VIC da zero +64

a cura di Tommaso Pantuso

The Disk Editor

di Luigi Tavolato

Chissà quante volte avranno desiderato andare a curiosare in un qualche floppy e scoprire i piccoli segreti che un esperto programmatore vi ha racchiuso, o ritoccare qua e là qualche sigla, qualche scritta, qualche protezioncina, o scoprire di settore in settore come i file sono registrati, cosa effettivamente una directory contiene, o...?

Ora potete.

Il Disk Editor rende facilmente accessibili tutti i settori del disco: visualizzarli, modificarli, cancellarli... non sarà che questione di un attimo.

Tramite il Disk Editor potrete leggere (N) e quindi editare ciascun settore usando due diverse modalità: il modo Testo (T), quando i caratteri da inserire sono accessibili da tastiera, o byte per byte (M), digitando il corrispondente valore ASCII, nel caso in cui non lo siano. Il comando «S» vi permetterà quindi di riscriverlo esattamente con tutte le modifiche apportate.

Tramite i comandi «+» e «–» potrete spostarvi avanti ed indietro su di una traccia.

Con la «L» potrete scorrere il contenuto di un file, link per link. L'Hard Copy (H) riprodurrà su carta l'immagine dello schermo. In più una piccola sopresa: un generatore di errori di lettura (E).

Il programma si autodocumenta più che a sufficenza, attenzione però a riprodurre esattamente l'immagine video poiché il Disk Editor fa largo uso della memoria schermo e se non trova le informazioni al punto giusto potrebbero esser problemi.

La generazione degli errori di lettura

Quando il DOS formatta un disco scrive su ciascuna traccia tutta una serie di riferimenti che gli permetteranno successivamente di individuare univocamente ciascun settore.

Ogni settore ha i suoi riferimenti caratteristici (numero della traccia, numero del settore, ID), diversi, nella loro globalità, da quelli di tutti gli altri.

Un carattere di sincronismo permetterà al DOS di posizionarsi all'inizio di ciascun settore, dopodiché, letti i riferimenti suddetti, deciderà se è quello che gli interessa o no, nel qual caso andrà a leggere il successivo. Se lo trova e tutte le informazioni coincidono con quelle che si aspettava di trovare, ricerca il carattere di sincronismo della parte del settore dedicata a contenere i dati e li legge, immettendoli nel buffer selezionato, dove rimarranno a disposizione.

Se qualcosa non va durante la ricerca o la lettura, richiama la routine di errore e comunica che cosa ha riscontrato, interrompendo qualsiasi operazione stesse compiendo.

Dunque, per creare un errore di lettura, bisogna cancellare una o più di quelle informazioni a cui il DOS fa riferimento per portare a termine le operazioni richieste.

Gli errori di lettura che la routine di Generazione Errori del Disk Editor è in grado di riprodurre sono:

20: non riesce a trovare i riferimenti per identificare il settore (cancella le informazioni del Block Header).

21: non trova il carattere di sincronismo per potersi posizionare sul settore (cancella il primo Sync).

22: non trova il carattere di sincronismo del blocco dati (cancella il secondo Sync).

23: quando viene scritto un blocco dati, viene generato un carattere di controllo, detto Checksum. Se rileggendo un blocco dati, il Checksum trovato non corrisponde a quello che invece il DOS dovrebbe riscontrare, viene generato questo errore (cancella alcuni byte del blocco dati).

Uso degli errori nelle protezioni

Dunque, con il Disk Editor, ognuno potrà creare le proprie protezioni (o ricreare quelle altrui!). Come?

È piuttosto semplice. Basta richiamare la routine per generare l'errore (E), effettuare l'operazione su di un settore del disco su cui verrà registrato il programma che vi interessa ed inserire in quest'ultimo il test per controllare se l'errore o gli errori previsti si trovano nella giusta posizione. Attenzione però a non rovinare quelli contenenti informazioni vitali per il funzionamento delle routine che si intendono proteggere. Normalmente tutti i programmi (o i file in generale) vengono registrati a cominciare dalla prima traccia disponibile, il più vicina possibile alla 18, per poi svilupparsi verso la traccia 1 o la 35. Per cui queste saranno le ultime ad essere occupate, e

Read Errors Generator

```
0500  JMP ($0207)   Alla routine di errore ($0516 o $0539)

****  Seleziona Traccia e Settore su cui operare ed attiva l'Interrupt

0503  LDA $0203     Traccia
0506  STA $0A       Traccia per buffer 2
0508  LDA $0204     Settore
050B  STA $0B       Settore per buffer 2
050D  LDA #$E0      Comando di scrittura
050F  STA $02       Comando per buffer 2, attiva l'Interrupt del DOS
0511  LDA $02       Ad esecuzione terminata viene azzerato il comando
0513  BMI $0511     Terminato ?
0515  RTS           Fine

****  Routine per errori 20 22 23

0516  JSR $F50A     Cerca l'inizio del blocco dati
0519  BIT $1C00     Legge la Control Port
051C  BPL $0519     e aspetta finche' non sia disponibile
051E  JSR $F556     Cerca il carattere di sincronismo
0521  LDY $0205     Legge il valore del loop di posizionamento testina
0524  LDX $0206     Legge il valore del loop di cancellazione

****  Loop di Posizionemto Testina

0527  CLV
0528  BVC $0528     Aspetta che venga completata la lettura di un byte
052A  DEY
052B  BNE $0527     Ci sono altri bytes da leggere ?

052D  JSR $0547     Seleziona il Modo Scrittura

****  Loop di cancellazione

0530  CLV
0531  BVC $0531
0533  DEX
0534  BNE $0530
0536  JMP $0541

****  Routine per errore 21

0539  JSR $0547     Seleziona il Modo Scrittura
053C  TAX
053D  TAY
053E  JSR $FDB9     Loop di cancellazione

****  Fine lavoro
                                  - 2 -
0541  JSR $FE00     Seleziona il Modo Lettura
0544  JSR $F969     Rientra nel ciclo di lavoro del DOS

****  Seleziona la Porta di lettura/scrittura (Port A)

0547  LDA #$FF
0549  STA $1C03     Seleziona la Porta A in OUTPUT
054C  LDA $1C0C
054F  AND #$1F
0551  ORA #$C0
0553  STA $1C0C     Seleziona Registro di Controllo Porta (PCR) a scrittura
0556  LDA #$00     Carattere di Cancelllazione
0558  STA $1C01     Immette nella Porta A (R/W) il carattere da scrivere
055B  RTS
REGISTRI : VIA 6522 2. controllo motore e testina di lettura/scrittura

1C00 : Port B. registro di controllo della Porta (Status + cmd).
1C01 : Port A. Porta di lettura/scrittura dati (Per la R/W HEAD).
1C02 : registro di direzione dati (inPut/outPut) della Porta B.
1C03 : registro di direzione dati (inPut/outPut) della Porta A.
1C0C : registro di controllo Porta (PCR) : seleziona lettura/scrittura.
```

Uso della Porta B (1C00)	Bit
Led & motor : STP 1	0
STP 0 movimento testina	1
(STePPer motor ON/OFF)	
Motor : Pilota9gio motore (MTR)	2
Led ON/OFF (ACT)	3
Switch di Protezione scrittura	4
Sincronismo (SYNC)	7

Disassemblato della routine di generazione degli errori di lettura inserita nel Disk Editor per il drive 1541.

solo quando il floppy sarà veramente pieno.

Dunque è meglio orientare la propria scelta verso queste anche se è molto probabile che i lavori che si intende proteggere difficilmente possano occupare tutto il dischetto, settori errati compresi.

Ad ogni modo, tramite il Disk Editor, è piuttosto semplice da controllare se un settore è disponibile o no.

Se però voleste proprio utilizzare un settore più interno di quelli suddetti, prima di scrivere qualsiasi cosa sul disco, bisogna allocarlo nella BAM tramite un comando di Block-Allocate (B-A), in modo che il DOS non tenti di utilizzarlo, generare l'errore e quindi procedere con le successive operazioni.

Molti programmi di backup, ma non tutti, non sono in grado di riprodurre questi errori, in particolar modo quelli che copiano file per file, e quindi il programma si renderà facilmente conto se è stato o no duplicato.

Le conseguenze di una scoperta di questo genere (la vendetta!) dipenderà dalla fantasia (o dalla cattiveria) del programmatore: uno scratch del disco, soluzione diretta e piuttosto brutale ma non definitiva (si può sempre fare un'altra copia e stare più attenti), o un errore ogni tanto su informazioni importanti, danno più sottile e a volte estremamente pericoloso, o un reset del sistema, piuttosto banale ma efficace, o...

Se il programma è in linguaggio macchina sarà abbastanza difficile trovare dove il test viene effettuato, praticamente impossibile se è compilato; il BASIC però non offre nascondigli sicuri.

Un test potrebbe essere, ad esempio, del tipo:

```
110 OPEN 15,8,15
120 OPEN 6,8,6 "#"
130 FOR I=1 TO 4
140 READ TR,SE,ER
150 PRINT#15,"U1:"6;0;TR;SE
160 INPUT#15,E
170 IF E<>ER THEN .... (scegliete voi!)
180 NEXTI
190 CLOSE6:CLOSE1
200 DATA 1,0,20,4,7,23
220 DATA 35,11,21,2,16,23
```

Un test abbinato ad un autostart, magari con il blocco del RUN/STOP RESTORE, renderà la vendetta ancora più difficile da fermare.

Il Disk Editor è dotato anche di una routine (R) per ricercare e visualizzare eventuali errori presenti sul disco, con tutte le informazioni necessarie per sapere dove andare a ricrearli. Questo poiché moltissime protezioni fanno largo uso di tecniche di questo genere e per superarle è necessario sapere quali operazioni eseguire e dove effettuarle.

A questo punto... in bocca al lupo! A proposito, nel programma Disk Editor, per uscire dalla Routine di Ricerca Errori mentre sta ancora operando premere X.

Variabili e funzioni:

AP$:contiene i doppi apici in forma grafica. Viene usato dall'Hard Copy.
BS: locazione di base dello schermo per le operazioni sul settore in memoria
C1$ C2$ C3$ C4$: contengono comandi cursore ripetitivi.
CR: contatore di caratteri per l'input pilotato.
E$<: contiene i valori dei loop e dei puntatori per il generatore di errori.
ER$: contiene la routine di generazione degli errori di lettura.
LO: locazione di memoria schermo di base per visualizzare in reverse il comando richiesto.

MC: numero massimo di caratteri accettabili per l'input pilotato.
ME: numero massimo di errori visualizzabili contemporaneamente.
NE: numero di errori riscontrati.
O: posizione orizzontale.
SE: settore selezionato.
SS: settore successivo (link).
ST: status.
SX: settore precedente a quello su cui generare l'errore selezionato.
TR: traccia selezionata.
TS: traccia successiva (link).
V: posizione verticale.
W: variabile in output dall'input numerico.
Z$: contiene uno zero binario.
FNP<: restituisce il valore ASCII di un carattere decrementato o incrementato di 128 a seconda che il suo valore iniziale sia maggiore o minore di esso rispettivamente.
FNT<: restituisce il numero massimo di settori disponibili per una traccia.

Questo programma è disponibile su disco presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 204.

Nota

I codici di controllo dei listati sono riportati in forma «esplicita», in conseguenza dell'impiego della stampante Star NL-10 e relativa interfaccia per Commodore. Ovviamente, nella digitazione del programma è necessario usare i consueti tasti che corrispondono alle indicazioni fra parentesi: ad esempio cursore destro per (RGHT), CTRL-3 per (RED) eccetera.

La memoria del 1541

Il 1541 è una unità disco intelligente, dotata di un suo proprio Sistema Operativo, il DOS, che occupa 16K di memoria, supportato dal proprio microprocessore (6502). Ogniqualvolta impartiamo un ordine è il DOS del drive ad eseguirlo, sganciandosi totalmente dall'Unità Centrale (il C64).

Grazie a tecniche di Interrupt, il DOS è in grado di effettuare molte operazioni contemporaneamente: controlla il traffico di dati tra C64 e se stesso, interpreta ed esegue le operazioni richieste, gestisce i file ed i buffer, esegue tutta una serie di operazioni dette «Hardware-Oriented» quali la formattazione, le operazioni di lettura/scrittura...

Ad esempio, quando inviamo un qualsiasi comando al drive viene generato sulla linea di ATN un segnale di Interrupt che comunica al DOS di interrompere momentaneamente ciò che sta facendo per prender nota della nostra richiesta ed inserirla nella cosiddetta Job Queue (coda di lavoro). Verrà eseguita soltanto quando saranno state portate a termine tutte le operazioni che la precedono.
La mappa della memoria del 1541 si può riassumere in questo modo:

$0000-07FF: RAM utente (2K)
$0000-$00FF: pagina zero, contiene i flag, le costanti, i riferimenti per la gestione dell'I/O e delle routine di Interrupt.
$0100-$014F: area di stack
$0200-$0228: buffer di input per stringhe comando
$024A-$02B0: area per la gestione dei puntatori RAM e dei file
$02D5-$02F9: buffer per i messaggi d'errore
$0300-$03FF: buffer 0

```
1000 rem****************************
1010 rem* the disk editor rel. 3.3 *
1020 rem*                          *
1030 rem*    luigi     tavolato    *
1040 rem*                          *
1050 rem****************************
1060 :
1070 clr:poke53280,0:poke53281,0:z$=chr$(0):bs=1366:dime$(30)
1080 c1$="(home)(lgrn)(down)(down)(down)":c2$=c1$+"(down)(down)(down)"
:c3$=c2$+"(down)(down)(down)":c4$=c3$+"(down)(down)(down)(down)(
down)(down)(down)"
1090 :
1100 deffnt(t)=20+2*(t>17)+(t>24)+(t>30):rem*** n.max settori per trac
cia
1110 :
1120 deffnm(p)=p+128*((p)=128)-(p<128)):rem*** video reverse on/off
1130 :
1140 print"(clr)(dish)(swlc)(grn)┌──────────────────────┐ ┌──────────
─┐"
1150 print"│(yel)Disk Editor by(grn)├────┤Drive│Dos│Trk│Sec│
1160 print"├──────────────┤(lblu)1986(grn)├────┼───┼───┼───┤"
1170 print"│(yel)Luigi Tavolato(grn)├────┘   │   │   │   │"
1180 print"└──────────────┘        └───┴───┴───┴───┘"
1190 print"┌──┬──┬──┬──┬──┬──┬──┬──┐"
1200 print"│Pos│ │Byte│ │ │ID│ │Link│ │ │"
1210 print"└──┴──┴──┴──┴──┴──┴──┴──┘"
1220 print"  ┌──────────────────────┐"
1230 for i=1to8:print"  │";:spc(32)"│":next
1240 print"  └──────────────────────┘"
1250 print"  ┌──────────────────────┐"
1260 print"  │(yel)N nuovo settore  T modo testo   (grn)│
1270 print"  │(yel)+ set. seguente  M modifica byte(grn)│
1280 print"  │(yel)- set. preced.   H hard copy     (grn)│
1290 print"  │(yel)L link seguente  E genera errori(grn)│
1300 print"  │(yel)S scrivi sett.   R ricerca err. (grn)│
1310 print"  └──────────────────────┘";
1320 :
1330 rem*** init ***
1340 :
1350 for i=0to7:forj=4to35:poke55656+i*40+j,13:nextj,i:rem** colour **
1360 :
1370 rem*** read error generator ****
1380 for i=0to99:reada:d1=d1+a:er$=er$+chr$(a):next:ifd1<>9076then1520
1390 :
1400 rem***  loops & pointers  ******
1410 forj=0to3:for i=0to3:reade:e$(j)=e$(j)+chr$(e):d2=d2+e:nexti,j
1420 ifd2<>348then1520
1430 :
1440 rem*** apici in forma grafica **
1450 for i=0to8:readа:ap$=ap$+chr$(a):d3=d3+a:next:ifd3<>951then1520
1460 :
1470 rem*** clear screen parziale ***
1480 for i=0to78:readа:poke49152+i,a:d4=d4+a:next:ifd4<>10582then1520
1490 :
1500 poke55554,13:poke55559,13:poke55560,13:goto1530
1510 :
1520 printc3$spc(13)"(down)(down)(down)Data  Error":end
1530 :
1540 printc3$;"(down)(rght)(rght)(rght)(rght)(rght)   Inserisci il dis
co da
1550 print"(down)(rght)(rght)(rght)(rght)(rght) modificare e premi un
tasto":poke198,0:wait198,1:sys49152
1560 :
1570 open1,8,15:close1:ifst<>0thenend
1580 open15,8,15,"i0":gosub2570:iferthen2000
1590 open8,8,8,"#0":open6,8,6,"#2"
1600 :
1610 :rem*** legge drive e dos ***
1620 :
1630 print#15,"m-r"chr$(192)chr$(229)chr$(8):input#15,a$
1640 printc1$spc(22)mid$(a$,5,3):chr$(asc(right$(a$,1))-128);"(rght)":
left$(a$,3)
1650 :
1660 :
1670 rem*** write protect on ? ****
1680 :
1690 print#15,"m-r"chr$(0)chr$(28):rem**** byte di controllo ***
1700 get#15,p$:p=asc(p$+z$)and16:rem*** isola bit di protezione scritt
ura ***
1710 ifp=0thenprintc3$spc(11)"(down)(down)(down)WRITE PROTECT ON !!":g
oto2000
1720 :
1730 rem**** scrive nel controller ****
1740 rem*** il generatore di errori ***
1750 :
1760 print#15,"b-p"6;0:print#6,er$
1770 :
1780 rem* pilotaggio cursore e opzioni *
1790 :
1800 getx$
1810 ifx$="(left)"theno=o-31*(o<=0)+(o>0)
1820 ifx$="(rght)"theno=o-(o<31)+o*(o=>31)
1830 ifx$="(up)"thenv=v-7*(v<=0)+(v>0)
1840 ifx$="(down)"thenv=v-(v<7)+v*(v=>7)
1850 ifx$="h"thenlo=1885:gosub2960
1860 ifx$="s"thenlo=1948:gosub2430
1870 ifx$="l"thenlo=1908:gosub2210
1880 ifx$="r"thenlo=1965:gosub3490
1890 ifx$="m"thenlo=1845:gosub2630
1900 ifx$="n"thenlo=1788:gosub2840
1910 ifx$="t"thenlo=1805:gosub2920:gosub2770:gosub2920
1920 ifx$="+"thenlo=1828:se=se-(se<fnt(tr))+se*(se>fnt(tr)):gosub2260
1930 ifx$="-"thenlo=1868:se=se+(se>0)-fnt(tr)*(se<0):gosub2260
1940 ifx$="e"thenlo=1925:gosub3240:ife=4thengosub2920:sys49152
1950 ifti>tm+6thengosub2040
1960 gosub2160:on-(x$<>"f")goto1800
1970 :
1980 rem**** fine programma ****
1990 :
2000 close15:close8:close3:end
2010 :
2020 :rem*** flash ***
2030 :
2040 l=bs+v*40+o
2050 pokel,fnm(peek(l)):tm=ti
2060 on-(ti<tm+6)goto2060:pokel,fnm(peek(l)):tm=ti:return
2070 :
2080 rem** clear informazioni blocco **
2090 :
2100 gosub2120:printc2$:spc(5);"  ";spc(6);"   (rght) (rght)(rght)(rg
ht)(rght)  ";spc(6);"    (rght)   "
2110 sys49152:return
2120 printc1$:spc(31);"    (rght)   ":return
2130 :
2140 :rem*** aggiornamento riferimenti **
2150 :
2160 printc2$;spc(5);right$("  "+str$(v*32+o),3):pokel282,peek(bs+v*40
+o)
2170 printc2$;spc(14);right$("  "+str$(peek(bs+v*40+o)),3):return
2180 :
2190 rem******  link successivo  ******
2200 :
2210 gosub2920:ifts<1orts>35orss>fnt(ts)thengosub2920:return
2220 tr=ts:se=ss:goto2280
2230 :
2240 :rem*** lettura di un settore ****
2250 :
2260 gosub2920
2270 iftr<1ortr>35orse>fnt(tr)then2350
2280 gosub2100:gosub2390
2290 print#15,"u1:"8;0;tr;se:gosub2570:iferthen2350
2300 print#15,"m-r"chr$(22)chr$(0):get#15,i$:poke1287,asc(i$+z$)
2310 print#15,"m-r"chr$(23)chr$(0):get#15,i$:poke1288,asc(i$+z$)
2320 fori=0to7:forj=0to31:get#8,a$:a=asc(a$+z$):pokebs+i*40+j,a:nextj,
i
2330 ts=peek(bs):ss=peek(bs+1)
2340 printc2$;spc(32);right$(str$(ts),2);"(rght)";right$(" "+str$(ss),
3)
2350 gosub2920:o=0:v=0:return
```

$0400-$04FF: buffer 1
$0500-$05FF: buffer 2
$0600-$06FF: buffer 3
$0700-$07FF: buffer 4
$C100-$C44E: interprete comandi
$C44F-$C822: gestione directory, buffer e numero drive
$C823-$CAF7: routine comandi disco (S,D,N,C,R,)
$CAF8-$CDD1: routine comandi Memory, User e Block
$CDD2-$E4FB: gestione dei file, del disco e dei canali di comunicazione
$E4FC-$E609: tabella messaggi di errore
$E60A-$E77F: routine di gestione errori
$E780-$FF0F: routine di gestione dell'I/O, dell'Interrupt, della BAM e per la formattazione
$FF10-$FFFF: vettori user ed hardware

Per comunicare con l'esterno vi sono due Versatile Interface Adapter (VIA) che la CPU vede come locazioni di memoria: ciascuno di essi ne occupa 16. Il primo ($1800-$180F) controlla il bus seriale e agisce da tramite dal bus interno a quello esterno. I dati che inviamo o riceviamo dal drive transitano per questa porta.

Al secondo ($1C00-$1C0F) è affidato l'incarico di controllo dell'hardware: infatti pilota lo Stepper Motor, verifica la presenza del notch (la linguetta adesiva di protezione scrittura) e controlla la logica di lettura/scrittura delle testine. I segnali hardware sono trasmessi direttamente dal bus dati interno dei drive.

Ciascun VIA è dotato di temporizzatori (CLOCK) indirizzabili a seconda delle esigenze del DOS.

Vediamo in questa pagina e nella successiva lo schema dei VIA e la mappa delle locazioni più significative del 1541.

Schema dei VIA

VIA 6522 1 - Porta per il BUS seriale

$1800	Porta B (IEEE Port) data reg.
$1801	Porta A
$1802	Direzione dati Porta B
$1803	Direzione dati Porta A
$1804	Timer low
$1805	Timer high
$1806	Timer low latch
$1807	Timer high latch
$180B	Timer 1 ON/OFF
$180C	PCR - control register
$180D	Interrupt flags register
$180E	IRQ
PB 0	Dati in entrata (DATA IN)
PB 1	Dati in uscita (DATA OUT)
PB 2	Clock inserito (CLOCK IN)
PB 3	Clock escluso (CLOCK OUT)
PB 4	ATN A
PB 5,6	Indirizzo Periferica
CB 2	ATN IN

VIA 6522 2 - Pilotaggio motore e Controllo Testina di Lettura/Scrittura (R/W)

$1C00	Porta B, controllo Porta I/O
$1C01	Porta A, Porta di I/O
$1C02	Direzione dati Porta A
$1C03	Direzione dati Porta B (R/W)
$1C04	Timer 1 low
$1C05	Timer 1 high
$1C06	Timer 1 low latch
$1C07	Timer 1 high latch
$1C0B	Timer 1 ON/OFF
$1C0C	PCR - control register (I/O)
$1C0D	Interrupt Flags Register
$1C0E	IRQ (Interrupt ON/OFF - IER)
PB 0	Steps del motore
PB 1	Direzione della R/W Head
PB 2	Stepper Motor ON/OFF
PB 3	LED ON/OFF
PB 4	Flag di Protezione scrittura
PB 5,6	Densita' di registrazione
PB 7	Controllo di sincronismo
CA 1	Byte disponibile
CA 2	SOE

```
2360 :
2370 rem*** traccia e settore ***
2380 :
2390 printc1$;spc(31);" ";right$(str$(tr),2);"(rght) ";right$(str$(se
),2):return
2400 :
2410 :rem*** scrittura di un settore **
2420 :
2430 gosub2920:iftr<1ortr>35orse>fnt(tr)then2530
2440 cr=1-(tr>9):wn$=right$(str$(tr),cr):mc=2
2450 printc1$;spc(31);"    (left)(left)(left)";wn$:lc=1175:gosub2690:if
w>35then2450
2460 ifw=0thenon-(tr>0)gosub2390:gosub2920:return
2470 cr=1-(se>9):wn$=right$(str$(se),cr):tr=w:gosub2390
2480 printc1$;spc(35);"    (left)(left)(left)";wn$:lc=1179:gosub2690:if
w>fnt(tr)then2480
2490 se=-w*(cr>0)-se*(cr<1):gosub2390
2500 print#15,"b-p:"8;0:gosub2570
2510 fori=0to7:forj=0to31:print#8,chr$(peek(bs+i*40+j));:nextj,i
2520 print#15,"u2:"8;0;tr;se:gosub2570
2530 gosub2920:return
2540 :
2550 :rem**** disk  status ******
2560 :
2570 input#15,a$,b$,c$,d$:ifa$="00"thener=0:return
2580 sys49152:printc3$;"(down)(down)(rght)(rght)(rght)(rght)Errore (rv
s)"a$"(off) Traccia (rvs)"c$"(off) Settore (rvs)"d$
2590 printspc(20-len(b$)/2);"(down)(rvs)"b$:er=-1:tr=0:ts=0:on-(a$="70
")goto2000:return
2600 :
2610 :rem**** change value ******
2620 :
2630 lc=1278:mc=3:gosub2920:printc2$:spc(14);"   ":p=peek(1282):poke12
82,32
2640 gosub2680:w=-w*(cr>0)-p*(cr<1):pokebs+v*40+o,w:gosub2920:return
2650 :
2660 :rem**** input numerico < 255 ****
2670 :
2680 wn$="":cr=0
2690 getx$:x=asc(x$+z$):ifti>tm+10then1=lc+cr:gosub2050
2700 ifcr<mcandx>47andx<58thenwn$=wn$+x$:pokelc+cr,x:cr=cr+1
2710 ifx=20andcr>0thencr=cr-1:wn$=left$(wn$,cr):pokelc+cr,32
2720 ifx=13thenw=val(wn$):on-(w>255)goto2690:return
2730 goto2690
2740 :
2750 :rem**** input alfanumerico ******
2760 :
2770 getx$:ifti>tm+10thengosub2040
2780 x=asc(x$+z$):on-(x<1)goto2770:ifx=13thenreturn
2790 pokebs+v*40+o,x:o=o-(o<31)+o*(o=31):ifo=0thenv=v-(v<7)+v*(v=>7)
2800 gosub2160:goto2770
2810 :
2820 :rem***  new block ****
2830 :
2840 gosub2920
2850 printc1$;spc(31);"    ":lc=1175:mc=2:gosub2680:w1=w:ifw>35then2850
2860 ifw=0thenon-(tr>0)gosub2390:gosub2920:return
2870 printc1$:spc(35);"    ":lc=1179:mc=2:gosub2680:w2=w:ifse>fnt(tr)th
en2870
2880 tr=w1:se=w2:goto2270
2890 :
2900 rem*** evidenzia opzioni *******
2910 :
2920 fori=0to14:pokelo+i,fnm(peek(lo+i)):next:return
2930 :
2940 rem******** hard copy ************
2950 :
2960 gosub2920:open4,4:print#4:close4:ifst<>0then3060
2970 open4,4,7:print#4:print#4
2980 fori=0to17:print#4,chr$(15);
2990 forj=0to39:p=peek(1024+j+i*40)
3000 ifp>128thenp=p-128:print#4,"(rvs)";
3010 ifp=34thenprint#4,ap$;:goto3040
3020 p=p-(p>63andp<96)*32-(p<32orp>95)*64
3030 print#4,chr$(p);"(off)";
3040 nextj:print#4,chr$(8)
3050 nexti:print#4,chr$(15):print#4
3060 close4:gosub2920:return
3070 :
3080 :rem****  data varie  *******
3090 :
3100 data108,7,2,173,3,2,133,10,173,4,2,133,11,169,224,133,2,165,2,48
3110 data252,96,32,10,245,44,0,28,16,251,32,86,245,172,5,2,174,6,2,184
3120 data80,254,136,208,250,32,71,5,184,80,254,202,208,250,76,65,5,32,
71,5
3130 data170,168,32,185,253,32,0,254,76,105,249,169,255,141,3,28,173,1
2,28,41
3140 data31,9,192,141,12,28,169,0,141,1,28,96,0,0,0,0,0,0,0,0,3,3,22,5
,0,0,57
3150 data5,15,15,22,5,162,7,22,5
3160 data 8,128,135,128,135,128,128,15,146,169,107,133,251,169,5,133,2
52,169
3170 data 32,72,162,0,104,160,32,145,251,136,208,251,232,224,8,240,14
72,165
3180 data 251,105,40,133,251,144,234,230,252,76,13,192,162,0,189,57,19
2,233
3190 data30,32,210,255,232,224,22,208,243,96,0,49,48,48,48,189,189,60,
235
3200 data116,104,102,104,63,243,96,117,110,107,96,115,110,31
3210 :
3220 :rem****  errori di lettura ******
3230 :
3240 gosub2920:gosub2100:e=0:tr=0:printc3$;"(rght)(rght)(rght)(rght)(r
ght)Creazione Errori di Lettura
3250 print"(down)(rght)(rght)(rght)(rght)(rght)20   Manca la testa del
 blocco
3260 print"(rght)(rght)(rght)(rght)(rght)21   Manca il sincronismo
3270 print"(rght)(rght)(rght)(rght)(rght)22   Manca il blocco dati
3280 print"(rght)(rght)(rght)(rght)(rght)23   Manca il cecksum
3290 print"(rght)(rght)(rght)(rght)(rght)00   Fine
3300 getx$:e=e+(x$="(down)")*((e<4)-e*(e=4))+(x$="(up)")*(4*(e=0)-(e>0
))

3310 ifx$=chr$(13)thenon-(e<>4)gosub3380:o=0:v=0:ts=0:tr=0:gosub2120:r
eturn
3320 ifti>tm+6thengosub3350
3330 goto3300
3340 :
3350 poke1472+e*40,108:tm=ti
3360 on-(ti<tm+6)goto3360:poke1472+e*40,32:tm=ti:return
3370 :
3380 ps=1469+e*40:pokeps,peek(ps)+128:pokeps+1,peek(ps+1)+128
3390 printc1$:spc(31);"    ":lc=1175:mc=2:gosub2680:w1=w:ifw>35then3390
3400 ifw=0thenon-(tr>0)gosub2390:sys49152:gosub2920:return
3410 printc1$:spc(35);"    ":lc=1179:mc=2:gosub2680:se=w:ifse>fnt(tr)th
en3410
3420 tr=w1:gosub2390:sx=-(se-1)*(se>0)-fnt(se)*(se=0)
3430 print#15,"u1:"8;0;tr;se:gosub2570:iferthengosub2920:return
3440 print#15,"ud:"chr$(tr)chr$(sx)e$(e)
3450 print#15,"u1:"8;0;tr;se:sys49152:gosub2570:gosub2920:return
3460 :
3470 :rem*** ricerca blocchi errati ***
3480 :

3490 gosub2920:gosub2100:me=7:er=0:ne=0:fort=1to35:fors=0tofnt(t)
3500 printc1$;spc(31)" "right$(str$(t),2)"(rght) "right$(str$(s),2)
3510 print#15,"u1:"8;0;t;s:input#15,a$,b$,c$,d$
3520 ifa$<>"00"thengosub3600
3530 ifa$="21"thens=fnt(t)
3540 gety$:ify$="x"orne=30thens=fnt(t):t=35
3550 nexts,t:ifne+1=erorer=0then3580
3560 gosub3640:fori=ne+1toer:printspc(5);e$(i):ifi>=methengosub3640
3570 nexti
3580 gosub2120:o=0:v=0:gosub2920:return
3590 :
3600 e$(er)="(rvs)"+a$+"(off) "+b$+" (rvs)"+c$+"(rght)"+d$
3610 iferr<=methenprintleft$(c4$,11+er);spc(5);e$(er):ne=er
3620 er=er+1:return
3630 :
3640 poke198,0:wait198,1:sys49152:printc3$;:me=me+8:return
```

Mappa delle locazioni del 1541

```
$00         Comando per buffer 0
$01         Comando per buffer 1
$02         Comando per buffer 2
$03         Comando per buffer 3
$04         Comando per buffer 4
$05-$06     Traccia e settore buffer 0
$07-$08     Traccia e settore buffer 1
$09-$0A     Traccia e settore buffer 2
$0B-$0C     Traccia e settore buffer 3
$0D-$0F     Traccia e settore buffer 4
$12-$13     ID drive 0
$14-$15     ID drive 1
$16-$17     ID
$1C         Flag protezione scrittura (bit 4)
$20-$21     Flag per movimento testina
$22-$23     Traccia e settore di destinazione
$24-$2C     Buffer per l'Header nella ricerca del settore richiesto
$30-$31     Puntatore al buffer per Disk Controller (DC)
$39         Costante 8, identifica il Block Header
$3A         Segnale di Parita' per i dati del buffer
$3D         N. drive per DC
$3F         N. buffer per DC
$43         N. settori per la traccia durante la formattazione
$47         Costante 7, identifica il Data Block
$49         Stack Pointer
$4A         STEPS counter per R/W HEAD
$51         N. traccia durante la formattazione
$62-$63     Puntatore salti indiretti per DC
$65-$66     Puntatore per U9 (UI)
$69         AMPIEZZA intersector GAP (10)
$6A         N. tentativi di lettura (4)
$6D-$6E     Puntatore al buffer per la BAM
$6F-$70     Puntatore per comandi Memory o Block
$77         N. Periferica +32 per LISTEN  sul BUS seriale (IEEE)
$78         N. Periferica +64 per TALK (IEEE)
$79         Flag per LISTEN (IEEE)
$7A         Flag per TALK (IEEE)
$7C         Flag per ATN (ricezione dati da IEEE)
$7D         Flag per EOI (End Of InPut su IEEE)
$7F         N. di Periferica
$80         N. di traccia
$81         N. di settore
$82         N. di canale
$83         Indirizzo secondario
$84         Indirizzo secondario
$85         Byte di dati
$8B-$8D     Area di appoggio per divisione
$94-$95     Attuale Buffer Pointer
$99-$9A     Indirizzo buffer 0 ($0300)
$9B-$9C     Indirizzo buffer 1 ($0400)
$9D-$9E     Indirizzo buffer 2 ($0500)
$9F-$A0     Indirizzo buffer 3 ($0600)
$A1-$A2     Indirizzo buffer 4 ($0700)
$A3-$A4     Puntatore buffer di INPUT ($0200)
$A5-$A6     Puntatore buffer messaggi ($02D5)
$A7-$AD     Puntatori buffer per canale in $82 (tabella 1)
$AE-$B4     Puntatori buffer per canale in $82 (tabella 2)
$B5-$BA     Record basso, blocco basso
$BB-$C0     Record alto, blocco alto
$C1-$C6     Puntatore per scrittura file Relative
$C7-$CC     Lunghezza Record di un file Relative
$D4         N. Record attuale di un file Relative
$D5         N. di Side Sector (dati del Record)
$D6         Puntatore al Record nel Side Sector
$D7         Puntatore al Record nel file Relative
$E7         TIPO file (drive 0)
$E8         TIPO file (drive 1)
$F2-$F5     Tabella comandi per IEEE sul canale in $82
$F6         R/W flag (drive 0)
$F7         R/W flag (drive 1)
$F9         N. di buffer
$FF         Flag di errore (drive 0)
$100        Flag di errore (drive 1)
$101        DOS che ha formattato il floppy (65 -> 2A)
$100-$145   Area di stack
$200-$229   Buffer per stringa di comando
$22A        N. di parametri nella stringa di comando
$22B-$23B   Tabella dei canali
$24A        TIPO file
$24D        Codice di comando attuale ($80=READ, $90=WRITE...)
$24E        N. di settori per traccia
$251        Flag per BAM modificata (drive 0)
$252        Flag per BAM modificata (drive 1)
$253        Flag per file found
$255        Codice comando per IEEE
$258        Lunghezza record
$259-$25A   Traccia e Settore del Side Sector
$25B        Codice di comando per DC
$26C-$26D   Flags per errori
$274        Lunghezza stringa di input
$278        Puntatore nella stringa di comando al parametro in esame
$279        Flag per controllo di un file
$27A        Posizione nella stringa di comando dei due Punti (:)
$27B-$27F   Pos. ultimo byte di ogni parametro nella stringa di comando
$280-$284   Traccia di un file
$285-$289   Settore di un file
$28A        Flag per Wildcard (* o ?)
$28B        Flags per il check della linea di input
$28C        N. di drives
$28E        Ultimo drive utilizzato (0/1)
$291        N. settore
$292        Buffer Pointer per lettura directory
$293        N. settore per lettura della directory
$296        TIPO file (1-4)
$297        0=LOAD, 1=SAVE
$298        Flag di errore
$2B0-$2CB   Buffer per la Directory
$2D5-$2F9   Buffer per messaggi sullo STATUS del drive
$2FA-$2FC   N. blocchi liberi
```

N.B. Alcune delle locazioni della seconda pagina di memoria, quelle usate per la gestione dei file, possono cambiare di significato a seconda di che tipo di operazione venga eseguita.

La programmazione del 1541

Come detto in precedenza, il drive, o meglio il DOS del drive, fa largo uso delle tecniche di interrupt per svolgere contemporaneamente tutte le operazioni che gli competono.
La routine di Interrupt, oltre ad effettuare le operazioni precedentemente viste, va a controllare, tutte le volte che viene richiamata, se nelle prime cinque locazioni di memoria ($00-$04), ognuna corrispondente ad un ben determinato buffer, è presente un codice di comando: se lo trova, va ad eseguirlo non appena terminato il proprio lavoro (se di priorità superiore).

Questi codici sono:

Cod.	cmd	Descrizione
$80	(128)	Legge un Settore
$90	(144)	Scrive un Settore
$A0	(160)	Verifica un Settore
$B0	(176)	Cerca una Traccia
$B8	(184)	Cerca Traccia e Settore
$C0	(192)	Si posiziona sulla Traccia 1
$D0	(208)	Esegue il programma nel buffer
$E0	(224)	Si posiziona sul Settore richiesto ed esegue la routine LM nel buffer

Terminata l'operazione, il DOS riporterà, nella medesima locazione di comando che avevamo impostata, il codice identificativo dello STATUS:

Cod.	Descrizione
$01	Nessun errore
$02	Header Block non trovato
$03	Sincronismo non trovato
$04	Data Block non trovato
$05	Errore di Checksum nel Data Block
$06	Errore di decodifica di un byte
$07	Errore di verifica scrittura
$08	Errore, disco protetto da scrittura
$09	Errore di Checksum nell'Header Block
$0A	Blocco dati troppo lungo
$0B	Errore, ID errata

I numeri di traccia e settore vanno inseriti nelle locazioni corrispondenti al buffer da utilizzare (vedi mappa).

Per muovere la testina dalla prima traccia alla quarantesima (e non oltre pena il disallineamento della testina) possiamo usare questa piccola routine:

```
10 OPEN1,8,15,"I0": FOR I=1TO40
20 PRINT#1,"M-W"CHR$(5)CHR$(0)CHR$(1)CHR$(I):  REM IMPOSTA TRACCIA
30 PRINT#1,"M-W"CHR$(0)CHR$(0)CHR$(1)CHR$(176):REM MUOVE STEP. MOTOR
40 FOR J=1TO500: NEXTJ,I: CLOSE1
```

Supponiamo ora di voler caricare nel buffer 1 il Settore 11 della traccia 16 e di volervo verificare:

```
10 OPEN1,8,15
15 PRINT#1,"M-W"CHR$(7)CHR$(0)CHR$(2)CHR$(16)CHR$(11):REM TRK & SEC
20 PRINT#1,"M-W"CHR$(1)CHR$(0)CHR$(1)CHR$(128):      REM CMD LETTURA
25 PRINT#1,"M-R"CHR$(1)CHR$(0):                      REM STATUS
30 GET#1,A$:A=ASC(A$+CHR$(0)):IFA>127THEN25:         REM FINITO ?
35 IFA>1THENPRINT"ERRORE",A:CLOSE1:END
40 PRINT#1,"M-W"CHR$(1)CHR$(0)CHR$(1)CHR$(160):      REM CMD VERIFICA
45 PRINT#1,"M-R"CHR$(1)CHR$(0):                      REM STATUS
50 GET#1,A$:A=ASC(A$+CHR$(0)):IFA>127THEN45:         REM FINITO ?
55 IFA>1THENPRINT"ERRORE",A
60 CLOSE 1
```

Questo è un uso piuttosto banale di questa ulteriore possibilità che il 1541 offre. Ben di più si può fare imparando ad usare le porte di I/O, ed in particolare il VIA 2. Ciò però comincia ad implicare una buona conoscenza dell'assembler del 6052 e delle porte di Input/Output.

Vediamo come selezionare questa porta in lettura e scrittura:

```
Write Mode  0500 LDA #$FF
            0502 STA  $1C03 ; Porta A in OUTPUT
            0505 LDA  $1C0C
            0508 AND #$1F
            050A ORA #$C0
            050C STA  $1C0C ; PCR in modo scrittura
```

Dopodiché per scrivere un qualsiasi carattere su disco bisogna immetterlo nella porta I/O ($1C01). Il nuovo byte che verrà scritto sul disco sarà costituito non da 8 bit ma da 9: l'ulteriore bit è un bit di controllo e serve da riscontro per verificare che il byte sia stato registrato correttamente, è comunque un'informazione che viene totalmente gestita dal 1541.

Quando è stata completata la scrittura del byte viene messo in ON il bit di overflow del 6502.

Scriviamo, ad esempio, 3 volte $90 sul floppy:

```
0510 LDX #$03
0512 LDA #$90
0514 STA  $1C01 ; carattere da scrivere
0517 CLV
0518 BVC  $0518 ; byte completo ?
051A DEX
051B BNE  $0517
```

Attenzione però ad usare questa routine solo dopo aver posizionato la testina di registrazione al punto giusto, altrimenti...

```
Read Mode   0520 LDA  $1C0C
            0523 ORA #$E0
            0525 STA  $1C0C ; PCR in modo lettura
            0528 LDA #$00
            052A STA  $1C03 ; Porta A in INPUT
```

Possiamo passare in modo lettura anche effettuando un semplice JSR $FE00.

Per leggere 32 byte ed immetterli nel buffer 0 ($300) scriveremo:

```
0530 LDY #$20
0532 LDX #$00
0534 CLV
0535 BVC  $0535
0537 LDA  $1C01
053A STA  $0300,X
053D INX
053E DEY
053F BNE  $0534
```

Componiamo ora queste routine in modo da generare l'errore di lettura 20, diciamo, sul settore 3 della traccia 6. Dovremo pertanto cancellare uno o più caratteri del Block Header in modo che il DOS non sia più in grado di riconoscerlo.

Supponiamo di voler cancellare solo il terzo carattere dell'Header di questo settore. Il programma sarà costituito da due parti, una in BASIC, residente nel C64 con compito di pilota, ed una in LM, residente nel 1541 con il compito di esecutrice, LM$ sia la variabile stringa contenente la routine da scrivere nel drive:

```
10 OPEN1,8,15,"I0": OPEN6,8,6,"#0"
20 INPUT#1,ER$: IFER>19THEN90
30 PRINT#1,"B-P"6;0
40 PRINT#6,LM$;                                          REM LM ROUTINE
50 PRINT#1,"M-W"CHR$(5)CHR$(0)CHR$(2)CHR$(6)CHR$(2): REM TRK 6 & SEC 2
60 PRINT#1,"M-W"CHR$(0)CHR$(0)CHR$(1)CHR$(224):       REM CMD $E0
70 PRINT#1,"M-R"CHR$(0)CHR$(0)
80 GET#1,A$: A=ASC(A$): IFA>127THEN70:                   REM WAIT
90 PRINT A: CLOSE6: CLOSE1

La routine LM sara':

0300 JSR $FE00 ; seleziona modo lettura (Per Precauzione)
0303 JSR $F50A ; siamo nel'Header GAP del settore 2, la routine $F50A
                 Posiziona la testina all'inizio del suo blocco dati
0306 BIT $1C00
0309 BPL $0306 ; aspetta che la Control Port sia disponibile
030B JSR $F556 ; cerca il sincronismo del settore successivo (6)
030E LDX #$03  ; numero di bytes da ignorare
0310 CLV       ; azzera il bit di overflow
0311 BVC $0311 ; aspetta che un byte venga letto
0313 DEX
0314 BNE $0310 ; sono stati letti tutti ?
0316 LDA #$FF
0318 STA $1C03 ; seleziona la Porta A in OUTPUT
031B LDA $1C0C ; PCR
031E AND #$1F
0320 ORA #$C0
0322 STA $1C0C ; seleziona PCR in scrittura
0324 LDA #$00  ; carattere di cancelazione
0326 STA $1C01 ; I/O Port
0329 CLV
032A BVC $032A ; hai cancellato ?
032C JSR $FE00 ; seleziona il modo lettura
032F JMP $F969 ; rientra nel ciclo di lavoro del DOS
```

Se ora tenteremo di leggere il settore appena modificato, il DOS, dopo aver sbatacchiato un po' la testina sul fermo pensando ad un disallineamento, ci risponderà laconicamente «20,READ ERROR,2,6».

Per ottenere un errore 22 dovremo cancellare i caratteri dopo il ventesimo, mentre per il 23 quelli dopo il 50 circa.

Per ottenere il 21 il metodo più semplice è di cancellare una traccia o una buona porzione di essa. Per far ciò è comodo utilizzare la routine del DOS a $FDB9 che effettua due loop annidati: il registro X dovrà contenere il valore per il loop esterno mentre il registro Y per quello interno. Un altro modo, meno semplice, è di cancellare il sincronismo dell'Header del settore prescelto.

Gli errori di lettura 27 e 29 sono un po' più complessi da realizzare, presupponendo conoscenze estremamente approfondite delle tecniche e delle routine di formattazione. Per ottenere il primo su una traccia bisognerebbe formattarla adottando un diverso criterio di codifica del Checksum dell'Header, mentre, per ottenere il secondo dovremmo formattarla con un ID diverso. Tentativi di ottenere questi errori tramite semplici cancellazioni danno risultati evanescenti: qualche volta riesce ma nella maggior parte dei casi non si approda a nulla.

La precedente routine si può ottimizzare parametrizzandola, in modo da non doverla riscrivere ogni volta che si voglia creare un nuovo errore. La stringa di comando, come abbiamo visto nella mappa del 1541, viene memorizzata nel buffer ad essa riservato, residente da $200 a $228, pertanto in esso verranno immessi i nuovi parametri.

La versione migliorata è:

```
10 OPEN 1,8,15,"I0"
20 OPEN 6,8,6,"#2"
30 PRINT#1,"B-P"6;0
40 PRINT#6,LM$
50 L1=3 :        REM    LOOP 1
55 L2=4 :        REM    LOOP 2
60 TR=6 :        REM   TRACCIA
65 SE=2 :        REM SETTORE-1
70 CMD = 224 :   REM   COMANDO
80 PRINT#6,"UD:"CHR$(TR)CHR$(SE)CHR$(CMD)CHR$(L1)CHR$(L2)
90 CLOSE6: CLOSE1


0500 JMP $0516
0503 LDA $0203
0505 STA $0A    ; TRACCIA
0507 LDA $0204
050A STA $0B    ; SETTORE
050C LDA $0205
050F STA $02    ; COMANDO
0511 LDA $02    ; STATUS
0513 BMI $0511  ; FINITO ?
0515 RTS
0516 JSR $F50A
0519 BIT $1C00
051C BPL $0519
051E JSR $F556 ; Cerca il carattere di sincronismo
0521 LDY $0205 ; LOOP 1
0524 LDX $0206 ; LOOP 2
0527 CLV
0528 BVC $0528
052A DEY
052B BNE $0527
052D LDA #$FF
052F STA $1C03
0532 LDA $1C0C
0535 AND #$1F
0537 ORA #$C0
0539 STA $1C0C
053C LDA #$00
053E STA $1C01
0541 CLV
0542 BVC $0542
0544 DEX
0545 BNE $0541
0546 JSR $FE00
0544 JSR $F969
```

Abbiamo così visto un altro modo di usare i comandi USER, così da poter inviare più parametri di quelli previsti sfruttando contemporaneamente la possibilità che hanno di effettuare dei salti all'interno del buffer #2.

MC

QUEL CHE SI DICE AVERE LE SPALLE AL COPERTO

Il primo personal computer, l'Apple][, giunge in Italia nel 1979.
Nel 1979 si costituisce la Automazione Sistemi Elettronici Microcomputers sdf, con l'intento di sviluppare la propria presenza principalmente nel mercato dell'elettronica industriale. Aver scommesso nell'informatica già da subito, seguendo l'onda montante dei primi PET COMMODORE, APPLE][e dell'M20 OLIVETTI fa della ASEM, nel frattempo trasformata in S.p.A., una delle aziende più "vecchie" nel settore. Un'azienda a cui di "rumore" piace farne con le cose, con i risultati piuttosto che con le chiacchiere. Vi presentiamo pertanto la ASEM di oggi, risultato di indubbi successi di vendita, di capacità imprenditoriale e tecnica: un'azienda dal futuro sicuro.
La progettazione della piastra elettronica, la realizzazione del master e degli impianti da cui si ottiene il circuito stampato, la successiva saldatura dei componenti, le fasi di collaudo sono gli "steps" che il prodotto compie prima di essere immesso sul mercato o di essere avviato all'assemblaggio dei computers.
Nella scelta dei componenti si vincono e si perdono molte battaglie. Alla ASEM abbiamo salda una convinzione: il costo di un computer non è il prezzo che si paga: guasti, malfunzionamenti e inaffidabilità sono cose che devono essere messe in bilancio al momento dell'acquisto, come la serietà del produttore. È per questo che utilizziamo solo drives TOSHIBA, dischi fissi della NEC, tastiere CHERRY e PREH, alimentatori switching costruiti in conformità alle normative europee su nostre specifiche da una primaria azienda italiana.

Una fase del collaudo delle schede prima del montaggio dei computers. Sotto, una fase della realizzazione del master.

La forza vendite Asem è presente, in Italia, in quasi tutte le regioni: è attualmente allo studio la realizzazione di centri territoriali di assistenza tecnica.

Piemonte/Val d'Aosta	Mar Due Snc	011/3290769
Lombardia	PC Plus Srl	02/2841544/5
Veneto/Trentino A.A.	ECO Srl	045/916476
Friuli V.G.	Lucio Rodaro	0432/962282
Emilia Romagna	Marco Giannasi	0522/294805
Toscana/Umbria	G. Presentini e C.	055/973151
Lazio	Luigi Ricci	06/6237040
Campania/Calabria	S. Barbagallo	081/414994
Puglia/Basilicata	N.R. Cavallo	080/330499

æ ASEM

Asem spa
Zona artigianale - Buia (UD)
telef. 0432/962282 telex 450608

COME USARCI AL MEGLIO

La espansione territoriale della ASEM in Italia e la costante crescita del numero di clienti e di fatturato ci ha imposto l'introduzione di alcuni aspetti organizzativi che devono essere conosciuti dai nostri clienti affinchè essi possano trarre il massimo dei vantaggi da questi nuovi servizi.

Ordini
Per l'inoltro degli ordini, i signori rivenditori quando non vogliano contattare l'Agente di zona, possono trasmetterli direttamente all'Ufficio Vendite, che provvede inoltre all'invio della documentazione anche agli utenti finali avendo cura di indicare il punto vendita più vicino.

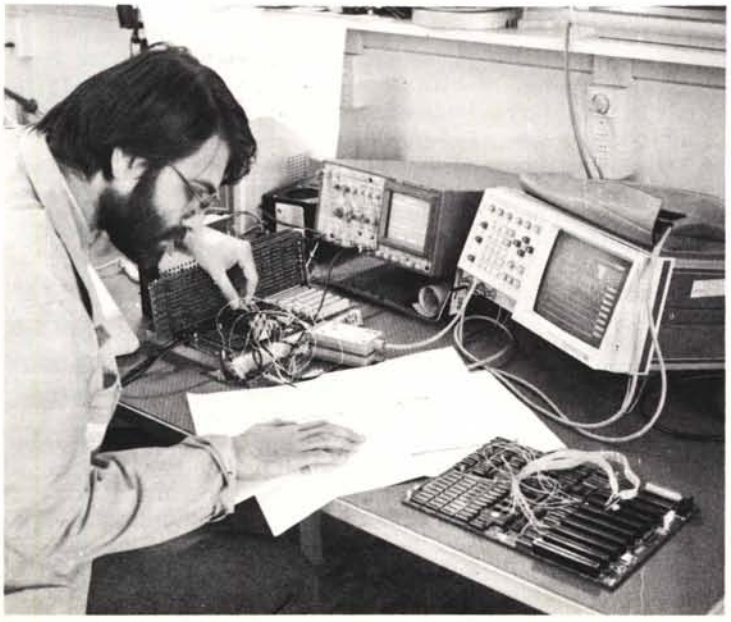
Hot Line
Una hot line facente capo all'Ufficio Tecnico è a disposizione per risolvere ogni problema di natura hardware. La hot line è attiva il pomeriggio e risponde al numero 0432/961014.

Assistenza
Sia le parti meccaniche sia le parti elettroniche sono soggette a guastarsi: è nell'ordine delle cose. Ciò che importa è avere le risorse tecniche ed umane in grado di intervenire in tempi accettabili; a tale fine va contattato l'Ufficio Assistenza.

Spedizioni
ASEM si è sempre distinta per la celerità e la flessibilità nell'evasione degli ordini (di questa stessa opinione sono i clienti che ci seguono da anni); ciò non vuol dire che alle volte non serva sollecitare una spedizione o una riparazione che tarda.

Telefonateci o veniteci a trovare: imparerete a chiamarci per nome.

ASEM
LO
STANDARD
E LA
DIFFERENZA



THOMPSON È UNA MANNA PER GLI OCCHI

Cinque sono i modelli che compongono la gamma di monitor a colori della THOMPSON. Le caratteristiche comuni a tutti i modelli sono l'utilizzo di CRT di alta qualità antiriflesso e con sfondo nero per far meglio risaltare colori e caratteri.
Due sono i modelli di monitor a colori; il CM31311 SIR ed il CM31481 SIR, entrambi a 12". Si distinguono per la capacità risolutiva del tubo: il primo modello ha capacità pari a 0.31 mm DOT PITCH e 14 MHz di banda passante, mentre il secondo a 0.48 mm DOT PITCH e stessa banda passante. Entrambi vengono pilotati dall'adattatore grafico colore del PC IBM. I due modelli a 14", CM36382 SIR e CM36512 VPIR, anch'essi si distinguono per la definizione del tubo, rispettivamente di 0.38 mm e di 0.51 mm DOT PITCH, con la stessa banda passante di 12 MHz. I dati relativi alla risoluzione sono analoghi ai 2 modelli a 12". Per la scheda ENHANCED GRAPHIC ADAPTER prodotta dalla IBM, THOMPSON ha progettato un modello specifico in grado di rispondere al meglio alle caratteristiche grafiche dell'adattatore. Riesce pertanto a visualizzare 650 H per 350 V punti con ben 64 colori possibili, disponendo di sincronismi automatici a 15.7 e a 22.0 KHz.