Il Basic Applesoft, seppure molto esteso, è piuttosto lontano dall'essere completo. Molte istruzioni di grafica, di gestione dei dati o dei caratteri sarebbero desiderabili. Non ci dobbiamo però scoraggiare, con qualche semplice artificio, e un po' di conoscenza del linguaggio macchina, è possibile agganciare al Basic Applesoft un gran numero di routine, sia scritte da noi che scopiazzate in giro, e creare un mini interprete in grado di eseguirle direttamente dall'ambiente Basic. Il tutto senza perdere neppure un byte della amata area variabili dell'Applesoft.

Il modo di gestire queste nuove istruzioni si deduce facilmente dal programma di questa puntata: 12 nuove istruzioni vengono aggiunte al Basic tramite il comando & e risiedono nella RAM della Language Card (occupando appena due dei dodici Kappa disponibili). A causa di ciò non è purtroppo possibile utilizzare questa estensione sotto ProDOS, a meno di qualche piccola modifica (si deve accorciare un po' il programma e metterlo nella pagina di RAM non utilizzata dal ProDOS).

Visto che il codice oggetto da inserire è piuttosto lungo è a disposizione dei lettori il disco con il programma bell'e pronto.

G-Basic II

di Riccardo Ciacomazzi - Lido (VE)

Il G-BASIC II è un estensione al Basic Applesoft e si collega a quest'ultimo grazie al comando <&> al quale è agganciato l'interprete G-BASIC II.

Per non occupare parte della memoria RAM destinata ai programmi Basic, il G-BASIC II risiede nella Language Card ed è collegato al comando <&> tramite una serie di routine che risiedono in pagina $300.

Nelle seguenti istruzioni i nomi delle variabili servono solo da esempio in quanto per qualunque istruzione l'interprete riconosce qualunque variabile e la sua posizione in memoria nel formato Basic Applesoft.

& FLIP1

Questa istruzione permette la visualizzazione della prima pagina grafica (HGR) senza cancellarla in modo solo grafici e non seleziona il modo scrittura il quale rimane per la pagina scelta in precedenza (con un semplice POKE 230, 32 si seleziona anche il modo scrittura in HGR).

& FLIP2

È identica alla precedente tranne per il fatto che si riferisce alla seconda pagina grafica (in questo caso serve un POKE 230, 64 per scrivere in HGR2).

& REVERSE

Esegue il reverse video della pagina grafica selezionata in modo scrittura.

& CHAR (CH, ST$)

Grazie a questa istruzione è possibile definire la forma di un set di caratteri utilizzabile nelle due pagine grafiche.

Nella sintassi dell'istruzione, CH può essere una variabile o un'espressione numerica il cui valore può variare tra 0 e 255, mentre ST$ è una stringa composta da 16 caratteri esadecimali e può essere rappresentata da una variabile di stringa (come nell'esempio) oppure dalla stessa stringa posta tra virgolette (es: & CHAR (CH, "0123456789ABCDEF")). In questa istruzione CH rappresenta il codice del carattere, mentre ST$ rappresenta la forma del carattere codificata in 8 byte i quali prenderanno il valore scritto nella stringa (i primi due caratteri si riferiscono al primo byte, il terzo e il quarto carattere al secondo by-

```
0300-  AD 81 C0   LDA   $C081      034C-  91 00      STA   ($00),Y
0303-  AD 81 C0   LDA   $C081      034E-  E6 00      INC   $00
0306-  60         RTS              0350-  A5 00      LDA   $00
0307-  AD 83 C0   LDA   $C083      0352-  C9 FF      CMP   #$FF
030A-  20 00 D0   JSR   $D000      0354-  D0 F4      BNE   $034A
030D-  AD 81 C0   LDA   $C081      0356-  E6 01      INC   $01
0310-  60         RTS              0358-  A5 01      LDA   $01
0311-  AD 81 C0   LDA   $C081      035A-  C9 00      CMP   #$00
0314-  4C C9 DE   JMP   $DEC9      035C-  D0 EC      BNE   $034A
0317-  AD 81 C0   LDA   $C081      035E-  60         RTS
031A-  20 95 D9   JSR   $D995      035F-  00         BRK
031D-  60         RTS              0360-  AD 83 C0   LDA   $C083
031E-  85 09      STA   $09        0363-  A9 00      LDA   #$00
0320-  A5 19      LDA   $19        0365-  85 00      STA   $00
0322-  8D 30 03   STA   $0330      0367-  85 02      STA   $02
0325-  A5 1A      LDA   $1A        0369-  A9 E2      LDA   #$E2
0327-  8D 31 03   STA   $0331      036B-  85 01      STA   $01
032A-  AD 81 C0   LDA   $C081      036D-  A9 40      LDA   #$40
032D-  A5 09      LDA   $09        036F-  85 03      STA   $03
032F-  20 D4 E5   JSR   $E5D4      0371-  A0 00      LDY   #$00
0332-  85 09      STA   $09        0373-  B1 00      LDA   ($00),Y
0334-  AD 83 C0   LDA   $C083      0375-  91 02      STA   ($02),Y
0337-  A5 09      LDA   $09        0377-  E6 00      INC   $00
0339-  60         RTS              0379-  E6 02      INC   $02
033A-  AD 81 C0   LDA   $C081      037B-  D0 F6      BNE   $0373
033D-  AD 81 C0   LDA   $C081      037D-  E6 01      INC   $01
0340-  A9 00      LDA   #$00       037F-  E6 03      INC   $03
0342-  85 00      STA   $00        0381-  A5 01      LDA   $01
0344-  A9 F0      LDA   #$F0       0383-  C9 EA      CMP   #$EA
0346-  85 01      STA   $01        0385-  D0 EC      BNE   $0373
0348-  A0 00      LDY   #$00       0387-  AD 81 C0   LDA   $C081
034A-  B1 00      LDA   ($00),Y    038A-  60         RTS
```

Questa parte dell'interprete risiede in pagina tre perchè deve attivare e disattivare la RAM della Language Card dove si trovano le routine aggiunte.

te, ecc...), per questo motivo bisogna far attenzione che la stringa sia composta da 16 caratteri altrimenti l'istruzione tornerà con un ?SYNTAX ERROR. (P.S. L'istruzione è molto simile alla rispettiva istruzione del fu TI 99/4A).

& WRITE (CH, X, Y)

Serve a stampare un singolo carattere sulla pagina grafica selezionata in scrittura.

CH rappresenta il codice carattere e sono valide le stesse regole dell'istruzione precedente.

X rappresenta la coordinata orizzontale ed il suo valore può variare tra 0 e 39.

Y rappresenta la coordinata verticale ed il suo valore può variare tra 0 e 184.

X e Y possono essere o variabili intere e reali oppure espressioni numeriche valide per il formato Basic Applesoft.

& SCRIBE (X, Y, ST$)

Stampa una stringa nella pagina grafica selezionata in scrittura alle coordinate rappresentate da X e Y.

Per le coordinate X e Y sono valide le stesse regole dell'istruzione precedente.

ST$ rappresenta la stringa da stampare e può essere solo una variabile di stringa e non la stringa stessa messa tra virgolette e neanche uno dei manipolatori di stringa ammessi da Applesoft (RIGHT$, LEFT$, MID$, ecc...).

& CH$ (CH, ST$)

Restituisce sulla stringa ST$ una serie di valori esadecimali che rappresentano la forma del carattere con codice CH.

CH deve rispettare le stesse regole dell'istruzione & CHAR, mentre ST$ deve essere una variabile di stringa nella quale si troverà la forma del carattere scelto.

& KEY (K%, S%)

Legge la tastiera e restituisce su K% il codice ASCII del tasto battuto, mentre S% vale 1 se è stato battuto un tasto e 0 se non è stato battuto. K% e S% devono essere due variabili numeriche intere.

(P.S. L'istruzione è simile alla CALL KEY del TI 99/4A).

& SOUND (DU, FR)

Emette una nota in base ai valori delle due variabili.

DU rappresenta la durata (0-255) e FR la frequenza (0-255) purtroppo non in Hertz.

DU e FR possono essere variabili numeriche reali e intere oppure espressioni numeriche.

& ESA (ND, ST$)

Carica nella stringa ST$ il valore esadecimale e binario del numero decimale rappresentato da ND.

ND deve essere un valore tra 0 e 255 e può essere una variabile o un'espressione numerica.

ST$ deve essere una variabile di stringa nella quale dopo aver eseguito l'istruzione troveremo il valore esadecimale nei primi due caratteri e quello binario negli altri 8.

& DEC (ST$, ND%)

Questa istruzione carica nella variabile intera ND% il valore decimale della stringa esadecimale. ST$.

ST$ deve essere una variabile di stringa composta da due caratteri esadecimali.

ND% deve essere una variabile numerica intera nella quale troveremo dopo aver eseguito l'istruzione un valore tra 0 e 255.

```
D000-  A9 FF      LDA  #$FF
D002-  85 08      STA  $08
D004-  85 06      STA  $06
D006-  A9 E9      LDA  #$E9
D008-  85 07      STA  $07
D00A-  E6 08      INC  $08
D00C-  A5 08      LDA  $08
D00E-  0A         ASL
D00F-  A8         TAY
D010-  C8         INY
D011-  B9 00 EE   LDA  $EE00,Y
D014-  C9 00      CMP  #$00
D016-  D0 03      BNE  $D01B
D018-  4C 11 03   JMP  $0311
D01B-  20 34 D0   JSR  $D034
D01E-  90 EA      BCC  $D00A
D020-  A5 08      LDA  $08
D022-  0A         ASL
D023-  A8         TAY
D024-  B9 01 EE   LDA  $EE01,Y
D027-  8D 32 D0   STA  $D032
D02A-  B9 00 EE   LDA  $EE00,Y
D02D-  8D 31 D0   STA  $D031
D030-  4C CF D2   JMP  $D2CF
D033-  60         RTS
D034-  A0 FF      LDY  #$FF
D036-  A2 00      LDX  #$00
D038-  C8         INY
D039-  E6 06      INC  $06
D03B-  D0 02      BNE  $D03F
D03D-  E6 07      INC  $07
D03F-  A1 06      LDA  ($06,X)
D041-  C9 00      CMP  #$00
D043-  F0 06      BEQ  $D04B
D045-  D1 B8      CMP  ($B8),Y
D047-  D0 04      BNE  $D04D
D049-  F0 ED      BEQ  $D038
D04B-  38         SEC
D04C-  60         RTS
D04D-  E6 06      INC  $06
D04F-  D0 02      BNE  $D053
D051-  E6 07      INC  $07
D053-  A1 06      LDA  ($06,X)
D055-  C9 00      CMP  #$00
D057-  F0 03      BEQ  $D05C
D059-  4C 4D D0   JMP  $D04D
D05C-  18         CLC
D05D-  60         RTS
```

Seconda parte dell'interprete, una volta attivata la RAM ausiliaria il controllo del Basic passa a questa routine che si occupa di riconoscere ed eseguire il comando incontrato dopo la &.

& BIN (ST$, ND%)

Carica nella variabile intera ND% il valore decimale della stringa binaria ST$.

ST$ deve essere una variabile di stringa composta da 8 caratteri binari (0 e 1).

Per ND% valgono le stesse regole dell'istruzione precedente.

Funzionamento del G-Basic II

L'interprete G-Basic II come ho già detto, risiede nella Language Card la quale non può avere un collegamento diretto con la ROM del sistema nella quale risiede l'Applesoft, questo collegamento avviene tramite una serie di routine che risiedono a partire dalla locazione $300.

Le routine hanno i seguenti scopi:

> adr. $300

Seleziona la Language Card in modo scrittura e viene richiamata per caricare in memoria l'interprete o un nuovo set di caratteri.

> adr. $307

Commuta la Language Card in modo lettura-scrittura; questa routine viene richiamata dal comando <&>1219 ogni volta che deve venir eseguita un'istruzione G-BASIC II.

> adr. $311

Seleziona la ROM in modo lettura ed esegue la stampa del messaggio "?SYNTAX ERROR".

> adr. $317

Seleziona la ROM in lettura ed esegue la routine DATA del Basic Applesoft, la quale legge i vari caratteri di una linea Basic finché non trova un carattere <:> oppure un <RETURN>.

> adr. $31E

Seleziona la ROM in letture ed esegue la routine il cui indirizzo è posto in $19 e $1A, dopodiché seleziona la Language in lettura-scrittura e torna all'indirizzo di chiamata. L'Accumulatore e i registri X e Y rimangono invariati.

> adr. $33A

Copia il MONITOR dell'Applesoft nella Language Card.

```
D100- AD 50 C0 AD 54 C0 AD 57
D108- C0 AD 52 C0 4C 17 03 AD
D110- 50 C0 AD 55 C0 AD 57 C0
D118- AD 52 C0 4C 17 03 A5 E6
D120- 85 1C 18 69 20 85 1D A9
D128- 00 85 1B A8 A9 FF 51 1B
D130- 91 1B E6 1B A5 1B C9 00
D138- D0 F2 E6 1C A5 1C C5 1D
D140- D0 EA 4C 17 03 A2 00 20
D148- B1 00 E8 E0 06 D0 F8 A9
D150- 4F 85 19 A9 E7 85 1A 20
D158- 1E 03 86 EB 20 B1 00 20
D160- 1E 03 86 1E 20 B1 00 20
D168- 1E 03 86 1F 20 72 D1 4C
D170- 17 03 A9 E2 85 D7 A9 00
D178- 85 D6 A6 D7 A4 EB B1 D6
D180- 85 EC 20 9F D1 A5 1F A6
D188- CF A4 CE 20 11 F4 A5 EC
D190- A4 1E 91 26 E6 1F E6 D7
D198- A5 D7 C9 EA D0 DC 60 A5
D1A0- 1E 85 CE A9 00 85 CF 06
D1A8- CE 26 CF 06 CE 26 CF 06
D1B0- CE 26 CF A5 CE 38 E5 1E
D1B8- B0 02 C6 CF 85 CE 60 A2
D1C0- 00 20 B1 00 E8 E0 05 D0
D1C8- F8 A9 4F 85 19 A9 E7 85
D1D0- 1A 20 1E 03 86 ED 20 B1
D1D8- 00 A0 00 B1 B8 C9 22 F0
D1E0- 23 A9 E3 85 19 A9 DF 85
D1E8- 1A 20 1E 03 85 EE 84 EF
D1F0- A0 00 B1 EE 85 F9 C8 B1
D1F8- EE AA C8 B1 EE 86 EE 85
D200- EF 4C 1A D2 20 B1 00 A5
D208- B8 85 EE A5 B9 85 EF A0
D210- FF C8 B1 B8 C9 22 D0 F9
D218- 84 F9 A5 F9 38 E9 10 B0
D220- 03 4C 11 03 A9 E2 85 FE
D228- A9 00 85 FD A0 00 B1 EE
D230- 85 F9 38 E9 30 85 FA 29
D238- F0 C9 00 F0 10 A5 F9 38
D240- E9 37 85 FA 29 F0 C9 00
D248- F0 03 4C 11 03 84 FB 98
D250- 6A B0 11 06 FA 06 FA 06
D258- FA 06 FA A5 FA 85 FC C8
D260- 4C 2E D2 00 A5 FA 05 FC
D268- 85 FA A4 ED A5 FA 91 FD
D270- E6 FE A4 FB C8 C0 10 D0
D278- B5 4C 17 03 A2 00 20 B1
D280- 00 E8 E0 04 D0 F8 A9 E3
D288- 85 19 A9 DF 85 1A 20 1E
D290- 03 85 EE 84 EF 20 B1 00
D298- 20 1E 03 85 F9 84 FA 20
D2A0- C4 D2 A5 FF 29 80 C9 00
D2A8- F0 0F A5 FF 29 7F A0 01
D2B0- 91 EE A9 01 91 F9 4C 17
D2B8- 03 A9 00 A0 01 91 EE 91
D2C0- F9 4C 17 03 AD 00 C0 85
D2C8- FF A9 00 8D 10 C0 60 A2
D2D0- 00 20 B1 00 E8 E0 06 D0
D2D8- F8 A9 4F 85 19 A9 E7 85
D2E0- 1A 20 1E 03 86 EB 20 B1
D2E8- 00 20 1E 03 86 EC A4 EB
D2F0- AD 30 C0 A6 EC CA E0 00
D2F8- D0 FB 88 C0 00 D0 F1 4C
D300- 17 03 A2 00 20 B1 00 E8
D308- E0 04 D0 F8 A9 4F 85 19
D310- A9 E7 85 1A 20 1E 03 86
D318- ED 20 B1 00 A9 E3 85 19
D320- A9 DF 85 1A 20 1E 03 85
D328- 1B 84 1C A0 00 A9 10 91
D330- 1B C8 A9 BA 91 1B C8 A9
D338- 03 91 1B A9 BA 85 EE A9
D340- 03 85 EF A9 00 85 06 85
D348- F9 A9 E2 85 07 A4 ED B1
D350- 06 85 08 A9 F0 25 08 18
D358- 6A 6A 6A 6A AA BD 7E D3
D360- A4 F9 91 EE A9 0F 25 08
D368- AA BD 7E D3 C8 91 EE E6
D370- F9 E6 F9 E6 07 A5 07 C9
D378- EA D0 D2 4C 03 D4 30 31
D380- 32 33 34 35 36 37 38 39
D388- 41 42 43 44 45 46 A2 00
D390- 20 B1 00 E8 E0 04 D0 F8
D398- A9 4F 85 19 A9 E7 85 1A
D3A0- 20 1E 03 86 ED 20 B1 00
D3A8- A9 E3 85 19 A9 DF 85 1A
D3B0- 20 1E 03 85 1B 84 1C A0
D3B8- 00 A9 0A 91 1B C8 A9 CC
D3C0- 91 1B C8 A9 03 91 1B A9
D3C8- CC 85 EE A9 03 85 EF A9
D3D0- F0 25 ED 18 6A 6A 6A 6A
D3D8- AA BD 7E D3 A0 00 91 EE
D3E0- A9 0F 25 ED AA BD 7E D3
D3E8- C8 91 EE A9 CE 85 EE A0
D3F0- 00 06 ED A2 00 90 02 A2
D3F8- 01 BD 7E D3 91 EE C8 C0
D400- 08 D0 EE A0 00 A9 52 85
D408- 19 A9 E4 85 1A 20 1E 03
D410- A5 1B 85 AB A5 1C 85 AC
D418- A9 D4 85 19 A9 E5 85 1A
D420- A5 71 85 EE A5 72 85 EF
D428- 20 1E 03 A0 01 A5 EE 91
D430- 1B C8 A5 EF 91 1B 4C 17
D438- 03 A2 00 20 B1 00 E8 E0
D440- 04 D0 F8 A9 E3 85 19 A9
D448- DF 85 1A 20 1E 03 85 EE
D450- 84 EF 20 B1 00 20 1E 03
D458- 85 F9 84 FA A0 00 B1 EE
D460- C9 02 F0 03 4C 11 03 C8
D468- B1 EE AA C8 B1 EE 86 EE
D470- 85 EF A0 00 B1 EE 29 F0
D478- C9 40 D0 0E 38 B1 EE E9
D480- 37 0A 0A 0A 0A 85 ED 4C
D488- 95 D4 B1 EE 38 E9 30 0A
D490- 0A 0A 0A 85 ED C8 B1 EE
D498- 29 F0 C9 40 D0 0C 38 B1
D4A0- EE E9 37 05 ED 85 ED 4C
D4A8- B3 D4 B1 EE 38 E9 30 05
D4B0- ED 85 ED A0 00 A9 00 91
D4B8- F9 C8 A5 ED 91 F9 4C 17
D4C0- 03 A2 00 20 B1 00 E8 E0
D4C8- 04 D0 F8 A9 E3 85 19 A9
D4D0- DF 85 1A 20 1E 03 85 EE
D4D8- 84 EF 20 B1 00 20 1E 03
D4E0- 85 F9 84 FA A0 00 B1 EE
D4E8- C9 08 F0 03 4C 11 03 C8
D4F0- B1 EE AA C8 B1 EE 86 EE
D4F8- 85 EF A0 00 B1 EE 29 0F
D500- 18 C9 00 F0 01 38 26 ED
D508- C8 C0 08 D0 EF A0 00 A9
D510- 00 91 F9 C8 A5 ED 91 F9
D518- 4C 17 03 A2 00 20 B1 00
D520- E8 E0 07 D0 F8 A9 4F 85
D528- 19 A9 E7 85 1A 20 1E 03
D530- 86 1E 20 B1 00 20 1E 03
D538- 86 FA 20 B1 00 A9 E3 85
D540- 19 A9 DF 85 1A 20 1E 03
D548- 85 EE 84 EF A0 00 B1 EE
D550- 85 F9 C8 B1 EE AA C8 B1
D558- EE 85 EF 86 EE A9 00 85
D560- ED A4 ED A5 FA 85 1F B1
D568- EE 85 EB 20 72 D1 E6 1E
D570- A5 1E C9 28 D0 13 A9 00
D578- 85 1E A5 FA 18 69 08 85
D580- FA 38 E9 BA 90 03 4C 94
D588- D5 E6 ED A5 ED C5 F9 D0
D590- D0 4C 17 03 A9 00 85 EE
D598- A9 08 85 EF A0 00 A2 00
D5A0- A5 EE 20 11 F4 A5 26 85
D5A8- EC A5 27 85 ED A0 00 A2
D5B0- 00 A5 EF 20 11 F4 A0 00
D5B8- B1 26 91 EC C8 C0 28 D0
D5C0- F7 E6 EE E6 EF A5 EF C9
D5C8- C0 D0 D1 A9 00 85 1E A9
D5D0- 20 85 EB A9 B8 85 1F 20
D5D8- 72 D1 E6 1E A5 1E C9 28
D5E0- D0 F1 4C 17 03
```

Dump delle routine che eseguono le varie istruzioni.

> adr. $360

Copia il set di caratteri nelle locazioni a partire da $4000 in modo da poterlo registrare su disco da programma.

> adr. $3F5

Puntatore per il comando <&>, deve puntare alla locazione $307.

L'interprete G-Basic risiede in memoria a partire dall'indirizzo $D000 e termina all'indirizzo $D05D, poi fino alla fine della pagina la memoria è libera per eventuali modifiche.

Le routine per l'esecuzione delle varie istruzioni cominciano dalla locazione $D100 e nell'ordine sono:

$D100 : FLIP1
$D10F : FLIP2
$D11E : REVERSE
$D145 : WRITE
$D1BF : CHAR
$D27C : KEY
$D2CF : SOUND
$D302 : CH$
$D38E : ESA
$D439 : DEC
$D4C1 : BIN
$D51B : SCRIBE

La tabella delle istruzioni risiede a partire dall'indirizzo $EA00 ed ogni istruzione è divisa dalla seguente con un <$00>. La tabella degli indirizzi risiede a partire dall'indirizzo $EE00 e deve finire con una coppia di <$00> che ne indicano la fine.

Costruzione di una nuova istruzione

Per creare delle nuove istruzioni bisogna tener presenti le seguenti regole:

1 - Se nell'istruzione bisogna utilizzare le routine grafiche del Basic Applesoft queste sono presenti nella Language perché vengono copiate assieme al MONITOR.

2 - Se la routine occorrente non è presente nella Language per richiamarla bisogna caricarne l'indirizzo nelle locazioni $19 (byte meno significativo) e $1A (byte più significativo) dopodiché si richiama la routine all'indirizzo $31E, la quale esegue la routine richiesta.

3 - Se per far spazio a nuove istruzioni si vuole cancellare il MONITOR, bisogna far attenzione a non cancellare anche le routine grafiche perché vengono utilizzate dalle istruzioni già presenti per calcolare la posizione in cui stampare i caratteri.

4 - Ricordarsi che il set di caratteri comincia dalla locazione $E200 e quindi non bisogna scriverci sopra.

5 - Infine ricordarsi di scriverne il

```
EE00- 00 D1 0F D1 1E D1 45 D1
EE08- BF D1 7C D2 CF D2 02 D3
EE10- 8E D3 39 D4 C1 D4 1B D5
EE18- 00 00 00

EA00- 46 4C 49 50 31 00 46 4C  FLIP1 FL
EA08- 49 50 32 00 52 45 56 45  IP2 REVE
EA10- 52 53 45 00 57 52 49 54  RSE WRIT
EA18- 45 28 00 43 48 41 52 28  E( CHAR(
EA20- 00 4B 45 59 28 00 53 4F   KEY( SO
EA28- 55 4E 44 28 00 43 48 24  UND( CH$
EA30- 28 00 45 53 41 28 00 44  ( ESA( D
EA38- 45 43 28 00 42 49 4E 28  EC( BIN(
EA40- 00 53 43 52 49 42 45 28  SCRIBE(
EA48- 00 00 00 00 00 00 00 00
```

Elenco dei comandi e dei relativi punti di entrata.

Demo di installazione ed attivazione del G-Basic II. ▶

```
10 D$ =  CHR$ (4): GOSUB 60
20  PRINT D$;"BLOAD CALL-NEW.BASI
    C"
30  CALL 768: CALL 826
40  PRINT D$;"BLOAD NEW.BASIC"
50 A$ = " G-BASIC ][ A C T I V E
     D ": VTAB 9: HTAB 7: INVERSE
     : FOR A = 1 TO 27: PRINT  MID$
     (A$,A,1);: & SOUND(20,80): NEXT
     : NORMAL : HOME : NEW
60  TEXT : HOME
70  PRINT  TAB( 15);"G-BASIC ][":
     PRINT
80  PRINT  TAB( 8);"Extended Basi
    c Applesoft": PRINT
90  PRINT  TAB( 10);"GIACOMAZZI
    SOFTWARE": PRINT
100  PRINT  TAB( 16);"[-1986-]"
110  POKE 34,10: RETURN
```

nome (senza il carattere <&>) subito dopo il nome dell'ultima istruzione dividendolo da questo con un $00, e metterlo anche alla fine del nome appena scritto.

6 - Scrivere l'indirizzo di entrata alla routine dell'istruzione subito dopo l'indirizzo dell'ultima istruzione e ricordarsi di terminare la tabella con due $00.

Per copiare il linguaggio consiglio di scrivere per prima la serie di routine presenti in pagina $300 e registrarle sul disco come segue:

BSAVE CALL-NEW.BASIC, A$300, L$FF <RETURN>

poi dal MONITOR dare il comando:

33AG <RETURN>

il quale copierà il monitor nella Language. Infine digitare:

C083 <RETURN>

e se tutto funziona ci si troverà ad utilizzare il MONITOR inserito nella Language.

Quindi digitare il listato dell'interprete e delle varie istruzioni comprese le due tabelle per i nomi e gli indirizzi.

Una volta terminata la digitazione sempre rimanendo nel MONITOR-Language digitare il seguente comando DOS:

BSAVE NEW.BASIC, A$D000, L$2000 <RETURN>

Quindi ritornare in Basic con il comando:

C081 <RETURN>
ctrl-<C> <RETURN>

Ora digitare il listato in Basic e salvarlo con il comando:

SAVE START /# NEW BASIC <RETURN>

Dando il RUN a quest'ultimo programma partirà il G-BASIC II.

Questo programma è disponibile su disco presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 160.

Apple-posta

Token da Basic?

Cari amici di MCmicrocomputer, vi scrivo per porvi i seguenti quesiti:

1) È possibile tokenizzare una stringa passata con un INPUT tramite le routine dell'interprete Basic ed eventualmente come?

2) Perché non pubblicate una versione dell'Heapsort in linguaggio macchina che sfrutti appieno la memoria negli Apple con 64K e prodos?

3) Penso che per molti sarebbe utile un programmino in linguaggio macchina di ricerca di una sottostringa all'interno di una stringa, voi non ne avete sottomano nessuno?

4) Come si usano le CALL con i compilatori (TASC, EXPEDITER, ecc.)?

5) Sarebbe possibile compilare programmi sotto ProDOS sostituendo i comandi di sistema operativo con delle CALL? Se sì sapete quali sono gli indirizzi dei vari comandi?

Vi ringrazio.

Alberto del Corona, Livorno

1) Si può fare semplicemente da Basic sostituendo alle parole chiave il relativo TOKEN, ma non si può poi inserire nel programma una tale riga. Infatti una volta in fase di RUN il programma Basic non può essere modificato; o più precisamente non si può cambiarne la lunghezza. Per creare un programma Basic automodificante si devono scrivere prima le righe che dovranno contenere le parti automodificanti in questo modo:

1520::::::::::::::::::::::
1530::::::::::::::::::::::

Il numero di riga può essere qualsiasi, mentre il numero di due punti deve essere maggiore o uguale alla istruzione da creare. Eventuali due punti in più non hanno alcun effetto sul RUN. Una volta scritte le routine contenenti righe automodificanti si possono sostituire i puntini con i relativi token (che si trovano sul manuale applesoft) mentre le scritte o i valori costanti sono scritti normalmente in ASCII. Per individuare il punto di partenza delle righe da modificare si possono usare svariati sistemi: si può leggere da Basic il numero di riga (terzo e quarto byte di ciascuna riga) oppure sostituire i primi due punti con un carattere particolare (ad esempio la &) e poi scorrere la memoria di programma (solitamente da 2049 in poi) alla ricerca del carattere speciale. Ricordarsi che ogni modifica alle righe precedenti sposta il punto di inizio delle successive!

Per scrivere a questo punto dentro la riga la nuova istruzione sono sufficienti un po' di POKE.

2) L'Heapsort in linguaggio macchina di Bo Arnklit (MC n. 3) sfrutta già al massimo la memoria, la colpa della limitata area variabile è l'applesoft che non utilizza ancora le espansioni di memoria.

3) Detto fatto.

4) Col TASC non ci sono problemi, a patto di avere le routine in L.M. al posto giusto. L'EXPEDITER non lo

Programma di Prova

```
10  PRINT  CHR$ (4)"BLOAD R.OBJ0"
20  POKE 11,0: POKE 12,3
30  INPUT "STRINGA ORIGINARIA ";A$
35  INPUT "STRINGA DA CERCARE ";B$
40 Z =  USR (0);A$,B$
50  IF Z = 0 THEN  PRINT "LA STRINGA RICERCATA NON E' PRESENTE ": PRINT : GOTO 30
60  PRINT "LA STRINGA DA CERCARE E' PRESENTE A PARTIRE DAL "Z" CARATTERE"
70  PRINT : GOTO 30
```

```
                        2 ************************
0000:                   3 *      POS ROUTINE       *
0000:                   4 *                        *
0000:                   5 *    BY MARCO MERLER     *
0000:                   6 *  FOR MC MICROCOMPUTER  *
                        7 ************************
0000:                   8 *
0000:                   9 *
0000:                  10 *
0000:                  11 * LOCAZIONI USATE
0000:                  12 *
DFE3:                  13 PTRGET  EQU  $DFE3
0083:                  14 VARPNL  EQU  $83
0006:                  15 POINT   EQU  $06
00FA:                  16 LEN1    EQU  $FA
00FB:                  17 LEN2    EQU  $FB
0200:                  18 BUFF1   EQU  $200
0380:                  19 BUFF2   EQU  $380
00B1:                  20 CHRGET  EQU  $B1
0007:                  21 RESULT  EQU  $07
00FE:                  22 XSAVE   EQU  $FE
00FF:                  23 YSAVE   EQU  $FF
00FD:                  24 YSAVE1  EQU  $FD
0000:                  25 *
0000:                  26 * ROUTINE CON ORIGINE IN $300
0000:                  27 *
----- NEXT OBJECT FILE NAME IS POS ROUTINE.OBJ0
0300:                  28         ORG  $300
0300:86 FE             29         STX  XSAVE
0302:84 FF             30         STY  YSAVE
0304:20 B1 00          31         JSR  CHRGET
0307:20 E3 DF          32         JSR  PTRGET
030A:A0 02             33         LDY  #$02
030C:B1 83             34         LDA  (VARPNL),Y
030E:85 07             35         STA  POINT+1
0310:88                36         DEY
0311:B1 83             37         LDA  (VARPNL),Y
0313:85 06             38         STA  POINT
0315:88                39         DEY
0316:B1 83             40         LDA  (VARPNL),Y
0318:85 FA             41         STA  LEN1
031A:A8                42         TAY
031B:88                43         DEY
031C:B1 06             44 NEXT1   LDA  (POINT),Y
031E:99 00 02          45         STA  BUFF1,Y
0321:88                46         DEY
0322:10 F8             47         BPL  NEXT1
0324:20 B1 00          48         JSR  CHRGET
0327:20 E3 DF          49         JSR  PTRGET
032A:A0 02             50         LDY  #$02
032C:B1 83             51         LDA  (VARPNL),Y
032E:85 07             52         STA  POINT+1
0330:88                53         DEY
0331:B1 83             54         LDA  (VARPNL),Y
0333:85 06             55         STA  POINT
0335:88                56         DEY
0336:B1 83             57         LDA  (VARPNL),Y
0338:85 FB             58         STA  LEN2
033A:A8                59         TAY
033B:88                60         DEY
033C:B1 06             61 NEXT2   LDA  (POINT),Y
033E:99 80 03          62         STA  BUFF2,Y
0341:88                63         DEY
0342:10 F8             64         BPL  NEXT2
0344:A0 00             65         LDY  #$00
0346:B9 00 02          66 NEXT3   LDA  BUFF1,Y
0349:CD 80 03          67         CMP  BUFF2
034C:F0 09             68         BEQ  UGUALI
034E:C8                69 NEXT5   INY
034F:C4 FA             70         CPY  LEN1
0351:D0 F3             71         BNE  NEXT3
0353:A0 00             72 NOTFOUND LDY #$00
0355:F0 1F             73         BEQ  END2
0357:84 FD             74 UGUALI  STY  YSAVE1
0359:A2 00             75         LDX  #$00
035B:E8                76 NEXT4   INX
035C:C8                77         INY
035D:E4 FB             78         CPX  LEN2
035F:F0 12             79         BEQ  END
0361:C4 FA             80         CPY  LEN1
0363:F0 EE             81         BEQ  NOTFOUND
0365:B9 00 02          82         LDA  BUFF1,Y
0368:DD 80 03          83         CMP  BUFF2,X
036B:F0 EE             84         BEQ  NEXT4
036D:A4 FD             85         LDY  YSAVE1
036F:D0 DD             86         BNE  NEXT5
0371:F0 DB             87         BEQ  NEXT5
0373:A4 FD             88 END     LDY  YSAVE1
0375:C8                89         INY
0376:A9 00             90 END2    LDA  #$00
0378:20 F2 E2          91         JSR  $E2F2
037B:A6 FE             92         LDX  XSAVE
037D:A4 FF             93         LDY  YSAVE
037F:60                94         RTS
```

conosco ma non credo che ci siano problemi (le CALL in un Apple sono indispensabili!).

5) Sì è possibile, gli indirizzi, e il modo di raggiungerli saltando il BASIC SYSTEM, è spiegato nel Beneath the Apple ProDOS del «solito» Don Worth edito dalla Quality Software.

Detto fatto

Vi invio questa piccola routine che, sulla scia delle precedenti, aggiunge un'altra istruzione al Basic Applesoft. Come la IN$ dell'MS-DOS, verifica la presenza o meno di una stringa in un'altra, restituendo tramite la funzione USR la posizione di quest'ultima o altrimenti zero se la stringa da cercare non è presente.

Marco Merler - Gardolo (TN)

Un sort più veloce!

Sono alla ricerca di una procedura (possibilmente in ProDOS) che ordini alfabeticamente i file di testo; vedendo il programma di TRE X TE (che sto usando) ed il mio si nota subito la differenza: infatti il programma citato impiega circa 6 secondi ad ordinare 450 record, il mio non saprei ma certamente molto, molto di più.

Vorrei anche sapere se il programma INTEGRATO pubblicato a pag. 114 del N. 49 di Febbraio funziona in ProDOS e possibilmente come si deve procedere.

Colgo altresì l'occasione per ringraziarvi anticipatamente ed inviarvi i miei cordiali saluti.

Giovanni Zanuso - Valleggia (SV)

Il programma di Heapsort in linguaggio macchina di Bo Arnklit, pubblicato sul numero 3 di MC, è sicuramente uno dei più veloci, riuscendo ad ordinare un vettore di 1000 stringhe in poco meno di cinque secondi. Sostituendo l'istruzione HIMEM:37800 in HIMEM:37632 in testa al programma di lancio, si può usare anche sotto ProDOS (il ProDOS vuole che Himen sia multiplo di 256).

Il programma INTEGRATO invece non funziona sotto ProDOS a causa del diverso modo di intercettamento dei registri di I/O di quest'ultimo, perché sia utilizzabile occorrerebbe modificare tutta la parte di inizializzazione del programma binario. Se l'autore, o qualche volonteroso lettore, ci invieranno le modifiche saremo lieti di pubblicarle.

MC

LA SOFTCOM È LIETA DI PRESENTARTI....

... I SUOI PRODOTTI

NOVITÀ
- **ANTIRAM** POTENTISSIMO SPROTETTORE DISCO-NASTRO, DISCO-DISCO, NASTRO-DISCO, NASTRO-NASTRO

NOVITÀ
- **VIDEODIGITAL 64** PER DIGITALIZZARE E STAMPARE IMMAGINI CON IL TUO CBM 64
- **SPROTECT 64** PER SPROTEGGERE QUALSIASI PROGRAMMA SU DISCO O SU NASTRO
- **TURBO DISK** VELOCIZZA OLTRE 5 VOLTE IL TUO DRIVE COMMODORE
- **TURBO DOS II** VELOCIZZA OLTRE 10 VOLTE IL TUO DRIVE COMMODORE
- **DUPLICATORI** PER 2 REGISTRATORI COMMODORE

... LE SUE OFFERTE

- **ATARI 520 STM** CON MODULATORE VIDEO - DRIVE MONITOR MONOCROMATICO

  L. 1.490.000+ IVA
- **OKIMATE 20** STAMPANTE A COLORI PER COMMODORE - IBM **L. 550.000**
- **PANASONIC KX 1080 IBM** COMPATIBILE - NLQ - 100 CPS **L. 490.000**
- **DISCHI** $5^1/_4$ NASHUA A

  L. * 1.900 *

TUTTE LE NOVITÀ SOFTWARE
ATARI - COMMODORE 64 - 128
IBM - MSX - AMIGA

... I SUOI PC/XT COMPATIBILI

NOVITÀ
- **PC/XT TURBO** TURBO MAINBOARD ESP 640K 2 DRIVE - 256K - TASTIERA SCHEDA COLOR

L. 1.590.000 +IVA

– SCHEDA VIDEO COLOR	L.	180.000 +IVA
– ESPANSIONE 512K	L.	140.000 +IVA
– MOUSE PER IBM	L.	185.000 +IVA
– SCHEDA I/O PLUS	L.	195.000 +IVA
– SCHEDA MULTIFUNCTION	L.	220.000 +IVA
– MONITOR MONOCROMATICO HANTAREX	L.	180.000 +IVA
– CHIP 4164 PER ESP. MEMORIA	L.	3.500 +IVA

SCONTI AI SIG. RIVENDITORI
SI CERCANO DISTRIBUTORI DI ZONA

VENDITA PER CORRISPONDENZA

... E IL SUO VASTO ASSORTIMENTO

- COMMODORE 64 - 128
- REGISTRATORE 1530 COMMODORE
- STAMPANTI PANASONIC - OKI - STAR
- DISCHI MAGNETICI 3M - NASHUA - MASTER $5^1/_4$; $3^1/_2$; HIGH DENSITY
- MOUSE PER C - 64
- AZIMUTH CONTROLLER
- JOYSTICK, TAGLIADISCHETTI, COPRITASTIERA, CONTENITORI DISCHETTI, ACCESSORI MSX

DISTRIBUTORE PROGRAMMI ED ACCESSORI MASTERTRONIC * ULTIME NOVITÀ SOFTWARE *

SOFTCOM S.n.c. VIA PAOLINI 11 - 10138 TORINO - TEL. 011/ 44.55.43