



INTRODUZIONE AL BASIC DEL PC 128S

Uno dei linguaggi più versatili e potenzialmente utili ad un primo approccio al computer è senz'altro il BASIC. Per questo ottimo home computer, l'Olivetti Prodest ha scelto una delle versioni BASIC più estese e di più facile acquisizione

Il BASIC di cui è dotato il PC 128S, è il BASIC IV. Esso è la necessaria evoluzione dei vari BASIC I, II e III e perciò vedremo ora di compararne alcuni comandi e di scoprirne le facilitazioni nella gestione delle più diverse procedure, anche le più complesse, tramite l'uso di pochi ma appropriati comandi. Questa sezione della rivista sarà particolarmente utile a chi si avvicina per la prima volta al mondo dei computer e ai suoi linguaggi, ma potrà interessare anche coloro che, più esperti, vogliono conoscere a fondo le peculiarità del BASIC del PC 128S. In questo articolo, vedremo di esaminare globalmente questo linguaggio accennando sommariamente alle sue caratteristiche.

Il comando ON

Con questo comando è possibile richiamare delle procedure ed anche delle sobroutines.

Per esempio:

```
120 ON in%—129 PROCleft,
PROCright, PROCdown, PROCup,
ELSE PROCerr
```

La chiamata PROC ha esattamente la stessa sintassi delle normali procedure di chiamata, può avere cioè parametri separati da vir-

gole fra parentesi:

```
1240 ON menù % PROCadd (name$, age%),
PROCdel (name$),
PROCshow (name$)
```

La costruzione LIST IF

Il comando LIST è stato esteso e contempla una funzione di "cross-reference". Il comando LIST del tipo (LIST oppure LIST 200,1000 può essere seguito dalla parola chiave IF, la quale può essere a sua volta seguita da una stringa. Solo le linee di programma contenenti la stringa verranno listate. I caratteri che seguono l'IF, sono scritti così come vengono usati, in tal modo possono essere create le parole chiave. Per esempio:

```
LIST IF lista i comandi IF
LIST IF TIME lista le linee contenenti
TIME come funzione
LIST IF name$ (la array name$)
LIST 100,200 IF var = lista i valori
attribuiti a "var" nelle linee da 100 a
200
```

Il comando EDIT

Adesso è possibile usufruire di potenti agevolazioni del "text editor", per redigere un programma BASIC

senza dover fare lo *SPOOL dello stesso, usano il nuovo comando EDIT. EDIT, edita a proprio carico l'intero programma, ma può essere usato con gli stessi parametri del LIST (inclusa la parte IF) per editare solo parte del programma.

Data l'importanza di questo comando, una sua trattazione più ampia verrà fatta in un prossimo numero

La pseudo variabile TIME\$

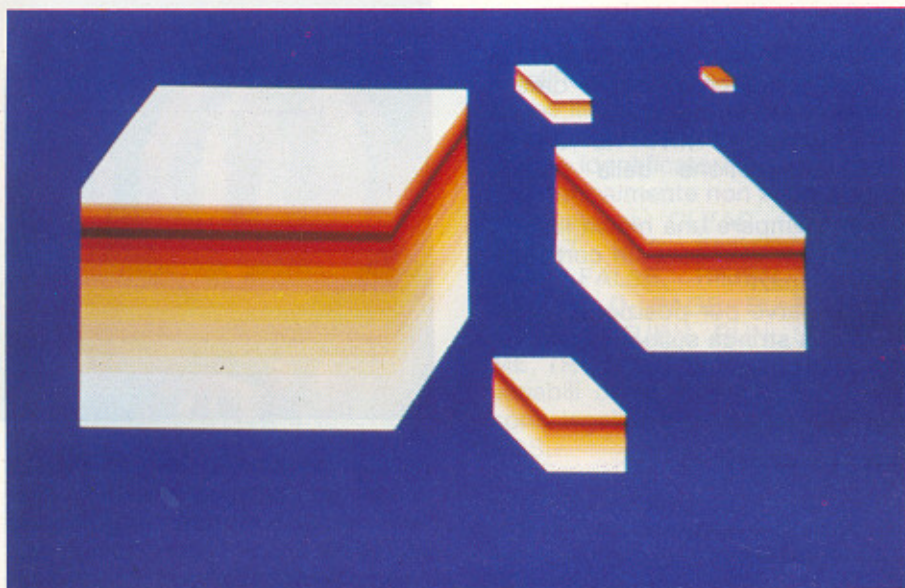
È riferita al tempo, e provvede all'accesso al tempo reale del sistema. Può essere usata come una funzione (ex. PRINT TIME\$) o come un'istruzione (ex. TIME\$ = "...").

EXT # =

Nel BASIC II, la funzione EXT # è usata per visualizzare la lunghezza di un file sequenziale aperto. La suddetta facilitazione può essere usata solamente con dei filing system idonei (ADSF, Network, ma non cassetta).

COLOR

Nel BASIC IV l'istruzione COLOUR



può essere anche scritta come COLOR. In ogni caso sarà egualmente accettata, ma verrà comunque listata come COLOUR.

Estensioni all'assembler

Il microprocessore usato nel PC 128S è il 65C12, una versione CMOS del 6502 con un set di istruzioni arricchito. Per avvantaggiarsi di ciò, l'assembler costruito nel BASIC IV è stato esteso per poter accettare le nuove istruzioni.

I listati dell'assembler sono ora strutturati in una maniera più leggibile, tanto che le label possono avere fino a nove caratteri incluse le iniziali.

Si possono usare i caratteri minuscoli in ogni parte del programma sorgente, come `lda &70, x and equus "fred"`.

L'assembler non va più in stato confusionale a causa del modo d'indirizzamento dell'accumulatore. Per esempio nelle versioni precedenti ASL ALFRED, veniva letto come ASL a seguito dal commento LFRED. Ciò non avviene più.

Il BASIC attiva alcuni indirizzi puntati a delle routine di floating point. Questi possono essere richiamati da programmatori in linguaggio macchina.

ALTRI CAMBIAMENTI MINORI

Il SAVE può prendere qualsiasi stringa come proprio argomento (ex. `SAVE a $ + b$`), mentre il "!" e "?" possono essere usati come parametri formali (`DEF FN fred (!&70)`).

Adesso si può usare il codice ASCII 141 nei commenti e nelle stringhe, per esempio per produrre caratteri in doppia altezza nei modi teletext:

```
100 REM <&8D> Big comment
110 REM <&8D> Big comment
```

(Prima il RENUMBER e il LIST sarebbero stati confusi da questo codice.)

In aggiunta, LIST potrà listare commenti che includono codici di teletext colour, come linee colorate nei modi teletext; non c'è più bisogno di chiudere fra virgolette le REM.

INTRODUZIONE PER I PROGRAMMATORI DEL MICROSOFT BASIC

Aiuti per la programmazione strutturata.

Il BASIC del PC 128S, dispone di diverse caratteristiche che aiutano il programmatore nella stesura di programmi leggibili e ben strutturati. I comandi in questione sono:

- l'istruzione `IF...THEN...ELSE...`
- la costruzione `REPEAT...UNTIL...`
- l'istruzione `ON...PROC...`
- procedure definibili
- funzioni definibili
- parametri di procedure e di funzioni e variabili locali.

I vantaggi di aiutare la stesura di programmi chiari sono molteplici:

- la possibilità di adoperare nomi di variabili lunghi e significativi
- identificazione automatica di loop, se richiesta
- un potente comando `RENUMBER`

In aggiunta a questi, ci sono i soliti comandi BASIC quali `GOTO`, `GOSUB`, `ON...GOTO/GOSUB` e `FOR...NEXT`.

Accessi al sistema operativo

Il computer dispone di un potente sistema operativo (MOS), che controlla l'hardware della macchina (monitor, tastiera, convertitori analogico-digitali, stampanti, ecc.). Il MOS inoltre, supporta il filing system, usato per immagazzinare o richiamare informazioni nei floppy-disc, tapes, cartridge, ecc. Il BASIC del PC 128S, dispone di un pieno accesso al MOS e ai filing system per mezzo di istruzioni come `PLOT`, `INKEY`, `ADVAL`, `OSCLI`, ecc. Questi comandi sono molto più facili da usare e da capire dei loro "oscuro" predecessori `PEEK` e `POKE`.

Accessi al codice macchina

Sebbene il BASIC IV del PC 128S è attualmente la più veloce versione di BASIC a otto bit disponibile, ci sono dei casi, come per esempio l'estrazione veloce di array molto vaste, in cui è richiesta la velocità del linguaggio macchina. A tal fine, il BASIC dispone di un potente as-



sembrer 65C12, che può essere usato per sviluppare diversi tipi di programmi in tale linguaggio.

L'istruzione CALL, può essere usata per chiamare le routine in linguaggio macchina e per passare loro dei parametri BASIC d'ogni tipo.

La funzioneUSR può essere usata per chiamare le routine del codice macchina inizializzando i registri del 6502 con le variabili del BASIC e riportando il contenuto del registro in uscita.

CALL eUSR dispongono di un facile accesso alle routine del sistema operativo, che non sono direttamente supportate dalle funzioni BASIC.

Trattamento dei dati

Il BASIC del PC 128S supporta numeri interi a quattro byte (a differenza dei soliti due byte), e numeri floating point a cinque byte. Le stringhe possono essere assegnate dinamicamente e possono arrivare a 255 caratteri di lunghezza. Le array multi-dimensionali sono utilizzabili come array di singoli byte.

L'accesso alla memoria è possibile grazie agli operatori indiretti "?", "!" e "\$." ha degli usi simili al PEEK e POKE (i quali non sono né necessari né supportati dal BASIC del PC 1286). "!" è simile, ma agisce su quattro byte, alla volta, invece di un singolo byte. "\$" agisce sulle stringhe di carattere (questo è provvisto in aggiunta alle normali variabili stringa).

Un ricco set di funzioni numeriche e stringa prevede funzioni logaritmiche e trigonometriche, gestione delle stringhe e funzioni di ricerca.

Formati di stampa

I numeri possono essere rappresentati in tre formati (generale, esponente, fisso), con campi di ampiezza variabili. I numeri interi possono essere stampati in esadecimale, e la funzione STR\$ può esse-

re usata per convertire numeri in stringhe esadecimali o per dare il formato della stringa.

L'istruzione PRINT ha diversi modi di gestione della pagina schermo:

- per stampare una nuova linea
- SPC per stampare un certo numero di spazi
- TAB serve per posizionare l'inizio di una stringa sullo schermo da una data posizione.

Trattamento degli errori

ON ERROR è una facilitazione fornita per individuare gli errori non fatali e per trattarli per mezzo del programma. Il comando è supportato dalle funzioni ERR e ERC, che danno il numero dell'errore e il numero di linea dell'ultimo errore commesso e dalla funzione REPORT che stampa l'ultimo messaggio d'errore.

Il modo diretto

Il prompt del BASIC è ">", e la sua presenza all'inizio di una linea indica che il BASIC è pronto ad accettare l'input. Il programmatore può digitare dei comandi come AUTO e LIST, i quali saranno immediatamente eseguiti, come PRINT LOG (12), o linee BASIC che devono essere inserite nel programma. Quest'ultime dovranno essere precedute da un numero di linea nel campo 0-32767. Le singole linee



del programma possono essere cancellate digitando il numero di linea immediatamente seguito da RETURN.

A parte riportiamo una lista di alcuni comandi con i loro significati. Notare che un comando non può essere eseguito da dentro un programma, e non può essere preceduto da un altro comando o da un'istruzione. Se seguito da una istruzione, quest'ultima verrà ignorata.

LE VARIABILI

In questa parte dell'articolo descriveremo i tipi di dati che possono essere usati nei programmi in BASIC IV del PC 128S e gli operatori e le funzioni previste per trattare gli stessi.

Tipi di variabili in BASIC

Nel BASIC del PC 128S, ci sono tre

Comandi	Significati
AUTO	Genera automaticamente i numeri di linea
DELETE	Cancella una gamma di numeri di linea da un programma
EDIT	Chiama il system editor per redigere il programma BASIC
LIST	Lista il programma
LISTO	Setta l'opzione indentation del LIST
LOAD	Carica un programma BASIC
NEW	Cancella il corrente programma
OLD	Richiama il programma cancellato da NEW



tipi di dati fondamentali: numeri reali, numeri interi, stringhe. Per ognuno c'è un tipo di variabile corrispondente: reale, intera e stringa. Esiste una facilitazione che permette di dichiarare delle array multi-dimensionali di ogni tipo. Ogni tipo di variabile dispone di un set di funzioni e di operatori con i quali operare. Notare che i numeri reali e gli interi, sono largamente intercambiabili; il BASIC, quando richiesto, converte automaticamente i valori reali in valori interi e viceversa.

Numeri reali

Le variabili di tipo reale sono semplicemente degli identificatori. In BASIC, un identificatore è una sequenza di uno o più caratteri nel set [A,B..Y,Z,a,b,..y,z,0,1..8,9,—,£]. Il primo carattere non può essere una cifra. Esempi di variabili reali sono:

```
£amount
numVars
A—Long—Variable—Name
var 1234-21z
```

Si può vedere come i caratteri speciali (underline) e £ (pound), agiscano come lettere extra. Una restrizione sugli identificatori è che non possono cominciare con tutte le parole riservate. Per esempio, GETADDR è illegale poiché GET è una parola riservata. Tuttavia le parole riservate possono essere incastonate in un identificatore, come in NAMELIST. Poiché le parole ri-

servate devono essere in maiuscolo, gli identificatori come getaddr e anche list possono essere usati. Le parole riservate permesse all'inizio di un identificatore, sono quelle che normalmente non sono seguite da qualcosa: CLEAR, CLG, CLS, COUNT, END, ENDPROC, ERL, ERR, FALSE, HIMEN, LOMEN, NEW, OLD, PAGE, PI, POS, REPORT, RETURN, RUN, STOP, TIME, TRUE, VPOS. In questo modo, variabili come COUNTER e POST sono legali e distinte da COUNT e POS.

Le array reali sono semplicemente degli identificatori seguiti da sotto scritte tra parentesi:

```
counts (char-128)
mat (i%,j%)
numVars (Ø)
```

Notare che la array numVars () e la semplice variabile numVars possono essere entrambe nello stesso programma; esse sono entità completamente separate. Una costante reale è una sequenza di cifre con una parte decimale opzionale e una parte esponente opzionale. Esempi sono:

```
1224
12.34
-0.000001
.123
10293.1234
+0.1E10
-112.32E-4
```

Il segno del numero, se non dato esplicitamente, è considerato sempre positivo. Il numero <n> dopo la E significa "moltiplicare il numero per 10 alla potenza di n". La grandezza massima di un numero reale che può essere rappresentata dal BASIC è 1.7014118344E+38. Il tentativo di generare un valore più grande di questo produrrà un errore. La grandezza minima di un numero reale (questo è il numero più vicino a 0) è 1,469367939E-39. Il tentativo di generare un numero inferiore a questo non avrà successo.

Numeri interi

Le variabili intere sono degli identi-

ficatori seguiti dal segno "%". Esempi sono:

```
check-sum%
£pennies%
cx%
YCOORD1%
```

Gli elementi delle array intere sono indicati nello stesso modo delle array reali: Facendo eguire alla variabile una lista di sotto scritti. Esempi sono:

```
lookup% (i%)
board% (row, col)
```

Le costanti intere vengono scritte come sequenze tra una e dieci cifre significative, opzionalmente precedute da un segno. Esempi sono:

```
-99
234134112
-532354
+42
```

Le costanti intere possono essere scritte anche in esadecimale, facendo precedere alla parte costante una "&". Il numero è una sequenza da una a otto cifre esadecimali significative (0,1..8,9,A,B..E,F). Esempi sono:

```
&0D
&FF7FF80
-&55AA1
&80000000
```

Non ci sono spazi fra la "&" e la prima cifra.

Le variabili intere del sistema

Ci sono 27 variabili intere, che sono definite permanentemente e vengono appunto chiamate: Variabili Intere del Sistema. Esse sono A%—Z% e a %..

a è usata per controllare il formato di stampa (vedi l'istruzione PRINT). 0% e P% hanno un significato particolare quando si programma in codice macchina. A%, C%, X% e Y% hanno un significato particolare quando si usano i programmi in codice macchina. Le altre variabili intere del sistema, sono completamente libere all'uso del programmatore, come anche



A%,C%,O%,P%,X% e Y%, se non usate in linguaggio macchina.

Il vantaggio principale delle variabili di sistema è che non vengono azzerate da CLEAR, NEW, OLD, RUN, LOAD e CHAIN, pertanto possono essere usate per passare informazioni attraverso i programmi. Un altro vantaggio è che il BASIC permette un'entrata molto veloce delle variabili di sistema, perché conosce esattamente la loro posizione in memoria. È pertanto una buona idea usare queste variabili quando risulta importante la velocità (come all'interno di un loop FOR... NEXT).

Altre variabili possono essere create usando un'assegnazione oppure l'istruzione INPUT e possono essere cancellate usando le istruzioni sopra menzionate.

Numeri interi indiretti

C'è un'altra forma di variabili intere, accessibile per mezzo del prefisso "!". Il formato di questa variabile è !<factor>, dove <factor> è un numero, una variabile o una funzione di chiamata o anche un'espressione fra parentesi. Alcuni tipici numeri interi indiretti sono:

```
!&70
!ptr%
!words% (i%)
```

Il valore di un numero intero indiretto, è il valore dei quattro bytes all'indirizzo specificato. In questo modo, !&70 ha un valore determinato dai quattro bytes, complemento di due, immagazzinati alla locazione &70-&73. Analogamente, !ptr% assume ptr% come indirizzo di un valore a 4 byte alla locazione ptr%-ptr%+3 e da il valore di questo numero intero.

C'è un'estensione alla notazione "!" : essa ha la forma <variabile>!<factor>.

Questa volta, <variabile>, è una variabile numerica (reale o intera). Il valore ottenuto è un numero intero a 4 byte alla locazione <variabile>+<factor>-<variabile>+<factor>+3. Da questo si può vedere che <variabile>

le>!<factor> è solamente un altro modo di scrivere! (<variabile>+<factor>). Ecco alcuni esempi:

```
table! 10
vals%!i%
address%! (2*index%)
```

Stringhe

Le variabili stringa possono supportare stringhe lunghe fino a 255 caratteri. L'estensione minore della stringa, è la stringa nulla che ha una lunghezza pari a zero. L'indirizzo di una variabile stringa, effettivamente l'indirizzo del suo "String information Block" (SIB), è lungo quattro byte, ed ha il formato:

```
Address of string text SIB + 0
Byte allocated to the string SIB + 2
Current length of string SIB + 3
```

Per ottenere il meglio dal BASIC, è utile sapere esattamente quante sono le stringhe immagazzinate. Questo è particolarmente importante se si stanno scrivendo programmi piuttosto lunghi, con parecchie stringhe, poiché il BASIC del PC 128S non esegue nessun "garbage collection". Ci sono però delle tecniche di programmazione che permettono un notevole risparmio di memoria.

Quando una variabile stringa viene creata (per esempio grazie ad un'istruzione di assegnazione), il BASIC respinge il suo immagazzinamento. Se la lunghezza iniziale è inferiore agli otto caratteri, l'immagazzinamento (il valore dei byte allocati) è lo stesso della lunghezza. Se la lunghezza iniziale è di otto o più caratteri, i byte allocati saranno maggiorati di otto caratteri rispetto all'originale, fino ad un limite massimo di 255 caratteri. Cioè per esempio:

```
a$ = "123456"
b$ = "1234567890"
```

verranno usati 6 byte per a\$ (gli stessi della sua lunghezza) e 18 byte per b\$ (8 in più dell'originale).

Collocando più caratteri dell'originale lunghezza, il BASIC permet-

te alla stringa di "crescere" prima di doverle trovare una nuova area di immagazzinamento. Questo è importante perché se la stringa sorpassa la sua collocazione originale, essa viene spostata in un'area più grande che la possa contenere. In tal modo, l'area che essa occupava in precedenza non è più accessibile e non può essere riusata, dando così luogo ad un enorme spreco di memoria.

Consideriamo ciò che accade quando \$ e b\$ vengono allungate di due byte, per esempio attraverso le istruzioni:

```
a$ = a$ + "AB"
b$ = b$ + "AB"
```

la stringa a\$ occuperà ora 8 byte. Precedentemente a\$ aveva a disposizione uno spazio di memoria pari a 6 bytes. Dopo l'operazione di somma delle stringhe, a\$ verrà rilocata in un altro spazio di memoria grande 8+8 byte, lasciando però inutilizzabili i primi 6 byte originali.

Quando b\$ è riassegnata, la sua nuova lunghezza è di 10 byte, ma lo spazio assegnatole dal BASIC era di 18 byte, cosicché essa non dovrà essere spostata in un altro luogo della memoria e di conseguenza non ci sarà spreco di preziosi byte.

L'unica eccezione è che se una variabile stringa è l'ultima variabile dinamica ad essere impostata, essa può crescere fino alla massima lunghezza permessa senza richiedere nuovo spazio. Questo accade perché il BASIC sa che non c'è nulla dopo di essa e che può essere estesa senza che ciò alteri alcun dato. È logico che se dobbiamo adoperare una stringa con un ampio campo di variazioni, sarà opportuno inserirla per ultima nelle variabili create dal programma. Bisogna però fare attenzione, perché le variabili LOCAL e i parametri procedura/funzione, possono creare variabili anche se non ancora esistenti.

Riassumendo: sarà utile assegnare subito ad una variabile stringa la massima lunghezza che si presume possa raggiungere. Natu-

ralmente, immediatamente dopo, si dovrà resettarla come segue:

```
a$ = STRING$(30,"*")
a$ = ""
```

Usando questo metodo sarà possibile:

- accelerare l'esecuzione del programma
- prevenire gli errori "No room" causati dal continuo spostamento delle stringhe in memoria.

Le costanti stringa sono sequen-

ze di caratteri tra 0 e 255, racchiuse tra virgolette. Il numero massimo dei caratteri in una stringa, è dato anche dal numero massimo di caratteri accettati dalla linea BASIC" e cioè 238.

Per poter includere in una stringa le virgolette, dovranno essere digitate due volte.

Esempi di costanti stringa sono
 "A string constant"
 "A string "" with a quote in"

** : REM stringa nulla

**** : REM doppie virgolette. ■

PERIFERICHE PER PC 128 E PC 128S

- 1 Monitor a colori
- 2 Stampante
- 3 Mouse
- 4 Joystick

