
# i trucchi del CP/M

*di Pierluigi Panunzi*

## MBASIC

Come già accennato nella scorsa puntata, la prima riguardante questo argomento, presenteremo una nuova istruzione dell'MBASIC, molto utile e che senza dubbio colma una lacuna esistente nel set di istruzioni.

Diciamo innanzitutto che è **assolutamente indispensabile** effettuare quelle modifiche riportate la scorsa puntata, modifiche atte ad introdurre uno spazio di memoria tra la fine dell'interprete Basic e l'area predisposta per i buffer dei file, e cioè ben prima della zona riservata ai programmi scritti in Basic.

È proprio in questo spazio che andremo ad inserire la routine che esegue la nuova istruzione, nonché quelle che presenteremo nei prossimi numeri (è una promessa).

La nuova istruzione in questione si chiama **DSK(.)** e consente di conoscere quanto spazio è ancora disponibile sul dischetto: ciò può essere molto utile, specie dopo aver digitato un programma in Basic molto lungo, per vedere se potrà essere salvato su disco. Infatti se il dischetto è quasi pieno, il salvataggio **non** potrà avvenire ed il più delle volte (per la famosa Legge di Murphy) perderemo addirittura il nostro bel programma...

Invece con la nuova istruzione potremo vedere in qualsiasi istante lo spazio a disposizione sul dischetto e se non lo riterremo sufficiente (con un po' di pratica si può arrivare a ciò), potremo cambiare il dischetto, eseguire l'**istruzione RESET** (attenzione, ciò è ben diverso dal premere il tasto di Reset!!!), riverificare lo spazio a disposizione ed in caso positivo salvare il programma, che nel frattempo sarà rimasto immutato al suo posto.

Per i meno esperti diciamo che l'istruzione **fre(x)** in questi casi non serve a niente, in quanto dà soltanto lo spazio disponibile *in memoria*.

Ricordando che ci riferiamo alla versione **5.21** dell'MBASIC e che con pochissime modifiche si può adattare alle altre *release*, diamo subito l'elenco delle locazioni da correggere nonché la routine stessa, da inserire nel "buco" di memoria tra gli indirizzi **6000H** e **60FFH.**

È a questo punto che i non esperti si possono fermare, avendo a disposizione una nuova istruzione, per l'appunto la **DSK**, nonché una vecchia, la **DELETE**, che ora si chiama **DEL.**

### Le modifiche da effettuare

Dopo aver eseguito il comando **A>ZSID MBASIC.COM** oppure **A>DDT MBASIC.COM** inseriamo la routine tra gli indirizzi **6000H** e **603DH**, e i seguenti valori (sempre espressi in esadecimale):

| | | | |
|---|---|---|---|
| a) | 01DD | 00 | 00 |
| | 01DE | 00 | 60 |
| b) | 0288 | 4C | CC |
| | 0289 | 45 | AA |
| | 028A | 54 | 53 |
| | 028B | C5 | CB |
| | 028C | AA | 20 |

Usciamo dallo ZSID con **^C** e salviamo la nuova versione "customizzata" dell'MBASIC con un nome qualsiasi: noi abbiamo scelto (in analogia alla nuova opzione /P della scorsa puntata...) il nome **PBASIC.COM.** Per far ciò dovremo impostare

**A>SAVE 96 PBASIC.COM**

in modo da salvare su disco la zona di memoria compresa tra **0100H** e **60FFH.**

Lanciato perciò il "PBASIC", possiamo andare a vedere quanto spazio abbiamo sul disco digitando:

**PRINT DSK(0)**

per il drive A: (che in alcuni sistemi è chiamato "0") oppure **PRINT DSK(1)** per il drive B:.

Analogamente alle altre istruzioni potremo invece porre, ad esempio:

**A=DSK(0)**

per poi elaborare l'informazione così ottenuta.

Il valore ottenuto rappresenta il numero di "Blocchi" liberi (gli "Allocation blocks" del CP/M), che a seconda del computer usato potranno essere da 1 o 2 o addirittura 4 kbyte.

Fin qui, dicevamo, per i non esperti e per coloro ai quali non interessa una barbosa descrizione tecnica...

### I "dettagli tecnici" della nuova istruzione

Prima di analizzare la piccola routine in assembler di cui al listato allegato, occupiamoci di quella mezza dozzina di byte che abbiamo modificato e che non sono altro che la "punta dell'iceberg" di un insieme di informazioni per spiegare le quali non basterebbe un volume...

Comunque cercheremo di sintetizzare (ma non troppo!), pensando tuttavia che queste informazioni ci saranno molto utili per le prossime puntate, appunto per inserire nuove istruzioni.

Innanzitutto bisogna sapere alcune notizie riguardanti l'interprete MBASIC.

A partire dalla locazione **0107H** fino alla locazione **0469H** sono presenti parecchie informazioni di notevole importanza: in particolare abbiamo:

— da **0107H** a **0206H** abbiamo una tabella di indirizzi a due byte ai quali il controllo passa quando viene eseguita una delle tante istruzioni dell'MBASIC ed a partire dai quali vi sono le vere e proprie routine che eseguono una certa determinata funzione.

— da **0207H** a **023AH** è presente un'altra tabella di 26 puntatori a due byte che "puntano" a zone consecutive di memoria dove si possono trovare le parole chiave dell'MBASIC, disposte in ordine "quasi" alfabetico e seguite ognuna dal rispettivo "token": tanto per fare un esempio il **nono** puntatore, che si riferisce alla lettera "I" come iniziale di istruzioni MBASIC, punta alla zona dove sono memorizzate le parole chiave inizianti per "I" e cioè INPUT, INSTR, INT, INP, eccetera, ognuna "troncata" dell'iniziale (per risparmiare lo spazio!) ed avente di seguito il valore del token corrispondente.

È così che INPUT è codificata con

| | | | | |
|---|---|---|---|---|
| 4E | 50 | 55 | D4 | 85 |
| N | P | U | T | |

dove la "T" finale è contraddistinta dal bit più significativo posto ad 1 (rispetto al valore ASCII che era 54, sempre in esadecimale) e dove l'"85" è il byte (token) con cui l'istruzione INPUT è memorizzata in un programma.

— Da **023BH** a **0469H** è presente per l'appunto tale zona dove si trovano tutti questi nomi troncati dell'iniziale e con il proprio token alla fine.

Purtroppo però non vi è una corrispondenza biunivoca tra gli indirizzi delle routine e i nomi posti in ordine pseudo-alfabetico, ma tale corrispondenza è ben più cervellotica.

Per comprenderla dobbiamo in pratica costruire una tabella di corrispondenza token-istruzioni: ci accorgiamo così che i "comandi" (quali ad esempio RUN, LIST, IF ecc.) hanno token i cui valori sono maggiori di 81H, mentre le altre funzioni han-

no valori alcuni maggiori di CEH ed altri compresi tra 01H e 34H (si tratta delle SIN, LEFT$, VARPTR, le istruzioni aritmetiche e logiche ecc.).

Ora la corrispondenza si ha tra gli indirizzi di partenza delle routine ed i rispettivi token...

Detto così suona maledettamente sinistro, mentre in realtà vuol dire che l'indirizzo posto a **0107H** si riferisce al token **81H** e cioè all'istruzione END, l'indirizzo posto a **0109H** si riferisce al token **82H** corrispondente invece a FOR e così via.

Al termine dei comandi, inizia lo stesso discorso per le istruzioni vere e proprie: ecco che ad esempio a **019FH** c'è l'indirizzo che si riferisce al token **01H** corrispondente all'istruzione LEFT$.

Semplice no?! A parte gli scherzi, una volta capito il meccanismo non c'è più nulla da eccepire...!

Tutto questo discorso più o meno contorto (la colpa è dei progettisti della premiata Microsoft...) per arrivare finalmente alla nuova istruzione.

Vediamo perciò quale è stato il ragionamento che abbiamo fatto per poterla inserire nell'ambito delle altre già esistenti.

Dal momento che è assolutamente **vietato** oltrepassare i limiti di memoria segnalati in precedenza per le varie aree, pena un brutale inchiodamento del sistema, bisogna andare a vedere se esistono delle istruzioni aventi un nome lungo almeno sei lettere per poter creare **due** istruzioni di tre lettere l'una: dato che volevamo creare la DSK abbiamo cercato tra le istruzioni inizianti per "D". Abbiamo così trovato la DELETE, la quale può diventare tranquillamente "DEL" per lasciar posto alla nuova.

Ecco che si potranno utilizzare le istruzioni "esose" tipo STRING$, RESUME, COMMON, ecc., da sei lettere, per non parlare dell'orribile (ma ora meravigliosa!) RANDOMIZE che con le sue nove lettere permette l'introduzione di ben due nuove istruzioni le quali (se ci siamo spiegati bene...) dovranno tutte iniziare per "R" ed essere lunghe tre lettere.

Ma non precorriamo i tempi...

Trovata dunque l'istruzione DELETE, che era presente in memoria come sequenza di byte:

| 45 | 4C | 45 | 54 | C5 | AA |
|---|---|---|---|---|---|
| E | L | E | T | E | token |

per lasciar posto alla DSK (di token pari a 20H) dovrà diventare

| 45 | CC | AA | 53 | CB | 20 |
|---|---|---|---|---|---|
| E | L | token | S | K | token |

dove, ricordiamo, "EL" e "SK" sono ciò che rimane di DEL e di DSK quando eliminiamo l'iniziale e dove il token 20H è stato scelto tra quelli liberi.

Corrispondentemente al token così prescelto si è trovato l'indirizzo **01DDH** nel quale si è posto l'indirizzo di partenza della routine (**6000H**).

## La routine in assembler

Fatte queste doverose premesse, andiamo ora ad affrontare l'analisi, questa volta sommaria, della routine in linguaggio macchina, che sfrutterà alla perfezione alcune informazioni fornite dal modulo BDOS, parte principale del CP/M.

In particolare, dopo aver selezionato il disco in base al valore dell'argomento dell'istruzione DSK, si ricava l'indirizzo del DPB (Disk Parameter Block) e cioè di una tabella in cui sono fornite alcune interessanti informazioni riguardanti il dischetto in esame.

In particolare il sesto e settimo byte della tabella ci forniscono il valore del massimo numero di "Blocchi" effettivamente disponibili nel disco in esame.

Successivamente si ricava l'indirizzo della zona di memoria in cui è codificata l'utilizzazione o meno di un determinato "blocco": ogni blocco occupato è rappresentato da un bit posto ad "1" in tale zona di memoria.

Se per esempio il numero massimo di blocchi è 20 allora saranno riservati 20 bit consecutivi di 3 byte (8 bit per ogni byte!). Basterà contare quanti "zeri" vi sono tra questi byte ed il gioco è fatto!

Bisogna però analizzare nel nostro caso **solo** i 20 bit dovuti e **non** tutti e 24 i bit di 3 byte, altrimenti otterremmo un valore maggiore di quello effettivo.

Fatto questo conto, il valore ottenuto è posto in HL e si effettua un salto alla routine posta a **29CDH**, con il quale si ha il corretto ritorno all'MBASIC.

Complicatuccio, eh?!

A conclusione di questa bella chiacchierata, sono doverose due considerazioni: la prima è che il programma presentato è **assolutamente inedito** (se non andiamo errati, una funzione come la DSK non è presente in sistemi dotati di MBASIC), mentre la seconda considerazione è che questa routine è perfettamente adattabile a sistemi dotati sia di dischetti da 5" che di dischi da 8" sia infine di hard disk!

Infatti la routine può contare fino a 65535 blocchi ed anche se fossero ognuno da 1kbyte, potrebbe "contabilizzare" un sistema hard disk della bellezza di 64 MByte!

L'unico problema semmai è dato dal fatto che altre *release* dell'MBASIC richiederanno due valori diversi per gli indirizzi posti nella prima e nell'ultima istruzione del programma, ma con un minimo di pratica si può sopperire a tale inconveniente. Altra "portatilità" ad altri sistemi (dimenticavamo che il "nostro" è un Osborne I) si ha nel fatto che si possono indirizzare fino alle 16 unità a dischi consentite dal CP/M.

Nel nostro caso (due sole unità a dischi, 0 ed 1) ha senso indirizzare solo l'unità A: (con 0) e l'unità B: (con 1).

In generale l'**AND OFH** posto come seconda istruzione della routine previene l'utilizzazione di valori maggiori di 15.

È evidente però che se in un sistema dotato di due unità a dischi si richiede la DSK(5) allora non ci si deve lamentare se si esce dall'MBASIC per ottenere un perentorio

**BDOS error on F:Select**!

In questo caso infatti abbiamo tentato di riferirci ad un inesistente disco F:.

A risentirci dunque alla prossima puntata con altre istruzioni!! MC

**Dal prossimo numero ci sarà anche il**

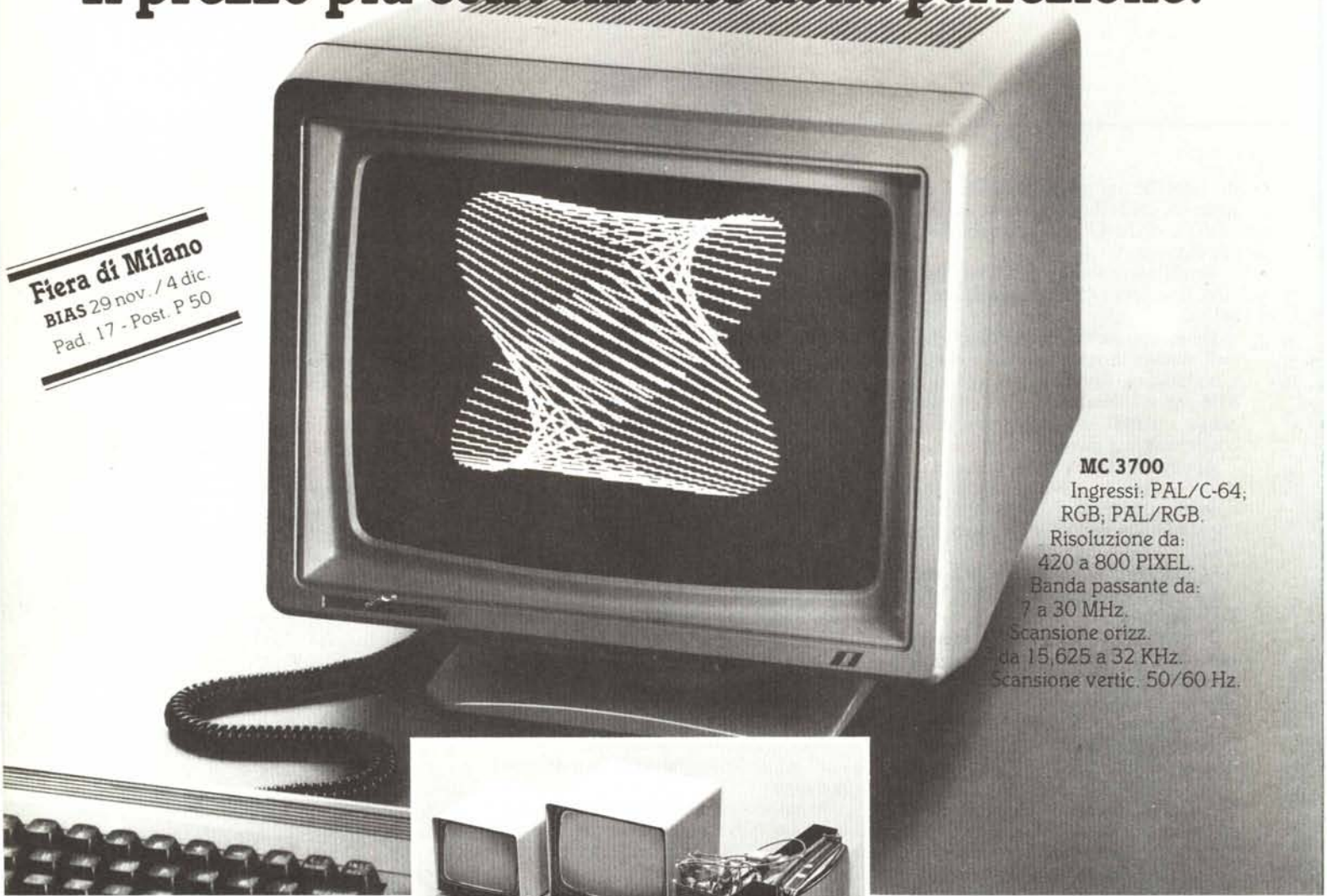
# SOFTWARE MBASIC

Inviateci i vostri programmi
(e routine); come consueto,
quelli pubblicati saranno ricompensati.
Se inviate il dischetto,
non dimenticate di indicare
su quale macchina può essere letto;
in ogni caso,
allegate sempre le spiegazioni
e quando possibile il listato.
Al prossimo mese!

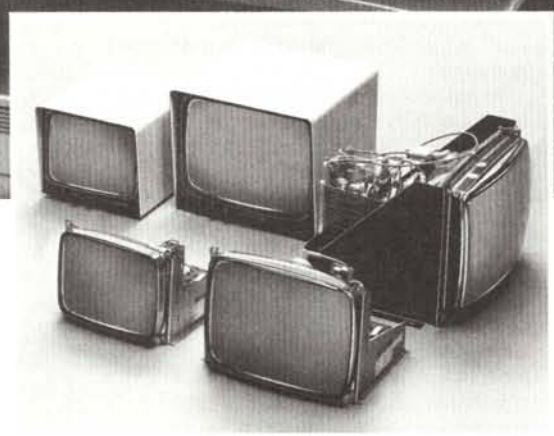
# Monitor Cabel.

## Il prezzo più conveniente della perfezione.

**Fiera di Milano**
**BIAS** 29 nov. / 4 dic.
Pad. 17 - Post. P 50



**MC 3700**
Ingressi: PAL/C-64; RGB; PAL/RGB.
Risoluzione da: 420 a 800 PIXEL.
Banda passante da: 7 a 30 MHz.
Scansione orizz. da 15,625 a 32 KHz.
Scansione vertic. 50/60 Hz.



Se per il vostro home-personal computer utilizzate lo schermo del televisore, riflettete. Con meno di quello che pensate potete avere un monitor Cabel. La nuova serie MC 3700 unisce al raffinato design caratteristiche di assoluta avanguardia: basso consumo, alta risoluzione, affidabilità, video orientabile, comandi frontali e non sul retro.

Aggiungiamo che la serie MC 3700 può collegarsi con tutti i personal e home computers e funzionare con segnali provenienti da telecamere, videoregistratori e sintonizzatori TV.

Scegliere un Cabel, anche per applicazioni speciali, significa scegliere monitors monocromatici e a colori apprezzati dal mercato professionale di tutt'Europa.

CONCESSIONARI ED ASSISTENZA TECNICA

**MILANO E PROVINCIA**
- BRESCIANI AMEDEO
  Via A. Stoppani, 34 - 20128 Milano
  Tel. 02/2043459
- TECHNEX s.r.l.
  Via Teocrito, 46 - 20128 Milano
  Tel. 02/2575315

**EMILIA ROMAGNA - MARCHE**
- ONDAELLE s.n.c.
  Via Faccini, 4 - 40128 Bologna
  Tel. 051/373513 - 359649

CABEL® electronic
24035 CURNO (Bergamo) - Tel. 035/612103
Telex 316370 CABEL I

**LIGURIA**
- R. e R. ELECTRONICS s.r.l.
  Via F.lli Canepa, 94
  16010 Serra Riccò - GE
  Tel. 010/750729 - 750866
  Telex 216530 COGE I

**TOSCANA - UMBRIA**
- FGM ELETTRONICA s.r.l.
  Via Silvio Pellico, 9/11
  50121 Firenze
  Tel. 055/245371
  Telex 573332 FGM I

**LAZIO**
- HI-REL s.r.l.
  Via Amatrice, 15
  00199 Roma
  Tel. 06/8395671 - 8395581
  Telex 614676
- GIU.PA.R
  di G. Pastorelli e figli
  Via dei Conciatori, 36
  00154 Roma
  Tel. 06/5758734

**CAMPANIA - PUGLIA - BASILICATA - CALABRIA**
- C.F. ELETTRONICA PROFESSIONALE
  Corso V. Emanuele, 54
  80122 Napoli
  Tel. 081/683728

**SICILIA**
- RICCOBONO EMANUELE
  Via Onorato, 46
  90139 Palermo
  Tel. 091/331464 - 325813

TARGET