COMMODORE 64

I segreti del disco

Trattamento file col driver floppy disk 1541

di Andrea de Prisco

In tutto l'armamentario Commodore (sezione Home Computer) se c'è un elemento che davvero brilla per le sue alte doti di scarsezza qualitativa, questo è il manuale di istruzioni dell'unità a floppy disk 1541. Le inesattezze si contano a decine, e non solo a carattere editoriale. Zeri scambiati per "o", uno per "i", virgole in luogo di punto e virgola. Un vero pasticcio. Quanti lettori sono in grado di usare i file relativi? Cercheremo di mettere un po' d'ordine nella faccenda, svelandovi anche qualcosa in più.

SAVE e LOAD

Prima di parlare di unità a dischi, diamo uno sguardo a come il Commodore 64 salva e recupera programmi da memoria di massa (disco o nastro che sia). Come tutti ben sappiamo, l'istruzione che ci consente di salvare un programma è:

```
SAVE "Nome Programma",Periferica
```

Periferica è il numero device: 1 per il nastro, 8 o un numero maggiore per l'unità a dischi. Fin qui tutto normale. Più interessante è conoscere cosa il computer fa quando gli si dice "SAVE !".

Le prime 256 locazioni della memoria del 64 (per intenderci: da PEEK(Ø) a PEEK(255)), costituiscono la cosiddetta Pagina Zero. È usata dal microprocessore per conservare le variabili di sistema: un insieme di informazioni necessarie al buon funzionamento di tutta la macchina.

Fra le tante cose che il processore conserva in pagina zero, alcune indicano l'inizio della memoria Basic, la fine, dove sono stivate le variabili, gli array, le stringhe, dove termina il programma Basic in memoria (qual è l'ultimo byte occupato). Queste ed altre, permettono al microprocessore di non fare confusione durante l'esecuzione di un programma, stivando sempre in locazioni libere le nuove variabili Basic usate, senza perderne nessuna (sarebbe il colmo ...). In figura 1 è mostrata la tabella delle variabili di sistema più utili.

Ritornando alla nostra SAVE, quando da Basic è impartito questo comando il processore non fa altro che trasferire sulla periferica (nastro o disco) tutto quello che è compreso tra l'inizio programma Basic (questa informazione è mantenuta nelle locazioni 43 e 44) e la fine programma Basic (locazioni 45 e 46).

All'accensione della macchina, i due puntatori sono (per così dire) azzerati: puntano entrambi alla prima locazione Ram disponibile, la 2049 (la 2048 deve sempre contenere 0). Provando a inserire una qualsiasi linea Basic, possiamo controllare che il puntatore fine programma "avanza". Se digitiamo, dopo aver inserito una linea:

```
PRINT PEEK(46)•256+PEEK(45)
```

Troveremo un numero maggiore di 2049. Man mano che inseriremo nuove linee, il nostro puntatore di fine continuerà ad avanzare, mantenendo sempre l'informazione della locazione del primo byte libero a ridosso del nostro programma Basic. Variando di mano nostra i due puntatori di inizio e fine, è possibile salvare anche qualcosa di diverso dal programma Basic, o un programma Basic che (sempre per nostra scelta) non inizia a 2049.

Volendo ad esempio salvare su disco il contenuto della Rom dell'interprete Basic, locata tra 40960 e 49151, non dovremo far altro che posizionare il puntatore di inizio a 40960, quello di fine a 49151 e digitare SAVE "ROM BASIC",8. Per posizionare i due puntatori, poniamo:

```
POKE 44,160:POKE 43,0:POKE 46,192:POKE 45,0
```

Assieme al programma vero e proprio (anzi, per l'esattezza: a tutto quanto compreso tra i due puntatori di cui sopra) il 64 provvede a memorizzare sulla periferica anche il puntatore di inizio. Se per rileggere il programma useremo un normalissimo LOAD, il programma sarà caricato a partire dalla corrente locazione puntata da 44 e 43 (è ignorata la locazione di inizio salvata insieme al "malloppo"). Se si usa l'istruzione LOAD nel formato:

```
LOAD "Nome programma",Periferica,1
```

il programma verrà posizionato nella stessa zona di memoria dalla quale era stato salvato. Resta inteso che, avendo salvato un normalissimo programma Basic, usare il LOAD in uno dei formati non cambia nulla, sempreché non sia stato spostato volutamente il puntatore di inizio.

Conoscendo come avviene il trasferimento di programmi su periferica, possiamo già cominciare a fare qualcosa di più contorto: salvare in un solo colpo un programma Basic e uno o più programmi in linguaggio macchina adoperati dal primo con il comando SYS. Non dovrebbe essere molto difficile intuire il trucchetto: una volta digitato il programma Basic, si interroga il puntatore di fine (45 e 46) per conoscere l'ultimo byte occupato dal programma. Di seguito a questo, posizioneremo le nostre routine in linguaggio macchina avendo l'accortezza, terminata questa operazione, di spostare il puntatore di fine oltre lo spazio occupato dal l.m.. Salvando, salveremo tutto, né ci saranno problemi con l'uso di variabili: in ogni caso Basic e l.m. convivranno amichevolmente insieme. Unica limitazione è data dal fatto che non è possibile inserire nuove linee Basic senza "pestare" le routine. Quindi inserire il l.m. a pié del Basic solo quando si è certi che non si dovranno apportare modifiche al programma originale.

I comandi del disco

Più che una semplice periferica, il 1541 è un vero e proprio computer (con tanto di possibilità di programmarlo). La sua stessa struttura lo grida a gran voce: Microprocessore 6502, due VIA (processori di I/O) 6522, 16K Rom contenenti un proprio sistema operativo, 2 K di Ram per

inserire mini routine o da usare come buffer. Accoppiato al 64 forma di fatto un sistema multi-processo molto flessibile. Ad esempio, è possibile mandare in esecuzione un programma, mentre il disco formatta un minifloppy; cosa che non accade per quei computer (leggi APPLE e molti dei più grandi) che caricano da disco il DOS nella memoria utente, gravando il microprocessore della macchina di gestire anche l'unità a dischi.

Fra 64 e disco, si stabiliscono veri e propri contatti verbali. Quando è richiesto un programma, il 64 chiede al disco di passarglielo. Quest'ultimo controlla se è presente (in caso contrario comunica: "capo ..., non c'è"), provvede a cercarlo tra le tracce e quando l'ha trovato inizia a trasferirlo byte dopo byte. Al termine di questa operazione, l'unità centrale ringrazia e dice: "a risentirci!". Da questo momento il disco resta in religioso silenzio ad aspettare un nuovo ordine di "sua eccellenza".

Per dialogare con il disco, così come per la stampante e ogni altra periferica intelligente (l'unità a nastri è scema!), è necessario aprire un canale di comunicazione. Per il disco, il canale da aprire è il n. 15 e l'apertura avviene con il comando OPEN, specificando un parametro di riferimento da usare per le successive operazioni di PRINT#, il numero della periferica (in genere 8) e 15 che comunica al disco che vogliamo dialogare. Quindi:

```
OPEN 1,8,15
```

è la prima operazione da compiere.

A questo punto, col comando PRINT#1 possiamo spedire al driver i nostri messaggi. Per accogliere eventuali messaggi del disco, sempre dopo aver stabilito la comunicazione, basterà eseguire da Basic un INPUT #1, A, B$, C, D. "A" conterrà il numero del messaggio; "B$" il messaggio vero e proprio; "C" e "D" la traccia e il settore (quando serve) dove si è verificato l'inconveniente di "B$". Ciò proprio perché la maggior parte dei possibili messaggi (56 in tutto) riguarda errori. Se nel corso di una operazione la spia rossa inizia a lampeggiare, si è verificato un errore. Conviene "sentire" il driver per saperne di più. I comandi accettati dal disco sono in tutto 6 e si spediscono al driver tramite l'istruzione PRINT #, specificando tra apici il comando e i parametri dello stesso. Senza entrare nei dettagli di tutti, diamo la descrizione di quelli meno digeribili dall'utente col solo ausilio del tremendo manuale di istruzioni.

Il comando VALIDATE, serve per fare pulizia sul disco di tutte le cose inutili. Per esempio, file aperti e non richiusi, programmi mal salvati (apertura dello sportellino prima del termine dell'operazione) e altro. Il comando NEW, da non confondere col corrispondente gemello Basic, serve per formattare o riformattare un disco. Se infatti questo è già stato formattato precedentemente, non specificando l'identificazione avremo una formattazione della sola directory, con conseguente risparmio di

```
10 OPEN4,8,4,"SEQUENTIAL,S,W"
20 INPUTA$:IFA$="x"THENCLOSE4:END
30 PRINT#4,A$+CHR$(13);:GOTO20
```

Listato 1

```
10 OPEN4,8,4,"SEQUENTIAL,S,R"
20 INPUT#4,A$:PRINTA$
30 IFST<>64THEN20
40 CLOSE4
```

Listato 2

Creazione e lettura di un file sequenziale.

Listato 3

```
10 DIMA$(255),B$(255),I%(255)
20 PRINT"■■■ MENU■"
30 PRINT" 1: CREA ARCHIVIO"
40 PRINT" 2: LEGGI ARCHIVIO"
50 PRINT" 3: MODIFICA ARCHIVIO"
60 PRINT" 4: EXIT"
150 GETA$:IFA$<"1"ORA$>"4"THEN150
160 A=VAL(A$):ON A GOTO 1000,2000,3000
170 END
200 REM
210 REM ********************************
220 REM *                              *
230 REM *     R  E  L  A  R  C  H      *
240 REM *                              *
250 REM *          (C)  1984           *
260 REM *                              *
270 REM *        ADP  SOFTWARE         *
280 REM *                              *
290 REM ********************************
300 REM
1000 FORI=0TO255:A$(I)="■":I%(I)=-1:NEXT
1010 HH=HH+1
1015 A$="":A=0:PRINTHH:INPUT"CHIAVE ";A$:IFA$=""THEN1110
1020 B$="":INPUT"TESTO  ";B$
1030 A$=LEFT$(A$+"                              ",18)
1040 FORI=1TOLEN(A$):A=A+ASC(MID$(A$,I,1)):NEXT:A=AAND255
1050 AA=-1
1060 IFA$(A)="■"THEN1090
1065 IFA$(A)=A$THENPRINT"CHIAVE ESISTENTE !":GOTO1015
1070 AA=A:IFI%(A)<>-1THENA=I%(A):GOTO1060
1080 A=(A+1)AND255:IFA$(A)<>"■"THEN1080
1090 B$(A)=LEFT$(B$,254):A$(A)=A$:IFAA<>-1THENI%(AA)=A
1100 GOTO1010
1110 OPEN4,8,4,"@:INDICE,S,W"
1120 FORJ=0TO255:PRINT#4,A$(J):NEXT
1130 FORJ=0TO255:PRINT#4,I%(J):NEXT:CLOSE4
1140 OPEN1,8,15,"I"
1150 OPEN4,8,8,"ARCH,L,"+CHR$(80)
1160 INPUT#1,A,A$,C,D
1170 FORI=255TO0STEP-1:IFA$(I)="■"THEN1200
1180 PRINT#1,"P"CHR$(8)CHR$(I)CHR$(0)CHR$(1)
1190 PRINT#4,B$(I)
1200 NEXT:CLOSE4:CLOSE1:GOTO20
2000 GOSUB2080
2010 A$="":A=0:INPUT"CHIAVE ";A$:IFA$=""THENCLOSE4:CLOSE1:GOTO20
2020 A$=LEFT$(A$+"                              ",18)
2030 FORI=1TOLEN(A$):A=A+ASC(MID$(A$,I,1)):NEXT:A=AAND255
2040 IFA$(A)=A$THEN2070
2050 A=I%(A):IFA=-1THENPRINT"NON ESISTE !":GOTO2010
2060 IFA$(A)<>A$THEN2050
2070 GOSUB2140:GOTO2010
2080 OPEN4,8,4,"INDICE,S,R"
2090 FORJ=0TO255:INPUT#4,A$(J):NEXT
2100 FORJ=0TO255:INPUT#4,I%(J):NEXT:CLOSE4
2110 OPEN1,8,15,"I"
2120 OPEN4,8,8,"ARCH"
2130 INPUT#1,B,A$,C,D:RETURN
2140 PRINT#1,"P"CHR$(8)CHR$(A)CHR$(0)CHR$(1)
2150 INPUT#4,A$:PRINTA$
2160 RETURN
2170 CLOSE1:OPEN1,8,15:INPUT#1,A,A$:PRINTA,A$:CLOSE1
3000 GOSUB2080
3010 A$="":A=0:INPUT"CHIAVE ";A$:IFA$=""THENCLOSE4:CLOSE1:GOTO20
3020 A$=LEFT$(A$+"                              ",18)
3030 FORI=1TOLEN(A$):A=A+ASC(MID$(A$,I,1)):NEXT:A=AAND255
3040 IFA$(A)=A$THEN3070
3050 A=I%(A):IFA=-1THENPRINT"NON ESISTE !":CLOSE1:CLOSE4:GOTO20
3060 IFA$(A)<>A$THEN3050
3070 INPUT"TESTO ";A$
3140 PRINT#1,"P"CHR$(8)CHR$(A)CHR$(0)CHR$(1)
3150 PRINT#4,A$
3160 CLOSE1:CLOSE4:GOTO20
```

Programma RELARCH.

POKE	CONTENUTO
43 - 44	Inizio Programmi Basic
45 - 46	Fine Programmi Basic
47 - 48	Fine Variabili
49 - 50	Fine Array
51 - 52	Inizio Spazio Attivo Stringhe
53 - 54	Fine Spazio Attivo Stringhe
55 - 56	Fine Memoria Basic

Figura 1 - Puntatori Basic.

Numero Traccia	Settori	Totale
1 - 17	0 - 20	21
18 - 24	0 - 18	19
25 - 30	0 - 17	18
31 - 35	0 - 16	17
Totale Settori		664

Figura 2 - Distribuzione settori per traccia.

tempo. A proposito di identificatore: non tutti sanno che l'ID è importantissimo e deve categoricamente essere diverso per ogni dischetto formattato; pena innumerevoli problemi, con tanto di perdita di file, usando più dischetti con lo stesso ID, di seguito.

Altro comando interessante e mal spiegato sul manuale, è il comando COPY. Serve per due scopi: fare una copia sullo stesso dischetto di un qualsiasi file (PRG o SEQ) già presente, o unire più file sequenziali in un unico nuovo file, somma dei precedenti. La sintassi è la seguente (nei due casi):

```
PRINT#1,"COPY:NuovoFile=VecchioFile"
PRINT#1,"COPY:NuovoFile=VecchioFile1,
VecchioFile2"
```

sapendo che NuovoFile è il nome che avrà il file copia e che VecchioFile è il nome di un file già presente sul dischetto. Si possono riunire in un unico file fino a 4 file sequenziali già esistenti.

Trattamento dei file

Una delle possibili traduzioni dall'inglese della parola file è cartellina, raccoglitore. Potremmo dire che come termine è abbastanza azzeccato: un file è un insieme di dati, opportunamente raccolti.

Esistono due tipi di file, a struttura sequenziale e ad accesso diretto. Sequenziale vuol dire che possiamo recuperare i dati solo e soltanto nello stesso ordine in cui l'abbiamo inseriti: inoltre lo stesso caricamento dei dati su memoria di massa avviene sequenzialmente, si può inserire la decima registrazione solo dopo aver inserito le prime 9.

Usando un file ad accesso diretto, si può accedere ad una qualsiasi registrazione dell'archivio, per leggerla o modificarla, semplicemente specificando la posizione o una chiave che identifichi univocamente un determinato dato. Esistono poi altre organizzazioni più complesse, quali l'organizzazione ad albero o le cosiddette organizzazioni ISAM (Indexed Sequential Access Method), che riguardano però (in generale) la gestione di grosse quantità di dati simultaneamente.

Il driver 1541 consente, a livello di hardware, la gestione di file sequenziali e relativi. I file relativi sono ad accesso diretto: relativo vuol dire che è sufficiente specificare la posizione relativa (all'interno del file) della registrazione cercata.

Per quanto concerne il trattamento di archivi sequenziali, c'è davvero poco da dire: è una delle poche cose ben spiegate sul manuale. Qualche consiglio, comunque, non guasta mai. Per prima cosa, è bene parlare col disco solo in termini di stringhe e non di interi, reali e stringhe. Prima di spedire un dato in un file, conviene convertirlo in stringa con l'istruzione STR$. Allo stesso modo, in lettura, otterremo stringhe, eventualmente da riconvertire in interi o in reali a seconda del caso.

Come separatore, fra un dato e il successivo è bene usare un CHR$(13), corrispondente a un [RETURN]. Tutte le stringhe potranno essere riversate sul disco, separandole con un CHR$(13), ma concatenandole con il punto e virgola. Per intenderci:

```
PRINT#2,A$+CHR$(13);
```

inserirà la stringa contenuta in A$, e lascerà il file già pronto a ricevere la successiva stringa, con un comando simile. Per ripescare la registrazione, il comando 2, A$ non fallirà un colpo: il [RETURN] come separatore funziona molto bene. Viene preso il dato voluto e il file resta pronto per la successiva richiesta con analogo comando. È possibile inoltre sapere se la registrazione letta è l'ultima dell'archivio, semplicemente interrogando la variabile ST. Questa, come la TI e la TI$, è una variabile di sistema, e quindi riservata per usi specifici. ST conterrà il numero 64 solo quando avremo letto l'ultimo dato di un file. I programmini 1 e 2 mostrano un esempio di creazione di file sequenziale, e di lettura dello stesso.

E veniamo ai nostri tanto amati file relativi. Per adoperarli, le operazioni da compiere sono due: la creazione del file e il caricamento dei dati. Quando si crea un archivio relativo, bisogna specificare al driver un nome e la lunghezza massima di ogni registrazione. Per inserire dati nel file, è sufficiente specificare quale posizione occuperà il dato, nonché il dato stesso.

Supponiamo di dover creare un archivio indirizzi, che per semplicità supporremo formato da tante stringhe contenenti le varie informazioni (nome, cognome, via, città, telefono) codificate alla maniera sequenziale: campi concatenati, separati da

```
CHIAVE ? LUCIA
VIA CRISPI - PISA

CHIAVE ? FABIO
VIA S.MADDALENA - PISA

CHIAVE ? LEO
VIA PANARO - ROMA

CHIAVE ? MANUELA
VIA DEL LANTE - PISA

CHIAVE ? SANDRO
VIA LIDONNICI - CATANZARO

CHIAVE ? ARCIBALDO
NON ESISTE !

CHIAVE ? ■
```

Accesso ad un file relativo.

Listato 4

```
10 INPUT"TRACCIA & SETTORE ";TR,SE
20 OPEN1,8,15
30 OPEN5,8,5,"#"
40 PRINT#1,"U1:";5;0;TR;SE
50 FORI=0TO255:
60 GET#5,A$:IFA$=""THENA$=CHR$(0)
70 A$=STR$(ASC(A$))
80 PRINTRIGHT$("  "+A$,4);:NEXT
90 CLOSE5:CLOSE1
```

Esempio di lettura di un file relativo col programma RELARCH.

CHR$(13). Diciamo che 200 caratteri per ogni dato dovrebbero bastare. La prima cosa da fare è stabilire la comunicazione col disco. A tale scopo, eseguiremo un:

```
OPEN 1,8,15
```

Per creare il file basterà:

```
OPEN 4,8,4,"INDIRIZZI,L,"+CHR$(200)
```

il primo 4 è un parametro di riferimento: serve per riferire a questo file le successive operazioni di PRINT#. L'8 indica l'unità a dischi; il secondo 4 è il canale di servizio che il disco adopererà per ricevere e restituire dati. Fra apici il nome del file e la specifica che si tratta di un archivio relativo ("L"); infine, sotto forma di CHR$, la lunghezza di ogni registrazione. Appena dato il secondo OPEN, per motivi non troppo chiari, il driver segnala un errore, che la Commodore stessa consiglia di ignorare: la spia inizia a lampeggiare e l'unico modo per farla tacere è eseguire un 1, A,B$,C,D per leggere il messaggio del disco. Per inserire una registrazione, bisogna specificare la posizione. Ciò avviene utilizzando il canale di "dialogo" col disco, con il comando PRINT#1. Se ad esempio, vogliamo occupare la quinta posizione, comunicheremo un:

```
PRINT#1,"P"CHR$(4)CHR$(5)CHR$(0)
```

la "P" indica che stiamo dando la posizione; il 4 in forma di CHR$, specifica il canale adoperato dal nostro file (l'abbiamo specificato al momento della creazione); di seguito la posizione, codificata con due byte, essendo possibili più di 256 posizioni diverse. Come per il linguaggio macchina, la posizione effettiva è data moltiplicando per 256 il secondo valore e sommandoci il primo. Nel nostro caso, 0*256+5 è appunto la quinta posizione; volendo occupare la 600-sima avremmo dovuto scrivere CHR$(88)CHR$(2), dato che 2*256+88 = 600. Creato il nostro file, possiamo accedere alle singole registrazioni sia per scriverle che per leggerle. Dopo il posizionamento, eseguendo un PRINT#4,A$ salveremo sul disco il valore di A$; eseguendo INPUT#4,A$ preleveremo da disco l'informazione contenuta nel record (registrazione) sulla quale ci siamo posizionati. Per maggior sicurezza, dato che a volte il 1541 fa un po' di confusione, è bene specificare, prima della lettura, oltre alla posizione relativa al file, anche la posizione relativa al record: quale byte sarà letto per primo (presumibilmente il n. 1). La specifica della posizione avviene così:

```
PRINT#1,"P"CHR$(4)CHR$(5)CHR$(0)CHR$(1)
```

supposto che eravamo interessati alla quinta registrazione. Come esempio di gestione archivio relativo, è stato preparato il programma RELARCH (listato 3). L'accesso avviene specificando una chiave, non la posizione relativa. Per realizzare ciò abbiamo adoperato un file ausiliario di tipo sequenziale denominato INDICE.

Dando RUN al programma, appare il menu; 4 in tutto le opzioni: creazione, lettura, modifica e ... EXIT per uscire. Anche se solo un esempio, questo programma si presta molto bene per parecchi scopi, forse proprio per la sua eccessiva semplicità. Quando si crea un archivio, bisogna, per ogni registrazione, specificare una chiave (diversa per ognuna) e il testo (max. 80 car.) da associare alla chiave. Ad esempio, se vogliamo un archivio indirizzi, potremmo usare come chiave il nome e cognome e come testo tutto il rimanente: via, telefono, città, etc. Per recuperare una registrazione, da menu si va in lettura, e alla richiesta CHIAVE si indica il nominativo voluto. Per tornare al menu basta rispondere [RETURN] a qualsiasi richiesta di chiave. È possibile inserire al più 256 record.

Accesso per chiave primaria

Il programma RELARCH listato a pag. 114, come descritto nell'articolo, permette un'organizzazione con accesso per chiave alla registrazione cercata. Ciò significa che per individuare all'interno dell'archivio relativo un record, è sufficiente specificare una qualsiasi stringa alfabetica, ad esempio il nome per conoscere telefono, via, città. Chiaramente il programma non si diverte a cercare a tentoni la registrazione, ma sfrutta l'informazione della chiave per conoscere la posizione (al momento del caricamento e all'atto della ricerca). Queste organizzazioni sfruttano le cosiddette trasformate Hash: semplici funzioni che, data una stringa, restituiscono un numero compreso nell'intervallo di variabilità della dimensione del file. Nel nostro caso, essendo limitato a soli 256 elementi, una buona funzione Hash potrebbe essere quella di sommare il codice ascii dei caratteri che formano la stringa, il tutto modulo 256 (il resto della divisione della somma con 256).

Cosa succede se per due chiavi diverse è generata la stessa posizione?

Siamo nel caso delle collisioni, peraltro inevitabili. In casi come questi, un metodo molto semplice per aggirare l'ostacolo è quello di cercare la prima posizione libera di seguito a quella già occupata. In lettura, se la posizione generata non è quella richiesta, si scandisce l'archivio a cominciare da questa posizione: sempre meglio che scandire tutto il file. Inoltre dato che le collisioni, statisticamente parlando, non si presentano subito, ma solo dopo aver inserito un certo numero di record, un piccolo miglioramento delle prestazioni si ha inserendo prima le registrazioni che interrogheremo più di sovente e poi quelle che ci serviranno di rado (ad es. il telefono delle persone antipatiche!).

Per migliorare ulteriormente l'andazzo delle collisioni, esistono altri metodi ausiliari, come quello delle catene confluenti, adoperato in RELARCH. In questo caso, oltre all'archivio relativo, sono mantenuti in memoria anche un indice di tutte le chiavi e un array ausiliario per memorizzare le posizioni delle chiavi che hanno generato una collisione. In questo modo, si accede al disco (che è la cosa che fa perdere più tempo) solo quando si conosce la posizione esatta della chiave e non una approssimata, come abbiamo visto sopra.

I comandi U1 e U2

Il driver 1541 ha anche la possibilità di leggere e/o modificare qualsiasi settore del dischetto. Grazie a questo sono possibili vari trucchetti, alcuni usati anche per proteggere programmi da copie illecite. In questa sede ci occuperemo di come cambiare il nome a un dischetto, senza riformattarlo, ossia senza perdere i programmi presenti. Per leggere un blocco, esiste un apposito comando detto User1 (abbr. "U1"). Tanto per cambiare, anche in que-

```
TRACCIA & SETTORE ? 18,0

  18   1  65   0  21 255 255  31  21 255
 255  31  21 255 255  31  21 255 255  31
  21 255 255  31  21 255 255  31  21 255
 255  31  21 255 255  31  21 255 255  31
  21 255 255  31  21 255 255  31  14  87
  95  21   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0
   0   0  16 236 255   7  19 255 255   7
  19 255 255   7  19 255 255   7  19 255
 255   7  19 255 255   7  19 255 255   7
  18 255 255   3  18 255 255   3  18 255
 255   3  18 255 255   3  18 255 255   3
  18 255 255   3  17 255 255   1  17 255
 255   1  17 255 255   1  17 255 255   1
  17 255 255   1  49  53  52  49  32  65
  82  84  73  67  79  76  79  32  32  32
 160 160  49  65 160  50  65 160 160 160
 160   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0
```

Programma 4 in esecuzione; è stata richiesta la traccia 18, settore 0.

```
LOAD"CATALOG",8

SEARCHING FOR CATALOG
LOADING
READY.
RUN

1541 ARTICOLO
PROVA.SEQ.W
DISK NAME
PROVA.SEQ.R
1541
CATALOG
TR.SE
AUTOCATALOG
RELARCH
? ■
```

Selezione programma con l'Utility Catalog.

sto caso, la prima cosa da fare è stabilire una comunicazione col disco, quindi:

```
OPEN 1,8,15
```

Oltre a questo, abbiamo bisogno di un canale e di un buffer per trasferire i 256 byte di un settore. Per far questo è sufficiente impartire il comando:

```
OPEN 4,8,4,"#"
```

in questo caso, il canale scelto è il 4 e il buffer allocato sarà scelto dal 1541, arbitrariamente (a noi, comunque, non importa). Per trasferire il blocco nel buffer, si impartirà il comando:

```
PRINT#1,"U1:"4;0;traccia;settore
```

Il 4 indica il canale, lo 0 il n. driver, di seguito si specificano la traccia e il settore interessati. In figura 2 sono riportati il n. di settori per le varie tracce. A questo punto, per accedere ai byte del blocco basta eseguire 256 volte il comando GET#4,A$ tenendo presente che A$="" indica che è stato letto un CHR$(0). Il programma 4 mostra come è possibile leggere un blocco qualsiasi. Analogamente si può scrivere un blocco, semplicemente caricando il buffer con 256 caratteri e impartendo il comando U2, nello stesso formato di U1.

Per cambiare il nome e l'ID di un dischetto, basta leggere il blocco 0 della traccia 18 (dove stanno memorizzati), apportare le dovute modifiche e riscrivere il blocco, s'intende nella stessa posizione. Il programma 7 fa appunto questo: permette di cambiare nome e ID di un dischetto.

Il disco da L.M.

Chi si occupa almeno un po' di linguaggio macchina avrà certamente sentito, prima o poi, la necessità di accedere all'unità a dischi. Per fare ciò si ricorre alle routine Kernal del sistema operativo. Sono un insieme di routine, direttamente richiamabili dall'utente col comando JSR, che riguardano appunto l'I/O: è impensabile che qualcuno se le costruisca in proprio, quando sono già belle e fatte da mamma Commodore.

Anche da L.M. per accedere al disco bisogna aprire canali e file. Per aprire un file, bisogna innanzitutto scrivere in qualche zona di memoria il nome del file, come insieme contiguo di byte rappresentanti i codici Ascii dei caratteri di cui è formato il nome. Le routine interessate all'apertura di un file sono tre e si trovano agli indirizzi $FFBA, $FFBD e $FFCØ.

La prima setta i tre parametri di una OPEN: il n. del file, il n. della periferica e l'indirizzo secondario (0 se non bisogna specificarlo). La seconda serve per specificare dove è stivato il nome del file e quanto è lungo (lunghezza 0 se non c'è). Se vogliamo da L.M. eseguire un:

```
OPEN 3,8,3,"ARCHIVIO,S,R"
```

basterà dapprima scrivere da qualche parte il nome (es. a $C243) e usare la sequenza:

```
LDA #$03
LDX #$08
LDY #$03
JSR $FFBA
LDA #$0C
LDX #$43
LDY #$C2
JSR $FFBD
JSR $FFC0
```

Prima di JSR $FFBA, si fissano in A il numero file, in X il device, in Y l'indirizzo secondario (il canale, nel nostro caso). Prima di JSR $FFBD abbiamo fissato in A la lunghezza del nome ($0C è 12) e nei registri X e Y la posizione del primo byte del nome. JSR $FFC0 apre, di fatto, il file. Per trasfe-

Cambiamo il numero di device

Il driver 1541, così come viene scartato dalla confezione, è il device n. 8 del bus IEEE/RS 232, utilizzato per comunicare con le periferiche Commodore: Stampante, Plotter, Unità a dischi e, speriamo, altro. Per utilizzare più di un floppy bisogna dare ad ognuno un diverso numero, in modo da poter selezionare, di volta in volta, la periferica che ci interessa. Si può procedere in due distinti modi: a livello software o a livello hardware. Nel primo caso, si ha lo svantaggio di dover inizializzare ad ogni accensione il suddetto numero. Procedendo invece col cacciavite e un buon taglierino la modifica resterà stabile, a meno di interventi "saldatorecci".

In questo caso la Commodore l'ha combinata grossa: la pagina 40 del manuale può sicuramente essere cestinata, non c'è una sola briciola di vero!

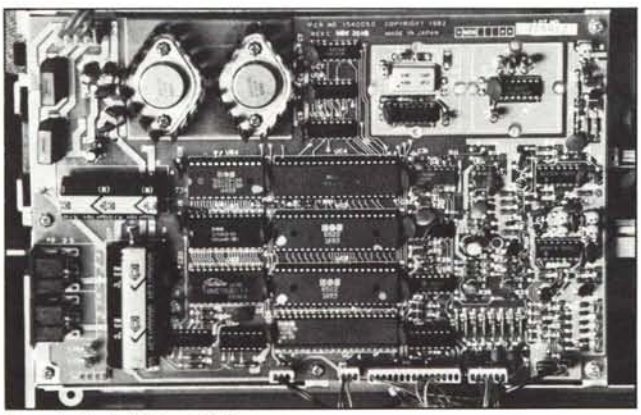
Per il procedimento software non c'è da dilungarsi in parole: per cambiare numero device è sufficiente far girare il programmino inserito in questo riquadro (lungi dall'essere stato ricopiato dal manuale!).

A seguire, di contro, le indicazioni del manuale per il procedimento hardware c'è quasi da impazzire: sono segnalate viti e coperchi in metallo inesistenti, jumper da cercare nel margine sinistro della scheda a mezz'altezza ... tutto falso!

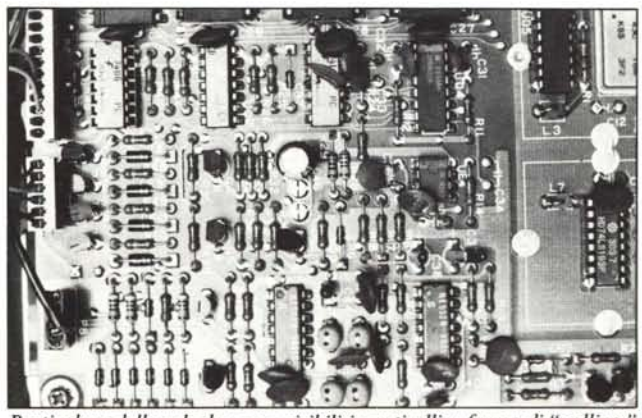
Per cambiare numero device, dopo aver aperto il driver svitando quattro viti poste sul fondo, bisogna tagliare con un taglierino molto affilato uno o tutti e due i ponticelli a forma di "pallina" situati nella parte inferiore della scheda (lato sportellino), ben visibili fra un transistor e un condensatore, come mostrato nelle foto. Tagliando il pallino inferiore, avremo il device 9, tagliando quello superiore, il 10; incidendoli tutti e due, il device n. 11. Per riottenere il device 8 bisogna far cadere una goccia di stagno sulle palline tagliate, al fine di ristabilire il contatto.

Col metodo software, la scelta è molto più ampia: si può anche arrivare a 20, se non di più.

```
10 OPEN 1,8,15:INPUT "N. DEVICE ";A
20 PRINT#1,"M-W"CHR$(119)CHR$(0)CHR$(2)
CHR$(A+32)CHR$(A+64)
30 CLOSE 1
```



La scheda del Drive 1541



Particolare della scheda: sono visibili i ponticelli a forma di "pallino."

Listato 5

```
10 INPUT"FORMATTO  ";A$
20 IFA$<>"SI"THENS0
30 INPUT"NOME & ID ";A$,B$
40 OPEN15,8,15, "N:"+A$+","+B$
50 OPEN1,8,1, "CATALOG"
60 PRINT#1,CHR$(1);CHR$(8);
70 FORI=0TO250
80 READA:PRINT#1,CHR$(A);
90 NEXT:CLOSE1:CLOSE15
100 REM
110 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
120 REM x                              x
130 REM x    A U T O C A T A L O G     x
140 REM x                              x
150 REM x          (C)  1984           x
160 REM x                              x
170 REM x        ADP  SOFTWARE         x
180 REM x                              x
190 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
200 REM
1000 DATA25,8,10,0,158,50,49,50,56,58,13
3,65,36,58,139,65,36,178,34,34,167
1010 DATA49,48,0,77,8,20,0,153,34,147,17
,17,17,76,79,65,68,34,199,40,51,52,41
1020 DATA170,65,36,170,199,40,51,52,41,3
4,44,56,44,49,19,34,58,151,49,57,56,44
1030 DATA49,58,151,54,51,49,44,49,51,0,0
,0,81,169,1,162,8,160,0,32,186,255,169
1040 DATA1,162,96,160,163,32,189,255,32,
192,255,162,1,32,198,255,169,13,32,210
1050 DATA255,32,207,255,201,34,208,12,32
,207,255,201,34,240,237,32,210,255,208
1060 DATA244,165,197,201,60,240,7,32,183
,255,201,64,208,224,32,204,255,169,1
1070 DATA32,195,255,96,0,0,187,33,251,65
,235,9,195,73,0,140,10,80,0,0,69,84,32
1080 DATA69,77,85,32,66,79,79,84,34,32,3
2,32,32,32,80,82,71,32,32,0,223,8,17
1090 DATA0,32,32,34,69,77,85,76,65,84,79
,82,34,32,32,32,32,32,32,32,32,32,80
1100 DATA82,71,32,32,32,0,255,8,35,0,32,
32,34,68,65,84,65,66,65,83,69,34,32,32
1110 DATA32,32,32,32,32,32,32,80,82,71,3
2,32,32,0,31
```

Programma AUTOCATALOG.

rire o prelevare dati dal disco (sempre un byte alla volta) esistono due apposite routine: $FFD2 e $FFCF; entrambe utilizzano l'accumulatore per il trasferimento.

Prima di utilizzarle, però, è obbligatorio specificare se il file appena aperto è di input o output.

Le routine interessate sono rispettivamente $FFC6 e $FFC9 e prima di invocarle bisogna mettere in X il numero del file cui si riferiscono (potremmo aver aperto più file!).

Nel caso si abbia a che fare con file sequenziali, per interrogare la variabile ST si può usare la routine $FFB7 che restituisce in A il valore dello status (quando è 64, l'ultimo carattere del file è stato letto).

Terminate le operazioni di lettura o scrittura, per chiudere canale e file si utilizzano le routine $FFCF (resetta tutti i canali di input e di output) e $FFC3 (chiude il file specificato nell'accumulatore).

Leggiamo la directory

Abbiamo già visto alcuni usi dell'indirizzo secondario, specificato all'atto dell'apertura di un file. Anche i programmi possono essere trattati come dei file, anzi per l'esattezza, sono dei file di tipo programma.

Per aprire un file di tipo PRG si usa come indirizzo secondario 0 o 1 a seconda che si voglia leggere o scrivere: c'è una netta corrispondenza con il LOAD e il SAVE vero e proprio.

Di fatto, quando si salva un programma, il 64 apre un file di tipo PRG (con indirizzo secondario 1) e riversa sul disco tutto quanto contenuto tra i due puntatori descritti all'inizio dell'articolo.

Volendo leggere un programma come se fosse un file, la prima operazione da compiere è:

```
OPEN 4,8,0,"NomeProgramma"
```

successivamente, ogni GET#4,A$ restituirà un carattere del programma, ricordando che A$= "" è CHR$(0) e che i primi due byte trasferiti sono il puntatore all'inizio area Basic dalla quale era stato salvato il programma.

Con questo stesso metodo, è possibile leggere la directory di un dischetto, semplicemente usando il comando:

```
OPEN 4,8,0,"$"
```

e ripetere GET#4,A$ fintantoché ST è diverso da 64.

Il programma 5, AUTOCATALOG, serve per formattare (se si desidera) un dischetto e inserire su di esso un programma "CATALOG" che semplifica l'operazione di selezione programma da caricare.

Una volta inizializzato il minifloppy con questa utility, digitando LOAD "CATALOG",8 e dopo il caricamento RUN, vedremo scorrere sul video la directory.

Alla fine un punto interrogativo e il cursore lampeggiante segnala che si è pronti a procedere.

Per selezionare un programma è sufficiente salire col cursore sul nome prescelto e premere [RETURN].

Per interrompere lo scroll prima del termine (per directory più lunghe di una schermata) basta premere la barra spaziatrice: per rivedere il tutto bisogna premere [RETURN] quando il cursore è accanto al punto interrogativo.

Giuro che è molto facile: provare per credere. MC

Listato 6

```
.,  0850 A9 01     LDA #$01
.,  0852 A2 08     LDX #$08
.,  0854 A0 00     LDY #$00
.,  0856 20 BA FF  JSR $FFBA
.,  0859 A9 01     LDA #$01
.,  085B A2 60     LDX #$60
.,  085D A0 A3     LDY #$A3
.,  085F 20 BD FF  JSR $FFBD
.,  0862 20 C0 FF  JSR $FFC0
.,  0865 A2 01     LDX #$01
.,  0867 20 C6 FF  JSR $FFC6
.,  086A A9 0D     LDA #$0D
.,  086C 20 D2 FF  JSR $FFD2
.,  086F 20 CF FF  JSR $FFCF
.,  0872 C9 22     CMP #$22
.,  0874 D0 0C     BNE $0882
.,  0876 20 CF FF  JSR $FFCF
.,  0879 C9 22     CMP #$22
.,  087B F0 ED     BEQ $086A
.,  087D 20 D2 FF  JSR $FFD2
.,  0880 D0 F4     BNE $0876
.,  0882 A5 C5     LDA $C5
.,  0884 C9 3C     CMP #$3C
.,  0886 F0 07     BEQ $088F
.,  0888 20 B7 FF  JSR $FFB7
.,  088B C9 40     CMP #$40
.,  088D D0 E0     BNE $086F
.,  088F 20 CC FF  JSR $FFCC
.,  0892 A9 01     LDA #$01
.,  0894 20 C3 FF  JSR $FFC3
.,  0897 60        RTS
```

Questo è il cuore del programma AUTOCATALOG.

Listato 7

```
100 DIMA$(255):OPEN1,8,15,"I"
110 OPEN3,8,3,"#"
120 PRINT#1,"U1:"3;0;18;0
130 FORI=0TO255:GET#3,A$:IFA$=""THENA$=CHR$(0)
140 A$(I)=A$:NEXT:A$="":PRINT"■■"
150 CLOSE3:CLOSE1:FORI=144TO159:A$=A$+A$(I):NEXT
160 PRINT"■■VECCHIO NOME = ";A$:A$=""
170 INPUT"■NUOVO   NOME = ";A$
180 A$=LEFT$(A$+"                ",16)
190 FORI=1TOLEN(A$):A$(143+I)=MID$(A$,I,1):NEXT
200 I$=A$(162)+A$(163):A$=""
210 PRINT"■■VECCHIO ID. = ";I$
220 INPUT"■NUOVO   ID. = ";A$
230 A$=A$+"  ":A$(162)=LEFT$(A$,1):A$(163)=MID$(A$,2,1)
240 OPEN1,8,15,"I"
250 OPEN3,8,3,"#"
260 PRINT#1,"B-P":3;0
270 FORI=0TO255:PRINT#3,A$(I);:NEXT
280 PRINT#1,"B-P":3;0
290 PRINT#1,"U2:"3;0;18;0
300 CLOSE3:PRINT#1,"I":CLOSE1:END
```

Il programma consente di cambiare nome e ID di un dischetto.