L'HOME COMPUTER TI 99/4A

Basic
Basic esteso
TMS 9900

Carmine Elefante

GRUPPO EDITORIALE JACKSON

L'HOME COMPUTER TI 99/4A

Basic
Basic esteso
TMS 9900

Carmine Elefante

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

L'autore ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca Di Fiore e l'ing. Roberto Pancaldi.

Idea grafica di copertina a cura di Carmine Elefante.

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

Ai miei genitori

SOMMARIO

CAPITOLO I

L'HOME COMPUTER TI 99/4A

1.1 PRIMA DI COMINCIARE

Il libro si pone un duplice obiettivo:

- insegnare il linguaggio BASIC anche a coloro che si avvicinano per la prima volta alla programmazione e al computer;
- guidare l'utente, anche esperto, alla conoscenza del TI 99/4A come sistema per la sua migliore utilizzazione.

Il primo obiettivo viene perseguito presentando i comandi del sistema operativo e le istruzioni del linguaggio BASIC in modo semplificato, in cui si cerca di avvicinare il lettore alla terminologia specialistica e al suo significato logico.
La metodologia adottata si avvale, inoltre, di una serie di esercizi svolti caratterizzati da un livello di difficoltà crescente.

Al secondo obiettivo del libro si tende attraverso due vie:

- la prima è quella di far conoscere l'architettura del TI 99/4A, di cui vengono forniti gli schemi a blocchi nell'APPENDICE F, con particolare riferimento all'organizzazione della memoria e alle caratteristiche della CPU;
- la seconda mettendo a disposizione del lettore un'esperienza biennale maturata nella realizzazione di software applicativo sul sistema TI 99/4A, per cui alla fine di ciascun capitolo vengono dati dei consigli di utilità pratica.

Il libro è organizzato in cinque capitoli; il primo capitolo comprende descrizione e installazione del sistema TI 99/4A; i capitoli secondo e terzo illustrano rispettivamente il linguaggio BASIC e BASIC ESTESO; nel capitolo quarto viene descritta la architettura del TI 99/4A e relativi schemi a blocchi, il microprocessore a 16 bit TMS 9900 utilizzato per la realizzazione della CPU, e le istruzioni ASSEMBLER che sono tipiche del TMS 9900 e della sua filosofia.

Questa parte è rivolta a coloro che, partendo dalla conoscenza potenziale del microprocessore, vogliono utilizzare il linguaggio ASSEMBLER del 9900 per poter gestire al meglio tutte le risorse: dai registri interni al memory mapped del video, alle memorie di massa o periferiche che costituiscono il sistema TI 99/4A.

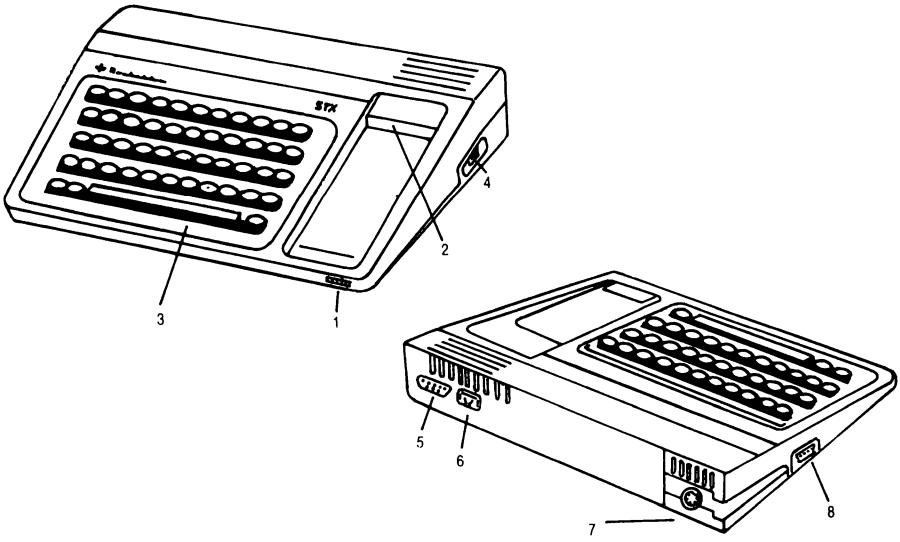
Il capitolo quinto sviluppa programmi di utilità e giochi in linguaggio BASIC (di cui vengono forniti i listati sorgenti) da far girare sul sistema TI 99/4A. Questo capitolo è legato logicamente al terzo e al quarto, in quanto fornisce delle applicazioni complete del linguaggio BASIC nonchè evidenzia la grande flessibilità dell'HOME COMPUTER TI.

1.2 L'INSTALLAZIONE DEL TI 99/4A

Prima di installare l'home computer TI 99/4A trovate una postazione al riparo da luce intensa e disponete il televisore o monitor insieme al computer su di un tavolo o scrivania, vicino a una presa di corrente multipla.

L'home computer TI 99/4A è fornito di un modulatore che da un lato va inserito nel Jack dell'antenna del televisore e dall'altro alla presa posteriore del computer e dell'alimentatore, il cui cavo va inserito direttamente nella parte posteriore del computer.

Vediamo in dettaglio le prese e porte presenti nella parte posteriore e laterali della consolle, qui riprodotta in vista frontale e laterale:



Consolle TI 99/4A

1) È l'interruttore ON/OFF con spia luminosa rossa, accesa nello stato ON; funge anche da RESET.

2) Porta per Moduli SSS(Solid State Software) contenenti sia programmi applicativi sia software di sistema.

3) Tastiera Qwerty per digitare le informazioni da inviare al computer.

4) Porta per l'espansione del sistema.

5) Presa 9-pin per collegare il cavetto a due canali per registratori a cassetta.

6) Presa 4-pin per il cavo dell'alimentatore.

7) Presa 6-pin per inserire il cavo del modulatore.

8) Presa 9-pin per inserire il cavetto dei comandi a distanza (Joystick).

N.B. Non confondere LA PRESA LATERALE dei comandi a distanza con la Presa posteriore del cavetto per registratori.

Procedere ai seguenti collegamenti:

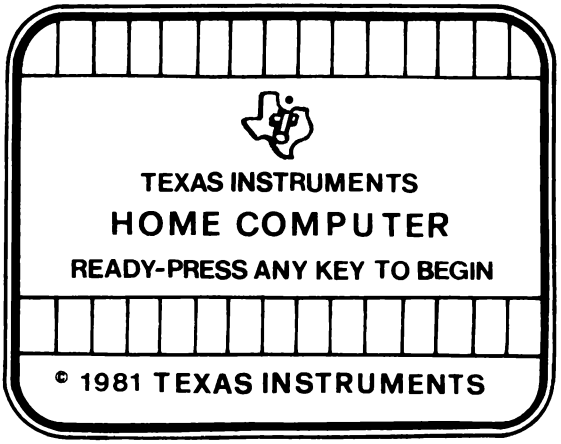
1) Inserite i cavetti del modulatore PAL nella parte posteriore del computer e nella presa dell'antenna VHF del televisore.

2) Inserite l'alimentatore nella rete e nella presa posteriore del computer.

3) Accendete il televisore.

4) Accendete il TI 99/4A mediante l'interruttore ON/OFF anteriore.

5) Sintonizzate il televisore nel canale 36 ricercando il segnale del computer, apparirà sul video la sottostante immagine:



Qualora non riusciate ad ottenere il segnale video indicato, controllate che:

1) I collegamenti fra computer, alimentazione, modulatore e rete siano corretti;
2) la rete sia attivata;
3) spegnete e accendete diverse volte, mediante l'interruttore anteriore ON/OFF del computer.

Se utilizzate il Monitor invece del televisore, non sarà più necessario il modulatore e sarà sufficiente un cavetto da collegare direttamente al vostro computer TI 99/4A.

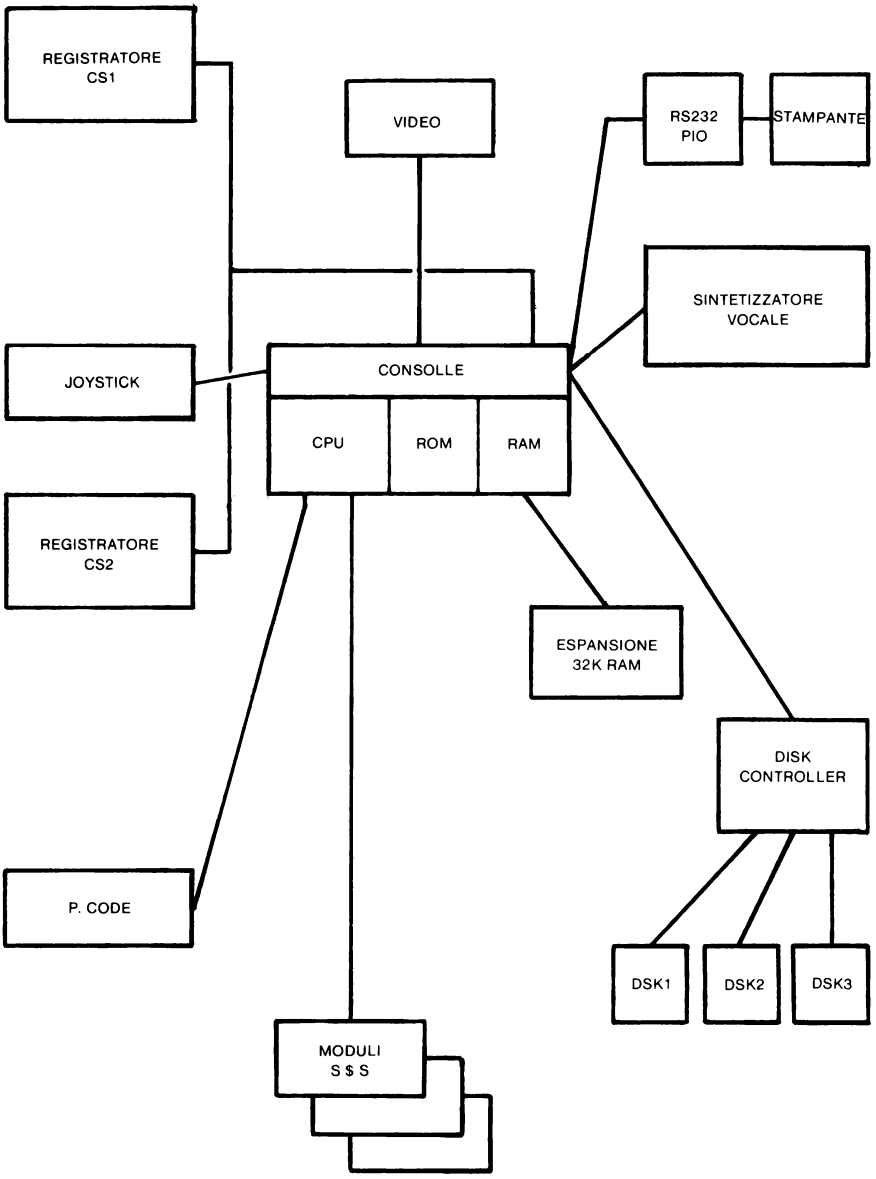
1.3 CARATTERISTICHE DEL COMPUTER TI 99/4A

- CPU (Central Processing Unit), ALU (Arithmetic Logic Unit) realizzate con il microprocessore 9900 TI a 16 bit (che sarà descritto nella seconda parte del libro).
- Tastiera Qwerty a 48 tasti con ripetizione automatica, tasti di controllo, tasti di funzione, lettere maiuscole e minuscole.
- Memoria interna di elaborazione RAM (Random Access Memory) 16 Kbyte espandibile fino a 48 Kbyte con scheda da 32 Kbyte.
- Memoria interna ROM (Read Only Memory) 26 Kbyte espandibile con i moduli SSS (Solid State Software) ad oltre 60 Kbyte.
- Risoluzione immagine 32 caratteri su 24 linee, ciascun carattere rappresentato con matrice 8×8 punti, per totali 49152 punti.
- Software di base:
  Interprete BASIC (14 Kbyte)
  Interprete L. Grafico (3 Kbyte)
  Monitor (4.4 Kbyte).
- Sottoprogrammi di libreria che permettono di definire caratteri propri con codice esadecimale, di utilizzare 16 colori sia per lo sfondo che per i caratteri, di programmare tre suoni simultaneamente (accordi) su cinque ottave con una ottava di rumori a spettro diverso.
- Possibilità di collegare due registratori a cassette per salvare programmi e file di dati.
- Possibilità di collegare il sintetizzatore vocale per ascoltare frasi inserite nel corpo del programma o qualsiasi fonema italiano o inglese.
- Possibilità di collegare l'interfaccia RS232 con porta seriale per collegamento stampanti standard RS232 Centronix o comunicazioni via modem su cavo telefonico fra diversi computer; l'interfaccia è anche fornita di una porta parallela PIO di I/O.
- Possibilità di collegare un disk controller per unità a dischi floppy da 89 Kbyte e 5" 1/4, un controller gestisce fino a tre unità a dischi.
- Possibilità di inserire direttamente in consolle il modulo SSS per il BASIC ESTESO che potenzia la capacità grafica e le istruzioni Basic.
- Possibilità di inserire direttamente in consolle una MINI MEMORY di espansione RAM di 4 Kbyte, con possibilità di programmare routine Assembler da richiamare da programma Basic, nonchè indirizzare la VDP RAM (Video Display Processor RAM) per grafica ad alta risoluzione, ciascun punto può essere colorato con 16 colori a scelta.

- Possibilità di utilizzare il linguaggio TI LOGO come linguaggio di didattica e l'Assembler del TMS 9900 per indirizzare e utilizzare tutte le risorse del computer.
- Possibilità di interfacciarsi con una scheda P-CODE per utilizzare il compilatore UCSD Pascal.

1.4 PERIFERICHE ED INTERFACCE

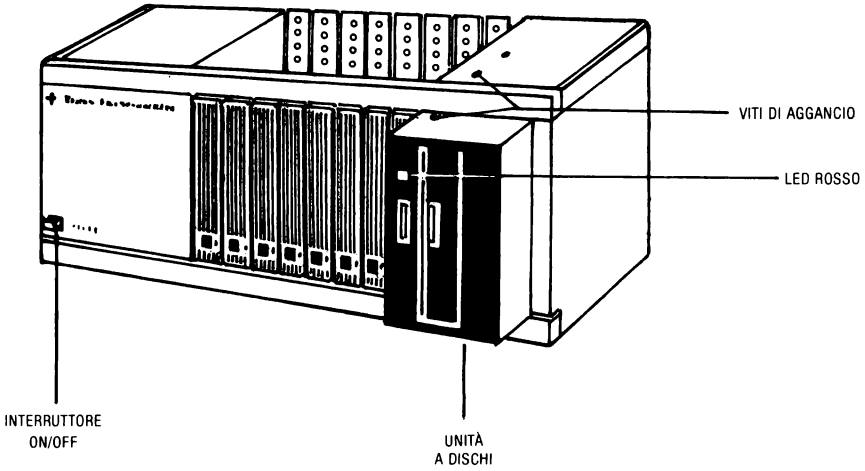
Lo schema in figura rappresenta una configurazione espansa del sistema TI 99/4A:

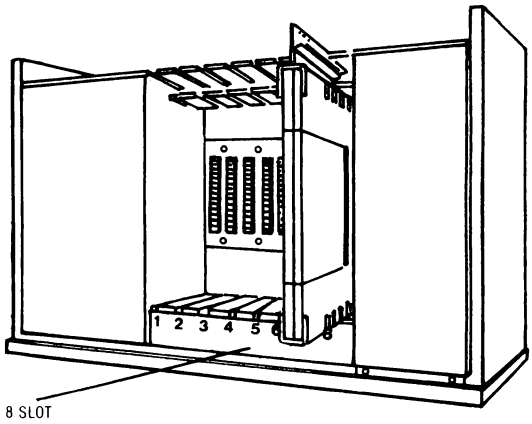


– Il registratore a cassette è un comune registratore con le seguenti caratteristiche:
  1) tre uscite a Jack per microfono, cuffia, comando a distanza;
  2) contagiri per poter controllare il nastro utilizzato;
  3) regolatore di toni e volume;
  4) monofonico.

– I moduli SSS si inseriscono direttamente in consolle nella porta predisposta e contengono sia programmi applicativi sia giochi sia sistema operativo.

– Le interfacce RS232, il controller per l'unità a dischi, un driver e l'espansione di memoria RAM sono inseribili nelle slot del Peripheral BOX (ESPANSIONE DEL SISTEMA) che contiene 8 slot per alloggiare altrettante schede di espansione del sistema; vedi figura:





– L'interfaccia RS232 è fornita di due porte: una seriale RS232 e una parallela PIO (Parallel Input Output), con la possibilità di collegare un cavo a Y per modificare la PIO in una seconda porta seriale RS232, quest'ultima è chiamata RS232/2.

È possibile anche inserire una seconda scheda RS232 nel sistema di periferiche, se ciò avviene la porta seriale è identificata con RS232/3 e la seconda porta parallela è identificata con PIO/2. Nota che RS232/1 e PIO/1 sono identificate dal computer senza specificare l'unità.

I vari comandi e istruzioni per gestire i file di stampa verranno illustrati in seguito; qui si dà una descrizione delle opzioni software ammesse dall'interfaccia RS232:

BAUD RATE – Velocità di trasmissione delle informazioni;

DATA BITS – Numero di bit di informazione per esprimere il dato;

PARITY – Bit di controllo di parità nella trasmissione dei dati;

STOP BITS – Numero di bit inviati alla fine di ogni carattere;

NULLS – Sei caratteri "nulli" per la sincronizzazione del ritorno carrello della stampante;

CARRIAGE RETURN E LINE FEED OFF – Disabilita l'aggiunta automatica di un ritorno carrello e di passaggio alla linea successiva di un blocco di informazioni;

LINE FEDD OFF – Disabilita il passaggio alla linea successiva alla fine di un blocco di informazioni.

Si fa riferimento all'APPENDICE A per ulteriori indicazioni relative alla porta seriale RS232 e quella parallela PIO.

– L'interfaccia controller dell'unità a dischi si inserisce nella slot n° 8 del sistema di espansione e riesce a controllare tre unità a dischi per floppy da 89 Kbyte che vengono installati secondo le norme descritte nei rispettivi manuali.

– L'unità a dischi inserita nel Sistema di espansione è identificata come "DSKI" e come tale viene riconosciuta per utilizzarla da memoria di massa dove salvare programmi e file di dati.

Per la gestione dei dischi, insieme al driver, è fornito un modulo SSS contenente il sistema operativo DOS (Disk Operative System) che consente di inizializzare i dischi a 40 tracce, fare il catalogo del disco, assegnare il nome al disco, cancellare programmi o file, copiare i dischi, riassegnare i nomi ai programmi, stampare la mappa del disco.

I comandi sono visualizzati in menù facili da seguire; si consulti il manuale dell'unità a dischi per tutte le informazioni relative al DOS.

- L'espansione di memoria RAM di 32 Kbyte è inseribile nella slot n° 2 del sistema di espansione; per l'organizzazione della memoria si fa riferimento nell'APPENDICE B.

- Il registratore a cassetta è un comune registratore il quale abbia le caratteristiche descritte precedentemente. Il cavetto per registratori consente di collegare due unità separate, CS1 e CS2; CS1 può essere utilizzata in lettura/scrittura, CS2 solo in lettura.

  Inserite i tre Jack nel registratore CS1:
  il ROSSO nella presa microfono
  il NERO nella presa comando a distanza (Jack più piccolo)
  il BIANCO nella presa cuffia.

- La stampante può essere una standard seriale RS232C o parallela a 80 colonne, che va collegata mediante cavo all'interfaccia RS232.
  Il manuale della stampante consente un settaggio dei vari switch presenti in essa per regolare la velocità, il salto pagina ecc.

- Il sintetizzatore vocale che si collega direttamente nella porta laterale della consolle consente di ascoltare frasi inserite da programma Basic. È disponibile anche il modulo SSS Terminal Emulator II che interpreta frasi e fonemi in lingua italiana o inglese.

- Il televisore o il monitor vanno installati come descritto nel primo paragrafo; non c'è alcuna differenza fra di essi se non la soppressione del modulatore nel caso si usi il monitor.
  Sedici sono i colori utilizzabili in FOREGROUND (Primo piano) e in BACKGROUND (sfondo).

- I comandi a distanza (Joystick) possono essere programmati da software nei loro spostamenti NORD, EST, OVEST, SUD e vanno utilizzati sempre con il tasto ALPHALOCK alzato.

1.5 CONSIGLI PRATICI

1) Qualora il registratore non legga o scriva i vostri programmi, controllate che i collegamenti corrispondano effettivamente alle modalità descritte; mettete il volume al massimo, distanziatelo di almeno mezzo metro dal video; se state registrando, provate a staccare il Jack della cuffia, se state leggendo provate a staccare il Jack rosso del microfono, se avete il comando a distanza staccatelo e date manualmente lo start al registratore. Qualora ancora il registratore non legga o scriva correttamente, verificate la sua compatibilità.
   Utilizzate nastri C46 oppure C60 di ottima qualità. Per evitare la smagnetizzazione dei nastri non avvicinateli a sorgenti elettromagnetiche e pulite bene le testine del registratore.

2) Per controllare che le interfacce nel Sistema di Espansione sono attivate, controllate che le rispettive luci frontali del sistema siano accese, nonché il led rosso dell'unità a dischi qualora sia attivato.
   Tutte le schede devono essere inserite fino in fondo nelle slot ed essere allineate per poter chiudere il Box e attivarlo con la alimentazione prevista. Nella fase di installazione o di disinserimento delle schede, togliete sempre l'alimentazione e controllate le operazioni sui rispettivi manuali. Prima dell'attivazione del Sistema di Espansione con tutte le schede, verificate che sia chiuso perfettamente.
   Per disinserire il controller dell'unità a dischi dalla slot n° 8 dovete sempre prima sfilare l'unità a dischi che è ad esso connesso con un cavetto interno che ha un solo verso di connessione.

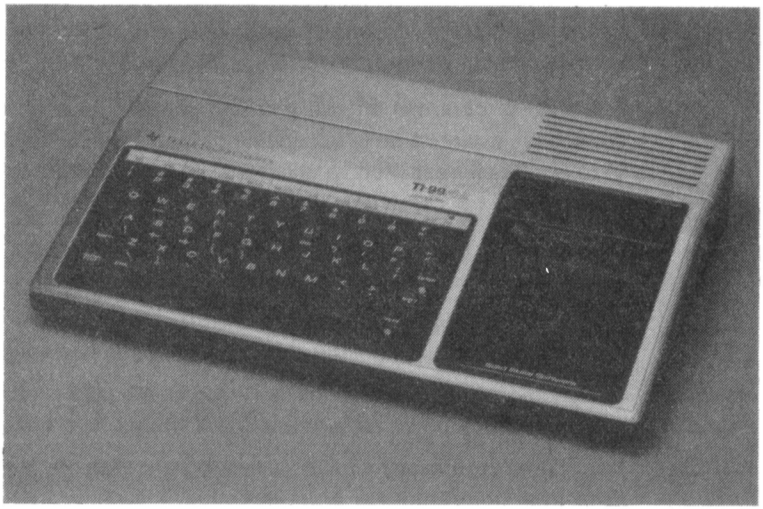
3) È disponibile un modulo SSS di diagnostica che effettua i test delle memorie, dei colori, della tastiera, delle porte, dei suoni.

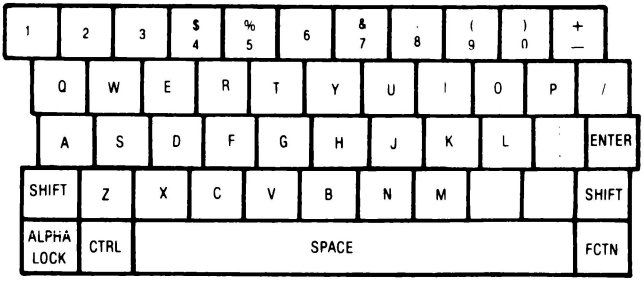
CAPITOLO II

PROGRAMMANDO IN TI BASIC

2.1 DESCRIZIONE DELLA TASTIERA

La tastiera è una Qwerty a 48 tasti, con la possibilità della ripetizione automatica qualora si tiene premuto qualsiasi tasto più di un secondo. In figura è mostrata la fotografia della tastiera e sotto lo schema per evidenziarne i tasti:





Descrizione dei tasti:

SPACE	Barra spaziatrice per cancellare o lasciare degli spazi.
ENTER	Serve per inviare qualsiasi comando o istruzione al computer, quindi va premuto dopo la battitura dei comandi o istruzioni.
ALPHA LOCK	È utilizzato per la selezione dei caratteri maiuscoli, i caratteri numerici rimangono inalterati.
SHIFT	Sono due tasti con uguale funzione; premendo uno di essi contemporaneamente al tasto dove sono presenti due simboli permette di scrivere quello in alto.
FCTN	È il tasto di funzione che va utilizzato sempre contemporaneamente ai tasti che presentano dei simboli frontalmente esempio E, D, S, X ecc., per realizzare le funzioni indicate; oppure contemporaneamente ai tasti numerici da 1 a 9 e Zero per realizzare le funzioni descritte nella striscia di plastica mobile inserita nella cornice superiore. Per una facile individuazione, frontalmente al tasto c'è un dischetto grigio, così pure sulla striscia mobile in corrispondenza delle diciture INS, DEL ecc.
CTRL	È il tasto di controllo che va premuto soltanto se sono definite nella striscia mobile delle funzioni in corrispondenza del dischetto rosso. Per il Basic non va utilizzato. Va utilizzato per le telecomunicazioni fra computer.
I tasti FCTN e +	Realizzano la funzione QUIT, ossia la cancellazione di tutto ciò che è in memoria RAM e si ritorna al "Titolo principale".
I tasti FCTN e 1	Realizzano la funzione DEL, ossia la cancellazione di una lettera, numero o simbolo sulla linea che state battendo o correggendo.
I tasti FCTN e 2	Realizzano la funzione INS, ossia l'inserimento di una lettera, numero o simbolo sulla linea che state battendo o correggendo.
I tasti FCTN e 3	Realizzano la funzione ERASE, ossia la cancellazione dell'intera linea che state battendo o correggendo.
I tasti FCTN e 4	Realizzano la funzione CLEAR, ossia fermano l'esecuzione di un programma o lo scorrere del listato sorgente Basic; questa operazione è detta "BREAKPOINT"
FCTN e S ←	Realizzano la funzione LEFT, ossia permettono di scorrere il cursore sulla linea nella direzione di sinistra senza cancellare i caratteri sullo schermo.
FCTN e D →	Realizzano la funzione RIGHT, ossia permettono di scorrere il cursore sulla linea nella direzione di destra senza cancellare i caratteri sullo schermo.

FCTN e E↑ Realizzano la funzione UP, ossia lo scrollo in su delle istruzioni componenti il listato sorgente Basic.

FCTN e X↓ Realizzano la funzione DOWN, ossia lo scrollo in giù delle istruzioni componenti il listato sorgente Basic.

I seguenti tasti realizzano delle funzioni speciali utilizzate soprattutto quando si inseriscono i moduli SSS in consolle:

FCTN e 5	BEGIN ossia iniziare
FCTN e 6	PROC'D ossia proseguire
FCTN e 7	AID ossia aggiungere
FCTN e 8	REDO ossia rifare
FCTN e 9	BACK ossia tornare indietro

Tasti per operazione aritmetiche

Questi sono utilizzati per eseguire espressioni numeriche con operazioni di somma, moltiplicazione, elevamento a potenza, divisione, sottrazione, confronto numerico:

\+ addizione
* moltiplicazione
– sottrazione
/ divisione
= uguale
^ elevamento a potenza
> maggiore
< minore

La tastiera così descritta nei suoi singoli tasti è nella configurazione standard e potete avvalervi di tutte le funzioni sopra descritte per redigere a velocità elevata e correttamente il programma sorgente Basic da voi realizzato.

Questa configurazione corrisponde a unità '0' ossia i codici dei caratteri sono quelli Ascii da 32 a 127 descritti in APPENDICE C. Questo perchè la configurazione della tastiera può essere cambiata e corrispondere a unità 1, 2, 3, 4, 5.

2.2 COMANDI DI SISTEMA OPERATIVO

Prima di iniziare la descrizione dei singoli comandi si sottolinea, per coloro che si avvicinano per la prima volta alla programmazione, la distinzione tra comandi di Sistema Operativo e istruzioni di linguaggio BASIC; i primi consentono di eseguire, stampare, visualizzare, correggere, controllare i programmi che, nel vostro caso, sono costituiti da un insieme di istruzioni BASIC.

Altra distinzione da fare è tra software del sistema operativo, o di base, e software applicativo. Il primo è inteso come l'insieme di tutti i programmi memorizzati in ROM o caricati in parte in RAM che la CPU utilizza per gestire il sistema, inviando e ricevendo segnali ed informazioni. Il secondo è inteso come l'insieme di programmi sviluppati per la risoluzione al Computer di problemi specifici, come calcolo telai, inventario, fatturazione, calcoli numerici ecc.

Dopo aver installato l'Home Computer nella configurazione base, consolle, video, modulatore, alimentatore, e come memoria di massa un registratore a cassetta, siete pronti per battere le prime istruzioni Basic e salvare su nastro quello che avete digitato per esercitarvi.

Dal "Titolo principale" passate all'interprete Basic premendo un tasto qualsiasi e sarà visualizzato:

```
                    TI BASIC READY
```

```
battete: 10 PRINT "STO IMPARANDO A PROGRAMMARE IN BASIC"
                                                  premete ENTER
         20 PRINT "IL MIO NOME È FRANCESCO"
                                                  premete ENTER
         30 PRINT "IL COMPUTER È UN MEZZO POTENTE"
                                                  premete ENTER
battete:    RUN                                   premete ENTER
```

RUN

è il comando per mandare in esecuzione il vostro programma sorgente presente in RAM.

Essendo PRINT l'istruzione di stampa, sullo schermo sarà visualizzato:

```
STO IMPARANDO A PROGRAMMARE IN BASIC
IL MIO NOME È FRANCESCO
IL COMPUTER È UN MEZZO POTENTE
```

*** * DONE * *** questo messaggio avverte che è finita l'esecuzione del programma.

Se volete listare le tre linee 10, 20, 30 sullo schermo battete il comando LIST e premete ENTER.

LIST

è il comando che permette di visualizzare il vostro programma BASIC a partire da una linea specificata.

Il formato di LIST è il seguente:

LIST	lista tutte le linee di programma
LIST 10	lista solo la linea 10
LIST 10-50	lista le linee comprese fra 10 e 50
LIST 10–	lista tutte le linee a partire da 10
LIST –10	lista le linee precedenti a 10

Il 10 è stato posto come esempio ed è sostituibile da qualsiasi numero di linea presente nel programma.

Se avete la stampante, e volete listare il programma sulla carta,

battete: LIST "RS232.BA=1200"

premete ENTER

BA=1200 indica che la velocità di trasmissione è 1200 BAUD.

Qualora abbiate la stampante settata a 300 BAUD, il computer automaticlamente assegna la velocità di trasmissione, per cui se volete listare il programma sulla carta, battete:

LIST "RS232"

Qualora vogliate listare soltanto un insieme di linee di programma, battete:

LIST "RS232" :100-300

in questo caso vengono listate le linee da 100 a 300.

I comandi che permettono di leggere e scrivere su memorie di massa, sia registratore a cassetta, sia unità a dischi, sono:

- OLD
- SAVE

OLD CS1 permette di trasferire il programma residente su nastro nella memoria centrale di elaborazione.

OLD

rappresenta il comando di lettura, CS1 l'unità a registratore n° 1 che è contraddistinta dall'essere collegata con tre fili al computer (mentre CS2 è collegata con due fili ed è utilizzabile solo in lettura). Il nome del programma da caricare non va specificato in quanto il nastro deve essere precedentemente posizionato sul programma da leggere.

Non è prevista la ricerca del programma per mezzo di nomi.

Dopo aver premuto ENTER, sul video appaiono le seguenti scritte:

* REWIND CASSETTE TAPE CSI THEN PRESS ENTER
  (Riavvolgi il nastro dell'unità CS1 e poi premi ENTER).
* PRESS CASSETTE PLAY CS1 THEN PRESS ENTER
  (Premi il tasto avanti (Play) del registratore e premi ENTER).
* READING
  (Il computer avverte che sta leggendo).
* DATA OK
* PRESS CASSETTE STOP CS1
  THEN PRESS ENTER
  (Il computer ha letto bene i dati e avverte di premere lo STOP del registratore e il tasto ENTER).

Qualora si verifichi un errore di lettura, comparirà uno dei due messaggi di errore:

ERROR – NO DATA FOUND
(I dati non sono stati trovati).
ERROR IN DATA DETECTED
(Parte dei dati non sono registrati)

poi il computer vi chiede se volete fare un controllo o finire le operazioni:

* PRESS R TO READ CS1
  (Premi il tasto R per leggere)
* PRESS C TO CHECK
  (Premi il tasto C per rileggere)
* PRESS E TO EXIT
  (Premi il tasto E per finire).

Se disponete dell'unità a dischi, il comando di lettura è:

OLD DSKI. Nome programma.

OLD

rappresenta il comando per trasferire il programma dal disco nella memoria centrale; DSK1 l'unità a dischi n° 1, le altre unità a dischi sono identificate da DSK2, DSK3.

Per la lettura di file di dati, si rimanda al paragrafo "Gestione dei file".

Il comando che permette di salvare i programmi su memoria di massa quale il registratore a cassette è:

SAVE CS1

Questa fase di scrittura, che permette di conservare i programmi presenti in memoria centrale su nastro, è seguita dal computer con le stesse scritte indicate nella pagina precedente, con la sola differenza che adesso dovete premere il tasto RECORD del registratore:

* PRESS CASSETTE RECORD CS1
  THEN PRESS ENTER

* RECORDING
  (Il computer avverte che sta registrando).

  * CHECK TAPE (Y OR N)
  (Il computer vi chiede se volete fare un controllo, premete Y per si, N per no).

In caso di errore compare uno dei seguenti messaggi:

* I/O ERROR xy
  (Errore dovuto al cattivo collegamento del registratore).

  * CAN'T DO THAT
  (Il programma non è presente in memoria)

poi il computer vi chiede se volete fare un controllo o finire le operazioni:

PRESS R TO RECORD CS1
PRESS C TO CHECK
PRESS E TO EXIT

Se disponete dell'unità a dischi, il comando di scrittura è:

SAVE DSK1. Nome programma

SAVE

rappresenta il comando per trasferire il programma dalla memoria centrale sul disco; DSK1 rappresenta l'unità a dischi n° 1, le altre unità sono identificate da DSK2, DSK3.

Dovete sempre assegnare un nome al programma, costituito da caratteri alfanumerici, che permette di identificarlo ogni volta che lo si richiama.

Attenzione!!! se indicate con lo stesso nome due programmi diversi, il computer trascrive il secondo e cancella il primo.

Quando si scrive un programma in BASIC, si hanno, a partire dal numero di linea, quattro righe di schermo, che corrispondono a **28×4** caratteri utili.

CALL CLEAR è il sottoprogramma che permette di pulire lo schermo.

Quando si aggiunge nel programma questa istruzione, il computer cancella il listato e visualizza i risultati dei calcoli e tutte le espressioni volute.

Quando si scrivono le istruzioni con il numero di linea, i numeri possono essere anche scritti in ordine non crescente, il computer li ordina automaticamente:

```
20 PRINT "STO PARTENDO"                    premete ENTER
5  CALL CLEAR                              premete ENTER
10 PRINT "CARO FRANCO"                     premete ENTER
30 PRINT "PER L'AMERICA"                   premete ENTER
```

Battete il comando LIST e vedrete che le istruzioni sono disposte in ordine. Per eseguire le istruzioni dovete battere RUN e ottenete la stampa di quanto è indicato fra gli apici.

NUM

È il comando che assegna automaticamente i numeri di linea, potete anche specificare il numero di partenza e il passo, altrimenti il computer parte da 100 e incrementa di 10 in 10. Sperimentate battendo NUM e NUM 10,5 premete ENTER per inviare al computer il COMANDO.

D'ora in avanti ometterò "premete ENTER" perchè ormai sapete che per inviare o un comando o un'istruzione al computer dovete premere il tasto ENTER. Qualora commettiate degli errori sintattici o semantici, riferitevi all'APPENDICE D per leggere il significato di essi.

battete:

```
NUM
      100  PRINT "DEVO ESEGUIRE LE OPERAZIONI"
      110  A=100
      120  B=200
      130  C1=A*B
      140  C2=A+B
      150  C3=B-A
      160  C4=B/A
      170  PRINT C1,C2,C3,C4
```

Premete due volte ENTER e uscirete dalla numerazione automatica, battete RUN, il programma va in ESECUZIONE e sarà stampato:

```
 DEVO ESEGUIRE LE OPERAZIONI
 20000          300
 100            2

* * DONE * *
```

Aggiungete 90 CALL CLEAR e sarà pulito lo schermo quando mandate in esecuzione con RUN il programma.

Alle linee 110 e 120 abbiamo espresso le variabili numeriche A e B che hanno assunto il valore rispettivamente di 100 e di 200.

Alle linee 130-160 abbiamo espresso le variabili numeriche C1, C2, C3, C4 che conterranno il risultato delle rispettive operazioni indicate.

Alle linee 100 e 170 l'istruzione PRINT fa stampare il messaggio racchiuso fra " " e i valori delle variabili C1, C2, C3, C4.

RES

è il comando che assegna automaticamente una nuova sequenza dei numeri di linea in base al numero iniziale e all'incremento specificati; permette di cambiare i numeri di linea al programma.

Sperimentate con il programma precedente battendo

```
RES 10,5
```

listate il programma con LIST: la prima istruzione partirà dalla linea 10 e le successive avranno un incremento di 5.

I numeri di linea che potete utilizzare vanno da 1 a 32767; finora avete battuto dei numeri di linea con incremento 10 perchè ogni qualvolta volete inserire altre linee con nuove istruzioni, avete già lo spazio, senza alterare i numeri delle linee presenti.

Se avete l'esigenza di cancellare la memoria RAM di elaborazione per scrivere un nuovo programma, avete a disposizione due comandi BYE e NEW. Vediamo la differenza:

BYE

è il comando che cancella tutto ciò che è in memoria RAM, chiude tutti i file dati aperti (vedi paragrafo 'gestione dei file') e rimanda al "Titolo principale".

NEW

è il comando che cancella tutto ciò che è in memoria RAM, chiude tutti i file dati aperti (vedi il paragrafo 'gestione dei file') e rimanda all'interprete TI BASIC.

Proviamo a scrivere un semplice programma:

battete

```
NUM  10,10
     10  A$="HO IMPARATO MOLTI COMANDI"
     20  PRINT A$
     30  A=500
     40  B=300
     50  PRINT A;B
     60  C=A+B-50+40*2
     70  PRINT "C=";C
```

battete RUN

e il programma viene eseguito e i risultati appariranno sullo schermo.

```
 HO IMPARATO MOLTI COMANDI
 500      300
 C=830
* * DONE * *
```

Alla linea 10 abbiamo utilizzato una variabile alfanumerica il cui valore è rappresentato dal contenuto fra gli " ".

Tutte le variabili alfanumeriche devono terminare col simbolo $ es.: A$, CAR$, NOM$, TRENO$, J$, B$ ecc.

Alla linea 70 abbiamo scritto "C=" perchè oltre la stampa del valore di C volevamo anche la stampa della variabile C.

Se volete correggere una linea del programma, avete a disposizione il comando EDIT molto potente per scorrere istruzione per istruzione, intervenire sulla linea e modificarla.

EDIT

è il comando che consente di richiamare la linea da correggere o cambiare.

Se dopo aver scritto il programma precedente, battete EDIT 30 e premete ENTER verrà visualizzato:

```
30 A=500
```

Con i tasti FCTN e → potete scorrere il cursore sulla linea nella direzione di destra e correggere o cambiare i caratteri, ad es:

```
30 A=850          premete ENTER
```

quest'ultima rappresenta la nuova linea e potete verificarlo con LIST.

Qualora volete cambiare più linee, anzichè premere ENTER, premete i tasti FCTN e ↑ (su), FCTN e ↓ (giù) per scorrere tutte le linee del programma e solo alla fine delle correzioni di tutte le linee premete ENTER.

Avete la possibilità di un EDIT molto potente soprattutto se utilizzate la ripetizione automatica dei tasti e i tasti FCTN e 1 (INS) e i tasti FCTN e 2 (DELETE) per inserire o cancellare caratteri sulla linea.

BREAK

è il comando che permette di interrompere l'esecuzione del programma nei punti tuare dei controlli, è molto utile nella fase di correzione.

Il computer trovando il comando BREAK 120 ferma l'esecuzione in corso all'istruzione 120 e risponde:

```
* BREAKPOINT AT 120
```

Esempio:

```
NUM
      100  A$="DEVO IMPARARE IL BASIC"
      110  B$="PENSO CHE SIA FACILE"
      120  PRINT A$&"  "&B$
```

```
130  A=100*2
140  B=150
150  PRINT A;B
160  END
```

battete il comando BREAK 140 premete ENTER
battete il comando RUN.

L'esecuzione del programma si fermerà all'istruzione 140 consentendovi di effettuare dei controlli, è molto utile nella fase di correzione.

Alla linea 120 il simbolo di concatenazione & permette di collegare i due messaggi in A$ e B$, nonchè lo spazio fra " ".

CON

è il comando che permette di continuare l'esecuzione del programma una volta che è stato interrotto con BREAK.

Potete sperimentarlo con l'esempio precedente, battendo CON non appena avete controllato le istruzioni volute con l'arresto dell'esecuzione.

UNBREAK

è il comando che permette di annullare i punti di arresto definiti dal BREAK.

Nell'esempio precedente se battete

```
UNBREAK 140
```

otterrete l'esecuzione completa del programma senza interruzioni.

Oltre il BREAKPOINT ottenuto da software, ossia utilizzando il comando BREAK, potete ottenere il BRAKPOINT anche con i tasti FCTN e 4(CLEAR) che interrompono l'esecuzione in corso fornendovi il numero dell'ultima istruzione eseguita prima del BREAKPOINT e appare il messaggio:

```
* BREAKPOINT AT 150
```

150 è l'ultima istruzione eseguita.

TRACE

è il comando che consente di vedere sullo schermo i numeri di linea nell'ordine con cui le istruzioni vengono eseguite.

UNTRACE

è il comando che annulla il TRACE.

I comandi BREAK e TRACE sono utilissimi nella fase di DEBUGGING (correzioni) dei vostri programmi, soprattutto quando dovete trovare errori logici di programmazione dove non è sufficiente leggere il listato sorgente staticamente,ossia come scritto, ma bisogna effettuare una lettura dinamica seguendo l'ordine delle istruzioni così come vengono eseguite, e questo sarà evidente quando nel paragrafo successivo tratteremo le istruzioni di salto condizionato e salto incondizionato.

Sintatticamente i comandi vanno battuti sempre senza numero di linea e vanno seguiti da uno spazio bianco.

DELETE

è il comando che consente di cancellare programmi e file su disco o nastro.

Esempi:

```
DELETE  "DSK1.nome file"
DELETE  "DSK1.nome programma"
DELETE  CSI
```

Esso è utilizzabile anche da istruzione, es.:

```
10  INPUT"Nome programma":A$
20  DELETE A$
```

MODO IMMEDIATO

In tutti gli esempi analizzati sinora, abbiamo utilizzato sempre PRINT come istruzione, cioè col numero di linea.

Il computer può operare anche in Modo Immediato. Pensate ad una calcolatrice; volendo effettuare dei calcoli, potete battere in prossimità del prompt (>)

PRINT 30 * 40+1000
premete ENTER e ottenete: 2200

PRINT "STO LAVORANDO IN MODO IMMEDIATO"
ENTER
PRINT "NON C'È BISOGNO DEL COMANDO RUN"
ENTER

e leggerete tutto ciò che è espresso fra " ".

Effettuate altri esempi per fare le vostre verifiche; importante è notare che non state scrivendo un set di istruzioni con i numeri di linea, per cui non dovete mandare in esecuzione le linee con RUN, esse vanno in esecuzione in Modo Immediato.

Se sul video compaiono messaggi d'errore, consultate sempre l'Appendice D dove sono descritti tutti i messaggi d'errore, e adopo aver capito il significato dell'errore in esame procedete alle correzioni di sintassi, di semantica o di logica.

Comandi Analizzati in Questo Paragrafo:

RUN	ESEGUIRE
LIST	LISTARE
NUM	NUMERARE
RES	RINUMERARE
EDIT	REDIGERE, EDIFICARE
BREAK	FERMARE, STOPPARE
CON	CONTINUARE
TRACE	TRACCIARE
SAVE	SALVARE
OLD	RICHIAMARE
PRINT	STAMPARE
UNTRACE	ELIMINA IL TRACE
UNBREAK	ELIMINA IL BRREAK
DELETE	CANCELLARE

Coloro che si avvicinano per la prima volta ad un linguaggio di programmazione, possono ripetere mentalmente il significato italiano delle parole chiavi per ottenere una elasticità di utilizzo di tutti i comandi quando la pratica li richieda.

2.3 VARIABILI, TIPO E FORMATO

Battete

```
NUM
        100  A=30
        110  B$="CARMINE"
        120  C$="HO DECISO DI STUDIARE"
        130  PRINT A,B$,C$
        140  END
```

battete RUN e premete ENTER

Saranno scritti i valori delle variabili A,B$,C$ ossia rispettivamente 30, CARMINE, HO DECISO DI STUDIARE.

Quindi A,B$,C$ rappresentano le variabili e i risultati il valore che esse assumono nel corso del programma.

Il computer, quando si assegna il valore ad una variabile, memorizza in una locazione di memoria il valore ed esso sarà sempre uguale finchè non lo si aggiorna; così se nel programma precedente aggiungete le linee:

```
132  C=A*100
135  D$=B$&C$
137  PRINT C;D$
```

La variabile C conterrà il valore 30*100 ossia 3000 e D$ conterrà CARMINE HO DECISO DI STUDIARE, mentre i valori delle variabili A, B$ e C$ rimangono inalterati.

Il computer ad ogni nome di variabile associa un indirizzo e uno solo che conterrà il valore o numerico o alfanumerico, con la sola differenza che i nomi di variabili alfanumeriche devono terminare con $. Es: CAS$, A$, C$, BRAVO$, NOME$, IN$.

Invece i nomi di variabili numeriche sono Es: CASSA, A, BB, CCC, L, TRONCO, CARDINE, TAL, X,Y.

Il numero di caratteri consentiti sia per i nomi di variabili numeriche che alfanumeriche è 15.

Le parole chiavi del BASIC non possono essere utilizzate come nomi di variabili.

Le costanti numeriche possono avere valori compresi fra – 9.9999999999999 E127 e –1E–128 e

fra 1E-128 e 9.9999999999999 E127

Possono essere numeri reali positivi o negativi.

La notazione esponenziale permette di scrivere numeri molto piccoli e molto grandi; importante è ricordare che l'esponente deve variare fra −99 e +99 per la rappresentazione sul video.

dove

$$N = \text{Mantissa} \times 10^{\text{ESPONENTE}}$$

Esempi:

$35.4 \times 10^4 = 35.4\ E4$
$5.7 \times 10^{-4} = 5.7\ E4$
$2750 \times 10^5 = 275\ E4$
$-34.7 \times 10^2 = -34.7\ E2$

Le costanti alfanumeriche (o stringhe) possono essere lunghe fino a quattro righe di schermo, ossia 28×4 caratteri; 112 caratteri che possono contenere spazi, lettere, numeri, simboli; importante è che le stringhe siano racchiuse fra apici " ".

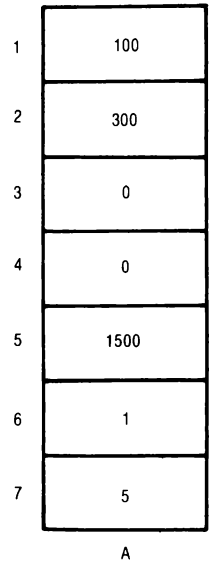
Utilizzando il carattere & avete sperimentato che potete collegare le stringhe e lasciare degli spazi con la stringa vuota " ".

Le variabili ad indici sono:

1) ARRAY (vettori monodimensionali)
2) MATRICI (vettori bidimensionali)

1) ARRAY

Pensate di avere una struttura con tanti cassetti sequenziali:



Nell'esempio sono 7 cassetti che contengono 7 valori numerici: 100,300,0,0,1500,1,5.
Le posizioni dei cassetti sono individuate da una variabile indice che varia da 1 a 7.
In questo caso A(1)=100, A(2)=300, A(3)=0, A(4)=0, A(5)=1500, A(6)=1, A(7)=5.

La struttura prende il nome di ARRAY e A(I) è la variabile ad indice I, A(I) rappresenta l'elemento dell'ARRAY (un cassetto) e A il nome dell'ARRAY numerico.
Per dichiarare al computer che volete utilizzare una variabile ad indice di lunghezza 7 dovete battere

```
10 DIM A(6)
```

In questo caso il computer riserverà 7 locazioni di memoria, in quanto parte dalla locazione 0, per contenere altrettanti valori numerici; potete assegnare i valori sia con

```
20  A(1)=100
30  A(2)=300
40  A(3)= 0
50  A(4)= 0
60  A(5)=1500
70  A(6)= 1
80  A(7)= 5
```

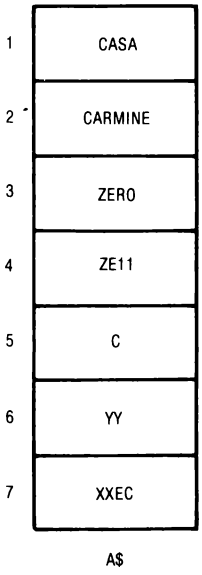
sia con le istruzioni READ e DATA:

```
20  READ   A(1),A(2),A(3),A(4),A(5),A(6),A(7);
30  DATA   100, 300, 0, 0, 1500, 1, 5.
```

l'associazione fra variabile e valore avviene per corrispondenza sequenziale, ossia ad A(1) sarà assegnato 100; A(2) sarà 300 ecc.

Se avete una struttura con sette elementi contenenti nomi o stringhe, l'ARRAY è di tipo alfanumerico per cui il nome deve terminare con $.

Esempio:



Per dichiarare al computer che volete utilizzare una variabile ad indice alfanumerica di lunghezza 7, dovete battere:

```
10 DIM A$(6)
```

A$ rappresenta il nome dell'ARRAY alfanumerico e A$(1) il primo elemento dell'ARRAY, il valore CASA è assegnato con:

```
20  A$(1) = "CASA"
```

oppure con:

```
20 READ A$(1)="CASA"
30  DATA CASA,CARMINE
```

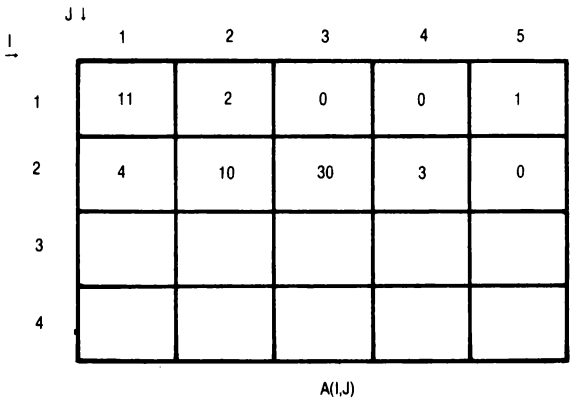
e questo vale per tutti e sette gli elementi.

Più riservate locazioni di memoria con DIM(Dimensione), meno spazio rimarrà in memoria RAM di elaborazione.

Fino a 10 elementi il computer assegna automaticamente il DIM.

2) MATRICI

Pensate di avere una tabella con righe e colonne; questa variabile prende il nome di matrice bidimensionale, in cui due indici scorrono le righe e le colonne.



La matrice A è scandita dall'indice di riga I che nel nostro esempio varia da 1 a 4, e dall'indice di colonna J che varia da 1 a 5.

Nel nostro esempio gli elementi della matrice sono J×I ossia 20; A rappresenta il nome della matrice numerica e A(I,J) è l'elemento della matrice.

Per dichiarare le dimensioni della matrice battete:

```
 5 OPTION BASE 1
10  DIM A(4,5)
```

Option Base 1 permette di far partire il conteggio dalla base 1.

La linea 10 dichiara la dimensione massima della matrice, ossia quanto spazio deve essere riservato in memoria; nell'esempio 20 locazioni per variabili numeriche.

Per assegnare i valori agli elementi della matrice battete:

```
20  A(1,1)=11
30  A(1,2)=2
40  A(1,3)=0
50  A(1,4)=0
60  A(1,5)=1
70  A(2,1)=4
    •
    •
    •
    •
```

Oppure battete:

```
20  READ A(1,1), A(1,2), A(1,3) .....
30  DATA 11, 2, 0 .....
```

dove l'associazione fra l'elemento A(1,1), e il valore 11 avviene per corrispondenza, così avanti per tutti gli altri elementi.

Queste variabili ad indice analizzate, ben si prestano alla logica del computer per contenere dati di tabelle o archivi di nomi.

Infatti pensate sempre ai cassetti dove potete mettere tutte le informazioni desiderate (numeriche, alfanumeriche) e qualora vogliate analizzare un cassetto è sufficiente richiamare l'elemento iesimo del vettore (o ARRAY) e leggerne il contenuto.

Un modo veloce per leggere i dati (numerici o alfanumerici) o per inserire i dati è quello di utilizzare i cicli FOR.....NEXT

Confrontate le due sequenze di istruzioni:

```
 5  DIMENSIONE A(5)          5 DIM A(5)
10  PER I=1 a 5             10 FOR I=1 TO 5
20  LEGGI A(I)              20 READ A(I)
30  CONTINUA FINO A I=5     30 NEXT I
40  DATI 100,200,2,3,7      40 DATA 100,200,2,3,7
50  FINE                    50 END
```

I è la variabile che controlla il ciclo, ossia permette di contare da 1 a 5 incrementandosi di una unità per volta (I= I+1).

In generale FOR.....NEXT permette di ripetere tutte le operazioni interne al ciclo, vuoi che siano calcoli vuoi che siano operazioni di letture o stampe.

Lo schema si presenta così:

```
10  FOR I=1        TO  N

            CALCOLI

            LETTURE

            STAMPE

50  NEXT I
```

Tutte le istruzioni interne al ciclo a partire dal FOR I fino al NEXT I vengono ripetute N volte.

Se volete la stampa di un elenco di nomi memorizzati in un vettore battete:

```
10  DIM NOMI$(5)
20  FOR I=1 TO 5
30  READ NOMI$(I)
40  PRINT NOMI$(I);
50  NEXT I
60  DATA CARLO, LUIGI,FRANCO,ADA,ENNIO
70  END
```

battete

```
RUN
```

Per pulire lo schermo aggiungete: 5 CALL CLEAR

In questo esempio NOMI$ rappresenta il nome del vettore alfanumerico, NOMI$(1) il primo elemento del vettore e CARLO il valore che assume la variabile NOMI$(I) dove I è il contatore del ciclo.

Per ogni FOR ci deve essere sempre un NEXT che chiude il ciclo.

Esempio:

```
NUM  10
      10  REM UTILIZZO MATRICE
      20  CALL CLEAR
      30  DIM M$(12)
      40  FOR N=1 TO 12
      50  READ M$(N)
      60  PRINT M$(N)
      70  NEXT N
      80  DATA GENNAIO,FEBBRAIO,MARZO,APRILE,MAGGIO,
          GIUGNO,LUGLIO,AGOSTO
      90  DATA SETTEMBRE,OTTOBRE,NOVEMBRE,DICEMBRE
     100  END
```

La linea 30 permette di riservare 13 locazioni alfanumeriche.

Alla linea 10 l'istruzione REM permette di scrivere dei commenti al programma.

Vediamo le istruzioni analizzate in questo paragrafo con il loro significato:

REM	COMMENTO
DIM	DIMENSIONE
OPTION BASE 1	BASE 1 PER MATRICI O ARRAY

2.4 ISTRUZIONI DI INPUT/OUTPUT

Vengono dette istruzioni di input le istruzioni di lettura, ossia quelle che vi permettono di effettuare l'introduzione di dati nel programma.

Esistono due modi per assegnare i valori di ingresso:

1) MODO STATICO 2) MODO DINAMICO

È detto modo Statico quello realizzato con le istruzioni READ e DATA, in quanto all'interno del programma, prima di mandarlo in esecuzione con RUN, dovete assegnare i valori alle variabili.

ESEMPIO:

```
NUM
      100  REM TABELLA CODICI
      110  DIM A(7)
      120  FOR I=1 TO 7
      130  READ A(I)
      140  PRINT I; "A(I)="; A(I)
      150  NEXT I
      160  DATA 10,20,50,100,200,300,400
      170  END
```

battete RUN

Alla linea 100 è stata introdotta una frase di commento con l'istruzione REM, avete a disposizione quattro righe di 28 caratteri per scrivere i vostri commenti al programma; alle linee 130 e 160 avviene la lettura dei codici che vengono memorizzati nell'ARRAY A(I) di otto elementi, la corrispondenza fra le variabili del READ e i valori del DATA avviene in sequenza; le linee comprese fra il 120 FOR I e 150 NEXT I vengono eseguite sette volte per la presenza del ciclo.

battete

```
NUM   100 REM TABELLA NOMI
      110  DIM T$(5)
      120  FOR K=1 TO 5
      130  READ T$(K)
      140  PRINT K; "T$(K)="; T$(K)
      150  NEXT K
      160  DATA CARMINE,GINO,ROSA,MARIA,ADA
      170  END
RUN
```

L'istruzione REM permette di scrivere le frasi di commento; DIM permette di dichiarare le dimensioni dell'ARRAY alfanumerico T$(K); le linee 130 e 160 permettono di leggere i nomi da memorizzare nell'ARRAY T$(K).

In entrambi gli esempi visti se volete cambiare i dati degli ARRAY A(I) e T$(K) dovete richiamare con EDIT 160 la linea del DATA e assegnare dei nuovi valori da voi prescelti.

2) È detto invece modo Dinamico quello realizzato con l'istruzione INPUT, che vi permette di assegnare i valori alle variabili del programma durante l'esecuzione di esso.

ESEMPIO:

```
NUM
      100  REM     CALCOLO
      110  INPUT A,B
      120  C=A*B+20
      130  PRINT "C=";C
      140  END
RUN
```

In questo esempio i valori numerici da dare alle variabili A e B vengono richiesti sul video da un punto interrogativo ?, digitate ad esempio 5,10 e premete ENTER, in questo caso A varrà 5 e B varrà 10, sarà stampato C=70 come risultato del calcolo della linea 120.

Qualora venga visualizzato il messaggio "TRY AGAIN" ("Battete ancora"), significa che non avete assegnato i valori nel formato corretto o tipo numerico/alfanumerico dichiarato.

Tutte le variabili che seguono l'istruzione INPUT devono essere battute nello stesso ordine seguite dalla virgola

ESEMPIO:

```
100  INPUT     A, B$, AST
```

in questo caso la variabile A numerica è seguita dalla variabile alfanumerica B$ e ancora dalla variabile numerica AST. Qualora vogliate ricordare a cosa è associato il valore della variabile espressa nell'INPUT potete aggiungere fra apici frasi di commento:

```
NUM
      100  INPUT "DIGITA IL NOME":B$
      110  INPUT "NUMERO":A
      120  INPUT "VIA":C$
```

```
        130   PRINT B$,A,C$
        140   END
RUN
```

Durante l'esecuzione del programma vi sarà richiesto il valore di B$, A, C$ e in funzione di ciò che battete otterrete la stampa.

INPUT è un'istruzione molto potente per la capacità di rendere i dati d'ingresso del tutto generali.

Un'istruzione di assegnazione è LET, viene utilizzata opzionalmente, esempi:

```
100  LET  A=200          equivalente  100  A=200
100  LET  B$="CARLO"     equivalente  100  B$="CARLO"
100  LET  C=A*1000       equivalente  100  C=A*1000
```

Vengono dette istruzioni di output le istruzioni di stampa o visualizzazione: PRINT, DISPLAY.

Nel corso degli esempi finora analizzati abbiamo più volte utilizzato l'istruzione PRINT, vediamo una nuova opzione per scrivere il dato alla colonna che desideriamo e poi riassumiamo tutte le opzioni possibili:

```
10  PRINT TAB(15);19
20  PRINT TAB(7);"CARLO"
```

In questo esempio 19 è scritto a partire dalla 15ª colonna e CARLO a partire dalla 7ª colonna.

TAB(N)	dove N è il numero di colonna da cui cominciare a stampare $1 \leq N \leq 28$
;	fa spaziare di 2 caratteri i numeri e collega le stringhe
,	fa spaziare di mezzo video (14 colonne) i numeri o lettere
:	fa saltare di un rigo
&	concatena le stringhe
" "	lascia degli spazi quanti sono gli spazi della stringa vuota
" ESEMPIO"	stampa il contenuto fra " "

con TAB(N) per N >28 è calcolato Modulo(N); per la stampa su carta $1 \leq N \leq 80$

Queste opzioni sono valide sia con PRINT che con DISPLAY; esercitatevi con

molte istruzioni di PRINT per imparare bene a impaginare i dati con tutti i formati possibili.

Ricordate che sono accettati anche i caratteri minuscoli e molte volte evidenziano bene le frasi di commento che volete visualizzare.

Nel cap. III del BASIC ESTESO analizzeremo le istruzioni PRINT AT, DISPLAY AT, PRINT USINS, DISPLAY USING.

2.5 ISTRUZIONI DI CONTROLLO

Volendo riassumere la struttura di un programma sorgente in BASIC alla luce degli argomenti sinora trattati, avremo a disposizione il seguente schema:

```
NUM
  100   Dichiarazione del tipo variabile (DIM,A(I),A,A$)
        Descrizione e Commenti (REM)
        Operazioni di lettura (INPUT,READ,DATA)


        Calcoli ed Elaborazioni


    .   Operazioni di stampa e scrittura(PRINT,DISPLAY)
  190   END
RUN
  * * DONE * *          Risultati
```

In questo schema tutte le istruzioni vengono eseguite sequenzialmente, dal numero di linea più piccolo al numero di linea più grande.

Esistono due tipi di istruzioni GOTO e IF THEN ELSE che vengono dette rispettivamente:

SALTO INCONDIZIONATO
SALTO CONDIZIONATO

che permettono di eseguire una o più istruzioni non sequezialmente, ossia permettono di saltare a numeri di linea presenti nel programma per effettuare delle operazioni e poi ritornare laddove era stata interrotta la sequenza o saltare in altri punti del programma.

Vediamo nei dettagli le istruzioni di salto:

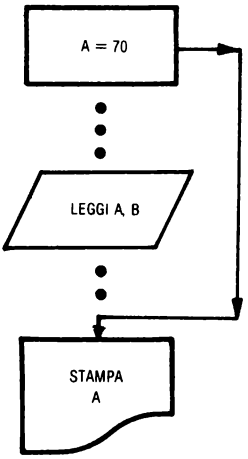
1) GOTO n — (Salta all'istruzione n)

2) IF THEN n ELSE m — (Se verificata la condizione salta all'istruzione n, altrimenti salta ad m)

3) ON GOTO n1,n2,n3,n4 — (Salta all'istruzione n1,n2,n3,n4 al variare di x da 1 a 4)

1) GOTO n

```
 90  A=70
100  GOTO 500
110  READ A,B
120  DATA 15,20
  .
  .
  .
  .
  .
500  PRINT "A";A
```

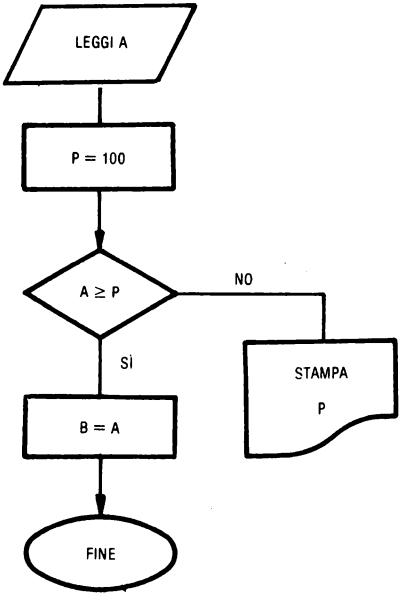


In questo esempio, visualizzato anche dal diagramma di flusso, c'è il salto all'istruzione di stampa effettuato con GOTO 500, questo permette di alterare l'esecuzione sequenziale delle istruzioni.

2) IF THEN n1 ELSE n2

ESEMPIO:

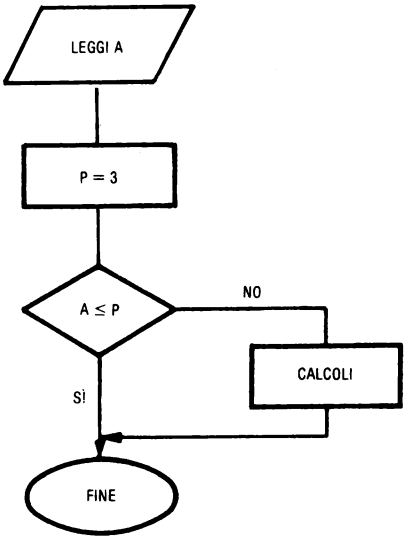
```
  5  INPUT A
 10  P=100
 20  IF A > = P THEN 50 ELSE 100
  .
  .
  .
 50  B=A
 60  GOTO 200
100  PRINT P
200  END
```



Alla linea 20 c'è il test fra A e P, se A > = P allora l'esecuzione continua all'istruzione 50 altrimenti a 100.

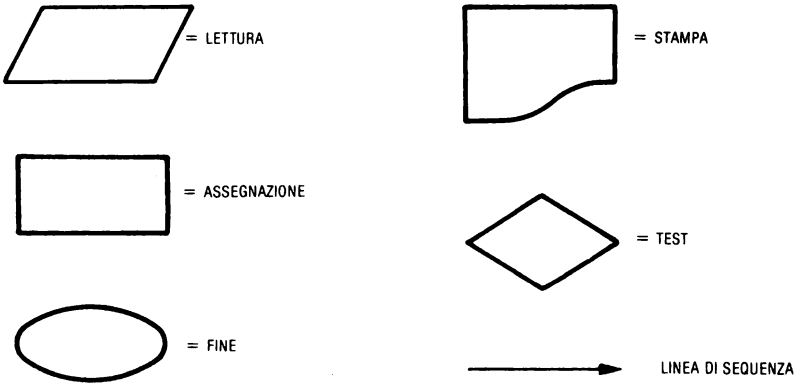
ESEMPIO:

```
10   INPUT A
20   P=3
30   IF A > = P THEN 100
40   C=A*P
.
.
.
100  END
```



Alla linea 30 c'è il test fra A e P, se A = < P allora l'esecuzione continua all'istruzione 100 altrimenti continua in sequenza alla istruzione 40.

Negli esempi visti abbiamo utilizzato anche la visualizzazione mediante diagrammi di flusso, vediamo il significato dei simboli grafici:

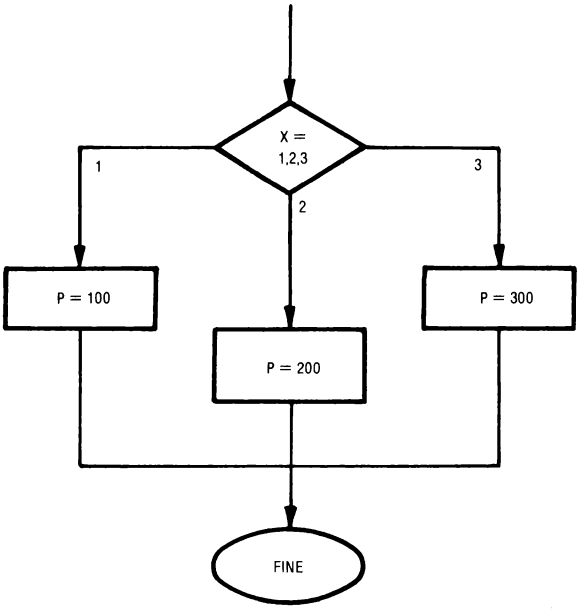


ESEMPIO: Ricerca del minimo

```
NUM
      100  INPUT "A=":A
      110  INPUT "B=":B
      120  IF A=B THEN 160
      130  IF A<B THEN 180
      140  PRINT"B è minore di A"
      150  GOTO 190
      160  PRINT "B=A"
      170  GOTO 190
      180  PRINT "B è maggiore di A"
      190  END
RUN
```

3) ONGOTO N1,n2,n3 .....

Questa istruzione di salto condizionato prende anche il nome di salto calcolato, infatti se dopo una serie di calcoli o scelta la variabile X assume il valore 1,2,3,4 ecc. si ha il salto rispettivamente alla linea n1,n2,n3,n4 ecc.

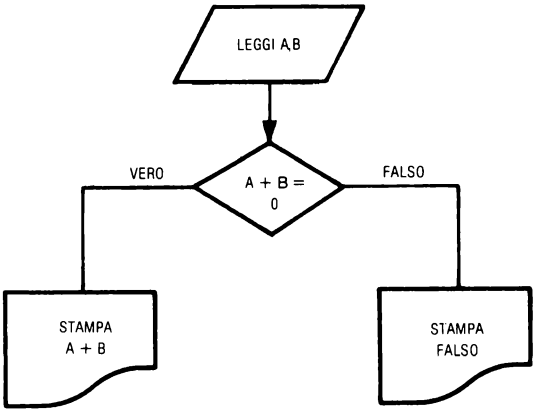



```
10   ONGOTO 60,70,80
.
.
.
.
60   P=100
65   GOTO 100
70   P=200
75   GOTO 100
80   P=300
100  END
```

OPERATORI RELAZIONALI AMMESSI:		*ESEMPIO:*
=	uguale	A = B
<	minore	A < B
>	maggiore	5 > 3
< >	diverso	A < > C
<=	minore o uguale	T < = P
>=	maggiore o uguale	T > = P
vero		A = —1
falso		A = 0

Analizziamo il caso vero o falso:

Il computer assegna ad una relazione falsa lo zero, ad una vera —1

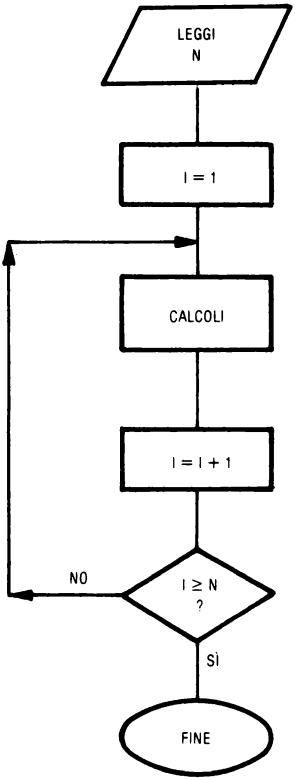


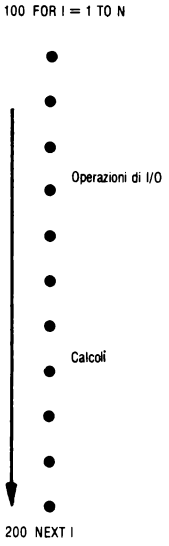

```
100 INPUT "A=  ":A
110 INPUT "B=  ":B
120 IF A+B THEN 150
130 PRINT "FALSO"
140 GOTO 200
150 PRINT A,B,"VERO"
200 GOTO 100
```

Alla linea 120 se l'espressione A+B è diversa da 0(ossia VERO) si ha il salto alla linea 150 altrimenti A+B=0 (ossia FALSO) si ha il salto alla linea 130. Alla linea 200 c'è l'istruzione GOTO che rimanda all'inizio di programma, per cui se volete fermare l'esecuzione dovete utilizzare i tasti FCTN e 4 (BREAKPOINT).

Alla luce dello studio dei salti condizionati vediamo il ciclo FOR NEXT, studiato nel paragrafo precedente, come è realizzato logicamente:

```
100 FOR I=1 TO N
  .
  .
  .
  .  Operazioni di I/O
  .
  .
  .
  .  Calcoli
  .
  .
  .
  .
200 NEXT I
```





Il ciclo FOR NEXT contiene implicitamente il controllo o test della variabile I che deve variare da 1 a N con passo unitario.

C'è da aggiungere che potete variare il passo aggiungendo STEP m, dove m è un numero relativo positivo o negativo.

ESEMPIO:

```
100  FOR I=1 TO 10 STEP 2
      .
      .  Operazioni di I/O
      .
      .  Calcoli
      .
200  NEXT I
```

Il ciclo è ripetuto 5 volte essendo il passo uguale a 2.

Qualora vogliate fermare l'istruzione in esecuzione per alcuni secondi, es. la stampa di risultati, potete utilizzare il seguente ciclo che prende il nome di ciclo di ritardo:

```
100  FOR I=1 TO 1000
110  NEXT I
```

In tal caso non inserendo altre istruzioni fra 100 e 110, il computer perderà del tempo contando fino a 1000, questo ritarderà l'esecuzione delle istruzioni successive. Potete variare 1000 con numeri più grandi o più piccoli a seconda se volete ritardare di molti secondi o pochi secondi l'esecuzione.

ESEMPIO:

```
NUM  10,10

      10  REM Utilizzo Cicli
      20  DIM M$(12,10)
      30  FOR N=I TO 12
      40  READ M$(N,10)
      50  PRINT M$(N,10)
      60  NEXT N
      70  DATA Ariete,Toro,Gemelli,Cancro,Leone,Vergine
      80  DATA Bilancia,Scorpione,Sagittario,Capricorno,Acquario,Pesci
      90  FOR K=1 TO 2000
     100  NEXT K
     110  END
RUN
```

Il primo ciclo permette la lettura e la stampa dei dodici segni zodiacali, il secondo ciclo ritarda l'esecuzione dell'istruzione END.

ESEMPIO:

Creazione del vocabolario

Le linee 40-90 permettono la creazione di due vettori ING$ e ITA$ che contengono rispettivamente i termini inglesi ed italiani. La variabile Q$ contiene la parola da tradurre.

La linea 140 effettua la ricerca della parola inglese nel vettore ING$.

```
10 CALL CLEAR
20 REM  VOCABOLARIO
30 DIM ING$(10),ITA$(10)
40 FOR I=1 TO 10
50 READ ING$(I),ITA$(I)
60 NEXT I
70 DATA HOUSE,CASA,UNCLE,ZIO,DOOR,PORTA,PENCIL,PENNA
80 DATA TABLE,TAVOLO,APPLE,MELA,MEAT,CARNE
90 DATA HORSE,CAVALLO,DOG,CANE,CAT,GATTO
100 INPUT "Parola da tradurre? ":Q$
110 PRINT :::::
120 R$="                              PAROLA NON PRESENTE"
130 FOR I=1 TO 10
140 IF Q$=ING$(I)THEN 160
150 GO TO 170
160 R$=ITA$(I)
170 NEXT I
180 PRINT Q$;" significa : ";R$:::
190 GOTO 100
```

Potete creare un vocabolario a vostro piacimento allungando i vettori e inserendo le parole da voi richieste.

ESEMPIO:

Tabella da ordinare in senso crescente.

Le linee 45-65 permettono l'inserimento dei valori da ordinare nel vettore VA. Il valore 9999 è fittizio ed indica che non vogliamo inserire 100 valori da ordinare ma di meno. La variabile I contiene il numero di valori inseriti.

Le linee 80-120 permettono l'ordinamento dei valori effettuando il confronto del primo con il successivo, nel caso che il primo valore sia maggiore del successivo avviene lo scambio di essi per mezzo delle linee 95-105.

```
10 REM TABELLA DA ORDINARE
15 REM VA(I) CONTIENE I VALORI DA ORDINARE
16 REM TEMP VARIABILE TEMPORANEA
20 CALL CLEAR
25 OPTION BASE 1
30 DIM VA(100)
35 PRINT "INSERIMENTO VALORI","Digita 9999 per finire": : : :
40 FOR I=1 TO 100
45 INPUT "VALORE ? ":VA(I)
50 IF VA(I)=9999 THEN 60
55 NEXT I
60 I=I-1
65 CALL CLEAR
70 PRINT "ORDINAMENTO  CRESCENTE": : : :
75 FOR J=1 TO I
80 FOR K=J+1 TO I+1
85 IF VA(J)<=VA(K)THEN 105
90 TEMP=VA(J)
95 VA(J)=VA(K)
100 VA(K)=TEMP
105 NEXT K
110 PRINT "VA(";STR$(J);")=";VA(J)
115 NEXT J
```

RUN

Questo metodo di ordinamento crescente è detto "GORGOGLIAMENTO" perchè alla fine di tutti gli scambi fra valori successivi il vettore risulta ordinato.

ESEMPIO di cicli nidificati:

```
NUM
      100  REM Cicli Nidificati
      110  FOR I=1 TO 2
      120  FOR K=1 TO 4
      130  READ VAL1,VAL2
      140  PRINT "VAL1=";VAL1;"VAL2=";VAL2
      150  NEXT K
      160  PRINT
      170  RESTORE 200
      180  NEXT I
      190  DATA 10,20,30,40,50
      200  DATA 3,4,5,6,7
      210  DATA 100,200,300,400
      220  END
RUN
```

In questo esempio abbiamo due cicli nidificati, nel senso che uno è interno all'altro, per cui per ogni valore di I (ciclo esterno) viene eseguito il ciclo interno controllato da K (ciclo interno).

Alla linea 170 abbiamo introdotto l'istruzione RESTORE 200 che ha la capacità di dire al computer quale linea di DATA deve utilizzare per il READ, nel nostro caso il DATA è quello di linea 200.

Con le istruzioni IF THEN ELSE e FOR NEXT potete implementare molte strutture di controllo che hanno il Test in testa, in coda, nel corpo del programma. Vediamo una serie di possibilità:

```
10  J=N
20  Q(J)=P(J)
30  J=J-1
40  IF J>=1 THEN 20
```

```
10  J=N
20  Q(J)=P(J)
30  J=J-1
40  IF J< > THEN 20
```

```
10  J=N
20  Q(J)=P(J)
30  IF J<= THEN 60
40  J=J-1
50  GOTO 20
```

```
10  J=0
20  J=J+1
30  Q(J)=P(J)
40  IF J < N THEN 20
```

```
10  J=1
20  Q(J)=P(J)
30  J=J+1
40  IF J< >N+1 THEN 20
```

```
10  J=N+1
20  IF J=1 THEN 60
30  J=J-1
40  Q(J)=P(J)
50  GOTO 20
```

In questo paragrafo abbiamo analizzato le istruzioni di controllo, per coloro che si avvicinano la prima volta è bene ripetere mentalmente il significato associato a ciascuna di esse:

GOTO n
Salta alla linea n

IF A > B THEN n
Se A è maggiore di B salta alla linea n, altrimenti continua in sequenza.

IF A > B THEN n ELSE m
Se A è maggiore di B salta alla linea n, altrimenti alla linea m.

FOR I=1 TO N STEP 1 NEXT I
Per I che parte da 1 fino ad N esegui tutte le istruzioni fino all'istruzione NEXT I con passo 1.

ON X GOTO n1,n2,n3

Al variare di X da 1 a 3 salta alla linea rispettivamente n1,n2,n3.

RESTORE n

Ripristina la linea n di DATA.

2.6 SOTTOPROGRAMMI PER I COLORI

Prima di descrivere i sottoprogrammi di libreria, ossia quelli presenti in ROM della consolle TI 99/4A, è bene che analizziamo la funzione e la struttura di un sottoprogramma.

Pensate ad un calcolo all'interno del programma principale che debba essere utilizzato più volte o ad una funzione richiamata più volte per aggiornare i valori, questo è il caso tipico in cui si deve creare un sottoprogramma che svolga la specifica funzione e richiamarlo quante volte ci occorre nel programma principale.

Osservate lo schema:

```
 10 PROGRAMMA PRINCIPALE
  .
  .
  .
 50  CHIAMATA SOTTOPROGRAMMA
  .
  .
  .
 80  CHIAMATA SOTTOPROGRAMMA
  .
  .
  .
150  FINE
  .
  .
  .
200  SOTTOPROGRAMMA
  .      (FUNZIONE)
  .
  .
  .
300  RETURN
```

Un sottoprogramma è un insieme di istruzioni, terminante con l'istruzione RE-

TURN, che realizza una specifica funzione o calcolo da poter utilizzare più volte nel corso dell'esecuzione del programma principale.

Le istruzioni che permettono il salto ai sottoprogrammi sono:

1) GOSUB n
2) ONGOSUB n_1, n_2, n_3....

– GOSUB n
permette di saltare al sottoprogramma iniziante all'istruzione numero n fino all'istruzione RETURN che riporta il controllo al programma principale, in particolare all'istruzione successiva al GOSUB n.
— ONGOSUB n_1, n_2, n_3....
permette di saltare a diversi sottoprogrammi inizianti alle istruzioni n_1, n_2, n_3 al variare di X da 1 a 3. In ogni caso eseguito il sottoprogramma selezionato dal valore di X, il controllo ritorna al programma principale all'istruzione successiva al ON x GOSUB n_1, n_2, n_3.

I sottoprogrammi di libreria, che permettono di cambiare colore allo schermo e ai caratteri creando la possibilità di combinazioni di colori a scelta dell'utente, sono:

1) CALL SCREEN (I)
2) CALL CLEAR
3) CALL COLOR (S,F,B)

1) CALL SCREEN (I)
permette di colorare lo schermo con uno dei sedici colori previsti, I è la variabile che varia da 1 a 16.

Codice I	Colore
1	Trasparente
2	Nero
3	Verde
4	Verde chiaro
5	Blu scuro
6	Blu chiaro
7	Rosso scuro
8	Viola

9	Rosso
10	Rosso chiaro
11	Giallo scuro
12	Giallo chiaro
13	Verde scuro
14	Magenta
15	Grigio
16	Bianco

2) CALL CLEAR

permette di pulire lo schermo, è molto utile per cancellare tutte le scritte o disegni dallo schermo per visualizzare nuovi caratteri o simboli.

3) CALL COLOR (S,F,B)

permette di assegnare il colore al carattere specificato dal set del codice ASCII.

Premesso che ciascun carattere è costituito da una matrice di punti 8x8, di cui una parte rappresenta il carattere vero e proprio (FOREGROUND, primo piano) e i rimanenti lo sfondo (BACKGROUND, sfondo), potete assegnare il colore fra i sedici disponibili visti precedentemente sia al carattere sia al suo sfondo.

Nell'utilizzo standard tutti i caratteri hanno colore nero (codice 2) per il FOREGROUND e il colore trasparente (codice 1) per il BACKGROUND.

È importante tener presente che ciascun carattere standard ha il proprio codice ASCII, vedi APPENDICE C, e tutti i caratteri, anche quelli definiti dall'utente con CALL CHAR, sono suddivisi in 16 set ciascun di 8 caratteri:

Set	Codici caratteri	Caratteri Ascii
1	32-39	Spazio -'
2	40-47	(- /
3	48-55	Ø - 7
4	56-63	8 - ?
5	64-71	C - G
6	72-79	H - O
7	80-87	P - W
8	88-95	X - =
9	96-103	` - g
10	104-111	h - o
11	112-119	p - w
12	120-127	x - DEL
13	128-135	Definiti dall'utente

14	136-143	Definiti dall'utente
15	144-151	Definiti dall'utente
16	152-159	Definiti dall'utente

ESEMPI

```
NUM 10
    10 FOR U=1 TO 14
    20 CALL COLOR (U,3,2)
    30 NEXT U
    40 CALL SCREEN (2)
    50 PRINT "GIOVANNI"
    60 GO TO 10
RUN
```

Questo esempio realizza il campo inverso, schermo nero e caratteri verdi. Infatti il ciclo permette di assegnare i 12 set dei caratteri ASCII più 2 set di caratteri definiti dall'utente.

Cambiando il codice del colore potete ottenere tutte le combinazioni di colori fra caratteri e schermo.

```
NUM 100 REM ASSEGNAZIONE COLORI
    110 PRINT "SCEGLI I COLORI DESIDERATI":::
    120 INPUT "COLORE SCHERMO?" :F
    130 INPUT "COLORE CARATTERE?" :C
    140 INPUT "COLORE SFONDO?" :V
    150 CALL CLEAR
    160 CALL SCREEN (F)
    170 CALL COLOR (5,C,V)
    180 CALL HCHAR (10,10,65,30)
    190 GO TO 110
RUN
```

Questo esempio permette di scegliere i colori dei caratteri e dello schermo come desiderate in quanto i codici dei colori sono assegnati con INPUT e posti nelle variabili F,C,V che vanno a definire i parametri dei sottoprogrammi CALL, SCREEN, CALL COLOR.

Alla linea 180 il CALL HCHAR permette di ripetere 30 volte in orizzontale il carattere A (cod. ASCII 65, set n 5) a partire da riga 10 e colonna 10.

2.7 SOTTOPROGRAMMI PER I SUONI

Il sottoprogramma che permette di ascoltare suoni di note e accordi è CALL SOUND.

1) CALL SOUND (Durata, Frequenza, Volume)

2) CALL SOUND (Durata, FR1, VOL1, FR2, VOL2, FR3, VOL3).

Premesso che i campi di variabilità dei parametri: Durata, Frequenza, Volume sono i seguenti:

D	1 - 4250	−1 −4250
F	110 - 44733	−1 −8
	(per le note)	(per i rumori)
V	0	30 (massimo)

È possibile suonare delle note (vedi APPENDICE C per la tabella delle note nelle rispettive scale) utilizzando il formato 1; e suonare degli accordi con il formato 2 in cui va espressa solo una volta la durata e massimo 3 note (frequenze) e altrettanti volumi.

La durata negativa permette di ascoltare velocemente le note.

ESEMPIO:

```
NUM  10
      10 REM MACCHINA IN PARTENZA E ARRIVO
      20 CALL CLEAR
      30 FOR N=1 TO 8
      40 CALL SOUND(60,220,8,-5,0)
      50 CALL SOUND(60,220,8,-5,5)
      60 NEXT N
      70 CALL SOUND(80,220,8,-5,0)
      80 FOR F=1000 TO 5000 STEP 20
      90 CALL SOUND(-99,111,30,111,30,F,30,-8,0)
     100 NEXT F
     110 FOR F=4000 TO 8000 STEP -50
     120 CALL SOUND(-99,111,30,111,30,F,30,-8,0)
     130 NEXT F
     140 GOTO 20
RUN
```

Questo programma permette di ascoltare la sequenza di rumori di una macchina in partenza nonché il rallentare dei giri del motore e chiusura della chiave d'accensione.

Potete cambiare i parametri e ottenere una partenza più veloce o rallentare la corsa.

2.8 SOTTOPROGRAMMI PER LA GRAFICA E GIOCHI

CALL CHAR (COD, "PATTERN")	definisce i caratteri
CALL HCHAR (R,C,COD,RIP)	ripete i caratteri in orizzontale
CALL VCHAR (R,C,COD,RIP)	ripete i caratteri in verticale
CALL GCHAR (R,C,V)	ritorna il codice del carattere
CALL KEY (U,RIT,STATO)	controlla la tastiera
CALL JOYST (U,X,Y)	assegna le posizioni ai joystic

– CALL CHAR (COD, "PATTERN")

Questo sottoprogramma permette di creare caratteri propri indicando il codice numerico (COD) che è un numero variante da 32 a 159, cioè potete utilizzare sia i set dei codici ASCII da 1 a 12, sia i set 13,14,15,16 disponibili per i caratteri propri.

Per illustrare la "PATTERN" dobbiamo fare una digressione sul codice esadecimale binario:

Codice esadecimale	Simboli
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Il codice è esadecimale perché utilizza 16 simboli da 0 ad F (le lettere da A ad F rappresentano i numeri 10-11-12-13-14-15), è binario perchè utilizza solo le cifre 1 e 0.

Come nell'aritmetica decimale così anche in quella binaria ciascuna cifra in funzione della sua posizione ha un determinato peso;

F = 1111

questo è ottenuto da:

$2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1 = 15$

8 4 2 1

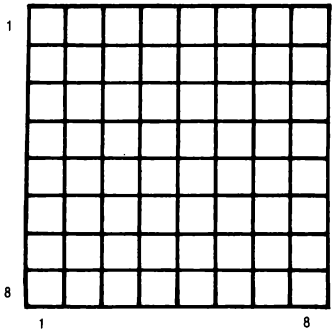
Potete ricavarvi qualsiasi codice ricordando questa regola:

1 in quarta posizione vale 8
1 in terza posizione vale 4
1 in seconda posizione vale 2
1 in prima posizione vale 1

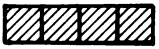
ESEMPIO:

9 = 1001
A = 1010
3 = 0011
8 = 1000

Premesso ciò, sapendo che ciascun carattere è costituito da una matrice 8×8 punti (64) vediamo com'è possibile rappresentare con il codice esadecimale la "PATTERN" di un carattere.

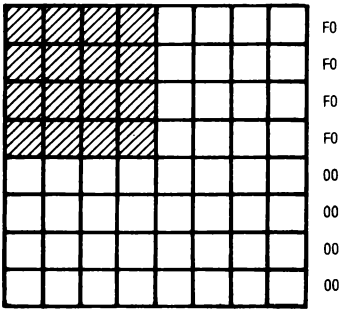


Gli zero sono rappresentati da caselle bianche, gli uno da caselle annerite.

1111 = F 

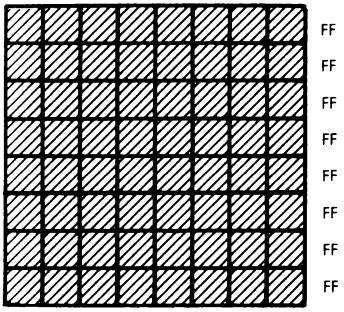
0000 = 0 

ESEMPI: Definizione di 1/4 di quadratino



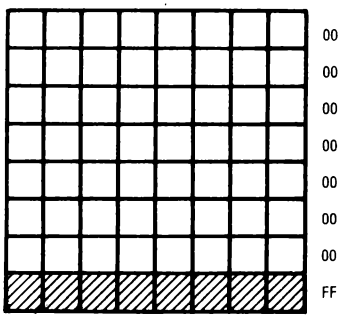

```
10 P$ = "F0F0F0F000000000'
20 CALL CHAR (129,P$)
```

Definizione di un quadratino



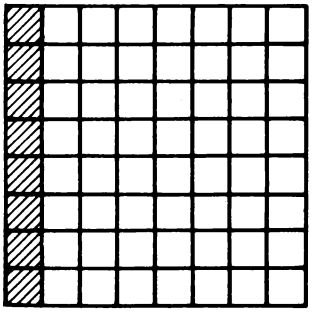

```
10 P$ = "FFFFFFFFFFFFFFFF"
20 CALL CHAR (130,P$)
```

Definizione di una linea orizzontale




```
10 P$ = "00000000000000FF"
20 CALL CHAR (131,P$)
```

Definizione di una linea verticale



```
10 P$ = "8080808080808080"
20 CALL CHAR (132,P$)
```

Potete con il CALL CHAR definirvi un alfabeto greco, figure per giochi ecc., per avere automaticamente le Pattern dei caratteri utilizzate l'esercizio 5.4.

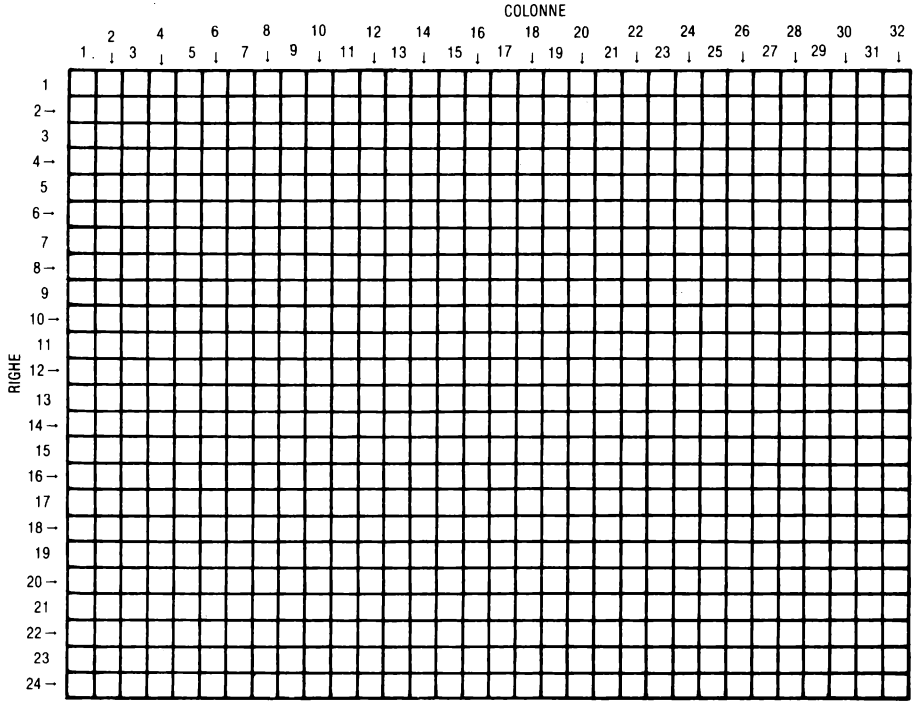
Per poter vedere il carattere sullo schermo nella posizione da voi indicata dovete utilizzare i sottoprogrammi:

CALL VCHAR (Riga, Colonna, COD, Ripetizioni)
CALL HCHAR (Riga, Colonna, COD, Ripetizioni)

– CALL HCHAR (Riga, Colonna, COD, Rip)
permette di ripetere il carattere in ORIZZONTALE il numero di volte indicato in Rip a partire da riga e colonna indicate.

– CALL VCHAR (Riga, Colonna, COD, Rip)
permette di ripetere il carattere in VERTICALE il numero di volte indicato in Rip a partire da riga e colonna indicate.

Lo schermo è suddiviso in 32 colonne x 24 righe:



per cui il totale dei caratteri rappresentabili sullo schermo è 24 x 32 = 768.

Se il numero di ripetizioni non è assegnato, il computer per default assegna 1.

Se utilizzate nella definizione di caratteri i codici 32-127 ASCII, effettuando il Breakpoint o finendo l'esecuzione, il computer ritorna i caratteri ASCII corrispondenti.

Nella definizione della PATTERN se utilizzate meno di sedici caratteri, il computer assegna degli zero, i caratteri in più vengono eliminati.

ESEMPI:

```
10 CALL CLEAR
20 P$ = "FF"
30 CALL CHAR (129,P$)
40 CALL HCHAR (1,1,129,28)
50 FOR I= 1 TO 500
60 NEXT I
70 END
```

Nell'esempio illustrato è tracciata una linea di 28 caratteri a riga e colonna 1.

```
10 CALL CLEAR
20 P$ = "FFFFFFFFFFFFFFFF"
30 CALL CHAR (130,P$)
40 CALL VCHAR (10,1,130,56)
50 FOR I=1 TO 500
60 NEXT I
70 END
```

In questo esempio è visualizzato 56 volte il carattere ■ definito a linea 30.

Utilizzando i codici dei caratteri Ascii (32-127) potete ripetere in verticale e in orizzontale tutti i caratteri presenti nella tabella ASCII.

— **CALL GCHAR (Riga, Colonna, Variabile numerica)**

permette di conoscere qual'è il codice ASCII del carattere rappresentato a riga e colonna indicate.

— **CALL KEY (U,Rit, STATO)**

permette di controllare la tastiera.

Premesso che ci sono 6 configurazioni possibili della tastiera, U varia da 0 a 5, per la configurazione standard U = 0.

Rit rappresenta la variabile numerica di ritorno, ossia contiene il codice numerico del carattere corrispondente al tasto premuto.

STATO rappresenta la variabile numerica con valori (1, −1,0) indicanti lo stato della tastiera, esso indica se un tasto è stato premuto o no.

ESEMPIO:

```
10 CALL CLEAR
20 PRINT "PREMI UN TASTO":::
30 CALL KEY (0,R,ST)
40 IF ST = 0 THEN 30
50 PRINT "NOME E COGNOME"
60 END
```

in questo programma l'esecuzione è fermata dalla linea 30 e continua quando premete un tasto.

ESEMPIO:

```
10 CALL CLEAR
20 PRINT "PREMI A":::
30 CALL KEY (0,R,ST)
40 IF R <> 65 THEN 30
50 PRINT "NOME E COGNOME"
60 END
```

in questo programma l'esecuzione è fermata dalla linea 30 e continua quando premete il tasto A che ha codice ASCII 65.

ESEMPIO

Questo programma permette di trasformare la tastiera in una pianola, associa ad alcuni tasti le note della scala Do.

La linea 150 controlla la tastiera, la variabile AA contiene il codice ASCII 65,66,67,68,69,70,71 rispettivamente dei tasti A,B,C,D,E,F,G quando sono battuti. La linea 180 permette di non far interrompere il programma qualora battete un tasto errato. La linea 170 seleziona la nota da suonare insieme alla linea 190 e 330 suona la nota con durata 100 e volume 2.

Potete interrompere il programma con il FCTN-CLEAR e assegnare altre note ad altri tasti.

```
100 CALL CLEAR
110 PRINT "PUOI SUONARE USANDO I TASTI":::::::
120 PRINT " A   B   C   D   E   F   G "::
130 PRINT "LA  SI  DO  RE  MI  FA  SOL":::
140 PRINT " Per terminare batti ";" un numero"
150 CALL KEY(0,AA,SS)
160 IF SS=0 THEN 150
170 NOTE=AA-64
180 IF NOTE)7 THEN 150
```

```
190 ON NOTE GOTO 300,320,200,220,240,260,280
200 NOTE=262
210 GOTO 330
220 NOTE=294
230 GOTO 330
240 NOTE=330
250 GOTO 330
260 NOTE=349
270 GOTO 330
280 NOTE=392
290 GOTO 330
300 NOTE=440
310 GOTO 330
320 NOTE=494
330 CALL SOUND(100,NOTE,2)
340 GOTO 150
350 END
```

ESEMPIO

Questo programma permette la possibilità di sperimentare i sottoprogrammi per la costruzione di figure o disegni colorati. Il ciclo 990-1020 realizza la costruzione dei caratteri che costituiscono la rappresentazione di una porta. Il COLOR assegna i colori ai caratteri. Il VCHAR ripete i caratteri definiti col DATA.

```
890 REM  ** PORTA **
892 DIM CT$(13)
900 DATA FF80BFA0A0A7A4A4,FF00FF0000FF
910 DATA FF01FD0505E52525,A4A4A4A4A4A4A4A4
920 DATA 2525252525252525,A4A4A4A4A7A0A0AC
930 DATA 00000000FF,25252525E5050505
940 DATA ACA0A0A7A4A4A4A4,000000FF
950 DATA 050505E525252525,A4A4A4A4A7A0A0FF
960 DATA 00000000FF0000FF,25252525E50505FF
970 M=130
980 RESTORE 900
990 FOR S=0 TO 13
1000 READ CT$(S)
1010 CALL CHAR(M+S,CT$(S))
1020 NEXT S
1030 CALL COLOR(13,2,9)
1040 CALL COLOR(14,2,9)
1050 CALL COLOR(15,2,9)
1060 CALL COLOR(16,2,9)
1070 R=6
1080 C=13
1090 CALL VCHAR(R,C,M)
1100 CALL VCHAR(R+1,C,M+3)
1110 CALL VCHAR(R+2,C,M+5)
1120 CALL VCHAR(R+3,C,M+8)
1130 CALL VCHAR(R+4,C,M+3,4)
1140 CALL VCHAR(R+8,C,M+11)
1150 CALL VCHAR(R,C+1,M+1)
1160 CALL VCHAR(R+2,C+1,M+6)
1170 CALL VCHAR(R+3,C+1,M+9)
1180 CALL VCHAR(R+8,C+1,M+12)
1190 CALL VCHAR(R,C+2,M+1)
1200 CALL VCHAR(R+2,C+2,M+6)
1210 CALL VCHAR(R+3,C+2,M+9)
1220 CALL VCHAR(R+8,C+2,M+12)
1230 CALL VCHAR(R,C+3,M+2)
1240 CALL VCHAR(R+1,C+3,M+4)
1250 CALL VCHAR(R+2,C+3,M+7)
1260 CALL VCHAR(R+3,C+3,M+10)
1270 CALL VCHAR(R+4,C+3,M+4,4)
1280 CALL VCHAR(R+8,C+3,M+13)
```

```
1290 CALL COLOR(13,2,1)
1300 CALL COLOR(14,2,1)
1310 CALL COLOR(15,2,1)
1320 CALL COLOR(16,2,1)
```

ESEMPIO

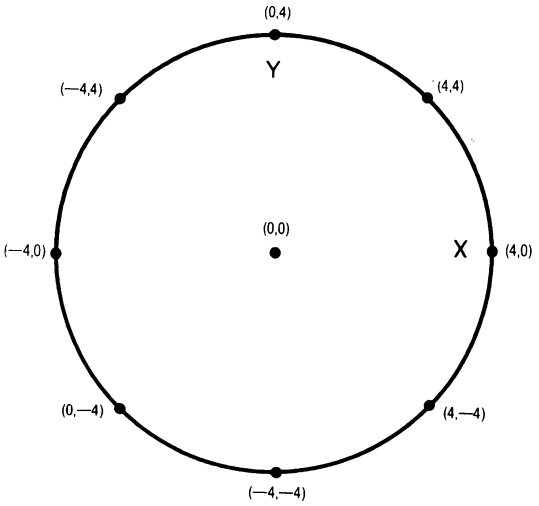
Questo programma realizza il modello di una bolla. Le linee 90-190 permettono l'inserimento delle voci della bolla. Le linee 210-380 assegnano il colore ed il formato alla bolla.

Interrompendo il programma o finendo l'esecuzione, sullo schermo appaiono i caratteri ASCII dei codici utilizzati col VCHAR e HCHAR.

```
10 CALL CLEAR
20 CALL SCREEN(10)
30 PRINT "    GENERAZIONE   BOLLA
40 FOR Z=1 TO 500
50 NEXT Z
60 REM  CREAZIONE BOLLA DI SCARICO
70 CALL CLEAR
80 CALL SCREEN(4)
90 INPUT " DATA ":X$
100 PRINT ::
110 INPUT " BOLLA N ":A$
120 PRINT ::
130 INPUT " ARTICOLO N ":B$
140 PRINT ::
150 INPUT " DESCRIZIONE:":C$
160 PRINT ::
170 INPUT " QUANTITA':":D$
180 PRINT ::
190 INPUT " PREZZO  L ":E$
200 PRINT ::
210 CALL COLOR(2,16,5)
220 CALL HCHAR(2,3,42,28)
230 CALL VCHAR(2,3,42,21)
240 CALL HCHAR(23,3,42,28)
250 CALL VCHAR(2,30,42,21)
260 CALL HCHAR(3,10,83,1)
270 CALL HCHAR(3,10,83,1)
280 CALL HCHAR(3,12,67,1)
290 CALL HCHAR(3,14,65,1)
300 CALL HCHAR(3,16,82,1)
310 CALL HCHAR(3,18,73,1)
320 CALL HCHAR(3,20,67,1)
330 CALL HCHAR(3,22,79,1)
340 CALL HCHAR(20,23,70,1)
350 CALL HCHAR(20,24,73,1)
360 CALL HCHAR(20,25,82,1)
370 CALL HCHAR(20,26,77,1)
380 CALL HCHAR(20,27,65,1)
390 FOR I=1 TO 1000
400 NEXT I
410 CALL CLEAR
420 L$="S"
430 INPUT "NUOVA BOLLA ? S/N  ":P$
440 IF P$=L$ THEN 60
450 END
```

– CALL JOYST(U,X,Y)

Questo sottoprogramma permette di assegnare le posizioni ai Joystick al variare di X e Y nel modo seguente:



U = 1 Joystick 1
U = 2 Joystick 2

Per l'utilizzo del tasto del fuoco, dovete utilizzare:
CALL KEY (O,R,ST)
dove R assumerà il VALORE 18 quando premete il tasto del fuoco.

Vedete gli esercizi 5.5,5.6 per delle applicazioni complete.

Grafica ad alta risoluzione

Finora abbiamo trattato l'indirizzamento di 24 x 32 = 768 punti dello schermo con le funzioni VCHAR, HCHAR, GCHAR; nell'ambito del carattere 8 x 8 = 64 punti con l'istruzione CALL CHAR, abbiamo analizzato il modo di indirizzare i singoli pixel, avendo anche la possibilità di definire delle figure proprie o caratteri speciali con le "PATTERN" in codice esadecimale. L'indirizzabilità di 32 x 8 x 24 x 8 = 49152 punti, però risulta molto laboriosa e lenta per la realizzazione di grafici e curve con le istruzioni residenti nell'interprete BASIC, per cui l'unico modo per risolvere il problema è quello di ricorrere all'Assembler del TMS 9900 e ciò richiede una delle due configurazioni Hardware:

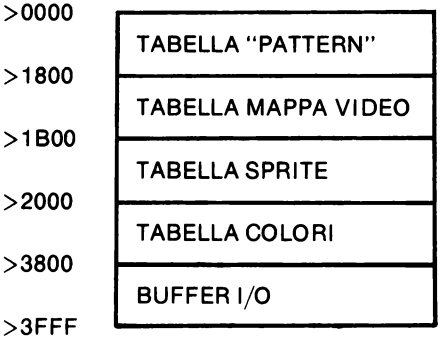
- MODULO SSS Miny Memory provvisto di Assemblatore line by line
- MODULO EDITOR/ASSEMBLER, dischi con le utility, Unità a dischi, espansione 32K RAM

Disponendo di una delle due configurazioni, dovete predisporre la VDP RAM (16K) per gestire la grafica ad alta risoluzione, ossia predisporla a contenere tutte le

informazioni utili per gestire i 49152 pixel e i 16 colori da utilizzare. Per questo esistono i VDP WRITE ONLY REGISTER che consentono la predisposizione della VDP RAM, nel nostro caso dovete porre il bit 6 del R0 ad 1 e i bit 3 e 4 del R1 a zero.

Il microprocessore grafico TMS 9918A che gestisce il video, legge nella tabella Mappa del Video il nome del carattere da visualizzare in un punto dello schermo, poi la "PATTERN" che lo definisce, e infine i colori da utilizzare.

Questo modo di operare è detto Bit Map e la configurazione dei 16K RAM di VDP è la seguente:



VDP RAM IN BIT MAP MODE

In essa la tabella delle "PATTERN" contiene le forme dei caratteri, occupa 6K byte, in quanto ciascuna informazione occupa 8byte e la tabella è suddivisa in tre sezioni da 256 informazioni. Questa tabella è alloccata all'indirizzo >0000 o all'indirizzo >2000.

La tabella Mappa del Video contiene i nomi dei caratteri, occupa 0.8K byte, in quanto ciascuna informazione occupa 1 byte e la tabella è suddivisa in tre sezioni da 256 informazioni. Questa tabella è alloccata all'indirizzo > 1800.

Premesso che gli Sprite sono le figure in movimento, a cui si può assegnare un colore fra i sedici, la velocità e la direzione volute, nonché sovrapporli e movimentarli contemporaneamente sul video, agli Sprite é dedicata la tabella alloccata all'indirizzo >1B00 (Per ulteriori informazioni sugli Sprite leggete il paragrafo 3.6).

La tabella Colori contiene i colori con i quali saranno visualizzati i caratteri o pixel, occupa 6K byte, in quanto ciascuna informazione occupa 8 byte e la tabella è suddivisa in tre sezioni da 256 informazioni. Questa tabella è alloccata all'indirizzo >2000 o >0000 in quanto come area di memoria è scambiabile con la tabella delle "PATTERN" con il registro R4 WRITE ONLY. Volendo quindi rappresentare un carattere esso deve essere descritto nelle sue caratteristiche nelle tre tabelle che occupano complessivamente 13K byte di VDP RAM. L'ultima porzione di memoria all'indirizzo >3800 è utilizzata come memoria di transito per le periferiche (informazioni di I/O). Per ulteriori informazioni sulle modalità di comunicazione fra CPU e VDP leggete il cap. 4 e le Appendici B e F.

2.9 FUNZIONI DI STRINGA

Nell'analisi dei testi e delle stringhe in generale, molto utili sono le funzioni che permettono di analizzare e manipolare i caratteri alfanumerici. Le funzioni preposte sono:

ASC("STRINGA")

ritorna il codice ASCII di un carattere.

ESEMPIO

```
10  PRINT  ASC("X")        stampa 88 (cod. ASCII di x)
10  PRINT  ASC("CAR")      stampa 67 (cod. ASCII di C)
```

CHR$ (NUMERO)

ritorna il carattere del codice ASCII corrispondente o il carattere definito dall'utente.

ESEMPIO

```
10  PRINT CHR$(75)              stampa il carattere K
20  CALL CHAR(129,"FF")
30  PRINT CHR$(129)             stampa il carattere definito alla linea 20
```

LEN("STRINGA")

ritorna la lunghezza della stringa, nella lunghezza vengono contati anche i caratteri bianchi.

ESEMPIO

```
10  PRINT LEN("LIBRO")          stampa il valore 5
```

POS("STRINGA 1", "STRINGA 2", n)

ritorna la posizione in cui si incontra la stringa 2 nella stringa 1 la prima volta o l'ennesima volta

ESEMPIO

```
10  PRINT POS("CASA", "A", 2)   stampa il valore 4 perchè la posizione
                                della A incontrata la seconda volta è 4.
```

SEG$("STRINGA", posizione partenza, lunghezza)

effettua la segmentazione della stringa indicata a partire dalla posizione assegnata e lunghezza desiderata.

ESEMPIO

```
10  PRINT SEG$("BUON GIORNO",6,6)
```

stampa la parola GIORNO, cioè 6 caratteri a partire dalla 6ª posizione.

STR$(espressione numerica)

permette di considerare i numeri come stringhe che perdono tutte le regole numeriche di calcolo e per i quali valgono le regole delle stringhe.

VAL("STRINGA")

è l'inverso della funzione STR$, in quanto permette di assegnare il valore alla stringa.

ESEMPIO

```
10  PRINT STR$(VAL("1511.10")
```

stampa la stringa 1511.10

Nota esplicativa:

ASC	Codice ASCII
CHR$	Carattere
LEN	Lunghezza
POS	Posizione
SEG$	Segmentazione della stringa
STR$	Stringa
VAL	Valore

2.10 FUNZIONI ARITMETICHE

ABS(espressione numerica)	**calcola il valore assoluto di un numero**
ATN(X)	**calcola l'arcotangente di X**
COS(X)	**calcola il coseno di un angolo X espresso in radianti**
SIN(X)	**calcola il seno di un angolo X espresso in radianti**
TAN(X)	**calcola la tangente di un angolo X espresso in radianti**
INT(espressione numerica)	**calcola la parte intera di un numero**
LOG(espressione numerica)	**calcola il logaritmo naturale**
EXP(espressione numerica)	**calcola l'esponenziale e^X**
SGN(espressione numerica)	**calcola il segno algebrico del numero**
SQR(espressione numerica)	**calcola la radice quadrata del numero**
RND	**è la funzione che permette di generare numeri casuali compresi fra 0 e 1**
RANDOMIZE	**è la funzione che genera sequenze diverse di numeri; è utilizzata in coppia con RND**

Essendo i numeri generati dalla funzione RND dei numeri decimali, per poter avere dei numeri interi dovete applicare la funzione INT.

```
INT(RND)
```

Volendo inserire l'intervallo di generazione dei numeri, dovete moltiplicare RND per il numero desiderato:

INT(10 * RND)	genera numeri compresi fra 0 e 9
INT(10 * RND)+1	genera numeri compresi fra 1 e 10

Volendo generare numeri compresi in un intervallo (A,B) dovete applicare la seguente formula:

```
INT((B–A+1)*RND)+A
```

ESEMPIO:

```
10  FOR I=1 TO 10
20  PRINT INT(10*RND)+1;
```

```
30  NEXT I
40  END
```

Questo programma stampa 10 numeri casuali generati da RND; ogni volta che eseguite il programma la sequenza dei numeri generati rimane inalterata, se volete cambiare la sequenza dovete inserire l'istruzione RANDOMIZE. Nell'esempio precedente inserite la linea:

```
5  RANDOMIZE
```

ora il programma genererà sequenze di numeri sempre diversi.

Molto utile è l'istruzione DEF che permette all'utente di definirsi tutte le funzioni che ritiene utili e che può richiamare in più punti del programma:

DEF nome [parametro] = espressione numerica
DEF nome [parametro] = espressione di stringa

DEF permette di definire sia funzioni numeriche sia funzioni di stringa, il parametro rappresenta la variabile locale della funzione definita, per utilizzare la funzione dovete richiamarla con il nome assegnato e il valore viene trasferito e aggiornato tramite il parametro.

Il numero di linea di DEF deve essere più piccolo del numero di linea dell'istruzione che deve utilizzare la funzione.

Ecco alcuni esempi utili di funzioni definite:

– Generazione di chiave segreta

```
10 CALL CLEAR
20 DEF CHIAVE$(X)=SEG$(NOME$,X,5)
30 INPUT "NOME O CODICE  ":NOME$
40 INPUT "POSIZIONE X= ":X
50 FOR I=1 TO 5
60 K$(I)=CHIAVE$(X)&STR$(I)
70 PRINT ::
80 PRINT K$(I)
90 NEXT I
100 END
```

– Funzione derivata

```
90 DEF F(Y)=Y*2
100 DEF DERIVATA(X)=(F(X+H)-F(X-H))/2*H
110 INPUT "PASSO H= ":H
120 INPUT "ASCISSA X= ":X
130 PRINT "F'(";STR$(X);")=";DERIVATA(X)
```

– Funzione PI=3.14

DEF PI=4*ATN(1)

per passare da gradi a radianti moltiplicate i gradi per PI/180.

– Programma per la conversione del numero decimale in binario:

```
10 CALL CLEAR
20 INPUT "NUMERO  ":N
30 PRINT :::N;" In binario e'= "::::
40 FOR I=15 TO 0 STEP -1
50 B=INT(N/2^I)
60 N=N-B*2^I
70 A$=STR$(B)
80 PRINT A$;
90 NEXT I
100 FOR I=1 TO 3000
110 NEXT I
120 GOTO 10
```

– Programma per il calcolo del logaritmo di un numero:

```
10 CALL CLEAR
20 INPUT "BASE  ":B
30 IF B(=0 THEN 20
40 INPUT "NUMERO  ":N
50 IF N(=0 THEN 40
60 X=LOG(N)/LOG(B)
70 CALL CLEAR
80 PRINT "LOGARITMO IN BASE ";B;" DI";N;" E'";X:::::
90 END
```

Funzioni logiche AND e OR

Essendo AND la funzione logica che corrisponde al prodotto potete ottenere essa

con * ES: IF (A>10) * (B<5) THEN 200.

Essendo OR la funzione logica che corrisponde alla somma potete ottenere essa

con + ES: (A<3) + (A>10) THEN 300.

Routine PRINT AT

```
60 CALL CLEAR
70 INPUT "Riga,Colonna,Parola  : ":RIGA,COL,PAR$
80 GOSUB 100
90 END
100 REM  **DISPLAY AT**
110 Z2=RIGA
120 Z3=COL-1
130 FOR Z1=1 TO LEN(PAR$)
140 Z3=Z3+1
150 IF Z3<=32 THEN 180
160 Z2=Z2+1
170 Z3=1
180 CALL HCHAR(Z2,Z3,ASC(SEG$(PAR$,Z1,1)))
190 NEXT Z1
200 RETURN
```

Riga, colonna e parola da visualizzare vanno separate da virgole.

Routine ACCEPT AT

```
1960 CALL CLEAR
1970 INPUT "RIGACOLONNATIPOLUNGHEZZA ":PAR$
1980 GOSUB 2010
1990 END
2000 REM  *** ROUTINE ACCEPT AT ***
2010 RIGA=VAL(SEG$(PAR$,1,2))
2020 COL=VAL(SEG$(PAR$,3,2))
2030 TIPO$=SEG$(PAR$,5,1)
2040 LUNG=VAL(SEG$(PAR$,6,2))
2050 K=COL
2060 CALL HCHAR(RIGA,COL,95)
2070 CALL KEY(0,TASTO,ST)
2080 IF ST=0 THEN 2070
2090 IF TASTO=13 THEN 2210
2100 IF TASTO=8 THEN 2170
2110 IF TIPO$="A" THEN 2130
2120 IF (TASTO<45)+(TASTO)58)+(TASTO=47)THEN 2070
2130 CALL HCHAR(RIGA,COL,TASTO)
2140 COL=COL+1
2150 IF COL-K)LUNG THEN 2170
2160 GOTO 2060
2170 CALL HCHAR(RIGA,COL-1,32)
2180 CALL SOUND(15,440,5)
2190 COL=COL-1
2200 GOTO 2060
2210 CALL HCHAR(RIGA,COL,32)
2220 FOR COLON=K TO COL-1
2230 CALL GCHAR(RIGA,COLON,COD)
2240 PAR$=PAR$&CHR$(COD)
2250 NEXT COLON
2260 PAR$=SEG$(PAR$,8,28)
2270 IF TIPO$="A" THEN 2290
2280 NUMERO=VAL(PAR$)
2290 RETURN
```

TIPO A = alfanumerico / N = numerico ; cod. 13 = tasto ENTER ; cod. 8 = tasti FCTNS ; cod. 95 = tasto – ; cod. 32 = spazio
PAR$ contiene valori alfanumerici e NUMERO valori numerici

Es Input: 0203A05/1312N11

2.11 GESTIONE DEI FILE

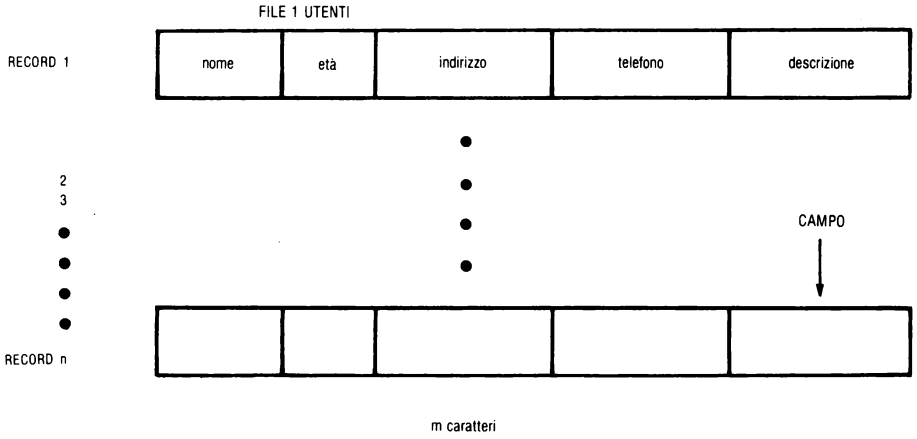
Pensate di avere un archivio di nomi e di informazioni relative a ciascun nome; per poterlo gestire esso deve essere identificato dal nome, ordinato, deve consentire la ricerca di un nome, deve poter essere aggiornato; ciascun elemento o scheda deve occupare un certo spazio, può essere memorizzato su diversi supporti fisici.

Così dualmente quando col computer volete creare un FILE(flusso di dati) per conservare delle informazioni numeriche o alfanumeriche, dovete definire tutte le caratteristiche del FILE e il dispositivo dove voler accedere.

Un FILE è costituito da un insieme di RECORD, ciascun RECORD è suddiviso in CAMPI numerici e alfanumerici.

Il FILE sarà identificato dal nome e numero, sarà costituito da un RECORD, ciascun RECORD sarà lungo m caratteri.

ESEMPIO:



RECORD 1
2
3
.
.
.
.
RECORD n

In questo caso abbiamo il FILE 1 UTENTI costituito da n RECORD e ciascun RECORD costituito da 5 CAMPI per totali m caratteri; ciascun RECORD conterrà le in-

formazioni rispettivamente del sig. Rossi, del sig. Esposito, del sig. Vanga ecc., questa organizzazione permetterà di accedere all'archivio UTENTI per poter conoscere le informazioni memorizzate.

Vediamo come creare e gestire un FILE in linguaggio Basic:

```
OPEN # n: "DSK1.nome",OPZIONI
OPEN # n: "CS1",OPZIONI
```

dove n è un numero compreso fra 1 e 255 e le opzioni descrivono il file; OPEN # n è l'istruzione che crea il file n o apre il file n sul supporto o dispositivo espresso fra gli apici.

```
PRINT # n [,REC p]:lista variabili
```

dove n è compreso fra 1 e 255, REC p rappresenta il RECORD p, la lista variabili contiene variabili numeriche, alfanumeriche e array; PRINT # n è l'istruzione che permette di scrivere le informazioni nel file n.

```
INPUT # n [,REC p]:lista variabili
```

dove n è compreso fra 1 e 255, REC p rappresenta il RECORD p, la lista variabili contiene variabili numeriche, alfanumeriche, array, INPUT# n è l'istruzione che permette di leggere le informazioni dal file n.

Quando avete effettuate tutte le operazioni di lettura/scrittura nel file n esso va sempre chiuso, l'istruzione preposta è:

```
CLOSE # n
```

I comandi BYE e NEW permettono di chiudere automaticamente tutti i file aperti. Non utilizzate mai FCTN(QUIT) perchè potete perdere le informazioni in transizione dei file e non effettuate la chiusura dei file.

Vediamo la struttura tipica di una sequenza di scrittura e lettura:

```
OPEN#1 : "DSK1.UTENTI",OPZIONI
PRINT #1 : A,B,C$
CLOSE#1
```

sequenza di scrittura nel file 1 di nome UTENTI, utilizzata l'unità a dischi 1.

```
OPEN # 1 :"DSK1.UTENTI",OPZIONI
INPUT#1 :A,B,C$
CLOSE#1
```

sequenza di lettura dal file 1 di nome UTENTI, utilizzata l'unità a dischi 1.

Vediamo tutte le caratteristiche dei file che potete creare su unità a nastro o disco e la descrizione delle OPZIONI consentite.

Numero del file e nome

Il numero può variare da # 1 a # 255
Il nome dipende dal dispositivo dove volete aprire il file.

Le comunicazioni fra unità centrale e periferiche (dischi, registratori, stampanti RS232, modem, sintetizzatore vocale) avvengono tramite apertura di un canale di trasmissione, dotato di Buffer di I/O (registro di transito); è possibile aprire da 1 a 255 canali, per ogni file associato al # n il computer assegna un'area di transito che consentirà tutte le operazioni di INPUT/OUTPUT. Ciascun file aperto impegna 518 byte.

Organizzazione ed accesso

SEQUENTIAL oppure RELATIVE (solo per l'unità dischi).

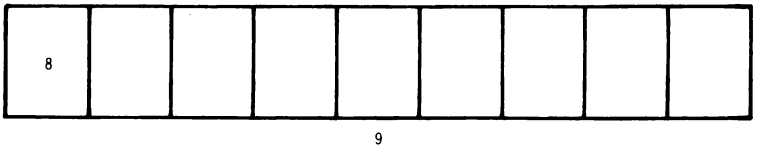
Un File organizzato sequenzialmente è quello in cui le informazioni sono memorizzate una dopo l'altra, per cui l'accesso alla ennesima informazione richiede lo scorrimento delle n-1 informazioni precedenti. Il caso tipico di supporto che consente solo file sequenziali è il nastro, infatti per accedere all'informazione all'ennesimo giro devono essere svolti i primi n-1 giri.

Un file organizzato in modo RELATIVE o RANDOM è quello in cui le informazioni sono memorizzate in modo casuale mediante puntatori, per cui l'accesso all'ennesima informazione richiede un solo passo senza alcuna differenza fra la prima e l'ultima informazione. Il file RELATIVE può essere creato solo su disco, che consente anche file SEQUENTIAL.

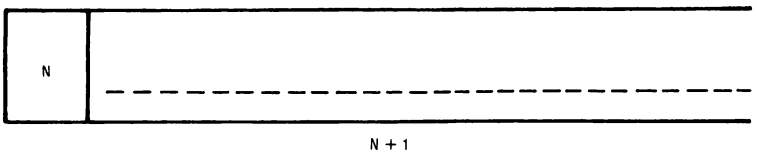
Tipo di file

INTERNAL oppure DISPLAY

Il file di tipo INTERNAL memorizza le informazioni nel formato interno-macchina per cui le variabili numeriche occupano sempre 9 byte, uno di controllo e otto utili per il dato:

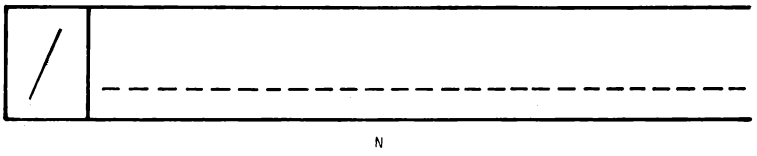


le variabili alfanumeriche occupano N+1 byte, uno di controllo e N byte quanti sono i caratteri espressi:



la lunghezza massima della stringa è di 112 caratteri. Il primo byte a sinistra di controllo contiene la lunghezza del dato.

Il file di tipo DISPLAY memorizza i dati così come sono rappresentati sul video, per cui non c'è differenza fra dati numerici e alfanumerici, essi vanno sempre separati dalla virgola. I dati numerici impegnano un carattere in più per il segno.



la lunghezza N complessiva è data dal numero di caratteri espressi e dalle virgole.

Il tipo DISPLAY è più lento del tipo INTERNAL, in quanto richiede una doppia conversione DIS/INT - INT/DIS.

Tempo di scrittura su nastro in relazione al tipo e alla lunghezza del RECORD.

Tipo	Num. caratteri	Secondi
	1	6
DISPLAY (64)	64	6
	65	13
	1	7
DISPLAY (128)	128	7
	129	15
	1	8
DISPLAY (192)	192	8
	193	17
	1	6
INTERNAL (64)	64	6
	65	ERRORE
	1	7
INTERNAL (128)	128	7
	129	ERRORE
	1	8
INTERNAL (192)	192	8
	193	ERRORE

Questa tabella evidenzia che non c'è molta differenza di tempo fra RECORD lunghi 64 caratteri e quelli lunghi 192 caratteri.

Un nastro C-60, 30 minuti di registrazione, può contenere fino a 43K byte di informazioni.

Formato del record

FIXED [m] oppure VARIABILE [m]

Il formato FIXED del RECORD indica che esso deve necessariamente essere di lunghezza fissa, la lunghezza prevista per i file sequenziali su nastro è 64, 128, 192 caratteri.

Per default il computer assegna la lunghezza 64, se specificate [m] il computer

riempirà ogni RECORD con gli zeri per raggiungere la lunghezza m. Ogni dispositivo assegna al RECORD una propria lunghezza massima, così l'unità a dischi ammette RECORD fino a 255 caratteri.

Il formato VARIABILE consente di avere dei RECORD a lunghezza variabile, ma diminuisce la velocità rispetto al formato FIXED.

Su disco l'assegnazione automatica della lunghezza del RECORD è 80 byte.

Potete indicare anche il numero di RECORD costituenti il File, questo va inserito dopo RELATIVE.

Mentre per i file SEQUENTIAL sono previsti sia RECORD fissi che variabili, per i file RELATIVE sono previsti solo RECORD fissi.

Modo di apertura del file

INPUT oppure OUTPUT oppure UPDATE.

Il modo di apertura INPUT specifica al computer che si vuole aprire il file per prelevare dei dati (lettura), il modo OUTPUT specifica che si vuole scrivere nel file, il modo UPDATE prevede sia la lettura che la scrittura.

Per i file sequenziali su nastro è previsto solo il modo INPUT e OUTPUT.

ESEMPIO:

```
10 OPEN # 3: "RS232.BA=1200", OUTPUT, FIXED 80
20 PRINT # 3: "PROVA STAMPA", 100, 5000
30 CLOSE # 3
```

Il computer apre il file # 3 per la stampante RS232 che consente una velocità di trasmissione di 1200 Baud, stampa le informazioni di linea 20 nel formato previsto; per la tabellazione potete utilizzare TAB (n).

ESEMPIO:

FASE SCRITTURA

```
10 INPUT "NOME": NOM$
20 INPUT "VALORE": V
30 CALL CLEAR
40 OPEN # 1: "CSI", OUTPUT, INTERNAL
50 PRINT # 1: NOM$, V, "AGENDA"
60 CLOSE # 1
```

FASE LETTURA

```
100 OPEN # 1: "CS1", INPUT, INTERNAL
110 INPUT # 1: NOM$, V, L$
120 CLOSE # 1
130 PRINT NOM$, V, L$
```

Nella fase di scrittura a linea 40 abbiamo aperto il file # 1 su nastro CS1 di tipo INTERNAL, OUTPUT, per default il computer prevede SEQUENTIAL e FIXED 64. Perciò l'unica attenzione va posta nella lunghezza delle variabili che non devono complessivamente eccedere i 64 caratteri e nelle caratteristiche del file ogni qualvolta lo volete utilizzare.

Essendo il file INTERNAL la variabile V occupa 9 byte, L$ occupa 7 byte e NOM$ occupa N+1 byte; complessivamente non si devono superare 64 byte.

ESEMPIO:

```
10 CALL CLEAR
20 REM  ARCHIVIO
30 OPEN #3:"CS1",SEQUENTIAL,INTERNAL,OUTPUT,FIXED 128
40 A$="NOME E COGNOME "
50 CALL CLEAR
60 B$="INDIRIZZO "
70 READ M,T,CONTO
80 IF M=99 THEN 140
90 PRINT #3:A$,B$,M,T,CONTO
100 REM  M=MATRICOLA,T=TELEFONO,CONTO=C.C.
110 PRINT :A$;" "&B$,:" "&STR$(M);" "&STR$(T);" "&STR$(CONTO)
120 GO TO 70
130 DATA 1,2423545,12,2,545436,345,99,99,99
140 CLOSE #3
150 END
```

La linea 30 crea un file SEQUENTIAL, INTERNAL, con RECORD a lunghezza fissa di 128 caratteri; per la fase di lettura dovete aprire il file # 3 con le stesse caratteristiche tranne l'OUTPUT che diventa INPUT.

Per i file RELATIVE è possibile specificare a quale RECORD accedere con l'opzione REC posta nelle istruzioni:

INPUT # n [,RECp] PRINT # n [,RECp]

è possibile effettuare il ripristino del RECORD con

RESTORE # n [,REC p]

nonché verificare la fine del file con

EOF (n)

Il conteggio dei RECORD parte da zero, per cui dovete fare attenzione quando specificate REC p. L'istruzione CALL FILES (N) dove N = 1÷9 consente di tener aperti contemporaneamente fino a 9 file; per default il computer assegna N = 3.

ESEMPIO

```
10 CALL CLEAR
20 OPEN #1:"DSK1.ARCHIVIO",INTERNAL,RELATIVE
30 FOR A=1 TO 20
40 INPUT "NOME E COGNOME  ":X$
50 CALL CLEAR
60 PRINT #1:X$,A
70 NEXT A
80 CALL CLEAR
90 INPUT "QUALE UTENTE ? ":A
100 IF A<1 THEN 150
110 IF A>20 THEN 150
120 INPUT #1,REC A-1:A$,B
130 PRINT A$,B
140 GO TO 90
150 CLOSE #1
160 END
```

La linea 20 consente la creazione di un file RELATIVE, INTERNAL sull'unità a dischi uno, con RECORD FIXED 80 e UPDATE.

Le linee 30-60 consentono di inserire nei 20 RECORD i nomi e i cognomi e le posizioni.

Le linee 90-120 consentono di prelevare le informazioni dal file chiedendo la posizione del RECORD da leggere.

La linea 130 effettua la stampa sul video delle informazioni.

La linea 150 chiude il file.

Caratteristiche dei file

Su nastro	*Su disco*
SEQUENTIAL	*SEQUENTIAL*/RELATIVE
FIXED	FIXED/VARIABILE
INPUT/OUTPUT	INPUT/OUTPUT/UPDATE
INTERNAL	INTERNAL/DISPLAY
Max lunghezza Record 192 byte	Max lunghezza Record 255 byte
1-255	1-255

Un esempio completo della gestione dei file è rappresentato dall'esercizio 5.1 che vi consente di creare un ARCHIVIO PERSONALE o SU DISCO o SU NASTRO.

ESEMPI:

OPEN ≠ 1 : "RS232"

apertura del file≠ 1 su porta seriale RS232; per default la velocità di trasmissione è di 300 Baud.

OPEN ≠ 1 : "RS232.BA=9600.CH", FIXED 80

apertura del file # 1 su porta seriale RS232; velocità di trasmissione 9600 Baud con verifica del controllo di parità (CH); RECORD di stampa 80 caratteri fissi.

OPEN # 1 : "PIO.EC"

apertura del file # 1 su porta parallela I/O con disattivazione della funzione ECO

OPEN # 1: "SPEECH", OUTPUT

apertura del file # 1 sul sintetizzatore vocale per l'emissione della voce.

```
10 CALL CLEAR
20 OPEN #1:"SPEECH",OUTPUT
30 OPEN #2:"ALPHON",INTERNAL
40 INPUT "FRASE= ":A$
50 IF A$="" THEN 40
60 PRINT #1:A$
70 INPUT #2:B$
80 Z=LEN(B$)
90 FOR R=4 TO Z
100 PRINT ASC(SEG$(B$,R,1))
110 NEXT R
120 GOTO 40
```

Inserendo il modulo TERMINAL EMULATOR II nella consolle e selezionando l'interpetre BASIC, con questo programma potete ascoltare con il sintetizzatore vocale collegato al computer tutte le frasi italiane e inglesi che digitate.

La linea 100 effettua la stampa del codice Ascii delle lettere.

2.12 CONSIGLI PRATICI

- Potete ottenere l'EDIT della linea semplicemente battendo il numero di linea e premendo i tasti FCTN E (↑).
- Potete ottenere lo scrollo verticale delle istruzioni sia con l'EDIT e premendo i tasti FCTN E (↑), FCTN X (↓); sia con il comando NUM e il tasto ENTER, avendo cura prima di fare un RESEQUENCE per essere sicuri che il passo d'incremento delle linee sia uguale a quello del NUM.
- Potete allungare di oltre 50 caratteri le linee delle istruzioni lunghe 28 × 4 = 112 caratteri seguendo questa procedura: scrivete un carattere qualsiasi nell'ultima posizione della 4ª linea dell'istruzione in esame, premete ENTER, richiamate la linea con l'EDIT e il cursore potrà scorrere anche sulla 5ª linea; potete seguire la stessa procedura per la 6ª linea.
- Quando utilizzate le matrici e i vettori ricordate che il computer utilizza la base 0, ad esempio DIM A (6,6) è una matrice di 7×7 = 49 elementi, DIM (99) è un vettore di 100 elementi; OPTION BASE 1 permette di cambiare la base a 1.
- Un vettore A (1000) occupa 8K byte di RAM, una matrice A (40,40) occupa 12.5K, una matrice A (11,11,11) occupa 14K; nel caso di matrici alfanumeriche vuote A$ (1000) occupa 2K, A$ (40,40) occupa 3.4K, A$ (11,11,11) occupa 3.5K. Sono ammesse matrici fino a tre dimensioni.
- Nella configurazione base sono disponibili 14K RAM per l'utente; collegando il sistema di periferiche sono disponibili 12K RAM, per evitare di andare in * * MEMORY FULL utilizzate una matrice fittizia che possa contenere lo spazio utile per non eccedere i 14K RAM e vi permette di conoscere per differenza lo spazio occupato dal vostro programma.
- Per la ricerca degli errori utilizzate:
  Il BREAK che realizza l'interruzione del programma dove vi interessa;
  il TRACE che segue dinamicamente e visualizza il numero delle istruzioni appena eseguite;
  i tasti FCTN e CLEAR che realizzano il Breakpoint e con il comando PRINT stampate le variabili di cui volete conoscere il contenuto attuale.
- Per la cancellazione dei programmi potete utilizzare DELETE sia da comando sia da istruzione senza ricorrere al DISK MANAGER.
- Per la gestione dei FILE utilizzate sempre una buona pianificazione degli spazi occupati, del tempo di ricerca, della chiusura corretta dei singoli file e riferitevi alle informazioni fornite nel paragrafo 2.11.
- Se disponete dei moduli PERSONAL RECORD KEEPING o STATISTICS, selezionando il loro interprete BASIC avete a disposizione diversi sottoprogrammi che aumentano la velocità di esecuzione, tra essi c'è il DISPLAY AT e ACCEPT AT nel seguente formato:

CALL D (Riga, Colonna, Lungh. stringa, "stringa")

questo sottoprogramma visualizza a riga e colonna desiderate la stringa di lunghezza fissata.

CALL A (Riga, Colonna, Lungh. Campo, C, Variabile di ritorno, intervallo di validità)

questo sottoprogramma permette di accettare a riga e colonna desiderate i valori fissati in un certo intervallo di validità.

ESEMPI:

```
10 CALL D (10,10,5, "SI/NO")
```

visualizza a riga e colonna 10 la stringa di 5 caratteri

```
10 CALL A (10,10,2,C,T$)
20 IF T$="SI" THEN 50
```

permette di accettare un valore alfanumerico di 2 caratteri a riga e colonna 10, suona il Beep, C assume il valore 0 o 1.

```
10 CALL A (10,10,2,C,T,1,15)
```

permette di accettare un valore numerico di due cifre compreso fra 1 e 15, suona il Beep.

- I TASTI CTRL (A), CTRL (,) non devono essere battuti in quanto essi determinano dei segnali di controllo utilizzati dalla trasmissione seriale per il registratore a cassette.
- Se disponete dell'EDITOR/ASSEMBLER potete utilizzare 80 colonne di schermo, avete a disposizione un EDITOR completo dei comandi FIND, REPLACE, MOVE, SHOW, INSERT, COPY, DELETE, EDIT, ADJUST, HOME, TAB, che agiscono sulle linee e sui caratteri di un testo, nonché una configurazione della tastiera con funzioni di scrollo orizzontale, verticale, spostamento del cursore in relazione al TAB, "finestra" per analizzare il contenuto di un testo.
- Se disponete del modulo TERMINAL EMULATOR II potete implementare una rete di comunicazione fra HOME COMPUTER, nonché insieme al SINTETIZZATORE VOCALE creare una qualsiasi espressione italiana o inglese per poi ascoltarla; l'espressione vocale è del tutto generale in quanto il modulo accetta qualsiasi composizione di fonemi.
- Per l'ALTA RISOLUZIONE GRAFICA, 49152 punti tutti indirizzabili e colorabili, potete utilizzare sia l'ASSEMBLER sia la MINI MEMORY che consente l'utilizzo dell'ASSEMBLER LINE BY LINE.

CAPITOLO III

IL TI BASIC ESTESO

Premesso che tutti i comandi, funzioni, sottoprogrammi, istruzioni analizzate nel capitolo II del TI BASIC sono valide anche nel BASIC ESTESO, ossia che un programma scritto in TI BASIC è eseguibile anche in BASIC ESTESO, in questo capitolo analizzeremo solo quelle istruzioni, comandi, sottoprogrammi e funzioni che appartengono al BASIC ESTESO.

3.1 COMANDI DI SISTEMA OPERATIVO

SIZE

permette di conoscere quanto spazio di memoria RAM è disponibile.

RUN "DSK1.nome"

permette di mandare in esecuzione il programma specificato dal nome sull'unità a dischi n. 1.

RUN n. linea

permette di mandare in esecuzione un programma dall'istruzione n. in poi.

EDIT n. linea

è realizzato esprimendo il n. di linea e battendo i tasti (FCTN 1E), dove n rappresenta il numero di linea n da visualizzare, il comando EDIT scritto per intero non è ammesso.

SAVE

è potenziato con l'opzione protected che permette di salvare i programmi su dischi o nastro in modo protetto, ossia non è possibile richiamare il listato sorgente BASIC nè tantomeno copiare su altri dischi o nastri.

SAVE CS1,PROTECTED

SAVE "DSK1.nome",PROTECTED

SAVE "DSK1.nome",MERGE

L'opzione MERGE permette di salvare i programmi su disco con la possibilità di essere richiamati e "fusi" con altri programmi presenti in memoria principale.

MERGE

è il comando che permette di fare la "fusione" di un programma presente su memoria di massa (UNITÀ A DISCHI) con un programma in memoria principale.

MERGE DSK1.nome

il nome identifica il programma memorizzato su dischetto nell'unità 1, detto programma è fuso con quello presente in memoria centrale.

3.2 ISTRUZIONI

DISPLAY AT(Riga,Colonna) [,OPZIONI] : variabili

è l'istruzione che permette di visualizzare sullo schermo (24 righe x 28 colonne) valori numerici, variabili e stringhe alfanumeriche nella posizione desiderata, assegnando la riga e la colonna. È un'estensione dell'istruzione PRINT.

Il formato completo si avvale anche di altre opzioni:

ERASE ALL	permette di cancellare tutto ciò che è sullo schermo prima di visualizzare i caratteri.
BEEP	permette di ascoltare un segnale sonoro prima di visualizzare i valori.
SIZE(n)	assegna la lunghezza dei caratteri da visualizzare a partire da una riga e colonna assegnata. Se non utilizzate tutta la riga questa è completata con gli spazi.

Esempio:

```
10  DISPLAY AT(5,10) ERASE ALL SIZE(6) BEEP : X$
```

questa istruzione visualizza a riga 5 e colonna 10 il valore di X$ lungo 6 caratteri, suona il BEEP e pulisce lo schermo.

ACCEPT AT(Riga, Colonna) [,OPZIONI] :variabili

è l'istruzione che permette di accettare i dati durante l'esecuzione del programma da una posizione assegnata sullo schermo (24 righe x 28 colonne). È un'estensione dell'istruzione INPUT. Il formato completo si avvale di molte opzioni:

ERASE ALL	pulisce l'intero schermo prima di accettare i valori.
BEEP	permette di ascoltare un segnale sonoro prima di accettare i valori.
VALIDATE	permette di definire il tipo di caratteri accettabili o di definire la limitazione ad alcuni caratteri desiderati. I tipi sono:
UALPHA	per caratteri alfabetici maiuscoli
DIGIT	per numeri da 0 a 9
NUMERIC	per numeri da 0 a 9 e "+","—",".","E"
LIMITAZIONE	a caratteri racchiusi fra " "

SIZE (n) rappresenta la lunghezza massima di caratteri accettabili a partire dalla riga e colonna assegnate. Se il numero n è positivo, il campo assegnato è pulito prima di accettare i dati. Se il numero n è negativo, il campo assegnato non cancella il valore preesistente in quella posizione.

Esempi:

```
10  ACCEPT AT(10,10) ERASE ALL : V
```

questa istruzione accetta il valore numerico a riga 10 e colonna 10 pulendo prima lo schermo; il valore è posto nella variabile V.

```
10 ACCEPT AT(5,4) BEEP VALIDATE ("ABC") : S$
```

questa istruzione accetta il valore alfabetico combinazione delle lettere A,B,C a riga 5 e colonna 4 suonando il BEEP; il valore è posto nella variabile S$.

```
10 ACCEPT AT(2,3) VALIDATE(UALPHA) SIZE(5) : L$
```

questa istruzione accetta il valore alfabetico di lunghezza 5 caratteri a riga 2 e colonna 3; il valore è posto nella variabile L$.

```
10 ACCEPT AT(2,5) VALIDATE(DIGIT,"ABC") SIZE(-3) : X$
```

questa istruzione accetta il valore alfanumerico limitato ai numeri e alle lettere A,B,C di lunghezza 3 caratteri, senza cancellare il preesistente valore sullo schermo a riga 2 e colonna 5; il valore è posto nella variabile X$.

Esempio:

```
10 REM TABELLA NOMI E INDIRIZZI
15 REM NAM$ CONTIENE I NOMI,ADDR$ CONTIENE GLI INDIRIZZI
20 DIM NAM$(20),ADDR$(20)
25 DISPLAY AT(5,1)ERASE ALL:"NOME :"
30 DISPLAY AT(7,1):"INDIRIZZO :"
35 DISPLAY AT(23,1):"BATTI 99 PER FINIRE"
40 FOR I=1 TO 20
45 ACCEPT AT(5,7)VALIDATE(UALPHA,"99")BEEP SIZE(13):NAM$(I)
50 IF NAM$(I)="99" THEN 70
55 ACCEPT AT(7,12)SIZE(13):ADDR$(I)
60 DISPLAY AT(7,12):"                "
65 NEXT I
70 CALL CLEAR
75 DISPLAY AT(2,1):"NOME","INDIRIZZO"
80 CALL CHAR(135,"FF")
85 CALL HCHAR(3,1,135,28)
90 FOR T=1 TO I-1
95 DISPLAY AT(T+3,1):NAM$(T):: DISPLAY AT(T+3,15):ADDR$(T)
100 NEXT T
105 PRINT "Batti un tasto per finire"
110 CALL KEY(0,K,SS)
115 IF SS=0 THEN 110
120 END
```

PRINT USING

è l'istruzione che permette di assegnare il formato alle variabili da stampare, è un'estensione del PRINT.

È utilizzato in un duplice formato:

1) **PRINT USING** "lettere alfanumeriche *###.###* " :variabili

2) **PRINT USING** n : variabili

   n **IMAGE** lettere alfanumeriche *###.###*

Per assegnare il formato alle variabili da stampare si utilizza il simbolo # ripetuto tante volte quante sono le cifre intere e decimali richieste, il primo # della sequenza rappresenta il segno meno.

Con *###.###* si possono rappresentare i numeri da –99.999 a 999.999.

Con *###* si possono rappresentare i numeri da –99 a 999

Regole da tener presente:

– Se il numero eccede il formato vengono stampati gli * * * * *.
– Se i numeri sono decimali questi vengono arrotondati in caso di troncamento.
– Se si stampano caratteri alfanumerici, questi vengono normalizzati a sinistra.
– Si possono aggiungere numeri o lettere alla sequenza di # # #

Esempio:

```
10 PRINT USING "IL VALORE È ##.# " :A
```

I valori ammessi di A sono –9.9 a 99.9

Valori di A	*Stampa ottenuta*
5.29	IL VALORE È 5.3
0	IL VALORE È .0
–3.72	IL VALORE È –3.7
78.689	IL VALORE È 78.7
101	IL VALORE È * * * * *

Esempio:

```
10  PRINT USING 20:A$
20  IMAGE ATTENZIONE AL #####
```

I valori ammessi di A$ sono stringhe di lunghezza massima 5 caratteri, le stringhe sono normalizzate a sinistra.

Valori di A$	*Stampa ottenuta*
CANE	ATTENZIONE AL CANE
BUE	ATTENZIONE AL BUE
LEONE	ATTENZIONE AL LEONE
PESCECANE	ATTENZIONE AL * * * *

Esempio:

```
10 PRINT USING " ###.## " : 63.7
```

stampa il valore 63.70

Esempio:

```
10 PRINT USING " ###.# " : 234.99
```

stampa il valore 235.0

Esempio:

```
100  PRINT USING 110 : 32.9,497.567
110  IMAGE I VALORI SONO ### E ###.##
```

si ottiene la seguente stampa:

```
I VALORI SONO 33 E 497.57
```

Esempio:

```
200  PRINT USING 210 : 85.39,388.777
210  IMAGE LIRE ##.# E ###.#
```

si ottiene la seguente stampa:

```
LIRE 85.4 E 388.8
```

PRINT USING nel seguente formato:

```
PRINT # n [,REC] USING " ### " : variabili
PRINT # n[,REC] USING n : variabili
```

è utilizzato per scrivere i dati nel file # n nel formato desiderato.

Esempio:

```
10 REM **USO DEL PRINT USING**
20 CALL CLEAR
30 IMAGE L ###.##
40 IMAGE "  ###.##"
50 DATA 12.67,13.456,34.19,0.234,1.3458,00.001,100.45
60 TOTALE=0
70 FOR A=1 TO 7
80 READ ADDENTI
90 TOTALE=TOTALE+ADDENTI
100 IF A=1 THEN PRINT USING 30:ADDENTI ELSE PRINT USING 40:ADDENTI
110 NEXT A
120 PRINT "----------"
130 PRINT USING "L####.##":TOTALE
```

RPT$ ("stringa",variabile numerica)

Questa funzione permette di ripetere una stringa il numero di volte espresso dalla variabile numerica.

Esempio:

```
10  PRINT RPT$("FRANCO",5)
```

questa istruzione stampa 5 volte Franco.

LINPUT ["**espressioni alfanumeriche**":] **variabile stringa**
LINPUT [#**file**] [,**REC n**] : **variabile stringa**

questa istruzione è un'estensione dell'istruzione INPUT e permette alle variabili di stringa di contenere oltre i simboli alfanumerici anche le virgole, i punti, gli spazi, gli apici, i punti e virgola.

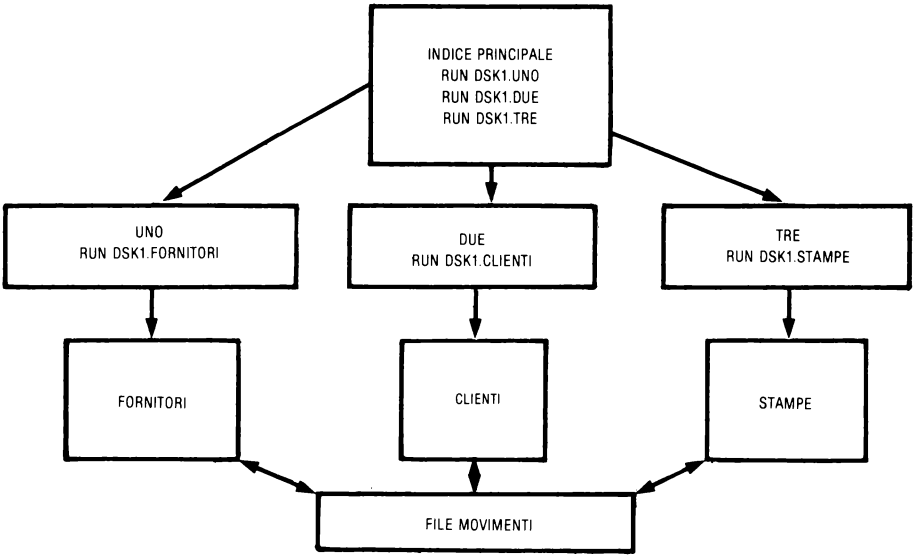
Come esempio vedete il WORD PROCESSING nel Cap. V

RUN "Dispositivo.nome"

RUN Numero di linea

RUN oltre ad essere utilizzato come comando, può essere utilizzato come istruzione, il che comporta la possibilità di segmentare un programma e implementare una struttura ad albero per accedere solo a parti di programma, aumentare la velocità di esecuzione e dedicare a ciascuna parte di programma una specifica funzione con aumento di leggibilità e modularità del programma.

Esempio:



I dispositivi indicati nell'istruzione possono essere tutte e tre le unità a dischi (DSK1, DSK2, DSK3).

3.3 CREAZIONE SOTTOPROGRAMMI

Le istruzioni che permettono la creazione di sottoprogrammi e il loro utilizzo sono:

SUB nome sottoprogramma (lista parametri)

SUBEND

CALL nome sottoprograma (lista parametri)

Un sottoprogramma è un insieme di istruzioni che realizzano uno specifico calcolo o funzione e può essere chiamato più volte nel corso di un programma principale, esso è identificato dal nome e lo scambio di informazioni o dati avviene tramite la lista dei parametri presenti sia nel sottoprogramma sia nel programma principale.

L'istruzione – CALL nome – serve a chiamare il sottoprogramma e l'istruzione SUBEND chiude il sottoprogramma (deve essere l'ultima istruzione; è l'equivalente del RETURN analizzato nel Cap. 2.6) e rimanda l'esecuzione al programma principale, in particolare all'istruzione successiva al – CALL nome.–

Vediamo la struttura tipica implementata con Programma principale e Sottoprogrammi:

```
PROGRAMMA PRINCIPALE
  .
  .
  .
  .
CHIAMATA SOTTOPROGRAMMA UNO(lista parametri)
  .
  .
CHIAMATA SOTTOPROGRAMMA DUE(lista parametri)
  .
  .
CHIAMATA SOTTOPROGRAMMA UNO(lista parametri)
  .
  .
END

SUB UNO(lista parametri)
  .
  .
  .
SUBEND
SUB DUE(lista parametri)
  .
```

```
SUBEND
```

Le variabili numeriche, alfanumeriche, Array definite nel sottoprogramma hanno valore locale nell'ambito del proprio sottoprogramma, nel senso che uscendo dal sottoprogramma le variabili perdono di significato.

Il trasferimento o comunicazione di valori fra programma principale e sottoprogramma avviene tramite l'ordine sequenziale delle variabili nelle liste di parametri, che devono contenere necessariamente nello stesso ordine lo stesso tipo di variabili(numerico, alfanumerico,Array).

Cosi ad esempio:

```
CALL FRANCO(A,B,S$,L(1),x)
SUB FRANCO(C,H,A$,T(1),x)
```

fra le due liste di parametri è rispettato lo stesso ordine del tipo, solo il nome può essere diverso. Inoltre se viene trasferito un valore numerico, questo può essere alterato nell'ambito del sottoprogramma ma rimanere inalterato nel programma principale.

Un sottoprogramma chiamato più volte nel corso dell'esecuzione di un programma principale, aggiorna le variabili locali solo alla nuova chiamata.

Esempio:

```
100  CALL CLEAR
110  INPUT"VALORE":VA
120  INPUT"NOME ":NOM$
130  CALL CLEAR
140  CALL CALCOLO(VA,NOM$)
150  PRINT"VA=";VA
160  PRINT"NOM = ";NOM$
170  END
200  SUB CALCOLO(X,L$)
210  X=X*100
220  L$ =L$ & "NUMERICO"
230  PRINT"UN MOMENTO PREGO": : :
240  SUBEND
```

Qualora si voglia uscire dal sottoprogramma prima della fine di esso si può utilizzare l'istruzione SUBEXIT.

Il controllo è trasferito all'istruzione successiva alla chiamata sottoprogramma del programma principale.

3.4 SOTTOPROGRAMMI PER L'INDIVIDUAZIONE DI ERRORI

ON WARNING PRINT

ON WARNING STOP

ON WARNING NEXT

Questa istruzione consente di prendere una decisione in caso d'errore, come continuare l'esecuzione se si verifica un errore. L'opzione PRINT permette la visualizzazione del messaggio e vale anche per default; l'opzione STOP ferma l'esecuzione in corso e visualizza il messaggio; l'opzione NEXT permette di continuare l'esecuzione senza visualizzare messaggi.

ON ERROR STOP

ON ERROR n **n è il numero di linea**

Questa istruzione consente di prendere una decisione in caso d'errore, così il controllo può essere passato ad un sottoprogramma senza interrompere l'esecuzione. L'opzione STOP interrompe la esecuzione e visualizza il messaggio e vale anche per default; l'opzione numero di linea permette di trasferire il controllo dell'esecuzione alla linea desiderata se si verifica un errore.

ON ERROR n corrisponde ad un salto al sottoprogramma iniziante alla linea n, per cui la routine a linea n deve terminare con RETURN.

Si possono utilizzare diverse opzioni:

RETURN

RETURN m

RETURN NEXT

m è la linea dove viene passato il controllo dopo che si è verificato l'errore; l'opzione NEXT passa il controllo all'istruzione successiva a quella dove si è verificato l'errore, senza opzioni il controllo passa all'istruzione dove si è verificato l'errore.

CALL ERR (C,T) è il sottoprogramma che permette di conoscere il codice dell'errore e il tipo di errore ultimo verificatosi. È utilizzato insieme all'ON ERROR. Consultate l'APPENDICE G per i codici di ERRORI PREVISTI.

3.5 SOTTOPROGRAMMI PER ACCEDERE A ROUTINE ASSEMBLER

CALL INIT

CALL LINK (nome sottoprogramma, parametri)

CALL LOAD ("nome di accesso", [indirizzo, byte] ... [, FILE])
CALL PEEK (indirizzo, variabili numeriche)

Il sottoprogramma INIT permette di verificare la connessione dell'ESPANSIONE DI MEMORIA 32K RAM, prepara il computer a mandare in esecuzione un programma Assembler e carica routine di utilità nell'ESPANSIONE DI MEMORIA.

Il sottoprogramma LINK permette lo scambio di parametri fra programmi in BASIC ESTESO e sottoprogrammi Assembler.

Il nome sottoprogramma rappresenta il sottoprogramma Assembler precedentemente caricato con CALL LOAD e i parametri rappresentano variabili o numeri da essere scambiati con il sottoprogramma Assembler.

Il sottoprogramma LOAD carica in memoria un programma oggetto Assembler o dati direttamente da file costituiti dall'indirizzo e dal contenuto del byte associato. Ciascun indirizzo e dato del byte sono separati dalla virgola. Il nome del file permette di accedere al file dove sono contenuti i codici oggetti.

Il sottoprogramma PEEK permette di conoscere il contenuto delle locazioni di memoria. Assegnando l'indirizzo del byte di partenza e le variabili numeriche si ottiene che il valore contenuto a quell'indirizzo e i successivi sono caricati nelle variabili numeriche indicate.

ESEMPIO:

```
10 CALL PEEK (8010,N1,N2,N3,N4,N5)
20 PRINT N1;N2;N3;N4;N5

Stampa    220    160    136    0    6
```

questi numeri rappresentano i contenuti dei byte agli indirizzi 8010,8011, 8012, 8013, 8014.

3.6 SOTTOPROGRAMMI PER GIOCHI

Premesso che nel Basic Esteso valgono sempre i sottoprogrammi definiti per i giochi nel Cap. II, qui illustriamo tutti i sottoprogrammi che permettono di creare e movimentare le figure (SPRITE).

1) CALL SPRITE	**permette di definire gli Sprite (figure in movimento)**
2) CALL CHARPAT	**permette di conoscere il codice esadecimale (PATTERN) dei caratteri Ascii e propri.**
3) CALL CHARSET	**permette di ripristinare i codici e colori dei caratteri Ascii (32-95).**
4) CALL COINC	**permette di conoscere se c'è stata coincidenza fra due figure in movimento (Sprite).**
5) CALL DELSPRITE	**permette di cancellare gli Sprite.**
6) CALL DISTANCE	**permette di conoscere la distanza tra due Sprite.**
7) CALL LOCATE	**permette di riposizionare gli Sprite.**
8) CALL MAGNIFY	**permette di ingrandire gli Sprite.**
9) CALL MOTION	**permette di assegnare le velocità agli Sprite.**
10) CALL PATTERN	**permette di cambiare il codice esadecimale (PATTERN) degli Sprite.**
11) CALL POSITION	**permette di conoscere la posizione degli Sprite.**
12) CALL SAY	**permette di far parlare il Computer (occorre collegare il sintetizzatore vocale).**
13) CALL SPGET	**permette di conoscere le parole pronunciate dal CALL SAY.**

— **CALL SPRITE** (# numero sprite, codice carattere, colore sprite, riga, colonna [, velocità di riga, velocità di colonna]).

Premesso che lo SPRITE è una figura in movimento (FOLLETTO), esso è molto utilizzato per creare giochi dove si possono sovrapporre, movimentare, accelerare, fermare, ingrandire, colorare, cancellare le figure.

Il numero dello sprite varia da 1 a 28 ed è sempre preceduto dal simbolo # . Il # numero identifica uno e un solo sprite. Definendo con lo stesso numero un secondo sprite, questo cancella il precedente e conserva la stessa velocità se non viene definita una nuova. Gli sprite scorrono sopra le figure o caratteri fissi. Su due sprite

coincidenti prevale quello con numero minore, su di una stessa riga dal quinto sprite in poi è cancellato quello con numero maggiore.

Il codice dello sprite è un numero che varia da 32 a 143 (si ricordi che nel Basic esteso sono ammessi 15 set) e deve essere predefinito necessariamente con CALL CHAR. Ripetiamo qui il formato di CALL CHAR: CALL CHAR (codice carattere, "PATTERN carattere") il codice numerico varia da 32 a 143 e la PATTERN di un carattere è rappresentata da 16 cifre esadecimali che codificano il carattere. Qualora la PATTERN sia minore di 16 cifre, il computer inserisce fino al completamento tutti zero; la PATTERN di 32 cifre definisce due caratteri, quella di 64 definisce quattro caratteri. Per maggiori informazioni consultate il Paragarafo 2.

Il colore dello sprite è un numero che varia da 1 a 16, esso definisce il FOREGROUND, il BACKGROUND è sempre 1 (colore trasparente).

Il numero di riga e di colonna variano rispettivamente da 1 a 192 e da 1 a 256 a partire dall'angolo superiore sinistro. La posizione dello sprite è rappresentata dal suo angolo superiore sinistro. Gli sprite fermati con il BREAKPOINT non riappaiono con il comando CONTINUE.

La velocità di riga e di colonna permette di movimentare gli sprite in tutte le direzioni, cioè:
la velocità di riga positiva muove lo sprite verso il basso,

la velocità di riga negativa muove lo sprite verso l'alto,

la velocità di colonna positiva muove lo sprite verso destra,

la velocità di colonna negativa muove lo sprite verso sinistra.

Se sono assegnate entrambe le velocità lo sprite si muove nell'angolazione determinata da esse.

I numeri che rappresentano la velocità variano da −127 a 127. Il valore 0 coincide con lo sprite fermo, allontanandosi da 0 aumenta la velocità.

– CALL CHARPAT (variabile numerica, variabile alfanumerica)

Questo sottoprogramma permette di conoscere il codice esadecimale (PATTERN) che definisce il carattere Ascii (32-95) o il carattere proprio assegnando il codice numerico. La PATTERN che definisce il carattere è posta nella variabile alfanumerica.

– CALL CHARSET

Questo sottoprogramma ripristina i codici e i colori standard dei caratteri Ascii (32-95).

– CALL COINC (# numero sprite, # numero sprite, tolleranza, variabile numerica)
– CALL COINC (# numero sprite, riga, colonna, tolleranza, variabile numerica)
– CALL COINC (ALL, variabile numerica)

Questo sottoprogramma permette di conoscere se c'è coincidenza fra due o più sprite o fra uno sprite e una data locazione definita da riga e da colonna. ALL permette di considerare tutti gli sprite definiti.

Due sprite sono coincidenti quando il loro angolo sinistro superiore cade all'interno di una assegnata tolleranza.

Uno sprite e una locazione sono coincidenti quando l'angolo superiore sinistro dello sprite e la locazione cadono all'interno di una assegnata tolleranza. La variabile numerica è posta a 1 quando c'è coincidenza a 0 nel caso contrario.

– CALL DELSPRITE (# numero sprite [,....])
– CALL DELSPRITE (ALL)

Questo sottoprogramma permette di cancellare uno o più sprite definiti, ALL li cancella tutti.

– CALL DISTANCE (# numero sprite, # numero sprite, variabile numerica)
– CALL DISTANCE (# numero sprite, riga, colonna, variabile numerica)

Questo sottoprogramma permette di conoscere la distanza fra due sprite o fra uno sprite e una locazione definita da riga e da colonna.

La variabile numerica conterrà la distanza fra due punti calcolata come la differenza di coordinate elevate al quadrato.

– CALL LOCATE (# numero sprite, riga, colonna [,...])

Questo sottoprogramma permette di cambiare la posizione assegnata agli sprite. Il numero di riga varia da 1 a 192, il numero di colonna da 1 a 256. La posizione dello sprite coincide con il suo angolo superiore sinistro.

– CALL MAGNIFY (fattore d'ingrandimento)

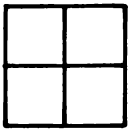
Questo sottoprogramma permette di ingrandire i caratteri o gli sprite, specificando il fattore d'ingrandimento che varia da 1 a 4.

Il fattore 1 è quello assegnato per default dal computer, con cui ciascun carattere occupa una matrice di 8×8 punti.

Il fattore 2 permette allo sprite di essere quadruplicato, ossia occupa lo spazio di quattro caratteri, e ciascun punto del carattere definito è ripetuto in quattro posizioni.

Il fattore 3 permette allo sprite di essere moltiplicato per quattro, a differenza del fattore 2, adesso è il carattere definito che si ingrandisce occupando quattro spazi di caratteri.

Il fattore 4 permette allo sprite di essere ingrandito di lunghezza quattro e di essere ripetuto quattro volte, il che significa avere lo sprite occupante 16 spazi di caratteri.



– CALL MOTION (# numero sprite, velocità di riga, velocità di colonna [,...])

Questo sottoprogramma permette di assegnare o variare la velocità allo sprite. Velocità di riga e velocità di colonna sono numeri varianti da –127 a 127, lo zero rappresenta lo stato di riposo per lo sprite, allontanandosi da zero aumenta la velocità. Valgono le stesse regole del CALL SPRITE per quanto riguarda le direzioni di essi.

– CALL PATTERN (# numero sprite, codice carattere [,...])

Questo sottoprogramma permette di cambiare la PATTERN agli sprite senza modificare le altre caratteristiche dello sprite. Il codice carattere è quello definito con il CALL CHAR (codice carattere, "PATTERN") insieme alla PATTERN.

– CALL POSITION (# numero sprite, riga, colonna [,...])

Questo sottoprogramma permette di conoscere la posizione occupata dagli sprite.

Riga e colonna sono due variabili in cui verrà posta la posizione di riga e di colonna. La posizione dello sprite è rappresentata dal suo angolo superiore sinistro.

– CALL SAY ("STRINGA" [, variabili stringa])

Questo sottoprogramma permette di far parlare il computer, ma è necessario che ad esso sia collegato lo SPEECH SYNTHESIZER (sintetizzatore vocale), il vocabolario utilizzabile da questo sottoprogramma è quello illustrato nell'APPENDICE H.

Possono essere inserite sia le stringhe da ascoltare che le variabili alfanumeriche contenenti frasi da voler ascoltare.

Per le parole non presenti nel vocabolario inglese il computer legge lettera per lettera.

— **CALL SPGET ("STRINGA" [, variabili stringa])**

Questo sottoprogramma è utilizzato insieme al CALL SAY per conoscere le frasi che il computer sta leggendo.

3.7 FUNZIONI ARITMETICHE-LOGICHE

ABS Calcola il valore assoluto di un numero.
ASC Ritorna il codice numerico Ascii dei caratteri.
ATN Calcola l'arcotangente degli angoli espressi in radianti.
COS Calcola il coseno degli angoli espressi in radianti.
INT Calcola la parte intera di un numero.
LOG Calcola il logaritmo naturale di un numero.
MAX Ritorna il valore numerico più grande fra due numeri.
MIN Ritorna il valore numerico più piccolo fra due numeri.
PI È il valore $\pi = 3.141592654$.
RND Genera numeri casuali fra 0 e 1.
SGN Ritorna il segno di un numero.
SIN Calcola il seno degli angoli espressi in radianti.
SQR Calcola la radice quadrata di un numero.
TAN Calcola la tangente degli angoli espressi in radianti.
AND Moltiplicazione logica.
OR Somma logica.
NOT Negazione logica.
XOR OR esclusivo logico.

Le funzioni logiche OR, AND, XOR, NOT permettono di implementare i seguenti formati di IF:

```
IF A < B AND B> C THEN 200
IF 3 < 4 OR 5 > 6 THEN X=4
IF 3 < 4 XOR 5 < 6 THEN 100 ELSE 200
IF (A OR B) AND (C XOR D) THEN 200
IF NOT A = B THEN 200
```

sono ammesse tutte le combinazioni di OR, AND, XOR, NOT.

La funzione PI = $\pi = 3.14$ permette la conversione immediata di radianti in gradi e viceversa.

Le funzioni MIN e MAX permettono il confronto fra variabili numeriche:

ESEMPI:

```
PRINT MIN(12,–2)
PRINT MAX(20,30).
```

Molto utile è l'istruzione DEF che vi permette di definire le vostre funzioni. Di seguito vengono riportate le funzioni trigonometriche da definire:

Secant

```
DEF SEC (X) =1/COS(X)
```

Cosecant

```
DEF CSC(X)=1/SIN(X)
```

Cotangent

```
DEF COT(X)=1/TAN(X)
```

Inverse Sine

```
DEF ARCSIN(X)=ATN(X/SQR(1-X*X))
```

Inverse Cosine

```
DEF ARCCOS(X)=-ATN(X/SQR(1-X*X))+PI/2
```

Inverse Secant

```
DEF ARCSEC(X)=ATN(SQR(X*X-1))+(SGN(X)-1)*PI/2
```

Inverse Cosecant

```
DEF ARCCSC(X)=ATN(1/SQR(X*X-1))+(SGN(X)-1)*PI/2
```

Inverse Cotangent

```
DEF ARCCOT(X)=PI/2-ATN(X) or =PI/2+ATN(-X)
```

Hyberbolic Sine

```
DEF SINH(X)=(EXP(X)-EXP(-X))/2
```

Hyberbolic Cosine

```
DEF COSH(X)=(EXP(X)+EXP(-X))/2
```

Hyperbolic Tangent

```
DEF TANH(X)=-2*EXP(-X)/(EXP(X)+EXP(-X))+1
```

Hyperbolic Secant

```
DEF SECH=2/(EXP(X)+EXP(-X))
```

Hyperbolic Cosecant

```
DEF CSCH=2/(EXP(X)-EXP(-X))
```

Hyperbolic Cosecant

```
DEF CSCH=2/(EXP(X)-EXP(-X))
```

Hyperbolic Cotangent

```
DEF COTH(X)=2*EXP(-X)/(EXP(X)-EXP(-X))+1
```

Inverse Hyperbolic Sine

```
DEF ARCSINH(X)=LOG(X+SQR(X*X+1))
```

Inverse Huperbolic Cosine

```
DEF ARCCOSH(X)=L0G(X+SQR(X*X-1))
```

Inverse Hyperbolic Tangent

```
DEF ARCTANH(X)=LOG((1+X)/(1-X))/2
```

Inverse Hyperbolic Secant

```
DEF ARCSECH(X)=LOG((1+SQR(1-X*X))/X)
```

Inverse Hyperbolic Cosecant

```
DEF ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1)+1)/X)
```

Inverse Hyperbolic Cotangent

```
DEF ARCCOTH(X)=LOG((X+1)/(X-1))/2
```

3.8 CONSIGLI PRATICI

- Potete utilizzare l'EDIT semplicemente battendo il numero di linea desiderato e premendo contemporaneamente i tasti FCTN ed E(↑), la parola EDIT non è prevista.

- Potete cambiare numero di linea alle istruzioni effettuando un replace con queste operazioni: richiamate la linea con il numero di linea e FCTN E (↑), premete ENTER, premete i tasti FCTN REDO, il cursore si posizionerà sul numero di linea da voi selezionato.

- Sono previste 5 linee per ogni istruzione; potete aumentare la lunghezza delle istruzioni di 25 caratteri, scrivendo un carattere nell'ultima posizione della 5ª linea e facendo l'EDIT della linea, il cursore continuerà a correre per la 6ª linea.

- Potrete aumentare di molto la velocità di esecuzione utilizzando:
  - le routine ACCEPT AT e DISPLAY AT con tutte le opzioni previste;
  - scrivere più istruzioni separate da: : su di una stessa linea;
  - RUN come istruzione che vi consente di segmentare il programma;
  - inserire delle routine Assembler;
  - eliminare il "pre-scan" con ! ○ — per le istruzioni che lo consentono; le istruzioni che necessitano il "pre-scan" sono DIM, DATA, SUB, DEF, CALL: ! ○ P + rappresenta (pre-scan ON);

- Potete facilitare la fase di correzione dei programmi utilizzando:
  - i comandi TRACE e BREAK;
  - i sottoprogrammi ON ERROR, ON WARNING, CALL ERR.

- Potete salvare un programma su disco con il nome LOAD, il computer automaticamente all'accensione richiamerà il programma e lo manderà in esecuzione.

- Potete utilizzare molte frasi di commento inserendo il ! sulla linea, che è l'equivalente del REM.

- Quando utilizzate il PRINT con i due punti (: : :), per far saltare i righi di stampa, i due punti vanno separati da uno spazio.

- Potete utilizzare molti sottoprogrammi per i giochi ricordando che avete a disposizione 15 Set di caratteri grafici.

- Potete implementare programmi con struttura ad albero utilizzando RUN da istruzione, molto utile per procedure commerciali e gestionali.

- IF THEN ELSE è anche valido nel formato – :

```
IF X=2 THEN 100 ELSE X=X+1
```

- È consentita anche l'assegnazione:

```
A,B,C=0          A$,B$="CASA"
```

- Potete proteggere i vostri programmi con l'opzione protected utilizzata con il SAVE; non sarà più possibile accedere al listato sorgente né effettuare copie.
- Potete fare il MERGE (fusione) fra programmi su disco e in memoria RAM.
- Potete conoscere in ogni istante lo spazio libero in memoria con il comando SIZE.
- Potete richiamare delle routine Assembler con LOAD, LINK, PEEK.
- Potete implementare i sottoprogrammi con variabili locali che vi consentono una struttura altamente modulare dei programmi, ciò aumenta la leggibilità e la facilità di correzioni o successivo sviluppo.
- Se disponete dello SPEECH SYNTHESIZER (Sintetizzatore vocale) potete inserire molte espressioni vocali o parole nei vostri programmi con CALL SAY. Il vocabolario contiene 373 parole e lettere ma sono ammesse anche le combinazioni (SOME+THING produce SOMETHING) e l'aggiunte dei suffissi ING, S, ED (ANSWERING, ANSWERS, ANSWERED).

CAPITOLO IV

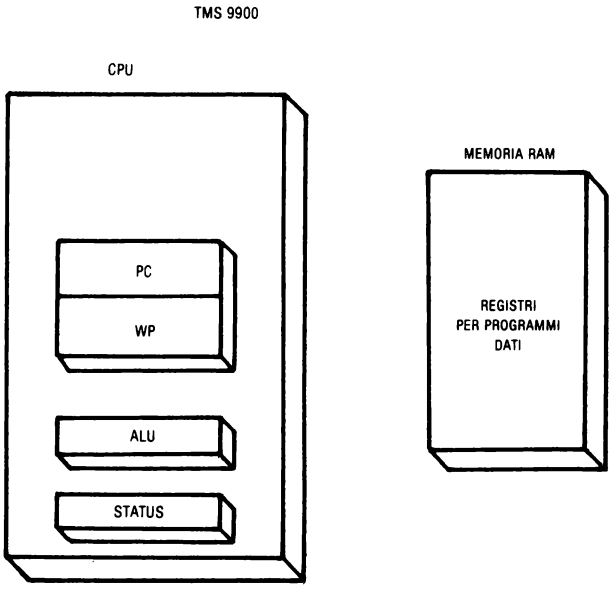
IL TMS 9900

4.1 ARCHITETTURA DEL TMS 9900

Il TMS 9900 è un microprocesore a tecnologia MOS LSI (Large Scale Integration) a 16 bit. La capacità di indirizzamento è di 32K parole (una parola è 16 bit).

Questo microprocessore non dispone di registri interni di lavoro (WORKSPACE REGISTERS) essi vengono creati dinamicamente in RAM, ciò favorisce il controllo di più processi ciascuno con la sua area di lavoro che viene conservata all'atto dell'interruzione.

Diamo uno sguardo all'architettura interna del 9900:

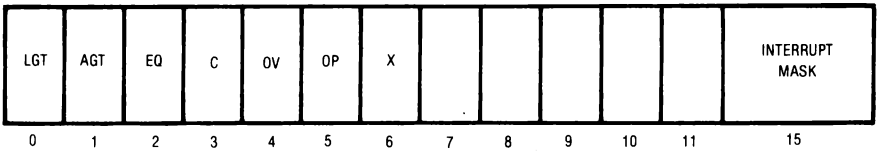


I 16 registri definiti in RAM esterna (WORKSPACE REGISTER) vengono detti registri "SOFTWARE", quelli interni (WP, PC, STATUS) vengono detti "HARDWARE".

I registri interni, tutti a 16 bit, sono:

— **Program Counter (PC)** contiene l'indirizzo della prossima istruzione da eseguire. Esso si incrementa automaticamente all'esecuzione di ciascuna istruzione, ad eccezione di quelle istruzioni che vanno direttamente a modificarlo.

— **Satus Register** contiene i bit di flag indicanti i risultati dell'ultima operazione aritmetica o logica eseguita, nonché 4 bit di maschera per Interrupt.



LGT (Logical Greather Than) — esso è alto qualora è verificata la diseguaglianza fra numeri non relativi.

AGT (Aritmetic Greather Than) — esso è alto qualora è verificata la diseguaglianza fra numeri relativi.

EQ (Equal) — Esso è alto quando due byte o parole comparate risultano eguali.

C (Carry) — Esso è alto se il bit più significativo viene sciftato o c'è il riporto di un'operazione.

OV (Overflow) — Esso è alto quando si verifica un traboccamento del valore numerico.

OP (Odd Parity) — Esso è alto quando c'è un numero dispari di bit 1 in un risultato numerico (controllo di parità).

X (Extended Operation) — Esso è alto quando i registri di PC e WP richiedono istruzioni di diverse parole macchine.

7 - 11 — sono riservati.

12 - 15 — (Interrupt Mask) — Sono settati alti per definire i livelli di interrupt.

WORKSPACE Pointer (WP) — Contiene l'indirizzo del primo registro della WORKSPACE AREA costituita da — 16 registri —

L'area di lavoro (W. REGISTERS) è costituita da 16 registri allocati esternamente in RAM. Essi, da 0 a 15 (WR0, WR15) sono general purpose per memorizzare indirizzi e dati temporanei; in particolare il registro 0 (WR0) può contenere il numero di posizioni di scorrimento definito nelle istruzioni di shift (SLA, SRA, SRC, SRL).

WR11 contiene l'indirizzo di ritorno che viene caricato ogni qualvolta si esegue un'istruzione di branch e link (BL).

WR13, WR14, WR15 contengono rispettivamente WP, PC, ST di un processo interrotto con Context Swiching.

WR12 contiene l'indirizzo base per le istruzioni di CRU.

In questo schema sono riportati i bit dello STATUS REGISTER che vengono utilizzati dalle istruzioni ASSEMBLER. Il simbolo x identifica il bit interessato.

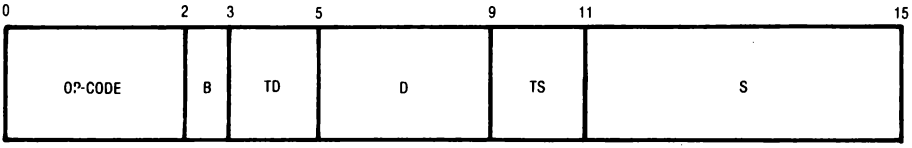
Mnemonic	L>	A>	EQ	C	OV	OP	X	Mnemonic	L>	A>	EQ	C	OV	OP	X
A	X	X	X	X	X	–	–	JOP	–	–	–	–	–	–	–
AB	X	X	X	X	X	X	–	LDCR	X	X	X	X	–	2	–
ABS	X	X	X	X	X	–	–	LI	X	X	X	–	–	–	–
AI	X	X	X	X	X	–	–	LIMI	–	–	–	–	–	–	–
ANDI	X	X	X	–	–	–	–	LWPI	–	–	–	–	–	–	–
B	–	–	–	–	–	–	–	MOV	X	X	X	–	–	–	–
BL	–	–	–	–	–	–	–	MOVB	X	X	X	–	–	X	–
BLWP	–	–	–	–	–	–	–	MPY	–	–	–	–	–	–	–
C	X	X	X	–	–	–	–	NEG	X	X	X	X	X	–	–
CB	X	X	X	–	–	X	–	ORI	X	X	X	–	–	–	–
CI	X	X	X	–	–	–	–	RTWP	X	X	X	X	X	X	X
CLR	–	–	–	–	–	–	–	S	X	X	X	X	X	–	–
COC	–	–	X	–	–	–	–	SB	X	X	X	X	X	X	–
CZC	–	–	X	–	–	–	–	SBO	–	–	–	–	–	–	–
DEC	X	X	X	X	X	–	–	SBZ	–	–	–	–	–	–	–
DECT	X	X	X	X	X	–	–	SETO	–	–	–	–	–	–	–
DIV	–	–	–	–	X	–	–	SLA	X	X	X	X	X	–	–
INC	X	X	X	X	X	–	–	SOC	X	X	X	–	–	–	–
INCT	X	X	X	X	X	–	–	SOCB	X	X	X	–	–	X	–
INV	X	X	X	–	–	–	–	SRA	X	X	X	X	–	–	–
JEQ	–	–	–	–	–	–	–	SRC	X	X	X	X	–	–	–
JGT	–	–	–	–	–	–	–	SRL	X	X	X	X	–	–	–
JH	–	–	–	–	–	–	–	STCR	X	X	X	–	–	2	–
JHE	–	–	–	–	–	–	–	STST	–	–	–	–	–	–	–
JL	–	–	–	–	–	–	–	STWP	–	–	–	–	–	–	–
JLE	–	–	–	–	–	–	–	SWPB	–	–	–	–	–	–	–
JLT	–	–	–	–	–	–	–	SZC	X	X	X	–	–	–	–
JMP	–	–	–	–	–	–	–	SZCB	X	X	X	–	–	X	–
JNC	–	–	–	–	–	–	–	TB	–	–	X	–	–	–	–
JNE	–	–	–	–	–	–	–	X	3	3	3	3	3	3	3
JNO	–	–	–	–	–	–	–	XOP	3	3	3	3	3	3	3
JOC	–	–	–	–	–	–	–	XOR	X	X	X	–	–	–	–

BIBLIOGRAFIA:
TMS 9900 Family System Development manual, Texas Inst., pag. 1-6 11-15.

4.2 MODI D'INDIRIZZAMENTO

Un'istruzione Assembler occupa 16 bit(1 parola) di memoria. Ciascuna parola è suddivisa in campi che variano a seconda del modo di indirizzamento e dal tipo di istruzione (salto, somma, funzione logica ecc.).

Premesso che esistono 9 formati di istruzioni, vediamo come si presenta un'istruzione Assembler nel formato I e II.



Formato I

Formato II

OP–CODE	– Codice Operativo
B	– Bit indicante istruzioni a 1 byte(1), a 2 byte(0).
TD	– Tipo di indirizzamento dell'operando Destinazione
D	– Operando Destinazione
TS	– Tipo di indirizzamento dell'operando Sorgente
S	– Operando Sorgente
DISP	– Costante di spiazzamento

L'operando SORGENTE è un numero, un indirizzo, ecc. su cui si può operare per ottenere un dato risultato. L'operando DESTINAZIONE è l'indirizzo dove il risultato di un'operazione aritmetico/logica è memorizzato. L'istruzione Assembler specifica gli operandi S e D e il modo di indirizzamento di essi, nonchè il codice operativo che specifica l'operazione aritmetico/logica da effettuarsi.

Ci sono 8 modi di indirizzamento, vediamo i primi 5 che riguardano ciascun operando:

Modi d'indirizzamento	*Campo T*	*Esempio*
1) WORKSPACE REGISTER	00	7
2) WORKSPACE REGISTER INDIRETTO	01	* 9
3) MODO SIMBOLICO	10	@ CAR
4) MODO INDICIZZATO	10	@CAR(6)
5) WORKSPACE REGISTER INDIRETTO CON AUTOINCREMENTO	11	* 7+

I modi SIMBOLICO e INDICIZZATO presentando il primo bit del campo T alto (10), richiedono un'ulteriore parola (2 byte) contenuta in memoria.

Nel modo SIMBOLICO i quattro bit del campo S e D seguenti il campo T sono settati a zero.

Nel modo INDICIZZATO i quattro bit del campo S e D seguenti il campo T rappresentano il registro INDICE.

Il Registro 0 non può essere utilizzato per il modo INDICIZZATO.

1) Indirizzamento al WORKSPACE REGISTER

Questo è un modo diretto per cui il contenuto del WR è l'operando.

ESEMPIO:

MOV	R3,R7	Copia il contenuto del WORKSPACE REGISTER 3 nel W. REGISTER 7
COC	R15,R8	Compara il contenuto dei bit 1 del WR8 con quelli del WR15.

2) Indirizzamento al WORKSPACE REGISTER INDIRETTO

Questo è un modo indiretto per cui il contenuto del WR è l'indirizzo dell'operando, i registri R sono preceduti dal simbolo (*).

ESEMPIO:

A	* R6, * R2	Addiziona il contenuto dell'indirizzo indicato nel WR6 con il contenuto dell'indirizzo indicato nel WR2 e il risultato caricalo all'indirizzo indicato nel WR2
MOV	*R6, R0	Copia il contenuto dell'indirizzo indicato dal WR6 nel WR0

3) Indirizzamento SIMBOLICO

In questo modo l'indirizzo di memoria che contiene l'operando è un nome simbolico. L'indirizzo simbolico è preceduto da (@).

ESEMPIO:

S	@LAB, @CAR	Sottrae il contenuto della locazione di memoria LAB dal contenuto della locazione di memoria CAR e la differenza caricala alla locazione CAR
C	RO, @LEM	Compara il contenuto del WRO con il contenuto della locazione LEM
MOV	@12, @7C	Copia il contenuto della locazione >000C nella locazione >007C

4) Indirizzamento INDICIZZATO

In questo modo l'indirizzo è la somma del contenuto del WORKSPACE REGISTER e dell'indirizzo simbolico di memoria.

L'indirizzo è preceduto da (@) e seguito da un Registro R in parentesi.

ESEMPIO:

A @ 2(R7),R5	Addiziona il contenuto dell'indirizzo, ottenuto come somma di 2 più il contenuto del WR7, al contenuto del WR5
MOV R7, @CAR–6(R5)	Copia il contenuto del WR7 alla locazione di memoria ottenuta come somma del contenuto del WR5 e del valore contenuto alla locazione CAR-6

5) Indirizzamento al WR INDIRETTO CON AUTOINCREMENTO

In questo modo dopo che l'indirizzo è ottenuto dal WR, il WR è incrementato di 1 o 2 rispettivamente per istruzioni a uno o due byte — L'indirizzo è preceduto dall' * e seguito da +.

ESEMPIO:

S	* R4+,R1	Sottrae il contenuto dell'indirizzo indicato nel WR4 dal contenuto del WR1, poni il risultato nel WR1 ed incrementa l'indirizzo nel WR4 di due

Analizziamo gli altri tre modi di indirizzamento:

6) INDIRIZZAMENTO IMMEDIATO
7) INDIRIZZAMENTO AL PROGRAM COUNTER
8) INDIRIZZAMENTO AL CRU BIT

6) Indirizzamento IMMEDIATO

Nel modo immediato uno degli operandi è costituito dal valore, qualora l'istruzione sia ad un solo operando il valore del secondo operando è contenuto nella istruzione successiva.

ESEMPIO:

LI	R4, >2010	Carica nel WR4 il valore esadecimale >2010

7) Indirizzamento al P.C.

Questo modo d'indirizzamento è utilizzato solo nel caso di istruzioni di salto (JUMP).

ESEMPIO:

JMP	CAR	Salto incondizionato alla locazione CAR

8) Indirizzamento al CRU BIT

In questo modo l'istruzione può settare, resettare e testare i bit di CRU o trasferire dati fra memoria e CRU. L'indirizzo base sotware di CRU è contenuto nei bit (3-14) del WR12 e da questo è ricavato l'indirizzo hardware di CRU. Le istruzioni di CRU contengono una definita costante di spiazzamento (da −128 a 127) che è sommata all'indirizzo base del WR12.

ESEMPIO:

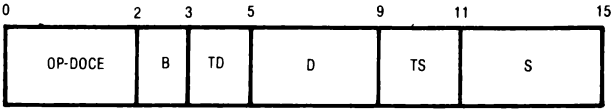
SBO	8	Pone il bit di CRU a 1 all'indirizzo di CRU ottenuto dalla somma dell'indirizzo base più la costante di spiazzamento 8

BIBLIOGRAFIA:

Manuale Editor/Assembler — Texas Instruments, Cap. IV.

4.3 ISTRUZIONI ASSEMBLER

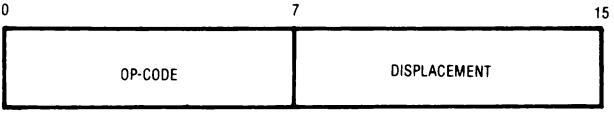
FORMATO I



In questo formato le istruzioni contengono due operandi separati da virgola, di cui il primo è l'operando SORGENTE e il secondo l'operando DESTINAZIONE. Appartengono a questo formato le seguenti istruzioni:

Cod. mnemonico	*Significato*
A	Add Words
AB	Add Bytes
C	Compare Words
CB	Compare Bytes
MOV	Move Word
MOVB	Move Byte
S	Subtract Words
SB	Subtract Bytes
SOC	Set Ones Corresponding
SOCB	Set Ones Corresponding, Byte
SZC	Set Zeros Corresponding
SEZCB	Set Zeros Corresponding,Byte

FORMATO II



In questo formato le istruzioni utilizzano l'indirizzamento al P.C. e la costante di spiazzamento. Appartengono a questo formato le istruzioni di salto (Jump):

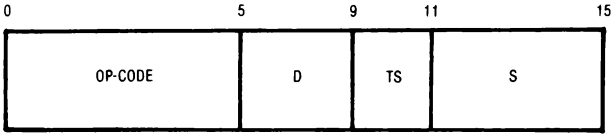
Cod. mnemonico	*Significato*
JEQ	JUMP if Equal
JGT	JUMP if Greater than
JH	JUMP if Logical High

JHE	JUMP	if High or Equal
JL	JUMP	if logical Low
JLE	JUMP	if Low or Equal
JLT	JUMP	if Less than
JMP	JUMP	inconditional
JNC	JUMP	if No Carry
JNE	JUMP	if Not Equal
JNO	JUMP	if NO Overflow
JOC	JUMP	On Carry
JOP	JUMP	if Odd Parity

Allo stesso formato appartengono anche le istruzioni che agiscono sul CRU bit INPUT/OUTPUT:

SBO	Set Bit to logic One
SBZ	Set Bit to logic Zero
TB	Test Bit

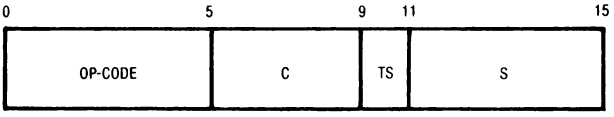
FORMATO III



In questo formato le istruzioni contegnono l'operando SORGENTE del tutto generale e l'operando DESTINAZIONE costituito da un WORKSPACE REGISTER. Appartengono a questo formato le istruzioni logiche:

COC	Compare Ones Corresponding
CZC	Compare Zeros Corresponding
XOR	Exclusive OR

FORMATO IV

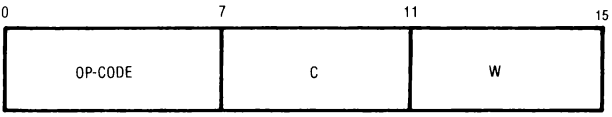


In questo formato le istruzioni contengono un indirizzo di memoria e un'espressione numerica. L'indirizzo di memoria rappresenta la locazione dove o da dove vengo-

no trasferiti i bit, l'espressione numerica rappresenta il numero di bit da trasferire, essa può valere da 0-15.

LDCR	Load CRU
STCR	Store CRU

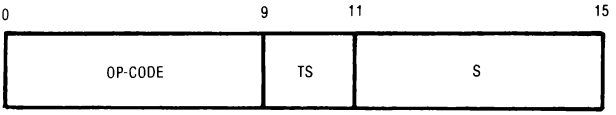
FORMATO V



In questo formato le istruzioni contengono un indirizzo di WORKSPACE REGISTER e un'espressione numerica. Il contenuto del WR è scorso di tante posizioni(bit) quante sono indicate nell'espressione numerica. Appartengono a questo formato le istruzioni di SHIFT:

SLA	Shift Left Arithmetic
SRA	Shift Right Arithmetic
SRC	Shift Right Circular
SRL	Shift Right Logical

FORMATO VI

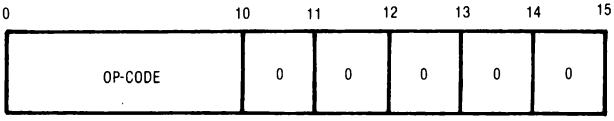


In questo formato le istruzioni contengono un solo operando:

Cod. mnemonico	*Significato*
ABS	ABSolute Value
B	Branch
BL	Branch and Link
BLWP	Branch and Load W. POINTER
CLR	Clear
DEC	DECrement
DECT	DECrement by Two
INC	INCrement
INCT	INCrement by TWO

INV	INVert
NEG	NEGate
SETO	Set to One
SWPB	Swap Bytes
X	Execute

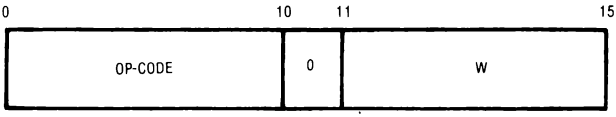
FORMATO VII



In questo formato le istruzioni non contengono operandi, sono istruzioni di controllo:

CKOF	Clock Off
CKON	Clock On
IDLE	IDLE
LREX	Load or Restart execution
RSET	Reset
RTWP	Return with W. Pointer

FORMATO VIII



In questo formato le istruzioni contengono l'indirizzo del WORKSPACE Register e l'operando immediato. Il W. Register rappresenta l'operando DESTINAZIONE e l'operando immediato il valore.

AI	Add Immediate
ANDI	And Immediate
CI	Compare Immediate
LI	Load Immediate
ORI	OR Immediate

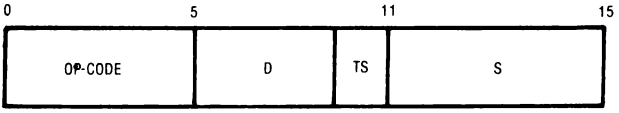
Due istruzioni particolari richiedono soltanto l'operando immediato:

LIMI	Load Interrupt Mask Immediate
LWPI	Load Workspace Pointer Immediate

Due istruzioni particolari richiedono soltanto l'indirizzo del W. Register:

STST	Store Status
STWP	Store Workspace Pointer

FORMATO IX



In questo formato l'istruzione contiene due operandi: l'indirizzo del moltiplicatore o divisore e l'indirizzo del W. Register che contiene il moltiplicando o il dividendo:

MPY	Multiply
DIV	Divide

Una particolare istruzione è:

XOP	Exended Operation

che contiene l'operando indirizzo e l'espressione numerica.

BIBLIOGRAFIA:
Manuale Editor/Assembler – Texas Instruments, Cap. V.

4.4 CONTEXT SWITCH

L'operazione di Context Switching è essenziale nella filosofia del TMS 9900, in quanto utilizza la capacità di questo processore di definire diverse aree di memoria esterna per allocare i suoi registri di lavoro (16 WORKSPACE REGISTER) e salvare lo status della CPU (PC, STATUS REGISTER, WORKSPACE POINTER) creando così la possibilità per ciascun programma utente di avere i propri registri. Questo è molto importante nella gestione degli Interrupt.

"Context" è l'equivalente della parola "ENVIRONMENT" per cui "Context switching" rappresenta un passaggio, cambio di controllo per la CPU, che può controllare un'altra area di lavoro (W. REGISTER) definendo un nuovo puntatore (W. POINTER) e un nuovo P. COUNTER e salvando lo status attuale (PC, STATUS, W.P.) che permetterà di ripristinare il processo interrotto.

Premesso che il Context switching può essere realizzato sia in modo HARDWARE (con i segnali LOAD, RESET, IC0 – IC3 che definiscono diversi livelli di Interrupt) sia in modo SOFTWARE (con le istruzioni XOP e BLWP), esso necessita in ogni modo di un'area di memoria detta TWO-WORD VECTOR che contenga gli indirizzi (PC) dei programmi da processare e gli indirizzi (WP) dei registri di lavoro (W. REGISTER).

La sequenza delle operazioni richieste dal "CONTEXT SWITCHING" è la seguente:

- Il nuovo WP è caricato in CPU.
- Il vecchio STATUS REGISTER è salvato nel WR15 della nuova area di lavoro.
- Il vecchio PC è salvato nel WR14 della nuova area di lavoro.
- Il vecchio WP è salvato nel WR13 della nuova area di lavoro.
- Il nuovo PC è caricato in CPU e l'esecuzione del nuovo programma inizia.

Quando l'esecuzione del nuovo programma termina, il ritorno (RTWP) al vecchio programma comporta il ricaricamento dello STATUS,PC,WP con i valori salvati nei WR15,WR14,WR13 che ripristinano il vecchio stato della CPU.

Un esempio di organizzazione di memoria (TWO-WORD VECTOR) per gestire il context è mostrato in Figura 1.

In Figura 1 il vettore TWO-WORD per le istruzioni XOP è memorizzato dall'indirizzo >0040 all'indirizzo >007E; volendo eseguire l'istruzione:

```
XOP DT,2.
```

L'OPERANDO 2 indica quale vettore utilizzare, nel nostro caso l'indirizzo >0360 è posto nel nuovo WP per individuare la nuova area di programma dei registri RO-R15,

e l'indirizzo >032E è posto nel nuovo PI per indicare la prossima istruzione da eseguire. (vedi Figura 2).

L'operando DT consente di salvare nel registro R11 della nuova W.A. l'indirizzo >0242, questo potrebbe essere utile per lo scambio di parametri fra più programmi.

RTWP effettua il ritorno alla vecchia area di lavoro (W.A.) (vedi Figura 3).

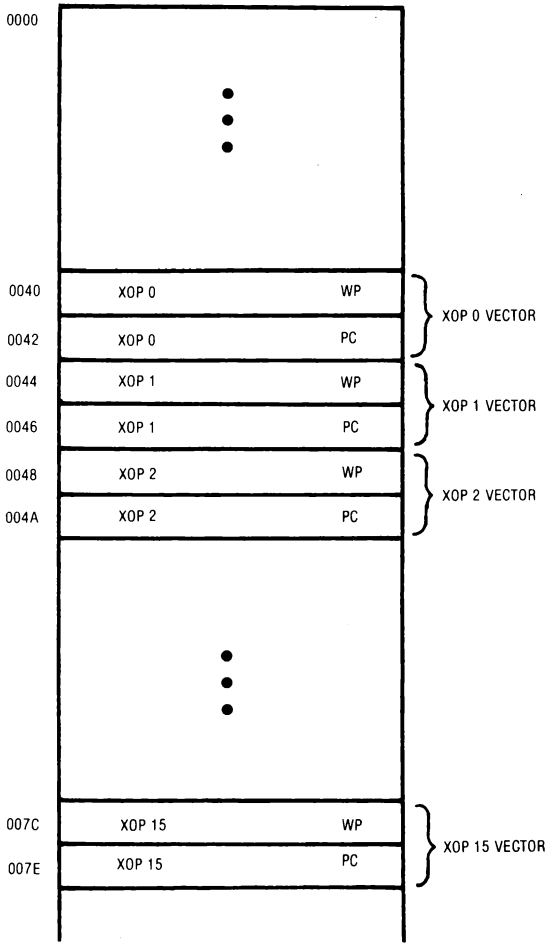


Figura 1

Memorizzazione del vettore TWO-WORD

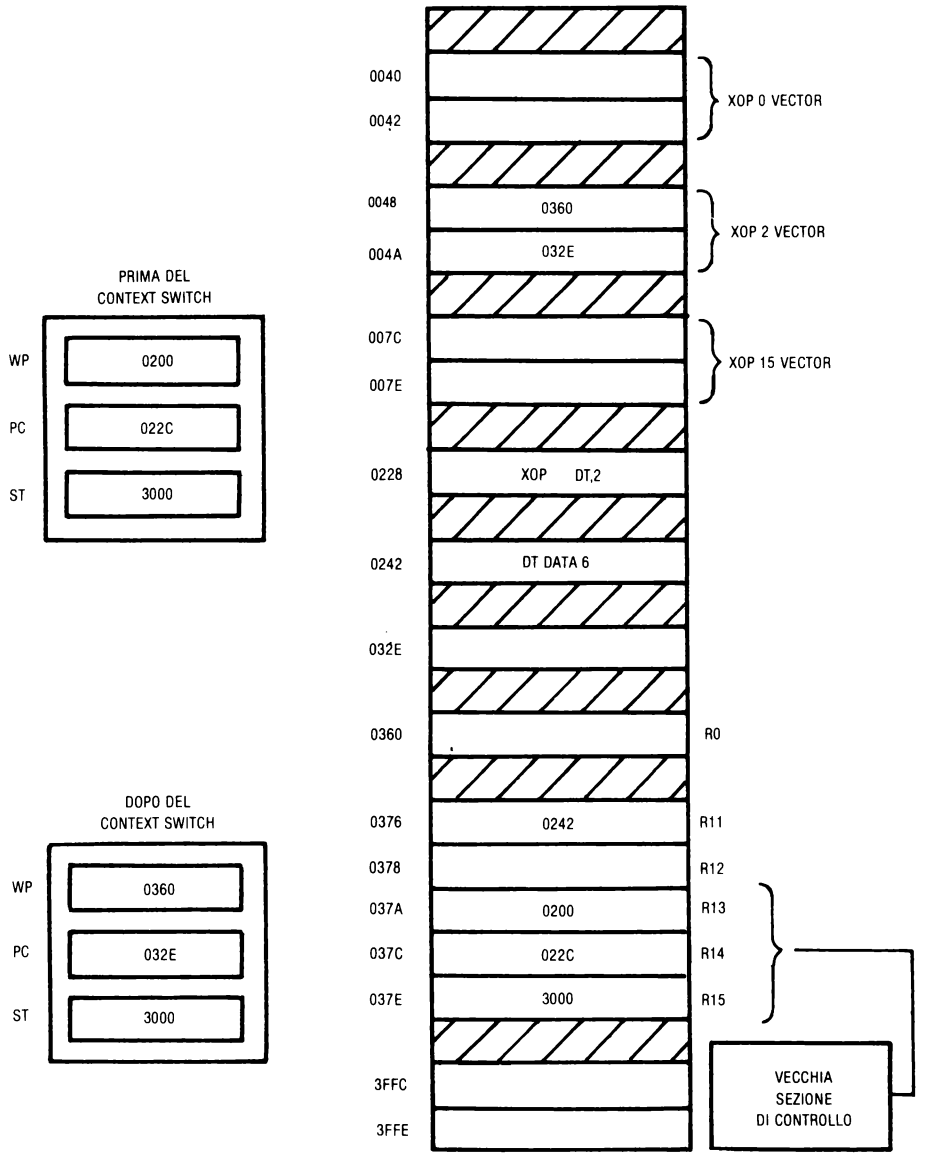


Figura 2

Context SWITCH con l'istruzione XOP DT, 2

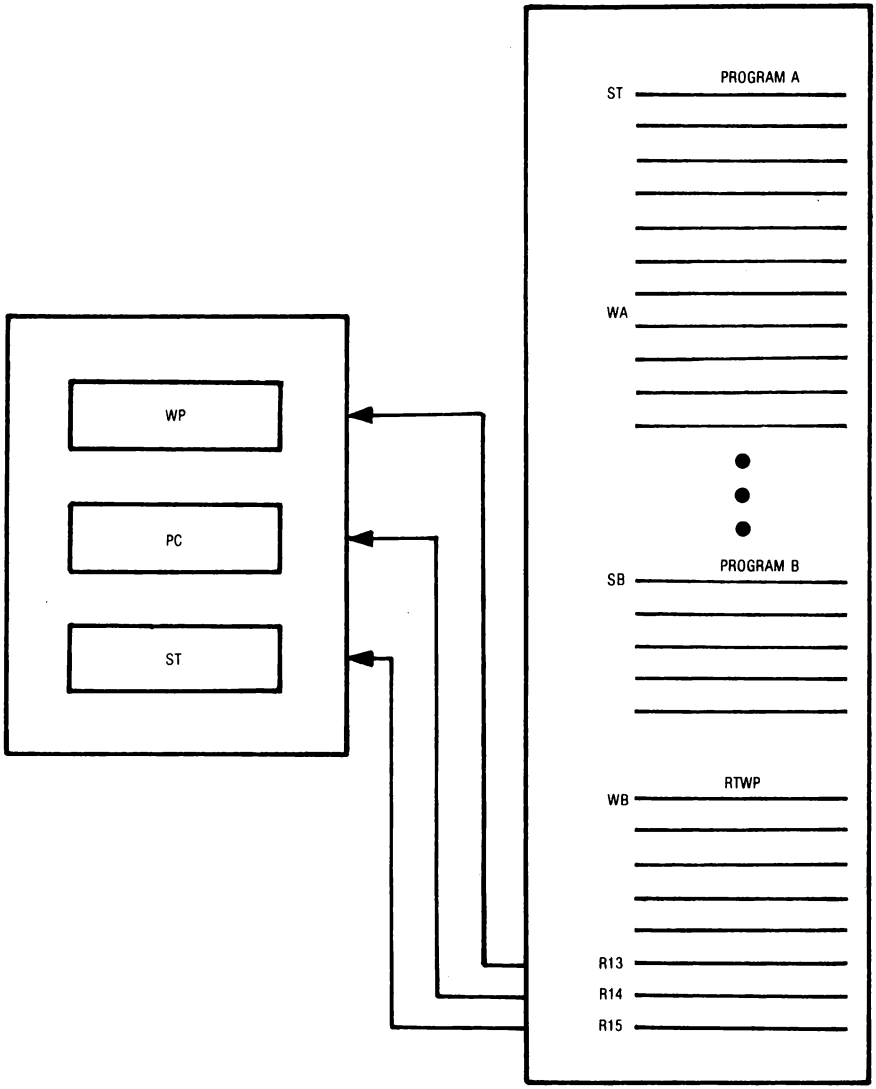


Figura 3

Ritorno da un'operazione di Context SWITCH:
RTWP Ripristina il vecchio stato.

BIBLIOGRAFIA:
«TM 990 Introduction To Microprocessors" – Texas Instruments, pag. 46-47, 285-288, 369-383

CAPITOLO V

PROGRAMMI PER IL TI 99/4A

5.1 ARCHIVIO PERSONALE

Questo programma vi consente di creare un archivio personale di 100 elementi, sia su nastro sia su disco.

Ciascun elemento può essere descritto con un massimo di 380 caratteri su nastro e 500 caratteri su disco; per ciascun elemento sono utilizzati due RECORD di lunghezza fissa, di tipo INTERNAL, SEQUENZIALI.

Mandate in esecuzione il programma e apparirà la seguente scritta:

NOME ARCHIVIO –CSI–
oppure –DSK1.nome–
va indicato fra apici
battete "DSK1.PROVA"
NO Elementi
battete 3

Selezione
1 – PER LA CREAZIONE
2 – PER LA LETTURA
3 – PER FINIRE

Selezionate l'opzione 1 e potete inserire i vostri elementi:

NOME .....
NOTA 1 .....
NOTA 2 .....
NOTA 3 .....
NOTA 4 .....

I campi alfanumerici NOME,NOTA 1,NOTA 2 costituiscono il primo RECORD, i campi alfanumerici NOTA 3 e NOTA 4 il secondo RECORD.

La linea 350 effettua il controllo della lunghezza solo del primo RECORD. Selezionate l'opzione 2 e potete ricercare i vostri elementi per nome; quando avete ultimate tutte le ricerche, battete 999 e ritornate alla selezione principale.

```
10 REM  ARCHIVIO PERSONALE
20 CALL CLEAR
30 DIM A$(100),B$(100),C$(100),D$(100),E$(100)
40 X$="                "
50 CALL CLEAR
60 PRINT "NOME ARCHIVIO  -CS1-  "::,"oppure -DSK1.NOME-"::
70 PRINT "Va indicato fra apici "::::
80 INPUT "   ":NOME$
90 PRINT X$,X$
100 PRINT "   No Elementi"::::
110 INPUT "   ":NM
120 IF NOME$="CS1" THEN 130 ELSE 150
130 LU=192
140 GOTO 160
150 LU=250
160 CALL CLEAR
170 PRINT TAB(9);"SELEZIONE  :"::::
180 PRINT "  1- PER LA CREAZIONE"::
190 PRINT "  2- PER LA LETTURA"::
200 PRINT "  3- PER FINIRE"::::
210 INPUT X$:XX
220 IF XX>3 THEN 160
230 CALL CLEAR
240 ON XX GOTO 250,500,740
250 FOR I=0 TO NM-1
260 PRINT X$
270 PRINT TAB(9);"ELEMENTO No ";I+1::::
280 INPUT "NOME    ":A$(I)
290 PRINT X$
300 INPUT "NOTA1 ":B$(I)
310 INPUT "NOTA2 ":C$(I)
320 INPUT "NOTA3 ":D$(I)
330 INPUT "NOTA4 ":E$(I)
340 LNGH=LEN(A$(I))+LEN(B$(I))+LEN(C$(I))
350 IF LNGH>LU-1 THEN 360 ELSE 410
360 PRINT " LUNGHEZZA ECCESSIVA ! MAX = ";LU;" e non ";LNGH
370 PRINT "  RIPETERE L'INSERIMENTO   "::::::
380 FOR T=1 TO 800
390 NEXT T
400 GOTO 280
410 NEXT I
420 OPEN #1:NOME$,SEQUENTIAL,FIXED LU,OUTPUT,INTERNAL
430 FOR I=0 TO NM-1
440 PRINT #1:A$(I),B$(I),C$(I)
450 PRINT #1:D$(I),E$(I)
460 NEXT I
470 CLOSE #1
480 CALL CLEAR
490 GOTO 170
500 OPEN #1:NOME$,SEQUENTIAL,FIXED LU,INTERNAL,INPUT
510 FOR I=0 TO NM-1
520 INPUT #1:A$(I),B$(I),C$(I)
530 INPUT #1:D$(I),E$(I)
540 NEXT I
550 CLOSE #1
560 CALL CLEAR
570 PRINT "  Nome o Codice di ricerca"::
580 PRINT "  999 Per Finire"::
590 INPUT "    ":Z$
```

```
600 IF Z$="999" THEN 160
610 F$="  ELEMENTO NON PRESENTE"
620 FOR I=0 TO NM-1
630 IF A$(I)=Z$ THEN 680
640 NEXT I
650 PRINT #1, X$,F$
660 FOR G=1 TO 500
670 NEXT G
680 CALL CLEAR
690 PRINT A$(I)::B$(I)::C$(I)::D$(I)::E$(I)::
700 PRINT X$,"Premi Enter per continuare"
710 INPUT X$:L$
720 CALL CLEAR
730 GOTO 560
740 END
```

Potete aggiungere facilmente altre opzioni e routine per l'aggiunta di elementi, per la stampa degli elementi, per la cancellazione ecc. Potete utilizzare anche dei campi numerici per descrivere in modo più diversificato i singoli elementi.

5.2 WORD PROCESSING

Configurazione richiesta: ESPANSIONE 32KRAM
UNITÀ A DISCHI
STAMPANTE
INTERFACCIA RS232
MODULO SSS ED/ASSEMBLER

Selezionando l'opzione Editor/Assembler appare sullo schermo l'indice principale di selezione:

```
    Press
1 TO EDIT
2    ASSEMBLER
3    LOAD AND RUN
4    RUN
5    RUN PROGRAM FILE
```

Inserite nell'unità a dischi il dischetto Parte A dell'Editor/Assembler e selezionate l'opzione 1 TO EDIT, appare il seguente indice:

```
  * EDITOR *
    press
1 TO LOAD
2    EDIT
3    SAVE
4    PRINT
5    PURGE
```

Selezionate l'opzione 2 EDIT e il computer caricherà dal dischetto Parte A il programma EDIT 1 e sullo schermo appare il cursore che predispone l'inizio al file da inserire su 80 colonne e l'istruzione * EOF (fine del file, END OF FILE) che indica l'ultima riga del file quando deve essere chiuso.

Portate con il cursore I FCTN X la linea * EOF a fine pagina del video e inserite il testo che desiderate avvalendovi delle seguenti funzioni svolte dai tasti:

FCTN 4	roll-up	Scrollo verticale di 24 linee in avanti
FCTN 6	roll-down	Scrollo verticale di 24 linee indietro
FCTN 5	next-window	Scrollo orizzontale
FCTN 7	tab	Muove il cursore in funzione di una tabellazione assegnata, per default utilizza le colonne 1,8,13,26,31,46,60 e 80.
FCTN 8	insert line	inserisce una linea

FCTN 9	esc	permette di retrocedere di una pagina di schermo e di passare dal MODO EDIT al MODO COMANDI
FCTN S	left-arrow	sposta il cursore a sinistra di un carattere
FCTN D	right-arrow	sposta il cursore a destra di un carattere
FCTN X	down-arrow	sposta il cursore in giù di una linea
FCTN E	up-arrow	sposta il cursore in su di una linea
ENTER	return	invia le informazioni e i dati dopo che sono stati digitati sulla tastiera
FCTN 1	del	cancella un carattere
FCTN 2	ins	inserisce un carattere
FCTN 3	erase	cancella l'intera linea
FCTN =	quit	per lasciare l'EDITOR/ASSEMBLER

Utilizzando i tasti FCTN 9(Escape) potete passare dal Modo EDIT al Modo COMANDI che prevede queste funzioni:

EDIT, FIND, REPLACE, MOVE, INSERT, COPY, SHOW, DELETE, ADJUST, TAB, HOME.

Premesso che ciascun comando può essere digitato soltanto con la lettera iniziale (E,F,R,M,I,C,D,A,T,H,) analizziamo le funzioni svolte dai rispettivi comandi:

EDIT

permette di ritornare al Modo EDIT.

FIND

permette di trovare una stringa in una riga e colonna assegnate; il formato richiesto è:

FIND <CNT> (<COL,COL>)/STRING/

dove CNT rappresenta il numero di volte che è incontrata la stringa e varia da 1 a 9999; COL le colonne dove ricercare la stringa e varia da 1 a 80; STRING la stringa da ricercare.

REPLACE

permette di cambiare la stringa desiderata, è richiesto il seguente formato:

REPLACE<V,> <CNT> (<COL,COL>)/OLD/NEW/

dove CNT rappresenta il numero di volte che è incontrata la stringa; COL le colonne dove ricercare la stringa; OLD la stringa da sostituire; NEW la nuova stringa; se specificate V il computer vi chiede la verifica

REPLACE STRING(Y/N/A)

Y per cambiare la stringa, N per non effettuare la sostituzione, A per cambiare la stringa tutte le volte che è incontrata.

MOVE

permette di spostare le linee del testo. È richiesto il seguente formato:

MOVE START LINE, STOP LINE, AFTER LINE

START LINE rappresenta la prima linea del blocco da spostare; STOP LINE l'ultima linea del blocco da spostare; AFTER LINE rappresenta il numero di linea dopo il quale va trasferito il blocco di linee. Dopo lo spostamento delle linee il computer automaticamente rinumera i numeri di linea. La linea EOF è rappresentata con la lettera E.

INSERT

permette di inserire le linee del testo o l'intero file presente su dischetto a partire da un numero di linea specificato nel file aperto dall'EDIT. È richiesto il seguente formato:

INSERT BEFORE LINE, FILE NAME

BEFORE LINE rappresenta il numero di linee prima del quale è inserito il file specificato dal DSK1.nome

ESEMPIO:

20,DSK1. POESIE

inserisce nel file aperto dall'EDIT prima della linea 20 il File POESIE memorizzato sul dischetto dell'unità a dischi 1.

COPY

permette di copiare le linee del testo, è utilizzato lo stesso formato del MOVE, con la differenza che non cancella le linee copiate.

SHOW

permette di leggere le linee di testo a partire da un numero di linea specificato. È richiesto il seguente formato:

SHOW LINE

LINE rappresenta il numero di linea dal quale inizia la visualizzazione del testo.

DELETE

permette di cancellare parte del testo. È richiesto il seguente formato:

```
DELETE START LINE, STOP LINE
```

START LINE rappresenta la prima linea e STOP LINE l'ultima linea da cancellare.

ADJUST

permette di ritornare al Modo EDIT.

TAB

permette di modificare i valori di tabellazione del cursore, per default sono inserite le seguenti colonne: 1,8,13,26,31,46,60,80.

HOME

permette di ritornare al Modo EDIT.

Il tasto ESCAPE permette di ritornare alle pagine precedenti degli indici di selezione.

Se volete ritornare al file aperto battete il comando HOME o EDIT che permettono di uscire dal Modo COMANDI.

Dopo aver inserito il testo o qualsivoglia modello di documento nel file aperto dall'EDIT, togliete il dischetto Parte A dall'unità a dischi e inserite il dischetto dati da voi precedentemente formattato (inizializzazione a 40 tracce).

Con il tasto ESCAPE ritornate all'indice di selezione e scegliete l'opzione 3 SAVE che vi consente di salvare sul disco il vostro file appena creato:

```
    SAVE
VAR 80 FORMAT(Y/N)?
    N
```

Rispondete N per il formato fisso dei Record a 80 caratteri, Y per il formato variabile.

```
FILE NAME?
DSK1.nome
```

Assegnate un nome al file che memorizzate sul dischetto dati inserito nell'unità a dischi 1.

Qualora vogliate stampare il file creato e memorizzato su disco, selezionate l'opzione 4 PRINT:

```
    PRINT
FILE NAME?
DSK1.nome
```

Richiesta del nome del file da stampare memorizzato sul dischetto inserito nell'unità a dischi 1.

```
DEVICE NAME ?
RS232.BA=9600
```

Richiesta del nome del dispositivo di stampa; in questo caso è utilizzata la stampante RS232 con velocità di trasmissione a 9600 Baud.

In seguito, in sostituzione all'opzione PRINT, è fornito un programma in BASIC ESTESO che effettua la stampa del testo o file nelle copie da voi desiderate.

Se selezionate l'opzione 5 PURGE, potete cancellare il file in memoria:

```
    PURGE
ARE YOU SURE(Y/N)?
```

Siete sicuri di voler cancellare ?
Y per il sì, N per il NO.

L'opzione 1 TO LOAD vi consente di caricare in memoria i file che avete memorizzato sul vostro dischetto dati:

```
    LOAD
FILE NAME ?
DSK1.nome
```

Richiesta del nome del file che volete leggere sul dischetto dati inserito nell'unità a dischi 1.

(Questo paragrafo è tratto dal manuale "Editor/Assembler")

BIBLIOGRAFIA:

Manuale "Editor/Assembler" – Texas Instruments, Cap. II

Questo programma sostituisce l'opzione PRINT analizzata precedentemente e consente di stampare la documentazione nelle copie desiderate e controlla la capacità del buffer della stampante RS232 settata a 1200 Baud. Infatti alla linea 115 è definito il sottoprogramma CHIUDI che prima chiude il file 2 e poi lo riapre.

```
10 CALL CLEAR
15 REM *****DOCUMENTI******
20 DISPLAY AT(3,5):"******STAMPA*******"
25 DISPLAY AT(6,3):"NOME DOCUMENTAZIONE :"
30 ACCEPT AT(8,3):NOME$
35 DISPLAY AT(12,3):"NUMERO COPIE :"
40 ACCEPT AT(12,20):NN
45 OPEN #2:"RS232.BA=1200",OUTPUT
50 FOR I=1 TO NN
55 OPEN #1:SEG$(NOME$,1,15),INPUT ,FIXED 80
60 R=1
65 LINPUT #1:PAR$
70 PRINT #2:PAR$
75 R=R+1
80 IF R)26 THEN CALL CHIUDI
85 IF EOF(1)THEN 90 ELSE 65
90 DISPLAY AT(18,6):"***";I;"***"
95 CLOSE #1
100 NEXT I
105 CLOSE #2
110 END
115 SUB CHIUDI
120 CLOSE #2
125 R=1
130 OPEN #2:"RS232.BA=1200",OUTPUT
135 SUBEND
```

La variabile R controlla le linee del testo e la linea 85 effettua il controllo della FINE DEL FILE.

Il programma richiede il modulo del BASIC ESTESO; dopo aver inserito il dischetto con i file della documentazione, mandate in esecuzione con RUN questo programma e apparirà:

*** STAMPA ***
NOME DOCUMENTAZIONE:

battete

DSK1.POESIE

NUMERO COPIE:

battete

2

In questo esempio è richiesta la doppia copia dei file POESIE memorizzato nell'unità a dischi 1.

Esempio di stampa ottenuta su 80 colonne su carta normale:

```
--------------------------------------------------------------------------------
RACCOLTA   DI POESIE E PENSIERI    DATA       EDIZIONE       AUTORE       EDITORE
--------------------------------------------------------------------------------
LA MUSICA E' LA MIGLIORE CONSOLAZIONE GIA' PER IL FATTO CHE
NON CREA NUOVE PAROLE .
ANCHE QUANDO ACCOMPAGNA DELLE PAROLE,LA SUA MAGIA PREVALE ED ELIMINA
IL PERICOLO DELLE PAROLE.MA IL SUO STATO PIU' PURO E' QUANDO RISUONA DA SOLA.
LE SI CREDE SENZA RISERVE,POICHE' CIO' CHE AFFERMA RIGUARDA I SENTIMENTI.
IL SUO FLUIRE E' PIU' LIBERO DI QUALSIASI ALTRA COSA CHE SEMBRI UMANAMENTE
POSSIBILE,E QUESTA LIBERTA' REDIME.
QUANDO PIU' FITTAMENTE LA TERRA SI POPOLA,E QUANTO PIU' MECCANICO DIVENTA
IL MODO DI VIVERE,TANTO PIU' INDISPENSABILE DEVE DIVENTARE LA MUSICA.
VERRA' UN GIORNO IN CUI ESSA SOLTANTO PERMETTERA' DI SFUGGIRE ALLE STRETTE
MAGLIE DELLE FUNZIONI,E CONSERVARLA COME POSSENTE ED INTATTO SERBATOIO DI
LIBERTA' DOVRA' ESSERE IL COMPITO PIU' IMPORTANTE DELLA VITA INTELLETTUALE
FUTURA.
LA MUSICA E' LA VERA STORIA VIVENTE DELL' UMANITA',DI CUI ALTRIMENTI POSSEDIAMO
SOLO PARTI MORTE.
NON C'E' BISOGNO DI ATTINGERVI ,POICHE' ESISTE GIA' DA SEMPRE IN NOI,E BASTA
SEMPLICEMENTE ASCOLTARE,PERCHE' ALTRIMENTI SI STUDIA INVANO.

                                                          Elias Canetti

da  La Provincia dell'uomo

                                        1972 edizione ADELPHI


SPESSO IL MALE DI VIVERE HO INCONTRATO:
ERA IL RIVO STROZZATO CHE GORGOGLIA,
ERA L'INCARTOCCIARSI DELLA FOGLIA
RIARSA,ERA IL CAVALLO STRAMAZZATO.


BENE NON SEPPI,FUORI DEL PRODIGIO
CHE SCHIUDE LA DIVINA INDIFFERENZA:
ERA LA STATUA NELLA SONNOLENZA
DEL MERIGGIO,E LA NUVOLA,
```

E IL FALCO ALTO LEVATO.

Eugenio Montale

da Ossi di seppia

1971 edizione EINAUDI

Un ulteriore esempio è dato dalla stampa di un listino prezzi

* * * STAMPA * * *
NOME DOCUMENTAZIONE:
DSK1.LISTINO
NUMERO COPIE:
3

*******LISTINO PREZZI******** VALEVOLE DAL 1-3-83

TEXAS INSTRUMENTS

HARDWARE			
CONSOLLE TI 99/4A		PHC004	399.000
PERIPHERAL BOX		PHP1200	420.000
DISK CONTROLLER	CARD	PHP1240	350.000
DISK DRIVE	CARD	PHP1250	680.000
JOYSTICKS		PHP1100	52.000
RAM EXPANSION	CARD	PHP1260	270.000
RS232	CARD	PHP1220	280.000
P.CODE	CARD	PHP1270	450.000
SINTETIZZATORE VOCALE		PHP1500	220.000
CASSETTE CABLE		PHA2000	25.000
MONITOR CABLE		PHA2010	25.000
SOFTWARE			
SOCCER		PHM3024	59.000
SPEECH EDITOR		PHM3011	48.000
STATISTIC		PHM3014	120.000
TERMINAL EMULATOR		PHM3035	120.000
VIDEO GAMES 1		PHM3018	59.000
VIDEO GAMES 2		PHM3025	36.000
YATHZEE		PHM3039	59.000
ZEROZAP		PHM3036	59.000
TOMBSTONE CITY		PHM3052	59.000
OTHELLO		PHM3067	69.000
CAR WARS		PHM3054	69.000
TI INVADERS		PHM3053	39.000
EDITOR ASSEMBLER		PHM3055	215.000
AMAZING		PHM3030	59.000
ADD-SUB 1		PHM3027	69.000
ADD-SUB 2		PHM3028	69.000
ATTACK		PHM3031	59.000
BEGINNER GRAMMAR		PHM3003	48.000
BLASTO		PHM3032	59.000
SCACCHI		PHM3008	96.000
CONNECT FOUR		PHM3038	59.000
DEMO		PHM3031	49.000
DIAGNOSTIC		PHM3000	24.000

DISK MANAGER	PHM3019	89.000
EARLY READING	PHM3015	69.000
EXTENDED BASIC	PHM3026	215.000
HUSTLE	PHM3024	59.000
TI LOGO	PHM3040	340.000
MULTIPLICATION 1	PHM3029	69.000
MUSIC MAKER	PHM3020	69.000
P.R.G.	PHM3044	120.000
P.R.K.	PHM3013	120.000
ADVENTURELAND	PHT6046	59.000
MISSION IMPOSSIBLE	PHT6047	59.000
VOODOO CASTLE	PHT6048	59.000
THE COUNT	PHT6049	59.000

5.3 ISTOGRAMMI

Questo programma vi consente di avere graficamente sugli assi ortogonali X e Y l'istogramma delle quantità digitate; si possono inserire un massimo di 25 dati; l'unità di misura è calcolata in rapporto alle quantità digitate.

```
10 REM  ISTOGRAMMA
20 RANDOMIZE
30 CALL CHAR(129,"FF03030303030303")
40 CALL CHAR(130,"FF18181818181818")
50 DIM DATI(25)
60 CALL CLEAR
70 CALL CHAR(144,"007E7E7E7E7E7E7E")
80 CALL COLOR(15,7,3)
90 FOR C=4 TO 28
100 H=RND*13+2
110 CALL VCHAR(H,C,144,20-H)
120 CALL SOUND(10,RND*2000+400,2)
130 NEXT C
140 PRINT "    G  R  A  F  I  C  I"
150 FOR C=0 TO 300
160 NEXT C
170 FOR C=0 TO 100
180 NEXT C
190 CALL CLEAR
200 CALL COLOR(5,2,1)
210 CALL COLOR(6,2,1)
220 CALL COLOR(7,2,1)
230 PRINT "  UN MOMENTO , PREGO"
240 PRINT
250 CALL CHAR(145,"00007E7E7E7E7E7E")
260 CALL CHAR(146,"0000007E7E7E7E7E")
270 CALL CHAR(147,"000000007E7E7E7E")
280 CALL CHAR(148,"00000000007E7E7E")
290 CALL CHAR(149,"0000000000007E7E")
300 CALL CHAR(150,"000000000000007E")
310 CALL CHAR(151,"0000000000000000")
320 FOR C=0 TO 100
330 NEXT C
340 CALL CLEAR
350 INPUT "QUANTI DATI (MAX 25) ? ":NUMERO
360 PRINT
370 PRINT NUMERO;" DATI":::
380 FOR C=0 TO 400
390 NEXT C
400 FOR C=1 TO NUMERO
410 PRINT C;"- DATO  ";
420 INPUT "   ":DATI(C)
430 NEXT C
440 FOR C=2 TO 300
450 NEXT C
460 FOR C=1 TO NUMERO
470 IF MAGGIORE)DATI(C)THEN 490
480 MAGGIORE=DATI(C)
490 NEXT C
500 COST=MAGGIORE/20
510 CALL CLEAR
520 CALL SCREEN(16)
530 PRINT TAB(5);"UNITA' :";CHR$(144);" = ";COST
540 CALL COLOR(15,11,1)
550 CALL HCHAR(22,3,130,25)
560 CALL VCHAR(1,3,129,22)
570 FOR C=4 TO NUMERO+3
580 H=DATI(C-3)/COST
590 H1=INT(((H-INT(H))*10))
600 H1=(H1/5)*4
610 H1=H1+1
620 H1=INT(H1)
```

```
630 ON H1 GOSUB 700,720,740,760,780,800,820,840
640 CALL VCHAR(22-INT(H),C,144,INT(H))
650 CALL VCHAR(21-INT(H),C,Q,1)
660 NEXT C
670 CALL KEY(1,A,B)
680 IF B=0 THEN 670
690 END
700 Q=151
710 RETURN
720 Q=150
730 RETURN
740 Q=149
750 RETURN
760 Q=148
770 RETURN
780 Q=147
790 RETURN
800 Q=146
810 RETURN
820 Q=145
830 RETURN
840 Q=144
850 RETURN
```

5.4 COSTRUZIONE CARATTERI

Questo programma vi consente di rappresentare su di un reticolo 8×8 i caratteri da voi definiti ed avere la "PATTERN" in codice esadecimale da utilizzare nel sotto-programma CALL CHAR (Cod, "PATTERN").

```
10 REM   COSTRUZIONE CARATTERI
20 DIM B(8,8)
30 CALL CHAR(100,"")
40 CALL CHAR(101,"FFFFFFFFFFFFFFFF")
50 CALL COLOR(9,2,16)
60 CALL CLEAR
70 LIN$="CODICE DEI CARATTERI PROPRI"
80 Y=3
90 X=4
100 GOSUB 630
110 LIN$="12345678"
120 Y=8
130 GOSUB 630
140 GOSUB 680
150 LIN$="0=BASSO=BIANCO"
160 Y=22
170 X=4
180 GOSUB 630
190 LIN$="1=ALTO=NERO "
200 Y=23
210 GOSUB 630
220 FOR R=1 TO 8
230 CALL HCHAR(8+R,5,100,8)
240 NEXT R
250 FOR R=1 TO 8
260 FOR C=1 TO 8
270 CALL HCHAR(8+R,4+C,30)
280 CALL KEY(0,K,ST)
290 IF ST=0 THEN 280
300 IF (K()8)+(K()9)(=-2 THEN 330
310 GOSUB 730
320 GOTO 270
330 K=K-48
340 IF (K(0)+(K)1)(=-1 THEN 280
350 B(R,C)=K
360 CALL HCHAR(8+R,4+C,100+K)
370 NEXT C
380 NEXT R
390 COD$="0123456789ABCDEF"
400 LIN$=""
410 FOR R=1 TO 8
420 BAS=B(R,5)*8-B(R,6)*4+B(R,7)*2+B(R,8)+1
430 ALT=B(R,1)*8+B(R,2)*4+B(R,3)*2+B(R,4)+1
440 LIN$=LIN$&SEG$(COD$,ALT,1)&SEG$(COD$,BAS,1)
450 NEXT R
460 Y=16
470 X=12
480 GOSUB 630
490 LIN$="BATTI F PER FINIRE"
500 Y=18
510 X=12
520 GOSUB 630
530 LIN$="UN TASTO QUALSIASI"
540 Y=19
550 GOSUB 630
560 LIN$="PER CONTINUARE "
570 Y=20
580 GOSUB 630
590 CALL KEY(0,K,ST)
600 IF ST=0 THEN 590
610 IF K()70 THEN 50
620 STOP
630 FOR I=1 TO LEN(LIN$)
```

```
640 CODE=ASC(SEG$(LIN$,I,1))
650 CALL HCHAR(Y,X+I,CODE)
660 NEXT I
670 RETURN
680 FOR I=1 TO LEN(LIN$)
690 CODE=ASC(SEG$(LIN$,I,1))
700 CALL HCHAR(Y+I,X,CODE)
710 NEXT I
720 RETURN
730 CALL HCHAR(8+R,4+C,100+B(R,C))
740 IF K=9 THEN 820
750 C=C-1
760 IF C<>0 THEN 880
770 C=8
780 R=R-1
790 IF R<>0 THEN 880
800 R=8
810 GOTO 880
820 C=C+1
830 IF C<>9 THEN 880
840 C=1
850 R=R+1
860 IF R<>9 THEN 880
870 R=1
880 RETURN
```

5.5 COSTRUZIONE FIGURE

Questo programma vi consente di rappresentare sullo schermo rettangolare i disegni ottenuti dal movimento dei Joystic; tenete alzato il tasto ALPHA LOCK.

Potete cambiare i colori ai disegni pigiando il tasto del fuoco del Joystic.

```
10 REM  DISEGNI CON I JOYSTIC
20 REM  UTILIZZO DEL PULSANTE DI FUOCO PER CAMBIARE COLORE
30 KK=1
40 X=15
50 Y=13
60 CALL SCREEN(3)
70 CALL CLEAR
80 CALL COLOR(1,9,9)
90 CALL COLOR(2,3,3)
100 CALL JOYST(1,DX,DY)
110 CALL KEY(1,KEY,STATUS)
120 IF KEY<>18 THEN 170
130 KK=-KK
140 CALL COLOR(2,14,14)
150 IF KK=-1 THEN 170
160 CALL COLOR(2,12,12)
170 X=X+DX/4
180 Y=Y-DY/4
190 X=INT(32*((X-1)/32-INT((X-1)/32)))+1
200 Y=INT(24*((Y-1)/24-INT((Y-1)/24)))+1
210 CALL HCHAR(Y,X,42)
220 GOTO 100
```

5.6 GENERATORE DI SUONI

Questo programma vi consente di comporre dei brani musicali spostando i Joystic in tutte le direzioni. Le note suonate vengono visualizzate sullo schermo.

Tenete alzato il tasto ALPHA LOCK.

```
10 REM    SUONI CON I JOYSTICK
15 REM   LA NOTA SUONATA E' VISUALIZZATA
20 A$="DOREMIFASOLASI"
25 DIM NOTE(9),M(8,8)
30 FOR I=1 TO 9
35 READ NOTE(I)
40 NEXT I
45 DATA 262,294,330,349,392,440,494,524,40000
50 FOR I=1 TO 9
55 READ X,Y,INDICE
60 M(X,Y)=INDICE
65 NEXT I
70 DATA 8,4,1,8,8,2,4,8,3
75 DATA 0,8,4,0,4,5,0,0,6
80 DATA 4,0,7,8,0,8,4,4,9
85 CALL JOYST(1,X1,Y1)
90 CALL JOYST(2,X2,Y2)
95 X1=X1+4
100 Y1=Y1+4
105 X2=X2+4
110 Y2=Y2+4
115 CALL SOUND(-1000,NOTE(M(X1,Y1)),0,NOTE(M(X2,Y2)),0)
120 PRINT SEG$(A$,M(X1,Y1),2);SEG$(A$,M(X2,Y2),2)
125 GOTO 85
```

5.7 BILANCIO FAMILIARE

Questo programma, in una forma molto semplice, vi consente di effettuare il bilancio familiare mensile, tenendo conto dei parametri più consueti di una famiglia media.

```
90 REM  **CONTABILITA' FAMILIARE **
100 CALL CLEAR
101 CALL SCREEN(3)
110 PRINT "  CONTABILITA'  FAMILIARE  ":::::::
120 FOR I=1 TO 500
130 NEXT I
140 CALL CLEAR
150 INPUT "NOME FAMIGLIA - ":A$
155 PRINT :::::
160 INPUT "NUCLEO FAMILIARE NUM. ":B$
165 PRINT :::
166 FOR I=1 TO 500
167 NEXT I
170 CALL CLEAR
180 PRINT "     ENTRATE   MENSILI     ":C$:::
190 INPUT "STIPENDI L. ":D
200 PRINT :::
210 INPUT "REDDITI DA IMMOB. L. ":E
215 PRINT :::
220 INPUT "VARIE L. ":F
225 FOR I=1 TO 900
227 NEXT I
230 CALL CLEAR
240 X=D+E+F
245 PRINT "REDDITO COMPLESSIVO MENSILE"::"      LIRE",X
246 FOR I=1 TO 900
247 NEXT I
250 CALL CLEAR
255 CALL SCREEN(11)
260 PRINT "     USCITE    MENSILI     ":C$:::
270 INPUT "COSTO CASA L. ":G
280 PRINT :
290 INPUT "MANUTENZIONE CASA L. ":H
300 PRINT :
310 INPUT "VITTO L. ":I
320 PRINT :
330 INPUT "ABBIGLIAMENTO L. ":L
340 PRINT :
350 INPUT "COSTO AUTO L. ":M
360 PRINT :
370 INPUT "HOBBY L. ":N
380 PRINT :
390 INPUT "VARIE L. ":O
400 Y=G+H+I+L+M+N+O
410 CALL CLEAR
420 PRINT " SPESE COMPLESSIVE MENSILI ":::"    LIRE ",Y
425 FOR I=1 TO 1000
430 NEXT I
440 CALL CLEAR
441 CALL SCREEN(4)
450 Z=X-Y
460 IF Z>0 THEN 510
470 PRINT "QUESTO MESE HAI SPESO TROP "
472 PRINT "PO IN QUANTO LE USCITE SUPE"
473 PRINT "RANO LE ENTRATE":::
475 PRINT "    LIRE ";Z::::
480 PRINT ::::
490 FOR I=1 TO 800
500 NEXT I
505 GOTO 580
510 PRINT "QUESTO MESE HAI RISPARMIATO":"       LIRE ",Z::
511 PRINT ::
```

```
512 Q$="NO"
520 INPUT "VUOI INVESTIRE IL RISPARMIO          SI       NO      ?":Q$
530 IF Q$="NO" THEN 578
540 CALL CLEAR
550 PRINT "VALUTANDO IL 10% DI INTERES-SE PUOI RICAVARE :          "::::
560 Z=Z+Z*10/100
570 PRINT "       LIRE ",Z::
575 FOR I=1 TO 500
577 NEXT I
578 CALL CLEAR
580 PRINT "FAMIGLIA ",A$::
590 PRINT "REDDITO MENSILE LIRE ";X::
600 PRINT "SPESE MENSILI LIRE ";Y::
610 PRINT "UTILE  LIRE  ";Z
620 FOR I=1 TO 3000
630 NEXT I
640 CALL CLEAR
650 K$="SI"
660 INPUT "      ANCORA ? (SI/NO) ":K$
680 IF K$="SI" THEN 100
690 END
```

5.8 DIETE

Questo programma vi consente di calcolare le calorie di un pasto, tenendo conto delle calorie spese nelle attività quotidiane da ciascuno di voi con la finalità di una dieta equilibrata.

```
10 REM  **DIETA EQUILIBRATA **
20 CALL CLEAR
30 PRINT "    DIETA   EQUILIBRATA    "::::::::
40 FOR I=1 TO 500
50 NEXT I
60 CALL CLEAR
70 PRINT " UNA DIETA E' EQUILIBRATA"::
80 PRINT " QUANDO FORNISCE TUTTI GLI"::
90 PRINT " ELEMENTI ESSENZIALI:"::
100 PRINT " PROTEINE, VITAMINE, MINERA "::
110 PRINT " LI, CARBOIDRATI,GRASSI "::" NELLE QUANTITA' SUFFICIENTI "
120 FOR I=1 TO 1000
130 NEXT I
140 CALL CLEAR
150 PRINT " UN UOMO CONSUMA MEDIAMENTE "::
160 PRINT " DALLE 2500/3000 CALORIE AL"::" GIORNO,UNA DONNA NE CONSU"::
170 PRINT " MA 2300/2600 ":::
180 FOR I=1 TO 700
190 NEXT I
200 CALL CLEAR
210 PRINT " ENERGIA MEDIA CONSUMATA NELLE SEGUENTI ATTIVITA'      (CALORIE /
1h):":::
220 PRINT " - DORMIRE =  65":" - STARE IN PIEDI = 120":" - GUIDARE = 170":"
 - CAMMINARE = 320":
230 PRINT " - CORRERE = 600":" - BALLARE = 320":" - SPOLVERARE = 190":"
 - RIFARE  IL LETTO = 300":
240 PRINT " - SCAVARE = 350":" - STARE SEDUTI = 90"::
250 INPUT "QUANTE CALORIE CONSUMI AL DI":A
260 CALL CLEAR
270 PRINT "    COSTITUZIONE  PASTO    "::
280 PRINT."    N.RO  CALORIE/100gr    "::::
290 INPUT "- PANE/PASTA 350/100gr ":B
300 INPUT "- VEGETALI 25/100gr ":C
310 INPUT "- CARNE 150/100gr ":D
320 INPUT "- PESCE 110/100gr ":E
330 INPUT "- FORMAGGI 300/100gr ":F
340 INPUT "- LEGUMI 350/100gr ":G
350 INPUT "- GRASSI 900/100gr ":H
360 INPUT "- FRUTTA F 60/100gr ":I
370 INPUT "- DOLCI 400/100gr ":L
380 INPUT "- LATTE 60/100gr ":M
390 INPUT "- UOVA 148/(2) ":N
400 INPUT "- CIOCCOLATO 550/100gr ":O
410 CALL CLEAR
420 P=B+C+D+E+F+G+H+I+L+M+N+O
430 PRINT "CALORIE PASTO",P::
440 PRINT  "CALORIE SPESE",A::
450 X=A-P
460 PRINT "CONFRONTO FRA CALORIE SPESE E CALORIE INGERITE =";X:::::
470 END
```

5.9 GIOCO I

Questo programma 5.9 vi consente di sperimentare tutti i sottoprogrammi per i suoni e giochi.

```
10 REM  GIOCO E SALUTE
15 RANDOMIZE
20 CALL CHAR(128,"1C3F7CF13C7F1810")
25 CALL CHAR(129,"287C7EFE1CFE7CF8")
30 CALL CHAR(130,"18F63C78FE183E38")
35 CALL CLEAR
40 CALL SCREEN(2)
45 CALL CHAR(133,"3F7FFFFFFFFF7F3F")
50 CALL CHAR(136,"E0C080808080C0E0")
55 CALL CHAR(137,"0000000000000000")
60 CALL CHAR(153,"03070F0F0F0F0703")
65 CALL CHAR(144,"80C0E0E0E0E0C080")
70 CALL COLOR(13,15,1)
75 CALL COLOR(14,15,16)
80 CALL COLOR(15,9,1)
85 CALL COLOR(16,9,16)
90 CALL HCHAR(21,12,133,1)
95 CALL HCHAR(21,13,136,1)
100 CALL HCHAR(21,14,137,3)
105 CALL HCHAR(21,17,153,1)
110 CALL HCHAR(21,18,144,1)
115 Q=0
120 FOR S=20 TO 1 STEP -1
125 CALL SOUND(1,RND*100+110,1)
130 CALL SOUND(1,RND*100+110,1)
135 CALL SOUND(1,RND*100+110,1)
140 Q=Q+.5
145 CALL HCHAR(S,18-Q,RND*2+128,Q*2+RND*2)
150 NEXT S
155 CALL CHAR(97,"FF8181818181FF")
160 CALL CHAR(96,"FFFFE7E73C3C1818")
165 CALL COLOR(9,2,11)
170 CALL HCHAR(6,14,96,1)
175 CALL HCHAR(6,18,96,1)
180 CALL HCHAR(9,14,97,5)
185 CALL HCHAR(10,14,97,5)
190 FOR C=0 TO 400
195 NEXT C
200 CALL SCREEN(8)
205 CALL COLOR(5,14,1)
210 CALL COLOR(6,14,1)
215 CALL COLOR(1,14,1)
220 CALL COLOR(7,14,1)
225 FOR K=3 TO 20 STEP 3
230 CALL HCHAR(10,14,32,5)
235 CALL HCHAR(11,14,97,5)
240 CALL HCHAR(11,14,32,5)
245 CALL HCHAR(10,14,97,5)
250 CALL SOUND(1,300,1)
255 CALL SOUND(1,400,1)
260 CALL HCHAR(K,4,67,1)
265 CALL HCHAR(K,5,79,1)
270 CALL HCHAR(K,6,85,1)
275 CALL HCHAR(K,7,71,1)
280 CALL HCHAR(K,8,72,1)
285 CALL HCHAR(K,9,33,1)
290 CALL HCHAR(K,10,33,1)
295 NEXT K
300 FOR C=0 TO 500
305 NEXT C
310 CALL CLEAR
315 CALL COLOR(5,2,1)
320 CALL COLOR(6,2,1)
325 CALL COLOR(1,2,1)
330 CALL COLOR(7,2,1)
```

```
335 CALL SCREEN(13)
340 PRINT "       F  U  M  I  ? ":::::::
345 INPUT "       SI O NO ? ":K$
350 IF K$="NO" THEN 530
355 IF K$()"SI" THEN 630
360 CALL CLEAR
365 INPUT " QUANTE NE FUMI AL GIORNO?      ":SN
370 IF SN=0 THEN 460
375 PRINT ::::
380 INPUT " QUANTO COSTA UN PACCHETTO?     ":COSTO
385 CALL CLEAR
390 PRINT "   NON FUMANDO AVRESTI          RISPARMIATO : ":::
395 PRINT
400 PRINT ((SN*7)/20)*COSTO;" LIRE ALLA SETTIMANA"
405 PRINT
410 PRINT ((SN*30)/20)*COSTO," LIRE AL MESE"
415 PRINT
420 PRINT ((SN*365)/20)*COSTO," LIRE ALL'ANNO"
425 FOR T=1 TO 1000
430 NEXT T
435 CALL CLEAR
440 IF SN)25 THEN 480
445 IF SN)20 THEN 495
450 IF SN)15 THEN 505
455 IF SN)5 THEN 470
460 PRINT "       O K ! "
465 GOTO 540
470 PRINT "NON FUMI TROPPO "
475 GOTO 540
480 PRINT "ATTENZIONE LA TUA SALUTE"
485 PRINT "E' IN SERIO PERICOLO !"
490 GOTO 540
495 PRINT "   SONO TROPPE !!"
500 GOTO 540
505 PRINT "SONO TANTE !! "
510 PRINT "TI CONSIGLIO DI LIMITARTI !"
515 FOR C=0 TO 1000
520 NEXT C
525 GOTO 15
530 CALL CLEAR
535 PRINT "******** B E N E *********"
540 FOR K=110 TO 3000 STEP 30
545 CALL SOUND(1,K,1)
550 NEXT K
555 FOR C=0 TO 500
560 NEXT C
565 CALL CHAR(121,"E1A6E41818E4A6E1")
570 FOR Q=4 TO 20
575 CALL HCHAR(10,Q-3,32,1)
580 CALL HCHAR(10,Q-2,121,1)
585 CALL HCHAR(10,Q-1,32,1)
590 CALL HCHAR(10,Q,133,1)
595 CALL HCHAR(10,Q+1,136,1)
600 CALL HCHAR(10,Q+2,137,3)
605 CALL HCHAR(10,Q+5,153,1)
610 CALL HCHAR(10,Q+6,144,1)
615 NEXT Q
620 CALL HCHAR(10,20,32,8)
625 GOTO 15
630 PRINT ::
635 PRINT "PARTECIPA SERIAMENTE !!"
640 FOR C=0 TO 1000
645 NEXT C
650 GOTO 15
```

5.10 GIOCO II

Questo programma simula le estrazioni del lotto. Potete utilizzarlo più volte per simulare i numeri ritardatari sulle varie ruote.

La linea 150 genera i numeri casuali da 1 a 90 e le linee 160-180 controllano se su di una stessa ruota escono numeri uguali. Le linee 210-270 consentono un'esatta tabellazione.

```
10 REM   ***GIOCO DEL LOTTO***
20 CALL CLEAR
30 OPTION BASE 1
40 DIM M(90)
50 PRINT TAB(5);"ESTRAZIONI DEL LOTTO"::
60 RANDOMIZE
70 FOR U=1 TO 10
80 READ R$(U)
90 NEXT U
100 DATA "BARI","CAGLIARI","FIRENZE","GENOVA","MILANO"
110 DATA "NAPOLI","PALERMO","ROMA","TORINO","VENEZIA"
120 FOR I=1 TO 10
130 PRINT R$(I);
140 FOR J=1 TO 5
150 A(J)=INT(90*RND)+1
160 FOR U=1 TO 90
170 IF M(U)=A(J)THEN 150 ELSE 180
180 NEXT U
190 M(A(J))=A(J)
200 NEXT J
210 C1=LEN(STR$(A(1)))
220 C2=LEN(STR$(A(2)))
230 C3=LEN(STR$(A(3)))
240 C4=LEN(STR$(A(4)))
250 C5=LEN(STR$(A(5)))
260 PRINT TAB(11-C1);A(1);TAB(15-C2);A(2);TAB(19-C3);A(3);
270 PRINT TAB(23-C4);A(4);TAB(27-C5);A(5)::
280 NEXT I
290 PRINT "batti un tasto per finire"
300 CALL KEY(0,AA,ST)
310 IF ST=0 THEN 300
320 END
```

5.11 GIOCO III

Battaglia navale — richiede il modulo del BASIC ESTESO. In esso sono evidenziate tutte le funzioni applicabili agli Sprite. Per il fuoco utilizzate il tasto F; FCTN S e FCTN D permettono lo spostamento della nave.

```
100 CALL SCREEN(16):: CALL CLEAR :: CALL MAGNIFY(2):: RANDOMIZE
105 SD=2000 :: SPEED=7
110 FOR CHAN=1 TO 8 :: CALL COLOR(CHAN,16,2):: NEXT CHAN
115 DISPLAY AT(1,1):"PUNTI 0"
120 CALL CHAR(96,"FFFFFFFFFFFFFFFF0000342822FF7E3CC8")
125 CALL CHAR(104,"FFFFFFFFFFFFFFFF000008181818FFFF7E")
130 CALL CHAR(112,"FFFFFFFFFFFFFFFF4701342910483156416692642434265476536746524345466547384167926344244234343141 3")
135 CALL COLOR(9,8,1,10,6,1,11,5,1)
140 FOR A=24 TO 20 STEP -1
145 CALL HCHAR(A,1,112,32)
150 NEXT A
155 FOR B=19 TO 16 STEP -1
160 CALL HCHAR(B,1,104,32)
165 NEXT B
170 FOR C=15 TO 12 STEP -1
175 CALL HCHAR(C,1,96,32)
180 NEXT C
185 DISPLAY AT(24,1)SIZE(16):"BATTAGLIA NAVALE"
190 IF RND(.5 THEN CALL SPRITE(#1,97,15,75,RND*190+1,0,-5)ELSE CALL SPRITE(#1,97,15,75,RND*190+1,0,10)
195 G=INT(RND*56)+100
200 M=INT(RND*170)+10
205 V=INT(RND*2)+1
210 IF V=2 THEN CALL SPRITE(#2,105,2,G,M,0,-SPEED)ELSE CALL SPRITE(#2,105,2,G,M,0,SPEED)
215 CALL SOUND(-4250,110,15,-1,15,SD,10)
220 IF GM=1 THEN SD=SD-40
225 CALL POSITION(#3,Y1,X1):: IF Y1)183 THEN 285
230 CALL COINC(ALL,HIT):: IF HIT=-1 THEN 360
235 CALL KEY(1,K,S):: IF S=0 OR S=-1 THEN 215
240 IF K=2 THEN CALL MOTION(#1,0,-7)
245 IF K=3 THEN CALL MOTION(#1,0,7)
250 IF K=12 AND X=0 THEN 260
255 GOTO 235
260 REM
265 CALL POSITION(#1,Y,X)
270 CALL SPRITE(#3,46,7,Y+10,X,10,0)
275 X,GM=1 :: SD=SD-20
280 GOTO 215
285 REM
290 CALL POSITION(#3,Y,X):: CALL DELSPRITE(#3)
295 CALL SPRITE(#4,113,12,183,X)
300 FOR A=1 TO 2
305 CALL PATTERN(#4,114):: CALL COLOR(#4,RND*12+3)
310 CALL PATTERN(#4,115):: CALL COLOR(#4,RND*12+3)
315 CALL PATTERN(#4,113):: CALL COLOR(#4,RND*12+3)
320 CALL PATTERN(#4,46):: CALL COLOR(#4,RND*12+3)
325 CALL SOUND(-500,-5,0)
330 NEXT A
335 G=G-6 :: CALL POSITION(#2,A,B):: CALL LOCATE(#2,G,8)
340 CALL POSITION(#2,Y,X)
345 IF Y(100 THEN 465
350 GM,X,S=0 :: SD=2000
355 GOTO 215
360 REM
365 CALL DELSPRITE(#3)
370 IF V=2 THEN CALL MOTION(#2,5,-10)ELSE CALL MOTION(#2,5,10)
375 CALL SOUND(-200,-6,0):: CALL COLOR(#2,10)
380 CALL POSITION(#2,A,B):: IF A)163 THEN 385 ELSE 380
385 CALL PATTERN(#2,113):: CALL COLOR(#2,12)
390 CALL SOUND(-1000,-4,-0)
395 CALL SOUND(-1000,-6,-0)
```

```
400 CALL MOTION(#2,0,0)
405 CALL POSITION(#2,Y,X):: CALL DELSPRITE(#2)
410 FOR B=5 TO 10
415 CALL SPRITE(#6,46,RND*2+9,Y,X,-10,SGN(RND-.5)*20)
420 NEXT B
425 FOR D=5 TO 10
430 FOR DEL=1 TO 20 :: NEXT DEL
435 CALL DELSPRITE(#D)
440 NEXT D
445 SCORE=SCORE+1 :: DISPLAY AT(1,1):"PUNTI";SCORE
450 SPEED=SPEED+3
455 GM,X,S=0 :: SD=3000
460 GOTO 195
465 REM
470 CALL MOTION(#1,0,0)
475 CALL POSITION(#1,C,D)
480 CALL POSITION(#2,A,B)
485 CALL SPRITE(#5,46,16,A,B)
490 U=C-A :: V=D-B :: D1X=SGN(U):: D1Y=SGN(V):: D2X=D1Y :: D2Y=0
495 M=ABS(U):: N=ABS(V)
500 IF M>N THEN 510
505 D2X=0 :: D2Y=SGN(V):: M=ABS(V):: N=ABS(U)
510 S=INT(M/2)
515 FOR I=0 TO M
520 CALL LOCATE(#5,A,B)
525 S=S+N :: IF S<M THEN 535
530 S=S-M :: A=D1X :: B=B+D1Y :: GOTO 540
535 A=A+D2X :: B=B+D2Y
540 NEXT I
545 FOR AS=1 TO 10
550 CALL SOUND(200,-4,0,110,15,210,15,310,15)
555 CALL PATTERN(#1,115)
560 CALL COLOR(#1,RND*12+3)
565 CALL SCREEN(RND*12+3)
570 CALL COLOR(#1,RND*12+3)
575 CALL PATTERN(#1,113):: CALL COLOR(#1,RND*12+3)
580 NEXT AS
585 CALL SOUND(-1000,-5,0)
590 CALL SCREEN(16)
595 CALL POSITION(#1,Y,X):: CALL DELSPRITE(#1,#5)
600 FOR DEL=1 TO 500 :: NEXT DEL
605 CALL PATTERN(#1,97)
610 CALL COLOR(#1,15)
615 DISPLAY AT(10,7):"GIOCHI ANCORA ?(Y/N)"
620 CALL DELSPRITE(ALL)
625 CALL KEY(0,K,S):: IF S=0 THEN 625
630 IF K=89 OR K=121 THEN 650
635 IF K=78 OR K=110 THEN CALL CLEAR :: CALL DELSPRITE(ALL):: STOP
640 CALL SOUND(10,110,0)
645 GOTO 625
650 X=0 :: GM=0 :: SCORE=0 :: SPEED=7 :: SD=2000
655 DISPLAY AT(1,1):"PUNTI";SCORE
660 CALL HCHAR(10,5,32,22)
665 GOTO 185
670 END
```

APPENDICE A

PORTA SERIALE RS232

PIN	MNEMONIC	OUTPUT/INPUT	FUNZIONE
1		Ground	Terra di protezione
2	RD	Input	Dati seriali entrati in RS232/1
3	TX	Output	Dati seriali da RS232/1
5	CTS	Output	Clear da inviare a RS232/1
6	DSR	Output	Data set ready (resistore di "pull-up" da 1.8 Kohm e + 12 V).
7		Ground	Massa di logica o di segnale
8	DCD	Output	Rivelazione di supporto dati per RS232/1
20	DTR	Input	Terminale dati pronto per RS232/1

IMPOSTAZIONE DI OPZIONI DA SOFTWARE

Opzioni	Impostare	Applicabile a Porta Parallela
Velocità di BAUD = 110, 300, 600 1200, 2400, 4800, or 9600	.BA = 300 (o velocità voluta)	NO
Bits di dati = 7 or 8	.DA = 7 (or 8)	NO
Parità = Dispari, Pari or Nessuna	.PA = O (or E or N)	NO
Due Bits di Stop	.TW	NO
Zeri	.NU	SI
Controllo parità	.CH	NO
Eco Escluso	.EC	SI
CRLF Escluso	.CR	SI
LF Escluso	.LF	SI

VALORE DI ASSEGNAZIONE AUTOMATICA DEGLI INTERRUTTORI DA SOFTWARE

Opzione	Open/List	Old/Save	Applicabile a I/O Parallela
Velocità di BAUD	Velocità di BAUD = 300	Velocità di BAUD = 300	NO
Bits di dati	Bits di dati = 7	Opzione non consentita	NO
Parità	Parità=Dispari	Opzione non consentita	NO
Bits di stop	1 Bit di stop	1 Bit di stop	NO
Zeri	Nessuno zero	Opzione non consentita	SI
Controllo Parità	Nessun controllo parità	Opzione non consentita	NO
Eco escluso	Eco	Opzione non consentita	SI
CRLF OFF	Ritorno di Carrello	Opzione non consentita	SI
LF OFF	Line a successiva	Opzione non consentita	SI

Esempi: Per Interfaccia Seriale RS232: OPEN # 1: "RS232.BA=1200" – SAVE "RS232/2.BA=600.TW".
Per Interfaccia I/O Parallelo: OLD "PIO" – LIST "PIO".

BIBLIOGRAFIA:

Manuale "RS232" – Texas Instruments, pag. 70.

APPENDICE B

ORGANIZZAZIONE DELLA MEMORIA

Prima di presentare lo schema della memoria interna del TI 99/4A vediamo la terminologia utilizzata:

1) La CPU (Central Processing Unit) è rappresentata dal TMS 9900 il cui compito è di processare ed eseguire i comandi e le istruzioni, nonchè indirizzare le locazioni di memoria.

2) La RAM è la memoria di elaborazione dove vengono scritti e letti i programmi e i dati da elaborare.

3) La ROM è la memoria a sola lettura dove sono memorizzati i comandi di sistema operativo, l'interprete BASIC, le routine di utilità e gestione del sistema.

4) La memoria GROM (Graphic Read Only Memory) è un tipo di ROM che contiene i programmi eseguiti dall'interprete grafico GPL (Graphic Programming Language).

5) Il processore VDP (Video Display Processor) gestisce le funzioni del video e le tabelle dei simboli e dei colori.

6) La memoria VDP RAM è dedicata al VDP; in essa vengono caricati i programmi sorgenti BASIC, le tabelle dei smboli, i valori dello stack (puntatori), le stringhe vuote. Una parte della VDP RAM è utilizzata come registro di transito (buffer) per i dati e un'altra parte per trasferire blocchi di informazioni (PAB) tra file e dispositivi esterni tramite DSR (Device Service Routine).

7) Il registro CRU (Communication Register Unit) è un registro interno alla CPU dedicato alla porta seriale di CRU.

8) Speech Synthetizer è il sintetizzatore vocale che permette la sintesi della voce.

9) Sound Processor è il processore per la generazione dei suoni su cinque ottave.

10) PAB (Peripheral Access Block).

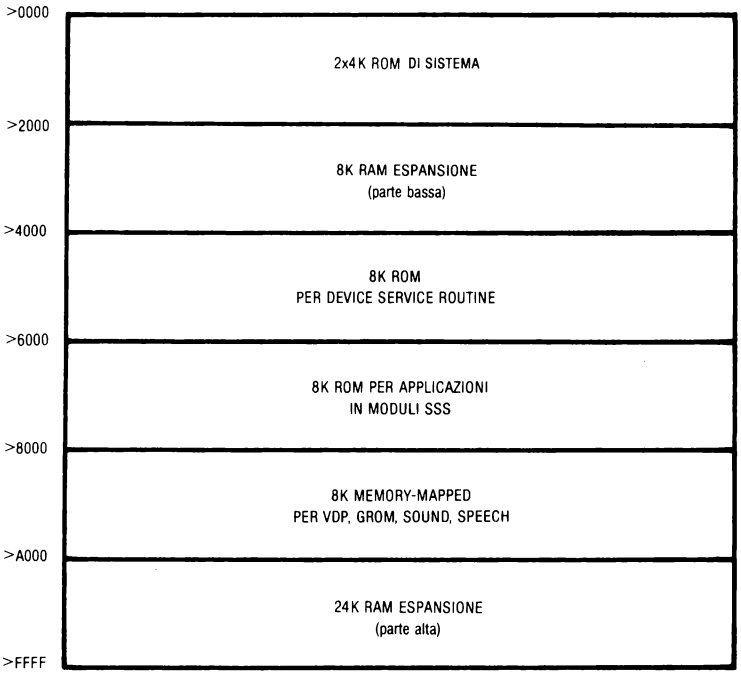
11) SSS MODULE (Modulo Solid State Software).

Il TMS 9900 non è in grado di gestire da solo la capacità massima del sistema, 110K tra RAM e ROM e quindi si avvale della collaborazione del microprocessore

grafico TMS 9918A, che oltre a svolgere le routine di I/0 su video, ha anche un banco di memoria dedicato VDP RAM.

Ed è proprio in questi 16K RAM che vengono caricati i programmi sorgenti Basic, la VDP RAM è indirizzata del TMS 9900 tramite registri di comunicazione e buffer con il TMS 9918A.

Vediamo la memoria (64K) direttamente indirizzabile dal 9900:



>1FFF = 8K

La memoria è vista come 8 blocchi di 8K byte ciascuno; vediamo le specifiche di ciascun blocco:

>0 – 1FFF	Sono contenuti in ROM l'interprete BASIC e ROUTINE d'utilità.
>2000 – 3FFF	Rappresenta la parte bassa di memoria RAM quando è inserita l'espansione di memoria di 32K RAM.
>4000 – 5FFF	Sono contenute in ROM le DSR (Device Service Routine) che danno accesso alle periferiche come dischi, stampante ecc. Queste ROM sono selezionate dal CRU anche in parallelo.

>6000 – 7FFF Utilizzato per i moduli SSS (permettono la connessione alle GROM).

>8000 – 9FFF Memory-mapped per selezionare il processore VDP,GROM,SOUND,SPEECH.

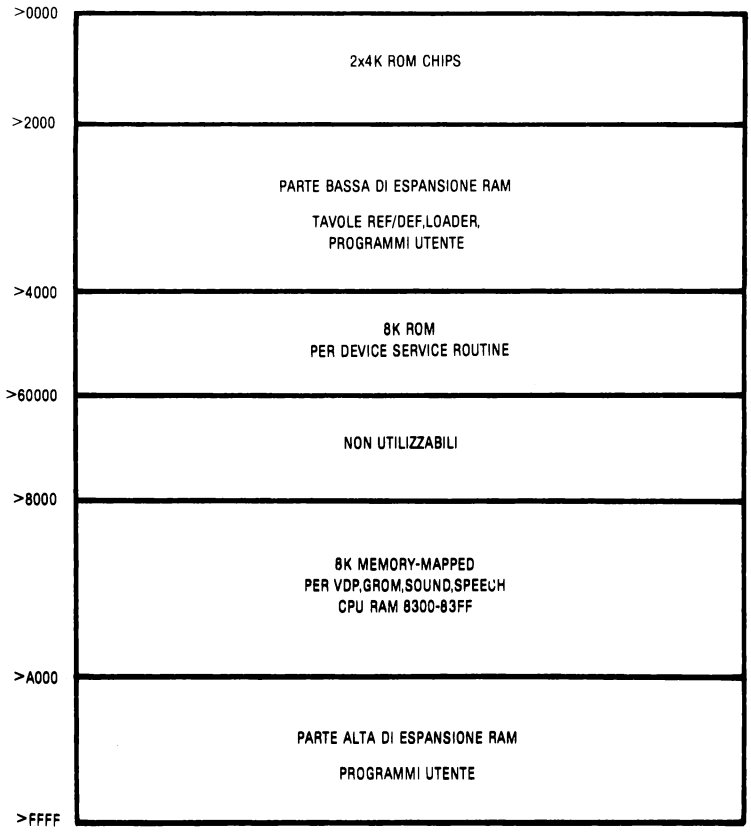
>A000 – BFFF Sono tre blocchi di 8K che rappresentano la parte alta di memoria RAM quando è inserita l'espansione di memoria di 32K RAM.

>C000 – DFFF

>E000 – FFFF

Qualora volete utilizzare l'EDITOR/ASSEBLER inserendo direttamente in consolle il modulo SSS dell'assemblatore, espandendo la memoria RAM di 32K, otterrete la seguente organizzazione di memoria indirizzabile direttamente da CPU:

Selezionate l'opzione LOAD AND RUN



L'espansione di memoria 32K RAM è suddivisa in un blocco di 8K dall'indirizzo >2000 a >3FFF contenente il LOADER, tabelle dei simboli REF/DEF, parte di programma utente e da un blocco di 24K contenente il programma utente. Gli indirizzi da >FFD8 a >FFFF sono utilizzati per istruzioni XOP.

Le due ROM da 4K indirizzate da >0000 a >1FFF contengono i comandi di sistema operativo, l'interprete GPL, l'interprete BASIC.

Il blocco di 8K dall'indirizzo >4000 a >5FFF è assegnato alle ROM, che gestiscono le periferiche attraverso le DSR (Device Service Routine), selezionate dai bit di CRU.

Il blocco di 8K dall'indirizzo >8000 a >A000 contiene una memory-mapped per gestire la VDP RAM, le GROM, il processore del SOUND e il processore dello SPEECH.

Le GROM che possono essere gestite dalla CPU sono otto, di cui tre sono presenti in consolle e cinque possono essere aggiunte con i moduli SSS, ciascuna contiene 6K byte di memoria. Le GROM sono indirizzabili dall'indirizzo >0000 a >F7FF, la GROM 0 è indirizzabile da >0000 a >17FF, la GROM 1 da >2000 a >37FF ecc.

La memoria VDP RAM è un'area di 16K (dall'indirizzo >0000 a >3FFF) gestita dal processore VDP che è in comunicazione con la CPU.

È sufficiente un solo indirizzo in CPU RAM per leggere o scrivere uno specifico blocco di dati in VDP RAM in quanto gli indirizzi di quest'ultima sono autoincrementati. In CPU RAM esistono già gli indirizzi per ciascuna funzione di I/O, così otteniamo la seguente MEMORY-MAPPED:

>8000	CPU RAM (>8300 – >83FF)
>8400	SOUND
>8800	VDP READ DATA
>8802	VDP READ STATUS
>8C00	VDP WRITE DATA
>8C02	VDP WRITE ADDRESS
>9000	SPEECH READ
>9400	SPEECH WRITE
>9800	GROM READ DATA
>9802	GROM READ ADDRESS
>9C02	GROM WRITE ADDRESS

Programando in TI BASIC la VDP RAM contiene i programmi sorgenti, i puntatori, la tavola dei simboli.

Programmando in ASSEMBLER i 16K VDP RAM vengono predisposti a contenere gli indirizzi degli SPRITE (figure in movimento) la velocità e posizioni degli SPRITE e le routine di gestione delle periferiche.

BIBLIOGRAFIA:

Manuale "Editor/Assembler" – Texas Instruments, pag. 398-403.

SINTESI DELLE ROUTINE DI UTILITÀ E SOTTOPROGRAMMI BASIC

COMANDI PER IL DEBUGGING

COMANDI	*CODICE*
Load Memory with ASCII	A
Breakpoint Set/Clear	B
CRU Inspect/Change	C
Execute	E
Find Word or Byte	F
GROM Base Change	G
Inspect Screen Location	I
Find Data Not Equal	K
Memory Inspect/Change	M
Move Block	N
Compare Memory Blocks	P
Quit Debugger	Q
Inspect or Change WP,PC, and SR	R
Execute in Step Mode	S
Trade Screen	T
Toggie Offset to and from TI BASIC	U
VDP Base Change	V
Inspect or Change Registers	W
Change Bias	X, Y, or Z
Hexadecimal to Decimal Conversion	>
Decimal to Hexadecimal Conversion	.
Hexadecimal Arithmetic	H

ROUTINE D'UTILITÀ

COD. MNEMONICO	*SIGNIFICATO*
DSRLNK	Links your program to Device Service Routines.
GPLLNK	Links your program to Graphics Programming Language routines.
KSCAN	Scans the keyboard.
LOADER	Links your program to the Loader to load TMS9900 tagged object code.
VMBR	Reads multiple bytes from VDP RAM.
VMBW	Writes multiple bytes to VDP RAM.
VSBR	Reads a single byte from VDP RAM.
VSBW	Writes a single byte to VDP RAM.
VWTR	Writes a single byte to a VDP Register.
XMLLNK	Links your program to the assembly language routines in the console ROM or in RAM.

SOTTOPROGRAMMI TI BASIC

CALL CHARPAT (character-number, string-variable[...])
CALL INIT
CALL LINK("program-name"[,parameter-list])
CALL LOAD(,"object-filename"["object-filename",...]) or
(address, value[,value...[,"", address, value[,value,...]]])
CALL PEEK(address, variable-list[,"",...])
CALL PEEKV(address, vairable-list["",...])
CALL POKEV(address, value-list[,"",...])

ROUTINE TI BASIC

CODICE	*SIGNIFICATO*
ERR	Reports errors.
NUMASG	Makes a numeric assignment.
NUMREF	Gets a numeric parameter.
STRASG	Makes a string assignment.
STRREF	Gets a string parameter.

TAG OGGETTO

TAG	*USO*	*CAMPO 1*	*CAMPO 2*
0	Program Identification	Program Lenghth	Program ID
1	Entry Point Definition	Absolute Address	
2	Entry Point Definition	Relocatable Address of Chain	Symbol
4	External References	Absolute Address of Chain	Symbol
5	External Definitions	Relocatable Address	Symbol
6	External Definitions	Absolute Address	Symbol
7	Checksum Indicator	Checksum	
8	Checksum Ignore	Any Value	
9	Load Address	Absolute Value	
A	Load Address	Relocatable Address	
B	Data	Absolute Value	
C	Data	Relocatable Address	
F	End of Record		

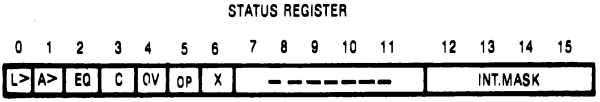
INDIRIZZI PREDEFINITI

CODICE	*INDIRIZZO*	*DATI CONTENUTI*
GPLWS	>83E0	GPL Interpreter Workspace.
GRMRA	>9802	GROM/GRAM read address.

GRMRD	>9800	GROM/GRAM read data.
GRMWA	>9C02	GROM/GRAM write address.
GRMWD	>9C00	GROM/GRAM write data.
PAD	>8300	The scratch pad used by TI BASIC, GPL, TI BASIC, and other programs. You may use some areas. See the Appendix for a detalled description of this area.
SCAN	>000E	Entry address of the keyboard Scan utility.
SOUND	>8400	Sound chip.
SPCHRD	>9000	Speech read.
SPCHWT	>9400	Speech write.
UTLTAB	>2022	Utility variable table.
VDPRD	>8800	VDP RAM read data.
VDPSTA	>8802	VDP RAM status.
VDPWA	>8C02	VDP RAM write address.
VDPWD	>8C00	VDP RAM write data.

SINTESI DELLE DIRETTIVE ASSEMBLER

REFERENCE	*INDIRIZZO*	*CODICE*	*SIGNIFICATO*
UTLTAB	>2022	UTLTAB	Entry address.
UTLAB+>2	>2024	FSTH	First free address in high memory.
UTLTAB+>4	>2026	LSTH	Last free address in high memory.
UTLTAB+>6	>2028	FSTLOW	First free address in low memory.
UTLTAB+>8	>202A	LSTLOW	Last free addres in low memory.
UTLTAB+>A	>202C	CHKSAV	Checksum.
UTLTAB+>>C	>202E	FLGPTR	Pointer to the flag in the PAB.
UTLTAB+>E	>2030	SVGPAT	GPL return address.
UTLTAB+>10	>2032	SAVCRU	CRU address of the peripheral.
UTLTAB+>12	>2034	SAVENT	Entry address of the DSR or subprogram.
UTLTAB+>14	>2036	SAVLEN	Device or subprogram name length.
UTLTAB+>16	>2038	SAVPAS	Pointer to the device or subprogram name in the PAB.
UTLTAB+>18	>203A	SAVVER	Version number of the DSR.



NOME	*BIT*	*SIGNIFICATO*
L>	0	Logical greater than
A>	1	Arithmetic greater than
EQ	2	Equal
C	3	Carry
OV	4	Overflow
OP	5	Odd parity
X	6	Extended operation
—	7–11	Reserved
INT. MASK	12–15	Interrupt mask

SIGNIFICATO	*COD. MNEMONICO*	*SINTASSI*
Absolute Origin	AORG	AORG <wd-exp>
Block Ending with Symbol	BES	BES<wd-exp>
Block Starting with Symbol	BSS	BSS<wd-exp>
Initialize Byte	BYTE	BYTE<exp>[,<exp>]...
Common Segment	CEND	CEND
Common Segment End	CSEG	CSEG
Copy File	COPY	COPY"<file name>"
Initialize Word	DATA	DATA <exp>[,<exp>]...
External Definition	DEF	DEF <symbol>[,<symbol>]...
Data Segment End	DEND	DEND
Dummy Origin	DORG	DORG <exp>
Data Segment	DSEG	DSEG
Define Extended Operation	DXOP	DXOP <symbol>,<term>
Program End	END	END [<symbol>]
Define Assembly-Time Constant	EQU	<label> EQU <exp>
Word Boundary	EVEN	EVEN
Program Identifier	IDT	IDT'<string>'
List Source	LIST	LIST
Force Load	LOAD	LOAD <symbol>[,<symbol>]...
Page Eject	PAGE	PAGE
Program Segment End	PEND	PEND
Program Segment	PSEG	PSEG

External Reference	REF	REF <symbol>[,<symbol>]...
Relocatable Origin	RORG	RORG [<exp>]
Secondary External Reference	SREF	SREF <symbol>,[<symbol>]...
Initialize Text	TEXT	TEXT[-]'<string>'
Page Titie	TITL	TITL'<string>'
No Source List	UNL	UNL

SINTESI DELLE ISTRUZIONI E PSEUDO ISTRUZIONI ASSEMBLER

SIGNIFICATO	*COD. MNEMONICO*	*OP. CODE*	*FORMATO*	*STATUS BIT UTILIZZATI*
Load immediate	LI	0200	VIII	0–2
Add immediate	AI	0220	VIII	0–4
And immediate	ANDI	0240	VIII	0–2
Or immediate	ORI	0260	VIII	0–2
Compare immediate	CI	0280	VIII	0–2
Store Workspace pointer	STWP	02A0	VIII	–
Store Status	STST	02C0	VIII	–
Load Workspace pointer immediate	LWPI	02E0	VIII	–
Load interrupt mask immediate	LIMI	0300	VIII	12–15
Idle	IDLE	0340	VII	–
Reset	RSET	0360	VII	–
Return with Workspace Pointer	RTWP	0380	VII	0–15
Clock on	CKON	03A0	VII	–
Clock off	CKOF	03C0	VII	–
Load or restart execution	LREX	03E0	VII	–
Branch and load Workspace pointer	BLWP	0400	VI	–
Branch	B	0440	VI	–
Return	RT	045B	VI	–
Execute	X	0480	VI	–
Clear	CLR	04C0	VI	–

SIGNIFICATO	*COD. MNEMONICO*	*OP. CODE*	*FORMATO*	*STATUS BIT UTILIZZATI*
Negate	NEG	0500	VI	0–2,4
Invert	INV	0540	VI	0–2
increment	INC	0580	VI	0–4
increment by two	INCT	05C0	VI	0–4
Decrement	DEC	0600	VI	0–4
Decrement by two	DECT	0640	VI	0–4
Branch and link	BL	0680	VI	–
Swap bytes	SWPB	06C0	VI	–
Set to one	SET0	0700	VI	–
Absolute value	ABS	0740	VI	0–2,4
Shift right arithmetic	SRA	0800	V	0–3
Shift right logical	SRL	0900	V	0–3
Shift left arithmetic	SLA	0A00	V	0–4
Shift right circular	SRC	0B00	V	0–3
Unconditional jump	JMP	1000	II	–
No operation	NOP	1000	II	–
Jump if less than	JLT	1100	II	–
Jump if low or equal	JLE	1200	II	–
Jump if equal	JEQ	1300	II	–
Jump if high or equal	JHE	1400	II	–
Jump if greater than	JGT	1500	II	–
Jump if not equal	JNE	1600	II	–
Jump if no carry	JNC	1700	II	–
Jump on carry	JOC	1800	II	–
Jump if no overflow	JNO	1900	II	–
Jump if logical low	JL	1A00	II	–
Jump if logical high	JH	1B00	II	–
Jump if odd parity	JOP	1C00	II	–
Set CRU bit to one	SBO	1D00	II	–
Set CRU bit to zero	SBZ	1E00	II	–
Test bit	TB	1F00	II	2
Compare ones corresponding	COC	2000	III	2
Compare zeros corresponding	CZC	2400	III	2
Exclusive or	XOR	2800	III	0–2
Extended operation	XOP	2C00	IX	6
Load CRU	LDCR	3000	IV	0–2,5
Store CRU	STCR	3400	IV	0–2,5

SIGNIFICATO	*COD. MNEMONICO*	*OP. CODE*	*FORMATO*	*STATUS BIT UTILIZZATI*
Multiply	MPY	3800	IX	–
Divide	DIV	3C00	IX	4
Set zeros corresponding	SZC	4000	I	0–2
Set zeros corresponding byte	SZCB	5000	I	0–2,5
Subtract words	S	6000	I	0–4
Subtract bytes	SB	7000	I	0–5
Compare words	C	8000	I	0–2
Compare bytes	CB	9000	I	0–2,5
Add words	A	A000	I	0–4
Add bytes	AB	B000	I	0–5
Move word	MOV	C000	I	0–2
Move byte	MOVB	D000	I	0–2,5
Set ones corresponding	SOC	E000	I	0–2
Set ones corresponding, byte	SOCB	F000	I	0–2,5

BIBLIOGRAFIA:

Manuale "Editor/Assembler" – Texas Instruments, Indice principale.

APPENDICE C

CODICI CARATTERI ASCII

I caratteri definiti per l'Home Computer TI-99/4A sono quelli standard ASCII con codici da 32 a 127. Nella seguente tabella sono elencati questi cartteri e i codici corrispondenti.

CODICE ASCII	CARATTERE	CODICE ASCII	CARATTERE	CODICE ASCII	CARATTERE
32	(spazio)	65	A	97	a
33	! (punto esclamativo)	66	B	98	b
34	" (virgolette)	67	C	99	c
35	# (numero o libbra)	68	D	100	d
36	$ (dollaro)	69	E	101	e
37	% (per cento)	70	F	102	f
38	& ("e" commerciale)	71	G	103	g
39	' (apostrofo)	72	H	104	h
40	((parentesi aperta)	73	I	105	i
41	) (parentesi chiusa)	74	J	106	j
42	* (asterisco)	75	K	107	k
43	+ (segno più)	76	L	108	l
44	, (virgola)	77	M	109	m
45	- (segno meno)	78	N	110	n
46	. (punto)	79	O	111	o
47	/ (barra obliqua)	80	P	112	p
48	0	81	Q	113	q
49	1	82	R	114	r
50	2	83	S	115	s
51	3	84	T	116	t
52	4	85	U	117	u
53	5	86	V	118	v
54	6	87	W	119	w
55	7	88	X	120	x
56	8	89	Y	121	y
57	9	90	Z	122	z
58	: (due punti)	91	[(parentesi quadra aperta)	123	{ (parentesi graffa aperta)
59	; (punto e virgola)	92	\ (barra obliqua inversa)	124	
60	< (minore)	93	] (parentesi quadra chiusa)	125	} (parentesi graffa chiusa)
61	= (uguale)	94	^ (elevamento a potenza)	126	~ (tilde)
62	> (maggiore)	95	_ (trattino)	127	DEL (visualizzato come spazio)
63	? (punto interrogativo)	96	` (accento grave)		
64	@ (segno at)				

Questi caratteri sono raggruppati in dodici *insiemi* per l'uso nei programmi di grafica a colori.

Insieme	*Codici Caratteri*	*Insieme*	*Codici Caratteri*	*Insieme*	*Codici Caratteri*
1	32-39	5	64-71	9	96-103
2	40-47	6	72-79	10	104-111
3	48-55	7	80-87	11	112-119
4	56-63	8	88-95	12	120-127

Esistono infine altri due caratteri del TI-99/4A, il *cursore*, con codice 30, e il *carattere di margine*, con codice 31.

TABELLA DEI CODICI DEI COLORI

Codice-colore	*Colore*
1	Trasparente
2	Nero
3	Verde
4	Verde chiaro
5	Blu scuro
6	Blu chiaro
7	Rosso scuro
8	Viola
9	Rosso
10	Rosso chiaro
11	Giallo scuro
12	Giallo chiaro
13	Verde scuro
14	Magenta
15	Grigio
16	Bianco

FREQUENZE DELLE NOTE MUSICALI

Frequenza	*Nota*		*Frequenza*	*Nota*	
110	LA		440	LA	(un'ottava sopra DO centrale)
117	LA #, SI♭		466	LA #, SI♭	
123	SI		494	SI	
131	DO	(un'ottava sotto DO centrale)	523	DO	(un'ottava sopra DO centrale)
139	DO #, RE♭		534	DO #, RE♭	
147	RE		587	RE	
156	RE #, MI♭		622	RE #, MI♭	
165	MI		659	MI	
175	FA		698	FA	
185	FA #, SOL♭		740	FA #, SOL♭	
196	SOL		784	SOL	
208	SOL #, LA♭		831	SOL #, LA♭	
220	LA	(un'ottava sotto DO centrale)	880	LA	(un'ottava sopra LA corista)
220	LA	(un'ottava sotto DO centrale)	880	LA	(un'ottava sopra LA corista)
233	LA #, SI♭		932	LA #, SI♭	
247	SI		988	SI	
262	DO	(DO centrale)	1047	DO	
277	DO #, RE♭		1109	DO #, RE♭	
294	RE		1175	RE	
311	RE #, MI♭		1245	RE #, MI♭	
330	MI		1319	MI	
349	FA		1397	FA	
370	FA #, SOL♭		1480	FA #, SOI ♭	
392	SOL		1568	SOI.	
415	SOL #, LA♭		1661	SOL #, LA♭	
440	LA	(un ottava sopra DO centrale)	1760	LA	

APPENDICE D

MESSAGGI DI ERRORI

Errori sintattici e semantici delle istruzioni prima dell'esecuzione con RUN:

*INCORRECT STATEMENT *ISTRUZIONE SCORRETTA

1) Su una riga ci sono una o più variabili senza il separatore virgola o punto e virgola.
2) In una stringa mancano le virgolette finali.
3) Caratteri o separatori non validi seguono i comandi LIST, NUM, RES, CON, RUN.
4) Una variabile è seguita da una costante numerica senza separatore.
5) Un comando non è la prima parola di una linea.
6) LIST non è seguito dai due punti volendo listare su un dispositivo esterno.

*BAD LINE NUMBER *NUMERO DI LINEA SCORRETTO

1) Il numero di linea è fuori dell'intervallo ammesso 1 – 32767.
2) Il RES genera un numero maggiore di 32767.

*BAD NAME *NOME SCORRETTO

1) Il nome di una variabile è più lungo di 15 caratteri.

*CAN'T CONTINUE *NON POSSO CONTINUARE

Il comando CON è stato dato inopportunamente, dopo l'interruzione è stata corretta qualche istruzione.

*CAN'T DO THAT *NON POSSO FARE CIÒ

1) Avete utilizzato dei comandi come istruzioni.
2) Avete utilizzato delle istruzione Basic come comandi di sistema operativo.

3) Avete utilizzato uno dei comandi RUN, LIST o SAVE senza programma presente in memoria principale RAM.

*LINE TOO LONG *LINEA TROPPO LUNGA

La linea d'ingresso eccede lo spazio previsto dal registro di buffer.

*MEMORY FULL *MEMORIA OCCUPATA

Avete superato la capacità di memoria disponibile.

Errori commessi durante la creazione della tabella dei simboli.

(Questa denota l'area di memoria riservata alle variabili, matrici, funzioni). Durante questa fase il computer riconosce gli errori sintattici e semantici e visualizza il messaggio corrispondente indicando il numero della linea errata:

*BAD VALUE IN 200 *VALORE SCORRETTO ALLA LINEA 200

1) Una dimensione di una matrice è scorretta.

2) Una dimensione di una matrice è zero, con OPTION BASE = 1.

*CAN'T DO THAT *NON POSSO FARE CIÒ

1) Nel programma avete inserito più OPTION BASE.

2) L'istruzione OPTION BASE ha un numero di linea più alto di un'istruzione che dimensiona la matrice.

*FOR NEXT ERROR *ERRORE DEL CICLO

Non avete chiuso o aperto lo stesso numero di cicli, avete inserito un diverso numero di FOR e NEXT.

*INCORRECT STATEMENT *ISTRUZIONE ERRATA

1) Nell'istruzione DEF manca qualche simbolo ")" di parentesi o l'uguaglianza "=" o qualche parametro ha un nome scorretto.

2) Nell'istruzione DIM manca qualche simbolo ")" di parentesi, la dimensione non è un numero, il nome della matrice è scorretto.

3) L'istruzione OPTION BASE non è completa di 1 o 0 o manca il termine BASE.
MEMORY FULL *MEMORIA PIENA

1) La dimensione di una matrice è troppo grande.

2) Lo spazio di memoria per variabili o funzioni è insufficiente.

*NAME CONFLICT	*CONFLITTUALITÀ FRA NOMI DI VARIABILI

1) Più matrici hanno lo stesso nome.

2) Variabile semplice e matrice hanno lo stesso nome.

3) Variabile semplice e funzione hanno lo stesso nome.

4) Il riferimento all'elemento della matrice non corrisponde alle dimensioni dichiarate.

Errori commessi nell'esecuzione del programma, ossia dopo aver battuto RUN.

Il Computer trovando l'errore ferma l'esecuzione e visualizza il messaggio d'errore col numero della linea errata:

Qui si riporta solo la traduzione dei messaggi che sono visualizzati, per i dettagli consultare il manuale d'uso da pag. 135 a 138.

* BAD ARGUMENT	*L'ARGOMENTO DELLE FUNZIONI È SCORRETTO
*BAD LINE NUMBER	*IL NUMERO DI LINEA È SCORRETTO O NON ESISTE
*BAD NAME	*IL NOME DI UN SOTTO-PROGRAMMA È SCORRETTO
*BAD SUBSCRIPT	*L'INDICE DELLA MATRICE O VETTORE È SCORRETTO
*BAD VALUE	*IL VALORE INDICATO NON CORRISPONDE AL VALORE AMMESSO NEI SOTTOPROGRAMMI O FUNZIONI O CODICI
*CAN'T DO THAT	*NON POSSO FARE CIÒ
*DATA ERROR	*ERRORE NELL'ASSEGNARE VALORI COL DATA
*FILE ERROR	*ERRORE NELLA GESTIONE DEI FILE
*INCORRECT STATMENT	*ISTRUZIONE ERRATA SINTATTICAMENTE O SEMANTICAMENTE
*MEMORY FULL	*MEMORIA PIENA

*NUMBER TOO BIS	*NUMERO TROPPO GRANDE
*STRING-NUMBER MISMATCH	*STRINGHE O NUMERI NON CORRISPONDONO AL TIPO DELLE VARIABILI UTILIZZATE

Errori commessi durante il trasferimento di dati o programmi dal Computer alle periferiche o viceversa.

Vengono detti di I/O (Input/Output), e il messaggio si presenta nella forma:

I/O ERROR X Y dove X indica l'operazione che ha causato l'errore

VALORI DI X	*OPERAZIONE*
0	OPEN
1	CLOSE
2	INPUT
3	PRINT
4	RESTORE
5	OLD
6	SAVE
7	DELETE

e Y indica il tipo di ERRORE

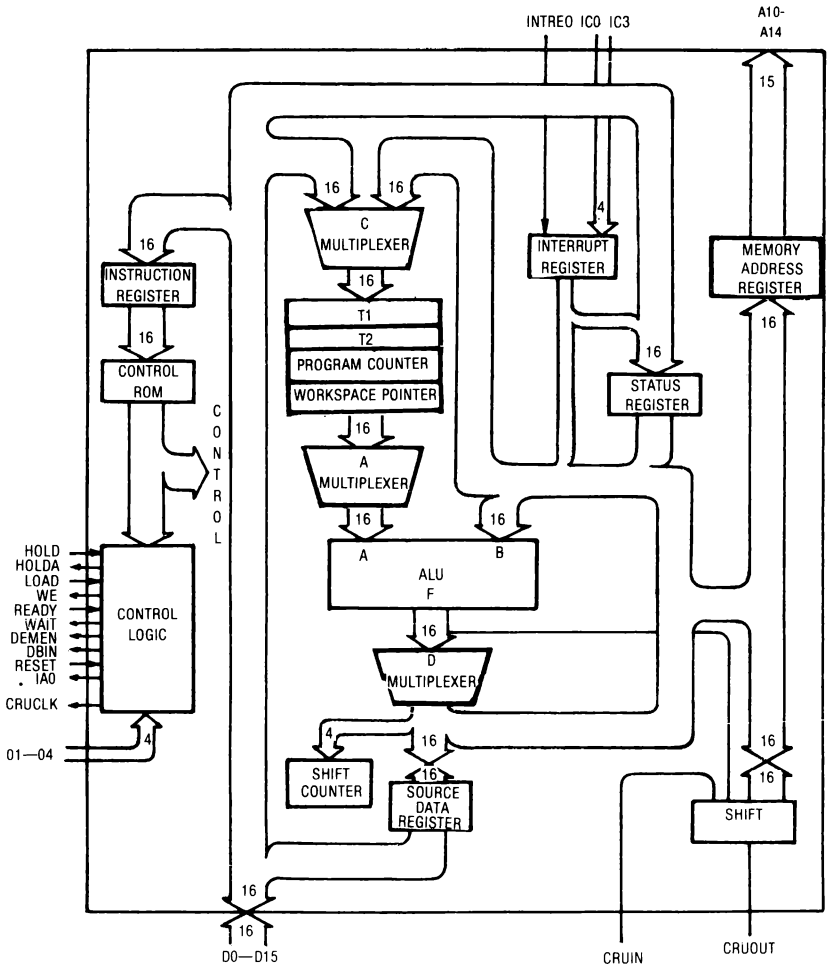
VALORI DI Y	*TIPO DI ERRORE*
0	Il nome del dispositivo utilizzato nei comandi DELETE, LIST, OLD e SAVE è errato.
1	Tentate di scrivere su un file protetto.
2	Una o più opzioni di OPEN sono errate o non corrispondono alle caratteristiche del file.
3	Un comando di I/0 non è valido.
4	Lo spazio della memoria di massa è insufficiente
5	Tentate di leggere oltre la fine del file.
6	Un dispositivo è scollegato o guasto.
7	Il file non esiste o è errato il tipo di file.

Questa appendice è tratta dal "Manuale d'uso".

BIBLIOGRAFIA:
Manuale d'uso – Texas Instruments, pag. 134-138.

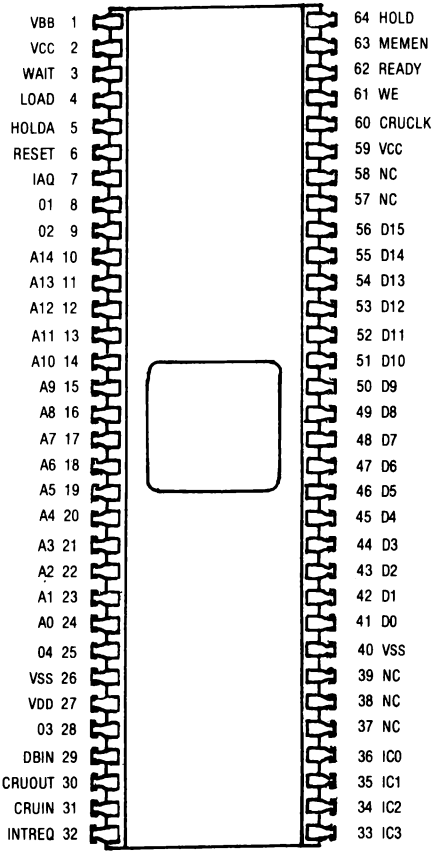
APPENDICE E

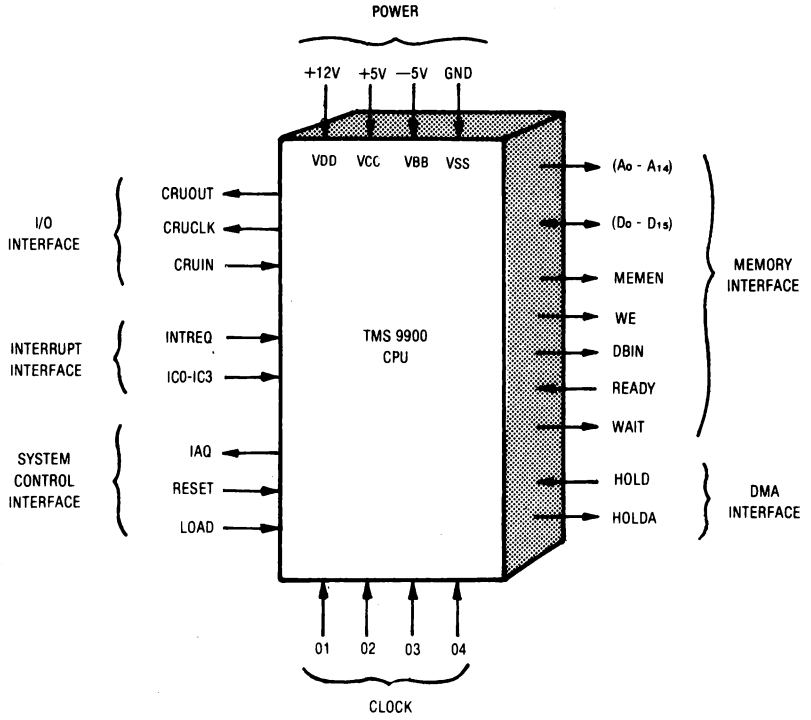
CARATTERISTICHE DEL TMS 9900



SCHEMA INTERNO DEL TMS 9900

DESCRIZIONE DEI PIN DEL TMS 9900





BIBLIOGRAFIA:

9900 Micro Processor Series — Texas Instruments — pag. 8-17/8-19

DESCRIZIONE DEI PIN

ADDRESS BUS

A0 — A14
A15 / CRU OUT — Costituiscono l'ADDRESS BUS, che indirizzano la memoria esterna quando $\overline{\text{MEMEN}}$ è alto. L'A15 è anche utilizzato come CRU DATA OUT. Tutti i pin sono di OUT, il bit più significativo (MSB) è A0.

DATA BUS

D0 — D15 — Costituiscono il DATA BUS bidirezionale, che trasferiscono i dati da leggere e scrivere in memoria quando $\overline{\text{MEMEN}}$ è alto. D0 è il bit più significativo (MSB).

CONTROL BUS

$\overline{\text{MEMEN}}$ — MEMORY ENABLE. Abilitazione all'accesso in memoria.
DBIN — DATA BUS IN. È alto quando i buffer e 9900 ricevono i dati.
$\overline{\text{WE}}$ — WRITE ENABLE. $\overline{\text{WE}}$ abilita a scrivere in memoria.
$\overline{\text{MBE}}$ — MEMORY BLOCK ENABLE. $\overline{\text{MBE}}$ indica l'accesso al blocco di memoria.
$\overline{\text{CRUCLK}}$ — CRU clock. Abilita la CRU OUT line a inviare dati.
CRUIN — CRU DATA IN. Dati di CRU entrano in CPU.

MEMORY CONTROL

READY / HOLD — READY (quando $\overline{\text{MEMEN}}$ è alto) indica che la memoria è pronta per l'accesso.
HOLD (quando $\overline{\text{MEMEN}}$ è basso) indica una richiesta all'uso del data bus.

HOLDA / IAQ — HOLD ACKNOWLEDGE (quando $\overline{\text{MEMEN}}$ è basso) indica che la CPU è in stato di attesa.
Instruction Acquisition (quando $\overline{\text{MEMEN}}$ è alto) indica che la CPU sta acquisendo un'istruzione.

WAIT — Quando è alto indica che la CPU è in stato di attesa.

TIMING e CONTROL

$\overline{\text{LOAD}}$ — Quando è alto, $\overline{\text{LOAD}}$ fa realizzare alla CPU un interrupt non mascherabile.

$\overline{\text{RESET}}$ — Quando è alto, $\overline{\text{RESET}}$ causa il reset (azzeramento) della CPU.

EXT INT / $\overline{\text{(INTREQ)}}$ — EXTERNAL INTERRUPT, quando è alto fa realizzare alla CPU un interrupt.

01,0,03;04 — CPU Clock - Fasi di clock.

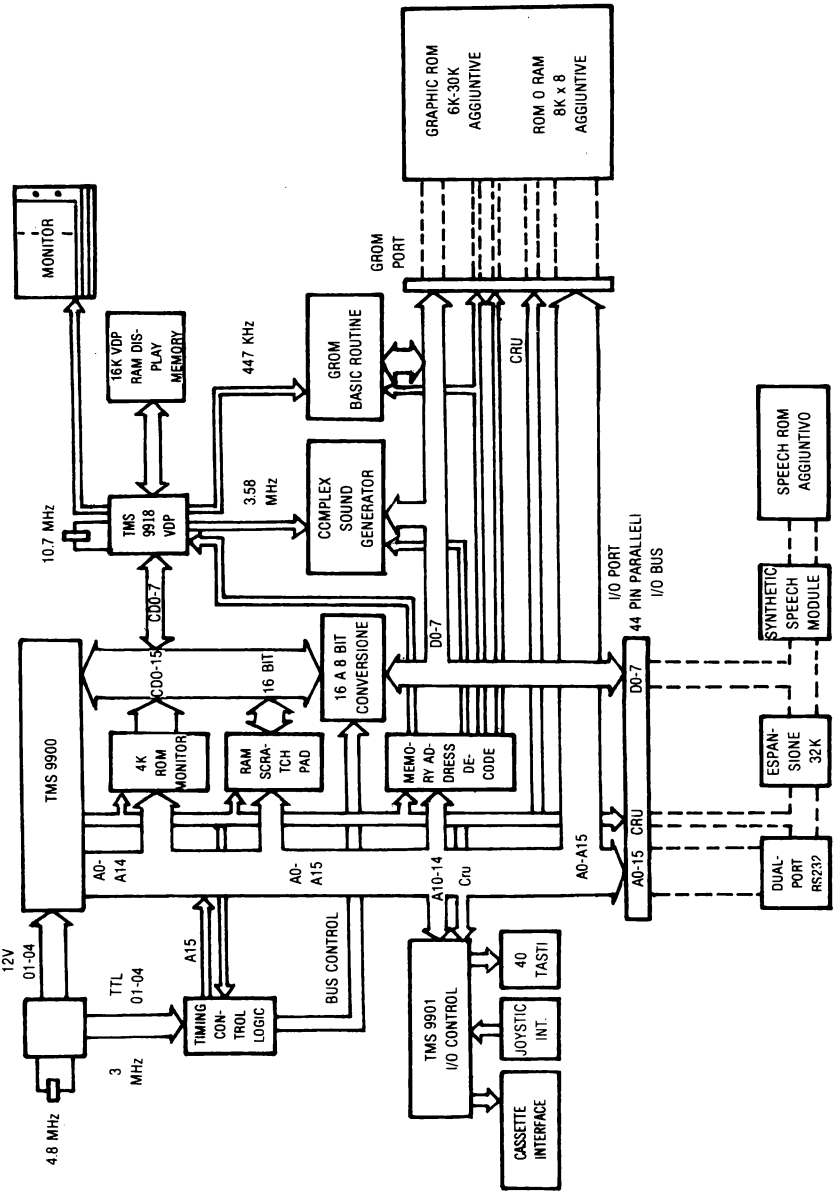
IC0 — IC3 — Realizzano le priorità di interrupt.

GND — GROUND. Terra

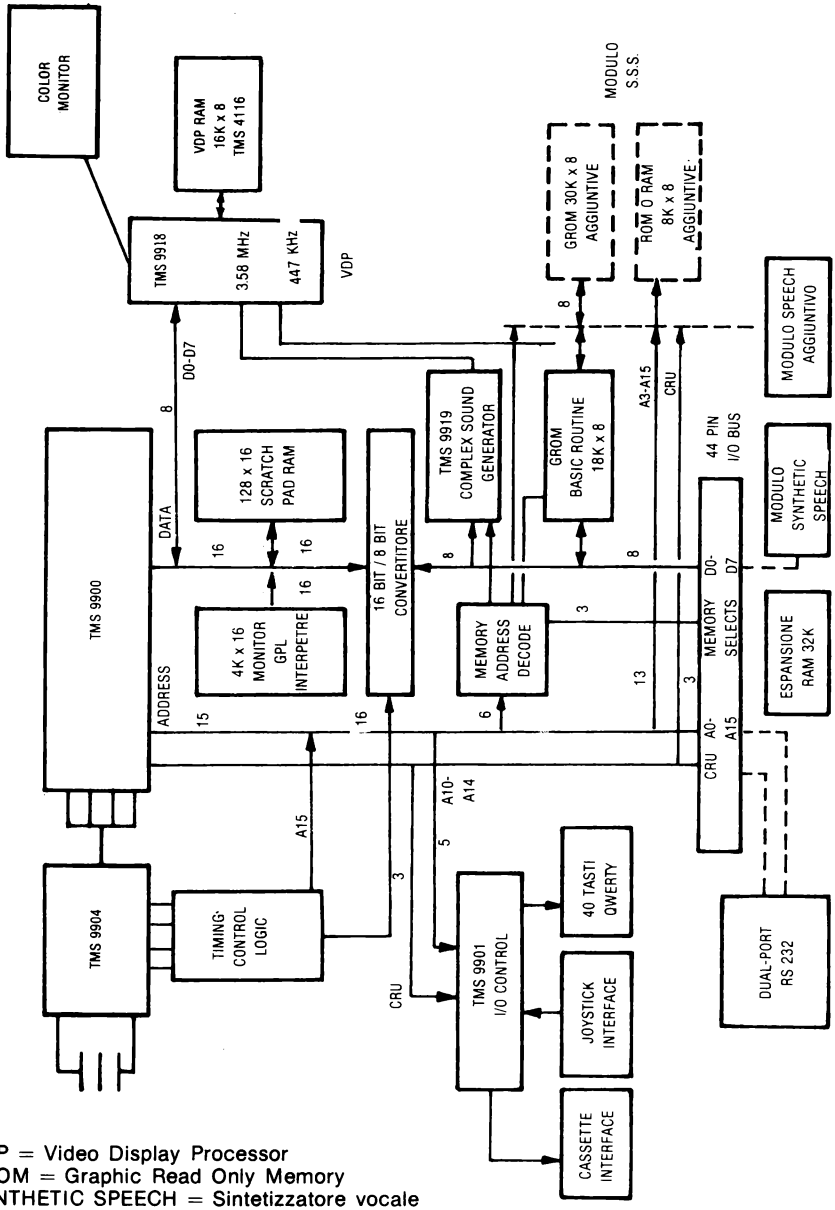
VDD,VCC,VBB — Alimentazioni +122,+5,—5

APPENDICE F

TI 99/4A SCHEMA A BLOCCHI



TI 99/4A SCHEMA A BLOCCHI



VDP = Video Display Processor
GROM = Graphic Read Only Memory
SYNTHETIC SPEECH = Sintetizzatore vocale
CRU = Communication Register Unit
S.S.S. = Solid State Software

APPENDICE G

MESSAGGI D'ERRORE DEL BASIC ESTESO

Codice		
10	Numeric overflow	Traboccamento del numero
14	Syntax error	Errore sintattico
16	Illegal after subprogram	Non utilizzabile dopo il sub
17	Unmatched quotes	Apici mancanti
19	Name too long	Nome troppo lungo
20	Unrecognized character	Carattere non riconosciuto
24	String number mismatch	Conflittualità fra stringa e numero
25	Option base error	Errore nell'option base
28	Improperly used name	Nome usato impropriamente
36	Image error	Errore nel formato dell'image
39	Memory full	Memoria piena
40	Stack overflow	Traboccamento dello stack
43	Next without for	Numero non uguale di FOR e NEXT
44	FOR-NEXT nesting	Cicli nidificati
47	Must be in sub program	Presente nel sottoprogramma
48	Recursive subprogram call	Sottoprogramma ricorsivo
49	Missing subend	Subend mancante
51	Return without Gosub	Return senza Gosub
54	String truncated	Stringa troncata
56	Speech string too long	Espressione vocale lunga
57	Bad subscript	Scritta scorretta
60	Line not found	Linea non presente
61	Bad line number	Numero di linea scorretto
62	Line too long	Linea troppo lunga
67	Can't continue	Non posso continuare
69	Command illegal in program	Comando illegale nel programma
70	Only legal in a program	Utilizzabile solo nel programma
74	Bad argument	Argomento scorretto
78	No program present	Programma non presente
79	Bad value	Valore scorretto
81	Incorrect argument list	Argomento scorretto del list
83	Input error	Errore di Input
84	Data error	Errore nel Data

97	Protection violation	Programma protetto
109	File error	Errore nel File
130	I/O Error	Errore di I/O
125	Subprogram not found	Sottoprogramma non presente

Nota:

1) Si ha un traboccamento (OVER FLOW) quando il numero eccede la rappresentazione max prevista.

2) Un sottoprogramma è ricorsivo quando chiama se stesso.

3) Lo STACK è un registro LIFO (LAST INPUT FIRST OUTPUT) dove vengono caricati i puntatori per poi essere prelevati.

APPENDICE H

VOCABOLARIO PER LA SINTESI VOCALE

– (NEGATIVE)
+ (POSITIVE)
. (POINT)
0
1
2
3
4
5
6
7
8
9

A

A (a)
Al ( )
ABOUT
AFTER
AGAIN
ALL
AM
AN
AND
ANSWER
ANY
ARE
AS
ASSUME
AT

B

BACK
BASE
BE
BETWEEN
BLACK
BLUE
BOTH
BOTTOM
BUT
BUY
BYE

C

CAN
CASSETTE
CENTER
CHECK
CHOICE
CLEAR
COLOR
COME
COMES
COMMA
COMMAND
COMPLETE
COMPLETED
COMPUTER
CONNECTED
CONSOLE
CORRECT
COURSE
CYAN

D

DATA
DECIDE
DEVICE
DID
DIFFERENT
DISKETTE
DO
DOES
DOING
DONE
DOUBLE
DOWN
DRAW
DRAWING

E

EACH
EIGHT
EIGHTY
ELEVEN
ELSE
END
ENDS
ENTER
ERROR
EXACTLY
EYE

F

FIFTEEN
FIFTY
FIGURE
FIND
FINE
FINISH
FINISHED
FIRST
FIT
FIVE
FOR
FORTY
FOUR
FOURTEEN
FOURTH
FROM
FRONT

G

GAMES
GET
GETTING
GIVE
GIVES
GO
GOES
GOING
GOOD
GOOD WORK
GOODBYE
GOT
GRAY
GREEN
GUESS

H

HAD
HAND

HANDHELD UNIT
HAS
HAVE
HEAD
HEAR
HELLO
HELP
HERE
HIGHER
HIT
HOME
HOW
HUNDRED
HURRY

I

I WIN
IF
IN
INCH
INCHES
INSTRUCTION
INSTRUCTIONS
IS
IT

J

J
JOYSTICK
JUST

K

KEY
KEYBOARD
KNOW

L

LARGE
LARGER
LARGEST
LAST
LEARN
LEFT
LESS
LET
LIKE
LIKES
LINE
LOAD
LONG
LOOK
LOOKS
LOWER

M

MADE
MAGENTA
MAKE
ME
MEAN
MEMORY
MESSAGE
MESSAGES
MIDDLE
MIGHT
MODULE
MORE
MOST
MOVE
MUST

N

NAME
NEAR
NEED
NEGATIVE
NEXT
NICE TRY
NINE
NINETY
NO
NOT
NOW
NUMBER

O

OF
OFF
OH
ON
ONE
ONLY
OR
ORDER
OTHER
OUT
OVER

P

PART
PARTNER
PARTS
PERIOD
PLAY
PLAYS
PLEASE
POINT
POSITION
POSITIVE
PRESS
PRINT
PRINTER
PROBLEM
PROBLEMS
PROGRAM
PUT
PUTTING

Q

R

RANDOMLY
READ (read)
READ1 (red)
READY TO START
RECORDER
RED
REFER
REMEMBER
RETURN
REWIND
RICHT
ROUND

S

SAID
SAVE
SAY
SAYS
SCREEN
SECOND
SEE
SEES
SET
SEVEN
SEVENTY
SHAPE
SHAPES
SHIFT
SHORT
SHORTER
SHOULD
SIDE
SIDES
SIX
SIXTY
SMALL
SMALLER
SMALLEST
SO
SOME
SORRY
SPACE

SPACES
SPELL
SQUARE
START
STEP
STOP
SUM
SUPPOSED
SUPPOSED TO
SURE

T

TAKE
TEEN
TELL
TEN
TEXAS INSTR.
THAN
THAT
THAT IS INC.
THAT IS RICHT
THE (the)
THE1 (tha)
THEIR
THEN
THERE
THESE
THEY
THING
THINGS
THINK
THIRD
THIRTEEN
THIRTY
THIS
THREE
THREW
THROUGH
TIME
TO
TOGETHER
TONE
TOO
TOP
TRY
TRY AGAIN
TURN
TWELVE
TWENTY
TWO
TYPE

U

UHOH
UNDER
UNDERSTAND
UNTIL
UP
UPPER
USE

V

VARY
VERY

W

WAIT
WANT
WANTS
WAY
WE
WEIGH
WEIGHT
WELL
WERE
WHAT
WHAT WAS THAT
WHEN
WHERE
WHICH
WHITE
WHO
WHY
WILL
WITH
WON
WORD
WORDS
WORK
WORKING
WRITE

X

Y

YELLOW
YES
YET
YOU
YOU WIN
YOUR

Z

ZERO

BIBLIOGRAFIA:

Manuale ex Basic – Texas Instruments, pag. 203-205.

Indice di ricerca per argomenti

Comandi, Funzioni, Istruzioni, Sottoprogrammi, del TI BASIC

Comandi, Istruzioni, Sottoprogrammi, Funzioni del BASIC ESTESO

Indice Bibliografico

TEXAS INSTRUMENTS, *9900 Family Systems Design*, Houston 1976.
TEXAS INSTRUMENTS, *TMS 9900 Family System Development Manual*, 1977.
TEXAS INSTRUMENTS, *TM 990 Introduction to Microprocessors*, 1979.
TEXAS INSTRUMENTS, *TI Extended Basic*, 1982.
TEXAS INSTRUMENTS, Home Computer Texas Instruments TI-99/4A *Manuale d'uso*, Milano 1982.
TEXAS INSTRUMENTS *Editor/Assembler del TI 99/4A*, 1982.
TEXAS INSTRUMENTS, *Manuale della scheda RS232*, 1982.
TEXAS INSTRUMENTS, *Manuale del Disk Driver*, 1982.
'99' er Magazine Rivista

PROGRAMMI DI MATEMATICA E STATISTICA

Leggendo questo libro il lettore potrà formarsi quella logica di base indispensabile per la risoluzione di problemi di matematica e statistica.
Ad ogni programma viene preposta un'esposizione schematica del metodo numerico e delle tecniche di programmazione utilizzate, il diagramma a blocchi relativo all'algoritmo, il listato (anch'esso ottenuto da calcolatore) in cui tra l'altro vengono specificati il tempo e la quantità di memoria impiegati.

Cod. 522D **L. 16.000 Pagg. 228**

INTRODUZIONE AL PASCAL

Il volume, incentrato su numerosissimi esempi che verificano costantemente l'apprendimento del lettore, insegna a conoscere, capire ed usare tutte le particolarità e i vantaggi di questo linguaggio. Nel corso della trattazione vengono ampiamente utilizzate le tecniche di programmazione strutturata, come pure tecniche particolari, quali il trattamento dei file, l'utilizzazione della recursività e il trattamento grafico.

Cod. 516A **L. 30.000 Pagg. 484**

COMPUTER GRAFICA

Si può dire che la computer grafica si pone nel contesto più generale del trattamento dell'informazione, avendo individuato nell'immagine un contenuto informativo che è possibile elaborare.
Quest'opera, con il suo rigore informativo e scientifico, si pone come fondamentale nel carente panorama italiano; inoltre le informazioni e gli spunti contenuti nel testo contribuiranno certamente alla divulgazione ed alla formazione di idee nuove e feconde.

Cod. 519P **L. 29.000 Pagg. 174**

APPLE II - Guida all'uso

Se possedete un Apple e volete conoscerlo a fondo se volete comprarlo, o se semplicemente volete imparare la sua programmazione, troverete in questo libro, tutte le risposte, comprese alcune vere "primizie" che vi occorrono per una perfetta operatività del sistema. Conoscerete i vari componenti del sistema e come usarli al meglio. Verrete guidati alla programmazione in BASIC e a usare le caratteristiche grafiche e sonore del sistema. Imparerete a memorizzare su disco sia programmi che archivi dati, come ad inserire un programma scritto in assembler in uno scritto in BASIC.
E poi ancora, tutte le istruzioni e funzioni BASIC e ben 12 appendici veramente basilari.

Cod. 331P **L. 26.000 Pagg. 400**

CEDOLA DI COMMISSIONE LIBRARIA

Ritagliare (o fotocopiare) e inviare a
Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano

Nome e Cognome _____

_____

Indirizzo _____

_____

Cap. _____ Città _____ Provincia _____

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

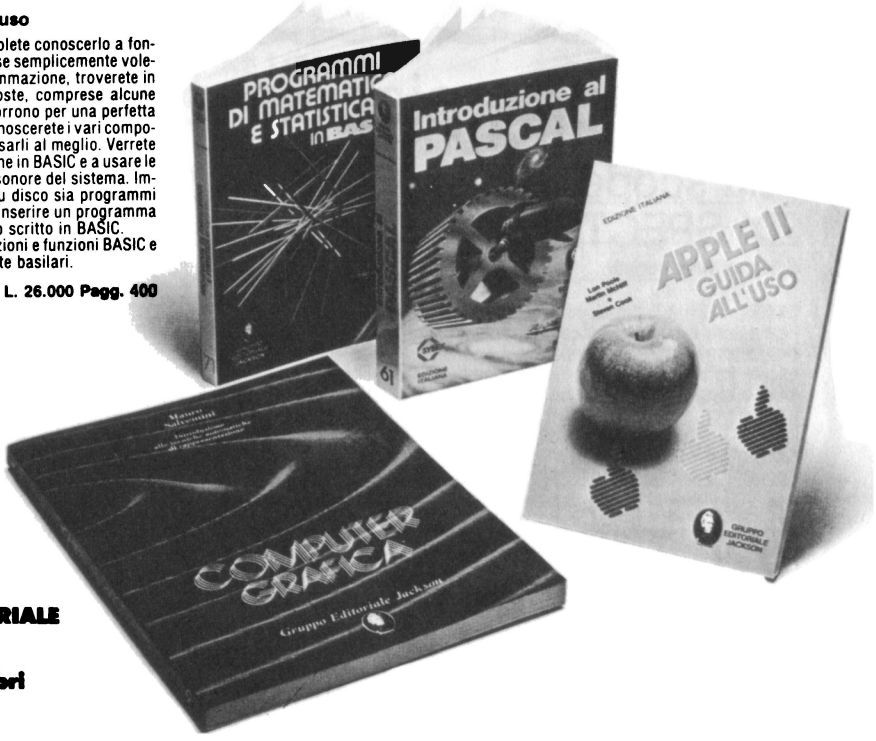
Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L 2 000 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L .....

Data ..... Firma .....

... dalla libreria JACKSON



GRUPPO EDITORIALE JACKSON

Divisione Libri

CEDOLA DI COMMISSIONE LIBRARIA

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città** **Provincia**

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L. 2.000 per contributo fisso spese di spedizione.

☐ Allego assegno n° ..... di L. .....

☐ Non abbonato ☐ Abbonato sconto 20% ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor
☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

Data ..... Firma .....

CEDOLA DI COMMISSIONE LIBRARIA

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città** **Provincia**

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L. 2.000 per contributo fisso spese di spedizione.

☐ Allego assegno n° ..... di L. .....

☐ Non abbonato ☐ Abbonato sconto 20% ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor
☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

Data ..... Firma .....

-*Thanks to:* ***Gianfranco Mazzarello (www.microatena.it)***

for the Scan of this book.

- *Revisited by TI99 Italian User Club (info@ti99iuc.it) in* 2015

Il primo obiettivo di questo libro è di insegnare il BASIC a coloro che si avvicinano per la prima volta alla programmazione e al computer.
Ciò viene ottenuto presentando i comandi del sistema operativo e le istruzioni del linguaggio BASIC in modo molto semplice, avvicinando il lettore alla terminologia specialistica e al suo significato logico.
Molti sono gli esercizi proposti, caratterizzati da un livello di difficoltà crescente.
Il secondo obiettivo è quello di far conoscere l'architettura del TI 99/4A di cui vengono forniti gli schemi a blocchi, con particolare riferimento all'organizzazione delle memorie e alle caratteristiche della CPU. Inoltre il lettore alla fine di ogni capitolo troverà dei consigli di utilità pratica maturati da numerose esperienze nella realizzazione di software applicativo sul sistema TI 99/4A.

Carmine Elefante laureato in ingegneria elettronica presso il Politecnico di Napoli, si è interessato sin dagli anni universitari alla filosofia e ai linguaggi dei microprocessori, maturando diverse esperienze nell'ambito della grande e media industria nel controllo del software di computer dedicati e nell'analisi del sotware applicativo.
Attualmente si interessa di mini e personal computer sia in fase di consulenza EDP sia in fase didattica e addestrativa degli utenti.

L. 15.000 Cod. 343 B ISBN 88-7056-026-0