VIC da zero

di Tommaso Pantuso

RESET non distruttivo

Lavorando molto con un computer sorge l'esigenza di alcuni accessori (hard o soft) che evitino delle solenni arrabbiature o che in ogni caso ci aiutino nella programmazione. A proposito di arrabbiature, vi è mai capitato durante la prova di un programma appena battuto, magari in L/M, che la macchina non vi restituisca più il controllo per un errore commesso o che essa si blocchi per qualunque altra causa?

Leggendo questo articolo imparerete come in questi casi la si possa fare in barba al computer con ... il suo stesso aiuto!

A volte dopo l'apertura di un file di stampa di un programma, ad es.:

```
OPEN 5,4 : CMD 5 : LIST
```

succede che se alla fine della stampa, dimentichiamo, prima di effettuare una qualunque operazione, di digitare

```
PRINT#5 : CLOSE 5 : (Return),
```

provando ad usare la tastiera alcune volte la macchina si blocca e ritorna nel mondo dei "vivi" solo spegnendola e riaccendendola dopo qualche istante. Naturalmente il programma va perduto e, se non abbiamo provveduto a registrarlo, dovremo ricominciare tutto daccapo. Altre volte, per distrazione o per troppa sicurezza, capita di far girare un programma in L/M che non è stato ancora registrato, e per via di qualche errore il sistema entra in un loop infinito che ci costringe a spegnere il calcolatore con la conseguente perdita di tutti i dati e di tempo prezioso. Se siamo stati previdenti e possediamo il programma registrato su nastro o disco, perderemo lo stesso del tempo a ricaricarlo in memoria, specialmente se esso è conservato su un supporto a nastro magnetico, considerando i tempi di caricamento da cassetta!

L'ideale sarebbe poter sbloccare la macchina senza perdere i dati: è proprio quello che ci proponiamo di insegnarvi in questo articolo e nel prossimo, spiegando come aggiungere al VIC 20 un pulsante di RESET SISTEMA ed una routine che recuperi automaticamente il programma in memoria ristabilendo le condizioni della macchina prima del blocco. Trarremo spunto da questo argomento per analizzare il modo in cui vengono formattati ed inseriti i programmi Basic in memoria e come, in base alla loro struttura, essa venga organizzata.

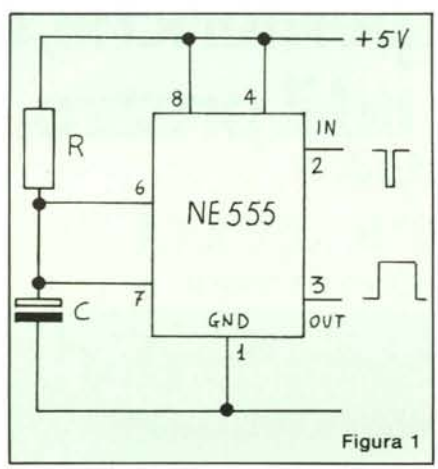


Figura 1

Il reset

Con un'operazione di *reset* sostanzialmente si ristabiliscono le condizioni iniziali (o quasi) del computer ed in ogni caso, dopo di essa, siamo in grado di operare come all'accensione.

Accendendo il VIC il sistema compie una serie di operazioni di inizializzazione eseguendo delle apposite routine che predispongono la macchina al corretto input/output.

La sequenza di inizializzazione è avviata da un apposito circuito di reset ottenuto tramite un integrato tuttofare siglato NE 555 che i lettori di VIC da ZERO già conoscono. Questo circuito, subito dopo l'accensione della macchina, tiene bassa una linea del microprocessore 6502, chiamata *linea di reset*, per un tempo non inferiore a 6 cicli macchina, comunicando, con tale operazione, al sistema di mandare in esecuzione le routine preposte all'inizializzazione, il cui indirizzo di partenza è puntato dal contenuto dei registri 65532 e 65533. Essi contengono i valori 34 e 253 (decimali) e quindi inviano alla locazione $34+256*253 = 64802$; provate infatti ad effettuare

```
SYS 64802
```

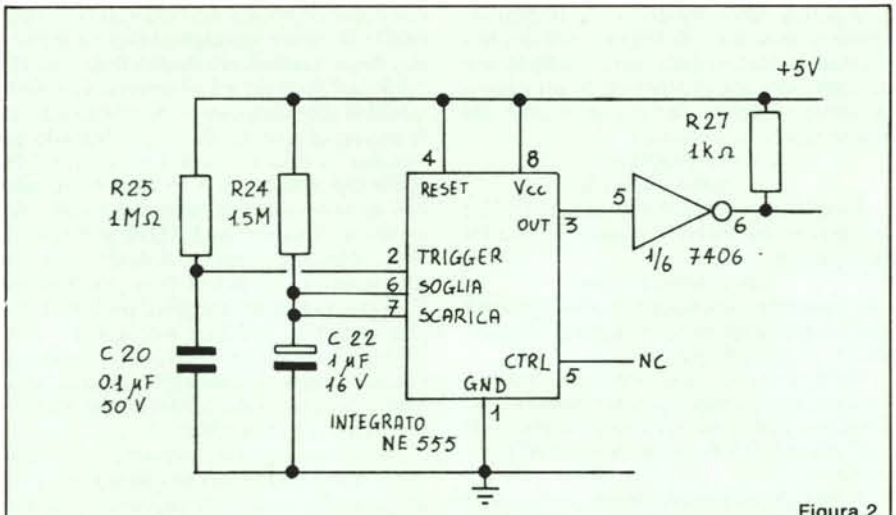
ed otterrete di porre il sistema nelle condizioni di partenza. Diamo un'occhiata più ravvicinata all'NE 555 e più precisamente alla funzione specifica che svolge nel VIC, dove viene utilizzato in configurazione monostabile. Questa parola *"difficile"* significa che, facendo riferimento alla figura 1 dove è riportato un NE 555 collegato come monostabile secondo uno schema base, se mandiamo un impulso negativo sul terminale 2, collegandolo per un istante alla massa, sull'uscita 3 la tensione passerà da 0 volt a +5 volt e resterà a tale valore, prima di ritornare nello stato iniziale, per un tempo programmabile tramite la scelta dei valori di R e C secondo la formula

$$t = -RC \ln(1/3) \simeq 1.1\ RC \text{ secondi}$$

con R e C espressi in ohm e farad.

In sostanza otteniamo quindi un impulso positivo sulla linea 3 in seguito ad uno negativo sulla 2.

Lo schema completo del circuito di reset utilizzato dalla Commodore nella versione E del VIC 20 è riportato in figura 2. In esso, alla accensione, il condensatore C 20, inizialmente scarico, mantiene collegato a massa il piedino 2; quando esso raggiunge le corrette condizioni di carica tramite R 25 disaccoppia tale terminale dalla massa portandolo a livello alto: in pratica R25 e C20 generano l'impulso negativo sull'ingresso di trigger ed R24 e C22 determinano il tempo durante il quale, in base ad esso, l'uscita 3 dovrà stare a livello alto, che secondo la formula fornita risulta 1.65 s circa. La condizione dell'uscita sarà invertita (cioè trasformata da alta a bassa) da una sezione del circuito integrato 7406, un buffer open collector, la cui uscita è direttamente collegata alla linea di reset del 6502 la quale risulterà quindi attivata per



CIRCUITO DI RESET DEL VIC 20

Figura 2

tutto il tempo in cui il pin 3 dell'NE 555 permarrà a livello alto.

Succede ora, e non a caso, che il famigerato piedino 40 del 6502, cioè la linea di reset, è accessibile all'utente poiché essa compare sull'uscita 3 della *user port* e sull'uscita X della *porta di espansione della memoria* del VIC.

Per inizializzare il sistema in qualunque momento basterà quindi pigiare un apposito pulsante collegato tra una delle linee citate e la massa (1,12,A ed N per la user port; 1,22,A e Z per la porta di espansione). Nella figura 3 è indicato il semplice collegamento da effettuare utilizzando un connettore 12+12.

Per il momento non diciamo altro sull'argomento reset ed andiamo a descrivere come è strutturata la memoria utente del computer e come in essa vengono immagazzinati i programmi, cosa che potrebbe sembrare scollegata dai fatti trattati, ma a cui è strettamente connessa e vedremo tra breve come.

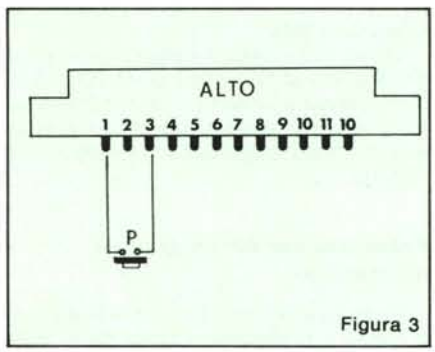


Figura 3

Memoria utente e puntatori

Un'area della memoria RAM del VIC è naturalmente utilizzata dall'utente per la stesura dei programmi ed in configurazione base il computer, come è noto, mette a disposizione per tale scopo 3.5K a cui se ne possono aggiungere altri 24 tramite le apposite cartridge da inserire nella porta di espansione.

In ogni configurazione, il sistema usa quest'area per memorizzarvi il programma, le variabili, i vettori e le stringhe, in un modo facilmente riconoscibile dall'esterno decodificando opportunamente le informazioni fornite da appositi puntatori, tramite i quali è possibile risalire al modo in cui le zone contenenti i vari elementi sono suddivise quando un programma è stato posto in memoria. Esaminiamo la suddivisione di tali zone facendo riferimento alla figura 4. Dalla parte più bassa della RAM utente (come numero, non sul disegno!) parte il programma vero e proprio formattato, come vedremo tra breve, in maniera opportuna; subito in coda ad esso vengono memorizzate le variabili semplici e di seguito quelle con indice (array). Dalla parte più alta della memoria partono invece le stringhe che procedono verso gli indirizzi più bassi. Quando la zona "stringhe" entra in collisione con quella "variabili" non avremo più spazio a disposizione per il programma e qualunque tentativo di inserzione provocherà l'emissione del messaggio "out of memory".

I numeri indicati sulla destra del disegno indicano le locazioni di pagina zero in cui è memorizzato il puntatore che delimita il corrispondente margine di memoria. Il contenuto di alcune di queste locazioni naturalmente varia man mano che si compone il programma (o dopo il run per i dimensionamenti di vettori) aggiornando il sistema sulle variazioni che avvengono di volta in volta in memoria. Facciamo un esempio per capirci riferendoci al VIC inespanso, cosa che non toglie nessuna generalità al discorso.

Se, dopo aver acceso la macchina, leggiamo il contenuto delle locazioni 43 e 44, ci accorgeremo che esse contengono i valori 1 e 16. Il contenuto di tali locazioni, opportunamente interpretato, indica il punto della memoria da cui inizia il programma; infatti

$$1 + 256 \times 16 = 4097$$

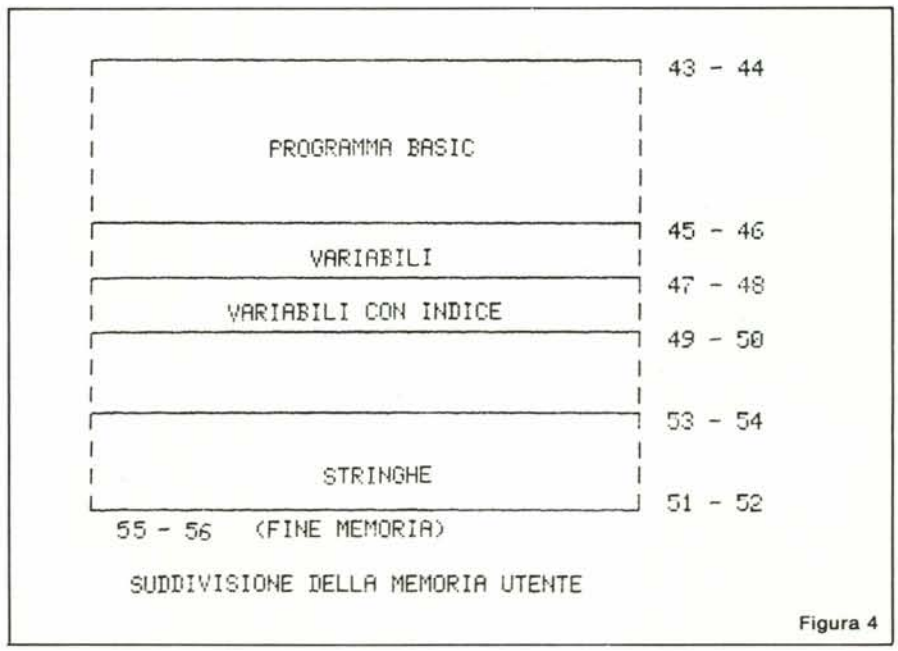
che è proprio lo start della memoria utente nel VIC base.

Il contenuto delle locazioni 55 e 56 è 0 e 30, per cui il sistema capisce che la memoria disponibile per i programmi termina all'indirizzo

$$0 + 256 \times 30 = 7680$$

Se ad esempio vogliamo conoscere lo spazio occupato da un vettore dimensionato in memoria dovremo verificare l'ampiezza della zona destinata alle variabili con indice tramite le seguenti linee:

```
10 A=PEEK(49) :B=PEEK(50)
20 C=PEEK(47) :D=PEEK(48)
30 H=A+256*B :L=C+256*D
40 AMP=H-L:PRINT AMP
```



SUDDIVISIONE DELLA MEMORIA UTENTE

Figura 4

```
INIZIO BASIC                  43 - 44
INIZIO VARIABILI              45 - 46
INIZIO VARIABILI CON INDICE   47 - 48
FINE VARIABILI CON INDICE     49 - 50
INIZIO STRINGHE               51 - 52
FINE STRINGHE                 53 - 54
FINE MEMORIA                  55 - 56

          ALCUNI PUNTATORI
```

Figura 5

e così per qualsiasi altra zona delimitata da due puntatori.

Come avrete potuto notare sono tutte cose molto semplici ma molto utili da ricordare quando si vuol sezionare a proprio piacimento una determinata zona di memoria nascondendola al sistema.

Tenendo presente un fattore molto importante, e cioè che le locazioni di RAM che contengono i puntatori sopra indicati sono accessibili tramite le istruzioni PEEK e POKE, supponiamo di voler selezionare un certo numero di byte in memoria proteggendoli da eventuali sovrapposizioni, ad esempio 100 byte, che prenderemo a partire dalla parte più alta della memoria. Dovremo allora comunicare al sistema di ignorare gli ultimi 100 byte imponendo che la memoria destinata al programma Basic termini (nel VIC base) a 7580 invece che a 7680. Inoltre, dato che le stringhe vengono memorizzate a partire dalla fine della memoria in giù, come già spiegato, dovremo imporre anche che esse vengano conservate a partire dall'indirizzo 7580. In pratica, come si suol dire, dobbiamo *abbassare* i puntatori alla fine della memoria ed all'inizio delle stringhe di 100 byte memorizzando nelle locazioni che li contengono dei nuovi valori: ricaviamoli.

$$A = INT(7580/256) = 29 ;$$
$$B = 7580 - A \cdot 256 = 156.$$

I valori da sostituire nei registri 51-52 e 55-56 sono quindi *rispettivamente* 156 e 29. Si ha infatti:

$$156 + 256 \cdot 29 = 7580$$

che il sistema considererà come indirizzo di fine memoria ed inizio stringhe proteggendo gli ultimi 100 byte.

Nella figura 6 riportiamo un comodo programma tramite il quale è possibile selezionare delle aree di memoria nella parte più alta con il VIC in qualsiasi configurazione.

Come ultima cosa vogliamo farvi notare che in effetti un puntatore di fine zona indica l'inizio di una nuova sezione. In altre parole, il puntatore di fine memoria fornisce il valore 7680, cioè il valore del punto d'inizio della memoria schermo, quindi la memoria utente termina effettivamente a 7679.

Prima di esaminare come recuperare un programma se si verifica un blocco del sistema, vediamo come esso viene formattato in memoria proponendoci di legare insieme nell'ultimo paragrafo i tre argomenti trattati.

Formattazione dei programmi in memoria

Quando scriviamo una linea di un programma e premiamo il tasto RETURN, essa viene prelevata dal buffer di tastiera e, prima di venire immagazzinata in memoria, fatta passare attraverso il cosiddetto *buffer del Basic* dove è compressa e formattata. La compressione viene effettuata sulle parole chiave del Basic attribuendo ad ognuna di esse un codice chiamato TOKEN che occupa un solo byte, ottenendo così un notevole risparmio di memoria. Ad esempio all'istruzione PRINT in memoria viene attribuito il valore decimale 153 con un risparmio di ben 4 byte; l'istruzione END viene conservata con il numero 128 e con risparmio di 2 byte e così via.

Quando il programma viene listato, il sistema esegue l'operazione inversa, cioè quella di espansione, trasformando i valori numerici in parole chiave.

Dopo questa breve premessa, esaminiamo come è codificato in memoria il programma riferendoci alle seguenti due linee:

```
10 PRINT"PROVA"
20 END
```

ed alla figura 7 (VIC inespanso).

L'inizio della memoria utente è a partire dalla locazione 4096 dove troveremo *sempre* memorizzato uno 0. Le locazioni 4097 e 4098 puntano al nuovo blocco che codifica una nuova linea; nel nostro caso esse contengono 14 e 16 quindi il nuovo blocco parte da

$$14 + 256 \cdot 16 = 4110$$

come è semplice verificare osservando la figura 7.

In 4099 e 4100 troviamo rispettivamente 10 e 0 che rappresentano il byte basso ed alto del numero di linea Basic:

$$10 + 256 \cdot 0 = 10$$

In 4101 troviamo il token di PRINT che è 153.

```
10 REM ------------------------------------
20 REM ------------ MEM  BOT -------------
30 REM ---------- (C) TPS 1984 ------------
40 REM ------------------------------------
50 PRINT"□ ØESPANSIONE:"
60 INPUT" ØØ,3,8,16 O 24K";M
70 IFM=0ORM=3THENI=7680:GOTO90
80 I=8192+M*1024
90 INPUT" ØQUANTI BYTE   ";H
100 A=PEEK(55)+PEEK(56)*256
110 B=A-H:C=INT(B/256):D=B-C*256
120 POKE51,D:POKE52,C
130 POKE55,D:POKE56,C:PRINT"Ø"
140 PRINT"START ZONA PROT.";B
150 PRINT"END   ZONA PROT.";I-1
160 PRINT"ØP51 =";PEEK(51),"P52 =";PEEK(52)
170 PRINT"ØP55 =";PEEK(55),"P56 =";PEEK(56)
180 PRINT"BYTE PROT.";(I-1)-B
190 CLR
```

Figura 6

```
10 PRINT"PROVA"
20 END


4096   0 : START
4097  14 : LINK AL PROSSIMO BLOCCO, PARTE BASSA
4098  16 : LINK AL PROSSIMO BLOCCO, PARTE ALTA
4099  10 : NUMERO DI LINEA, PARTE BASSA
4100   0 : NUMERO DI LINEA, PARTE ALTA
4101 153 : TOKEN DELLA PAROLA CHIAVE PRINT
4102  34 : CODICE ASCII DELLE VIRGOLETTE
4103  80 : CODICE ASCII DELLA LETTERA P
4104  82 : CODICE ASCII DELLA LETTERA R
4105  79 : CODICE ASCII DELLA LETTERA O
4106  86 : CODICE ASCII DELLA LETTERA V
4107  65 : CODICE ASCII DELLA LETTERA A
4108  34 : CODICE ASCII DELLE VIRGOLETTE
4109   0 : INIZIO NUOVA LINEA BASIC
4110  20 : LINK AL PROSSIMO BLOCCO, PARTE BASSA
4111  16 : LINK AL PROSSIMO BLOCCO, PARTE ALTA
4112  20 : NUMERO DI LINEA, PARTE BASSA
4113   0 : NUMERO DI LINEA, PARTE ALTA
4114 153 : TOKEN DELLA PAROLA CHIAVE END
4115   0 : INIZIO NUOVA LINEA BASIC
4116   0 :
4117   0 :
```

Figura 7

128	END	166	SPC(
129	FOR	167	THEN
130	NEXT	168	NOT
131	DATA	169	STEP
132	INPUT#	170	+
133	INPUT	171	-
134	DIM	172	*
135	READ	173	/
136	LET	174	
137	GOTO	175	AND
138	RUN	176	OR
139	IF	177	>
140	RESTORE	178	=
141	GOSUB	179	<
142	RETURN	180	SGN
143	REM	181	INT
144	STOP	182	ABS
145	ON	183	USR
146	WAIT	184	FRE
147	LOAD	185	POS
148	SAVE	186	SQR
149	VERIFY	187	RND
150	DEF	188	LOG
151	POKE	189	EXP
152	PRINT#	190	COS
153	PRINT	191	SIN
154	CONT	192	TAN
155	LIST	193	ATN
156	CLR	194	PEEK
157	CMD	195	LEN
158	SYS	196	STR$
159	OPEN	197	VAL
160	CLOSE	198	ASC
161	GET	199	CHR$
162	NEW	200	LEFT$
163	TAB(	201	RIGHT$
164	TO	202	MID$
165	FN	203	GO

Figura 9 - Codifica delle parole chiave

4096	0
4097	0
4098	0
4099	10
4100	0
4101	153
4102	34
4103	80
4104	82
4105	79
4106	86
4107	65
4108	34
4109	0
4110	20
4111	16
4112	20
4113	0
4114	128
4115	0
4116	0
4117	0

Figura 8

Il contenuto delle locazioni che vanno da 4102 a 4108 è semplicemente interpretabile e comunque sufficientemente commentato nella figura 7. In 4109 troviamo uno 0 tramite il quale il sistema capisce che è finito il primo blocco e comincia la formattazione del secondo.

La fine di un programma è sempre indicata da due zeri consecutivi posti di seguito allo zero di fine blocco.

Tiriamo le somme

Dopo tutte le nozioni fornite non ci resta che ritornare sull'argomento "reset" ed esaminare come questo influisca sul sistema quando in macchina è memorizzato un programma. Per prima cosa bisognerà collegare al computer il pulsante che connetta a massa, se premuto, la linea 40 del microprocessore provocando il ripristino del sistema per mezzo delle apposite routine preposte, come già detto, a tale scopo.

Per far questo si possono seguire le indicazioni della figura 3 che illustra come collegare il pulsante sopra indicato saldandolo ad un connettore 12+12 da inserire nella user port oppure eseguire l'analoga operazione tra la linea X ed una qualsiasi linea di massa su un connettore 22+22 per la porta di espansione.

Collegato il pulsante, introduciamo in macchina le due linee di programma (le solite) indicate in figura 7.

Se ora premiamo tale pulsante il sistema si bloccherà per un paio di secondi e poi rivisualizzerà la scritta che appare normalmente all'accensione ridando il controllo all'utente.

Se però provate a listare il programma che avevate inserito prima del ripristino, esso non vi sarà mostrato, anche se si trova ancora in memoria: il perché è molto semplice.

Osservate attentamente la figura 8, la quale mostra la formattazione del programma dopo il reset; essa è identica a quella originaria per tutto tranne che per il contenuto della seconda e della terza locazione che ora contengono entrambe uno 0. Ciò significa che il sistema non ha più l'informazione che gli permette di concatenare il primo blocco di programma al secondo (ribadiamo che le prime due locazioni di ogni blocco contengono un puntatore al blocco successivo) e così via.

Osservando la figura 7 possiamo notare che originariamente in 4097 e 4098 erano contenuti i numeri 14 e 16 e per riportare il formato nelle condizioni originali non ci resta che effettuare

POKE 4097,14 : POKE 4098,16.

Se dopo tale operazione provate a listare il programma, esso verrà integralmente visualizzato sullo schermo.

Ma non abbiamo ancora finito!

Se provate a far girare il programma il sistema ignorerà il comando RUN perché non vede nessun programma in memoria.

L'operazione di reset infatti riassetta tutti i puntatori di sistema compresi quindi quelli di *inizio* e *fine programma* posti in pagina zero agli indirizzi 43-44 e 45-46. Dopo il ripristino avremo:

inizio programma

PEEK(43)+256*PEEK(44)=4097

fine programma

PEEK(45)+256*PEEK(46)=4099

quindi il sistema leggendo questi due valori penserà che non è presente in macchina nessun programma da elaborare perché ad esso non è stata destinata nessuna area. Dovremo quindi ripristinare anche le locazioni 45 e 46 memorizzando in esse i valori che contenevano prima del reset. Essi erano, per il nostro programma, 22 e 16 che indicavano il punto 22+256*16=4118 stabilendo 22 byte per il programmino di prova. I nostri problemi saranno quindi risolti da

POKE 45,22 : POKE 46,16

Da questo momento in poi potrete continuare le operazioni come se nulla fosse accaduto! È tutto per questo mese.

Nel prossimo numero presenteremo, insieme ad altre cose nuove, la conclusione di questo articolo, consistente in una routine che avvalendosi della possibilità di modificare la gestione dell'interrupt, permetterà il recupero automatico del programma dopo un reset hardware del sistema.

Appuntamento quindi al prossimo numero e ... buone vacanze! MC

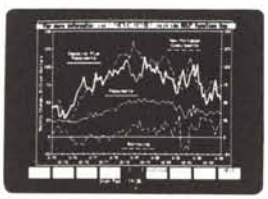
HP computer

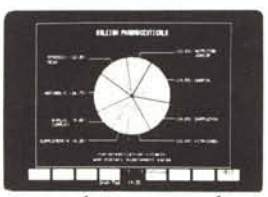
Ipersonal: ipersensibile
perché lo tocchi sullo schermo
e lui ti obbedisce.
Ipergestionale perché ricco
di programmi di utilità
aziendale dalla grafica alla
contabilità.
Iperfacile perché ti capisce
subito e in italiano. Iper...

dorland italia

: l'ipersonal

Tu ti siedi al tuo Personal, e colloqui con lui. È il tuo nuovo HP 150, quanto di più progredito ci sia nel settore. Spesso non hai neanche bisogno di toccare la tastiera. Tocchi direttamente lo schermo, cambi, cancelli, sposti, risolvi e se vuoi, inoltre, puoi disporre di una stampante termica incorporata ed una vasta gamma di periferiche.





Insomma, hai un vero "alter-ego", ora.

Il touch-screen è solo la punta dell'iceberg, la parte visibile, la prova... tangibile della maneggevolezza ed elementarietà dei comandi, contrapposte ad una tecnologia tanto avanzata.

Già, perché l'HP 150 congiunge la sua qualità di essere "alla mano" (ci vai "d'accordo" subito, ti capisce e ti segue anche se non hai mai usato un personal) con l'assortimento dei programmi che vanno dalla contabilità alla gestione di magazzino; dal trattamento dei testi ai programmi tecnico-scientifici, a una vasta gamma di applicazioni, che potrai scegliere secondo le tue necessità di oggi e domani.

L'ipersonal continua la tradizione HP, aperta al suo pubblico.
Perciò, per ogni necessità, potrai telefonare all'HP e avrai un'assistenza gratuita.
La garanzia, poi, è estesa a 12 mesi.

HP 150 è facilmente collegabile anche come terminale coi principali elaboratori, e grazie al sistema operativo MS/DOS* ti permette di utilizzare una vasta fonte di software già disponibile. Per saperne di più, prendi contatto con il rivenditore più vicino.

Hewlett-Packard Italiana S.p.A.
Via G. Di Vittorio, 9 - 20063 Cernusco S/N
Milano - Tel. 02/92369362

*Trade Mark

HP-soluzioni produttive

Se vuoi saperne di più sul personal HP 150
invia questo tagliando a Hewlett-Packard Italiana S.p.A.
Marketing Communication - C.P. 10190 - 20100 Milano.

Nome e cognome _____

Società _____

Indirizzo _____

MC MICRO/F

HEWLETT PACKARD