L'ADP Basic

51 nuovi comandi per le vostre periferiche

di Andrea de Prisco

Con questo numero inizia una serie di articoli sull'implementazione di nuove istruzioni Basic sul vostro Commodore 64.
Il tool presentato (unico nel suo genere) permette una gestione facilitata delle periferiche Commodore che, normalmente, intendono solo in termini di OPEN, CMD e PRINT#. Con l'ADP Basic per conoscere la Directory di un dischetto basterà digitare CAT, per stampare su carta si userà LPRINT (di sapore vagamente "ZXiano"...) per tracciare una linea col plotter il comando DRAW.
Procediamo con ordine...

Prima parte

Tre routine

Iniziamo subito col dire che, per aggiungere nuove istruzioni all'interprete Basic, bisogna mettere le mani in un bel po' di roba, quindi facile-facile non è. È necessario innanzitutto capire bene come funzionano tre routine del sistema operativo del 64: la Tokenize Routine, l'Execute Statements e la Perform List.

Anche se solo accennate nel numero scorso, queste tre sono a capo di tutto il funzionamento dell'interprete Basic.

La prima trasforma le linee da noi digitate in una forma più compatta per risparmiare spazio in memoria. Tutte le parole proprie del linguaggio sono trasformate in un opportuno codice a un solo byte. La seconda routine è invocata ogni qualvolta si deve eseguire uno statement Basic. È prelevato il codice token che identifica l'istruzione e si cede il controllo al pezzettino di interprete che la esegue. I linguaggi di programmazione interpretati funzionano così: per ogni statement si scrive una porzione di programma in linguaggio macchina che lo esegue, raccolte tutte queste "porzioni" se ne invoca una o un'altra a seconda di quale comando bisogna eseguire. In figura 1 è mostrato il diagramma a blocchi della Execute Statements. Ricordiamo che quando questa va in esecuzione, la linea è già stata tokenizzata.

Come riportato nel numero scorso, i codici token utilizzati vanno da 128 a 203. Per l'esattezza, da 128 a 162 sono istruzioni che possiamo trovare a inizio statement, mentre per valori superiori a 162 l'istruzione può stare solo nel corpo di un comando più complesso o dentro una espressione. Eccezion fatta per GO, codice token 203.

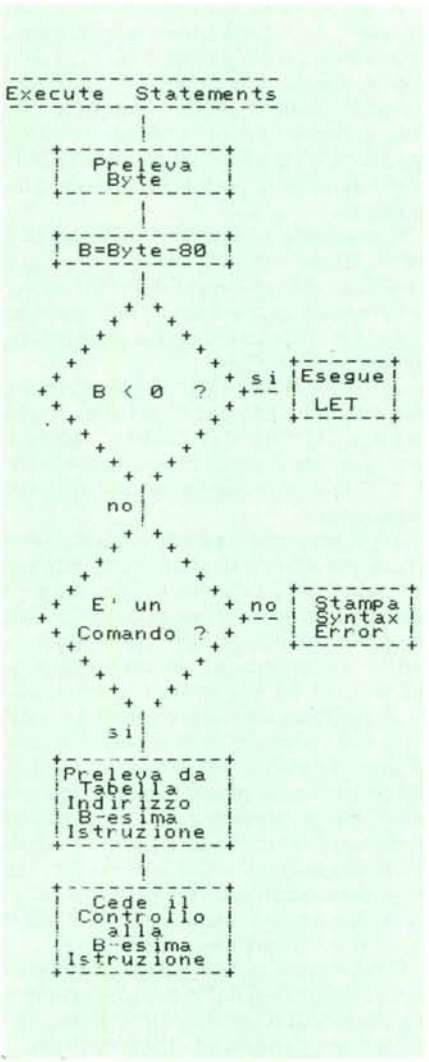

Figura 1 - Flow chart di Execute Statements.

La prima delle due diramazioni del diagramma di figura 1, seleziona il caso in cui il primo codice dell'istruzione non sia un token: in questo caso è invocato il comando LET (assegnamento variabile). La seconda diramazione controlla che il primo codice token appartenga a un comando e non a una funzione, pena segnalazione Syntax Error.

Se non siete convinti provate a digitare THEN 100 [Return].THEN ha codice token 167, maggiore quindi di 162, e non può stare ad inizio linea.

In figura 2 è mostrato il diagramma a blocchi della routine Perform List. Serve per riconvertire, all'atto di un LIST, in comandi e funzioni Basic i codici token mantenuti in memoria. La variabile APICI è usata come flag e serve per by-passare la detokenizzazione dopo l'apertura degli apici. "In che senso?"... qualcuno potrebbe obiettare. Chiariamo con un esempio: avete presente il cuoricino in reverse che identifica il tasto CLR (clear screen) dopo aver aperto gli apici?

Ebbene, ha come codici ASCII 147, lo stesso valore del codice token di LOAD. Quando si esegue un LIST, se il 64 incontra un 147 deve sapere se visualizzare un cuoricino in reverse o scrivere LOAD: in altre parole se sulla stessa linea siano stati aperti apici (e non richiusi) o meno.

Le tre routine sopra descritte fanno uso di alcune tabelle in rom contenenti la lista di tutte le istruzioni e la lista degli indirizzi di partenza di ognuna. In teoria, per aggiungere nuovi comandi, è sufficiente modificare le tabelle di cui sopra con i nomi e gli indirizzi di partenza dei nuovi statement. È anche vero però che essendo queste su rom, qualsiasi modifica è impossibile. Mamma Commodore ancora una volta ci viene incontro ponendo in alcune loca-

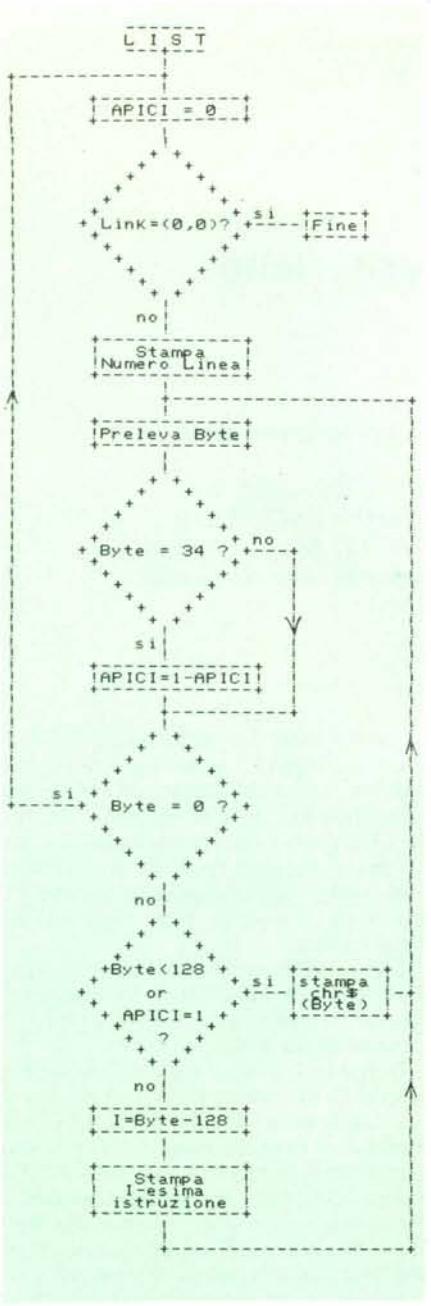

Figura 2 - Flow chart dell'istruzione List.

zioni ram gli indirizzi di partenza delle tre routine descritte: invece di modificare le tabelle o le singole routine, per inserire nuovi comandi le riscriveremo ex-novo (facenti riferimento a nuove tabelle estese), avendo l'accortezza di modificare gli indirizzi di partenza locati in ram. I byte 772 e 773 ($0304 e $0305) contengono l'indirizzo della Tokenize Routine, i byte 776 e 777 ($0308 e $0309) l'indirizzo della Execute Statements e i byte 774 e 775 ($0306 e $0307) l'indirizzo di Perform List. Basterà praticamente copiare queste routine in ram (dalla rom) modificando i puntatori alle tabelle, anch'esse opportunamente rilocate in ram. L'ADP Basic, per non occupare ram utente (come ormai è consuetudine della ADP SOFTWARE) si posiziona nei 4K ram liberi posti all'indirizzo $C000 (49152 dec.). È in questa zona di memoria che inseriremo le nuove routine ram e tutte le porzioncine di interprete, una per ogni nuova istruzione. I codici token utilizzati saranno quelli compresi tra 204 e 254, appositamente lasciati liberi dalla Commodore per espansioni del linguaggio Basic.

L'ADP Basic

La cosa più importante è inserire nuovi comandi Basic senza compromettere il regolare funzionamento delle istruzioni già esistenti: in altre parole scongiurare la ben più minima incompatibilità col CBM Basic di cui è fornita la macchina. Per quel che riguarda le tre routine sopra mostrate, non si corrono rischi, avendo la possibilità di ricopiarle integralmente dalla rom, apportando naturalmente le dovute modifiche. Il problema più grosso resta però l'integrazione dei nuovi comandi col resto del Basic: la possibilità di inserire su una stessa linea istruzioni di diversa paternità usando naturalmente come separatore i ":".

Curiosando tra metri e metri di disassemblato dell'interprete Basic CBM, si scopre che c'è una istruzione che possiamo sfruttare per arricchire il nostro linguaggio standard. Anche se a qualcuno potrà sembrare strano, si sta parlando del comando DATA. Questo infatti non è altro che uno statement nullo: quando l'interprete l'incontra, non fa altro che passare a nuova istruzione. È quando si esegue un READ che l'interprete la cerca tra le linee del programma.

Si potrebbe obiettare: "Anche REM è un'istruzione nulla...". Esatto! ma ha il considerevole svantaggio di far procedere l'interprete a nuova linea, e non a nuova istruzione. Per convincerci di ciò digitate:

```
10 REM : PRINT "CIAO"
```

e date il RUN. Non si avrà alcun effetto; incontrata la REM (per l'esattezza: il codice token di REM) il Basic del 64 ignora il resto della linea, come se fosse un commento. I ":" in linee di questo tipo non vengono considerati.

Resta però sempre l'interrogativo principale: perché mai sfruttare l'esistenza del comando DATA. La risposta è semplice (almeno si spera!): incontrando un comando ADP BASIC, l'interprete modificato cederà il controllo al pezzettino di programma in L.M. che esegue tale istruzione. Al termine, per continuare l'esecuzione del resto della linea, ci sarà un salto brutale all'interpretazione dell'istruzione DATA, che come visto fa proprio quello che a noi serve. Resta inteso che il tutto funziona perfettamente anche se usiamo comandi ADP singolarmente. Nulla ci vieta di imbrogliare ugualmente l'interprete facendogli credere che sia una linea DATA, dopo aver svolto la funzione richiesta.

Questo per quanto riguarda la maggior parte dei comandi ADP. I rimanenti sfruttano altri stratagemmi, come vedremo, per assicurare ugualmente la piena compatibilità.

```
C03B 20 CC FF JSR $FFCC
C03E 20 E7 FF JSR $FFE7
C041 A9 48    LDA #$48
C043 A2 08    LDX #$08
C045 A0 0F    LDY #$0F
C047 20 BA FF JSR $FFBA
C04A A9 00    LDA #$00
C04C 20 BD FF JSR $FFBD
C04F 20 C0 FF JSR $FFC0
C052 60       RTS
```

Listato 1 - Questa routine apre il canale di comunicazione col disco (OPEN 1, 8, 15).

```
C400 20 3B C0 JSR $C03B
C403 A2 48    LDX #$48
C405 20 C9 FF JSR $FFC9
C408 A9 49    LDA #$49
C40A 20 D2 FF JSR $FFD2
C40D 20 E7 FF JSR $FFE7
C410 4C F8 A8 JMP $A8F8
```

Listato 2 - Comando INIT.

```
C413 20 3B C0 JSR $C03B
C416 A2 48    LDX #$48
C418 20 C9 FF JSR $FFC9
C41B A9 56    LDA #$56
C41D 20 D2 FF JSR $FFD2
C420 20 E7 FF JSR $FFE7
C423 4C F8 A8 JMP $A8F8
```

Listato 3 - Comando VDATE.

```
C426 20 3B C0 JSR $C03B
C429 A2 48    LDX #$48
C42B 20 C9 FF JSR $FFC9
C42E A9 53    LDA #$53
C430 20 D2 FF JSR $FFD2
C433 D0 43    BNE $C478
```

Listato 4 - Comando ERASE.

I comandi

Possiamo suddividere l'intero set di comandi dell'ADP Basic in tre categorie:

a) Comandi che non necessitano passaggio di parametri

b) Comandi che necessitano passaggio di parametri

c) Comandi che restituiscono valori sul video.

Analizzeremo dapprima i comandi appartenenti alla prima delle tre categorie, essendo, di fatto, la più semplice.

Un comando che non necessita di parametri è INIT, ed è usato per inizializzare il driver. Listato 2 rappresenta l'implementazione di tale istruzione. Come si può notare, non occupa molti byte.

JSR $C03B è un salto a una subroutine (listato 1) che apre il canale di comunica-

zione col disco, dopo aver resettato gli altri file aperti. Il file usato per comunicare ha numero logico 72 ($48 in hex). Tornando alla nostra INIT di listato 2, dopo il "salto" a $C03B, si seleziona come canale di output il file $48. Si esegue cioè LDX #$48 e JSR $FFC9. A questo punto, per l'inizializzazione è sufficiente "sparare" una "I" al disco: basta un LDA #$49 e un JSR $FFD2; infatti $49 è il codice ASCII della "I" e $FFD2 è l'indirizzo della routine che "spara". Per l'elenco e la descrizione delle routine Kernal adoperate, si dia uno sguardo al riquadro di questa pagina.

Non resta che chiudere il file aperto con un JSR $FFE7 e saltare come promesso (JMP $A8F8) all'implementazione dell'istruzione DATA per proseguire la normale esecuzione di altri statement Basic.

Il listato 3 rappresenta l'istruzione VDATE, anch'essa priva di parametri, da utilizzare per convalidare i blocchi usati e liberi di un dischetto. L'unica differenza con la INIT, sta nella "V" inviata al disco in luogo della "I". Il codice ASCII della "V" è appunto $56, come si può notare dalla linea C41B.

Analizziamo ora i comandi appartenenti alla seconda categoria, ossia comandi che necessitano di passaggio parametri.

Il listato 5 mostra l'istruzione FMAT, da usare (con attenzione) per formattare un dischetto. Ricordiamo che l'operazione di formattazione cancella qualsiasi informazione contenuta sul dischetto. Come per l'uso normale, per formattare un dischetto occorre specificare un nome e facoltativamente un identificatore di due caratteri. L'identificatore è obbligatorio solo se il dischetto è nuovo, ossia mai formattato.

Facendo un esempio, se intendiamo chiamare un dischetto PIPPO e dare ad esso l'identificatore PP, basterà digitare:

```
FMAT "PIPPO,PP"
```

se non si vuol specificare l'ID, è sufficiente:

```
FMAT "PIPPO"
```

Vediamo passo-passo come funziona la

```
C435 20 3B C0  JSR $C03B
C438 A2 48     LDX #$48
C43A 20 C9 FF  JSR $FFC9
C43D A9 4E     LDA #$4E
C43F 20 D2 FF  JSR $FFD2
C442 A9 3A     LDA #$3A
C444 20 D2 FF  JSR $FFD2
C447 20 D4 E1  JSR $E1D4
C44A A0 00     LDY #$00
C44C B1 BB     LDA ($BB),Y
C44E 20 D2 FF  JSR $FFD2
C451 C8        INY
C452 C4 B7     CPY $B7
C454 D0 F6     BNE $C44C
C456 20 CC FF  JSR $FFCC
C459 4C F8 A8  JMP $A8F8
```

Listato 5 - Comando FMAT.

FMAT. Torniamo dunque al listato 5. La prima operazione è il solito "salto" a $C03B per stabilire comunicazione col disco.

Indi, come per la INIT e la VDATE, si dichiara il file $48 come canale di output. Segue una "doppietta" costituita da una "N" e dai ":" (codici $4E e $3A).

A questo punto bisogna inviare al disco il nome del dischetto e eventualmente l'ID. In un sol colpo, tutto quanto contenuto tra apici. Il programmino FMAT deve recuperare dalla linea Basic la stringa che segue la parola FMAT (nulla vieta di mettere una variabile o una espressione contorta quanto si vuole, purché di tipo stringa). Per fare ciò ci avvaliamo di una routine del sistema operativo usata dai comandi LOAD, SAVE e VERIFY per leggere (dal video o dal listato) il nome del programma "incriminato".

Questa routine è locata a $E1D4 e, una volta invocata, restituisce nel byte $B7 la lunghezza e in ($BB, $BC) l'indirizzo in memoria dove è stata trasferita la stringa. Il ciclo compreso tra C44C e C454 non fa altro che inviare al disco la stringa (carattere per carattere) posta dopo FMAT. Il tutto si conclude con un JSR $FFCC per chiudere il canale aperto e, tanto per cambiare, un salto a $A8F8.

Un piccolo quiz per i lettori: perché la VDATE non è stata chiamata VALIDATE e la FMAT, FORMAT? Non è per risparmiare byte, il motivo è assai più serio...

```
C46B 20 3B C0  JSR $C03B
C46E A2 48     LDX #$48
C470 20 C9 FF  JSR $FFC9
C473 A9 52     LDA #$52
C475 20 D2 FF  JSR $FFD2
C478 A9 3A     LDA #$3A
C47A 20 D2 FF  JSR $FFD2
C47D 20 D4 E1  JSR $E1D4
C480 A0 00     LDY #$00
C482 B1 BB     LDA ($BB),Y
C484 20 D2 FF  JSR $FFD2
C487 C8        INY
C488 C4 B7     CPY $B7
C48A D0 F6     BNE $C482
```

Listato 7 - Comando RENAME.

Il comando RENAME (listato 7), è molto simile al comando FMAT: uniche sostanziali differenze sono la "R" in luogo della "N" alla linea C473 e la mancanza del JMP $A8F8 finale: al termine l'istruzione casca nell'istruzione FLASH che, come vedremo più avanti, visualizza i messaggi del disco. Ciò è stato necessario per poter implicitamente controllare che il RENAME abbia dato buon esito.

La sintassi è:

```
RENAME "NuovoNome=VecchioNome"
```

dove NuovoNome e VecchioNome sono rispettivamente il ... nuovo nome e il vec-

Le routine del Kernal

Essendo l'ADP Basic un tool di comandi orientati all'uso semplificato delle periferiche, per l'implementazione si è fatto largo uso delle routine Kernal del sistema operativo del 64. La stessa Commodore l'ha disposte ben ordinate e facilmente utilizzabili dall'utente per interfacciare (a livello soft) il mondo della programmazione in linguaggio macchina con i vari device di cui si dispone.

Vediamo il funzionamento delle principali.

$FFBA: set logical, first and second address
Registri di Comunicazione: A, X, Y
Descrizione: Setta il numero logico, l'indirizzo primario (il n. di device) e l'indirizzo secondario di un file prima dell'apertura. In A si pone il n. File, in X il n. device e in Y l'ind. secondario.

$FFBD: set file name information
Registri di comunicazione: A, X, Y
Descrizione: Setta le informazioni circa il nome di un file prima dell'apertura. Si inserisce in A la lunghezza del nome, in X e in Y l'indirizzo dove è stivato (X parte bassa e Y parte alta).

$FFC0: open logical file
Registri di comunicazione: nessuno
Preroutine: $FFBA, $FFBD
Descrizione: Apre il file specificato con le routine $FFBA e $FFBD.

$FFC3: close logical file
Registri di comunicazione: A
Descrizione: Chiude il file il cui numero è specificato in A.

$FFC6: open channel for input
Registri di comunicazione: X
Preroutine: $FFC0
Descrizione: Predispone un file precedentemente aperto come canale di input (trasferimento dati da periferica a CPU).

$FFC9: open channel for output
Registri di comunicazione: X
Preroutine: $FFC0
Descrizione: Predispone un file precedentemente aperto come canale di output (trasferimento dati da CPU a periferica).

$FFCC: close input and output channel
Registri di comunicazione: nessuno
Descrizione: Resetta i canali di input e di output (i file aperti restano tali).

$FFCF: input character from channel
Registri di comunicazione: A
Descrizione: ogni chiamata a questa routine provoca un get per il file input precedentemente aperto. In A il codice ASCII del carattere letto.

$FFD2: output character to channel
Registri di comunicazione: A
Descrizione: Scarica nel file output precedentemente aperto il carattere contenuto in A.

$FFE7: close all file
Registri di comunicazione: nessuno
Descrizione: chiude tutti i file aperti (resetta la file table del 64)

$FFB7: Read I/O status word
Registri di comunicazione: A
Descrizione: trasferisce in A il valore delle variabili ST. Usata per conoscere se è stato letto l'ultimo carattere di un file (A = 64).
Se nessun file è aperto, si assume come canale di output il video e come canale di input la tastiera.

chio nome del file "preso di mira". I comandi ERASE e COPY, rispettivamente listato 4 e 6, sfruttano l'esistenza della RENAME per non sprecare spazio. Essendo in buona parte identica, a un certo punto, sia l'una che l'altra saltano nel corpo della RENAME.

La sintassi di ERASE è:

```
ERASE "NomeFile"
```

dove "NomeFile" è "naturalmente" il nome del file da eliminare.

La sintassi di COPY è:

```
COPY "NuovoFile=VecchioFile"
```

oppure:

```
COPY "NuovoFile=VecchioFile1, VecchioFile2"
```

a seconda che si voglia creare sul dischetto un file a partire da uno o più file già esistenti.

Per quanto riguarda l'istruzione DLOAD, la sintassi è:

```
DLOAD "NomeProgramma"
```

e permette di caricare programmi da disco. Per implementare tale comando, ci rifaremo al normalissimo LOAD del 64, che, come arcinoto, per caricare un programma da disco, necessita della specifica "virgola otto" a piè del "NomeProgramma". È il listato 9. Sì, tutto lì. Il byte $0A indica se vogliamo un LOAD o un VERIFY. Segue un JSR $E1D4 per leggere il "NomeProgramma", e un LDA #$08 STA $BA per scegliere come periferica il disco (che ha appunto come numero di device 8). Infine un salto a $E16F, nel cuore della normalissima LOAD, figlia di mamma Commodore.

Di stessa fattura anche DSAVE (listato 10) e DVER (listato 11) per salvare e verificare un programma sul disco.

La loro sintassi, come intuibile, è:

```
DSAVE "NomeProgramma"
```

e

```
OVER "NomeProgramma"
```

Gli ultimi due comandi presentati in questa puntata, appartengono alla terza delle tre categorie viste precedentemente. Restituiscono cioè valori sul video: l'interazione è tra driver e schermo. Il primo è il comando FLASH e si usa quando la spia del drive inizia a lampeggiare (flashing) segnalando il verificarsi di un errore. Numero, tipo, traccia e settore vengono mostrati su video e la spia smette di lampeggiare. Se il comando FLASH viene impartito senza il verificarsi del lampeggio, un tranquillizzante messaggio 00, OK, 00, 00 sarà visualizzato.

Il listato 8 è l'implementazione di tale comando. Notare l'estrema semplicità. Per motivi prettamente estetici, la prima operazione che si compie è di stampare un [RETURN] per far iniziare a nuova riga la scrittura del messaggio di errore. Subito dopo, il classico JSR $C03B stabilisce la comunicazione col disco. LDX #$48 e JSR

Formattiamo un floppy con nome DISCO4 e identificatore D4.

Supponendo di aver lasciato lo sportellino del driver aperto, la spia inizierà a lampeggiare. Il comando FLASH indicherà l'errore rilevato.

```
C45C 20 3B C0 JSR $C03B
C45F A2 48    LDX #$48
C461 20 C9 FF JSR $FFC9
C464 A9 43    LDA #$43
C466 20 D2 FF JSR $FFD2
C469 D0 0D    BNE $C478
```

Listato 6 - Comando COPY.

```
C4AC A9 00    LDA #$00
C4AE 85 0A    STA $0A
C4B0 20 D4 E1 JSR $E1D4
C4B3 A9 08    LDA #$08
C4B5 85 BA    STA $BA
C4B7 4C 6F E1 JMP $E16F
```

Listato 9 - Comando DLOAD.

```
C4BA 20 D4 E1 JSR $E1D4
C4BD A9 08    LDA #$08
C4BF 85 BA    STA $BA
C4C1 4C 59 E1 JMP $E159
```

Listato 10 - Comando DSAVE.

```
C4C4 A9 01    LDA #$01
C4C6 D0 E6    BNE $C4AE
```

Listato 11 - Comando DVER.

```
C48C A9 0D    LDA #$0D
C48E 20 D2 FF JSR $FFD2
C491 20 3B C0 JSR $C03B
C494 A2 48    LDX #$48
C496 20 C6 FF JSR $FFC6
C499 20 CF FF JSR $FFCF
C49C 20 D2 FF JSR $FFD2
C49F 20 B7 FF JSR $FFB7
C4A2 C9 40    CMP #$40
C4A4 D0 F3    BNE $C499
C4A6 20 3B C0 JSR $C03B
C4A9 4C F8 A8 JMP $A8F8
```

Listato 8 - Comando FLASH.

Senza l'ADP ...

I comandi dell'ADP BASIC presentati su questo numero provvedono a spedire comandi al disco, interrogarlo circa segnalazioni di errore, a leggere la Directory, a salvare o caricare un programma da disco senza l'assillo del "virgola otto".

Senza ADP BASIC, per spedire un comando al disco è necessario innanzitutto aprire un canale di comunicazione. Ciò avviene di solito col comando:

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

a questo punto, ognuno dei 6 comandi è spedibile tramite l'istruzione:

```
PRINT#1, <Comando>
```

<Comando> può essere:

— "N:NomeDisco, ID" (Formatta un dischetto di nome = NomeDisco e Identificatore = ID)

— "V" (Convalida dei Blocchi liberi o occupati di un dischetto)

— "I" (Inizializzazione Driver)

— "S:NomeFile" (Cancella dal dischetto il File denominato NomeFile)

— "R:NuovoNome=VecchioNome" (si usa per cambiare il nome a un file su dischetto)

— "C:NuovoFile=VecchioFile" oppure:

— "C:NuovoFile=VecchioFile1,VecchioFile2" (si usa per creare copie di file sullo stesso dischetto a partire da uno o più file già esistenti, concatenandoli).

Per conoscere la directory si usa invece leggere il file programma "$". Il maggior svantaggio di questa procedura è la perdita del programma Basic mantenuto in memoria.

Per conoscere i messaggi di errore segnalati dal frenetico lampeggio della spia, si usa il comando:

```
INPUT1,A,B$,C,D:PRINT A,B$,C,D
```

naturalmente inserito in una linea di programma Basic essendo l'istruzione INPUT non direttamente eseguibile da tastiera.

I comandi ADP BASIC corrispondenti a tutto il macello sovrastante sono:

FMAT per formattare
VDATE per validare i blocchi
INIT per inizializzare il driver
ERASE per cancellare un file
RENAME per ridenominare un file
COPY per copiare file
CAT per visualizzare la directory (senza perdite)
FLASH per leggere messaggi di errore del disco (spia rossa flashing)
DLOAD per caricare un programma
DSAVE per salvarlo
DVER per verificare la buona riuscita di un "salvataggio"

Tutto qui. S'intende per questo numero!

Il comando CAT mostra la directory di un dischetto.

A questo punto, se si vuol leggere un programma è sufficiente digitare DLOAD + SPAZIO + [RETURN].

$FFC6 indicano come input il canale appena aperto. Siamo ora nel vivo del "ciclo": si esegue la lettura di un carattere dal disco e la stampa del medesimo su video, fino a quando l'ultimo carattere (del messaggio di errore) non è stato letto. La subroutine JSR $FFB7 restituisce in A il valore della parola di stato I/O, detta in Basic ST, che assume valore 64 ($40 in hex) appena si è letto l'ultimo carattere di un file.

La routine di FLASH termina con un nuovo salto a $C03B per resettare il canale di comunicazione, lasciandolo aperto per futuri dialoghi.

L'ultimo comando è CAT, e serve per visualizzare la directory di un dischetto su video, senza perdere il programma mantenuto in memoria, come generalmente accade con la sequenza:

```
LOAD "$", 8
LIST
```

Sul numero 33 abbiamo già visto, parlando del trattamento file col driver 1541, come è possibile leggere da programma la directory di un dischetto. In sunto è sufficiente aprire un file programma (indirizzo secondario 0) di nome "$" e leggere e stampare uno per volta i caratteri passati dal driver. In maniera molto simile al comando FLASH.

Per raffinare un po' la soluzione e rendere più elegante la visualizzazione della directory (e come vedremo anche più pratica), per ogni file è stampato dapprima il numero di blocchi occupati, di seguito il tipo del file (PR, SE, RE o US) e in ultimo il nome. Ciò per permettere di posizionarsi col cursore sul file programma da caricare, digitare DLOAD (più uno spazio) e battere [RETURN]. Digitando infatti DLOAD, pestiamo il numero di blocchi e il tipo del file, lasciando al momento del [RETURN] la linea pronta per essere eseguita.

Il listato 13 è una routine utilizzata dal comando CAT per visualizzare il numero dei blocchi in modo formattato, con le unità, decine e centinaia incolonnate (dite la verità, incolonnato è bello!).

All'indirizzo $C513 ha inizio il programma del comando CAT (listato 12). La prima operazione che si compie è l'apertura di un file programma di nome "$". Nel byte $A360 c'è appunto il codice ASCII del simbolo "$". Il file è naturalmente di INPUT, perciò la sequenza LDX #$41 e JSR $FFC6. Seguono qualche JSR $FFCF per posizionarsi su caratteri utili del file dollaro che stiamo leggendo. Il JSR $FFE1 e $C53E serve per controllare la pressione del tasto RUN/STOP: in seguito a questa è fermata la lettura e il 64 torna allo stato di READY.

La parte "ciclica" della routine si svolge in breve così:

— si legge il numero di blocchi di un file, caricando la parte bassa in A e la parte alta in X.

— un salto a $C4C8 stampa tale valore in modo formattato.

— si legge il nome del file, come tutti i caratteri compresi tra l'apertura e la chiusura degli apici, e lo si parcheggia momentaneamente a partire dal byte $02C3.

— è letto il tipo del file e posizionato (i primi due caratteri) nelle locazioni $02C1 e $02C2.

— il contenuto delle celle $02C0 e seguenti è stampato sul video.

Il tutto si ferma quando al posto degli apici si incontra la "B" di "BLOCKS FREE" e quindi tutta la directory è stata letta.

Nei prossimi numeri

Questo mese ci fermiamo qui. Possiamo darvi qualche anticipazione sul seguito. Sul prossimo numero continueremo la nostra carrellata sugli ADP-comandi rivolti al disco. Vedremo l'APPEND, per attaccare due programmi; il VIEW, per vedere un listato da disco senza occupare la memoria; DISKNAME, per cambiare nome e ID a un dischetto; TRSE, per curiosare tra le tracce; e ancora: BSAVE e BLOAD per salvare e caricare programmi in linguaggio macchina o più in generale porzioni di me-

```
C513 A9 41     LDA #$41
C515 A2 08     LDX #$08
C517 A0 00     LDY #$00
C519 20 BA FF  JSR $FFBA
C51C A9 01     LDA #$01
C51E A2 60     LDX #$60
C520 A0 A3     LDY #$A3
C522 20 BD FF  JSR $FFBD
C525 20 C0 FF  JSR $FFC0
C528 A2 41.    LDX #$41
C52A 20 C6 FF  JSR $FFC6
C52D 20 CF FF  JSR $FFCF
C530 20 CF FF  JSR $FFCF
C533 20 CF FF  JSR $FFCF
C536 20 CF FF  JSR $FFCF
C539 A9 0D     LDA #$0D
C53B 20 D2 FF  JSR $FFD2
C53E 20 E1 FF. JSR $FFE1
C541 F0 68     BEQ $C5AB
C543 20 B7 FF  JSR $FFB7
C546 D0 63     BNE $C5AB
C548 20 CF FF  JSR $FFCF
C54B 85 FE     STA $FE
C54D 20 CF FF  JSR $FFCF
C550 A6 FE     LDX $FE
C552 20 C8 C4  JSR $C4C8
C555 20 CF FF  JSR $FFCF
C558 EA        NOP
C559 EA        NOP
C55A C9 22     CMP #$22
C55C F0 0E     BEQ $C56C
C55E C9 42     CMP #$42
C560 D0 F3     BNE $C555
C562 20 D2 FF  JSR $FFD2
C565 20 CF FF  JSR $FFCF
C568 D0 F8     BNE $C562
C56A F0 C7     BEQ $C533
C56C A2 00     LDX #$00
C56E 9D C3 02  STA $02C3,X
C571 E8        INX
C572 20 CF FF  JSR $FFCF
C575 9D C3 02  STA $02C3,X
C578 C9 22     CMP #$22
C57A D0 F5     BNE $C571
C57C 20 CF FF  JSR $FFCF
C57F C9 20     CMP #$20
C581 F0 F9     BEQ $C57C
C583 8D C0 02  STA $02C0
C586 20 CF FF  JSR $FFCF
C589 8D C1 02  STA $02C1
C58C A9 20     LDA #$20
C58E 8D C2 02  STA $02C2
C591 EA        NOP
C592 A9 00     LDA #$00
C594 9D C3 02  STA $02C3,X
C597 A2 FF     LDX #$FF
C599 E8        INX
C59A BD C0 02  LDA $02C0,X
C59D F0 05     BEQ $C5A4
C59F 20 D2 FF  JSR $FFD2
C5A2 D0 F5     BNE $C599
C5A4 20 CF FF  JSR $FFCF
C5A7 D0 FB     BNE $C5A4
C5A9 F0 88     BEQ $C533
C5AB 20 CC FF  JSR $FFCC
C5AE A9 41     LDA #$41
C5B0 20 C3 FF  JSR $FFC3
C5B3 4C F8 A8  JMP $A8F8
```

Listato 12 - Comando CAT.

```
100 FORI=49152TO49785:READII
110 POKEI,II:NEXT
120 FORI=50176TO50613:READII
130 POKEI,II:NEXT
140 FORI=52480TO52785:READII
150 POKEI,II:NEXT
290 SYS49152:END
300 ++++++++++++++++++++++++++++++++++++++++++++++++++
310 +                                                +
320 +   ***  ****  ****     ****   ***   ***  ***  ***   +
330 +  *   * *   * *   *    *   * *   * *      *   *   *  +
340 +  ***** *   * ****     ****  *****  ***   *   *      +
350 +  *   * *   * *  *     *   * *   *    *  *   *   *  +
360 +  *   * * **** *  *    ****  *   *  ***  ***  ***   +
370 +                                                +
380 +            ---------------------               +
390 +            (C)  1984     ADP SOFTWARE          +
400 +            ---------------------               +
410 +                                                +
500 ++++++++++++++++++++++++++++++++++++++++++++++++++
1000 DATA120,169,30,141,4,3,169,193,141,5,3,169,68,141,6,3,169,194,141,7,3,234
1010 DATA234,169,249,141,8,3,169,192,141,9,3,208,48,32,204,255,32,231,255,169
1020 DATA68,162,4,160,0,32,186,255,169,0,32,189,255,32,192,255,96,32,204,255
1030 DATA32,231,255,169,72,162,8,160,15,32,186,255,169,0,32,169,255,32,192,255
1040 DATA96,169,11,141,32,208,141,33,208,169,155,32,210,255,96,0,0,0,0,0,0,0
1050 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,195,18,196,37,196
1060 DATA52,196,91,196,106,196,139,196,171,196,185,196,195,196,18,197,0,0,0,0
1070 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1080 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1090 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,115,0,201,128
1100 DATA48,4,201,204,16,4,56,76,231,167,56,233,204,10,168,169,167,72,169,233
1110 DATA72,165,129,192,72,165,123,192,72,76,115,0,166,122,160,4,132,15,189,0
1120 DATA2,16,7,201,255,240,73,232,208,244,201,32,240,66,133,8,201,34,240,97
1130 DATA36,15,112,56,201,63,208,4,169,153,208,48,201,48,144,4,201,60,144,40
1140 DATA132,113,160,0,169,204,133,254,169,1,133,253,132,11,136,134,122,202,208
1150 DATA208,2,230,254,232,189,0,2,56,241,253,240,242,201,128,208,48,5,11,164
1160 DATA113,232,200,153,251,1,185,251,1,240,60,56,233,58,240,4,201,73,208,2
1170 DATA133,15,56,233,85,208,148,133,8,189,0,2,240,223,197,8,240,219,200,153
1180 DATA251,1,232,208,240,166,122,230,11,198,253,200,208,2,230,254,177,253,16
1190 DATA247,230,253,177,253,208,175,189,0,2,16,184,153,253,1,198,123,169,255
1200 DATA133,122,96,144,6,240,4,201,171,208,247,32,107,169,32,19,166,32,121,0
1210 DATA240,12,201,171,208,232,32,115,0,32,107,169,208,224,104,104,165,20,5
1220 DATA21,208,6,169,255,133,20,133,21,160,1,132,15,177,95,240,67,32,44,168
1230 DATA32,215,170,200,177,95,170,200,177,95,197,21,208,4,228,20,240,2,176,44
1240 DATA132,73,32,205,189,169,32,164,73,41,127,32,71,171,201,34,208,6,165,15
1250 DATA73,255,133,15,200,240,17,177,95,208,16,168,177,95,170,200,177,95,134
1260 DATA95,133,96,208,181,76,134,227,108,6,3,16,215,201,255,240,211,36,15,48
1270 DATA207,56,233,127,170,132,73,160,255,169,204,133,255,169,1,133,254,202
1280 DATA240,11,200,208,2,230,255,177,254,16,247,48,242,200,208,2,230,255,177
1290 DATA254,48,164,32,71,171,208,242
1999 REM ************************************************************
2000 DATA32,59,192,162,72,32,201,255,169,73,32,210,255,32,231,255,76,248,168
2010 DATA32,59,192,162,72,32,201,255,169,86,32,210,255,32,231,255,76,248,168
2020 DATA32,59,192,162,72,32,201,255,169,83,32,210,255,208,67,32,59,192,162,72
2030 DATA32,201,255,169,78,32,210,255,169,58,32,210,255,32,212,225,160,0,177
2040 DATA187,32,210,255,200,196,183,208,246,32,204,255,76,248,168,32,59,192,162
2050 DATA72,32,201,255,169,67,32,210,255,208,13,32,59,192,162,72,32,201,255,169
2060 DATA82,32,210,255,169,58,32,210,255,32,212,225,160,0,177,187,32,210,255
2070 DATA200,196,183,208,246,169,13,32,210,255,32,59,192,162,72,32,198,255,32
2080 DATA207,255,32,210,255,32,183,255,201,64,208,243,32,59,192,76,248,168,169
2090 DATA0,133,10,32,212,225,169,8,133,186,76,111,225,32,212,225,169,8,133,186
2100 DATA76,89,225,169,1,208,230,133,98,134,99,162,144,56,32,73,188,32,223,189
2110 DATA133,253,132,254,160,255,162,0,200,177,253,208,251,132,252,56,169,3,229
2120 DATA252,168,240,9,169,32,157,60,3,232,136,208,249,177,253,157,60,3,232,200
2130 DATA201,0,208,245,202,169,32,157,60,3,232,169,0,157,60,3,169,60,160,3,76
2140 DATA30,171,169,65,162,3,160,0,32,186,255,169,1,162,96,160,163,32,189,255
2150 DATA32,192,255,162,65,32,198,255,32,207,255,32,207,255,32,207,255,32,207
2160 DATA255,169,13,32,210,255,32,225,255,240,104,32,183,255,208,99,32,207,255
2170 DATA133,254,32,207,255,166,254,32,200,196,32,207,255,234,234,201,34,240
2180 DATA14,201,96,208,243,32,210,255,32,207,255,208,248,240,199,162,0,157,195
2190 DATA2,232,32,207,255,157,195,2,201,34,208,245,32,207,255,201,32,240,249
2200 DATA141,192,2,32,207,255,141,193,2,169,32,141,194,2,234,169,0,157,195,2
2210 DATA162,255,232,189,192,2,240,5,32,210,255,208,245,32,207,255,208,251,240
2220 DATA136,32,204,255,169,65,32,195,255,76,248,168
2999 REM ************************************************************
3000 DATA0,69,79,196,70,79,210,78,69,88,212,68,65,84,193,73,78,80,85,64,163,73
3010 DATA78,80,85,212,68,73,205,82,69,65,196,76,69,212,71,79,84,207,82,85,206
3020 DATA73,198,82,69,83,84,79,82,197,71,79,83,85,194,82,69,84,85,82,206,82,69
3030 DATA205,83,84,79,208,79,206,87,65,73,212,76,79,65,196,63,65,86,197,86,69
3040 DATA82,73,70,217,68,69,198,80,79,75,197,80,82,73,78,84,163,80,82,73,78,212
3050 DATA67,79,78,212,76,73,83,212,67,76,210,67,77,196,83,89,211,79,80,69,206
3060 DATA67,76,79,83,197,71,69,212,78,69,215,84,65,66,168,84,207,70,206,83,80
3070 DATA67,158,84,72,69,205,78,79,212,83,84,69,208,171,173,170,175,222,65,78
3080 DATA196,79,210,190,189,188,83,71,206,73,78,212,65,66,211,85,83,210,70,82
3090 DATA197,80,79,211,83,81,210,82,78,196,76,79,193,69,86,208,67,79,211,83,73
3100 DATA206,84,65,206,65,84,206,80,69,69,203,76,69,206,83,84,82,164,86,65,204
3110 DATA65,83,195,67,72,82,164,78,69,70,84,164,82,73,71,72,84,164,77,73,68,164
3120 DATA71,207,73,79,79,212,86,69,65,84,197,69,82,65,83,197,70,77,65,212,67
3130 DATA79,80,217,82,69,78,65,77,197,70,76,65,83,200,68,78,79,65,196,68,83,65
3140 DATA86,197,68,8F,69,210,67,65,212
READY.
```

Per implementare i comandi presenti in questa puntata, è sufficiente digitare e far partire questo programma.

```
C4C8 85 E2     STA $62
C4CA 86 E3     STX $63
C4CC A2 90     LDX #$90
C4CE 38        SEC
C4CF 20 49 BC  JSR $BC49
C4D2 20 DF BD  JSR $BDDF
C4D5 85 FD     STA $FD
C4D7 84 FE     STY $FE
C4D9 A0 FF     LDY #$FF
C4DB A2 00     LDX #$00
C4DD C8        INY
C4DE B1 FD     LDA ($FD),Y
C4E0 D0 FB     BNE $C4DD
C4E2 84 FC     STY $FC
C4E4 38        SEC
C4E5 A9 03     LDA #$03
C4E7 E5 FC     SBC $FC
C4E9 A8        TAY
C4EA F0 00     BEQ $C4F5
C4EC A9 20     LDA #$20
C4EE 9D 3C 03  STA $033C,X
C4F1 E8        INX
C4F2 88        DEY
C4F3 D0 F9     BNE $C4EE
C4F5 B1 FD     LDA ($FD),Y
C4F7 9D 3C 03  STA $033C,X
C4FA E8        INX
C4FB C8        INY
C4FC C9 00     CMP #$00
C4FE D0 F5     BNE $C4F5
C500 CA        DEX
C501 A9 20     LDA #$20
C503 9D 3C 03  STA $033C,X
C506 E8        INX
C507 A9 00     LDA #$00
C509 9D 3C 03  STA $033C,X
C50C A9 3C     LDA #$3C
C50E A0 03     LDY #$03
C510 4C 1E AB  JMP $AB1E
```

Listato 13 - Questa routine è invocata dal comando CAT per stampare il numero dei blocchi occupati o liberi.

moria. EXE per caricare e far partire insieme un programma; RANGE per conoscere l'inizio e la fine di dove verrà locato nella memoria un programma (di qualsiasi genere) mantenuto su un dischetto.

Per le stampanti MPS-801 e MPS-802: PRON e PROFF per attivare o disattivare l'output su carta invece che su video; LPRINT per stampare su carta direttamente; HCOPY, per avere una copia su carta del contenuto del video. E molti altri ancora in fase di preparazione.

Per il plotter avremo PLON e PLOFF per output su plotter; MOVE e DRAW per muovere la penna o scrivere tra due coordinate; BLACK, BLUE, RED e GREEN o COLOR [0...3] per cambiare penna; SIZE [10,20,40,80] per scegliere il numero di caratteri per riga; ORSET per definire un punto di origine fittizio per iniziare un disegno ... e tante altre istruzioni, alcune ancora da inventare. La fantasia non mancherà. Arrivederci. MC