LE PERIFERICHE COMMODORE

E.V.M. COMPUTERS

LE PERIFERICHE COMMODORE

E.V.M. COMPUTERS

I Edizione Maggio 1984

E.V.M. Computers
Via Marconi 9/a
52025 MONTEVARCHI (AR)

INTRODUZIONE

E' normalmente molto difficile che un computer possa essere utilizzato come sola unita' centrale. In questo caso infatti ci si dovrebbe limitare ad utilizzare un mezzo sicuramente meno potente, ma altrettanto sicuramente piu' facile da usare, come ad esempio una calcolatrice delle quali ne esistono versioni sempre piu' sofisticate e con grandi capacita'.
Esiste pur sempre, utilizzando un computer piccolo o grande che sia, il problema di immagazzinare programmi scritti dall' utente stesso oppure la necessita' di caricare programmi acquistati.

Percio' un computer diventa realmente un SISTEMA DI ELABORAZIONE DATI solo quando dispone come minimo di un' unita' di massa o di memoria esterna.
Questo libro e' stato scritto con lo scopo di insegnare ad utilizzare le principali periferiche che possono essere applicate ad un computer della serie COMMODORE.
La prima parte e' stata scritta perche' l' utente impari ad adoperare, via via sempre con maggior sicurezza, abilita' e soddisfazione, lo strumento piu' diffuso sugli HOME e sui PERSONAL: la cassetta di registrazione o DATASETTE o CN2.
Di norma infatti questo e' il primo accessorio che si acquista e le tecniche di lavoro sono eguali per tutti i tipi di computer COMMODORE.
Passeremo poi ad esaminare il trattamento di dati e programmi su disco.
Data la grande diffusione dei VIC 20 e dei CBM 64 ci riferiremo in modo particolare alle unita' a dischi 1540/1541 senza per questo mancare di

riportare esempi e dati relativi alle unita' 3040/4040/8050 e 8250 oltre a brevissimi accenni sugli HARD DISK sui quali pero' fino a questo punto le politiche di costruzione e di conseguenza quelle commerciali della COMMODORE non sembrano ancora chiaramente stabilizzate.

Dato che la manipolazione dei dati su disco era di particolare complessita' sulle unita' cosi' dette maggiori e data anche l' abitudine (tutta europea) ad utilizzare in forme sempre piu' spinte in particolare il CBM64, abbiamo approfondito in misura notevole anche tutte le descrizioni di funzionamento per i DUAL FLOPPY, visto che si tende ad impiegare queste unita' anche sul CBM 64.
Per ultimo tratteremo delle stampanti, ma in questo caso con riferimento all' uso della MPS 801 e 4023.

La seconda parte del manuale, e si tratta di una seconda parte LOGICA, nel senso che, dividendo il manuale stesso in tre grandi settori: CASSETTA, DISCO, STAMPANTE, quanto diciamo e' riportato subito al termine di ogni settore, e' dedicata invece ad un' approfondimento dei concetti espressi nella prima parte, ad una serie di precisazioni, alle routines oltre che ad una maggiore serie di spiegazioni relative alla parte HARDWARE.
Abbiamo deciso di non effettuare subito un discorso molto approfondito, ma di riservarlo alla seconda parte del manuale stesso perche' la lettura del corso ne risulta cosi' facilitata anche da parte di chi non desideri o non sia in grado di sorbirsi una grande serie di notizie che inevitabilmente finirebbero per confonderlo.
L' ultima parte e' riservata poi ad una serie di programmi applicativi e di utili routines oltre che alle necessarie tavole.

Abbiamo cercato di spiegare con parole facili una serie di concetti, quelli relativi ai files, che risultano da sempre lo scoglio maggiore ad un corretto apprendimento della programmazione, ma senza i quali a nostro modesto parere, l' utente resta solo un mezzo programmatore, non potendo utilizzare in pieno le reali capacita' di un computer di qualsiasi tipo si tratti.

Altra cosa un po' noiosa di tutte le riviste e manuali in commercio e' quella relativa ai programmi la cui digitazione risulta spesso non priva di errori e comunque sempre noiosa.
Per questo motivo abbiamo messo a punto un servizio, da utilizzare presso il fornitore oppure direttamente presso la nostra sede, con le cartoline che si trovano al termine del manuale, con le quali si possono richiedere su nastro o su disco i programmi presenti piu' altri di cui per brevita' non abbiamo riportato il listato.

Questo nostro lavoro, frutto di una ricerca non superficiale su tutto il materiale oggi disponibile, nonche' derivato da esperienze personali non e' certamente privo di errori e di imprecisioni.
Per questo, come per gli altri volumi dedicati ai prodotti COMMODORE confidiamo nella collaborazione dei lettori per segnalarci tutte le imprecisioni e per fornirci tutti i suggerimenti che essi riterranno opportuni.

Da parte nostra l'impegno a rispondere a tutte le richieste che ci verranno fatte ed invitiamo quindi i cortesi lettori a prendere nota dei numeri telefonici e dell' indirizzo.

UNITA' A CASSETTE

CAPITOLO PRIMO

INTRODUZIONE

Un sistema di computer e' composto da qualcosa in piu' di una tastiera, di uno schermo e della stessa unita' centrale, cuore del computer.

Per avere un programma a disposizione tutte le volte che si vuole senza doverlo reinserire nella memoria della unita' centrale con una lunga e noiosa serie di operazioni durante la quale e' oltretutto facile sbagliarsi e' necessario disporre di una unita' di registrazione che potra' essere una unita' a cassetta o un floppy disk.
Prendiamo per ipotesi un programma per la gestione di indirizzi o MAIL PROGRAM.

Questo programma, per non riscriverlo tutte le volte, dovra' essere immagazzinato in una cassetta o in un dischetto.
Inoltre un MAIL PROGRAM e' usato per creare una lista di nomi e di indirizzi che per essere utilizzata deve essere immagazzinata anche questa.
Infine per un completo utilizzo sara' necessario disporre di una stampante per le lettere, gli elenchi.
Per questo motivo difficilmente potremo limitarci alla sola unita' centrale.

Volendo sintetizzare un computer ha essenzialmente tre grandi capacita':

CALCOLO

SCELTA

COMUNICAZIONE

Fino a questo momento, dato che ci rivolgiamo ad utenti che si suppone abbiano almeno una certa conoscenza del BASIC, dovrebbero essere conosciute le prime due capacita', vediamone ora la terza.
La capacita' di comunicare con l' esterno e', in un computer, certamente la piu' complessa perche' e' necessario per prima cosa conoscere una serie di regole precise per l' invio e la ricezione dei dati, e poi adeguarsi a queste regole, cioe' metterle in pratica, renderle operative.
Le capacita' di un computer di comunicare con il mondo esterno (esterno al microprocessore naturalmente) sono veramente notevoli, tuttavia noi ci limiteremo per il momento a prendere in considerazione solo la possibilita' di leggere, scrivere e modificare dati e programmi su nastro che e' la periferica piu' usata per il suo costo, mentre nel resto del manuale vedremo sia il disco che la stampante.

NOTA

La capacita' del processore di comunicare con la memoria RAM e ROM e con il video e la tastiera sono state abbondantemente trattate nei precedenti volumi di questa serie per i quali rimandiamo alla bibliografia al termine del volume.

Per quanto riguarda le altre periferiche collegabili ad un computer come:

MODEM

RS-232

LINEE SERIALI E IEEE

oltre a quanto altro si puo' collegare come Paddles, Joystick e Light-Pen, ci rendiamo conto che nell' ambito di questo volume abbiamo potuto dare solo una descrizione sommaria e rimandiamo ad altri volumi di prossima (speriamo) pubblicazione e che riguarderanno l' Hardware.

IL CONCETTO DI FILE

Sia su dischi che su cassette le informazioni sono immagazzinate come " FILES".
Per comprendere il concetto di FILE immaginiamo un comune schedario da tavolo entro il quale conserveremo, per esempio delle schede contenenti indirizzi di nostri amici, clienti o altro.
Avremo quindi che lo schedario sara' il contenitore del nostro FILE, cioe' la cassetta o il dischetto. Non l' unita' a cassette ma il nastro stesso.
L' insieme delle schede sara' il FILE vero e proprio e che quindi potra' ampliarsi in rapporto alla grandezza fisica del contenitore.
E' intuitivo che le singole schede saranno i RECORDS, mentre i dati della scheda saranno i campi o FIELDS.

Per un utente di computer il concetto di FILE deve essere considerato come di primaria importanza, ma non e' difficile da comprendere.
Quando si apre (OPEN) un file, tutte le informazioni ivi immagazzinate diventano accessibili e lo rimangono fino a quando non si chiude (CLOSE).
Rifacendosi all' esempio iniziale e' come aprire il contenitore.

Quando un computer scrive un programma o dei dati su cassetta o disco, crea un nuovo file o aggiunge qualcosa ad uno vecchio.
Un file puo' avere una lunghezza qualsiasi, limitata sola dalla capacita' della cassetta o del disco.

Si puo' creare un nuovo file senza scrivere niente dentro, cio' equivale ad avere una serie di schede bianche senza alcun contenuto.
Si possono avere numerosi files per ogni dischetto (dipende da che tipo di unita' CBM si sta adoperando), mentre non esiste limite per la cassetta.
L' ammontare della memoria non ha nessun effeto sulla grandezza di un file di dati.
Un file di dati puo' essere piu' grande della memoria disponibile del vostro computer. Aprendo un file di dati infatti si puo' leggere un solo carattere o piu' informazioni e passarle alla memoria centrale del computer.
Quando si scrive un file di dati, le informazioni che passano dalla memoria del computer a quella di massa possono essere, e di norma e' cosi',aggiunte a dati immagazzinati in precedenza sulla cassetta o sul floppy.

FILES PROGRAMMI

Ci sono due differenti tipi di files:

FILES PROGRAMMI

FILES DATI

Un file programmi, come e' implicito nel nome contiene un serie di comandi in Basic, in Assembler, in Pascal o qualsiasi linguaggio si stia usando, messi insieme come programma.
Si crea un file programmi utilizzando un comando SAVE. Cioe' quando si usa il comando SAVE automaticamente viene creato sulla periferica un file PROGRAMMA.
Per creare un file di dati e' invece indispensabile usare piu' di un comando il che ci riporta al concetto di programma piu' o meno piccolo.

FILES PROGRAMMA

Un file puo' avere un nome, per cui il nome che assegnerete ad ogni file programma sara' posto in testa al programma.

NOTA

Il discorso del nome e' riferito SOLO alla cassetta perche', come vedremo si puo' anche registrare un gruppo di informazioni SENZA dare nessun nome.
Non cosi' su disco.

Stante alle specifiche fornite dalle case costruttrici i computers CBM dovrebbero riconoscere nomi di files fino ad un massimo di 128 caratteri, ma solo i primi 16 caratteri sono visualizzati sullo schermo.
Tuttavia non si riesce a capire bene come facciano per i 128 caratteri in quanto da nessuna parte del sistema operativo di cassetta o sulla directory del disco e' possibile trovare tanto spazio.
Esperimenti condotti dagli autori in tal senso hanno dato sempre esito negativo.
I nomi dei files disco possono avere 16 o meno caratteri per questo sara' un buon principio di restringere tutti i nomi di files a 16 caratteri o meno.
L' ammontare di memoria del vostro computer ha effetto sulla grandezza massima di un PROGRAM FILE. Cio' e' perche' si crea un singolo file programma quando si salva (SAVE) un programma su cassetta o su disco.

Quando si carica un programma in memoria si carica l' intero contenuto del file programma. Non si puo' caricare una parte di un file programma in memoria. Per questo la grandezza massima di un file programma deve essere minore della capacita' di memoria programma del vostro computer.
La domanda e' come si faccia allora a caricare un programma molto grande.
Se e' necessario caricare un programma molto grande e quindi non e' disponibile una memoria del computer sufficientemente vasta si puo' suddividere

il programma in tanti sottoprogrammi, ognuno dei quali potra' essere contenuto nella memoria del computer.
Quando ogni sezione del programma ha completato il lavoro, cioe' la sua esecuzione, semplicemente caricheremo la sezione successiva in memoria e la faremo girare.
In questo modo si puo' eseguire l' intero programma.
Successivamente descriveremo i passi necessari per eseguire un programma in questo modo.

Un vantaggio dei files programma e' che il salvataggio ed il caricamento avviene tramite il DOS, cioe' il DISK OPERATING SYSTEM,' o per la cassetta, tramite il sistema operativo del computer.
Sara' quindi necessario applicare un identificatore al file programma, tramite il suo nome o la sua locazione, per caricarlo nella memoria, ma e' l' unica operazione necessaria.

FILE MAILIG LIST			
RECORD 1	cognome	Campo	1
	nome	"	2
	via	"	3
	CAP	"	4
	città	"	5
RECORD 2	cognome	Campo	1
	nome	"	2
	via	"	3
	CAP	"	4
	città	"	5
RECORD 3	cognome	Campo	1
	nome	"	2
	via	"	3
	CAP	"	4
	città	"	5
RECORD 4	cognome	Campo	1
	nome	"	2
	via	"	3
	CAP	"	4
	città	"	5

Fig. 1 - Come vengono distribuiti i Campi ed i Records all'interno di un File.

CAPITOLO SECONDO

COMANDI PER I FILES PROGRAMMI

I comandi per i files programmi sono in verita' molto semplici ed essenzialmente si riducono a 3:

LOAD

SAVE

VERIFY

Il formato di LOAD e':

LOAD"nome del programma",d

dove:

d = il numero della periferica interessata. Nel caso della cassetta non e' necessario mettere il parametro d.

Esempio

Caricare da cassetta il programma "MAILING LIST".

```
LOAD"MAILING LIST"
```

Il sistema rispondera' con un messaggio:

```
PRESS PLAY ON TAPE
```

e dopo aver eseguito l' operazione sul tasto indicato verra' visualizzato un:

OK

quindi il sistema iniziera' il caricamento del programma che sara' copiato dal nastro sulla memoria RAM del computer.

*** APPROFONDIMENTO ***

Il Sistema Operativo del computer genera automaticamente un' istruzione di OPEN (vedi dopo) usando l' indirizzo secondario appropriato per effettuare l' operazione di LOAD.
Sulla periferica cosi' attivato viene iniziata una ricerca per trovare il programma il cui nome e' specificato nell' istruzione LOAD.
Dopo che il programma e' stato trovato esso viene letto automaticamente dalla periferica e caricato nella memoria partendo dall' indirizzo specificato nell' intestazione del file.

Gli errori di lettura che possono verificarsi durante l' esame del primo blocco vengono automaticamente corretti dal secondo blocco perche' ricordiamo che le registrazioni sono fatte in doppio.
Alla fine del ciclo viene eseguita una somma di prova o:

CHECKSUM

Se si ha un errore di Checksum o se siamo in presenza di un errore che non e' correggibile il sistema operativo visualizza un messaggio di:

? LOAD ERROR

ed arresta il caricamento del programma.

Se il caricamento del programma viene fatto in modo diretto, alla fine dell' operazione viene eseguita la funzione di CLEAR (CLR) inizializzando tutte le variabili.

Se l' operazione di caricamento e' invece chiamata da un programma, allora il computer COMMODORE tratta questa operazione come una OVERLAY.
Il nuovo programma cioe' viene caricato nello spazio usato dal precedente programma, ma i valori di tutte le variabili vengono mantenuti uguali a quelli raggiunti nel programma precedente.
Cio' permette ad un programma di chiamarne un' altro passandogli tutti i parametri.
L' unico vincolo a quanto appena esposto sta nel fatto che il programma chiamato deve avere dimensioni uguali o minori del programma chiamante.
Poiche' il Basic rimpiazza completamente il programma corrente, non e' possibile avere un singolo programma principale e diverse subroutines in overlay. Tuttavia includendo il programma principale in ciascuna overlay, si puo' ottenere lo stesso effetto con una picola perdita di velocita.
L' uso dei nomi dei files e degli overlay permettono di scrivere dei programmi strutturati di grande ampiezza e con una notevole complessita'.

VERIFY

Questo comando ha la stessa sintassi del comando LOAD:

VERIFY"nome del programma",d

In effetti l' istruzione di VERIFY e' un caso speciale di LOAD che dovrebbe venir eseguita dopo aver registrato qualsiasi programma.
Il comando VERIFY fa si che il Basic esegua le stesse operazioni dell' istruzione LOAD, con la differenza che i dati non vengono caricati in memoria ma vengono confrontati con il contenuto della memoria.
Se vengono incontrati degli errori, sia nel primo che nel secondo passo, il computer visualizzera' un messaggio:

? VERIFY ERROR

e sara' necessario registrare una seconda volta il programma in quanto la copia precedente non e' utilizzabile.

SAVE

Anche l' istruzione SAVE provoca un' operazione di apertura e chiusura automatica di un file.

Ha il seguente formato:

SAVE"nome del programma",d

dove d e' il numero della periferica.
Se la periferica e' una unita' a nastro il Sistema Operativo del Computer inizia automaticamente a registrare, naturalmente dopo aver premuto i relativi tasti, un' intestazione ed apre un file su nastro con un nome appropriato.
Se il dispositivo e' un' unita' IEEE-488, cioe' di norma un disco, viene inviato uno speciale messaggio di apertura che sta ad indicare che il computer sta inviando un file programma.
Immediatamente dopo il programma viene scritto direttamente dalle sue locazioni di memoria o sul nastro o sullla periferica collegata alla porta IEEE-488.

NOTA

Per quanto riguarda il caricamento di programmi su cassetta da parte di unita' VIC 20 e CBM 64 si ha un BLANK di schermo al momento stesso che inizia l' operazione di ricerca.
E' un problema che la COMMODORE avrebbe potuto benissimo evitare e che spesso induce il principiante a spaventarsi ingiustificatamente.
Quando il programma viene trovato si ha un' attesa di circa 30 secondi e poi il nastro riparte da solo.
Se si vuole evitare questa attesa e' sufficiente premere il tasto con il simbolo della COMMODORE posto a sinistra del computer.

Per un ulteriore approfondimento di concetti HARDWARE vedi quanto riportato al termine di questa sezione.

CAPITOLO TERZO

DATA FILES

Un DATA FILE o un files di dati come dovrebbe essere implicito nel nome, contiene informazioni che devono essere interpretati come dati in opposizione a comandi di programma.
I files dati sono creati, scritti e letti tramite i programmi, cioe' non possono essere scritti o letti direttamente come un file programma per mezzo delle istruzioni LOAD e SAVE.

RECORDS E FIELDS

I data files sono divisi in records che a sua volta sono suddivisi in FIELDS o campi.
Un singolo field contiene informazioni che possono essere rappresentate tramite il nome di una singola variabile.
Per questo motivo un field puo' contenere un numero intero, un numero in virgola mobile o una singola variabile stringa.
Un record contiene uno o piu' field.
I records di solito rappresentano unita' d' informazioni ripetitive entro il file, ma puo' non essere sempre cosi'.
Prendiamo per esempio una mailing list.
L' intera mailig list puo' essere considerata come un singolo file di dati.
Ogni nome e indirizzo entro la mailing list sara' un record entro il file.

Un esempio di file che gestisce un indirizzario potrebbe essere caricato con i seguenti dati. In questo caso ogni record conterra' 5 campi:

Il cognome

Il nome

La via

Il CAP

La citta'

Un file puo' contenere uno o piu' record. Ogni record puo' contenere uno o piu' fields.
Il numero di records in un file e la massima lunghezza di un record varia con il tipo di file come descriviamo successivamente in questo manuale. Tuttavia in pratica la grandezza di un file e' limitata solo dalla capacita' di memoria di massa. Nessuna restrizione invece alla lunghezza di un record su cassetta.
Un record puo' avere una lunghezza qualsiasi che entri pero' nella lunghezza del nastro e questo per il semplice motivo che non puo' essere diviso in due nastri fisici.

*** APPROFONDIMENTO ***

TRASFERIMENTO DATI VERSO E DA PERIFERICHE.

Istintivamente ci si puo' aspettare che sia il disco che il nastro si muovano in risposta ad ogni comando di lettura o di scrittura. Si intende come movimento fisico.
Cioe' l' unita' a cassetta dovrebbe muovere fisicamente il nastro e lo stesso per il disco.
Qualche volta si possono osservare entrambe queste attivita', mentre altre volta rimane tutto invisibile. Cio' e' dovuto al fatto che un piccolo ammontare di memoria agisce come DATA BUFFER connettendo il computer con l' unita' a cassetta o con il disco.

NOTA

Il BUFFER e' una zona di memoria che serve di solito come magazzino temporaneo di dati.

Quando il computer legge da una di queste periferiche sono letti abbastanza dati per riempire il buffer.
I dati che devono essere scritti su cassetta o su disco sono prima scritti sul buffer data.
Non appena il data buffer e' pieno, tutto il contenuto del buffer e' inviato alla cassetta o al disco, ecco che allora vedremo qualche attivita' delle periferiche. Vedremo cioe' qualcosa in funzione fisicamente.
Mentre non vedremo nessuna altra attivita' fino a

quando il BUFFER non sia riempito.
Il buffer di cassetta e' contenuto nella memoria del computer, e' lungo 192 bytes e puo' contenere percio' fino a 191 bytes di data.
Invece il buffer del disco e' nella stessa unita' a disco e non nella memoria del computer.
Ogni buffer data del disco e' grande 256 bytes e puo' manipolare fino a 254 bytes di dati.
I Buffer del disco e della cassetta sono relativamente grandi di conseguenza queste periferiche sono inattive per la maggior parte del tempo che il computer impiega per leggere o scrivere data.
Cio' e' di notevole vantaggio perche' consente in pratica di risparmiare la meccanica delle periferiche ed accellera inoltre le procedure di lettura/scrittura.

FILES LOGICI E UNITA' FISICHE

Si usa il termine "INPUT/OUTPUT PROGRAMMING" per descrivere la logica di programmazione che consente il trasferimento dati fra il computer e le unita' esterne.
I dischi, le cassette e le stampanti sono quindi unita' fisiche esterne.
Per consentire una qualsiasi operazione di INPUT/OUTPUT (ingresso/uscita) il programma deve identificare l' unita' fisica esterna alla quale si deve accedere.
Pensiamo il problema in termini di programmazione.
Questo concetto e' facile da capire se si pensa alla tastiera ed al video come unita' esterne rispetto al computer propriamente detto, come in effetti esse sono.
Quando viene eseguito un comando INPUT i dati che abbiamo immesso con la tastiera sono specificati in un parametro di input.

Quando il comando :

```
10 INPUT A
```

e' eseguito, alcuni numeri che l' operatore fa entrare attraverso la tastiera sono assegnati a una variabile (in virgola mobile).
Nello stesso modo il comando di PRINT visualizzera' variabili o costanti.

Cosi' il comando PRINT:

```
20 PRINT A
```

prende i valori assegnati alla variabile in virgola mobile A e mostra questo valore sullo schermo.
Quando viene eseguito un comando di INPUT l' unita' fisica esterna e' vista come nel caso precedente e' stato per la tastiera.

Quando viene eseguito invece un comando di PRINT l' unita' fisica esterna viene vista come il video.
La programmazione degli INPUT/OUTPUT diventa molto piu' complessa quando i dati sono trasferiti da/a e dalla cassetta, il disco, la stampante e altre unita' fisiche esterne al di fuori della tastiera e del video.
Per queste piu' complesse operazioni di INPUT o OUTPUT dovrete prima di tutto aprire un :

CANALE DI COMUNICAZIONE

tra il programma e l' unita' fisica selezionata.
Dopo aver eseguito l' operazione richiesta di INPUT/OUTPUT dovrete richiudere il canale.
Il Basic del CBM identifica canali singoli usando un numero di canali che puo' andare da 0 a 255.

Si apre un canale usando il comando :

OPEN

I Parametri di questo comando identificano l' unita' fisica alla quale si deve accedere, mentre per la natura di questo accesso vedremo in dettaglio nelle parti seguenti.

Fino a che il canale non sia chiuso, ogni comando di INPUT/OUTPUT necessita solo che sia specificato

il numero del canale per descrivere completamente la natura della operazione di input o di output.
Ogni unita' fisica ha di per se un solo numero di riconoscimento fisico.Questo numero e' usato come un parametro quando si apre un canale per identificare l' unita' fisica alla quale si vuole accedere.
I numeri di canale sono per questo riportati frequentemente come "LOGICAL FILES NUMBERS" o "LOGICAL UNITS NUMBERS".

Il nome LOGICAL FILE descrive un canale molto accuratamente, perche' un canale stabilisce un legame tra un programma e un file di dati.
I FILES LOGICI sono un concetto di programmazione.
Si puo' iniziare una qualsiasi operazione di I/O usando un comando di OPEN.
Uno dei parametri del comando OPEN e' il canale o il numero di FILE logico.
Gli altri parametri identificano l' unita' fisica, i dati ai quali si deve avere accesso ed il mezzo in cui occorre questo accesso.

Dopo che una operazione di input o output e' stata completata bisogna eseguire un comando CLOSE che richiudera' il canale.
Il comando CLOSE richiede solo un parametro:

IL CANALE O NUMERO DI FILE LOGICO

Questo numero di file logico unisce quindi un comando CLOSE ad un comando OPEN.
Tra un comando OPEN ed un CLOSE tutti i comandi di I/O usano un canale o un logical file number per identificare l' unita' alla quale si deve accedere e l' operazione che deve essere eseguita.

Il LOGICAL FILE NUMBER mette in relazione i comandi di :

OPEN

CLOSE

GET #

PRINT #

INPUT #

Con qualsiasi altro.
Fino a quando state usando un numero di file logico in un comando, non potete riutilizzare lo stesso numero di file logico per fissare un diverso canale di I/O fino a che il LOGICAL FILE non sia chiuso.
Se lo farete il Basic del computer rispondera' con un :

FILE OPEN ERROR

D'altra parte nessun altra limitazione e' presente nel metodo di assegnare un numero di File Logico entro il vostro programma.
Il numero di DEVICE o di periferica identifica l' unita' fisica alla quale il computer inviera' i suoi dati o dalla quale li ricevera'.
Il numero di device appare come un parametro nel comando OPEN.

Ogni unita' fisica che possa comunicare con un computer CBM ha assegnato in permanenza un numero di DEVICE.

Non appena venga trovato un numero di device in un comando OPEN il computer attiva un' appropiata logica elettronica per stabilire una comunicazione con l' unita' specifica identificata nel numero di device.
Teoricamente sono disponibili 256 numeri di periferiche in un range compreso fra 0 e 255.
Tuttavia solo i numeri di device fra 0 e 30 sono correntemente usati.
In aggiunta alla definizione del numero di device, molte unita' fisiche rispondono ad un vasto gruppo di indirizzi secondari.

Device	Device Number	Secondary Address	Operation Performed
Keyboard	0	None	
Cassette Drive #	1 (Default)	0	Open per letture
		1	Open per scritture
Cassette Drive #2	2	2	Open per scritture ma con EOT dopo la chiusura.
Video Display	3	None	
Line Printer Models 2022 and 2023	4	0	
		1	
		2	
		3	VEDERE CAPITOLO STAMPANTI
		4	
		5	
		6	
Disk Drives (all models)	8	0	LOAD di un file programma
		1	SAVE di un file programma
		2-14	Non assegnati
		15	OPEN sul canale di comando
Other devices connected to IEEE 488 Bus	5,6,7 and 9 through 31		VEDERE SPECIFICHE SUI MANUALI DELLE SINGOLE PERIFERICHE
	32 to 255 unavailable at this time		

Fig. 2 - Numero di periferiche con indirizzo secondario.

INDIRIZZO SECONDARIO

Oltre ad avere un numero di unita' fisica a molte periferiche puo' essere assegnato un indirizzo secondario o SECONDARY ADRESS.

L' indirizzo secondario e' un comando che parte dal computer e che dice all' unita' fisica quale operazione deve prepararsi ad eseguire.
Gli indirizzi secondari sono riportati in sommario in appendice per le unita' fisiche che sono piu' comunemente connesse ad un computer CBM.
Non dovreste impegnarvi in uno studio particolare degli indirizzi secondari, perche' successivamente quando descriveremo i programmi di I/O in dettaglio la funzione di indirizzo secondario diventera' familiare ed ovvia per il suo frequente uso.
Il programma seguente illustra molto bene l' uso dei parametri nei comandi di I/O.

```
100 OPEN4,1,2,"MAILING LIST"
200 PRINT#4,CN$
210 PRINT#4,NO$
220 PRINT#4,VP$
230 PRINT#4,CA$
240 PRINT#4,LO$
300 CLOSE4
```

I 5 comandi di PRINT # che appaiono nelle linee dalla 200 alla 240 scrivono 5 parti di nome ed indirizzo in un file chiamato :

MAILING LIST

localizzato su un nastro dell' unita' a cassetta.

Tutte le volte che si incontra un comando di PRINT# il computer sa cosa deve fare perche' controlla il numero di File logico che appare dopo il carattere #.
Nel programma questo numero di File logico e' 4, percio' nel comando di OPEN e' specificato il file logico 4 che descrive la natura dell' operazione.
Questo comando di OPEN e' presente nella linea 100 del nostro programma.
Se il computer non dovesse trovare un comando di OPEN con il richiesto numero di unita' logica, questi non potrebbe mettere in funzione le operazioni di I/O poiche' non saprebbe cosa fare.
Nel programma c'e' un comando di OPEN sul File Logico n. 4. Questo comando specifica l' unita' logica n. 1 che appunto sta ad indicare che e' selezionata la cassetta.
L' indirizzo secondario e' 2 perche' in questa occasione e' possibile scrivere sulla cassetta del drive 1 ma non e' possibile leggerci.
Quando questa operazione e' chiusa verra' scritto un fine nastro sulla cassetta per prevenire che un qualsiasi dato possa essere successivamente aggiunto.

Il comando OPEN specifica inoltre che il Data file al quale si deve accedere ha il nome MAILIG LIST.
Sulla linea 300 e' presente un comando di CLOSE (chiudi). Questo comando specifica il numero 4 come File Logico, di conseguenza tutto quanto e' stato aperto con il comando OPEN nella linea 100 sara' chiuso con questo comando alla linea 300.
Poiche' il comando OPEN alla linea 100 specifica un numero di indirizzo secondario 2, il comando CLOSE alla linea 300, quando sara' eseguito causera' una EOF (END OF FILE) sulla cassetta.

In questo modo il file logico n 4 che e' presente nei comandi dalle linee 200 alla linea 300 congiunge questi comandi con un comando OPEN alla linea 100.
Parametri addizionali appaiono sui comandi OPEN alla linea 100 per descrivere le operazioni che devono essere eseguite.

Prima di procedere oltre con la programmazione vediamo alcuni concetti, in particolare per le variabili di controllo che sono essenziali per la gestione delle periferiche.

FISICAL UNIT STATUS

Una stampante puo' ricevere informazioni da un computer, cioe' si possono preparare delle stringhe da far stampare su una stampante, tuttavia da una stampante i dati non possono passare ad un computer.
Per questo motivo non e' necessario specificare il numero di indirizzo secondario quando si esegue un comando di OPEN su una periferica tipo stampante.
Al contrario una cassetta puo' ricevere dati dal computer o trasmetterglieli, per cui l' indirizzo secondario usato nel comando OPEN che inizializza la cassetta dovra' specificare se l' operazione e' di lettura o di scrittura.
Quando si esegue un comando di PRINT , GET o INPUT e' necessario fare attenzione a quello che si vuol fare. In altre parole non sara' possibile eseguire dei comandi di INPUT o di GET quando l' unita' a cassetta sara' stata preparata solo per operazioni di scrittura.
Se questo dovesse avvenire avremo una registrazione

di errore di STATUS.
L' unita' fisica riporta l' informazione sullo status di seguito ad ogni operazione di INPUT o di OUTPUT quando questa sia stata eseguita con successo o con insuccesso.
In pratica tutte le volte che si accede ad una unita' periferica, considerando pero' in questo caso come periferiche anche la tastiera ed il video, viene attivato un registro di 8 bit che e' appunto:

REGISTRO DI STATUS

Questo registro ha come riferimento la variabile Basic ST. Per esempio il comando 10:

```
10 X= ST
```

Assegnera' al registro di status il valore della variabile X.

Per quanto riguarda le cassette riportiamo gli errori che tramite ST possono essere rilevati:

ST=4 Blocco Corto.

Leggendo un blocco di dati da nastro si incontra un segnali di spaziatura prima di aver letto i caratteri che ci si aspettava di trovare.

ST=8 Blocco Lungo.

Leggendo un blocco non si trova il segnale di

spaziatura dopo aver letto il numero di caratteri che si aspettava di trovare.

ST=16 Errore di lettura irrecuperabile.
Si sono riscontrati piu' di 31 errori nel primo blocco del blocco di controllo oppure si e' trovato un errore che non puo' essere corretto perche' presente in tutti e due i blocchi di registrazione. Ricordiamo che la registrazione su nastro e' SEMPRE doppia.

ST=32 Errore di Checksum.

Durante la lettura o il LOAD di dati viene calcolata una somma di controllo sui bits dei bytes letti e viene confrontata con la somma di controllo registrata sul nastro al momento della scrittura. Se viene generato questo errore le due somme non coincidono.

NOTA

Questi ultimi due non sono, come si vede dalla descrizione dei veri e propri errori.

ST=64 End of File.

Viene cioe' incontrato il fine di un file.

ST=-128 End of Tape.

Viene incontrato il segnale di fine nastro.

Da quanto detto e' evidente che il nostro computer CBM non e' in grado di riconoscere errori durante la scrittura su nastro.

La normale tecnica di programmazione e' quella che fa eseguire ad un' istruzione di INPUT #o GET #un test sul valore dello stesso. Poiche' la parola di stato e lo stato cambia per ciascun nuovo comando di I/O lo stato e' una grandezza variabile rapidamente. Ad esempio:

```
100 INPUT#2,A
110 INPUT#5,B
120 IF TS=0 THEN 200
```

Questo programma testa il risultato del trasferimento di dati SOLO dal file logico 5. Il risultato della lettura del file logico 2 e' perso. Un sistema corretto per usare la parola ST e' la seguente:

```
100 INPUT#2 ,A,B,C
110 IF ST=0 THEN 200
120 IF ST=64 THEN 300
130 IF ST=2 THEN 400
```

In tal modo ciascun errore puo' essere individuato e si possono intraprendere le successive operazioni che si rendano necessarie attraverso l' istruzione:

140 IF ST AND mask THEN nnn

dove mask rappresenta il bit che si vuol testare.

Device Operation	Status							
	00000001 Read as 1	00000010 Read as 2	00000100 Read as 4	00001000 Read as 8	00010000 Read as 16	00100000 Read as 32	01000000 Read as 64	10000000 Read as -128
Read from Cassette drive #1 or #2	Operation OK	Operation OK	Short Block. Data block read had fewer bytes than expected	Long Block Data block read had more bytes than expected	Unrecoverable read error	Checksum error. One or more data bits read incorrectly	End of file encountered	End of tape encountered
Verify cassette drive #1 or #2					Any verify mismatch		None	
Disk drives (all models)	Receiving device not available	Transmitting device not available	None	None	None	None	End of file	Disk drive not present
IEEE 488 Bus	Time out on listener	Time out on talker					End of Identify	Device not present

Fig. 3 - Valore del byte di Status letto da periferiche con la variabile ST.

CAPITOLO QUARTO

MANIPOLAZIONE DI DATI SU CASSETTA

Veniamo ora a descrivere i passi di programma necessari per la manipolazione dei files su cassetta.
Descriveremo come i FILES DI DATI sono creati, letti o modificati sotto controllo di programma.
Molti dei comandi Basic che appaiono in questo capitolo sono dati per scontati, cioe' si suppone che il lettore abbia una discreta conoscenza generale del Basic.
In appendice a questo volume e' riportata una lista generale di questi comandi tuttavia e' bene rifarsi ai manuali relativi alla programmazione.
Potete programmare il computer per scrivere dati su cassetta o per rileggerli, ma non potete programmare il movimento fisico della cassetta.
E' importante che comprendiate il modo in cui opera fisicamente il DRIVE, cioe' l' unita'.
In altre parole dovete tenere presente che per eseguire operazioni sulla cassetta non sarete mai in grado di manipolare il movimento fisico del nastro.

NOTA

In effetti questo discorso non e' completamente vero perche' si puo' programmare, ad esempio agendo su determinati registri, l' arresto del motore.

I files sono immagazzinati in modo sequenziale su nastro, cioe' uno di seguito all' altro.
Un HEADER cioe' una testata, precede il primo file e un fine nastro (EOT) segue l' ultimo file.
Ogni fine di file e' segnato da un EOF.
La testata e' scritta automaticamente all' inizio del nastro, cioe' quando ci scrivete per la prima volta.
A questo punto potete notare che l' attivita' della cassetta o almeno questa parte di attivita' della cassetta, non vi riguarda.
In altre parole l' esistenza di un HEADER viene scritta automaticamente, cioe' non ha bisogno di vostre operazioni.

Il computer puo' trovare i File mentre il nastro sta girando piano, cioe' alla velocita':

PLAY

ma non e' in grado di trovarli quando sta girando in FF, cioe' in :

FAST FORWARD

E questo perche' durante la fase di LETTURA/SCRITTURA non e' in contatto fisicamente con il nastro.

Come abbiamo detto la fine file e' identificata da un segno particolare (o meglio da un carattere particolare).
In altre parole, come si puo' vedere dalla precedente tabella, uno status pari a 64 identifica

un END OF FILE.
Uno Status di -128 identifica una fine nastro.

Il computer non puo' eseguire una operazione di riavvolgimento diretto veloce ne puo' trovare niente sulla cassetta mentre il nastro si sta riavvolgendo.
Si deve iniziare il movimento sulla cassetta manualmente premendo il tasto relativo in seguito ad una istruzione fornita dalla unita' centrale.
Si raccomanda di non premere nessun tasto prima che un messaggio venga visualizzato.
In seguito potremo comportarci diversamente dopo aver pero' preso un po' di pratica nelle operazioni.
Esaminiamo ora l' impatto sulle operazioni dell' unita' a cassetta.
Quando si stanno scrivendo dati sulla cassetta, il nastro deve essere correttamente posizionato ad inizio scrittura e cio' e' messo sotto la responsabilita' dell' operatore.
A questo punto e' necessario ricordarsi che se non si posiziona correttamente il nastro e' facile avere delle sovrascritture.
Inoltre se la parte iniziale trasparente e' posizionata sulla testina di scrittura, l' unita' cerchera' di scrivere le informazioni che pero' non saranno registrate.
E' importante ricordarsi di questo perche' il computer non e' capace di distinguere la superfice magnetica dalla superfice non magnetica.
Il metodo che consente la massima sicurezza e' di iniziare a scrivere su nastro vergine o su una cassetta i cui dati non servano piu' e di posizionare il nastro all' inizio della superfice magnetica.
Si possono cosi' tranquillamente scrivere records e files uno dietro l' altro fino al termine fisico

del nastro stesso.
Il Sistema Operativo dell' unita' centrale nella parte relativa all' uso dell' unita'a a dischi si assicurera' che venga lasciato uno spazio sufficiente fra la fine di un record o di un file e l' inizio del successivo, per cui non sara' necessario che l' operatore si occupi di questo.
Quando si leggono files di dati gia' registrati, e' necessario assicurarsi che il nastro sia riavvolto fino all' inizio del primo file che si vuole rileggere.
Il computer puo' trovare un qualsiasi definito file di dati purche' questo sia DOPO il punto in cui l' abbiamo fatto partire, ma non puo' certo tornare indietro a cercarselo da solo.

Non si deve mai cercare di riscrivere anche una piccola parte di file su nastro perche' l' operazione e' troppo rischiosa.
Supponiamo per esempio di aver immagazzinato su un file cassetta 10 nomi ed indirizzi e che si desideri variare il quinto nome ed il relativo indirizzo.
Teoricamente si potrebbe leggere i primi 4 nomi ed indirizzi e questa operazione ci dovrebbe lasciare il nastro posizionato all' inizio del quinto nome.
Si potrebbe quindi scrivere il nuovo quinto nome sul vecchio.
In pratica e' meglio non farlo.
Infatti l' unita'a a cassetta non e' molto precisa e c' e' una buona probabilita' che il nuovo nome ed indirizzo sia scritto un po' prima o un po' dopo del vecchio.
Infatti, a parte il fatto che dovrebbero essere della stessa identica lunghezza e questo risulta difficile da ottenere, e' sufficiente che un solo carattere vada fuori posto, cioe' o troppo prima o dopo, che non saremo in grado di rileggere i dati.

Per aggiornare quindi un file di dati e' necessario caricare il file stesso nella memoria del computer, aggiornarlo e quindi riscriverlo.
E' molto improbabile che sul CBM 64 a differenza di quanto avveniva per il VIC 20, sia necessaria una espansione di memoria per cui l' utilizzo della ricordata tecnica di OVERLAY dovrebbe essere piu' che sufficiente.

Potrebbe sembrare che la trattazione di questi problemi sia stata troppo lunga, ma per esperienza diretta e per le numerose lettere che ci sono pervenute, possiamo assicurarvi che i guai sulla registrazioni su nastro possono far perdere piu' tempo che non la scrittura stessa dei programmi.

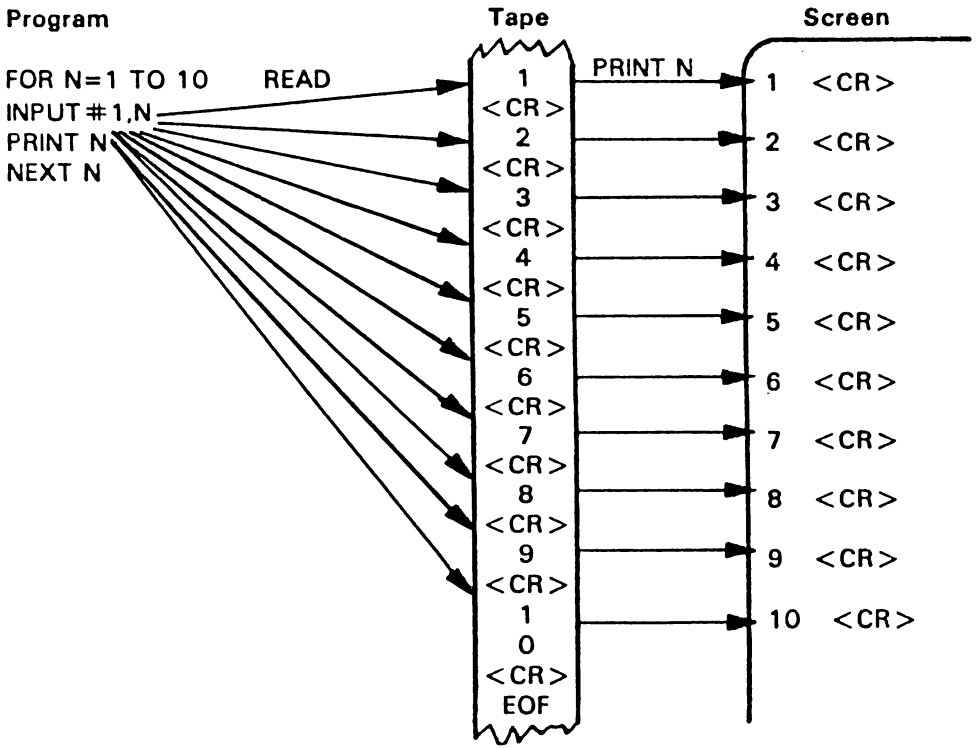


Fig. 4 - Diagramma di distribuzione e di comparazione di dati numerici su nastro e su schermo.

CAPITOLO QUINTO

IL FORMATO DEI FILE SU CASSETTA

La descrizione dei file dati presentata all' inizio di questo capitolo e' una descrizione concettualmente esatta e completa di come i dati sono trattati in generale da un computer.
I file di dati sono suddivisi in RECORD e FIELDS.
Si puo' mantenere questo tipo di organizzazione che definiremo CLASSICA usando dei programmi appropriati e ci permettiamo di raccomandarvelo.

FIELDS NUMERICI

Ogni field numerico deve essere seguito da un ritorno carrello effettuato con l' inserzione del carattere (CHR$(13)).
Percio' un file che consiste solo di campi numerici dovrebbe essere visto come una sequenza di numeri separati da un carattere di ritorno carrello come e' illustrato nell' esempio seguente dove N sta' per il numero e CR per il carattere di ritorno carrello:

N--CR--N--CR--N--CR

Per questo all' interno di un file di dati puramente numerico non esiste nessuna ripartizione di fields in record o distinzione fra i vari records.

E' interamente demandato quindi alla logica del programma la formazione di record intesi come sequenza di fields sempre che questo sia necessario.
Le variabili stringa da memorizzarre possono invece essere divisi facilmente in fields e questi eventualmente raggruppati in records.
Si possono usare i due punti (CHR$(44)) per separare i fields entro un record, mentre un ritorno carrello (CHR$(13)) seguira' l' ultimo field del record stesso.Nell' esempio seguente viene mostrato la struttura fisica di registrazione su nastro di un file che contiene solo variabili stringa con 5 file per record. Ricordiamo che, come nell' esempio precedente CR sta per il ritorno carrello mentre S per la stringa:

CR--S--:--CR--S--:--CR--S--:

Se si usano i due punti ed il ritorno carrello come separatori per dividera il file stringa in Fields e record come illustrato precedentemente, allora tutti i fields di ogni record devono essere letti con un singolo comando di INPUT#.

Non e' obbligatorio usare come separatori di variabili stringa i due punti e il ritorno carrello, ma sara' probabilmente meglio il tenere separato tutti i campi di una variabile stringa usando un ritorno carrello.
Come per i file di dati numerici,e' rimesso alla logica del programma che dovra' gestirli il raggruppamento di campi in records.
Prendendo come esempio il prgramma di gestione indirizzi si vede come la logica di programmazione richieda essa stessa che vari fields siano organizzati in record in maniera evidente.

Non e' necessario insegnare come un programmatore debba vedere che ogni nome ed indirizzo diventa un record, mentre parte del nome e dell' indirizzo devono essère trattati come singolo field.
Le strade per dividere un nome ed indirizzo nei singoli fields sono numerose e in pratica, purche' risponda allo scopo che ci siamo prefissi una strada vale l' altra.
L' organizzazione di un file deve essere dettata dalle necessita' del programma piuttosto che dalla struttura dei data files su cassetta del vostro computer.
Le difficolta' eventuali di programmazione saranno relative alla sintassi dei comandi PRINT# eINPUT#.

Ora partendo da un semplice programma cercheremo di approfondire la sintassi dei comandi e tramite piccole continue modifiche spiegheremo cio' che e' concesso e cio' che non e' concesso fare.
Digitiamo il seguente programma:

```
10 OPEN1,1,1
20 FORI=1TO10
30 PRINT#1,I+100
40 NEXT
50 CLOSE 1
60 STOP
70 OPEN1
80 FORI=1TO10
90 INPUT#1,J
100 PRINTJ
110 NEXT
120 CLOSE1
132 STOP
```

Il comando OPEN alla linea 10 apre il file logico

1, selezionando l' unita' a cassetta per una operazione di scrittura.
Il ciclo di FOR-NEXT alle linee 20,30 e 40 scrivono 10 numeri su nastro.
I numeri sono seguiti da un carattere di ritorno carrello per far si che il comando di PRINT # alla linea 30 forzi un ritorno carrello ad ogni scrittura su nastro, allo stesso modo che un comando di PRINT forza un a capo sullo schermo.
Il file logico e' chiuso alla linea 50.
La struttura fisica della registrazione su nastro sara' la seguente:

CR--1--CR--2--CR--3...10--CR

I comandi dalla linea 70 alla linea 120 leggono e visualizzano i dieci numeri che erano stati scritti su nastro dai comandi che vanno dalle linea 20 alla linea 50.
Eseguite il programma e osservate cosa succede seguendo le indicazioni che daremo.
Digitate il programma e salvatelo su una cassetta programmi. Prendete poi un cassetta vergine ed inseritela facendo in modo che la superfice magnetica della cassetta sia posizionata sulla finestrella perche' altrimenti scriverebbe sulla parte bianca/trasparente del nastro e quindi non scriverebbe affatto.
Assicuratevi che nessun tasto sia premuto e digitate RUN.
Verra' visualizzato il seguente messaggio:

PRESS PLAY AND RECORD ON TAPE #1

Premere i tasti indicati sulla unita' a cassette.
Il computer visualizzara' un :

OK

sotto la frase precedente.
Il nastro comincera' a girare mentre i numeri da 101 a 110 saranno registrati.
Dopo che i 10 numeri sono stati scritti la cassetta si ferma e sullo schermo viene visualizzato il seguente messaggio:

BREAK IN 60

READY.

con il cursore che lampeggia sotto la scritta Ready.
Il comando di STOP alla linea 60 ha causato l' interruzione.
Premere ora il tasto di STOP della cassetta per far rialzare i tasti di PLAY e di RECORD.
Riavvolgere la cassetta con il tasto REWIND e ripremere il tasto STOP per far rialzare il REWIND.
Infatti anche se quest' ultimo si rialza da se la tensione sul nastro puo' far si che questo si spezzi.
Eseguiamo la seconda parte del programma digitando:

GOTO 70

Sara' visualizzato il messaggio:

PRESS PLAY ON TAPE 1

Fate attenzione a premere SOLO il PLAY sulla

cassetta ed il computer rispondera' con un:

OK

dopo l' operazione il nastro comincera' a girare e dopo aver superato la parte bianca della cassetta e letto i 10 numeri scritti prima visualizzera' su una colonna verticale dello schermo:

```
101
102
103
104
105
106
107
108
109
110
BREAK IN 130
READY.
```

Il messaggio finale e' dovuto all' esecuzione del comando STOP che e' presente alla linea 130 del nostro programma.
Se avete dimenticato di riavvolgere il nastro prima di digitare GOTO 70,1' unita' cerchera' per tutto il nastro, che si e' detto doveva essere vergine per questo esempio, e non trovando niente andra' fino alla fine fisica del nastro.
Se siete incorsi in questo errore dovete per prima cosa premere il tasto STOP della unita' a cassetta e poi fermare il programma premendo il RUN/STOP del calcolatore.
Successivamente riavvolgere il nastro, ma prima di far ripartire il programma dovete eseguire una

operazione di chiusura in forma diretta digitando:

```
CLOSE 1
```

e quindi ripartire con un:

```
GOTO 70
```

Chiarito il primo errore che si puo' commettere e che e' piu' frequente di quanto non si creda passiamo ad effettuare delle piccole modifiche al nostro programma.
Listiamo il programma ed aggiungiamo al comando PRINT nella linea 100 un punto e virgola in modo da avere:

```
100 PRINT J;
```

Riavvolgiamo il nastro e digitiamo di nuovo:

```
GOTO 70
```

Dopo avere eseguito l' operazione di lettura come precedentemente descritto sullo schermo apparira':

```
101 102 103 104 105 106 107 108 109 110

BREAK IN 130
READY.
```

A titolo sperimentale proviamo a cambiare i comandi dalla linea 80 alla 110 in modo tale che i dieci numeri siano inseriti usando un solo comando di INPUT, per cui il nostro programma sara' ora come segue:

```
10 OPEN1,1,1
20 FORI=1TO10
30 PRINT#1,I+100
40 NEXT
50 CLOSE 1
60 STOP
70 OPEN1
80 FORI=1TO10
90 INPUT#1,J
100 PRINTJ
110 NEXT
120 CLOSE1
132 STOP
```

Riavvolgiamo ancora la cassetta ed eseguiamo la seconda parte del programma, cioe' la parte di sola lettura.
Come nell' esempio appena visto sull' uso del punto e virgola, i dieci numeri saranno letti dal nastro e visualizzati in una sola riga.
Questo esempio serve a dimostrare che non esiste nessuna differenza nella lettura dei dieci numeri sia eseguendo il comando INPUT #con 10 variabili come suoi parametri che eseguendo lo stesso comando con una sola variabile ma 10 volte.

Continuiamo il nostro ciclo di esperimenti modificando il sistema di separazione all' interno

del file numerico.
Proviamo ora a cambiare la prima parte, cioe' quella relativa alla scrittura.Il programma sara' come segue:

```
10 OPEN1,1,1
20 FOR I =1 TO 10
30 M(I) =I + 100
40 NEXT
45 C$=CHR$(59)
48 PRINT#1,  M(1);C$;  M(2);C$;  M(3);C$;  M(4);C$;
M(5)
47 PRINT#1,  M(6);C$;  M(7);C$;  M(8);C$;  M(9);C$;
M(10)
50 CLOSE 1
60 STOP
70 OPEN 1
80 FOR I =1 TO 10
90 INPUT#1, J
100 PRINT J
110 NEXT
120 CLOSE 1
130 STOP
```

dove CHR$(59) rappresenta un punto e virgola(;).
Eseguiamo nuovamente l' operazione di scrittura nastro ricordando di far apparire sulla finestrella la prima parte della superfice magnetica.
Dopo che sara' apparso il solito:

PRESS PLAY AND RECORD ON TAPE#1

eseguiamo le operazioni abituali e dopo la registrazione avremo:

```
BREAK IN 60
READY.
```

Riavvolgiamo il nastro e digitiamo il GOTO 60 per far eseguire l' operazione di lettura.
Dopo aver premuto il tasto PLAY sulla cassetta sara' visualizzata un scritta:

```
FILE DATA ERROR IN 90
READY
```

Cio' vorra dire che i dati non sono stati letti correttamente. Perche'?.
Perche' non si puo' usare nessuna altra forma di puntegiatura e quindi di separazione nel trattamento di dati numerici all' infuori del ritorno carrello.

Si possono invece usare i due punti o il ritorno carrello per separare i campi in una stringa.

Proviamo a cambiare il programma come segue:

```
5 DATA UNO, DUE, TRE, QUATTRO, CINQUE, SEI,  SETTE,
OTTO, NOVE, DIECI
6 REM ----------------------------
10 OPEN1,1,1
20 FOR I =1 TO 10
30 READ M$ (I)
40 NEXT
45 C$=CHR$(44)
46 PRINT#1, M$(1);C$; M$(2);C$; M$(3);C$; M$(4);C$;
M$(5)
47 PRINT#1, M$(6);C$; M$(7);C$; M$(8);C$; M$(9);C$;
M$(10)
50 CLOSE 1
60 STOP
70 OPEN 1
80 FOR I =1 TO 10
90 INPUT#1, J$
100 PRINT J$
110 NEXT
120 CLOSE 1
130 STOP
```

Riavvolgiamo quindi il nastro e eseguiamo la prima parte di scrittura del programma.
Vedremo che i dati saranno scritti correttamente e sara' visualizzata la scritta:

```
BREAK IN 60
READY
```

A questo punto riavvolgiamo la cassetta ed eseguiamo il solito GOTO 70
Dopo aver premuto il tasto PLAY avremo la seguente visualizzazione:

UNO SEI

ed immediatamente sotto sullo schermo:

```
STRING TOO LONG ERROR IN 90
READY.
```

Cosa e' successo?

Il problema sta nel comando INPUT# della linea 90. Un comando INPUT # leggera' tutti campi della stringa fino al primo ritorno carrello.
Quindi le variabili da M$(1) a M$(5) sono in INPUT alla prima esecuzione del comando INPUT# alla riga 90. Tuttavia solo il valori di M$(1) e' stato letto da J$ perche' la virgola e' interpretata come un separatore di campo e non come un segnale di termine.
La seconda volta il comando INPUT£ alla linea 90 e' eseguito, da M$(6) fino a M$(10) sono in input, poiche' questi sono due campi situati fra due ritorni carrello. Ancora una volta solo M$(6) e' assegnato a J$ poiche' la virgola e' interpretata come campo.
La terza volta che il comando INPUT£ alla linea 90 viene eseguito non ci sono piu' dati da leggere e viene segnalato pertanto un errore.
Per risolvere questo problema e' necessario eseguire i comandi di INPUT£ con lo stesso numero di variabili presenti nel comando PRINT#.

Consideriamo il seguente programma:

```
5 DATA UNO, DUE, TRE, QUATTRO, CINQUE, SEI,  SETTE,
OTTO, NOVE, DIECI
6 REM --------------------------
10 OPEN1,1,1
20 FOR I =1 TO 10
30 READ M$(I)
40 NEXT
45 C$=CHR$(44)
46 PRINT#1, M$(1);C$; M$(2);C$; M$(3);C$; M$(4);C$;
M$(5)
47 PRINT#1, M$(6);C$; M$(7);C$; M$(8);C$; M$(9);C$;
M$(10)
50 CLOSE 1
60 STOP
70 OPEN 1
80 INPUT#1, N$(1), N$(2), N$(3), N$(4), N$(5)
90 INPUT#1, N$(6), N$(7), N$(8), N$(9), N$(10)
100 FOR I = 1 TO 10
105 PRINT N$(I);" ";
110 NEXT
120 CLOSE 1
130 STOP
```

Se non commetteremo nessun errore nel posizionamento del nastro avremo, stavolta correttamente:

UNO DUE TRE QUATTRO ecc.

sul video seguiti da:

```
BREAK IN 130
READY.
```

E' probabile che siamo stati noiosi, ma vi assicuriamo che la manipolazione dei dati su cassetta o su disco e' veramente la parte piu' importante della programmazione.
Per questo vi consigliamo di continuare con gli esperimenti fra cui vi consigliamo:

1)Puo' un singolo comando di INPUT# leggre un gruppo di variabili stringa separate da un ritorno carrello?

Per provare basta cambiare nell' ultimo programma la linea 45 in modo tale che a C$ sia assegnato il valor di CHR$(13).

2)Cosa accade se si mischiano numeri e stringhe in un singolo file di dati?

3)Creare 10 variabili stringa M$(I) come nell' ultimo programma, ma aggiungere 10 variabili numeriche M(I) aggiungendo per esempio:

```
35 M(I)=I+100
```

LETTURA DI DATI DA CASSETTA

Ci sono tre passi di programma necessari per leggere un file dati da cassetta:

APRIRE IL FILE con un OPEN

LEGGERE IL FILE con INPUT #

CHIUDERE IL FILE con un CLOSE

Un file di dati deve essere aperto per la lettura utilizzando lo stesso nome (NAME FILE) che era stato adoperato per scriverlo.
Mentre puo' essere assegnato un diverso numero di file logico, l' indirizzo secondario deve essere posto a 0 appunto per l' opzione di lettura.

Ricordiamo come regola generale:

SCRITTURA:

```
OPEN1,1,2,"DATA"
```

LETTURA:

```
OPEN1,1,0,"DATA"
```

Sono disponibili due comandi di lettura da cassetta:

```
INPUT #
```

e

```
GET #
```

Per leggere un campo (FIELD) numerico o una stringa useremo il comando INPUT#.
Il comando GET#leggera' invece un carattere per volta.
Ricordiamo poi di eseguire un CLOSE dopo che il file e' stato letto.
E naturalmente chiudiamo lo stesso file logico che e' stato aperto. per cui nell' esempio di prima:

```
CLOSE 1
```

Un buon sistema di chiudere un file e' quello di controllare l' esistenza di un carattere EOF (end-of-file) attraverso il registro di STATUS.
Quando un file viene scritto, un carattere di EOF viene posto alla fine del file.
Quando viene letto un carattere di EOF, il registro di status assume il valore di 64 ed il file puo' essere chiuso.
Si puo' controllare con questo semplice comando inserito alla fine della lettura:

```
IF ST=64 THEN CLOSE 1
```

Cioe' quando lo status e' 64 il file viene chiuso.

Precedentemente abbiamo scritto un programma per scrivere una serie di numeri da 1 a 10 in un file dati su cassetta chiamato.
Ora scriveremo un programma per leggere i dieci numeri dal file di dati NUMBERS e visualizzarli sullo schermo.

```
10 PRINT"** LETTURA DI DATI NUMERICI"
15 PRINT
20 PRINT"INSERIRE LA CASSETTA E PREMERE IL RETURN
QUANDO SIETE PRONTI":
25 PRINT
30 GET A$:IFA$=""THEN 30
40 PRINT"APERTURA":OPEN1,1,0 "NUMERI"
45 PRINT
50 FOR I=1 TO10
60 INPUT #1,N
70 PRINT N
80 NEXT I
90 PRINT"CHIUSURA": CLOSE 1
100 END
```

Il comando INPUT#legge un campo per volta.
Le prime 3 linee di del programma dicono all' utente come deve operare.
I comandi sono identici a quelli del programma di scrittura.
Alla linea 30 c'e' un loop di attesa che da all' operatore il tempo di montare il nastro sulla cassetta.
Dopo il montaggio fisico del nastro, premere RETURN e il programma passa alla linea successiva.
Prima che ogni dato sia letto, il file deve essere aperto.
I comandi alla linea 40 apriranno il FILE#1,sulla periferica 1, con l' indirizzo secondario 0 per l'

operazione di lettura sul file con il nome NUMBERS.

Nelle linee da 50 a 80, il cuore del programma, con il ciclo FOR-NEXT vengono letti i primi 10 dati dal nastro e visualizzati sullo schermo.

Il comando INPUT#1 della linea 60 legge un numero per l' esecuzione.

Dopo che i dati sono stati letti il file viene chiuso alla linea 90, mentre alla linea 100 si trova un END che ricordiamo pero' e' opzionale.

Il comando INPUT # puo' anhe leggere campi che contengano variabili stringa.
Come abbiamo visto in precedenza con il programma di numeri alfabetizzati, invece dei numeri nella loro rappresentazione decimale li abbiamo scritto in forma letterale.
Per leggere le stringhe, in questo caso, e' sufficiente una piccola modifica al programma di lettura precedente.
Come possiamo vedere dal listato che segue sono necessari solo dei cambiamenti alla linea 40 per fargli rileggere un file diverso da quello di prima ed alla linea 60 per far rileggere una stringa anziche' un numero.
Cambieremo pertanto:

```
60 INPUT#1,N
70 PRINTN
```

con:

```
60 INPUT#1,N$
70 PRINTN$
```

CAPITOLO SESTO

PROGRAMMAZIONE DI FILE DI DATI

Per accedere ai dati su cassetta sono necessari tre blocchi di programma:

-1 Aprire il File (OPEN)

-2 Eseguire un comando di lettura o scrittura (INPUT o PRINT)

-3 Chiudere il file (CLOSE)

vediamo i singoli passi uno per uno.

APERTURA O OPEN

Si deve aprire un comando OPEN per aprire un file di dati qualsiasi operazione si desideri eseguire. Il formato di questo comando e':

OPEN N,D,S,Nome del File

Cioe' apri il file logico N, seleziona sulla periferica D il file di dati scelto con "Nome del file" e predisponi per eseguire l' operazione specificata con l' indirizzo secondario S.

Si puo' usare il comando OPEN con una qualsiasi

combinazione di questi parametri.
N e' il solo parametro che deve essere presente, mentre se D e' assente viene ritenuto uguale a 1. Se e' assente S viene assunto per 0, mentre se manca il nome del file verra' selezionato il primo file che si incontra sulla cassetta.
Quando viene eseguito un comando OPEN su cassetta per la lettura di dati, verra' visualizzato il seguente messaggio:

PRESS PLAY ON TAPE

e dopo che il tasto della cassetta sara' premuto sara' visualizzato:

OK

Il computer incomincera' allora a leggere il nastro. Nell' ipotesi che il comando sia dato in modo immediato e che il file cercato non sia il primo sul quale si e' posizionata la testina di lettura, avremo la visualizzazione dei seguenti messaggi:

SEARCHING FOR (Nome del file)

FOUND Test

FOUND Ross

FOUND Chess

FOUND

dove i primi tre stanno ad indicare i nomi di files incontrati, mentre il quarto indica che si e' incontrato un file senza nome.

FOUND (Nome del file scelto)

Ready.

L' ultimo messaggio prima del Ready sara' visualizzato quando il file scelto sara' trovato.

NOTA

In modo programma invece questo blocco di messaggi non sara' visualizzato.

Quando il comando OPEN viene eseguito per una operazione di scrittura il computer visualizzera' il seguente messaggio:

PRESS PLAY & RECORD ON TAPE

Anche questo messaggio sara' seguito da un OK quando vengano premuti i tasti.

Il computer scrivera' il TAPE HEADER, poi si fermera' in attesa di dati.

Vediamo alcuni esempi di utilizzo del comando OPEN commentandoli:

OPEN 1

Vine aperto il file logico 1. Nessuna periferica e' specificata per cui come periferica sara' assunta la cassetta n. 1.
Non essendo inoltre specificato nessun indirizzo seconadrio il computer si prepara ad una operazione di indirizzo seconadrio 0 cioe' di lettura.
E dato che nemmeno il nome del file e' specificato, verra' letto il primo file che incontra su cassetta.

OPEN 1,1

Come per il precedente, ma in questo caso e' specificata la periferica.

OPEN 1,1,0

Come il precedente salvo che in questo caso sono dichiarati tre parametri.

OPEN 1,1,0,"data"

Come sopra ma con una apertura per leggere il file di nome "data" sulla cassetta.

OPEN 3,1,2

Apre il file logico 3 per la cassetta 1.Scrive un nuovo file e un carattere End of Tape alla fine del file. Il file che si registra non ha nessun nome.

OPEN 3,1,2,"Paolo"

Come sopra ma questa volta con un file di nome PAOLO.

CHIUSURA DI UN FILE

I comandi di apertura e di chiusura di un file sono fra se strettamente correlati.
Ricordiamoci che il comando CLOSE deve essere l' ultimo comando che si da nella sequenza logica della programmazione e che mentre se non si esegue l' apertura di un file e' immediatamente tangibile che non si puo' accedere ad esso, il fatto di usare il comando CLOSE e' lasciato alla accortezza del programmatore, perche' il sistema non dara' alcuna segnalazione.
Il formato del comando e':

CLOSE N

dove N e' l' unico parametro da specificare e deve corrispondere al numero di file logico precedentemente dichiarato nel comando OPEN.

Ricordiamo che quando e' stata eseguita un' operazionedi chiusura su un file dopo la lettura non sono piu' consentiti accessi per la lettura sullo stesso file, per cui sar' necessario riaprirlo.

NOTA

Non e' indispensabile eseguire la chiusura di un

file dopo la lettura, tuttavia non sara' cattiva pratica di programmazione prendere costante confidenza con questa operazione che in altri casi e' invece indispensabile.

Come si puo' vedere piu' agevolmente nella parte di approfondimento e' invece ASSOLUTAMENTE necessario eseguire la chiusura di un file dopo una operazione di scrittura.
Si vedra' infatti nell' approfondimento che i dati non vengono mai trasferiti direttamente dalla memoria centrale del computer al nastro, ma passano attraverso un BUFFER, cioe' attraverso una zona polmone che provvede a comunicare i dati stessi a gruppi di 192 Bytes.
Quando questo BUFFER e' statoriempito allora i dati passano al nastro. Ma solo allora, oppure quando si esegue un CLOSE per chiudere appunto il file al quale stiamo riferendoci.
Per questo motivo accade quasi sempre che se al termine di un' operazione di scrittura non si esegue la chiusura un po' di bytes che quasi sicuramente sono sul nastro vengono persi.

Inoltre quando si esegue una chiusura di un file nastro dopo un' operazione di scrittura, verra' anche inviato un carattere di fine file o END OF FILE (EOF) che verra' quindi scritto sul nastro stesso.
Al sistema infatti necessita questo carattere di separazione fra un file e il successivo. Senza di questo infatti il computer potrebbe, successivamente in fase di lettura, continuare a leggere i dati del file successivo sul quale magari siamo andati a scrivere sopra.

NOTA

E' importante notare infatti che mentre in fase di input dati siamo certi della quantita' di dati da scrivere, altrettanto non avviene in fase di rilettura.

Quando si chiude un file preventivamente con un INDIRIZZO SECONDARIO 2, su cassetta verra' scritto un fine nastro END OF TAPE (EOT) alla fine del file stesso.
In questo caso il computer non andra' ad eseguire una ricerca di altri files.

COME ACCEDERE AI DATA FILES

Dopo aver aperto un file con un comando OPEN si puo' accedere a questo file sia per leggerci che per scriverci fino a quando il file stesso non sia chiuso con un comando CLOSE.
Ricordiamoci ancora una volta che sia che si scriva sia che si legga, queste operazioni devono essere fatte in modo SEQUENZIALE.
Cioe' il primo record scritto o letto sara' sempre il primo record del FILE, per cui se si vuole leggere il decimo record di un file sara' necessario prima leggere i primi nove.
Nessun tasto della cassetta deve essere premuto prima che l' apposito messaggio non sia apparso sullo schermo.
E' bene ricordare ancora una volta che la posizione esatta della cassetta e' sotto la responsabilita' dell' operatore. Infatti l' unita' a cassette iniziera' a scrivere immediatamente i dati non

appena saranno premuti gli appositi tasti senza controllare ne che sul nastro sia scritto qulacosa ne che il nastro sia correttamente posizionato sull' inizio della superfice magnetica.

Per scrivere data su cassetta e' necessario usare il comando PRINT#che ha il seguente formato:

PRINT#f,data

dove f e' il numero di file logico che e' stato assegnato con un comando OPEN e che sara' riutilizzato con il comando CLOSE. Puo' avere un valore fra 1 e 255.

Data saranno invece i dati da caricare sul file.

Il comando PRINT£ non puo' essere scritto nella forma abbreviata ?#, ma deve essere digitato per intero.
Il comando PRINT# trasferisce i dati dalla memoria centrale del computer al buffer di cassetta. Quando il buffer e' stato completamente caricato nei suoi 191 Bytes di capacita' i dati sono scaricati su nastro appunto in BLOCCHI di 191 caratteri per volta.

Con il comando appena visto e' possibile scrivere sia dati numerici che alfanumerici o stringhe.

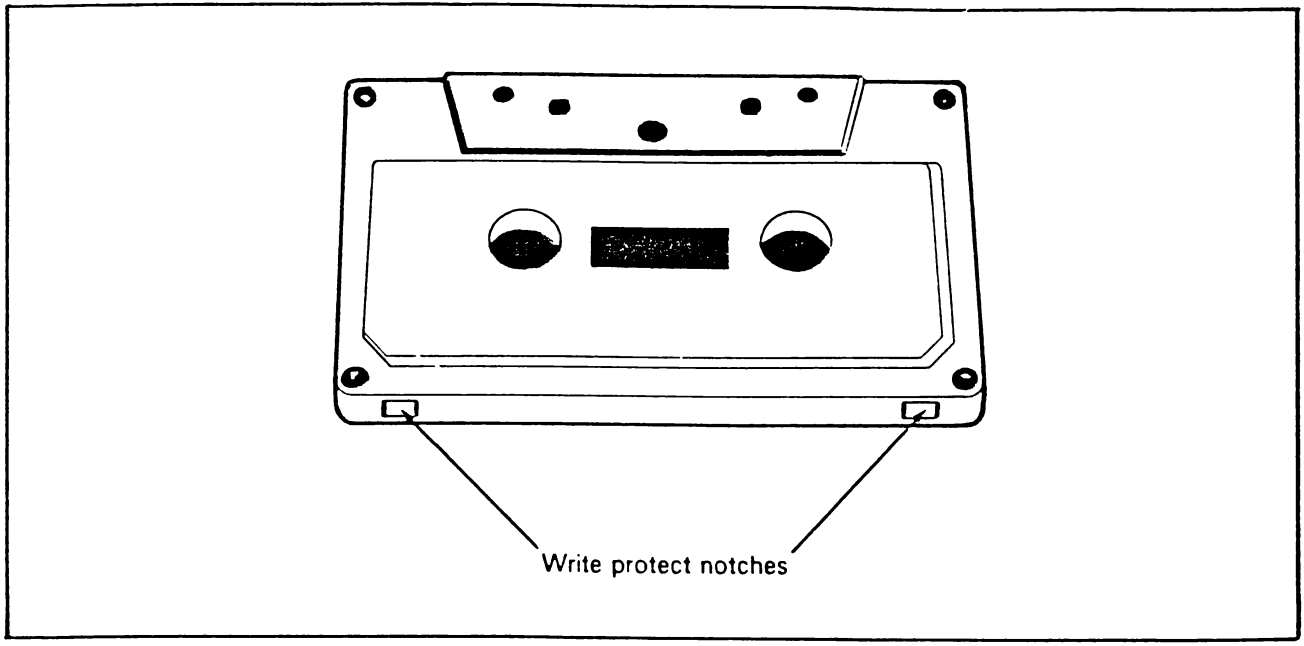


Fig. 5 - Linguette da togliere per proteggere da scrittura i FILES su cassetta.

CAPITOLO SETTIMO

APPROFONDIMENTO HARD/SFTWARE

Tutti i computers COMMODORE, comprese le serie professionali e con la sola esclusione attuale del CBM64 SX (quello portatile) hanno la possibilita' di connettere una singola unita' a cassette esterna che puo' essere utilizzata per immagazzinare programmi e dati.
L' unita' e' connessa al computer per mezzo delle linee:

WRITE

READ

MOTOR

SENSE

due linee di alimentazione composte da una terra (GND) e un +5 Volts.

La cassetta e' controllata nelle sue linee di I/O per mezzo di due integrati di tipo VIA.
Le linee di alimentazione del motore sono connesse agli integrati di interfaccia attraverso 3 piloti transistor utilizzati per elevare la tensione ed il voltaggio e che consentono quindi che il motore sia alimentato e guidato direttamente dal computer.
L' uscita al motore e' un +9 Volts non regolata a una potenza di 500Ma.

Al termine di questo manuale e' riportata una tabella che indica i comandi e le locazioni di memoria da manipolare per consentire il controllo del motore tramite software.

Il controllo della linea di Input, la linea PA6 sul VIA 1, e' connessa ad un' interrutore sull' unita' a cassette che controlla quando sia il tasto PLAY, REWIND o FAST FORWARD sono stati premuti.
Purtroppo pero' la connessione dell' interruttore viene controllata solo durante le fasi di lettura e scrittura per vedere se il tasto PLAY e' premuto .
Per questo motivo se il tasto di REWIND e di FAST FORWORD vengono premuti accidentalmente invece del tasto PLAY, il sistema non e' in grado di comprendere la differenza ed agisce come se il tasto PLAY fosse premuto.
Per lo stesso motivo, durante l' esecuzione di una routine di registrazione il tasto RECORD dovrebbe essere premuto prima del tasto PLAY perche' la fase di registrazione inizia nello stesso momento in cui il tasto PLAY viene premuto senza controllare che sia stato premuto anche il tasto di RECORD.

La linea di lettura "READ LINE" e' connessa al CIA numero 1 tramite l' integrato VIA 2, e la linea di scrittura "WRITE LINE" alla linea PB3 del VIA 2.
Durante un' operazione di lettura il Sistema Operativo utilizza il fissaggio dell' INTERRUPT FLAG del CIA numero 1 per controllare il passaggio di dati sulla linea di lettura cassetta.
Il funzionamento delle linee di lettura scrittura e' controllato interamente dal S.O. e la sola operazione Hardware richiesta e' l' amplificazione del segnale.

OPERAZIONI SU CASSETTA

Nel normale utilizzo all' unita' a cassetta e' assegnato un numero di periferica di Input/Output. La cassetta e' la periferica numero 1, il numero della periferica utilizzata in quel momento e' immagazzinata nella locazione decimale 186 per il VIC 20 e per il CBM64.

NOTA

Fino a quando non differiscono non staremo, d'ora in poi, a specificare VIC 20 o CBM64, anche perche' molte delle locazioni e indirizzi sono uguali.
D' altra parte e' logico che ci si riferisca per la cassetta a questi computer, perche' sulle serie professionali e' molto piu' facile trovare collegata l' unita' a dischi.
Abbiamo comunque riportato una tavola di comparazione per tutti i computer COMMODORE piu' diffusi che potra' essere utile anche per passare programmi da una unita' all' altra.

Il numero di periferica, il numero di file logico e l' indirizzo secondario sono utilizzati quando si salvano o si ricercano file di dati da cassetta.
Il numero di file logico puo' essere uno qualsiasi purche' compreso nell' intervallo da 1 a 255 ed e' utilizzato per consentire la manipolazione di Files multipli sulla stessa periferica.
Tutto questo e' di poco utilizzo per l' unita' a cassette, perche' non si possono manipolare contemporaneamente piu' files.

Lavorando normalmente e' usuale avere lo stesso numero sia per il file logico che per la periferica.Il numero di file logico del file sul quale si sta lavorando e' immagazzinato nella locazione decimale 184.
L' indirizzo secondario e' importante perche' determina il modo di operare della cassetta. L' indirizzo secondario e' immagazzinato nella locazione di memoria 185 il cui valore normale dovrebbe essere 0.
Se l' indirizzo secondario e' 0 allora il nastro e' aperto per un' operazione di lettura. Se e' messo a 1 allora il nastro e' aperto per un' operazione di scrittura mentre se il contenuto della locazione decimale 185 e' 2 sul nastro verra' eseguita un' operazione di scrittura con l' immissione di un carattere di END OF TAPE quando il file viene chiuso.

Il Sistema Operativo del computer e' configurato in modo tale da consentire che due diversi tipi di file possano essere immagazinati su cassetta:

Files programmi

Files data

L' utilizzo di questi due nomi e' pero' una convenzione poiche' un programma puo' essere immagazzinato come file di dati mentre dei dati possono essere immagazzinati come file programma..
La differenza fra questi due tipi di files non e' nella loro applicazione ma nel sistema in cui i contenuti della memoria del computer sono registrati.

Infatti invece di FILES PROGRAMMI e FILES DATA si dovrebbe parlare piuttosto di FILES BINARI e FILES ASCII.
Un file binario e' normalmente utilizzato per immagazzinare programmi perche' questo tipo di files viene creato e gestito dal sistema operativo che immagazzina i contenuti di memoria presenti e compresi fra una locazione o un' indirizzo di inizio ed un' indirizzo di fine.

FILES BINARI

E' chiamato file binario perche' immagazzina su nastro il valore di ogni locazione di memoria compresa nell' intervallo richiesto o desiderato.
I comandi Basic sono immagazzinati in memoria utilizzando i TOKENS.

NOTA

Ricordiamo che con la procedura di TOKENIZZAZIONE, cioe' con l' uso dei TOKENS, si riduce un comando Basic ad occupare un solo Byte in luogo dei quattro o cinque che si vedono sullo schermo.
Per un approfondimento vedi la GUIDA AL CBM 64 o il MANUALE D' USO DEL VIC 20, entrambi ed. EVM.

L' uso dei TOKENS consente che i comandi non siano quindi immagazzinati nello stesso sistema in cui sono listati cioe' visualizzati oppure immessi tramite la tastiera.
I comandi sono invece immagazzinati in memoria in una maniera che potremmo definire parzialmente

codificata.

Essendo parzialmente codificata un file binario e' un sistema veloce e molto efficente di immagazzinare programmi.
Inoltre i files binari sono ESSENZIALI, questo nel senso che si devono obbligatoriamente usare, quando si caricano o si salvano programmi in codice macchina.
L' indirizzo di inizio dal quale un file di tipo binario deve essere salvato e' immagazzinato sia per il VIC 20 che per il CBM 64 nelle locazioni di memoria 172 e 173.
Queste locazioni di memoria sono utilizzate dalla routine di SAVE che conterra' l' indirizzo di inizio della BASIC TEXT AREA diversa pero' da macchina a macchina.
Ad esempio nel VIC 20 questi indirizzi conterranno i valori 1 e 16 in quanto l' area di Basic incomincia dall' indirizzo decimale 4097.
I contenuti delle locazioni di memoria 172 e 173 possono essere cambiati per consentire che la routine di SAVE punti ad una qualsiasi scelta locazione di memoria.
L' indirizzo di fine dell' area di memoria che deve essere salvata su nastro e' immagazzinata nelle locazioni 174 e 175.
Normalmente quando si salva un programma Basic queste due locazioni contengono l' indirizzo in cui nel programma si trovano due Bytes a ZERO che corrispondono al punto di fine o di termine del programma Basic stesso.
Anche l' indirizzo di fine puo' essere cambiato per far si che la routine di SAVE punti ad un' altro qualsiasi desiderato indirizzo.
Per cambiare entrambi questi indirizzi non possiamo utilizzare la normale routine di SAVE poiche' questa automaticamente inizializza queste

locazioni, cioe' punta o indirizza automaticamente a queste locazioni.
Dovremo quindi scrivere una piccola routine in codice macchina per modificare i valori presenti in detti indirizzi.
In mancanza di modifiche il comando SAVE scrivera' un file binario da quegli indirizzi ed un comando LOAD li leggera' come file binario.

FILES ASCII

I files ASCII sono normalmente utilizzati per immagazzinare dati, (ma possono essere utilizzati per immagazzinare anche programmi).
Il formato dei dati e' lo stesso che viene visualizzato sullo schermo o immesso tramite tastiera.
I files ASCII sono creati o letti esclusivamente tramite istruzioni contenute entro un programma Basic.
Un file binario e' creato o letto molto spesso per mezzo di istruzioni date in modo diretto, sebbene i comandi LOAD e SAVE possano essere utilizzati entro un programma.
Un file ASCII deve per prima cosa essere aperto con un comando OPEN che specifichi il file logico, il numero di periferica, l' indirizzo secondario ed il nome del file.
Il Sistema Operativo interpreta questi parametri, cioe' quelli contenuti nel comando OPEN, e consente all' utente di leggere o scrivere il file su una data periferica.
Mentre un file binario e' quindi caricato come una serie di successive locazioni di memoria, un file ASCII e' caricato come una serie di variabili

stringa.
Per questo motivo il file ASCII dovrebbe essere manipolato immagazzinando pochi Bytes per volta e fermando e facendo quindi ripartire il nastro.
Il Sistema Operativo del computer supera questo problema perche' dispone di un BUFFER di nastro di 192 Bytes entro il quale tutti dati che devono essere scritti su nastro o da questi letti vengono caricati.
Solo quando questo Buffer e' pieno il motore del nastro si avvia.

I dati sono quindi immagazzinati su nastro in blocchi di 192 Bytes e cosi' il motore si attiva e si spegne tra i blocchi in un intervallo di due secondi cosa che consente al motore stesso un' accurato movimento di accelerazione e di decelerazione.
L' inizio del buffer contenente i 192 caratteri e' posto all' indirizzo 828. Con questo intendiamo naturalmente dire che l' area del Buffer inizia da qui.
Il puntatore all' inizio del Buffer e' dato dal contenuto dei puntatori 178 e 179.
Il numero dei caratteri o il numero di Bytes presenti nel buffer e' immagazzinato nell' indirizzo di memoria 166.
Questa locazione di memoria puo' essere utilizzata dal programmatore per controllare l' ammontare di spazio lasciato in un file di dati.

Sia che il file da immagazzinare sia di tipo binario o ASCII il metodo di registrazione e' lo stesso.
Si tratta infatti di un metodo di codifica messo a punto dalla COMMODORE per consentire la massima sicurezza di registrazione e quindi poi di rilettura.

Ogni Byte di dati o di programma e' codificato dal Sistema Operativo del computer utilizzando impulsi di 3 distinte audiofrequenze.
Questi sono impulsi di lunga durata con una frequenza di impulsi di 1488 Hz, impulsi di media durata con una frequenza di 1953 Hz ed impulsi di corta durata con una frequenza di 2480 Hz.
Tutti questi impulsi sono del tipo ad ONDA QUADRA con un' intervallo costante (cioe' di 1:1).
Il Sistema Operativo richiede circa 9 millisecondi per registrare un Byte di dati che consiste di 8 bits, una parola di segnale, ed un bit di parita'.
I bits di dati possono essere composti sia da 1 che da 0 e sono codificati da una sequenza di impulsi medi e corti.
Un UNO e' composto da un ciclo di impulsi di media lunghezza seguito da un ciclo d' impulsi di corta lunghezza, mentre uno ZERO e' composto da un ciclo d' impulso di corta lunghezza seguito da un ciclo d' impulso di lunghezza media.
Ogni bit consiste quindi di due cicli d' impulsi ad onda quadra uno corto ed uno medio per una durata totale di 864 mcs.

Il BIT di parita' o PARITY BIT e' richiesto per il controllo d' errore ed e' codificato come 8 bits di dati utilizzando un impulso lungo ed uno corto.
Il suo stato e' determinato dal contenuto degli 8 data bits. Il WORD MARKER, cioe' il segnalatore di parola, separa ogni Byte di dati e segnala anche al Sistema Operativo l' inizio di ogni Byte.

Il WORD MARKER e' codificato come un ciclo d' impulsi lunghi seguito da un ciclo d' impulsi medi.

NOTA

Poiche' un BYTE di dati e' registrato in appena 8.96 millisecondi, un blocco di 192 Bytes di dati di un file ASCII dovrebbe essere registrato in poco piu' di 1.7 secondi.
Tuttavia ad un controllo di temporizzazione risulta che un Byte di dati richiede circa 5.7 secondi.
Ci sono due cause per questa discrepanza nella temporizzazione.

Prima di tutto per ridurre la possibilita' di errori audio i dati sono registrati due volte.
Secondo fra due record di 192 bytes viene immesso un INTER RECORD GAP della lunghezza di circa 2 secondi.

CONTROLLO DI ERRORE

L' uso massiccio delle tecniche di controllo errore (ERROR CHECKING) e' una delle ragioni per le quali il sistema a nastri implementato sui computer COMMODORE e' di gran lunga migliore di quello disponibile in molti altri personal computer.

Ci sono due livelli di controllo d' errore.
Il primo divide i dati in blocchi di 8 bytes e successivamente inserisce un nono Byte chiamato CHECKSUM DIGIT.
Il CHECKSUM e' ottenuto aggiungendo all' ottavo Byte un nuovo Byte.
Il CHECKSUM e' il Byte meno significativi di risultato.

Il secondo livello di controllo errore consiste nel registrare ogni blocco di dati due volte.

Questo consente di rilevare gli errori controllando il digit che deve essere corretto durante la seconda lettura del blocco di 192 Bytes di dati.
Registrando i dati due volte una verifica e' possibile confrontando il contenuto dei due blocchi.
Cio' consentira' di rilevare quei piccoli errori che non sono stati controllati dal CHECKSUM.

L' utilizzo di sequenze di impulsi invece di due differenti frequenze come normalmente avviene nelle registrazioni, ha un grande vantaggio perche' consente al Sistema Operativo di compensare facilmente la variazione o le variazioni nella velocita' di registrazione.
I computers COMMODORE utilizzano infatti il sistema SOFTWARE per controllare la corretta registrazione dei dati.
Prima di iniziare la registrazione dei dati o dei programmi su nastro viene scritta una testata di 10 secondi.
Questa testata ha due funzioni. Prima di tutto consente che il motore del nastro possa incominciare a scorrere ad una velocita' corretta.
Secondariamente la sequenza di impulsi corti scritta nella testata e' utilizzata per sincronizzare la routine di lettura temporizzata in modo tale che il nastro sia correttamente temporizzato.
In questo modo il Sistema Operativo puo' quindi produrre un fattore di correzione che consente un largo raggio di variazioni nella velocita' del nastro senza che ne abbia a risentire la fase di lettura.

La temporizzazione di sistema utilizzata per consentire sia le operazioni di scrittura che di lettura e' molto accurata perche' e' basata su un

sistema di CLOCK e sui TIMER 1 e 2 (cioe' sui TEMPORIZZATORI 1 e 2) del VIA n.2.
Gli INTERRECORD GAPS sono utilizzati solamente con i Files ASCII e la loro funzione e' di permettere che il motore dell' unita' a nastri possa decelerare prima che sia fermato ed accellerare alla corretta velocita' prima che riparta.
In questo modo viene consentita una migliore lettura o scrittura di dati.
Ogni INTERRECORD GAP ha una durata di circa 2 secondi ed e' registrato come una sequenza di impulsi corti allo stesso modo dei 10 secondi della testata.
E' presente anche un GAP , cioe' un' interruzione fra i blocchi.

Quando il primo blocco di 192 Bytes e' stato registrato, questi e' seguito da 50 cicli di impulsi corti e poi inizia la registrazione del secondo blocco di 192 Bytes.
Sia che stiamo registrando un file ASCII che un File di tipo binario, il primo record scritto su nastro dopo la testata di 10 secondi e' un file di 192 caratteri chiamato FILE HEADER BLOCK.
Il FILE HEADER contiene il nome del file, l' indirizzo di partenza e l' indirizzo di fine.
In un file ASCII questi indirizzi sono l' inizio e la fine del BUFFER di nastro, mentre in un file di tipo binario questi indirizzi contengono l' indicazione dell' area di memoria nel quale il programma deve essere immagazzinato.
Il nome del file puo' essere lungo al massimo 128 Bytes. La lunghezza del nome del file e' immagazzinata nell' indirizzo di memoria decimale 183 e quando viene eseguita un' operazione di lettura, questa lunghezza verra' confrontata con il nome del file del comando LOAD.
Se il nome del file trovato e' identico allora il

Sistema Operativo consentira' la lettura del file.
Se e' invece diverso proseguira' nella ricerca.

Durante un' operazione di lettura o scrittura il nome del file e' immagazzinato in un blocco di memoria il cui indirizzo di partenza e' dato dal contenuto delle locazioni di indirizzo 187 e 188.
Una volta compiuta questa operazione le due locazioni di memoria precedenti sono azzerate per puntare ad una locazione indicata dal Sistema Operativo.

L' indirizzo di partenza e' normalmente fissato all' inizio della memoria utente che pero' varia da macchina a macchina.
Tuttavia questa puo' essere cambiato per puntare ad un' altra qualsiasi locazione.
Questo sistema e' impiegato quando si registra un programma in codice macchina utilizzando il monitor e quando si eseguono delle protezioni sui programmi.

Nel resto di questa sezione pubblichiamo un' elenco di programmi che il lettore potra' digitare oppure richiedere su nastro, ricordandosi di specificare il tipo di computer, con le schede al termine dell' opera.

CAPITOLO OTTAVO

INTRODUZIONE

Con l' acquisto di un drive per floppy disk la potenzialita' operativa del vostro sistema e' fortemente aumentata.
Malgrado all' inizio avessimo deciso di scrivere solo la parte relativa all' uso dei drive 1540/1541 ci siamo resi conto che non poteva essere trascurata la possibilita' che utenti VIC-20 e CBM64 disponessero di interfacce da potersi collegare anche agli altri floppy COMMODORE.
Per questo motivo e anche per il fatto che comunque era necessario un manuale sulle periferiche importanti CBM la trattazione non sara' limitata ai soli utenti descritti, ma si rivolge a tutti gli utenti CBM.

Molti concetti vengono anche ripetuti piu' volte, ma altri sono dati per scontati, in particolare quelli trattati abbondantemente nella precedente sezione del manuale quella relativa alla gestione dell' unita' a cassette.
Anche per questa sezione avvertiamo che molte tavole sono riportate al termine del volume.

FILES DISCO

I dischetti possono immagazzinare sia files programmi che files di dati. A differenza che sulle cassette sui dischi si possono immagazzinare i files di dati in tre diversi modi, sempre sotto controllo di un programma:

FILES SEQUENZIALI

FILES RELATIVES

FILES RANDOM

I files relatives sono gestibili direttamente solo con le unita' che abbiano il DOS 4. Per gli utenti VIC e CBM64 abbiamo pero' messo a punto una routine che consente di utilizzarli comunque.

CONFRONTO FRA MANIPOLAZIONE DI FILES SU DISCO E SU CASSETTA.

La manipolazione di files su cassetta differisce in modo sostanziale da quella degli stessi files su disco per le seguenti ragioni:

1 - La velocita' di accesso ai dischetti e' molto piu' alta di quella delle cassette.

2 - Non esistono INIZI e FINE sulla superfice

magnetica dei dischetti. Una unita' a dischi accede con facilita' a qualsiasi punto del disco cosa che ovviamente non avviene per la cassetta.

3 - La manipolazione di files di dati su cassetta o su disco differisce perche' la formattazione dei dati ed i metodi di accesso sono sostanzialmente diversi.

Non facciamoci ingannare dalla velocita' meccanica di rotazione del disco o della cassetta che se non e' uguale non determina comunque una grande differenza.
La cassetta registra i dati in maniera sequenziale durante lo scorrimento del nastro e nello stesso modo li rilegge.
Al contrario il disco immagazzina dati su un gran numero di tracce concentriche.
La testina di lettura della cassetta e' ferma ed aspetta che il nastro si posizioni per poter eseguire operazioni di lettura o scrittura, al contrario la testina dell' unita' a dischi si sposta avanti ed indietro mentre il dischetto gira per trovare il giusto punto in cui operare.

Per usare l' unita' a dischi non e' indispensabile sapere come le informazioni sono immagazzinate sulla superfice magnetica del supporto, tuttavia le conoscenze di questi argomenti renderanno piu' efficiente la programmazione.
Per questo inizieremo la nostra discussione sui files disco descrivendo il modo in cui i dati sono immagazzinati sulla superfice magnetica del floppy.

COME IL DISCO IMMAGAZZINA I DATI.

Un dischetto immagazzina i dati su un numero di tracce circolari concentriche.
Queste tracce sono a loro volta divise in settori.
Differenti unita' scrivono un diverso numero di tracce sul disco.
Alcuni drives scrivono su ambedue le facce del dischetto come gli 8250, gli altri come il 2040,3040,8050 e 1540/1541 su una sola.
I drives dell' unita' a dischi non scrivono dati lungo l' intera lunghezza della traccia, che e' utilizzata invece per la memorizzazione dei segnali di riferimento.
Per far questo sarebbe necessario un complesso sistema di indirizzamento.
Infatti se si utilizzasse l' intera superfice della traccia dovremo presumere che dato che ci troviamo di fronte a tracce concentriche nessuna traccia avrebbe la lunghezza di un' altra e pertanto non potrebbe contenere la stessa quantita' di dati.
Per risolvere questo problema che inevitabilmente porterebbe ad un appesantimento eccessivo della gestione del Sistema Operativo su disco, le tracce sono state divise in settori.
Ogni settore contiene esattamente lo stesso ammontare di informazioni.
Nel caso dei computer CBM ogni settore contiene un blocco di dati pari a 256 Bytes.

DIRECTORY DEL DISCO E BAM

Due tracce di ogni disco sono usate per l' indice del dischetto stesso.

La prima o DIRECTORY TRACK contiene il nome che e' stato assegnato al dischetto, seguito dai nomi di tutti i files e dall' indirizzo di inizio del loro settore.

La seconda traccia contiene la BAM o BLOCK AVAILABILITY MAP che identifica i blocchi dove sono o non sono allocati i files.

Come abbiamo detto la BAM e' in pratica la rappresentazione della memoria disponibile su disco e della distribuzione degli spazi.
Quando il sistema deve immagazzinare dati su disco, la BAM viene automaticamente collegata con il DOS per determinare quale spazio e' disponibile e quindi quanti blocchi possono essere salvati.
Se e' disponibile un spazio sufficiente per immagazzinare un dato file, allora l' operazione sara' coronata da successo e la BAM aggiornata per tener conto dello spazio utilizzato.

Se invece il DOS riterra' che lo spazio non e' sufficiente allora verra' riportato un errore e l' operazione stessa di salvataggio non avra' effetto e verra' solo registrato il nome del programma nella Directory con un asterisco.

I files immagazzinati su cassetta non necessitano di una Directory o indice all' inizio del nastro.
Se dieci files sono immagazzinati su cassetta e il programma specifica un accesso particolare al sesto file, il fatto di avere una directory all' inizio del nastro non aiuta certo l' unita' a cassette a trovarlo meglio!
Poiche' un file su cassetta puo' avere una lunghezza qualsiasi, non esiste mezzo di associare il numero del file ad una determinata posizione del

nastro.
Questo anche perche' non e' possibile avere dei comandi che facciano andare il nastro prima forte e poi piano senza usare i tasti dell' unita' a cassette.
Ne' per quanto utile, c' e' da fidarsi eccessivamente del contametri della DATASETTE tutto meno che preciso e che comunque non potrebbe tener conto dell' allungamento o dell' accorciamento del nastro dovuto alle variazioni termiche o al punto di scrittura.
Come abbiamo detto in precedenza l' unico sistema per evitare di leggere i files che precedono quello oggetto della nostra ricerca e' di posizionarsi con il tasto FORWARD un po' prima dell' inizio (probabile) del file.
In caso contrario e' necessario far rileggere tutti i files precedenti.

Al contrario con una unita' a dischi si puo' andare direttamente all' inizio di un qualsiasi file sulla superfice del dischetto stesso, perche' ogni settore del disco e' egualmente accessibile.
Per rendere possibile questo e' quindi necessario che ogni dischetto abbia un indice che contenga il nome di tutti i files ivi registrati e l'indirizzo del settore di partenza.
Questo indice, simile quindi all' indice di un libro, e' appunto la DIRECTORY che fornisce anche il tipo di file che e' stato memorizzato e l' occupazione in blocchi di questo.

Quando un file di dati su disco e' aperto, l' unita' prima di tutto leggera' la Directory dalla quale ottiene l' indirizzo del settore in cui ha inizio il file. Poi la testina di lettura/scrittura potra' posizionarsi direttamente all' inizio del file aperto.

La Directory contiene le seguenti informazioni:

-Nome del disco

-Identificatore (ID) del disco

-Numero di versione del DOS

-Nome dei Files

-Tipo dei Files

-Numero dei blocchi usati

-Puntatore al primo blocco dei Files

-Numero dei blocchi disponibili

Vediamo ora come vengono trattati i records di un file su disco.

FILES RELATIVES

Tutti i records presenti in un file relative hanno la stessa lunghezza.
Per questo e' facile calcolare l' indirizzo di settore per un singolo record di un file relative.
Supponiamo di avere un file relative in cui i singoli records occupino mezzo settore. Cioe' che ne entri due per settore.
Allora il decimo record di questo file relative sara' semplicemente rintracciabile sul quinto

settore dall' inizio del file.

!!!ATTENZIONE!!!

I files relatives sono disponibili solo con le CBM BASIC VERSION 4.0 e oltre, usando il DOS 2.0 e oltre.
In questo manuale e' riportata una subrotine che consentira' anche agli utenti del VIC-20 e CBM64 di utilizzare i Files Relatives.

FILES SEQUENZIALI

I records di un file sequenziale possono avere differenti lunghezze.
Per questo non si puo' calcolare il settore sul quale deve essere rintracciato un particolare record di un file sequenziale, appunto perche' la lunghezza del singolo record e' sconosciuta.

La testina del dischetto puo' andare direttamente all' inizio di un file sequenziale, poiche' l' indirizzo del settore e' dato dalla Directory, ma una volta trovato questo inizio il file deve essere letto fino a quando non si trova il record desiderato.
La ricerca e' quindi sequenziale e quindi simile a quella su nastro.
Per trovare il decimo record di un file e' quindi necessario leggere i precedenti 9 records.

NOTA

Tutte le versioni dei dischi della Commodore sono abilitate alla gestione dei files sequenziali.

FILES SEQUENZIALI E RELATIVES

Se i records di un file dati sequenziale devono essere letti sequenzialmente cioe' uno dopo l' altro, gran parte dei vantaggi dell' accesso casuale tipico dei dischi viene perso. Ed allora perche' usarli?.
Prima di tutto perche' non su tutte le unita' a dischi della Commodore e' possibile la gestione dei FILES Relatives.
Secondo, e molto piu' importante e' che con i files sequenziali si riesce ad immagazzinare molte piu' informazioni che con i relatives. Si immagazzinano in forma piu' densa.
Si sfrutta cioe' meglio la capacita' di memorizzazione del disco.
Consideriamo il seguente esempio.
Si abbiano due nomi ed indirizzi come segue:

ALESSANDRO MARCELLOZZI
VIA MARTIRI DELLA LIBERAZIONE 7
20036 MILANO

e

MARIO ROSSI
VIA ROMA 1
31073 ROMA

Supponiamo che questi due nomi ed indirizzi facciano parte di un file MAILIG LIST. Ogni nome ed indirizzo diventera' quindi un record entro il file dati.
Gestendo il file con il metodo relative si dovra' assegnare lo stesso spazio per ogni nome ed indirizzo.
Per questo si dovra' tenere conto non della media di occupazione ma dell' indirizzo piu' lungo. Di conseguenza i nomi e gli indirizzi piu' corti lasceranno parte del disco inutilizzato.
Nell' esempio precedente, per immagazzinare il primo nome ed indirizzo avremo bisogno di 69 bytes (considerando gli spazi di separazione) mentre per il secondo solo di 34, e quindi circa la meta'.
Ma usando un file relative dovremo dimensionarlo a records di 69 almeno per cui il secondo nominativo lascera' meta' del suo spazio inutilizzato.

Al contrario un file sequenziale assegna ad ogni record solo lo spazio che effettivamente gli necessita.
Per cui lo sfruttamento e' massimo e su grandi quantita' di dati si fa indubbiamente sentire.

INDIRIZZAMENTO DEL DISCO

I settori assegnati su disco ad un file di dati non sono FISICAMENTE sequenziali sulla superfice del dischetto anche quando si utilizza un file di tipo SEQUENZIALE.
Per esempio, quando si aggiungono records ad un file esistente, questi devono essere registrati

senza andare a cadere sul file successivo.
Per questo il file dovra' essere proseguito, in casi di aggiunte, dovunque esistano settori liberi sulla superfice del dischetto.
Il file si contrae quando si cancellano records per cui, con questa operazione si rendono disponibili nuovamente dei settori precedentemente allocati.
Alla funzione di distribuzione del file sulla superfice del disco e' preposto il DOS cioe' Disk Operating System per cui la distribuzione su tutta la superfice del disco non presenta nessun problema quando si lavora con i files sequenziali.
E' presente un puntatore in ogni settore che dice in pratica dove indirizzarsi per la successiva lettura o scrittura.

APERTURA DI UN FILE SU DISCO

Dodici buffers di memoria sono disponibili su ogni unita' a disco per la manipolazione dei files.

****NOTA****

Un numero minore di buffers sono invece disponibili per i 1540/41. A questo proposito vedi le tavole nella seconda parte.

Non appena si accede ad ogni file disco due di questi buffers sono usati per operazioni di controllo.
Cio' lascia dieci buffers in ogni unita' attraverso cui si puo' accedere ai files data stessi.
Due buffers sono necessari per ogni operazione di

apertura di un file sequenziale.
Tre buffers sono necessari per le operazioni di apertura di un file relative.
Per questo il Basic puo' avere fino a cinque files sequenziali aperti simultaneamente su ogni unita' a disco.

Si puo' incrementare, ma solo fino ad un certo punto il numero di files aperti contemporaneamente. Infatti ogni file aperto richiede un suo proprio indirizzo secondario e solo 13 indirizzi secondari sono disponibili per i files di dati di qualsiasi tipo.

INDIRIZZI SECONDARI

Il Basic usa 16 indirizzi secondari: da 0 a 15. Ogni comando di OPEN nel Basic deve specificare un indirizzo secondario.Gli indirizzi secondari sono usati nella seguente maniera:

1-L' indirizzo secondario 0 e' usato per caricare i programmi dal disco alla memoria centrale del computer.

2-L' indirizzo 1 e' usato per salvare i programmi dalla memoria centrale del computer all' unita' a disco.

3-Gli indirizzi secondari da 2 a 14 sono usati per accedere ai files di dati (sono appunto 13 come ricordavamo prima).Si puo' selezionare uno qualunque di questi indirizzi secondari, ricordando che pero' poi non possono essere usati per un'

altra operazione di OPEN su altro file di dati.

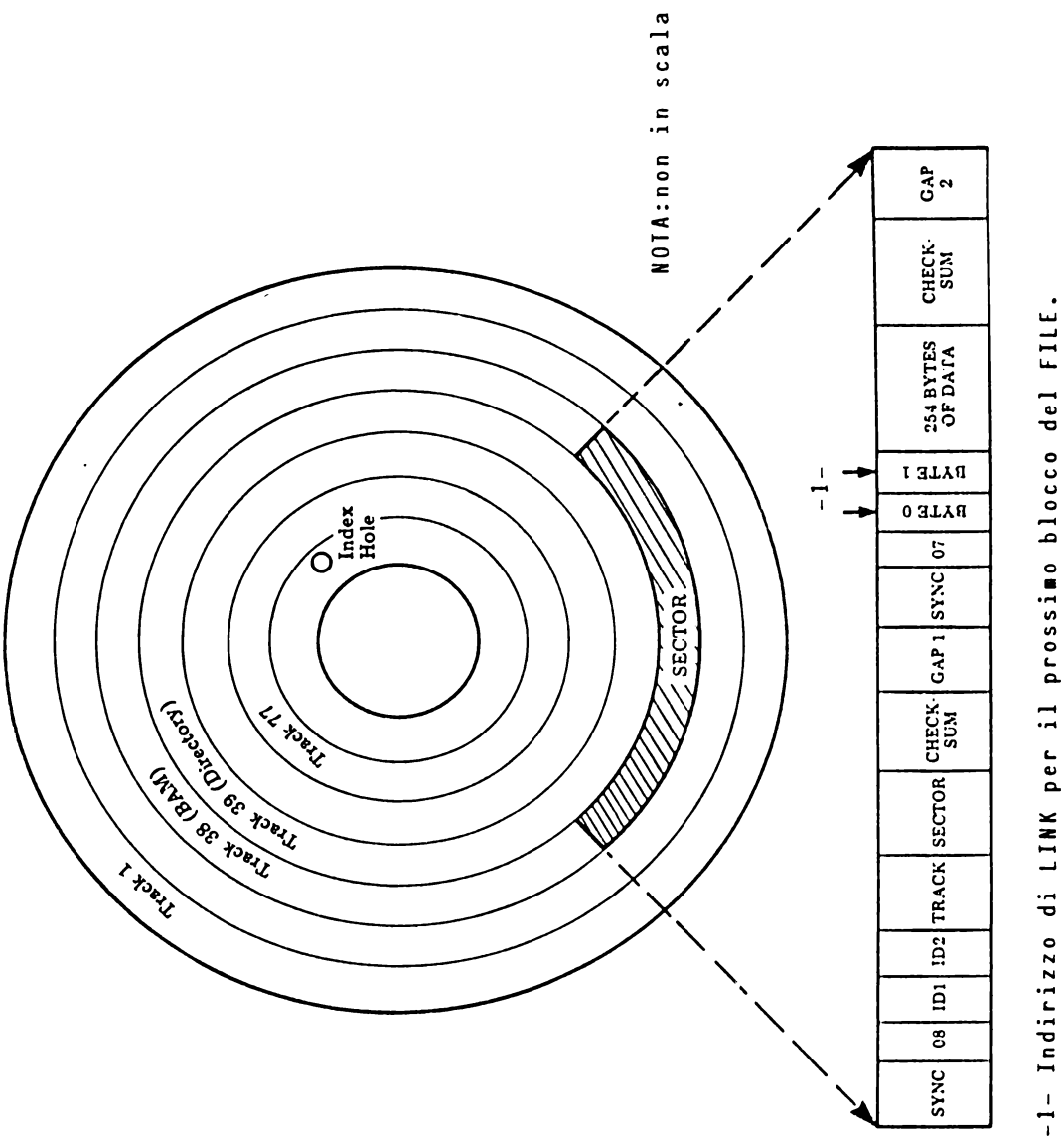
4-L' indirizzo secondario numero 15 apre uno speciale " CANALE DI COMANDO" che e' usato per accedere allo STATUS del dischetto e per consentire una delle speciali operazioni che vedremo successivamente.

IL CANALE DI COMANDO (15)

Il canale di comando necessita di una particolare attenzione perche' e' veramente molto importante.
Usando il disco del VIC, come del resto quelli della serie 3000, si dovrebbe sempre aprire il canale di comando per effettuare una qualsiasi operazione su disco.
Si dovrebbe inoltre lasciare questo canale aperto fino a quando si operi comunque ed in qualsiasi modo su disco.
Inoltre come abbiamo detto si usa il canale di comando per le operazioni speciali su disco e per interrogarlo sullo STATUS.

-2- Indirizzo di LINK per il prossimo blocco del FILE.

Fig. 6 - Formato dei dati in un singolo settore.
1540/1541-2031-2040-3040-4040



-1- Indirizzo di LINK per il prossimo blocco del FILE.

Fig. 7 - Formato dei dati in un singolo settore.
8050-8250

CAPITOLO NONO

OPERAZIONI SU DISCO

In aggiunta alle operazioni di scrittura e lettura files su dischetto che vedremo separatamente e dettagliatamente per ogni tipo di accesso , il Basic della Commodore relativo a questo tipo di unita' consente le seguenti operazioni:

1-Preparazione di un nuovo dischetto.

2-Cancellazione di un disco vecchio e preparazione per un nuovo uso.

3-Visualizzazione della Directory del disco per vedere quali file sono immagazzinati, quanto spazio questi hanno occupato e quindi quanto ne resta utilizzabile.

4-Copia di un file

5-Copia di un intero dischetto

6-Cancellazione di un file o rimpiazzo dei files

PREPARAZIONE DI UN DISCO E INIZIALIZZAZIONE

Introduzione

A differenza di quanto avviene per la cassetta non si puo' prendere un dischetto vergine, inserirlo nel drive ed incominciare a scrivere i dati.
Per prima cosa infatti la superfice magnetica deve essere preparata ad accogliere i dati, i settori devono essere fissati e poi devono essere scritte la Directory e la BAM.
Inoltre al dischetto deve essere assegnato un nome.

Si possono usare i dischi di una marca che si desidera, ma la casa che li costruisce non sa a priori se saranno adoperati su una marca di computer invece che su un' altra e all' interno della stessa marca essi possono venire utilizzati per un numero diverso di traccce e settori, ecco il motivo della preparazione o meglio della loro IDENTIFICAZIONE.

Inoltre si puo' ripreparare per un nuovo uso un vecchio dischetto, naturalmente purche' sia in condizioni fisiche integre e sopratutto non sia rigato.
Questa operazione cancella naturalmente tutti i dati vecchi, compresa la BAM e la Directory.

Di norma la preparazione di un dischetto per il suo uso viene fatta in modo diretto, anche se questa routine puo' essere inserita in un menu' di programma.

Preparazione

Per preparare un dischetto si deve per prima cosa eseguire un OPEN sul canale di comando.
Poi si eseguira' un comando PRINT# usando il file logico specificato nella lista dei parametri del comando OPEN.
Il comando PRINT# deve avere la seguente lista di caratteri, o parametri, racchiusa fra virgolette:

NEW o N

che identifica appunto l' operazione da eseguire.

: (due punti)

di separazione

NOME DEL DISCO

un nome qualsiasi che vogliamo dare

, (virgola)

anche questa di separazione

XX

identificatore del disco

E quindi il formato generale del comando che segue il PRINT#sara':

"NEW:NOME DEL DISCO,XX"

NOTA

NEW puo' essere rimpiazzato o abbreviato con la sola lettera N.
Il nome del disco deve essere una stringa di lunghezza non superiore ai 16 caratteri.
XX deve essere un coppia di caratteri alfanumerici.

Nel comando OPEN con il quale si accede al canale di comando si puo' mettere un qualsiasi numero di file logico, ma si deve specificare che l' unita' fisica e' la numero 8 e l' indirizzo secondario che deve essere il numero 15.

Il comando NEW viene usato su un dischetto non FORMATTATO oppure su un dischetto che l' utente vuole riformattare e del quale quindi non interessano piu' i dati.
Quando si usa il modo RIFORMATTAZIONE di un disco vecchio sara' cancellata la Directory preesistente e reinizializzata la BAM rendendo quindi nuovamente disponibili tutti i blocchi del dischetto.
In questo caso non dovremo specificare XX cioe' l' identificatore.
Vediamo qualche esempio.

```
OPEN 1,8,15

PRINT#1,"NEW:ESEMPIO,10"
```

RISULTATO: Viene aperto il canale di comando, formattato un disco che avra' per nome ESEMPIO e per identificatore 01.

Esempio:

```
OPEN3,8,15

PRINT#3,"N:TEST,11"
```

RISULTATO: Come il precedente ma con un disco di nome TEST e identificatore 11.

Riportiamo inoltre un esempio di riformattazione di un dischetto usato.

```
OPEN15,8,15

PRINT#15,"N:NUOVO"
```

RISULTATO: Con questo comando al disco usato viene assegnato un nome NUOVO, la Directory e la BAM vengono riformattate e resta solo l' identificatore preesistente.

NOTA

E' bene ricordare che questa ultima procedura potra' funzionare SOLO se il dischetto e' gia'

stato formattato in precedenza e su quello stesso tipo di unita'.
Il tempo necessario per formattare un disco e' di circa 2 minuti.
Molto inferiore e' invece l' operazione di riformattazione.

NOTA

Il comando NEW al disco, che infatti viene dato fra virgolette non deve essere confuso con il NEW del BASIC che cancella il programma in memoria ed azzera le variabili.

ERRORI

Se per una qualsiasi ragione il disco non puo' essere formattato allora il LED rosso presente sulla parte frontale del drive, iniziera' a lampeggiare (nelle unita' a doppio floppy si accende solo la luce rossa).

La ragione puo' essere una delle seguenti:

1 - Si e' dimenticato di inserire il disco nell' unita' oppure si e' dimenticato di chiudere le alette o si e' inserito il disco al rovescio.
A questo proposito e' bene ricordare che il metodo di aprire un' altra finestrella per poter utilizzare il dischetto da tutte e due le parti e' pratica assolutamente da evitare.

2-I comandi non sono stati dati correttamente.

E' bene porre particolare attenzione alla punteggiatura che e' l' errore in cui si cade piu' di frequente. Essa infatti deve essere SCRUPOLOSAMENTE osservata nelle sue regole.

3-Il disco ha la fascetta di protezione.

4-Il dischetto e' realmente difettoso.

5-L' unita'a a dischi e' difettosa.

NOTA

Sara' bene vedere in fondo al manuale le tavole di errore per un maggior approfondimento.

INIZIALIZZAZIONE

Benche' non sia indispensabile questa funzione sul drive 1540 o 1541 puo' accadere talvolta di doverla usare.
Per inizializzare il dischetto e' necessario eseguire un OPEN sul canale di comando e successivamente un comando di PRINT#seguito dalle parole fra virgolette "INITIALIZE" o dalla letterea "I".
Il comando "I" allinea la testina di lettura/scrittura con la traccia 1 del dischetto.
Successivamente la testina si posiziona sulla traccia 18, legge l' etichetta del disco ed il suo identificatore e carica queste informazioni nella memoria del Disk Operating System.

I dischi sono normalmente inizializzati in modo

programma e nessun dato sulla superfice del disco e' variata durante questa operazione che quindi puo' anche avvenire con la finestrella coperta.

Esempio

```
10 OPEN1,8,15

20 PRINT#1,"INITIALIZE"
```

oppure

```
10 OPEN3,8,15

20 PRINT#1,"I"
```

Si puo' anche usare la forma abbreviata:

```
OPEN15,8,15,"I"
```

che si adopera in forma diretta quando si inizia ad operare su disco.

VALIDATE

Dopo che un dischetto e' stato usato per molto tempo, puo' succedere che la Directory debba essere riorganizzata.
Infatti quando dati e programmi sono stati ripetutamente salvati (SAVE) e cancellati

(SCRATCH), di queste operazioni possono esserci rimaste numerose tracce, in particolare in piccoli blocchi sparpagliati, appunto troppo piccoli perche' possano essere riutilizzati.
In effetti la funzione di VALIDATE e' quella di cancellare tutti i files presenti nella DIRECTORY e di ricostruirne una nuova.
Se durante questa operazione viene incontrato un errore allora la funzione di ricostruzione viene sospesa e si ritorna alle condizioni di partenza.

Il comando VALIDATE riorganizzerra' allora il dischetto in modo tale che si possa disporre del massimo spazio effettivamente disponibile.
Un' altra importante funzione di questo comando e' quella di rendere accessibili tutti i files che erano stati aperti ma non chiusi.

!!!ATTENZIONE!!!

C' e' un pericolo nell' uso di questo comando. Quando si usino i FILES RANDOM i blocchi ALLOCATI saranno DE-ALLOCATI con questo comando. (vedere per questo anche il capitolo sui FILES-RANDOM). Per questo motivo il VALIDATE non dovrebbe mai essere usato quando in un dischetto sono presenti i files random.

Naturalmente e' un comando che si usa in forma diretta in una delle due forme:

```
PRINT#15, "VALIDATE"

PRINT#15,"V"
```

RENAME

Questo comando consente di cambiare nome ad un file di programmi o di dati.
In effetti si tratta di una operazione molto veloce perche' l' unico cambiamento che avviene e' nella Directory del disco.
Sul disco naturalmente non deve esistere gia' un file con lo stesso nome utilizzato nel RENAME perche' in questo caso l' operazione non potra' avvenire ed avremo una segnalazione di errore:

FILE EXISTS.

Il formato di RENAME e':

PRINT#15,"RENAMEO:vecchio nome=nuovo nome"

o nella forma abbreviata R al posto della lettera RENAME.

SCRATCH

Questo comando consente di cancellare files e programmi dal disco rendendo disponibili i blocchi per nuove informazioni.
Si possono cancellare programmi uno alla volta o in gruppo come possiamo vedere dagli esempi. Il formato generale del programma e' il seguente:

PRINT#15,"SCRATCHO:nome del programma"

o abbreviando, S al posto della parola SCRATCH.

ESEMPI

Ammettiamo che siano presenti i seguenti files:

TEST,TRAIN,TRUCK,TAIL

possiamo usare:

```
PRINT#15,"S0:TR*"
```

se si desideri cancellare sia TRAIN che TRUCK. Usando invece:

```
PRINT#15,"S0:T*"
```

li cancelleremo tutti. Cancelleremo cioe' tutti i files che iniziano per T.

Se per esempio la directory contenesse i files KNOW e GNAW usando:

```
PRINT#15,"S0:?N?W"
```

cancelleremo ambedue i programmi in quanto il ? sostituisce i caratteri ignoti all' inizio o nel mezzo del nome del file.

COPY

Questo comando consente, come del resto e' implicito nel nome, di effettuare una copia di un qualsiasi file di dati o programmi.
Nel caso si disponga di un solo drive e' ovvio che la copia puo' essere fatta solo sullo stesso dischetto.
Il formato di questo comando e':

```
PRINT#15,"COPY0:nuovo file=0:vecchio file"
```

oppure usando la lettera C al posto della parola COPY.

Questo comando puo' anche essere usato in modo interessante per la concatenazione di files sequenziali.

Esempio:

```
PRINT#15,"CO:MAILING
FILE=0:NOME,0:INDIRIZZO,0:TELEFONO"
```

Il comando COPY, in particolare per le unita' a singolo floppy presenta non pochi inconvenienti. Per questo e' stato messo a punto una procedura particolare (programma) che si trova in vendita con relativa facilita'.

CAPITOLO DECIMO

I COMANDI DEL BASIC 4.0

Introduzione

Riportiamo ora i comandi disco disponibili SOLO sulle versioni BASIC 4.0

DIRECTORY

Questo comando fara' apparire la Directory sullo schermo senza disturbare il contenuto della memoria.
Si puo' usare il formato completo del comando:

DIRECTORY che ci dara' la Directory di ambedue i dischetti.

DIRECTORY D1 o D0 che ci dara' la Directory di un solo drive.

Oppure di puo' usare la forma abbreviata:

diRd0 o diRd1

NOTA. Al posto del Directory puo' essere adoperata la parola CATALOG per intera o con l' abbreviazione:

CaD0 o CaD1

COLLECT

Questo comando ha la stessa funzione del VALIDATE. Le operazioni a cui da luogo sono le seguenti:

-Ricrea la mappa di disponibilita' dei blocchi in accordo con i dati validi su disco.

-Cancella dalla Directory i files che non sono stati chiusi in modo corretto.

Il formato dell' istruzione COLLECT e':

COLLECT Dx

x= il numero del drive.

Nel caso non si specifichi il numero di drive viene assunto il valore 0, cioe' l' operazione di Collect verra' fatta sul dischetto presente sul drive 0.

DUPLICATE

Questo comando esegue prima di tutto la formattazione del dischetto di destinazione e poi trasferisce ciascun blocco di informazioni dal dischetto sorgente al dischetto destinazione, creando una copia esatta del disco sorgente. Poiche' il DOS 1, DOS 2, ed il DOS 2.5 usano

protocolli di formattazione diversi questo comando non puo' essere usato indifferentemente con dischetti preparati su drives diversi.
Per evitare confusioni riportiamo una breve guida di riferimento.

Il comando DUPLICATE puo' essere usato con:

-2040 e 3040 quando si usino dischetti formattati sulla 2040 e 3040.

-4040 quando si usino dischetti formattati sulla 4040.

Il comando DUPLICATE non puo' essere usato con:

-2040 e 3040 quando si usino dischetti formattati sulla 4040.

-4040 usando dischetti formattati sulla 2040 e3040.

NOTA

Per riprodurre dischetti qualora ci si trovi nelle condizioni appena descritte e' necessario utilizzare il programma COPY ALL DISK che si trova sul dischetto TEST DEMO fornito con il drive.

Il formato dell' istruzione DUPLICATE e' il seguente:

```
PRINT#1,"DUPLICATE ddr=sdr"
```

oppure con la notazione abbreviata:

```
PRINT#1,"Dddr=sdr"
```

dove:

ddr= dischetto di destinazione

sdr= dischetto sorgente.

E' molto importante non rovesciare l' ordine in cui viene dato il numero di drive perche' in questo caso si andrebbe incontro ad una perdita dei dati e tale errore non sarebbe rimediabile in alcun modo. Sara' quindi buona norma proteggere con l' apposita etichetta sulla finestra il disco sorgente.

BACKUP

Il comando BACKUP esegue le stesse funzioni del comando visto in precedenza. Il suo formato e':

```
BACKUP Dsdr TO Dddr
```

dove sdr e ddr hanno gli stessi significati visti per il DUPLICATE.
E' importante notare che il formato dell' istruzione BACKUP differisce dal formato dell' istruzione DUPLICATE nel senso che l' ordine dei drive e' rovesciata.

CONCAT

Questo comando permette all' utente di concatenare files. Il formato di questa istruzione e':

```
CONCAT Dsdr,"sfn" TO Dddr, "dfn"
```

A fine operazione il file chiamato dfn sul drive ddr conterra' sia il contenuto del file dfn che del file sfn.

Esempio:

```
CONCAT D0,"ALFA" TO D1,"BETA"
```

Dara' come risultato in BETA del drive 1 il contenuto in dati di BETA e ALFA.
ALFA restera' inalterato.

Questo comando puo' essere molto utile per la gestione della programmazione strutturata per la quale prevediamo di uscire con un apposito volume.

PROCEDURA DI CARICAMENTO VELOCE

Questo comando e' valido con il DOS 2 ed il Basic 4.0 e con le unita' periferiche 4040 e 8050.
Questo comando carica il primo file che si trova sul disco del drive 0 nella memoria. Per essere sicuri di accedere effettivamente al primo programma che si trova sul dischetto il comando deve essere il primo che viene dato dopo un

avviamento.
Eseguire le fasi seguenti:

- Spegnere il computer e riaccenderlo.

- Assicurarsi che il disco che si trova nel drive 0 contenga un programma come primo file e che naturalmente la porta del drive sia chiusa.

- Premere contemporaneamente il tasto SHIFT ed il tasto RUN/STOP.

Il computer eseguira' per prima cosa la inizializzazione del disco presente nel drive 0, andra' alla ricerca del primo programma su questo disco e lo carichera' in memoria.

!!!ATTENZIONE!!!

Fare bene attenzione a che il primo file della Directory sia un programma perche' in caso di file dati sia esso sequenziale, relativo o USR avremo una segnalazione di errore che pero' puo' essere dovuta anche ad altre cause.
In caso di una qualunque segnalazione di errore ripetere l' operazione dall' inizio o inizializzare normalmente il disco.

Eseguita l' operazione di SHIFT e RUN/STOP sullo schermo:

```
dl"*

        searching for 0:*
        loading
```

```
run
```

Quando si esegue quindi questa operazione il computer esegue automaticamente le operazioni di LOAD e RUN e non e' necessario dare alcun comando.

CAPITOLO UNDICESIMO

COMANDI BASIC PER LA MANIPOLAZIONE DI DATI SU DISCO.

I seguenti comandi consentono all' utente di operare su disco per comunicare alla periferica un qualsiasi tipo di dati e per poi riutilizzarli.

In tutte le versioni del DOS cioe' del Disk Operating System (Sistema Operativo su Disco) sono disponibili i seguenti comandi:

```
OPEN lfn,8,sa,"dr:fn"
CLOSElfn
LOAD "dr:fn",8
SAVE "dr:fn",8
VERIFY"dr:fn",8

PRINT #
GET #
INPUT #
```

Questi comandi sono invece disponibili solo per i DOS superiori cioe' per quelli delle serie 4/8000, che nel seguito di questo capitolo possono essere indicati, per brevita' anche se non proprio con assoluta correttezza di termini, come appartenenti al BASIC 4.0.

```
DOPEN#lfn,"fn"
DCLOSE#lfn
```

```
DLOAD "fn"
DSAVE "fn"
RECORD#lfn,R,B
```

Dove:

```
lfn = numero di file logico
fn  = nome del file
dr  = numero del drive
sa  = indirizzo secondario
lf  = file logico
```

Ricordiamo che come numero di periferica il disco ha il numero 8, la stampante il 4 (o il 5) e la seconda cassetta il 2.
SAVE e DSAVE

Questo comando consente di trasferire un programma dalla memoria del computer al dischetto per il suo immagazzinamento.
Cio' puo' essere ottenuto con SAVE in tutti i tipi di Basic e con DSAVE nei computer con BASIC 4.0.

Ogni dato trasferito con i comandi SAVE e DSAVE e' automaticamente definito e quindi registrato dal Disk Operating System come file programma e viene registrato nella Directory del disco con la lunghezza in blocchi, il nome, e l' indicatore PRG.
Quindi entrambi i comandi trasferiscono dati in forma di programmi dalla memoria del computer ad un dato dischetto.
Sara' necessario specificare il numero del drive, il nome del programma ed il numero di periferica.
Il formato del comando SAVE e':

SAVE "dr:fn",dn

Dove:

dr = numero del drive (1 o 0)
fn = e' un nome qualsiasi, assegnato dall' utente, di non piu' di 16 caratteri.

Ricordiamo che il nome del file per il disco e' obbligatorio a differenza di quanto accade per la cassetta e che sono contati, come numero di caratteri anche gli spazi bianchi.

dn = e' il numero della periferica disco che normalmente, ma non necessariamente deve essere 8.

Il seguente esempio e' stato messo per far prendere un minimo di pratica nell' uso di questo comando. Si compone infatti di una linea di programma da digitare e di un comando di SAVE che sara' dato invece in modo diretto.

```
10 PRINT"PROVA DI REGISTRAZIONE"

SAVE"0:PROVA",8
```

Il comando DSAVE esegue le stesse funzioni viste in precedenza solo che si da in una forma diversa e piu' semplice ed e' valido solo per i dischi BASIC 4.0.

Il formato e':

```
DSAVE"fn"Ddr
```

Per cui il precedente esempio sarebbe stato scritto:

```
10 PRINT"PROVA DI REGISTRAZIONE"
DSAVE"PROVA"D0
```

LOAD e DLOAD

Un programma o comunque una serie di dati immagazzinati su dischetto come programma puo' essere riletto con uno dei comandi LOAD (per tutti) o DLOAD (per i BASIC 4.0).

I comandi LOAD e DLOAD trasferiscono quindi un file PRG da un dato dischetto nella memoria interna RAM del computer.
Si deve specificare il numero del drive, il nome del programma ed il numero della periferica.
Il formato del comando LOAD e':

```
LOAD "dr:fn",dn
```

Dove per i parametri valgono le regole precedentemente esposte tranne per il fatto che il nome del file (parametro fn) in questo caso dovra'

essere un programma salvato precedentemente con un comando SAVE.

Il seguente esempio illustra come un programma viene caricato da disco nella memoria del computer e quindi eseguito.
Dato che si tratta del seguito dell' esempio visto in precedenza sara' necessario procedere come segue.

Digitare NEW e premere RETURN per eseguire un CLEAR, cioe' la pulizia della memoria del computer e vedere cosi' senza alcuna confusione come il comando stesso operi.

ATTENZIONE!!!

Non confondere il NEW del Basic con il NEW del comando disco che serve per la sua formattazione.

```
LOAD"0:PROVA",8

LOADING PROVA

READY
```

dopo il:

```
RUN
```

Sullo schermo apparira' il messaggio:

```
PROVA DI REGISTRAZIONE
```

Il comando DLOAD esegue le stesse funzioni solo che deve essere dato in un' altra forma:

```
DLOAD"fn",Ddr
```

Ricordiamo a proposito di questo comando le seguenti particolarita'.

Dato che come si vede non e' necessario specificare il numero della periferica sara' bene ricordare che questa automaticamente sara' presa come periferica 8.
Inoltre, non dichiarando nemmeno il numero del DRIVE, la ricerca sara' fatta sul DRIVE 0.

NOTA

La corretta esecuzione di un comando LOAD o DLOAD porta anche alla chiusura di tutti i files aperti. Per questo sara' necessario un nuovo comando di OPEN per poter continuare a comunicare con il disco sia per l' invio di comandi che per ripristinare il canale di errore.

Ricordiamo inoltre che questi comandi portano alla distruzione del programma eventualmente presente in precedenza nella memoria del computer.

VERIFY

Il formato di questo comando e':

```
VERIFY"dr:fn",8
```

Questo comando verifica che un programma, il cui nome e' specificato nel parametro "fn", immagazzinato su floppy disk contenga le stesse informazioni di un programma presente nella memoria del computer.
E' lo stesso di quello visto in precedenza per la cassetta.

NOTA

E' molto importante ricordarsi di eseguire un VERIFY dopo aver salvato un programma, ma sopratutto e' essenziale non dimenticare che fra la fase di SAVE e quella di VERIFY non devono assolutamente essere apportate variazioni sul programma in memoria, o comunque sulla memoria stessa.

Continuando con l' esempio precedente, dopo aver dato il SAVE procedere come segue:

```
VERIFY"0:PROVA",8
```

sullo schermo apparira':

```
SEARCHING FOR 0:PROVA

VERIFYNG
```

E se e' stata verificata la corretta rispondenza fra il programa presente nella memoria RAM del computer e quello precedentemente registrato su disco, allora sullo schermo sara' visualizzato:

```
OK
READY.
```

con dopo il solito cursore lampeggiante.

Se invece verra' visualizzato un messaggio di :

```
VERIFYNG ERROR
```

sara' necessario eseguire nuovamente un SAVE del programma e operare di nuovo la verifica dello stesso.

Il comando VERIFY puo' essere utilizzato anche nella forma piu' semplice:

```
VERIFY"*",8
```

che consentira' di verificare l' ultimo programma salvato su disco senza dover riscrivere il nome del file.

NOTA

E' importante ricordare che non si puo' salvare su disco un programma due volte con lo stesso nome a meno di non ricorrere a uno di questi due artifizi.

Prima si effettua la cancellazione (SCRATCH) del programma che ha dato luogo ad un errore di verifica e poi si esegue nuovamente la procedura di SAVE.

Oppure si esegue direttamente il SAVE facendo pero' precedere il numero del drive dal segno § (carattere chiocciola) come verra' spiegato in seguito.

OPEN

Come e' stato gia' spiegato in precedenza e piu' volte, questo comando fissa una corrispondenza univoca tra un numero di file logico ed un file esistente su disco.
Questo comando riserva anche uno spazio nel buffer dell' unita' a disco per le operazioni sul file che deve essere aperto.

Il formato completo del comando OPEN e' il seguente:

OPENlfn,dn,sa,"dr:fn,ft,modo"

dove:

lfn = e' il numero di file logico

dn = e' il numero della periferica e in questo caso normalmente il numero 8.

sa = e' l' indirizzo secondario che puo' essere un numero qualsiasi fra 2 e 14 e deve essere usato sia per l' input che per l' output dei dati come viene specificato nel modo. (vedi anche la nota seguente).

dr = numero del drive che potra' essere 0 o 1 nelle unita' a doppio drive.

fn = nome del file.

ft = tipo del file che potra' essere :

SEQ = Sequenziale
USR = Utente
REL = Relative
PRG = Programma

modo = che descrive in che modo il canale di trasferimento dati debba essere utilizzato. Potra' essere quindi una delle opzioni:

READ (R) per la lettura

WRITE (W) per la scrittura.

NOTA

L' indirizzo secondario 15 si riferisce al canale di comando e di errore ed ha uno speciale utilizzo che sara' visto a parte.
Gli indirizzi secondari 0 e 1 sono riservati dal Sistema Operativo sia dell' unita' centrale che del

disco (DOS) per il caricamento ed il salvataggio di programmi.
In pratica sono associati automaticamente ai comandi SAVE e LOAD.

Vediamo ora alcuni esempi di utilizzo rimandando alla parte della cassetta per l' approfondimento dell' uso di OPEN:

Esempio 1

```
OPEN2,8,2,"0:FILE1,SEQ,WRITE"
```

che potra' essere dato anche nella forma abbreviata:

```
OPEN2,8,2,"0:FILE1,S,W"
```

che indica di aprire il file logico numero 2 sulla periferica 8 (il disco), con un indirizzo secondario 2 sul drive 0.
Il file di nome FILE1 sara' un file sequenziale aperto per scriverci (WRITE).

Esempio 2

```
OPEN3,8,9,"1:PROVA,PRG,WRITE"
```

che potra' essere dato anche nella forma abbreviata:

```
OPEN3,8,9,"1:PROVA,P,W"
```

e che indica di aprire il file logico numero 3 sulla periferica 8, con indirizzo secondario 9 sul drive 1.
Il file trattato in questo caso sara' il file PROVA di tipo programma anche questo aperto per scriverci.

Esempio 3

```
OPEN8,8,8,"0:PIPPO,USR,READ"
```

che nella forma abbreviata potra' essere scritto equivalentemente:

```
OPEN8,8,8,"0:PIPPO,U,R"
```

e che ordina di aprire il file logico numero 8 sull' unita' a dischi con indirizzo secondario 8 sul drive 0.
Il file in questo casi si chiama PIPPO e' un file USER o utente ed e' stato aperto per eseguire un' operazione di lettura.

Il contenuto di un file esistente puo' essere rimpiazzato facendo precedere il numero del drive, in questo caso nel comando OPEN, dal segno § (at o a commerciale o chiocciola) come mostrato nel

seguente esempio:

```
OPEN3,8,5,"§O:PIPPO,USR,WRITE",
```

NOTA

E' buona norma ricordarsi che questa operazione potrebbe dare esiti negativi e potrebbe arrivare anche a distruggere uno o piu' files nel dischetto se non viene lasciato un numero di blocchi liberi almeno superiore di uno o due alla grandezza del file che si vuole rimpiazzare.
Questo sistema inoltre puo' essere utilizzato anche con il comando SAVE.

Nel caso che il file specificato da rimpiazzare non esista la procedura di OPEN sara' normalmente eseguita e, nel caso precedente scritto un file UTENTE di nome PIPPO.

Alcuni dei parametri presenti nel comando OPEN possono essere assegnati ad una variabile come mostrato negli esempi seguenti.

Esempio 1

```
FL$="O:PROVA,SEQ,READ"
```

Per cui potremo usare in programma:

```
OPEN1,8,3,FL$
```

Esempio 2

```
FL$="0:PLUTO"

OPEN2,8,5,FL$+",SEQ,WRITE"
```

Questo tipo di utilizzo e' particolarmente conveniente in programma.

DOPEN

Questo comando e' disponibile solo sulle macchine che impiegano il BASIC 4.0.
Quando e' utilizzato quindi con dischi 4040 o 8050/8250 deve essere impiegato per creare files relatives o sequenziali di lunghezza fissa.

Il formato di DOPEN e':

```
DOPEN#lfn,"fn",Ddr,Lrl,(ON Udn)(,W)
```

dove:

lfn,fn,dr sono gli stessi parametri spiegati in precedenza per il comando OPEN.

Lrl = definisce la lunghezza del record che deve appunto essere uguale a rl.

ON Udn = specifica il numero di periferica e che se

manca sara' automaticamente posto uguale a 8.

W deve essere specificato per consentire il modo scrittura.

Se W non e' specificato per i files sequenziali, questi verranno aperti per una operazione di lettura.

CLOSE

Questo comando serve per chiudere un file precedentemente aperto con un comando OPEN.
Il suo formato e' il seguente:

```
CLOSE lfn
```

dove:

lfn = e' il numero di file logico di un file aperto con il comando OPEN.

Ricordarsi di chiudere sempre un file dopo aver terminato di lavorarci.
Non e' infatti possibile avere piu' di 10 file contemporaneamente aperti sul computer e piu' di 5 sul disco.
Per questo e' buona norma prendere l' abitudine di chiudere un file al piu' presto possibile perche' questo metodo ci consentira' di avere una riserva da poter opportunamente impiegare quando se ne manifesti la necessita' e quindi di avere anche il massimo numero dispobile di files.

DCLOSE

Anche questo comando e' disponile solo per gli utenti di macchine con il Basic 4.0 e serve per chiudere i files aperti con il comando DOPEN.

La sintassi del comando DCLOSE e' la seguente:

```
DCLOSE#lfn
```

dove:

lfn = e' il file che deve essere chiuso.

Il comando DCLOSE puo' essere impiegato anche nella seguente maniera e formato:

```
DCLOSE ON Udn
```

dove:

dn = e' il numero di periferica dell' unita' a dischi (di norma ed in mancanza di indicazioni e' 8).

Quando e' utilizzato in questa forma il comando DOPEN chiudera' tutti i files attivi sull' unita' specificata.

Vediamo alcuni esempi:

Esempio 1

```
DCLOSE
```

chiudera' tutti i files aperti sull' unita' a dischi 8.

Esempio 2

```
DCLOSE#5
```

chiudera' solo il file logico numero .

CHIUSURA DEL CANALE DI COMANDO

Chiudendo il canale di comando si ottiene la chiusura di tutti i canali associati con l' unita' a dischi.

Il seguente esempio illustra un tipo di situazione in cui piu' canali sono chiusi da un singolo comando CLOSE.

Esempio

```
A       OPEN1,8,15
B       OPEN3,8,2,"0:FILE1,S,W"
C       OPEN4,8,5,"0:FILE2,S,W"
```

```
D       PRINT#3,"DATI IMPORTANTI"
E       PRINT#4,"ALTRI DATI"


F       OPEN3,4

G       ?FILE OPEN ERROR
        READY.
```

Vediamo un breve commento:

A - Viene aperto il canale di comando.

B/C Vengono aperti dei canali di dati per operazioni di scrittura.

D/E Vengono inviati dei dati ai files aperti in precedenza.

F - A questo punto viene aperto, supponiamo per errore, un canale sulla stampante

G - Viene visualizzato un messaggio di errore sullo schermo ed il computer va in Ready.

Poiche' e' stato incontrato un errore, tutti i files logici del computer verranno chiusi, ma i canali di comunicazione con l' unita' a dischi sono ancora aperti.
Per chiudere i canali del disco sara' necessario digitare:

```
OPEN1,8,15
```

CLOSE1

E solo a questo punto che tutti i canali di dati sull' unita' a disco saranno appropriatamente chiusi.

CHIUSURA DEL CANALE DATI

Il comando CLOSE chiude un file ed il canale di comando o il canale di dati associati con il file stesso.
Ogni volta che si chiude un file aperto con un canale di scrittura, questa operazione, cioe' la chiusura del file, scrivera' il BLOCCO DI FINE DATI sul disco stesso ed aggiornera' la Directory del disco.
Quando invece si chiude un file che era stato precedentemente aperto per la lettura (cioe' utilizzando un canale di lettura), allora quel canale sara' semplicemente chiuso senza nessuna altra operazione automatica.

NOTA

Quando un drive e' inizializzato con comandi INITIALIZE,NEW,DUPLICATE o VALIDATE tutti i canali associati con quel drive saranno chiusi.
Per questo motivo i precedenti comandi detti di inizializzazione non dovrebbero essere utilizzati fino a quando tutti i files non siano correttamente chiusi.
In caso contrario infatti avremo la distruzione fisica del file stesso.

PRINT #

Questo comando, gia' visto a proposito della cassetta, serve ad inviare una stringa di comando alla periferica disco.

Il formato e':

PRINT#lfn,"stringa di comando"

dove:

lfn = numero di file logico preventivamente aperto utilizzando l' indirizzo secondario 15.

stringa di comando = e' appunto un comando di manipolazione di un file disco o semplicemente un comando al disco. Questi comandi sono stati gia' visti in precedenza e saranno anche approfonditi in seguito nella parte relativa alla trattazione dei vari tipi di files.

PRINT£ deve anche essere usato per trasmettere dati ad un file sequenziale o relative preventivamente aperto.
Deve essere utilizzato un punto e virgola (;) come carattere terminatore per ogni comando PRINT# quando si stia utilizzando computer provvisti di BASIC 3.0 per evitare di inviare LINE FEEDS estranei al dischetto.
Questi caratteri sono scritti sul dischetto come

parte del TERMINE DATI dalla routine BASIC PRINT#.

E' importante essere a conoscenza di questo aspetto, perche' il ritorno carrello da solo e' visto come CARATTERE DI TERMINE dal DOS.
Il LINE FEED e' allora immagazzinato nel file come primo carattere del successivo record.
Per evitare cio' utilizzate il seguente formato:

```
PRINT#2,"PIPPO";CHR$(13)
```

Il CHR$(13) e' il ritorno carrello necessario per un' appropriato termine del record sul disco.

Gli utenti del BASIC 4.0 non hanno la necessita' di seguire questa procedura, sebbene il suo uso non procuri niente di male.
Infatti nel BASIC 4.0 ogni file aperto con un lfn inferiore a 128 sopprime automaticamente il LINE FEED.
Per questo motivo si dovra' utilizzare il seguente formato:

```
PRINT#lfn,A$
```

Che assolvera' la funzione di scrivere la stringa correttamente senza per questo interferire con il record successivo.

Piu' variabili possono essere scritte su disco allo stesso tempo. Vediamo due esempi.

Esempio 1

```
PRINT#lfn,A$,B$,C$
```

che portera' come risultato la scrittura di una variabile composta dalla somma delle singole variabili e cioe':

```
A$+B$+C$
```

e che percio' potra' essere ricercata e riletta come unica variabile.

Esempio 2

```
PRINT#lfn,A$CHR$(13)B$CHR$(13)C$
```

invece avra' come risultato l' inserimento di 3 variabili separate da un ritorno carrello che quindi cosi' saranno scritte e poi di conseguenza cosi' ricercate all' interno del file.

INPUT

Il comando INPUT# e' utilizzato per trasferire informazioni da una periferica come appunto un disco nella memoria del computer.
INPUT# e' valido solo quando e' utilizzato in un programma e solo quando si riferisce ad un file logico che sia stato aperto (con OPEN) per l' ingresso dati.

Il formato di INPUT#e':

```
INPUT#lfn,A$
```

oppure

```
INPUT#lfn,A
```

dove:

lfn = e' un file preventivamente aperto utilizzando l' indirizzo secondario 15.

A$ = e' una normale variabile stringa che contiene i dati da trasferire.

A = e' una normale variabile numerica che contiene i dati da trasferire.

INPUT puo' essere utilizzata per trasferire piu' stringhe allo stesso tempo.

Esempio

```
INPUT#lfn,A$,B$,C$
```

dove

A$,B$ e C$ conterranno i dati che devono essere riletti dal computer.

Nell' esempio qui sopra e' necessario ricordarsi che i dati devono essere stati scritti con un CHR$(13) perche' se non troveranno il carattere di ritorno carrello come separatore, allora avremo la

la rilettura di una stringa intera o di una somma di stringhe.
Se si vogliono cioe' trovare tante stringhe separate si devono anche scrivere separatamente.

Ricordiamo che nessuna stringa puo' contenere piu' di 80 caratteri in fase di INPUT.
Vediamo ora qualche esempio.

Esempio 1

```
.
.
20 INPUT#2,A
.
.
```

Leggi da disco il prossimo dato che deve essere in forma numerica ed assegna alla variabile A il valore contenuto nel dato stesso.

Esempio 2

```
.
.
40 INPUT#2,C$
.
.
```

Leggi da disco il prossimo dato che deve essere in forma alfanumerica ed assegna alla variabile C$ il valore contenuto nel dato stesso.

Esempio 3

```
.
.
60 INPUT#7,B,D$
.
.
```

Leggi da disco i prossimi due dati che deve essere il primo numerico ed il secondo alfanumerico (o almeno verra' trattato come tale) ed assegnali rispettivamente alle variabili B e D$.

Per stringhe piu' lunghe di 80 caratteri e' necessario utilizzare il comando GET#.

GET #

Il comando GET# ha la stessa funzione del comando INPUT# visto in precedenza solo che opera su un Byte di dati per volta.
Anche questo comando non puo' essere utilizzato che in modo programma, e quindi come per il GET normale, non e' possibile adoperarlo in modo diretto, e solo quando e' riferito ad un file che sia gia' stato aperto.

Il formato e':

```
GET#lfn,A$
```

dove:

lfn = e' il solito numero di file logico preventivamente aperto con OPEN e che utilizzi un

indirizzo secondario 15.

A$ = e' la variabile stringa che conterra' i dati trasferiti.

GET£ deve anche essere usato per trasferire piu' bytes di informazioni o di dati ed e' quindi utile per ricercare stringhe che sono state scritte in precedenza su disco in un formato non accettabile per un comando di INPUT. Cioe' normalmente per stringhe superiori a 80 caratteri.
Vediamo un esempio commentato.

Esempio

```
10AA$=""
20FORI=1TO254
30GET#1fn,A$
40AA$=AA$+A$
50NEXT
```

Con questo programma sara' possibile trasferire da disco all' unita' centrale una stringa della lunghezza di 254 caratteri ed immetterne il contenuto in memoria, dell' unita' centrale, nella variabile AA$.

RECORD #

Il comando RECORD#e' utilizzato prima di comandi INPUT#, PRINT# o GET#per posizionare il puntatore del file (FILE POINTER) ad un desiderato record di

un file relative.
Per esempio se il puntatore del record e' fissato al di la' dell' ultimo record e viene utilizzato il PRINT#, viene generato il giusto numero di records per espandere il file al record richiesto.

Ricordiamo che questo comando, come del resto la gestione di file relatives sono disponibili solo per gli utenti che abbiano unita' equipaggiate con il BASIC 4.0.

Il formato del comando e':

RECORD # lfn,r,b

dove:

lfn = e' il numero di file logico preventivamente aperto con un comando DOPEN.

r = e' il desiderato numero di record.
Questo parametro puo' essere sia un nome di variabile che un valore.
Se e' un nome deve essere racchiuso fra parentesi, mentre se e' un valore deve essere compreso in un' intervallo fra 0 e 65535.

b = e' la posizione del byte richiesto entro il record. La posizione del byte e' opzionale.
b deve essere compreso fra 1 e 254.

Il seguente esempio illustra come il comando RECORD e' utilizzato con un INPUT#.

Esempio

```
10RECORD#1,120
20INPUT#1,A$
```

Dove nella linea 10 si usa il comando RECORD per selezionare il record interessato.
Nella linea 20 si esegue l' input del prossimo gruppo di dati come stringa e lo assegna ad una variabile A$.

Esempi piu' dettagliati sia di questo comando che dell' utilizzo dei files relative sono riportati nell' apposito capitolo.
Ricordiamo inoltre che si possono simulare i files relative anche su unita' 1540/1541 con apposita routine.

CAPITOLO DODICESIMO

I FILES RANDOM

I files ad accesso RANDOM o semplicemente RANDOM FILES sono creati indirizzando i blocchi di dati sul dischetto ed usando i Buffers di memoria disponibili sull' unita' stessa.
Ogni blocco di dati occupa un singolo settore, quindi i files RANDOM possono indirizzare direttamente questi blocchi di dati attraverso il loro indirizzo di traccia e settore. Nello stesso modo diretto e' possibile indirizzare i Buffers del disco. Ricordiamo che ogni unita' a disco ha 16 Buffers di memoria (a parte i 1540/1541) di 256 Bytes ciascuno.
I files Random sono creati usando un numero di subroutines che accedano direttamente alla superfice del dischetto ed ai Buffers.

!! ATTENZIONE!!!

Ancora una volta vi raccomandiamo di NON usare i files RANDOM in procedure operative se non siete degli esperti programmatori.
Tuttavia la fatica di imparare questo metodo di accesso al disco sara' ampiamente ripagato dalla capacita' di manipolazione dati che si otterra' al termine.

L' accesso RANDOM e' programmato usando comandi PRINT #con appropriati codici di testo nei parametri come potremo vedere in seguito.

I comandi PRINT#hanno accesso al canale di comando tramite l' indirizzo secondario 15.
I files RANDOM sono aperti con uno specifico Buffer assegnato ad ogni file logico tramite il suo indirizzo secondario.
La lista dei parametri del comando PRINT #usa l' indirizzo secondario per identificare i files logici ed assegnare i buffers.

Vediamo ora il formato del comando OPEN quando lo si usi con i files RANDOM.

```
100 OPEN lf,dev,sa,"#(bu)"
```

dove

lf= il numero di file logico specificato nel canale di comando dell'OPEN.

dev= il numero della periferica (normalmente 8)

se= indirizzo secondario (SECONDARY ADRESS) che dovrebbe avere un valore fra 2 e 14

bu= se e' presente sta ad indicare il numero del Buffer.

Sono presenti 16 Buffers da 256 bytes ognuno di cui i primi 3 sono usati dal Disk Operating System, mentre dal 3 al 15 sono disponibili. Se questo parametro non viene specificato, allora il DOS assegnera' il primo Buffer disponibile all' indirizzo secondario.

Si puo' eseguire un comando GET # immediatamente dopo l' apertura di un file RANDOM per determinare il numero di Buffer assegnato.

Tuttavia un comando GET #deve essere eseguito PRIMA di una qualsiasi operazione di INPUT o OUTPUT d' accesso ai files logici.

Vediamo un esempio di programma:

```
5 REM ASSEGNARE IL BUFFER 5 ALL' INDIRIZZO
SECONDARIO 4 USANDO IL FILE LOGICO 5
10 OPEN 2,8,4,"#5"
20 REM CONTROLLO STATUS
30 PRINTD$
40 REM VISUALIZZA N. BUFFER PER CONTROLLO
50 GET#2,A$:PRINTASC(A$)
60 PRINTD$
70 CLOSE2
80 STOP
```

I comandi di accesso ai Files Random sono inviati in seguito all' uso del PRINT # che assume la seguente forma generale:

```
10 OPEN 1f,8,15
20 PRINT#1f,"Parametro"
```

Dove "Parametro" identifica l' operazione di accesso al file Random. Il parametro poi e' composto di due parti:

Un comando

Una lista di parametri veri e propri.

Il comando puo' essere espresso in forma estesa che deve pero' terminare con i due punti (:), nel qual

caso il DOS fissa che a partire dal IV carattere della stringa "Parametro" inizi appunto la lista dei parametri e che quindi non deve essere usato il carattere (:).

I parametri effettivi della lista possono essere separati da una virgola, da uno spazio o da un carattere di salto.
Si usano le seguenti abbreviazioni per descrivere i parametri:

sa= indirizzo secondario

dr= numero del drive

t= numero di traccia

s= numero di settore della traccia selezionata.

p= puntatore del Buffer o selettore della posizione del carattere, che deve avere un valore fra 0 e 255.

adl= Byte basso (LOW-ORDER) dell' indirizzo di memoria

adh= Byte alto (HIGH-ORDER) dell' indirizzo di memoria.

nc= numeri di caratteri.

data= stringa dati con un certo numero di caratteri.

BLOCK-READ

Questo comando carica il settore del disco specificato entro un buffer. Usandolo in unione con gli altri comandi BLOCK permette di creare in Basic un sistema di files ad accesso casuale.

```
PRINT#1f,"BLOCK-READ:sa,dr,t,s"
```

oppure nella forma abbreviata:

```
PRINT#1f,"B-Rsa,dr,t,s"
```

oppure:

```
PRINT#1f,"B-R";sa;dr;t;s
```

Nel seguente esempio si apre il file logico 2, assegnando il buffer 5 all' indirizzo secondario 4, poi si carica il settore 0 della traccia 18 sul drive 0 entro il buffer 5.

Esempio:

```
5  REM---------------------------
10 REM APRE IL FILE LOGICO 2 ASSEGNANDO IL BUFFER 5
ALL' INDIRIZZO SECONDARIO 4
15 REM---------------------------
20 OPEN 2,8,4,"#5"
25 REM---------------------------
30 REMCARICA IL SETTORE  0  DELLA  TRACCIA  18  DEL
DRIVE 1 ENTRO IL BUFFER 5
31 REM
32 REM---------------------------
40 OPEN 15,8,15
50 PRINT#15,"B-R4,1,18,0"
55 REM---------------------------
56 REM
```

```
60 REM VISUALIZZA IL CONTENUTO DEL BUFFER
65 REM
70 REM VISUALIZZA I 256 BYTES DEL BUFFER IN 8 RIGHE
DI 32 NUMERI CIASCUNA
71 REM
72 REM----------------------------
75 PRINT"(SHIFT/CLEAR HOME)";
80 FOR I= 1 TO 8
90 FOR J=1 TO 32
100 GET#2,A$:IF A$="" THEN 100
110 PRINT ASC(A$)
120 NEXTJ
130 PRINT
140 NEXT I
150 CLOSE 2
160 CLOSE 15
170 STOP
```

BLOCK-WRITE

Questo comando serve a scrivere i contenuti di un buffer in uno specifico settore. Ha il seguente formato:

```
PRINT#lf,"BLOCK-WRITE:sa,dr,t,s"
```

o nella forma abbreviata.

Nel sottoindicato esempio i comandi eseguono la funzione di aprire il file logico 2, assegnando il buffer 8 all' indirizzo secondario 7.
I contenuti del buffer 8 sono scritti nel settore 10 della traccia 35 naturalmente sul drive 0.(Il discorso non e' valido per gli 8000).

```
10 OPEN 15,8,15,"IO"
20 OPEN 2,8,7,"#8"
30 PRINT#15,"B-P";7;0
40 PRINT#2,"PROVA DI SCRITTURA";CHR$(13);
50 PRINT#15,"B-WW";7;0;35;0
60 CLOSE 2
70 CLOSE 15
80 STOP
```

BLOCK-ALLOCATE

Questo comando serve per allocare i blocchi.
La mappa di disponibilita' dei blocchi e' scritta sul dischetto quando il file logico viene chiuso.
Se il blocco richiesto e' gia' stato allocato, allora il canale di errore identifica il prossimo blocco disponibile, mentre un errore NO BLOCK (error 65) viene visualizzato.
Se non ci sono blocchi disponibili, allora viene restituito uno 00 per i parametri di traccia e settore. Il comando ha il seguente formato:

```
PRINT #1f,"BLOCK-ALLOCATE:dr,t,s"
```

oppure:

```
PRINT #1f,"B-A";dr;t;s
```

Esempio:

```
10 OPEN15,8,15,"IO"
20 FOR I= 0 TO 10
30 PRINT#15,"B-A";0;35;I
40 NEXT
50 PRINT#15,"IO"
```

60 CLOSE 15

BLOCK-FREE

Questo comando e' l' opposto del precedente.

BUFFER-POINTER

Questo comando muove il puntatore dall' inizio del Buffer alla posizione di un carattere qualsiasi contenuto entro il Buffer stesso.
Il formato e':

PRINT#1f,"BUFFER-POINTER:sa,p".

PROGRAMMAZIONE DEL CONTROLLER DISCO

Il programmatore puo', in seguito ai comandi che vedremo scrivere delle routines che risiedano ed operino sul controller del disco. Vedi anche la parte di Approfondimento riportata al termine di questa Sezione.

BLOCK-EXECUTE

Questo comando permette che parte del DOS o

routines scritte dall' utente risiedano sul disco, siano caricate nella memoria del controller del disco e possano essere quindi eseguite.
E' lo stesso del BLOCK-READ tranne che i dati letti dal settore del disco devono essere il codice oggetto di un programma in Linguaggio Macchina.
B-E legge un programma che deve terminare con RTS, cioe' con un Ritorno da Subroutines.

Esempio:

```
100 PRINT#15,"B-E6,0,1,10
```

Legge un blocco dal drive 0, traccia 1,settore 10 nel Buffer del canale 6 ed esegue il suo contenuto a partire dalla posizione 0 del Buffer.

!!!ATTENZIONE!!!

Tutti i tre comandi MEMORY sono orientati al trattamento del singolo Byte in modo che l' utente possa utilizzare programmi in Linguaggio Macchina.
Usando i comandi MEMORY si puo' accedere per mezzo di istruzioni BASIC alle informazioni usando la funzione CHR$. Il sistema accetta solo codici abbreviati.

MEMORY-READ

Questo comando consente la lettura di 1 byte di dati dal dischetto.
Ricordando che ci sono disponibili sul drive ben 16k-Bytes di ROM e 2k-Bytes di RAM, si puo' quindi accedere direttamente a questi o ai buffers che il DOS ha fissato in RAM usando i comandi MEMORY.

Il formato di questo comando e':

```
PRINT#1f,"M-R",adl/adh
```

L' indirizzo del Byte da leggere deve essere specificato nella lista dei parametri che segue il comando usando le funzioni CHR$. Deve essere cioe':

adl/adh= CHR$(adl)CHR$(adh).

Il byte stesso e' quindi letto usando un comando GET# tramite il canale di controllo (15).

Esempio:

```
100 PRINT#15,"M-R"CHR$(8)CHR$(18)
110 GET#15,A$
```

In questo esempio viene letto un Byte di dati dal buffer di indirizzo 1808.

PROGRAMMA PER LEGGERE LA MEMORIA DEL CONTROLLER DEL DISCO.

```
10 OPEN15,8,15
20 INPUT"LOCAZIONE?";A
30 A1=INT(A/256):A2=A-A1x256
40 PRINT#15,"M-R:"CHR$(A2)CHR$(A1)
50 FORL=1TO5
60 GET#15,A$
70 PRINT ASC(A$+CHR$(0));
80 NEXT
90 INPUT"CONTINUA";A$
100 IF LEFT$(A$,1)="S" THEN50
110 GOTO 20
```

MEMORY-WRITE

Questo comando serve per scrivere dati entro il Buffer del disco. Si possono inserire fino ad un massimo di 34 Bytes per volta.
Il formato del comando e':

```
PRINT#1f,"M-W"adl/adh/nc/data
```

Supponendo di voler scrivere all' indirizzo $1800 i valori 32,0,17,96 e ricordandoci sempre che il Byte meno significativo dell' indirizzo deve precedere quello piu' significativo:

```
100
PRINT#15,"M-W"CHR$(00)CHR$(18)CHR$(4)CHR$(32)CHR$(0
)CHR$(17)CHR$(96).
```

MEMORY-EXECUTE

Serve per eseguire una subroutine in Linguaggio Macchina che deve essere presente o in ROM o in RAM.
Ha il seguente formato:

```
PRINT#1f,"M-E"adl/adh
```

dove adl e adh danno l'indirizzo dipartenza della routine nel buffer della memoria.
Ricordiao ancora una volta che una routines in L.M. deve terminare con una istruzione RTS

RTS = $60 = CHR$(96).

USER

Questo comando fornisce un collegamenteo con il codice macchina Commodore in accordo con una tabella di salto cui si accede con uno speciale puntatore. Il secondo carattere di questo comando e' usato quale indice in tale tabella.
Possono esssere usati i caratteri ASCII da 0 a 9.

CAPITOLO TREDICESIMO

FILES SEQUENZIALI

Un file di dati sequenziali puo' essere aperto sia per eseguire una lettura che una scrittura di dati, ma mai contemporaneamente per tutte e due le operazioni.
Quando viene aperto un nuovo file sequenziale di dati per la scrittura il procedimento stesso lo crea fisicamente.
Nel caso infatti si tenti di aprire un file esistente per la scrittura avremo come risultato anche se tutta l'operazione e' corretta un errore di tipo:

FILE EXIST

Di conseguenza un file esistente puo' essere solo aperto per leggerlo, mentre un file che non sia presente nella Directory puo' essere solo aperto per scriverlo.

SEPARATORI

Le variabili numeriche in un file sequenziale devono terminare con un carattere di ritorno carrello, mentre le stringhe possono anche terminare con una virgola.
E' tuttavia sempre raccomandabile usare il carattere di ritorno carrello (RETURN o il suo equivalente ASCII) per separare i vari campi di un

file dati sequenziale perche' l'uso della virgola, mentre non da nessun vantaggio particolare puo' creare dei problemi durante la stesura di un programma.

Se tutti i campi terminano con un ritorno carrello, allora le regole di scrittura di un file sequenziale sono molto semplici.
Si usa infatti il comando PRINT#con una lista di parametri come abbiamo visto, allo stesso modo che si adopera il PRINT quando si desideri visualizzare una lista di variabili sullo schermo in una singola colonna verticale.
Per leggere poi questi campi useremo i comandi INPUT# e GET#.
In appendice riportiamo alcuni programmi di files sequenziali, tuttavia e' intuibile che questo tipo di gestione non e' difficile almeno in fase di lettura e scrittura.
Si consiglia anche di rivedere quanto scritto sui Files Sequenziali su cassetta con tutti gli esempi riportati perche' la gestione e' molto simile.
Riteniamo importante invece dedicare alcune righe alla modifica di un file sequenziale di dati.

INSERIENTO DATI ENTRO UN FILE SEQUENZIALE.

Per aggiungere dati ad un file sequenziale esistente sara' necessario procedere in maniera tale da non incorrere negli errori detti in precedenza.

Supponiamo di avere un file di dati, che chiameremo DATA1 contenente un elenco di nomi, un elenco di programmi, un gruppo di indirizzi o comunque un elenco di dati che si desideri aggiornare.

Date le limitazioni viste in precedenza non si puo' chiaramente aggiungere dati perche' appunto su di un file esistente si puo' solo eseguire una operazione di lettura e non di scrittura.
Sara' quindi necessario creare un file DATA2 e procedere di conseguenza:

1 - Per prima cosa controllare che non esista un file data di nome DATA2 e se esiste cancellarlo.

2 - Eseguire un comando OPEN di lettura sul file DATA1.

3 - Eseguire un comando OPEN di scrittura sul file DATA2.

4 - Leggere sequenzialmente i record dal file DATA1 e riscriverli sempre sequenzialmente sul file DATA2.

5 - Trovare la fine del file DATA1 controllando lo ST che deve essere uguale a 64 ed a questo punto scrivere i nuovi records sul file DATA2.

6 - Chiudere DATA1

7 - Cancellare (SCRATCH) DATA1

8 - Eseguire il RENAME del DATA2, assegnandogli appunto il nome nuovo DATA1.

Naturalmente se i dati sono da aggiornare invece che semplicemente da aggiungere, sara' necessario eseguire il programma passo-passo trovando i campi da variare, con il sistema di rileggerli uno per uno, visualizzarli, correggerli e poi riscriverli

nel file DATA2.

R E L A T I V E S

Introduzione

Mano a mano che il numero di dati registrati in un file aumenta, aumenta anche il tempo medio richiesto per accedere a questi dati finche' si raggiunge un punto in cui il disco viene detto essere "DISK BOUND".
Il sistema operativo a disco non puo' manipolare cio' che gli si chiede con sufficiente velocita' a causa dell' eccessivo tempo d' attesa richiesto per trovare un file e leggere il suo contenuto. Questo e' un inconveniente tipico inerente all' uso su disco di files sequenziali che registrano e rileggono i dati nella stessa maniera in cui vengono registrati e letti i dati su una cassetta magnetica.
I dati vengono cioe' individuati eseguendo una ricerca attraverso ciascun file traccia per traccia e settore per settore finche' l' informazione desiderata non viene trovata. Ovviamente se un file possiede un nome identificabile che e' stato registrato nella Directory la ricerca di questo file e' molto semplificata.

Non e' raro tuttavia per dei file su disco contenere delle informazioni che non siano memorizzate sotto un nome di file e che quindi non siano facilmente reperibili mediante l' uso di una directory.
Quali esempi di questo tipo si possono citare i sistemi di prenotazioni di biglietti e vari tipi di files contenenti informazioni personali. In queste circostanze il vantaggio del drive a disco di una

maggiore velocita' di individuazione dell' informazione rispetto alla cassetta magnetica si attenua.
Una parziale causa di cio' risiede nell' architettura del Disk Operating System. Fortunatamente pero' esistono metodi piu' efficaci di ricerca e di lettura.

FILES RELATIVES

L' accesso diretto su files relatives e' un metodo che permette al programmatore di posizionarsi su qualsiasi record sul disco con riferimento all' inizio di un file.
I due componenti principali nell'organizazione dei files relatives sono il concatenamento dei blocchi SIDE SECTOR ed il concatenamento dei blocchi dati.
Ambedue sono collegati attraverso dei puntatori in modo simile a quello usato con i files sequenziali.
La lunghezza del record quando viene fissata dall' utente puo' essere compresa fra 1 e 254 bytes, mentre il numero di records e' limitato dalle capacita' del disco.
I SIDE SECTORS non contengono informazioni sul record, ma contengono la posizione dei blocchi di dati.

Quando un record viene richiamato, la posizione del puntatore viene stabilita dalla lunghezza del record stesso, poiche' tale lunghezza viene usata in un algoritmo per calcolare il valore che il puntatore deve assumere. Il numero di record cercato viene assegnato attraverso il comando RECORD. Il SIDE SECTOR contiene anche una tavola di puntatori a tutti gli altri SIDE SECTOR del file.

Per muoversi da un SIDE SECTOR ad un' altro il puntatore e' modificato attraverso un appropriato comando DOS e la traccia ed il settore del corrispondente SIDE SECTOR viene letto nella memoria.
In successione, usando le informazioni contenute nel SIDE SECTOR letto, i puntatori dei blocchi di dati possono essere posizionati ed usati per leggere il blocco di dati che contiene il record cercato.
I puntatori dei blocchi di dati del file relative contenuti nel SIDE SECTOR permettono al DOS di muoversi da un record ad un'altro attraverso due comandi di lettura del disco realizzando in tal modo un considerevole risparmio del tempo richiesto per individuare una data informazione.

L' espansione e' l' operazione chiave di un file relative che aiuta a ridurre le operazioni di lettura/scrittura richieste per individuare e leggere dati.
Prima di esaminare come questa operazione del DOS diminuisca il tempo di accesso e' necessario vedere come i canali di I/O sono utilizzati.

Quando viene aperto un canale su un file che gia' esisteva in precedenza, il DOS fara' in modo che ci si posizioni sul primo record in maniera che i parametri che verranno successivamente assegnati siano interpretati in modo corretto.
Non e' necessario assegnare la lunghezza del record variabile nell' istruzione OPEN se il file era gia' stato creato, tuttavia il DOS eseguira' sempre un controllo rispetto alla lunghezza del record che era stata data in origine durante la fase di creazione del file.
Se non sara' trovata concordanza allora verra' segnalato l'errore:

50 RECORD NOT PRESENT

L' uso di files relatives richiede 3 buffers di memoria mentre i files sequenziali ne richiedono solo 2.
Per questo non piu' di tre canali di comunicazione possono essere aperti contemporaneamente nell' uso dei files relatives.
Se un record si trova al confine fra due blocchi di dati, cioe' esso parte in un blocco di dati e termina in un' altro, allora il DOS leggera' il primo segmento e proseguira' la lettura nel secondo blocco di dati.
In pratica i records di parecchi files relatives possono essere suddivisi sui blocchi di dati. L' unica eccezzione si ha con una dimensione di record 1,2,127 e 254. Tali lunghezze infatti sono dei divisori esatti della dimensione 254 del blocco di dati e quindi la suddivisione su piu' blocchi non e' necessaria.

Questo metodo di suddivisione ha il vantaggio di non richiedere nessuna memoria ulteriore dal sistema al di fuori di quella richiesta per i blocchi di SIDE SECTOR nei files relatives.
Quando un record viene scritto attraverso l' istruzione PRINT # il blocco di dati non e' immediatamente scritto sul disco. Esso viene scritto solo quando il DOS si muove al di fuori del particolare blocco di dati in cui il record risiede.
Cio' puo' accadere sia attraverso delle uscite successive e sequenziali di record o posizionando il puntatore ad un' altro record che si trovi al di fuori di quel particolare blocco.

A causa della procedura di suddivisione si raccomanda che non siano aperti due canali su un

file relatives allo stesso tempo se entrambi i canali devono scrivere sullo stesso file.

!!!ATTENZIONE!!!

L' istruzione PRINT # lf,RECORD incrementa il puntatore del record per cui deve essere dato un unico PRINT#.

CREAZIONE DI UN FILE RELATIVE

NOTA

I seguenti esempi si applicano a quei sistemi che siano equipaggiati con unita' centrale ed unita' a dischi che consentono il diretto uso dei files relatives e pertanto i comandi :

DOPEN
DCLOSE
RECORD

E' possibile tuttavia gestire i relatives anche con il VIC20 e con il CBM 64 immettendo via software delle semplici routines che simulino i comandi suddetti non presenti in queste unita'.

Quando un file relative viene aperto per la prima volta tale file deve essere inizializzato dal programmatore.
Vediamo un semlpice esempio di programma e commentiamolo:

```
110 DOPEN#1,"ESEMPIO",DO,L50
120 GOSUB 190
130 RECORD# 1,100
140 GOSUB 190
150 PRINT#1,CHR$(255);
160 GOSUB 190
170 DCLOSE#1
180 END
190 IF DS 20 THEN RETURN
200 PRINT DS$
210 IF DS=50 THEN RETURN
220 STOP
```

In questo esempio viene creato alla linea 110 un file con il nome "ESEMPIO" ed una lunghezza di record di 50 bytes.

Le linee 120, 140, 160 rimandano alla subroutine che tratta la manipolazione degli errori che vedremo in fondo.

La linea 130 posiziona il puntatore di file al record 100 non ancora esistente.

La linea 150 scrive sul record 100. Durante questa operazione di scrittura il DOS individua che i record da 1 a 99 non esistono ed automaticamente li inizializza scrivendo un carattere chr$(255) nel primo byte.

La linea 170 chiude il file e fa si che lo spazio sia allocato nella BAM e nella Directory del file.

Le linee da 190 a 220 contengono la subroutine di manipolazione degli errori. Se DS 20 non vi e' alcuna condizione di errore ed il controllo viene

rimesso al programma principale. La linea 200 stampa il messaggio di errore e la linea 210 rinvia al programma principale se l' errore individuato era 50 RECORD NOT PRESENT.
Nel caso si verifichi un qualsiasi altro tipo di errore la linea 220 fa si che il programma termini ed il programmatore possa quindi correggerlo.

ESPANSIONE DI UN FILE RELATIVE

Dopo che un file e' stato inizializzato i dati possono esservi scritti. L' inizializzazione di un file fatta nella maniera decritta puo' essere eseguita dopo che il file sia stato originariamente creato.
Se il programmatore vuole espandere un file esistente si puo' adoperare la stessa procedura con un numero di record (nell' esempio seguente la linea 130) modificata in modo da essere il nuovo ultimo record.

L' esempio che segue con il disco che contiene il file ESEMPIO (con 100 record) nel drive 0 lascera' i primi 100 records immutati e creera' nuovi record fra la posizione 101 e 200 che conterranno solamente il carattere CHR$(255) nella prima posizione.

```
110 DOPEN#1,"ESEMPIO",D0,L50
120 GOSUB 190
130 RECORD#1,200
140 GOSUB 190
150 PRINT#1,CHR$(255);
160 GOSUB 190
```

```
170 DCLOSE#1
180 END
190 IF DS 20 THEN RETURN
200 PRINT DS$
210 IF DS=50 THEN RETURN
220 STOP
```

NOTA

Quando un programmatore desideri operare nei confronti di un file che gia' esiste non e' necessario specificare la lunghezza del record.
Nel caso che tale lungheza venga specificata essa deve corrispondere a quella lunghezza che era stata dischiarata al tempo della creazione del file perche' in caso contrario si avra' una segnalazione di errore.

Quando si espande un file nella maniera descritta, anche i SIDE SECTORS vengono automaticamente creati.
I SUDE SECTORS sono trasparenti per il programmatore perche' essi sono generati automaticamente ed esaminati dal DOS.

ACCESSO AD UN FILE RELATIVE

Allo scopo di rendere pratici i files relatives deve essere possibile al programmatore di accedere al file per la lettura o la scrittura dei dati.
Ambedue queste operazioni sono semplificate con i files relatives ed ambedue possono usare il comando RECORD per posizionarsi sul desiderato record prima

delle operazioni.

Per scrivere un dato in un determinato record di un file deve essere assegnata una costante per posizionarsi su tale record. Operare come segue:

```
110 DOPEN#1,"ESEMPIO",DO
120 GOSUB 190
130 RECORD#1,25
140 GOSUB 190
150 PRINT#1,"RECORD 25"
160 GOSUB 190
170 DCLOSE#1
180 END
190 IF DS 20 THEN RETURN
200 PRINT DS$
210 IF DS=50 THEN RETURN
220 STOP
```

Il programma che segue illustra in che maniera si possa accedere ad uno specifico byte del record:

```
110 DOPEN#1,"ESEMPIO",DO
120 GOSUB 900
130 RECORD#1,25,1
140 GOSUB 900
150 PRINT#1,"CAMPO 1"
160 GOSUB 900
170 RECORD#1,25,10
180 GOSUB 900
190 PRINT#1,"CAMPO 2"
200 GOSUB 900
210 RECORD#1,25,30
220 GOSUB 900
230 PRINT#1"CAMPO 3"
```

```
240 GOSUB 900
250 DCLOSE#1
260 END
900 IF DS 20 THEN RETURN
910 PRINT DS$
920 STOP
```

Le linee 130, 170 e 210 posizionano il puntatore di file sul record 25 in corrispondenza a diversi bytes.

NOTA

E' importante che i campi siano scritti in ordine poiche' la scrittura di un byte all' inizio di un record fa scorrere il resto del record nella memoria.
Cio' significa in sostanza che e' possibile scrivere sul byte 1 e successivamente sul byte 20 ma non e' possibile scrivere prima sul byte 20 e poi sul byte 1.
Poiche' il ritorno carrello e' riconosciuto come un carattere di separazione dal Basic i dati possono essere riletti nella seguente sequenza:

```
110 DOPEN#1,"ESEMPIO",DO
120 GOSUB 290
130 RECORD#1,25
140 GOSUB 290
150 RECORD#1,25,1:GOSUB 290
160 INPUT#1,A$:GOSUB 290
170 RECORD#1,25,10:GOSUB 290
180 INPUT#1,B$:GOSUB 290
190 RECORD#1,25,30:GOSUB 290
200 INPUT#1,C$:GOSUB 290
```

```
210 DCLOSE#1
220 END
290 IF DS 20 THEN RETURN
300 PRINT DS$
320 STOP
```

Le linee 160, 180 e 200 fanno si che i valori memorizzati sul disco siano letti e memorizzati rispettivamente nelle variabili A$, B$, C$.

E' molto comune dover accedere a un byte la cui posizione viene detrminata solo durante l' esecuzione.
La seguente routine illustra una procedura che richiede all' operatore il numero di record ed il dato da scrivere

```
100 PRINT"SCRIVI N. RECORD E DATA"
105 INPUT R,D$
110 DOPEN#1,"ESEMPIO",DO
120 GOSUB 190
130 RECORD#1,(R)
140 GOSUB 190
150 PRINT#1,D$
160 GOSUB 190
170 DCLOSE#1
180 END
190 IF DS 20 THEN RETURN
200 PRINT DS$
220 STOP
```

La linea 130 posiziona il puntatore di file al numero di record (R) specificato dal programmatore. Si noti che la variabile usata nel comando RECORD deve essere chiusa in parentesi.

La linea 150 fa si che i dati memorizzati in D$ siano trascritti su disco.

Notare inoltre che il comando RECORD puo' essere se il file deve essere esaminato in maniera sequenziale. In tal modo si risparmia del tempo durante l' esecuzione del programma. Ed un esempio di cio' si puo' avere quando si legge un gran numero di dati dal file su disco.
Si supponga che nel programma D$ sia stata dimensionata come una matrice di 100 elementi e che questi elementi siano stati scritti su disco nei record dal numero 1 al 100 del file "ESEMPIO".
Questa operazione puo' essere fatta con la seguente routine:

```
110 DOPEN#1,"ESEMPIO",DO
120 GOSUB 190
130 FOR I=1 TO 100
150 PRINT#1,D$(I)
160 GOSUB 190
165 NEXT I
170 DCLOSE#1
180 END
190 IF DS 20 THEN RETURN
200 PRINT DS$
220 STOP
```

Poiche' il puntatore di un record viene automaticamente posizionato sul record 1, quando il file viene aperto, il record 1 e' il primo record che viene scritto.
Se nessun comando RECORD viene eseguito il DOS posiziona automaticamente il puntatore sul record successivo dopo ciascun PRINT. Di conseguenza il

contenuto di D$ verra' scritto dal record 1 al 100
sul file "ESEMPIO".

CAPITOLO QUATTORDICESIMO

I MESSAGGI DI ERRORE DISCO

Riportiamo qui la serie di messaggi di errore disco, le possibili cause e gli eventuali rimedi.
Al termine di questo capitolo e' riportato un breve programma che aiuta ad individuare il tipo di errore che il DOS puo' aver generato in risposta ad un nostro messaggio o ad un comando diretto o indiretto, cioe' sotto forma di programma.
Per un' accurato esame degli errori è' necessario consultare le tecniche di registrazione e rilettura dati esposte nei capitoli seguenti.

Su due piccoli particolari ci permettiamo di insistere.
Il primo non farsi prendere dal panico quando si ha la generazione di un errore e, tanto meno, procedere al RESET di sistema se la luce rossa del LED frontale appena incomincia a lampeggiare. Spesso infatti e' solo la testina che si deve posizionare correttamente.
Secondo e' bene cercare di gestire SEMPRE anche da programma il controllo degli errori.

SOMMARIO DEI MESSAGGI DI ERRORE DEL DISK OPERATING SYSTEM

0 OK, nessun errore
1 Risposta per il file cancellato.Non e' un errore
2-19Non usati
20 Intestazione non registrata
21 Caratter di sincronizzazione non registrato.
22 Blocco di dati assente
23 Errore di checksum in un blocco di dati
24 Errore di decodifica di byte
25 Errore di verifica in scrittura
26 Si e' cercato di scrivere su un disco protetto.
27 Errore di checksum nella testata
28 Blocco dati troppo lungo
29 Identificatore del disco errato
30 Errore di sintassi generico
31 Comando errato
32 Linea lunga
33 Nome del file illegale
34 File non assegnato
39 File di comando non trovato
50 Record inesistente
51 Overflow sul record
52 File troppo grande
60 File aperto per scrittura
61 File non aperto
62 File inesistente
63 File esistente
64 File inesatto
65 Blocco non disponibile
66 Traccia o settore illegali
67 Come sopra ma di sistema
70 Nessun canale disponibile
71 Errore nella Directory
72 Disco o directory piene
73 Messagio relativo all' alimentazione

74 Unita' non pronta

DESCRIZIONE DEI MESSAGGI DI ERRORE DOS

I messaggi di errore con numero inferiore a 20 dovrebbero essere ignorati con l' eccezione del messaggio 01 che fornisce informazioni sul numero di files cancellati con il comando SCRATCH.

20 READ ERROR

Non e' stata trovata l' intestazione del blocco.Il controller del disco non e' in grado di individuare l' intestazione del richiesto blocco di dati.
Questo errore puo' essere causato da un numero di settore illegale oppure l' intestazione e' andata distrutta.

21 READ ERROR

Mancanza di carattere di sincronismo.
Il controller del disco non riesce ad individuare il carattere di sincronismo nella traccia desiderata.
Puo' essere dovuto ad un disallineamento della testina, oppure non e' stato inserito il dischetto o non e' stato formattato o e' comunque sciupato.
Spesso l' errore indica anche un guasto Hardware dell' unita' a disco.

22 READ ERROR

Blocco di dati assente

E' stato richiesto al controller di leggere un blocco di dati che non e' stato in precedenza scritto in modo appropriato.
Questo errore avviene in corrispondenza di comandi BLOCK ed indica una richiesta illegale di traccia o settore.

23 READ ERROR

Errore di CHECKSUM in un blocco di dati. Indica la presenza di un errore in uno o piu' Bytes del blocco di dati.
Il blocco e' stato letto nella memoria del Sistema Operativo del disco (DOS), ma al controllo la somma di prova e' risultata errata.
Questo errore puo' esere dovuto anche ad una messa a terra errata dell' unita'.

24 READ ERROR

Errore di decodifica di byte.
I dati o l' intestazione sono stati letti nella memoria DOS, ma e' stato trovato un errore HARDWARE a causa di un errata configurazione di bit nel byte di dati.
Anche questo errore puo' derivare, come il precedente da un errata messa a terra.

25 WRITE ERROR

Errore di verifica di scrittura.
Questo messaggio viene generato se il controller individua una mancata corrispondenza fra quanto scritto e quanto contenuto nella memoria del DOS.

26 WRITE PROTECT ON

Scrittura su disco protetto.
Si ottiene questo messaggio tentando di scrivere su un dischetto con la protezione di scrittura attivata.
Cioe' quando la finestrella sulla destra del dischetto e' coperta.

27 READ ERROR

Errore di controllo somma (CHEKSUM) nella testata.
Il controller ha trovato un errore nella testata del blocco di dati richiesto.
Il blocco non e' stato correttamente letto nella memoria DOS.
Anche questo tipo di errore puo' indicare problemi HARDWARE derivanti da una errata messa a terra.

28 WRITE ERROR

Blocco di dati troppo lungo.
Il controller, dopo aver scritto un blocco di dati, cerca di individuare il carattere di sincronismo della intestazione del blocco successivo.
Se tale carattere non viene individuato entro un certo periodo di tempo e' generato il messaggio di errore.
L' errore puo' essere causato da una cattiva formattazione del disco, da un guizzo di tensione durante una fase di registrazione o da altro guasto HARDWARE.
I dati si estendono al blocco successivo che pertanto e' privo del carattere di sincronismo.

29 DISK IS MISMATCH

Errore frequente quando si tenti di usare un dischetto che non e' stato inizializzato.
Questo messaggio appare anche quando il dischetto ha una testata errata.

30 SYNTAX ERROR

Errore di tipo generale.
Il DOS non riesce ad interpretare l' ordine inviato sul canale di comando.
Generalmente questo errore e' provocato da un numero illegale di file,da nomi o configurazioni non ammessi.
Un esempio puo' essere nell' usare due nomi di files contemporaneamente con il comando COPY.

31 SYNTAX ERROR

Comando errato
Il DOS non riconosce il comando.
Si ricordi che il comando deve iniziare nella prima posizione della linea.

32 SYNTAX ERROR

Linea troppo lunga.
Il comando inviato supera la lunghezza massima di 58 caratteri.

33 SYNTAX ERROR

Nome di file non legale.

In un comando di LOAD, SAVE o VERIFY sono stati usati dei parametri illegali.

34 SYNTAX ERROR

File non assegnato.
E' stato omesso il nome del file oppure il DOS non lo riconosce.
Spesso nel comando mancano gli apici o i due punti.

39 SYNTAX ERROR

Comando non valido.
Questo errore puo' verificarsi se l' ordine inviato al canale di comando (indirizzo secondario 15) non e' interpretabile dal DOS.

50 RECORD NOT PRESENT

Record inesistente.
Questo messaggio viene generato quando si tenti una lettura al di la' dell' ultimo record tramite una operazione di INPUT#o di GET#.
Questo messaggio puo' anche aver luogo dopo il posizionamento ad un record al di la' della fine di un file quando si usino i files relatives.
Se l' intenzione era quella di espandere il file aggiungendo un nuovo record (con un comando PRINT#), il messaggio di errore puo' essere ignorato.

Le operazioni di INPUT o di GET non devono essere eseguite dopo che questo errore e' stato scoperto, se prima non si sia provveduto ad un riposizionamento.

51 OVERFLOW IN RECORD

Se l' istruzione PRINT#va oltre la dimensione del record l' informazione viene troncata.
Poiche' il ritorno carrello che serve come carattere terminatore del record deve essere contabilizzato nella dimensione del record stesso, questo messaggio di errore puo' aver luogo se il numero totale di caratteri nel record (includendo quindi anche il ritorno carrello finale) eccede la lunghezza precedentemente assegnata.

52 FILE TOO LARGE

File troppo grande.
Quando si presenta questo messaggio nel posizionamento su un record di un file relative cio' significa che si e' avuto un OVERFLOW su disco.

60 WRITE FILE OPEN

Si ha questo messaggio quando un file che era stato aperto per scriverci non e' stato richiuso e si tenta di leggerlo.

61 FILE NOT OPEN

Si ha questo messaggio quando si tenti di accedere ad un file in lettura o in scrittura e lo stesso file per il DOS non risulti aperto.
Non sempre avremo pero' questa segnalazione di errore. In qualche caso il messaggio non viene generato ed il comando di accesso e' semplicemente ignorato, e cio' e bene ricordarlo in fase di

scrittura delle eventuali routines di controllo dei programmi di accesso ai dischi.

62 FILE NOT FOUND

Il file non e' stato trovato.
Il file richiesto non e' stato trovato nel drive indicato o perche' non esiste o perche' e' distrutto.

63 FILE EXISTS

Avverte che sul dischetto esiste gia' un file con lo stesso nome.

64 FILE TYPE MISMATCH

Il tipo di file specificato nella richiesta non coincide con quello presente nella Directory.

65 NO BLOCK

Tale messaggio si puo' presentare nell' uso di un comando di BLOCK-ALLOCATE (abbreviato B-A).
Sta ad indicare che il blocco da allocare e' gia' stato allocato in precedenza.
I parametri stanno ad indicare traccia e settore del blocco disponibili piu' in alto, cioe' con un numero maggiore.
Se i parametri riportati sono 0 cio' significa che tutti i blocchi di numero piu' alto sono stati utilizzati.

66 ILLEGAL TRACK AND SECTOR

Sta ad indicare che il Sistema Operativo del Disco ha tentato di accedere ad una traccia o ad un settore che non esistono nel formato utilizzato.
Puo' anche rivelare dei problemi di lettura del puntatore al blocco successivo.

67 ILLEGAL SYSTEM T(track) OR S(sector)

Questo speciale messaggio di errore indica una traccia o un settore del sistema non legali.

70 NO CHANNEL(available)

Nessun canale richiesto e' disponibile.
Quindi o il canale richiesto non e' disponibile o tutti i canali sono gia' impegnati.
Come abbiamo visto il DOS consente di aprire solo un numero di canali limitato anche dal tipo di accesso.
Ricordiamo che possono essere aperti contemporaneamente al massimo 5 files sequenziali o 6 files ad accesso diretto.

71 DIR ERROR

Errore nella Directory.
La BAM (BLOCK AVAILABILITY MAP) non puo' essere accoppiata con il contatore interno.
L' errore puo' essere generato per problemi di allocazione della BAM stessa o quando si sia sovrascritto sulla BAM nella memoria del DOS.
Per correggere questo problema e' necessario reinizializzare il dischetto per riposizionare

giustamente la BAM nella memoria DOS.
Alcuni files aperti devono essere chiusi con la fase di correzione.

72 DISK FULL

Il dischetto e' stato completamente riempito oppure , anche con spazio ancora su disco, si e' pero' utilizzta l' intera capacita' del Directory di memorizzare i nomi di 144 files.

73 DOS MISMATCH

La versione del DOS dell' unita' a dischi 1540 e 1541 e' la CBM DOS 2.6.
Le versioni DOS 2.6 e DOS 1.0 (cioe' quelle delle unita' CBM 2040 e 3040) sono compatibili in lettura ma non in scrittura.
I dischi cioe' possono essere letti con intercambiabilita' con i due DOS, ma un disco formattato con una delle due versioni non puo' essere scritto con l' altra perche' la disposizione varia.
Questo errore viene segnalato quando si tenti di scrivere su un disco che e' stato formattato in un' altra versione.
Purtroppo pero' i DOS 2.6 e 2.5 (cioe' quelli presenti sulle unita' Commodore 8050 e 8250) non sono compatibili neppure in lettura.
Ricordiamo che questo messaggio puo' apparire anche dopo l' accensione.

74 DRIVE NOT READY

Unita' non pronta nel senso che all' interno manca il dischetto.

ERRORI SUI FILES

Ci sono delle condizioni che possono causare errori quando si aprono files di dati.

1-Si avra' una segnalazione di :

FILE NOT FOUND

se si apre un nuovo file di dati per operazione di lettura senza aver chiuso il vecchio.

2-Se si tenta di aprire un file esistente indicandone pero' la natura in modo errato avremo una segnalazione di :

FILE TYPE MISMATCH

Questo errore capita in modo particolare quando si tenti una operazione di OPEN su un file programma, ma identificandolo come file dati sia sequenziale che random.

3-Non si puo' aprire un file sequenziale esistente per scriverci.
Se si tenta una operazione del genere avremo un errore di :

FILE EXIST

Si puo' infatti solo scrivere entro un NUOVO file

di dati.

Sbagliando un nome di file in un comando OPEN si incorre in un tipo di errore che puo' causare un mucchio di guai perche' non sempre si ha una segnalazione di errore.
Infatti il DOS considera che il nome errato del file sia a tutti gli effetti un nuovo file con le conseguenze immaginabili in particolare se si verifica su un programma che sta girando e non su una semplice prova.

Si ha errore su disco quando nell' unita' 1540 o 1541 incomincia a lampeggiare il LED rosso che nella sua normale funzione e' acceso ma con luce fissa.
Nessuna operazione su disco dovrebbe essere gestita prima di avere riportato l' indicatore LED nella sua normale posizione a luce fissa.

Per riportare al normale funzionamenteo l' unita'a dischi e' necessario fermare l' esecuzione del programma con il tasto di RUN/STOP e poi leggere lo stato di errore del dischetto.

Riteniamo che sarebbe una buona idea quella di leggere lo STATUS dopo ogni operazione su disco, in particolare dopo le operazioni di scrittura.
Per far cio' rimandiamo al breve programma mostrato qui sotto che dovrebbe essere inserito come subroutine di ogni programma di manipolazione dati su disco.

Per esaminare lo stato di errore, si deve aprire un file logico specificando l' unita' fisica 8 con un

indirizzo secondario 15.
Si deve eseguire l' input delle 4 variabili stringa e visualizzarle.
Il programma e' il seguente:

```
10 OPEN1,8,15
20 INPUT#1,A$,B$,C$,D$
30 PRINT A$,B$,C$,D$
40CLOSE 1
```

Questo programma dovrebbe essere inserito come subroutine in procedure piu' grosse.
Il controllo di errore evita infatti spiacevoli sorprese in fase di rilettura anche se rallenta l' esecuzione.

Il comando INPUT#1 non puo' essere eseguito in modo immediato. Le stringhe da visualizzare indicheranno:

A$ = numero del messaggio di errore

B$ = messaggio d' errore

C$ = numero della traccia

D$ = numero del settore

ATTENZIONE!!

I nomi di queste variabili sono messi arbitrariamente per cui usando anche altri nomi il risultato non cambia.
Questa precisazione e' necessaria perche' utilizzando questo programma come subroutine di un

piu' vasto programma (MAIN-PROGRAM), puo' darsi che queste variabili siano gia' state utilizzate.

E' importante ricordare che nelle serie dotate di BASIC 4.0 il messaggio di errore completo, cioe' come quello riportato appena sopra puo' aversi con la richiesta del valore di DS$.

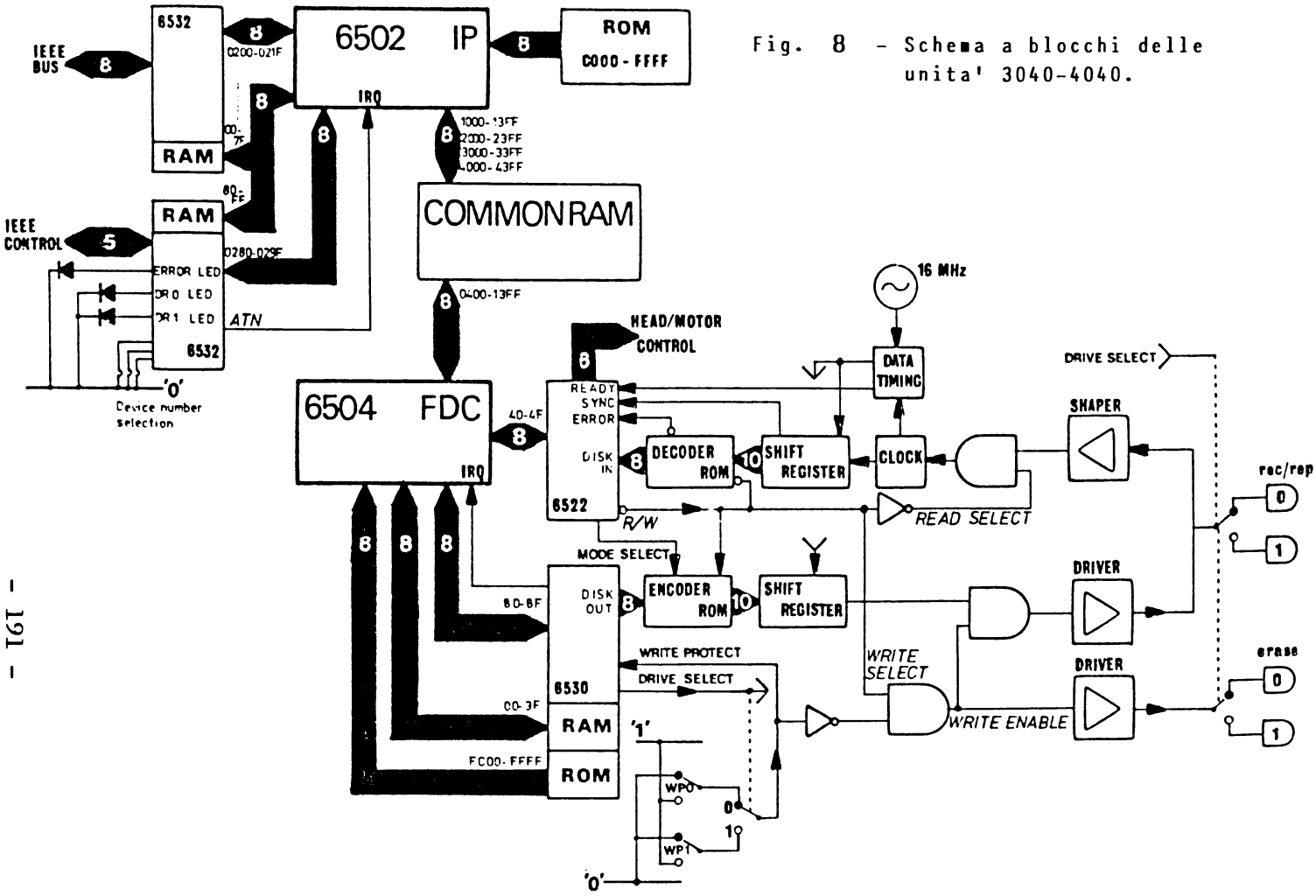


Fig. 8 - Schema a blocchi delle unita' 3040-4040.

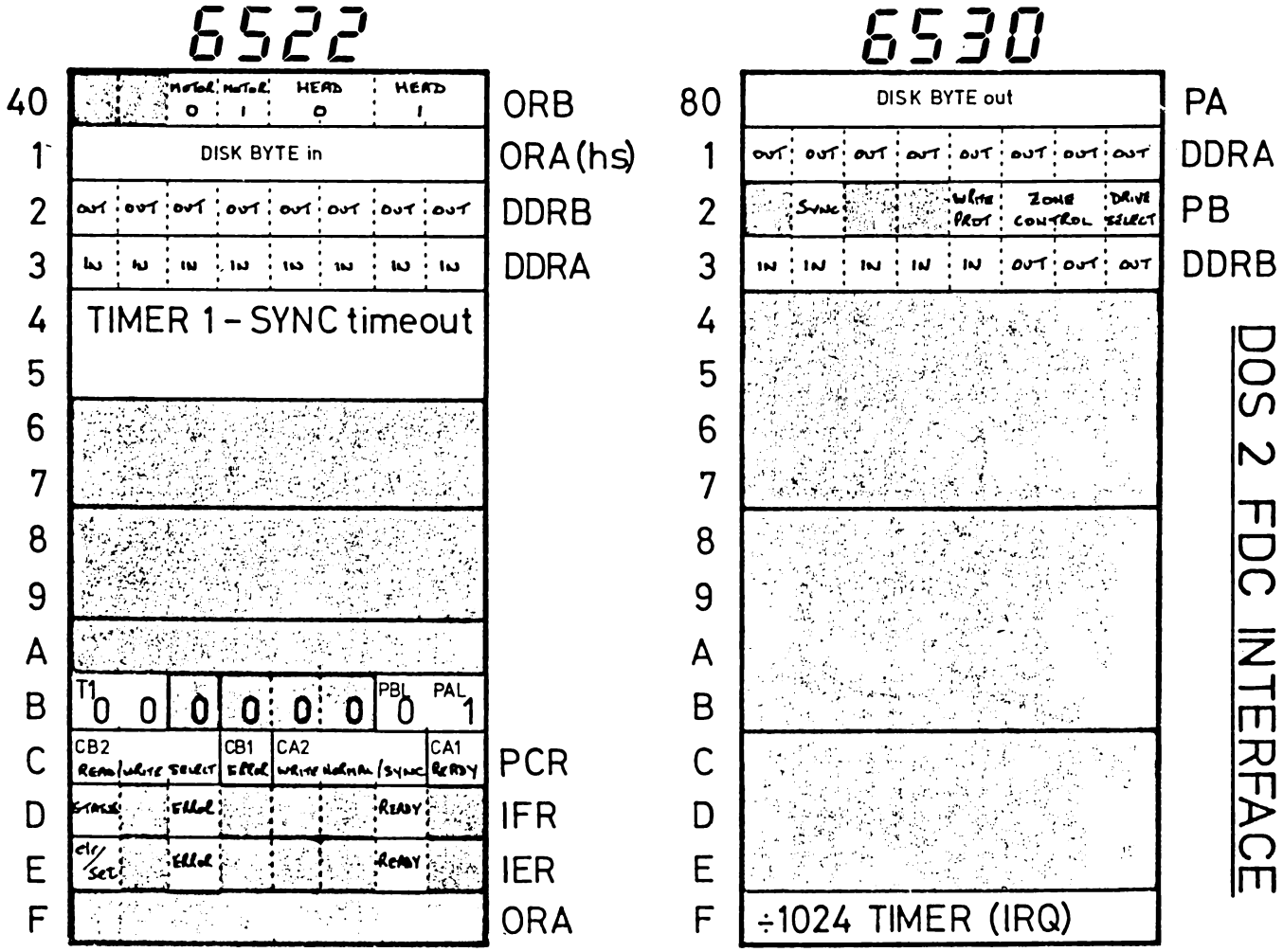


Fig. 9 - Registri interni del FLOPPY DISK CONTROLLER.

CAPITOLO QUINDICESIMO

A partire da questo capitolo iniziano una serie di approfondimenti sia sulla parte HARDWARE che SOFTWARE dei dischi.
Come ripetermo piu' volte e' necessario sottolineare che cercheremo di insistere sugli aspetti schematici del funzionamento e non su un tipo di disco in particolare anche grazie all' evoluzione continua del mercato che obbliga la COMMODORE, come del resto tutti gli altri costruttori a continue modifiche.

Nelle pagine precedenti e' riportato uno schema di diagramma per il funzionamento concettuale del 4040.
Questo schema puo' essere assimilato al 3040 ed alle serie 8000 mentre per il 1540/1541 va preso con molta attenzione dato il diverso funzionamento.

Da questo diagramma si puo' vedere come l' unita' a dischi impieghi ben due processori oltre ad un certo numero di altri integrati che vedremo nel corso del capitolo.
L' integrato 6502 e' l' INTERFACE PROCESSOR o IP e manipola tutte le comunicazioni tramite la porta IEEE-488.
Questo integrato ha inoltre il pieno controllo su tutta la movimentazione dei caratteri sia in fase di RICEZIONE che di TRASMISSIONE.
Esegue anche l' organizzazione del disco potremmo dire A POSTERIORI come la MANIPOLAZIONE DEI FILES, L' ALLOCAZIONE DEI BLOCCHI, ecc.
L' altro integrato, il processore 6504, e' il FLOPPY DISK CONTROLLER o FDC ed ha essenzialmente il controllo di tutte le operazioni di lettura e

scrittura in funzione dei comandi che gli vengono inviati dall' INTERFACE PROCESSOR.
L' IP utilizza poi due integrati periferici (2 chips 6532) ognuno dei quali e' anche provvisto di 128 Bytes di memoria RAM oltre a 2 porte di I/O ed un temporizzatore (o TIMER) programmabile.
Il solo INTERRUPT (che spiegato in termini molto semplici altro non e' che una interruzione temporanea in un funzionamento o nell' invio di dati) utilizzato sull' IP viene da un integrato 6530 ed interrompe appunto le operazioni del 6502 IP tutte le volte che viene fissata una linea di ATN (cioe' di ATTENTION) sulla IEEE-488.
A questo punto entra infatti in funzione la routine di HANDSHAKE della IEEE-488 che permette di ricevere dati dal BUS.
Il DOS 2 utilizza 12 K di ROM (il DOS 1 solo 8 K) oltre a 4 K di RAM divisi in 256 blocchi.
Una parte di questo spazio e' utilizzato come WORKSPACE o spazio di lavoro, mentre 14 di questi blocchi, ognuno dei quali di 256 bytes (15 blocchi nel DOS 1) sono utilizzati come BUFFERS per passare i dati DA e AL FDC.
Il CONTROLLER utilizza solo 64 Bytes di memoria RAM presenti nel 6530.
Questi, cioe' l' integrato 6530, ha 64 Bytes di RAM, 1 K di ROM, 2 Buffers di Input/Output ed un TIMER.
Nella piastra e' presente anche un 6522 con due porte di I/O e 2 TIMERS.
Il FLOPPY DISK CONTROLLER utilizza solo l' interrupt che proviene dal 6530 e che e' utilizzato per lo HEAD STEPPING, cioe' la temporizzazione e la guida del motore del drive stesso.
Per ragioni di economia l' elettronica e' posta fra i due drives e la linea di DRIVE SELECT cioe' di scelta fra quale unita' del disco adoperare e' utilizzata appunto per determinare quale dei due e'

il drive selezionato o in funzione o comunque aperto in quel momento.

Per scrivere qualcosa su disco il FDC per prima cosa controlla la linea di protezione scrittura o WRITE LINE PROTECT collegata a due microinterruttori il WP0 e il WP1 che inibisce l' invio di impulsi elettronici.
La linea di R/W (Read/Write) e' fissata all' operazione scrittura (WRITE) che inibisce la lettura dei dati, con il READ SELECT e disabilita la scrittura con il WRITE SELECT.
Un segnale di WRITE ENABLE, cioe' di disabilitazione scrittura e' quindi generato e disattiva la porzione cancellata della testata di lettura scrittura.
Il FDC attende il momento giusto, o meglio la giusta temporizzazione, utilizzando la READY LINE, e quindi immette i dati nella porta di DISK OUT.
Questi dati per prima cosa passano attraverso una ROM di codifica dove sono tradotti nella forma a 10-bits e da qui proposti allo SHIFT REGISTER.
Il risultato nella forma di 10-bits seriali passano attraverso il DRIVER e quindi al disco stesso.

Ritornando indietro, i dati DA disco prendono forma e vengono presentati alla circuiteria di clock dove viene estratta la temporizzazione per i dati in ingresso.

I dati sono temporizzati (funzione che qui si intende eseguita dal CLOCK) nel registro di shift e presentati alla ROM di decodifica che fa arrivare 8 BITS alla porta DATA IN del 6522.
Se la ROM di decodifica non riesce a riconoscere il Byte, la linea di errore e' fissata per indicare un "BYTE DECODING ERROR# 24".

Il FDC utilizza la linea di READY per determinare quando un Byte di dati valido e' disponibile sulla porta di DATA IN e solo a quel momento esegue la lettura di quel dato.

Uno speciale segnale di sincronizzazione o SYNC SIGNAL proviene dallo SHIFT REGISTER per controllare l' inizio di un blocco su disco. Questo e' ottenuto scrivendo una serie di 10 bits tutti a 1.
La procedura di decodifica assicura che i bytes di dati non siano mai tutta di 1 in modo tale che il CONTROLLER non si possa sbagliare con il Byte di sincronizzazione.
La ROM che serve alla codifica ed alla decodifica e' la stessa e per farla funzionare in lettura o scrittura e' necessario solo selezionare la linea R/W per l' uso che se ne vuol fare.

Le comunicazioni fra i due processori sono consentite attraverso la RAM che e' comune ad entrambi
I segnali dell' INTERFACE PROCESSOR al FDC per la scrittura o la lettura di un dato blocco sono immessi in una coda di lavoro o JOB QUEUE con un Byte di controllo lavoro o JOOB CONTROL BYTE.
Il FDC esegue quindi le operazioni di scrittura o lettura utilizzando i valori presenti in un dato buffer.
Quando ha completato il lavoro, restituisce un codice per indicare che l' operazione e' stata correttamente eseguita oppure, in caso negativo, restituisce un codice d' errore.
E' importante pero' ricordare che i blocchi di memoria RAM comune sono viste dai due processori con indirizzi distinti. Vediamoli nella seguente tabella che riporta anche le funzioni relative al DOS 1.2 e 2.1.

INDIRIZ.	INDIRIZ.	FUNZIONI	
IP	FDC	DOS1	DOS2
$1000	$0400	WORKSPACE	WORKSPACE
$1100	$0500	BUFFER 0	BUFFER 0
$1200	$0600	BUFFER 1	BUFFER 1
$1300	$0700	BUFFER 2	BUFFER 2
$2000	$0800	BUFFER 3	BUFFER 3
$2100	$0900	BUFFER 4	BUFFER 4
$2200	$0A00	BUFFER 5	BUFFER 5
$2300	$0B00	BUFFER 6	BUFFER 6
$3000	$0C00	BUFFER 7	BUFFER 7
$3100	$0D00	BUFFER 8	BUFFER 8
$3200	$0E00	BUFFER 9	BUFFER 9
$3300	$0F00	BUFFER 10	BUFFER 10
$4000	$1000	BUFFER 11	BUFFER 11
$4100	$1100	BUFFER 12	BUFFER 12
$4200	$1200	BUFFER 13	BUFFER 13
$4300	$1300	BUFFER 14	WORKSPACE

NOTA
Le BAM (0) e (1) sono:

BAM	DOS1	DOS2
0	BUFFER 13	BUFFER 12
1	BUFFER 14	BUFFER 13

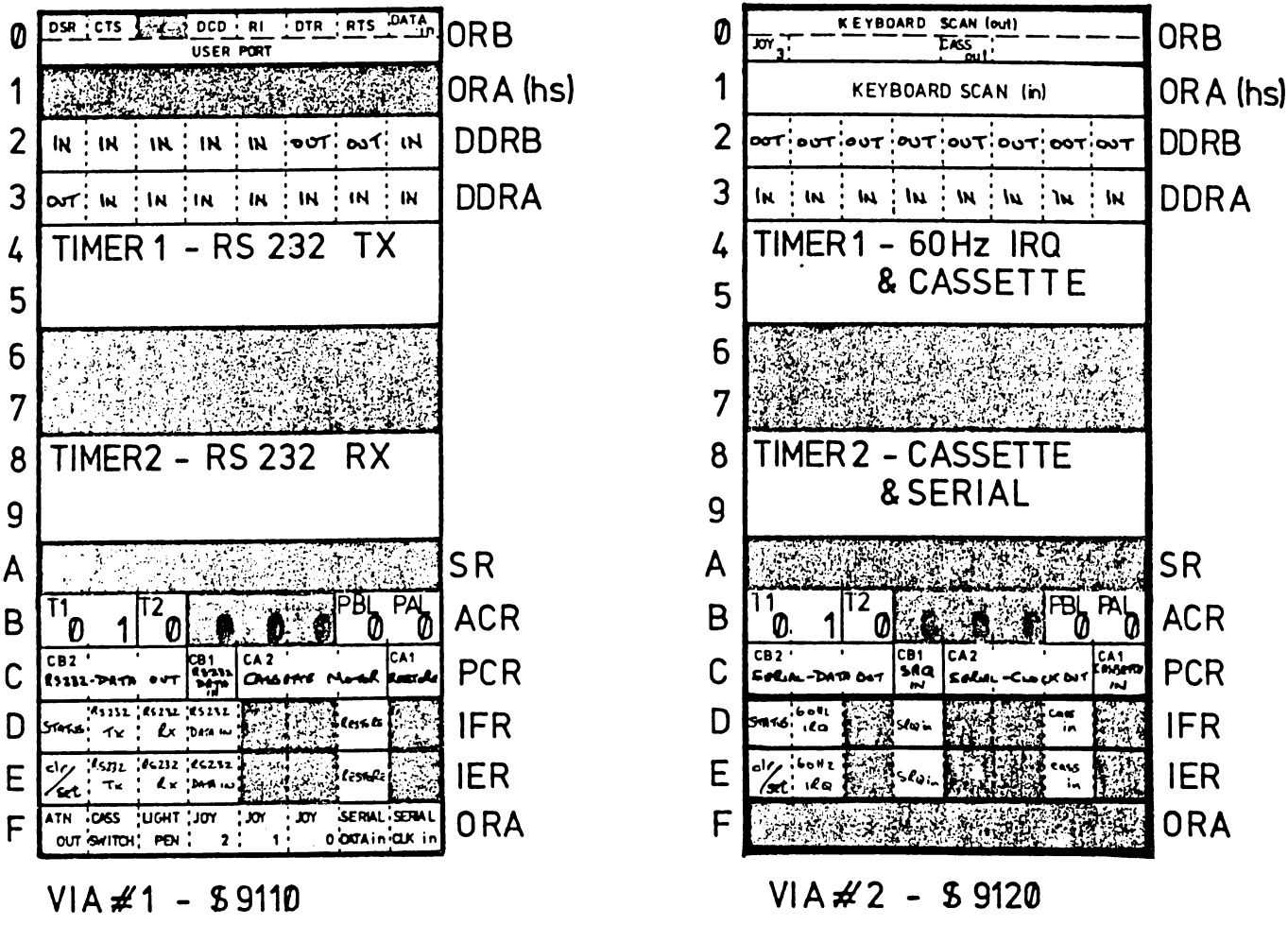


Fig. 10 - Registri interni degli integrati via 6522 usati dal VIC-20.

CAPITOLO SEDICESIMO

I FILES E L' HARDWARE

Vediamo di illustrare i principi base con i quali vengono manipolati i files da un punto di vista HARDWARE.

Come abbiamo visto nella spiegazione di files data nella prima parte, la memorizzazione su disco assomiglia molto a quella su cassetta.
Ci rivolgiamo infatti sempre ad una superfice magnetica sia per registrare che per rileggere e l' unica differenza sembra essere quella che il disco non debba essere scorso tutto per rileggere qualcosa come il nastro, ma dato che ha una testina di lettura scrittura indirizzabile, la ricerca stessa possa essere fatta in maniera piu' veloce.

Tutto questo e' vero fino ad un certo punto, ma sul disco ci attendiamo comunque di memorizzare molte piu' informazioni, con piu' sicurezza e quindi di utilizzarne la maggiore velocita' piu' che per la registrazione per la ricerca.
Il normale metodo di registrazione su cassetta e' quello di codificare i dati come una sequenza di zeri e uni in diverse frequenze di toni.
Tutto questo richiede un grande impiego di tempo e di spazio perche' sono necessari diversi cicli di ciascun tono per determinarne la frequenza e decidere se quello che e' stato codificato e' un 1 o uno 0.
E' vero che la tecnica COMMODORE riduce di molto la codifica di questo metodo perche' la registrazione

avviene in due soli cicli di tono.
Quando si opera su disco queste tecniche non sono efficienti.

Quando si registrano dati il fattore limitativo e' generalmente la densita' del flusso magnetico di passaggio che il mezzo puo' registrare con affidabilita'.
Cio' perche' il numero di volte che il campo magnetico registra sulla superfice magnetica puo' cambiare in un dato spazio.
La codifica dei tono richiede molti passaggi per bit. La tecnica COMMODORE ne richiede 4 per ogni bit.
Alcune tecniche disco ne utilizzano 2 per bit, ma le piu' moderne, e la COMMODORE fra queste, ne usano solo una per bit per incrementare ulteriormente la densita' di dati.

I dischi a 5.25" ruotano alla velocita' di 300 rpm e possono registrare dati con una densiata' di circa 400 bits per pollice.
Sia i 3040 che i 4040 come i 1540 e 1541 registrano piu' o meno con questa densita' mentre la densita' dell' 8050 e' leggermente piu' alta.
L' unita' a dischi 8050/8250 ottengono gran parte delle loro maggiori capacita' di memorizzazione dal fatto di utilizzare in DOPPIO le tracce.
Questo e' ottenuto attraverso l' uso di un piu' accurato sistema di posizionamento della testina di lettura e di meccanismi,cioe' di parti meccaniche migliori.

Vediamo ora di osservare piu' accuratamente il percorso di un byte di 8-bits (ed in questo caso dobbiamo precisare la particolarita' del Byte che come abbiamo visto e come vedremo puo' essere utilizzato in maniera diversa dal normale) dopo che

questi ha lasciato il microprocessore 6502 FDC.

Come abbiamo detto in precedenza e come possiamo vedere dal diagrammma presentato il byte di 8 Bits e' per prima cosa codificato in 10 bits e presentato all' input parallelo dello shift register.
Viene quindi eseguito un procedimento di trattamento con il clock ad una frequenza di circa 250k-bits per secondo.
Il dato che ne risulta e' usato per ingabbiare il clock stesso e produce come risultato una serie di dati in cui tutti gli "1" siano codificati dalla presenza di un impulso di clock e gli "0" dall' assenza di un impulso di clock come e' mostrato nell' associato diagramma di temporizzazione.
Questi, cioe' il dato stesso, potrebbe essere immagazzinato su disco come si trova, ma e' invece piu' grande della densita' di registrazione poiche' ci sono due flussi di passaggio per ogni 1 registrato.
In questo caso il flusso di dati e' inviato ad un FLIP-FLOP che taglia ogni impulso presente nel dato stesso e produce un singolo flussso di passaggio per ogni 1 nel dato codificato.
E' questo il segnale che e' fornito alla circuiteria elettronica che in quel momento guida la testina di lettura.

I Bytes non sono registrati uno alla volta ma in gruppi di 256.
Per consentire al FDC di presentare il Byte allo shift-register esattamente ed al momento giusto viene generato uno speciale impulso di temporizzazione (sulla liea READY) al termine di ogni 10 impulsi di clock.
Questo impulso e' disponibile per l' FDC ed e' anche utilizzato per caricare il Byte entro il

registro di SHIFT.

Il nuovo Byte deve essere presentato immediatamente al canale di codifica perche' i dati continuano ad essere trattati con il clock del registro di shift ed ogni interruzione fra i Bytes causerebbe problemi durante la fase di lettura.

La rilettura dei dati registrati su disco e' un po' piu' difficile.

Nelle pagine seguenti e' riportato uno schema di funzionamento per i cicli di lettura e scrittura che abbiamo cercato di evidenziare nella maniera piu' chiara possibile.

Ripetiamo ancora una volta che ci riferiamo in generale alle unita' a dischi COMMODORE ma il principio di funzionamento, anche se in alcuni casi possono variare gli integrati, e' identico.

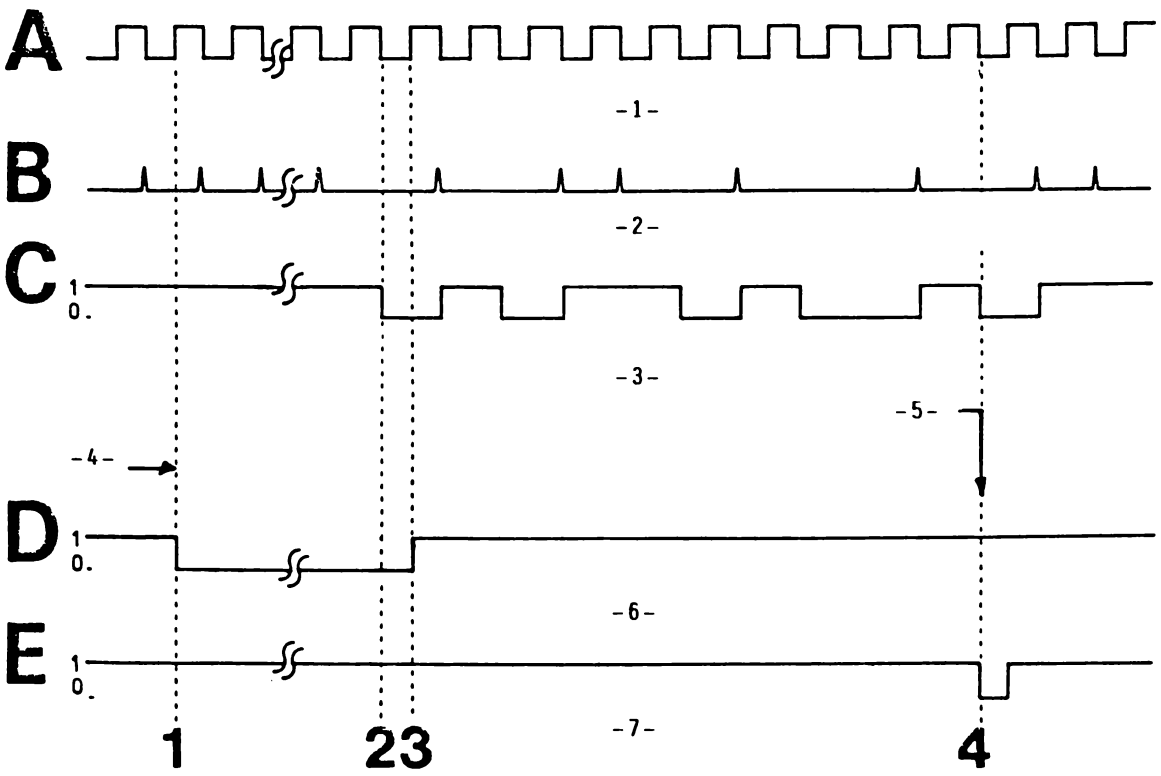


Fig. 11 - Ciclo di lettura.

- CICLO DI LETTURA -

-1- CLOCK.

-2- Arrivo dei dati da disco.

-3- Dati separati dai CLOCK di sincronismo e posti nel registro di SHIFT.

-4- Recezione di 10 impulsi consecutivi.

-5- I dati provenienti dalla ROM di decodifica sono immagazzinati nel 6522 nel fronte di discesa dell'impulso di READY.

-6- Segnale di sincronizzazione(SYNC)- Viene posto a livello basso,quando tutte le uscite del registro di SHIFT sono a "1".

Fig. 11 a.

Diagram 2 - READ

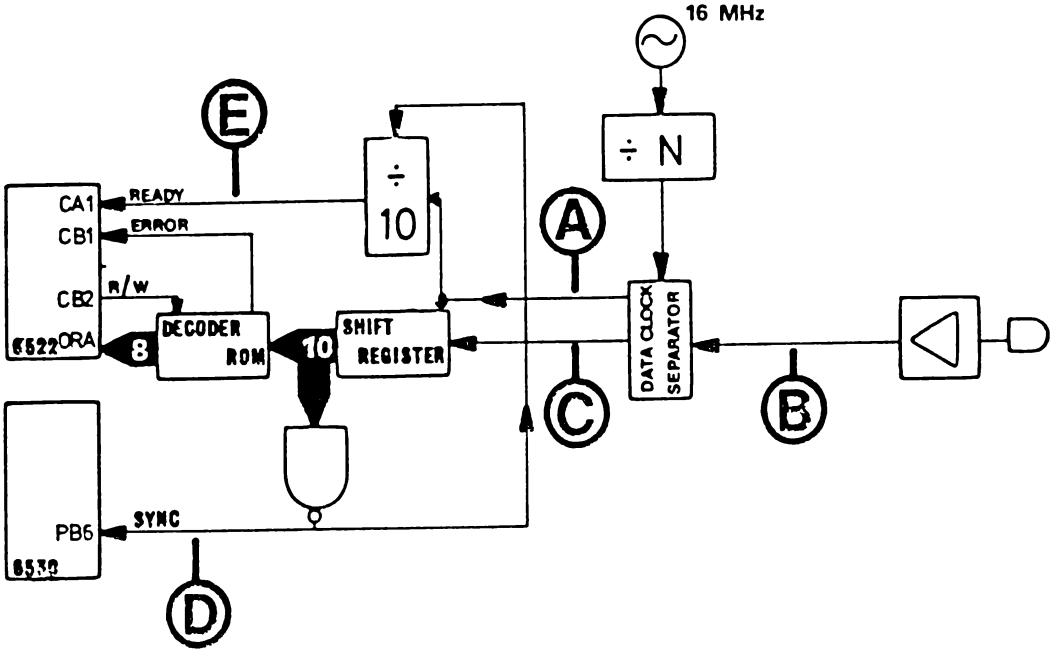


Fig. 11 b - Ciclo di lettura.

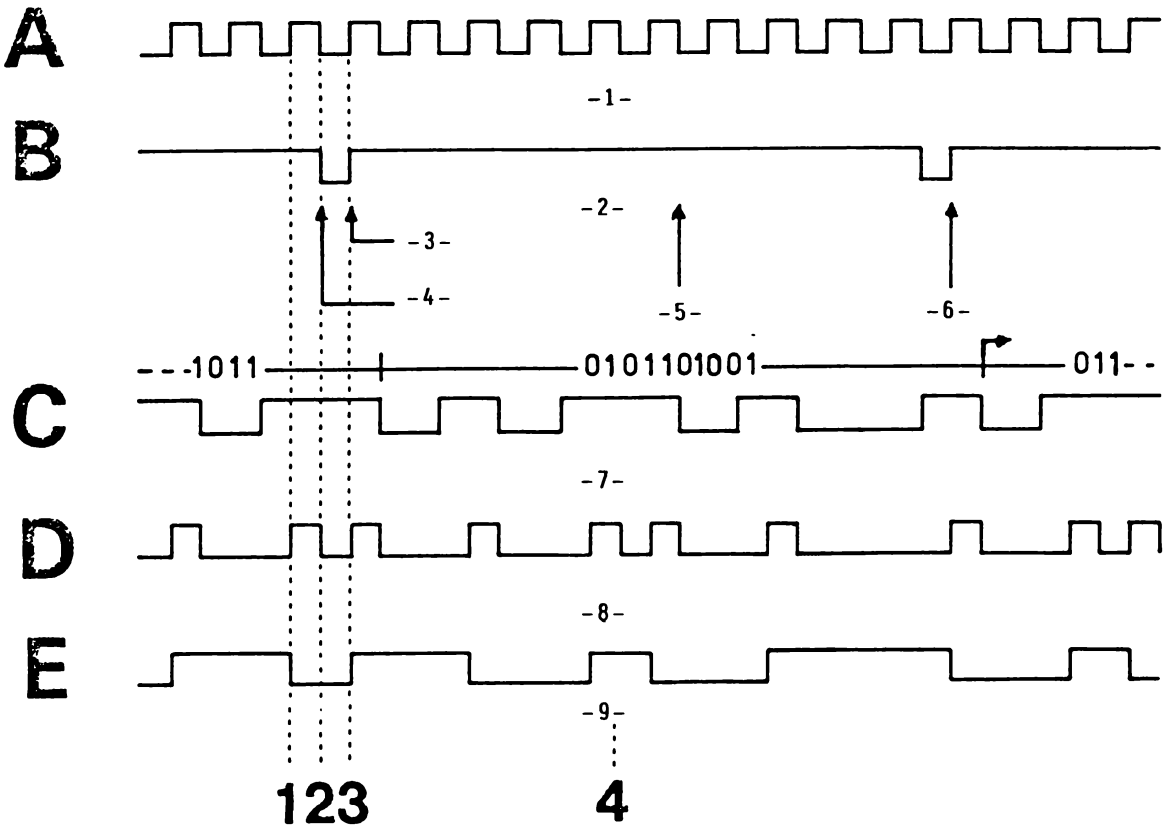


Fig. 12 - CICLO DI SCRITTURA.

- CICLO DI SCRITTURA -

-1- CLOCK- Gira a circa 250KHZ.

-2- LINEA di READY- Ogni 10 CLOCK.

-3- BYTE successivo nel registro di SHIFT.

-4- FLAG di READY del floppy disk controller.messo a"1" a questo punto.

-5- BYTE successivo messo in INPUT nel registro di SHIFT.

-6- BYTE posto entro il registro di SHIFT.

-7- Uscita di dati dal registro di SHIFT.

-8- Impulsi del CLOCK dalla porta AND.

-9- Uscita dal FLIP-FLOP.

Fig. 12 a.

Diagram 1 - WRITE

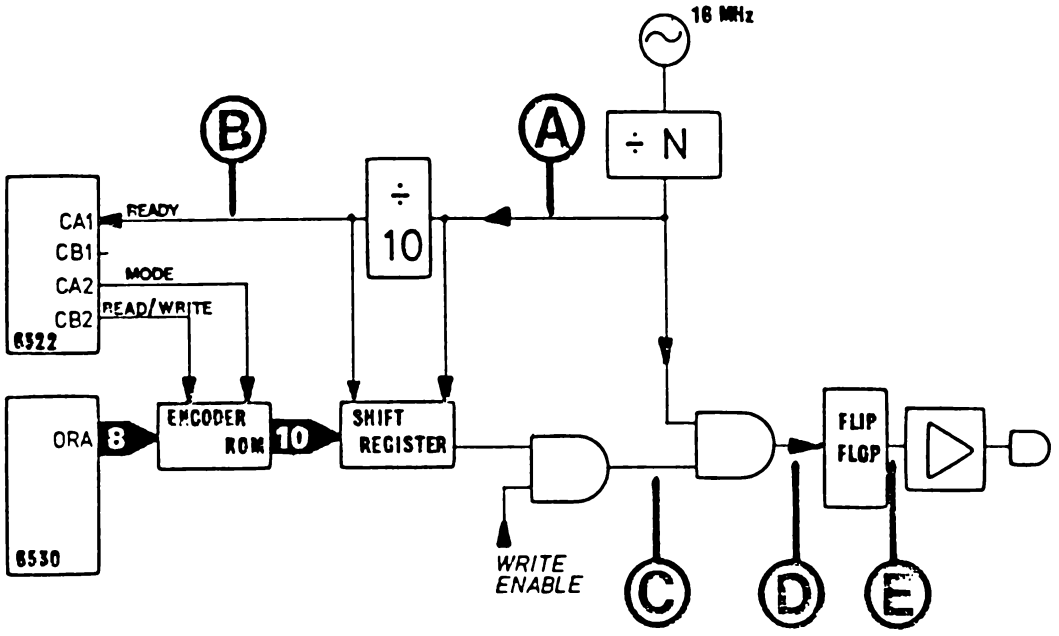


Fig.12 b- Ciclo di scrittura.

RILETTURA DEI DATI

Dobbiamo,per prima cosa ripristinare il flusso di passaggio dati dal disco e pulirlo.
Questo non e' un lavoro facile poiche' i segnali sono a basso livello quando essi lasciano il disco e consistono quindi in uno stretto impulso per ogni flusso di passaggio.
Questi segnali vengono puliti (CLEANED) e riconvertiti in impulsi che assomigliano ai dati originali.
Per gli "1" avremo quindi un' impulso, cioe' un segnale, mentre per gli "0" nessun impulso.
Il secondo diagramma mostra che i dati in ingresso sono forniti in un' impulso seriale di un' altro registro di 10-bits.
Il risultato, il Byte di 10-Bits e' ridecodificato, rimesso nella forma di Byte a 8 bits e reso disponibile per il processore.
Da cio' ne discendono 2 problemi.

Primo come facciamo noi a sapere quando inizia il CLOCKING cioe' la temporizzazione dei dati nel registro ?

Secondo a quale velocita' dovrebbe essere il CLOCK ?

Consideriamo per prima cosa la seconda questione.

Gli impulsi di clock dovrebbero essere generati alla stessa frequenza in cui i dati erano registrati, ma variazioni nella velocita' di registrazione e lettura potrebbero consentire che il CLOCK possa andar fuori di sincronismo con i dati stessi.

Cioe',sebbene il CLOCK possa lavorare bene per la prima coppia di bits alla fine dei 256 Bytes il CLOCK potrebbe essere di molti Bytes fuori.
Si rende quindi necessario un metodo per sincronizzare gli impulsi di CLOCK con i dati in ingresso.
Questo risultato viene ottenuto controllando il generatore di CLOCK ad ogni impulso di dati in ingresso.
Cio' consente il sincronismo fra il movimento di dati ed il CLOCK.
Sfortunatamente se un corretto Byte di 8-bits era codificato con questo sistema potrebbe non esserci nessun impulso per un certo periodo di tempo cosa che ci manderebbe nuovamente fuori sincronismo con il CLOCK.

Immaginiamoci che cosa potrebbe succedere se tutti i 256 Bytes fossero stati a" 0" a partire dall' ultimo.
Con quel Byte il clock potrebbe facilmente andare fuori sincronismo di diversi Bytes ed i dati non sarebbero riletti correttamente.
Per superare questo problema il Byte di 8 -bits e' codificato in maniera tale da garantire che non ci siano piu' di 2 "0" consecutivi entro un Byte.
Per ottenere questo naturalmente c' e' bisogno di piu' di 8-bits. Ecco perche' il dato viene codificato in Byte di 10-bits.
Questa codifica e' effettuata tramite la ROM ed una serie di manipolazioni di bits.
Le due meta' del Byte di 8 bits finiscono per essere codificate nella stessa maniera.
In questo modo "0000" e' codificato come "010100".
L' altra meta' del Byte, poniamo a "0101" e' codificato come "01111" e cosi' via.
Utilizzando questa tecnica avremo la sicurezza che il CLOCK non possa mai andare fuori sincronismo.

C' e' anche un' altro vantaggio poiche' l' insieme di 10-bits consente ora ben 1024 possibili combinazioni, mentre il processo di decodifica ne richiede solo 256.
Cio' lascia 768 combinazioni da 10-bits che non dovrebbereo essere registrati su disco.
Se una di queste combinazioni viene trovata nel processo di lettura allora incorreremo in un errore e il meccanismo di decodifica mettera' la linea di errore a "1".
Se si incorre in un errore di questo tipo l' unita' a dischi generera' un numero di errore "24" cioe' un:

BYTE DECODING ERROR

Non appena i Bytes sono ricevuti, la linea di READY e' utilizzata dal processore per indicare che tutti i dieci bits sono stati ricevuti e i dati sono quindi disponibili per il CONTROLLER.
Dopo di che il CONTROLLER deve rileggere i dati al piu' presto possibile perche' il rinfresco o REFRESH ed i dati saranno quindi disponibili per un periodo di tempo molto breve come si puo' vedere dal diagramma di temporizzazione.
Tutto cio' pero' non risponde alla domanda circa l' identificazione dell' inizio di questo processo, domanda che ci eravamo proposti per prima.

La conoscenza del processo di inizio della sequenza di 256 Bytes (quelli da 10-bits per intenderci) e' vitale perche' basta un solo bit fuori e nessuno dei dati sara' letto correttamente.
Subito prima di un blocco di dati esiste uno speciale carattere di sincronismo (SYNC CHARACTER) che consiste in una serie consecutiva di 11 "1".
Poiche' il processo di codifica come abbiamo viisto

garantisce che nessun normale BYTE di dati sia registrato con tutti "1", questi Bytes, cioe' i bytes di sincronizzazione sono unici.
Una gabbia di AND e' usata nell' uscita parallela del registro di SHIFT per controllare che tutti i 10-bits siano a 1.
Non appena viene verificato questo stato di cose la linea di SINCRONIZZAIONE viene messa bassa.
Viene inoltre "CONGELATO" cioe' bloccato il contatore da 10-bits usato per generare il segnale di READY.
Questo e' congegnato in modo tale che il primo Byte di dati che segue il caratttere di sincronizzazione avra' uno "0" come primo Byte.
Non appena questo viene ricevuto la linea di sincronizzaione viene posta alta, si ha un RESTART del contatore di 10-bits e il segnale di READY e' ora sincronizzato con i dati in arrivo.
L' impulso di sincronizzazione e' anche disponibile per il CONTROLLER in modo tale che questi possa sincronizzarsi con la sua operazione di lettura.
Di seguito diamo un breve sommario delle linee di controllo.
Tutte queste linee sono disponibili per il CONTROLLER il cui lavoro consiste nel presentare i BYTES di dati all' ingresso e di controllarne la concatenazione.
Quindi di rileggere i Bytes di dati dall' uscita in modod tale di avere la giusta concatenazione nel giusto momento.

READY

Questa linea va bassa ogni decimo impulso di CLOCK per indicare una sequenza completa di 10-bits.
Nel modo scrittura (WRITE MODE) sta ad indicare che il registro SHIFT e' pronto per ricevere il

successivo Bytes di dati.
Nel modo di lettura (READ MODE) indica al CONTROLLER che e' disponibile un Byte di dati valido.

SYNC

Questa linea va bassa tutte le volte che la circuiteria elettronica riconosce 10 "1" consecutivi.
Il termine del carattere di sincronizzazione e' indicato dal porsi in alto della linea di SYNC che in conseguenza sincronizza e mette in opera la generazione di un segnale di READY.

ERROR

Indica che un Byte di 10-bits illegale (come quelli che abbiamo visto prima) e' stato letto dal disco.

MODE

Per una normale codifica di dati, la linea MODE e' uguale a "0".
MODE = 1 evita che il Byte di sincronizzaizone sia codificato nella normale procedura.
E' usato quindi tutte le volte che i bytes di sincronizzazione devono essere scritti come tali.

READ/WRITE

Queste linee sono separate dalle normali linee HARDWARE READ e WRITE.
Sono fissate dal SOFTWARE del CONTROLLER per

determinare in che modo le operazioni LOGICHE di entrambe le funzioni dovrebbero porsi in essere.
Sono messi a "1" per READ ed a "0" per WRITE.

Parliamo ora un attimo del CLOCK che viene gestito dal registro di SHIFT.
La frequenza di CLOCK e' di circa 250Kb per secondo.
La frequenza di CLOCK normale varia a secondo il punto in cui i dati sono registrati su disco.
Quando la testina e' posizionata sulle tracce piu' esterne, il disco passa sotto la testina a circa 2 volte la velocita' con la quale essa passa sulle tracce piu' interne.
Percio' la densita' di registrazione deve essere cambiata per tenere conto di questo fatto.
Per lo stesso motivo e' comprensibile che riducendo il numero di settori nelle tracce piu' interne la velocita' campione di CLOCK debba variare.
Ricordiamo che il disco e' diviso in 4 zone e che la frequenza di CLOCK e il numero di settori per traccia cambiano a secondo della zona.
L' attuale CLOCK e' generato da un' oscillatore a 16 Mhz. Questa frequenza e' divisa secondo la tavola sottoriportata.
Il CLOCK che ne risulta e' quindi diviso ulteriormente per 4 per produrre una frequenza di CLOCK operativo di circa 250Khz.

ZONA DISCO	TRACCE	NUMERO SETTORI	CLOCK DIV	FREQUENZA ATTUALE
0	1-17	21	13	307.692Kb/s
1	18-24	19	14	285.714Kb/s
2	25-30	18	15	266.667Kb/s
3	31-35	17	16	250.000Kb/s

CAPITOLO DICIASSETTESIMO

IL FORMATO DEL DISCO

Polemiche lunghissime sono sorte fra i puristi di questa materia, in particolare in INGHILTERRA, circa il formato dei dischetti ed i relativi DOS.
La forma piu' corretta dovrebbe essere quella di riferci al formato del dischetto come viene trattato dall' unita' stessa.

Ad esempio in quanto riportiamo il formato 1 si riferisce ai dischi 2031 e 3000, il 2a ai dischi della serie 4000 e 1540/1541 ed il 2c alla serie 8000.
Ricordiamo tuttavia che malgrado i 4000 e 1500 abbiano il formato 2a hanno pero' differenti DOS.

Il formato 1 o 2a di un disco ha 35 tracce concentriche numerate appunto da 1 a 35.
La traccia 1 sara' quindi quella piu' esterna e la 35 quella piu' interna rispetto al centro del dischetto stesso.
Il formato 2C ha invece 77 tracce.

NOTA

E' da notare che quindi per i dischi 8050 e 8250 e' l' incremento nella densita', cioe' nel numero delle tracce,, ad aumentare la capacita' e non l' incremento della densita' stessa per ogni traccia, se non in piccola parte.

Le tracce hanno un numero variabile di settori a secondo di dove sono presenti su disco.

Le tracce piu' esterne hanno il maggior numero di settori perche' esse sono effettivamente piu' lunghe.
La suddivisione in tracce e settori e' la seguente:

FORMATO 1		FORMATO 2A		FORMATO 2C	
tr	se	tr	se	tr	se
1-17	0-20	1-17	0-20	1-39	0-28
18-24	0-19	18-24	0-18	40-53	0-26
25-30	0-17	25-30	0-17	54-64	0-24
31-35	0-16	31-35	0-16	65-77	0-22

Come possiamo notare dalla precedente tabella, la sola differenza fra il formato 1 e 2A e' che ognuna delle tracce nell' intervalllo 18-24 ha un settore in meno.
Il risultato di questo e' che un tentativo di scrivere su un disco 2a da parte del DOS 1 puo' richiedere 1' accesso ad un settore non esistente mentre il tentativo di scrivere su un formato 1 usando un Sistema Operativo che si aspetta di trovare una formattazione di tipo 2A non potra' usare i settori extra.

A causa di questi problemi, non sara' possibile un' operazione di scrittura tra un DOS e 1' altro, mentre invece sara' possibile un' operazione di lettura sul formato 2A in quanto il Sistema Operativo ne riconosce comunque la compatibilita7 di formato e ne accetta i settori extra.

NOTA

E' bene notare che i numeri delle tracce iniziano da 1 mentre la numerazione dei settori incomincia da 0.

Ogni settore su disco e' composto da 2 blocchi:

- Un blocco corto (HEADER BLOCK) conosciuto anche come blocco indirizzi.

- Un blocco lungo conosciuto anche come DATA BLOCK.

Il blocco di testa (HEDER BLOCK) e' composto da:

**Un carattere di sincronizzazione o SYNC CHARACTER.

**Un byte di identificazione blocco o BI che per quanto riguarda la testata assume il valore esadecimale 08.

**Un byte di controllo somma o CHECKSUM BYTE (CKS)

**I numeri di traccia e settore o SCT e TRK

**E di due BYTES dell' identificatore disco o ID che viene immesso quando il disco e' formattato.

NOTA

E' da notare che, seguendo la norma ,ID e' diviso in IDL o IDENTIFICATOR LOW e IDH o IDENTIFICATOR HIGH.

A questo segue una interruzione conosciuta come

HEADER GAP a cui segue l' inizio del blocco dei dati.
Questo blocco di dati inizia con un' altro carattere di sincronizzazione e un identificatore di blocco che assumera' il valore esa 07.
Questo e' seguito da 256 Bytes di dati i primi due quali sono utilizzati come Bytes di LINK, cioe' come collegamento.
Al termine c' e' il CHECKSUM DIGIT.
Prima di passare al successivo HEADER avremo un' INTER SECTOR GAP.
Quando un disco e' formattato con un comando di NEW, tutti i blocchi di HEADER sono scritti su disco e correttamente posizionati.

NOTA

Nel DOS 1 vengono scritti dei blocchi di dati DUMMY e che contengono tutti 0, ma nelle successive versioni non e' presente nessun blocco di dati ma solo un' intervallo.
Comunque e' bene rivedere le tavole di distribuzione dei valori nelle singole testate per conoscere come e dove sono scritti i dati.

La dimensione dell' intervallo fra HEADER e il blocco di dati al quale si riferisce e' fissato dal Sistema Operativo.
Questo GAP o interruzione non puo' essere troppo corto perche' in questo caso il carattere di sincronizzazione del blocco dati passerebbe la testa e quindi il DOS non riuscerebbe a leggere correttamente dall' inizio il blocco dati.

Non puo' essere neppure troppo lungo perche' lo spazio fra le traccce sarebbe tale da diminuire la

capacita' di memorizzazione del disco.

Nel DOS 1 gli INTER SECTOR GAPS sono anche fissati in lunghezza, ma questo crea dei problemi su delle tracce dove il numero dei settori era stato immesso al limite delle capacita'.
Infatti se la velocita' di formattazione del drive e la velocita' di scrittura dati sono molto diverse e' possibile che l' ultimo settore di una traccia possa andare a sovrammettersi sul primo settore della traccia successiva.

NOTA

Alcuni dischi non COMMODORE usano il sistema detto SECTOR INTERLEAVING. Con questo sistema i settori non sono in ordine numerico sequenziale, ma in una speciale sequenza per esempio:

0-3-6-9-12-15-18-1-4-7- ecc.

questo per evitare la completa rotazione fisica del dischetto e quindi dei lunghi ritardi dovuti a fattori meccanici di allineamento e di ricerca.

La Commodore non utilizza questo sistema.Infatti sui drive CBM i settori sono uno dietro l' altro e viene utilizzata una tecnica SOFTWARE per scegliere il settore piu' efficiente, da un punto di vista della ricerca successiva, sul quale scrivere i dati.
E' da notare che l' FDC usa i blocchi di HEADER per eseguire un controllo sulle tracce e per trovare il

settore richiesto oltre che per mantenere un controllo di identificatore su disco.
L' identificatore o ID mostrato nella DIRECTORY e' solo un' informazione per la memoria dell' utente.

LA CODA DI LAVORO

Per comunicare con il FLOPPY DISK CONTROLLER l' IP utilizza la normale area di memoria RAM dove vengono immagazzinati i dati.

Ogni BUFFER di 256 Bytes e' associato ad un JOB REQUEST BYTE in una JOB QUEUE TABLE.
Chiariremo successivamente, anche con l' ausilio di una tavola questi nomi.
Ogni BUFFER ha anche un punto d' ingresso in una tavola di HEADER o di TESTATA e per leggere un dato settore all' interno di un BUFFER, l' IP immette un Byte di richiesta lavoro nella coda, inserisce la testata (HEADER) del settore atteso in una tavola e quindi aspetta.
Il bit 7 del Byte di richiesta lavoro e' messo a 1 per indicare al CONTROLLER che questa e' appunto una richiesta di lavoro ed il CONTROLLER riporta un codice di stato con il bit 7=0 quando il lavoro e' terminato.
Il sistema con il quale il CONTROLLER manipola questa coda di lavoro e' un po' complesso.

Infatti il CONTROLLER esegue continuamente una scansione, cioe' una lettura, della JOB QUEUE, decide quale operazione possa essere eseguita nella maniera migliore oppure quale e' la piu' efficace azione da eseguire.

Nell' unita' a doppio drive viene controllato se e'

in funzione il motore del Drive 1 o dello 0, quale traccia e di quale settore si trova sotto la testina e quale combinazione di traccia/settore passera' subito dopo sotto la testina.

I codici utilizzati nella coda di lavoro sono riportati di seguito.
Notare che il numero del drive richiesto in lavoro e' contenuto nel bit 0 cosi' che il comando di READ sul drive 1 dovrebbe avere un byte di richiesta lavoro pari a $81.

$80 - READ di un settore
$90 - WRITE di un settore
$A0 - VERIFY di un settore
$B0 - SEEK cioe' ricerca di un settore
$C0 - BUMP cioe'riporto della testina alla traccia 1
$D0 - JUMP salto diretto ad un codice macchina nel buffer
$E0 - EXECUTE di un codice nel buffer con messa a punto di testina,motore, ecc.

Le operazioni eseguite da $80 e $90 si spiegano da sole. $A0 e' utilizzata solo alla fine di un' operazione di scrittura per essere sicuri che i dati immessi possano essere poi riletti esattamente come erano stati scritti.

Il comando SEEK, cioe' ricerca di un settore, consente di accedere all' identificatore del disco.

Il comando BUMP produce quello strano rumore perche' la testina fisicamente ritorna alla traccia 1 cosi' che da quel punto, sul drive, si sappia esattamente dove ci si trova.

Il comando JUMP costringe il controller ad eseguire il codice macchina presente nel BUFFER non appena il comando stesso viene trovato nella coda di lavoro.

NOTA

Quest' ultimo comando non e' implementato nel DOS 1.

Il comando EXECUTE attende che il drive sia disponibile e che la testina di lettura/scrittura sia posizionata sulla traccia giusta.
Non e' possibile utilizzare comandi M-R direttamente sugli spazi indirizzati dal CONTROLLER.
Per questo deve essere scritta una piccola routine e quindi trasferita all' interno di uno dei Buffers e poi eseguita usando i comandi EXECUTE o JUMP.
Questa routine dovrebbe trasferire la memoria indirizzata dal CONTROLLER entro l' area di RAM comune a tutti e due i processori.(vedi anche capitolo precedente).
Cio' consentira' che questi dati siano resi accessibili dall' IP e successivamente dall' unita' centrale.

Allo stesso modo che abbiamo visto per il bit 7=0 i codici di errore sono nell' intervallo da 1 a 16 ($01/$10) e, sono convertiti in numeri in un' intervallo fra 20 e 29 perche' possano essere avviati all' unita' centrale quando viene interrogato il canale di errore.

SCANSIONE DELLA CODA DI LAVORO

La coda viene costantemente esaminata e non appena una richiesta di lavoro viene trovata questa e' controllata.
Se questa richiesta di lavoro e' un comando di JUMP allora l' esecuzione continua fino al primo Byte dell' appropriato Buffer.
Se questo non avviene allora il giusto drive viene messo in funzione e parte un ciclo di ritardo.
Viene quindi effettuato un controllo per vedere se la testina e' stata fissata correttamente ed il ciclo continua fino a quando non viene trovato il lavoro, cioe' l' operazione, per il quale la testina di lettura/scrittura era stata selezionata.

Se l' operazione richiesta era BUMP, allora la testina e' messa in posizione e fissato l' appropriato drive.
L' FDC portera' la testina indietro sulla traccia 1 e reiniziera' il ciclo principale.
Se non e' un' operazione di BUMP viene effettuato un controllo per vedere se la testina e' sulla stessa traccia del lavoro richiesto.
Se e' cosi' viene effettuato un controllo per conferamre che il drive e' disponibile ed e' presente l' adatta routine per effettuare il lavoro richiesto.

Se l' operazione da eseguire non e' su questa traccia, allora la coda di lavoro (JOB QUEUE) e' di nuovo esaminata per l' operazioneo il dato piu' vicino all' attuale traccia e viene effettuata una richiesta per una routine di INTERRUPT che fara' muovere la tastina alla traccia richiesta.
Il ciclo e' quindi ripreso e viene lasciato solo quando la testina e' arrivata nella giusta

posizione.

Quando viene trovata la traccia viene fissata la necessaria zona di temporizzazione e il numero di settore sulla traccia stessa.
E' a questo punto che viene identificato il comando di EXECUTE e l' esecuzione continua con il primo Byte dell' appropriato buffer.
E' quindi letto il primo HEADER disponibile su disco, controllato il suo CHECKSUM e la traccia registrata del CONTROLLER stesso viene fissata.

Se il lavoro richiesto era SEEK, allora lo HEADER letto da disco e' trasferito nello HEADER associato con l' attuale Buffer ed il lavoro e' completo.

Le sole operazioni lasciate indietro sono READ, WRITE e VERIFY e il CONTROLLER procede a controllare che l' identificatore appena letto da disco sia lo stesso presente nella testata del Buffer.
Se tutto e' OK viene considerato come giusto l' attuale settore e viene ricercata la coda di lavoro per una qualsiasi richiesta di questo settore.

LETTURA DI UN BLOCCO

Per prima cosa viene effettuata una chimata ad un subroutine. Questa richiesta rimane in attesa fino a quando lo HEADER del settore richiesto non sia letto.
Cio' e' fatto attraverso la lettura di ogni blocco di testa (HEADER BLOCK) sulla traccia fino a quando il suo contenuto non corrisponda esattamente alla richiesta testata.

La routine attende quindi che le arrivi l' impulso di sincronismo del blocco successivo e legge il primo Byte che dovrebbe essere l' identificatore di blocco 07.
Se cosi' non e' viene generato un' errore di tipo 07.

Quindi, cioe' dopo l' esecuzione di questo lavoro, tutti i 256 Bytes di dati sono letti e immessi nell' appropriato Buffer.
Viene quindi effettuato un controllo di CHECKSUM e un controllo sul FLAG di errore.
Se tutto va bene un codice di OK viene riportato nella coda di lavoro.

SCRITTURA DI UN BLOCCO

Per prima cosa viene controllata la tacca posta in alto a destra del dischetto.
Se questa tacca non e' coperta allora la routine attende di leggere un corretto HEADER che deve essere trovato esattamente nello stesso sistema viso per READ.
Prima di iniziare a scrivere i dati viene generato un ciclo di ritardo per consentire che il GAP di INTER-BLOCK passi la testina e quindi il drive sia posizionato nel modo scrittura.
Vengono scritti 3 bytes di sincronismo seguiti da un identificatore di blocco dati e quindi vengono scritti i 256 Bytes dal buffer seguiti dal relativo CHECKSUM.
Quando tutto questo lavoro e' stato completato la routine di WRITE si posiziona sul codice da $90 a $A0 e si ricongiunge al ciclo principale.
Cio' consentira' che i dati appena scritti possano

essere verificati alla successiva rotazione del disco.

VERIFICA DEI DATI

Questa funzione agisce esattamente come la routine di READ eccetto che ogni byte letto e' confrontato con i contenuti del buffer.

SERVIZIO DI INTERRUPT

La sola sezione del CONTROLLER non ancora spiegata e' quella relativa alla routine di INTERRUPT.
Questa routine e' chiamata in funzione circa 65 volte al secondo ed ha due distinte sezioni di funzionamento.
La prima sezione manipola e controlla i motori del drive.
Ci sono due ritardi associati con ogni motore.
C' e' un ritardo di velocita' (UP TO SPEED DELAY) che controlla che il motore sia appena messo in funzione e fissa i bits 7 delle locazioni 03/04 (rispettivamente del drive 0 e 1) fino a quando il ritardo non sia terminato.

L' altro ritardo e' quello connesso al fatto che il drive sia in funzione o meno. La seconda meta' della routine controlla il funzionamento dei motori passo passo.
Un byte presente nelle locazioni 05/06 (rispettivamente quindi per i motori 0 e 1) viene fissato in modo tale che il CONTROLLER possa controllare i movimenti della testina.

Questi Bytes indicano quante tracce ed in quale direzione la testina si dovrebbe muovere.
L' interrupt manipola questi passi (ricordiamo che si tratta di motori particolari con movimento non continuo ma passo passo).
Fissa quindi il bit 6 delle locazioni 03/04 per i drives 0 e 1 fino a quando la testina non sia posizionata sulla giusta traccia.
La parte rimanente dei contenuti di 03/04 e' utilizzata dal CONTROLLER per prendere nota della traccia attuale sulla quale la testina stessa e' in quel momento posizionata.
Questa informazione non e' utilizzata dalla routine di INTERRUPT.

Nel prossimo capitolo vedremo alcune particolarita' nell7 uso dei dischetti.

CAPITOLO DICIOTTESIMO

COMPATIBILITA' FRA 8250 E 8050

Ci sono numerose differenze fra i modelli COMMODORE 8050 e 8250.
La piu' ovvia e' che nell' 8250 vengono utilizzate entrambe le facce del disco mentre nell' 8050 viene usata solo una faccia del disco.
Anche la struttura dei files relative sulle macchine e' notevolmente diversa e richiede una certa cura nel suo uso.
La migliore capacita' nell' 8250 e', almeno in teoria, quella di poter usare l' intero disco per immagazzinare FILES di tipo RELATIVES, mentre al contrario sull' 8050 un file relative puo' avere una massima lunghezza di 182K.
Mentre e' possibile utilizzare nell' 8250 dischi scritti sull' 8050, ci sono alcune precauzioni che e' necessario siano prese.
La prima precauzione e' di carattere che potremo definire qualitativo o commerciale.
Infatti dobbiamo ricordare che sull' 8250 e' necessario utilizzare dischi di alta qualita' che siano ASSOLUTAMENTE:

DOPPIA FACCIA

DOPPIA DENSITA'

77 TRACCE

Il primo tentativo di accedere ad un disco scritto

e formattato su una unita' 8050 con una unita' 8250 potra' dare un' errore disco, ma continuando a provare con questi tentativi normalmente il secondo riesce.
Non e' altrettanto semplice quando invece si tenti di utilizzare files relatives scritti in precedenza sull' 8050.
Questo perche' il CBM 8250 deve essere messo in grado di manipolare i files relatives non espandibili utilizzando una routine come la seguente:

```
OPEN 15,8,15
PRINT#15,"M-W";CHR$(164)CHR$(67)CHR$(1)CHR$(255)
CLOSE 15
```

Con questo sistema diventa relativamente facile eseguire la copia.
Continuiamo ora con le istruzioni per la copia o meglio per il trasferimento di dati da un tipo di disco all' altro.
Inserire e far girare il piccolo programmino per disabilitare i FILES RELATIVES, quindi con il disco 8050 nel drive 0 ed il disco formattato 8250 nel drive 1 copiare il file relative con il normale sistema.
Si puo' usare l' istruzione:

```
COPY D0,"programma sorgente" TO D1," programma destino"
```

NOTA

Nel caso si abbia una segnalazione di errore disco

al primo tentativo ripetere semplicemente il comando.

Quando la copia e' stata completata sara' necessario RESETTARE l' 8250 e questo potra' essere fatto nella maniera piu' semplice spegnendo e poi riaccendendo l' unita' a dischi.

Inserire quindi il DEMO DISK fornito dalla COMMODORE stessa con la macchina, nel drive 0.
Caricare ora e far girare il programma :

EXPAND RELATIVE

Per quanto la COMMODORE affermi che non esistono problemi di copia e per questo come abbiamo visto, fornisca anche un programma per la copia di files piu' complessi come sono appunto i relatives, alcuni programmi possono dare delle seccature.
I programmi da noi provati come VISICALC e WORDCRAFT hanno funzionato correttamente, mentre la stessa cosa non e' stata verificata con il SILICON OFFICE di cui e' inibita la copia.
Secondo noi l' utente, ad esempio in questo caso, dovrebbe contattare la :

BRISTOL SOFTWARE FACTORY

che ne ha curato la messa a punto per l' assistenza nel caso il rivenditore sia sprovvisto di apposito dischetto di duplicazione.

Per altro dobbiamo concludere che la copia di files

ordinari da 8050 a 8250 non presenta, dopo lunga esperienza, grandi difficolta' sopratutto se ci si ricorda di accedere per prima cosa al disco formattato sull' 8050 per il problema dell' errore come segnalato all' inizio.

Ricordiamo quindi di procedere in questo modo. (La procedura e' consigliata, NON OBBLIGATORIA).

- Inserire il disco 8050 nell' unita' 0 e il disco 8250 nell' unita' 1.

- Formattare il disco 8250 nella maniera usuale magari utilizzando il comando HEADER e, sarebbe l' ideale, conservando lo stesso ID e lo stesso nome del disco.

*** ATTENZIONE ***

RICORDARE DI PRENDERE LA PRECAUZIONE, CHE DOVREBBE ESSERE ABITUALE, DI PROTEGGERE LA FINESTRELLA DEL DISCO SORGENTE.
IN QUESTO CASO IL DRIVE 0.

--Eseguire il CATALOG D0, cioe' del disco 8050, facendo l' operazione 2 volte se necessario per spegnere il LED che potrebbe segnalare una condizione di errore.

--Eseguire la copia con:

COPY D0 TO D1

Dobbiamo segnalare un' ultima differenza fra le due unita' relativamente al codice di errore riportato nel DS$ o attraverso il canale di errore.
Mentre nell' 8050 avremo un abituale messaggio di 4 dati, cioe' una risposta di 4 parametri, nell' 8250 ne avremo 5 secondo il formato di seguito indicato:

NUMERO D' ERRORE

MESSAGGIO D' ERRORE

TRACCIA

SETTORE

DRIVE

ANTI COPIA

Uno dei principali problemi per chi crei del SOFTWARE e' quello di inibire la possibilita' di copia.
Premettiamo che questo paragrafo non e' stato scritto per spiegare dei metodi anticopia. Anche perche' essendo questo un libro di larga diffusione cio' sarebbe una contraddizione in se.
Ci sono 4 sistemi per copiare un programma da un disco ad un' altro.
Non in tutte ma in alcune unita' CBM esistono i comandi di BACKUP e di DUPLICATE che fanno parte del sistema operativi del disco e che eseguono un lavoro simile.
Eseguono cioe' la copia BLOCCO PER BLOCCO dell' originale.

Se durante questo processo di copia viene incontrato un' errore sul disco originale, il procedimento stesso si ferma.
Cioe' la copia non viene effettuata. Per cui per prevenire questo tipo di copia esiste il semplicissimo metodo di ROVINARE O SCIUPARE comunque una parte del disco.
Si tratta di una forma molto poco elegante di protezione che puo' essere eseguita con un magnete, cioe' con una calamita.
Con questo magnete e' possibile cancellare alcune sezioni della prima traccia.
Come si vede si tratta di un sistema estremamente impreciso oltre che come avevamo detto poco elegante.

In alternativa il disco puo' essere formattato omettendo la prima coppia di tracce.
Cio' puo' essere fatto facilmente iniziando il processo di formattazione su un secondo disco e dopo che sono state formattate due o tre tracce inserire il disco che si vuole proteggere in modo tale che la formattazione continui.

Il secondo metodo di copia e' di usare il comando COPY del DOS che legge il file originale, carattere per carattere, e lo riscrive sul disco copia.

Il terzo metodo coinvolge l' utilizzo dei comandi U1 e U2 del DOS per leggere e poi per scrivere ciascun blocco su disco.
Questo metodo eseguira' anche la correzione di ogni errore di CHECKSUM nei data e pertanto e' un buon metodo per recuperare alcune forme di errore su disco.

Il quarto metodo di copia e' di far rileggere il programma dal computer e dopo di riscriverlo sul

nuovo disco.

In effetti pero' a questo metodo sono state trovate le prime e piu' comuni forme di protezione.
Infatti e' stato inserito il RUN automatico e sono stati disabilitati i tasti di RUN/STOP e sulle serie VIC e CBM 64 RUN/STOP e RESTORE. Inoltre sono state ottenute altre precauzioni per prevenire che questo programma quando gira possa essere arrestato nella sua esecuzione.

Vediamo quindi che la protezione contro le copie coinvolge una varieta' di tecniche complesse come la rottura o corruzione di tracce logiche del disco, la scrittura di programmi in forma non usuale, la codifica di programmi o l' immissione di identificatori non copiabili su disco; questi identificatori si riferiscono a quella parte del disco che viene costantemente controllata.

In effetti spesso vengono utilizzate tutte queste forme di protezione in combinanazione piu' o meno spinta.

Tuttavia il grande svantaggio di un sistema di protezione anticopia e' che comunque esiste la necessita' che definiremo essenziale di far rileggere il programma dal computer.
Per questo motivo molto spesso non ci si puo' spingere oltre determinati limiti nelle tecniche di protezione anticopia.
Un' altra necessita' e' che l' utente possa comunque eseguire con una certa facilita' delle copie di sicurezza anche se questo problema puo' essere superato da accordi o convenzioni particolari fra la ditta fornitrice del SOFTWARE e l' utente stesso.

Vogliamo accennare ancora ad altri due sistemi uno dei quali particolarmente in voga per i giochi.

Con il primo di questi sistemi viene utilizzato un programma chiamato :

LOADER

che verra' caricato nella memoria del computer ed al momento del RUN carichera' il programma principale, che poi deve essere quello esecutivo, e controllera' che alcuni identificatori, e questo per qualsiasi tipo di identificatori si tratti,cioe' alcune parole chiave siano immesse correttamente nel disco.

Pero' per chi abbia una certa pratica, cioe' anche senza essere dei tecnici altamente qualificati, e' possibile modificare il programma LOADER per togliere questi controlli e determinare quindi il sistema di codifica utilizzato all' atto del salvataggio di questo programma.

Alcuni programmi utilizzano delle chiavi di protezione HARDWARE chiamate anche DONGLE che consentono che il programma sia copiato ma non ammettono che il programma stesso possa girare su una macchina diversa da quella per la quale e' stato costruito o comunque su una macchina sprovvista di chiave.

Al momento attuale, anche se non e' una forma di protezione sicura al 100 % e' pero', in particolare con le DONGLES EVM SPECIALS,(e possiamo assicurarvi che non e' pubblicita' gratuita), la forma che meno si presta a manipolazioni in quanto molto spesso le chiavi sono inserite in contenitori

ricoperti di resine particolari che non ne permettono l' apertura.

Tuttavia e' possibile togliere la chiave di protezione e per esempio immagazzinare una piccola routine che incorpori il controllo della chiave.

A conclusione di questo nostro breve discorso possiamo dire che se e' consigliabile utilizzare, e questo in rapporto al prezzo ed al tipo di programma, protezioni sempre piu' complesse ed in definitiva costose, l' ultima delle quali la chiave HARDWARE, in pratica non esiste MAI la sicurezza assoluta di avere una cassaforte a prova di ladro.
Per questo motivo si consiglia di non spendere piu' tempo del necessario a proteggere i programmi.
Esercizi di protezione sui programmi sono invece altamente raccomandabili per imparare ad utilizzare sia tecniche di compattazione che metodi dii lavoro su disco.

NOMI DI FILE: APPROFONDIMENTO

Quando si esegue un' operazione di OPEN, SAVE o LOAD di un file disco, la prima informazione che e' inviata dal computer all' unita' a dischi e' il nome del file e in tutti e tre i casi e' inviata nella stessa identica maniera.
La sola eccezione puo' essere costituita dall' uso del comando OPEN senza un nome di file, quando niente di niente e' inviato al drive.

I comandi di LOAD e SAVE iniziano come OPEN tranne che per lo specifico indirizzo secondario

utilizzato:

0 per LOAD

1 per SAVE

La prima cosa che accade con OPEN e' che la linea di ATTENTION (ATN LINE) viene fissata (messa a 1) per indicare che sta per essere inviato un comando e un comando di LISTEN va sul BUS.
Questi e' nella forma 001x xxxx dove x e' il numero di periferica.
In teoria questo numero puo' andare da 0 a 31, ma da 0 a 3 sono riservati dalla COMMODORE per le proprie periferiche (vedi tavole) ed il valore 31 e' utilizzato come UNLISTEN standard.

Con il disk drive in posizione di LISTENING, viene inviato ora l' indirizzo secondario.Ricordiamo che l' ATN e' a 1.
Questo indirizzo e' nella forma:

1111 XXXX

dove per X si intende l' indirizzo secondario selezionato.
L' 1111 sta ad indicare al disco che questo e' un comando di OPEN e che seguira' il nome di un file.

NOTA

Ricordiamo infatti che per le operazioni su disco, a differenza che per quelle relative alla cassetta

il nome del file e' OBBLIGATORIO.

La linea ATN e' quindi messa a ZERO e viene inviato il nome del file. L' ultimo carattere viene inviato con un EOI.
Al termine di questa complessa manovra viene inviato un comando di UNLISTEN:

0011 1111

E LA LINEA DI ATN RIMESSA A 1.

Dopo che queste operazioni sono state eseguite l' unita' a dischi consente le necessarie manovre in conseguenza del tipo di file.
Attende di ricevere o inviare dati quando viene richiesto.

Ad esempio un comando PRINT£ inizia con l' invio di un comando di LISTEN alla periferica come abbiamo visto ed invia l' indirizzo secondario nella forma:

0110 XXXX

dove per X si intende l' indirizzo secondario.
Questo attiva il giusto canale entro il disco poi procede a prelevare il carattere dal bus fino a quando non viene fissato un EOI con l' ultimo carattere.

Per terminare viene inviato un UNLISTEN dal computer stesso.

SAVE

Lavora esattamente allo stesso modo.
I dati sono solo inviati e l' indirizzo secondario e' a l cosa che forza il disco a eseguire un' azione di :

PRG, WRITE file

cioe' a scrivere un file programma.

CLOSE

In questo caso viene inviato un LISTEN seguito dall' indirizzo secondario, ma questa volta nella forma 1100 XXXX.

INPUT e LOAD

Lavorano nella stessa maniera, ma dopo che e' stato inviato il nome del file e quindi il disco e' pronto per inviare dati, la sequenza inizia con un comando di TALK nella forma:

0101 XXXX

dove X e' l' indirizzo secondario.

La forma standard di UNTALK e':

0101 1111

L' azione precisa che otteniamo di conseguenza ad un comando OPEN oppure con SAVE o LOAD dipende da cio' che e' contenuto nel nome del file e dall' indirizzo secondario utilizzato.

Il FILENAME e' analizzato per identificare il numero del drive, il nome del file, il tipo del file ed il modo del file.
Se il modo file e' omesso verra' selezionata l' operazione READ.
Se entrambi i parametri sono omessi verra' scelto l' ultimo tipo di operazione effettuata sull' ultimo file.

LETTURA DELLA DIRECTORY

E' spesso importante per un programma essere in grado di leggere la directory del disco e questo puo' essere fatto in vari modi.

Come primo metodo puo' essere aperto un file come file programma, utilizzando il comando:

```
OPEN 2,8,0,"$0"
```

Dopo di che sara' possibile prelevare i caratteri dalla Directory usando un comando GET£.
Questo sistema riportera' la Directory nel suo formato programma, con gli indirizzi di collegamento (LINK) e con la scrittura del tipo di file.
Questo metodo puo' essere utilizzato per determinare il numero di blocchi liberi utilizzando semplicemente un:

```
OPEN 2,8,0,"0:% & %"
```

Cio' riportera' il nome del disco e l' ID, sempre in formato programmae quindi verranno listati tutti i file con "% & %", che naturalmente non dovrebbero essere presenti.
La successiva linea sara' il messaggio di BLOCKS FREE, dalla quale potra' essere letta la capacita' del disco.

E' possibile tuttavia eseguire un comando di OPEN sulla Directory come un file sequenziale nel sistema normale:

```
OPEN 2,8,2, "$0"
```

caso questo in cui il comando GET # riportera' direttamente i singoli Bytes dalla Directory.
Infatti il primo blocco di Bytes sara' la BAM seguita dal nome del disco e dagli ingressi dei files e completata dai loro puntatori d' inizio.

CAPITOLO DICIANNOVESIMO

TAVOLE DISCO

Nelle pagine seguenti riportiamo delle tavole relative alla distribuzione dei contenuti delle tracce principali del disco.
In pratica i contenuti delle BAM e delle DIRECTORY dei vari tipi di dischetto.

Come abbiamo gia' detto non esistono grandi approfondimenti per le unita' a disco rigido perche' la COMMODORE ancora non ha stabilizzato una politica commerciale e di conseguenza e' inutile dare delle notizie che sarebbero superate in breve.

Particolarmente utile dovrebbe essere l' ultima tavola, con la distribuzione e la descrizione relativa ai FILES RELATIVES.
Infatti la comprensione di questi files e' essenziale sia per gli utenti delle serie professionali sia per gli utenti delle serie 1540/1541 dove sara' necessario ricorrere a gestione SOFTWARE, cioe' a delle simulazioni vere e proprie.

FORMATO DELLA BAM

1540/1541 traccia 18 Settore 00

Byte	Dato	Descrizione
0-1	18-01	Traccia e settore del primo blocco della Directory.
2	65	ASCII di "a" che indica il formato DOS.
3	00	Riservato per usi futuri del DOS.
4-143		Bit MAP dei blocchi disponibili per le tracce da 1 a 35. Ogni bit rappresenta un blocco. 1= blocco disponibile. 0= blocco non disponibile.

2031 Traccia 18 Settore 00

Byte	Dato	Descrizione
0-1	18-00	Traccia e settore del primo blocco della Directory.
2	65	ASCII di "a" che indica il formato DOS 2.6
3	00	Riservato per futuri usi del DOS
4-143		Bit MAP dei blocchi disponibili per le tracce da 1 a 35. Ogni bit rappresenta un blocco. 1= blocco disponibile. 0= blocco non disponibile.

2040/3040 Traccia 18 Settore 00

Byte	Dato	Descrizione
0-1	18-01	Traccia e settore del primo blocco della Directory.
2	1	ASCII di "1" che indica il formato DOS 1
3	00	Riservato per futuri usi del DOS
4-143		Bit MAP dei blocchi disponibili per le tracce da 1 a 35. Ogni bit rappresenta un blocco. 1= blocco disponibile. 0= blocco non disponibile.

4040 Traccia 18 Settore 00

Byte	Dato	Descrizione
0-1	18-01	Traccia e settore del primo blocco della Directory.
2	65	ASCII di "a" che indica il formato DOS 2.1
3	00	Riservato per futuri usi del DOS
4-143		Bit MAP dei blocchi disponibili per le tracce da 1 a 35. Ogni bit rappresenta un blocco. 1= blocco disponibile. 0= blocco non disponibile.

8050 Primo Blocco BAM Traccia 38 Settore 00

Byte	Dato	Descrizione
0-1	38-03	Traccia e settore del secondo blocco della BAM
2	67	ASCII di "c" che identifica il DOS 2.5
3	00	Riservato per future espansioni
4	01	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	51	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 1.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 1.
11-255	var.	BAM per le tracce 2-50,5 Bytes per traccia.

8050 Secondo Blocco BAM Traccia 38 Settore 03

Byte	Dato	Descrizione
0-1	39-01	Traccia e settore del primo blocco della BAM
2	67	ASCII di "c" che identifica il DOS 2.5
3	00	Riservato per future espansioni
4	51	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	78	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 51.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 51.
11-140	var.	BAM per le tracce 52-77,5 Bytes per traccia.
141-255		Non utilizzato.

8250 Primo Blocco BAM Traccia 38 Settore 00

Byte	Dato	Descrizione
0-1	38-03	Traccia e settore del secondo blocco della BAM
2	67	ASCII di "c" che identifica il DOS 2.7
3	00	Riservato per future espansioni
4	01	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	51	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 1.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 1.
11-255	var.	BAM per le tracce 2-50,5 Bytes per traccia.

8250 Secondo Blocco BAM Traccia 38 Settore 03

Byte	Dato	Descrizione
0-1	38-06	Traccia e settore del terzo blocco della BAM
2	67	ASCII di "c" che identifica il DOS 2.7
3	00	Riservato per future espansioni
4	51	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	101	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 51.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 51.
11-255	var.	BAM per le tracce 52-100,5 Bytes per traccia.

8250 Terzo Blocco BAM Traccia 38 Settore 06

Byte	Dato	Descrizione
0-1	38-09	Traccia e settore del quarto blocco della BAM
2	67	ASCII di "c" che identifica il DOS 2.7
3	00	Riservato per future espansioni
4	101	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	151	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 101.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 1.
11-255	var.	BAM per le tracce 102-150,5 Bytes per traccia.

8250 Quarto Blocco BAM Traccia 38 Settore 09		
Byte	Dato	Descrizione
0-1	39-01	Traccia e settore del primo blocco della Directory.
2	67	ASCII di "c" che identifica il DOS 2.7
3	00	Riservato per future espansioni
4	151	Numero della traccia piu' bassa rappresentato in questo blocco della BAM.
5	155	Numero della traccia piu' alta +1 presente in questo blocco della BAM.
6	var.	Numero dei blocchi inutilizzati sulla traccia 151.
7-10	var.	Bit map dei blocchi disponibili sulla traccia 51.
11-25	var.	BAM per le tracce 152-154,5 Bytes per traccia.
26-255		Non utilizzati.

9060/9090 Formati della BAM Traccia 1 Settore 00	
Byte	Dato	Descrizione
0-1		Puntatore di traccia e settore al successivo blocco BAM. $FFFF = ultimo blocco BAM.
2-3		Puntatore di traccia e settore al precedente blocco BAM. $FFFF = primo blocco BAM.
4		Numero della traccia piu' bassa in questo blocco della BAM.
5		Numero della traccia piu' alta in questo blocco della BAM.
6		Numero dei blocchi inutilizzati in questa traccia.
7-10		Bit map dei blocchi disponibili su questa traccia.
11-255		Bit map delle successive 49 tracce.

STRUTTURA DEGLI INGRESSI BAM PER OGNI TRACCIA

Quanto segue e' valido per tutte le versioni dei dischi COMMODORE.
Ogni traccia ha 5 Bytes che ne disegnano il contenuto.
Un bit con contenuto =1 ci dice che il bloco e' disponibile, mentre un bit con contenuto =0 ci dice che il blocco e' stato utilizzato.

BYTE	DESCRIZIONE
1	Numero dei blocchi disponibili per questa traccia in q uesto momento.
2	Bit map dei blocchi 0-7
3	Bit map dei blocchi 8-15
4	Bit map dei blocchi 16-23
5	Bit map dei blocchi 24-31

FORMATO DELLA DIRECTORY

2031 Testata Directory Traccia 38 Settore 00

Byte	Dato	Descrizione
1-143		Riservato alla BAM del 2031. (Vedi prima)
144-161		Nome del disco racchiuso fra spazi shiftati
162-163		Numero di ID del dischetto.
164	160	Spazio shiftato
165-166	50,65	ASCII di "2a" che identifica la versione DOS e il formato.
167-170	160	Spazi shiftati
171-255	00	Non utilizzati.

2040/3040 Testata Directory Traccia 38 Settore 00

Byte	Dato	Descrizione
1-143		Riservato alla BAM del 2031. (Vedi prima)
144-161		Nome del disco racchiuso fra spazi shiftati
162-163		Numero di ID del dischetto.
164	160	Spazio shiftato
165-166	50,65	ASCII di "2a" che identifica la versione DOS e il formato.
167-170	160	Spazi shiftati
171-255	00	Non utilizzati.

4040 Testata Directory Traccia 38 Settore 00

Byte	Dato	Descrizione
1-143		Riservato alla BAM del 2031. (Vedi prima)
144-161		Nome del disco racchiuso fra spazi shiftati
162-163		Numero di ID del dischetto.
164	160	Spazio shiftato
165-166	50,65	ASCII di "2a" che identifica la versione DOS e il formato.
167-170	160	Spazi shiftati
171-255	00	Non utilizzati.

NOTA

E' da notare che dati ASCII possono apparire nei Bytes compresi fra 180 e 191 sullo stesso dischetto.

8050 Testata Directory Traccia 39 Settore 00

Byte	Dato	Descrizione
0-1	38-00	Puntatore traccia-settore al primo blocco della BAM
2	67	ASCII di "c" che identifica il formato DOS 2.
5.		
3	00	Riservato per futuri usi del DOS.
4-5		Non utilizzato
6-21		Nome del dischetto
22-23	160	Spazi shiftati
24-25		Numero di ID del dischetto.
26	160	Spazi shiftati
27-28	50-67	ASCII "2c" che identifica la versione.
29-32	160	Spazi shiftati
33-255	00	Non utilizzati

8250 Testata Directory Traccia 39 Settore 00

Byte	Dato	Descrizione
0-1	38-00	Puntatore traccia-settore al primo blocco della BAM
2	67	ASCII di "c" che identifica il formato DOS 2.
5.		
3	00	Riservato per futuri usi del DOS.
4-5		Non utilizzato
6-21		Nome del dischetto
22-23	160	Spazi shiftati
24-25		Numero di ID del dischetto.
26	160	Spazi shiftati
27-28	50-67	ASCII "2c" che identifica la versione.
29-32	160	Spazi shiftati
33-255	00	Non utilizzati

D9060/D9090 Testata Directory -Traccia 0 Settore 0

Byte	Dato	Descrizione
0-1		Puntatore traccia/settore per tracce e settori rovinati.
2-3	00-255	Identifica il formato DOS 3.0
4-5	76-00	Tracci/settore del primo blocco della Directory.
6-7	00-00	Non utilizzati
8-9	01-00	Traccia/settore del primo blocco BAM.

FORMATI BLOCCHI DELLA DIRECTORY

PER TUTTI I DOS

2031 Blocchi Directory -Traccia 18 Settori da 1 a 18

2040 Blocchi Directory -Traccia 18 Settori da 1 a 18

3040 Blocchi Directory -Traccia 18 Settori da 1 a 18

4040 Blocchi Directory -Traccia 18 Settori da 1 a 18

8050 Blocchi Directory -Traccia 39 Settori da 1 a 29

8250 Blocchi Directory -Traccia 39 Settori da 1 a 29

D9060/9090 I Blocchi della Directory il cui inizio e' nella traccia 76, utilizzano tutte le Tracce- Settori utilizzati da 00 a 31. Dopo ci si puo' espandere su blocchi addizionali se necessario, avendo quindi a disposizione una directory illimitata.

FORMATI BLOCCHI DELLA DIRECTORY

PER TUTTI I DOS

1540/1 Blocchi Directpry -Traccia 18 Settori da 1 a 18

2031 Blocchi Directory -Traccia 18 Settori da 1 a 18

2040 Blocchi Directory -Traccia 18 Settori da 1 a 18

3040 Blocchi Directory -Traccia 18 Settori da 1 a 18

4040 Blocchi Directory -Traccia 18 Settori da 1 a 18

8050 Blocchi Directory -Traccia 39 Settori da 1 a 29

8250 Blocchi Directory -Traccia 39 Settori da 1 a 29

D9060/9090 I Blocchi della Directory il cui inizio e' nella traccia 76, utilizzano tutte le Tracce- Settori utilizzati da 00 a 31. Dopo ci si puo' espandere su blocchi addizionali se necessario, avendo quindi a disposizione una directory illimitata.

Byte	Dati	Descrizione
0-1		Puntatore traccia/settore al successivo blocco della Directory.
2		Tipo di file
3-4		Puntatore traccia/settore al primo blocco del file
5-20		Nome del file.
21-22		Traccia/settore del primo SIDE SECTOR se trattasi di file RELATIVE.
23		Lunghezza del record se e' un file RELATIVE.
24-27		Riservato per future informazioni sul file.
28-29		Puntatore traccia/settore per rimpiazzare il file quando si utilizzi la ù.
30-31		Numero di blocchi utilizzati dal file.
32-255		Punti di ingresso degli altri file da 32 Byte s ciascuno con la stessa struttura, formato e contenuti di quello esaminato in precedenza e che va da 2 a 32. Inoltre 2 bytes addizionali non utilizzati.

NOTE al formato dei blocchi per tutte le versioni DOS.

- Tavola VIII -

1- 32 Bytes di File Entry tranne che per il primo di 30.

2- Totale di 8 File entries per blocco.

3- I tipi di file sono:

Files cancellati	$00
Dati sequenziali	$01
Files programmi	$02
Files User	$03
Records relatives	$04

4- Sui codici relativi ai tipi di files visti inprecedenza viene eseguito un OR con $80 quando ilfile e' chiuso in maniera corretta.

5- Il valore 00 relativo alla traccia nel Byte ZERO indica l' ultimo blocco utilizzato nella Directory.
Il valore relativo al settore indica quindi il prossimo Byte da usare.

FORMATO DEI FILES DATI SU DISCO

TUTTE LE VERSIONI DOS

FILES PROGRAMMI

Byte	Descrizione
0-1	Puntatore traccia/settore al prossimo blocco programma.
2-255	Fino a 254 Bytes di programma Basic. La fine del file e' indicata dalla presenza di tre bytes consecutivi con valore $00.

FILES DATI SEQUENZIALI E RELATIVE

Byte	Descrizione
0-1	Puntatore traccia/settore al successivo blocco di dati sequenziali
2-255	Fino a 254 bytes di dati con un ritorno carrello come separatore fra i blocchi di dati.

NOTA

Se il Byte ZERO contiene il valore $00 cio' indica l'ultimo blocco di dati.
Il byte 1, che e' il byte di collegamento settore indichera' allora la posizione prossima per ricevere dati.
La fine di un record RELATIVE e' indicata dalla lettura di $FF.

FORMATO DEL FILE RELATIVE-SIDE SECTOR

Byte	Descrizione
0-1	Puntatore traccia settore al prossimo SIDE SECTOR.
2	Numero di SIDE SECTOR- se e7 una unita' 4040 o 8050 s i tratta di file relative- Con costante $FE-se dos2.7 DOS 3.0 files relatives.
3	Lunghezza del record relative
4-5	Puntatore traccia/settore-Primo Side Sector.
6-7	Puntatore traccia/settore-Secondo Side Sector.
8-9	Puntatore traccia/settore-Terzo Side Sector.
10-11	Puntatore traccia/settore-Quarto Side Sector.
12-13	Puntatore traccia/settore-Quinto Side Sector.
14-15	Puntatore traccia/settore-Sesto Side Sector.
16-255	Puntatore traccia/settore a 120 blocchi dati.

Totale di 720 blocchi per file.

CAPITOLO VENTESIMO

LA PORTA IEEE-488

In questo e nei successivi capitoli, prima di passare all' esame dell' ultima periferica importante, la stampante, esamineremo le possibilita' di collegamento dei nostri computer. Vedremo la porta IEEE-488 e quella seriale, cercando di evidenziare gli schemi dei collegamenti.

Abbiamo ritenuto infatti che le possibilita' di collegamenti in definitiva non siano per nulla meno importanti dell' esame della stampante.
Ecco il motivo perche' questi capitoli sono stati inseriti in questo punto.

Questo capitolo sulla porta IEEE-488 non e' tanto utile agli utenti del VIC 20 e del CBM 64 in quanto questo tipo di porta non esiste su questi computer almeno nella forma standard o comunque molto vicina agli standard.
Sara' molto utile invece ci auguriamo almeno ai possessori di PET e di CBM professionali, benche' anche utenti del CBM 64 sentano spesso la necessita' di collegare strumentazioni o periferiche su questo tipo di uscita, non ultimo vedi il disco 1001 da 1 MB.

Sul CBM64 e sul VIC 20 esistono infatti delle porte IEEE ma di tipo particolare come vedremo nella specifica trattazione a loro riservata.

La porta IEEE-488 e' la principale porta di I/O sulle serie COMMODORE, almeno fino all' uscita degli HOME, ed e' stata messa a punto per consentire ai CBM di connettersi ad una larga gamma di periferiche, dalla stmpante, ai floppy, agli strumenti scientifici.
Il bus IEEE-488 o come e' spesso conosciuto come Hp-IB BUS e' stato sviluppato dai laboratori di ricerca della HEWLETT-PACKARD intorno agli anni '70 per semplificare l' integrazione di strumenti di rilevamento misure con i sistemi di calcolo come i computer.
E' stato anche adottato come BUS internazionale Standard dall' AMERICAN INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEER e gli e' stato, in questa occasione, assegnato il numero di protocollo 488 da qui il nome con il quale e' normalmente conosciuto.
Essendo quindi un BUS universale e standard consente che dovrebbe essere possibile collegare una qualsiasi periferica IEEE-488 ad un' altra qualsiasi periferica IEEE-488.
Questo fatto ha reso disponibile molti fabbricanti in tutto il mondo a mettere a punto prodotti con interfacce IEEE-488.

NOTA

Abbiamo usato il condizionale perche' spesso questo concetto di universalita' non e' applicato da tutti. Vedi appunto il caso COMMODORE dove le funzioni della IEEE-488 non sono state implementate ne' completamente ne' come funzioni standard.

L' altra interfaccia che invece vediamo implementata sul VIC 20 e sul CBM 64, sebbene in

maniera particolare e' la RS232.
In altro capitoli parleremo anche della RS-232 in particolare per le applicazioni sul VIC 20 e sul CBM 64.

L' impiego della IEEE-488 sulla serie COMMODORE e' stato variamente commentato in tutto il mondo.
Tuttavia non e' difficile da costruire un' interfaccia IEEE-488 che possa essere connessa alla RS-232 ed in questo modo le serie COMMODORE possono essere equipaggiate per colloquiare con una serie quasi infinita di periferiche.

LE LINEE DELLA IEEE-488

Le 16 linee attive della porta IEEE derivano la loro funzione dall' integrato 6520 numero 2.
Solo 4 linee sono connesse direttamente ai CHIPS d' interfaccia o bus di controllo processore, mentre le rimanenti sono connesse al sistema tramite 3 linee di buffers bidirezionali.
I BUFFERS bidirezionali sono utilizzati per combinare due linee una di INPUT e una di OUTPUT dai CHIPS periferici di I/O per produrre le linee bidirezionali richieste dal BUS-IEEE.
Dal punto di vista del processore la porta IEEE consiste di 8 linee di INPUT dati e 8 linee di OUTPUT dati piu' 4 HANDSHAKE di OUTPUT e 4 HANDSHAKE INPUT.
Le rimanenti quattro linee di controllo sono unidirezionali.
I CHIPS dei buffers bidirezionali sono periferiche tre stati (TRI-STATE) nello stato non attivato le linee bidirezionali della porta IEEE sono in una stato di alta impedenza.

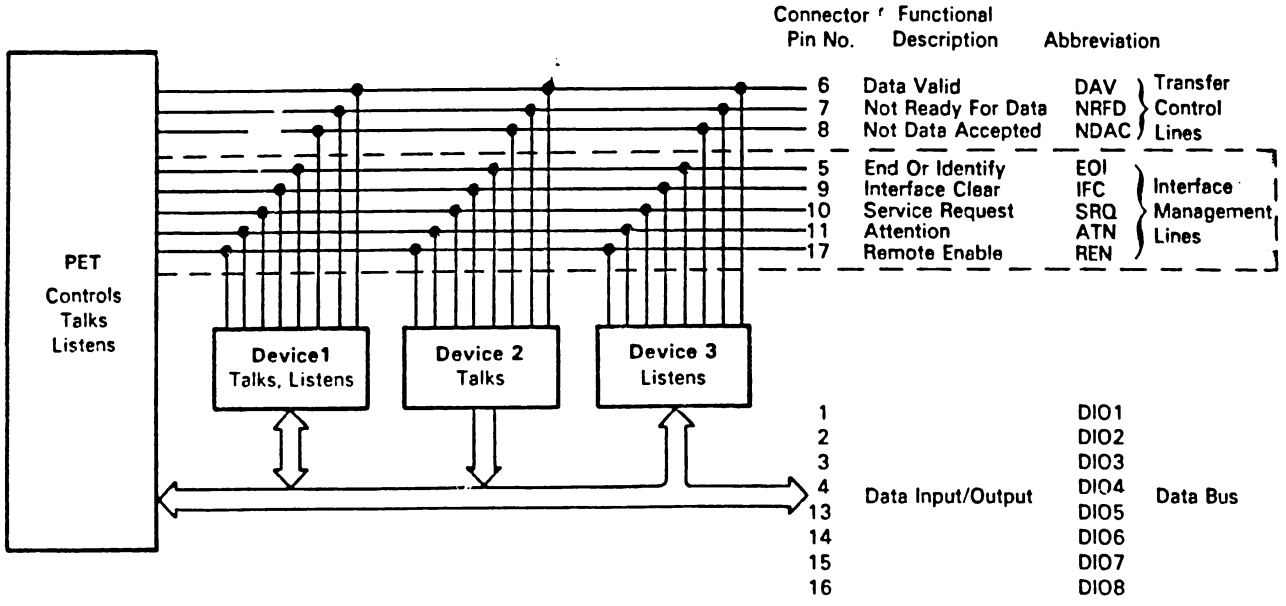


Fig. 13 - Diagramma a blocchi delle connessioni fra CBM e BUS/IEEE-488.

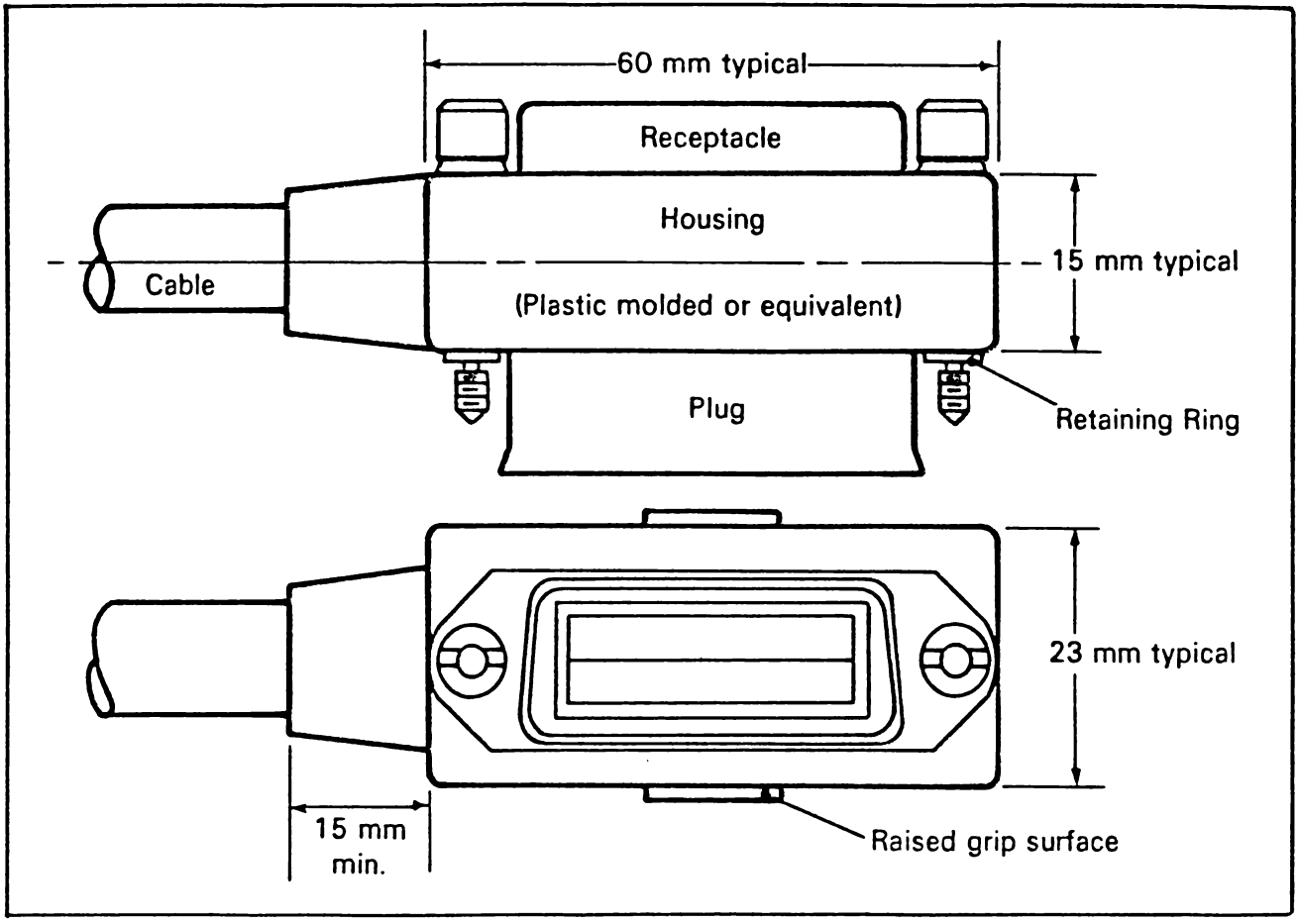


Fig. 14 - Disegno dei doppi connettori maschio/femmina della IEEE.

Questo consente di avere un livello di voltaggio intermedio fra lo stato alto e lo stato basso (HIGH STATE e LOW STATE) consentendo ad ogni periferica di manipolare il bus in uno stato "TRUE" cioe' vero o "1".
Il connettore standard IEEE non e' utilizzato sui COMMODORE.

Dalle tavole allegate si puo' notare inoltre che neppure il connettore utilizzato sulle serie COMMODORE e' standard.

La massima lunghezza di cavo utilizzabile per connettere periferiche sul bus IEEE non dovrebbe superare i 5 metri, mentre la lunghezza del cavo fra l' unita' centrale COMMODORE , che come vediamo e' l' unico che puo' agire da CONTROLLER, e l' ultima periferica sul BUS non dovrebbe essere piu' lunga di 15 metri.
Il grande vantaggio delle porta IEEE e' che si puo' usare per connettere piu' di una periferica al computer (bene inteso che stia funzionando come CONTROLLER di gestione) ed e' per questa ragione che ci si riferisce spesso alla porta come BUS.
Ogni periferica e' identificata sul bus con un determinato numero (appunto il numero di periferica o DEVICE NUMBER).

Il CBM consente all' utente di connettere fino ad un massimo di 15 periferiche sul bus IEEE.
Le periferiche connesse sul bus IEEE devono essere capaci di consentire almeno una delle seguenti funzioni:

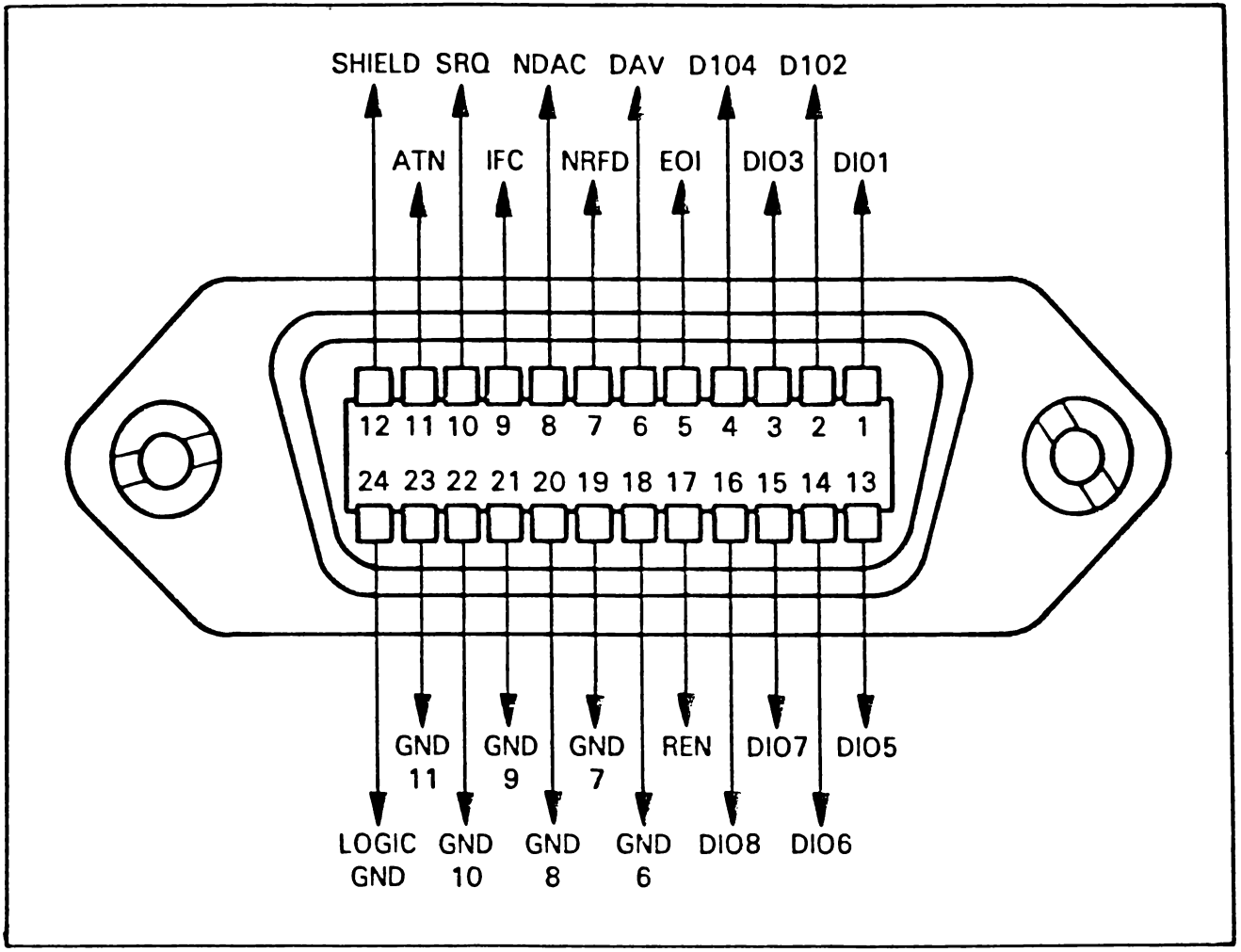


Fig. 15 - Piedinatura del connettore IEEE-488 COMMODORE.

IEEE Bus (GPIB) Contact	PET J1 Contact	GPIB Functional Division	IEEE Function Assignment	IEEE Function Description
1	1	Data Bus	DIO1	Data Input/Output Wire 1
2	2		DIO2	Data Input/Output Wire 2
3	3		DIO3	Data Input/Output Wire 3
4	4		DIO4	Data Input/Output Wire 4
5	5	Interface Management Bus	EOI	End Or Identify
6	6	Transfer Control Bus	DAV	Data Valid
7	7		NRFD	Not Ready For Data
8	8		NDAC	Not Data Accepted (Data not Accepted)
9	9	Interface Management Bus	IFC	Interface Clear
10	10		SRQ	Service Request
11	11		ATN	Attention
12	12		SHIELD	GPIB Cable Shield and Chassis Ground
13	A	Data Bus	DIO5	Data Input/Output Wire 5
14	B		DIO6	Data Input/Output Wire 6
15	C		DIO7	Data Input/Output Wire 7
16	D		DIO8	Data Input/Output Wire 8
17	E	Interface Management Bus	REN	Remote Enable (Always at ground in the PET)
18	F	Grounds	Gnd 6	DAV)
19	H		Gnd 7	NRFD)
20	J		Gnd 8	NDAC } Matching grounds for
21	K		Gnd 9	IFC) these control lines
22	L		Gnd 10	SRQ)
23	M		Gnd 11	ATN)
24	N		Logic Gnd	EOI and REN Gnds

Fig. 16 - Connessioni da eseguire tra GPIB ed il connettore PET J1. Quando il PET è utilizzato come CONTROLLER di periferiche.

LISTENER

Una periferica che e' definita come LISTENER (cioe' in ascolto) deve essere capace di ricevere dati da altre periferichee connesse sul bus.
Il migliore esempio di una periferica che agisce unicamente come LISTENER e' la stampante.

TALKER

Una periferica capace di trasmettere dati ad un' altra periferica sul bus IEEE.
Un esempio di questo potrebbe essere un voltmetro digitale. Altri potrebbero essere un contatore oppure un lettore di nastro su carta.

CONTROLLER

Una periferica che manipola comunicazioni sul BUS IEEE come periferiche di indirizzi e invia comandi.
Il CBM e' la sola periferica che puo' agire come CONTROLLER.
Da cio' e' chiaro che la periferica in funzione di CONTROLLER puo' agire sia come TALKNER che come LISTENER.

Sebbene fino a 15 periferiche possano essere inserite sul BUS IEEE, solo una periferica per volta puo' agire come TALKER.
Tutte le altre periferiche possono agire come LISTENERS consentendo che i dati possano essere inseriti a piu' di una periferica per volta.

Le 16 linee di segnali del BUS IEEE possono essere divise in 3 gruppi.

IL DATA TRASMISSION BUS o bus di trasmissione dati.

IL TRANSFER BUS o bus di trasferimento dati.

IL MANAGEMENT BUS o bus di manipolazione.

Le restanti 8 linee sul connettore ad 8 linee sono GROUNDS, cioe' a terra.

Il DATA BUS consiste di 8 linee bidirezionali per la trasmissione di segnali di dati in modo parallelo.
I segnali sono attivi bassi (ACTIVE LOW) ed il bit piu' significativo e' sulla linea D 108.
I dati sono trasmessi un BYTE per volta con un codice ASCII a 7 bits e con l' ottavo bit disponibile per il controllo di parita'.
La trasmissione dei dati viene effettuata alla velocita' della periferica piu' lenta presente sul BUS e che e' attiva in quel particolare momento.
Sebbene la massima velocita' di trasferimento dati su un BUS IEEE sia di circa un milione di bytes per secondo, sui prodotti Commodore questo e' limitato dalla velocita' del processore.
Su queste macchine avremo quindi una velocita' di trasmissione di circa 5000 Bytes al secondo che il BASIC a sua volta riduce a circa 100 Bytes al secondo.
Il BUS DATI e' anche utilizzato per trasmettere indirizzi alle periferiche.

NOTA

E' evidente che quindi, utilizzando il Linguaggio Macchina, possiamo avere una velocita' di trasmissione molto piu' elevata.
Ecco uno dei motivi di questo capitolo, come di quello che si riferisce alla RS-232 e della spiegazione delle routine Kernal oltre alle tavole.

Questi sono indirizzi utilizzati per consentire che una a periferica con il BUS, cioe' sia accessibile al BUS.

La differenza fra dati e informazioni di controllo tipo quelle che abbiamo visto sta nel fatto che durante la trasmissione la linea ATN viene mantenuta bassa. Questo durante la trasmissione delle informazioni di controllo.
Il BUS di trasferimento consiste di 3 linee utilizzate per controllare il trasferimento di dati sul BUS DATI.
Allo stesso modo che sulle linee di dati, questi segnali sono attivi quando sono bassi.

La funzione delle linee sul BUS di trasferimento possono essere date in sommario come segue:

DAV - Data valid

Quando questa linea e' bassa, il segnale sara' un dato valido sul bus dati.

NRFD - Not ready for data

Questa linea e' tenuta bassa per tutto il tempo in cui una o piu' periferiche sul BUS IEEE definite come LISTENER cioe' come pronte per accettare dati non sono invece nella condizione di accettare questi dati.
Non appena tutte le periferiche sono pronte, cioe' non esiste nemmeno una periferica non disponibile ad accettare dati, allora la linea NFRD viene posta alta.

NDAC - Data not accepted

Questa linea e' tenuta bassa da una periferica in ascolto mentre sta leggendo dati.
Non appena i dati sono stati letti, la periferica in ascolto pone NDAC alta in modo tale da segnalare alla periferica che stava trasmettendo (cioe' quella in TALKER), che i dati sono stati accettati.

Poiche' i dati sono trasferiti sul BUS della IEEE in modo asincrono, la funzione delle tre linee del BUS di trasferimento e' di trattare o controllare il trasferimento di dati fra l' unita' TALKER e la LISTENER cioe' fra l' unita' di trasmissione e quella in ascolto.
La sequenza di temporizzazione per la manipolazione dei dati oppure chiamata anche sequenza di HANDSHAKE o anche HANDSHAKING SEQUENCE e' molto importante ma esula dagli scopi non specialistici di questo volume per cui si consiglia di leggere per chi desideri approfondire, trattazioni specifiche sull' argomento.

In linea di massima possiamo dire che il termine

HANDSHAKING e' utilizzato per descrivere metodi che assicurino la sincronizzazione di impulsi di Input o di Output tra il computer ed una periferica esterna.

Vediamo ora le 5 linee del management BUS che hanno il compito di inviare comandi alle periferiche e di controllare lo stato corrente del DATA BUS.
Come al solito spiegheremo, sebbene in sommario, le funzioni di queste linee.

ATN - Attention

Questa linea e' messa bassa dal controller quando sta inviando comandi e indirizzi periferici sul BUS DATI.
Non appena la linea ATN va alta la periferica preventivamente selezionata puo' trasferire dati tra essa stessa ed il CONTROLLER.

EOI - End of identify

Questa linea e' messa bassa dalla periferica in TALKER mentre l' ultimo Byte di dati sta per essere trasferito e indica quindi alla LISTENER che siamo alla fine del messaggio.

IFC - Interface clear

Sui Commodore la linea IFC ovvero di pulizia dei dati contenuti nell' interfaccia e' collegata al RESET di sistema in modo tale che quando l' unita' centrale viene accesa questa linea va bassa per

circa 100 millisecondi.
Mettendo bassa la linea di IFC tutte le periferiche connesse sul BUS IEEE sono quindi inizializzate in maniera ottimale.

SRQ - Service request

Alcune periferiche connesse al bus IEEE hanno la capacita' di REQUEST SERVICE dal controller, cosa che viene ottenuta mettendo bassa la linea SRQ.
Questa linea tuttavia non e' implementata dal Basic sul CBM ma e' connessa all' Input CB1 sulla 6520#2 e puo' essere utilizzata scrivendo una subroutine in codice macchina per controllare lo stato di questa linea come parte del processo di scansione dell' INTERRUPT di tastiera che ricordiamo e' di 60 Hz.
Se piu' di una periferica collegata al BUS IEEE e' in grado di eseguire questa funzione, cioe' di mettere bassa la linea di SRQ allora il Controller, che ricordiamo in questo caso essere la sola unita' centrale, deve controllare le periferiche per trovare quale di queste ha fatto la richiesta di servizio.
Il CONTROLLER eseguira' questa verifica trasmettendo un comando in modo seriale con un codice esadecimale 18.
Ogni periferica viene controllata mettendo in funzione la linea ATN, indirizzando la periferica come TALKNER e rimuovendo poi l' ATN.
Se viene incontrata la periferica che aveva selezionato la SRQ allora questa rispondera' mettendo bassa la linea di dati 7.
Questo metodo di controllo seriale viene quindi disabilitato dal CONTROLLER stesso con l' invio di un comando esadecimale 19.

REN - Remote enable

Questa linea e' tenuta bassa dal CBM e' non e' disponibile per il controllo utente.

Il CBM essendo il solo CONTROLLER attivo sul BUS manipola tutte le comunicazioni fra periferiche. Cio' viene fatto inviando comandi a queste periferiche tramite le linee di dati.
I comandi vengono distinti dai dati dallo stato della linea ATN.

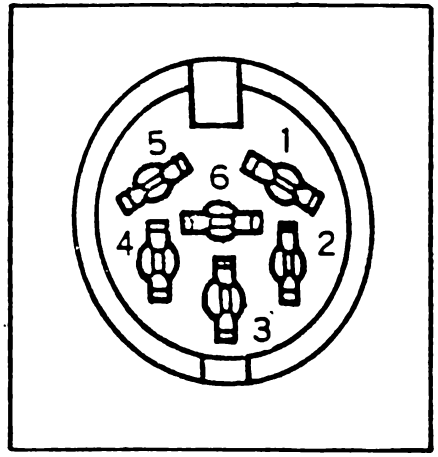
Quando la linea ATN e' bassa allora il BUS DATI e' in modo comando, il controller e' la sola periferica attiva mentre tutte le altre periferiche stanno attendendo istruzioni.
Questi comandi sono eseguiti automaticamente dal Sistema Operativo del CBM quando il Bus IEEE e' utilizzato con il Basic.
La conoscenza di questi comandi e' necessaria invece se il Bus deve essere controllato e in parte utilizzato in maniera migliore con l' impiego del linguaggio macchina. Ecco perche' abbiamo ravvisatola necessita' di una pur breve spiegazione del dettaglio di questi comandi.

Il piu' semplice gruppo di comandi e':

ADDRESS

UNADDRESS

di cui ne esistono 4:



PIN#	TYPE
1	SERIAL SRQ IN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	NC

Fig. 17 - Piedinatura del connettore IEEE seriale.

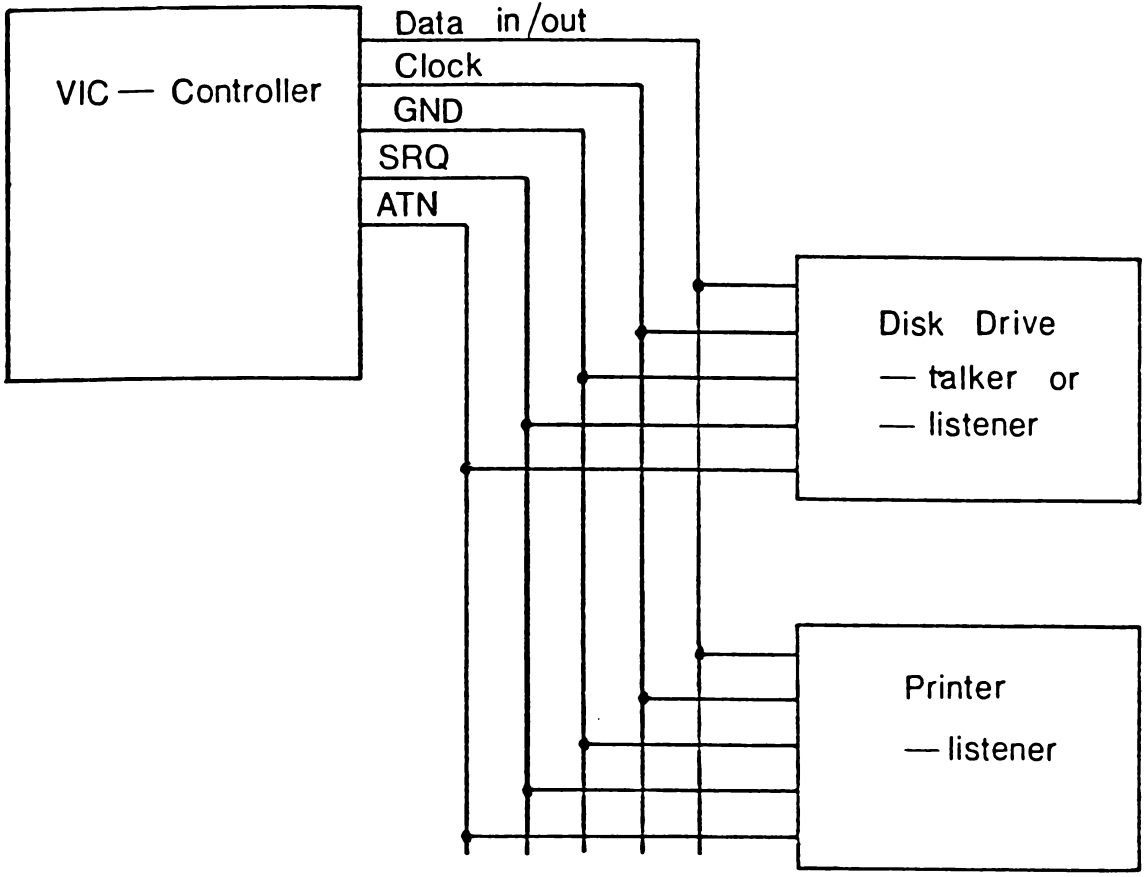


Fig. 18 - Schema di collegamento delle periferiche sul BUS IEEE.

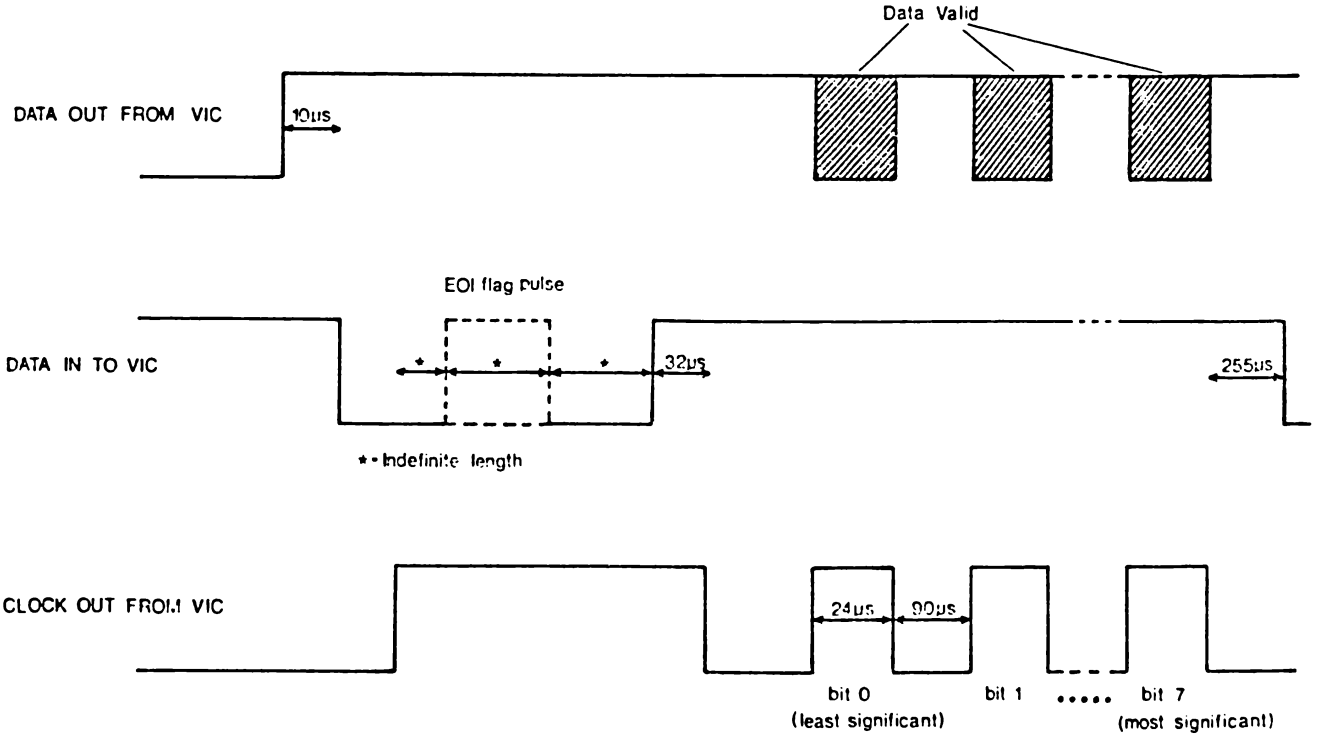


Fig. 19 - Forma d'onda dei dati e linee di controllo sulla porta IEEE del VIC.

TALKER ADDRESS

LISTENER ADDRESS

UNLISTEN ADDRESS

UNTALK ADDRESS

Il TALKER ADDRESS o indirizzo di chiamata e' trasmesso come un codice di 7 bits e abilita una data periferica a parlare, cioe' a trasmettere.
Ora poiche' una sola periferica alla volta puo' agire come TALKER, cioe' puo' parlare, e' evidente che questo comando dovra' anche automaticamente disabilitare le altre.

Da un punto di vista funzionale il TALKER ADDRESS e' assimilabile, almeno come concetto, al numero di periferica (DEVICE NUMBER) utilizzato normalmente nel Basic. Ma mentre un DEVICE NUMBER puo' essere un numero qualsiasi fra 1 e 30, il TALKER ADDRESS e' un numero qualsiasi di un gruppo di 31 caratteri ASCII di un BYTE da 7 bits che e' definito come appunto TALK ADDRESS dal fatto di avere il bit sei = 0 e il bit sette = 1.
Ogni periferica ha il suo stesso e unico TALK ADDRESS che puo' essere fissato dall' utente e sara' quindi utilizzato dal software di controllo per selezionare proprio quella periferica.

Anche il LISTENER ADDRESS o indirizzo di ascolto e' trasmesso come codice di 7 bits ed abilita quindi una scelta periferica ad agire in funzione di unita' di ricezione o di ascolto.
Anche per il LISTENER ADDRESS valgono le considerazioni e la tecnica di trasmissione riportatat in precedenza solo che in questo caso

avremo invece il sesto bit = 1 ed il settimo bit = 0.

NOTA

E' importante notare che in Basic la differenza fra periferica in TALKER o in LISTENER cioe' in trasmissione o in ascolto, e' determinata dal contenuto dell' indirizzo secondario che poi il sistema operativo del computer tradurra' opportunamente nei contenuti dei bits 6 e 7 mettendoli nella giusta segquenza, come visto prima, in funzione della necessita' di trsamettere o ricevere dati.

Quando una periferica puo' agire sia come TALKER che come LISTENER, naturalmente le verranno assegnati indirizzi identici tranne che per i contenuti dei bits 6 e 7 visti.
Per questo una periferica selezionata come una unita' di ascolto o LISTENER dal carattere ASCII "&" avra' un indirizzo di trasmissione o TALKER ADDRESS selezionabile dal carattere "F".

NOTA

Via HARDWARE e' sempre possibile, ma sconsigliabile ai non esperti, cambiare gli indirizzi visti in precedenza spostando degli interruttori interni alla periferica stessa.

Una periferica posta in TALK o in LISTEN puo'

essere resettata da un comando UNADDRESS.
Il comando UNLISTEN che equivale nella forma esadecimale a $3F esegue il CLEAR di tutto il Bus per TUTTE le unita' che fossero state poste in LISTENERS.
Un comando di UNTALK che e' equivalente a $5F disabilita' l' attuale periferica selezionata come TALKER.

NOTA

Questo effetto puo' essere ottenuto anche selezionando un' indirizzo non utilizzato.

Una periferica non necessita di essere indirizzata per rispondere ad un gruppo di comandi conosciuto come comandi universali e tutte le periferiche risponderanno ad uno di questi comandi provenienti dal controller sia che siano selezionate nelle forme precedenti o che non siano selezionate.
Vediamo ora in sommario questi 5 comandi universali e le loro funzioni e forme di utilizzo.

DCL - Device Clear

Questo comando restituisce a tutte le periferiche presenti sul Bus IEEE la capacita' di rispondere ad un dato ordine sia che siano state selezionate, cioe' indirizzate, o no. Del resto anche la semplice traduzione letterale del comando ne rende evidente la funzione.

SPE - Serial poll enable

Abilita il modo seriale di richiesta sul Bus.
E' utilizzato solo quando la linea SRQ, vista in precedenza, e' implementata sul CBM.
In pratica questo comando consente al controller di trovare quale periferica abbia generato la SERVICE REQUEST.

SPD - Serial poll disable

La ricerca da parte del controller vista appena sopra viene disabiltata con questo comando.

LLO - Local lockout

Su alcune periferiche, in particolare sulle stampanti, e' presente sul pannello frontale un pulsante di RESET che puo' essere disabilitato, naturalmente quando la periferica e' in fase di LISTENER, con questo comando.

PPU - Parallel poll unconfigured

Questo comando consente al Controller di effettuare un' indagine CONTINUA e parallela sulle periferiche che abbiano effettuato il SERVICE REQUEST.
Purtroppo non e' disponibile direttamente sull' interprete Basic dei CBM.

TAVOLA DEGLI EQUIVALENTI ESA DEI COMANDI

COMANDO	VALORE ESA
DCL	14
SPE	18
SPD	19
LLO	11
PPU	15

Altri comandi specifici per una qualsiasi particolare periferica possono essere dati sul bus IEEE.
Questi sono i comandi di indirizzo secondario utilizzati nell' OPEN per comunicare ad una periferica intelligente di funzionare in un numero di diversi modi.
La forma e la natura di un comando di indirizzo secondario dato sia in Basic che utilizzando il linguaggio macchina dipende esclusivamente dalla periferica alla quale si fa riferimento.
Infatti ogni periferica ha una serie di indirizzi secondari che potremo definire convenzionali e che possono essere conosciuti solo utilizzando il manuale di quella particolare periferica.
Cio' e' in parte il motivo perche' nel presente maunale abbiamo cercato di riportare gli indirizzi di quante piu' periferiche in uso presso gli utenti CBM era possibile.

Come abbiamo piu' volte visto un indirizzo secondario in Basic puo' avere un valore compreso in un' intervallo fra 0 e 31.

Il comando OPEN nel Basic e' utilizzato per selezionare una periferica sul bus IEEE che abbia un numero fra 4 e 30.
Infatti ricordiamo che se il numero di periferica e' inferiore a 4 il Sistema Operativo indirizzera' la tastiera, la cassetta o il video.
Viene anche inizializzato il Sistema Operativo stesso, cioe' viene messo in grado di eseguire determinate operazioni, in modo tale che la periferica potra' comuicare con un particolare file logico identificato da un numero fra 1 e 255.

L' utilizzo dell' indirizzo secondario e del nome del file e' opzionale, tuttavia un indirizzo secondario e' inviato solo se viene utilizzato un FILE NAME.
Infatti, in questo caso,il Sistema Operativo inviera' un comando di LISTEN ad una data periferica seguito dall' indirizzo secondario.
Se dalla periferica non c' e' nessuna risposta al comando ATN, il Sistema Operativo rispondera' con un messaggio di DEVICE NOT PRESENT e mettera' ad 1 il bit 7 della STATUS WORD (ST).
Dopo aver inizializzato il Sistema Operativo, il sistema stesso di comunicazione, specificato la periferica che si intende utilizzare e scelto l' indirizzo per la particolare funzione che si desidera eseguire, sara' possibile trasferire i dati sul bus IEEE utilizzando i comandi INPUT#, PRINT# e GET#.
Quando uno di questi comandi viene incontrato in un programma il Sistema Operativo chiamera' in funzione la routine di INPUT della IEEE-488.
I comandi di INPUT# e GET# specificano un particolare numero di file logico, la routine di input invia un comando di TALK alla periferica dichiarata nell' OPEN per quel particolare file logico, fissando percio' la periferica indirizzata

come TALKER DEVICE e l' unita' centrale come LISTENER.

L' unita' centrale attende quindi che la linea DAV venga messa bassa cosa che gli indichera' che la periferica in TALKER ha messo un byte di dati valido sul bus IEEE.
Un input sulla linea DAV deve essere ricevuto entro 64 millisecondi se quel Byte di dati deve esere immesso nel Buffer di input del Basic.
Se il segnale non e' ricevuto nel periodo indicato allora la sequenza di input della IEEE avra' termine e la routine di manipolazione di errore fissera' il Byte di Status della variabile ST del Basic, (gia' vista in precedenza) a 2, il che stara' ad indicare un fuori tempo (TIME OUT) della periferica in TALKER.
Il Byte di status o ST e' immagazzinato in diverse locazioni a secondo della unita' centrale che si sta utilizzando.

SPINOTTI DEL BUS SERIALE

PIN	DESCRIZIONE
1	Ingresso SRQ seriale
2	GND
3	Ingresso/Uscita ATN seriale
4	Ingresso/Uscita CLK seriale
5	Ingresso/Uscita dati seriali
6	Nessuna connessione

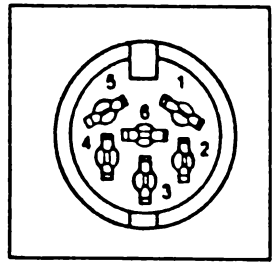


Fig. 20 - Spinotti del bus seriale.

Se il comando Basic era INPUT # dopo essere andato a cercare un carattere e dopo averlo immesso nel Buffer di Input, la routine di input della IEEE e' chiamata ancora in funzione e viene immesso un altro carattere.

Questo processo continua fino a quando la routine di Input identifica un livello basso di segnale sulla linea di EOI che sta ad indicare la fine del trasferimento di dati.

NOTA

Non tutte le periferiche generano un segnale di EOI.

Non appena viene sentito un' impulso sulla EOI il Sistema Operativo mettera' a 1 il bit 6 dello Status Byte e forzera' un ritorno carrello entro il Buffer fino a quando l' attuale comando non sia terminato.

Il comando di INPUT # e' limitato nella sua capacita' operativa dalla lunghezza del Buffer di Input che previene il trasferimento di piu' di 80 caratteri per volta a meno che un Ritorno Carrello separi ogni blocco di 80 caratteri.
Ogni tentativo di scrivere piu' di 80 caratteri nel Buffer di Input causera' un malfunzionamento di sistema.
Se la periferica collegata sul bus IEEE invia piu' di 80 caratteri senza un ritorno carrello fra i blocchi, allora deve essere utilizzato un comando di GET #.
Questo comando infatti chiama in esecuzione la

routine di Input della IEEE di volta in volta che viene incontrato nel programma Basic e quindi l' invio di dati viene fatto uno alla volta.

Con l' utilizzo di questo comando, cioe' del GET #, si supera la limitazione dovuta alla dimensione del Buffer. Purtroppo il comando e' di lenta esecuzione.
Al termine di un comando di Input, sia che esso sia stato un INPUT # che un GET # viene eseguito un salto, cioe' viene chiamata in funzione la:

TERMINATION ROUTINE IEEE

cioe' la routine di fine esecuzione di operazioni sulla IEEE. Questa riportera' in una determinata locazione di memoria situata in pagina 0-4 e che anche in questo caso ovviamente varia da unita' centrale ad un' altra, ma che comunque e' la locazione che contiene il numero della periferica e che potrete trovare sulle tavole, il valore 0 (ZERO) restituendo quindi in Input la periferica tastiera.
Viene poi inviato un comando di UNTALK sul Bus IEEE rendendo il canale quindi libero perche' possa far passare un successivo comando.

Dopo aver aperto un file logico su una data periferica l' unita' centrale puo' ricevere dati da quella periferica con un comando di PRINT # che chiama in funzione la subroutine OUTPUT della IEEE. Questa subroutine fissa la periferica specificata dal File logico nel comando PRINT in modo LISTENER. Quindi il Sistema Operativo cambiera' ilnumero di periferiche in Output dal 3 che e' il video alla periferica che e' stata scelta in funzione di LISTENER sul Bus IEEE.

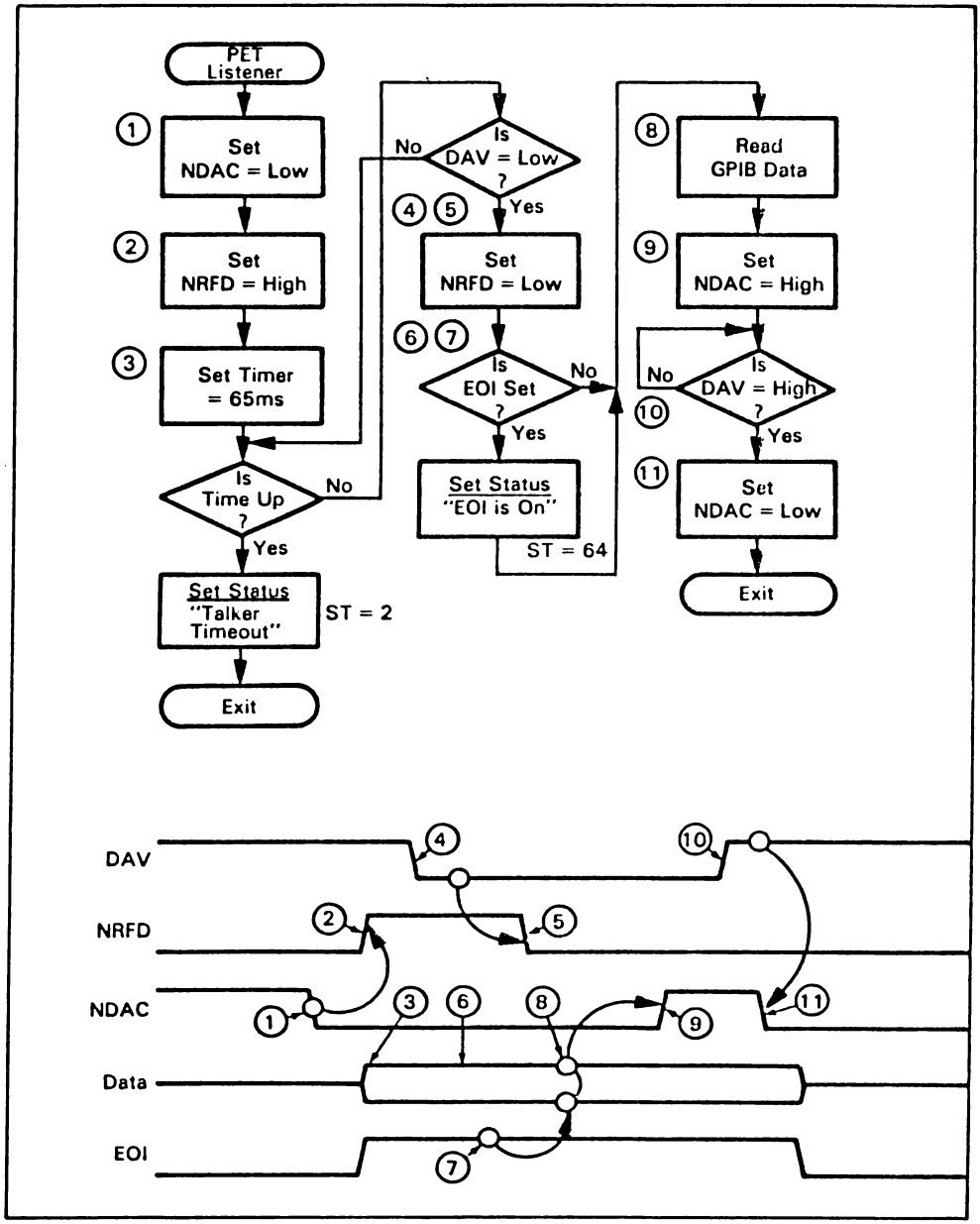


Fig. 21 - CBM COME CONTROLLER/LISTENER
FLOW CHART e diagramma di temporizzazione della funzione HANDSHAKE.

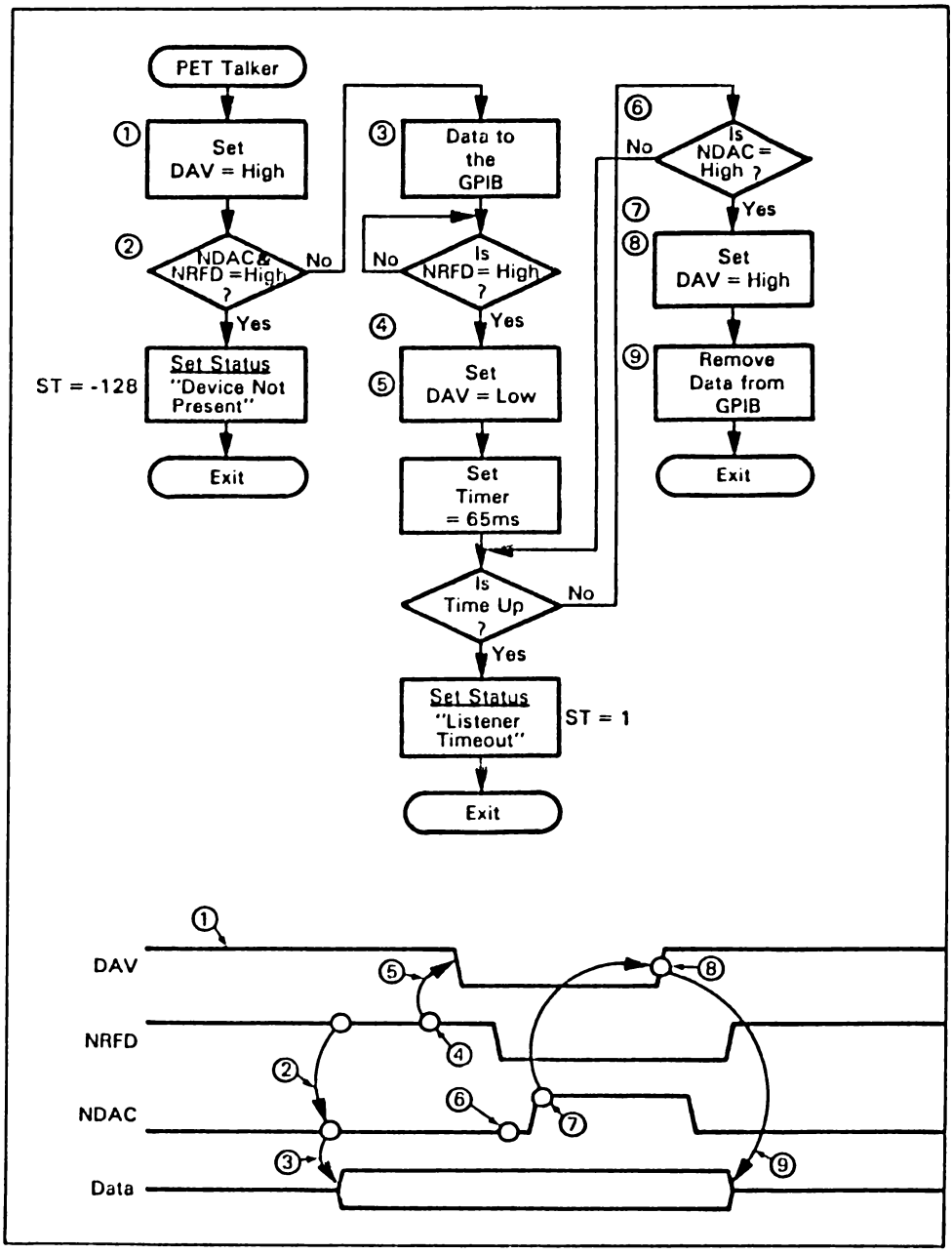


Fig. 22 - COM COMC CONTROLLER/TALKER
FLOW CHART e diagramma di temporizzazione della funzione HANDSHAKE.

Il Basic puo' ora trasferire i dati, un carattere per volta alla routine di IEEE che attende che la linea NFRD vada bassa, cosa che stara' ad indicare che le periferiche in LISTENING sul Bus sono pronte per accettare dati.
Un singolo Byte di dati e' quindi immesso sul Bus IEEE e viene generato un impulso DAV per indicare che sono disponibili dei dati validi sul Bus stesso.
La routine di Output della IEEE attende quindi che la linea NDAC vada alta a dimostrazione che i dati sono stati ricevuti dalla unita' in LISTENER.
Tuttavia anche in questo se l' impulso NDAC non e' ricevuto entro 64 millisecondi, la linea NFRD viene posta bassa e viene generato un' errore.
Il bit 0 dello Status Byte viene messo a 1 per indicare che la periferica in LISTENER e in fuori tempo (TIME OUT).
Quando tutti i caratteri sono stati trasferiti dal programma il Sistema Operativo passa il controllo, fa cioe' intervenire, la routine di END della IEEE che invia un impulso di EOI con l' ultimo carattere immagazzinato nel Buffer di uscita.
Dopo aver eseguita questa operazione viene mandato un comando di UNLISTEN in modo tale che il Bus venga reso disponibile per una successiva operazione e la linea delle periferiche sia resettata cosi' da avere come periferica in uscita di nuovo il video che, ripetiamo, e' la periferca di DEFAULT o normale, allo stesso modo che la tastiera e' la periferica normale di ingresso dati.

Dopo aver terminato tutti gli INPUT e gli OUTPUT, cioe' tutti gli ingressi e le uscite di dati tra un file logico nell' unita' centrale (CONTROLLER) ed una o piu' periferiche connesse al Bus IEEE il file relativo ad ogni periferica deve essere chiuso.

Questa operazione viene effettuata con un comando CLOSE.
Ad esempio un comando CLOSE 5 chiudera' il canale con la periferica associata con il file logico 5 aperto dal comando OPEN.
Al ricevimento di un CLOSE il Sistema Operativo inviera' un comando LISTEN a quella scelta periferica seguito da un comando di indirizzo secondario che segnalera' a quella periferica di arrestare la funzione di I/O dati appena eseguita e di riportarsi allo stato che definiremo di inizializzazione.

CAPITOLO VENTUNESIMO

LA PORTA SERIALE IEEE

Come abbiamo visto nel capitolo precedente di norma la porta IEEE-488 e' una porta parallela.
I dati cioe' che passano attraverso questa porta o come abbiamo piu' spesso sottolineato, vengono inviati sul BUS IEEE, lo sono in forma parallela.
Tuttavia sul VIC 20 e sul CBM 64 questo tipo di uscita e' stata implementata come porta seriale.

La differenza e' costituita dal fatto che mentre sul bus dati di una porta standard i dati stessi sono trasferiti come sequenza di 8 dati sulle macchine menzionate la trasmisione avviene con una linea di dati seriali.
Il bus IEEE del VIC e del 64 e' formato da sei linee, di cui 3 di Input e tre di Output.
Le tre linee di INPUT naturalmente controllano le linee di invio dati da una periferica all' unita' centrale,mentre le tre linee di OUTPUT hanno la stessa funzione per i dati che passano dall' unita' centrale alle relative periferiche.

Queste tre linee, da entrambe le parti o tipo di funzioni le si voglia guardare sono formate da :

-Una linea per l' invio o il ricevimento di dati informa seriale.

-Una linea di CLOCK che serve appunto per la temporizzazione dei dati che passano sul BUS

seriale.

-Una linea di SERVICE REQUEST o di ATTENTION.

Quindi la funzione della porta seriale IEEE e' solo approssimativamente confrontabile con la stessa uscita implementata sulle serie CBM maggiori che abbiamo visto in precedenza, ma e' stata modificata per numerose applicazioni che consentono l' interfacciamento e la comunicazione fra il VIC 20 e il CBM64 e qualsisasi altro tipo di periferica che possono essere sia unita' centrali dello stesso tipo che computer piu' grossi.
Nel caso venga richiesta, cioe' ci sia la necessita' di una completa interfaccia IEEE-488 standard sara' necessario utilizzare il modulo di espansione :

IEEE-488 EXPANSION MODULE

che, dato la grande diffusione dei computer COMMODORE dovrebbe essere reperibile normalmente e senza grandi difficolta' presso un qualsiasi rivenditore.
Questo modulo consentira' quindi di collegarsi a tutte le periferiche delle serie maggiori della COMMODORE che utilizzano questo tipo di uscita ma nella forma parallela.

Una porta IEEE-488 o un BUS di dati, sianella semplice versione seriale, dovuta in gran parte al software del Sistema Operativo, sia nella piena implementazione consentita o sulle serie maggiori o su queste macchine, ma con il modulo di espansione, presenta come abbiamo gia' avuto occasione di

accennare in precedenza, notevoli vantaggi operativi rispetto sia ad un' uscita seriale RS-232 che ad altri tipi di uscita parallela.

Come gia' detto il grandissimo vantaggio e' costituito dal fatto che sul BUS della IEEE-488 sia ha la possibilita' di collegarsi a piu' periferiche contemporaneamente attraverso un unico gruppo di linee di Input/Output.
Anche in questo caso, cioe' nel caso di una uscita seriale anziche' parallela ci sono 3 tipi di periferiiche che possono essere collegate sul bus IEEE:

-CONTROLLER cioe' quel tipo di unita' che controlla il movimento dei dati.

-LISTENER cioe' quel tipo di periferiche che sono in ricezione di dati sul BUS.

-TALKER cioe' quel tipo di periferiche che trasmettono dati sul BUS.

Anche nel caso dell' implementazione seriale solo il computer, cioe' l' unita' centrale costituita dal VIC 20 o dal CBM64, posono agire come CONTROLLER, anche se in caso di collegamento fra piu' unita' centrali si puo' far agire solo come LISTENER o TALKER.
Tutte le periferiche possono essere rispettivamente in posizione di LISTENER o di TALKER, cioe' in ascolto o in trasmissione di dati.
Ovviamente per esempio una stampante potra', anche in questo caso essere solo in posizione di ricezione dati.

CONNESSIONI DELLA PORTA SERIALE

Le sei linee di I/O della porta seriale IEEE sono collegate ai due integrati 6522 del VIC 20.
La seguente tavola mostra le funzioni ed i collegamenti:

VIA No	LINEA No	FUNZIONI DELLA LINEA
VIA 1	PA 1	DATI INPUT
VIA 2	CB 2	DATI OUTPUT
VIA 1	PA 0	CLOCK INPUT
VIA 2	CA 2	CLOCK OUTPUT
VIA 1	PA 7	ATTENTION OUT
VIA 2	CB 1	SERVICE REQ. IN

E' da notare che la linea ATTENTION IN e' semplicemente connessa al PIN 9 del connettore ma non e' implementata.
Nel caso venga richiesta la funzione menzionata il PIN 9 dovra' essere collegata ad una linea di HANDSHAKE non utilizzata dalla USER PORT e dovra' essere scritto un programma per implementare questa funzione.

Quando viene utilizzata la porta IEEE sul VIC 20 o sul CBM64 sia questo nella versione standard che nella espansione esterna con il modulo, la sintassi dei comandi e le relative funzioni sono identiche a quelle viste in precedenza in quanto la differenza e' solo sul modo di trasmettere dati.

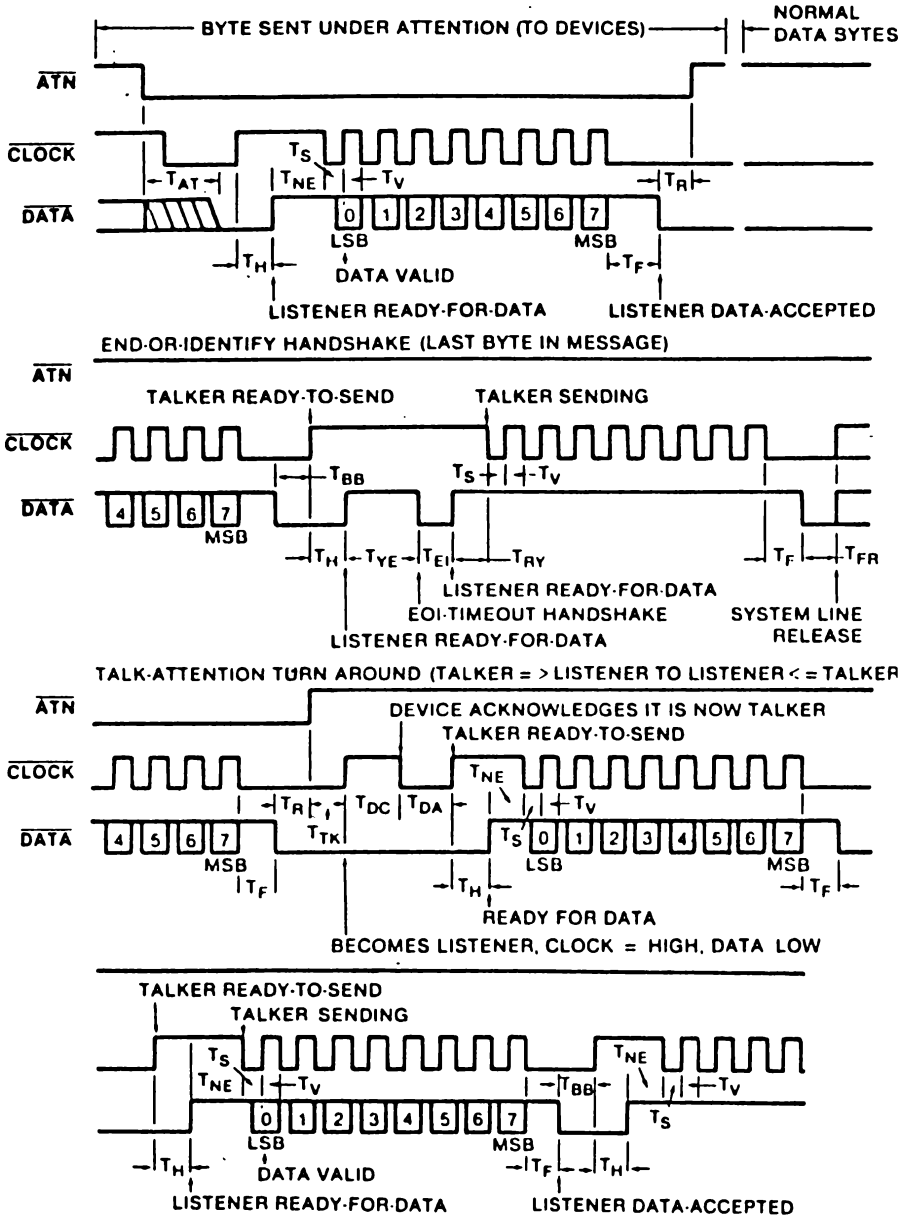


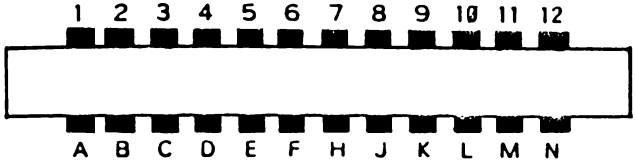
Fig. 23 - Tempi di risposta del bus seriale.

DESCRIZIONE	SIMBOLO	MIN	MED	MAX
Risposta ATN (richiesta) (1)	Tat	-	-	1000 ns
Predisposizione Ricevente	Th	0	-	**
Risposta non-EOI a RFD (2)	Tne	-	40 ns	200 ns
Predisposizione Trasmittente	Ts	20 ns	70 ns	-
Convalida dati	Tv	20 ns	20 ns	-
Handshake pacchetto (3)	Tf	0	20 ns	1000 ns
Pacchetto di ATN da rilasciare	Tr	20 ns	-	-
Intervallo fra due byte	Tbb	-	-	-
Risposta di EOI	Tye	200 ns	250 ns	-
Trattenimento risposta di EOI	Tei	60 ns	-	-
Limite di risposta del trasmittente	Try	0	30 ns	60 ns
Riconoscimento del byte	Tpr	20 ns	30 ns	-

Note:

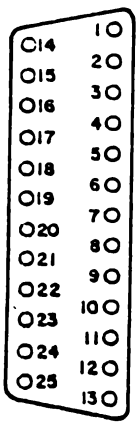
1. Se si supera il massimo tempo consentito, errore di dispositivo non presente.
2. Se si supera il massimo tempo consentito, richiesta di risposta EOI.
3. Se si supera il massimo tempo consentito, errore di pacchetto.
4. Affinche' un dispositivo esterno sia assunto come trasmittente, il valore minimo di Tv e Tpr deve essere 60 ns.

Fig. 24 - Tempi di risposta del bus seriale.



PIN	TYPE	NOTE	PIN	TYPE	RS232 FUNCTION
1	GND		A	GND	
2	+5V	100mA MAX	B	CB1	
3	RESET		C	PB0	⊥ Sin
4	JOY3		D	PB1	— RTS
5	JOY1		E	PB2	— DTR
6	JOY2		F	PB3	— RI
7	LIGHT PEN		H	PB4	— DCD
8	CASSETTE SWITCH		J	PB5	
9	SERIAL ATN IN		K	PB6	— CTS
10	+9V	120mA MAX	L	PB7	-- DSR
11	GND		M	CB2	— Sout
12	GND		N	GND	— GND

Fig.25 - Schema connettore VIC-RS232 e relativi PIN



PIN		
1	Protective Ground	AA
2	Transmitted Data	BA
3	Received Data	BB
4	Request To Send	CA
5	Clear To Send	CB
6	Data Set Ready	CC
7	Signal Ground	AB
8	Carrier Detect	CF
9	(not used)	
10	"	
11	"	
12	"	
13	"	
14	"	
15	"	
16	"	
17	"	
18	"	
19	"	
20	Data Terminal Ready	CD
21	(not used)	
22	"	
23	"	
24	"	
25	"	

Fig.26 - Connettore standard RS-232 e linee di codifica EIA.

CAPITOLO VENTIDUESIMO

USCITA SERIALE RS-232

Il VIC 20 ed il CBM 64 sono in grado di comunicare con unita' periferiche siano esse stampanti, modem ed altro usando una porta di cominicazioni seriali conosciuta come RS 232 Input/Output port.

Allo stesso modo che abbiamo visto per la IEEE, il nome RS-232 si riferisce semplicemente ad uno standard di un prodotto industriale, fra l' altro molto diffuso, messo a punto per comunicazioni seriali con periferiche di computer.

Una porta seriale di I/O e' composta da un minimo di tre linee:

---Una linea di OUTPUT o di trasmissione dati.

---Una linea di INPUT o di ricezione dati.

---Una linea di massa.

I dati sono trasmessi o ricevuti come treno o sequenza d'impulsi.
Ad esempio un singolo Byte diventa una stringa o un insieme di 8 impulsi.

Sebbene una porta seriale possa funzionare con

appena tre linee spesso sono utilizzate altre linee per controllare e trasferire informazioni o dati.

Il VIC ed il 64 sono in grado di ricevere segnali di controllo.
Quando viene utilizzata la RS-232, e vedremo poi il perche' di questo discorso, tutte le linee sono connesse alla porta B del VIA numero 1, le stesse linee usate per la user port.
Normalmente una interfaccia RS-232 sara' utilizzata per connettere fra loro una porta parallela ed un connettore standard TS-232. L' interfaccia provvedera' anche al Buffer necessario ed all' HIGHER DRIVE VOLTAGE.

NOTA

Per necessita' di comunicazioni con periferiche che utilizzino il modo semplice a tre linee un circuito di interfaccia puo' essere facilmente costruito usando una coppia di integrati BUFFER/DRIVER.

La linea RS-232 normalmente trasmette i dati usando un segnale a 12 Volts, tuttavia, utilizzando dei cavi si puo' lavorare con un segnale di 5 volts.
Nelle tavole vediamo il connettore standard RS-232, mentre di seguito mostriamole funzioni ed il PIN ASSIGNEMENT di ognuna di queste linee.

VIA l.n	RS232 pin.	VIA ABB pin	EIA	IN OUT	FUNZIONE
GND	1	A	GND	AA -	1,2 Protezione
CB1	3	B	Sin	BB In	1,2 Ric. dati
PB0	3	C	Sin	BB In	1,2 Connes.a Sin
PB1	4	D	RTS	CA Out	2 Rich. invio
PB2	20	E	DTR	CD Out	2 Data term.pr.
PB3	18	F	RI	CE In	3 Indic.Ring
PB4	8	H	DCD	CF In	2 Lin.ric.segn.
PB6	5	K	CTS	CB In	2 Clear per In.
PB7	6	L	DSR	CC In	2 Dati pronto.
CB2	2	M	SoutBA	Oul	1,2 Trasm. dati
GND	7	N	GND	AB	2,3 Segnale GRD.

NOTA

Vediamo il significato dei valori riportati nella penultima colonna.

1-Interfaccia tre linee(RTS e DTR sono entrambi tenuti alti durante questo modo)

2-Interfaccia X.

3-Disponibile solo attraverso programmazione utente e quindi non implementata o usata nel Sistema Operativo.

L' implementazione della porta seriale sui nostri computers e' molto interessante perche' consente tramite l' uso del software di emulare una periferica Hardware.
L' Hardware di cui si parla e' il 6551 UNIVERSAL

ASYNCHRONOUS TRANSMITTER AND RECEIVER che e' anche conosciuto in termini abberviati come UART.

I progettisti del VIC 20 intendevano utilizzare questo integrato per generare una porta di ingresso/uscita RS-232, tuttavia la MOS non fu in grado di mettere a punto in tempo e quindi di consegnare alla COMMODORE l' integrato necessario.
Si dovette ricorrere pertanto ad una simulazione Software.
L' idea consenti' comunque di risparmiare un integrato ed in definitiva una decisione presa sotto l' incalzare della necessita' si dimostro' utile ed interessante, tanto e' vero che lo stesso concetto fu poi trasferito anche sul CBM 64.
Per cui il 6551 non esiste e noi, nel resto dello scritto ci riferiremo quindi ad uno PSEUDO registro 6551 anche perche' chi abbia conoscenze di questo tipo di integrato trovera' piu' semplice comprenderne le singole funzioni.

Come per gli altri integrati di I/O cosi' le funzioni del 6551 sono controllate da registri in locazioni di memoria specifiche.
Gli pseudo registri del 6551 sono localizzati in varie parti, e vedremo poi dove, dell' area di immagazzinamento variabili nella parte piu' alta della memoria.
Le routines operative, cioe' le routines che consentono alla RS-232 di funzionare richiedono 2 zone di Buffer, ciascuna delle quali di grandezza pari a 256 Bytes.
Questi Buffers avranno la funzione di immagazzinamento temporaneo dei dati rispettivamente uno per i dati in ricezione, l' altro per quelli in trasmissione attraverso la porta seriale.

I 512 Bytes di memoria occupati da questi Buffers sono localizzati nella parte alta della memoria RAM, mentre l' indirizzo di partenza,cioe' l' indirizzo della zona di inizio di questi Buffers sono contenuti in 4 locazioni di memoria.
I due piu' importanti registri di questa porta sono:

-REGISTRO DI CONTROLLO

-REGISTRO DI COMANDO

che controllano le operazioni della porta stessa e che descriviamo di seguito.

I REGISTRI DELLO PSEUDO 6551

Pseudo registro di controllo 6551.

-Indirizzi: $0293- Dec. 659

La funzione del registro di controllo e' quella di fissare la velocita' di trasmissione e di ricezione dei dati e di fissare il numero di Bits necessari per trasmettere ogni carattere.
La velocita' con la quale i dati sono spediti o ricevuti e' chiamata BAUD RATE, cioe' il numero di Bits per secondo.

Se il BAUD RATE e' fissato a 300 Baud, questo vorra' dire che si trasmette o si riceve alla velocita' di trecento Baud.

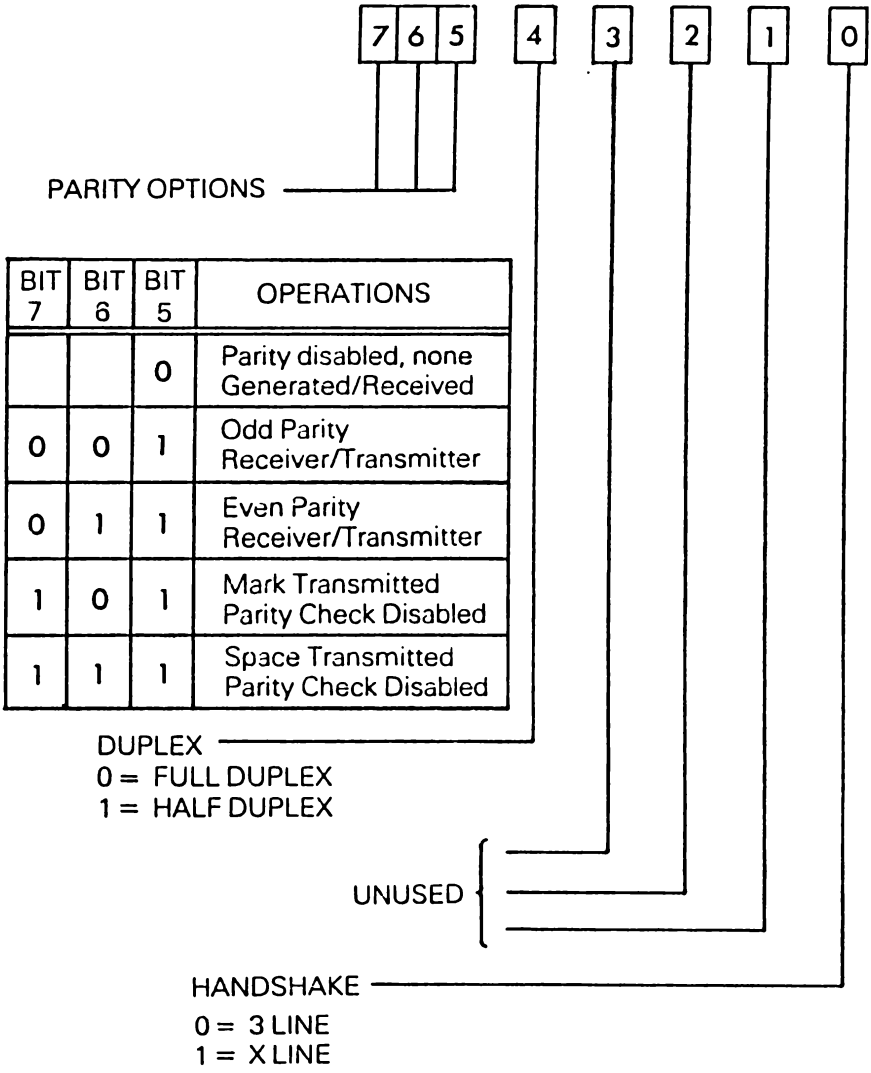


Fig. 27 - Funzione dei bits nel registro di comando del VIC per RS-232.

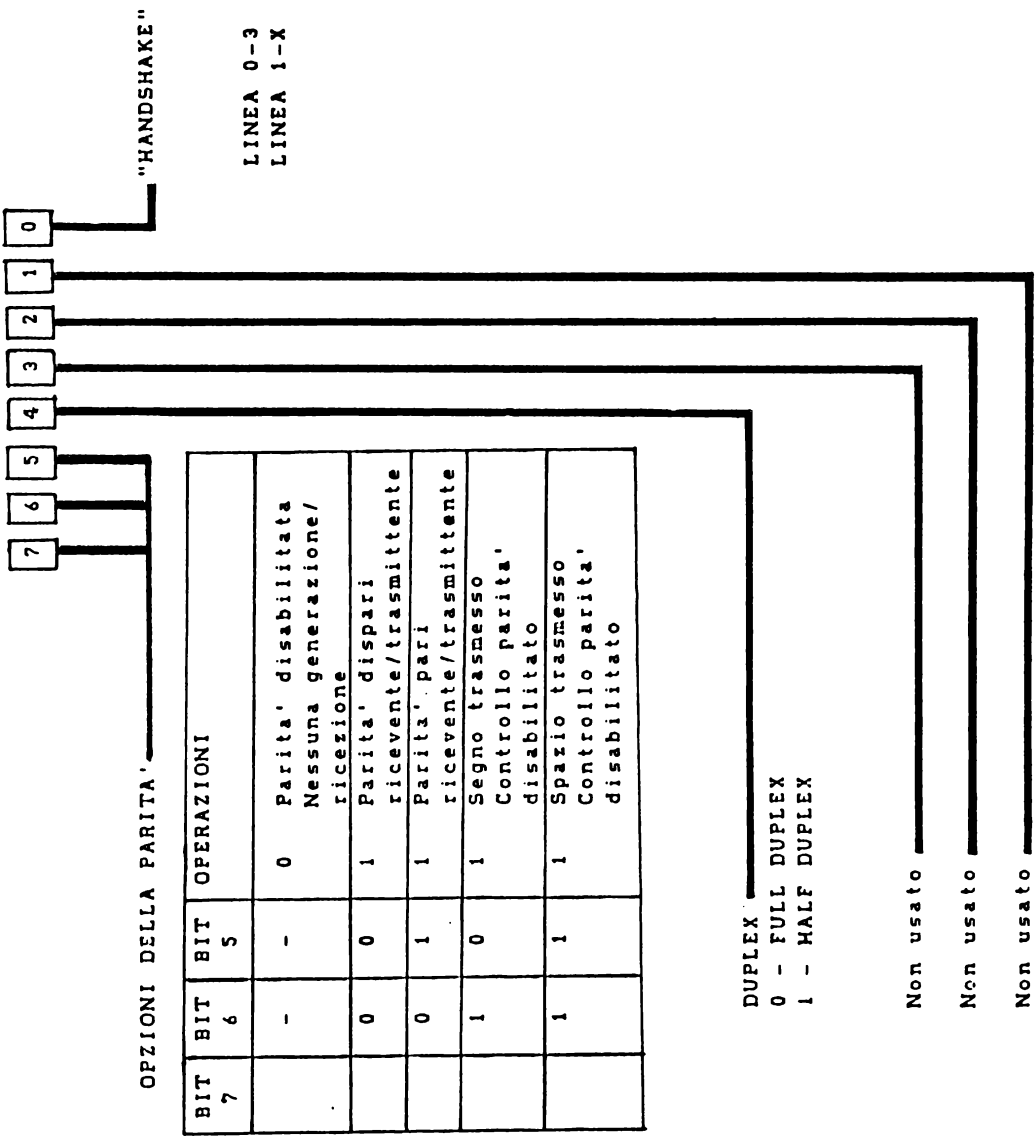
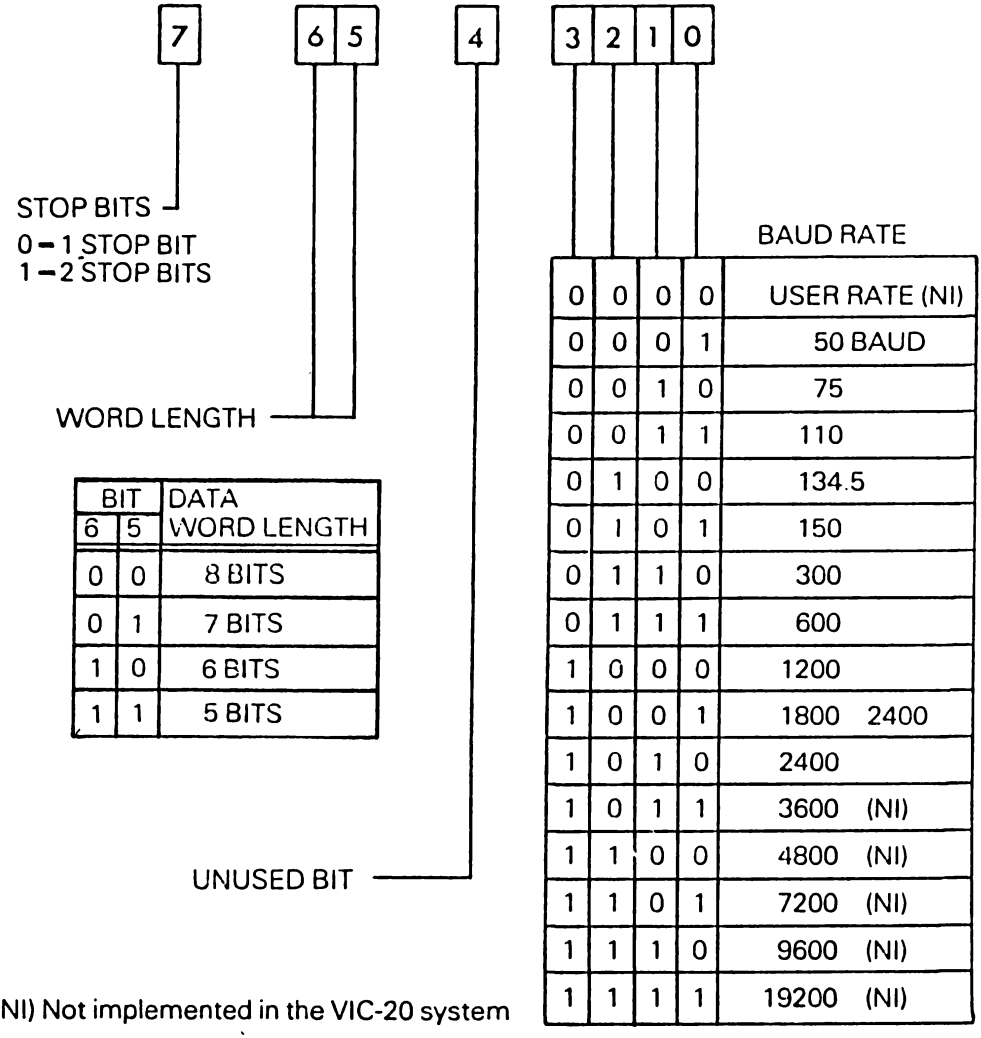


Fig. 28 - Mappa del registro di comando.



BIT 6	BIT 5	DATA WORD LENGTH
0	0	8 BITS
0	1	7 BITS
1	0	6 BITS
1	1	5 BITS

0	0	0	0	USER RATE (NI)
0	0	0	1	50 BAUD
0	0	1	0	75
0	0	1	1	110
0	1	0	0	134.5
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	1800 2400
1	0	1	0	2400
1	0	1	1	3600 (NI)
1	1	0	0	4800 (NI)
1	1	0	1	7200 (NI)
1	1	1	0	9600 (NI)
1	1	1	1	19200 (NI)

(NI) Not implemented in the VIC-20 system

Fig. 29 - Funzione dei bits nel registro di controllo del VIC per RS-232.

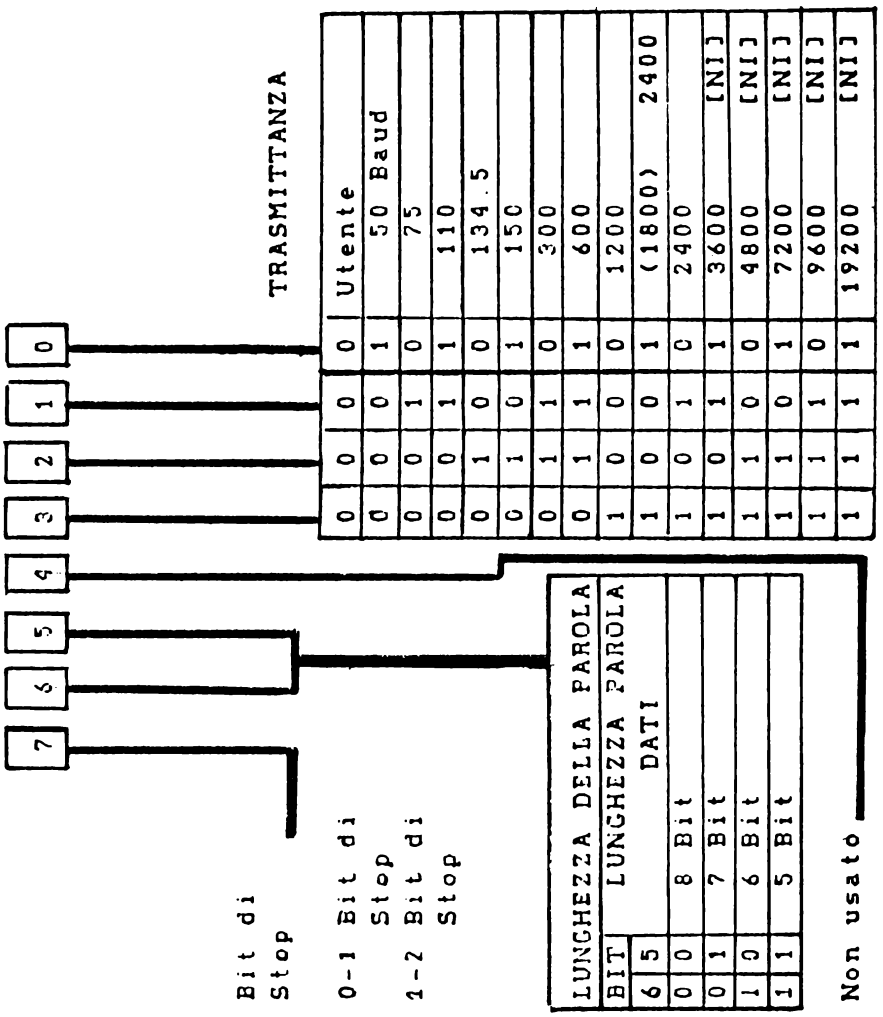


Fig. 30 - Mappa del registro di controllo.

Cioe' ogni carattere e' trasmesso come gruppo di 8 Bits piu' il bit di STOP ed il bit di PARITA' e quindi per un totale di 10 Bits, vorra' dire che a questa velocita' si trasmetteranno 30 caratteri al secondo.
La selezione del livello di trasmissione, cioe' della velocita' dipende dalle caratteristiche della periferica che comunica con l' unita' centrale tramite la RS-232 per cui sara' necessario controllare queste caratteristiche, magari nel manuale della periferica che si desidera collegare.

I Bits 5,6 e 7 di questo registro controllano il numero di bits necessari per trasmettere o ricevere dati tra l' unita' centrale e la periferica.

Pseudo registro di comando 6551.

-Indirizzi: $0294- Dec. 660

Il contenuto di questa locazione di memoria, cioe' questo registro, controlla il modo di trasmissione e di ricezione dei dati.
Il bit 0 fissa il modo a 3 o a X linee come visto in precedenza.
Il contenuto del Bit 4 fissa il cosi' detto modo DUPLEX come segue:

DUPLEX PIENO o FULL DUPLEX

cioe' con trasmissione e ricezione simultanea dei dati.

MEZZO DUPLEX o HALF DUPLEX

cioe' con trasmissione alternata alla ricezione.

I Bits 5,6 e 7 (naturalmente il loro contenuto) determinano la natura del Bit di parita' e quando sono trasmessi i punti e gli spazi.
Il bit di parita' e' trasmesso dopo la trasmissione dei bits di dati ed ha una funzione di controllo di errore.
La scelta del bit di parita' e del suo funzionamento e di quando deve essere fissato o disabilitato, dipende dal tipo di periferica collegato con il controller alla porta RS-232.

Vediamo dopo nelle tavole le altre locazioni dello pseudo 6551.

USO DELLA PORTA RS-232

APERTURA DI UN CANALE

Comando: OPEN lf,2,0,"(rc) (rm)"

dove:

-lf = numero di file logico

-rc = registro di controllo. Deve essere un carattere ASCII equivalente a quello richiesto per fissare la funzione (vista in precedenza) del REGISTRO DI CONTROLLO.

Esempio:

Per fissare un BAUD RATE a 300 e trasmettere un codice di 7 bits si utilizzera' CHR$(6+32). Questo comando fissera' i bits 1,2 e 5 allo stato logico "1" e lascera' i restanti a "0".

-rm = regsitro di comando. Deve esere un carattere ASCII equivalente a quello richiesto per fissare la funzione del REGISTRO DI COMANDO.

Esempio:

Per fissare l' uscita al MARK PARITY e l' utilizzo del DUPLEX PIENO usare CHR$(32+128). Questo comando fissera' i bits 5 e 7 allo stato logico "1" e lascera' gli altri a "0".

PUNTO D' INGRESSO PER IL VIC 20 $FFC0

PUNTO D' INGRESSO PER IL CBM 64 $FFC0

NOTE D' USO

Dovrebbe essere aperto solo un canale di comunicazione RS-232 per volta poiche' un comando OPEN resetta i puntatori del Buffer.
Quindi all' esecuzione di un secondo comando OPEN i dati fissati con il primo comando saranno distrutti.
Inoltre questo comando deve esere obbligatoriamente adoperato, anche in questo caso per evitare perdite

o sovrapposizioni di dati, prima della dichiarazione di variabili o prima di comandi di dimensionamento (DIM).
Cio' perche' il comando di apertura di un canale sulla RS-232 esegue una funzione di pulizia, in pratica un CLEAR, prima di collocare i 512 Bytes utilizzati dai due Buffer che, come ricordiamo, sono messi nella parte alta della memoria RAM.
Nel caso che in questa parte della memoria non ci sia sufficiente spazio i dati verranno persi.
L' ultima parte del parametro di questo comando, cioe' quella relativa al campo con il nome del file racchiuso fra virgolette, puo' avere un massimo di 4 caratteri di cui solo due saranno utilizzati dal Sistema Operativo nelle attuali condizioni mentre gli altri 2 sono riservati per utilizzi futuri.

Nessun errore di controllo viene eseguito dal Sistema Operativo del computer circa il contenuto delle locazioni di memoria assegnato ai rispettivi registri dai caratteri relativi ai caratteri di comando e di controllo. Comunque un errore nella selezione del BAUD RATE provochera' un malfunzionamento del sistema stesso di trasmissione o di ricezione.

Se non verra' selezionato nessun BAUD RATE potremo avere un salto ad un gruppo di dati errati mentre l' uscita sul canale sara' automaticamente fissata a 300 BAUD.

RICEZIONE DATI DA UN CANALE RS-232

Comando: GET £ lf,(variabile stringa)

dove:

lf- file logico ID utilizzato nel comando di apertura del canale (OPEN).

PUNTO D' INGRESSO PER IL VIC 20 $FFCF

PUNTO D' INGRESSO PER IL CBM 64 $FFCF

Questi relativamente all' apertura di un canale per l' INPUT.

PUNTO D' INGRESSO PER IL VIC 20 $FFE4

PUNTO D' INGRESSO PER IL CBM 64 $FFE4

Questi relativamente alla funzione GET di un carattere dal Buffer.

NOTE D' USO

I dati ricevuti sono immessi nel Buffer apposito di 255 Byte fissato durante l' esecuzione del comando OPEN visto in precedenza.
I dati in oggetto sono sotto il controllo dei temporizzatori degli integrati 6522 e dei relativi INTERRUPTS e sono messi a punto nel loro giusto stato durante la fase di RUN del programma BASIC.

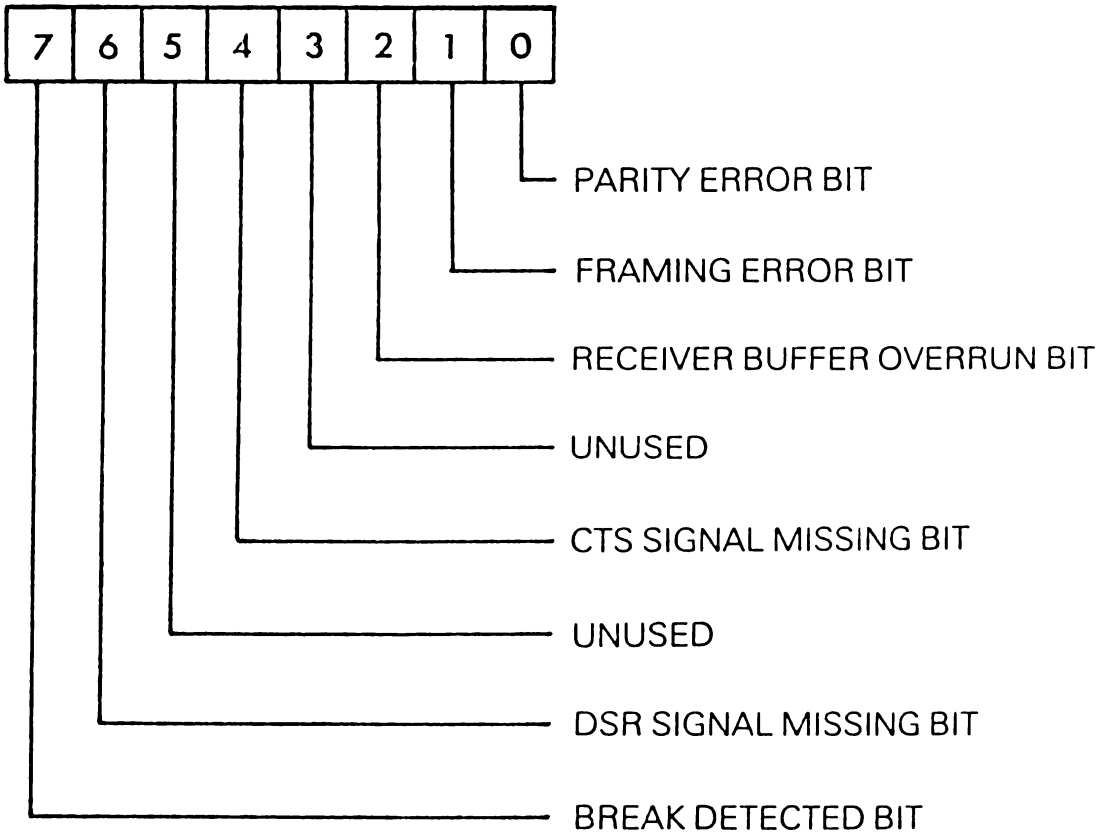
Questa operazione viene eseguita perche' abbiamo la linea di ingresso dati della RS-232 collegata alla

linea di HANDSHAKE CB1 per cui un input sul CB1 provochera' un INTERRUPT DI SISTEMA del tipo NMI.
L' utilizzo quindi degli INTERRUPTS NMI e' la ragione per cui la cassetta ed il BUS seriale non dovrebbero essere utilizzati durante la fase di comunicazione dati della RS-232.

Il buffer di ricevimento e' organizzato , in relazione alla manipolazione dei dati, con il concetto FIFO (FIRST IN FIRST OUT).
Il buffer elimina la necessita' che il Basic attenda l' ingresso dei dati.

Il Basic ha accesso al Buffer utilizzando il comando GET per trasferire un singolo Byte di dati inuna variabile.
Se non ci sono dati nel Buffer il comando GET restituira' un carattere nullo.
Se abbiamo una condizione di OVERFLOW (in pratica troppi dati), allora tutti i caratteri ricevuti durante la presenza di questa condizione sono persi.

Una condizione di OVERFLOW e' indicata dal bit numero 2 a "1" del Registro di Stato della RS-232.
Questo tipo di condizione si presenta spesso utilizzando il Basic a causa del rapporto di lentezza esecutiva tipica dell' interprete nell' adoperare il comando GET che di norma si impiega nella concatenazione di stringhe.
E' quindi consigliabile implementare delle piccole routine in Linguaggio Macchina.



RS-232 STATUS REGISTER — $0297

Fig. 31 - Funzione dei bits nel registro di stato del VIC per RS-232.

(Locazioni SDD00 - SDD0F del dispositivo 2 6526)

PIN	6526	DESCRIZIONE	EIA	ABBR.	IN/OUT	MODI
C	PB0	Dati ricevuti	(BB)	Sin	IN	1 2
D	PB1	Richiesta di invio	(CA)	RTS	OUT	1(*) 2
E	PB2	Terminale dati disponibile	(CD)	DTR	OUT	1(*) 2
F	PB3	Indicatore anello	(CE)	RI	IN	3
H	PB4	Ricevuto segnale di linea	(CF)	DCD	IN	2
J	PB5	Non asegnato	-	XXX	IN	3
K	PB6	Azzera per invio	(CB)	CTS	IN	2
L	PB7	Insieme dati disponibile	(CC)	DSR	IN	2
B	FLAG2	Dati ricevuti	(BB)	Sin	IN	1 2
M	PA2	Dati trasmessi	(BA)	Sout	OUT	1 2
A	GND	Linea di terra (protezione	(AA)	GND	-	1 2
N	GND	Terra del segnale	(AB)	GND	-	1 2 3

MODI:

1) Interfaccia linea 3 (Sin, Sout, GND)
2) Interfaccia linea X
3) Disponibile solo all'Utente (Non usata / non implementata nel codice)

*) Queste linee sono mantenute alte durante il modo LINEA-3.

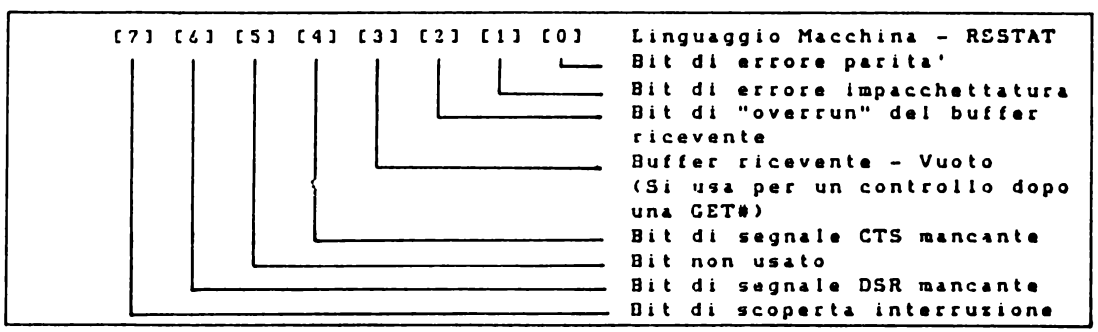


Fig. 32 - (sopra) Linee Porta Utente.
(sotto) Registro di stato dell'RS-232.

TRASMISSIONE DATI SU CANALE RS-232

Comando: CMD lf

PRINT # lf,(lista di variabili)

dove:

lf- file logico ID utilizzato nel comando di apertura del canale (OPEN).

PUNTO D' INGRESSO PER IL VIC 20 $FFC9

PUNTO D' INGRESSO PER IL CBM 64 $FFC9

Questi relativamente all' apertura di un canale per OUTPUT

PUNTO D' INGRESSO PER IL VIC 20 $FFD2

PUNTO D' INGRESSO PER IL CBM 64 $FFD2

Questi relativamente all' output di un carattere su un canale.

NOTE D' USO

Quando uno di questi due comandi viene utilizzato i dati vengono per prima cosa trasferiti dalla selezionata stringa in memoria al Buffer di trasmissione che abbiamo visto all' inizio del capitolo.

Da questo Buffer va in uscita sul canale RS-232 utilizzando il formato ed il BAUD RATE selezionati con il comando di apertura canale OPEN.

Il dato in uscita e' completamente trasparente al Basic poiche' la temporizzazione e' eseguita dai TIMERS del 6522 e l' uscita di ogni Byte inizializzata da un INTERRUPT NMI.

CHIUSURA DI UN CANALE DI DATI SU RS-232

Comando: CLOSE lf

dove:

lf- file logico ID utilizzato nel comando di apertura del canale (OPEN).

PUNTO D' INGRESSO PER IL VIC 20 $FFC3

PUNTO D' INGRESSO PER IL CBM 64 $FFC3

NOTE D' USO

La chiusura di un canale di dati sulla RS-232 comporta che tutti i restanti dati vengano scartati, arresta la trasmissione o il ricevimento dei dati stessi, implementa un comando RTS (RETURN FROM SUBROUTINE) e la linea Sout alta. Rende infine disponibile l' area di memoria utilizzata in precedenza per i buffers che abbiamo visto.

La chiusura di un file RS-232 consentira' di nuovo l' uso della cassetta e della porta IEEE.
Per questi risultati e' chiaro che bisognera' assicurarsi che tutti i dati presenti nel buffer siano stati trasmessi.
Questo controllo puo' essere fatto con l' esame della variabile Basic ST, che deve essere ST=0.
La seconda parte del controllo va fatta sul bit 6 della PORTA PARALLELA A DEL VIA numero 1 di indirizzo decimale 37151 che deve essere a 1.
Se entrambe le condizioni viste risul tano vere allora ci sono ancora dati nel Buffer.

Nell' ultima tavola riportiamo lo schema di piedinatura della porta d' espansione del CBM 64.

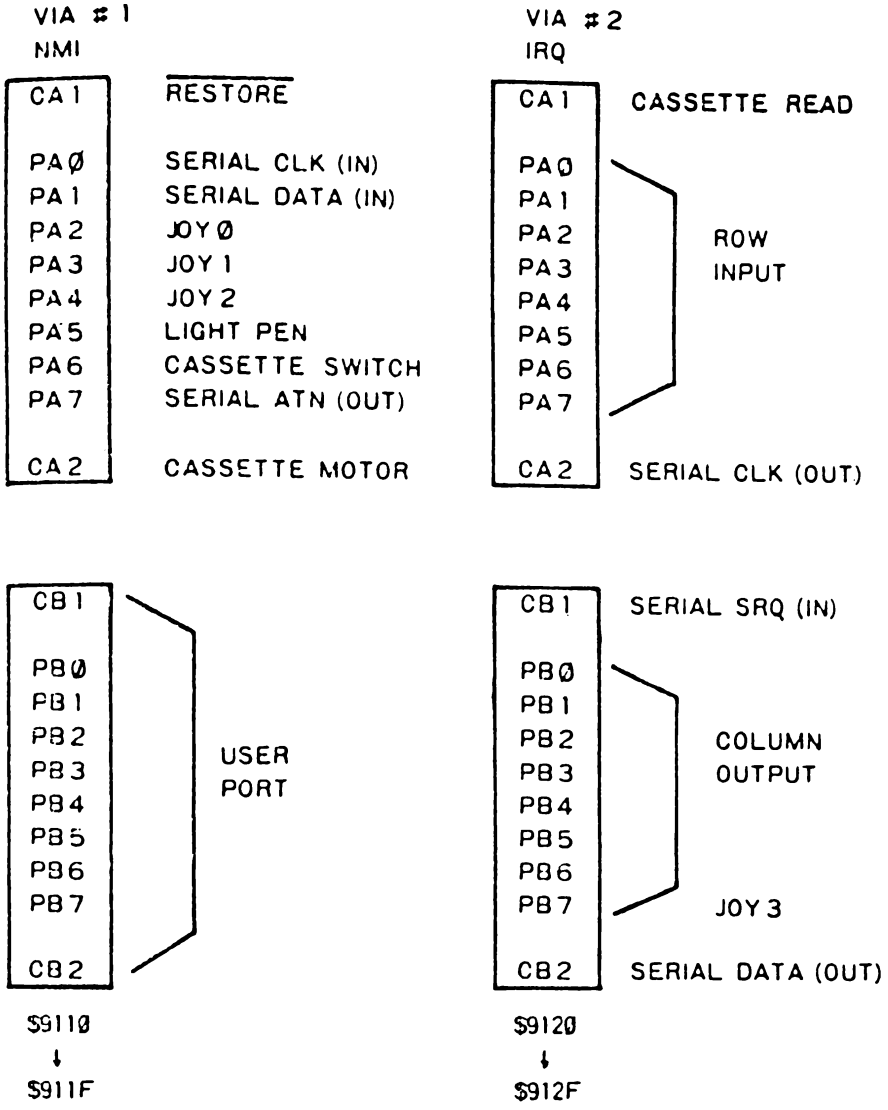


Fig. 33 - Distribuzione delle linee di I/O dagli integrati 6522 per il VIC 20

NOME	PIN	DESCRIZIONE
GND	1	Terra del sistema
+5 VDC	2	(Assorbimento massimo supportabile dai dispositivi
+5 VDC	3	PORTA UTENTE e CARTUCCIA: 450 mA)
IRQ	4	Linea di RIchiesta Interruzione per 6502
R/W	5	Lettura/Scrittura
DOT CLOCK	6	Timer punti video (8.18 MHz)
I/O1	7	Blocco 1 di I/O ($DE00-$DEFF) non bufferizzato @
GAME	8	Input ls ttl
EXROM	9	Input ls ttl
I/O2	10	Blocco 2 di I/O ($DE00-$DEFF) bufferizzato @ output ls ttl
ROML	11	Blocco di 8K RAM/ROM decodificato ($8000) @ output ls ttl
BA	12	Segnale di bus disponibile dal circuito VIC-II non bufferizzato - carico massimo 1 ls
DMA	13	Linea di richiesta di accesso diretto alla memoria input ls ttl
D7	14	Bit 7 del bus dati
D6	15	Bit 6 del bus dati
D5	16	Bit 5 del bus dati
D4	17	Bit 4 del bus dati
D3	18	Bit 3 del bus dati
D2	19	Bit 2 del bus dati
D1	20	Bit 1 del bus dati
D0	21	bit 0 del bus dati
GND	22	Terra del sistema
GND	A	
ROMH	B	Blocco bufferizzato di 8K di RAM/ROM decodificati @ ($E000)
RESET	C	Pin di RESET del 6502 out ttl bufferizzato/in non bufferizzato
NMI	D	Interruzione non mascherabile del 6502 out ttl bufferizzato, in non bufferizzato
02	E	Timer di sistema per Fase 2
A15	F	Bit 15 del bus indirizzi
A14	H	Bit 14 del bus indirizzi
A13	J	Bit 13 del bus indirizzi
A12	K	Bit 12 del bus indirizzi
A11	L	Bit 11 del bus indirizzi
A10	M	Bit 10 del bus indirizzi
A9	N	Bit 9 del bus indirizzi
A8	P	Bit 8 del bus indirizzi
A7	R	Bit 7 del bus indirizzi
A6	S	Bit 6 del bus indirizzi
A5	T	Bit 5 del bus indirizzi
A4	U	Bit 4 del bus indirizzi
A3	V	Bit 3 del bus indirizzi
A2	W	Bit 2 del bus indirizzi
A1	X	Bit 1 del bus indirizzi
A0	Y	Bit 0 del bus indirizzi
GND	Z	Terra del sistema

I nomi soprasegnati sono attivi a segnale basso

Fig. 34 - Porta di espansione del CBM 64.

STAMPANTE

CAPITOLO VENTITRESIMO

L A S T A M P A N T E

Introduzione

La stesura di questa parte del manuale sulle periferiche ha richiesto un certa ponderazione in quanto sono numerose e di diverso utilizzo le stampanti che si possono collegare ai computer COMMODORE.

Per questo motivo avevao deciso di tenerci su linee generali, ma poiche' questo volume e' dedicato particolarmente agli utenti abbiamo preferito dare descrizioni, se non proprio molto approfondite di 3 gruppi principali e molto simili fra loro:

1) GP-100 1525 1515

2) 2022 3022 4022 4023 1526 MPS 802

3) MPS 801

E con questo crediamo di aver compiuto un esame quasi completo.

USO DELLA STAMPANTE

Prima di adoperare la stampante si dovrebbe essere sicuri di conoscere quanto segue:

-Saper usare il computer

-Conoscere almeno la programmazione elementare in Basic

-Saper scrivere dei files da e per una periferica quale ad esempio un registratore a cassetta o un floppy disk.

L' unita' di stampa, come del resto qualsiasi periferica, va collegata, a computer spento tramite l' apposito cavo o l' interfaccia.
Successivamente accendere la stampante e far eseguire, la prima volta naturalmente, l' AUTO TEST.
Ricordarsi, prima di questa operazione di inserire la carta e controllare che il nastro sia correttamente posizionato.
Controllare quindi che tutti caratteri siano stampati uniformemente e senza difetti.

La vostra stampante vi da molto di piu' che copie chiare e veloci. Poiche' e' fornita internamente di un sistema a microprocessore essa e' molto versatile.
Se correttamente usata potrete usarla per:

-Stampare listati dei programmi

-Stampare risultati dei programmi

-Stampare grafici

Potrete inoltre usare la vostra periferica per formattare i dati in stampa e per caratteri maggiorati e sarete in grado di creare un vostro set di caratteri.

COMANDI PER LA STAMPANTE

Quando volete stampare qualcosa quello che dovete fare e' essenzialmente trasferire le funzioni del video alla stampante. Per questo sono previsti alcuni comandi Basic speciali, del resto visti in precedenza per l' uso delle periferiche.
La maggior parte degli altri comandi e regole del Basic rimangono le stesse.
Ricordatevi sempre di premere il tasto RETURN dopo aver battuto ciascun comando.

IL COMANDO OPEN

La sintassi di questo comando e':

OPEN lfn,dn,(sa)

Questo comando stabilisce una corrispondenza fra il numero del file e la periferica.
Il parametro lfn o NUMERO DI FILE LOGICO puo' essere un qualsiasi numero compreso fra 1 e 255 e non importa quale numero scegliate purche' rimanga

coerente con tutto il blocco dei comandi.
Il dn o DEVICE NUMBER o numero della periferica (chiamato anche indirizzo primario) si riferisce al numero della periferica alla quale desiderate inviare il file o gruppo di dati da stampare. Nel caso della GP-100 il numero puo' essere sia 4 che 5.

NOTA

Se volete cambiare il numero della periferica e' bene che facciate eseguire questa operazione ad un tecnico o comunque consultatelo se non siete veramente degli esperti.
Questa necessita' puo' presentarsi se avete due stampanti da collegare allo stesso computer.
Infatti, poiche' e' necessario riferirsi individualmente all' una o all' altra delle stampanti e' necessario cambiare il numero di una delle due.

Il SA o (Secondary Adress) INDIRIZZO SECONDARIO e' un concetto applicabile a molte stampanti. Esso infatti avverte il sistema a microprocessore che deve avvenire una formattazione. Con la stampante GP-100 si puo' selezionare uno dei seguenti modi:

0:Modo "CURSOR UP"

7:Modo "CURSOR DOWN"

NOTA

Dato che questo ultimo parametro e' opzionale, se non viene selezionato esso e' automaticamente fissato a 0.

IL COMANDO CMD

La sintassi del comando e':

```
CMD lfn
```

CMD trasferisce il controllo del computer alla stampante. Il parametro lfn deve essere lo stesso del comando OPEN con il quale e' associato.
A differenza del comando PRINT la linea o bus verso la ricevente rimane aperta.
La linea o il bus della periferica ricevente (in questo caso la stampante) e' quindi attivata in posizione di ricezione. Quando date questo comando la stampante scrive Ready. Se a questo comando fate seguire un comando PRINT o LIST l' uscita avvera' invece che sul video direttamente sulla stampante.

IL COMANDO PRINT

La sintassi di PRINT#e':

```
PRINT#lfn,data
```

Il comando PRINT#lavora come il comando PRINT con la differenza che l' uscita di dati avviene, in

questo caso sulla stampante invece che sul video.
La linea della stampante deve essere chiusa dopo la stampa dei dati assegnati. Perciò se avete usato il comando CMD e' necessario farlo seguire da un PRINT£ per chiudere il collegamento fra la stampante ed il computer.

NOTA

Nel Basic standard della Commodore il comando PRINT puo' essere abbreviato con un punto interrogativo (?), mentre il comando PRINT#deve essere digitato per intero.

IL COMANDO CLOSE

La sintassi del comando e':

CLOSE lfn

Dovete sempre chiudere il file dopo averlo usato per stampare.
Non dovete aprire piu' di 10 file e sara' buona abitudine di chiuderli mano a mano che finite di stampare su di essi.
Questo modo di procedere vi permettera' di avere il massimo numero di files disponibili all' uso. Come vedremo piu' avanti un file puo' essere aperto sotto diversi numeri di file logico allo stesso tempo.

NOTA

Ricordiamo ancora una volta che poiche' il comando CMD non chiude la linea con la stampante, dovete sempre far precedere il comando CLOSE da un PRINT# allo scopo di chiudere correttamente un file.

Riportiamo ora una serie di esempi in versione corretta ed errata sull' utilizzo dei comandi visti:

CORRETTO	ERRATO
OPEN5,4 PRINT#5,"ESEMPIO" CLOSE5	OPEN5,4 ?#5,"ESEMPIO" CLOSE5
OPEN5,4 CMD5,"ESEMPIO" PRINT#5:CLOSE5	OPEN5,4 CMD5,"ESEMPIO" CLOSE5
OPEN5,4 CMD5,"ESEMPIO" PRINT#5,"ESEMPIO" CLOSE5	OPEN5,4 CMD5,"ESEMPIO" PRINT#5,"ESEMPIO" PRINT#5:CLOSE5
OPEN5,4 PRINT#5,"ESEMPIO" CMD5,"ESEMPIO" PRINT#5:CLOSE5	OPEN5,4 PRINT#5,"ESEMPIO" CMD5,"ESEMPIO" CLOSE5

Con le concise descrizioni di come operano questi comandi possiamo ora procedere ad esaminare nella parte seguente per poter imparare ad usare gli stessi comandi per controllare la stampante. Per una definizione piu' approfondita dei comandi come dei parametri ad essi collegati vi rimandiamo alla parte iniziale di questo volume oppure al manuale Basic.

STAMPA IN MODO DIRETTO

Il modo diretto vi permette di far entrare i comandi di stampa dalla tastiera. L' esempio seguente mostra un intero processo di stampa in modo diretto di un breve programma Basic. Il file contenente un singolo comando Basic e' inserito nella memoria del computer. Il file viene aperto, e' aperto il canale di uscita ed il file viene listato.
Dopo che il file e' stato stampato vengono chiusi il canale di uscita ed il file. La stampante e' ora disattivata ed il computer e' pronto per ricevere nuovi comandi.

```
A)DIGITARE      VIDEO       STAMPANTE

B)IO?"TEST"     IO?"TEST"

C)OPEN3,4       OPEN3,4
Ready.

D)CMD3          CMD3        Ready.
```

```
E)LIST          LIST          10PRINT"TEST"
Ready.

F)PRINT#3       PRINT#3

G)CLOSE3        CLOSE3
Ready.
```

Commento alle singole fasi:

A) Si immettono dati nella memoria del computer.

B) Si apre un file e si da un lfn 3. Il 4 rende disponibile il file per la stampante.

C) Il canale della stampante e' in "LISTEN" cioe' e' aperto per ricevere dati.

D) Il programma viene stampato. La stampante e' sempre in "LISTEN".

E) Si usa il comando PRINT # per mettere la stampante in posizione "UNLISTEN".

F) Si chiude il file cosi' che il canale 3 puo' essere usato per qualsiasi altro motivo.

STAMPA SOTTO IL CONTROLLO DI PROGRAMMA

Come avete visto potete controllare la stampante direttamente dalla tastiera. Si puo' anche controllarla con un programma Basic. Nell' esempio seguente questo piccolo programma si trova nella memoria del computer dove puo' esservi immesso o direttamente da tastiera o caricato dal

registratore o dal disco.

```
10 OPEN 3,4
20 CMD 3
30 PRINT"PROGRAMMA DI CONTROLLO"
40 LIST
```

Dopo aver dato il RUN avremo sulla stampante il seguente risultato:

```
PROGRAMMA DI CONTROLLO

10 OPEN 3,4
20 CMD 3
30 PRINT"PROGRAMMA DI CONTROLLO"
40 LIST
```

NOTA

Ricordiamo che il comando LIST all' interno di un programma conclude l' esecuzione del programma stesso in qualunque punto si trovi. Quindi nel caso precedente, quando avrete finito di far eseguire un programma dovete battere il comando PRINT # per chiudere il canale e dopo digitare il comando CLOSE per chiudere il file.

MODI DI STAMPA E CODICI DI CONTROLLO

Oltre al modo standard le stampanti del primo gruppo possono operare nei seguenti modi selezionabili con il comando PRINT#:

1- Caratteri in doppia grandezza con controllo software.

2- Capacita' grafiche

3- In modo grafico, un disegno formato da DATA puo' essere ripetuto tutte le volte che si vuole con un singolo comando.

4- Posizione di stampa indirizzabile con un carattere o con un punto colonna.

5- Grafici, caratteri standard, caratteri doppi possono essere mescolati in una sola linea.

6- Stampa dei caratteri in reverse

7- Stampa automatica.

CODICI DI CONTROLLO

CODICE	CHR$	DESCRIZIONE
NL	10	LINE FEED DOPO LA STAMPA
CR	13	DITTO
BS	8	COMANDO PER MODO GRAFICO
SO	14	COMANDO PER IL CARATTERE DOPPIO
SI	15	COMANDO CARATTERE STANDARD
POS	16	INDIRIZZO DI INIZIO STAMPA

ESC	27	IND. DI PUNTO DOPO POS.
SUB	26	REPEAT DEL COMANDO GRAPHIC
CURS.UP	145	MAIUSCOLO
CURS.DOWN	17	MINUSCOLO
RVS ON	18	ABILITA IL REVERSE
RVS OFF	146	DISABILITA IL REVERSE

ATTENZIONE!!!

Esamineremo ora alcuni dei vari formati di stampa ma escluderemo quelli che si ricavano molto semplicemente dai vari manuali. In particolare non ci soffermeremo a lungo sulla parte grafica.

CARATTERI IN DOPPIA GRANDEZZA

Quando la stampante viene accesa automaticamente si trova nel modo di caratteri standard. Tuttavia quando vengono selezionati i vari tipi di caratteri, questi rimangono selezionati fintanto che non ordiniamo di tornare al modo standard.

I Esempio: Si desidera stampare una frase o stringa di caratteri in caratteri a doppia grandezza.

Programma:

```
10 OPEN2,4
20 PRINT#2,CHR$(14)"GP-100VC PRINTER"
30 CLOSE2
```

Dopo aver dato il RUN apparira' sulla stampante la seguente scritta in formato doppio:

GP-100VC PRINTER

II Esempio: Tenuto conto che il programma precedente NON chiude il canale, cioe' non resetta la stampante, eseguire le stesse funzioni di cui sopra, ma riportare la stampante in modo standard ed eseguire il LIST del programma.

Programma:

```
10 OPEN2,4
20 PRINT#2,CHR$(14)"GP-100VC PRINTER"
30 PRINT#1,CHR$(15)
40 CMD1:LIST
```

Dopo il RUN verra' scritto il messaggio precedente il carattere doppio e la lista del programma in modo normale.

GP-100VC PRINTER

```
10 OPEN2,4
20 PRINT#2,CHR$(14)"GP-100VC PRINTER"
30 PRINT#1,CHR$(15)
40 CMD1:LIST
```

STAMPA IN UNA DETERMINATA POSIZIONE

Con un CHR$(16) la posizione di inizio stampa può' essere determinata.
La distanza dall' inizio della riga nel seguente esempio e' data dai valori congiunti di CHR$(48) che e' 0 e CHR$(56) che e' 8 nella prima parte che infatti inizia a stampare dall' ottavo carattere.
Nella riga 70 da CHR$(51) che equivale a 3 e CHR$(48) che e' 0. Infatti l' inizio di stampa della seconda stringa e' a 30 caratteri dall' inizio.

Esempio:

```
10 OPEN4,4
20 FOR I=1TO4
30 PRINT#4,"0123456789"

50 PRINT#4,CHR$(10)
60 PRINT#4,CHR$(16) CHR$(48)CHR$(56)"GP-100VC";
70 PRINT#4,CHR$(16) CHR$(51)CHR$(48)"PRINTER";
80 CLOSE4
```

Dopo aver eseguito il RUN.

```
0123456789012345678901234567890 ...

GP-100VC                PRINTER
```

MODO CURSOR UP

Con il codice CHR$(145) viene selezionato il set di caratteri presente nel modo "CURSOR UP". Questo

modo, che e' poi il maiuscolo entra automaticamente in funzione all' accensione della stampante. Non e' necessario fornire alcun esempio, ma ricordiamo solo che questo comando puo' essere utile per stampe miste cioe' di minuscoli e maiuscoli o minuscoli e caratteri grafici.

MODO CURSOR DOWN

Si seleziona con CHR$(17).

Esempio:

```
10 OPEN8,4,7
20 PRINT#8,CHR$(17)"PERSONAL COMPUTER"
30 PRINT#8,CHR$(17)"GRAPHIC PRINTER"
40 CMD8
```

Dopo il RUN verra' stampato:

personal computer
graphic printer

Tralasciando gli altri modi tipici di questa stampante e che possono essere meglio individuati nel manuale relativo riportiamo un esempio di esecuzione di HARD-COPY schermo.

- HARD COPY OF THE SCREEN -

```
60000 REM SCREEN COPY
60010 SI$=CHR$(15):BS$=CHR$(8):PO$=CHR$(16)
60020 RV$=CHR$(18):RO$=CHR$(146):QT$=CHR$(34)
60030 MF$=CHR$(145):VR=PEEK(648)*256
60040 OPEN4,4:PRINT#4
60050 FORCL=0TO22:QF=0:AS$=MF$:FORRO=0TO21
60060 SC=PEEK(VR+22*CL+RO)
60070 IFSC=34THENQF=1-QF
60080 IFSC<>162THEN60110
60090 QF=1-QF:IFQF=1THENAS$=AS$+RV$+QT$:GOTO60170
60100 AS$=AS$+QT$+RO$:GOTO60170:GOTO60130
60110 IFQF=1AND(SC>=128)THENSC=SC-128:GOTO60130
60120 IFSC>=128THENSC=SC-128:RF=1:AS$=AS$+RV$
60130 IFSC<32ORSC>95THENAS=SC+64:GOTO60160
60140 IFSC>31ANDSC<64THENAS=SC:GOTO60160
60150 IFSC>63ANDSC<96THENAS=SC+32:GOTO60160
60160 AS$=AS$+CHR$(AS)
60170 IFRF=1THENAS$=AS$+RO$:RF=0
60180 NEXTRO
60190 IFQF=0THENPRINT#4,SI$PO$"20"AS$:GOTO60210
60200 PRINT#4,SI$;PO$;"20";AS$;QT$
60210 NEXTCL:CLOSE4:RETURN
```

CAPITOLO VENTIQUATTRESIMO

SECONDO GRUPPO DI STAMPANTI

2022 3022 4022 4023 1526 MPS802

Riportiamo ora le indicazioni relative alle stampanti del secondo gruppo.
Attraverso il controllo di formato opzionale della stampante, si puo' modificare l' interpretazione dei dati inviati alla stampante.
L' opzione del controllo di formato da la possibilita' di stampare numeri in colonna, stabilire il numero di linee per pagina e compiere altri utili operazioni di formattazione.
Per utilizzare l' opzione di controllo del formato e' necessario usare il terzo parametro che abbiamo visto a proposito del comando OPEN, cioe' l'indirizzo secondario o SA.

Come indirizzo secondario possiamo dare uno dei seguenti 11 valori:

0 Stampa i dati esattamente come ricevuti.

1 Stampa i dati secondo un formato definito in precedenza.

2 Immagazzina le istruzioni di formattazione dei dati.
3 Fissa il numero di linee da stampare per ogni pagina.

4 Abilita la stampante a fornire messaggi diagnostici.

5 Definisce un carattere programmabile.

6 Stabilisce lo spazio fra le linee.

7 Caratteri maiuscoli/minuscoli.

8 ASCII/GRAFICA.

9 Disabilita la stampa di messaggi diagnostici.

10 Resetta la stampante.

Dopo aver trasmesso il giusto comando di OPEN, e' necessario inviare un comando di PRINT # per trasmettere l' appropriato indirizzo secondario alla periferica che in questo casi sara' la Stampante.

NOTA

Ricordate che e' possibile avere fino a 10 files aperti contemporaneamente per cui esiste la possibilita' di eseguire molteplici funzioni.

SA = 0

STAMPA DI DATI ESATTAMENTE COME RICEVUTI.

Come abbiamo gia' detto in precedenza il valore di

questo indirizzo secondario puo' essere omesso.
Che lo includiate o meno nel vostro comando di OPEN, la stampante stampera' i dati esattamente come ricevuti.
Fino ad ottanta caratteri sono stampabili sulla stessa riga. Se l' ottanunesimo non e' un ritorno carrello, quest' ultimo viene eseguito automaticamente ed i rimanenti caratteri sono stampati su una nuova linea.

Esempio.

```
10 OPEN 5,4
20 PRINT#5, "QUESTA E' UNA PROVA"
```

Il risultato sara':

```
QUESTA E' UNA PROVA
```

SA = 1

STAMPA DEI DATI IN UN FORMATO PREDEFINITO.

L' indirizzo secondario 1 attiva le possibilita' di formattazione della vostra stampante.
I dati che devono essere stampati sono manipolati in accordo con un preventivo formato specificato usando l' indirizzo secondario SA = 2.
Se dovete trasmettere una stringa di dati con SA=1 e non vi e' alcuna specifica di formato nella memoria della stampante, la stringa di dati sara' stampata esattamente come ricevuta.

Quando formattate una stringa dal computer, deve

essere messo un carattere separatore (CHR$(29)) per delimitare i campi della stringa.
Dovendo saltare un campo di stampa e' necessario inserire un carattere (CHR$(160)), per far si che la successiva stringa sia stampata lasciando un campo vuoto.

```
10 OPEN2,4,2
20 OPEN1,4,1
30 PRINT#2,"AAA    AAA    AAA"
40 PRINT#1,"ABC"CHR$(29)CHR$(160)CHR$(29)"DEF"
50 CLOSE2:CLOSE1
```

Risultato:

```
ABC                 DEF
SA = 2
```

IMMAGAZZINAMENTO DEL FORMATO DEI DATI

Una delle piu' importanti caratteristiche di questa stampante e' la sua capacita' di formattazione dei dati.
Questa capacita' consente di allineare i dati a destra o a sinistra oppure di allineare i dati numerici a secondo della posizione del punto decimale.

Esempio:

```
10 OPEN2,4,2
20 OPEN1,4,1
30 PRINT#2,"$$$.99"
40 PRINT#1,.05
```

Risultato:

```
$.05
```

In questo esempio sono stati utilizzati dei comandi

per il trasferimento delle istruzioni di formattazione dei dati dal computer alla memoria della stampante in modo che questi siano correttamente interpretati.
Una volta completata la stampa comparira' sullo schermo la scritta READY.

CARATTERI PER LA FORMATTAZIONE

Un formato e' specificato da alcuni caratteri che definiscono il tipo di stampa che volete ottenere.
I caratteri per la formattazione si dividono in 3 gruppi:

NUMERICI 9,Z,$,S,.,-

ALFANUMERICI A

INTERVALLI (vuoti)

I campi sono definiti dalla combinazione di questi caratteri di formattazione.
E' possibile preparare delle stringhe di formattazione di 136/250 caratteri.
Quelli in eccesso saranno stampati su una riga successiva.

N U M E R I C I

9 - Specifica la posizione di una cifra in un campo numerico.
Se non vi e' alcuna cifra da stampare non viene stampato nessun carattere, ma viene lasciato uno spazio.

Z - Specifica anche questa la posizione di una cifra in un campo numerico. A differenza del 9, questo carattere forza la stampa di uno "0" se in quella posizione non e' presente alcuna cifra da stampare.
Cio' puo' essere utile nel caso in cui si desideri far precedere la cifra da stampare da ZERI.

$ - Se e' specificato un $ solo allora il campo e' trattato come un importo in dollari con una posizione fissa del segno DOLLARO ($).

$123
$1234.00

E' un parametro raramente utilizzabile nelle nostre stampe.

S - Quando precede un campo numerico, fa si che in questa posizione sia stampato il segno di + o -.

. - Definisce la posizione del punto decimale e viene stampato nella posizione specificata.

- - Specifica un segno in coda al numero. Se il numero e' positivo viene stampato uno spazio bianco. Un campo numerico non puo' avere contemporaneamente i segni S e -, perche' in questo caso sara' riconosciuto solo il simbolo S.

A L F A N U M E R I C I

La lettera A rappresenta una posizione di un campo alfabetico.
All' interno del campo gli spazi che precedono sono troncati. Il campo e' incolonnato a sinistra ed e' riempito a destra da spazi bianchi.
Il carattere (CHR$(160)) non viene cancellato se e' messo nella prima posizione del campo.

Esempio:

```
10 OPEN 2,4,2
20 OPEN 1,4,1
30 PRINT#2,"A  AA  AAA"
40 PRINT#1,"CBM"CHR$(29)"CBM"CHR$(29)"CBM
50 CLOSE2:CLOSE1
```

Il risultato:

```
        C CB  CBM
```

CARATTERI FISSI NELLE STRINGHE DI FORMATTAZIONE.

Questi sono caratteri che devono essere stampati esattamente come appaiono nella stringa di formattazione.

Questi caratteri sono preceduti nelle stringhe dal carattere (RVS cioe' da un REVERSE).
Un utilizzo di questi e' l' incolonnamento e la stampa di tabelle con caratteri verticali che delimitino i singoli campi.

SA = 3

NUMERO DI LINEE PER PAGINA

Questo indirizzzo secondario permette di variare il numero di linee per pagina.
Affinche' si possa utilizzare questa opzione e' necessario iniziare la stampa con uno speciale carattere (CHR$(147)) che consente di resettare il contarighe della stampante e di abilitare la funzione.
Quando non viene specificato la lunghezza della pagina e la funzione viene attivata la stampante e' fissata per carta a 66 righe (11 pollici).

Esempio

```
10 OPEN 4,4
20 OPEN 1,4,1
30 OPEN 2,4,2
40 OPEN 3,4,3
50 A$="999    9999    99.99999999    99.99999999"
55 PRINT#2,A$
60 PRINT#3,CHR$(60)
70 PRINT#4,CHR$(147)
80 FOR I=1TO99
90 PRINT#1,I;I*I;SQR(I);I↑(1/3)
100 NEXTI
110 PRINT#4,CHR$(19)
120 CLOSE4:CLOSE3:CLOSE2:CLOSE1
```

SA = 4

ABILITAZIONE DEI MESSAGGI DIAGNOSTICI

Trasmettendo questo indirizzo secondario , ogni volta che si verifica una condizione di errore appare un messaggio diagnostico.
Quando si ha un errore di formattazion ne viene stampato il tipo, viene disabilitato il formato di stampa e viene segnalato un puntatore in corrispondenza del campo errato.
Se questo indirizzo secondario non e' stato inserito nel programma, nel caso si incorra in un errore, i caratteri saranno scaricati direttamente in stampa senza tener conto della formattazione.
Sia che i messaggi di errore con questo parametro siano abilitati o no, un OVERFLOW di un campo numerico, (naturalmente da stampare) e' sempre indicato da un campo con asterischi (*****).
In questo caso, come in altri casi di errori che non portino comunque al RESET di sistema, l' indirizzo secondario viene posto =0 e tutti i dati percio' ricevuti dalla stampante saranno stampati ESATTAMENTE come vengono ricevuti.

Esempio

```
*PE:C*

 10 OPEN4,4,4:PRINT#4:CLOSE4
 20 OPEN25,4,25
 30 PRINT#25
 40 CLOSE25
 50 OPEN1,4:CMD1:LIST
READY.
```

MESSAGGI DIAGNOSTICI POSSIBILI

PE:L Linee per pagina fuori campo.

E' stato tentato di fissare un numero di linee per pagina fuori dell' intervallo compreso fra 13 e 128 tramite l' impiego dell' indirizzo secondario SA=3. Ilcomando in questo caso viene ignorato e rimane la lunghezza di pagina fissata in precedenza.

PE:C Comando errato.

E' stato dato alla stampante un indirizzo secondario sconosciuto.
Questo comando viene ignorato.

PE:M Errore sul formato dei dati.

E' stato ordinato di stampare un dato ALFANUMERICO in campo numerico.
Il primo carattere stampato dopo questo messaggio d' errore e' il carattere sbagliato.

PE:E Errore di esponente.

Il dato numerico inviato alla periferica per essere stampato comunque in un campo di formattazione numerica, ha un esponente non valido.
La forma corretta di un numero esponenziale e':

n.nnn+EE o n.nnnn-EE

PE:F Errore di formattazione.

Il dato inviato con l' uso di SA=2 contiene dati non correttamente formattati o con una sintassi non

riconoscibile.

PE:T Errore sul carattere di termine.

Per carattere di termine si intende un ritorno carrello CHR$(13), un LINE FEED CHR$(10) o un' insieme dei due caratteri in sequenza.
Prima di inviare un nuovo SA e' necessario ricorrere all' invio di uno dei caratteri di TERMINE appena detti. Se questa azione non verra' compiuta, ecapita anche spesso, avremo quest' ultimo errore.

SA = 5

DEFINIZIONE DI UN CARATERE PROGRAMMABILE.

Per questo indirizzo vedere quanto e' riportato nell' esame della prossima stampante.

SA = 6

FISSA L' INTERLINEA

Questo indirizzo secondario controlla lo spazio fra una linea di stampa e la successiva.
Ci sono 144 passi per pollice e possono essere assegnati dei valori da 1 a 127.
Con un valore 18 viene prodotta una spaziatura pari a 8 linee per pollice.
Il valore di DEFAULT e' 24 che produce una spaziatura standard di 6 linee per pollice.

Esempio:

```
10 OPEN4,4
20 OPEN6,4,6
30 FORI=1TO127STEP4
40 PRINT#6,CHR$(I)
50 PRINT#4,"HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH"
60 NEXT
```

SA = 7 e SA = 8

SELEZIONA IL MODO MAIUSCOLO E MINUSCOLO

Il modo normale della stampante e' la stampa in maiuscolo.
Con SA=7 avremo la stampa in minuscolo, mentre con SA=8 in maiuscolo.
L' utilizo alternato di questi due SA permette di ottenere qualsiasi tipo di carattere si desideri dei due SET grafici disponibili.

SA = 9

DISABILITA I MESSAGGI DIAGNOSTICI

Se e' importante l' abilitazione di messaggi di errore durante la fase di DEBUG di un programma, non altrettanto puo' dirsi quando esso e' operativo.
Comunque con l' invio di questo indirizzo secondario e' completamente disabilitato la stampa di messaggi diagnostici.

SA = 10

RESETTA

Serve per resettare la stampante.

FUNZIONI SPECIALI

Nella tavola seguente sono riportati i caratteri CHR$ da inviare alla stampante per ottenere particolari funzioni.

CARATTERI ALLARGATI

Si puo' raddoppiare la larghezza di un qualsiasi carattere utilizzando il CHR$(1).
Mentre normalmente la matrice di stampa e' 8x8, con questo sistema si ottiene una matrice di 8x16.
Qualora questo comando venga ripetuto piu' volte in una linea, sara' raddoppiato ogni volta il numero di punti in orizontale, come si puo' vedere dall' esempio sottostante.

Esempio:

```
10 OPEN4,4
20 PRINT#4,"H"CHR$(1)"E"CHR$(1)"L"CHR$(1)"L"CHR$(1)"O"
```

Risultato:

```
HELLO
```

PAGING

Tramite il carattere CHR$(147) come accennato nell' indirizzo secondario 3, si utilizza la possibilita' della stampante di impaginare lasciando libere 3 linee in testa al foglio e 3 linee in fondo, cioe'

al piede di pagina.
Attraverso il SA=3 e' possibile variare il numero di linee per pagina.

CARATTERI MAIUSCOLI E MINUSCOLI

All' accensione della stampante i caratterei disponibili sono quelli ASCII/Graphics.
Benche' non sia possibile sullo schermo utilizzare simultaneamente le minuscole ed i simboli grafici, la stampante invece vi permette di usarli sulla stessa linea.
Questo risultato e' ottenuto con il carattere CURSOR/UP per le maiuscole e CURSOR/DOWN per le minuscole.

Esempio

```
10 OPEN4,4
20 PRINT#4,"□C■OMMODORE"
```

Risultato:

Commodore

CAMPO INVERSO

Immettendo in una stringa un carattere CHR$(18) (RVS/ON), e' possibile far stampare dei caratteri in campo inverso nello stesso modo che si puo' fare sul video.
Per disabilitare questo tipo di stampa e' sufficiente, come per il video, mettere un carattere CHR$(146) (RVS/OFF).
E' consigliabile non utilizzare il tipo di stampa REVERSE per piu' di 5 linee consecutive perche' puo' essere dannoso per la testina.

Esempio

```
10 OPEN5,4
20 PRINT#5,"■C■OMMODORE"
```

Risultato:

RITORNO CARRELLO

Se si cerca di stampare piu' di 80 caratteri per linea dopo l' ottantesimo carattere verra' forzato un ritorno carrello (RETURN), con l' interlinea ed i caratteri in piu' saranno stampati nella linea successiva.
E' possibile pero' riscrivere 2 o piu' volte sulla solita linea terminando la prima parte con un CHR$(141) (RETURN SHIFT).
Questo comando annulla la scrittura in campo inverso, l' allargamento dei caratti e il campo fra apici.

GLI APICI o virgolette

Se e' trasmesso un numero dispari di APICI, i caratteri di controllo vengono stampati.
Questo puo' essere particolarmente utile quando si esegue un LIST di un programma.

CAPITOLO VENTICINQUESIMO

In questo e nel capitolo che segue daremo una descrizione, e ripetendo anche concetti dei quali abbiamo gia' parlato, della stampante MPS 801 della COMMODORE, indubbiamente la piu' diffusa sulle serie VIC 20 e CBM 64.
Il fatto di ripetere tanti concetti e' dovuto proprio alla diffusione di questa stampante.

PROBLEMI E CONSIGLI

In questo breve paragrafo vogliamo presentare alcuni consigli per ottenere un migliore funzionamento della vostra stampante e la soluzione ad alcuni problemi che si possono verificare.

PRECAUZIONI DA OSSERVARE

*Eseguendo la fase di accensione e spegnimento in successione attendere almeno 2/3 secondi prima di riaccendere perche' la stampante potrebbe non essere propriamente inizializzata.

*Non esporre mai la stampante, come del resto i dischi direttamente ed in maniera prolungata alla luce solare.

*Cercare di non spegnere la stampante mentre lavora in particolare durante la fase di ritorno carrello.

*Non cercare di operare sulla testina quando c'e' tensione, cioe' prima di aver spento la stampante.

*Non cercare di stampare senza carta o senza nastro perche' potreste danneggiare la testina di stampa.

*Durante la fase di stampa, cercare di non spingere al massimo la densita' di punti, che si verifica in modo particolare o stampando in REVERSE o programmando dei caratteri troppo pieni.
In questo modo si accorcia la vita della testina e dei meccanismi stessi.

PROBLEMI E POSSIBILI RIMEDI

Ci auguriamo che non si presentino dei problemi.Comunque vogliamo dare dei consigli per non dover ricorrere sempre alla manutenzione anche quando, con un minimo di calma e di attenzione potete risolvere da voi.

NESSUNA STAMPA - INDICATORE SPENTO

1-La stampante e' realmente spenta oppure la connessione con il cavo di alimentazione, sia alla stampante stessa o alla spina, non e' perfetta.
Controllare che l' interruttore sia nella giusta posizione e che i cavi di alimentazione siano correttamente collegati e ci sia tensione sulla linea.

2-E' partito il fusibile ed e' quindi necessario sostituirlo con uno della stessa qualita' e dimensione.
Cercare di utilizzare solo lo stesso tipo di fusibili.

NESSUNA STAMPA - INDICATORE ACCESO

1-Controllare la connessione fra la stampante ed il computer.

2-Puo' essere stato male inserito o fissato il nastro. Vedere di rimetterlo a posto senza forzare.

FUNZIONA MA LA CARTA NON AVANZA

Normalmente si e' incastrata la carta. Rimuovere il foglio e riposizionarlo correttamente.

CARATTERI CHE SPORCANO O LABILI

1-Controllare con l' apposita levetta la pressione di stampa.
Questa levetta si trova sulla sinistra della stampante.

2-Cattivo inserimento del nastro.
Rimuovere e rimettere correttamente

3-Cartuccia o troppo vecchia ed avremo allora dei

caratteri troppo leggeri e quindi scarsamente leggibili. Oppure nastro difettoso che non scorre o che e' troppo carico di inchiostro.
Ricordarsi che le cartucce invecchiano anche se non si stampa molto perche' l' inchiostro si asciuga comunque con l' andare del tempo. In particolare questo fenomeno si verifica in ambienti particolarmente caldi o con la vicinanza di un radiatore o con l' esposizione alla luce diretta del sole.

A partire dal successivo paragrafo e terminando nel prossimo capitolo, cercheremo di dare un panorama quanto piu' dettagliato possibile della stampante MPS 801.
Siamo certi che la ripetizione e/o l' approfondimento di alcuni concetti non sara' inutile.

COMANDI ASSOCIATI CON LA STAMPA

Quando si desidera stampare qualcosa si tratta essenzialmente di trasferire quanto e' visualizzabile nello schermo sulla stampante stessa, anche se dobbiamo sottolineare che questo concetto e' molto generico.
Una piccola serie di comandi BASIC consentono di effettuare questo trasferimento. Infatti la maggior parte della sintassi dei comandi BASIC resta la stessa.
Non dimenticate di premere il tasto RETURN dopo ogni linea di informazione.

IL COMANDO OPEN

Questo comando crea una corrispondenza univoca fra il numero di file e una periferica.
Il NUMERO DI FILE LOGICO o (lfn) puo' essere un qualsiasi numero fra 1 e 255.
Il numero della periferica o DEVICE NUMBER (dn) si riferisce fisicamente alla periferica alla quale si invia il file, in questo caso cioe' le informazioni da stampare.

NOTA

Non ci dilungheremo sul concetto di file perche' e' stato abbondantemente spiegato nel resto del volume.

Il primo numero di periferica e' anche conosciuto con il nome di INDIRIZZO PRIMARIO.
Se si sta utilizzando una MPS-801 normalmente il numero della periferica e' il 4.
L' INDIRIZZO SECONDARIO o (sa) e' opzionale ed e' unico.
L' indirizzo secondario su questa stampante consente le seguenti opzioni di stampa:

Sa =0 Stampa i dati esattamente come ricevuti.

Sa =6 Fissa gli spazi fra le linee.

Sa =7 Seleziona il modo normale di stampa o BUSINNES MODE

Sa =8 Seleziona il modo grafico di stampa o GRAPHIC MODE.

Sa =10 Esegue il reset della stampante.

La sintassi del comando OPEN e' la seguente:

OPEN lfn,dn oppure OPEN lfn,dn,sa

Esempi:

OPEN 99,4 OPEN 1,4,0

OPEN 2,4 OPEN 26,4,7

IL COMANDO CMD

Questo comando trasferisce il controllo dal computer alla stampante.
Il numero di file logico o lfn deve essere lo stesso di quello che e' stato scelto per il comando OPEN.

NOTA

Se si sta usando piu' di un comando OPEN nel programma, ogni comando CMD deve avere lo stesso numero del corrispondente comando OPEN.

A differenza che nell' uso di un comando PRINT£, che vedremo in seguito, la linea di ricezione dati sulla periferica, in questo caso la stampante, e' lasciata aperta.
Si dice che la stampante e' in LISTENING cioe' in ascolto, dopo l' apertura del canale di trasmissione dati a cui ci siamo riferiti con il comando CMD.
Questo spiega perche' ogni volta che si da il comando CMD la stampante stampa un READY. esattamente come se fosse il video. Restera' quindi il canale aperto ed in attesa di istruzioni.
Quindi a questo punto un qualsiasi comando PRINT o LIST sara' direttamente eseguito.

La sintassi del comando CMD e' la seguente:

CMD lfn

Esempio:

```
CMD 99
CMD 2
```

NOTA

Notare che i numeri di files logici del comando CMD corrispondono ai numeri di files logici del comando OPEN.

IL COMANDO PRINT #

Il comando PRINT #opera allo stesso modo del PRINT in un normale programma BASIC tranne che la visualizzazione del risultato (OUTPUT) avviene sulla stampante (in questo caso) invece che sul video.
Dopo che i dati inviati alla stampante sono terminati, il lfn e' automaticamente chiuso.
Questa operazione e' chiamata UNLISTENING e ci permette di inviare altri dati alla stampante solo se viene riaperto il lfn.
Notare che il comando PRINT # deve essere scritto esattamente cosi' cioe' senza lasciare spazi fra il termine PRINT e il puond altrimenti non otteremo alcun risulatato.

NOTA

E' anche importante ricordare che non si puo' abbreviare, come normalmente si fa nei programmi

BASIC, il comando con un punto interrogativo.

La sintassi del comando PRINT#e':

```
PRINT#lfn,dati
```

Esempio:

```
PRINT#99,"BUONGIORNO"

PRINT#2,CHR$(124),123,63,76
```

IL COMANDO CLOSE

STAMPA IN MODO DIRETTO

Dopo avere esaminato brevemente i vari comandi di stampa, vediamo il sistema di applicarli.
Il modo diretto consente di comunicare con la stampante inserendo comandi di stampa direttamente dalla tastiera del computer.

Il seguente esempio illustra l' intero sistema di modo DIRETTO in un programmino BASIC.
In questo programma, un file che contiene un singolo comando BASIC e' scritto nella memoria del computer.
Successivamente il file viene aperto.
Viene quindi aperto il canale di uscita che va alla stampante e la stampante stessa e' messa in posizione di ascolto o ricevimento dati (LISTENING) utilizzando il comando CMD.

Il file viene ora listato, cioe' ne viene eseguito il LIST e poi il canale di uscita e' chiuso utilizzando il comando PRINT#.
Al termine il file viene chiuso con un comando CLOSE.
A questo punto il collegamento con la stampante sara' chiuso, cioe' ne verra' chiuso il canale di trasmissione dati ed il computer sara' pronto per eseguire nuovi comandi.
plotter

Digitare	Visualizzazione	Stampa
① 10?"TEST"	10?"TEST"	
② OPEN3,4	OPEN3,4 READY.	
③ CMD3 .	CMD3	READY.
④ LIST	LIST	10 PRINT"TEST" READY.
⑤ PRINT#3	PRINT#3	
⑥ CLOSE3	CLOSE3 READY.	

Vediamo di spiegare il programma punto per punto. Il lettore dovrebbe sopratutto cercare di comprendere appieno il processo completo di funzionamento.

1 - I dati vengono immessi nella memoria del computer.

2 - Viene eseguita l' apertura del file a cui viene assegnato il numero di file logico 3. Il 4, cioe' il numero che segue la virgola dice che il file e' disponibile per la stampante.

3 - La stampante viene messa in posizione di

LISTENING cioe' di ascolto.

4 - Viene eseguito il listato su stampante. Dopo l' esecuzione di questa fase la stampante resta in posizione di ascolto.

5 - Viene usato il comando PRINT #per mettere la stampante in posizione di non ascolto UNLISTENING.

6 - Il file viene chiuso in modo tale che il file logico 3 possa essere utilizzato in altra maniera.

STAMPA SOTTO CONTROLLO DI PROGRAMMA

Ora che abbiamo visto come operare direttamente sulla stampante da tastiera vediamo come e' possibile lavorare con un programma BASIC che ci consenta di stampare.
Il seguente programma puo' essere residente sia nellaa memoria dl computer che su floppy disk o su cassetta.

```
10 OPEN 3,4
20 CMD3
30 PRINT"PROGRAMMA"
40 LIST
```

Eseguendo il RUN avremo il seguente risultato:

NOTA IMPORTANTE

Quando si usa un comando LIST durante l' esecuzione di un programma e' necessario ricordarsi di

eseguire la chiusura del canale con un PRINT #. Digitare quindi un comando di CLOSE per chiudere il file dopo che il programma ha terminato di girare. Ricordarsi che quindi le operazioni di chiusura sono 2, quella del canale di comunicazione con la stampante e quella del file.
Dobbiamo sottolineare tuttavia che sebbene il programma presentato giri correttamente non e' certo quanto di meglio si potrebbe sperare in quanto a metodologia di corretta programmazione.
Percio' si suggerisce di utilizzare solo il comando CMD in modo diretto per eseguire in particolare il listato di un programma.

INDIRIZZO SECONDARIO

Questa stampante ha come abbiamo visto diversi tipi di indirizzi secondari due dei quali molto importanti e che sono:

Sa = 0 o modo grafico

Sa = 7 o modo BUSINESS

Il seguente esempio mostra come utilizzare l' indirizzo secondario.

```
100 OPEN4,4
110 PRINT#4,"      ASCII CODE TABLE"
120 PRINT#4
130 A$="0123456789ABCDEF"
140 PRINT#4,"  | ";
150 FORI=1TO16:PRINT#4,MID$(A$,I,1)" ";:NEXT
160 PRINT#4
170 PRINT#4,"--+",
```

```
180 FOR I=1TO16:PRINT#4,"──";:NEXT
190 PRINT#4
200 FOR I=1TO16
210 PRINT#4,MID$(A$,I,1)"│ ";
220 FOR J=I-1TO255STEP16
230 IF J<32 THEN GOSUB330:GOTO260
240 IF J>127 AND J<160 THEN GOSUB330:GOTO260
250 PRINT#4,CHR$(J)" ";
260 NEXT J
270 PRINT#4
280 NEXT I
290 PRINT#4:PRINT#4
300 CLOSE4
310 END
320 :
330 PRINT#4,"  ";
340 RETURN
```

Facendo girare il precedente programma avremo come risultato:

ASCII CODE TABLE

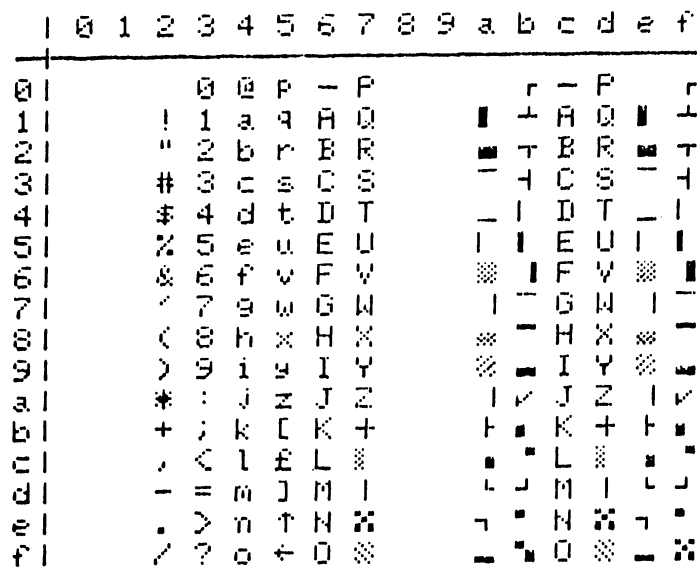
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	─	┐				┌	─	┐		┌
1			!	1	A	Q	♠	●			▌	┴	♠	●	▌	┴
2			"	2	B	R	│	─			▀	┬	│	─	▀	┬
3			#	3	C	S	─	♥			▔	┤	─	♥	▔	┤
4			$	4	D	T	─	│			_	│	─	│	_	│
5			%	5	E	U	─	╱			▕	▌	─	╱	▕	▌
6			&	6	F	V	─	╳			▒	▐	─	╳	▒	▐
7			'	7	G	W	│	○			▕	▔	│	○	▕	▔
8			(	8	H	X	│	♣			▒	▀	│	♣	▒	▀
9			)	9	I	Y	╲	│			◤	▄	╲	│	◤	▄
A			*	:	J	Z	╲	◆			▕	┘	╲	◆	▕	┘
B			+	;	K	[	╱	+			├	▪	╱	+	├	▪
C			,	<	L	£	└	▒			▪	▪	└	▒	▪	▪
D			-	=	M	]	╲	│			└	┘	╲	│	└	┘
E			.	>	N	↑	╱	π			┐	▪	╱	π	┐	▪
F			/	?	O	←	┌	◥			▁	▪	┌	◥	▁	π

Se si cambiano le linee 100 e 110 nel precedente programma:

```
100 OPEN4,4,7
110 PRINT#4,"      ASCII CODE TABLE"
```

Facendo girare otterremo:

ascii code table



CAPITOLO VENTISEIESIMO

MODI DI STAMPA E CODICI DI CONTROLLO

Alcuni dei programmi visti in precedenza piu' il comando PRINT #possono essere utilizzati insieme ad una serie di codici CHR$ che consentiranno quindi le seguenti funzioni:

DESCRIZIONE	CODICE
Inserimento modo grafico	CHR$(8)
Line feed dopo la stampa	CHR$(10)
Ritorno carrello	CHR$(13)
Caratteri in doppia grandezza	CHR$(14)
Modo caratteri standard	CHR$(15)
Fissa il tab in testata	CHR$(16)
Modo cursor DOWN	CHR$(17)
Inizio campo in REVERSE	CHR$(18)
Ripetizione selezione grafica	CHR$(26)
Dichiar. indirizzo punti (vedi)	CHR$(27)
Modo cursor UP	CHR$(145)
Fine utilizzo campo REVERSE	CHR$(146)

MODO DI CARATTERI STANDARD

Questo e' il modo di stampa normale che quindi

abbiamo selezionato naturalmente all' accensione della stampante.
Tuttavia quando si passa ad altro modo di stampa sara' necessario poi, per tornare al modo standard inviare alla stampante un codice CHR$(15).
Nel seguente esempio selezioniamo il modo di caratteri in doppia grandezza per scrivere la frase contenuta fra virgolette, ma poi per evitare di continuare con questo tipo di stampa, che evidentemente ci serviva solo per un rigo, magari per un titolo, dobbiamo tornare ad un carattere normale con l' invio di un codice CHR$(15).

```
10 OPEN1,4
20 PRINT#1,CHR$(14)"MPS-801 PRINTER"
30 PRINT#1,CHR$(15)
40 CMD1:LIST

READY.
```

MODO CARATTERI IN DOPPIA GRANDEZZA

Questo tipo di stampante puo' essere utilizzata per ottenere dei caratteri in doppia grandezza.
Questo sara' una peculiarita' interessante per titoli, evidenziazioni ed altro che potranno essere quindi utilizzati con relativa facilita'.

```
10 OPEN2,4
20 PRINT#2,CHR$(14)"MPS-801 PRINTER"
30 CLOSE2

RUN

MPS-801 PRINTER
```

MODO GRAFICO

Utilizzando il codice CHR$(8) si entra nel modo grafico.
Questo modo consente di disegnare e stampare segni grafici con l' uso di valori inseriti in un normale comando DATA.
Ogni comando DATA e' composto di numeri che rappresentano una riga di punti e che verranno quindi utilizzati con un comando READ.
Per disegnare questi nuovi caratteri non compresi fra quelli a disposizione, diamo ora un piccolo schema da seguire accuratamente.
Fare particolare attenzione al fatto che ogni numero nel comando DATA corrisponde ad una riga del carattere.

PASSI E SCHEMA DI DISEGNO CARATTERI

1 - Munirsi di un foglio di carta possibilmente a quadretti.

2 - Numerare otto righe come segue:

1
2
4
8
16
32
64
128

3 - Disegnare un grafico a punti come nell' esempio sottoriportato

```
  1  o o * * o o o
  2  o * o o * * o
  4  * o o o * o o
  8  * o o o o o o
 16  * o o o * o o
 32  o * o o * * o
 64  o o * * o o o

128
```

4 - Eseguire la somma dei numeri corrispondenti al disegno ricordandosi di eseguire alla fine il complemento a 128 aggiungendo tale numero al primo risultato ottenuto.
Partendo quindi con le colonne da sinistra verso destra avremo i seguenti risultati:

I colonna 4+8+16+128=156

II colonna 2+32+128 =162

III coloonna 1+64+128 =193

IV colonna 1+64+128 =193

V colonna 2+4+16+32+128=182

VI colonna 2+32+128 =162

5 - Di conseguenza il comando DATA relativo al carattere disegnato sara':

DATA 156,162,193,182,162

che andra' inserito in programma.

Nell' esempio seguente avremo la stampa del carattere disegnato seguito dal contenuto della stringa fra virgolette per 4 volte.
Infatti il primo ciclo di FOR...NEXT serve per eseguira la lettura dei valori del DATA, mentre il secondo serve per la stampa di 4 volte del messaggio e del caratere speciale disegnato.

```
10 DATA156,162,193,193,182,162
20 FOR I=1TO6
30 READ A
40 A$=A$+CHR$(A)
50 NEXT I
60 OPEN3,4
70 FOR I=1TO4
80 PRINT#3,CHR$(8)A$;
90 PRINT#3,CHR$(15)" COMMODORE"
100 NEXT I
110 CLOSE3
```

RUN

```
  C COMMODORE
  C COMMODORE
  C COMMODORE
  C COMMODORE
```

STAMPA IN UNA DATA POSIZIONE

Con il codice CHR$(16) si puo' determinare l' inizio della posizione di stampa.
Questa possibilita' e' di conseguenza all' assegnazione di un numero di 2 digit che segue il

carattere CHR$(16).

Esempi.

```
10 OPEN4,4
20 FOR I=1TO4
30 PRINT#4,"0123456789";
40 NEXT I
50 PRINT#4,CHR$(10);
60 PRINT#4,CHR$(16)CHR$(48)CHR$(56)
   "MPS-801";
70 PRINT#4,CHR$(16)CHR$(51)CHR$(48)
   "PRINTER"
80 CLOSE4
 RUN

 01234567890123456789012345678901234567890123456789

 . .        MPS-801                    PRINTER
```

In questo esempio e' importante fare attenzione al fatto che nelle linee 50, 60 e 70 i codici CHR$ hanno i seguenti significati.

CHR$(10) = Line feed

CHR$(16) = Selezione del modod di stampa

CHR$(48) = Equivalente al valore 0

CHR$(56) = Equivalente al valore 8

CHR$(51) = Equivalente al valore 3

Se osserviamo infatti la linea 60 ordina di eseguire la stampa all' ottavò carattere (08) mentre la linea 70 fa eseguire la stampa alla posizione 30.

Nel secondo esempio, identico al primo possiamo notare che i valori di assegnazione della posizione di stampa possono essere dati anche all' interno della stringa stessa da stampare.
Questo puo' sembrare di primo avviso un si stema piu' semplice, ma e' bene notare invece che il valore, sempre di due caratteri all' interno del CHR$ come nel primo esempio puo' essere dato come variabile da incrmentare per successivi posizionamenti, cosa che invece non si puo' evidentemente fare all' interno di una stringa.

```
10 OPEN4,4
20 FOR I=1TO4
30 PRINT#4,"0123456789";
40 NEXT I
50 PRINT#4,CHR$(10);
60 PRINT#4,CHR$(16)"08MPS-801";
70 PRINT#4,CHR$(16)"30PRINTER"
80 CLOSE4
```

RUN

```
0123456789012345678901234567890123456789

        MPS-801                  PRINTER
```

.POSIZIONE DI INIZIO DELLA STAMPA

Usando il carattere CHR$(27) l' indirizzo assoluto di inizio stampa puo' essere variato, cioe' e' possibile stampare dei caratteri con una spaziatura di X punti (DOTS) seguendo il formato specificato in questa tabella:

CHR$(27) CHR$(16) CHR$(HP) CHR$(LP)

I due caratteri che seguono CHR$(27) e CHR$(16) stanno ad identificare la posizione espressa in binario (BYTE ALTO - BYTE BASSO) della posizione di stampa in punti (DOTS).

Esempio:

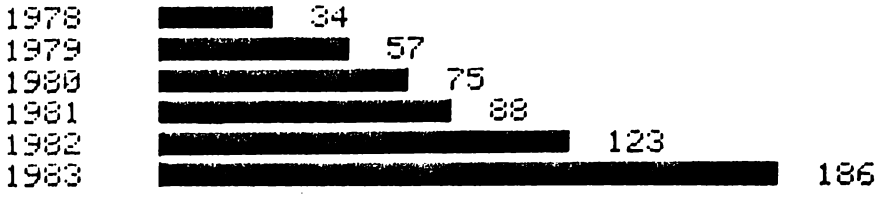
Dovendo stampare un carattere con posizione X=300 (cioe' a 300 punti dall' inizio riga) PH e PL saranno uguali:

PH = INT(X/256):PL = X-(PH*256)

PH = 1 : PL = 44

RIPETIZIONE DEI DATI GRAFICI

Utilizzando il codice CHR$(26) si possono eseguire delle ripetizioni di dati in forma grafica dove si desidera.
Vediamo subito con un esempio come possa essere utilizzato questo sistema:




```
10 OPEN6,4
20 FORI=1TO6:READA:A$=A$+CHR$(A):NEXT
30 FORJ=1TO4:READB:B$=B$+CHR$(B):NEXT
40 FORK=1TO6:READC:C$=CHR$(C)
50 D$=STR$(1977+K)
60 PRINT#6,CHR$(15)D$A$C$B$C
70 NEXT
80 CMD6:LIST
90 DATA 8,27,16,0,53,26
100 DATA 255,59,15,32
110 DATA 34,57,75,88,123,186

READY.
```

MODO DI CURSOR UP

Inviando alla stampante il carattere CHR$(145), i caratteri che verranno stampati di seguito, saranno in GRAPHIC MODE (CURSOR UP MODE), fino a quando non effettueremo un ritorno carrello.

Esempio

```
10 OPEN7,4,7:CU$=CHR$(145):CD$=CHR$(17)
20 PRINT#7,CU$"♠      "CD$"SPADE"
30 PRINT#7,CU$"♥      "CD$"HEART"
40 PRINT#7,CU$"♦      "CD$"DIAMOND"
50 PRINT#7,CU$"♣      "CD$"CLUB"
60 CLOSE7
```

```
♠      spade
♥      heart
♦      diamond
♣      club
```

MODO DI CURSOR DOWN

Inviando un carattere CHR$(17) verra' effettuata la stampa in modo MAIUSCOLO/MINUSCOLO fino a quando non viene trovato un CURSOR/UP.

Esempio.

```
10 OPEN8,4:CU$=CHR$(145):CD$=CHR$(17)
20 PRINT#8,CD$"SPADE     "CU$"♠"
30 PRINT#8,CD$"HEART     "CU$"♥"
40 PRINT#8,CD$"DIAMOND   "CU$"♦"
50 PRINT#8,CD$"CLUB      "CU$"♣"
60 CLOSE8
```

```
sPade     ♠
heart     ♥
diamond   ♦
club      ♣
```

STAMPA IN REVERSE

Selezionando il codice CHR$(18) si puo' stampare una qualsiasi informazione in REVERSE.
Come si puo' agevolmente vedere dall' esempio si ha una stampa di una striscia nera con i caratteri in bianco.

```
10 open9,4,7
20 print#9,chr$(18)" Personal Computer  "
30 print#9,chr$(18)" Dot Matrix Printer "
40 cmd9:list

ready.
```

RESET DEL MODO REVERSE

Allo stesso modo che e' stato fissato il modo REVERSE puo' essere tolto cioe' posto in OFF con il codice CHR$(146).
Vediamone un' esempio applicativo:

```
 Personal Computer
 Dot Matrix Printer
```

```
10 open10,4,7
20 print#10,chr$(18)" Personal Computer  "
30 print#10,chr$(146)" Dot Matrix Printer "
40 cmd10:list

ready.
```

LINE FEED

E' possibile avere due tipi di interlinea:

Caratteri normali 6 linee per pollice.

Cratteri grafici 9 linee per pollice.

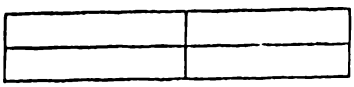
Normalmente e' fissata la prima funzione. Infatti le linee stampate normalmente sono spaziate l' una da l' altra. E' possibile comunque eliminare lo spazio

esistente fra le due linee in modo da poter effettuare la stampa di tabelle o linee verticali con tratto continuo.

Esempio:

```
10 OPEN12,4:SI$=CHR$(15):BS$=CHR$(8)
20 PRINT#12,SI$"┌──────────┬──────────┐"BS$
30 PRINT#12,SI$"│          │          │"BS$
40 PRINT#12,SI$"├──────────┼──────────┤"BS$
50 PRINT#12,SI$"│          │          │"BS$
60 PRINT#12,SI$"└──────────┴──────────┘"
70 CLOSE4
```

RUN



BUFFER DELLA STAMPANTE

Questa stampante dispone di una parte di memoria RAM che viene utilizzata come area di transito di dati. In pratica come Buffer.
Questo Buffer e' di 90 caratteri interamente utilizzabili per la stampa tranne che per 1 o piu' Bytes che vengono riservati per i codici di controllo.
Normalmente la stampante lavora in modo automatico per cui non e' da preoccuparsi in che modo vengono trattati dati e non c' e' realmente pericolo di incorrere in un overflow.
Non sara7 comunque male sapere dell' esistenza di questo buffer e per chi vuole opeerare in linguaggio piu' evoluto approfittarne.
Vediamo qualche approfondimento nel paragrafo seguente.

STAMPA AUTOMATICA

Avremo la stampa automatica a 3 condizioni.
Per capire esattamente cosa succede e' necessario conoscere un po' come lavora la stampante.
Prima di tutto ogni carattere e' fatto da sei righe di punti. Per ogni linea di stampa possiamo avere fino ad un massimo di 80 caratteri. Ricordiamo che gli spazi contano come un carattere di 6 righe.
Questo ci consente di affermare che ci sono 480 caratteri per linea.
Vediamo ora le condizioni presentate all' inizio.

a - Quando il Buffere si riempe durante la fase di ingresso dati.

b - Quando la stampante SENTE che state usando piu' di 480 punti per linea. Naturalmente parliamodi punti ricordandoci che possono essere selezionati questi 480 punti singolarmente.

c - Quando entrambi i casi visti in precedenza , a e b, si verrificano allo stesso tempo.

Cosa accade dunque quando si verificano queste condizioni?

A - Quando il buffer si riempe durante la fase di stampa.....
Ma RICORDA dove aveva cessato di stampare in modo tale che possa continuare a stampare quando e se si desidera.
Oppure la stampa continuera' quando il Buffer sara' di nuovo pieno.

B - Quando la stampante utilizza piu' di 480

caratteri allora viene stampata la linea e poi si ha un arresto ed e' pronta per altre informazioni.

C - Quando entrambi i casi si verificano allora vengono SCARICATI solo i primi 80 caratteri e quindi stampati.
La testina di stampa si muove quindi sulla successiva linea.
A questo punto la stampante puo' fare due cose:

1 Riprendere un qualsiasi carattere lasciato nel buffer, che e' piu' lungo di 80 caratteri come abbiamo visto prima, e lo stampa.

2 Andare in READY ed attende per ulteriori informazioni.

Tavole

Cassetta

$F542-$F646

Routine RAM per LOAD

Esegue un caricamento da cassetta o da periferica seriale secondo il contenuto di $BA (attuale numero di periferica).
In alternativa carica, se $B9=0, i registri X e Y che contengono l' indirizzo di caricamento.
L' indirizzo alto di caricamentoo e' riportato in X e Y.

$F675-$F734

Routine RAM per SAVE

Esegue la registrazione su cassetta o su periferica seriale sempre secondo il contenuto di $BA.
L' inizio della memoria da salvare e' presente nell' Accumulatore come indirizzo indiretto, mentre l' indirizzo di fine memoria e' nei registri X e Y.

$F7AF-$F889

Ricerca le informazioni necessarie dalla memoria di nastro.
Legge il nastro fino a quando non viene trovato uno dei seguenti tipi di blocchi:

-Testata di un file di dati basic.

-File di caricamento prog. basic.

$F88A-$F89E

Gruppo misto di routine di controllo.Con i seguenti indirizzi e funzioni:

$F8AB

Controllo motore cassetta.

$F8B7

Controllo per i tasti PLAY e RECORD. Vedere pero' quanto scritto in precedenza circa la reale portata di questo controllo.

$F8C0

Lettura della testata del blocco in ingresso.

$F8C9

Lettura e caricamento del blocco in ingresso.

$F8E3

Scrive la testata del blocco.

$F8F4

Punto d' ingresso per inizio operazioni su nastro.

$F95D

Fissa la temporizzazione per dipolo.

$F98E-$FABC

Routine di lettura. Il carattere letto vine

trasferito alla routine di indirizzo $BF.

$FABD-$FBE9

Routine che consente la manipolazione di bytes per la lettura cassetta.

$FBEA-$FD21

Routines di scrittura cassetta.
La locazione di indirizzo $BE e7 il contatore per i record.

Se $BE=2 allora sara' la prima testata.
Se $BE=1 allora sara' il primo dato.
Se $BE=0 sara' allora il secondo dato.

NOTA

Ricordiamo che i vettori di IRQ vengono cambiati durante le operazioni su cassetta per cui se l' utente ha resettato questi vettori questi devono essere ripristinati ai loro valori dopo aver utilizzato la cassetta.

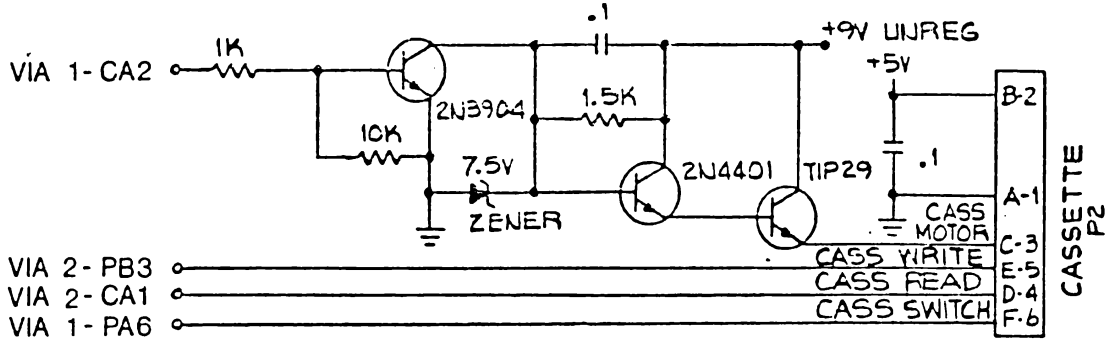
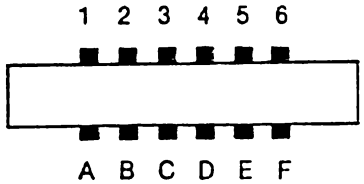


Fig. 35 - Circuito della cassetta e collegamenti agli integrati 6522.



PIN #	TYPE
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SWITCH

Fig. 36 - Funzioni e descrizioni dei pins del connettore cassette.

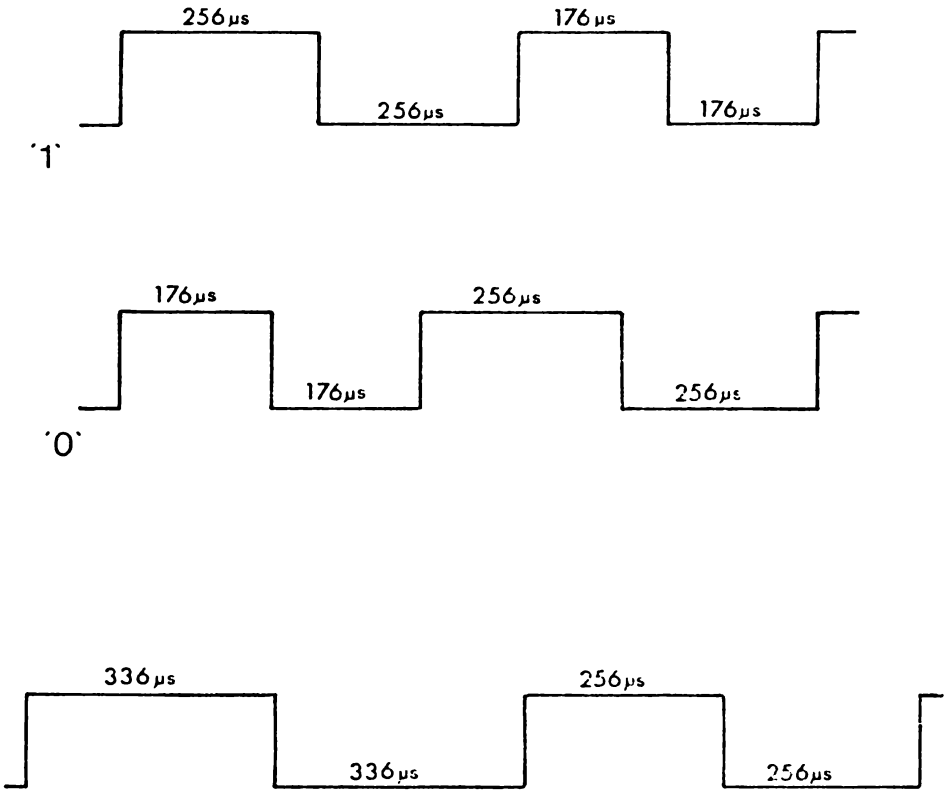


Fig. 37 - Forme d'onda per il registratore.

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
RIDATA	00AA	170	00AA	00C2	Buffer ingresso byte per RS232/Tempor izzazione cassetta
RIPRTY	00AB	171	00AB	00C3	INPUT di parita' per RS232/Controllo cassetta
SAL	00AC-00AD	172-173	00AC	00C7	Puntatore: Buffer del nastro/scrollin g di schermo
EAL	00AE-00AF	174-175	00AE	00C9	Indirizzo di fine nastro/ Fine del pr ogramma
TAPE1	00B2-00B3	178-179	00B2	00D6	Puntatore inizio Buffer nastro
BITTS	00B4	180	00B4	00CE	Cont. bit uscita per RS232/Flag tempo orizzazione lettura nastro
NXTBIT	00B5	181	00B5	00CF	Prossimo bit da inviare a RS232/Lett nastro per EOT
RODATA	00B6	182	00B6	00D0	Buffer per Byte in uscita su RS232/Fl ag errore di lettura nastro
FNLEN	00B7	183	00B7	00D1	N. di caratteri del nome del File
LA	00B8	184	00B8	00B2	N. dell'attuale file logico
SA	00B9	185	00B9	00D3	Attuale indirizzo secondario
FA	00BA	186	00BA	00D4	Attuale n. di periferica
FNADR	00BB-00BC	187-188	00BB	00D5	Puntatore: attuale indirizzo del nome del file
ROPRTY	00BD	189	00BD	00DD	Parita' di uscita per RS232/Temp. di cassetta
FSBLK	00BE	190	00BE	00DE	Contatore dei blocchi in I/O per nast

- Tavola XI -

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
					ro
MYCH	00BF	191	00BF	00DF	Buffer per porta seriale
CAS1	00C0	192	00C0	00F9	Interruttore per motore cassetta
STAL	00C1-00C2	193-194	00C1	00FB	Indirizzo di partenza per LOAD e per scrittura cassetta
MEMUSS	00C3-00C4	195-196	00C3	----	Temp. di LOAD per cassetta
SCHAR	00D7	215	00D7	00D9	Valore dell' attuale carattere in ingresso/ Ultimo carattere uscito
ASCWRK	00FF-010A	255-266	00FF	00FF	Area assembler per conversione virgola mobile in ASCII
BAD	0100-013E	256-***	0100	0100	Errore di input da nastro
STACK	0100-01FF	256-***	0100	0100	Area di STACK hardware per 6510
BSTACK	013F-01FF	319-***	013F	013F	Area di STACK per il Basic
LAT	0259-0262	601-610	0259	0251	Tavola KERNAL: numero del file logico attivo
FAT	0263-026C	611-620	0263	025B	Tavola KERNAL: numero di periferica per ogni file
SAT	026D-0276	621-630	026D	0265	Tavola KERNAL: indirizzo secondario di ogni file
IRQTMP	029F-02A0	671-672	029F	----	Immagazzinamento temporaneo del vettore IRQ durante le operazioni su nastro
TODSNS	02A2	674	----	----	Controllo sensore cassetta durante le operazioni di I/O

- Tavola XII -

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
TRDTMP	02A3	675	----	----	Immagazzinamento temporaneo durante l a lettura nastro
TD1IRQ	02A4	676	----	----	Indicatore temporaneo di IRQ durante la lettura del nastro
TBUFFR	033C-03FB	828-1019	033C	027A	Buffer del nastro per operazioni di I /O

- Tavola XIII -

Programmi

MENUEV/U

```
 10 REM*********************************
 20 REM                                 *
 30 REM PROGRAMMA PER LA CREAZIONE      *
 35 REM                                 *
 36 REM DI UN MENU DELLA DIRECTORY      *
 38 REM                                 *
 39 REM       SOLO SU DISCO             *
 40 REM                                 *
 41 REM*********************************
 43 REM
 44 REM
 50 REM*********************************
 51 REM                                 *
 52 REM VIENE ESEGUITO ANCHE UN OR-     *
 54 REM DINAMENTO ALFABETICO DEI        *
 56 REM PROGRAMMI PRESENTI NELLA        *
 58 REM DIRECTORY STESSA CON LA SUC-    *
 60 REM CESSIVA CREAZIONE DI UN FI-     *
 62 REM LE SEQUENZIALE SU DISCO         *
 64 REM                                 *
 66 REM*********************************
 68 REM
 70 REM
100 POKE53280,6
101 POKE53281,6
102 DIM TB$(150)
103 PRINT"(CLEAR)"CHR$(14)
104 DI=8
105 C=
110 OPEN8,DI,8,":    DIRECTORY    "
111 OPEN15,DI,15
112 INPUT# 15,EN$,ED$,ET$,ES$
120 INPUT# 8,DN$
121 IFEN$<>"00"THENCLOSE8:CLOSE15:GOTO 420

130 INPUT# 8,TB$(C)
131 IFST=66THEN160
```

```
140 IFASC(LEFT$(TB$(C),1))=10THENTB$(C)=RIGHT$
    (TB$(C),LEN(TB$(C))-1):GOTO 140
150 C=C+1
151 GOTO 130
160 CLOSE8
161 CLOSE15
162 M=C-1
163 TB$(C)="
164 L1$="(RVS)"
165 FOR C=1TO40
166 L1$=L1$+" "
167 NEXT
170 L1$=L1$+"(RVSOFF)"
171 A=1
180 POKE53281,1
181 PRINT"(CLEAR)■";
182 FOR C=1TO4
183 PRINTL1$;
184 NEXT
185 FOR C=1TO2
186 PRINT"■"L1$;
190 NEXT
191 PRINT"(HOME)";
200 PRINTSPC(8)"(RVS)■IREZIONE ♥VILUPPO
     V64"SPC(10)"(C)1984   ┐└┐│└┌/╲┐♠ X♠└┘
    ";
210 PRINT"♠┘/┌♥┘ ┐X╲."SPC(11)"♠UTORE: ┘RAN
    Z │┘┐/     "SPC(57)""DN$SPC(8)L1$;
220 FOR C=ATOA+14
221 PRINT"(RVS)"RIGHT$("      "+STR$(C),5)"(RVSOFF)
    "TB$(C)SPC(29-LEN(TB$(C)));
230 PRINT"■(RVS)       ";
231 NEXT
232 PRINTL1$;
240 FOR A=1TO2
241 PRINT"■"L1$;
242 NEXT
```

MENUEV/U

```
243 PRINTLEFT$(L1$,40)"(UP)"
250 PRINTSPC(9)"(RVS)(2 UP)-1 = PAGINA SUC
    CESSIVA"SPC(18)"-3 = CREA LA DIRECTORY
    ";
260 PRINTSPC(18)"-5 = USCITA (HOME)"
261 POKE2023,160
262 POKE56295,6
263 POKE198,0
264 B$=""
270 GETA$
271 IFA$=""THEN270
280 IFASC(A$)=13THEN370
290 IFA$="■"THEN350:REM" F1 ******
300 IFA$="■"THEN410:REM" F3 ******
310 IFA$="■"THENPOKE53280,3:PRINT"(CLEAR)(3 DOWN)■
    RUN(HOME)":END :REM" F5 ******
320 IFASC(A$)=20THENB$="":PRINT"(HOME)■
    (RVS)   (RVSOFF)":B$="":GOTO 270
330 IFA$<"0"ORA$>"9"ORLEN(B$)=3THEN270
340 B$=B$+A$
341 PRINT"(HOME)(4 DOWN)■"B$
342 GOTO 270
350 IFC>MTHENA=1:GOTO 180
360 A=C
361 GOTO 180
370 IFVAL(B$)>MTHENPRINT"(HOME)(4 DOWN)■??
    ?":FOR C=1TO1500:NEXT :A$=CHR$(20):GOTO
    320
380 POKE53281,6
381 POKE53280,14
382 PRINT"(RVSOFF)(CLEAR)(3 DOWN)■■LOAD"CHR$
    (34)TB$(VAL(B$))CHR$(34)",8"
390 REM   PINTRIGHT$(STR$(DI),LEN(STR$(DI
    ))-1)
400 PRINT"(HOME)"
401 POKE198,4
402 POKE631,13
```

```
403 POKE632,82
404 POKE633,213
405 POKE634,13
406 END
410 PRINT"(HOME)(4 DOWN)■WAIT"
420 OPEN8,DI,0,"$0"
421 FOR C=1TO8
422 GET#8,A$
423 NEXT
424 C=1
425 DN$=""
426 FOR C=1TO16
430 GET#8,A$
431 DN$=DN$+A$
432 NEXT
433 GET#8,A$
434 GET#8,A$
435 $=DN$+"   "
436 GET#8,A$
440 DN$=DN$+A$
441 GET#8,A$
442 DN$=DN$+A$
443 GET#8,A$
444 GET#8,A$
445 DN$=DN$+"   "+A$
450 GET#8,A$
451 DN$=DN$+A$
452 GET#8,A$
453 C=1
460 FOR A=1TO4
461 GET#8,A$
462 NEXT
463 PN$=""
464 TY$=""
470 GET#8,A$
471 IFST<>0THEN560
480 IFA$=""THEN560
```

```
490 ASC(A$)<>34THEN470
500 GET#8,A$
501 IFASC(A$)<>34THENPN$=PN$+A$:GOTO 500
510 GET#8,A$
511 IFASC(A$)=32THEN510
520 TY$=TY$+A$
521 GET#8,A$
522 IFA$<>""THEN520
530 IFLEFT$(TY$,3)<>"PRG"THEN460
540 IFLEFT$(PN$,1)=" "THEN460
550 TB$(C)=PN$
551 C=C+1
552 IFST=0THEN460
560 CLOSE8
561 OPEN15,DI,15,"S:    DIRECTORY    "
562 CLOSE15
570 Z$="◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆"
571 OPEN8,DI,8,":    DIRECTORY    ,S,W"
572 PRINT#8,DN$
580 FOR A=1TOC-1
581 C$=Z$
582 FOR B=1TOC-1
583 IFC$<TB$(B)THEN600
590 C$=TB$(B)
591 D=B
600 NEXT
601 PRINT#8,C$
602 TB$(D)=Z$
603 NEXT
604 CLOSE8
605 CLR
606 GOTO 100
```

```
DONE

 30 REM      UN-COMPACTOR
 50 REM    BY:ROBERT W.BAKER
 70 REM 15 WINDSOR DR.,ATCO,NJ 08004
100 REM>MOD.RAY THOMAS.830116 TO GIVE  CHO
    ICE OF DRIVES & FIX 'LAST LINE' BUG <
110 GOTO  270
120 :
130 REM>>>>>>> SUBROUTINES <<<<<<<<<
140 :
150 GOSUB 160:V1=V
160 GET#5,C$:GOSUB 190  170 IF C$="" THEN V=0:RETURN
180 V=ASC(C$):RETURN
190 INPUT# 15,EN,EM$,ET,ES
200 IF EN=0 THEN RETURN
210 PRINT"(CLEAR)(RVS)DISK ERROR":PRINT
220 PRINTEN;EM$;ET;ES
230 GOTO  1030
240 :
250 REM ***** INITIALALIZATION *****
260 :
270 PRINT"(CLEAR)";SPC(10);"(RVS)UN-COMPAC
    TOR
280 INPUT (2 DOWN)" (RVS)INPUT(RVSOFF) FIL
    E IN DRIVE#";D1$
290 INPUT "(RVS)OUTPUT(RVSOFF) FILE IN DRI
    VE#";D2$
300 INPUT "(RVS)INPUT FILENAME(RVSOFF)";FL
    $
310 DIM  C(256)
320 OPEN 15,8,15
330 OPEN 5,8,5,D1$+":"+FL$+",P,R"
340 GOSUB 190
350 PRINT:PRINT"OK, WORKING ON LINE# .....

360 FO$=LEFT$(FL$,14)+(DOWN)"/U"
370 PRINT#15,"S"+D2$+":"+FO$
```

```
DONE

380 OPEN 6,8,6,D2$+":"+FO$+",P,W"
390 GOSUB 190
400 GOSUB 150:PRINT#6,CHR$(V1);C$;
410 F=1:GOTO  80
420 :
430 REM ***** OUTPUT THIS LINE#
440 :
450 LN=NL:IF LK=0 THEN 1010
460 PRINTLN,
470 PRINT#6,CHR$(1);CHR$(1);
480 PRINT#6,CHR$(LL);CHR$(LH);
490 :
500 REM ***** READ THIS BASIC PGM LINE
510 :
520 X=1
530 GOSUB 160:C(X)=V
540 IF V>0 THEN X=X+1:GOTO  530
550 :
560 REM ***** GET NEXT LIN & LINE#
570 :
580 GOSUB 150:LK=V+V1:IF Z=1 THEN 600
585 IF LK=0 THEN Z=1
590 GOSUB 150:NL=V1+(256*V):LL=V1:LH=V
600 IF F THEN F=0:GOTO  450
610 :
620 REM ***** BREAK UP LINE IF POSSIBLE
630 :
640 X=1
650 :
660 REM SKIP IF NOT COLON
670 :
680 IF C(X)<>58 THEN 810
690 IF X=1 THEN 950
700 LN=LN+1:IFLN>=NL THEN 950
710 PRINT#6,CHR$(0);CHR$(1);CHR$(1);
720 H=INT(LN/256):L=LN-(256*H)
730 PRINT#6,CHR$(L);CHR$(H);
```

DONE

```
 740 X=X+1:IF C(X)=32 OR C(X)=58 THEN 740
 750 GOTO  680
 760 :
 770 REM COPY REST OF LINE IF ---
 780 REM GOTO, END, RUN, IF, RETURN
 790 REM REM, STOP, LIST, CONT
 800 :
 810 IF C(X)<128 OR CX))>155 THEN 910
 820 IF C(X)=128 OR C(X))>153 THEN 850
 830 IF C(X)<137 OR C(X))>144 THEN 910
 840 IF C(X)=140 OR C(X)=141 THEN 910
 850 PRINT#6,CHR$(C(X));
 860 IF C(X))>0 THEN X=X+1:GOTO  850
 870 GOTO  450
 880 :
 890 REM SKIP IF NOT QUOTE
 900 :
 910 IF C(X)<>34 THEN 950
 920 PRINT#6,CHR$(C(X));:X=X+1
 930 IF C(X)=34 OR C(X)=0 THEN 950
 940 GOTO  920
 950 PRINT#6,CHR$(C(X));
 960 IF C(X))>0 THEN X=X+1:GOTO  680
 970 GOTO  450
 980 :
 990 REM *** END OF BASIC PROGRAM
1000 :
1010 PRINT#6,CHR$(0);CHR$(0);
1020 PRINT"(CLEAR)(RVS)DONE":PRINT:PRINT
1030 CLOSE 5:CLOSE 6:CLOSE 15
```

COPIA FILE

```
1000 REM UN DRIVE
1010 POKE52,31:POKE56,31:CLR
1020 FOR I=0 TO 173
1030 READ  D:POKE 7936+I,D:NEXT
1040 POKE 785,0:POKE 786,31
1050 PL$=CHR$(19)
1060 FOR  I=1 TO 24 :PL$=PL$+CHR$(17):NEXT

1070 FOR  I=1 TO 39:CL$=CL$+" ":NEXT
1080 CL$=CL$+CHR$(13)+CL$+CHR$(13)
1090 CL$=LEFT$(PL$,21)+CL$+CHR$(145)+CHR$(1
     45)
1100 OPEN 15,8,15:REM OPE ERRORE DI CANALE

1110 GOTO  1830
1120 REM LEGGE LA DIRECTORY
1130 POKE 785,0:REM SELEZIONA PER LA LETTUR
     A
1140 PRINT#15,"I":OPEN2,8,2,"$"
1150 NB=USR(2):REM CARICA LA DIRECTORY DENT
     RO LA MEMORIA
1160 CLOSE 2:GOSUB 1680:REM SEGNALA ERRORE
1170 RETURN
1180 REM STAMPA IL NOME DEL DISCO
1190 POKE785,6:REM LEGGE IL ARATTERE
1200 PRINTLEFT$(PL$,3);CHR$(18);
1210 FOR  K=142 TO 169
1220 PRINT CHR$(USR(K));:NEXT
1230 PRINT CHR$(146)
1240 RETURN
1250 REM STAMPA DIRECTORY
1260 POKE 785,6:REM
1270 IX=(DE-1)*32+254-INT((DE-1)/8)*2
1280 IF USR(IX)<128 THEN RETURN
1290 TY=USR(IX)-128:IF TY=0 OR TY>3 THEN RETURN

1300 FOR  K=IX+3 TO IX+18
```

```
1310 PRINT CHR$(USR(K));
1320 NEXT :PRINT TAB(16);"";
1330 PRINT MID$("SPU",TY,1):RETURN
1340 REM
1350 REM DA LA DESTINAZIONE DEL DISCO
1360 PRINT CL$;
1370 PRINT "INSERISCI LA DESTINAZIONE DEL D
     ISCO."
1380 PRINT "PREMI ";CHR$(18);"RETURN";
1390 PRINT CHR$(146);" QUANDO SEI PRONTO"
1400 GET DM$:IF DM$=""THEN GOTO  1400
1410 IF DM$<>CHR$(13)THEN GOTO  1400
1420 PRINT#15,"I":RETURN 1430 REM
1440 REM
1450 PRINT CL$;:IF EN=63 GOTO  1500
1460 IF UE=128 THEN PRINT NM$;" LARGO ";
1470 IF EN=62 THEN PRINT NM$;" NOT FOUND
        ";
1480 PRINT "ENTER FILE TYPE":NM$="":EN=0:UE
     =0
1490 INPUT  NM$,TY$:RETURN
1500 PRINT NM$;"ESISTE, CAMBIA IL NOME"
1510 EN=0:NM$="":INPUT  NM$: RETURN
1520 REM
1530 REM LEGGI I FILE
1540 POKE 785,0:REM SELEZIONA LA FUNZIONE C
     HE LEGGE
1550 OPEN 2,8,2,NM$+","+TY$+",R"
1560 GOSUB 1760:IF EN<>0 THEN CLOSE 2:RETURN

1570 NB=USR(2):REM LEGGI I FILE
1580 CLOSE 2:GOSUB 1680:REM CONTROLLA IL RI
     SULTATO
1590 RETURN
1600 REM
1610 REM SCRIVI IL FILE
```

```
1620 POKE 785,3:REM SELEZIONA LA SRITTURA
1630 OPEN 2,8,2,NM$+",+TY$+",W"
1640 GOSUB 1760:IF EN<>0 THEN CLOSE 2:RETURN

1650 NB=USR(2):REM SCRIVI I BYTES
1660 CLOSE 2:GOSUB 1680:REM  CONTROLLA IL R
     ISULTATO
1670 RETURN
1680 REM CONTR. USR( ) I VALORI
1690 NB=NB-(NB<0)*65536
1700 IF NB<61440 THEN UE=0:RETURN
1710 IF NB=61440 THEN UE=128:RETURN
1720 PRINTCL$;
1730 IF PEEK(785)=0 THEN PRINT "RED";
1740 IF PEEK(785)=3 THEN PRINT "WRITE";
1750 PRINT " ERRORE # ";NB-61440:END
1760 REM LEGGI ERROREC DI CANALE
1770 INPUT# 15,EN,EM$,ET,ES
1780 IF EN=0 OR EN=62 OR EN=63 THEN RETURN
1790 CLOSE 15:PRINT CL$;
1800 PRINT "DISCO IN ERRORE#";EN
1810 PRINT LEFT$(PL$,22);EM,M$:END
1820 REM
1830 REM ROTIN
1840 PRINT CHR$(147);:REM CLEAR SCHERMO
1850 PRINT "(RVS)          DRIVE FILE COPY
                    (RVSOFF)";
1860 INPUT "INSERITO DISCO SORGENTE(2 RIGHT)Y
     (3 LEFT)";A$:PRINT"(2 DOWN)"
1870 PRINT"
       (UP)"
1880 PRINT:PRINT "READING DIRECTORY"
1890 GOSUB 1120:REM LEGGI LA DIRECTORY
1900 GOSUB 1180:REM TITOLO
1910 NS=INT((NB+1)/254)
1920 IF NS=1THEN GOTO 400:REM NO FILE
1930 DS=1
```

```
1940 IF NS<=DS THEN DS=1:REM
1950 PRINTLEFT$(PL$,4);
1960 BE=(DS-1)*8+1:LE=BE+15
1970 IFDS+1=NS THEN LE=BE+8
1980 FOR  DE=BE TO LE
1990 GOSUB 1250:NEXT :REM DISPLAY FILE
2000 FOR  I=PEEK(214)TO 20:REM CLEAR
2010 PRINT "                          "
2020 NEXT
2030 GOSUB 1440:REM
2040 IF NM$="" THEN DS=D+2:GOTO  1940
2050 REM COPY IL FILE
2060 PRINT CL$;
2070 PRINT "READING ";NM$
2080 GOSUB1530:REM LEGGI I FILE
2090 IF EN=62 OR UE=128 GOTO  2030
2100 BC=NB:REM SAVE COUNT
2110 GOSUB1350:REM
2120 PRINT CL$;"WRITING ";NM$
2130 GOSUB 1610:REM WRITE IL FILE
2140 IF EN=0 GOTO  2200
2150 GOSUB 1440:REM
2160 IF NM$<>""GOTO  2120
2170 PRINTCL$;"FILE NON COPIATO"
2180 GOTO  2270
2190 REM
2200 REM DISP. VCOMPLET. STATUS
2210 PRINT CL$;
2220 IF NB<>BC GOTO   2250
2230 PRINT "COPY SUCCESFUL.  ";
2240 GOTO  2260
2250 PRINT "ERRORE ";NB;" OUT ";
2260 PRINT BC;"BYTES COPIED ."
2270 PRINT "PREMI ";CHR$(18);
2280 PRINT "RETURN";CHR$(146);
2290 PRINT " COIO UN ALTRO FILE"
2300 GET DM$:IF DM$="" GOTO  1820
```

```
2310 END
2320 REM FILE ERRRORE
2330 PRINT LEFT$(PL$,21);
2340 PRINT "NO FILES NEL DISCO"
2350 PRINT "PREMI RETURN PER TRY ";
2360 PRINT "UN ALTRO DISCO ";
2370 GET DM$:IF DM$="" GOTO
2380 IF DM$=CHR$(13) GOTO  1830
2390 END
2400 END :REM NESSUN FILE
2410 DATA  76,15,31,7,88,31,76,154
2420 DATA  31,108,3,0,108,5,0,32
2430 DATA  9,31,152,170,32,198,255,168
2440 DATA  176,57,169,0,133,251,169,32
2450 DATA  133,252,160,0,32,207,255,145
2460 DATA  251,230,251,208,2,230,252,165
2470 DATA  144,208,13,165,252,201,160,144
2480 DATA  235,32,204,255,160,0,240,19
2490 DATA  64,144,32,204,255,192,64,208
2500 DATA  10,164,251,165,252,56,233,32
2510 DATA  76,12,31,169,240,76,12,31
2520 DATA  32,9,31,152,170,32,201,255
2530 DATA  168,176,240,165,251,141,174,31
2540 DATA  165,252,141,175,31,169,0,133
2550 DATA  251,169,32,133,252,160,0,177
2560 DATA  251,32,210,255,164,144,208,20
2570 DATA  230,251,208,2,230,252,165,252
2580 DATA  205,175,31,144,234,165,251,205
2590 DATA  174,31,144,227,32,204,255,76
2600 DATA  73,31,32,9,31,132,251,24
2610 DATA  105,32,133,252,160,0,177,251
2620 DATA  168,169,0,76,12,31
```

RECUPERO

```
100 POKE53280,0:POKE53281,0:PRINT"(CLEAR)■
    "
140 DIM FE$(32),TF$(5):X$="":R$=""
150 FOR I=0TO5:READ TF$(I):NEXT :CC=0
160 K=0:CH=9:S=1:TR=18:T=TR:DE$="8":DR$="0
    ":DR=VAL(DR$)
165 M$="--------------------------------"
170 IN$="TIPO   NOME FILE      TR.SET.LUNG.
    "
180 PRINT"(CLEAR)";TAB(13)"(RVS)RECUPERO
    ILE"
182 PRINT"(3 DOWN)   1 - RECUPERO FILE CAN
    CELLATO(DOWN)"
190 PRINT"   2 - RECUPERO TUTTI FILE CANCE
    L.(DOWN)"
200 PRINT"   3 - CONVALIDA(DOWN)"
210 PRINT"   4 - DIRECTORY AMPLIATA(DOWN)"

220 PRINT"   5 - CANCELLAZIONE FILE(DOWN)"
    :PRINT"   6 - ERRORE DOS(DOWN)"
222 PRINT"   7 - FINE(DOWN)"
230 PRINT"     (SPACE) PER TORNARE AL MENU
    (DOWN)"
240 PRINT"(RVS)1:RE1 2:RECT 3:CONV 4:DIR
    5:CANC 6:ERR 7:FINE (RVSOFF)";
245 GETR$:IFR$=""THEN245
260 IFR$("1"ORR$)"7"THEN245
270 R=VAL(R$):PRINTR$:RT=0
280 ONRGOTO 450,960,1000,1500,1800,2010,20
    00
290 END
300 TT=ASC(FE$(1)):SS=ASC(FE$(2))
310 FT=ASC(FE$(0))AND127
320 IFFT>5THENPRINT"TIPO ERRAT":FT=5
330 TF$=TF$(FT)
340 LF=ASC(FE$(28))+256*ASC(FE$(29))
350 TT$=RIGHT$("  "+STR$(TT),3)
```

```
360 SS$=RIGHT$("   "+STR$(SS),3)
370 LF$=RIGHT$("      "+STR$(LF),5)
380 PRINT"(RVS)"TF$"(RVSOFF) "NF$TAB(20);
390 IFSPTHENPRINT#4,"(RVS)"TF$"(RVSOFF) "N
    F$;
400 PRINTTT$;SS$;LF$
410 IFSPTHENPRINT#4,TT$SS$LF$
420 RETURN
450 PRINT"(CLEAR)(RVS)RICERCA DI UN CERTO
    FILE"
460 GOSUB750
470 PRINT"NOME DEL FLEA RECUPERARE"
480 INPUT NO$:L=LEN(NO$):FOR K=1TOL
490 IFMID$(NO$,K,1)="*"THENL=K-1:K=80
500 NEXT K:IFK<80THENL=16
510 NO$=LEFT$(NO$+"                    ",L)
520 PRINT"STO CERCANDO    (RVS)"NO$
530 REM
540 T=TR:OPEN1,DE,15:PRINT#1,"I"+DR$
550 S=1:OPENCH,DE,CH,"#"+DR$
560 PRINT#1,"U1:"CH;DR;T;S
570 PRINT#1,"B-P:"CH;0:GET#CH,T$,S$
580 FOR FE=1TO8
590 GETR$:IFR$=" "THEN710
600 FOR CC=0TO31:GET#CH,X$
610 IFX$=""THENX$=CHR$(0)
620 FE$(CC)=X$:NEXT :NF$=""
630 FOR K=3TO18:NF$=NF$+FE$(K):NEXT
640 IFNO$=LEFT$(NF$,L)THEN1200
650 NEXT FE
660 IFT$=""THENT$=CHR$(0)
670 IFS$=""THENS$=CHR$(0)
680 T=ASC(T$):S=ASC(S$)
690 IFT<>0THEN560
700 PRINT"(RVS)NON TROAO(RVSOFF)"NF$
710 CLOSECH:CLOSE1:CLOSE4:PRINT:PRINT:GOTO
    240
```

RECUPERO

```
 720 REM
 730 INPUT# 1,CE$,ER$,ET$,ES$:IFCE$="00"THEN
     RETURN
 740 PRINTCE$,ER$,ET$,ES$:GOTO 710
 750 PRINT"(CLEAR)"
 760 INPUT "DISPOSITIVO   8(3 LEFT)";DE$
 770 IFDE$(>"8"ANDDE$(>"9"ANDDE$(>"10"ANDDE
     $(>"11"THEN760
 780 DE=VAL(DE$)
 790 RETURN
 800 REM
 810 PRINT"(RVS)0=DEL 1=SEQ 2=PRG 3=USR 4=R
     L 5=MENU"
 820 INPUT "(RVS)TIPO(RVSOFF)   2(3 LEFT)";
     R:IFR(0ORR)5THEN820
 830 IFR=5THEN710
 840 PO=2+32*(FE-1):TP=R+128
 850 PRINT#1,"U1:"CH,DR,T,S
 860 PRINT#1,"B-P:"CH;PO
 870 PRINT#CH,CHR$(TP);
 880 PRINT#1,"U2:"CH;DR;T;S
 890 RETURN
 900 REM
 910 IFMID$(NF$,1,1)=CHR$(0)THEN1420
 920 INPUT "(RVS)DA RECUPERARE (S/N/END)(RVSOFF)
       N(3 LEFT)";R$
 930 IFR$=S"THENGOSUB800:PRINT#1,"B-P:"CH;
     PO-32
 940 IFLEFT$(R$,1)="E"THEN1420
 950 RETURN
 960 REM
 970 PRINT"(CLEAR)(RVS)RECUPERO DI TUTTI I
     FILE"
 980 RT=-1:GOTO 1510
1000 PRINT"(CLEAR)(RVS)CONVALIDA (RVSOFF)";
     :GOSUB750
1010 CLOSE1:OPEN1,DE,15:PRINT#1,"V"+DR$
```

```
1020 GOSUB720:GOTO 710
1200 REM
1210 PRINT"TROVATO IL FILE "NF$
1220 PRINTIN$
1230 GOSUB300
1330 PRINT"CHE TIPO ERA  ?"
1340 GOSUB800
1420 INPUT "CONVALIDA (S/N)   S(3 LEFT)";R$

1430 IFR$="S"THEN1000
1440 GOTO 710
1500 PRINT"(CLEAR)   (RVS)VISUALIZZAZIONE/S
     TAMPA  DIRECTORY"
1510 GOSUB750:INPUT "STAMPA   N(3 LEFT)";R$

1520 SP=0:IFR$="S"THENSP=-1:OPEN4,4
1530 GOSUB1900:PRINTM$:PRINTIN$:PRINTM$:S=1

1540 IFSPTHENPRINT#4,M$:PRINT#4,IN$:PRINT#4
     ,M$
1550 IFSTTHENPRINT"STAMPANTE SPENTA":END
1560 OPEN1,8,15:PRINT#1,"I"+DR$
1570 OPENCH,8,CH,"#"+DR$:GOSUB720
1580 PRINT#1,"U1:"CH;DR;T;S
1590 PRINT#1,"B-P:"CH=0:GET#CH,T$,S$
1600 FOR FE=1TO8
1610 GETR$:IFR$=" "THEN1730
1620 FOR CC=0TO31:GET#CH,X$
1630 IFX$=""THENX$=CHR$(0)
1640 FE$(CC)=X$:NEXT :NF$=""
1650 FOR K=3TO18:NF$=NF$+FE$(K):NEXT
1660 GOSUB300:IFRT AND(FT=0)THENGOSUB900
1670 IFFE$(3)=(0)THEN1730
1680 NEXT FE
1690 IFT$=""THENT$=CHR$(0)
1700 IFS$=""THENS$=CHR$(0)
1710 T=ASC(T$):S=ASC(S$)
```

```
1720 IFT<>0THEN1580
1730 IFRT THEN1420
1740 T=TR:GOTO 710
1800 PRINT"(CLEAR)(RVS)CANCELLAZIONE FILE":
     GOSUB750
1810 PRINT"NOME DEL FILE DA CANCELLARE"
1820 INPUT R$:CLOSE1:OPEN1,DE,15
1830 PRINT#1,"S:"+R$
1840 GOSUB720:GOTO 710
1900 REM
1910 PO=6:IFTR=18THENPO=144
1920 S=0:OPEN1,DE,15,"I"+DR$:GOSUB720
1930 OPENCH,DE,CH,"#":PRINT#1,"U1:"CH;DR;TR
     ;S
1940 PRINT#1,"B-P:"CH;PO:NF$=""
1950 FOR K=1TO23:GET#CH,X$:NF$=NF$+X$:NEXT

1955 PRINT"(CLEAR)DR NOME DISCHETTO     ID F
     ORM.":PRINTM$
1956 IFSPTHENPRINT#4,"DR NOME DISCHETTO
     ID FORM.":PRINT#4,M$
1960 PRINT"(RVS)"DR$"   "NF$"       ":IFSPTHEN
     PRINT#4,"(RVS)"DR$"   "NF$"         "
1970 CLOSECH:CLOSE1:RETURN
1980 DATA DEL,SEQ,PRG,USR,REL,XXX
1990 REM
2000 PRINT"(CLEAR)(RVS)FINE":CLOSE1:CLOSECH
     :CLOSE4:END
2010 GOSUB750:OPEN1,DE,15:INPUT# 1,CE$,ER$,
     ET$,ES$
2020 PRINT"NUMERO ERRORE     "CE$
2030 PRINT"TIPO ERRORE       "ER$
2040 PRINT"TRACCIA           "ET$
2050 PRINT"SETTORE           "ES$
2060 GOTO 710
```

DOS HEX DUMPER

```
  1 OPEN15,8,15
  4 FOR T=32768TO33767:IFPEEK(T)=32THENNEXT

  5 CO=8:LL=1:IFT>32768+50THENCO=16:LL=2
  6 GOSUB1000
  7 GOSUB2000
 10 INPUT "START ADDRESS";AD
 15 INPUT "END ADDRESS";EA
 20 INPUT "(DOWN)(RVS)S(RVSOFF)CREEN OR (RVS)P
    (RVSOFF)RINTER";P$
 21 DV=3:IFP$="P"THENDV=4
 22 OPEN4,DV:IFDV=4THENCO=16
 23 PRINT#4,"(CLEAR)START ADDRESS=";MID$(STR$
    (AD,2);"  END ADDRESS=";MID$(STR$(EA)
    ,2)
 24 PRINT#4,"--------------";:FOR T=1TOLEN(
    STR$(AD)):PRINT#4,"-";:NEXT
 25 PRINT#4,"  ------------";:FOR T=1TOLEN(
    STR$(EA)):PRINT#4,"-";:NEXT
 27 PRINT#4:PRINT#4
 30 GOSUB 490
 40 CLOSE4:PRINT"(2 DOWN)":GOTO 10
300 A%=AD/B:AD$=H$(A%)+H$(AD-A%*B):RETURN
490 GOSUB300:PRINT#4,"> ";AD$;" ";
500 FOR  T=1 TO CO
510 GOSUB10000:AD=AD+1:IP(T)=D
520 PRINT#4,H$(D);" ";
530 NEXT  T
550 PRINT#4,"(RVS)";
560 FOR  T=1 TO CO:N=IP(T)ANDNOT128
565 IFN<32THENN=46
570 PRINT#4,CHR$(N);
580 NEXT :PRINT#4
581 IF AD>=EA THEN RETURN
582 GETX$:IFX$=""THEN490
584 GETX$:IFX$=""THEN584
600 GOTO 490
```

DOS HEX DUMPER

```
 1000 PRINT"(CLEAR)";CHR$(160);
 1005 A$="****                          ****":PRINTTAB(
      (20*LL)-LEN(A$)/2);A$
 1010 A$="*****  DOS HEX DUMPER  *****":PRINT
      TAB((20*LL)-LEN(A$)/2);A$
 1020 A$="****                          ****":PRINTTAB(
      (20*LL)-LEN(A$)/2);A$
 1030 PRINT"
 1035 A$=(DOWN)"BY ALLAN YATES":PRINTTAB((20
      *LL)-LEN(A$)/2);A$
 1037 PRINT"
 1040 A$=(DOWN)"(C)COPYRIGHT 1983":PRINTTAB(
      (20*LL)-LEN(A$)/2);A$
 1050 PRINT
 1060 A$="RIGHT TO COPY BUT NOT TO SLL":PRINT
      TAB((20*LL)-LEN(A$)/2);A$
 1070 PRINT"
 1080 RETURN
 2000 REM(2 DOWN) ***** SET HEX VALUES *****

 2010 REM
 2020 DIM  D$(15),H$(255),IP(17)
 2040 FOR J=0TO15:READ D$(J):NEXT
 2070 FOR J=0TO15:FOR D=0TO15:H$(J*16+D)=D$(
      J)+D$(D):NEXT D,J
 2080 B=256:RETURN
 8000 DATA  0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
10000 REM *****  GET BYTE  *****
10010 REM
10020 PRINT#15,"M-R";CHR$(AD-INT(AD/256)*256
      )CHR$(AD/256)
10030 GET#15,D$:D=ASC(D$+CHR$(0)):RETURN
```

CROSS-REF

```
100 DIM A$(15),B$(3),X$(1500),C(255)
110 PRINT"(CLEAR)CROSS-REF     JIM BUTTERFI
    ELD"
120 Q$=CHR$(34):S$="       ":B$(1)=Q$:B$(3)=
    CHR$(58)
130 INPUT "VARIABLES OR LINES";Z$:C2=5:IFASC
    (Z$)=76THENC2=6
140 FOR J=1TO255:C(J)=4:NEXT J:FOR J=48TO5
    7:C(J)=6:NEXT J
150 IFC2=5THENFOR J=65TO90:C(J)=5:NEXT J:FOR
    J=36TO38:C(J)=7NEXT J:C(40)=8
160 C(34)=1:C(143)=2:C(131)=3
170 INPUT "PROGRAM NAME";P$:OPEN1,8,3,"0:"
    +P$+",P,R"
180 GET#1,A$,B$:IFASC(B$)<>4THENCLOSE1:STOP

190 IFB=0THEN240
200 PRINTL$;:K=X:FOR J=BTO1STEP-1:PRINT" "
    ;A$(J);:X$=A$(J)
205 IFC2=6ANDLEN(X$)<5THENX$=" "+X$:GOTO 2
    05
206 X$=X$+L$
210 IFX$(K)>=X$THENX$(K+J)=X$(K):K=K-1:GOTO
    210
220 X$(K+J)=X$:NEXT J:X=X+B:PRINT:B=0
230 REM GET NEXT LINE, TEST END
240 GET#1,A$,B$:IFLEN(A$)+LEN(B$)=0THEN530

250 REM GET LINE NUMBER
260 GET#1,A$:L=LEN(A$):IFL=1THENL=ASC(A$)
270 GET#1,A$:A=LEN(A$):IFA=1THENA=ASC(A$)
280 C=C2:C1=-1:L=A*256+L:L$=STR$(L):IFLEN(
    L$)<6THENL$=LEFT$(S$,6-LEN(L$))+L$
290 REM GET BASIC STUFF
300 GET#1,A$:A=LEN($:IFA=1THENA=ASC(A$)
310 C9=C(A):IFC9>C1THEN380
320 K=0:IFB=0THEN360
```

```
330 FOR J=1TOB:IFA$(J)=M$THEN370
340 IFA$(J)<M$THENNEXT J:K=B:GOTO 360
350 FOR K=BTOJSTEP-1:A$(K+1)=A$(K):NEXT K
360 B=B+1:A$(K+1)=M$
370 C=C2:C1=-1:M$=""
380 IFC2=5THEN420
390 IFA=137ORA=138ORA=141ORA=167THENC=6:GOTO
    470
400 IFA=44ORA=32THEN470
410 IFC9<>C=9:GOTO 470
420 IFC9=CTHENC=-1:C1=4
430 IFC>6THEN470
440 IFC<0ANDC9>C1ANDC9>6THENC1=C9:GOTO 460

450 IFC2=5THENIFLEN(M$)>20RC>0THEN470
460 M$=M$+A$
470 ONC9+1GOTO 190,480,480,480:GOTO 300
480 B$=B$(C9):C$=""
490 GET#1,A$:IFA$=""THEN190
500 IFA$=B$THEN300
510 IFA$<>Q$THEN490
520 A$=B$:B$=C$:C$=A$:GOTO 490
530 CLOSE1:INPUT PINTER";Z$
540 C=3:Z=6:IFASC(Z$)=89THENC=4:Z=12
550 OPEN2,C:PRINT#2:PRINT#2,"CROSS REFEREN
    CE - PROGRAM ";P$
560 X$="":FOR J=1TOX:A$=X$(J)
565 IFC2=6THENK=6:GOTO 580
570 FOR K=1TOLEN(A$):IFMID$(A$,K,1)<>" "THEN
    NEXT K:STOP
580 B$=LEFT$(A$,K-1):C$=MID$(A$,K+1):IFX$=
    B$THEN600
590 PRINT#2:Y=0:X$=B$:PRINT#2,X$;LEFT$(S$,
    5-LEN(X$));
600 Y=Y+1:IFY<ZTHEN620
610 Y=1:PRINT#2:PRINT#2,S$;
620 PRINT#2,LEFT$(S$,6-LEN(C$));C$;
630 NEXT J:PRINT#2:CLOSE2
```

DATA BASE

```
   10 OPEN15,8,15,"I0"
  100 DIM A$(300)
  110 REM*** DATA BASE 29/03/83 ***
  120 POKE36879,42:PRINT"■";
  130 PRINT"(CLEAR)(3 DOWN)"
  140 PRINTTAB(11)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":PRINTTAB(
      11)"▒╲──────────────╱▒"
  150 PRINTTAB(11)"▒ |              | ▒"
  160 PRINTTAB(11)"▒ | DATA BASE | ▒":PRINTTAB(
      11)"▒ |              | ▒"
  170 PRINTTAB(11)"▒╱──────────────╲▒"
PRINTTAB(11)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
  190 PRINTTAB(17)"(2 DOWN)BY":PRINTTAB(11)"
      (DOWN)E.V.M. COMPUTERS":PRINTTAB(14)"(DOWN)(
      C) 1983"
  200 FOR T=1TO3000:NEXT
  210 POKE650,128:POKE36879,110:PRINT"■";
  220 GOTO 530
  230 REM *** LETTURA FILE ***
  240 PRINT"(CLEAR)          (RVS)      LETTUR
      A FILE      (RVSOFF)"
  250 PRINT"(4 DOWN)              D - DA DISCO
      "
  260 PRINT"(2 DOWN)              N - DA NASTR
      O"
"(2 DOWN)              T - RITORNA
      MENU"
  280 GETA$:IFA$=""THEN280
  290 IFA$="D"THEN2520
  300 IFA$="N"THEN330
  310 IFA$="T"THEN530
  320 GOTO 280
  330 PRINT"(CLEAR)             (RVS)■  RICERC
      A   FILE  (RVSOFF)"
  340 INPUT "(2 DOWN)(RVS)FILE:(RVSOFF)";F$
  350 OPEN1,1,0,F$:PRINT"(DOWN)FILE: (RVS)"F
      $"(RVSOFF)"
```

```
360 INPUT# 1,CA:FOR M=1TOCA:INPUT# 1,CA$(M
    ):INPUT# 1,DD(M):NEXT M
370 INPUT# 1,Y:FOR I=1TOYINPUT# 1,A$(I):NEXT
    I:CLOSE1:GOTO 530
380 REM *** REGISTRA FILE ***
390 PRINT"(CLEAR)          (RVS)    REGISTR
    A  FILE    (RVSOFF)"
400 PRINT"(4 DOWN)             D - SU DISCO
    "
410 PRINT"(2 DOWN)             N - SU NASTR
    O"
420 PRINT"(2 DOWN)             T - TORNA ME
    NU"
430 GETA$:IFA$=""THEN430
440 IFA$="D"THEN2650
450 IFA$="N"THEN480
460 IFA$="T"THEN530
470 GOTO 430
480 PRINT"(CLEAR)            (RVS)  REGIST
    RA  FILE  (RVSOFF)"
490 PRINT"(2 DOWN)SCRIVI NOME   FILE:":INPUT
    "(DOWN)(RVS)FILE:(RVSOFF)";F$
500 OPEN1,1,1,F$:PRINT"(DOWN)FILE: (RVS)"F
    $"(RVSOFF)"
510 PRINT#1,CA:FOR M=1TOCA:PRINT#1,CA$(M):
    PRINT#1,DD(M):NEXT M
520 PRINT#1,Y:FOR I=1TOY:PRINT#1,A$(I):NEXT
    I:CLOSE1:GOTO 530
530 PRINT"(CLEAR)       (RVS)   DATA -
    BASE  (RVSOFF)"
540 PRINT"(DOWN)             1- LEGGO FILE "

550 PRINT"(DOWN)             2- REGISTRO FIL
    E"
560 PRINT"(DOWN)             3- CREO ARCHIVI
    O"
570 PRINT"(DOWN)             4- CONTINUO ARC
    HIVIO"
```

```
580 PRINT"(DOWN)                5- ELENCO RECOR
    D"
590 PRINT"(DOWN)                6- SORT RECORD"

600 PRINT"(DOWN)                7- RICERCAECO
    RD"
610 PRINT"(DOWN)                S- STAMPA RECOR
    D"
620 PRINT"(DOWN)                M- MEMORIA FREE
     "
622 PRINT"(DOWN)                E- END "
630 PRINT"(DOWN)              (RVS)CAMPI=(RVSOFF)"
    CA:PRINTTAB(20)"(UP)(RVS)RECORD=(RVSOFF)"
    Y
640 GETE$:IFE$=""THEN640
650 IFE$="1"THEN230
660 IFE$="2"THEN380
670 IFE$="3"THEN750
680 IFE$="4"THEN970
690 IFE$="5"THEN8000
700 IFE$=""THEN2280
710 IFE$="7"THEN1690
720 IFE$="M"THEN1330
730 IFE$="S"THEN1450
732 IFE$="E"THENPRINT"(CLEAR)(10 DOWN)
            (RVS)FINE  PROGRAMMA.(RVSOFF)"
    :END
740 GOTO 640
750 REM *** CREO ARCHIVIO ***
760 PRINT"(CLEAR)(DOWN)              ▒▒▒(RVS)
    NOTA BENE (RVSOFF)▒▒▒▒"
770 PRINT"(2 DOWN)DEFINISCI IL NUMERO ED I
     NOMI DEL CAMP,(DOWN)(MASSIMO (RVS)9(RVSOFF)
    CAMPI DA (RVS)10(RVSOFF) ";
780 PRINT"CARATTERI).":PRINT"(DOWN)POI DEF
    INISCI LA LUNGHEZZA DEL CAMPO, ";
781 PRINT"AL(DOWN)MASSIMO DI (RVS)25(RVSOFF)
    CARATTERI OGNUNO."
```

```
790 PRINT"(DOWN)LA LUNGHEZZA DI UN RECORD
    PUO' ESSERE DI(DOWN)(RVS)88(RVSOFF)  C
    ARATTERI IN TOTALE.";
792 PRINT" (ESCLUSI I NOMI(DOWN)DEI CAMPI)
    ."
800 PRINT(2 DOWN)              (RVS)PREMI
    UN TASTO(RVSOFF)"
810 GETAU$:IFAU$=""THEN810
820 PRINT"(CLEAR)          (RVS)  CREO  A
    RCHIVIO  (RVSOFF)":PRINT"(2 DOWN)QU
    ANTI CAMPI ? (RVS)";
830 GETR$:IFR$=""THEN830
840 CA=VAL(R$)
850 PRINTCA:PRINT"(DOWN)  (RVS) NOME DEL C
    AMPO (RVSOFF)     (RVS) LUNGHEZZA CAMP
    O(RVSOFF)":PRINT
860 AH=0:FOR K=1TOCA
870 PRINTK;:INPUT " ";CA$(K):IFLEN(CA$(K))
    >10THENPRINT"(2 UP)":GOTO 870
880 PRINT"(UP)(25 RIGHT)";:INPUT DD(K):IFD
    D(K)>25THEN880
890 AH=AH+(DD(K)):IFAH>88THEN880
900 NEXT
910 BH=88-AH:PRINT"(HOME)(18 DOWN)
     "BH"(RVS)CARATTERI LIBERI(RVSOFF)"
920 PRINT"            (DOWN)VUOI CORREGGERE
    ?"
930 GETAP$:IFAP$=""THEN930
940 IFAP$="S"THEN961
950 IFAP$=N"THEN970
960 GOTO 930
961 INPUT "(DOWN)N. DEL CAMPO:";NC:IFNC>CA
    +1THEN961
962 INPUT "(DOWN)(RVS)NOME CAMPO(RVSOFF)";
    CA$(NC):INPUT "(DOWN)(RVS)LUNGHEZZA(RVSOFF)"
    ;DD(NC)
963 PRINT"(CLEAR)(DOWN)  (RVS) NOME DEL CA
    MPO (RVSOFF)";:PRINT"     (RVS) LUNGHE
    ZZA (RVSOFF)":PRINT:IFNC>CATHENCA=NC
```

```
 964 AH=0:FOR K=1TOCA:PRINT"       ";CA$(K)
 965 PRINT"(UP)(25 RIGHT)"DD(K):AH=AH+DD(K)
     :NEXT
 966 BH=88-AH:PRINT"(HOME)(18 DOWN)
      "BH"(RVS)CARATTERI LIBERI(RVSOFF)"
 968 PRINT"            (DOWN)VUOI CORREGGERE
     ?"
 969 GOTO 930
 970 IFCA(1THEN640
 980 PRINT"(CLEAR)             (RVS)  CREO  A
     RCHIVIO  (RVSOFF)  "
 990 Y=Y+1:I=Y
1000 PRINT"(2 DOWN)  PER TORNARE BATTI (RVS)
     * (RVSOFF)"
1010 PRINT"(DOWN)NUMERO "I:PRINT"(UP)(12 RIGHT)
     (RVS)123456789012345678901234 5(RVSOFF)"
     :PRINT
1020 A$(I)="":FOR K=1TOCA
1030 PRINT"(RVS)"CA$(K)"(RVSOFF)":PRINT"(UP)
     (12 RIGHT)";:FOR E=1TODD(K):PRINT"-";:
     NEXT E
1040 FOR U=1TODD(K)+2:PRINT"(LEFT)";:NEXT U

1050 INPUT AB$(K)
1060 IFLEN(AB$(K))>DD(K)THEN1030
1070 A$(I)=A$(I)+AB$(K)
1080 IFLEFT$(AB$(K),1)="*"THENY=Y-1:GOTO 53
     0
1090 NEXT K:IFCC=99THEN2500
1100 GOTO 980

1120 PRINT"(DOWN)(RVS)I=INSERT  D=DEL. C=CO
     RR.  M=MENU  V=VEDI(RVSOFF)"
1130 GETA$:IFA$=""THEN1130
1140 IFA$="I"THEN2460
1150 IFA$="C"THEN1390
1152 IFA$="D"THEN3000
```

```
 1160 IFA$="M"THEN530
 1170 IFA$="V"THEN1190
 1180 GOTO 1130
 1190 G=0:PRINT"(CLEAR)":GOTO 1260
 1200 REM
 1202 PRINT"(CLEAR)          (RVS)        ELEN
      CO          (RVSOFF)"
"(DOWN)QUANTI DATI VIDEO  ?"
 1212 GETGG$:IFGG$=""THEN1212
 1214 GG=VAL(GG$)
 1220 G=0:FOR I=1TOY:S=1:PRINT"(UP)
         RECORD N.(RVS)"I"(RVSOFF)":PRINT
 1230 FOR M=1TOCA:AB$(M)=MID$(A$(I),S,DD(M))
      :S=S+DD(M)
 1240 PRINT"(RVS)"CA$(M)" (RVSOFF)":PRINT"(UP)
      (10 RIGHT)"AB$(M)
 1250 NEXT M:G=G+1:IFG=GGTHEN1120
 1260 PRINT"(DOWN)":NEXT I
 1270 PRINT"(DOWN)(RVS)I=INSERT  =CRREGGI
        D=DELETE   M=MENU(RVSOFF)"
 1280 GETA$:IFA$=""THEN1280
 1290 IFA$="I"THEN2460
 1300 IFA$="C"THEN1390
 1310 IFA$="M"THEN530
 1312 IFA$="D"THEN3000
 1320 GOTO 1280
 1330 REM** MEMORIA **
 1340 PRINT"(CLEAR)":FR=FRE(0)
 1350 PRINT"(5 DOWN)             (RVS)MEMORIA
      (RVSOFF) ="FR
 1360 PRINT"(2 DOWN)            (RVS) PREMI U
      N TASTO (RVSOFF)"
 1370 GETA$:IFA$="THEN1370
 1380 GOTO 530
 1390 I=0:INPUT "(DOWN)NUM. DA CORREGGERE";I
      :IFI>YTHEN1390
 1400 PRINT"(CLEAR)(2 DOWN)"I:A$(I)="":FOR M
      =1TOCA
```

```
1410 PRINT"(DOWN)(RVS)"CA$(M)"(RVSOFF)";:PRINT
     "    ";:FOR E=1TODD(M):PRINT"-";:NEXT E

1420 FOR U=1TODD(M)+2:PRINT"(LEFT)";:NEXT U
     :INPUT AB$(M)
1430 A$(I)=A$(I)+AB$(M)
1440 NEXT :GOTO 1380
1450 REM*** STAMPA DATI ***
1460 PRINT"(CLEAR)(4 DOWN)   LA STAMPATE E
     '  COLLEGATA ?"
1470 GETAL$:IFAL$=""THEN1470
1480 IFAL$="N"THEN530
1490 IFAL$="S"THEN1510
1500 GOTO 1470
1510 OPEN4,4:GOSUB2130
1520 FOR I=1TOY:S=1
1530 GOSUB1540:NEXT I:CLOSE4:GOTO 530
1540 PRINT#4,I;:A$(I)=A$(I)
1550 FOR M=1TOCA:AB$(M)=MID$(A$(I),S,DD(M))
     :S=S+DD(M)
1560 PRINT#4,LEFT$(DC$,DD);
1570 PRINT#4,AB$(M);
1580 PRINT#4,LEFT$(DC$,DD);
1590 NEXT M:PRINT#4"":RETURN
1600 OPEN4,4
1610 PRINT#4,CHR$(14)"              ELENCO  D
     ATI"
1620 PRINT#4,CHR$(15)
1630 FOR I=1TOY:S=1
1640 PRINT#4,"   DATI N."I
1650 FOR M=1TOCA:AB$(M)=MID$(A$(I),S,DD(M))
     :S=S+DD(M)
1660 PRINT#4,CA$(M);
1670 PRINT#4,CHR$(16)"20: "AB$(M)
1680 NEXT :PRINT#4,"":NEXT I:CLOSE4:GOTO 53
     0
1690 REM*** RICERCA DATI ***
```

```
1700 PRINT"(CLEAR)                    (RVS)NOTA
     BENE(RVSOFF)"
1710 PRINT"(DOWN) LA RICERCA DEI DATI  PUO'
      ESSERE FATTA (DOWN)SU  OGNI CAMPO."
1720 PRINT"(DOWN)BASTA UN CARATTERE PER ELE
     NCARE TUTTI I (DOWN)DATI CHE INIZIANO
     CON QUEL ";
1730 PRINT"CARATTERE.":PRINT"(DOWN)PIU' CAR
     ATTERI RENDONO LA RICERCA PIU'  (DOWN)S
     ELETTIVA."
1740 PRINT(4 DOWN)                 (RVS)PREMI
     UN TASTO(RVSOFF)"
1750 GETA$:IFA$=""THEN1750
1760 PRINT"(CLEAR)          (RVS)     RICERCA
        DATI     (RVSOFF)"
1770 JJ=0:PRINT"(DOWN)PREMI (RVS)S(RVSOFF)
     PER STAMPARE, O UN TASTO PER     (DOWN)P
     ROSEGUIRE."
1780 GETA$:IFA$=""THEN1780
1790 IFA$="S"THEN2070
1800 INPUT "(RVS)(DOWN)DATI:(RVSOFF)";AW$:A
     W=LENK AW$)
1810 SR=0:FOR I=1TOY:S=1:FOR M=1TOCA
1820 IFA$=MID$(A$(I),S,DD(M))THEN1970
1840 IFAW$=LEFT$(MID$(A$(I),S,DD(M)),AW)THEN
     1970
1850 S=S+DD(M)
1920 NEXT M
1930 NEXT I
1940 PRINT"(DOWN)                  (RVS) FINE D
     ATI (RVSOFF)":PRINT"              (RVS)PR
     EMI PER TORNARE(RVSOFF)    "
1941 CLOSE4
1950 GETA$:IFA$=""THEN1950
1960 GOTO 530
1970 S=1:A$(I)=A$(I):PRINT"(DOWN) NUM."I
1980 FOR N=1TOCA:AB$MID$(A$(I),S,DD(N)):S=
     S+DD(N)
```

```
1990 PRINT"(RVS)"CA$(N)" (RVSOFF)":PRINT"(UP)
     (11 RIGHT)"AB$:NEXT N:IFJJ=999THENI=I:
     GOTO 2090
2000 IFJJ=999THEN1930
2010 I=I:SR=SR+1:IFSR=2THEN2030
2020 GOTO 1930
2030 PRINT"(DOWN)              (RVS) PREMI PER
      VEDERE (RVSOFF)"
2040 GETA$:IFA$=""THEN2040
2050 PRINT"(CLEAR)           (RVS)     RICERCA
        DATI     (RVSOFF) "
2060 SR=0GOTO 2020
2070 OPEN4,4:JJ=999:GOSUB2130
2080 GOTO 1800
2090 PRINT#4,I;
2100 FOR M=1TOCA:PRINT#4,LEFT$(DC$,DD);:PRINT#
     4,AB$(M);:PRINT#4,LEFT$(DC$,DD);
2110 NEXT :PRINT#4,""
2120 GOTO 2000
2130 DA=0:DB=0:FOR M=1TOCA:CA$(M)=CA$(M):DA
     =DA+LEN(CA$(M)):DB=DB+DD(M)
2140 NEXT
2150 IFDA>75THEN1600
2160 IFDD>75THEN1600
2170 DC=75-DA:DC=DC/CA/2:DC=INT(DC)
2180 D=75-DB:DD=DD/CA/2:DD=INT(DD):DC$="
                                    "
2190 PRINT#4,CHR$(14)"              ELENCO  D
     ATI"
2200 PRINT#4,CHR$(15)
2210 FOR M=1TOCA
2220 PRINT#4,LEFT$(DC$,DC);
2230 PRINT#4,CA$(M);
2240 PRINT#4,LEFT$(DC$,DC);
2250 NEXT :PRINT#4,""
2260 FOR F=1TO79:PRINT#4,"─";:NEXT F:PRINT#
     4,""
```

```
2270 RETURN
2280 PRINT"(CLEAR)(DOWN)  (RVS) NOME DEL CA
     MPO (RVSOFF)";:PRINT"      (RVS) LUNGHE
     ZZA (RVSOFF)":PRINT:IFNC>CATHENCA=NC
2282 FOR K=1TOCA:PRINTK;"     ";CA$(K)
2284 PRINT"(UP)(25 RIGHT)"DD(K):NEXT
2290 PRINT"(DOWN)   (RVS)SU QUALE CAMPO VUO
     I IL SORT ?(RVSOFF)"
2292 GETR$:IFR$=""THEN2292
2294 R=VAL(R$):S=1:FOR M=1TOCA:S=S+DD(M):A(
     M)=S-DD(M):NEXT :A(1)=1:A1=A(R)
2296 INPUT "(2 DOWN)   (RVS)QUNTE LETTERE
     DI PRECISIONE(RVSOFF)";B1
2310 PRINT"(CLEAR)(6 DOWN)"TAB(14)"(RVS)SHE
     LL  SORT(RVSOFF)":PRINT"(4 DOWN)"TAB(1
     5)"(RVS)ATTENDERE(RVSOFF)"
2312 TI$="000000":FOR J=1TOY :M=1
2313 M=2*M:IFM<=YTHEN2313
2314 M=INT(M/2):IFM=0THEN2430
2320 FOR J=1TOY-M:C=J
2330 D=C+M:IFMID$(A$(C),A1,B1)<=MID$(A$(D),
     A1,B1)THEN2350
2340 AA$=A$(C):A$(C)=A$(D):A$(D=AA$:C=C-M:
     IFC>0THEN2330
2350 NEXT :GOTO 2314
2430 PRINTTAB(6)"(4 DOWN)"RIGHT$(TI$,4)"  S
     ECONDI PER IL SORT!":FOR T=1TO2000:NEXT

2450 GOTO 530
2460 CC=0:C=0:INPUT "N.DA INSERIRE";C
2480 Y=Y+1:FOR I=YTOCSTEP-1:A$(I)=A$(I-1):NEXT

2490 I=C:CC=99:PRINT"(CLEAR)":GOTO 1010
2500 CC=0:GOTO 1270
2520 PRINT"(CLEAR)          (RVS)  LETTURA F
     ILE DISCO  (RVSOFF)"
2530 PRINT"(2 DOWN)SCRIVI NOME FILE:":INPUT
     "(DOWN)(RVS)■FILE(RVSOFF)■";F$
```

```
2540 OPEN2,8,2,F$+",S,R":GOSUB2740
2550 INPUT# 2,CA:RS=ST:GOSUB2740:FOR M=1TOC
     A
2560 INPUT# 2,CA$(M):RS=ST:GOSUB2740
2570 INPUT# 2,DD(M):RS=ST:GOSUB2740:NEXT M
2580 INPUT# 2,Y:RS=ST:GOSUB2740:FOR I=1TOY
2590 INPUT# 2,A$(I)
2600 IFRS=64THEN2630
2610 IFRS<>0THEN2640
2620 NEXT :CLOSE2:GOTO 530
2630 CLOSE2:GOTO 530
2640 PRINT"ERRORE DI STATO":CLOSE2:GOTO 550

2650 PRINT"(CLEAR)          (RVS)  REGISTRA
     FILE DISCO (RVSOFF)":CR$=CHR$(13)
2660 PRINT"(2 DOWN)SCRIVI NOME FILE:":INPUT
     "(DOWN)(RVS)FILE(RVSOFF) ";F$
2670 OPEN2,8,2,"@:"+F$+",S,W":GOSUB2740
2680 PRINT#2,STR$(CA)CR$;:GOSUB2740:FOR M=1
     TOCA
2690 PRINT#2,CA$(M)CR$;
2700 PRINT#2,STR$(DD(M))CR$;:NEXT M:GOSUB27
     40
2710 PRINT#2,STR$(Y)CR$;:GOSUB2740:FOR I=1TO
     Y
2720 PRINT#2,A$(I)CR$;:GOSUB2740
2730 NEXT I:CLOSE2:GOTO 530
2740 INPUT# 15,EN,EM$,ET,ES
2750 IFEN=0THENRETURN
2760 PRINT"OPERAZIONE ERRATA":CLOSE2:FOR T=
     1TO1000:NEXT :RETURN
3000 INPUT "(DOWN)NUMERO DA CANCELLARE";C:Y
     =Y-1
3010 FOR I=CTOY:A$(I)=A$(I+1):NEXT :GOTO 53
     0
8000 PRINT"(CLEAR)              (RVS)   ELEN
     CO   (RVSOFF)"
```

DATA BASE

```
8100 PRINT"(DOWN)    ";:FOR =1TOCA:PRINT"(RVS)"
     CA$(M)"     ";:NEXT :PRINT
8220 G=0:FOR I=1TOY:S=1:PRINT"(RVS)"I"(RVSOFF)"
     ;
8230 FOR M=1TOCA
8240 PRINTMID$(A$(I),S,DD(M))"    ";:S=S+DD(
     M)
8250 NEXT M:G=G+1:IFG=20THEN8300
8260 PRINT"(DOWN)":NEXT I
8270 GOTO 1270
8300 PRINT"(DOWN)(RVS)I=INSERT  D=DEL. C=CO
     RR.  M=MENU  V=VEDI(RVSOFF)"
8310 GETA$:IFA$=""THEN8310
8340 IFA$="I"THEN2460
8350 IFA$="C"THEN1390
8360 IFA$="M"THEN530
8362 IFA$="D"THEN3000
8370 IFA$="V"THEN8400
8380 GOTO 1130
8400 PRINT"(DOWN)":NEXT I
8410 GOTO 1270
```

I N D I C E

ELENCO PROGRAMMI FORNIBILI SU DISCO O NASTRO

Come avevamo annunciato nella parte introduttiva mettiamo a disposizione del lettore di questo manuale un servizio che riteniamo utilissimo e per di piu' a prezzi particolari.
Utilizzando l' apposita cartolina potrete ordinare uno o piu' dei seguenti prodotti.

!!!!!ATTENZIONE!!!!

Ricordiamo che SOLO i possesori del manuale e quindi con CARTOLINA NUMERATA possono richiedere le confezioni riportate ai prezzi indicati.
Ricordarsi inoltre di specificare il modello di disco.

Diamo l' elenco riepilogativo dei programmi che verranno forniti su disco utilizzando l' apposita cartolina.
Oltre a quelli pubblicati sul presente volume sono comprese altre utility.

CODICE EVM 9901

MENUEV
DONE
COPIA FILE
RECUPERO
DOS HEX DUMPER
CROSS REFERENCE
DATA BASE
DISKMENU2
COPI 8 TO 9
COPY ALL
DIR UPDATE
BAM MAP
DUMP SEQ FILE
RILOCATORE
DISK ID
PERFORMANCE TEST
SEQUENTIAL FILE
RANDOM FILE

Elenco dei programmi forniti su nastro.

CODICE EVM 9902

TAPE LABEL
DATA BASE
TAPE PHONO FILE

CODICE EVM 9903

TURBO TAPE

Questo programma consente di aumentare di ben 7 volte la velocita' di caricamento e di registrazione di programmi su nastro.

CODICE EVM 9904

DISK DOCTOR PLUS

Questo package consente di controllare fisicamente lo stato ed il contenuto dei files su disco.
E' fornito insieme ad altri quattro interessanti utilities, fra le quali addirittura un DISASSEMBLER per il DOS.

DISK HEX DUMPER
*** DOS DISASSEMBLER ***
DISK LOGGER
DOS HEX DUMPER

NOME _____

VIA _____

CAP CITTA' _____

SEGNALAZIONE ERRORI _____

_____

_____

_____

DESIDERO RICEVERE

☐	CATALOGO E.V.M.	OMAGGIO
☐	DISCO COD. E.V.M. 9901	L. 35.000
☐	NASTRO COD. E.V.M. 9902	L. 20.000
☐	NASTRO COD. E.V.M. 9903	L. 20.000
☐	DISCO COD. E.V.M. 9904	L. 30.000

☐ EFFETTUATE LA SPEDIZIONE IN CONTRASSEGNO AGGIUNGENDO L. 4.000 QUALE CONTRIBUTO SPESE POSTALI

☐ ALLEGO ASSEGNO AGGIUNGENDO L. 1.500 AL TOTALE QUALE CONTRIBUTO SPESE POSTALI

N.B.; I PREZZI SI INTENDONO COMPRENSIVI DI IVA

F I R M A

SPEDIRE IN BUSTA A:

E.V.M Computers
Via Marconi 9/a
52025 MONTEVARCHI (AR)

Normalmente i manuali dedicati ai computer trattano le funzioni dell'unità centrale trascurando molto spesso le periferiche.
Con questo testo si ricopre questa lacuna prendendo in esame le più diffuse periferiche delle linea COMMODORE.
Le periferiche trattate in questo volume sono:

CASSETTA **C2N / DATASETTE**
FLOPPY DISK **1540/1541/2031/2040/3040/4040/8050/8250**
LE STAMPANTI **1515/1525/GP100/1526/MPS 801/MPS 802/ 2022/3022/4022/4023/8023/MPP 1361**
BUS DI COLLEGAMENTO **IEEE-488 PARALLELA / IEEE-488 SERIALE / RS-232**

L. 25.000

E.V.M.

LE PERIFERICHE COMMODORE