essere di lunghezze differenti.

3) Il numero totale dei bytes necessari per i caratteri delle stringhe memorizzate e per i controlli associati non puo' mai, durante l'esecuzione di un programma, superare il blocco di memoria assegnato dal sistema allo scopo. Se questo succede si ha un messaggio di errore. Dal momemto che il sistema alloca spazio di memoria per le stringhe in modo dinamico, puo' capitare che un programma venga usato piu' volte senza dare questo tipo di errore e poi in una esecuzione, durante la quale si usano stringhe molto lunghe, dia questo errore.

4) Le stringhe non possono contenere il carattere doppio apice (codice esadecimale 22); questo carattere puo' essere ottenuto usando la funzione di stringa CHR$(#22).

Seguono alcuni esempi di uso di stringhe. Ricordatevi di dare, prima di provare il comando CLEAR.

DIM A$(10,10) Predispone l'esistenza di una matrice di stringhe e alloca spazio per un puntatore per ogni elemento della stringa, ma le stringhe restano momentaneamente di lunghezza nulla.

A$ = "FOO" + V$ Assegna il valore della espressione, formata da stringhe, alla variabile stringa A$,

andando ad occupare uno spazio pari ai caratteri della stringa piu' uno.

IF A$ = B$ THEN STOP Rappresenta una comparazione di stringhe. La comparazione avviene carattere per carattere da sinistra verso destra (i caratteri sono ASCII) fino a quando non viene riscontrata una differenza. Se durante una comparazione tra stringhe, si esauriscono i caratteri di una delle due stringhe, prima di riscontrare una disuguaglianza, la piu' corta e' considerata minore dell'altra. Notate che "A " e' considerata maggiore di "A", cioe' contano gli spazi di riempimento.

INPUT X$ Legge una stringa dalla tastiera. La stringa non deve necessariamente essere scritta tra apici, a meno che non si voglia farla terminare con spazi di riempimento. Se la stringa non e' scritta tra apici essa termina con il carattere virgola.

READ X$ Legge una stringa dal blocco dati DATA del programma. Anche in questo caso il DATA non ha mantenuto gli spazi di riempimento a meno che gli elementi stringa non siano stati scritti tra apici e le stringhe terminano o con una virgola o con il carattere di fine linea.

PRINT X$ Visualizza sul video la stringa X$.

PRINT "FOO"+ A$ Visualizza il risultato della espressione fra stringhe.

6.2.2.3. OPERATORI

E' ovvio che il risultato dell'addizione di I% + J% sia 7 se I% contiene 3 e J% contiene 4. E' anche ragionevole aspettarsi che I + J dia come risultato il numero decimale 7.0 se I=3.0 e J=4.0. Negli esempi visti, gli operandi erano dello stesso tipo ed il risultato mantiene questo tipo. In altri casi il tipo del risultato dipendera' dai tipi degli operandi, che possono essere diversi tra loro e dall'operatore che agisce su di essi. Gli operatori possono produrre conversioni nel tipo dei dati sia prima di eseguire le operazioni che dopo.

Gli operatori relazionali e gli operatori logici producono risultati di tipo logico. Risultati di questo tipo non possono essere assegnati ad una variabile e possono solo essere usati nelle frasi IF, per operare delle scelte.

6.2.2.4. DESCRIZIONE DELLE FRASI BASIC

Nelle prossime pagine, nel corso della descrizione delle frasi BASIC, V e W definiscono delle variabili numeriche, X definisce una espressione numerica, I,J o K definiscono una espressione che viene troncata al valore intero prima di essere adoperata. A e B definiscono delle matrici senza alcun parametro. Una espressione e' costituita da una serie di variabili, operatori, chiamate di funzioni e costanti che, dopo l'esecuzione delle operazioni e delle funzioni, condotte a termine applicando le regole esposte, da' luogo ad un valore numerico o ad una stringa.
Una costante puo' essere sia un numero che una stringa.

6.2.2.5. ESPRESSIONI

Il principio che sta alla base del calcolo delle espressioni nel BASIC DAI e' che, se in una espressione compaiono solo numeri interi e/o variabili e operatori che lavorano su numeri interi, tutti i calcoli vengono svolti senza usare il modulo matematico delle operazioni floating-point. Questo rende piu' veloci i calcoli quando non servono i numeri decimali, come e' di solito per i controlli industriali e le applicazioni grafiche.

L'ordine di valutazione degli operatori e' il seguente:

espressioni tra parentesi
elevamento a potenza (rappresentato da una freccia verso l'alto)

```
*    /     MOD
+    -
SHL      SHR
IOR      IAND      IXOR
>        <         =          <=
AND      OR
```

Gli operatori appartenenti allo stesso livello sono valutati procedendo da sinistra a destra. Esempio: 3 * 5 MOD 2 = 1.

6.2.3. SEGNALAZIONE ERRORI

6.2.3.1. FORMATO DELLA SEGNALAZIONE DEGLI ERRORI

Quando si verifica un errore, viene stampato un messaggio che fornisce spiegazioni piu' o meno dettagliate a seconda dei casi.
(1)-Se si e' appena scritto un comando in modo immediato, non vengono fornite altre informazioni, oltre alla

segnalazione di errore.
(2)-Se si e' appena scritta una linea di programma errata, si ha una ripetizione della linea con 3 asterischi (***) dopo il numero di linea ed un punto interrogativo (?) dopo l'errore.
(3)-Se e' in esecuzione un comando in modo immediato, non vengono fornite altre informazioni, oltre alla segnalazione di errore.
(4)-Se e' in esecuzione una linea di programma memorizzato, si ottiene la frase:"IN LINE NUMBER" seguita dal numero di linea.

6.2.3.2. ELENCO MESSAGGI DI ERRORE

CAN'T CONT
Non ci sono programmi in sospeso che possano essere riavviati con CONT.

COLOUR NOT AVAILABLE
E' stato chiesto un colore, nel MODE a 4 colori, che non e' stato preselezionato con un comando COLORG.

COMMAND INVALID
Il comando in questione o non puo' essere usato in modo immediato e si e' in tale modo o viceversa non puo' essere usato sotto controllo del programma e si e' in tale modo.

DIVISION BY 0
Si e' tentato di dividere per 0.

ERROR LINE RUN
Una linea di programma errata e' stata lasciata in un programma e si ha errore in fase esecutiva.

INVALID NUMBER
L'argomento usato in una funzione VAL non e' un numero floating-point valido.

LINE NUMBER OUT OF RANGE
E' stato usato un numero di linea uguale a zero, o negativo o maggiore di 65535.

LINE TOO COMPLEX
La linea di programma scritta genera un codice programma piu' lungo di 128 bytes.

LOADING ERROR 0, 1, 2 o 3
Non si riesce a caricare o un programma o dei dati.
Per le cassette si ha:
-0 significa errore di Checksum nel nome del programma;
-1 significa mancanza di memoria;
-2 significa errore di Checksum nel programma;
-3 rignifica errore nel caricamento di dati.

NEXT WITHOUT FOR
E' stato eseguito un NEXT, ma non era preceduta dal FOR corrispondente.

NUMBER OUT OF RANGE
Sono stati usati dei numeri in un contesto dove risultano o troppo grandi o troppo piccoli.

OFF SCREEN
Si e' indirizzato un punto che non e' accessibile nel Mode attuale.

OUT OF DATA
Una frase READ ha cercato di leggere piu' dati dei DATA di quanti siano disponibili.

OUT OF MEMORY
Si e' tentato di usare piu' memoria di quanta sia disponibile o per il programma o per la tabella dei simboli o per la memoria di schermo o per le stringhe e le matrici o per il buffer di edit.

OUT OF SPACE FOR MODE
Si ha questo messaggio se un programma sta lavorando in Mode 1 o 2 con spazio di memoria libero insufficiente per girare in Mode 0, 1A o 2A e si tenta di stampare un messaggio. Il sistema cancella il programma come se si fosse dato il comando NEW e evidenzia il messaggio.

OUT OF STRING SPACE
Si e' usato piu' spazio per le stringhe di quanto disponibile.

OVERFLOW
Si e' avuto un supero di capacita' o per un numero o per una stringa.

RETURN WITHOUT GOSUB
Si e' eseguita una frase RETURN che non era stata preceduta da un corrispondente GOSUB.

STACK OVERFLOW
O si e' scritta una linea di programma troppo complicata o

il programma in corso ha usato troppo spazio nella Stack area.

STRING TOO LONG
Si e' creata una stringa di piu' di 255 caratteri.

SUBSCRIPT ERROR
E' stato calcolato un indice che esce dalle dimensioni dichiarate per la matrice oppure si e' usato il nome di una matrice con un numero di indici errato, oppure si e' richiesto un dimensionamento a 0.

SYNTAX ERROR
Esiste qualche errore nella linea che si e' appena scritta, oppure nella lista dei dati di una frase INPUT o READ.

TYPE MISMATCH
Una espressione da' dei risultati in disaccordo con il tipo dei dati aspettato.
UNDEFINED ARRAY
Si fa riferimento ad una matrice che non e' stata definita.

UNDEFINED LINE NUMBER
Si fa riferimento ad un numero di linea che non esiste.

6.2.4. IL BASIC DAI E' CONVERSAZIONALE

6.2.4.1. COMUNICAZIONE CON LO SCHERMO

Quando si accende il DAI ed appare sul video la scritta BASIC seguita dall'asterisco di PROMPT, il calcolatore si trova nel Mode 0, questo significa avere sul video 24 linee di 60 caratteri ciascuna. Per tornare in questo modo di funzionamento, basta dare, in qualunque momento, il comando MODE 0.

La posizione dello schermo dove apparira' il prossimo carattere e' determinata dal cursore lampeggiante. La lunghezza delle linee sullo schermo e' fisicamente di 60 caratteri, ma da un punto di vista logico si puo' avere un Output che occupa fino a 3 linee di continuazione. Queste linee, di "continuazione", hanno il primo carattere uguale a 0 ed il primo carattere valido si trova 7 posizioni piu' a destra. Il cursore si muove in avanti quando appare un carattere e all'indietro quando si fa un backspace (CHR$(#8)). Il carattere "carriage return" (ritorno carrello) chiude una linea (CHR$(#D)); nel seguito chiameremo tale tasto RETURN.

Il carattere "form feed" (avanzamento di una linea) (CHR$(#C)) ha lo speciale effetto di pulire tutta l'area dei caratteri e di posizionare il cursore nell'angolo in alto a sinistra. Il carattere "tab" (CHR$(#9)) non ha funzioni speciali.

Quando si supera la terza linea di continuazione, i caratteri restanti fino a: un RETURN, un "backspace" o un "form feed", sono ignorati. Quando il BASIC e' in attesa di un Input vengono accettati i caratteri che seguono il carattere di PROMPT. Se il carattere di PROMPT viene cancellato da un "backspace" qualunque carattere scritto in quella posizione viene ignorato e in generale si avra' un errore di sintassi. I colori con i quali appaiono i caratteri sono quelli disponibili all'accensione; essi possono essere modificati dal comando COLORT.

6.2.4.2. INPUT DI PROGRAMMI E DATI.

Quando il DAI e' in attesa di Input esso fa apparire un carattere di PROMPT, che di norma e' un asterisco (*), ma che durante l'esecuzione del comando INPUT e' un punto interrogativo (?). A questo punto l'utente puo' scrivere quello che desidera. Per cancellare l'ultimo carattere scritto, si puo' usare il tasto "CHAR DEL". Se i caratteri introdotti superano la linea, si prosegue sulla seguente dalla settima posizione e con "C" nella prima posizione, a segnalare che si tratta di una linea di continuazione. Si possono usare fino a 3 linee di continuazione, ottenendo cosi' 59 + 53 + 53 + 53 = 218 caratteri.

Se si usa il tasto BREAK mentre si sta scrivendo un comando si ha la stampa di " ! " e la linea viene ignorata. Se invece si usa BREAK durante l'esecuzione di un comando di INPUT si ha la sospensione del programma.

6.2.4.3. CREAZIONE, MODIFICA E PROVA DI UN PROGRAMMA

Quando il DAI e' pronto per accettare comandi, appare il carattere di PROMPT. L'utente puo' allora scrivere una o piu' linee di programma, una linea puo' contenere anche piu' comandi separati dai due punti (:), ed ogni linea deve terminare con un RETURN. I comandi dati in modo immediato vengono codificati immediatamente e, se non ci sono errori di sintassi, vengono eseguiti appena premuto il RETURN. Se la linea e' stata scritta facendola precedere da un numero di linea, essa viene subito codificata e memorizzata nel programma che si trova in memoria al posto giusto in base al numero di linea. Essa, eventualmente, rimpiazza una linea gia' esistente con lo stesso numero. Se la linea contiene errori di sintassi, questi vengono segnalati. Nel caso di un comando in modo immediato, dopo la segnalazione dell'errore,

non segue alcuna azione. Nel caso di una linea di programma, all'inizio di essa, dopo il numero di linea, vengono inseriti 3 asterischi (***) e la linea viene codificata (solo i primi 121 caratteri) come se fosse un REM. Se si tenta di eseguire la linea errata, si ha il messaggio di errore e viene inserito un punto interrogativo vicino all'errore riscontrato.

Se si desidera provare un programma gia' memorizzato, si deve dare il comando "RUN" se si desidera iniziare dalla prima linea di programma, oppure "RUN XXX" se si desidera partire dalla linea XXX.

Il programma va in esecuzione e termina per una delle seguenti cause:

(1)-Incontra una frase END. Allora appare il messaggio: END PROGRAM. Si puo' ripartire solo dando RUN di nuovo.
(2)-Incontra una frase STOP. Allora appare il messaggio:STOPPED IN LINE XXX. In questo caso il programma e' in uno stato di sospensione.
(3)-L'utente preme il tasto BREAK. Allora, a seconda dei casi, puo' accadere:
- viene stampato il messaggio: BREAK IN LINE XXX ed il programma e' sospeso;
- dopo una pausa il sistema fa apparire il messaggio: ***BREAK ed il programma non puo' continuare.

(4)-Il programma da' errore per un qualunque motivo.

Quando un programma e' stato sospeso, esso puo' continuare la sua esecuzione se si da' il comando CONT. Questo comando fa continuare il programma come se esso non si fosse mai fermato. Naturalmente se durante la sospensione l'utente ha modificato, in modo immediato, il valore di alcune variabili queste restano modificate.

Se si e' avuto un messaggio di errore del sistema, oppure se l'utente ha dato comandi del tipo: RUN, LOAD, SAVE, EDIT, CLEAR, NEW, allora il programma sospeso non puo' essere continuato con CONT. In tali casi se l'utente scrive CONT il sistema risponde: CAN'T CONT. Quando si usa uno dei comandi: RUN, SAVE, CLEAR, LOAD, EDIT, NEW, tutte le variabili numeriche sono riinizializzate a zero e le variabilli stringa divengono stringhe nulle; gli spazi assegnati alle matrici vengono annullati e queste riceveranno di nuovo assegnazione di spazio al momento dell'esecuzione della relativa DIM.

Per cancellare un programma dalla memoria basta il comando NEW.

Quando un programma e' in stato di sospensione si puo' usare il comando STEP per continuare l'esecuzione una linea per volta. Il sistema evidenzia le linee prima di eseguirle e le esegue quando si preme il tasto di spazio.

Al momento dell'accensione del DAI le variabili vengono considerate di tipo floating-point, quindi per avere variabili intere si deve usare il suffisso %. Si puo' naturalmente usare il comando IMP prima di caricare un programma per modificare lo stato delle variabili.

6.2.4.4. FUSIONE DI DUE PROGRAMMI BASIC

Si possono fondere due programmi BASIC, naturalmente i due moduli non devono avere numeri di linea coincidenti, con la seguente procedura:

```
CLEAR 10000
LOAD SEGMENTO-1 DEL PROGRAMMA DA FONDERE
EDIT + BREAK + BREAK
LOAD SEGMENTO-2 DEL PROGRAMMA DA FONDERE
POKE#135,2
```

6.2.4.5. FUSIONE DI PROGRAMMI IN BASIC E IN LINGUAGGIO MACCHINA

Si deve preparare il programma in linguaggio macchina e memorizzarlo dopo il programma BASIC con il quale si vuole fondere. Chiamiamo il programma in linguaggio macchina MLP/R.
Esempio:

Si abbia il seguente programma BASIC:

```
10 CLEAR 2000
20 DIM A (20,20)
30 LOADA A
40 CALLM #2F1
50 STOP
```

esso deve essere memorizzato con SAVE.

Si abbia il seguente programma MLP/R:

```
10 CLEAR 2000
20 DIM A (20,20)
30 FOR I% = 0 TO 9
40 READ B%: POKE (#2F1 +I%), B%:NEXT
50 SAVEA A "TEST": STOP
60 DATA #F5,#3E,#FF,#32
65 DATA #50,#BE,#F1,#C9,0,0
```

Notate che le dimensioni massime di una matrice con un solo indice sono di 256x4 bytes, mentre qui sono di 20x20x4=1600 bytes.

Nei due programmi ci sono la CLEAR e la DIM identiche; nel primo c'e' la LOADA del secondo programma.

Il primo programma carichera' il secondo con la sua linea 30 e lo mandera' in esecuzione con la sua linea 40. Voi potrete, ogni volta che volete fare girare il secondo

programma dare RUN 40. Se volete potete cancellare le prime 3 linee scrivendo: 10 RETURN, 20 RETURN, 30 RETURN.

E' importante ricordare che, dopo che il programma MLP/R e' stato caricato dal programma BASIC, non si deve passare in modo EDIT e non si devono mandare in esecuzione linee che contengano: CLEAR, DIM e LOADA.

Se volete usare un programma MLP/R e non avete gia' deciso dove localizzarlo in memoria, potete trovare la locazione di inizio scrivendo:
PRINT HEX$ (VARPTR (A(0,0))). Questa locazione di solito e' #2F0 per il primo MLP/R con una matrice ad una dimensione, #2F1 se la matrice e' a due dimensioni, sempre che non si usino i dischi.

6.2.5. COMANDI DI CONTROLLO

6.2.5.1. EDIT

Cominciamo con degli esempi:
(1) EDIT trasferisce tutto il programma BASIC nel Buffer dell'EDIT per poterlo modificare o stampare;
(2) EDIT 100 trasferisce solo la linea 100 del programma nel Buffer;
(3) EDIT 100 - trasferisce tutto il programma a partire dalla linea 100 nel Buffer;
(4) EDIT 100 - 130 trasferisce le linee di programma da 100 a 130 nel Buffer;
(5) EDIT - 130 trasferisce il programma dall'inizio e fino alla linea 130 nel Buffer.

L' EDIT serve per scrivere e modificare un programma BASIC. Esiste un Buffer interno nel quale vengono memorizzate le linee del programma; le prime 24 linee del Buffer vengono evidenziate sullo schermo. Il cursore viene posizionato al primo carattere della prima linea dello schermo.

Il cursore puo' essere posizionato dovunque nello schermo usando i 4 tasti con le frecce. Se si tenta di portare il cursore fuori dello schermo, appare sullo schermo la parte di programma cercata. La parte di programma visibile sullo schermo si chiama "finestra" (window). La "finestra" puo' essere cambiata usando insieme il tasto di controllo del cursore ed il tasto di shift. Il tasto CHAR DEL cancella il carattere prima del cursore; non ha effetto dopo un RETURN. Qualunque carattere premuto viene inserito a sinistra del cursore, se esso si trova a sinistra del RETURN.

Quando tutte le operazioni di EDIT sono terminate, si deve usare il tasto BREAK. Se si preme due volte il tasto BREAK si cancella (abolisce) tutto il lavoro fatto sotto EDIT. Se

dopo il BREAK si fa uno spazio viene cancellata la versione precedente del testo.
Vengono evidenziati sullo schermo i messaggi di errore, seguiti dal PROMPT.
Il comando EDIT puo' anche essere usato per fondere programmi, provenienti da diversi nastri.
Si raccomanda di evitare di premere qualunque tasto dopo aver dato il comando EDIT e prima che appaia il programma richiesto sullo schermo.

6.2.5.2. IMP

Valgono gli esempi dati nel paragrafo 6.2.2.

6.2.5.3. LIST

Consideriamo i seguenti esempi, sufficientemente esplicativi:
(1) LIST mostra sul video tutto il programma, usando lo scrolling, se il programma e' lungo. Si puo' arrestare lo scrolling premendo un qualunque tasto e farlo proseguire usando il tasto di spazio.
(2) LIST 100 mostra solo la linea 100
(3) LIST 100 - mostra il programma dalla linea 100 in poi.
(4) LIST 100 - 150 mostra il programma dalla linea 100 alla linea 150.
(5) LIST - 100 mostra il programma dall'inizio e fino alla linea 100.

6.2.5.4. NEW

Con questo comando si cancella il programma presente in memoria e vengono rimesse a zero o a stringa nulla tutte le variabili. Non si ha alcuna variazione riguardo alla memoria assegnata per stringhe e variabili con indice.

6.2.5.5. RUN

Vediamo alcuni esempi:
(1)-RUN fa partire l'esecuzione del programma dal numero di linea minore e riposiziona al valore iniziale le variabili.
(2)-RUN 100 come sopra, ma il programma inizia da 100. Se la linea 100 non esiste si ha un messaggio di errore.

6.2.6. ISTRUZIONI DI CONTROLLO NEL PROGRAMMA

6.2.6.1. END

Questo comando fa terminare l'esecuzione di un programma ed appare il messaggio:END PROGRAM. Se si vuole proseguire si deve dare il comando RUN.

6.2.6.2. FOR.....NEXT

Vediamo subito degli esempi:

```
(1)- FOR V = 1 TO 9.3 STEP .6
(2)- FOR V = 1 TO 9.3
(3)- FOR V = 10 * N TO 3.4/Q STEP SQR(R)
(4)- FOR V = 9 TO 1 STEP - 1
(5)- FOR W% = 1 TO 10 : FOR V = 0 TO 3 : NEXT : NEXT
```

La variabile del FOR viene inizializzata con la prima espressione data. Le frasi seguenti vengono eseguite fino a quando non si incontra una frase NEXT. Quello che succede al NEXT dipende da come e' scritta la frase FOR. Quando inizia la frase FOR vengono calcolate anche le espressioni dopo il TO e dopo lo STEP. Se STEP non e' presente, esso viene assunto = 1. Viene calcolato, dividendo l'intervallo per lo STEP, quante ripetizioni del ciclo devono essere eseguite, per poter creare un contatore per il ciclo. Si possono avere da 0 a 2 elevato a (23-1) iterazioni se il ciclo e' controllato da numeri reali, e da 0 a 2 elevato a (31-1) se il ciclo e' controllato da numeri interi. Si ottiene l'esecuzione ciclica tante volte quanto e' il valore calcolato per il contatore, ma se il ciclo e' male impostato, si ha almeno una esecuzione. Se non si e' usato lo STEP, il sistema usa una routine speciale che rende piu' veloce l'esecuzione. I cicli controllati da numeri interi sono piu' veloci di quelli controllati da numeri reali.

Casi speciali:

(a)- Se si hanno due FOR uno interno all'altro e manca il NEXT di quello piu' interno, il sistema considera terminato il FOR piu' interno al NEXT di quello piu' esterno:
FOR A = 1 TO 10 : FOR B = 0 TO 3 : NEXT A e' consentito.

(b)- Il sistema considera terminati i FOR piu' interni se viene riiniziato un FOR piu' esterno:

```
10 FOR A = 1 TO 10
20 FOR B = 0 TO 3
30 GOTO 10
```

(c)- I cicli FOR interni ai sottoprogrammi sono separati da

quelli esterni secondo le regole viste in a) e b)

(d)- Un ciclo FOR puo' essere abbandonato da un RETURN:
```
10 GOSUB 30
20 STOP
30 FOR A = 1 TO 10
40 RETURN
```

(e)- Alla fine di un ciclo FOR, la variabile del ciclo si trova all'ultimo valore aumentato dello STEP:
```
10 FOR I = 1 TO 10
15 NEXT
20 PRINT I
```
stampera' 11.0

6.2.6.3 GOSUB

Il comando GOSUB deve essere seguito da un numero di linea: GOSUB 500, per esempio. Esso fa proseguire l'esecuzione del programma dalla linea 500. Quando il programma incontra poi la frase RETURN esso prosegue dalla linea seguente a quella del comando GOSUB. Si possono avere catene di GOSUB ed il numero dei GOSUB concatenati dipende da quanta area STACK e' disponibile. Un programma puo' avere 10 GOSUB concatenati o 15 FOR concatenati senza difficolta'.

6.2.6.4. GOTO

Il comando GOTO 670, per esempio, fa proseguire l'esecuzione del programma dalla linea 670. Si tratta del comando di salto incondizionato.

6.2.6.5. IF....GOTO

Vediamo degli esempi:

(1)- IF X = Y + 23.4 GOTO 95 se la condizione proposta e' verificata (VERA) il programma prosegue dalla linea 95, se non e' verificata (FALSA) il programma prosegue dalla linea seguente.

(2)- IF X = 5 GOTO 50 : Z = A e' un errore di logica, infatti non verra' mai eseguito Z = A.

6.2.6.6. IF....THEN

Vediamo degli esempi:

(1)- IF X < 0 THEN PRINT "X < 0": GOTO 350 se la condizione e' vera viene stampato il messaggio ed il programma prosegue dalla linea 350, altrimenti va in sequenza.

(2)- IF X = Y + 23.4 THEN 95 in questo caso e' indifferente usare IF...THEN o IF...GOTO.

6.2.6.7. ON....GOSUB

Vediamo un esempio:

100 ON K GOSUB 1000, 2000, 3000 se K = 1 viene eseguito il sottoprogramma che inizia in 1000, se K = 2 viene eseguito il sottoprogramma che inizia in 2000 e se K = 3 viene eseguito il sottoprogramma che inizia in 3000. Si chiama istruzione di salto calcolato a sottoprogramma. Il RETURN del sottoprogramma fa ritornare all'istruzione che segue l'ON..GOSUB.

6.2.6.8. ON....GOTO

Vediamo alcuni esempi:

(1)- ON I GOTO 10, 20, 30, 40
se I=1 fa saltare alla linea 10
se I=2 fa saltare alla linea 20
se I=3 fa saltare alla linea 30
se I=4 fa saltare alla linea 40
se I < = 0 o > (numero dei numeri di linea),
fa proseguire dalla frase seguente.
Se il numero di linea selezionato non esiste si ha errore.

(2)- ON SGN(X)+2 GOTO 40, 50, 60 l'espressione tra ON e GOTO viene valutata, SGN(X) puo' essere uguale a -1,0,+1, e quindi l'intera espressione puo' valere o 1 o 2 o 3 e si ricade nel caso precedente.

6.2.6.9. RETURN

Questa istruzione fa tornare alla frase seguente l'ultimo GOSUB eseguito.

6.2.6.10. STOP

Questa istruzione fa arrestare l'esecuzione del programma; viene evidenziato il messaggio: STOPPED IN LINE .... Per far proseguire il programma si deve scrivere CONT.

6.2.6.11. WAIT

Questa istruzione serve per creare un'attesa nel programma. Vediamo alcuni esempi:

(1)- WAIT I, J, K viene letto lo stato della REAL WORLD INPUT Porta I, ne viene fatto l'OR ESCLUSIVO con la variabile K, poi viene fatto l'AND del risultato con la variabile J fino a quando l'esito dell'AND e' vero. A questo punto il programma prosegue dall'istruzione seguente. Se i parametri di WAIT sono solo 2, K e' preso uguale a 0. Se si vuole creare una attesa fino a quando un bit in una determinata posizione diventa zero si deve mettere un bit a 1 nella corrispondente posizione di K. Le tre variabili I, J e K devono essere > = 0 e < = 255.

(2)- WAIT MEM I, J, K funziona come in (1), ma I e' una locazione di memoria qualsiasi e quindi anche una Porta di I/O localizzata in memoria.

(3)- WAIT TIME I sospende l'esecuzione di un programma per un intervallo di tempo dato dall'espressione I. Il risultato deve essere compreso tra 0 e 65535. Il tempo e' misurato in intervalli di 20 millisecondi.

6.2.7. COMANDI PER ACCESSO FISICO ALLA MACCHINA

6.2.7.1. CALLM

Esempi:

(1)- CALLM 1234 manda in esecuzione una routine in linguaggio macchina a partire dall'indirizzo specificato 1234.

(2)- CALLM I, V manda in esecuzione una routine in linguaggio macchina a partire dall'indirizzo I. Dopo la chiamata della routine la coppia di registri H ed L contiene l'indirizzo della variabile che si trovava in V. La routine in linguaggio macchina deve memorizzare tutti i registri ed i flags dell'8080 e ripristinarli prima del RETURN. Se V e' una variabile, il puntatore e' a V. Se V e' una stringa, il puntatore punta al puntatore della stringa. La stringa e'

formata da un byte che da' la lunghezza della stringa, seguito dai caratteri della stringa. Se V e' una matrice, il puntatore si comporta come per una normale variabile.

6.2.7.2. INP (I)

Esempio:

A = INP (31) legge il byte presente sul DCE-BUS della CARD 3 PORTA 1 e pone il suo valore nella variabile A. Il numero della porta deve essere compreso tra 0 e 255.

6.2.7.3. OUT I, J

Esempio:

OUT 91, A manda il numero contenuto nella variabile A al DCE-BUS CARD 9 Porta 1. Sia I che J devono essere compresi tra 0 e 255.

6.2.7.4. PDL (I)

Esempio:

A = PDL(I) pone nella variabile A un numero compreso tra 0 e 255 che rappresenta la posizione di uno dei potenziometri delle paddles. I deve essere compreso tra 0 e 5.

6.2.7.5. PEEK (I)

Esempio:

A = PEEK (#13C2) pone nella variabile A il contenuto dell'indirizzo esadecimale 13C2. Se I non e' compreso nell'intervallo 0-65536 si ha errore. Un tentativo di leggere indirizzi inesistenti di memoria da' un valore non prevedibile.

6.2.7.6. POKE I, J

Esempio:

POKE I, J memorizza il byte individuato da J (secondo argomento) nella locazione di memoria di indirizzo I (primo argomento). J deve essere compreso tra 0 e 255. I deve

essere un indirizzo valido di memoria. Si devono usare le istruzioni POKE con una certa cautela per evitare danni al sistema e perdite di programmi in corso di prova.
Seguono alcuni esempi di uso del comando POKE.

```
POKE #131,0   da' un output sullo schermo e su RS 232
POKE #131,1   da' un output sullo schermo
POKE #131,2   da' un output nel Buffer di Edit
POKE #135,2   legge dal Buffer di Edit
POKE #13D,#10 seleziona la cassetta 1
              (usare #20 per cassetta 2)
POKE #40,#28  mette in ON il motore della cassetta 1
POKE #40,#18  mette in ON il motore della cassetta 2
POKE #40,#30  mette in OFF i motori delle due cassette
POKE #730,#30 attiva il drive 0 del floppy
POKE #730,#31 attiva il drive 1 del floppy
```

Andate a vedere altri esempi di POKE in 5.9.1./2./3.

6.2.7.7. UT

Esempio:

```
UT          chiama il Monitor del linguaggio macchina.
```

6.2.8. COMANDI BASIC PER DATI E INPUT/OUTPUT

6.2.8.1. DATA

Esempi:

(1)- DATA 1,3,-1E3,-0.4 specifica 4 dati numerici; questi dati vengono immagazzinati nello stesso ordine nel gruppo dati generato da tutti i DATA del programma.

(2)- DATA "F00", "Z00" specifica due stringhe da immagazzinare insieme agli altri dati. Per le stringhe si puo' evitare di usare gli apici delimitativi; essi sono necessari solo se nella stringa esistono virgole o ci sono spazi da conservare a destra.

6.2.8.2. GETC

Esempio:

```
A = GETC          pone nella variabile A l'ultimo carattere
```
premuto sulla tastiera. Se dopo l'ultimo GETC non sono stati premuti tasti A contiene zero. Questo comando provoca una

scansione della tastiera; se si premono i tasti a lungo sulla tastiera puo' capitare che un tasto appaia come premuto due volte dato che la scansione e' molto veloce.

6.2.8.3. INPUT

Esempi:

(1)- INPUT V, W, W2 richiede dati dal terminale. Rispondendo, ogni dato deve essere separato dal dato seguente, se esiste, da una virgola.Dopo l'ultimo valore si deve premere RETURN. Quando il programma esegue il comando INPUT appare come carattere di PROMPT il "?". Se si risponde con meno dati di quanti richiesti viene evidenziato un altro "?" e si deve continuare l'immissione di dati.

Se si risponde con piu' dati di quanti richiesti, il sistema avvisa che i dati in piu' sono ignorati. Nell'immissione delle stringhe valgono le stesse regole viste per i DATA.

(2)- INPUT "VALUE"; V stampa VALUE prima del punto interrogativo e si comporta come in (1). Se si interrompe un INPUT con il tasto BREAK si puo' scrivere CONT e viene rieseguito il comando di INPUT. Se capita un qualunque errore il comando di INPUT viene rieseguito completamente.

6.2.8.4. PRINT (Puo' essere sostituito da"?")

Esempi:

```
(1) PRINT X, Y, Z
(2) PRINT
(3) PRINT X, Y
(4) PRINT "IL VALORE E' "; A
(5) ? A2,B
```

Si ottiene la stampa del contenuto delle variabili o delle costanti elencate dopo la parola PRINT. Se la lista di variabili o/e costanti non termina con virgola o punto e virgola (, ;) dopo la stampa si ha l'avanzamento alla prossima linea.

Se gli elementi da stampare sono separati dal punto e virgola(;) essi sono stampati uno vicino all'altro, mentre se sono separati dalla virgola (,) dopo un elemento il cursore si posiziona all'inizio del prossimo campo. Se la lista di stampa e' vuota, si ha solo l'avanzamento alla prossima linea.

In ogni linea si hanno 5 campi con inizio nelle posizioni 0, 12, 24, 36, 48.

6.2.8.5. READ

Esempio:

READ V, W legge dati dal gruppo dati creato dalle frasi DATA del programma. Il gruppo dei dati (originati dai DATA) e' gestito da un puntatore interno che inizialmente precede il primo dato e viene spostato di un dato per ogni dato letto. La frase READ legge i dati in base alla posizione del puntatore e li trasferisce nelle variabili della lista. Se si tenta di leggere con READ piu' dati di quanti disponibili si ha errore esiste pero' un comando che permette di riposizionare il puntatore al primo dato del gruppo, esso e': RESTORE.

6.2.8.6. RESTORE

Con questo comando viene riposizionato il puntatore interno all'inizio del blocco dati, originato dai DATA.

6.2.9. COMANDI I/O PER CASSETTE E DISCHI

Se si usano i Resident Machine Utility Program sono disponibili altri comandi oltre quelli qui elencati, vedere 7.3. (UTILITY).

6.2.9.1. CHECK

Questo comando serve per scandire una casseta o un disco esaminando tutti i files. Vengono stampati il tipo ed il nome di ogni file seguiti dalla parola OK o BAD a seconda che il controllo di CHECKSUMM e' o meno corretto. La pressione del tasto BREAK interrompe l'operazione.

6.2.9.2. LOAD

Esempi:

(1)- LOAD "PIPPO" carica in memoria il programma PIPPO o da una cassetta o da un disco. Alla fine appare il carattere di PROMPT. Il nome del file puo' essere una qualunque stringa stampabile.

(2)- LOAD carica il primo programma che trova sulla cassetta. Il motore del registratore dovra' o meno essere avviato manualmente a seconda del tipo di registratore collegato.
Se questo comando viene eseguito in modo diretto vengono evidenziati il nome ed il tipo di ogni programma o blocco dati incontrati nella scansione del nastro. Se il

caricamento avviene correttamente appare il carattere di PROMPT altrimenti si ha il messaggio: LOADING ERROR seguito da un numero che specifica l'errore riscontrato. Durante il caricamento da nastro scompare il cursore lampeggiante.

6.2.9.3. LOADA

Carica in memoria matrici o programmi in linguaggio macchina memorizzati sotto forma di matrici; esempi:

```
LOADA A$ "PIPPO"  oppure  LOADA F$ + "J"
dove PIPPO o J sono nomi di matrici.

10 DIM A$(0,0)              100 DIM A$ (0,0)
20 INPUT A$                 110 LOADA A$
30 SAVEA A$ "INFO"          120 GOTO 100
40 GOTO 10
```

6.2.9.4. SAVE

Esempi:

```
(1) SAVE "PIPPO"
(2) SAVE A$
```

Memorizzano su cassetta o su disco il programma che sta in memoria. Il programma resta anche in memoria invariato. Si possono memorizzare successivamente piu' programmi, ognuno con il suo nome.

(3) SAVE il programma viene memorizzato su cassetta senza nome.

Il sistema risponde a questo comando con: SET RECORD, START TAPE, TYPE SPACE e si deve agire in conseguenza, rispettando le esigenze del tipo di registratore collegato. Alla fine appare il carattere di PROMPT.

6.2.9.5. SAVEA

Esempi:

```
(1) SAVEA "PIPPO"
(2) SAVEA A$
```

Memorizzano una matrice su cassetta o su disco.

```
(3) SAVEA A
(4) 10 INPUT A$
    20 SAVEA A$
    30 GOTO 10
```

Se il registratore e' automatico, quando date il RUN il nastro iniziera' a girare e verra' memorizzato quello che entra con la linea 20.

(5) Per copiare un programma seguito da una matrice (o routine in linguaggio macchina) con due registratori a cassetta, uno in stato PLAY e l'altro in stato RECORD:

```
POKE #40,#28:LOAD:POKE #40,#18:SAVE:POKE #40,#28
PRINT "FINE SAVE":CLEAR 2000:DIM A(20,20):LOADA A
POKE #40,#18
SAVEA A :POKE #40,#28
```

Premere RETURN. La matrice ha il nome A.

6.2.10. PROVA DEL PROGRAMMA E FRASI DI COMMENTO

6.2.10.1. CONT

Questa frase consente di continuare l'esecuzione di un programma che era stato interrotto da una istruzione STOP inserita nel programma stesso oppure dall'uso del tasto BREAK.

6.2.10.2. REM

Vediamo degli esempi:

```
(1) REM ORA PONI V = 0
```

Consente di inserire commenti nel programma. Non e' una frase esecutiva e viene saltata in esecuzione. La frase REM termina alla fine della linea e non puo' essere separata da un'altra istruzione dai due punti (:).

```
(2) REM PONI V = 0 : V = 0
```

Chiaramente la frase dopo i due punti non verra' mai eseguita, ma continua ad essere considerato un commento.

```
(3) V = 0 : REM PONGO V = 0
```

In questo caso V = 0 viene eseguito perche' viene prima di REM.

6.2.10.3. STEP

Con questo comando e' possibile procedere all'esecuzione del programma una istruzione per volta. Dopo uno STOP o un BREAK si puo' dare il comando STEP; dopo di cio', ad ogni pressione della barra di spazio viene evidenziata ed eseguita una istruzione.

6.2.10.4. TRON

Se scriviamo:

```
100 A = 0
105 TRON
110 A = 1
115 A = 2
120 TROFF
```

quando facciamo girare il programma con RUN, dopo la linea 105, cioe' dopo l'esecuzione della istruzione TRON (TRACE ON), le linee 110 e 115 vengono evidenziate ed eseguite. Alla linea 120 con l'istruzione TROFF (TRACE OFF) ha termine la funzione attivata da TRON.

6.2.10.5. TROFF

Vedi 6.2.10.4.

6.2.11. ISTRUZIONI PER MATRICI E VARIABILI

6.2.11.1. CLEAR

Possiamo scrivere:

```
CLEAR 700
```

Questa frase ha il seguente effetto: pone tutte le variabili numeriche al valore zero, trasforma tutte le stringhe in stringhe nulle, libera tutti gli spazi assegnati alle matrici. Lo spazio di memoria (HEAP) assegnato all'insieme delle stringhe e delle matrici viene ad essere uguale al numero che segue CLEAR (in bytes). Questo spazio puo' essere al minimo di 4 bytes ed al massimo di 32767 bytes.

6.2.11.2. DIM

Vediamo alcuni esempi:

```
(1) DIM A(3), B(10)
(2) DIM R3(5,5), D$(2,2,2)
```

Viene assegnato spazio alle 4 matrici A, B, R3, D$. Le matrici possono avere piu' di 1 dimensione. Gli indici iniziano dal valore 0, quindi DIM A(3) definisce una matrice a una dimensione con 4 elementi. Il valore massimo

consentito per una dimensione e' 254. Le dimensioni possono essere indicate sia come costanti che come variabili. La frase DIM puo' essere eseguita piu' volte in un programma per variare le dimensioni di una specifica matrice. La parte di memoria RAM usata per le matrici forma un unica area contigua (HEAP) con quella usata per le stringhe.

6.2.11.3. FRE

Vediamo alcuni esempi:

(1) A = FRE

La variabile A contiene il numero di bytes non occupati dal programma; tale numero non tiene conto dello spazio di memoria dedicato alle stringhe e alle matrici.

(2) PRINT FRE

Viene evidenziato il numero di bytes di memoria ancora liberi.

6.2.11.4. LET

Le frasi:

```
LET W = X
V = 5.23
```

assegnano un valore ad una variabile. L'uso della parola chiave LET e' opzionale.

6.2.11.5. VARPTR (V)

Vediamo alcuni esempi:

(1) A = VARPTR (B)

La variabile A contiene l'indirizzo di memoria della variabile B.

(2) A = VARPTR (B(3,4))

La variabile A contiene l'indirizzo dell'elemento 3,4 della matrice B.

6.2.12. ISTRUZIONI PER LA GRAFICA

Vedere come esempio il programma "LE TORRI DI HANOI"

6.2.12.1. MODE

Vediamo alcuni esempi:

(1) MODE 0

Pone il video nel modo per i caratteri.

(2) MODE 1A

Pone il video nello stato della grafica a 16 colori con 4 linee per i caratteri in basso.

Il DAI ha 3 diverse definizioni per la grafica e in ogni definizione ci sono 4 diverse configurazioni per il video. Due di queste configurazioni consentono solo la grafica, mentre le altre due mantengono le stesse caratteristiche per la grafica diminuendo lo spazio per la stessa a favore di 4 linee di caratteri nella parte bassa dello schermo. Dal punto di vista hardware si hanno due diverse possibilita' per la grafica: il modo a 16 colori ed il modo a 4 colori. Nel modo a 16 colori ogni punto dello schermo puo' essere posto in uno dei 16 colori disponibili, pero' ogni campo di 8 punti orizzontali vicini puo' contenere solo 2 (talvolta 3) colori diversi. Per i dettagli rivedete il paragrafo 3.2.2.1.. In ogni momento possono essere modificati i colori selezionati ottenendo effetti molto interessanti e l'animazione delle figure.

TABELLA DEFINIZIONE MODI

Modi	Grafica	Testo	Colori dei grafici
0	-	24 x 60	-
1	72,65	-	16
1A	72,65	4 x 60	16
2	72,65	-	4
2A	72,65	4 x 60	4
3	160,130	-	16
3A	160,130	4 x 60	16
4	160,130	-	4
4A	160,130	4 x 60	4
5	336,256	-	16
5A	336,256	4 x 60	16
6	336,256	-	4
6A	336,256	4 x 60	4

6.2.12.2. COLORG

Esempio:

```
COLORG 1 2 3 4
```

Pone i colori 1, 2, 3, 4 nei quattro registri colore.
Se lo schermo si trova gia' in uno dei modi a 4 colori, si ha il cambio immediato dei colori. Se lo schermo si trova invece in uno dei modi a 16 colori, non si ha alcun effetto immediato visibile. Comunque al primo cambio di MODE il colore 1 del registro 1 determina il colore dello sfondo (Vedi tabella dei colori a pagina 13 della seconda parte del manuale).

6.2.12.3. COLORT

Esempio:

```
COLORT 8 15 0 0
```

Predispone lo sfondo al colore 8 ed il testo al colore 15. Gli altri due colori sono normalmente a zero e non vengono usati. Essi possono venire usati in casi speciali da istruzioni POKE o da routine il linguaggio macchina.

6.2.12.4. ISTRUZIONI PER DISEGNARE

I punti dello schermo sono identificati da un sistema X,Y di coordinate aventi l'origine nell'angolo in basso a sinistra. Se si tenta di andare fuori dall'area massima si ha una segnalazione di errore. E' pero' possibile disegnare nella parte alta dello schermo non visibile nei MODE misti. Come avete gia' visto al paragrafo 3.2.12., con l'istruzione DOT si possono disegnare punti, linee e rettangoli colorati.

6.2.12.4.1. DOT

L'istruzione:

```
DOT 10, 20 15
```

posiziona un punto del colore 15 usando le coordinate X=10 e Y=20. Le dimensioni del punto dipendono dal grado di risoluzione selezionato.

6.2.12.4.2. DRAW

L'istruzione:

```
DRAW 91,73 42,77 15
```

disegna una linea del colore 15 tra i due punti di coordinate 91,73 e 42,77. La grossezza della linea dipende dal grado di risoluzione selezionato.

6.2.12.4.3. FILL

L'istruzione:

```
FILL 91,73 42,77 15
```

riempie con il colore 15 un rettangolo avente due angoli opposti nei punti di coordinate 91,73 e 42,77. La reale grandezza del rettangolo dipende dal grado di risoluzione selezionato.

6.2.12.5. ISTRUZIONI PER L'ANIMAZIONE DELLE FIGURE

In ciascuno dei modi a 4 colori lo stato di ogni punto dello schermo e' descritto da due bits. Il valore binario di questi due bits seleziona uno dei 4 colori possibili. Di norma le istruzioni DOT, DRAW e FILL pongono ambedue questi bits al loro nuovo valore. Il sistema consente di agire su uno solo di questi bits usando come numero di colore un numero oltre il 15 e cioe' 16,17,18 o 19.
Se scriviamo:

```
MODE 2A
COLORG 6 9 12 15
```

otteniamo per tutti i punti il colore 6 e le coppie di bits dei punti dello schermo sono tutte al valore 00 (colore BACKGROUND). Se ora scriviamo:

```
DOT 10,10 17
```

otteniamo la modifica del bit basso nel punto 10,10 e la coppia di bits passa al valore 01, colore 9. Se ora scriviamo:

```
DOT 10,10 19
```

otteniamo la modifica del bit alto nel punto 10,10 e la coppia di bits passa al valore 11 (cioe' 3), colore 15. Se scriviamo ancora:

```
DOT 10,10 16
```

otteniamo l'azzeramento del bit basso nel punto 10,10 e la coppia di bits passa al valore 10 (cioe' 2), colore 12. Se scriviamo ancora:

```
DOT 10,10 18
```

otteniamo l'azzeramento del bit alto nel punto 10,10 e la coppia di bits passa al valore 00, colore 6.

Con questa procedura si possono gestire due disegni indipendenti l'uno dall'altro sullo schermo, mantenendone uno invisibile (cioe' dello stesso colore dello sfondo) e l'altro visibile.

Il disegno invisibile puo' essere modificato mentre l'altro e' visibile, poi si scambiano istantaneamente i colori dei due disegni ottenendo l'effetto del movimento. Vediamo un esempio di programma:

```
 5  MODE 2
10  COLORG 0 0 0 0
20  FOR Q = 1 TO XMAX
30  DRAW 0,0 Q,YMAX 17+2*A: REM COLORE = 17 0 19
40  COLORG 0 15-15*A 15*A 15:REM COLORE = 18 0 16
50  DRAW 0,0 Q-1,YMAX 18-2*A:A=1-A:NEXT
```

Il programma precedente lavora in MODE 2. La linea 10 pone il colore 0 (nero) nei 4 registri colore e quindi tutti i bits colore dei punti dello schermo sono a 00. La linea 20 istituisce un ciclo facendo variare Q dal valore iniziale 1 al valore finale XMAX con incremento unitario ad ogni giro. La linea 30 disegna una linea partendo dal punto 0,0 ed arrivando la prima volta (Q=1) al punto 1,YMAX con il colore 17 (infatti A=0 inizialmente). Il 17 ha l'effetto di porre al valore 1 il bit basso dei bits colore del punto disegnato e quindi di selezionare il registro colore numero 1 che contiene il colore 0 (nero); per questa ragione la prima linea disegnata non si vede. La linea 40 modifica il contenuto dei registri colore ponendo, al primo giro, i colori 0, 15, 0, 15, e quindi fa apparire in bianco la linea disegnata prima. La linea 50 disegna una linea partendo dal punto 0,0 ed arrivando la prima volta al punto 0,YMAX con il colore 18. Il 18 ha l'effetto di porre al valore 0 il bit alto dei bits colore del punto disegnato e quindi di selezionare il registro colore numero 0 che contiene il colore 0 (nero); per questa ragione la linea disegnata non si vede. La nuova linea e' stata disegnata in una posizione antecedente a quella della liinea precedentemente resa visibile in bianco. A questo punto, sempre nella linea 50 il valore di A passa a 1 e si torna alla linea 30 con Q=2.

Viene ripercorso tutto il ciclo , ma dopo il primo giro la linea disegnata in 50 ricopre la linea resa visibile precedentemente e l'effetto combinato della 40 e della 50 rende animato il disegno.

Come avete visto nell'esempio precedente l'uso dei numeri di colore oltre il 15 ha il seguente effetto:

17	pone a 1 il bit basso
19	pone a 1 il bit alto
16	pone a 0 il bit basso
18	pone a 0 il bit alto

e quindi essi possono essere usati per riferirsi in modo indiretto ai colori contenuti nei 4 registri colore nelle frasi DOT, DRAW e FILL.

Siano rosso, giallo, verde e nero i 4 colori selezionati con la frase:

```
COLORG 3 14 5 0
```

per effetto di un programma grafico la situazione dei bits colore dei punti dello schermo puo' essere:

00, punto rosso
01, punto giallo
10, punto verde
11, punto nero.

Se in una delle frasi grafiche si usa il riferimento indiretto al colore tramite i codici colore da 16 a 19 si ha che:

con 16 i punti gialli diventano rossi,
i punti neri diventano verdi;
con 17 i punti rossi diventano gialli,
i punti verdi diventano neri;
con 18 i punti verdi diventano rossi,
i punti neri diventano gialli;
con 19 i punti rossi diventano verdi,
i punti gialli diventano neri.

COLORI DA 20 A 23

Se si usano i numeri di colore da 20 a 23 vengono resi attivi i 4 colori che erano stati selezionati dall'ultima frase COLORG usata. Il numero 20 si riferisce al colore selezionato dal primo registro, il numero 21 a quello selezionato dal secondo registro, il numero 22 a quello selezionato dal terzo registro ed il numero 23 a quello selezionato dal quarto registro.

Anche questo e' un modo di riferimento indiretto ai

colori, ottenuto modificando il riferimento dei punti dello schermo ai registri colore.

6.2.12.6. XMAX

L'istruzione:

```
A = XMAX
```

pone in A il valore massimo consentito per la variabile X nel modo grafico in essere.

6.2.12.7. YMAX

L'istruzione:

```
B = YMAX
```

pone in B il valore massimo consentito per la variabile Y nel modo grafico in essere.

6.2.12.8. SCRN(X,Y)

L'istruzione:

```
A = SCRN(31,20)
```

pone in A il numero del colore del punto di coordinate 31,20.

6.2.12.9. CURSOR

L'istruzione:

```
CURSOR 40,20
```

sposta il cursore nella quarantesima posizione della ventesima riga partendo dal basso dello schermo. Con questa istruzione si puo' spostare il cursore in qualunque punto dello schermo.

6.2.12.10. CURX

L'istruzione:

```
A = CURX
```

pone nella variabile A il valore della X della posizione del cursore sullo schermo. Il valore sara' < = 60.

6.2.12.11. CURY

L'istruzione:

```
B = CURY
```

pone nella variabile B il valore della Y della posizione del cursore sullo schermo. Il valore sara' < = 24>

6.2.13. ISTRUZIONI PER IL SONORO

6.2.13.1. ISTRUZIONI PER PROGRAMMARE SUONI

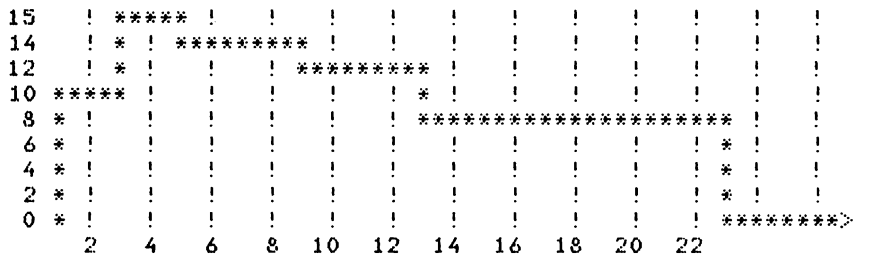
Il generatore di suoni del DAI e' supportato dal BASIC che fornisce un gruppo di comandi per controllare il sistema dei suoni e precisamente, 3 canali per i generatori di tono e un canale per il generatore di rumori bianchi. Il comando SOUND e' il primo metodo di controllo fornito. Esso specifica un canale, l'envelope, il volume e la frequenza da usare.
Per esempio, il comando:

```
SOUND 0 1 15 0 FREQ(1000)
```

specifica il canale 0, l'envelope 1, con volume 15 e frequenza 1000 Hz. A cosa serve il valore 0 dopo il 15 verra' spiegato piu' avanti.
Il comando ENVELOPE consente di cambiare rapidamente il volume di una nota come in uno strumento musicale. Cosi' si puo' specificare il crescendo e il decrescendo del volume di una nota. Viene specificato un gruppo di coppie di volume e tempo per volta. La costante di volume e' compresa tra 0 e 15 e il tempo e' misurato in intervalli unita' di 3.2 millisecondi. Per esempio:

```
ENVELOPE 0 10,2;15,2;14,4;12,5;8,10;0
```

predispone un envelope di volume di questo tipo:



tempo dopo il comando SOUND
(intervalli di 3.2 millisecondi)

Cosi' ogni comando SOUND che fa riferimento alla envelope 0 sopra disegnata produce un suono avente il volume come evidenziato dal grafico.

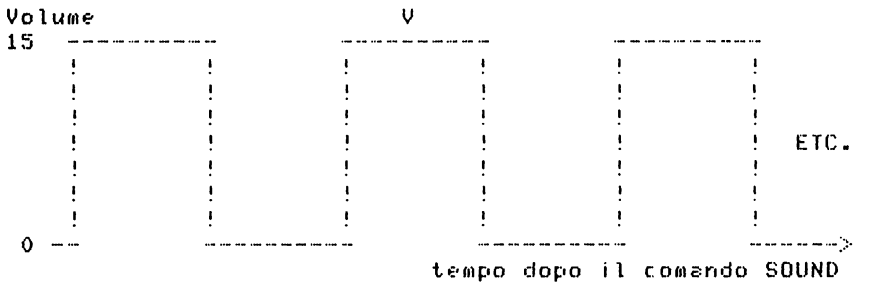
Variando l'envelope varia la qualita' del suono prodotto.

Il volume dato nel comando SOUND e' moltiplicato da quello dell'envelope. Se il comando SOUND richiede un volume di 8 unita', che e' 8/15 del volume pieno, e l'envelope richiede 4 unita', che e' circa 1/4 del massimo, allora questi valori vengono modificati e si avra' un volume pari a 2/15 del massimo (1/4 x 8/15 = 8/60 = 2/15).

Il comando ENVELOPE puo' finire, come nel caso precedente, con un solo volume o con una coppia di volume/tempo. Nel primo caso il volume finale rimane costante indefinitamente, mentre nel secondo si ha una ripetizione continua di tutta la ENVELOPE. Per esempio:

```
ENVELOPE 0 15,10;0,10;
```

determina una serie di volumi del tipo:



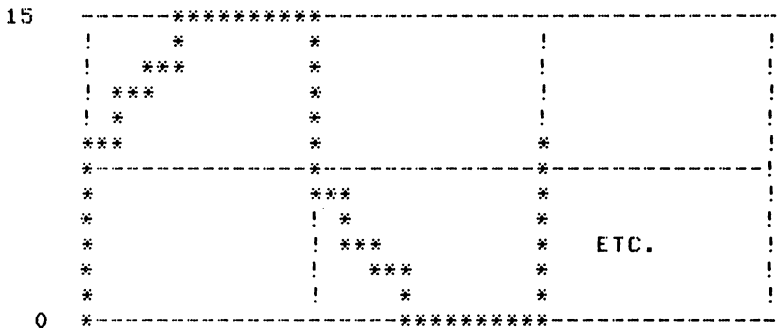
tempo dopo il comando SOUND

L'envelope piu' semplice e':

```
ENVELOPE 0 15
```

che non produce un effetto sonoro con il comando SOUND, dato che tutti i volumi sono moltiplicati per 15/15.

L'interprete BASIC pone delle limitazioni alla rapidita' con la quale puo' variare il volume in ogni canale. Il massimo cambiamento e' D/2+1, dove D e' la differenza tra il volume richiesto e quello in essere. Allora l'uscita reale di volume per l'ENVELOPE 0 15,10;0,10; sara':



Questo aiuta a ridurre i suoni spuri prodotti dai cambiamenti di volume.

Il generatore di rumore e' controllato dal comando NOISE che puo' controllare solo il volume e l'envelope. Per esempio:

```
NOISE 0 15
```

seleziona l'envelope 0 ed il volume 15.

Il comando SOUND controlla anche il TREMOLO ed il GLISSANDO.
Il TREMOLO e' semplicemente una rapida variazione di volume di + o - 2 unita'. Esso da' al suono un effetto WARBLING.
Il GLISSANDO e' un effetto per il quale una nota non parte immediatamente in un canale con la frequenza richiesta ma vi arriva "scivolando" dalla frequenza precedente.
Questo effetto e' proprio della Chitarra Hawaiana. Il GLISSANDO e il TREMOLO sono controllati da un parametro del comando SOUND e precisamente il quarto. Questo parametro e' formato da due bits; il bit basso attiva il tremolo ed il bit alto il glissando (Vedi anche paragrafo 6.2.13.2.).Nell'esempio:

```
SOUND 0 0 13 1 FREQ(1000)
```

si usa il canale 0, l'envelope 0, il volume 13 con tremolo, cosi' il volume varia rapidamente tra 11 e 15.

Nell'esempio:

```
SOUND 0 0 15 2 FREQ(5000)
```

il volume aumenta a 15 e scivola glissando alla frequenza 5000 Hz.

I tre comandi visti servono a controllare completamente le possibilita' sonore del DAI. Dal momento che e' molto facile modificare la frequenza ed il volume dei suoni, il DAI si presta per effettuare delle prove di imitazione della voce umana (Vedi anche paragrafo 6.2.13.6.).

6.2.13.2. SINTASSI DI SOUND

Vediamo alcuni esempi:

```
(1) SOUND <CHAN><ENV><VOL><TG>FREQ(PERIOD)
(2) SOUND <CHAN> OFF
(3) SOUND OFF
```

CHAN e' una espressione il cui valore deve essere compreso tra 0 e 2. Seleziona gli oscillatori 0,1 e 2.

ENV e' una espressione il cui valore deve essere 0 o 1. Seleziona quale dei due envelope precedentemente definiti deve essere usato (Vedi anche paragrafo 6.2.13.3.).

VOL e' una espressione il cui valore deve essere compreso tra 0 e 15. Seleziona il volume del suono. Esso viene moltiplicato per il volume specificato nell'envelope.

TG e' una espressione che puo' valere da 0 a 3, con il seguente significato:

0	NO TREMOLO e NO GLISSANDO
1	SI TREMOLO e NO GLISSANDO
2	NO TREMOLO e SI GLISSANDO
3	SI TREMOLO e SI GLISSANDO

PERIOD e' una espressione il cui valore deve essere compreso tra 0 e 65535. Seleziona il periodo del suono in unita' di intervalli di 1/2 microsecondo e corrisponde agli Hz desiderati.

6.2.13.3. SINTASSSI DI ENVELOPE

Vediamo alcuni esempi:

(1) ENVELOPE <ENV> [<V>,<T>;.....] <V>,<T>;
(2) ENVELOPE <ENV> [<V>,<T>;.....] <V>

ENV e' una espressione che puo' valere 0 o 1. Seleziona quale tra i due possibili envelope si sta definendo.

V e' una espressione il cui valore deve essere compreso tra 0 e 15. Seleziona un livello di volume per il quale va moltiplicato il volume definito dal comando SOUND.

T e' una espressione il cui valore deve essere compreso tra 1 e 254. Seleziona il tempo durante il quale deve essere usato il corrispondente volume. E' misurato in unita' di 3.2 millisecondi.

NOTA:Le parti del comando tra parentesi quadre sono opzionali e possono essere ripetute quante volte si vuole.

6.2.13.4. SINTASSI DI NOISE

Vediamo alcuni esempi:

(1) NOISE <ENV> <VOL>
(2) NOISE OFF

ENV e' un'espressione il cui valore puo' essere 0 o 1.

VOL e' una espressione il cui valore deve essere compreso tra 0 e 15. Esso e' un numero di 4 bits; i 2 bits alti, modificati dall'envelope specificato, controllano il volume, e i 2 bits bassi controllano la frequenza.

6.2.13.5. FREQ

L'istruzione:

A = FREQ(1000)

pone nella variabile A un numero che puo' essere inviato ad uno dei 3 canali dei generatori per impostare una frequenza di 1000 Hz.

6.2.13.6. SINTESI DEI SUONI VOCALI

6.2.13.6.1. TALK

INDIRIZZI TALK

```
Codice  Dati

 0      2 bytes FREQ canale 0
 2      2 bytes FREQ canale 1
 4      2 bytes FREQ canale 2
 8      1 byte VOL canale 0
 9      1 byte VOL canale 1
 A      1 byte VOL generatore rumori
 C      2 bytes ritardo in unita' di Millisecondi
 D      chiamata codice macchina
 FF     END
```

Blocchi di dati

```
   Indirizzo  contenuto commento
#2000   20 00     09C4   pone canale 0 freq. 800
        20 02     1A0A   pone canale 1 freq. 300
        20 08     0F     massimo vol. canale 0
        20 09     0F     massimo vol. canale 1
        20 0C     FEFE   pone e aspetta per....msec
        20 08     00     pone vol. basso
        20 09     00
        20 0D     0050   codice macchina in 5000
        20 FF            END

#5000   00         (LXI H, VARPTR(Q(0))) 21 00 20
#5004              (RETURN)              C9
```

Esempio:

```
  3   CLEAR 1000
  4   DIM Q(100)
  5   B% = VARPTR(Q(0))
 10   READ A%
 20   POKE B%, A% : B% = B% + 1
 30   IF A% <> #FF GOTO 10
 40   TALK VARPTR(Q(0))
 50   WAIT TIME 10
 60   GOTO 40
 80   DATA 0,9,#C4,2,#1A,#A,8,#F,9,#F
 90   DATA #C,#FE,#FE,8,0,9,0,#FF
```

6.2.14. FUNZIONI ARITMETICHE E DI STRINGA

Segue una lista delle funzioni fornite dal BASIC. Alcune sono funzioni matematiche, altre manipolano le stringhe. Per ogni funzione gli argomenti sono racchiusi tra parentesi. Le funzioni forniscono un risultato che puo' essere usato come una qualsivoglia variabile del suo tipo.

Esempi:

```
A = 3.0 + 2.1
A = SIN(3.0)+ 2.1
```

NOTA IMPORTANTE: Gli argomenti delle funzioni trigonometriche devono essere espressi in Radianti. 1 Radiante = 180/PI gradi = 57.2958 gradi. Pertanto se l'argomento X di una funzione, ad es. SIN(X), e' espresso in gradi si deve scrivere: SIN(X*PI/180).

6.2.14.1. ABS(X)

Fornisce un numero reale uguale al valore assoluto dell'espressione X.

6.2.14.2. ACOS(X)

Fornisce espresso in radianti,l'angolo il cui coseno e' X. Il risultato e' compreso tra -PI/2 e + PI/2.

6.2.14.3. ALOG(X)

Fornisce l'antilogaritmo in base 10 di X.

6.2.14.4. ASC(X$)

Fornisce il numero intero decimale che rappresenta il codice ASCII del primo carattere dellla stringa X$.

6.2.14.5. ASIN(X)

Fornisce l'arcoseno di X in radianti. Il risultato e' compreso tra -PI/2 e +PI/2, X deve essere tra -1 e +1.

6.2.14.6. ATN(X)

Fornisce l'arcotangente di X in radianti.

6.2.14.7. CHR$(I)

E' l'inverso della funzione ASC. Fornisce una stringa di 1 carattere il cui valore ASCII e' I. I deve essere compreso tra 0 e 255. Es. CHR$(65) fornisce il carattere "A".

6.2.14.8. COS(X)

Fornisce il coseno di X in radianti. X deve essere compreso tra 0 e 2PI.

6.2.14.9. EXP(X)

Fornisce il valore di "e" (2.71828) elevato a X. "e" e' la base dei logarirtmi naturali. Il valore massimo dell'argomento che puo' essere usato senza dare overflow dipende se si usano le opzioni matematiche hardware o software. Per opzioni hardware: -32<X<32. Per opzioni software: -43<X<43.

6.2.14.10. FRAC(X)

Fornisce un numero reale uguale alla parte frazionaria dell' argomento.

6.2.14.11. HEX$(I)

Fornisce una stringa di caratteri che rappresenta il valore esadecimale di I. I puo' variare tra 0 e 65535.

6.2.14.12. INT(X)

Fornisce il piu' grande numero reale intero (nella forma floating-point, ma intero) minore di X. Esempi:
INT(.23) = 0 INT(7) = 7.0 INT(-2.7) = -3.0
INT(1.1) = 1.0 INT(43.99) = 43.0
NOTARE CHE : INT(-1) = -2.0

6.2.14.13. LEFT$(X$,I)

Fornisce una stringa formata dagli I caratteri piu' a sinistra di X$.

6.2.14.14. LEN(X$)

Fornisce un numero intero che da' la lunghezza in caratteri della stringa X$.

6.2.14.15. LOG(X)

Calcola il logaritmo naturale (base e) di X.

6.2.14.16. LOGT(X)

Calcola il logaritmo in base 10 di X.

6.2.14.17. MID$(X$,I,J)

Fornisce J caratteri della stringa X$ partendo dalla posizione I. Il primo carattere della stringa X$ ha posizione 0.

6.2.14.18. PI

Fornisce il numero reale 3.14159.

6.2.14.19. RIGHT$(X$,I)

Fornisce una stringa formata dai caratteri piu' a destra della stringa X$.

6.2.14.20. RND(X)

Genera o via hardware o via software un numero a caso. E precisamente:

-Se X < 0 inizia una nuova sequenza di numeri a caso prendendo X come spunto, la sequenza e' generata via software. Lo stesso X produce la stessa sequenza. I numeri generati sono compresi tra 0 e 1.
-Se X > 0 fornisce il prossimo numero a caso disponibile nella sequenza in essere. Il numero e' compreso tra 0 e X.
-Se X = 0 fornisce un numero a caso generato via hardware e compreso tra 0 e 1.

Esempio:

```
10 CLEAR 1000
15 DIM B%(100)
20 INPUT C%
```

```
30 FOR A% = 1 TO 20
40 B%(A%) = RND(C%)
50 PRINT B%(A%)
60 NEXT A%
```

6.2.14.21. SGN(X)

Fornisce 1.0 se X>0, 0 se X=0, -1.0 se X<0.

6.2.14.22. SIN(X)

Calcola il seno di X. X deve essre espresso in radianti.

6.2.14.23. SPC(I)

Fornisce una stringa formata da I spazi. 0<I<=255.

6.2.14.24. SQR(X)

Fornisce la radice quadrata di I. Si ha errore se I<0.

6.2.14.25. STR$(X)

Fornisce il numero X sotto forma di stringa.

6.2.14.26. TAB(I)

Fornisce una stringa composta da tanti spazi quanti sono necessari per spostare il cursore sullo schermo nella colonna I. Lo spostamento puo' essere solo verso destra.

6.2.14.27. TAN(X)

Fornisce la tangente di X (espresso in radianti).

6.2.14.28. VAL(X$)

Fornisce un numero reale uguale al numero rappresentato dalla stringa X$.
X$ deve rappresentare un numero floating point.

6.2.15. OPERATORI LOGICI E ARITMETICI

NOTA: Nel seguito: int sta per intero, fpt sta per floating-point, str sta per stringa e logi sta per logico. Inoltre i valori int sono convertiti a fpt prima di usarli in espressioni miste int-fpt.

OPERATORE	USO	RISULTATO
+(addizione)	int + int	int
	fpt + int	fpt
	int + fpt	fpt
	fpt + fpt	fpt
	str + str	str
-(sottrazione)	stesse regole dell'addizione ma non lavora su stringhe	
/(divisione)	come sopra	
*(moltiplica)	come sopra	
freccia in su	come sopra	sempre fpt
IAND	int...int	stesse regole per i 6
IOR	int...fpt	operatori logici
IXOR	fpt...fpt	
MOD	fpt...int	sempre int (troncato automaticamente se fpt)
SHL		
SHR		
INOT	int	int
= uguale	str...str	
> maggiore	fpt...fpt	sempre logi
< minore	fpt...int	stesse regole per tutti
<> diverso	int...fpt	gli operatori
>= magg. ug.	int..int	relazionali
<= min.ug.		
AND	logi	logi
OR	logi	logi

Esempi:

Operazione	Risultato	Tipo risultato
1 + 2	3	int
1.0+2.0	3.0	fpt
1.0+2	3.0	fpt
3*4	12	int
3 elevato 4	81.0	fpt
12.0/4.0	3.0	fpt
12.0/4	3.0	fpt
12/4	3	int
11/4	2	int
3 IAND 2	2	int
3.0 IAND 6.0	2	int

Operazione	Risultato	Tipo risultato
3.14 IAND 6.72	2	int
3 SHL 2	12	int
3.2 SHL 2.1	12	int
7 = 4	FALSO	logi
3.0>2.1	VERO	logi
"AB"<"AH"	VERO	logi
"D"="D"	VERO	logi
7.1=7	FALSO	logi
7.0=7	VERO	logi
3<4 OR 7=8	VERO	logi
3=7 AND 9<10	FALSO	logi

Altri esempi:

	Risultato	Tipo risultato e note
63 IAND 16	16	int .Poiche' 63 in binario si scrive 111111 e 16 si scrive 10000, il risultato dell'IAND e' 10000 binario cioe' 16.
15 IAND 14	14	int .15 corrisponde in binario a 1111 e 14 a 1110, allora 15 IAND 14 da' come risultato 1110 cioe' 14.
-1 IAND 8	8	int .-1 corrisponde al numero binario 111...111 (tutti uno) e 8 al binario 1000, cosi' -1 IAND 8 da' 1000 ovvero 8.
4 IAND 2	0	int .4 e' in binario 100 e 2 e'10. In questo caso il risultato e' 0 perche' i due operandi non hanno alcun bit in posizione corrispondente
4 IOR 2	6	int .Il binario 100 composto con lo IOR con il binario 10 da per risultato 110 cioe' 6 decimale.
10 IOR 10	10	int .Qualsiasi numero IOR con se stesso da per risultato lo stesso numero
-1 IOR -2	-1	int .Il numero binario 111..111(-1) composto con 111..10(-2) da per risultato 111...111 cioe' -1.

Seguono le tabelle della verita' per gli operatori logici:

Operatore	Arg1	Arg2	Risultato
IAND	1	1	1
	0	1	0
	1	0	0
	0	0	0
IOR	1	1	1
	1	0	1
	0	1	1
	0	0	0
IXOR	1	1	0
	1	0	1
	0	1	1
	0	0	0
INOT	1	-	0
	0	-	1

Un tipico uso di questi operatori e' quello di controllare i bits della REAL WORLD delle porte di input.

Quando si lavora sui bytes il bit di posizione 7 e' il piu' significativo e quello di posizione 0 il meno significativo.

CAPITOLO 7

U T I L I T Y
I N L I N G U A G G I O
M A C C H I N A

7.1. INTRODUZIONE

Con l'ausilio di queste routine si puo' creare e correggere un programma in linguaggio macchina. Si ha la possibilita' di salvare i registri principali in qualunque momento. Essi possono essere evidenziati sul video e modificati e questo puo' essere necessario quando si lavora nel modo del REAL BUS WORLD e si controllano i TIMER e gli INTERRUPT. Si puo' evidenziare una parte del contenuto della memoria e modificarlo ed anche prelevarlo da disco o cassetta o memorizzarlo su disco o cassetta. Nel provare un programma in codice macchina si possono predisporre punti di arresto o inserire il Tracing.

7.2. INTERFACCIA CON IL BASIC

Quando si usa il comando UT in un programma BASIC e quindi si ricorre alle UTILITY si ottiene il messaggio:

P. C. UTILITY V3.3

Il messaggio e' seguito dal carattere di PROMPT ">". Quando appare questo comando le UTILITY sono in attesa di un comando da parte dell'utente. Nella maggior parte dei casi il formato dei comandi e' di una lettera seguita da uno o piu' numeri; non e' richiesto un separatore tra la lettera ed il primo numero. I numeri si scrivono sempre in esadecimale e terminano o con uno spazio o con il RETURN. Le utility usano sempre le ultime 4 o 2 cifre esadecimali scritte, a seconda del tipo di comando, ignorando eventuali altre cifre scritte prima di queste. Per questa ragione G12345678 e' considerato come G5678, dato che richiede un numero di 4 cifre. E, sempre per la stessa ragione, F0000 FFFF 5566 e' considerato come F0000 FFFF 66, dato che il terzo numero deve essere di 2 cifre. Se il numero richiesto, di 2 o 4 cifre, viene troncato ad un numero minore di cifre, esso viene ugualmente accettato. Quindi: G0003, G003, G03 e G3 sono equivalenti.

In caso di errore le utility evidenziano il carattere "?" e questo e' l' unico tipo di messaggio di errore previsto.

Quando si sta eseguendo un programma con Tracing o si sta evidenziando il contenuto della memoria si puo' usare il tasto BREAK per interrompere.

Alcune funzioni richiedono l'uso di un carattere terminale speciale al posto dello spazio o del RETURN. Questo carattere si chiama "ESCAPE" e si ottiene con il tasto "CURSOR LEFT" (freccia a sinistra).

Nel seguito, durante la descrizione dei comandi si usano le seguenti abbreviazioni:

SP per SPAZIO
CR per RETURN
ES per ESCAPE.

Per uscire dalle utility e tornare al BASIC si deve scrivere "B".

7.3. COMANDI UTILITY

I comandi utility appartengono a 3 classi:

(1) Comandi per la memoria;
(2) Comandi per i registri;
(3) Comandi per I/O esadecimale.

Nel seguito si usano le seguenti abbreviazioni:

ADR per ADDRESS
LADR per LOW ADDRESS
HADR per HIGH ADDRESS
DADR per DESTINATION ADDRESS
BADR per BASE ADDRESS di riferimento PROM

Gli indirizzi sono formati da 4 cifre esadecimali. Se si scrivono piu' di 4 cifre esadecimali vengono accettate solo le ultime quattro. Questo consente, in caso di errore, di continuare a digitare finche' le ultime 4 cifre non corrispondono all'indirizzo desiderato.

Gli argomenti dei comandi possono essere separati tra loro da spazio o virgola.

7.3.1. CLASSE 1: COMANDI PER LA MEMORIA

Con questi comandi e' possibile:
-provare un programma con il Tracing
-provare un programma eseguendolo passo passo
-evidenziare blocchi di memoria
-inserire programmi o dati.

7.3.1.1. LOOK

Il formato del comando e':

L ADR LADR HADR

Quando si manda in esecuzione il comando con il tasto RETURN ha inizio il trasferimento nel modo utente. Viene caricato il Registro Contatore del Programma (PROGRAM COUNTER) con l'indirizzo specificato. Dopo l'esecuzione di ogni istruzione viene evidenziato il contenuto dei registri della CPU (Central Processing Unit o Unita' Centrale del calcolatotre) cosi':

I = 1043 A = 02 F = 02 B = 00 C = 00 D = 00 E = 05 H = 00
L = 00 S = P = 1045

Dove I e' l'indirizzo della istruzione appena eseguita e si ha il Tracing di tutte le istruzioni comprese tra i due indirizzi specificati nel comando. Per sospendere temporaneamente l'esecuzione del programma si deve tenere premuto il tasto BREAK durante la scrittura di tutta la linea attuale. Per continuare dopo la sospensione basta premere L e poi RETURN. Il Tracing continuera' dall'indirizzo di P.

Dato che durante la sospensione del programma tutte le funzioni possono essere usate senza per questo influenzare la continuazione del programma dopo L e RETURN, l'utente ha molta liberta' di azione.

Se si desidera continuare l'esecuzione ripartendo da un altro punto si deve dare il comando:

L LADR HADR

e si ricomincia il Tracing dal nuovo indirizzo.

Ricordate che quando si usa la funzione L con i tre argomenti il sistema viene inizializzato come descritto in 4.1., mentre se si usano 2 argomenti o nessuno si ha solo il riposizionamento dei registri. Per questa ragione si puo' continuare l'esecuzione di un programma dopo il BREAK. L'uso del tasto BREAK consente di salvare il Registro Contatore del Programma forzando l'uscita da un ciclo errato.

7.3.1.2. DISPLAY

Il formato del comando e':

D LADR HADR

Dopo il RETURN il sistema evidenzia in esadecimale il contenuto dei bytes compresi tra i due indirizzi. Ogni linea

e' preceduta dall'indirizzo del primo byte mostrato. L'uso del tasto BREAK interrompe l'operazione.

7.3.1.3. GO

Il formato del comando e':

G ADR

Con questo comando parte l'esecuzione di un programma , dopo l'inizializzazione del sistema. Se nel comando c'e' l'indirizzo allora l'esecuzione parte da quell'indirizzo. Se l'indirizzo manca viene preso l'indirizzo contenuto nel Program Counter, vengono ripristinati dall'area di salvataggio solo i registri del microprocessore 8080 e non vengono toccati i bytes GIC e TICC. Si puo' usare il comando G senza indirizzo per ripartire dopo un BREAK.

7.3.1.4. FILL

Il formato del comando e':

F LADR HADR byte

Serve per riempire tutta la memoria indicata, da LADR a HADR, con il valore del byte. Se manca il byte (contenente la costante) viene preso lo zero per il riempimento.
Esempi:

F1010 101A FF

riempie i bytes da 1010 a 101A con FF

F1010 101A

riempie i bytes da 1010 a 101A con zero.

7.3.1.5. SUBSTITUTE

Il formato del comando e':

S ADR

ed esso puo' essere reso esecutivo o da spazio o da RETURN. Viene evidenziato sullo schermo il contenuto di ADR e si puo' sostituirlo con un nuovo valore scrivendolo alla tastiera e facendolo terminare con spazio o con virgola o con RETURN,
Appena terminata l'operazione relativa al byte ADR, viene evidenziato il contenuto del byte successivo e si puo' ancora sostituirlo. Questa sequenza di operazioni continua

fino all'uso del tasto ESC. Se non si vuole modificare il contenuto di un byte basta premese lo spazio o il RETURN.
  Esempio:

```
  S1000 3D-8F 1A  CB-3F 81-AE 78-FA
```

l'utente ha fatto entrare in ordine 8F, niente, 3F, AE, FA e quindi il contenuto dei bytes sara':

```
1000 contiene 8F     1001 contiene 1A    1002 contiene 3F
1003 contiene AE     1004 contiene FA
```

7.3.1.6. MOVE

  Il formato del comando e':

```
     M LADR HADR DADR
```

ed esso trasferisce il blocco di memoria compreso tra LADR e HADR in un blocco di memoria che inizia in DADR. Ovviamente il blocco di memoria origine resta invariato.
  Questo comando puo' essere molto utile in fase prova programmi per creare degli spazi in memoria per inserire nuove istruzioni. Analogamente puo' essere usato per inserire nuovi dati in blocchi di dati gia' esistenti. Bisogna pero' tener presente che se si inseriscono nuove istruzioni possono venire a crearsi degli errori dovuti al fatto che cambiano i riferimenti ad indirizzi considerati fissi precedentemente.

7.3.2. CLASSE 2: COMANDI PER I REGISTRI

7.3.2.1. EXAMINE

  Il formato del comando e':

```
     X
```

e serve per evidenziare sullo schermo i seguenti registri della CPU:

- -ACCUMULATORI
- -FLAGS
- -REGISTRI da B a L
- -STACK POINTER
- -PROGRAM COUNTER (PC)

  Esempio:

```
        X
A = 00   F = 46   B = 20   C = 44   D = 10
E = BF   H = 11   L = 7A   S = 11BE P = 1040
```

L'assegnazione dei bits di FLAG e' la seguente:

Posizione	Significato
7	segno
6	zero
5	sempre zero
4	carry ausiliario
3	sempre zero
2	parita'
1	sempre 1
0	carry

7.3.2.2. EXAMINE REGISTER

Il formato del comando e':

```
X registro
```

ed il suo effetto e' identico a quello del comando SUBSTITUTE, solo che lavora sui registri speciali.
Per esempio si voglia inizializzare l'accumulatore al valore 35 ed il registro B al valore FF. Possiamo procedere in due modi. PRIMO MODO:

```
XA 00-35 46- 20-FF
```

infatti dopo il contenuto di A appare il contenuto di F e lo si lascia inalterato, mentre si modifica il successivo che e' B. SECONDO MODO:

```
XA 00-35
XB 20-FF
```

e ricordate che dopo il primo comando si deve usare ESC per terminarlo e cosi' pure dopo il secondo.

7.3.2.3. VETTORE DELLE INTERRUZIONI EXAMINE

Il formato del comando e':

```
V
```

ed esso evidenzia i contenuti dei registri usati per l'inizializzazione e del vettore per i trasferimenti dovuti alle interruzioni (Interrupt Transfer Vector).
Esempio:

```
0=00  M=00  T=10  G=20  1=106F  2=1069
3=0040  4=0040  5=0040  6=0040  7=106F
```

7.3.2.4. VETTORE EXAMINE BYTES

Il formato del comando e':

V byte

ed esso consente di modificare il contenuto dei bytes di inizializzazione e/o del vettore delle interruzioni. Il comando si comporta come SUBSTITUTE e EXAMINE. Per uscire dal comando si deve usare il tasto ESC.

7.3.3. CLASSE 3: COMANDI I/O ESADECIMALE

7.3.3.1. READ

Il formato del comando e':

R ADR

ed il parametro ADR puo' anche mancare. Esso legge un file binario dalla cassetta o dal disco, non appena questi vengono avviati. Mentre legge il nastro viene fatto il controllo di Checksum per ogni record. Se si ha un errore il sistema si ferma e viene restituito il controllo all'utente. Si puo' cosi far tornare indietro il nastro e riavviare la lettura. Quando viene letto e riconosciuto il carattere di END OF FILE la lettura termina (Vedi anche paragrafo 7.3.3.3.).

7.3.3.2. WRITE

Il formato del comando e':

W LADR HADR

ed esso scrive il blocco di memoria da LADR a HADR sul nastro o sul disco. Il formato dell'output e' esadecimale impaccato secondo lo schema seguente:

```
     W600 60F

10060000B7C8CD380523C3000060E0DCD3805C50188
! !    ! !                                  !
! !    ! !                                  ---Checksum
! !    ! !
! !    ! ---Dati
! !    !
! !    ---Codice:00=dati,01=END
! !
! ---Indirizzo iniziale caricamento
!
---Numero di bytes del campo dati in esadecimale
```

7.3.3.3.ESEMPI

Negli esempi che seguono per dare chiarezza a quello che si scrive vengono divisi con uno spazio i diversi componenti del comando, ma nella realta' questi spazi non devono essere usati. Ricordiamo che SP sta per spazio e CR per RETURN.

```
     W 0 SP FFF SP GIORGIO  CR
```

Scrive la zona di memoria dal byte 0 al byte FFF su nastro o disco sotto il nome GIORGIO (nome del file).

```
     W 0 SP 1F  CR
```

Scrive la zona di memoria da 0 a 1F su cassetta senza nome di file. Sul disco non si puo' registrare un file senza nome.
I dati quando vengono riletti sono caricati nello stesso spazio dal quale erano stati prelevati.

```
     R 1000 SP GIORGIO
```

Rilegge il file GIORGIO, ma lo carica in memoria 1000 bytes piu' avanti di dove era stato prelevato.

```
     R CR
```

Viene letto in memoria il primo file disponibile da cassetta.
Tutti i files trattati da questi comandi sono di tipo 1. Essi non possono essere trattati dal BASIC. Questi comandi non possono agire su files non di tipo 1.
Nell'usare questi comandi si deve fare attenzione a non sbagliare i tasti perche' non si possono correggere i caratteri battuti.

7.3.4. TABELLA CARATTERI

Dec.	Carattere	Dec.	Carattere	Dec.	Carattere
000	NUL	040	(	084	T
001	SOH	041	)	085	U
002	STX	042	*	086	V
003	ETX	043	+	087	W
004	EOT	044	,	088	X
005	ENQ	045	-	089	Y
006	ACK	046	.	090	Z
007	BEL	047	/	091	[
008	CH DEL	048	0	092	barra sin.
009	TAB	049	1	093	]
010	LF	050	2	094	freccia su
011	VT	051	3	095	freccia sin.
012	FF	052	4	096	'
013	CR	053	5	097	a
014	SO	054	6	098	b
015	SI	055	7	099	c
016	Curs. su	056	8	100	d
017	Curs. giu'	057	9	101	e
018	Curs. sin.	058	:	102	f
019	Curs. dest.	059	;	103	g
020	shift+frec=	060	<	104	h
	cia su	061	=	105	i
021	shift+frec=	062	>	106	j
	cia giu'	063	?	107	k
022	shift+frec=	064	@	108	l
	cia sin.	065	A	109	m
023	shift+frec=	066	B	110	n
	cia destra	067	C	111	o
024	CAN	068	D	112	p
025	EM	069	E	113	q
026	SUB	070	F	114	r
027	lire	071	G	115	s
028	C+/	072	H	116	t
029	GS	073	I	117	u
030	RS	074	J	118	v
031	US	075	K	119	w
032	spazio	076	L	120	x
033	!	077	M	121	y
034	"	078	N	122	z
035	#	079	O	123	ap. graffa
036	$	080	P	124	barra vert.
037	%	081	Q	125	chiusa graf.
038	&	082	R	126	tilde
039	'	083	S	127	DEL

7.3.5. LISTA DI ALCUNI POKE UTILI

```
POKE #2C4,#FF   forza un BREAK
```

OUTPUT

```
POKE #131, 0  uscita su schermo + RS 232
         , 1  uscita su schermo
         , 2  uscita sul buffer di EDIT
         , 3  uscita su disco
```

INPUT

```
POKE #135, 0  input da tastiera
         , 1  input da stringa
         , 2  input dal buffer di EDIT in area programma
```

CONTROLLO NASTRO

```
POKE #40, #28   Cassetta 1 ON
        , #18   Cassetta 2 ON
        , #30   Cassetta 1 e 2 OFF

POKE #13D, #10  Attivazione porta 1 cassetta
         , #20  Attivazione porta 2 cassetta
```

ACCENSIONE UNITA' DISCHI

```
POKE #730, #30 Attivazione disco 0
         , #31 Attivazione disco 1
```

AM 9511

```
UT
>SFB00
>
>B
```

UNITA' DISCHI

```
UT
>Z3
>XA  30   uso del disco 0
     31   uso del disco 1
>G B6
>B

PARTE ALTA STACK #F900
PARTE BASSA STACK #F800
```

7.3.6. ATTIVAZIONE DISCHI E CASSETTE

ATTIVAZIONE DISCHI (da 2C5 a 2E2)

```
2C5 C3 58 05 C3 F2 05 C3 12 06 C3 A1
2D0 05 C3 FB 05 C3 FC 06 C9 00 00 C3 75 06 C3 29 06
2E0 C3 5C 06 (2E2)
2A0 08 5D 08 5E 08
```

ATTIVAZIONE CASSETTE (da 2C5 a 2E2)

```
2C5 C3 B8 D2 C3 F1 D2 C3 27 D4 C3 25
2D0 D3 C3 40 D3 C3 45 D4 C3 A2 D3 C9 00 00 C9 00 00
2E0 C3 B4 DD (2E2)
2A0 33 ED 03 F6 03 50 B3 C5 E8
```

7.3.7. PROTEZIONI SOFTWARE

Procedura operativa:

1) Scrivere un programma in BASIC e non scrivere frasi REM

2) UT

3) D2A1 2A4 (puntatori)

```
   2A1  #       #       #       #
      Basso  Alto   Basso    Alto
         VAL 1         VAL 2
```

4) Salvare con SAVE su cassetta con:

W(VAL1+1) (VAL2) NOME-FILE (senza doppi apici)

5) Proteggere:

F(VAL1+1) (VAL2) C (C=codice esadecimale per avanzamento)

6) B (ritorno al BASIC)

7) Salvare con SAVE su cassetta scrivendo:

SAVE "NOME-FILE"

Quando ricaricate il programma esso non puo' essere listato e non si puo' usare l'EDIT.

7.3.8. INTERVENTI DI EMERGENZA

Cosa fare se si ha un RESET accidentale durante l'introduzione di un programma o alla fine di un programma:

1) Premere BREAK
2) Scrivere UT e RETURN
3) Scrivere S29F e 6 spazi, si otterra' b a x x x x
4) Prendere nota di b a x x x x
5) Premere il tasto ESCAPE (freccia verso sinistra)
6) Scrivere S a b spazio, si otterra' x x
7) Prendere nota di x x
8) Premere il tasto ESCAPE (freccia verso sinistra)
9) Premere B (BASIC)

Se provocate voi un RESET operate cosi':

1) Scrivere UT e RETURN
2) Scrivere S29F e 6 spazi, si otterra' x y & & & &
3) Cambiare le 6 posizioni se diverse da quelle da voi annotate
4) Scrivere S a b spazio e cambiare le due posizioni se diverse da quelle annotate
5) Premere il tasto ESCAPE (freccia verso sinistra) e poi B
6) Scrivere EDIT e poi premere BREAK e spazio

COME SALVARE E RICARICARE UN DISEGNO

Per salvare un disegno operare cosi':

1) Premere BREAK
2) Scrivere MODE ? A (dove ? e' il MODE nel quale vi trovate)
3) Scrivere UT e poi RETURN
4) Scrivere W XXXX BFFF DISEGNO 1

dove XXXX e' per il MODE 2A B350
3A A440
5A 5670

CAPITOLO 8

E S E M P I D I
P R O G R A M M I

8.1. PREMESSA

I programmi che seguono mostrano molte delle capacita' del vostro calcolatore con particolare riguardo alla grafica ed al sonoro.

Troverete anche utili comandi in linguaggio macchina come supplemento a quelli illustrati nel manuale, e alcuni sottoporogrammi che potranno esservi utili.

8.2. AGENDA

QUESTO PROGRAMMA PERMETTE DI MEMORIZZARE NOME,COGNOME E INDIRIZZO DI AL MASSIMO 20 PERSONE. UNA VOLTA MEMORIZZATI I DATI E' POSSIBILE RIVEDERLI SINGOLARMENTE O IN BLOCCO.
NEL PRIMO CASO SI PUO' SPECIFICARE UNO QUALSIASI DEI DATI DELLA PERSONA CONSIDERATA (NOME, COGNOME, INDIRIZZO).
SI RICORRE,INVECE,AL SECONDO CASO QUANDO NON SI RICORDA NESSUN DATO SPECIFICO.
POICHE' I DATI SONO ORGANIZZATI IN VETTORI NON E' POSSIBILE MEMORIZZARLI PERMANENTEMENTE . ESSI VENGONO CANCELLATI OGNI VOLTA CHE E' CANCELLATO IL PROGRAMMA.

```
1     GOSUB 50000
2     CLEAR 15000
5     DIM NAME$(50),SURNAME$(50),ADRESS$(50)
10    PRINT CHR$(12)
99    PRINT"*****************************
      ********************************"
100   PRINT"*              Questo e' un programma
      dimostrativo            *"
110   PRINT"*                per gente che non si
      intende di              *"
120   PRINT"*                                COMP
      UTER                          *"
130   PRINT"*****************************
      ********************************"
140   GOSUB 10000
160   PRINT CHR$(12)
170   WAIT TIME 100
195   CURSOR 0,18
```

```
200  PRINT"################################
     ################################"
210  PRINT"#
                                        #"
220  PRINT"#           Faremo una lista
     di p.e. 20 persone con             #"
240  PRINT"#
                                        #"
250  PRINT"#        1) NOME
                                        #"
260  PRINT"#        2) COGNOME
                                        #"
270  PRINT"#        3) NUMERO
                                        #"
280  PRINT"#        4) INDIRIZZO
                                        #"
290  PRINT"#
                                        #"
300  PRINT"################################
     ################################"
400  GOSUB 10000
405  PRINT CHR$(12)
410  PRINT"################################
     ################################"
420  PRINT"# NOTE :-se fai un errore premi
               !CHAR DEL!          #"
430  PRINT"#        -NON PREMERE MAI il bottone
     reset                    #"
440  PRINT"#        -ogni comando al computer
     deve essere                #"
450  PRINT"#         seguito dal comando
      RETURN.                       #"
455  PRINT"#        -Quando hai scritto tutti i nomi
     che vuoi           #"
457  PRINT"#         introdurre premi HALT
                        #"
459  PRINT"#        -Se hai finito premi FINE
                        #"
460  PRINT"################################
     ################################"
470  GOSUB 10000
500  PRINT CHR$(12)
510  PRINT"============================
     ============================="
520  PRINT"+                     MENU
                                        +"
530  PRINT"+                    ------
                                        +"
540  PRINT"+        1) Nuovi dati di base
           ->> NUOVO                   +"
550  PRINT"+        2) Guarda i dati
           ->> GUARDA                  +"
```

```
560  PRINT"+        3) Cerca UNO dei dati
          ->> CERCA                +"
570  PRINT"          4)
          ->> FINE                 +"
580  PRINT"+
                                    +"
590  PRINT"++++++++++++++++++++++++++++
     ++++++++++++++++++++++++++++++++"
600  PRINT CHR$(13)
610  DIM OPTIE$(1):INPUT "PREMI ADESSO UNA DI QUELLE
     OPZIONI !";OPTIE$
630  IF OPTIE$="NUOVO" GOTO 1000
640  IF OPTIE$="GUARDA" GOTO 2000
650  IF OPTIE$="CERCA" GOTO 3000
660  IF OPTIE$="VUL" GOTO 4000
670  IF OPTIE$="FINE" GOTO 50600
680  PRINT
690  PRINT"Per favore rispondi solo con  - NUOVO"
691  PRINT"                              - GUARDA"
692  PRINT"                              - CERCA"
693  PRINT"                              - VUL"
694  PRINT"                              - FINE"
695  GOTO 600
1000 REM  -NUOVO-
1010 I%=1
1020 GOSUB 20000
1030 CURSOR 54,20
1040 PRINT I%
1050 CURSOR 8,21
1060 INPUT NAME$(I%)
1070 IF NAME$(I%)="HALT" GOTO 500
1080 CURSOR 12,20
1090 INPUT SURNAME$(I%)
1100 CURSOR 14,19
1110 INPUT ADRESS$(I%)
1120 I%=I%+1:IF I%<=20 GOTO 1020
1140 CURSOR 10,10:PRINT"Spiacente ma hai esaurito i
     dati di base"
1150 GOSUB 10000
1160 GOTO 500
2000 REM  -GUARDA-
2010 I%=1
2020 IF NAME$(I%)="HALT" GOTO 500
2025 GOSUB 20000
2030 CURSOR 54,20
2040 PRINT I%
2050 CURSOR 8,21
2060 PRINT NAME$(I%)
2070 CURSOR 18,20
2080 PRINT SURNAME$(I%)
2090 CURSOR 14,19
2100 PRINT ADRESS$(I%)
```

```
2110 GOSUB 10000
2130 IF I%<=20 GOTO 2020
2140 PRINT CHR$(12):PRINT"Ora hai visto
     le 50 persone   !"
2150 GOSUB 10000
2160 GOTO 500
3000 REM  -CARICA-
3005 PRINT CHR$(12)
3010 PRINT"VUOI CERCARE UNA PERSONA ."
3020 PRINT"Quali caratteristiche conosci ???"
3030 PRINT"     1) Nome          ->> NOME"
3040 PRINT"     2) Cognome       ->> COGN"
3050 PRINT"     3) Indirizzo     ->> INDI"
3060 PRINT"     4) Numero        ->> NUME"
3070 PRINT"     5) Nessuna....   ->> NESS"
3080 PRINT CHR$(13)
3090 DIM KOMMANDO$(1):INPUT KOMMANDO$
3100 IF KOMMANDO$="NOME" GOTO 3200
3110 IF KOMMANDO$="COGN" GOTO 3300
3130 IF KOMMANDO$="NUME" GOTO 3500
3140 IF KOMMANDO$="INDI" GOTO 3400
3150 IF KOMMANDO$="NESS" GOTO 2010
3160 PRINT:PRINT"Rispondi solo con NOME"
3161 PRINT"                    COGN"
3162 PRINT"                    NUME"
3163 PRINT"                    INDI"
3164 PRINT"                    NESS"
3180 GOTO 3080
3200 REM  -CARICA NOME-
3201 PRINT CHR$(12)
3202 DIM D$(1):INPUT"Conosci il nome <SI/NO>";D$
3203 IF D$="NO" GOTO 3210
3204 IF D$="SI" GOTO 7000
3205 PRINT:PRINT"Rispondi solo con SI o NO .":PRINT:
     GOTO 3202
3210 PRINT:PRINT"Segue la lista dei nomi :"
3220 I%=1
3225 REM
3230 PRINT I%;"  ";NAME$(I%)
3240 I%=I%+1
3250 IF I%<=20 GOTO 3225
3260 INPUT"Quale numero vuoi vedere";I%
3270 GOTO 3540
3300 REM  -CARICA COGNOME-
3301 PRINT CHR$(12)
3302 DIM F$(1):INPUT"Conosci il cognome <SI/NO>";F$
3303 IF F$="NO" GOTO 3320
3304 IF F$="SI" GOTO 7100
3305 PRINT:PRINT"Rispondi solo con SI o NO !!!":GOTO 3302
3320 PRINT:PRINT"Segue la lista dei cognomi : "
3330 I%=1
3340 IF NAME$(I%)<>"HALT" GOTO 3360
```

```
3345 GOTO 3385
3360 PRINT I%;"  ";SURNAME$(I%)
3370 I%=I%+1
3380 IF I%<=20 GOTO 3340
3385 INPUT"Quale numero vuoi vedere  ";I%
3390 GOTO 3540
3400 REM  -CARICA INDIRIZZO-
3401 PRINT CHR$(12)
3402 DIM G$(1):INPUT"Conosci l'indirizzo <SI/NO> ";G$
3403 IF G$="NO" GOTO 3420
3404 IF G$="SI" GOTO 7200
3405 PRINT:PRINT"Rispondi solo con SI o NO":PRINT:
     GOTO 3402
3420 PRINT:PRINT"Segue la lista di
     tutti gli indirizzi :  "
3430 I%=1
3440 IF NAME$(I%)<>"HALT" GOTO 3460
3445 GOTO 3490
3460 PRINT I%;"  ";ADRESS$(I%)
3470 I%=I%+1
3480 IF I%<=20 GOTO 3440
3490 INPUT"Quale numero vuoi vedere ";I%
3495 GOTO 3540
3500 REM  -CERCA IL NUMERO-
3510 PRINT CHR$(12)
3520 INPUT"Quale numero vuoi vedere ";I%
3540 GOSUB 20000
3545 GOSUB 30000
3570 GOSUB 10000
3580 GOTO 500
4000 REM  -PIENO-
5000 REM  -HALT-
7000 REM  -NOME CONOSCIUTO-
7010 I%=1:PRINT
7014 DIM GEKEND$(1):INPUT"Quale nome
     vuoi vedere ";GEKEND$
7020 IF NAME$(I%)=GEKEND$ GOTO 7050
7030 I%=I%+1
7040 IF I%<=20 GOTO 7020
7045 GOTO 500
7050 GOSUB 20000
7060 GOSUB 30000
7070 GOSUB 10000
7080 GOTO 7030
7100 REM  -COGNOME CONOSCIUTO-
7110 I%=1 :PRINT
7114 DIM GEKEND$(1):INPUT"Quale cognome vuoi vedere";
     GEKEND$
7120 IF SURNAME$(I%)=GEKEND$ GOTO 7150
7130 I%=I%+1
7140 IF I%<=20 GOTO 7120
7145 GOTO 500
```

```
7150 GOSUB 20000
7160 GOSUB 30000
7170 GOSUB 10000
7180 GOTO 7130
7200 REM  -INDIRIZZO CONOSCIUTO-
7210 I%=1:PRINT
7214 DIM GEKEND$(1):INPUT"Quale indirizzo vuoi vedere";
     GEKEND$
7220 IF ADRESS$(I%)=GEKEND$ GOTO 7250
7230 I%=I%+1
7240 IF I%<=20 GOTO 7220
7245 GOTO 500
7250 GOSUB 20000
7260 GOSUB 30000
7270 GOSUB 10000
7280 GOTO 7230
9999 REM  -RITORNO SUB-
10000 CURSOR 5,3
10010 PRINT"                    ---------------"
10020 CURSOR 5,2
10030 PRINT" ***    ORA PREMI !   RETURN   !    ***"
10040 CURSOR 5,1
10050 PRINT"                    ---------------"
10060 DIM TERUG$(1):INPUT TERUG$
10070 RETURN
19999 REM  -SEZIONE SUB-
20000 PRINT"*****************************
      *********************************"
20020 PRINT"* NOME :
                                *********"
20030 PRINT"* COGNOME :
                                *Nr.*   *"
20040 PRINT"* INDIRIZZO :
                                *********"
20050 PRINT"*****************************
      *********************************"
20060 RETURN
30000 REM  -STAMPA SUB-
30045 CURSOR 54,20:PRINT I%
30050 CURSOR 8,21:PRINT NAME$(I%)
30055 CURSOR 12,20:PRINT SURNAME$(I%)
30060 CURSOR 14,19:PRINT ADRESS$(I%)
30070 RETURN
50000 MODE 6:COLORG 0 9 8 15
50100 FILL 100,28 236,228 9
50200 FILL 118,158 212,178 8
50210 FOR I=1 TO 26:READ A,B,C,D
50220 DRAW A,B C,D 0:NEXT I
50230 FOR I=1 TO 9:READ A,B,C,D
50240 DRAW A,B C,D 9:NEXT I
50300 DATA 136,164,136,173,137,173,143,
      173,144,173,144,164
```

```
50301 DATA 137,167,143,167,147,164,147,
      173,148,173,154,173
50302 DATA 148,164,153,164,154,164,154,
      167,153,167,154,167
50303 DATA 157,164,157,173,158,173,164,
      173,158,168,164,168
50304 DATA 158,164,164,164,167,164,167,
      173,167,173,174,164
50305 DATA 167,164,167,173,167,173,174,
      164,174,164,174,173
50306 DATA 177,164,177,173,178,164,181,
      164,178,173,181,173
50307 DATA 182,165,184,167,182,172,184,
      170,184,171,184,168
50308 DATA 187,164,187,173,188,173,193,
      173,194,173,194,164
50309 DATA 188,167,193,167
50400 DATA 101,229,110,238,111,238,246,
      238,246,237,246,38
50401 DATA 106,233,241,233,241,33,241,
      232,241,33,237,33
50402 DATA 246,38,241,38,110,238,110,
      233,105,233,105,228
50500 FOR I=10 TO 1 STEP -1:DRAW 236-I,
      218+I 236,218+I 15:NEXT I
50530 FOR I=1 TO 10:DRAW 236-I,38-I 236
      ,38-I 15:NEXT I
50540 A=GETC:IF A=0 GOTO 50540
50550 RETURN
50560 END
```

8.3. POLIGONI

QUESTO PROGRAMMA DISEGNA POLIGONI DI N LATI SULLO SCHERMO. LA GESTIONE DEL COLORE SPETTA ESCLUSIVAMENTE ALL'UTENTE. FINITO UN POLIGONO PREMENDO QUALSIASI TASTO NE VIENE PROPOSTO UN'ALTRO AVENTE GLI STESSI COLORI E LO STESSO NUMERO DI LATI DEL PRECEDENTE. PREMENDO INVECE LA <BAR SPACE> IL PROGRAMMA RICOMINCIA DALL'INIZIO CIOE' CON LA RICHIESTA DEI COLORI DA UTILIZZARE. PREMENDO IL TASTO <RETURN> IL PROGRAMMA FINISCE.

E' POSSIBILE ANCHE INTERROMPERE IL PROGRAMMA DURANTE LA FASE DI PLOTTAGGIO. ANCHE IN QUESTO CASO SI DEVONO SEGUIRE LE MODALITA' VISTE PRIMA.

```
1     CLEAR 5000
2     PRINT CHR$(12):POKE #75,32:MODE 0:CURSOR 10,15
5     INPUT"QUANTI LATI            ";N
6     CURSOR 10,14:INPUT"RAGGIO  <TRA 4 E 120> ";R
7     CURSOR 10,13:INPUT"COLORE SFONDO         ";B
8     CURSOR 10,12:INPUT"COLORE POLIGONO       ";T
10    COLORG B T 0 0:MODE 5
60    DIM B(N),C(N)
90    P1=2*PI/N
100   FOR I=1 TO N
110   B(I)=R+10+R*COS((I-1)*P1)
120   C(I)=R+10+R*SIN((I-1)*P1)
130   NEXT I
140   FOR I=1 TO N
150   FOR J=1 TO N
160   DRAW B(J),C(J) B(I),C(I) T
165   A=GETC:IF A<>0 THEN 190
170   NEXT J:NEXT I
180   A=GETC:IF A=0 THEN 180
190   IF A=32 THEN 1
200   IF A<>13 THEN 10
```

8.4. SIMON

QUESTO PROGRAMMA SIMULA IL GIOCO DEL SIMON. IL GIOCO CONSISTE NEL RIPETERE UNA SEQUENZA DI SUONI PROPOSTA DAL COMPUTER. OGNI VOLTA CHE SI SENTE UN SUONO COMPARE SULLO SCHERMO UN RETTANGOLO COLORATO. QUANDO IL COMPUTER HA FINITO DI PROPORRE LA SUA SEQUENZA RIMANE IN ATTESA DELLA RISPOSTA DEL GIOCATORE CHE LA DEVE RIPETERE IN MODO ESATTO UTILIZZANDO SOLO I TASTI GRIGI. AD OGNI TASTO GRIGIO CORRISPONDE UN SUONO E UN RETTANGOLO SULLO SCHERMO.SE RIPETENDO LA SEQUENZA PROPOSTA SI COMMETTE UN ERRORE LO SCHERMO LAMPEGGIA.

PER RIPRENDERE IL GIOCO BASTA PREMERE QUALSIASI TASTO.

```
1     MODE 0:GOSUB 7000:MODE 4A:BST=0:CNT=0
2     COLORG 0 14 2 15:CURSOR 0,3:
      PRINT "       ULTIMA PARTITA";
3     CURSOR 40,3:PRINT "MIGLIOR RISULTATO"
4     GOSUB 5000
12    CLEAR 1000
15    ENVELOPE 0 3,10;3,10;3,10;0
19    REM COORDINATE DEI RETTANGOLI
20    DIM A(4),B(4)
35    B(2)=70 :A(3)=100 :B(3)=40
37    A(4)=70 :B(4)=10
40    DIM TUNE(100) , NOTE(4)
80    NOTE(4)=262 :NOTE(1)=330 :NOTE(3)=392 :NOTE(2)=523
100   DIM COLOR(4)
110   COLOR(1)=1 :COLOR(2)=5 :COLOR(3)=7 :COLOR(4)=11
450   CNT=0
480   CNT=CNT+1
490   TUNE(CNT)=INT(RND(4))+1
500   WAIT TIME 30
515   REM LINEE 520-560:PROPONE LA SEQUENZA AL GIOCATORE
520   FOR I=1 TO CNT
530   PLAY=TUNE(I)
540   GOSUB 2000
560   NEXT I
580   REM LINEE 590-650:RICEVE LA RISPOSTA DEL GIOCATORE
590   I=0
600   I=I+1
610   IF I>CNT THEN 480
630   GOSUB 6000
640   GOSUB 2000
645   IF BST<CNT THEN BST=CNT
650   IF PLAY=TUNE(I) THEN 600
669   REM LINEE 670-771:SEGNALA L'ERRORE AL GIOCATORE
670   GOSUB 5000
760   CURSOR 22,2 :PRINT "PARTITA INTERROTTA"
762   COLORG 0 0 0 0 :WAIT TIME 10 :COLORG 15 0 0 0:
763   A=GETC :IF A=0 THEN 762
```

```
764  CURSOR 22,2 :PRINT "                ";:CURSOR 44,2
770  IF BST>CNT THEN GOSUB 5010
771  GOTO 15
1999 REM SUONA E DISEGNA I RETTANGOLI
2000 SOUND 0 0 10 0 FREQ(NOTE(PLAY))
2020 SOUND 2 0 10 2 FREQ(NOTE(PLAY)*4)
2030 COLORG 0 COLOR(PLAY) 0 0
2039 REM DISEGNA IL RETTANGOLO
2040 FILL A(PLAY),B(PLAY) A(PLAY)+20,B(PLAY)+20
3000 WAIT TIME 20
3050 SOUND OFF
4039 REM CANCELLA IL RETTANGOLO
4040 FILL A(PLAY),B(PLAY) A(PLAY)+20,B(PLAY)+20 0
4100 RETURN
4999 REM STAMPA IL PUNTEGGIO
5000 CURSOR 10,2 :CNT%=CNT :PRINT CNT%; :PRINT"   ";
5010 CURSOR 44,2 :BST%=BST :PRINT BST%; :PRINT"   ";
5015 CURSOR 44,2
5020 RETURN
5999 REM RICEVE LA RISPOSTA DEL GIOCATORE
6000 G=GETC :IF G=0 GOTO 6000
6050 IF G=18 THEN PLAY=1 :GOTO 6120
6060 IF G=16 THEN PLAY=2 :GOTO 6120
6070 IF G=19 THEN PLAY=3 :GOTO 6120
6080 IF G=17 THEN PLAY=4
6120 RETURN
6999 REM STAMPA L'INTESTAZIONE
7000 PRINT CHR$(12)
7001 CURSOR 0,16
7010 PRINT "*********************************************
     ***************"
7020 PRINT "* *******************************************
     ************* *"
7030 PRINT "* *                    ----- SIMON -----
                * *"
7040 PRINT "* *******************************************
     ************* *"
7050 PRINT "*********************************************
     ***************"
7060 PRINT :PRINT "            PER IL GIOCO UTILIZZARE
     SOLO I"
7070 PRINT "                      TASTI GRIGI"
7080 CURSOR 0,0
7090 UI=GETC :IF UI=0 THEN 7090
7100 RETURN
```

8.5. FIGURE DI LISSAJOU

```
5     CLEAR 5000
10    MODE 6
16    DIM A(250),B(250)
20    COLORG 8 0 15 3
30    FOR X=0 TO 2*PI STEP 3E-2
40    A(N)=XMAX/2+100*COS(X):B(N)=YMAX/2
      +100*SIN(X*2)
45    N=N+1
50    NEXT
90    COLORG 8 0 15 3
100   FOR X=0 TO 209
110   DRAW 150,125 A(X),B(X) 0
115   DRAW 0,0 A(X),B(X) 3
116   DRAW A(X),B(X) XMAX,0 15
120   NEXT
300   FOR X=0 TO 50
320   COLORG 0 A 0 0
330   WAIT TIME 15
335   COLORG 0 0 A 0
337   WAIT TIME 15
338   COLORG 0 0 0 A
339   WAIT TIME 15
340   A=A+1:IF A=16 THEN A=1
345   NEXT X
400   FOR X=0 TO 50
410   COLORG RND(15) RND(15) RND(15)
      RND(15)
420   WAIT TIME 20
430   NEXT X
450   GOTO 90
```

8.6. PIRAMIDE ROTANTE

QUESTO PROGRAMMA RAPPRESENTA UNA PIRAMIDE CHE RUOTA SULLO SCHERMO SECONDO GLI ASSI X,Y,Z.

LA ROTAZIONE CHE AVVIENE SOTTO IL DIRETTO CONTROLLO DELL'UTENTE SI OTTIENE PREMENDO I SEGUENTI TASTI:
<1>,<2>,<3>,<4>,<5> E <6>

ESSI POSSONO ANCHE ESSERE PREMUTI INSIEME AL TASTO <REPT>. IL FATTORE SCALARE CHE VIENE CHIESTO VIA INPUT E' LA GRANDEZZA CHE FA VARIARE LE DIMENSIONI DELLA PIRAMIDE SULLO SCHERMO.

```
1   MODE  0:PRINT CHR$(12):CURSOR 5,15:
    POKE #75,32:REM IL CURSORE DIVENTA
    UN BLANK
6   INPUT"SFONDO   ";B:PRINT:CURSOR 5,14
8   INPUT"PIRAMIDE ";T:CURSOR 5,13
9   INPUT"FATTORE SCALARE";SF
85  MODE 6
86  COLORG B T B T
87  GOSUB 2000:REM INIZIALIZZAZIONE DATI
92  GOSUB 801:REM DISEGNO NUOVA FIGURA
95  COLORG B T*(1-Q)+B*Q T*Q+B*(1-Q) T
96  GOSUB 901:REM CANCELLAMENTO VECCHIA
    FIGURA
97  Q=1-Q
99  KS=ABS(KS)
100 A=GETC:IF A<ASC("0") GOTO 100
120 FOR P=1 TO NP
130 XX(P)=X(P):YY(P)=Y(P)
140 NEXT
150 ON A-ASC("0") GOTO 500,510,600,610,
    700,710
160 GOTO 100
500 KS=-KS
510 FOR P=1 TO NP
520 X=X(P):Y=Y(P)
530 X(P)=X*KC+Y*KS
540 Y(P)=Y*KC-X*KS
550 NEXT
560 GOTO 92
600 KS=-KS
610 FOR P=1 TO NP
620 Y=Y(P):Z=Z(P)
630 Y(P)=Y*KC+Z*KS
640 Z(P)=Z*KC-Y*KS
650 NEXT
660 GOTO 92
700 KS=-KS
710 FOR P=1 TO NP
720 Z=Z(P):X=X(P)
```

```
730 Z(P)=Z*KC+X*KS
740 X(P)=X*KC-Z*KS
750 NEXT
760 GOTO 92
801 REM DISEGNO NUOVA FIGURA
810 FOR L=1 TO NL
820 PA=LA(L)
830 PB=LB(L)
840 DRAW X(PA)+XC,Y(PA)+YC X(PB)+XC,
    Y(PB)+YC 17+Q*2
850 NEXT
860 RETURN
901 REM CANCELLAMENTO VECCHIA FIGURA
910 FOR L=1 TO NL
920 PA=LA(L)
930 PB=LB(L)
940 DRAW XX(PA)+XC,YY(PA)+YC XX(PB)+XC,
    YY(PB)+YC 18-2*Q
950 NEXT
960 RETURN
991 REM INIZIALIZZAZIONE DEI DATI
2000 PHI=PI/20
2010 KS=SIN(PHI)
2020 KC=COS(PHI)
2030 XC=XMAX/2
2040 YC=YMAX/2
2050 Q=1
2100 READ NP,NL
2110 DIM X(NP),Y(NP),Z(NP)
2120 DIM XX(NP),YY(NP)
2130 DIM LA(NL),LB(NL)
2200 FOR P=1 TO NP
2210 READ X(P),Y(P),Z(P)
2211 X(P)=X(P)*SF
2212 Y(P)=Y(P)*SF
2213 Z(P)=Z(P)*SF
2220 NEXT
2230 FOR L=1 TO NL
2240 READ LA(L),LB(L)
2250 NEXT
2260 GOSUB 801
2270 RETURN
2301 REM DATI
2800 REM NUMERO DI PUNTI E NUMERO DI
    LINEE
2900 DATA 5,8
2903 DATA 0,0,20
2904 DATA 20,20,-20
2905 DATA 20,-20,-20
2906 DATA -20,20,-20
2907 DATA -20,-20,-20
2910 DATA 1,2
```

```
2911 DATA 1,3
2912 DATA 1,4
2913 DATA 1,5
2914 DATA 2,3
2915 DATA 2,4
2916 DATA 3,5
2917 DATA 4,5
2999 DATA 8,12
4000 DATA 1,2
4001 REM DATI PER QUALCOSA D'ALTRO
4009 DATA 20,20,20
4010 DATA 20,20,-20
4020 DATA 20,-20,20
4030 DATA 20,-20,-20
4040 DATA -20,20,20
4050 DATA -20,20,-20
4060 DATA -20,-20,20
4070 DATA -20,-20,-20
4110 DATA 1,3
4120 DATA 1,5
4130 DATA 2,4
4140 DATA 2,6
4150 DATA 3,4
4160 DATA 3,7
4170 DATA 4,8
4180 DATA 5,6
4190 DATA 5,7
4210 DATA 7,8
9999 END
```

8.7. ISTRUTTORE MUSICALE

QUESTO PROGRAMMA GENERA MUSICA E VISUALIZZA LE NOTE. SE SI RISPONDE SI PREMENDO < S > ALLA PRIMA DOMANDA,I SOLI TASTI CHE SI POSSONO UTILIZZARE SONO QUELLI CHE VANNO DALLA (A) ALLA (F),DAL DO AL SI. MENTRE SE SI RISPONDE NO PREMENDO < N > TUTTI I TASTI ALFABETICI GENERANO NOTE. SI POSSONO ANCHE VISUALIZZARE LE NOTE IN GRANDI O PICCOLE SCALE. QUESTO AVVIENE RISPONDENDO < G > O < P > ALLA DOMANDA.

I TASTI NUMERICI HANNO LE SEGUENTI FUNZIONI:

```
1 = NOTE NORMALI
2 = TREMOLO
3 = GLISSANDO
4 = GLISSANDO+TREMOLO
5 = DISATTIVA LE NOTE CORTE
6 = NOTE CORTE
7 = INIZIA LA REGISTRAZIONE LUNGA MASSIMO 2000 NOTE
8 = FINISCE LA REGISTRAZIONE RIPROPONENDOLA SE VIENE PREMUTO
9 = FA SCORRERE LA PAGINA
0 = CANCELLA LA PAGINA
SHIFT+TASTO ALFABETICO = INVERTE LE NOTE
IL TASTO <TAB> FA RIPARTIRE IL PROGRAMMA
```

```
1     CLEAR 10000:LIMIT%=10:DIM ARRAY%(LIMIT%,200)
2     PAGE%=0:POINTER%=0:RECORD%=0:PLAYBACK%=0:TUTOR%=0
      ACCENT%=0
3     PRINT CHR$(12):PRINT:PRINT:PRINT"MODO ISTRUZIONE-SI O
      NO <S/N>"
4     ANS%=GETC:IF ANS%=0 THEN 4
5     IF ANS%=ASC("S") THEN TUTOR%=1:GOTO 7
6     IF ANS%<>ASC("N") THEN GOTO 1
7     PRINT:PRINT"SCALA-GRANDE O PICCOLA. <G/P>"
8     ANS%=GETC:IF ANS%=0 THEN 8
9     IF ANS%=ASC("G") THEN MODE 3:GOTO 15
10    IF ANS%=ASC("P") THEN MODE 5:GOTO 15
11    PRINT"RISPONDI SOLO CON 'G' O 'P'":GOTO 7
15    ENVELOPE 0 15,100;8,75;3,50;0:ENVELOPE 1 15,3;10,2;0:
      STYLE%=0
17    RESTORE:DIM NOTE(21,2),COMP%(21,1),SPOT%(21)
18    FOR I%=1 TO 13:FOR J%=0 TO 1:READ COMP%(I%,J%):NEXT J%
19    NOTE(I%,0)=FREQ(267*(2(I%/12)))
21    NOTE(I%,1)=2*NOTE(I%,0):NOTE(I%,2)=NOTE(I%,0)/2:NEXT
22    FOR I%=14 TO 21:FOR J%=0 TO 1:READ COMP%(I%,J%):NEXT:
      FOR J%=0 TO 2
23    READ CHORD%:NOTE(I%,J%)=NOTE(CHORD%,0):NEXT J%:NEXT I%
24    FOR I%=1 TO 21:READ SPOT%(I%):NEXT I%
25    GOSUB 15000
28    FOR TIMER%=1 TO 100-99*ACCENT%
30    GOSUB 10000:IF KEY%=0 THEN NEXT TIMER%:SOUND OFF:
```

```
     GOTO 28
31   IF KEY%=53 THEN ACCENT%=0:GOTO 30
32   IF KEY%=54 THEN ACCENT%=1:GOTO 30
33   IF KEY%=48 GOSUB 2000:GOTO 30
34   IF (KEY%=57) OR (WHERE=(-1)) THEN OFFSET=OFFSET-75:
     GOSUB 2010:GOTO 30
35   IF KEY%=9 THEN SOUND OFF:MODE 0:GOT 3
36   IF (KEY%>48) AND (KEY%<53) THEN STYLE%=KEY%-49:GOTO 30
37   OCTAVE%=1:IF (KEY%>96) OR (KEY%=60) THEN OCTAVE%=2:
     GOSUB 3000
38   FOR J%=1+13*TUTOR%*(1-ACCENT%) TO 21
39   IF KEY%<>COMP%(J%,TUTOR%) THEN NEXT J%:GOTO 3500
40   FOR I%=0 TO 2
41   SOUND I% ACCENT% 15-10*SGN(I%) STYLE% NOTE(J%/I%)/
     OCTAVE%:NEXT I%
42   IF (SPOT%(J%)=100) OR (WHERE=(-1)) OR (OFFSET<0)
     GOTO 100
48   GOSUB 4000
50   FILL AA,BB CC,DD EE
55   DRAW FF,GG HH,II JJ
60   WHERE=WHERE+10:IF WHERE>XMAX-10 THEN WHERE=-1
100  GOTO 28
200  NEXT TIMER%
210  SOUND OFF:GOTO 28
1000 DATA 90,67,83,67,88,68,68,67,67,69,86,70,71,67,66,71,
     72,67,78,65
1010 DATA 74,67,77,66,44,99,87,67,1,5,8,69,68,3,8,1,82,69,
     5,1,8,84,70
1015 DATA 6,10,13,89,71,8,1,5,85,65,10,1,6,73,66,12,3,8,79,
     99,13,5,8
1020 DATA -10,100,-5,100,0,5,100,10,100,15,100,20,25,-10,
     -10,-5,0,5,10,15,20,25
1500 OFFSET=YMAX-62:GOTO 2020
2000 FILL 0,0 XMAX,YMAX 0:GOTO 1500
2010 IF OFFSET<0 GOTO 1500
2020 WHERE=5
2030 FILL 0,OFFSET-12 XMAX,OFFSET+62 0
2040 FOR Z%=OFFSET TO OFFSET+40 STEP 10
2050 DRAW 0,Z% XMAX,Z% 12:NEXT Z%:RETURN
3000 KEY%=KEY%-32:IF KEY%=38 THEN KEY%=44
3010 RETURN
3500 TIMER%=TIMER%+1:GOTO 200
3510 RETURN
4000 AA=WHERE-2:BB=OFFSET+(OCTAVE%-1)*35+SPOT%(J%)-2
4010 CC=WHERE+2:DD=OFFSET+(OCTAVE%-1)*35+SPOT%(J%)+2
4020 EE=SPOT%(J%)/5+8
4030 FF=WHERE+6-4*OCTAVE%:GG=OFFSET+SPOT%(J%)+(OCTAVE%-1)*
     35
4040 HH=WHERE+6-4*OCTAVE%:II=OFFSET+SPOT%(J%)+20:
     JJ=SPOT%(J%)/5+8
4050 RETURN
5000 IF KEY%=56 THEN RECORD%=0:ARRAY%(PAGE%,POINTER%)=128
```

```
5010 RETURN
6000 IF POINTER%=200 THEN POINTER%=0:PAGE%=PAGE%+1:
     GOSUB 7000
6010 RETURN
7000 IF PAGE%>LIMIT% THEN PAGE%=LIMIT%:RECORD%=0:
     PLAYBACK%=0
7010 RETURN
10000 KEY%=GETC:IF KEY%=55 GOTO 30000
10002 IF (KEY%=56) AND (RECORD%=0) THEN PLAYBACK%=1:
      POINTER%=0:PAGE%=0
10005 IF RECORD%=1 THEN ARRAY%(PAGE%,POINTER%)=KEY%:
      GOSUB 5000
10010 IF PLAYBACK%=1 THEN KEY%=ARRAY%(PAGE%,POINTER%)
10015 IF (RECORD%=1) OR (PLAYBACK%=1) THEN POINTER%=
      POINTER%+1:GOSUB 6000
10020 IF KEY%=128 THEN PLAYBACK%=0
10030 RETURN
30000 RECORD%=1:PLAYBACK%=0:POINTER%=0:PAGE%=0
30010 KEY%=GETC:IF KEY%=0 GOTO 30010
30020 GOTO 10002
```

8.8. TORRE DI HANOI

IL GIOCO CONSISTE NEL MUOVERE TUTTE LE BARRE ORIZZONTALI DALLA COLONNA 1 ALLA COLONNA 3 SENZA METTERE MAI UNA BARRA GROSSA SOPRA UNA PIU' PICCOLA.
PER MUOVERE LA BARRA PREMI I TASTI <1>,<2> O <3>,DANDO IL NUMERO DELLA COLONNA DOVE PRECEDENTEMENTE ERA LA BARRA,SEGUITO DAL NUMERO DELLA COLONNA DOVE LA BARRA DEVE ANDARE.

```
1     MODE 0:PRINT CHR$(12):CURSOR 4,21:POKE #75,32
2     PRINT".................TORRE DI HANOI.............
      ..."
4     CURSOR 5,19:PRINT"UN'ESEMPIO DELLE CAPACITA' GRAFICHE
      ANIMATE DEL"
5     CURSOR 6,17:PRINT"           D A I PERSONAL COMPUTER"
18    CURSOR 4,10:PRINT"PREMI QUALSIASI TASTO PER COMINCIARE
      IL GIOCO"
19    T=GETC:IF T=0 GOTO 19
20    CLEAR 2000
21    DIM Z(100)
22    PRINT CHR$(12)
23    COLORT 7 0 0 0
24    COLORG 7 4 5 1
25    MODE 2A
30    JC1%=0:Y9=48:N=9:C1=4:C2=5:C3=1:C0=7
33    DRAW 0,0 70,0 C1
36    FOR I=1 TO 3
38    DRAW I*24-12,0 I*24-12,Y9 C2
40    Z(1)=0:Z(I*10)=10:NEXT
50    M=1:C=C3
60    FOR I=1 TO N
70    Z(1)=I:Z(10+I)=10-I
80    GOSUB 900:NEXT
90    GOTO 110
100   PRINT"MOSSA NON VALIDA"
110   JC1%=JC1%+1:PRINT"LA TUA MOSSA DA <1,2 O 3>  ";
111   P=GETC:WAIT TIME 5:IF P=0 GOTO 111
112   M1=P-48:M1%=M1:PRINT M1%;:PRINT"  A  ";
113   P=GETC:WAIT TIME 5:IF P=0 GOTO 113
114   M2=P-48:M2%=M2:PRINT M2%;:PRINT"   ";:PRINT JC1%;:
      PRINT" MOSSE"
120   IF M1<>INT(M1) OR M1<1 OR M1>3 GOTO 100
130   IF M2<>INT(M2) OR M2<1 OR M2>3 GOTO 100
140   IF M1=M2 OR Z(M1)=0 GOTO 100
150   P1=Z(M1)+10*M1
160   P2=Z(M2)+10*M2
170   IF Z(P1)>Z(P2) GOTO 100
200   M=M1:C=C0:GOSUB 900
210   Z(M2)=Z(M2)+1:Z(P2+1)=Z(P1)
220   Z(M1)=Z(M1)-1
```

```
230  M=M2:C=C3:GOSUB 900
240  G=G+1
250  IF Z(3)<N GOTO 110
300  PRINT"HAI USATO ",JC1%,"MOSSE"
310  STOP
900  X=M*24-12
910  X=5*Z(M)
920  X1=Z(Z(M)+10*M)+2
930  DRAW X-X1,Y X-1,Y C
940  DRAW X+1,Y X+X1,Y C
950  RETURN
```

8.9. DISEGNI COL CURSORE

IL PROGRAMMA CONSENTE DI DISEGNARE SULLO SCHERMO QUALSIASI FIGURA. PER MUOVERSI IN ORIZZONTALE E IN VERTICALE SI DEVONO UTILIZZARE I TASTI GRIGI. PER MUOVERSI IN DIAGONALE SI DEVONO UTILIZZARE I TASTI <1> <2> <3> <4>. DOPO AVER FATTO UN DISEGNO SE SI VUOLE COMINCIARNE UN ALTRO PREMERE LA BAR SPACE. SE SI VOGLIONO CANCELLARE DELLE LINEE PREMERE <CHAR DEL> E RIPASSARE SULLE LINEE STESSE.
PER RITORNARE IN MODO DISEGNO PREMERE <TAB>.

```
1     MODE0
3     PRINT CHR$(12):GOSUB 400
4     COLORG B T 0 0
5     MODE 3
10    A=GETC
12    IF A=32 THEN 200
13    IF A=8 THEN 220
14    IF A=9 THEN 320
15    GOTO 320
90    REM CAMBIO DELLE COORDINATE X E Y
100   Y=Y+1:IF Y>YMAX THEN Y=YMAX
105   RETURN
110   Y=Y-1:IF Y<0 THEN Y=0
115   RETURN
120   X=X-1:IF X<0 THEN X=0
125   RETURN
130   X=X+1:IF X>XMAX THEN X=XMAX
135   RETURN
200   FILL 0,0 XMAX,YMAX :Y=0:X=0
210   GOTO 10
215   REM CANCELLA LA LINEA
220   A=GETC:IF A=48 GOTO 1
221   IF A=32 GOTO 200
222   IF A=9 GOTO 320
223   IF A<16 OR A>52 GOTO 220
230   IF A=49 THEN GOSUB 100:GOSUB 130:GOTO 290
240   IF A=50 THEN GOSUB 100:GOSUB 120:GOTO 290
250   IF A=51 THEN GOSUB 110:GOSUB 120:GOTO 290
260   IF A=52 THEN GOSUB 110:GOSUB 130:GOTO 290
280   A=A-15:ON A GOSUB 100,110,120,130
290   DOT X,Y B:GOTO 220
295   REM DISEGNO DELLA LINEA
320   A=GETC:IF A=48 GOTO 1
321   IF A=8 GOTO 220
322   IF A<16 OR A>52 GOTO 320
323   IF A=32 GOTO 200
324   IF A=49 THEN GOSUB 100:GOSUB 130:GOTO 330
325   IF A=50 THEN GOSUB 100:GOSUB 120:GOTO 330
326   IF A=51 THEN GOSUB 110:GOSUB 120:GOTO 330
327   IF A=52 THEN GOSUB 110:GOSUB 130:GOTO 330
```

```
329  DOT X,Y T:A=A-15:ON A GOSUB 100,110,120,130
330  DOT X,Y B:DOT X,Y T:GOTO 320
400  CURSOR 20,12:INPUT"COLORE DELLO SFONDO ";B:
     PRINT CHR$(12)
490  CURSOR 21,12:INPUT"COLORE DELLA LINEA";T:
     PRINT CHR$(12)
491  A=GETC:IF A=0 GOTO 491
500  RETURN
```

8.10. LINEE CASUALI

QUESTO PROGRAMMA RAPPRESENTA SULLO SCHERMO UN NUMERO N DI LINEE CASUALI. LA GESTIONE DEL COLORE SPETTA ESCLUSIVAMENTE ALL'UTENTE. FINITA UNA SERIE DI LINEE PREMENDO QUALSIASI TASTO NE VIENE PROPOSTA UN'ALTRA AVENTE GLI STESSI COLORI DELLA PRECEDENTE. PREMENDO,INVECE,LA <BAR SPACE> IL PROGRAMMA RICOMINCIA DALL'INIZIO CIOE' CON LA RICHIESTA DEI COLORI DA UTILIZZARE. PREMENDO IL TASTO <RETURN> IL PROGRAMMA FINISCE.

E' POSSIBILE ANCHE INTERROMPERE IL PROGRAMMA DURANTE LA FASE DI PLOTTAGGIO. ANCHE IN QUESTO CASO SI DEVONO SEGUIRE LE MODALITA' VISTE PRIMA.

```
1     POKE #75,32
2     MODE 0:PRINT CHR$ (12):CURSOR 15,12:
      INPUT"COLORE SFONDO";B
3     PRINT CHR$(12):CURSOR 15,12:INPUT"COLORE LINEE";T
4     PRINT CHR$(12):CURSOR 15,12:INPUT"NUMERO LINEE";N
5     COLORG B T 15 0
10    MODE 6
100   S%=X% MOD (XMAX):T%=Y% MOD (YMAX)
105   FOR A%=0 TO N:X%=RND(XMAX):Y%=RND(YMAX)
110   DRAW S%,T% X%,Y% 15:DRAW S%,T% X%,Y% T:S%=X%:T%=Y%
115   P=GETC:IF P<>0 THEN 140
120   NEXT
130   P=GETC:IF P=0 THEN 130
140   IF P=32 GOTO 2
145   IF P=13 THEN END
150   GOTO 10
```

8.11. DAI TESTO IN GRAFICA

QUESTO PROGRAMMA DA UNA IDEA DELLE CAPACITA' GRAFICHE DEL DAI.E' UN PROGRAMMA RIPETITIVO CHE RAPPRESENTA TUTTO IL SET DI CARATTERI ALFANUMERICI PUR TROVANDOSI IN MODO GRAFICO.

ESSO VIENE REALIZZATO MEDIANTE UN SOTTOPROGRAMMA GRAFICO CHE SI AVVALE DI UN BLOCCO DI "DATA" NEL QUALE CIASCUN CARATTERE E' RAPPRESENTATO IN UNA MATRICE DI 5*7.

DATI PER SUBROUTINE:

X E Y - COORDINATE DEI PUNTI UTILIZZATE NELLA "DRAW"
VFLAG - INDICA SE IL CARATTERE DEVE ESSERE SCRITTO SECONDO L'ASSE VERTICALE O SECONDO L'ASSE ORIZZONTALE
A$ - STRINGA CONTENENTE I CARATTERI DA DISEGNARE
F - INDICA LA GRANDEZZA DEI CARATTERI DA DISEGNARE

```
1   CLEAR 10000
2   REM:DATI PER SUBROUTINE 40040:
    X / Y / C /VFLAG / A$ / F
5   COLORG 9 1 3 5
10  MODE 5
20  COLORG 9 0 14 1:FILL 0,YMAX/2-2 XMAX,YMAX 8
30  GOSUB 40012
31  FOR X=200 TO 230 STEP 3:DRAW X,10 X,45 0:NEXT
32  FOR Y=125 TO 150 STEP 2:FILL 260,Y XMAX,Y+1 Q:
    Q=Q+1:NEXT
33  X=10:Y=215:C=1:A$="DAI":VFLAG=0:F=2:GOSUB 40040
34  X=80:Y=215:C=6:A$="TESTO":GOSUB 40040
35  X=165:Y=215:C=5:A$="IN":GOSUB 40040
36  X=215:Y=215:C=0:F=2:A$="GRAFICA":GOSUB 4004
39  X=180:Y=190:C=2:F=1:A$="TEL. 02 / 2041051":
    GOSUB 40040
40  X=10:Y=200:C=0
41  A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ!#?$%&'()*=:
    -+;<>./1234567890"
50  GOSUB 40040
55  X=10:Y=170:C=3:F=2:GOSUB 40040
56  X=XMAX-10:Y=50:C=13:VFLAG=1:F=1:GOSUB 40040
60  VFLAG=0:X=10:Y=90:C=12:F=4:A$=LEFT$(A$,26):
    GOSUB 40040
63  FOR X=0 TO XMAX:DOT X,225+20*SIN(X/20) 15:NEXT
10000 WAIT TIME 600:GOTO 1
40012 DIM CAR$(90)
40021 FOR Z=32 TO 90:READ A$
40022 IF A$="STOP" THEN RETURN
40023 READ CAR$(Z):NEXT:RETURN
40040 X1=X:Y1=Y:IF F=0 THEN F=1
40041 FOR M=0 TO LEN(A$)-1
40042 T$=MID(A$,M,1)
40050 GR$=CAR$(ASC(T$))
40060 FOR N=0 TO LEN(GR$)-1 STEP 4
```

```
40065 IF VFLAG=1 GOTO 40120
40070 IF MID$(GR$,N,1)="/" THEN X=X+(8*F):GOTO 40100
40080 ZZ=VAL(MID$(GR$,N,1)):YY=VAL(MID$(GR$,N+1,1))
40082 JC5%=X+ZZ*F:JC6%=Y+YY*F
40083 JC7%=X+VAL(MID$(GR$,N+2,1))*F:
      JC8%=Y+VAL(MID$(GR$,N+3,1))*F
40084 DRAW JC5%,JC6% JC7%,JC8% C
40085 IF F<1.5 THEN GOTO 490
40086 JC9%=X+1+VAL(MID$(GR$,N+2,1))*F
40087 JC10%=Y+1+VAL(MID$(GR$,N+3,1))*F
40088 DRAW X+1+ZZ*F,Y+1+YY*F JC9%,JC10% C
40090 NEXT N
40100 IF X+8*F>=XMAX THEN X=X1:Y=Y-10*F
40102 NEXT M
40103 RETURN
40120 IF MID$(GR$,N,1)="/" THEN Y=Y-9*F:GOTO 40180
40130 JC1%=X+VAL(MID$(GR$,N+1,1))*F:
      JC2%=Y-VAL(MID$(GR$,N,1))*F
40131 JC3%=X+VAL(MID$(GR$,N+3,1))*F:
      JC4%=Y-VAL(MID$(GR$,N+2,1))*F
40133 DRAW JC1%,JC2% JC3%,JC4% C
40140 NEXT N
40180 IF Y-9*F<0 THEN Y=Y1:X=X-9*F
40190 NEXT M
40200 RETURN
50000 DATA BLANCO,/,UITROEP!,31313337/,
      QOUTES,25274547/,#
50001 DATA 1353155521274147/,$,12424253
      2444152626563137/
50010 DATA %,1727162612564151425 2/,&,12
      132131533115511627 3536/,'
50011 DATA 3536/,(,131513311537/
50020 DATA ),315353555537/,*,1256165231
      37/,+,32361454/,COMMA
50021 DATA 21323233/
50030 DATA -,1454/,.,31423241/,/,1256/,
      0,121621415256274712 56/
50040 DATA 1,214131372637/,2,1151123344
      44555647271627/,3
50041 DATA 122121415253345617574453/,4,
      414713531447
50050 5,122121415254154515171757/,6,214
      112151444525315373757/
50051 DATA 7,212223561757/,8,2141244427
      471213151652535556/
50060 DATA 9,113131535356245415162747/,
      :,33333535/,;,313232333533/
50061 DATA <,14471441/
50070 DATA =,13531555/,>,21545427/,?,16
      272747343331313456/,APE,/
50080 DATA A,111551551353153737 55/,B,1
      117174714441141525355 56/,C
```

```
50081 DATA 121627474756214141 52/,D,1117
      114152561747/
50090 DATA E,1117115114441757/,F,111714
      441757/,G,1216275721515153 5343/
50091 DATA H,111714545157/
50100 DATA I,214131372747/,J,1221214152
      57/,K,111713572451/,L,11171151/
50110 DATA M,11171735353435575751/,N,11
      1751571652/,O,1216274756522141/,P
50111 DATA 1117144417475556/
50120 DATA Q,1216274756532131 3351/,R,11
      171747565514442451/,S
50121 DATA 122121415253244415162747475 6
      /,T,17573137/
50130 DATA U,111721415157/,V,1317535713
      313153/,W,11175157113333513334/
50131 DATA X,111217165152575612561652/
50140 DATA Y,161756571634345631 34/,Z,17
      5712561151/
51000 DATA STOP
```

Altri libri editi dal Gruppo Editoriale Jackson

Esperimenti su circuiti logici e di memoria TTL
Codice 001A L. 22.000

Esperimenti su circuiti logici e di memoria TTL
Codice 002A L. 22.000

IL 3UGBOOK II - Esperimenti di interfacciamento e trasmissione dati
Codice 021A L. 4.500

IL BUGBOOK III - Interfacciamento e programmazione del microcomputer 8080
Codice 003A L. 19.000

Interfacciamento di microcomputer Esperimenti utilizzanti Chip 8255 PP I
Codice 004A L. 10.500

Esperimenti con TTL e 8080A, Vol. 1
Codice 005A L. 19.000

Esperimenti con TTL e 8080A, Vol. 2
Codice 006A L. 19.000

IL BUGBOOK VII - L'interfacciamento tra microcomputer e convertitori analogici
Codice 007A L. 15.000

Corso di elettronica fondamentale con esperimenti
Codice 201A L. 15.000

Comprendere l'elettronica a stato solido
Codice 202A L. 14.000

Introduzione pratica all'impiego dei circuiti integrati digitali
Codice 203A L. 7.000

Elettronica integrata digitale
Codice 204A L. 34.500

SC/MP
Codice 301D L. 9.500

Lessico dei microprocessori
Codice 302P L. 3.500

Introduzione al personal e business computing
Codice 303D L. 14.000

Introduzione ai microcomputer - Il libro dei principianti, Vol. 0
Codice 304A L. 14.000

Introduzione ai microcomputer - Il libro dei concetti fondamentali, Vol. 1
Codice 305A L. 16.000

Practical microprocessors
Codice 308B L. 35.000

Principi e tecniche di elaborazione dati
Codice 309A L. 15.000

NANOBOOK Z80, Vol. 1 - Tecniche di programmazione
Codice 310P L. 15.000

NANOBOOK Z80, Vol. 3 - Tecniche di interfacciamento
Codice 312P L. 18.000

DBUG
Codice 313P L. 6.000

Tecniche di interfacciamento dei microprocessori
Codice 314P L. 22.000

Microelettronica: la nuova rivoluzione industriale
Codice 315P L. 9.000

Elementi di trasmissione dati
Codice 316D L. 9.000

Impariamo a programmare in Basic con lo ZX80
Codice 317B L. 4.500

I microprocessori
Codice 320P L. 22.000

La programmazione dello Z8000
Codice 321D L. 22.000

TEA, un editor assembler residente per l'8080/8085
Codice 322P L. 12.000

8080A/8085 - Programmazione in linguaggio assembly
Codice 323P L. 24.000

Programmazione dello Z80 e progettazione logica
Codice 324P L. 19.000

Programmazione dell'8080 e progettazione logica
Codice 325P L. 16.500

Z80 programmazione in linguaggio assembly
Codice 326P L. 29.500

Usare il microprocessore
Codice 327A L. 15.000

Pascal - Manuale e standard del linguaggio
Codice 500P L. 10.000

Impariamo il Pascal
Codice 501A L. 10.000

Introduzione al Basic
Codice 502A L. 18.500

Applicazioni del 6502
Codice 504B L. 13.500

Impariamo a programmare in Basic con il PET/CBM
Codice 506A L. 10.000

Impariamo a programmare in Basic con il VIC/CBM
Codice 507A L. 11.000

Il Timer 555
Codice 601B L. 8.600

La progettazione dei circuiti amplificatori operazionali con esperimenti
Codice 602B L. 15.000

La progettazione dei filtri attivi con esperimenti
Codice 603B L. 15.000

La progettazione dei circuiti PLL con esperimenti
Codice 604H L. 14.000

Guida ai CMOS con esperimenti
Codice 605B L. 15.000

I tiristori - 110 progetti pratici
Codice 606D L. 8.000

Guida mondiale dei transistors
Codice 607H L. 20.000

Guida mondiale degli amplificatori operazionali
Codice 608H L. 15.000

Guida mondiale dei transistori ad effetto di campo JFET e MOS
Codice 609H L. 10.000

Amplificatori di Norton
Codice 610B L. 22.000

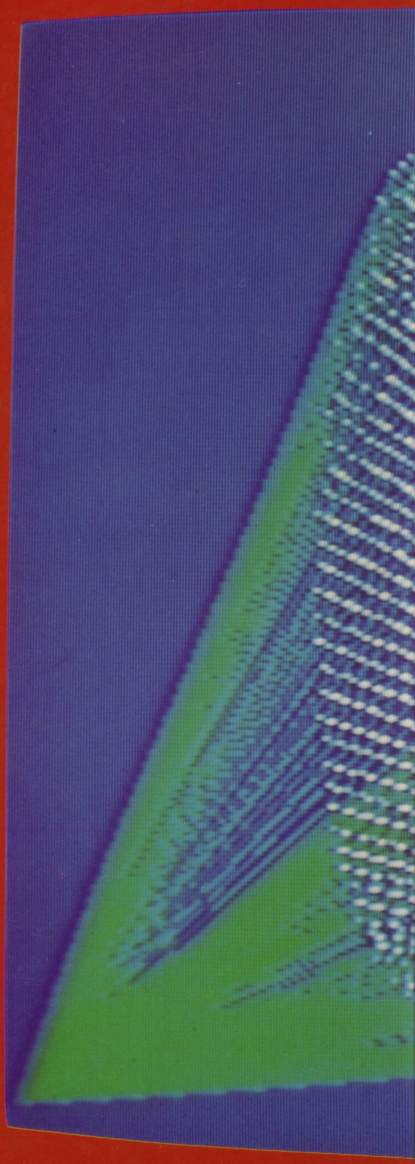
Manuale pratico del riparatore radio-TV
Codice 701P L. 18.500

Audio Handbook
Codice 702H L. 9.500

Audio e HI-Fi
Codice 703D L. 6.000

Potete acquistare i suddetti libri nelle migliori librerie oppure scrivendo direttamente a:
Gruppo Editoriale Jackson - Divisione Libri - Via Rosellini, 12 - 20124 Milano

L. 9.000



DAI

MANUALE DEL MICROCOMPUTER