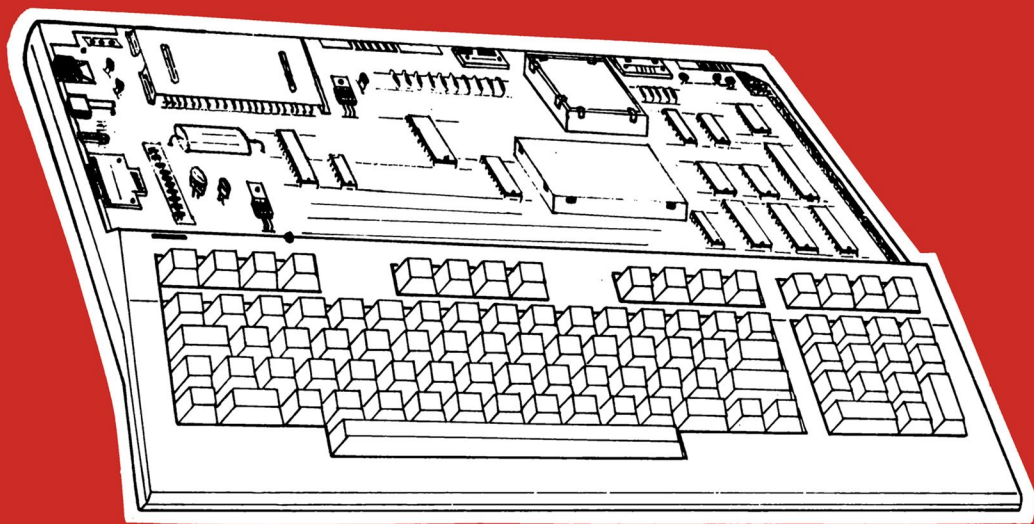


# IL SISTEMA OPERATIVO DEL commodore 128



EVM-COMPUTER









# **IL SISTEMA OPERATIVO DEL commodore 128**



**EVM-COMPUTER**

E.SOFT

Via IV Novembre 23

52025 MONTEVARCHI (AR)

EVM COMPUTERS SRL

Via Marconi 9/A

52025 MONTEVARCHI (AR)

Copyright 1986 by E.SOFT

Copyright 1986 by EVM COMPUTERS

Tutti i diritti sono riservati. Nessuna parte di questo manuale può essere riprodotta o posta in sistemi di archiviazione elettronici, meccanici o fotocopiata senza autorizzazione scritta.

I Edizione Ottobre 1986

## **INTRODUZIONE**

I motivi che hanno decretato il successo del COMMODORE 64 sono molto numerosi ed in parte almeno anche abbastanza noti.

Non possiamo qui elencarli tutti ma possiamo fissare la nostra attenzione almeno su tre motivazioni principali.

Il primo motivo e' che la Commodore era PRESENTE sul mercato nel momento in cui esplodeva, piu' o meno disordinatamente una domanda informatica sotto tutti i punti di vista delle applicazioni e da tutti gli strati sociali. Il motivo specifico del successo del C 64, o almeno uno dei piu' importanti, e' stata la risposta che appunto questo computer dava con il suo rapporto prezzo prestazioni.

Ma un motivo da non dimenticare e che d'altro canto ha segnato la morte piu' o meno prematura di altri computer, ed in qualche caso magari migliori, e' stato la straordinaria disponibilita' di SOFTWARE sia come numero di pacchetti che come prezzi.

Cio' in particolare e' stato possibile grazie anche alla notevole trasparenza della macchina con la messe di informazioni piu' o meno direttamente disponibile. In altre parole la copiosa documentazione predisposta da numerose case editrici. Queste informazioni sia a livello HARDWARE che SOFTWARE sono state anche il frutto della ricca collaborazione fra la EVM e la Commodore stessa.

Piu' lento, in verita' anche per altri motivi, il successo del C128, che pur

restando ad un ottimo livello come rapporto PREZZO/PRESTAZIONI, non ha mai avuto un gran che di programmi a disposizione oltre al relativo KNOW-HOW software ed hardware.

Il C128, in particolare ai prezzi attuali e' invece un computer di tutto rispetto, addirittura superiore al C64 come rapporto prezzo/prestazioni. In questo nuovo computer sono infatti stati messi insieme il notissimo C64, il nuovo C128 mentre l' utilizzo dello Z80 come coprocessore permette di usare i programmi in CP/M.

Malgrado la possibilita' di impiego del CPM, almeno sul nostro mercato, i modi piu' interessanti di lavoro sono il modo 64 ed il modo 128. Ora per operare, se non approfonditamente, ma almeno con una certa conoscenza di causa, su questi due computer e' necessario conoscerne il Sistema Operativo.

Il Sistema Operativo del C64 e' gia' stato pubblicato da noi ed ha riscosso un lusinghiero successo. Quello del C128 lo trovate nelle pagine seguenti e per di piu' e' la prima volta che queste notizie vengono pubblicate in italiano.

Questo libro esce con notevole ritardo rispetto ai tempi previsti sia per problemi tipografici che di messa a punto veri e propri, ce ne scusiamo con gli utenti come ci scusiamo subito degli errori e delle imperfezioni. Restiamo sempre a loro disposizione per ogni chiarimento.

## LE ROM KERNAL

L' intero Sistema Operativo del C128, almeno quando funziona in questo modo, e' residente su ROM come del resto e' tipica consuetudine della COMMODORE ed e' composto da ben 48 K Bytes, se si includono anche i 4K bytes della ROM relativa alla gestione dello Z-80, e risulta quindi diviso come segue:

|    |         |                   |
|----|---------|-------------------|
| 4  | K Bytes | per lo Z-80       |
| 28 | K       | " per il Basic    |
| 16 | K       | " per le routines |

Solo quest' ultima parte di memoria fa parte del presente manuale oltre ad una parte della ROM relativa alla gestione dello Z-80. Abbiamo impiegato un programma particolare per disassemblare quanto troverete nelle pagine seguenti e tutte le informazioni possibili per commentarle. In particolare per collegare con la Pagina zero (vedi descrizioni successive), anche questa ampiamente documentata. Riportiamo infatti oltre al commento le LABELS di riferimento in modo tale che il collegamento sia il migliore possibile e soprattutto facile e veloce.

Ci auguriamo che le abbreviazioni riportate siano di facile ed immediata comprensione essendo tutte in italiano. RTS sta per Return From Subroutine non e' commentata ed indica appunto la fine di una routine. Gli asterischi stanno a significare che continua l' operazione presentata nelle linee precedenti. Per chi e' alle prime armi in fatto di linguaggio macchina si consiglia il CORSO DI ASSEMBLER II PER C 64 che potra' essere utilmente impiegato anche in questo caso.

## VETTORI INGRESSO MONITOR

```

b000: 4c 21 b0 jmp * $b021
b003: 4c 09 b0 jmp * $b009
b006: 4c b2 b0 jmp * $b0b2
b009: 20 7d ff jsr * $ff7d [ primm ]
-----

```

```

b00c: 0d 42 52 45 41 4b 07 00
-----

```

## INIZIALIZZAZIONE MONITOR

```

b014: 68 pla
b015: 85 02 sta $02 [ bank ]
b017: a2 05 ldx #$05
b019: 68 pla
b01a: 95 03 sta $03,x [ pc-hi ]
b01c: ca dex
b01d: 10 fa bpl * $b019
b01f: 30 25 bmi * $b046
-----

```

## INIZIALIZZAZIONE PER INGRESSO NORMALE

```

b021: a9 00 lda #$00
b023: 8d 00 ff sta * $ff00
b026: 85 06 sta $06 [ a-reg ]
b028: 85 07 sta $07 [ x-reg ]
b02a: 85 08 sta $08 [ y-reg ]
b02c: 85 05 sta $05 [ s-reg ]
b02e: a9 00 lda #$00
b030: a0 b0 ldy #$b0
b032: 85 04 sta $04 [ pc-lo ]
b034: 84 03 sty $03 [ pc-hi ]
b036: a9 0f lda #$0f
b038: 85 02 sta $02 [ bank ]
b03a: 20 7d ff jsr * $ff7d [ primm ]
-----

```

## INIZIALIZZAZIONE MONITOR (GENERALE)

```

b03d: 0d 4d 4f 4e 49 54 4f 52
b045: 00
-----

```

```

b046: d8 cld
b047: ba tsx
b048: 86 09 stx $09 [ charac ]
b04a: a9 c0 lda #$c0
b04c: 20 90 ff jsr * $ff90 [ setmsg ]
b04f: 58 cli
b050: 20 7d ff jsr * $ff7d [ primm ]
-----

```

```

b053: 0d 20 20 20 20 50 43 20 CR PC
b05b: 20 53 52 20 41 43 20 58 SR AC X
b063: 52 20 59 52 20 53 50 0D R YR SP CR
b06b: 3B 20 1B 51 00 ;, <esc> q
-----

```

## VISUALIZZAZIONE CONTENUTO REGISTRI (MONITOR)

Ingresso monitor  
 Ingresso BREAK per monitor  
 Ingresso per Exmon  
 Output di stringa per PRINT  
 Messaggio iniziale del monitor. Segue  
 il BREAK  
 <cr> break <bell>

Immette il n. del Banco di memoria su  
 lo Stack in Pagina Zero  
 Questa parte della routine immette va  
 lori dei registri X e Y, Accumulatore  
 PS e PC nei corrispondenti Bytes di  
 pagina ZERO prelevandoli dallo STACK.  
 \*\*\*  
 Inizializzazione

Carica i registri con \$00 ed abilita  
 tutta la parte ROM.  
 Esegue un CLEAR in P.0 per Acc.  
 Come sopra per X  
 Come sopra per Y  
 Come sopra per PS  
 Carica A con LO/ADDR per il Monitor  
 Carica Y con HI/ADDR per il Monitor  
 Contatore di A in Mem.: PC LO  
 Contatore di Y in Mem.: PC HI  
 Fissa in Pag. 0 il num. di Banco di  
 a \$0F, Kernal, Basic, RAM 0, I/O  
 Vai a subr. PRINT  
 Messaggio per il MONITOR

<cr> monitor

Reset del modo Decimale  
 Immagazzina il puntatore dello STACK  
 in X ed in memoria.  
 Abilita i messaggi di controllo  
 Salta all subr. SETMSG  
 Abilitati tutti i sistemi di INTERRUPT  
 Salta alla subr. PRINT

Costanti per il monitor

|                |                |           |  |
|----------------|----------------|-----------|--|
| b070: a5 02    | lda \$02       | [ bank ]  | Immette il n. di BANCO in A            |
| b072: 20 d2 b8 | jsr * \$b8d2   |           | A= 2 byte ASCII: HI=A, LO=X            |
| b075: 8a       | txa            |           | Immette in A val. ASCII per Nib. LO    |
| b076: 20 d2 ff | jsr * \$ffd2   | [ bsout ] | Salta a BSOUT                          |
| b079: a5 03    | lda \$03       | [ pc-hi ] | PC-HI in Accumulatore                  |
| b07b: 20 c2 b8 | jsr * \$b8c2   |           | Converti A in 2 Bytes ASCII            |
| b07e: a0 02    | ldy #\$02      |           | Metti PC LO                            |
| b080: b9 02 00 | lda \$0002,y   |           | Metti P,A,X,Y,S, in A                  |
| b083: 20 a5 b8 | jsr * \$b8a5   |           | Uscita in A                            |
| b086: c8       | iny            |           | Incrementa la visualizzazione          |
| b087: c0 08    | cpy #\$08      |           | Controlla l'uscita da \$04 a \$09      |
| b089: 90 f5    | bcc * \$b080   |           | Legge il prossimo Byte.                |
| b08b: 20 b4 b8 | jsr * \$b8b4   |           | Lineefeed+ CLEAR al resto della linea  |
| b08e: a2 00    | ldx #\$00      |           | Piazza il puntatore.                   |
| b090: 86 7a    | stx \$7a       | [ dsdec ] | Il Buffer input = 0:****               |
| b092: 20 cf ff | jsr * \$ffcf   | [ basin ] | Va a Subr BASIN: leggi un carattere    |
| b095: 9d 00 02 | sta \$0200,x   |           | e metti in Buffer di ingresso Monitor  |
| b098: e8       | inx            |           | Visualizza l' incremento sul Buffer    |
| b099: e0 a1    | cpx \$a1       |           | Controllo per stampa di 160 Caratteri  |
| b09b: b0 1f    | bcs * \$b0bc   |           | Controllo prec. positivo = mess. err.  |
| b09d: c9 0d    | cmp #\$0d      |           | Controlla se prem. RETURN              |
| b09f: d0 f1    | bne * \$b092   |           | Controllo =no attendi prossimo caratt. |
| b0a1: a9 00    | lda #\$00      |           | Se il RETURN e' stato premuto, allora  |
| b0a3: 9d ff 01 | sta \$01ff,x   |           | la stringa comando termina con \$00    |
| b0a6: 20 e9 b8 | jsr * \$b8e9   |           | Controllo di Buffer Input per \$00     |
| b0a9: f0 e0    | beq * \$b08b   |           | Se contr. : o ? aspetta un Input.      |
| b0ab: c9 20    | cmp #\$20      |           | Control. se il caratt e' uno Spazio    |
| b0ad: f0 f7    | beq * \$b0a6   |           | Legge il successivo carattere          |
| b0af: 6c 2e 03 | jmp (\$032e)   |           | Vettore per routine MONITOR            |
| b0b2: a2 15    | ldx #\$15      |           | Il numero delle parole chiave in X e'  |
| b0b4: dd e6 b0 | cmp * \$b0e6,x |           | confrontato con la TAVOLA delle KEYW   |
| b0b7: f0 0c    | beq * \$b0c5   |           | Se e' trovata vala puntatore della tav |
| b0b9: ca       | dex            |           | la, lo decrementa di 1 fino al termine |
| b0ba: 10 f8    | bpl * \$b0b4   |           | della ricerca                          |
| b0bc: 20 7d ff | jsr * \$ff7d   | [ primm ] | Salto a subr. PRINT e uscita.          |

b0bf: 1d 3f 00 ?

Cursore a destra

#### FISSA INDIRIZZI PER COMANDI MONITOR

|                |                |           |  |
|----------------|----------------|-----------|--|
| b0c2: 4c 8b b0 | jmp * \$b08b   |           | Salta al ciclo di attesa di input      |
| b0c5: e0 13    | cpx \$13       |           | Controlla se e' L S o V                |
| b0c7: b0 12    | bcs * \$b0db   |           | Se si esegui                           |
| b0c9: e0 0f    | cpx \$0f       |           | Controllo se e' un car.di conv. \$+&%  |
| b0cb: b0 13    | bcs * \$b0e0   |           | Se si esegui                           |
| b0cd: 8a       | txa            |           | N. parola chiave in A                  |
| b0ce: 0a       | asl a          |           | Moltiplica A x2                        |
| b0cf: aa       | tax            |           | Sposta il valore in X                  |
| b0d0: bd fd b0 | lda * \$b0fd,x |           | Le due routines spostano gli indirizzi |
| b0d3: 48       | pha            |           | HI e LO nello STACK.                   |
| b0d4: bd fc b0 | lda * \$b0fc,x |           |  |
| b0d7: 48       | pha            |           |  |
| b0d8: 4c a7 b7 | jmp * \$b7a7   |           | Salta ai parametri di impiego          |
| b0db: 85 93    | sta \$93       | [ verck ] | Immag. il carattere del comando        |
| b0dd: 4c 37 b3 | jmp * \$b337   |           | Esegue un comando L, S o V             |
| b0e0: 4c b1 b9 | jmp * \$b9b1   |           | Esegue la conversione dei caratteri    |
| b0e3: 6c 00 0a | jmp (\$0a00)   |           | Vettore di WARM START del BASIC        |

b0e6: 41 43 44 46 47 48 4a 4d acdfghjm Parole chiavi accettate dal MONITOR  
 b0ee: 52 54 58 40 2e 3e 3b 24 rtx@.>;\$  
 b0f6: 2b 26 25 4c 53 56 +&%lsv

## INDIRIZZI DEI COMANDI DEL MONITOR

b0fc: 05 b4 (\$b405) A = ASSEMBLE  
 b0fe: 30 b2 (\$b230) C = COMPARE  
 b100: 98 b5 (\$b598) D = DISASSEMBLE  
 b102: da b3 (\$b3da) F = FILL  
 b104: d5 b1 (\$b1d5) G = GO TO  
 b106: cd b2 (\$b2cd) H = HUNT  
 b108: de b1 (\$b1de) J = JUMP  
 b10a: 51 b1 (\$b151) M = MONITOR  
 b10c: 4f b0 (\$b04f) R = REGISTER  
 b10e: 33 b2 (\$b233) T = TRANSFER  
 b110: e2 b0 (\$b0e2) X = EXIT  
 b112: 8f ba (\$ba8f) @ = DISC COMMAND  
 b114: 05 b4 (\$b405) . = ASSEMBLE  
 b116: aa b1 (\$b1aa) > = MODIFY MEMORY  
 b118: 93 b1 (\$b193) ; = MODIFY REGISTER

b11a: 8e b2 0a stx \$0ab2 Imm. temp. in X  
 b11d: a6 68 ldx \$68 [ facsgn ] N. banco rilevato da OP3  
 b11f: a9 66 lda #\$66 Indirizzo FETVEC in A  
 b121: 78 sei Disabilitaz. di TUTTI gli INTERRUPT  
 b122: 20 74 ff jsr \* \$ff74 [ indfet ] Salta a sub. INDFET  
 b125: 58 cli Abilitati tutti gli INTERRUPT  
 b126: ae b2 0a ldx \$0ab2 X caricato con il valore salv. prec.  
 b129: 60 rts  
 b12a: 8e b2 0a stx \$0ab2 Imm. temp. X.  
 b12d: a2 66 ldx #\$66 Carica STAVEC (ind LO) in X e inseris.  
 b12f: 8e b9 02 stx \$02b9 la rout. Indsta in STAVEC  
 b132: a6 68 ldx \$68 [ facsgn ] Rileva il n. banco da OP3  
 b134: 78 sei Disabilitaz. di TUTTI gli INTERRUPT  
 b135: 20 77 ff jsr \* \$ff77 [ indsta ] Salta a sub. INDSTA  
 b138: 58 cli Abilitati tutti gli INTERRUPT  
 b139: ae b2 0a ldx \$0ab2 X caricato con il valore salvato prec.  
 b13c: 60 rts

b13d: 8e b2 0a stx \$0ab2 Imm. tempo. X  
 b140: a2 66 ldx #\$66 Carica indirizzo CMPVEC in Y e valore  
 b142: 8e c8 02 stx \$02c8 CMPVEC per INDCMP  
 b145: a6 68 ldx \$68 [ facsgn ] Rileva il n. banco da OP3  
 b147: 78 sei Disabilit. di tutti gli INTERRUPT  
 b148: 20 7a ff jsr \* \$ff7a [ indcmp ] Salta a INDCMP  
 b14b: 58 cli Abilita tutti gli INTERRUPT  
 b14c: 08 php Risultato del confronto (CMP)  
 b14d: ae b2 0a ldx \$0ab2 Carica X con val.precedente  
 b150: 28 plp Riporta il risultato del confronto  
 b151: 60 rts

## COMANDO MONITOR M

b152: b0 08 bcs \* \$b15c Nessun parametro. Valori di default  
 b154: 20 01 b9 jsr \* \$b901 Copia contenuto OP! in OP#  
 b157: 20 a7 b7 jsr \* \$b7a7 Immetti in OPl  
 b15a: 90 06 bcc \* \$b162 Esegui conversione per passo  
 b15c: a9 0b lda #\$0b Carica OPl (ind. LO) con val. DEFAULT



|                |              |            |  |
|----------------|--------------|------------|--|
| bl5e: 85 60    | sta \$60     | [ tenexp ] | Carica S.C. 12                         |
| bl60: d0 15    | bne * \$b177 |            | Vai ad eseg. visualizz. memoria        |
| bl62: 20 0e b9 | jsr * \$b90e |            | Poni OP1-OP3 in OP1                    |
| bl65: 90 2a    | bcc * \$b191 |            | Se OP1-OP3> OP1 ERRORE                 |
| bl67: a2 03    | ldx #\$03    |            | Dividi 3 volte per 2 n. passi          |
| bl69: 24 d7    | bit \$d7     | [ mode ]   | Controllo per modo 40/80 colonne       |
| bl6b: 10 01    | bpl * \$b16e |            | 40 colonne (passo di divisione)        |
| bl6d: e8       | inx          |            | 80 colonne (numero di passi)           |
| bl6e: 46 62    | lsr \$62     | [ expsgn ] | Divisione di OP1 per 2 per la visualiz |
| bl70: 66 61    | ror \$61     | [ lowtr ]  | zazione dei valori in memoria di 8 o   |
| bl72: 66 60    | ror \$60     | [ tenexp ] | 16                                     |
| bl74: ca       | dex          |            | N. di divisioni per passo -1           |
| bl75: d0 f7    | bne * \$b16e |            | OP1 diviso per 8 o 16                  |
| bl77: 20 e1 ff | jsr * \$ffef | [ stop ]   | Vai a subr STOP: Contr. tasto di STOP  |
| bl7a: f0 12    | beq * \$b18e |            | Se STOP premuto vai a subr EXIT.       |
| bl7c: 20 e8 b1 | jsr * \$ble8 |            | Visualizza 1 linea di memoria          |
| bl7f: a9 08    | lda #\$08    |            | Carica A con operando                  |
| bl81: 24 d7    | bit \$d7     | [ mode ]   | Controllo per modo 40/80 colonne       |
| bl83: 10 01    | bpl * \$b186 |            | 40 col. aggiungi costante di 8.        |
| bl85: 0a       | asl a        |            | 80 col. aggiungi costante di 16        |
| bl86: 20 52 b9 | jsr * \$b952 |            | Somma: contenuto di A con OP3          |
| bl89: 20 22 b9 | jsr * \$b922 |            | Sottrazione: OP1 - costante <1>        |
| bl8c: b0 e9    | bcs * \$b177 |            | Ciclo fino a OP1<0                     |
| bl8e: 4c 8b b0 | jmp * \$b08b |            | Salta a ciclo di attesa Input          |
| bl91: 4c bc b0 | jmp * \$b0bc |            | Stampa ? e vai come sopra              |

## COMANDO MONITOR ;

|                |              |            |                                     |
|----------------|--------------|------------|-------------------------------------|
| bl94: 20 74 b9 | jsr * \$b974 |            | C=0 OP1 in pag. 0                   |
| bl97: a0 00    | ldy #\$00    |            | Fissa lo spost. per la Pag. 0       |
| bl99: 20 a7 b7 | jsr * \$b7a7 |            | Modifica OP1                        |
| bl9c: b0 0a    | bcs * \$bla8 |            | Carry set= ident. per sub. EXIT     |
| bl9e: a5 60    | lda \$60     | [ tenexp ] | Carica indirizzo di OP1             |
| bla0: 99 05 00 | sta \$0005,y |            | Puntatore di Status per B,A,X,Y     |
| bla3: c8       | iny          |            | Visualizza memoria CPU +1 in Pag.0  |
| bla4: c0 05    | cpy #\$05    |            | Controllo per var. mem. CPU         |
| bla6: 90 f1    | bcc * \$b199 |            | Se contr. negat. vai prossima rout. |
| bla8: 4c 8b b0 | jmp * \$b08b |            | Vai al ciclo di attesa input.       |

## COMANDO MONITOR DI MODIFICA MEMORIA

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| blab: b0 1c    | bcs * \$blc9 |            | Nessun param. quindi nessun cambiam.  |
| blad: 20 01 b9 | jsr * \$b901 |            | Copia il cont di OP1 in OP3           |
| blb0: a0 00    | ldy #\$00    |            | Metti a 0 il puntat. della vis. modif |
| blb2: 20 a7 b7 | jsr * \$b7a7 |            | Metti il valor modif in OP1           |
| blb5: b0 12    | bcs * \$blc9 |            | Se nessun altro val stampa la linea   |
| blb7: a5 60    | lda \$60     | [ tenexp ] | Prendi il valore da OP1               |
| blb9: 20 2a b1 | jsr * \$b12a |            | Immag A in ogni banco                 |
| blbc: c8       | iny          |            | Visual. punt per modif. Byte + 1      |
| blbd: 24 d7    | bit \$d7     | [ mode ]   | Controllo per modo 40/80 colonne      |
| blbf: 10 04    | bpl * \$blc5 |            | Massimo param. di lett.40 caratteri   |
| blcl: c0 10    | cpy #\$10    |            | Controllo variaz su 16 caratteri      |
| blc3: 90 ed    | bcc * \$blb2 |            | Se nessun cambiam. prossimo parametro |
| blc5: c0 08    | cpy #\$08    |            | Controllo variaz.su 8 caratteri       |
| blc7: 90 e9    | bcc * \$blb2 |            | Se nessun cambiamento prossimo param. |
| blc9: 20 7d ff | jsr * \$ff7d | [ primm ]  | Vai alla subr. di PRINT               |

blcc: 1b 4f 91 00 <esc> o <crsr up>

Esegui un Clear sui modi riportati

## COMANDI MONITOR G &amp; J

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| bld0: 20 e8 b1 | jsr * \$ble8 |            | Vis. delle variazioni                 |
| bld3: 4c 8b b0 | jmp * \$b08b |            | Vai al ciclo di attesa input          |
| bld6: 20 74 b9 | jsr * \$b974 |            | C=0 OP1 in pagina zero                |
| bld9: a6 09    | ldx \$09     | [ charac ] | Carica X con byte di Pag.0 per SP     |
| bldb: 9a       | txs          |            | Modifica SP con X                     |
| bldc: 4c 71 ff | jmp * \$ff71 | [ jmpfar ] | Vai alla subr JMPFAR                  |
| bldf: 20 74 b9 | jsr * \$b974 |            | C=0 OP1 in Pag. 0                     |
| ble2: 20 6e ff | jsr * \$ff6e | [ jsrfar ] | Vai a JSRFAR                          |
| ble5: 4c 8b b0 | jmp * \$b08b |            | Vai al ciclo di attesa per input      |
| ble8: 20 b4 b8 | jsr * \$b8b4 |            | Ritorno carrello + clear su resto lin |
| bleb: a9 3e    | lda #\$3e    |            | Carica A con '<'                      |
| bled: 20 d2 ff | jsr * \$ffd2 | [ bsout ]  | Vai a BSOUT                           |
| blf0: 20 92 b8 | jsr * \$b892 |            | OP3 in 5 Byte                         |
| blf3: a0 00    | ldy #\$00    |            | Cont di cicli messo a 0               |
| blf5: f0 03    | beq * \$blfa |            | Valore di 1 (esa)                     |
| blf7: 20 a8 b8 | jsr * \$b8a8 |            | SPAZIO e CR                           |
| blfa: 20 1a b1 | jsr * \$bl1a |            | Crshr up. A da un banco di mem.       |
| blfd: 20 c2 b8 | jsr * \$b8c2 |            | A visualizza in 2 Byte ASCII          |
| b200: c8       | iny          |            | Ciclo + n. spost + 1                  |
| b201: c0 08    | cpy #\$08    |            | Visual. di 8 in esa                   |
| b203: 24 d7    | bit \$d7     | [ mode ]   | Controllo per modo 40/80 colonne      |
| b205: 10 02    | bpl * \$b209 |            | Uscita a 40 colonne                   |
| b207: c0 10    | cpy #\$10    |            | Visualizz. di 16 in esa               |
| b209: 90 ec    | bcc * \$blf7 |            | Prendi prossimo valore esa            |
| b20b: 20 7d ff | jsr * \$ff7d | [ primm ]  | Vai a subr. PRINT                     |

b20e: 3a 12 00 : <rvs on>

Costante

|                |              |           |                                      |
|----------------|--------------|-----------|--------------------------------------|
| b211: a0 00    | ldy #\$00    |           | Ciclo e visualizz. cont. a 0         |
| b213: 20 1a b1 | jsr * \$bl1a |           | Carica A da banco di memoria         |
| b216: 48       | pha          |           | Immetti un carattere nello Stack     |
| b217: 29 7f    | and #\$7f    |           | Mask per bit 7                       |
| b219: c9 20    | cmp #\$20    |           | Prova per caratt. controllo          |
| b21b: 68       | pla          |           | Riprendi caratt. da STACK            |
| b21c: b0 02    | bcs * \$b220 |           | Se nessun car.contr. esegui Output   |
| b21e: a9 2e    | lda #\$2e    |           | Carica A con . (punto)               |
| b220: 20 d2 ff | jsr * \$ffd2 | [ bsout ] | Vai a subr. BSOUT                    |
| b223: c8       | iny          |           | Ciclo ed incr. cont +1               |
| b224: 24 d7    | bit \$d7     | [ mode ]  | Controllo per video 40/80 col.       |
| b226: 10 04    | bpl * \$b22c |           | Se 40 col.continua visualizz.        |
| b228: c0 10    | cpy #\$10    |           | Contr. se sono stampati 16 caratteri |
| b22a: 90 e7    | bcc * \$b213 |           | Altrimenti stampa il pross. caratt.  |
| b22c: c0 08    | cpy #\$08    |           | Contr. se sono stampati 8 caratt.    |
| b22e: 90 e3    | bcc * \$b213 |           | Altrimenti stampa il pross. caratt.  |
| b230: 60       | rts          |           |                                      |

## COMANDI MONITOR C &amp; T

|                |              |           |                                    |
|----------------|--------------|-----------|------------------------------------|
| b231: a9 00    | lda #\$00    |           | Fissa carattere per COMPARE        |
| b233: 2c a9 80 | bit \$80a9   |           | Salta a \$B236                     |
| b236: 85 93    | sta \$93     | [ verck ] | Fissa puntatore per Trasferimento  |
| b238: a9 00    | lda #\$00    |           | Puntatore di direzione per comando |
| b23a: 8d b3 0a | sta \$0ab3   |           | C/T a \$00                         |
| b23d: 20 83 b9 | jsr * \$b983 |           | Immettere e cont. passo in OPH,OP2 |
| b240: b0 05    | bcs * \$b247 |           | Se Carry set rilevato errore       |
| b242: 20 a7 b7 | jsr * \$b7a7 |           | Controllo separatori fra comandi   |

|                |     |          |        |               |  |
|----------------|-----|----------|--------|---------------|--|
| b245: 90 03    | bcc | *        | \$b24a |               | Operandi OK                              |
| b247: 4c bc b0 | jmp | *        | \$b0bc |               | Visual. ? e vai ciclo di attesa INPUT    |
| b24a: 24 93    | bit | \$93     |        | [ verck ]     | Controllo di trasferimento (-) o con     |
| b24c: 10 2c    | bpl | *        | \$b27a |               | fronto nella subr CMP                    |
| b24e: 38       | sec |          |        |               | Fissa il Carry per sottrazione           |
| b24f: a5 66    | lda | \$66     |        | [ indice ]    | Viene eseguito un controllo per vedere   |
| b251: e5 60    | sbc | \$60     |        | [ tenexp ]    | se il contenuto di entrambi i Bytes      |
| b253: a5 67    | lda | \$67     |        | [ facmo ]     | LO e HI ADDR sono > di OP3 o dei Bytes   |
| b255: e5 61    | sbc | \$61     |        | [ lowtr ]     | di indirizzo OP1                         |
| b257: b0 21    | bcs | *        | \$b27a |               | Se TO< FROM allora direzione OK          |
| b259: a5 63    | lda | \$63     |        | [ fac ]       | Aggiungi il contenuto di OP2 a \$65,\$64 |
| b25b: 65 60    | adc | \$60     |        | [ tenexp ]    | , \$63 al contenuto dell'operando OP1    |
| b25d: 85 60    | sta | \$60     |        | [ tenexp ]    | nelle locazioni \$62, \$61,\$60          |
| b25f: a5 64    | lda | \$64     |        | [ rightflag ] | ***                                      |
| b261: 65 61    | adc | \$61     |        | [ lowtr ]     | ***                                      |
| b263: 85 61    | sta | \$61     |        | [ lowtr ]     | Metti i risultati di qualsiasi somma,    |
| b265: a5 65    | lda | \$65     |        | [ facmoh ]    | che abbia dato OVERFLOW in OP1           |
| b267: 65 62    | adc | \$62     |        | [ expsgn ]    | Immagazzina il risultato della somma     |
| b269: 85 62    | sta | \$62     |        | [ expsgn ]    | in OP1                                   |
| b26b: a2 02    | ldx | #02      |        |               | Copia il contenuto degli operandi di     |
| b26d: bd b7 0a | lda | \$0ab7,x |        |               | indirizzo \$0Ab9 \$0AB8 \$0AB7 nell'oper |
| b270: 95 66    | sta | \$66,x   |        | [ indice ]    | ando OP3 (\$68, \$67,\$66)               |
| b272: ca       | dex |          |        |               | ***                                      |
| b273: 10 f8    | bpl | *        | \$b26d |               | ***                                      |
| b275: a9 80    | lda | #80      |        |               | Controlla che i dati di provenienza      |
| b277: 8d b3 0a | sta | \$0ab3   |        |               | siano diversi da quelli di destino       |
| b27a: 20 b4 b8 | jsr | *        | \$b8b4 |               | Vai ad indirizzo specificato             |
| b27d: a0 00    | ldy | \$00     |        |               | Metti il punt. di Spost a 0              |
| b27f: 20 e1 ff | jsr | *        | \$ffe1 | [ stop ]      | Vai alla subr. STOP                      |
| b282: f0 47    | beq | *        | \$b2cb |               | Se il tasto STOP e' premuto esci         |
| b284: 20 1a b1 | jsr | *        | \$b11a |               | Carica A da un banco di memoria          |
| b287: a2 60    | ldx | #60      |        |               | \$60 e = ind LO                          |
| b289: 8e b9 02 | stx | \$02b9   |        |               | Fissa STAVEC a questo indir.             |
| b28c: 8e c8 02 | stx | \$02c8   |        |               | Fissa CMPVEC a questo indir.             |
| b28f: a6 62    | ldx | \$62     |        | [ expsgn ]    | Carica X con byte banco                  |
| b291: 78       | sei |          |        |               | Disabilitati tutti gli INTERRUPT         |
| b292: 24 93    | bit | \$93     |        | [ verck ]     | Contr. per trasferim o confronto         |
| b294: 10 03    | bpl | *        | \$b299 |               | Confronto                                |
| b296: 20 77 ff | jsr | *        | \$ff77 | [ indsta ]    | Vai a subr. INDSTA                       |
| b299: a6 62    | ldx | \$62     |        | [ expsgn ]    | Carica X con byte di banco               |
| b29b: 20 7a ff | jsr | *        | \$ff7a | [ indcmp ]    | Vai a subr. INDCMP                       |
| b29e: 58       | cli |          |        |               | Abilita tutti gli INTERRUPT              |
| b29f: f0 09    | beq | *        | \$b2aa |               | Controllo che non sia dato un segnale    |
| b2a1: 20 92 b8 | jsr | *        | \$b892 |               | di uguale. Uscita di OP3 come ASCII      |
| b2a4: 20 a8 b8 | jsr | *        | \$b8a8 |               | con i contenuti di 5 Bytes per SPACE     |
| b2a7: 20 a8 b8 | jsr | *        | \$b8a8 |               | C/R, Cursor UP.                          |
| b2aa: 2c b3 0a | bit | \$0ab3   |        |               | Controllo per direz. di trasf.           |
| b2ad: 30 0b    | bmi | *        | \$b2ba |               | Invia nuovo ind. di ritorno              |
| b2af: e6 60    | inc | \$60     |        | [ tenexp ]    | Incrementa ind. di trasferimento         |
| b2b1: d0 10    | bne | *        | \$b2c3 |               | Se diverso vai ad errore                 |
| b2b3: e6 61    | inc | \$61     |        | [ lowtr ]     | Incrementa di 1                          |
| b2b5: d0 0c    | bne | *        | \$b2c3 |               | Se ind. HI in OVERFLOW = errore          |
| b2b7: 4c bc b0 | jmp | *        | \$b0bc |               | Vis ? e vai a ciclo attesa INPUT         |
| b2ba: 20 22 b9 | jsr | *        | \$b922 |               | Sottrazione: OP1 - costante 1            |
| b2bd: 20 60 b9 | jsr | *        | \$b960 |               | Sottrazione: OP3 - costante 1            |
| b2c0: 4c c6 b2 | jmp | *        | \$b2c6 |               | Vai a sottrazione OP2 - 1                |
| b2c3: 20 50 b9 | jsr | *        | \$b950 |               | Somma: costante lin OP3                  |
| b2c6: 20 3c b9 | jsr | *        | \$b93c |               | Sottrazione: OP2 - costante 1            |
| b2c9: b0 b4    | bcs | *        | \$b27f |               | Ciclo fino a tutti i passi eseguiti      |
| b2cb: 4c 8b b0 | jmp | *        | \$b08b |               | Salto a ciclo di attesa input            |

-----  
COMANDO MONITOR H

|                |              |              |   |
|----------------|--------------|--------------|---|
| b2ce: 20 83 b9 | jsr * \$b983 |              | Metti il valore di passo in OP1         |
| b2d1: b0 61    | bcs * \$b334 |              | Carica fissato = ident. Rilevato errore |
| b2d3: a0 00    | ldy #\$00    |              | Visual. caratt. ricerca in Buffer CPM   |
| b2d5: 20 e9 b8 | jsr * \$b8e9 |              | Leggi carattere da Buffer di input      |
| b2d8: c9 27    | cmp #\$27    |              | Controlla se e' car. = .                |
| b2da: d0 16    | bne * \$b2f2 |              | Se negativo non prendere come stringa   |
| b2dc: 20 e9 b8 | jsr * \$b8e9 |              | Leggi un caratt. dal Buffer di Input    |
| b2df: c9 00    | cmp #\$00    |              | E' un fine comando?                     |
| b2e1: f0 51    | beq * \$b334 |              | Se contr. prec pos. Visual. errore ?    |
| b2e3: 99 80 0a | sta \$0a80,y |              | Metti un car. nel buffer CPM            |
| b2e6: c8       | iny          |              | Sposta buffer CMP +1                    |
| b2e7: 20 e9 b8 | jsr * \$b8e9 |              | Contr. buffer input per fine comando    |
| b2ea: f0 1b    | beq * \$b307 |              | Contr se : o ? Se cosi' esegui ricerca  |
| b2ec: c0 20    | cpy #\$20    |              | Contr se 32 in Buffer CMP               |
| b2ee: d0 f3    | bne * \$b2e3 |              | Altrim. prendi il car successivo per    |
| b2f0: f0 15    | beq * \$b307 |              | rout. di ricerca                        |
| b2f2: 8c 00 01 | sty \$0100   |              | Immag lo spost. nel Buffer CMP          |
| b2f5: 20 a5 b7 | jsr * \$b7a5 |              | Metti operando CMP in OP1               |
| b2f8: a5 60    | lda \$60     | [ tenexp ]   | Trasmetti byte OP1 nel Buffer CMP       |
| b2fa: 99 80 0a | sta \$0a80,y |              | ***                                     |
| b2fd: c8       | iny          |              | Incrementa di 1 il buffer CMP           |
| b2fe: 20 a7 b7 | jsr * \$b7a7 |              | Immetti nuovo valore CMP in OP1         |
| b301: b0 04    | bcs * \$b307 |              | Esegui HUNT                             |
| b303: c0 20    | cpy #\$20    |              | Contr per 32 nel buffer CMP             |
| b305: d0 f1    | bne * \$b2f8 |              | Altrimenti prendi il succ. valore CMP   |
| b307: 84 93    | sty \$93     | [ verck ]    | Immag. contat. dei valori buffer CMP    |
| b309: 20 b4 b8 | jsr * \$b8b4 |              | ed esegui un CLEAR su resto linea       |
| b30c: a0 00    | ldy #\$00    |              | Visual. primo car. nel CMP Buffer       |
| b30e: 20 1a b1 | jsr * \$b11a |              | Carica A da un banco di mem.            |
| b311: d9 80 0a | cmp \$0a80,y |              | Confronta con car. nel Buffer CMP       |
| b314: d0 0e    | bne * \$b324 |              | Se diverso vai al prossimo passo        |
| b316: c8       | iny          |              | Visualizza prossimo valore Buffer CMP   |
| b317: c4 93    | cpy \$93     | [ verck ]    | ***                                     |
| b319: d0 f3    | bne * \$b30e |              | Se no prossimo passo di confronto       |
| b31b: 20 92 b8 | jsr * \$b892 |              | Contenuto di OP3                        |
| b31e: 20 a8 b8 | jsr * \$b8a8 |              | SPAZIO, CR,crsr-up                      |
| b321: 20 a8 b8 | jsr * \$b8a8 |              | SPAZIO, CR,crsr-up                      |
| b324: 20 e1 ff | jsr * \$ffef | [ stop ]     | Vai a sub. STOP: Controlla tasto STOP   |
| b327: f0 08    | beq * \$b331 |              | Se STOP premuto esci                    |
| b329: 20 50 b9 | jsr * \$b950 |              | Somma: costante 1 in OP3                |
| b32c: 20 3c b9 | jsr * \$b93c |              | Sottrazione: OP2-costante 1             |
| b32f: b0 db    | bcs * \$b30c |              | Ciclo attesa x esec tutti i passi       |
| b331: 4c 8b b0 | jmp * \$b08b |              | Salto al ciclo di attesa input          |
| b334: 4c bc b0 | jmp * \$b0bc |              | Visual. ?                               |
| b337: a0 01    | ldy #\$01    |              | Carica Y con \$01                       |
| b339: 84 ba    | sty \$ba     | [ fa ]       | Fissa n. periferi (1= nastro)           |
| b33b: 84 b9    | sty \$b9     | [ sa ]       | Fissa ind. secondario                   |
| b33d: 88       | dey          |              | Decrementa Y fino a \$00                |
| b33e: 84 c6    | sty \$c6     | [ ba ]       | Fissa n. banco per comandi L,V,S        |
| b340: 84 b7    | sty \$b7     | [ fnlen ]    | Fissa a 0 la lunghezza nome file        |
| b342: 84 c7    | sty \$c7     | [ fnbank ]   | Fissa banco per ind. nome file          |
| b344: 84 90    | sty \$90     | [ status ]   | Metti a 0 il Byte di status             |
| b346: a9 0a    | lda #\$0a    |              | Carica A in Pag. 0 con Hi Addr          |
| b348: 85 bc    | sta \$bc     | [ fnadr+ 1 ] | Nome del file caricato con \$0A         |
| b34a: a9 80    | lda #\$80    |              | Carica in Pag. 0 nome file per ind.     |
| b34c: 85 bb    | sta \$bb     | [ fnadr ]    | LO                                      |
| b34e: 20 e9 b8 | jsr * \$b8e9 |              | Controllo sul Buffer di inputi          |

```

b351: f0 58      beq * $b3ab
b353: c9 20      cmp #$20
b355: f0 f7      beq * $b34e
b357: c9 22      cmp #$22
b359: d0 15      bne * $b370
b35b: a6 7a      ldx $7a      [ dsdec ]
b35d: bd 00 02   lda $0200,x
b360: f0 49      beq * $b3ab
b362: e8         inx
b363: c9 22      cmp #$22
b365: f0 0c      beq * $b373
b367: 91 bb      sta ($bb),y [ fnadr ]
b369: e6 b7      inc $b7      [ fnlen ]
b36b: c8         iny
b36c: c0 11      cpy #$11
b36e: 90 ed      bcc * $b35d
b370: 4c bc b0   jmp * $b0bc

```

Se e' un fine comando vai ciclo INPUT  
 Contr. se il carattere letto e' SPAZIO  
 Se si continua e leggei il Pross. car.  
 Controllo se caratt e' "  
 Altrimenti errore in stringa com.  
 X caric. con visualiz da Buff. input  
 Legge il primo " in Buffer(=nomefile)  
 \$00=Fine stringa di comando  
 Punt. di buffer INPUT al pross. car.  
 Controllo rilevam secondo " (apici)  
 Apici trovato, prosegue valut.  
 Nome file immesso a \$0A80  
 Increm +1 contat. lungh. nome file  
 Increm. puntatore nome file  
 Contr. lungh.nome file >16 caratteri  
 Se minore leggi pross. carattere  
 Visual. ? e va ciclo attesa INPUT

## PARAMETRI DI VALUTAZIONE LVS

```

b373: 86 7a      stx $7a      [ dsdec ]
b375: 20 e9 b8   jsr * $b8e9
b378: f0 31      beq * $b3ab
b37a: 20 a7 b7   jsr * $b7a7
b37d: b0 2c      bcs * $b3ab
b37f: a5 60      lda $60      [ tenexp ]
b381: 85 ba      sta $ba      [ fa ]
b383: 20 a7 b7   jsr * $b7a7
b386: b0 23      bcs * $b3ab
b388: 20 01 b9   jsr * $b901
b38b: 85 c6      sta $c6      [ ba. ]
b38d: 20 a7 b7   jsr * $b7a7
b390: b0 3f      bcs * $b3d1
b392: 20 b4 b8   jsr * $b8b4
b395: a6 60      ldx $60      [ tenexp ]
b397: a4 61      ldy $61      [ lowtr ]
b399: a5 93      lda $93      [ verck ]
b39b: c9 53      cmp #$53
b39d: d0 d1      bne * $b370
b39f: a9 00      lda #$00
b3a1: 85 b9      sta $b9      [ sa ]
b3a3: a9 66      lda #$66
b3a5: 20 d8 ff   jsr * $ffd8 [ savesp ]
b3a8: 4c 8b b0   jmp * $b08b
b3ab: a5 93      lda $93      [ verck ]
b3ad: c9 56      cmp #$56
b3af: f0 06      beq * $b3b7
b3b1: c9 4c      cmp #$4c
b3b3: d0 bb      bne * $b370
b3b5: a9 00      lda #$00
b3b7: 20 d5 ff   jsr * $ffd5 [ loadsp ]
b3ba: a5 90      lda $90      [ status ]
b3bc: 29 10      and $10
b3be: f0 e8      beq * $b3a8
b3c0: a5 93      lda $93      [ verck ]
b3c2: f0 ac      beq * $b370
b3c4: 20 7d ff   jsr * $ffd7 [ primm ]

```

Punt.BUFFER input dopo il secondo "  
 Contr. Buffer x fine comando : ?  
 LOAD e VERIFY possono girare  
 Prendi i param da OPl  
 Nessun prametro, vai espress. L o V  
 Prendi val. OPl (Ind. LO)  
 Mettilo in Pagina zero  
 Prendi i param OPl (ind. di inizio)  
 Nessun parametro, vai a Espr per L V  
 Copia il cont. di OPl in OP3  
 Prendi il n. di banco da Pag. 0  
 Parametri LVS (fine Ind)  
 Nessun para. trovato, vai a Espre. LV  
 Esegui LFeed e clear sul resto riga  
 Carica X con val. LO per SAVE  
 Carica Y con val. HI per SAVE  
 Prendi il comando  
 E' un S ( per SAVE)?  
 No = errore  
 Carica A con 0, in pag. 0  
 per indirizzo secondario  
 N. banco da operando OP3  
 Vai a subr. SAVESP.  
 Vai a ciclo di attesa INPUT  
 Valuta il tipo di comando  
 E' V per Verify  
 Verifica a LOADSP se A <>0  
 E' L per Load  
 No, vedi se e' S per Save  
 Carica A con 0 per sub. LOADSP  
 Vai a subr. LOADSP  
 Carica STATUS in A  
 Bit mask per errore lettura  
 Nessun errore LV, vai ciclo attesa INP  
 Preleva car.come comando  
 Non e' un comando. ERRORE  
 Vai a subr PRINT

COSTANTE MONITOR

b3c7: 20 45 52 52 4f 52 00 error

Costante del MONITOR per ERRORE

## COMANDO MONITOR F

|                |       |        |            |  |
|----------------|-------|--------|------------|--|
| b3ce: 4c 8b b0 | jmp * | \$b08b |            | Vai ciclo di attesa INPUT              |
| b3d1: a6 66    | ldx * | \$66   | [ indice ] | Carica in X ind. di inizio LO          |
| b3d3: a4 67    | ldy * | \$67   | [ facmo ]  | Carica in Y ind. di inizio HI          |
| b3d5: a9 00    | lda * | \$00   |            | Scrivi ind. sec. (\$00=read)           |
| b3d7: 85 b9    | sta * | \$b9   | [ sa ]     | Metti in A in Pag. 0 per ind.sec. val. |
| b3d9: f0 d0    | beq * | \$b3ab |            | COMUNQUE prec. per comandi LV          |
| b3db: 20 83 b9 | jsr * | \$b983 |            | Metti prov. e dim. passo in OPH,OP2    |
| b3de: b0 23    | bcs * | \$b403 |            | Controllo per Carry set                |
| b3e0: a5 68    | lda * | \$68   | [ facsgn ] | Metti n. banco da OP3                  |
| b3e2: cd b9 0a | cmp * | \$0ab9 |            | Confronta n. banco con operando        |
| b3e5: d0 1c    | bne * | \$b403 |            | Se diverso e' errore di identific.     |
| b3e7: 20 a7 b7 | jsr * | \$b7a7 |            | Prendi il parametro di comando         |
| b3ea: b0 17    | bcs * | \$b403 |            | Se CARRY a 0 errore di ident.          |
| b3ec: a0 00    | ldy * | \$00   |            | Fissa visual. per comando, 0 in OP1    |
| b3ee: a5 60    | lda * | \$60   | [ tenexp ] | Indirizzo LO                           |
| b3f0: 20 2a b1 | jsr * | \$b12a |            | Rout di STA                            |
| b3f3: 20 e1 ff | jsr * | \$ffef | [ stop ]   | Vai a subr. di STOP. Contr per Tasto   |
| b3f6: f0 08    | beq * | \$b400 |            | Se premuto, vai ciclo att. INPUT       |
| b3f8: 20 50 b9 | jsr * | \$b950 |            | Somma: costante 1 in OP3               |
| b3fb: 20 3c b9 | jsr * | \$b93c |            | Sottrazione: OP2- costante 1           |
| b3fe: b0 ee    | bcs * | \$b3ee |            | Controllo per STOP premuto             |
| b400: 4c 8b b0 | jmp * | \$b08b |            | Vai a ciclo di attesa input            |
| b403: 4c bc b0 | jmp * | \$b0bc |            | Visualizza ? e poi come sopra          |

## COMANDO MONITOR A

|                |       |          |  |   |
|----------------|-------|----------|--|---|
| b406: b0 3a    | bcs * | \$b442   |  | Se CARRY a 0 = errore identific.        |
| b408: 20 01 b9 | jsr * | \$b901   |  | Copia OP1 in OP3                        |
| b40b: a2 00    | ldx * | \$00     |  | Esegui Clear su visual. BUFFER.         |
| b40d: 8e a1 0a | stx * | \$0aa1   |  | Metti a 0 il Bit ident. coman. abbrev   |
| b410: 8e b4 0a | stx * | \$0ab4   |  | Metti a 0 il contatore di ciclo         |
| b413: 20 e9 b8 | jsr * | \$b8e9   |  | Contr. Buffer per fine coman. : o ?     |
| b416: d0 07    | bne * | \$b41f   |  | Se non e' fine com. vai oltre           |
| b418: e0 00    | cpx * | \$00     |  | Visualizza ancora 0, nessun coman.      |
| b41a: d0 03    | bne * | \$b41f   |  | Se come sopra continua                  |
| b41c: 4c 8b b0 | jmp * | \$b08b   |  | Vai al ciclo di attesa input            |
| b41f: c9 20    | cmp * | \$20     |  | Il car. letto e' SPAZIO?                |
| b421: f0 e8    | beq * | \$b40b   |  | Se si leggi e inizializza               |
| b423: 9d ac 0a | sta * | \$0aac,x |  | Immetti caratt. nelBuffer mnem.         |
| b426: e8       | inx   |          |  | Il Buffer visual. punt. +1              |
| b427: e0 03    | cpx * | \$03     |  | Sono 3 caratteri?                       |
| b429: d0 e8    | bne * | \$b413   |  | Se no, vai al prossimo car.             |
| b42b: ca       | dex   |          |  | Vis. punt. a ultimo carattere           |
| b42c: 30 17    | bmi * | \$b445   |  | Operazione sui 3 caratteri              |
| b42e: bd ac 0a | lda * | \$0aac,x |  | Leggi ALL'INDIETRO i 3 car.             |
| b431: 38       | sec   |          |  | Fissa il carry per sottrazione          |
| b432: e9 3f    | sbc * | \$3f     |  | Valori car. alfa; A=1,b=2,ecc.          |
| b434: a0 05    | ldy * | \$05     |  | Cont. esegue spost di 5x 1 bit.         |
| b436: 4a       | lsr a |          |  | Spost. di 1 bit del valore della lett.  |
| b437: 6e a1 0a | ror * | \$0aa1   |  | fuori di A nei Byte pari di \$AA1-\$AA0 |
| b43a: 6e a0 0a | ror * | \$0aa0   |  | I 3 car. mnemonici saranno spostati     |
| b43d: 88       | dey   |          |  | nel byte pari appena detto e occupare   |
| b43e: d0 f6    | bne * | \$b436   |  | 3 gruppi di 5 bits in questi bytes      |
| b440: f0 e9    | beq * | \$b42b   |  | ***                                     |
| b442: 4c bc b0 | jmp * | \$b0bc   |  | Visual. ?; vai ciclo att. input         |
| b445: a2 02    | ldx * | \$02     |  | Fissa spost. buffer di uscita.          |

|       |          |                |               |  |
|-------|----------|----------------|---------------|--|
| b447: | ad b4 0a | lda \$0ab4     |               | Carica cont. di ciclo in A             |
| b44a: | d0 30    | bne * \$b47c   |               | Se diverso da 0, salta oltre           |
| b44c: | 20 ce b7 | jsr * \$b7ce   |               | Metti i param. di comando in OP1       |
| b44f: | f0 29    | beq * \$b47a   |               | Se = 0, salta a contr. fine com.       |
| b451: | b0 ef    | bcs * \$b442   |               | Fissa il carry per uscita err.         |
| b453: | a9 24    | lda #\$24      |               | Carica \$ in A e                       |
| b455: | 9d a0 0a | sta \$0aa0,x   |               | vai al Buffer uscita                   |
| b458: | e8       | inx            |               | Incrementa Buffer uscita +1            |
| b459: | a5 62    | lda \$62       | [ expsgn ]    | Immetti il val di OP1 in byte Banco    |
| b45b: | d0 e5    | bne * \$b442   |               | Se > 0 errore nell'indic OUTPUT        |
| b45d: | a0 04    | ldy #\$04      |               | Fattore di divisione esa               |
| b45f: | ad b6 0a | lda \$0ab6     |               | Carica n. base operando                |
| b462: | c9 08    | cmp #\$08      |               | Confronta con 8                        |
| b464: | 90 05    | bcc * \$b46b   |               | Se <8 metti OP1                        |
| b466: | cc b4 0a | cpy \$0ab4     |               | Confronta con contatore di ciclo       |
| b469: | f0 06    | beq * \$b471   |               | Se uguale salta oltre                  |
| b46b: | a5 61    | lda \$61       | [ lowtr ]     | Carica ind. alto di OP1                |
| b46d: | d0 02    | bne * \$b471   |               | Se <> 0 vai oltre                      |
| b46f: | a0 02    | ldy #\$02      |               | Fissa il contatore di ciclo per Byte n |
| b471: | a9 30    | lda #\$30      |               | Carica ASCII 0 in A e quindi immetti   |
| b473: | 9d a0 0a | sta \$0aa0,x   |               | nell'area di imm. temp coman. ASS.     |
| b476: | e8       | inx            |               | Incrementa il contat. dei com. ass     |
| b477: | 88       | dey            |               | Ciclo per OP                           |
| b478: | d0 f9    | bne * \$b473   |               | Ciclo per cont = 0                     |
| b47a: | c6 7a    | dec \$7a       | [ dsdec ]     | Vis. cont. carattere prev.             |
| b47c: | 20 e9 b8 | jsr * \$b8e9   |               | Contr. buffer input per fine comando   |
| b47f: | f0 0e    | beq * \$b48f   |               | Se e' un fine comando vai ad ind       |
| b481: | c9 20    | cmp #\$20      |               | Contr. se carat. e' spazio             |
| b483: | f0 c2    | beq * \$b447   |               | Se si immetti nuovo param. espress.    |
| b485: | 9d a0 0a | sta \$0aa0,x   |               | in imm. temp. dei com. assembler       |
| b488: | e8       | inx            |               | Il comando e' > di 9 carat.            |
| b489: | e0 0a    | cpx #\$0a      |               | Se no vai al pross. carat.             |
| b48b: | 90 ba    | bcc * \$b447   |               | Se si visual ? errore                  |
| b48d: | b0 b3    | bcs * \$b442   |               | Lunghezza comando in OP2 (lo)          |
| b48f: | 86 63    | stx \$63       | [ fac ]       | Lunghezza byte coman. in OP2 (LO)      |
| b491: | a2 00    | ldx #\$00      |               | Carica X con 0                         |
| b493: | 8e b1 0a | stx \$0abl     |               | Cont. ciclo confronto comandi          |
| b496: | a2 00    | ldx #\$00      |               | Carica X con 0 e usalo come visualiz.  |
| b498: | 86 9f    | stx \$9f       | [ ptr2 ]      | per Buffer comandi Ass.                |
| b49a: | ad b1 0a | lda \$0abl     |               | Carica A con cont. comandi             |
| b49d: | 20 59 b6 | jsr * \$b659   |               | Indiri. e lungh. cont prec.            |
| b4a0: | ae aa 0a | ldx \$0aaa     |               | Prendi lungh. punt comandi (0,1,2)     |
| b4a3: | 86 64    | stx \$64       | [ rightflag ] | Je immag. in OP2 (HI)                  |
| b4a5: | aa       | tax            |               | Contr. di risult. per confr.           |
| b4a6: | bd 61 b7 | lda * \$b761,x |               | Byte a tab.2                           |
| b4a9: | 20 7f b5 | jsr * \$b57f   |               | Confronta con Byte in buffer ASS.      |
| b4ac: | bd 21 b7 | lda * \$b721,x |               | Byte in Tab 1                          |
| b4af: | 20 7f b5 | jsr * \$b57f   |               | Confronta con byte in Buffer ASS       |
| b4b2: | a2 06    | ldx #\$06      |               | Cont. ciclo per ind. confr.            |
| b4b4: | e0 03    | cpx #\$03      |               | Controllo per 3 cicli compl.           |
| b4b6: | d0 14    | bne * \$b4cc   |               | Contr=no e' solo ind. di confr.        |
| b4b8: | ac ab 0a | ldy \$0aab     |               | Carica punt.(0,1,2)d. lungh. comandi   |
| b4bb: | f0 0f    | beq * \$b4cc   |               | Opera come comando di un Byte          |
| b4bd: | ad aa 0a | lda \$0aaa     |               | Carica ind. chiave                     |
| b4c0: | c9 e8    | cmp #\$e8      |               | Confronta con \$E8                     |
| b4c2: | a9 30    | lda #\$30      |               | 0 in ASCII in A                        |
| b4c4: | b0 1e    | bcs * \$b4e4   |               | Carry set per valut. corr.             |
| b4c6: | 20 7c b5 | jsr * \$b57c   |               | Confronto con byte nel Buffer ASS.     |
| b4c9: | 88       | dey            |               | Decr.lungh. cont. comandi di 1         |
| b4ca: | d0 f1    | bne * \$b4bd   |               | Se <> da 0 vai oltre                   |

|                             |     |            |               |                                       |
|-----------------------------|-----|------------|---------------|---------------------------------------|
| b4cc: 0e aa 0a              | asl | \$0aaa     |               | Sposta indirizzo chiave               |
| b4cf: 90 0e                 | bcc | * \$b4df   |               | Bit=0 poi salta al confr.             |
| b4d1: bd 14 b7              | lda | * \$b714,x |               | Carica ind. car 1 al tab              |
| b4d4: 20 7f b5              | jsr | * \$b57f   |               | Confronta con byte in buffer ASS      |
| b4d7: bd 1a b7              | lda | * \$b71a,x |               | Carica ind. car 2 al tab              |
| b4da: f0 03                 | beq | * \$b4df   |               | Se \$00, nessun confronto             |
| b4dc: 20 7f b5              | jsr | * \$b57f   |               | Confronta con byte in Buffer ASS      |
| b4df: ca                    | dex |            |               | Contatore cicli ind. -1               |
| b4e0: d0 d2                 | bne | * \$b4b4   |               | Se <> 0 continua ciclo                |
| b4e2: f0 06                 | beq | * \$b4ea   |               | 0, continua confr.                    |
| b4e4: 20 7c b5              | jsr | * \$b57c   |               | Confr. con byte in buffer ASS         |
| b4e7: 20 7c b5              | jsr | * \$b57c   |               | Confr. con byte in buffer ASS         |
| b4ea: a5 63                 | lda | \$63       | [ fac ]       | Carica lungh. immag. dei comandi ASS. |
| b4ec: c5 9f                 | cmp | \$9f       | [ ptr2 ]      | Confr. con visualizz.                 |
| b4ee: f0 03                 | beq | * \$b4f3   |               | Se uguale vai oltre                   |
| b4f0: 4c 8b b5              | jmp | * \$b58b   |               | Incrementa cont. cicli com            |
| b4f3: ac ab 0a              | ldy | \$0aab     |               | Preleva lungh.punt comandi            |
| b4f6: f0 32                 | beq | * \$b52a   |               | Se = 0 metti 1 nel Byte comando       |
| b4f8: a5 64                 | lda | \$64       | [ rightflag ] | Carica byte ind. HI da OP2            |
| b4fa: c9 9d                 | cmp | \$9d       |               | Confrontalo con \$9d                  |
| b4fc: d0 23                 | bne | * \$b521   |               | Se diverso vai aoltre                 |
| b4fe: a5 60                 | lda | \$60       | [ tenexp ]    | Carica ind. operando LO e             |
| b500: e5 66                 | sbc | \$66       | [ indice ]    | sottrai indirizzo com. LO             |
| b502: aa                    | tax |            |               | Metti risultato in X                  |
| b503: a5 61                 | lda | \$61       | [ lowtr ]     | Carica ind. operando HI e             |
| b505: e5 67                 | sbc | \$67       | [ facmo ]     | sottrai ind. com HI                   |
| b507: 90 08                 | bcc | * \$b511   |               | Valutazione dell'espressione          |
| b509: d0 6e                 | bne | * \$b579   |               | BRANCH OUT OF RANGE                   |
| b50b: e0 82                 | cpx | \$82       |               | Controlla dove l'espress. e' valida   |
| b50d: b0 6a                 | bcs | * \$b579   |               | Se risult. magg. cont. \$82 dai ?     |
| b50f: 90 08                 | bcc | * \$b519   |               | nella corrispondente espressione      |
| b511: a8                    | tay |            |               | Copia A in Y e                        |
| b512: c8                    | iny |            |               | incrementalo da 0 a 1                 |
| b513: d0 64                 | bne | * \$b579   |               | Se <>0 visualizza ? errore            |
| b515: e0 82                 | cpx | \$82       |               | Confronta con \$02                    |
| b517: 90 60                 | bcc | * \$b579   |               | Se minore di 2 visual. ?              |
| b519: ca                    | dex |            |               | Decrementa reg. X                     |
| b51a: ca                    | dex |            |               | Decrementa X                          |
| b51b: 8a                    | txa |            |               | Trasf valore in A                     |
| b51c: ac ab 0a              | ldy | \$0aab     |               | Carica in Y cont. lungh. com.         |
| b51f: d0 03                 | bne | * \$b524   |               | Se <>0 vai oltre                      |
| b521: b9 5f 00              | lda | \$005f,y   |               | Carica val. da OP1                    |
| b524: 20 2a b1              | jsr | * \$b12a   |               | Vai a Subr. STA per A in banco mem    |
| b527: 88                    | dey |            |               | Decrementa punt. lungh. com. di 1     |
| b528: d0 f7                 | bne | * \$b521   |               | Se <> 0 vai oltre                     |
| b52a: ad b1 0a              | lda | \$0abl     |               | Prendi valore da OP1                  |
| b52d: 20 2a b1              | jsr | * \$b12a   |               | Vai a Subr. STA per A in banco mem.   |
| b530: 20 ad b8              | jsr | * \$b8ad   |               | CR crsr-up                            |
| b533: 20 7d ff              | jsr | * \$ff7d   | [ primm ]     | Vai a Sub. PRINT per vis. stringa     |
| -----                       |     |            |               | Costanti monitor:                     |
| b536: 41 20 1b 51 00        |     |            |               | a esc-q                               |
| -----                       |     |            |               |                                       |
| INIZIO CONTR. PER ASSEMBLY  |     |            |               |                                       |
| b53b: 20 dc b5 jsr * \$b5dc |     |            |               | Uscita ind. e carica byte             |
| b53e: ee ab 0a inc \$0aab   |     |            |               | Incr. di 1 lungh. punt. opcode        |
| b541: ad ab 0a lda \$0aab   |     |            |               | Agg. lungh. a operando                |
| b544: 20 52 b9 jsr * \$b952 |     |            |               | Somma: contenuto di A +OP3            |
| b547: a9 41 lda #\$41       |     |            |               | Carica A con A di assembler           |



```

b549: 8d 4a 03 sta $034a
b54c: a9 20 lda #$20
b54e: 8d 4b 03 sta $034b
b551: 8d 51 03 sta $0351
b554: a5 68 lda $68 [ facsgn ]
b556: 20 d2 b8 jsr * $b8d2
b559: 8e 4c 03 stx $034c
b55c: a5 67 lda $67 [ facmo ]
b55e: 20 d2 b8 jsr * $b8d2
b561: 8d 4d 03 sta $034d
b564: 8e 4e 03 stx $034e
b567: a5 66 lda $66 [ indice ]
b569: 20 d2 b8 jsr * $b8d2
b56c: 8d 4f 03 sta $034f
b56f: 8e 50 03 stx $0350
b572: a9 08 lda #$08
b574: 85 d0 sta $d0 [ ndx ]
b576: 4c 8b b0 jmp * $b08b
b579: 4c bc b0 jmp * $b0bc
b57c: 20 7f b5 jsr * $b57f
b57f: 8e af 0a stx $0aaf
b582: a6 9f ldx $9f [ ptr2 ]
b584: dd a0 0a cmp $0aa0,x
b587: f0 0a beq * $b593
b589: 68 pla
b58a: 68 pla
b58b: ee b1 0a inc $0ab1
b58e: f0 e9 beq * $b579
b590: 4c 96 b4 jmp * $b496
b593: e6 9f inc $9f [ ptr2 ]
b595: ae af 0a ldx $0aaf
b598: 60 rts

```

## COMANDO MONITOR D

```

b599: b0 08 bcs * $b5a3
b59b: 20 01 b9 jsr * $b901
b59e: 20 a7 b7 jsr * $b7a7
b5a1: 90 06 bcc * $b5a9
b5a3: a9 14 lda #$14
b5a5: 85 60 sta $60 [ tenexp ]
b5a7: d0 05 bne * $b5ae
b5a9: 20 0e b9 jsr * $b90e
b5ac: 90 23 bcc * $b5d1
b5ae: 20 7d ff jsr * $ff7d [ primm ]

```

```

b5b1: 0d 1b 51 00 <cr> <esc> q

```

## CONTROLLO PER DISASSEMBLAGGIO

```

b5b5: 20 e1 ff jsr * $ffef [ stop ]
b5b8: f0 14 beq * $b5ce
b5ba: 20 d4 b5 jsr * $b5d4
b5bd: ee ab 0a inc $0aab
b5c0: ad ab 0a lda $0aab
b5c3: 20 52 b9 jsr * $b952
b5c6: ad ab 0a lda $0aab
b5c9: 20 24 b9 jsr * $b924
b5cc: b0 e0 bcs * $b5ae

```

nel buffer di proc. per pross. linea  
 Carica A con SPAZIO  
 nel buffer della procedura per la  
 prossima linea  
 Carica Byte banco di ind. in A  
 Acc in 2 Bytes ASCII: HI=A LO=X  
 Imm. nel buffer di proc il Byte di IND  
 alto per linea succ.  
 Acc in 2 Bytes ASCII: HI=A LO=X  
 Imm. nel Buffer di proc il Byte  
 alto per linea successiva  
 Incrementa il Buffer di proce. per  
 la prossima linea da elab.  
 A in 2 Byte ASCII: HI=A  
 LO=X  
 Immetti cont. Buffer e procedi  
 Buffer di tastiera fissato per  
 una lunghezza di 8 caratteri  
 Salto al ciclo di attesa input  
 Esegui 2 volte la seguente rout.  
 Imm. i cont del reg. X  
 Car. punt.visual. com. ass.  
 Confr. con cart. da buffer ASS  
 Se uguale esci  
 Preleva ind. RTS da STACK  
 Idem  
 Incrementa ciclo confronto comandi  
 Se >255 visualizza errore  
 Salta alla corrisp. espress.  
 Visualizza punt. comandi +1  
 Riportati al vecchio cont. X

Nessun operando valido  
 Copia OPl su OP3  
 Prendi operando OPl  
 Se e' un Oper. valido invia n. passi  
 Il valore del passo stand. e' \$14  
 nel contatore di passo basso  
 Salto incond. al Disass.  
 Immag. differenza OPl-OP3 in OPl  
 Esegui un CLEAR CARRY  
 Vai a sub. PRINT

Cr ESC-q  
 Sono costanti del MONITOR

Vai a Sub. STOP: controlla STOP prem  
 Se prem, vai ciclo att. INPUT  
 Prep. e vis. linea disass.  
 Incr. cont. lungh. cod.oper di 1  
 e immetti ind calc. in A  
 SOMMA: Cont acc+OP3  
 Lungh.punt. peramp.passo in A  
 Sottrazione: OPl-cont. acc  
 Continua disass. se nec.

```

b5ce: 4c 8b b0 jmp * $b08b
b5d1: 4c bc b0 jmp * $b0bc
b5d4: a9 2e lda #$2e
b5d6: 20 d2 ff jsr * $ffd2 [ bsout ]
b5d9: 20 a8 b8 jsr * $b8a8
b5dc: 20 92 b8 jsr * $b892
b5df: 20 a8 b8 jsr * $b8a8
b5e2: a0 00 ldy #$00
b5e4: 20 1a b1 jsr * $b11a
b5e7: 20 59 b6 jsr * $b659
b5ea: 48 pha
b5eb: ae ab 0a ldx $0aab
b5ee: e8 inx
b5ef: ca dex
b5f0: 10 0a bpl * $b5fc
b5f2: 20 7d ff jsr * $ff7d [ primm ]
-----

```

b5f5: 20

#### ROUTINE DI CONTROLLO BANCHI

```

b5e4: 20 1a b1 jsr * $b11a
b5e7: 20 59 b6 jsr * $b659
b5ea: 48 pha
b5eb: ae ab 0a ldx $0aab
b5ee: e8 inx
b5ef: ca dex
b5f0: 10 0a bpl * $b5fc
b5f2: 20 7d ff jsr * $ff7d [ primm ]
-----

```

b5f5: 20 20 20 00

#### ROUTINE PER ASSEMBLER/DISASSEMBLER

```

b5f9: 4c 02 b6 jmp * $b602
b5fc: 20 1a b1 jsr * $b11a
b5ff: 20 a5 b8 jsr * $b8a5
b602: c8 iny
b603: c0 03 cpy #$03
b605: 90 e8 bcc * $b5ef
b607: 68 pla
b608: a2 03 ldx #$03
b60a: 20 a1 b6 jsr * $b6a1
b60d: a2 06 ldx #$06
b60f: e0 03 cpx #$03
b611: d0 17 bne * $b62a
b613: ac ab 0a ldy $0aab
b616: f0 12 beq * $b62a
b618: ad aa 0a lda $0aaa
b61b: c9 e8 cmp #$e8
b61d: 08 php
b61e: 20 1a b1 jsr * $b11a
b621: 28 plp
b622: b0 1d bcs * $b641
b624: 20 c2 b8 jsr * $b8c2
b627: 88 dey
b628: d0 ee bne * $b618
b62a: 0e aa 0a asl $0aaa

```

Vai a ciclo di attesa INPUT  
 Vis. ? e poi come sopra  
 Carica Acc con .  
 Vai a subr.BSOUT  
 Vis. car. SPAZIO, CR, crsr up  
 Vis. da OP3 come 5 Bytes ASCII  
 stessi cara di \$B5D9  
 Carica Y per FETCH  
 Routine LDA per banchi mem  
 Controllo valid. per Bytes Cod. oper.  
 Metti risultato in STACK  
 Carica ciclo lungh. comando  
 Incrementa la lunghezza di 1  
 decrementa di 1  
 Visualizza val. com. <0 e' cost.  
 Vai a subr. PRINT

Costanti del MONITOR costituite da  
 3 SPAZI

Rout. di caric da un banco mem  
 Contr. di valid. byte cod. oper.  
 Metti il ris. nello STACK  
 Carica ciclo lungh. comando  
 Incrementa la lunghezza di 1  
 Decrementa la lunghezza di 1  
 Vis. val. comando. Se <0 costan.  
 Vai a subr. PRINT

Routine del monitor di 3 spazi

Salta a LDA per routine  
 Rout. LDA per A in banco di mem.  
 Vis. A come 2 bytes ASCII + Spazio  
 Incrementa X di 1  
 Confronta con \$03  
 Se <3 continua il ciclo  
 Prendi il risultato dallo STACK  
 Carica 3 car. per uscita mnem.  
 e per uscita car. in X  
 Inizial. ciclo cont. con 6  
 Dopo 3 cicli verra' visualizzato  
 l' attuale valore indir. va in uscita  
 Numero dei bytes dei com. operando  
 Se nessun bytes comando, salta  
 Indirizzo chiave del comando  
 Controllo per salto  
 Immetti il flag di CARRY nello STACK  
 Rout. di car. per un banco  
 Esegui un reset del FLAG del Carry  
 Se Carry set esegui salto  
 Trasf. A in oper. di 2 Bytes ASCII  
 Se il comando inogg. ha un operando  
 di 2 Bytes l' espres. del II Byte e'  
 mask. dalla chiave di indirizzo

|                |       |                  |                                     |
|----------------|-------|------------------|-------------------------------------|
| b62d: 90 0e    | bcc * | \$b63d           | Se il bit non e' fiss. esegui salto |
| b62f: bd 14 b7 | lda * | \$b714,x         | Carica car. per tipo indir.         |
| b632: 20 d2 ff | jsr * | \$ffd2 [ bsout ] | Vai a subr. BSOUT                   |
| b635: bd 1a b7 | lda * | \$b71a,x         | Carica car. per tipo indir.         |
| b638: f0 03    | beq * | \$b63d           | Se <> 0 salta                       |
| b63a: 20 d2 ff | jsr * | \$ffd2 [ bsout ] | Vai a subr. BSOUT                   |
| b63d: ca       | dex   |                  | Ciclo di uscita ind. -1             |
| b63e: d0 cf    | bne * | \$b60f           | Uscita di tutti e 6 i cicli         |
| b640: 60       | rts   |                  |                                     |

-----  
INDIRIZZI PER COMANDI DI SALTO

|                |          |                  |                                   |
|----------------|----------|------------------|-----------------------------------|
| b641: 20 4d b6 | jsr *    | \$b64d           | Calc. indirizzi X=HI A=LO         |
| b644: 18       | clc      |                  | Esegui clr carry per somma        |
| b645: 69 01    | adc      | #\$01            | Aggiun. 1 per correz. ind. LO     |
| b647: d0 01    | bne *    | \$b64a           | Controllo per overflow            |
| b649: e8       | inx      |                  | Aggiu. 1 per correz. HI           |
| b64a: 4c 9f b8 | jmp *    | \$b89f           | Dai val. di A + X come 4 Bytes    |
| b64d: a6 67    | ldx \$67 | [ facmo ]        | Carica ind. HI prov. operando OP3 |
| b64f: a8       | tay      |                  | Metti in X ind. salto             |
| b650: 10 01    | bpl *    | \$b653           | Continua con ind salto            |
| b652: ca       | dex      |                  | Decrementa ind. HI -1             |
| b653: 65 66    | adc      | #\$66 [ indice ] | Somma il salto a LO (OP3)         |
| b655: 90 01    | bcc *    | \$b658           | Se nessun OVERFLOW salta corr. HI |
| b657: e8       | inx      |                  | Correzione OVERFLOW per ind. HI   |
| b658: 60       | rts      |                  |                                   |

-----  
ROUTINE PER INDIRIZZI E LUNGHEZZA

|                |       |          |   |
|----------------|-------|----------|---|
| b659: a8       | tay   |          | Metti cod. prova in Y                   |
| b65a: 4a       | lsr a |          | Sposta bit0 e controlla                 |
| b65b: 90 0b    | bcc * | \$b668   | Se bit 0=0 Ok                           |
| b65d: 4a       | lsr a |          | Sposta bit 1                            |
| b65e: b0 17    | bcs * | \$b677   | Se bit 1=1 errore                       |
| b660: c9 22    | cmp   | #\$22    | Controllo pos. cod.\$89 usato in uscita |
| b662: f0 13    | beq * | \$b677   | Se pos. controllo non valid.            |
| b664: 29 07    | and   | #\$07    | Mask bits 3-7                           |
| b666: 09 80    | ora   | #\$80    | Mask bit 7                              |
| b668: 4a       | lsr a |          | Dividi cont A.per 2                     |
| b669: aa       | tax   |          | e visualizza in reg. X                  |
| b66a: bd c3 b6 | lda * | \$b6c3,x | Carica byte come tav. rif. ind.         |
| b66d: b0 04    | bcs * | \$b673   | Se c'e' resto da div. preced. salta     |
| b66f: 4a       | lsr a |          | Copia il contenuto del nibble UP        |
| b670: 4a       | lsr a |          | (bits 4-7) nel bit (4-3)                |
| b671: 4a       | lsr a |          | corrispondente a nibble LO              |
| b672: 4a       | lsr a |          | ***                                     |
| b673: 29 0f    | and   | #\$0f    | Esegui MASK OUT per bit 4-7             |
| b675: d0 04    | bne * | \$b67b   | Se diverso da 0 OK                      |
| b677: a0 80    | ldy   | #\$80    | Se codice non valido Carica Y con \$80  |
| b679: a9 00    | lda   | #\$00    | e l' A con \$00                         |
| b67b: aa       | tax   |          | Spostam. non eseguito in X              |
| b67c: bd 07 b7 | lda * | \$b707,x | Carica chiave di ind. da tabella ed     |
| b67f: 8d aa 0a | sta   | \$0aaa   | immetti in \$0AAA                       |
| b682: 29 03    | and   | #\$03    | Esegui MASK OUT per BIT 2-7             |
| b684: 8d ab 0a | sta   | \$0aab   | Immag. bits 0 e 1                       |
| b687: 98       | tya   |          | Copia codice testo in A                 |
| b688: 29 8f    | and   | #\$8f    | Esegui MASK OUT per bit 4,5,6           |
| b68a: aa       | tax   |          | Immetti valore preced in X              |
| b68b: 98       | tya   |          | Copia codice testo in A                 |
| b68c: a0 03    | ldy   | #\$03    | Inizial. cont. ciclo con 3              |

|             |              |                                   |
|-------------|--------------|-----------------------------------|
| b68e: e0 8a | cpx #\$8a    | Confronta val Mask con \$8A       |
| b690: f0 0b | beq * \$b69d | Se uguale salta                   |
| b692: 4a    | lsr a        | Dividi cont. A per 2              |
| b693: 90 08 | bcc * \$b69d | Se non c'è resto alla div. salta  |
| b695: 4a    | lsr a        | Dividi per 2 cont di A            |
| b696: 4a    | lsr a        | Idem                              |
| b697: 09 20 | ora #\$20    | Fissa il bit 5 di A               |
| b699: 88    | dey          | Decrementa il cont. di ciclo di 1 |
| b69a: d0 fa | bne * \$b696 | Se <>0 continua il ciclo          |
| b69c: c8    | iny          | Incrementa di 1 il cont. ciclo    |
| b69d: 88    | dey          | Decrementa cont ciclo 1           |
| b69e: d0 f2 | bne * \$b692 | Se <>0 esegui ancora la div.      |
| b6a0: 60    | rts          |                                   |

#### INVIO DI UN CARATTERE

|                |                        |                                      |
|----------------|------------------------|--------------------------------------|
| b6a1: a8       | tay                    | Cod. comando come vis in Y           |
| b6a2: b9 21 b7 | lda * \$b721,y         | Carica byte da tavola 1              |
| b6a5: 85 63    | sta \$63 [ fac ]       | e metti in OP2 (LO)                  |
| b6a7: b9 61 b7 | lda * \$b761,y         | Carica byte da tavola 2              |
| b6aa: 85 64    | sta \$64 [ rightflag ] | e metti in OP2 (HI)                  |
| b6ac: a9 00    | lda #\$00              | Carica A con \$00                    |
| b6ae: a0 05    | ldy #\$05              | Esegui lo spostamento verso sinistra |
| b6b0: 06 64    | asl \$64 [ rightflag ] | dei 5 bits di OP2 e mettili in A     |
| b6b2: 26 63    | rol \$63 [ fac ]       | Ciclo per seguire quanto sopra       |
| b6b4: 2a       | rol a                  | Idem                                 |
| b6b5: 88       | dey                    | Controllo per vedere se la somma del |
| b6b6: d0 f8    | bne * \$b6b0           | num. \$3F restituisce un car. valido |
| b6b8: 69 3f    | adc #\$3f              | o un ?                               |
| b6ba: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a subr. BSOUT                    |
| b6bd: ca       | dex                    | Esegui 3 cicli per le 3 lettere dal  |
| b6be: d0 ec    | bne * \$b6ac           | val. in OP2 (16-bit ind. LO/HI)      |
| b6c0: 4c a8 b8 | jmp * \$b8a8           | Vai a \$b8a8 e poi RTS               |

#### TAVOLA DI RIFERIMENTO INDIRIZZI

|                               |                 |
|-------------------------------|-----------------|
| b6c3: 40 02 45 03 d0 08 40 09 | @ e P @         |
| b6cb: 30 22 45 33 d0 08 40 09 | 0"e3P @         |
| b6d3: 40 02 45 33 d0 08 40 09 | @ e3P @         |
| b6db: 40 02 45 b3 d0 08 40 09 | @ e.P @         |
| b6e3: 00 22 44 33 d0 8c 44 00 | "d3P d          |
| b6eb: 11 22 44 33 d0 8c 44 9a | <crsr d> "d3P d |
| b6f3: 10 22 44 33 d0 08 40 09 | "d3P @          |
| b6fb: 10 22 44 33 d0 08 40 09 | "d3P @          |
| b703: 62 13 78 a9             | " <home> 8      |

#### TIPI DI INDIRIZZI E LUNGHEZZA CHIAVI

|                               |  |
|-------------------------------|--|
| b707: 00 21 81 82 00 00 59 4d | In questo gruppo sono presenti oltre che gli indirizzi e la lunghezza chiavi anche le tavole mnemoniche relative ai comandi tipi dell'Assembler.<br>Si tratta di 2 tavole per cui un byte della tavola 1 restituisce, in rapporto al corrispondente valore |
| b70f: 91 92 86 4a 85 9d 2c 29 |  |
| b717: 2c 23 28 24 59 00 58 24 |  |
| b71f: 24 00 1c 8a 1c 23 5d 8b |  |
| b727: 1b a1 9d 8a 1d 23 9d 8b |  |
| b72f: 1d a1 00 29 19 ae 69 a8 |  |
| b737: 19 23 24 53 1b 23 24 53 |  |
| b73f: 19 a1 00 1a 5b 5b a5 69 |  |

```

b747: 24 24 ae ae a8 ad 29 00
b74f: 7c 00 15 9c 6d 9c a5 69
b757: 29 53 84 13 34 11 a5 69
b75f: 23 a0 d8 62 5a 48 26 62
b767: 94 88 54 44 c8 54 68 44
b76f: e8 94 00 b4 08 84 74 b4
b777: 28 6e 74 f4 cc 4a 72 f2
b77f: a4 8a 00 aa a2 a2 74 74
b787: 74 72 44 68 b2 32 b2 00
b78f: 22 00 1a 1a 26 26 72 72
b797: 88 c8 c4 ca 26 48 44 44
b79f: a2 c8 0d 20 20 20

```

della tavola 2 un codice di 16 bit  
come carattere di 3 lettere

#### CONTROLO PER I SEPARATORI FRA I COMANDI

```

b7a5: c6 7a      dec $7a      [ dsdec ]   Punt. buffer input per car.
b7a7: 20 ce b7   jsr * $b7ce   Prendi operando OPl
b7aa: b0 16      bcs * $b7c2   Controllo per CARRY settato=errore
b7ac: 20 e7 b8   jsr * $b8e7   Ripris. ultimo car. letto
b7af: d0 09      bne * $b7ba   Se e' un fine comando continua
b7b1: c6 7a      dec $7a      [ dsdec ]   Come b7a5
b7b3: ad b4 0a   lda $0ab4     Carica flag ric. errori
b7b6: d0 11      bne * $b7c9   Se <>0 OK ed esci
b7b8: f0 0d      beq * $b7c7   Operando non valido. Errore e esci
b7ba: c9 20      cmp # $20     Contr. se carat. letto e' SPAZIO
b7bc: f0 0b      beq * $b7c9   Operatore valido, OK e esci
b7be: c9 2c      cmp # $2c     Contr. se caratt. e' una virgola
b7c0: f0 07      beq * $b7c9   Se e' un separatore valido, OK esci
b7c2: 68         pla          Sull' indir. nello STACK e' stato eseg
b7c3: 68         pla          un CLEAR, un ? viene vis. ed il progr.
b7c4: 4c bc b0   jmp * $b0bc     salta al ciclo di att. INPUT
b7c7: 38         sec          Contr. per carry set=errore

```

```

b7c8: 24         $          Salta a $b7ca

```

```

b7c9: 18         clc          Esegui un CLEAR di carry. Segn per OK
b7ca: ad b4 0a   lda $0ab4     Carica flag ric. errore
b7cd: 60         rts

```

#### VALUTAZIONE PARAMETRI DI UN COMANDO

```

b7ce: a9 00      lda # $00     Carica A con $0 per iniz. param
b7d0: 85 60      sta $60      [ tenexp ]   Esegui CLR su param com. di 3 Byte
b7d2: 85 61      sta $61      [ lowtr ]    N. 1 (OPl), inpag. zero da indirizzo
b7d4: 85 62      sta $62      [ expsgn ]   $62 (HI) a $(60) (LO)
b7d6: 8d b4 0a   sta $0ab4     Imm. temp. memo. per contr. errori
b7d9: 8a         txa          Metti X in A
b7da: 48         pha          Salva su STACK
b7db: 98         tya          Metti Y in A
b7dc: 48         pha          Salva su STACK
b7dd: 20 e9 b8   jsr * $b8e9   Contr. buffer input x fine comando
b7e0: d0 03      bne * $b7e5   Carat.: e ? non sono mark di fine, pros
b7e2: 4c 72 b8   jmp * $b87e   Esegui rout. usc. con clear di carry
b7e5: c9 20      cmp # $20     (puntatore). E' uno SPAZIO
b7e7: f0 f4      beq * $b7dd   Se si allora leggi pross. car.
b7e9: ad 03      ldx # $03     Carica vis. per 4 car. di conv.
b7eb: d2 f5 b0   cmp * $b0f5, x  Controllo di conversione (%&+$)
b7ee: f0 06      beq * $b7f6   fino a tr. caratt. di conv.

```

|       |          |      |                   |  |
|-------|----------|------|-------------------|--|
| b7f0: | ca       | dex  |                   | Visual. tavola calc. -1                |
| b7f1: | 10 f8    | bpl  | * \$b7eb          | Ciclo di ricerca attr. tavola          |
| b7f3: | e8       | inx  |                   | Reg X fissato a \$0                    |
| b7f4: | c6 7a    | dec  | \$7a [ dsdec ]    | Scarica punt. al buffer di input -1    |
| b7f6: | bc 8a b8 | ldy  | * \$b88a,x        | Carica reg.Y con sist. base            |
| b7f9: | bd 8e b8 | lda  | * \$b88e,x        | Carica A con fatt. di multiplic.       |
| b7fc: | 8d b6 0a | sta  | \$0ab6            | per sistema base ed immag.             |
| b7ff: | 20 e9 b8 | jsr  | * \$b8e9          | Contr. buffer input fine com. per : ?  |
| b802: | f0 7a    | beq  | * \$b87e          | Uscita per determ. operando            |
| b804: | 38       | sec  |                   | Carry settato per sottrazione          |
| b805: | e9 30    | sbcc | #\$30             | Conversione                            |
| b807: | 90 75    | bcc  | * \$b87e          | Se car.<0 esci                         |
| b809: | c9 0a    | cmp  | #\$0a             | Contr. per car se e' num fra 0 e 9     |
| b80b: | 90 06    | bcc  | * \$b813          | Se posit. vai a conv. esa              |
| b80d: | e9 07    | sbcc | #\$07             | Conv. di num. esa A-F                  |
| b80f: | c9 10    | cmp  | #\$10             | Se il valore non e' fra 0-F esci       |
| b811: | b0 6b    | bcs  | * \$b87e          | dalla subr. determ. operando           |
| b813: | 8d b5 0a | sta  | \$0ab5            | Immag. i valori esa                    |
| b816: | cc b5 0a | cpy  | \$0ab5            | Confronta la base con val. esa         |
| b819: | 90 61    | bcc  | * \$b87c          | Se val. base < carat. errore           |
| b81b: | f0 5f    | beq  | * \$b87c          | Se val. base = carat. errore           |
| b81d: | ee b4 0a | inc  | \$0ab4            | Incr. di 1 Byte ric. errore            |
| b820: | c0 0a    | cpy  | #\$0a             | Contr. per scelta input decimale       |
| b822: | d0 0a    | bne  | * \$b82e          | Se neg. vai a iniz. decim              |
| b824: | a2 02    | ldx  | #\$02             | Fissa a 2 il cont. di ciclo            |
| b826: | b5 60    | lda  | \$60,x [ tenexp ] | Copia l' operando di 3 byte (OPl)      |
| b828: | 9d b7 0a | sta  | \$0ab7,x          | nell' operando temporaneo              |
| b82b: | ca       | dex  |                   | per ingresso ind. decimale             |
| b82c: | 10 f8    | bpl  | * \$b826          | con valori (\$0AB9=HI,\$AB7=LO)        |
| b82e: | ae b6 0a | ldx  | \$0ab6            | Carica contatore per fattore di        |
| b831: | 06 60    | asl  | \$60 [ tenexp ]   | moltiplicazione. #-Byte                |
| b833: | 26 61    | rol  | \$61 [ lowtr ]    | Operando OPl                           |
| b835: | 26 62    | rol  | \$62 [ expsgn ]   | Moltiplica per 2                       |
| b837: | b0 43    | bcs  | * \$b87c          | Contr. per overflow, se pres= errore   |
| b839: | ca       | dex  |                   | Moltiplica per 2 cont. di ciclo        |
| b83a: | d0 f5    | bne  | * \$b831          | Ciclo a OPl per multipl.               |
| b83c: | c0 0a    | cpy  | #\$0a             | Contr. per n. base se decimale         |
| b83e: | d0 22    | bne  | * \$b862          | Se no vai a conv. decimale             |
| b840: | 0e b7 0a | asl  | \$0ab7            | Ciclo di conv. dec: i 3 byte di opera  |
| b843: | 2e b8 0a | rol  | \$0ab8            | temporaneo in \$AB9-\$AB7              |
| b846: | 2e b9 0a | rol  | \$0ab9            | sono moltiplicati per 2                |
| b849: | b0 31    | bcs  | * \$b87c          | Se overflow, errore                    |
| b84b: | ad b7 0a | lda  | \$0ab7            | --In questa routine viene eseguita la  |
| b84e: | 65 60    | adc  | \$60 [ tenexp ]   | somma dei 3 Bytes operando temporaneo  |
| b850: | 85 60    | sta  | \$60 [ tenexp ]   | negli ind. \$0AB9-\$0AB8-\$AB          |
| b852: | ad b8 0a | lda  | \$0ab8            | per il cont. di 3 Bytes dell' operando |
| b855: | 65 61    | adc  | \$61 [ lowtr ]    | OPl che viene controllato              |
| b857: | 85 61    | sta  | \$61 [ lowtr ]    | per la rilevazione di possibili        |
| b859: | ad b9 0a | lda  | \$0ab9            | OVERFLOW                               |
| b85c: | 65 62    | adc  | \$62 [ expsgn ]   | Il risultato della somma sara'         |
| b85e: | 85 62    | sta  | \$62 [ expsgn ]   | impresso in OPl                        |
| b860: | b0 1a    | bcs  | * \$b87c          | Se c'e' Overflow allora errore         |
| b862: | 18       | clc  |                   | Esec di CLEAR CARRY                    |
| b863: | ad b5 0a | lda  | \$0ab5            | Carica vaolre carat.                   |
| b866: | 65 60    | adc  | \$60 [ tenexp ]   | Aggiungi il valore dell' ultimo        |
| b868: | 85 60    | sta  | \$60 [ tenexp ]   | posizione di OPl                       |
| b86a: | 8a       | txa  |                   | Carica A con 0                         |
| b86b: | 65 61    | adc  | \$61 [ lowtr ]    | Controllo di Overflow sommando l'      |
| b86d: | 85 61    | sta  | \$61 [ lowtr ]    | ultima significat. pos. di OPl         |
| b86f: | 8a       | txa  |                   | Carica 0 in A                          |

```

b870: 65 62      adc $62      [ expsgn ]   Controllo di Overflow nella posizione
b872: 85 62      sta $62      [ expsgn ]   di OPl
b874: b0 06      bcs * $b87c   Se contr. overflow pos. errore
b876: 29 f0      and #$f0      Operaz. di mask out su nibble LO
b878: d0 02      bne * $b87c   Se nibble sup. <>0, errore
b87a: f0 83      beq * $b7ff   Valuta pos. succ. operando
b87c: 38        sec          Fissa il carry per seqn. di ril.errore

```

b87d: 24 \$ Salta a \$B87F

|       |    |       |            |                                      |
|-------|----|-------|------------|--------------------------------------|
| b87e: | 18 |       | clc        | Clear di carry per segn di par. OK   |
| b87f: | 8c | b6 0a | sty \$0ab6 | Immag. base sist. numerico in uso    |
| b882: | 68 |       | pla        | Ripristina il vecchio contenuto di Y |
| b883: | a8 |       | tay        | dallo stack                          |
| b884: | 68 |       | pla        | Ripristina il vecchio contenuto di X |
| b885: | aa |       | tax        | dallo Stack                          |
| b886: | ad | b4 0a | lda \$0ab4 | Carica A con punt. a errore          |
| b889: | 60 |       | rts        |                                      |

b88a: 10 0a 08 02 04 03 03 01                      Basi dei sistemi numerici

VISUALIZZAZIONE CONTENUTI DEL S. O.

|       |          |              |            |  |
|-------|----------|--------------|------------|--|
| b892: | a5 68    | lda \$68     | [ facsgn ] | Carica A con Byte di banco (HI di OP3) |
| b894: | 20 d2 b8 | jsr * \$b8d2 |            | A in 2 byte ASCII: HI=A,LO=X           |
| b897: | 8a       | txa          |            | Codice ASCII per valore LO in A        |
| b898: | 20 d2 ff | jsr * \$ffd2 | [ bsout ]  | Vai a sub. BSOUT                       |
| b89b: | a5 66    | lda \$66     | [ indice ] | Carica A con Byte di indir.(LO di OP3) |
| b89d: | a6 67    | ldx \$67     | [ facmo ]  | Carica X con Byte di indir.(HI di OP3) |
| b89f: | 48       | pha          |            | Immag. A in STACK                      |
| b8a0: | 8a       | txa          |            | Trasf in A valore di OP3 (ind. HI)     |
| b8a1: | 20 c2 b8 | jsr * \$b8c2 |            | Vis. A in 2 car. ASCII                 |
| b8a4: | 68       | pla          |            | Carica A con Ind. LO (OP3)             |
| b8a5: | 20 c2 b8 | jsr * \$b8c2 |            | Vis. A con 2 car. ASCII                |
| b8aa: | a9 20    | lda #\$20    |            | Carica un Blank in A                   |
| b8aa: | 4c d2 ff | jmp * \$ffd2 | [ bsout ]  | Vai a subr. BSOUT                      |
| b8ad: | 20 7d ff | jsr * \$ff7d | [ primm ]  | Vai a subr. PRIMM                      |

b8b0: 0d 91 00 <cr> <crsr u> Costanti

```

b8b4: a9 0d      lda #$0d          Carica codice car. Cr in A
b8b6: 4c d2 ff     jmp * $ffd2 [ bsout ] Vai a subr. BSOUT
b8b9: 20 7d ff     jsr * $ff7d [ primm ] Vai a subr. PRIMM

```

```
b8bc: 0d 1b 51 20 00 <cr> <esc> q
```

COSTANTI DEL MONITOR PER RITORNO  
CARRELLO E CLEAR DELLA SUCC. LINEA

```
b8cl: 60      rts      Chiusura e ritorno da subroutine
```

CONVERSIONE CONTENUTI ACCUMULATORE

|                |                        |                                     |
|----------------|------------------------|-------------------------------------|
| b8c2: 8e af 0a | stx \$0aaf             | Immag. cont.vecchi reg. X           |
| b8c5: 20 d2 b8 | jsr * \$b8d2           | Immetti A in 2 byte ASCII:HI=A,LO=X |
| b8c8: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a subr. BSOUT                   |
| b8cb: 8a       | txa                    | Carica in A car. da X               |

|       |          |                        |                                       |
|-------|----------|------------------------|---------------------------------------|
| b8cc: | ae af 0a | ldx \$0aaf             | Riprist. reg X                        |
| b8cf: | 4c d2 ff | jmp * \$ffd2 [ bsout ] | Vai a sub. BSOUT                      |
| b8d2: | 48       | pha                    | Imm. temp. cont. A                    |
| b8d3: | 20 dc b8 | jsr * \$b8dc           | Converti nibble basso in ASCII        |
| b8d6: | aa       | tax                    | Prec. val in reg X                    |
| b8d7: | 68       | pla                    | Riprist. cont. A                      |
| b8d8: | 4a       | lsr a                  | Rout. di spostam. a destra di 4 volte |
| b8d9: | 4a       | lsr a                  | in modo tale che il nibble HI (bits   |
| b8da: | 4a       | lsr a                  | 4-7) sia spostato nel LO nibble       |
| b8db: | 4a       | lsr a                  | ***                                   |
| b8dc: | 29 0f    | and #\$0f              | Mask out su nibble HI                 |
| b8de: | c9 0a    | cmp #\$0a              | Controllo per n. da 0 a 9             |
| b8e0: | 90 02    | bcc * \$b8e4           | Se posit. crea codice ASCII           |
| b8e2: | 69 06    | adc #\$06              | Convers car. per A-F                  |
| b8e4: | 69 30    | adc #\$30              | Gener. cod. ASCII per cont. A         |
| b8e6: | 60       | rts                    |                                       |

## ROUT. DI CARICAMENTO DA BUFFER DI INPUT

|       |          |              |           |                                      |
|-------|----------|--------------|-----------|--------------------------------------|
| b8e7: | c6 7a    | dec \$7a     | [ dsdec ] | Decr. -1 buffer input                |
| b8e9: | 8e af 0a | stx \$0aaf   |           | Imm. cont. reg. X                    |
| b8ec: | a6 7a    | ldx \$7a     | [ dsdec ] | Carica X in buff. input (vis)        |
| b8ee: | bd 00 02 | lda \$0200,x |           | Car. car dal buffer input comandi    |
| b8f1: | f0 06    | beq * \$b8f9 |           | E' stato trovato un \$00 (fine com.) |
| b8f3: | c9 3a    | cmp #\$3a    |           | Contr. se car. letto e' :            |
| b8f5: | f0 02    | beq * \$b8f9 |           | Se si esci con flag = fissato        |
| b8f7: | c9 3f    | cmp #\$3f    |           | Contr. se car. letto e' un ?         |
| b8f9: | 08       | php          |           | Imm. val. del flag =                 |
| b8fa: | e6 7a    | inc \$7a     | [ dsdec ] | Incr buffer input di 1 per succ. car |
| b8fc: | ae af 0a | ldx \$0aaf   |           | Ripristina X                         |
| b8ff: | 28       | plp          |           | Ripristina lo stato del flag =       |
| b900: | 60       | rts          |           |                                      |

## ROUTINE DI COPIA

|       |       |          |            |                                      |
|-------|-------|----------|------------|--------------------------------------|
| b901: | a5 60 | lda \$60 | [ tenexp ] | Questa routine serve per copiare il  |
| b903: | 85 66 | sta \$66 | [ indice ] | contenuto degli indirizzi di memoria |
| b905: | a5 61 | lda \$61 | [ lowtr ]  | \$62-\$61-\$60, cioe' di OP1 in OP3  |
| b907: | 85 67 | sta \$67 | [ facmo ]  | cioe' nelle locazioni \$68-\$67-\$66 |
| b909: | a5 62 | lda \$62 | [ expsgn ] | ***                                  |
| b90b: | 85 68 | sta \$68 | [ facsgn ] | ***                                  |
| b90d: | 60    | rts      |            |                                      |

## ROUTINE DI IMMAGAZZINAMENTO IN OP1

|       |       |          |            |                               |
|-------|-------|----------|------------|-------------------------------|
| b90e: | 38    | sec      |            | Sottrazione: carry set        |
| b90f: | a5 60 | lda \$60 | [ tenexp ] | Carica A con OP1 LO           |
| b911: | e5 66 | sbc \$66 | [ indice ] | Sottrai il val. +basso da OP3 |
| b913: | 85 60 | sta \$60 | [ tenexp ] | Imm. risul. in OP1(LO)        |
| b915: | a5 61 | lda \$61 | [ lowtr ]  | Carica A con \$61             |
| b917: | e5 67 | sbc \$67 | [ facmo ]  | Sottrai \$67 da OP3           |
| b919: | 85 61 | sta \$61 | [ lowtr ]  | Imm. ris. in OP3              |
| b91b: | a5 62 | lda \$62 | [ expsgn ] | Carica A con OP1 (HI)         |
| b91d: | e5 68 | sbc \$68 | [ facsgn ] | Sottrai \$68 da OP3           |
| b91f: | 85 62 | sta \$62 | [ expsgn ] | Imma. ris. in OP1 (HI)        |
| b921: | 60    | rts      |            |                               |

## ROUTINE DI SOTTRAZIONE

|       |       |           |  |                                       |
|-------|-------|-----------|--|---------------------------------------|
| b922: | a9 01 | lda #\$01 |  | Carica A con 1 e imm.lo come minuendo |
|-------|-------|-----------|--|---------------------------------------|



|       |          |              |               |                                       |
|-------|----------|--------------|---------------|---------------------------------------|
| b924: | 8d af 0a | sta \$0aaf   |               | in \$0AAF                             |
| b927: | 38       | sec          |               | Carry set per sottrazione             |
| b928: | a5 60    | lda \$60     | [ tenexp ]    | Carica A con OPl LO                   |
| b92a: | ed af 0a | sbc \$0aaf   |               | Sottrai minuendo da OPl LO            |
| b92d: | 85 60    | sta \$60     | [ tenexp ]    | Scrivi il risult. della sottrazione   |
| b92f: | a5 61    | lda \$61     | [ lowtr ]     | Carica A con \$61 (OPl MID)           |
| b931: | e9 00    | sbc #\$00    |               | Controllo per OVERFLOW                |
| b933: | 85 61    | sta \$61     | [ lowtr ]     | Scrivi il risult. della sottrazione   |
| b935: | a5 62    | lda \$62     | [ expsgn ]    | Carica A con OPl (HI)                 |
| b937: | e9 00    | sbc #\$00    |               | Controllo per Overflow                |
| b939: | 85 62    | sta \$62     | [ expsgn ]    | Scrive il risul. della sub.           |
| b93b: | 60       | rts          |               |                                       |
| ----- |          |              |               |                                       |
| b93c: | 38       | sec          |               | Esecuzione dell' operazione di sottra |
| b93d: | a5 63    | lda \$63     | [ fac ]       | zione della costante 1 dall' operando |
| b93f: | e9 01    | sbc #\$01    |               | OP2 nelle locazioni di memoria        |
| b941: | 85 63    | sta \$63     | [ fac ]       | \$65,\$64,\$63                        |
| b943: | a5 64    | lda \$64     | [ rightflag ] | ***                                   |
| b945: | e9 00    | sbc #\$00    |               | ***                                   |
| b947: | 85 64    | sta \$64     | [ rightflag ] | ***                                   |
| b949: | a5 65    | lda \$65     | [ facmoh ]    | ***                                   |
| b94b: | e9 00    | sbc #\$00    |               | ***                                   |
| b94d: | 85 65    | sta \$65     | [ facmoh ]    | ***                                   |
| b94f: | 60       | rts          |               |                                       |
| ----- |          |              |               |                                       |
| b950: | a9 01    | lda #\$01    |               | Addizione dei contenuti di A in OP3   |
| b952: | 18       | clc          |               | esguita con controlli di OVERFLOW     |
| b953: | 65 66    | adc \$66     | [ indice ]    |                                       |
| b955: | 85 66    | sta \$66     | [ indice ]    |                                       |
| b957: | 90 06    | bcc * \$b95f |               |                                       |
| b959: | e6 67    | inc \$67     | [ facmo ]     |                                       |
| b95b: | d0 02    | bne * \$b95f |               |                                       |
| b95d: | e6 68    | inc \$68     | [ facsgn ]    |                                       |
| b95f: | 60       | rts          |               |                                       |
| ----- |          |              |               |                                       |
| b960: | 38       | sec          |               | Subroutine di sottrazione della       |
| b961: | a5 66    | lda \$66     | [ indice ]    | costante 1 da OP3 di locazione        |
| b963: | e9 01    | sbc #\$01    |               | \$66-\$67-\$68                        |
| b965: | 85 66    | sta \$66     | [ indice ]    | ***                                   |
| b967: | a5 67    | lda \$67     | [ facmo ]     | ***                                   |
| b969: | e9 00    | sbc #\$00    |               | ***                                   |
| b96b: | 85 67    | sta \$67     | [ facmo ]     | ***                                   |
| b96d: | a5 68    | lda \$68     | [ facsgn ]    | ***                                   |
| b96f: | e9 00    | sbc #\$00    |               | ***                                   |
| b971: | 85 68    | sta \$68     | [ facsgn ]    | ***                                   |
| b973: | 60       | rts          |               |                                       |
| ----- |          |              |               |                                       |
| b974: | b0 0c    | bcs * \$b982 |               | Return se carry set                   |
| b976: | a5 60    | lda \$60     | [ tenexp ]    | Carica A con OPl ind LO               |
| b978: | a4 61    | ldy \$61     | [ lowtr ]     | Carica Y con OPl ind MID              |
| b97a: | a6 62    | ldx \$62     | [ expsgn ]    | Carica X con OPl ind HI               |
| b97c: | 85 04    | sta \$04     | [ pc-lo ]     | Imm in ind. di Pag. 0 per PC-LO (A)   |
| b97e: | 84 03    | sty \$03     | [ pc-hi ]     | Imm in ind. di Pag. 0 per PC-HI (Y)   |
| b980: | 86 02    | stx \$02     | [ bank ]      | Imm in ind. di Pag. 0 per n.banco(X)  |
| b982: | 60       | rts          |               |                                       |
| ----- |          |              |               |                                       |
| b983: | b0 2a    | bcs * \$b9af |               | Esci se errore in par. comando        |
| b985: | 20 01 b9 | jsr * \$b901 |               | Copia cont di OPl in OP3              |
| b988: | 20 a7 b7 | jsr * \$b7a7 |               | Vai a operando in OPl                 |
| b98b: | b0 22    | bcs * \$b9af |               | Se operando non valido errore         |

```

b98d: a5 60      lda $60      [ tenexp ]   Copia i contenuti dell'operando
b98f: 8d b7 0a    sta $0ab7
b992: a5 61      lda $61      [ lowtr ]     di 3 bytes (OP1) nelle seguenti locaz
b994: 8d b8 0a    sta $0ab8                                     di memoria $0AB9 - $0AB8 - $0AB7
b997: a5 62      lda $62      [ expsgn ]    ***
b999: 8d b9 0a    sta $0ab9                                     ***
b99c: 20 0e b9    jsr * $b90e                                     Diff. fra OP1- OP3 in OP1

```

```

b99f: a5 60      lda $60      [ tenexp ]   Copia i contenuti dei 3 bytes dell'
b9a1: 85 63      sta $63      [ fac ]       operando OP1 nell'operando OP3
b9a3: a5 61      lda $61      [ lowtr ]     Se da questa operazione risulta che
b9a5: 85 64      sta $64      [ rightflag ] OP1 maggiore di OP3 esegui uscita per
b9a7: a5 62      lda $62      [ expsgn ]    errore
b9a9: 85 65      sta $65      [ facmoh ]    ***
b9ab: 90 02      bcc * $b9af                                     Esegui un clear del Carry
b9ad: 18         clc

```

```

-----
b9ae: 24         $                                     Routine di uscita per ril errore
-----

```

```

b9af: 38         sec                                     Fissa il carry
b9b0: 60         rts

```

#### ROUTINE DI USCITA CONVERSIONE COMANDI

```

b9b1: 20 a5 b7    jsr * $b7a5                                     Conversione valore in OP1
b9b4: 20 b9 b8    jsr * $b8b9                                     Uscita simboli Cr, Esc-Q, Space
b9b7: a9 24      lda #$24                                     Carica A con <$>
b9b9: 20 d2 ff    jsr * $ffd2 [ bsout ]   Vai a BSOUT
b9bc: a5 62      lda $62      [ expsgn ]   Carica HI del val.di conversione
b9be: f0 07      beq * $b9c7                                     Se $00 cancella gli altri 2 gruppi
b9c0: 20 d2 b8    jsr * $b8d2                                     A in 2 Bytes ASCII: HI=A, LO=X
b9c3: 8a         txa                                             Metti in A ASCII per nibble LO
b9c4: 20 d2 ff    jsr * $ffd2 [ bsout ]   Vai a BSOUT
b9c7: a5 60      lda $60      [ tenexp ]   Carica LO dei 3 bytes di convers.
b9c9: a6 61      ldx $61      [ lowtr ]    Carica valore centrale
b9cb: 20 9f b8    jsr * $b89f                                     Uscita dei valori come car. ASCII
b9ce: 20 b9 b8    jsr * $b8b9                                     di Cr, Esc-Q, Space
b9d1: a9 2b      lda #$2b                                             Metti + in A
b9d3: 20 d2 ff    jsr * $ffd2 [ bsout ]   Vai a BSOUT
b9d6: 20 07 ba    jsr * $ba07                                     Converti OP1 in decimale
b9d9: a9 00      lda #$00                                             Cancella segnalatore
b9db: a2 08      ldx #$08                                             Uscita di 8 caratteri
b9dd: a0 03      ldy #$03                                             Y contiene il digit di output
b9df: 20 5d ba    jsr * $ba5d                                     Uscita di AA0-AA3 come n. decimale
b9e2: 20 b9 b8    jsr * $b8b9                                     Uscita deicaratteri c.s.
b9e5: a9 26      lda #$26                                             Metti & in A
b9e7: 20 d2 ff    jsr * $ffd2 [ bsout ]   Vai a Bscout
b9ea: a9 00      lda #$00                                             Cancella segnalatore
b9ec: a2 08      ldx #$08                                             Uscita di 8 caratteri
b9ee: a0 02      ldy #$02                                             Y contiene i 3 bit di output
b9f0: 20 47 ba    jsr * $ba47                                     Uscita di AA0-AA3 come ottale
b9f3: 20 b9 b8    jsr * $b8b9                                     Uscita di Cr, Esc-Q, Space
b9f6: a9 25      lda #$25                                             Metti % in A
b9f8: 20 d2 ff    jsr * $ffd2 [ bsout ]   Vai a BSOUT
b9fb: a9 00      lda #$00                                             Cancella segnalatore
b9fd: a2 18      ldx #$18                                             Uscita di 18 caratteri
b9ff: a0 00      ldy #$00
ba01: 20 47 ba    jsr * $ba47                                     ***

```

ba04: 4c 8b b0 jmp \* \$b08b

Vai a ciclo di attesa

-----  
ROUTINE DI CONVERSIONE

```

ba07: 20 01 b9 jsr * $b901
ba0a: a9 00 lda #$00
ba0c: a2 07 ldx #$07
ba0e: 9d a0 0a sta $0aa0,x
ba11: ca dex
ba12: 10 fa bpl * $ba0e
ba14: ee a7 0a inc $0aa7
ba17: a0 17 ldy #$17
ba19: 08 php
ba1a: 78 sei
ba1b: f8 sed
ba1c: 46 68 lsr $68 [ facsgn ]
ba1e: 66 67 ror $67 [ facmo ]
ba20: 66 66 ror $66 [ indice ]
ba22: 90 0f bcc * $ba33
ba24: 18 clc
ba25: a2 03 ldx #$03
ba27: bd a4 0a lda $0aa4,x
ba2a: 7d a0 0a adc $0aa0,x
ba2d: 9d a0 0a sta $0aa0,x
ba30: ca dex
ba31: 10 f4 bpl * $ba27
ba33: 18 clc
ba34: a2 03 ldx #$03
ba36: bd a4 0a lda $0aa4,x
ba39: 7d a4 0a adc $0aa4,x
ba3c: 9d a4 0a sta $0aa4,x
ba3f: ca dex
ba40: 10 f4 bpl * $ba36
ba42: 88 dey
ba43: 10 d7 bpl * $balc
ba45: 28 plp
ba46: 60 rts

```

Copia contenuti di OPl in OP3  
 Esegui un Clear AA0-AA3 per i  
 numeri decimali  
 Esegui un Clear AA4-AA7 usati come  
 contatori per conversione decim.  
 Inizializza una pos. contatore  
 Incrementalo di 1  
 Ciclo di controllo per passi convers.  
 Immagazzina status interrupt e dec.  
 Disabilita tutti gli Interrupt  
 Abilita il modo decimale  
 Dividi per 2 il valore di 3 Byte  
 presente in OPl  
 \*\*\*  
 Se nessun resto div.salta add. decim  
 Esegui un Clear di carry per somma  
 Se c'è un resto della divisione  
 somma i contenuti del contatore  
 nella memoria di output  
 \*\*\*  
 \*\*\*  
 Esegui un Clear di Carry per somma dec  
 Moltiplica il contenuto di 4 bytes  
 del contatore per 2  
 Controllo che il contenuto del contat.  
 sia una potenza di 2  
 \*\*\*  
 \*\*\*  
 Decrementa cont. di 1  
 Controllo esec. tutti i passi  
 \*\*\*

-----  
ROUTINEDI CONVERSIONE

```

ba47: 48 pha
ba48: a5 60 lda $60 [ tenexp ]
ba4a: 8d a2 0a sta $0aa2
ba4d: a5 61 lda $61 [ lowtr ]
ba4f: 8d a1 0a sta $0aa1
ba52: a5 62 lda $62 [ expsgn ]
ba54: 8d a0 0a sta $0aa0
ba57: a9 00 lda #$00
ba59: 8d a3 0a sta $0aa3
ba5c: 68 pla
ba5d: 8d b4 0a sta $0ab4
ba60: 8c b6 0a sty $0ab6
ba63: ac b6 0a ldy $0ab6
ba66: a9 00 lda #$00
ba68: 0e a3 0a asl $0aa3
ba6b: 2e a2 0a rol $0aa2
ba6e: 2e a1 0a rol $0aa1
ba71: 2e a0 0a rol $0aa0
ba74: 2a rol a
ba75: 88 dey

```

A su Stack  
 Copia OPl in OPA  
 \*\*\*  
 Continua la copia  
 \*\*\*  
 \*\*\*  
 \*\*\*  
 Carica A con 00  
 Immetti in OPA  
 Ripistina cont. A da Stack  
 Fissa flag per 0  
 Immag. n. bit  
 Numero bit in y  
 Iniz. A  
 Esegui uno spostamento verso sinistra  
 del contenuto operando 4 byte  
 un bit per volta  
 \*\*\*  
 \*\*\*  
 Decrem. di 1 bit counter

|                |                      |                              |
|----------------|----------------------|------------------------------|
| ba76: 10 f0    | bpl * \$ba68         | Esegui il ciclo fino a trasf |
| ba78: a8       | tay                  | Trasf. risult. in Y          |
| ba79: d0 09    | bne * \$ba84         | Se diverso da 0 output       |
| ba7b: e0 01    | cpx #\$01            | Controllo per posizione      |
| ba7d: f0 05    | beq * \$ba84         | Se e' prima pos. uscita bit  |
| ba7f: ac b4 0a | ldy \$0ab4           | Carica flag di zero          |
| ba82: f0 08    | beq * \$ba8c         | Controlla che sia attivo     |
| ba84: ee b4 0a | inc \$0ab4           | Disabilita bit di zero       |
| ba87: 09 30    | ora #\$30            | Metti SPACE in A             |
| ba89: 20 d2 ff | jsr * \$ffd2 [bsout] | Vai a BSOUT                  |
| ba8c: ca       | dex                  | Ciclo per n. di bit          |
| ba8d: d0 d4    | bne * \$ba63         | Contr. se diverso da zero    |
| ba8f: 60       | rts                  |                              |

## COMANDO MONITOR @

|                |                        |                                     |
|----------------|------------------------|-------------------------------------|
| ba90: d0 03    | bne * \$ba95           | Contr. identif. ind.perif           |
| ba92: a2 08    | ldx #\$08              | Fissa lo standard a 8               |
| ba94: 2c       | .byte \$2c             | Vai a \$BA97                        |
| ba95: a6 60    | ldx * \$60             | N. di periferica da OPl             |
| ba97: e0 04    | cpx #\$04              | Contr. se n. perif min. di 4        |
| ba99: 90 65    | bcc * \$bb00           | Vis. ? vai ciclo att.input          |
| ba9b: e0 1f    | cpx #\$1f              | Contr.n.perif magg. di 30           |
| ba9d: b0 61    | bcs * \$bb00           | Come sopra                          |
| ba9f: 86 60    | stx \$60 [ tenexp ]    | Immag. n.perif in OPl               |
| baa1: a9 00    | lda #\$00              | Carica n. banco                     |
| baa3: 85 62    | sta \$62 [ expsgn ]    | Immag. in OPl byte banco            |
| baa5: 85 b7    | sta \$b7 [ fnlen ]     | Metti a 0 lunghezza nome file       |
| baa7: aa       | tax                    | Metti a 0 A e X                     |
| baa8: 20 68 ff | jsr * \$ff68 [setbnk]  | Vai a SETBNK                        |
| baab: 20 e9 b8 | jsr * \$b8e9           | Lettura di 1 car.da buffer di input |
| baae: c6 7a    | dec \$7a [ dsdec ]     | Punt. del buffer input -1 e visual. |
| bab0: c9 24    | cmp #\$24              | Controlla se car. e' \$             |
| bab2: f0 4f    | beq * \$bb03           | Se pos. uscita directory            |
| bab4: a9 00    | lda #\$00              | Metti in A n. di file logico        |
| bab6: a6 60    | ldx \$60 [ tenexp ]    | Metti in X n.perif da OPl           |
| bab8: a0 0f    | ldy \$0f               | Fissa indirizzo secondario          |
| baba: 20 ba ff | jsr * \$ffba [setlfs]  | Vai a SETLFS                        |
| babd: 20 c0 ff | jsr * \$ffc0 [open]    | Vai a OPEN                          |
| bac0: b0 32    | bcs * \$baf4           | Uscita per errore di OPEN           |
| bac2: a2 00    | ldx #\$00              | Fissa uscita file logico            |
| bac4: 20 c9 ff | jsr * \$ffc9 [ckout]   | Vai a CKOUT                         |
| bac7: b0 2b    | bcs * \$baf4           | Se ril. errore esci                 |
| bac9: a6 7a    | ldx \$7a [dsdec]       | Fissa punt. visualizz.              |
| bacb: e6 7a    | inc \$7a [dsdec]       | Vai a success. carattere            |
| badc: bd 00 02 | lda \$0200,x           | Lettura car. da buffer di input     |
| bad0: f0 05    | beq * \$bad7           | Chiudi canale di comando            |
| bad2: 20 d2 ff | jsr * \$ffd2 [bsout]   | Vai a BSOUT                         |
| bad5: 90 f2    | bcc * \$bac9           | Contr. e uscita pross. caratt.      |
| bad7: 20 cc ff | jsr * \$ffc [clrch]    | Vai a CLRCH                         |
| bada: 20 b4 b8 | jsr * \$b8b4           | Rit. vcarr. e clear su resto linea  |
| badd: a2 00    | ldx #\$00              | Fissa file logico in input          |
| badf: 20 c6 ff | jsr * \$ffc6 [ chkin ] | Vai a routine CHKIN                 |
| bae2: b0 10    | bcs * \$baf4           | Se rilev. errore esci               |
| bae4: 20 cf ff | jsr * \$ffc [ basin ]  | Vai a BASIN                         |
| bae7: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a BSOUT                         |
| baea: c9 0d    | cmp #\$0d              | Controllo uscita CR                 |
| baec: f0 06    | beq * \$baf4           | Se pos. vai a CLRCH e esci          |
| baee: a5 90    | lda \$90 [ status ]    | Carica lo status in A               |
| baf0: 29 bf    | and #\$bf              | Esegui Mask out del bit 6           |

|                |                         |                                       |
|----------------|-------------------------|---------------------------------------|
| baf2: f0 f0    | beq * \$bae4            | Controllo se errore                   |
| baf4: 20 cc ff | jsr * \$ffcc [ clrch ]  | Vai a CLRCH                           |
| baf7: a9 00    | lda #\$00               | Chiudi file logico                    |
| baf9: 38       | sec                     | Fissa il carry per CLOSE              |
| bafa: 20 c3 ff | jsr * \$ffc3 [ close ]  | Vai a CLOSE                           |
| bafd: 4c 8b b0 | jmp * \$b08b            | Vai a ciclo attesa ingresso           |
| bb00: 4c bc b0 | jmp * \$b0bc            | Visualizza ? e ciclo di attesa input  |
| bb03: a0 ff    | ldy #\$ff               | Fissa lunghezza cont.nome file        |
| bb05: a6 7a    | ldx \$7a [ dsdec ]      | Vis. punt.a buffer di input e fissalo |
| bb07: ca       | dex                     | al carattere precedente               |
| bb08: c8       | iny                     | Incrementa contatore nome file        |
| bb09: e8       | inx                     | Puntatore al prossimo carattere       |
| bb0a: bd 00 02 | lda \$0200,x            | Lettura carattere                     |
| bb0d: d0 f9    | bne * \$bb08            | Se non e' un fine comando leggi ancor |
| bb0f: 98       | tya                     | Metti in A lungh. nome file           |
| bb10: a6 7a    | ldx \$7a [ dsdec ]      | Metti in X ind.nome file (LO)         |
| bb12: a0 02    | ldy #\$02               | Metti in Y ind.nome file (HI)         |
| bb14: 20 bd ff | jsr * \$ffbd [ setnam ] | Vai a SETNAM                          |
| bb17: a9 00    | lda #\$00               | Metti in A file logico                |
| bb19: a6 60    | ldx \$60 [ tenexp ]     | N. perif. in X da OPl (LO)            |
| bb1b: a0 60    | ldy #\$60               | Indirizzo secondario in Y             |
| bb1d: 20 ba ff | jsr * \$ffba [ setlfs ] | Vai a SETLFS                          |
| bb20: 20 c0 ff | jsr * \$ffc0 [ open ]   | Vai a OPEN                            |
| bb23: b0 cf    | bcs * \$baf4            | Se errore di CPEN esci                |
| bb25: a2 00    | ldx #\$00               | Fissa per input file logico           |
| bb27: 20 c6 ff | jsr * \$ffc6 [ chkin ]  | Vai a rout. CHKIN                     |
| bb2a: 20 b4 b8 | jsr * \$b8b4            | Ritorno carrello e clear resto        |
| bb2d: a0 03    | ldy #\$03               | Esegui la lettura dei primi 6 bytes   |
| bb2f: 84 63    | sty \$63 [ fac ]        | della directory                       |
| bb31: 20 cf ff | jsr * \$ffcf [ basin ]  | Vai a rout. BASIN                     |
| bb34: 85 60    | sta \$60 [ tenexp ]     | Immag. car. directory in OPl (LO)     |
| bb36: a5 90    | lda \$90 [ status ]     | Metti status in A                     |
| bb38: d0 ba    | bne * \$baf4            | Controllo se errore. Esci             |
| bb3a: 20 cf ff | jsr * \$ffcf [ basin ]  | Vai a rout. BASIN                     |
| bb3d: 85 61    | sta \$61 [ lowtr ]      | Immag. car. directory in OPl (HI)     |
| bb3f: a5 90    | lda \$90 [ status ]     | Poni status in A                      |
| bb41: d0 b1    | bne * \$baf4            | Se errore esci                        |
| bb43: c6 63    | dec \$63 [ fac ]        | Decrementa di 1 bytes directory       |
| bb45: d0 ea    | bne * \$bb31            | Se = a 0 leggi altri bytes            |
| bb47: 20 07 ba | jsr * \$ba07            | Prepara e visualizza contenuti di     |
| bb4a: a9 00    | lda #\$00               | OPl in forma decimale                 |
| bb4c: a2 08    | ldx #\$08               | Visualizza directory sia come ingre.  |
| bb4e: a0 03    | ldy #\$03               | che come n. di caratteri liberi       |
| bb50: 20 5d ba | jsr * \$ba5d            | ***                                   |
| bb53: a9 20    | lda #\$20               | Carica A con SPACE                    |
| bb55: 20 d2 ff | jsr * \$ffd2 [ bsout ]  | Vai a BSOUT                           |
| bb58: 20 cf ff | jsr * \$ffcf [ basin ]  | Vai a BASIN                           |
| bb5b: f0 09    | beq * \$bb66            | Segnale di \$0                        |
| bb5d: a6 90    | ldx \$90 [ status ]     | Metti status in X                     |
| bb5f: d0 93    | bne * \$baf4            | Controllo se errore. Esci             |
| bb61: 20 d2 ff | jsr * \$ffd2 [ bsout ]  | Vai a BSOUT                           |
| bb64: 90 f2    | bcc * \$bb58            | Uscita succ. carattere                |
| bb66: 20 b4 b8 | jsr * \$b8b4            | Esegui un ritorno carrello            |
| bb69: 20 e1 ff | jsr * \$ffe1 [ stop ]   | Vai a STOP                            |
| bb6c: f0 86    | beq * \$baf4            | Se STOP vai a uscita                  |
| bb6e: a0 02    | ldy #\$02               | Leggi contatore per bytes direc       |
| bb70: d0 bd    | bne * \$bb2f            | Salto incondiz. a lettura dir.        |

**NOTE A GRUPPO B**

Al termine della parte relativa al monitor rimangono alcune centinaia di caratteri liberi che al momento attuale non risultano utilizzati.

Pertanto da BB72 a BFFF tutto cio' che appare non e' da prendere in considerazione per un utilizzo pratico.

## TAVOLA DEI SALTI PER LE ROUTINES DELL' EDITOR

```

c000: 4c 7b c0 jmp * $c07b      CINT
c003: 4c 34 cc jmp * $cc34      DISPLAY
c006: 4c 34 c2 jmp * $c234      LP2
c009: 4c 9b c2 jmp * $c29b      LOOP
c00c: 4c 2d c7 jmp * $c72d      PRINT
c00f: 4c 5b cc jmp * $cc5b      SCRORG
c012: 4c 5d c5 jmp * $c55d      KEY
c015: 4c 51 c6 jmp * $c651      REPEAT
c018: 4c 6a cc jmp * $cc6a      PLOT
c01b: 4c 57 cd jmp * $cd57      CURSOR
c01e: 4c c1 c9 jmp * $c9c1      ESCAPE
c021: 4c a2 cc jmp * $cca2      PFKEY
c024: 4c 94 c1 jmp * $c194      IRQ a editor di IRQ
c027: 4c 0c ce jmp * $ce0c      INIT 80
c02a: 4c 2e cd jmp * $cd2e      SWAPPER
c02d: 4c 1b ca jmp * $calb      WINDOW

```

```

c030: ff ff ff      Liberi per future espansioni

```

## LINEE DI PARTENZA (BYTES LO)

```

c033: 00 28 50 78 a0 c8 f0 18
c03b: 40 68 90 b8 e0 08 30 58
c043: 80 a8 d0 f8 20 48 70 98
c04b: c0

```

## LINEE DI PARTENZA (BYTES HI)

```

c04c: 04 04 04 04 04 04 04 05
c054: 05 05 05 05 05 06 06 06
c05c: 06 06 06 06 07 07 07 07
c064: 07

```

## VETTORI DI TASTIERA ED USCITA CARATTERI

```

c065: b9 c7      ($c7b9)      Carattere con CTRL
c067: 05 c8      ($c805)      Carattere con SHIFT
c069: c1 c9      ($c9c1)      Carattere con ESC
c06b: e1 c5      ($c5e1)      Scansione tastiera
c06d: ad c6      ($c6ad)      Immagazzinamento tasto premuto

```

## PUNTATORI ALLA TAVOLA DI DECODIFICA TASTIERA

```

c06f: 80 fa      ($fa80)
c071: d9 fa      ($fad9)
c073: 32 fb      ($fb32)
c075: 8b fb      ($fb8b)
c077: 80 fa      ($fa80)
c079: e4 fb      ($fbe4)

```

ROUTINE CINT: INIZIALIZZA EDITOR E SCHERMO

|       |          |     |            |            |  |
|-------|----------|-----|------------|------------|--|
| c07b: | a9 03    | lda | #\$03      |            | Carica i 2 bits di HI della base       |
| c07d: | 0d 00 dd | ora | * \$dd00   |            | Fissa lo schermo LO                    |
| c080: | 8d 00 dd | sta | * \$dd00   |            | Immagazzina ancora                     |
| c083: | a9 fb    | lda | #\$fb      |            | Esegui un CLEAR del bit 2 del registro |
| c085: | 25 01    | and | \$01       | [ r6510 ]  | di direzione dati, quindi metti a 0 il |
| c087: | 09 02    | ora | #\$02      |            | bit 1 dello stesso registro ed esegui  |
| c089: | 85 01    | sta | \$01       | [ r6510 ]  | ancora il salvataggio                  |
| c08b: | 20 cc ff | jsr | * \$ffcc   | [ clrch ]  | Vai CLRCH:esegui il reset su cana. I/O |
| c08e: | a9 00    | lda | #\$00      |            | Esegui reset di filtro, volume ed ing. |
| c090: | 8d 18 d4 | sta | * \$d418   |            | nella tavola                           |
| c093: | 85 d8    | sta | \$d8       | [ graphm ] | Fissa flag di schermo in modo TESTO    |
| c095: | 85 d7    | sta | \$d7       | [ mode ]   | Fissa flag del modo 40/80 col. in 40   |
| c097: | 85 d0    | sta | \$d0       | [ ndx ]    | Esegui CLEAR sulla coda buffer tast.   |
| c099: | 85 d1    | sta | \$d1       | [ kyndx ]  | Esegui CLEAR su flag tasto funz.       |
| c09b: | 85 d6    | sta | \$d6       | [ crsm ]   | Reset su flag tastiera input           |
| c09d: | 8d 21 0a | sta | \$0a21     |            | Reset su flag PAUSA                    |
| c0a0: | 8d 26 0a | sta | \$0a26     |            | Reset su flag cursore                  |
| c0a3: | 85 d9    | sta | \$d9       | [ charen ] | Fissa punt. di car. in RAM/ROM         |
| c0a5: | 8d 2e 0a | sta | \$0a2e     |            | Fissa ind. base schermo (testo RAM)    |
| c0a8: | a9 14    | lda | #\$14      |            | Iniz. valore per punt base             |
| c0aa: | 8d 2c 0a | sta | \$0a2c     |            | Imm. punt.base testo schermo/car.      |
| c0ad: | a9 78    | lda | #\$78      |            | Iniz. val. base bit-map                |
| c0af: | 8d 2d 0a | sta | \$0a2d     |            | Iniz. base bit-map                     |
| c0b2: | a9 08    | lda | #\$08      |            | Iniz. val RAM                          |
| c0b4: | 8d 2f 0a | sta | \$0a2f     |            | Iniz. base RAM                         |
| c0b7: | ad 4c c0 | lda | * \$c04c   |            | Carica val. di inizial. (\$04)         |
| c0ba: | 8d 3b 0a | sta | \$0a3b     |            | Iniz. punt. sistema PAL                |
| c0bd: | a9 0a    | lda | #\$0a      |            | Car.val.iniz.amp.buffer tastiera       |
| c0bf: | 8d 20 0a | sta | \$0a20     |            | Iniz.flag per contr.buffer tastiera    |
| c0c2: | 8d 28 0a | sta | \$0a28     |            | Imm. puntatore per flash ursore        |
| c0c5: | 8d 27 0a | sta | \$0a27     |            | Flag per modo flash del cursore        |
| c0c8: | 8d 24 0a | sta | \$0a24     |            | Flag per ritardo funz.REPEAT tastiera  |
| c0cb: | a9 04    | lda | #\$04      |            | Valore part. per cont. vel.            |
| c0cd: | 8d 23 0a | sta | \$0a23     |            | Flag: vel. di ripet.                   |
| c0d0: | 20 83 c9 | jsr | * \$c983   |            | Iniz. pos. tabulatore                  |
| c0d3: | 8d 22 0a | sta | \$0a22     |            | Flag per punt. repeat di tastiera      |
| c0d6: | 0d 05 d5 | ora | * \$d505   |            | Fissa bit di controllo ser.(modo FAST  |
| c0d9: | 8d 05 d5 | sta | * \$d505   |            | ) nel MCR del Memory Man. Unit         |
| c0dc: | a9 60    | lda | #\$60      |            | Valore d' inizio per cursore           |
| c0de: | 8d 2b 0a | sta | \$0a2b     |            | Flag per modo cursore                  |
| c0e1: | a9 d0    | lda | #\$d0      |            | Val. di inizializ. per il sitema di    |
| c0e3: | 8d 34 0a | sta | \$0a34     |            | puntatori rel. movim. linea            |
| c0e6: | a2 1a    | ldx | #\$1a      |            | Cont.di ciclo per iniz. Pag 0          |
| c0e8: | bd 74 ce | lda | * \$ce74,x |            | Copia ROM di schermo modo 40 colonne   |
| c0eb: | 95 e0    | sta | \$e0,x     | [ pnt ]    | Copia val. inizio in pag. 0            |
| c0ed: | bd 8e ce | lda | * \$ce8e,x |            | Copia ROM di schermo modo 80 colonne   |
| c0f0: | 9d 40 0a | sta | \$0a40,x   |            | Copia iin RAM i valori iniziali        |
| c0f3: | ca       | dex |            |            | Decrementa di 1 cont. di ciclo         |
| c0f4: | 10 f2    | bpl | * \$c0e8   |            | Ciclo x contrl. tutti valori sia.tra.  |
| c0f6: | a2 09    | ldx | #\$09      |            | Controllo bit 6 flag di inizializz.    |
| c0f8: | bd 65 c0 | lda | * \$c065,x |            | Se bit fissato salta                   |
| c0fb: | 9d 34 03 | sta | \$0334,x   |            | Cont. di ciclo per iniz. pag 3         |
| c0fe: | ca       | dex |            |            | Decrem cont. ciclo di 1                |
| c0ff: | 10 f7    | bpl | * \$c0f8   |            | Cont. ciclo tutti i val.trasferiti     |
| cl01: | 2c 04 0a | bit | \$0a04     |            | Controlla bit 6 del flag inizializ     |
| cl04: | 70 1e    | bvs | *, \$cl24  |            | Se bit fissato vai oltre               |
| cl06: | a2 0b    | ldx | #\$0b      |            | Cont. ciclo iniz. pag.3                |
| cl08: | bd 6f c0 | lda | * \$c06f,x |            | Copia ROM decod. tastiera              |
| cl0b: | 9d 3e 03 | sta | \$033e,x   |            | Immag. vett. area RAM                  |



|                |                |                                     |
|----------------|----------------|-------------------------------------|
| cl0e: ca       | dex            | Decr. di 1 cont. di ciclo           |
| cl0f: 10 f7    | bpl * \$cl08   | Ciclo trasf. tutti i valori         |
| cl11: a2 4c    | ldx #\$4c      | Ciclo contr.iniz. tasti funz.       |
| cl13: bd a8 ce | lda * \$cea8,x | Copia la ROM della lunghezza tasto  |
| cl16: 9d 00 10 | sta \$1000,x   | funzione eimmag. in RAM             |
| cl19: ca       | dex            | Decr. cont.ciclo di 1               |
| cl1a: 10 f7    | bpl * \$cl13   | Ciclo contr. trasf. tutti i valori  |
| cl1c: a9 40    | lda #\$40      | Metti il bit 6 in pos. di ON e con. |
| cl1e: 0d 04 0a | ora \$0a04     | con flag inizializz.                |
| cl21: 8d 04 0a | sta \$0a04     | Metti risul. in flag di inizial.    |
| cl24: 20 2e cd | jsr * \$cd2e   | Vai al modo 40/80 colonne           |
| cl27: 20 83 c9 | jsr * \$c983   | Reset dei tabulatori                |
| cl2a: 20 24 ca | jsr * \$ca24   | Intero schermo come finestra        |
| cl2d: 20 42 c1 | jsr * \$cl42   | CLR/HOME                            |
| cl30: 20 2e cd | jsr * \$cd2e   | Vedi sopra                          |
| cl33: 20 24 ca | jsr * \$ca24   | Vedi sopra                          |
| cl36: 20 42 c1 | jsr * \$cl42   | Vedi sopra                          |
| cl39: 2c 05 d5 | bit * \$d505   | Contr. se modo 40/80 colonne        |
| cl3c: 30 03    | bmi * \$cl41   | Salta se siamo in 80 coolonne       |
| cl3e: 20 2e cd | jsr * \$cd2e   | Vai al modo 40/80 colonne           |
| cl41: 60       | rts            |                                     |

## ROUTINE DI CLEAR WINDOW

|                |              |                                    |
|----------------|--------------|------------------------------------|
| cl42: 20 50 c1 | jsr * \$cl50 | Cursore in posizione HOME          |
| cl45: 20 5e c1 | jsr * \$cl5e | Calcola ind. di inizio linea X     |
| cl48: 20 a5 c4 | jsr * \$ca45 | Esegui unclear su linea X          |
| cl4b: e4 e4    | cpx \$e4     | Confronto bordo inferiore finestra |
| cl4d: e8       | inx          | Incrementa punt. di linea          |
| cl4e: 90 f5    | bcc * \$cl45 | se bordo inf. non rintracciato     |

## ROUTINE DI CURSOR HOME

|                |                |            |                                   |
|----------------|----------------|------------|-----------------------------------|
| cl50: a6 e5    | ldx \$e5       | [ sctop ]  | Imm. bordo sup. in X              |
| cl52: 86 eb    | stx \$eb       | [ tblx ]   | Nota att. linea del cursore       |
| cl54: 86 e8    | stx \$e8       | [ lsxp ]   | Immag. come linea input iniziale  |
| cl56: a4 e6    | ldy \$e6       | [ sclf ]   | Carica in Y bordo sin. finestra   |
| cl58: 84 ec    | sty \$ec       | [ pntr ]   | Memor. colonna attt. cursore      |
| cl5a: 84 e9    | sty \$e9       | [ lstp ]   | come colonna iniziale             |
| cl5c: a6 eb    | ldx \$eb       | [ tblx ]   | Carica in X att. linea cursore    |
| cl5e: bd 33 c0 | lda * \$c033,x |            | Carica in A bit LO linea di part. |
| cl61: 24 d7    | bit \$d7       | [ mode ]   | Controllo modo 40/80 col.         |
| cl63: 10 01    | bpl * \$cl66   |            | Salta se siamo in 40 col.         |
| cl65: 0a       | asl a          |            | Altrimenti esegui un ASL di A     |
| cl66: 85 e0    | sta \$e0       | [ pnt ]    | Imm. byte LO                      |
| cl68: bd 4c c0 | lda * \$c04c,x |            | Metti in A byte HI linea iniziale |
| cl6b: 29 03    | and #\$03      |            | Mask out bits 2-7                 |
| cl6d: 24 d7    | bit \$d7       | [ mode ]   | Controllo modo 40/80 col          |
| cl6f: 10 06    | bpl * \$cl77   |            | Se siamo a 40 col. salta          |
| cl71: 2a       | rol a          |            | Spost a sin carry per byte HI     |
| cl72: 0d 2e 0a | ora \$0a2e     |            | Aggiungo q.s. aind. iniz. video   |
| cl75: 90 03    | bcc * \$cl7a   |            | Esegui un salto a sub. \$cl7a     |
| cl77: 0d 3b 0a | ora \$0a3b     |            | Agg. a ind. video 40 colonne      |
| cl7a: 85 e1    | sta \$e1       | [ pnt+ 1 ] | Immag. in A byte HI               |
| cl7c: a5 e0    | lda \$e0       | [ pnt ]    | Byte LO attuale linea di schermo  |
| cl7e: 85 e2    | sta \$e2       | [ user ]   | messo in A come ind. assegn       |
| cl80: a5 e1    | lda \$e1       | [ pnt+ 1 ] | Immag. in A byte HI               |
| cl82: 24 d7    | bit \$d7       | [ mode ]   | Contr per modo 40/80 col.         |
| cl84: 10 07    | bpl * \$cl8d   |            | E' attivo modo 40 colonne         |
| cl86: 29 07    | and #\$07      |            | Mask-out bits 2-7                 |

|                |                      |                            |
|----------------|----------------------|----------------------------|
| cl88: 0d 2f 0a | ora \$0a2f           | Assegna a RAM base         |
| cl8b: d0 04    | bne * \$cl91         | Se diverso esegui un salto |
| cl8d: 29 03    | and # \$03           | Mask-out bits 2-7          |
| cl8f: 09 d8    | ora # \$d8           | Agg. base colore RAM       |
| cl91: 85 e3    | sta \$e3 [ user+ 1 ] | Imm. valori byte HI        |
| cl93: 60       | rts                  |                            |

## ROUTINE DI IRQ

|                |                     |                                       |
|----------------|---------------------|---------------------------------------|
| cl94: 38       | sec                 | Fissa il flag di carry come punt.     |
| cl95: ad 19 d0 | lda * \$d019        | Carica IRR da VIC                     |
| cl98: 29 01    | and # \$01          | Controlla bit di interrupt RL         |
| cl9a: f0 07    | beq * \$cla3        | Se settaggio non eseguito vai oltre   |
| cl9c: 8d 19 d0 | sta * \$d019        | Esegui un clear del reg.              |
| cl9f: a5 d8    | lda \$d8 [ graphm ] | Controllo x modo testo o graf.        |
| clal: c9 ff    | cmp # \$ff          | Se e' stato abilitato lo schermo graf |
| cla3: f0 6f    | beq * \$c214        | vai alla rout. \$c214                 |
| cla5: 2c 11 d0 | bit * \$d011        | Controlla reg 1 del VIC               |
| cla8: 30 04    | bmi * \$clae        | ****                                  |
| claa: 29 40    | and # \$40          | *****                                 |
| clac: d0 31    | bne * \$cldf        | ****                                  |
| clae: 38       | sec                 | Fissa il flag di carry come punt.     |
| claf: a5 d8    | lda \$d8 [ graphm ] | Controlla modo testo grafica          |
| clb1: f0 2c    | beq * \$cldf        | Salto al modo testo                   |
| clb3: 24 d8    | bit \$d8 [ graphm ] | Ricontrolla                           |
| clb5: 50 06    | bvc * \$clbd        | Se bit 6=0 nessuna linea di raster    |
| clb7: ad 34 0a | lda \$0a34          | IRQ                                   |
| clba: 8d 12 d0 | sta * \$d012        | Refresh                               |
| clbd: a5 01    | lda \$01 [ r6510 ]  | Carica reg. direz. dati               |
| clbf: 29 fd    | and # \$fd          | Mask out bit 0-1                      |
| clcl: 09 04    | ora # \$04          | Metti a 1 bit 2 reg                   |
| clc3: 48       | pha                 | Salva conf su STACK                   |
| clc4: ad 2d 0a | lda \$0a2d          | Indirizzo di base per grafica         |
| clc7: 48       | pha                 | Salva su Stack detto ind.             |
| clc8: ad 11 d0 | lda * \$d011        | Carica reg. di contr. 1 del VIC       |
| clcb: 29 7f    | and # \$7f          | Esegui un clear su raster 1           |
| clcd: 09 20    | ora # \$20          | Fissa modo standard bit map           |
| clcf: a8       | tay                 | Trasf in Y reg. di controllo          |
| cld0: ad 16 d0 | lda * \$d016        | Cariica il reg. di contr. 2 del VIC   |
| cld3: 24 d8    | bit \$d8 [ graphm ] | Controllo modo testo/grafica          |
| cld5: 30 03    | bmi * \$clda        | Fissa modo multicolore                |
| cld7: 29 ef    | and # \$ef          | Metti a 0 bit modo multicolor         |
| cld9: 2c 09 10 | bit \$1009          | Vai a loc. \$ClDC, fissa bit modo mc  |
| cldc: aa       | tax                 | Registro di controllo in X            |
| cldd: d0 28    | bne * \$c207        | Salto incondizionato a \$C207         |

## MOD0 TESTO

|                |                     |                                    |
|----------------|---------------------|------------------------------------|
| cldf: a9 ff    | lda # \$ff          | Linea di raster nell' ultima linea |
| cle1: 8d 12 d0 | sta * \$d012        | Immagazzina come linea di RASTER   |
| cle4: a5 01    | lda \$01 [ r6510 ]  | Carica reg. direz. dati            |
| cle6: 09 02    | ora # \$02          | Fissa il bit 1 del reg.            |
| cle8: 29 fb    | and # \$fb          | Esegui un clear sul bit 2          |
| clea: 05 d9    | ora \$d9 [ charen ] | Il bit 2 e' =0 se CHAREN e'        |
| clec: 48       | pha                 | in RAM                             |
| clcd: ad 2c 0a | lda \$0a2c          | Carica ind. base di TESTO/GRAFICA  |
| clf0: 48       | pha                 | nello STACK                        |
| clf1: ad 11 d0 | lda * \$d011        | Carica reg. controllo VIC          |
| clf4: 29 5f    | and # \$5f          | Esegui un clear del CARRY          |

|                |     |   |        |                                       |
|----------------|-----|---|--------|---------------------------------------|
| clf6: a8       | tay |   |        | Metti in Y il reg. di contr.1         |
| clf7: ad 16 d0 | lda | * | \$d016 | Carica reg. controllo VIC n.2         |
| clfa: 29 ef    | and | # | \$ef   | Esegui un CLEAR sul BIT MULTICOLOR    |
| clfc: aa       | tax |   |        | Trasf. in X reg. di contr. 2          |
| clfd: b0 08    | bcs | * | \$c207 | Se CARRY=1 non eseg. ciclo attesa     |
| clff: a2 07    | ldx | # | \$07   | Considera X cont. ciclo ritardo       |
| c201: ca       | dex |   |        | Decrem. contat di 1                   |
| c202: d0 fd    | bne | * | \$c201 | Salta a pos. se oper. prec.non esegui |
| c204: ea       | nop |   |        | 2 pass.non operativi nel ciclo        |
| c205: ea       | nop |   |        | di attesa                             |
| c206: aa       | tax |   |        | Registro di controllo 2 in X          |

## FISSA I REGISTRI DI IRQ

|                |     |      |            |  |
|----------------|-----|------|------------|--|
| c207: 68       | pla |      |            | Ricarica ind. base                     |
| c208: 8d 18 d0 | sta | *    | \$d018     | Immetti quanto sopra in VIC            |
| c20b: 68       | pla |      |            | Carica dallo Stack registro di         |
| c20c: 85 01    | sta | \$01 | [ r6510 ]  | direzione dati. Esegui seg. salv.      |
| c20e: 8c 11 d0 | sty | *    | \$d011     | Registro 1 nel VIC                     |
| c211: 8e 16 d0 | stx | *    | \$d016     | Registro 2 nel VIC                     |
| c214: b0 13    | bcs | *    | \$c229     | Se Carry a 1 esegui il salto           |
| c216: ad 30 d0 | lda | *    | \$d030     | Carica reg di clock a 1/2 MHz          |
| c219: 29 01    | and | #    | \$01       | *****                                  |
| c21b: f0 0c    | beq | *    | \$c229     | Salta se siamo a 1 MHz                 |
| c21d: a5 d8    | lda | \$d8 | [ graphm ] | Carica modo TESTO/GRAFICA              |
| c21f: 29 40    | and | #    | \$40       | Controlla bit di interrupt della linea |
| c221: f0 06    | beq | *    | \$c229     | di RASTER. Se non e' nessun INTERRUPT  |
| c223: ad 11 d0 | lda | *    | \$d011     | Carica reg. di controllo 1             |
| c226: 10 01    | bpl | *    | \$c229     | Nessuna modifica al carry              |
| c228: 38       | sec |      |            | Fissa il CARRY come FLAG               |
| c229: 58       | cli |      |            | Disabilita tutti gli interrupt         |
| c22a: 90 07    | bcc | *    | \$c233     | Se FLAG non= 0 salta                   |
| c22c: 20 5d c5 | jsr | *    | \$c55d     | Richiama la rout. KEY                  |
| c22f: 20 e7 c6 | jsr | *    | \$c6e7     | Vai alla rout. di FLASH del curs.      |
| c232: 38       | sec |      |            | Rimetti a 1 il CARRY                   |
| c233: 60       | rts |      |            |  |

## CARATTERE DA ROUTINE KEY

|                |     |          |            |                                      |
|----------------|-----|----------|------------|--------------------------------------|
| c234: a6 d1    | ldx | \$d1     | [ kyndx ]  | Controllo se devono rilevarsi carat. |
| c236: f0 0c    | beq | *        | \$c244     | dal buffer di tastiera               |
| c238: a4 d2    | ldy | \$d2     | [ keyidx ] | Carica punt. al buffer KEY           |
| c23a: b9 0a 10 | lda | \$100a,y |            | Prendi un car. da TAVOLA KEY         |
| c23d: c6 d1    | dec | \$d1     | [ kyndx ]  | Decrem. cont. carattere              |
| c23f: e6 d2    | inc | \$d2     | [ keyidx ] | Increment il puntatore               |
| c241: 58       | cli |          |            | Disabilita tutti gli interrupt       |
| c242: 18       | clc |          |            | Esegui un CLEAR di CARRY             |
| c243: 60       | rts |          |            |                                      |

## CARATTERE DA BUFFER

|                |     |          |         |                                |
|----------------|-----|----------|---------|--------------------------------|
| c244: ac 4a 03 | ldy | \$034a   |         | Contr. n. caratteri in coda    |
| c247: bd 4b 03 | lda | \$034b,x |         | Prendi un caratt. da coda      |
| c24a: 9d 4a 03 | sta | \$034a,x |         | Immag. dopo uno spost.         |
| c24d: e8       | inx |          |         | Incrementa il cont.            |
| c24e: a4 d0    | cpx | \$d0     | [ ndx ] | Esegui lo spostamento di tutta |
| c250: d0 f5    | bne | *        | \$c247  | la coda per il car. prelevato  |
| c252: c6 d0    | dec | \$d0     | [ ndx ] | Decrementa la coda di 1        |
| c254: 98       | tya |          |         | Trasf. car. in A               |
| c255: 58       | cli |          |         | Disabil. tutti gli INTERRUPT   |

c256: 18            clc                            Esegui CLEAR di CARRY  
c257: 60            rts

-----  
LINEA DI INPUT DA LOOP4

|                |               |           |  |
|----------------|---------------|-----------|--|
| c258: 20 2d c7 | jsr * \$c72d  |           | Uscita carattere                       |
| c25b: 20 6f cd | jsr * \$cd6f  |           | Spostamento cursore                    |
| c25e: a5 d0    | lda \$d0      | [ ndx ]   | Carica in A n. car. di Buffer tast.    |
| c260: 05 d1    | ora \$d1      | [ kyndx ] | Sommaci n. car.del Buffer KEY          |
| c262: f0 fa    | beq * \$c25e  |           | Se vuoto attendi                       |
| c264: 20 9f cd | jsr * \$cd9f  |           | Fissaggio del cursore                  |
| c267: 20 34 c2 | jsr * \$c234  |           | Rileva car. dal buffer                 |
| c26a: c9 0d    | cmp #\$0d     |           | Controlla se e' un Rit. Carr.          |
| c26c: d0 ea    | bne * \$c258  |           | Se no vai al prossimo car.             |
| c26e: 85 d6    | sta \$d6      | [ crsm ]  | Immag. flag di input                   |
| c270: a9 00    | lda #\$00     |           | Esegui un CLEAR sul FLAG               |
| c272: 85 f4    | sta \$f4      | [ qtsw ]  | Immag. in A val. di QTSW               |
| c274: 20 c3 cb | jsr * \$cbbc3 |           | Calc. fine linea INPUT                 |
| c277: 8e 30 0a | stx \$0a30    |           | Salva in X ult. pos. col.              |
| c27a: 20 b5 cb | jsr * \$cbb5  |           | Fissa linea inizio                     |
| c27d: a4 e6    | ldy \$e6      | [ sclf ]  | Carica in Y bordo sin. fin. schermo    |
| c27f: a5 e8    | lda \$e8      | [ lsxp ]  | Carica in A inizio linea INPUT         |
| c281: 30 13    | bmi * \$c296  |           | Indirizzo linea inizio                 |
| c283: c5 eb    | cmp \$eb      | [ tblx ]  | Confr. con attuale linea curs.         |
| c285: 90 0f    | bcc * \$c296  |           | Il bordo non e' stato ragg.            |
| c287: a4 e9    | ldy \$e9      | [ lstp ]  | Inizio colonna input                   |
| c289: cd 30 0a | cmp \$0a30    |           | Confr. con ultima col in INPUT         |
| c28c: d0 04    | bne * \$c292  |           | Se non e' la stessa colonna            |
| c28e: c4 ea    | cpy \$ea      | [ indx ]  | esegui un confronto con la linea       |
| c290: f0 02    | beq * \$c294  |           | ricercata in prec.                     |
| c292: b0 11    | bcs * \$c2a5  |           | Fissa il FLAG di INPUT/GET             |
| c294: 85 eb    | sta \$eb      | [ tblx ]  | Scrivi l'attuale linea cursore         |
| c296: 84 ec    | sty \$ec      | [ pntr ]  | Immag. attuale col. cursore            |
| c298: 4c bc c2 | jmp * \$c2bc  |           | Prendi il caratt. dove e' ora il curs. |
| c29b: 98       | tya           |           | Trasf. in Y pos. col.                  |
| c29c: 48       | pha           |           | Salva sullo STACK                      |
| c29d: 8a       | txa           |           | Trasf. in X pos. linea                 |
| c29e: 48       | pha           |           | Salva sullo STACK                      |
| c29f: a5 d6    | lda \$d6      | [ crsm ]  | Carica FLAG di INPUT/GET               |
| c2a1: f0 b8    | beq * \$c25b  |           | Ciclo di rit. per GET                  |
| c2a3: 10 17    | bpl * \$c2bc  |           | Nessun Ritorno carrello                |
| c2a5: a9 00    | lda #\$00     |           | Metti a 0 il flag di INPUT/GET         |
| c2a7: 85 d6    | sta \$d6      | [ crsm ]  | tramite A                              |
| c2a9: a9 0d    | lda #\$0d     |           | Codice ASCII per Rit. Car.             |
| c2ab: a2 03    | ldx \$03      |           | Confronta codice schermo per vedere    |
| c2ad: e4 99    | cpx \$99      | [ dfltn ] | se come perf. e' lo schermo            |
| c2af: f0 04    | beq * \$c2b5  |           | Se la periferica e' lo schermo         |
| c2b1: e4 9a    | cpx \$9a      | [ dflto ] | confronta con STANDARD di OUTPUT       |
| c2b3: f0 03    | beq * \$c2b8  |           | Uscita sullo schermo                   |
| c2b5: 20 2d c7 | jsr * \$c72d  |           | Ingresso per BSOUT                     |
| c2b8: a9 0d    | lda #\$0d     |           | Codice ASCII per Rit. Carr.            |
| c2ba: d0 39    | bne * \$c2f5  |           | Salto incond. alla fine                |
| c2bc: 20 5c c1 | jsr * \$c15c  |           | Salta all' ind. linea att.             |
| c2bf: 20 58 cb | jsr * \$cb58  |           | Vai a car. e colore pos. att. curs.    |
| c2c2: 85 ef    | sta \$ef      | [ datax ] | Immag. temp. per stampa caratt.        |
| c2c4: 29 3f    | and \$3f      |           | Mask dei bit 6 e 7                     |
| c2c6: 06 ef    | asl \$ef      | [ datax ] | Conver. caratt. in                     |
| c2c8: 24 ef    | bit \$ef      | [ datax ] | ASCII                                  |
| c2ca: 10 02    | bpl * \$c2ce  |           | Contr. non e' caratt. REVERSE          |
| c2cc: 09 80    | ora #\$80     |           | Fissa il bit 7                         |

|                |              |           |  |
|----------------|--------------|-----------|--|
| c2ce: 90 04    | bcc * \$c2d4 |           | Controllo bit 7                        |
| c2d0: a6 f4    | ldx \$f4     | [ qtsw ]  | Attiva il FLAG apici                   |
| c2d2: d0 04    | bne * \$c2d8 |           | Se attivo salta a ind.                 |
| c2d4: 70 02    | bvs * \$c2d8 |           | Controllo bit 6                        |
| c2d6: 09 40    | ora #\$40    |           | Fissa il bit 6 per controllare se      |
| c2d8: 20 ff c2 | jsr * \$c2ff |           | e' car. apici. Quindi fissa il FLAG    |
| c2db: a4 eb    | ldy \$eb     | [ tblx ]  | Carica in Y att. pos. cursore          |
| c2dd: cc 30 0a | cpy \$0a30   |           | Controllo pos. ident. ult. colonna     |
| c2e0: 90 0a    | bcc * \$c2ec |           | Ultima colonna non raggiunta           |
| c2e2: a4 ec    | ldy \$ec     | [ pntr ]  | Carica in Y puntatore colonna          |
| c2e4: c4 ea    | cpy \$ea     | [ indx ]  | Confronta con la fine                  |
| c2e6: 90 04    | bcc * \$c2ec |           | La fine linea non raggiunta            |
| c2e8: 66 d6    | ror \$d6     | [ crsm ]  | Rotazione CARRY in BIT 7 di \$D6       |
| c2ea: 30 03    | bmi * \$c2ef |           | Controllo e quindi a nuova linea       |
| c2ec: 20 ed cb | jsr * \$cbcd |           | Sposta il cursore di una pos. a destra |
| c2ef: c9 de    | cmp #\$de    |           | Confronta con ASCII PI                 |
| c2f1: d0 02    | bne * \$c2f5 |           | se non e' PI                           |
| c2f3: a9 ff    | lda #\$ff    |           | Carica in A il codice giusto           |
| c2f5: 85 ef    | sta \$ef     | [ datax ] | Carica in A car. di stampa             |
| c2f7: 68       | pla          |           | Trasferisci in X il valore dello       |
| c2f8: aa       | tax          |           | STACK tramite A (linea)                |
| c2f9: 68       | pla          |           | Trasferisci in Y il valore dello       |
| c2fa: a8       | tay          |           | STACK tramite A (colonna)              |
| c2fb: a5 ef    | lda \$ef     | [ datax ] | Stampa il caratt. da immag. temp.      |
| c2fd: 18       | clc          |           | Esegui un clear del CARRY              |
| c2fe: 60       | rts          |           |  |

-----  
CONTROLLO PER APICI

|             |              |          |                                 |
|-------------|--------------|----------|---------------------------------|
| c2ff: c9 22 | cmp #\$22    |          | Confronta con car. apici        |
| c301: d0 08 | bne * \$c30b |          | Se altro carattere termina      |
| c303: a5 f4 | lda \$f4     | [ qtsw ] | Carica attuale modo apici       |
| c305: 49 01 | eor #\$01    |          | Modo reverse                    |
| c307: 85 f4 | sta \$f4     | [ qtsw ] | Immagazzina quanto sopra.       |
| c309: a9 22 | lda #\$22    |          | Carica A con val ASCII di apici |
| c30b: 60    | rts          |          |                                 |

-----  
IMPIEGO DI BSOUT

|                |              |            |                                    |
|----------------|--------------|------------|------------------------------------|
| c30c: a5 ef    | lda \$ef     | [ datax ]  | Salva l' attuale car. stampato     |
| c30e: 85 f0    | sta \$f0     | [ lstchr ] | come ultimo carattere              |
| c310: 20 57 cd | jsr * \$cd57 |            | Fissa il cursore all' att. colonna |
| c313: a5 f5    | lda \$f5     | [ insrt ]  | Carica A con modo inserimento      |
| c315: f0 02    | beq * \$c319 |            | Il modo prec. non e' attivato      |
| c317: 46 f4    | lsr \$f4     | [ qtsw ]   | Sposta a sin.FLAG modo apici       |
| c319: 68       | pla          |            | Prendi il primo valore dallo STACK |
| c31a: a8       | tay          |            | Trasferisci q.s. in Y              |
| c31b: 68       | pla          |            | Prendi il secondo valore da STACK  |
| c31c: aa       | tax          |            | Immettilo in X                     |
| c31d: 68       | pla          |            | Carica A con val. da STACK         |
| c31e: 18       | clc          |            | Clear CARRY                        |
| c31f: 60       | rts          |            |                                    |

-----  
CONVERSIONE DA ASCII A CODICE POKE

|             |              |           |                                |
|-------------|--------------|-----------|--------------------------------|
| c320: 09 40 | ora #\$40    |           | Fissa bit 2 di A               |
| c322: a6 f3 | ldx \$f3     | [ rvs ]   | Attiva FLAG del modo RVS       |
| c324: f0 02 | beq * \$c328 |           | Nessun caratt. RVS             |
| c326: 09 80 | ora #\$80    |           | Fissa bit di ordine HI per RVS |
| c328: a6 f5 | ldx \$f5     | [ insrt ] | Carica flag di modo INSERT     |

|       |          |          |              |  |                                     |
|-------|----------|----------|--------------|--|-------------------------------------|
| c32a: | f0 02    | beq *    | \$c32e       |  | Nessun modo di insert               |
| c32c: | c6 f5    | dec \$f5 | [ insrt ]    |  | Decrementa contatore                |
| c32e: | 24 f6    | bit \$f6 | [ insflg ]   |  | Controllo per FLAG di auto insert   |
| c330: | 10 09    | bpl *    | \$c33b       |  | Salta se disattivato                |
| c332: | 48       | pha      |              |  | Salva A in STACK                    |
| c333: | 20 e3 c8 | jsr *    | \$c8e3       |  | Vai al modo SOTTO IL CURSORE        |
| c336: | a2 00    | ldx #    | \$00         |  | Fissa il FLAG modo INSERT           |
| c338: | 86 f5    | stx \$f5 | [ insrt ]    |  | a 0                                 |
| c33a: | 68       | pla      |              |  | Ricarica A con valore in STACK      |
| c33b: | 20 2f cc | jsr *    | \$cc2f       |  | Uscita car. all' att. pos curs.     |
| c33e: | c4 e7    | cpy \$e7 | [ scrt ]     |  | Confr.per bordo destro finestra     |
| c340: | 90 0a    | bcc *    | \$c34c       |  | Bordo di cui sopra non trovato      |
| c342: | a6 eb    | ldx \$eb | [ tblx ]     |  | Imm. att. pos. curs in X            |
| c344: | e4 e4    | cpx \$e4 | [ scbot ]    |  | Confr. per bordo in basso finestra  |
| c346: | 90 04    | bcc *    | \$c34c       |  | Bordo di cui s. non trovato         |
| c348: | 24 f8    | bit \$f8 | [ locks+ 1 ] |  | Controllo FLAG di SCROLL            |
| c34a: | 30 16    | bmi *    | \$c362       |  | Nessun scrolling, vai a fine        |
| c34c: | 20 5c c1 | jsr *    | \$c15c       |  | Determ.ind. iniz.linea attuale      |
| c34f: | 20 ed cb | jsr *    | \$cbcd       |  | Sposta di 1 il cursore verso destra |
| c352: | 90 0e    | bcc *    | \$c362       |  | Controllo per nessuna linea nuova   |
| c354: | 20 74 cb | jsr *    | \$cb74       |  | Controllo per Bit di overflow       |
| c357: | b0 08    | bcs *    | \$c361       |  | Il bit di Overflow e' fissato       |
| c359: | 38       | sec      |              |  | Fissa bit di CARRY per non scroll   |
| c35a: | 24 f8    | bit \$f8 | [ locks+ 1 ] |  | Controllo per bit di scroll         |
| c35c: | 70 04    | bvs *    | \$c362       |  | Salta se non scroll                 |
| c35e: | 20 7c c3 | jsr *    | \$c37c       |  | Immetti una linea a X               |
| c361: | 18       | clc      |              |  | Esegui CLEAR di carry               |
| c362: | 60       | rts      |              |  |                                     |

-----  
POSIZIONAMENTO PER LINEFEED

|       |          |          |              |  |                                       |
|-------|----------|----------|--------------|--|---------------------------------------|
| c363: | a6 eb    | ldx \$eb | [ tblx ]     |  | Immetti att. linea curs. in X         |
| c365: | e4 e4    | cpx \$e4 | [ scbot ]    |  | Confronta per bordo basso schermo     |
| c367: | 90 0e    | bcc *    | \$c377       |  | Non trovato il bordo basso schermo    |
| c369: | 24 f8    | bit \$f8 | [ locks+ 1 ] |  | Controllo per bit di scroll           |
| c36b: | 10 06    | bpl *    | \$c373       |  | Scroll abilitato                      |
| c36d: | a5 e5    | lda \$e5 | [ sctop ]    |  | Carica A con bordo sup. schermo       |
| c36f: | 85 eb    | sta \$eb | [ tblx ]     |  | Scrivi att. pos. linea curs.          |
| c371: | b0 06    | bcs *    | \$c379       |  | Salto incond. a indir. indicato       |
| c373: | 20 a6 c3 | jsr *    | \$c3a6       |  | Esecuzione di scroll                  |
| c376: | 18       | clc      |              |  | Clear di carry                        |
| c377: | e6 eb    | inc \$eb | [ tblx ]     |  | Incr. att. linea curs. di 1           |
| c379: | 4c 5c c1 | jmp *    | \$c15c       |  | Determina ind.di part. linea attuale  |
| c37c: | a6 e8    | ldx \$e8 | [ lsxp ]     |  | Inizio linea di input                 |
| c37e: | 30 06    | bmi *    | \$c386       |  | La linea oggetto e' la seguente       |
| c380: | e4 eb    | cpx \$eb | [ tblx ]     |  | Confronto con att.pos. curs. su linea |
| c382: | 90 02    | bcc *    | \$c386       |  | Controllo di ricerca cursore          |
| c384: | e6 e8    | inc \$e8 | [ lsxp ]     |  | Incr.inizio linea input               |
| c386: | a6 e4    | ldx \$e4 | [ scbot ]    |  | Carica X con val.bordo basso schermo  |
| c388: | 20 5e c1 | jsr *    | \$c15e       |  | Fissa ind. attuale linea schermo      |
| c38b: | a4 e6    | ldy \$e6 | [ sclf ]     |  | Carica in Y bordo sinistro schermo    |
| c38d: | e4 eb    | cpx \$eb | [ tblx ]     |  | Confronto con attuale linea           |
| c38f: | f0 0f    | beq *    | \$c3a0       |  | Linea cursore e' bordo basso          |
| c391: | ca       | dex      |              |  | Decrementa linea di 1                 |
| c392: | 20 76 cb | jsr *    | \$cb76       |  | Controllo bit di overflow             |
| c395: | e8       | inx      |              |  | Ritorno alla linea attuale            |
| c396: | 20 83 cb | jsr *    | \$cb83       |  | Clear bit di Overflow                 |
| c399: | ca       | dex      |              |  | Ritorno alla linea precedente         |
| c39a: | 20 0d c4 | jsr *    | \$c40d       |  | Copia una linea di schermo            |
| c39d: | 4c 88 c3 | jmp *    | \$c388       |  | Ritorna al ciclo                      |

|       |          |                       |                                       |
|-------|----------|-----------------------|---------------------------------------|
| c3a0: | 20 a5 c4 | jsr * \$c4a5          | Esegui un clear sulla linea           |
| c3a3: | 4c 93 cb | jmp * \$cb93          | Fissa il bit di Carry                 |
| c3a6: | a6 e5    | ldx \$e5 [ sctop ]    | Carica in X bordo sup. schermo        |
| c3a8: | e8       | inx                   | Incrementa linea di 1                 |
| c3a9: | 20 76 cb | jsr * \$cb76          | Controllo bit di overflow             |
| c3ac: | 90 0a    | bcc * \$cb8           | Nessun overflow come ris. di contr.   |
| c3ae: | e4 e4    | cpx \$e4 [ scbot ]    | Confronta il bordo basso schermo      |
| c3b0: | 90 f6    | bcc * \$c3a8          | Bordo non raggiunto                   |
| c3b2: | a6 e5    | ldx \$e5 [ sctop ]    | Carica su X bordo superiore           |
| c3b4: | e8       | inx                   | Incrementalo di 1                     |
| c3b5: | 20 85 cb | jsr * \$cb85          | Fissa il bit di overflow              |
| c3b8: | c6 eb    | dec \$eb [ tblx ]     | Decr. di 1 attuale linea curs.        |
| c3ba: | 24 e8    | bit \$e8 [ lsexp ]    | Controlla bit 7 della linea di INPUT  |
| c3bc: | 30 02    | bmi * \$c3c0          | Salta se e' a 1                       |
| c3be: | c6 e8    | dec \$e8 [ lsexp ]    | Decrementa di 1 linea di INPUT        |
| c3c0: | a6 e5    | ldx \$e5 [ sctop ]    | Carica su X bordo alto schermo        |
| c3c2: | e4 df    | cpx \$df [ keytmp ]   | Confronta con linea cursore           |
| c3c4: | b0 02    | bcs * \$c3c8          | Se >= a bordo sup. salta a ind. indic |
| c3c6: | c6 df    | dec \$df [ keytmp ]   | Decrementa linea cursore di 1         |
| c3c8: | 20 dc c3 | jsr * \$c3dc          | Sposta resto dello schermo            |
| c3cb: | a6 e5    | ldx \$e5 [ sctop ]    | Carica in X bordo sup.                |
| c3cd: | 20 76 cb | jsr * \$cb76          | Controllo bit di overflow             |
| c3d0: | 08       | php                   | Salva i FLAGS sullo STACK             |
| c3d1: | 20 85 cb | jsr * \$cb85          | Esegui un CLEAR su bit di overflow    |
| c3d4: | 28       | plp                   | Riporta indietro i FLAGS              |
| c3d5: | 90 04    | bcc * \$c3db          | Se carry cleared = fine               |
| c3d7: | 24 f8    | bit \$f8 [ locks+ 1 ] | Controlla FLAG di scroll              |
| c3d9: | 30 cb    | bmi * \$c3a6          | Fissa il bit 7 e poi esegui SCROLL    |
| c3db: | 60       | rts                   |                                       |

-----  
ESEGUI UN CLEAR DELA LINEA X

|       |          |                    |                                    |
|-------|----------|--------------------|------------------------------------|
| c3dc: | 20 5e c1 | jsr * \$cl5e       | Clear su linea determ da X         |
| c3df: | a4 e6    | ldy \$e6 [ sclf ]  | Carica su Y bordo sinistro         |
| c3e1: | e4 e4    | cpx \$e4 [ scbot ] | Confronta val. bordo basso         |
| c3e3: | b0 0f    | bcs * \$c3f4       | Valore del bordo incontrato        |
| c3e5: | e8       | inx                | Il puntatore posiz. linea seguente |
| c3e6: | 20 76 cb | jsr * \$cb76       | Controllo bit di overflow          |
| c3e9: | ca       | dex                | Puntatore ripos. linea attuale     |
| c3ea: | 20 83 cb | jsr * \$cb83       | Clear bit di Overflow              |
| c3ed: | e8       | inx                | Puntatore ancora linea seguente    |
| c3ee: | 20 0d c4 | jsr * \$c40d       | Copia linea                        |
| c3f1: | 4c dc c3 | jmp * \$c3dc       | Copia linea succ.                  |
| c3f4: | 20 a5 c4 | jsr * \$c4a5       | Clear linea X                      |
| c3f7: | a9 7f    | lda #\$7f          | Flag per modo diretto              |
| c3f9: | 8d 00 dc | sta * \$dc00       | Lettura di tastiera                |
| c3fc: | ad 01 dc | lda * \$dc01       | Carica matrice di tastiera         |
| c3ff: | c9 df    | cmp #\$df          | Contr. per tasto CBM premuto       |
| c401: | d0 09    | bne * \$c40c       | Controllo negativo. Fine           |
| c403: | a0 00    | ldy #\$00          | Tasto CBM premuto                  |
| c405: | ea       | nop                | Nessuna operazione                 |
| c406: | ca       | dex                | Decremento X per ciclo ritardo     |
| c407: | d0 fc    | bne * \$c405       | Uscita ciclo                       |
| c409: | 88       | dey                | Ciclo da 0 a 65536 e dopo viene    |
| c40a: | d0 f9    | bne * \$c405       | provocato un arresto               |
| c40c: | 60       | rts                |                                    |

-----  
ROUTINE MOVLIN

|       |       |                   |                              |
|-------|-------|-------------------|------------------------------|
| c40d: | 24 d7 | bit \$d7 [ mode ] | Controllo modo 40/80 colonne |
|-------|-------|-------------------|------------------------------|

|                |                |            |                                       |
|----------------|----------------|------------|---------------------------------------|
| c40f: 30 25    | bmi * \$c436   |            | Salta se 80 colonne                   |
| c411: bd 33 c0 | lda * \$c033,x |            | Carica Byte basso linea attuale       |
| c414: 85 dc    | sta \$dc       | [ keynum ] | Immag. byte basso in \$DC             |
| c416: 85 da    | sta \$da       | [ keysiz ] | ed in \$DA                            |
| c418: bd 4c c0 | lda * \$c04c,x |            | Carica byte alto di ind. attuale      |
| c41b: 29 03    | and #\$03      |            | Mask out dei bit 2-7                  |
| c41d: 0d 3b 0a | ora \$0a3b     |            | Esegui un OR con ind. base video      |
| c420: 85 db    | sta \$db       | [ keylen ] | Salva lungh.tasto in A                |
| c422: 29 03    | and #\$03      |            | Esegui un AND fra bit 1 e 2 con ind.  |
| c424: 09 d8    | ora #\$d8      |            | base della RAM colore                 |
| c426: 85 dd    | sta \$dd       | [ keynxt ] | Immag. come byte alto                 |
| c428: b1 da    | lda (\$da),y   | [ keysiz ] | Carica carat. sorgente e salvalo come |
| c42a: 91 e0    | sta (\$e0),y   | [ pnt ]    | indirizzo di arrivo                   |
| c42c: b1 dc    | lda (\$dc),y   | [ keynum ] | Carica colore sorgente e salvalo come |
| c42e: 91 e2    | sta (\$e2),y   | [ user ]   | sorgente                              |
| c430: c4 e7    | cpy \$e7       | [ scrt ]   | Confronta val. bordo destro schermo   |
| c432: c8       | iny            |            | Increment. di 1 punt. colonna         |
| c433: 90 f3    | bcc * \$c428   |            | Salta a indir. spec. se non e' fine   |
| c435: 60       | rts            |            |                                       |

-----  
COPIA DI LINEA IN 80 COLONNE

|                |                |            |                                      |
|----------------|----------------|------------|--------------------------------------|
| c436: 8e 31 0a | stx \$0a31     |            | Immag. tempor. n. di linea           |
| c439: 8c 32 0a | sty \$0a32     |            | Idem per colonna                     |
| c43c: a2 18    | ldx #\$18      |            | Reg. 24 contiene bit di COPY         |
| c43e: 20 da cd | jsr * \$cd da  |            | Salta al valore di registro          |
| c441: 09 80    | ora #\$80      |            | Fissa il bit di COPY                 |
| c443: 20 cc cd | jsr * \$cdcc   |            | Immagaz. registro in VDC             |
| c446: 20 e6 cd | jsr * \$cde6   |            | Fissa ind. agg. ad attuale posiz.    |
| c449: ae 31 0a | ldx \$0a31     |            | Carica la linea da copiare           |
| c44c: bd 33 c0 | lda * \$c033,x |            | Byte basso della linea da copiare    |
| c44f: 0a       | asl a          |            | Ciclo di 2 tempi                     |
| c450: 85 da    | sta \$da       | [ keysiz ] | Immagaz. byte basso                  |
| c452: bd 4c c0 | lda * \$c04c,x |            | Carica byte alto linea da copiare    |
| c455: 29 03    | and #\$03      |            | Mask out dei bit 3-7                 |
| c457: 2a       | rol a          |            | Rotazione valore di Carry            |
| c458: 0d 2e 0a | ora \$0a2e     |            | Aggiungi base video RAM              |
| c45b: 85 db    | sta \$db       | [ keylen ] | Salva come byte alto                 |
| c45d: a2 20    | ldx #\$20      |            | Indirizzo alto blocco di part.       |
| c45f: 18       | clc            |            | Esegui un clear CARRY per somma      |
| c460: 98       | tya            |            | Immetti n.colonna in A               |
| c461: 65 da    | adc \$da       | [ keysiz ] | Sommacci il Byte basso               |
| c463: 85 da    | sta \$da       | [ keysiz ] | Ind di part+ colonna nel Byte basso  |
| c465: a9 00    | lda #\$00      |            | Carica 0 in A                        |
| c467: 65 db    | adc \$db       | [ keylen ] | Somma al byte alto                   |
| c469: 85 db    | sta \$db       | [ keylen ] | Salva risultato come nuovo byte HI   |
| c46b: 20 cc cd | jsr * \$cdcc   |            | Salva come indirizzo blocco partenza |
| c46e: e8       | inx            |            | Puntatore a ind. basso blocco part.  |
| c46f: a5 da    | lda \$da       | [ keysiz ] | Carica Byte basso ind. di destino    |
| c471: 20 cc cd | jsr * \$cdcc   |            | Vai al VDC                           |
| c474: 38       | sec            |            | Fissa il Carry per sottrazione       |
| c475: a6 e7    | ldx \$e7       | [ scrt ]   | Carica bordo sin. schermo in X       |
| c477: e8       | inx            |            | Incrementalo di 1                    |
| c478: 8a       | txa            |            | Trasferisci in A                     |
| c479: ed 32 0a | sbc \$0a32     |            | Sottrai attuale colonna              |
| c47c: 8d 32 0a | sta \$0a32     |            | Slavala come valore                  |
| c47f: a2 1e    | ldx #\$1e      |            | Registro VDC per contat. parole      |
| c481: 20 cc cd | jsr * \$cdcc   |            | Inizio copia                         |
| c484: a2 20    | ldx #\$20      |            | Indir. alto blocco part.             |
| c486: a5 db    | lda \$db       | [ keylen ] | Carica byte alto di ind. sorgente    |



|       |          |     |          |            |                                       |
|-------|----------|-----|----------|------------|---------------------------------------|
| c488: | 29 07    | and | #\$07    |            | Mascheramento bits 3-7                |
| c48a: | 0d 2f 0a | ora | \$0a2f   |            | Aggiungi a RAM                        |
| c48d: | 20 cc cd | jsr | * \$cdcc |            | Fissa i puntatori dei reg.            |
| c490: | e8       | inx |          |            | all' indirizzo basso blocchi partenza |
| c491: | a5 da    | lda | \$da     | [ keysiz ] | Carica indirizzo sorgente             |
| c493: | 20 cc cd | jsr | * \$cdcc |            | Fissa indirizzo sorgente              |
| c496: | 20 f9 cd | jsr | * \$cdf9 |            | Esegui aggior. indirizzo              |
| c499: | ad 32 0a | lda | \$0a32   |            | Carica n. caratt. da copiare          |
| c49c: | a2 le    | ldx | #\$1e    |            | Reg 31 come registro contat. parole   |
| c49e: | 20 cc cd | jsr | * \$cdcc |            | Esegui la copia                       |
| c4a1: | ae 31 0a | ldx | \$0a31   |            | Ricarica attuale linea                |
| c4a4: | 60       | rts |          |            |                                       |

-----  
ESEGUI IL CLEAR DI LINEA (40 COLONNE)

|       |          |     |          |           |                                     |
|-------|----------|-----|----------|-----------|-------------------------------------|
| c4a5: | a4 e6    | ldy | \$e6     | [ sclf ]  | Carica in Y bordo sinistro          |
| c4a7: | 20 85 cb | jsr | * \$cb85 |           | Esegui un Clear sul bit di overflow |
| c4aa: | 20 5e c1 | jsr | * \$cl5e |           | Indirizzo di part. linea X          |
| c4ad: | 24 d7    | bit | \$d7     | [ mode ]  | Controllo modo 40/80 colonne        |
| c4af: | 30 0f    | bmi | * \$c4c0 |           | Salta se 80 colonne                 |
| c4b1: | 88       | dey |          |           | Falso decremento                    |
| c4b2: | c8       | iny |          |           | Incrementa di 1 punt. colonna       |
| c4b3: | a9 20    | lda | #\$20    |           | Carica A con SPAZIO                 |
| c4b5: | 91 e0    | sta | (\$e0),y | [ pnt ]   | Immag. SPAZIO in RAM del video      |
| c4b7: | a5 f1    | lda | \$f1     | [ color ] | Carica A con codice col. carattere  |
| c4b9: | 91 e2    | sta | (\$e2),y | [ user ]  | Immag. il colore nella RAM colore   |
| c4bb: | c4 e7    | cpy | \$e7     | [ scrt ]  | Confronto per bordo destro          |
| c4bd: | d0 f3    | bne | * \$c4b2 |           | Salta se non uguale                 |
| c4bf: | 60       | rts |          |           |                                     |

-----  
ESEGUI IL CLEAR DI LINEA (80 COLONNE)

|       |          |     |          |            |                                       |
|-------|----------|-----|----------|------------|---------------------------------------|
| c4c0: | 8e 31 0a | stx | \$0a31   |            | Salva X                               |
| c4c3: | 8c 32 0a | sty | \$0a32   |            | Salva Y                               |
| c4c6: | a2 18    | ldx | #\$18    |            | Seleziona reg. 24                     |
| c4c8: | 20 da cd | jsr | * \$cdca |            | Prendi il valore attuale linea        |
| c4cb: | 29 7f    | and | #\$7f    |            | Esegui un clear sul bit di COPY       |
| c4cd: | 20 cc cd | jsr | * \$cdcc |            | Salva il nuovo valore                 |
| c4d0: | a2 12    | ldx | #\$12    |            | Aggiorna indirizzo alto               |
| c4d2: | 18       | clc |          |            | Clear di carry per somma              |
| c4d3: | 98       | tya |          |            | Colonna in A                          |
| c4d4: | 65 e0    | adc | \$e0     | [ pnt ]    | Aggiungi ind. basso di inizio         |
| c4d6: | 48       | pha |          |            | Immag. ind. basso sullo STACK         |
| c4d7: | 8d 3c 0a | sta | \$0a3c   |            | Immagazzina byte basso                |
| c4da: | a9 00    | lda | #\$00    |            | Carica A con 0                        |
| c4dc: | 65 e1    | adc | \$e1     | [ pnt+ 1 ] | Aggiungi il carry al Byte alto        |
| c4de: | 8d 3d 0a | sta | \$0a3d   |            | Immag. in A byte alto                 |
| c4e1: | 20 cc cd | jsr | * \$cdcc |            | Metti q.s. nel registro di indirizzo  |
| c4e4: | e8       | inx |          |            | Aggiorna indirizzo basso              |
| c4e5: | 68       | pla |          |            | Carica byte basso da STACK            |
| c4e6: | 20 cc cd | jsr | * \$cdcc |            | Immetti Byte basso in VDC             |
| c4e9: | a9 20    | lda | #\$20    |            | Carica A con SPAZIO                   |
| c4eb: | 20 ca cd | jsr | * \$cdca |            | Immettilo nel registro dati VDC       |
| c4ee: | 38       | sec |          |            | Fissa il carry per sottrazione        |
| c4ef: | a5 e7    | lda | \$e7     | [ scrt ]   | Carica in A bordo destro schermo      |
| c4f1: | ed 32 0a | sbc | \$0a32   |            | Sottrai colonna di inizio             |
| c4f4: | 48       | pha |          |            | Salva il numero sullo Stack           |
| c4f5: | f0 14    | beq | * \$c50b |            | La col. di partenza = al bordo destro |
| c4f7: | aa       | tax |          |            | Trasfer. in X il valore               |
| c4f8: | 38       | sec |          |            | Fissa il Carry per la somma           |

|                |              |             |                                       |
|----------------|--------------|-------------|---------------------------------------|
| c4f9: 6d 3c 0a | adc \$0a3c   |             | Somma Byte basso                      |
| c4fc: 8d 3c 0a | sta \$0a3c   |             | Salva il risultato su A               |
| c4ff: a9 00    | lda #\$00    |             | Metti 0 in A                          |
| c501: 6d 3d 0a | adc \$0a3d   |             | Somma il carry al Byte alto           |
| c504: 8d 3d 0a | sta \$0a3d   |             | Salva il byte alto                    |
| c507: 8a       | txa          |             | Trasf il n. caratteri in A            |
| c508: 20 3e c5 | jsr * \$c53e |             | A in reg. cont. parole                |
| c50b: a2 12    | ldx #\$12    |             | Aggiorna indirizzo alto               |
| c50d: 18       | clc          |             | Clear carry per somma                 |
| c50e: 98       | tya          |             | Colonna in A                          |
| c50f: 65 e2    | adc \$e2     | [ user ]    | Somma valore Byte basso               |
| c511: 48       | pha          |             | Salva byte basso su Stack             |
| c512: a9 00    | lda #\$00    |             | Carica A con 0                        |
| c514: 65 e3    | adc \$e3     | [ user+ 1 ] | Somma il Carry                        |
| c516: 20 cc cd | jsr * \$cdcc |             | Scrivi Byte alto in reg.              |
| c519: e8       | inx          |             | Aggiorna ind. alto                    |
| c51a: 68       | pla          |             | Prendi byte alto da Stack             |
| c51b: 20 cc cd | jsr * \$cdcc |             | Scrivilo nel registro                 |
| c51e: ad 3d 0a | lda \$0a3d   |             | Carica byte alto indirizzo di destino |
| c521: 29 07    | and #\$07    |             | Mask out bits 4-7                     |
| c523: 0d 2f 0a | ora \$0a2f   |             | Esegui un OR con ind. destino         |
| c526: 8d 3d 0a | sta \$0a3d   |             | Salva il risultato                    |
| c529: a5 f1    | lda \$f1     | [ color ]   | Cod.colore caratt. in A               |
| c52b: 29 8f    | and #\$8f    |             | Colore e bit di ALT                   |
| c52d: 20 ca cd | jsr * \$cdca |             | Prendi i cont. reg. dal reg. DATA     |
| c530: 68       | pla          |             | Prendi il numero dallo STACK          |
| c531: f0 03    | beq * \$c536 |             | Se = 0 salta                          |
| c533: 20 3e c5 | jsr * \$c53e |             | Uscita colore                         |
| c536: ae 31 0a | ldx \$0a31   |             | Ricarica reg X                        |
| c539: a4 e7    | ldy \$e7     | [ scrt ]    | Carica y con bordo destro             |
| c53b: 60       | rts          |             |                                       |

## SCRITTURA DI UN CARATTERE IN A

|                |              |  |                                     |
|----------------|--------------|--|-------------------------------------|
| c53c: a9 01    | lda #\$01    |  | Metti 1 nel contatore               |
| c53e: a2 1e    | ldx #\$1e    |  | Seleziona reg. cont. parole         |
| c540: 20 cc cd | jsr * \$cdcc |  | Determinane il valore               |
| c543: 2c 00 d6 | bit * \$d600 |  | Controllo bit di status             |
| c546: 10 fb    | bpl * \$c543 |  | Ciclo di attesa                     |
| c548: a2 12    | ldx #\$12    |  | Aggiorna indirizzo alto             |
| c54a: 20 da cd | jsr * \$cdca |  | Carica valore attuale               |
| c54d: cd 3d 0a | cmp \$0a3d   |  | Confr.con byte HI ind. destinazione |
| c550: 90 ea    | bcc * \$c53c |  | Se diver.correz.errore              |
| c552: a2 13    | ldx #\$13    |  | Aggiorna ind. basso                 |
| c554: 20 da cd | jsr * \$cdca |  | Carica val. attuale                 |
| c557: cd 3c 0a | cmp \$0a3c   |  | Confronta con ind. dest. basso      |
| c55a: 90 e0    | bcc * \$c53c |  | Se diver.correz.errore              |
| c55c: 60       | rts          |  |                                     |

## CONTROLLO MATRICE DI TASTIERA

|             |           |            |                                      |
|-------------|-----------|------------|--------------------------------------|
| c55d: a5 01 | lda \$01  | [ r6510 ]  | Carica bit 6 da pag. ZERO            |
| c55f: 29 40 | and #\$40 |            | NOTA: Il bit 6 indica se e' stato    |
| c561: 49 40 | eor #\$40 |            | selezionato il modo 40 o 80 colonne  |
| c563: 4a    | lsr a     |            | Inverti il bit 6                     |
| c564: 4a    | lsr a     |            | alla posizione 4                     |
| c565: 85 d3 | sta \$d3  | [ shflag ] | Immagazz. modo 40/80                 |
| c567: a0 58 | ldy #\$58 |            | Metti codice per NESSUN TASTO in p.0 |
| c569: 84 d4 | sty \$d4  | [ sfdx ]   | Imm. punt. per tasto premuto=        |

|       |          |     |          |             |                                       |
|-------|----------|-----|----------|-------------|---------------------------------------|
| c56b: | a9 00    | lda | #\$00    |             | Controllo valore per matrice linee    |
| c56d: | 8d 00 dc | sta | * \$dc00 |             | Imm. matrici linee 1-8                |
| c570: | 8d 2f d0 | sta | * \$d02f |             | Idem per linee 9-11                   |
| c573: | ae 01 dc | ldx | * \$dc01 |             | Porta B = ingresso matrice colonne    |
| c576: | e0 ff    | cpx | #\$ff    |             | Controllo se tasto premuto            |
| c578: | d0 03    | bne | * \$c57d |             | Controllo quale tasto e' premuto      |
| c57a: | 4c 97 c6 | jmp | * \$c697 |             | Se nessun tasto continua              |
| c57d: | a8       | tay |          |             | Vis.cont.inizio tavola tastiera       |
| c57e: | ad 3e 03 | lda | \$033e   |             | Copia indirizzo basso tavola decodifi |
| c581: | 85 cc    | sta | \$cc     | [ keytab ]  | di tastiera in pagina 0               |
| c583: | ad 3f 03 | lda | \$033f   |             | Come sopra per indirizzo              |
| c586: | 85 cd    | sta | \$cd     | [ keytab+1] | alto                                  |
| c588: | a9 ff    | lda | #\$ff    |             | Controllo valore matrice tastiera     |
| c58a: | 8d 2f d0 | sta | * \$d02f |             | Fissa contr. linee 9-11 in HI         |
| c58d: | 2a       | rol | a        |             | Metti a 0 bit posiz. linea controllo  |
| c58e: | 24 d3    | bit | \$d3     | [ shflag ]  | Puntatore controllo 1-8 o 9-11        |
| c590: | 30 05    | bmi | * \$c597 |             | Se controllo linee 9-11 salta         |
| c592: | 8d 00 dc | sta | * \$dc00 |             | Controllo porta A                     |
| c595: | 10 03    | bpl | * \$c59a |             | Controllo salto matrici 9-11          |
| c597: | 8d 2f d0 | sta | * \$d02f |             | Controllo porta A 9-11                |
| c59a: | a2 08    | ldx | #\$08    |             | Fissa cont. per matrice 8 colonne     |
| c59c: | 48       | pha |          |             | Carica in A val.linea contr           |
| c59d: | ad 01 dc | lda | * \$dc01 |             | Uscita matrici colonne                |
| c5a0: | cd 01 dc | cmp | * \$dc01 |             | Confronto porte A-B                   |
| c5a3: | d0 f8    | bne | * \$c59d |             | Attesa                                |
| c5a5: | 4a       | lsr | a        |             | Controlla valore matrice              |
| c5a6: | b0 17    | bcs | * \$c5b5 |             | Colonne bit per bit                   |
| c5a8: | 48       | pha |          |             | Immag.valore usc. matrici colonne     |
| c5a9: | b1 cc    | lda | (\$cc),y | [ keytab ]  | Carica codice tasto da tavola tastie  |
| c5ab: | c9 08    | cmp | #\$08    |             | Codice tasto '8' = ALT                |
| c5ad: | f0 08    | beq | * \$c5b7 |             | Esegui la funzione corrispondente     |
| c5af: | c9 05    | cmp | #\$05    |             | Controllo per SHIFT,CBM, Ctrl         |
| c5b1: | b0 09    | bcs | * \$c5bc |             | Se non e' continua                    |
| c5b3: | c9 03    | cmp | #\$03    |             | Controllo se e' un tasto BREAK        |
| c5b5: | f0 05    | beq | * \$c5bc |             | Se e' un BREAK esegue la funz.        |
| c5b7: | 05 d3    | ora | \$d3     | [ shflag ]  | Puntatore pg.0                        |
| c5b9: | 85 d3    | sta | \$d3     | [ shflag ]  | Immag. in A                           |
| c5bb: | 2c 84 d4 | bit | * \$d484 |             | Vai all' indirizzo                    |
| c5be: | 68       | pla |          |             | Carica valore matrice colonne         |
| c5bf: | c8       | iny |          |             | Incr.+1 cont. vis.tavola tastiera     |
| c5c0: | ca       | dex |          |             | Decr.-1 cont. ciclo matrice col.      |
| c5c1: | d0 e2    | bne | * \$c5a5 |             | Ciclo controllo tutte colonne         |
| c5c3: | c0 59    | cpy | #\$59    |             | Contr. sia linee che colonne          |
| c5c5: | b0 10    | bcs | * \$c5d7 |             | Se positivo valut. tasto premuto      |
| c5c7: | 68       | pla |          |             | Prendi dallo Stack val.li.controllo   |
| c5c8: | 38       | sec |          |             | Fissa il flag di carry                |
| c5c9: | 2a       | rol | a        |             | Valore linea di controllo             |
| c5ca: | b0 c2    | bcs | * \$c58e |             | Continua controllo linee 1-8          |
| c5cc: | 8d 00 dc | sta | * \$dc00 |             | Fissa porta A val. controllo HI       |
| c5cf: | 2d d3    | rol | \$d3     | [ shflag ]  | Bit 7 in pattern FLAG                 |
| c5d1: | 38       | sec |          |             | Linee restanti dalla matrice          |
| c5d2: | 66 d3    | ror | \$d3     | [ shflag ]  | Linee 9-11 controllate da porta A     |
| c5d4: | 2a       | rol | a        |             | Operazione di Clear bit per 9-11      |
| c5d5: | d0 b7    | bne | * \$c58e |             | Controllo matrice successiva          |
| c5d7: | 06 d3    | asl | \$d3     | [ shflag ]  | Valutaz. risultato tastiera           |
| c5d9: | 46 d3    | lsr | \$d3     | [ shflag ]  | Controllo porta A                     |
| c5db: | 68       | pla |          |             | Dallo Stack valore a linea controllo  |
| c5dc: | a5 d4    | lda | \$d4     | [ sfdx ]    | Codice tasto premuto in A             |
| c5de: | 6c 3a 03 | jmp | (\$033a) |             | Lettura vettore tastiera              |
| c5el: | c9 57    | cmp | #\$57    |             | Controllo tasto NO SCROLL             |

|       |          |              |            |  |
|-------|----------|--------------|------------|--|
| c5e3: | d0 13    | bne * \$c5f8 |            | Se negativo salta                      |
| c5e5: | 24 f7    | bit \$f7     | [ locks ]  | Bit 6 in Pag.0 per pausa:l=disabil     |
| c5e7: | 70 5a    | bvs * \$c643 |            | Se nes. pausa esegui un RTS            |
| c5e9: | ad 25 0a | lda \$0a25   |            | Carica A con ultimo Car.               |
| c5ec: | d0 55    | bne * \$c643 |            | Se non e'=0 esci con RTS               |
| c5ee: | a9 0d    | lda #\$0d    |            | Inverti i bits 0,1 e 3 del puntatore   |
| c5f0: | 4d 21 0a | eor \$0a21   |            | di pag. 0 ed immagazzinali nel puntat  |
| c5f3: | 8d 21 0a | sta \$0a21   |            | di pausa della Pag.0                   |
| c5f6: | 50 30    | bvc * \$c628 |            | Routine di ripetiz. tastiera           |
| c5f8: | a5 d3    | lda \$d3     | [ shflag ] | Carica A                               |
| c5fa: | f0 55    | beq * \$c651 |            | Valutazione normale                    |
| c5fc: | c9 10    | cmp #\$10    |            | Contr. per scelta set 40 car.          |
| c5fe: | f0 44    | beq * \$c644 |            | Se si esegui per 40 caratt.            |
| c600: | c9 08    | cmp #\$08    |            | Controllo pressione tasto ALT          |
| c602: | f0 42    | beq * \$c646 |            | Se posit. esegui                       |
| c604: | 29 07    | and #\$07    |            | Mascheramento bits 3-7                 |
| c606: | c9 03    | cmp #\$03    |            | Controllo pressione CBM shiftato       |
| c608: | d0 25    | bne * \$c62f |            | Se negativo ricontrolla                |
| c60a: | a5 f7    | lda \$f7     | [ locks ]  | Controllo per CBM shift SWITCH         |
| c60c: | 30 43    | bmi * \$c651 |            | Se si non ripetere routine             |
| c60e: | ad 25 0a | lda \$0a25   |            | Carica ultimo car.                     |
| c611: | d0 3e    | bne * \$c651 |            | Se non e' 0 ripeti routine             |
| c613: | 24 d7    | bit \$d7     | [ mode ]   | Controllo modo 40/80                   |
| c615: | 10 09    | bpl * \$c620 |            | Se positivo siamo a 40 colonne         |
| c617: | a5 f1    | lda \$f1     | [ color ]  | Carica in A codice colore car.         |
| c619: | 49 80    | eor #\$80    |            | Inverti bit 7 codice colore            |
| c61b: | 85 f1    | sta \$f1     | [ color ]  | Immag. codice colore                   |
| c61d: | 4c 28 c6 | jmp * \$c628 |            | Salta interr. caratt. VIC              |
| c620: | ad 2c 0a | lda \$0a2c   |            | Puntatore sistema per testo            |
| c623: | 49 02    | eor #\$02    |            | Inverti bit 2 del puntatore            |
| c625: | 8d 2c 0a | sta \$0a2c   |            | Carica la base del puntatore           |
| c628: | a9 08    | lda #\$08    |            | Inizializza il sistema di puntatori    |
| c62a: | 8d 25 0a | sta \$0a25   |            | con 8 per l' ultimo carattere          |
| c62d: | d0 22    | bne * \$c651 |            | Vai alla ripetizione della routine     |
| c62f: | 0a       | asl a        |            | Moltiplica per 2                       |
| c630: | c9 08    | cmp #\$08    |            | Confronto per SHIFT o CBM              |
| c632: | 90 12    | bcc * \$c646 |            | Se trovato carica tavola di decodifica |
| c634: | a9 06    | lda #\$06    |            | Valore di default CTRL in A            |
| c636: | a6 d4    | ldx \$d4     | [ sfdx ]   | Controlla salto tavola decodif         |
| c638: | e0 0d    | cpx #\$0d    |            | Controlla se 13mo tasto                |
| c63a: | d0 0a    | bne * \$c646 |            | Fissa il flag di pausa e salta         |
| c63c: | 24 f7    | bit \$f7     | [ locks ]  | Controllo consenso pausa/Ctrl-s        |
| c63e: | 70 06    | bvs * \$c646 |            | Nessun consenso, vai a tav. decod      |
| c640: | 8e 21 0a | stx \$0a21   |            | Imm X flag pausa valore 13             |
| c643: | 60       | rts          |            |  |

#### FISSA INDIRIZZO INIZIALE TAVOLA DI DECODIFICA

|       |          |              |               |  |
|-------|----------|--------------|---------------|--|
| c644: | a9 0a    | lda #\$0a    |               | Fissa il valore di default alla tav    |
| c646: | aa       | tax          |               | N. tavola di decodif in reg. X         |
| c647: | bd 3e 03 | lda \$033e,x |               | Copia indirizzo basso tavola di decodi |
| c64a: | 85 cc    | sta \$cc     | [ keytab ]    | fica nella memoria in pag. 0           |
| c64c: | bd 3f 03 | lda \$033f,x |               | Idem per indirizzo                     |
| c64f: | 85 cd    | sta \$cd     | [ keytab+ 1 ] | alto                                   |
| c651: | a4 d4    | ldy \$d4     | [ sfdx ]      | Inizio tavola in Y                     |
| c653: | b1 cc    | lda (\$cc),y | [ keytab ]    | Carica A con codice caratt. da tavola  |
| c655: | aa       | tax          |               | Immag. car. in X                       |
| c656: | c4 d5    | cpy \$d5     | [ ltsx ]      | Confronta con punt. per tasto attua.   |
| c658: | f0 07    | beq * \$c661 |               | Se uguale ripeti controllo             |
| c65a: | a0 10    | ldy #\$10    |               | Contatore ritardo rip. tasto premuto   |

|                |                |            |                                      |
|----------------|----------------|------------|--------------------------------------|
| c65c: 8c 24 0a | sty \$0a24     |            | Inizializza cont. con \$10           |
| c65f: d0 36    | bne * \$c697   |            | Vai alla rout. di tasto premuto      |
| c661: 29 7f    | and # \$7f     |            | Mask out bit 7                       |
| c663: 2c 22 0a | bit \$0a22     |            | Controllo punt. tasto premuto        |
| c666: 30 16    | bmi * \$c67e   |            | Tutti i tasti abilitati              |
| c668: 70 5a    | bvs * \$c6c4   |            | Continuazione rout. prec.            |
| c66a: c9 7f    | cmp # \$7f     |            | Controllo validita' carattere        |
| c66c: f0 29    | beq * \$c697   |            | Positivo, e' letto il val.di default |
| c66e: c9 14    | cmp # \$14     |            | Contr. tasto DEL                     |
| c670: f0 0c    | beq * \$c67e   |            | Se pos. ripetere                     |
| c672: c9 20    | cmp # \$20     |            | Controllo per barra spaziatrice      |
| c674: f0 08    | beq * \$c67e   |            | Se positivo ripeti valut.            |
| c676: c9 1d    | cmp # \$1d     |            | Controllo per CRSR-destro            |
| c678: f0 04    | beq * \$c67e   |            | Se positivo ripeti                   |
| c67a: c9 11    | cmp # \$11     |            | Controllo per CRSR-basso             |
| c67c: d0 46    | bne * \$c6c4   |            | Se positivo ripeti valutaz.          |
| c67e: ac 24 0a | ldy \$0a24     |            | Carica cont. per ritardo ripet.      |
| c681: f0 05    | beq * \$c688   |            | Se cont.=0 salta                     |
| c683: ce 24 0a | dec \$0a24     |            | Contatore -1                         |
| c686: d0 3c    | bne * \$c6c4   |            | Controllo per diverso da 0           |
| c688: ce 23 0a | dec \$0a23     |            | Decrem-1 velocita' contatore         |
| c68b: d0 37    | bne * \$c6c4   |            | Controllo per diverso da 0           |
| c68d: a0 04    | ldy # \$04     |            | Calcolo velocita' tasto premuto      |
| c68f: 8c 23 0a | sty \$0a23     |            | Reinizializza con \$04               |
| c692: a4 d0    | ldy \$d0       | [ ndx ]    | Metti in Y val.salto buffer coda tas |
| c694: 88       | dey            |            | Decrementa buffer di 1               |
| c695: 10 2d    | bpl * \$c6c4   |            | Se era maggiore esegui RTS           |
| c697: 4e 25 0a | lsr \$0a25     |            | Dividi ultimo car. per 2             |
| c69a: a4 d4    | ldy \$d4       | [ sfdx ]   | Copia visual. a inizio tav. decod    |
| c69c: 84 d5    | sty \$d5       | [ ltsx ]   | Carica Y con punt. attuale tasto     |
| c69e: e0 ff    | cpx # \$ff     |            | Controlla se era un carattere        |
| c6a0: f0 22    | beq * \$c6c4   |            | Se pos.leggi val. di default e RTS   |
| c6a2: a9 00    | lda # \$00     |            | Esegui reset del punt. pausa/Ctrl-s  |
| c6a4: 8d 21 0a | sta \$0a21     |            | Ricerca carattere valido             |
| c6a7: 8a       | txa            |            | Copia cod caratt. in A               |
| c6a8: a6 d3    | ldx \$d3       | [ shflag ] | Carica in X SHIFT-FLAG               |
| c6aa: 6c 3c 03 | jmp (\$033c)   |            | Vai alla routine KEY                 |
| c6ad: a2 09    | ldx # \$09     |            | Contatore di ciclo                   |
| c6af: dd dd c6 | cmp * \$c6dd,x |            | Confronta A con codice tasto         |
| c6b2: f0 16    | beq * \$c6ca   |            | Rilevato tastofunzione, elab.        |
| c6b4: ca       | dex            |            | Decrementa cont. ciclo di 1          |
| c6b5: 10 f8    | bpl * \$c6af   |            | Esecuz. ciclo per tutti i confronti  |
| c6b7: ae d0    | ldx \$d0       | [ ndx ]    | Indice coda buffer tastiera          |
| c6b9: ec 20 0a | cpx \$0a20     |            | Confronto con max. dimensione        |
| c6bc: b0 06    | bcs * \$c6c4   |            | Se e' max. dim. salta                |
| c6be: 9d 4a 03 | sta \$034a,x   |            | Immetti car. in buffer tastiera      |
| c6c1: e8       | inx            |            | Incr. di 1 coda buffer tastiera      |
| c6c2: 86 d0    | stx \$d0       | [ ndx ]    | Imm. in X 1 carattere                |
| c6c4: a9 7f    | lda # \$7f     |            | Controllo matrice tastiera           |
| c6c6: 8d 00 dc | sta * \$dc00   |            | Immag. valore di default             |
| c6c9: 60       | rts            |            |                                      |

#### ----- INIZIALIZZA BUFFER DI TASTIERA

|                |              |           |                                      |
|----------------|--------------|-----------|--------------------------------------|
| c6ca: bd 00 10 | lda \$1000,x |           | Carica in A lunghezza da rout. KEY   |
| c6cd: 85 d1    | sta \$d1     | [ kyndx ] | Immag. in KEY cont. carattere        |
| c6cf: a9 00    | lda # \$00   |           | La posizione della routine KEY nell' |
| c6d1: ca       | dex          |           | intera tavola e' determinata quando  |
| c6d2: 30 06    | bmi * \$c6da |           | tutte le lunghezze sono state aggu.  |
| c6d4: 18       | clc          |           | Clear del Carry per somma            |

|                |              |            |                              |
|----------------|--------------|------------|------------------------------|
| c6d5: 7d 00 10 | adc \$1000,x |            | Aggiungi la lunghezza di KEY |
| c6d8: 90 f7    | bcc * \$c6d1 |            | Se nessun Overflow continua  |
| c6da: 85 d2    | sta \$d2     | [ keyidx ] | Immagazzina il puntatore     |
| c6dc: 60       | rts          |            |                              |

## CODICI DEI TASTI FUNZIONE

|             |          |        |
|-------------|----------|--------|
| c6dd: 85 89 | (\$8985) | F1 F2  |
| c6df: 86 8a | (\$8a86) | F3 F4  |
| c6e1: 87 8b | (\$8b87) | F5 F6  |
| c6e3: 88 8c | (\$8c88) | F7 F8  |
| c6e5: 83 84 | (\$8483) | F9 F10 |

## CURSORE IN MODO FLASH

|                |              |           |                                       |
|----------------|--------------|-----------|---------------------------------------|
| c6e7: 24 d7    | bit \$d7     | [ mode ]  | Controllo modo 40/80 colonne          |
| c6e9: 30 41    | bmi * \$c72c |           | Se 80 colonne fine                    |
| c6eb: ad 27 0a | lda \$0a27   |           | Carica modo cursore VIC               |
| c6ee: d0 3c    | bne * \$c72c |           | Se disabilit. fine                    |
| c6f0: ce 28 0a | dec \$0a28   |           | Decrementa contatore di FLASH         |
| c6f3: d0 37    | bne * \$c72c |           | Se diverso da 0 fine                  |
| c6f5: ad 26 0a | lda \$0a26   |           | Carica cursore VIC                    |
| c6f8: 29 c0    | and #\$c0    |           | Mask out bits 0-5                     |
| c6fa: c9 c0    | cmp #\$c0    |           | Controllo cursore                     |
| c6fc: f0 2e    | beq * \$c72c |           | Se e' OFF fine                        |
| c6fe: a9 14    | lda #\$14    |           | Metti il contatore cursore FLASH del  |
| c700: 8d 28 0a | sta \$0a28   |           | VIC = 20 (\$14)                       |
| c703: a4 ec    | ldy \$ec     | [ pntr ]  | Metti in Y attuale colonna curs.      |
| c705: ae 2a 0a | ldx \$0a2a   |           | Carica alla pos. del cursore col.     |
| c708: b1 e0    | lda (\$e0),y | [ pnt ]   | Carattere attuale colonna             |
| c70a: 2c 26 0a | bit \$0a26   |           | Controllo modo cursore VIC            |
| c70d: 30 10    | bmi * \$c71f |           | Ancora carattere normale              |
| c70f: 8d 29 0a | sta \$0a29   |           | Car.in pos. curs. prima di FLASH      |
| c712: 20 7c c1 | jsr * \$c17c |           | Fissa ind. RAM colore                 |
| c715: b1 e2    | lda (\$e2),y | [ user ]  | Metti colore in pos. cursore          |
| c717: 8d 2a 0a | sta \$0a2a   |           | Salva il colore prima di FLASH        |
| c71a: a6 f1    | ldx \$f1     | [ color ] | Carica in X cod. car.colore           |
| c71c: ad 29 0a | lda \$0a29   |           | Carattere in pos.curs.prima del FLASH |
| c71f: 49 80    | eor #\$80    |           | Inverti bit negativo                  |
| c721: 20 40 cc | jsr * \$cc40 |           | Salva carattere e colore              |
| c724: ad 26 0a | lda \$0a26   |           | Carica modo cursore VIC               |
| c727: 49 80    | eor #\$80    |           | Condizione di FLASH negata            |
| c729: 8d 26 0a | sta \$0a26   |           | Salva di nuovo                        |
| c72c: 60       | rts          |           |                                       |

## INGRESSO ROUTINE BSOUT

|                |              |           |                                      |
|----------------|--------------|-----------|--------------------------------------|
| c72d: 85 ef    | sta \$ef     | [ datax ] | Salva car.da stamp. in Pag. 0        |
| c72f: 48       | pha          |           | Salva cont. di A in STACK            |
| c730: 8a       | txa          |           | Salva X in STACK                     |
| c731: 48       | pha          |           | Continua                             |
| c732: 98       | tya          |           | Salva Y in stack                     |
| c733: 48       | pha          |           | Continua                             |
| c734: ad 21 0a | lda \$0a21   |           | Contr.contenuti flag pausa in Pag. 0 |
| c737: d0 fb    | bne * \$c734 |           | Attendi fino a FLAG=0                |
| c739: 85 d6    | sta \$d6     | [ crsm ]  | Esegui CLEAR flag input/get          |
| c73b: a9 c3    | lda #\$c3    |           | Carica Byte HI                       |
| c73d: 48       | pha          |           | Immetti in Stack                     |
| c73e: a9 0b    | lda #\$0b    |           | Carica Byte basso                    |

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| c740: 48       | pha          |            | Immetti in Stack                      |
| c741: a4 ec    | ldy \$ec     | [ pntr ]   | Metti in Y attuale colonna cursore    |
| c743: a5 ef    | lda \$ef     | [ datax ]  | Immag.temp. carattere da stamp.       |
| c745: c9 0d    | cmp #\$0d    |            | Controllo se e' Rit. Carr.            |
| c747: f0 26    | beq * \$c76f |            | Se pos. esegui                        |
| c749: c9 8d    | cmp #\$8d    |            | Controllo se e' SHIFT Rit. Carr.      |
| c74b: f0 22    | beq * \$c76f |            | Se positivo esegui                    |
| c74d: a6 f0    | ldx \$f0     | [ 1stchr ] | Valore del precedente carattere       |
| c74f: e0 1b    | cpx #\$1b    |            | Se il carattere era una ESCAPE esegui |
| c751: d0 03    | bne * \$c756 |            | la manipolazione della sequenza di    |
| c753: 4c be c9 | jmp * \$c9be |            | ESCAPE saltando a \$C756              |
| c756: aa       | tax          |            | Carat.davis. in X                     |
| c757: 10 03    | bpl * \$c75c |            | Controllo se e' da 0-127              |
| c759: 4c 02 c8 | jmp * \$c802 |            | Se neg = ASCII esteso                 |
| c75c: c9 20    | cmp #\$20    |            | Controllo se car.< Blank              |
| c75e: 90 56    | bcc * \$c7b6 |            | Se pos. valuta codici controllo       |
| c760: c9 60    | cmp #\$60    |            | E' una lettera                        |
| c762: 90 03    | bcc * \$c767 |            | Se pos. visualizza                    |
| c764: 29 df    | and \$df     |            | Mask out bit5                         |
| c766: 2c 29 3f | bit \$3f29   |            | Vai a \$C769                          |
| c769: 20 ff c2 | jsr * \$c2ff |            | Controllo apici                       |
| c76c: 4c 22 c3 | jmp * \$c322 |            | Vis. carattere                        |
| c76f: 20 c3 cb | jsr * \$cbc3 |            | Cerca la fine linea input             |
| c772: e8       | inx          |            | Incrementa di 1                       |
| c773: 20 85 cb | jsr * \$cb85 |            | Clear bit di overflow                 |
| c776: a4 e6    | ldy \$e6     | [ sclf ]   | Carica bordo sinistro in Y            |
| c778: 84 ec    | sty \$ec     | [ pntr ]   | Immag. attuale posiz. cursore         |
| c77a: 20 63 c3 | jsr * \$c363 |            | Esegui un linefeed                    |
| c77d: a5 f1    | lda \$f1     | [ color ]  | Carica in A codice col.car            |
| c77f: 29 cf    | and \$cf     |            | Metti in OFF reverse e flash          |
| c781: 85 f1    | sta \$f1     | [ color ]  | Immag. codice colore                  |
| c783: a9 00    | lda #\$00    |            | Carica A con 0                        |
| c785: 85 f5    | sta \$f5     | [ insrt ]  | Esegui un clear dei BITS              |
| c787: 85 f3    | sta \$f3     | [ rvs ]    | Flag di Reverse                       |
| c789: 85 f4    | sta \$f4     | [ qtsw ]   | Flag apici                            |
| c78b: 60       | rts          |            |                                       |

## CODICI DI CONTROLLO

|             |          |                  |                    |
|-------------|----------|------------------|--------------------|
| c78c: 02 07 | (\$0702) | 2=sottolineatura | 7=segnale acustico |
| c78e: 09 0a | (\$0a09) | 9=tabulatore     | A=linefeed         |
| c790: 0b 0c | (\$0c0b) | B=Shift-1/CBM    | C=Shift-unlok/CBM  |
| c792: 0e 0f | (\$0f0e) | E=minuscolo      | F=Flash on         |
| c794: 11 12 | (\$1211) | 11=cursor up     | 12=Reverse on      |
| c796: 13 14 | (\$1413) | 13=home          | 14=delete          |
| c798: 18 1d | (\$1d18) | 18=tab           | 1D=Cursor a destra |

## INDIRIZZI DELLE ROUTINES CHE ESEGUONO I CODICI DI CUI SOPRA

|             |          |                         |
|-------------|----------|-------------------------|
| c79a: c6 c8 | (\$c8c6) | Sottolineatura          |
| c79c: 8d c9 | (\$c98d) | Tabulatore              |
| c79e: 4e c9 | (\$c94e) | Segnale acustico (BELL) |
| c7a0: b0 c9 | (\$c9b0) | Linefeed                |
| c7a2: a5 c8 | (\$c8a5) | Disabilita SHIFT/CBM    |
| c7a4: ab c8 | (\$c8ab) | Abilita SHIFT/CBM       |
| c7a6: 7f c8 | (\$c87f) | Minuscole               |
| c7a8: d4 c8 | (\$c8d4) | Flash on                |
| c7aa: 59 c8 | (\$c859) | Cursor UP               |
| c7ac: cl c8 | (\$c8cl) | Reverse ON              |

|             |          |                             |
|-------------|----------|-----------------------------|
| c7ae: b2 c8 | (\$c8b2) | Home                        |
| c7b0: 1a c9 | (\$c91a) | Delete                      |
| c7b2: 60 c9 | (\$c960) | Tabulatore (fissa/cancella) |
| c7b4: 53 c8 | (\$c853) | Cursore a destra            |

## ESECUZIONE CODICI DI CONTROLLO

|                |                    |                                     |
|----------------|--------------------|-------------------------------------|
| c7b6: 6c 34 03 | jmp (\$0334)       | Vettore uscita carattere            |
| c7b9: c9 1b    | cmp #1b            | Contr. se e' un ESCAPE              |
| c7bb: f0 38    | beq * \$c7f5       | Se pos. fine                        |
| c7bd: a6 f5    | ldx \$f5 [ insrt ] | Contr. se fissato modo inserim      |
| c7bf: d0 08    | bne * \$c7c9       | Pos. visual. carat. in REVERSE      |
| c7c1: c9 14    | cmp #14            | Controllo per DELETE                |
| c7c3: f0 0b    | beq * \$c7d0       | Esegui                              |
| c7c5: a6 f4    | ldx \$f4 [ qtsw ]  | Contr. flag modo apici inserito     |
| c7c7: f0 07    | beq * \$c7d0       | Se posit. vis. caratt. in REVERSE   |
| c7c9: a2 00    | ldx #00            | Eseg. CLEAR su ultimo caratt. vis.  |
| c7cb: 86 ef    | stx \$ef [ datax ] | Continua                            |
| c7cd: 4c 26 c3 | jmp * \$c326       | Vis. caratt. in REVERSE             |
| c7d0: a2 0d    | ldx #0d            | Car. X come cont. cod. controllo    |
| c7d2: dd 8c c7 | cmp * \$c78c,x     | Confronta con tavola                |
| c7d5: f0 1f    | beq * \$c7f6       | Se trovato esegui                   |
| c7d7: ca       | dex                | Decrem. di 1 il cont.               |
| c7d8: 10 f8    | bpl * \$c7d2       | Confronta con succ. valore          |
| c7da: a2 0f    | ldx #0f            | Esegui il confronto con i 16 codici |
| c7dc: dd 4c ce | cmp * \$ce4c,x     | colore per ev. cambiamento          |
| c7df: f0 04    | beq * \$c7e5       | Se trovato salta                    |
| c7e1: ca       | dex                | Decrem. cont. di 1                  |
| c7e2: 10 f8    | bpl * \$c7dc       | Confr. con valore succ.             |
| c7e4: 60       | rts                |                                     |

## FISSA I COLORI IN MODO 40 COLONNE

|             |                    |                                       |
|-------------|--------------------|---------------------------------------|
| c7e5: 24 d7 | bit \$d7 [ mode ]  | Controllo modo 40/80 colonne          |
| c7e7: 30 03 | bmi * \$c7ec       | Se 80 colonne salta                   |
| c7e9: 86 f1 | stx \$f1 [ color ] | Immag. codice colore per vis. caratt. |
| c7eb: 60    | rts                |                                       |

## FISSA I COLORI IN MODO 80 COLONNE

|                |                    |  |
|----------------|--------------------|--|
| c7ec: a5 f1    | lda \$f1 [ color ] | Metti in A cod. colore per vis.caratt. |
| c7ee: 29 f0    | and #f0            | Mask out del NIBBLE LO (bits 0-3)      |
| c7f0: 1d 5c ce | ora * \$ce5c,x     | Esegui OR con tavola cod. colore       |
| c7f3: 85 f1    | sta \$f1 [ color ] | Immag. codice colore per vis. caratt.  |
| c7f5: 60       | rts                |  |

## ESECUZIONE CODICI DI CONTROLLO

|                |                |                                      |
|----------------|----------------|--------------------------------------|
| c7f6: 8a       | txa            | Punt. in A e poi fine                |
| c7f7: 0a       | asl a          | Moltiplica per 2                     |
| c7f8: aa       | tax            | Deve essere trovato valore di 16 bit |
| c7f9: bd 9b c7 | lda * \$c79b,x | Car.byte basso di indir. partenza    |
| c7fc: 48       | pha            | Immetti in A                         |
| c7fd: bd 9a c7 | lda * \$c79a,x | Come sopra per Byte HI               |
| c800: 48       | pha            | Continua                             |
| c801: 60       | rts            |                                      |

## CONTROLLO PER ESTENSIONE ASCII

|                |              |                               |
|----------------|--------------|-------------------------------|
| c802: 6c 36 03 | jmp (\$0336) | Vettore car. uscita con SHIFT |
|----------------|--------------|-------------------------------|



|       |    |    |     |          |           |                                   |
|-------|----|----|-----|----------|-----------|-----------------------------------|
| c805: | 29 | 7f | and | #\$7f    |           | Mask out bit 7                    |
| c807: | c9 | 20 | cmp | #\$20    |           | Confronta con car. SPAZIO         |
| c809: | 90 | 09 | bcc | * \$c814 |           | Contr. se minore di 32            |
| c80b: | c9 | 7f | cmp | #\$7f    |           | Contr. se codice ASCII            |
| c80d: | d0 | 02 | bne | * \$c811 |           | Se neg. salta                     |
| c80f: | a9 | 5e | lda | #\$5e    |           | Codice ASCII                      |
| c811: | 4c | 20 | c3  | jmp      | * \$c320  | Visualizzazione                   |
| c814: | a6 | f4 | ldx | \$f4     | [ qtsw ]  | Carica flag modo apici            |
| c816: | f0 | 05 | beq | * \$c81d |           | Salta se non inserito             |
| c818: | 09 | 40 | ora | #\$40    |           | Fissa bit 6                       |
| c81a: | 4c | 26 | c3  | jmp      | * \$c326  | Vis. come caratt. REVERSE         |
| c81d: | c9 | 14 | cmp | #\$14    |           | Contr. se e' un car. INSERT       |
| c81f: | d0 | 03 | bne | * \$c824 |           | Se negativo salta                 |
| c821: | 4c | e3 | c8  | jmp      | * \$c8e3  | Se positivo esegui                |
| c824: | a6 | f5 | ldx | \$f5     | [ insrt ] | Carica flag modo inserim.         |
| c826: | d0 | f0 | bne | * \$c818 |           | Se pos. procedi come se con apici |
| c828: | c9 | 11 | cmp | #\$11    |           | Confronta per CURSOR UP           |
| c82a: | f0 | 3b | beq | * \$c867 |           | Salta se CURSOR UP e' in ON       |
| c82c: | c9 | 1d | cmp | #\$1d    |           | Confr. se cursor a sinistra       |
| c82e: | f0 | 45 | beq | * \$c875 |           | Se pos. esegui                    |
| c830: | c9 | 0e | cmp | #\$0e    |           | Confronta se maiuscolo            |
| c832: | f0 | 5e | beq | * \$c892 |           | Esegui                            |
| c834: | c9 | 12 | cmp | #\$12    |           | Contr. att. modo REVERSE          |
| c836: | d0 | 03 | bne | * \$c83b |           | Se neg. salta                     |
| c838: | 4c | bf | c8  | jmp      | * \$c8bf  | Clear modo RVS                    |
| c83b: | c9 | 02 | cmp | #\$02    |           | Contr. att. sottolineatura        |
| c83d: | d0 | 03 | bne | * \$c842 |           | Se neg. salta                     |
| c83f: | 4c | ce | c8  | jmp      | * \$c8ce  | Fissa modo sottoli.               |
| c842: | c9 | 0f | cmp | #\$0f    |           | Contr. FLASH                      |
| c844: | d0 | 03 | bne | * \$c849 |           | Se neg. Salta                     |
| c846: | 4c | dc | c8  | jmp      | * \$c8dc  | Clear del modo FLASH              |
| c849: | c9 | 13 | cmp | #\$13    |           | Contr. per CLR/HOME               |
| c84b: | d0 | 03 | bne | * \$c850 |           | Se neg. salta                     |
| c84d: | 4c | 42 | c1  | jmp      | * \$c142  | Esegui un CLEAR finestra          |
| c850: | 09 | 80 | ora | #\$80    |           | Clear bit 7                       |
| c852: | d0 | 86 | bne | * \$c7da |           | Vai a valutare se e' un colore    |
| c854: | 20 | ed | cb  | jsr      | * \$cbed  | Sposta di 1 il cursore a destra   |
| c857: | b0 | 04 | bcs | * \$c85d |           | Inizio nuova linea                |
| c859: | 60 |    | rts |          |           |                                   |

## CURSORE IN BASSO

|       |    |    |     |          |           |  |
|-------|----|----|-----|----------|-----------|--|
| c85a: | 20 | 63 | c3  | jsr      | * \$c363  | Prep. Linefeed                         |
| c85d: | 20 | 74 | cb  | jsr      | * \$cb74  | Contr. bit overflow-line               |
| c860: | b0 | 03 | bcs | * \$c865 |           | Trovata linea troppo lunga             |
| c862: | 38 |    | sec |          |           | Fissa il carry                         |
| c863: | 66 | e8 | ror | \$e8     | [ lsexp ] | Ruota il carry nella linea di inp.iniz |
| c865: | 18 |    | clc |          |           | Clear del carry                        |
| c866: | 60 |    | rts |          |           |  |

## CURSORE IN ALTO

|       |    |    |     |          |           |                                 |
|-------|----|----|-----|----------|-----------|---------------------------------|
| c867: | a6 | e5 | ldx | \$e5     | [ sctop ] | Carica in X bordo sup. finestra |
| c869: | e4 | eb | cpx | \$eb     | [ tblx ]  | Confronta con linea att. curs.  |
| c86b: | b0 | f9 | bcs | * \$c866 |           | Contr. se minore o =            |
| c86d: | 20 | 5d | c8  | jsr      | * \$c85d  | Fissa linea di status           |
| c870: | c6 | eb | dec | \$eb     | [ tblx ]  | Decr. di 1 att. linea curs.     |
| c872: | 4c | 5c | c1  | jmp      | * \$c15c  | Determ. ind. inizio att. linea  |
| c875: | 20 | 00 | cc  | jsr      | * \$cc00  | Cursore a sinistra              |
| c878: | b0 | ec | bcs | * \$c866 |           | Cursore immobile                |

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| c87a: | d0 e9    | bne * \$c865 |           | Cursore mosso ma non su nuova linea   |
| c87c: | e6 eb    | inc \$eb     | [ tblx ]  | Increment di 1 pos. curs.             |
| c87e: | d0 ed    | bne * \$c86d |           | Salto incond. a indirizzo             |
| c880: | 24 d7    | bit \$d7     | [ mode ]  | Controllo modo 40/80 col.             |
| c882: | 30 07    | bmi * \$c88b |           | Salta se 80 colonne                   |
| c884: | ad 2c 0a | lda \$0a2c   |           | Carica ind. base rout. CHARROM        |
| c887: | 09 02    | ora #\$02    |           | Fissa bit 0 e 1                       |
| c889: | d0 10    | bne * \$c89b |           | Salto incondiz. a ind.                |
| c88b: | a5 f1    | lda \$f1     | [ color ] | Metti in A cod. col. caratt.          |
| c88d: | 09 80    | ora #\$80    |           | Seleziona altro set caratt.           |
| c88f: | 85 f1    | sta \$f1     | [ color ] | Immag. cod. colo. per caratt. da vis. |
| c891: | 60       | rts          |           |                                       |

-----  
CONTROLLO TASTI SHIFT & COMMODORE

|       |          |              |          |                                |
|-------|----------|--------------|----------|--------------------------------|
| c892: | 24 d7    | bit \$d7     | [ mode ] | Controllo modo 40/80 colonne   |
| c894: | 30 09    | bmi * \$c89f |          | Salta se 80 colonne            |
| c896: | ad 2c 0a | lda \$0a2c   |          | Carica ind. base rout. CHARROM |
| c899: | 29 fd    | and #\$fd    |          | Esegui un clear su bits 0 e 1  |
| c89b: | 8d 2c 0a | sta \$0a2c   |          | Immag. come nuovi ind. base    |
| c89e: | 60       | rts          |          |                                |

-----  
C.S. PER 80 COLONNE

|       |       |           |           |  |
|-------|-------|-----------|-----------|--|
| c89f: | a5 f1 | lda \$f1  | [ color ] | Car. codice col. per vis. caratt. in A |
| c8a1: | 29 7f | and #\$7f |           | Clear bit 7                            |
| c8a3: | 85 f1 | sta \$f1  | [ color ] | Fissa cod. colore per vis. caratt.     |
| c8a5: | 60    | rts       |           |  |

-----  
ABILITAZIONE/DISABILITAZIONE TASTI C.S.

|       |       |              |           |                             |
|-------|-------|--------------|-----------|-----------------------------|
| c8a6: | a9 80 | lda #\$80    |           | Fissa bit 7                 |
| c8a8: | 05 f7 | ora \$f7     | [ locks ] | Esegui OR con flag registr. |
| c8aa: | 30 04 | bmi * \$c8b0 |           | Salto incond. a ind.        |
| c8ac: | a9 7f | lda #\$7f    |           | Clear su bit 7              |
| c8ae: | 25 f7 | and \$f7     | [ locks ] | Abilitat il flag            |
| c8b0: | 85 f7 | sta \$f7     | [ locks ] | Salva il valore             |
| c8b2: | 60    | rts          |           |                             |

-----  
CONTROLLO TASTI HOME

|       |          |              |            |                                |
|-------|----------|--------------|------------|--------------------------------|
| c8b3: | a5 f0    | lda \$f0     | [ 1stchr ] | Carica si A ult. caratt.       |
| c8b5: | c9 13    | cmp #\$13    |            | Contr. se HOME                 |
| c8b7: | d0 03    | bne * \$c8bc |            | Se neg. fine routine           |
| c8b9: | 20 24 ca | jsr * \$ca24 |            | Cancella finestra              |
| c8bc: | 4c 50 c1 | jmp * \$c150 |            | Vai a rout. di CURSOR HOME     |
| c8bf: | a9 00    | lda \$00     |            | Carica A con 0, clear modo RVS |
| c8c1: | 2c a9 80 | bit \$80a9   |            | Salta a \$C8C4                 |
| c8c4: | 85 f3    | sta \$f3     | [ rvs ]    | Immag. in A il flag            |
| c8c6: | 60       | rts          |            |                                |

-----  
ATTIVAZIONE SOTTOLINEATURA

|       |       |           |           |                                   |
|-------|-------|-----------|-----------|-----------------------------------|
| c8c7: | a5 f1 | lda \$f1  | [ color ] | Carica in A cod. col. per caratt. |
| c8c9: | 09 20 | ora #\$20 |           | Fissa bit 6 per sottolin.         |
| c8cb: | 85 f1 | sta \$f1  | [ color ] | Immag. cod. colore caratt.        |
| c8cd: | 60    | rts       |           |                                   |

-----  
DISATTIVAZIONE SOTTOLINEATURA

|       |       |           |           |                                    |
|-------|-------|-----------|-----------|------------------------------------|
| c8ce: | a5 f1 | lda \$f1  | [ color ] | Vedi sopra                         |
| c8d0: | 29 df | and #\$df |           | Esegui un clear su bit 6 per sott. |
| c8d2: | 85 f1 | sta \$f1  | [ color ] | Vedi sopra                         |
| c8d4: | 60    | rts       |           |                                    |

-----  
ATTIVAZIONE MODO FLASH

|       |       |           |           |                                   |
|-------|-------|-----------|-----------|-----------------------------------|
| c8d5: | a5 f1 | lda \$f1  | [ color ] | Vedi sopra                        |
| c8d7: | 09 10 | ora #\$10 |           | Fissa bit 4 per modo flash attivo |
| c8d9: | 85 f1 | sta \$f1  | [ color ] | Vedi sopra                        |
| c8db: | 60    | rts       |           |                                   |

-----  
DISATTIVAZIONE MODO FLASH

|       |       |           |           |  |
|-------|-------|-----------|-----------|--|
| c8dc: | a5 f1 | lda \$f1  | [ color ] | Vedi sopra                             |
| c8de: | 29 ef | and #\$ef |           | Esegui un clear su bit 4 dis.modo flas |
| c8e0: | 85 f1 | sta \$f1  | [ color ] | Vedi sopra                             |
| c8e2: | 60    | rts       |           |  |

-----  
ABILITA MODO INSERT

|       |          |              |            |                                  |
|-------|----------|--------------|------------|----------------------------------|
| c8e3: | 20 1e cc | jsr * \$cc1e |            | Copia coord. cursore             |
| c8e6: | 20 c3 cb | jsr * \$cbc3 |            | Cerca fine linea input           |
| c8e9: | e4 df    | cpx \$df     | [ keytmp ] | Confr. linea con linea curs.     |
| c8eb: | d0 02    | bne * \$c8ef |            | Se e' cambiata vai indir.        |
| c8ed: | c4 de    | cpy \$de     | [ keybnk ] | Confr. col. con att. colonna     |
| c8ef: | 90 21    | bcc * \$c912 |            | Se minore vai ad ind.            |
| c8f1: | 20 3e c3 | jsr * \$c33e |            | Cursore a fine linea             |
| c8f4: | b0 22    | bcs * \$c918 |            | Non esec. scroll                 |
| c8f6: | 20 00 cc | jsr * \$cc00 |            | Cursore una pos a sinistra       |
| c8f9: | 20 58 cb | jsr * \$cb58 |            | Pos. curs. rel. car. e col.      |
| c8fc: | 20 ed cb | jsr * \$cbcd |            | Curs. un passo a destra          |
| c8ff: | 20 32 cc | jsr * \$cc32 |            | Vis. carattere                   |
| c902: | 20 00 cc | jsr * \$cc00 |            | Cursore un passo a sin.          |
| c905: | a6 eb    | ldx \$eb     | [ tblx ]   | Carica su X att. linea cursore   |
| c907: | e4 df    | cpx \$df     | [ keytmp ] | Confronta con partenza           |
| c909: | d0 eb    | bne * \$c8f6 |            | Copia succ. carattere            |
| c90b: | c4 de    | cpy \$de     | [ keybnk ] | Confr. col. con col. di part.    |
| c90d: | d0 e7    | bne * \$c8f6 |            | Se no trovata continua           |
| c90f: | 20 27 cc | jsr * \$cc27 |            | Ins. SPAZIO a att. pos. curs.    |
| c912: | e6 f5    | inc \$f5     | [ insrt ]  | Incr. cont. per inserim.         |
| c914: | d0 02    | bne * \$c918 |            | Se div. da 0 salta               |
| c916: | c6 f5    | dec \$f5     | [ insrt ]  | Decr. di 1 per reset modo inser. |
| c918: | 4c 32 c9 | jmp * \$c932 |            | Reset vecchia pos. cursore       |
| c91b: | 20 75 c8 | jsr * \$c875 |            | Cursore a sinistra               |
| c91e: | 20 1e cc | jsr * \$cc1e |            | Copia coord. cursore             |
| c921: | b0 0f    | bcs * \$c932 |            | Non esec. cursore a sinistra     |
| c923: | c4 e7    | cpy \$e7     | [ scrt ]   | Confronto per bordo sin. fin.    |
| c925: | 90 16    | bcc * \$c93d |            | Bordo non trovato                |
| c927: | a6 eb    | ldx \$eb     | [ tblx ]   | Metti in X att. linea cursore    |
| c929: | e8       | inx          |            | Incrementa di 1 linea            |
| c92a: | 20 76 cb | jsr * \$cb76 |            | Controllo bit overflow           |
| c92d: | b0 0e    | bcs * \$c93d |            | Risulta linea seguente           |
| c92f: | 20 27 cc | jsr * \$cc27 |            | Caratt. SPAZIO in att. pos.      |
| c932: | a5 de    | lda \$de     | [ keybnk ] | Carica A colonna                 |
| c934: | 85 ec    | sta \$ec     | [ pntr ]   | Immag. att. pos. colonna         |
| c936: | a5 df    | lda \$df     | [ keytmp ] | Carica in A linea                |
| c938: | 85 eb    | sta \$eb     | [ tblx ]   | Scrivi att. linea cursore        |
| c93a: | 4c 5c c1 | jmp * \$c15c |            | Determina ind. iniz. linea       |
| c93d: | 20 ed cb | jsr * \$cbcd |            | Cursore un passo a destra        |

|                |              |          |                                  |
|----------------|--------------|----------|----------------------------------|
| c940: 20 58 cb | jsr * \$cb58 |          | Caratt. e colo. cursore          |
| c943: 20 00 cc | jsr * \$cc00 |          | Cursore un passo a sinistra      |
| c946: 20 32 cc | jsr * \$cc32 |          | Caratt. alla pos. del cursore    |
| c949: 20 ed cb | jsr * \$cbcd |          | Cursore indietro a destra        |
| c94c: 4c 23 c9 | jmp * \$c923 |          | Sposta linea a curs.             |
| c94f: a4 ec    | ldy \$ec     | [ pntr ] | Metti in Y att. col. curs.       |
| c951: c8       | iny          |          | Incr. punt. colonna              |
| c952: c4 e7    | cpy \$e7     | [ scrt ] | Confronta bordo destro finestra  |
| c954: b0 06    | bcs * \$c95c |          | Non ci sono piu' tabul. disp.    |
| c956: 20 6c c9 | jsr * \$c96c |          | Vai a succ. pos. tab.            |
| c959: f0 f6    | beq * \$c951 |          | Curs. e' a pos. tabulatore       |
| c95b: 2c a4 e7 | bit * \$e7a4 |          | Vai a \$C95E                     |
| c95e: 84 ec    | sty \$ec     | [ pntr ] | Metti in Y bordo destro finestra |
| c960: 60       | rts          |          |                                  |

-----  
ATTIVA/DISATTIVA TABULATORE

|                |              |            |                              |
|----------------|--------------|------------|------------------------------|
| c961: a4 ec    | ldy \$ec     | [ pntr ]   | Metti in Y col. att. cursore |
| c963: 20 6c c9 | jsr * \$c96c |            | Byte del tabulatore          |
| c966: 45 da    | eor \$da     | [ keysiz ] | Esegui rov. bit tabul.       |
| c968: 9d 54 03 | sta \$0354,x |            | Immagazzina in A             |
| c96b: 60       | rts          |            |                              |

-----  
DETERMINA POSIZIONE TABULATORE

|                |                |            |                                     |
|----------------|----------------|------------|-------------------------------------|
| c96c: 98       | tya            |            | Metti in A colonna                  |
| c96d: 29 07    | and #\$07      |            | Mask out bit 4-7                    |
| c96f: aa       | tax            |            | Bit prec in X come punt.            |
| c970: bd 6c ce | lda * \$ce6c,x |            | Carica in A potenza di 2            |
| c973: 85 da    | sta \$da       | [ keysiz ] | Immetti in \$DA                     |
| c975: 98       | tya            |            | Colonna di nuovo in A               |
| c976: 4a       | lsr a          |            | Questi tre passi servono a spostare |
| c977: 4a       | lsr a          |            | per 3 volte a destra A. Poi il val  |
| c978: 4a       | lsr a          |            | in INT                              |
| c979: aa       | tax            |            | Trasf. A in X come punt.            |
| c97a: bd 54 03 | lda \$0354,x   |            | Carica byte di tabulaz.             |
| c97d: 24 da    | bit \$da       | [ keysiz ] | Contr. fiss. ottavo tabul.          |
| c97f: 60       | rts            |            |                                     |

-----  
ESEGUI UN CLEAR SU TUTTI I TABULATORI

|                |              |  |                                       |
|----------------|--------------|--|---------------------------------------|
| c980: a9 00    | lda #\$00    |  | Carica A con 0 per op. di CLEAR       |
| c982: 2c a9 80 | bit \$80a9   |  | Viene eseg. un salto a \$C985. Viene  |
| c985: a2 09    | ldx #\$09    |  | controllato che ottava pos. e' un tab |
| c987: 9d 54 03 | sta \$0354,x |  | e tutte le pos. tab. sono rilev.      |
| c98a: ca       | dex          |  | Il contatore viene decr. di 1 e viene |
| c98b: 10 fa    | bpl * \$c987 |  | eseguito un salto se NN               |
| c98d: 60       | rts          |  |                                       |

-----  
ATTIVAZIONE BEEP

|                |              |            |                                      |
|----------------|--------------|------------|--------------------------------------|
| c98e: 24 f9    | bit \$f9     | [ beeper ] | Controllo flag acustico (BEEP)       |
| c990: 30 fb    | bmi * \$c98d |            | Nessun BEEP                          |
| c992: a9 15    | lda #\$15    |            | Metti a 15 (val. max) il volume del  |
| c994: 8d 18 d4 | sta * \$d418 |            | SID                                  |
| c997: a0 09    | ldy #\$09    |            | Carica costante di ATTACK/DECAY      |
| c999: a2 00    | ldx #\$00    |            | Carica X costante di SUSTAIN/RELEASE |
| c99b: 8c 05 d4 | sty * \$d405 |            | Immetti i valori nei corrispondenti  |
| c99e: 8e 06 d4 | stx * \$d406 |            | registri per voce 1                  |
| c9a1: a9 30    | lda #\$30    |            | Definisce byte HI di freq.           |

|                |              |                                    |
|----------------|--------------|------------------------------------|
| c9a3: 8d 01 d4 | sta * \$d401 | Immag. per voce 1                  |
| c9a6: a9 20    | lda #\$20    | Seleziona dente di sega            |
| c9a8: 8d 04 d4 | sta * \$d404 | Scrivi q.s. in SID                 |
| c9ab: a9 21    | lda #\$21    | Inizial. suono ottenuto mettendo a |
| c9ad: 8d 04 d4 | sta * \$d404 | uno il bit 0                       |
| c9b0: 60       | rts          |                                    |

-----  
CONTROLLO CURSORE

|                |               |          |                             |
|----------------|---------------|----------|-----------------------------|
| c9b1: a5 ec    | lda \$ec      | [ pntr ] | Metti in A att. col. curs.  |
| c9b3: 48       | pha           |          | Salva in A att. colonna     |
| c9b4: 20 c3 cb | jsr * \$cbcb3 |          | Ricerca fine linea          |
| c9b7: 20 63 c3 | jsr * \$c363  |          | Linefeed                    |
| c9ba: 68       | pla           |          | Rip. indietro att. colonna  |
| c9bb: 85 ec    | sta \$ec      | [ pntr ] | Immag. in A linea att. curs |
| c9bd: 60       | rts           |          |                             |

-----  
ESECUZIONE SEQUENZA DI ESCAPE

|                |                |                                  |
|----------------|----------------|----------------------------------|
| c9be: 6c 38 03 | jmp (\$0338)   | Vett. car. vis. con ESCAPE       |
| c9c1: c9 1b    | cmp #\$1b      | Contr. se caratt. e' ESC         |
| c9c3: d0 05    | bne * \$c9ca   | Salta se altro carattere         |
| c9c5: 46 ef    | lsr \$ef       | [ datax ]                        |
| c9c7: 4c 7d c7 | jmp * \$c77d   | Att. carat. molt. per 2          |
| c9ca: 29 7f    | and #\$7f      | Dis. tutte funz. speciali        |
| c9cc: 38       | sec            | Mask out bit 7                   |
| c9cd: e9 40    | sbc #\$40      | Fissa il carry per sottraz.      |
| c9cf: c9 1b    | cmp #\$1b      | Sottrai 64 da val. ASCII         |
| c9d1: b0 0a    | bcs * \$c9dd   | Confr. ris. con 27               |
| c9d3: 0a       | asl a          | Vai a fine se car. magg. di Z    |
| c9d4: aa       | tax            | Molt. val. A per 2               |
| c9d5: bd df c9 | lda * \$c9df,x | Metti ris. in X come punt.       |
| c9d8: 48       | pha            | Carica A byte HI per esec. rout. |
| c9d9: bd de c9 | lda * \$c9de,x | Salva su STACK                   |
| c9dc: 48       | pha            | Metti in STACK byte LO           |
| c9dd: 60       | rts            | Vai alla rout. ind. prec.        |

-----  
INDIRIZZI DELLE ROUTINE DI ESCAPE-----  
Ogni lettera o simbolo deve essere preceduta da ESCAPE

|             |        |                                     |
|-------------|--------|-------------------------------------|
| c9de: 9e ca | \$ca9e | @ - Clear del cursore a fine        |
| c9e0: ec ca | \$caec | A - Attiva modo di auto insert      |
| c9e2: 15 ca | \$ca15 | B - Fissa parte alta schermo        |
| c9e4: e9 ca | \$cae9 | C - Disabilita 1' auto insert       |
| c9e6: 51 ca | \$ca51 | D - Cancella linea attuale          |
| c9e8: 0a cb | \$cb0a | E - Disattivazione flash            |
| c9ea: 20 cb | \$cb20 | F - Attivazione flash               |
| c9ec: 36 cb | \$cb36 | G - Abilitazione BEEP               |
| c9ee: 39 cb | \$cb39 | H - Disabilitazione BEEP            |
| c9f0: 3c ca | \$ca3c | I - Inserisci una linea             |
| c9f2: b0 cb | \$cbb0 | J - Posiz. cursore a inizio linea   |
| c9f4: 51 cb | \$cb51 | K - Posiz. cursore a fine linea     |
| c9f6: el ca | \$cael | L - Abilita lo scrolling            |
| c9f8: e4 ca | \$cae4 | M - Disabilita lo scrolling         |
| c9fa: 47 cb | \$cb47 | N - Disattiva modo Reverse (80 col) |
| c9fc: 7c c7 | \$c77c | O - Modo apici, inst, RVS off       |
| c9ff: 8a ca | \$ca8a | P - Esegui un Clear a linea in.     |

|             |        |                                       |
|-------------|--------|---------------------------------------|
| ca00: 75 ca | \$ca75 | Q - Esegui un Clear a fine linea      |
| ca02: 3e cb | \$cb3e | R - Reverse di schermo in 80 col.     |
| ca04: f1 ca | \$caf1 | S - Block cursor                      |
| ca06: 13 ca | \$ca13 | T - Fissa il p. max di schermo        |
| ca08: fd ca | \$cafd | U - Sottolineamento cursore           |
| ca0a: bb ca | \$cabb | V - Attivazione scroll up             |
| ca0c: c9 ca | \$cac9 | W - Attivazione scroll down           |
| ca0e: 2b cd | \$cd2b | X - Passaggio 40/80 colonne           |
| ca10: 82 c9 | \$c982 | Y - Reset dei tabulatori (modo norm)  |
| ca12: 7f c9 | \$c97f | Z - Esegui un clear di tutti i tabul. |

## DEFINIZIONE DEI BORDI FINESTRA

|                |                      |                                      |
|----------------|----------------------|--------------------------------------|
| cal4: 18       | clc                  | La pos. del cursore e' max. sin.     |
| cal7: a6 ec    | ldx \$ec [ pntr ]    | Carica X con punt.                   |
| cal9: a5 eb    | lda \$eb [ tblx ]    | Carica A con pos. tab.               |
| calb: 90 11    | bcc * \$ca2e         | Contr. per clear di carry            |
| cald: 85 e4    | sta \$e4 [ scbot ]   | Definizione finestra                 |
| calf: 86 e7    | stx \$e7 [ scrt ]    | Definizione bordo destro             |
| ca21: 4c 32 ca | jmp * \$ca32         | Esegui resto della routine           |
| ca24: a5 ed    | lda \$ed [ lines ]   | Carica in A max. num. linee          |
| ca26: a6 ee    | ldx \$ee [ columns ] | Carica in X max. num. colonne        |
| ca28: 20 1d ca | jsr * \$cald         | Definisci punto max. destro          |
| ca2b: a9 00    | lda #\$00            | Punto sini. max = 0/0                |
| ca2d: aa       | tax                  | Trasf. A in X                        |
| ca2e: 85 e5    | sta \$e5 [ sctop ]   | Con queste due istruz. viene defini. |
| ca30: 86 e6    | stx \$e6 [ sclf ]    | il bordo sinistro (p. max)           |
| ca32: a9 00    | lda #\$00            | Metti 0 in A                         |
| ca34: a2 04    | ldx #\$04            | Metti 4 in X                         |
| ca36: 9d 5d 03 | sta \$035d,x         | Clear su bit di overflow line        |
| ca39: ca       | dex                  | Decrementa cont. di 1                |
| ca3a: d0 fa    | bne * \$ca36         | Salta se non es. clear su tutti bit  |
| ca3c: 60       | rts                  |                                      |

## INSERIMENTO LINEA

|                |                    |                                     |
|----------------|--------------------|-------------------------------------|
| ca3d: 20 7c c3 | jsr * \$c37c       | Sposta resto schermo a pos. X       |
| ca40: 20 56 c1 | jsr * \$c156       | Cursore sin. determ. ind. iniz.     |
| ca43: e8       | inx                | Incrementa linea                    |
| ca44: 20 76 cb | jsr * \$cb76       | Contr. bit di overflow-line         |
| ca47: 08       | php                | Salva il carry                      |
| ca48: 20 81 cb | jsr * \$cb81       | Controllo bit di overflow           |
| ca4b: 28       | plp                | Rprendi il carry da STACK           |
| ca4c: b0 03    | bcs * \$ca51       | Linea cursore = linea iniz.         |
| ca4e: 38       | sec                | Segna la vecchia o prec. linea come |
| ca4f: 66 e8    | ror \$e8 [ lsexp ] | linea seguente                      |
| ca51: 60       | rts                |                                     |

## CANCELLA LINEA ATTUALE

|                |                       |                                |
|----------------|-----------------------|--------------------------------|
| ca52: 20 b5 cb | jsr * \$cbb5          | Fissa ind. linea partenza      |
| ca55: a5 e5    | lda \$e5 [ sctop ]    | Carica in A max finestra       |
| ca57: 48       | pha                   | Salva sullo stack              |
| ca58: a5 eb    | lda \$eb [ tblx ]     | Carica in A att. linea cursore |
| ca5a: 85 e5    | sta \$e5 [ sctop ]    | Definiscilo come max finestra  |
| ca5c: a5 f8    | lda \$f8 [ locks+ 1 ] | Salva il flag di scroll        |
| ca5e: 48       | pha                   | Immetti in stack q.s.          |
| ca5f: a9 80    | lda #\$80             | Non eseguire lo scroll         |
| ca61: 85 f8    | sta \$f8 [ locks+ 1 ] | Salva il flag di scroll        |
| ca63: 20 b8 c3 | jsr * \$c3b8          | Abilita scroll in alto         |

|       |          |              |              |  |
|-------|----------|--------------|--------------|--|
| ca66: | 68       | pla          |              | Riprendi il flag di scroll             |
| ca67: | 85 f8    | sta \$f8     | [ locks+ 1 ] | Salva il flag di scroll                |
| ca69: | a5 e5    | lda \$e5     | [ sctop ]    | Carica in A max finestra               |
| ca6b: | 85 eb    | sta \$eb     | [ tblx ]     | Scrivi att. linea curs. in A           |
| ca6d: | 68       | pla          |              | Prendi max finestra                    |
| ca6e: | 85 e5    | sta \$e5     | [ sctop ]    | Riscrivi q.s.                          |
| ca70: | 38       | sec          |              | Fissa il carry                         |
| ca71: | 66 e8    | ror \$e8     | [ lsexp ]    | Segna come linea seguente              |
| ca73: | 4c 56 c1 | jmp * \$c156 |              | Poni il curs. bordo sin. finestra      |
| ca76: | 20 1e cc | jsr * \$cc1e |              | Salva coord. cursore                   |
| ca79: | 20 aa c4 | jsr * \$c4aa |              | Clear linea cursore                    |
| ca7c: | e6 eb    | inc \$eb     | [ tblx ]     | Incr. di 1 att. linea curs.            |
| ca7e: | 20 5c c1 | jsr * \$c15c |              | Determina ind. iniz. linea             |
| ca81: | a4 e6    | ldy \$e6     | [ sclf ]     | Carica in Y bordo sin. finestra        |
| ca83: | 20 74 cb | jsr * \$cb74 |              | Controllo bit di overflow-line         |
| ca86: | b0 f1    | bcs * \$ca79 |              | Esegui clear seg. linee                |
| ca88: | 4c 32 c9 | jmp * \$c932 |              | Fissa vecchio ind. cursore             |
| ca8b: | 20 1e cc | jsr * \$cc1e |              | Salva coord. cursore                   |
| ca8e: | 20 27 cc | jsr * \$cc27 |              | Metti SPAZIO a att.pos. curs.          |
| ca91: | c4 e6    | cpy \$e6     | [ sclf ]     | Confr. con bordo sin.finestra          |
| ca93: | d0 05    | bne * \$ca9a |              | Vai a ind. se non trovato              |
| ca95: | 20 74 cb | jsr * \$cb74 |              | Contr. bit overflow-line               |
| ca98: | 90 ee    | bcc * \$ca88 |              | Se nessun overflow fine                |
| ca9a: | 20 00 cc | jsr * \$cc00 |              | Cursore a sinistra                     |
| ca9d: | 90 ef    | bcc * \$ca8e |              | Clear linea                            |
| ca9f: | 20 1e cc | jsr * \$cc1e |              | Salva coord. cursore                   |
| caa2: | 20 aa c4 | jsr * \$c4aa |              | Cancella linea                         |
| caa5: | e6 eb    | inc \$eb     | [ tblx ]     | Incr. di 1 att. linea curs.            |
| caa7: | 20 5c c1 | jsr * \$c15c |              | Determ. ind. iniz. linea curs.         |
| caaa: | a4 e6    | ldy \$e6     | [ sclf ]     | Metti in Y bordo sin. finestra         |
| caac: | 20 74 cb | jsr * \$cb74 |              | Controlla bit di overflow-line         |
| caaf: | b0 f1    | bcs * \$caa2 |              | Se linea non trov. vai a ind.          |
| cabl: | a5 eb    | lda \$eb     | [ tblx ]     | Metti in A att. linea curs.            |
| cab3: | c5 e4    | cmp \$e4     | [ scbot ]    | Confronta bordo inf. finestra          |
| cab5: | 90 eb    | bcc * \$caa2 |              | Bordo inf. non trovato                 |
| cab7: | f0 e9    | beq * \$caa2 |              | Bordo inferiore trovato                |
| cab9: | 4c 32 c9 | jmp * \$c932 |              | Ripristina vecchio ind. curs.          |
| cabc: | 20 1e cc | jsr * \$cc1e |              | Salva corrd. cursore                   |
| cabf: | 8a       | txa          |              | Trasf. linea in A                      |
| cac0: | 48       | pha          |              | Salva linea su STACK                   |
| cacl: | 20 a6 c3 | jsr * \$c3a6 |              | Vai a scroll in alto                   |
| cac4: | 68       | pla          |              | Riprendi linea da stack                |
| cac5: | 85 df    | sta \$df     | [ keytmp ]   | Immagazzina in A                       |
| cac7: | 4c 32 c9 | jmp * \$c932 |              | Ripris. coord. vecchie del curs.       |
| caca: | 20 1e cc | jsr * \$cc1e |              | Salva coord. cursore                   |
| cadc: | 20 74 cb | jsr * \$cb74 |              | Controllo bit overflow-line            |
| cad0: | b0 03    | bcs * \$cad5 |              | Vai a ind. se non overflow             |
| cad2: | 38       | sec          |              | Segna. che linea di input non e' stata |
| cad3: | 66 e8    | ror \$e8     | [ lsexp ]    | rit. come linea di partenza            |
| cad5: | a5 e5    | lda \$e5     | [ sctop ]    | Metti in A max finestra                |
| cad7: | 85 eb    | sta \$eb     | [ tblx ]     | Immag. att. linea curs.                |
| cad9: | 20 7c c3 | jsr * \$c37c |              | Esegui un scroll in basso              |
| cadc: | 20 85 cb | jsr * \$cb85 |              | Esegui un clear su bit overf-line      |
| cadf: | 4c 32 c9 | jmp * \$c932 |              | Ripristina vecchie coordi. cursore     |
| cae2: | a9 00    | lda #\$00    |              | Abilit. funzione di scrolling          |
| cae4: | 2c       | .byte \$2c   |              | Salta a \$cae7                         |
| cae5: | a9 80    | lda #\$80    |              | Disabilita lo scrolling                |
| cae7: | 85 f8    | sta \$f8     | [ locks+ 1 ] | Immag. flag di scroll                  |
| cae9: | 60       | rts          |              |  |

## CONTROLLO AUTOINSERIMENTO

|                |              |            |                                     |
|----------------|--------------|------------|-------------------------------------|
| caea: a9 00    | lda #\$00    |            | Esegui clear su flag di auto-insert |
| caec: 2c a9 ff | bit * \$ffa9 |            | Vai a \$CAEF fis. flag sopra        |
| caef: 85 f6    | sta \$f6     | [ insflg ] | Immagaz. in A il flag               |
| cafl: 60       | rts          |            |                                     |

## ATTIVAZIONE BLOCCO CURSORE

|                |              |          |                                   |
|----------------|--------------|----------|-----------------------------------|
| caf2: 24 d7    | bit \$d7     | [ mode ] | Controllo modo 40/80 colonne      |
| caf4: 10 40    | bpl * \$cb36 |          | Se modo 40 colo. fine             |
| caf6: ad 2b 0a | lda \$0a2b   |          | Metti in A modo curs. da VDC      |
| caf9: 29 e0    | and #\$e0    |          | Mask out dei bits 0-4             |
| cafb: 4c 14 cb | jmp * \$cb14 |          | Cursor off                        |
| cafe: 24 d7    | bit \$d7     | [ mode ] | Controllo flag 40/80 colonne      |
| cb00: 10 34    | bpl * \$cb36 |          | Se 40 col. fine                   |
| cb02: ad 2b 0a | lda \$0a2b   |          | Metti in A modo curs. da VDC      |
| cb05: 29 e0    | and #\$e0    |          | Mask out bits 0-4                 |
| cb07: 09 07    | ora #\$07    |          | 7 e' la linea di inizio scansione |
| cb09: d0 09    | bne * \$cb14 |          | Salto inc. a ind.                 |
| cb0b: 24 d7    | bit \$d7     | [ mode ] | Controllo modo 40/80              |
| cb0d: 10 0b    | bpl * \$cb14 |          | Se 40 col. vai ad ind.            |
| cb0f: ad 2b 0a | lda \$0a2b   |          | metti in A modo cursore VDC       |
| cb12: 29 1f    | and #\$1f    |          | Mask out del bit di flash         |
| cb14: 8d 2b 0a | sta \$0a2b   |          | Salva su A                        |
| cb17: 4c 91 cd | jmp * \$cd91 |          | Metti in off modo e VIC           |
| cb1a: ad 26 0a | lda \$0a26   |          | Metti in A modo cursore del VIC   |
| cb1d: 09 40    | ora #\$40    |          | Fissa bit 6                       |
| cb1f: d0 12    | bne * \$cb33 |          | Salto incond.                     |
| cb21: 24 d7    | bit \$d7     | [ mode ] | Controllo modo 40/80 colonne      |
| cb23: 10 09    | bpl * \$cb2e |          | Se 40 colonne vai ad ind.         |
| cb25: ad 2b 0a | lda \$0a2b   |          | Carica in A modo curs. VDC        |
| cb28: 29 1f    | and #\$1f    |          | Mask out bit di flash             |
| cb2a: 09 60    | ora #\$60    |          | Definiz. per. puls. flash         |
| cb2c: d0 e6    | bne * \$cb14 |          | Salto inc. per immag.             |
| cb2e: ad 26 0a | lda \$0a26   |          | Metti modo curs. VIC              |
| cb31: 29 bf    | and #\$bf    |          | Mask out bit 6                    |
| cb33: 8d 26 0a | sta \$0a26   |          | Salva q.s.                        |
| cb36: 60       | rts          |          |                                   |

## ABILITA FLAG PER CAMPANELLO

|             |            |            |                              |
|-------------|------------|------------|------------------------------|
| cb37: a9 00 | lda #\$00  |            | Abilita segn acust.          |
| cb39: 2c    | .Byte \$2c |            | Vai a \$CB3C                 |
| cb3a: a9 80 | lda #\$80  |            | Disabilita campanello (BELL) |
| cb3c: 85 f9 | sta \$f9   | [ beeper ] | Immag. flag                  |
| cb3e: 60    | rts        |            |                              |

## 80 COLONNE IN REVERSE

|                |              |  |                               |
|----------------|--------------|--|-------------------------------|
| cb3f: a2 18    | ldx #\$18    |  | Sel. reg. 24                  |
| cb41: 20 da cd | jsr * \$cd4a |  | Prel. cont. attuale           |
| cb44: 09 40    | ora #\$40    |  | Fissa flag di REVERSE         |
| cb46: d0 07    | bne * \$cb4f |  | Salto inc. a ind.             |
| cb48: a2 18    | ldx #\$18    |  | Sel. registro 24              |
| cb4a: 20 da cd | jsr * \$cd4a |  | Prel. cont. attuale           |
| cb4d: 29 bf    | and #\$bf    |  | Clear su flag di RVS          |
| cb4f: 4c cc cd | jmp * \$cdcc |  | Immagazzina flag              |
| cb52: 20 c3 cb | jsr * \$cb33 |  | Determ.att. linea ini. indir. |
| cb55: 4c 3e c3 | jmp * \$c33e |  | Pos. curs. a fine linea       |



|       |       |              |            |                                |
|-------|-------|--------------|------------|--------------------------------|
| cb58: | a4 ec | ldy \$ec     | [ pntr ]   | Metti in Y col. att. curs.     |
| cb5a: | 24 d7 | bit \$d7     | [ mode ]   | Controllo modo 40/80           |
| cb5c: | 30 07 | bmi * \$cb65 |            | Salta a ind. se 80 col.        |
| cb5e: | b1 e2 | lda (\$e2),y | [ user ]   | Carica A con col. pos. cursore |
| cb60: | 85 f2 | sta \$f2     | [ tcolor ] | Salva su A                     |
| cb62: | b1 e0 | lda (\$e0),y | [ pnt ]    | Prendi cara. a pos. cursore    |
| cb64: | 60    | rts          |            |                                |

## CARATTERE E COLORE SOTTO IL CURSORE

|       |          |              |            |                          |
|-------|----------|--------------|------------|--------------------------|
| cb65: | 20 f9 cd | jsr * \$cdf9 |            | Fissa ind. aggiorn       |
| cb68: | 20 d8 cd | jsr * \$cdd8 |            | Prel. valore             |
| cb6b: | 85 f2    | sta \$f2     | [ tcolor ] | Immag. valore            |
| cb6d: | 20 e6 cd | jsr * \$cde6 |            | Fissa su video ind. agg. |
| cb70: | 20 d8 cd | jsr * \$cdd8 |            | Prel. car. da RAM video  |
| cb73: | 60       | rts          |            |                          |

## CONTROLLO LINEA DI OVERFLOW

|       |          |              |            |   |
|-------|----------|--------------|------------|---|
| cb74: | a6 eb    | ldx \$eb     | [ tblx ]   | Metti in X att. linea curs.             |
| cb76: | 20 9f cb | jsr * \$cb9f |            | Def. pot di 2                           |
| cb79: | 3d 5e 03 | and \$035e,x |            | Esegui un clear su bit di overflow-line |
| cb7c: | c9 01    | cmp #\$01    |            | Contr. per linea fissata                |
| cb7e: | 4c 90 cb | jmp * \$cb90 |            | Vai a fine rout.                        |
| cb81: | a6 eb    | ldx \$eb     | [ tblx ]   | Metti in X att. linea curs.             |
| cb83: | b0 0e    | bcs * \$cb93 |            | Salta ad ind. se flag fissato           |
| cb85: | 20 9f cb | jsr * \$cb9f |            | Defin. potenza di 2                     |
| cb88: | 49 ff    | eor #\$ff    |            | Esegui compl a 1 di A                   |
| cb8a: | 3d 5e 03 | and \$035e,x |            | Esegui un AND con tavola di overflow    |
| cb8d: | 9d 5e 03 | sta \$035e,x |            | Imm. in A                               |
| cb90: | a6 da    | ldx \$da     | [ keysiz ] | Metti in X per immag. tempor.           |
| cb92: | 60       | rts          |            |   |

## FISSA IL BIT DELLA LINEA DI OVERFLOW

|       |          |                |              |                                       |
|-------|----------|----------------|--------------|---------------------------------------|
| cb93: | 24 f8    | bit \$f8       | [ locks+ 1 ] | Controllo bit di scroll               |
| cb95: | 70 df    | bvs * \$cb76   |              | Salta se bit 6 fissato                |
| cb97: | 20 9f cb | jsr * \$cb9f   |              | Determ. potenza di 2                  |
| cb9a: | 1d 5e 03 | ora \$035e,x   |              | Fissa bit di overflow-line            |
| cb9d: | d0 ee    | bne * \$cb8d   |              | Aggiornalo                            |
| cb9f: | 86 da    | stx \$da       | [ keysiz ]   | Salva A                               |
| cbal: | 8a       | txa            |              | Trasf. X in A                         |
| cba2: | 29 07    | and #\$07      |              | Mask out bits 3-7                     |
| cba4: | aa       | tax            |              | A trasf. in X                         |
| cba5: | bd 6c ce | lda * \$ce6c,x |              | Prendi corr. potenza di 2             |
| cba8: | 48       | pha            |              | Salva A su stack                      |
| cba9: | a5 da    | lda \$da       | [ keysiz ]   | Riprist. valore orig.                 |
| cbab: | 4a       | lsr a          |              | A questo punto il valore viene        |
| cbac: | 4a       | lsr a          |              | diviso per 2 tre volte ed i risultati |
| cbad: | 4a       | lsr a          |              | posti in INT                          |
| cbae: | aa       | tax            |              | Risultati in X                        |
| cbaf: | 68       | pla            |              | Preleva potenza di 2 da Stack         |
| cbb0: | 60       | rts            |              |                                       |

## ESECUZIONE DI UN CLEAR DI OVERFLOW

|       |          |              |          |  |
|-------|----------|--------------|----------|--|
| cbb1: | a4 e6    | ldy \$e6     | [ sclf ] | Metti in Y bordo sin.finestra          |
| cbb3: | 84 ec    | sty \$ec     | [ pntr ] | Salva att. colo. cursore               |
| cbb5: | 20 74 cb | jsr * \$cb74 |          | Clear bit di ovefl-line di linea curs. |

|                |               |          |                                      |
|----------------|---------------|----------|--------------------------------------|
| cbb8: 90 06    | bcc * \$cbcb0 |          | Il carry a 0, se tutti bits=0        |
| cbba: c6 eb    | dec \$eb      | [ tblx ] | Decr. l att. linea cursore           |
| cbbc: 10 f7    | bpl * \$cbb5  |          | Se non e' la prima linea salta ind.  |
| cbbe: e6 eb    | inc \$eb      | [ tblx ] | Incr. att. linea curs.               |
| cbc0: 4c 5c cl | jmp * \$cl5c  |          | Ricerca ind.iniz.linea att.          |
| cbc3: e6 eb    | inc \$eb      | [ tblx ] | Incr. att. linea curs.               |
| cbc5: 20 74 cb | jsr * \$cb74  |          | Esegui clear su bit di overflow-line |
| cbc8: b0 f9    | bcs * \$cbcb3 |          | Se non e' ultima linea salta ad ind. |
| cbca: c6 eb    | dec \$eb      | [ tblx ] | Decr. att. linea curs.               |
| cbcc: 20 5c cl | jsr * \$cl5c  |          | Ricerca ind.iniz.linea att.          |
| cbcf: a4 e7    | ldy \$e7      | [ scrt ] | Carica Y bordo destro finestra       |
| cbd1: 84 ec    | sty \$ec      | [ pntr ] | Salva att. col. cursore              |
| cbd3: 20 58 cb | jsr * \$cb58  |          | Prendi car. e col. pos. cursore      |
| cbd6: a6 eb    | ldx \$eb      | [ tblx ] | Carica X con linea att. curs.        |
| cbd8: c9 20    | cmp #\$20     |          | Contr. se e' caratt. SPAZIO          |
| cbda: d0 0e    | bne * \$cbea  |          | Se neg. vai ad ind.                  |
| cbdc: c4 e6    | cpy \$e6      | [ sclf ] | Confronta con bordo sin.             |
| cbde: d0 05    | bne * \$cbe5  |          | Non trovato                          |
| cbe0: 20 74 cb | jsr * \$cb74  |          | Esegui clear su bit di ovfl-line     |
| cbe3: 90 05    | bcc * \$cbea  |          | Linea ancora libera                  |
| cbe5: 20 00 cc | jsr * \$cc00  |          | Pos. cursore un passo a destra       |
| cbe8: 90 e9    | bcc * \$cbd3  |          | Abil. mov. cursore                   |
| cbea: 84 ea    | sty \$ea      | [ indx ] | Fine att. linea di input             |
| cbec: 60       | rts           |          |                                      |

## SPOSTAMENTO DEL CURSORE 1 A DESTRA (SULLA FINESTRA)

|                |              |          |                                    |
|----------------|--------------|----------|------------------------------------|
| cbcd: 48       | pha          |          | Salva A in Stack                   |
| cbce: a4 ec    | ldy \$ec     | [ pntr ] | Metti in Y attuale linea cursore   |
| cbf0: c4 e7    | cpy \$e7     | [ scrt ] | Confronta con bordo des.finestra   |
| cbf2: 90 07    | bcc * \$cbfb |          | Contr. ric. bordo destro fin.      |
| cbf4: 20 63 c3 | jsr * \$c363 |          | Se neg.increm. curs. colonna       |
| cbf7: a4 e6    | ldy \$e6     | [ sclf ] | Metti in Y bordo sinistro finestra |
| cbf9: 88       | dex          |          | Decrementa di 1 Y                  |
| cbfa: 38       | sec          |          | Fissa carry per nuova linea        |
| cbfb: c8       | iny          |          | Incrementa di 1 colon. cursore     |
| cbfc: 84 ec    | sty \$ec     | [ pntr ] | Immag. att. colonna cursore        |
| cbfe: 68       | pla          |          | Rimetti A in Stack                 |
| cbff: 60       | rts          |          |                                    |

## SPOSTAMENTO DEL CURSORE 1 A SINISTRA (SULLA FINESTRA)

|                |              |           |                                    |
|----------------|--------------|-----------|------------------------------------|
| cc00: a4 ec    | ldy \$ec     | [ pntr ]  | Metti in Y att. col. cursore       |
| cc02: 88       | dex          |           | Decrementa col. di 1               |
| cc03: 30 04    | bmi * \$cc09 |           | Se risul. prec. neg.curs. in col.0 |
| cc05: c4 e6    | cpy \$e6     | [ sclf ]  | Confronta con bordo sin.finestra   |
| cc07: b0 0f    | bcs * \$cc18 |           | Se non ragg. punto max. OK         |
| cc09: a4 e5    | ldy \$e5     | [ sctop ] | Carica max finestra in Y           |
| cc0b: c4 eb    | cpy \$eb     | [ tblx ]  | Confronta con att. linea curs.     |
| cc0d: b0 0e    | bcs * \$cc1d |           | Contr.curs.sul max. fine           |
| cc0f: c6 eb    | dec \$eb     | [ tblx ]  | Decrem. att. linea curs.           |
| cc11: 48       | pha          |           | Salva A su Stack                   |
| cc12: 20 5c cl | jsr * \$cl5c |           | Cerca ind. part. linea             |
| cc15: 68       | pla          |           | Riprist. A da Stack                |
| cc16: a4 e7    | ldy \$e7     | [ scrt ]  | Metti in Y bordo destro fin.       |
| cc18: 84 ec    | sty \$ec     | [ pntr ]  | Salva att. pos. curs. colonna      |
| cc1a: c4 e7    | cpy \$e7     | [ scrt ]  | Confronto con bordo dest. fin      |
| cc1c: 18       | clc          |           | Clear di carry per mov. curs.      |
| cc1d: 60       | rts          |           |                                    |

## COPIA DI SERVIZIO PER IL CURSORE

|       |       |          |            |                               |
|-------|-------|----------|------------|-------------------------------|
| ccle: | a4 ec | ldy \$ec | [ pntr ]   | Metti in Y att. colonna curs. |
| cc20: | 84 de | sty \$de | [ keybnk ] | Copia q.s. in \$de            |
| cc22: | a6 eb | ldx \$eb | [ tblx ]   | Metti in X att. linea curs.   |
| cc24: | 86 df | stx \$df | [ keytmp ] | Copia ris. in \$df            |
| cc26: | 60    | rts      |            |                               |

## IMMETTI 1 SPAZIO IN POS. CURSORE

|       |          |              |           |                                     |
|-------|----------|--------------|-----------|-------------------------------------|
| cc27: | a5 f1    | lda \$f1     | [ color ] | Metti in A cod.col.per car. da vis. |
| cc29: | 29 8f    | and #\$8f    |           | Mask out bits 4-6                   |
| cc2b: | aa       | tax          |           | Trasf. q.s. in X                    |
| cc2c: | a9 20    | lda #\$20    |           | Carica A con SPAZIO                 |
| cc2e: | 2c       | .byte \$2c   |           | Vai a \$cc31                        |
| cc2f: | a6 f1    | ldx *f1      |           | Carica registro X con colore        |
| cc31: | 2c       | .byte \$2c   |           | Vai a \$cc34                        |
| cc32: | a6 f2    | ldx *f2      |           | Carica codice colore per insert/del |
| cc34: | a8       | tay          |           | Trasf A in Y                        |
| cc35: | a9 02    | lda #\$02    |           | Metti in A valore 2                 |
| cc37: | 8d 28 0a | sta \$0a28   |           | Contr. VIC per FLASH                |
| cc3a: | 20 7c c1 | jsr * \$c17c |           | Agg. indirizzo                      |
| cc3d: | 98       | tya          |           | Trasf Y in A                        |
| cc3e: | a4 ec    | ldy \$ec     | [ pntr ]  | Metti in Y att. col. curs.          |
| cc40: | 24 d7    | bit \$d7     | [ mode ]  | Contr. modo 40/80 colonne           |
| cc42: | 30 06    | bmi * \$cc4a |           | Se 80 col. salta a ind.             |
| cc44: | 91 e0    | sta (\$e0),y | [ pnt ]   | Immag. caratt. in 40 colonne        |
| cc46: | 8a       | txa          |           | Trasferisci RAM (video) e reg. X    |
| cc47: | 91 e2    | sta (\$e2),y | [ user ]  | per colore in mem. colore           |
| cc49: | 60       | rts          |           |                                     |

## CARATTERE SU SCHERMO A 80 COLONNE

|       |          |              |             |                                  |
|-------|----------|--------------|-------------|----------------------------------|
| cc4a: | 48       | pha          |             | Salva A su Stack                 |
| cc4b: | 8a       | txa          |             | Trasf reg. X (colore) in A       |
| cc4c: | 48       | pha          |             | Immag. su stack                  |
| cc4d: | 20 f9 cd | jsr * \$cdf9 |             | Fissa reg. aggiornato            |
| cc50: | 68       | pla          |             | Il colore da Stack in A          |
| cc51: | 20 ca cd | jsr * \$cdca |             | Immag. q.s. in RAM               |
| cc54: | 20 e6 cd | jsr * \$cde6 |             | Fissa ind. agg. per RAM video    |
| cc57: | 68       | pla          |             | Riprendi car. da Stack           |
| cc58: | 4c ca cd | jmp * \$cdca |             | Immag. car. in RAM video         |
| cc5b: | 38       | sec          |             | Fissa il carry                   |
| cc5c: | a5 e4    | lda \$e4     | [ scbot ]   | Metti in A min finestra          |
| cc5e: | e5 e5    | sbc \$e5     | [ sctop ]   | Sottrai max. linea               |
| cc60: | a8       | tay          |             | Trasf in reg. Y                  |
| cc61: | 38       | sec          |             | Fissa il carry                   |
| cc62: | a5 e7    | lda \$e7     | [ scrt ]    | Metti in A bordo destro finestra |
| cc64: | e5 e6    | sbc \$e6     | [ sclf ]    | Sottrai bordo si.fin. prec.      |
| cc66: | aa       | tax          |             | Trasf. Xn.car. per linea         |
| cc67: | a5 ee    | lda \$ee     | [ columns ] | Metti in A n. colonne            |
| cc69: | 60       | rts          |             |                                  |

## POSIZIONAMENTO CURSORE

|       |       |              |           |                                  |
|-------|-------|--------------|-----------|----------------------------------|
| cc6a: | b0 29 | bcs * \$cc95 |           | Contr. per carry fissato         |
| cc6c: | 8a    | txa          |           | Trasf. linea in A                |
| cc6d: | 65 e5 | adc \$e5     | [ sctop ] | Aggiungi max. finestra           |
| cc6f: | b0 14 | bcs * \$cc85 |           | Contr. per overflow.Pos. termina |
| cc71: | c5 e4 | cmp \$e4     | [ scbot ] | Confr. con min. finestra         |

|             |                   |                                     |
|-------------|-------------------|-------------------------------------|
| cc73: f0 02 | beq * \$cc77      | Se rilev. tutto bene                |
| cc75: b0 0e | bcs * \$cc85      | Contr. per overflow.Pos. termina    |
| cc77: 48    | pha               | Salva n. linea su STACK             |
| cc78: 18    | clc               | Esegui Clear del Carry per somma    |
| cc79: 98    | tya               | Immetti col. in A.                  |
| cc7a: 65 e6 | adc \$e6 [ sclf ] | Sommaci bordo sin. finestra         |
| cc7c: b0 06 | bcs * \$cc84      | Conr. per overflow. Se pos. termina |
| cc7e: c5 e7 | cmp \$e7 [ sscr ] | Confronta con bordo destro fin.     |
| cc80: f0 04 | beq * \$cc86      | Va bene se uguale                   |
| cc82: 90 02 | bcc * \$cc86      | Se invece overflow errore e fine    |
| cc84: 68    | pla               | Carica linea da Stack               |
| cc85: 60    | rts               |                                     |

-----  
ESEGUI UN CLEAR SULLA LINEA DI INPUT

|                |                    |                               |
|----------------|--------------------|-------------------------------|
| cc86: 85 ec    | sta \$ec [ pntr ]  | Immag. att. colonna curs.     |
| cc88: 85 e9    | sta \$e9 [ lstp ]  | Immag. linea iniz. input      |
| cc8a: 68       | pla                | Prendi linea da Stack         |
| cc8b: 85 eb    | sta \$eb [ tblx ]  | Riscrivi att. linea curs.     |
| cc8d: 85 e8    | sta \$e8 [ lsxp ]  | Immag. come linea iniz. input |
| cc8f: 20 5c cl | jsr * \$cl5c       | Determ. ind. att. linea       |
| cc92: 20 57 cd | jsr * \$cd57       | Fissa curs. att. linea        |
| cc95: a5 eb    | lda \$eb [ tblx ]  | Immetti A att. lienea curs.   |
| cc97: e5 e5    | sbc \$e5 [ sctop ] | Sottrai max. finestra         |
| cc99: aa       | tax                | Trasf. risult. in X           |
| cc9a: 38       | sec                | Fissa Carry per sottraz.      |
| cc9b: a5 ec    | lda \$ec [ pntr ]  | Metti in A att. col. curs.    |
| cc9d: e5 e6    | sbc \$e6 [ sclf ]  | Sottrai bordo sin. finestra   |
| cc9f: a8       | tay                | Meti risult. in Y.            |
| cca0: 18       | clc                | Esegui un Clear del Carry     |
| ccal: 60       | rts                |                               |

-----  
INGRESSO ROUTINE KERNAL PFKEY

|                |                         |                                       |
|----------------|-------------------------|---------------------------------------|
| cca2: ca       | dex                     | Decr. numero tasto funz.              |
| cca3: 86 dc    | stx \$dc [ keynum ]     | Immetti n. prec. in Pag. Zero         |
| cca5: 84 da    | sty \$da [ keysiz ]     | Immagg. lung. stringa in Pag. Zero    |
| cca7: 8d aa 02 | sta \$02aa              | Immag. q.s. in FETVEC                 |
| ccaa: a8       | tay                     | Trasf. punt.ind.stringa in Y          |
| ccab: b6 02    | ldx \$02,y [ bank ]     | Imm. n.banco stringa funz. in X       |
| ccad: 20 6b ff | jsr * \$ff6b [ getcgr ] | Vai a subr. GETCFG                    |
| ccb0: 85 de    | sta \$de [ keybnk ]     | Immag.in banco mem.byte per str.funz. |
| ccb2: a2 0a    | ldx #\$0a               | Metti n. tasti funz. in A             |
| ccb4: 20 20 cd | jsr * \$cd20            | Aggiorna lung.str.funzione            |
| ccb7: 85 db    | sta \$db [ keylen ]     | Immag.in Pag. 0lung. stringa          |
| ccb9: a6 dc    | ldx \$dc [ keynum ]     | Carica n. tasto funzione              |
| ccbb: e8       | inx                     | Crea n. reale per tasto funz.         |
| ccbc: 20 20 cd | jsr * \$cd20            | Aggiorna lung.str.funzione            |
| ccbf: 85 dd    | sta \$dd [ keynxt ]     | Immag. lunghezza stringa              |
| cccl: a6 dc    | ldx \$dc [ keynum ]     | Carica n. tasto funzione              |
| ccc3: a5 da    | lda \$da [ keysiz ]     | Carica lung.stringa tasto funz.       |
| ccc5: 38       | sec                     | Fissa il carry per sottraz.           |
| ccc6: fd 00 10 | sbc \$1000,x            | Sottrai lung. vecchia funz. stringa   |
| ccc9: f0 2b    | beq * \$ccf6            | Ness. mov necess. Cont.               |
| cccb: 90 16    | bcc * \$cce3            | Contr. nuova str. minore prec.        |
| cccd: 18       | clc                     | Clear Carry per somma                 |
| ccce: 65 db    | adc \$db [ keylen ]     | Somma:lung.tot. e differ. con vecch.  |
| ccd0: b0 4d    | bcs * \$cd1f            | Se lung. >256 errore                  |
| ccd2: aa       | tax                     | Immetti nuova lung. max. in X         |
| ccd3: a4 db    | ldy \$db [ keylen ]     | Immetti vecchio val.(max.lung) in Y   |

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| ccd5: c4 dd    | cpy \$dd     | [ keynxt ] | Contr. se entrambi val. prec. uguali  |
| ccd7: f0 ld    | beq * \$ccf6 |            | Se pos. ind. ultimo tasto funz.       |
| ccd9: 88       | dey          |            | Decr. vecchia lung.(max) di 1         |
| ccda: ca       | dex          |            | Increment. nuova lung.(max) di 1      |
| ccdb: b9 0a 10 | lda \$100a,y |            | Sposta stringa funz. dalla nuova pos. |
| ccde: 9d 0a 10 | sta \$100a,x |            | di inserimento                        |
| ccel: b0 f2    | bcs * \$ccd5 |            | Riserva spazio per nuova str.         |
| cce3: 65 dd    | adc \$dd     | [ keynxt ] | Somma la diff. lunghezze              |
| cce5: aa       | tax          |            | Metti nuova lung. in X                |
| cce6: a4 dd    | ldy \$dd     | [ keynxt ] | Immetti vecchia lung. in Y            |
| cce8: c4 db    | cpy \$db     | [ keylen ] | Confronta con vecchio max             |
| ccea: b0 0a    | bcs * \$ccf6 |            | Se confr. = allora spazio             |
| ccec: b9 0a 10 | lda \$100a,y |            | Inser. per nuova str. funzione        |
| ccef: 9d 0a 10 | sta \$100a,x |            | Esegui. funzione del tasto            |
| ccf2: c8       | iny          |            | Increment. vecchia e nuova lunghezza  |
| ccf3: e8       | inx          |            | di 1 bit per mov.                     |
| ccf4: 90 f2    | bcc * \$cce8 |            | Contr. spostam. funz. stringa         |
| ccf6: a6 dc    | ldx \$dc     | [ keynum ] | Carica n. tasto funz.                 |
| ccf8: 20 20 cd | jsr * \$cd20 |            | Somma lung. funz. stringa             |
| ccfb: aa       | tax          |            | Trasf in X nuovo tasto funz.          |
| ccfc: a4 dc    | ldy \$dc     | [ keynum ] | Carica n. tasto funzione              |
| ccfe: a5 da    | lda \$da     | [ keysiz ] | Carica lung. funz. stringa per inser. |
| cd00: 99 00 10 | sta \$1000,y |            | Sost. lungh. ingr.in tav.funz.stringa |
| cd03: a0 00    | ldy #\$00    |            | Iniz. spostamento puntatori           |
| cd05: c6 da    | dec \$da     | [ keysiz ] | Decr. di 1 lung. funz. stringa        |
| cd07: 30 15    | bmi * \$cdle |            | Trasf. tutti car. nella tavola.Esci   |
| cd09: 86 df    | stx \$df     | [ keytmp ] | Immag.lung. stringa                   |
| cd0b: a6 de    | ldx \$de     | [ keybnk ] | N. banco dove e' stringa              |
| cd0d: ad aa 02 | lda \$02aa   |            | Carica A con FETVEC                   |
| cd10: 78       | sei          |            | Disab. tutti interrupt                |
| cd11: 20 a2 02 | jsr \$02a2   |            | Carica carattere (funz.str)           |
| cd14: 58       | cli          |            | Abilita tutti interrupt               |
| cd15: a6 df    | ldx \$df     | [ keytmp ] | Carica in tavola pos.funz.str.        |
| cd17: 9d 0a 10 | sta \$100a,x |            | Inserisci cara.                       |
| cd1a: e8       | inx          |            | Incr. buffer str. di 1                |
| cd1b: c8       | iny          |            | Incr. buffer str. di 1                |
| cd1c: d0 e7    | bne * \$cd05 |            | Vai al ciclo trasf.stringa            |
| cd1e: 18       | clc          |            | Clear carry per Mark.                 |
| cd1f: 60       | rts          |            |                                       |

## CONTINUAZIONE PREC. PER LUNGHEZZA FUNZIONE STRINGA

|                |              |  |                              |
|----------------|--------------|--|------------------------------|
| cd20: a9 00    | lda #\$00    |  | Metti 0 nel contatore        |
| cd22: 18       | clc          |  | Clear di carry per somma     |
| cd23: ca       | dex          |  | Decr. val. prec. tasto       |
| cd24: 30 05    | bmi * \$cd2b |  | Se 0 aggiungi lungh.tot.     |
| cd26: 7d 00 10 | adc \$1000,x |  | Agg. lung. tasto X           |
| cd29: 90 f8    | bcc * \$cd23 |  | Esegui un salto inc. a subr. |
| cd2b: 60       | rts          |  |                              |

## ROUTINE KERNAL SWAPPER

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| cd2c: 85 f0    | sta \$f0     | [ 1stchr ] | Immag. in A ult.car.stamp.            |
| cd2e: a2 1a    | ldx #\$1a    |            | Scambia tipo monitor                  |
| cd30: bc 40 0a | ldy \$0a40,x |            | Immag. schermo att.                   |
| cd33: b5 e0    | lda \$e0,x   | [ pnt ]    | Rout. di esecuzione funz. prec. per   |
| cd35: 9d 40 0a | sta \$0a40,x |            | 26 volte pari al n.bytes da copiare   |
| cd38: 98       | tya          |            | Immag. interv. passivo da loc. \$0A40 |
| cd39: 95 e0    | sta \$e0,x   | [ pnt ]    | a \$0A5B                              |
| cd3b: ca       | dex          |            | Decrem. contatore                     |

|                |              |          |  |
|----------------|--------------|----------|--|
| cd3c: 10 f2    | bpl * \$cd30 |          | Contr. se niente e' cambiato           |
| cd3e: a2 0d    | ldx #\$0d    |          | Routine per l' esecuzione del cambia   |
| cd40: bc 60 0a | ldy \$0a60,x |          | fra schermo attivo e passivo. Riguarda |
| cd43: bd 54 03 | lda \$0354,x |          | sia bita map che tavole                |
| cd46: 9d 60 0a | sta \$0a60,x |          | L' area pass. inizia a \$0A60          |
| cd49: 98       | tya          |          | Esegui per 13 volte                    |
| cd4a: 9d 54 03 | sta \$0354,x |          | Continua                               |
| cd4d: ca       | dex          |          | Decrem. contatore e salta a \$CD40     |
| cd4e: 10 f0    | bpl * \$cd40 |          | se copia non eff.                      |
| cd50: a5 d7    | lda \$d7     | [ mode ] | Carica stato schermo                   |
| cd52: 49 80    | eor #\$80    |          | Inverti il bit Flag                    |
| cd54: 85 d7    | sta \$d7     | [ mode ] | Immag. di nuovo                        |
| cd56: 60       | rts          |          |  |

## ROUTINE DI POSIZIONAMENTO CURSORE

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| cd57: 24 d7    | bit \$d7     | [ mode ]   | Contr. modo 40/80 colonne             |
| cd59: 10 fb    | bpl * \$cd56 |            | Se 40 col. vai a fine                 |
| cd5b: a2 0e    | ldx #\$0e    |            | Posizione alta del cursore in X       |
| cd5d: 18       | clc          |            | Clear del carry                       |
| cd5e: a5 e0    | lda \$e0     | [ pnt ]    | Metti in A byte basso att.linea sche. |
| cd60: 65 ec    | adc \$ec     | [ pntr ]   | Sommaci col. curs.                    |
| cd62: 48       | pha          |            | Metti in A byte basso                 |
| cd63: a5 e1    | lda \$e1     | [ pnt+ 1 ] | Byte alto att. linea schermo          |
| cd65: 69 00    | adc #\$00    |            | Sommaci il Carry                      |
| cd67: 20 cc cd | jsr * \$cdcc |            | Immag.byte alto                       |
| cd6a: e8       | inx          |            | Increment punt. reg.                  |
| cd6b: 68       | pla          |            | Prendi byte basso da Stack            |
| cd6c: 4c cc cd | jmp * \$cdcc |            | Salva q.s. ancora                     |
| cd6f: 24 d7    | bit \$d7     | [ mode ]   | Contr. per modo 40/80 col.            |
| cd71: 10 26    | bpl * \$cd99 |            | Salta se siamo a 40 col.              |
| cd73: 20 7c cl | jsr * \$cl7c |            | Fissa indirizzi                       |
| cd76: a4 ec    | ldy \$ec     | [ pntr ]   | Metti in Y att. col. curs.            |
| cd78: 20 f9 cd | jsr * \$cdf9 |            | Agg. reg. con ind.                    |
| cd7b: 20 d8 cd | jsr * \$cdd8 |            | Continua e completa                   |
| cd7e: 8d 33 0a | sta \$0a33   |            | Immag. temporaneo                     |
| cd81: 29 f0    | and #\$f0    |            | Mask out per bit 0-3                  |
| cd83: 85 db    | sta \$db     | [ keylen ] | Immag. lunghezza                      |
| cd85: 20 f9 cd | jsr * \$cdf9 |            | Agg. reg. con ind.                    |
| cd88: a5 f1    | lda \$f1     | [ color ]  | Metti in A cod. colore caratt.        |
| cd8a: 29 0f    | and #\$0f    |            | Mask out bits 4-7                     |
| cd8c: 05 db    | ora \$db     | [ keylen ] | Esegui un OR con A                    |
| cd8e: 20 ca cd | jsr * \$cdca |            | Immag. ind.                           |
| cd91: a2 0a    | ldx \$0a     |            | Car. modo curs. e linea ini. scans.   |
| cd93: ad 2b 0a | lda \$0a2b   |            | Modo curs. 80 col.                    |
| cd96: 4c cc cd | jmp * \$cdcc |            | Immagazzina q.s.                      |
| cd99: a9 00    | lda #\$00    |            | Metti 0 in A                          |
| cd9b: 8d 27 0a | sta \$0a27   |            | Metti in off il cursore               |
| cd9e: 60       | rts          |            |                                       |

## ATTIVA CURSORE IN MODO 80 COLONNE

|                |              |          |                                  |
|----------------|--------------|----------|----------------------------------|
| cd9f: 24 d7    | bit \$d7     | [ mode ] | Controllo modo 40/80 colonne     |
| cdal: 10 10    | bpl * \$cdb3 |          | Salta se siamo a 40 col.         |
| cda3: 20 f9 cd | jsr * \$cdf9 |          | Agg.. reg. con ind.              |
| cda6: ad 33 0a | lda \$0a33   |          | Immag. temp. per MOVLIN          |
| cda9: 20 ca cd | jsr * \$cdca |          | Immag. indir. di cui sopra       |
| cdac: a2 0a    | ldx \$0a     |          | Modo curs. e linea inizio scans. |
| cdae: a9 20    | lda \$20     |          | Metti in A val. 32               |
| cdb0: 4c cc cd | jmp * \$cdcc |          | Imm. A in reg. dati VDC          |

|                |              |                                 |
|----------------|--------------|---------------------------------|
| cdb3: 8d 27 0a | sta \$0a27   | Curs. VIC attivo                |
| cdb6: ad 26 0a | lda \$0a26   | Contr. curs. fisso o lamp.      |
| cdb9: 10 0e    | bpl * \$cdc9 | Se fisso vai a fine             |
| cdbb: 29 40    | and #\$40    | Clear il flag modo flash        |
| cdbd: 8d 26 0a | sta \$0a26   | Immagazzina in A                |
| cdc0: ad 29 0a | lda \$0a29   | Carica car. VIC prima del Flash |
| cdc3: ae 2a 0a | ldx \$0a2a   | Carica col. VIC " " "           |
| cdc6: 20 34 cc | jsr * \$cc34 | Fissa vecchi valori             |
| cdc9: 60       | rts          |                                 |

-----  
ATTIVAZIONE REGISTRI VCR

|                |              |                         |
|----------------|--------------|-------------------------|
| cdca: a2 1f    | ldx #\$1f    | Carica reg dati VCR     |
| cdcc: 8e 00 d6 | stx * \$d600 | Tras. registri          |
| cdcf: 2c 00 d6 | bit * \$d600 | Controllo di status     |
| cdd2: 10 fb    | bpl * \$cdcf | Se non eseguito attendi |
| cdd4: 8d 01 d6 | sta * \$d601 | Immag. valore nel reg.  |
| cdd7: 60       | rts          |                         |

-----  
LETTURA VALORI DI VCR

|                |              |                                   |
|----------------|--------------|-----------------------------------|
| cdd8: a2 1f    | ldx #\$1f    | Questa routine esegue esattamente |
| cdca: 8e 00 d6 | stx * \$d600 | quanto sopra prendendo il valore  |
| cddd: 2c 00 d6 | bit * \$d600 | dall' A invece di caricarlo       |
| cde0: 10 fb    | bpl * \$cddd |                                   |
| cde2: ad 01 d6 | lda * \$d601 |                                   |
| cde5: 60       | rts          |                                   |

|                |                         |                                       |
|----------------|-------------------------|---------------------------------------|
| cde6: a2 12    | ldx #\$12               | Aggiorna indirizzo HI                 |
| cde8: 18       | clc                     | Esegui clear di carry per somma       |
| cde9: 98       | tya                     | Trasf. Y in A                         |
| cdea: 65 e0    | adc \$e0 [ pnt ]        | Somma byte LO a ind. att.             |
| cdec: 48       | pha                     | Trasferiscilo su Stack                |
| cded: a9 00    | lda #\$00               | Carica A con zero                     |
| cdef: 65 e1    | adc \$e1 [ pnt+ 1 ]     | Aggiungi il carry                     |
| cdf1: 20 cc cd | jsr * \$cdcc            | Immagazzina byte HI                   |
| cdf4: 68       | pla                     | Byte LO da Stack                      |
| cdf5: e8       | inx                     | Incrementa registro a \$13            |
| cdf6: 4c cc cd | jmp * \$cdcc            | Byte LO in registro aggiornato        |
| cdf9: a2 12    | ldx #\$12               | Aggiorna X con byte HI                |
| cdfb: 18       | clc                     | Esegui clear di Carry per somma       |
| cdfc: 98       | tya                     | Trasf. Y in A                         |
| cdfd: 65 e2    | adc \$e2 [ user ]       | Somma byte LO di ind.                 |
| cdff: 48       | pha                     | Immettilo su Stack                    |
| ce00: a9 00    | lda #\$00               | Carica A con 0                        |
| ce02: 65 e3    | adc \$e3 [ user+ 1 ]    | Somma q.s. al carry                   |
| ce04: 20 cc cd | jsr * \$cdcc            | Immag. byte HI                        |
| ce07: 68       | pla                     | Byte LO da Stack                      |
| ce08: e8       | inx                     | Incrementa X                          |
| ce09: 4c cc cd | jmp * \$cdcc            | Vai a indirizzo                       |
| ce0c: a9 00    | lda #\$00               | Carica A e Y con indirizzo di parte.  |
| ce0e: a0 d0    | ldy #\$d0               | che e' a \$D000                       |
| ce10: 85 da    | sta \$da [ keysiz ]     | Immagazzina valori prec. in Pag. 0    |
| ce12: 84 db    | sty \$db [ keylen ]     | agli ind. \$DA e \$DB                 |
| ce14: a2 12    | ldx #\$12               | Aggiorna reg. HI                      |
| ce16: a9 20    | lda #\$20               | Indir. iniz. del generatore di car.   |
| ce18: 20 cc cd | jsr * \$cdcc            | Q.s. in VDC                           |
| ce1b: e8       | inx                     | Incrementa X                          |
| ce1c: a9 00    | lda #\$00               | Carica A come Byte LO di ind. iniz.   |
| ce1e: 20 cc cd | jsr * \$cdcc            | del generatore di carattere (\$00)    |
| ce21: a0 00    | ldy #\$00               | Puntatore indice a linea/car          |
| ce23: a2 0e    | ldx #\$0e               | Seleziona CHARROM                     |
| ce25: a9 da    | lda #\$da               | Carica A con ind. pag. 0              |
| ce27: 20 74 ff | jsr * \$ff74 [ indfet ] | Vai a INDFET                          |
| ce2a: 20 ca cd | jsr * \$cdca            | Immagazzina in RAM                    |
| ce2d: c8       | iny                     | Incrementa Y. E' un puntatore ind.    |
| ce2e: c0 08    | cpy #\$08               | Contr. se tutti 8 car. sono copiati   |
| ce30: 90 f1    | bcc * \$ce23            | Se contr. neg. vai a pross. linea     |
| ce32: a9 00    | lda #\$00               | Carica A con zero                     |
| ce34: 20 ca cd | jsr * \$cdca            | Immagazzina valore in RAM             |
| ce37: 88       | dex                     | Esegui per 8 volte                    |
| ce38: d0 fa    | bne * \$ce34            | Vai a ind.se non eseguito             |
| ce3a: 18       | clc                     | Esegui clear di carry per somma       |
| ce3b: a5 da    | lda \$da [ keysiz ]     | Carica A con byte LO                  |
| ce3d: 69 08    | adc #\$08               | Sommaci 8                             |
| ce3f: 85 da    | sta \$da [ keysiz ]     | Immagazzina di nuovo                  |
| ce41: 90 e0    | bcc * \$ce23            | Contr. se carry fissato               |
| ce43: e6 db    | inc \$db [ keylen ]     | Se neg. continua                      |
| ce45: a5 db    | lda \$db [ keylen ]     | Controlla se byte HI punta al termine |
| ce47: c9 e0    | cmp #\$e0               | della routine di CHARROM              |
| ce49: 90 d8    | bcc * \$ce23            | Se neg. continua                      |
| ce4b: 60       | rts                     |                                       |

## TAVOLA ASCII DEI CODICI DI COLORE

ce4c: 90 05 1c 9f 9c 1e 1f 9e



ce54: 81 95 96 97 98 99 9a 9b

-----  
TAVOLA DEI CODICI DI COLORE PER VDC

ce5c: 00 0f 08 07 0b 04 02 0d

ce64: 0a 0c 09 06 01 05 03 0e

-----  
TAVOLA DELLE POTENZE DI 2

ce6c: 80 40 20 10 08 04 02 01

-----  
ce74: 00 04 00 d8 18 00 00 27

ce7c: 00 00 00 00 00 18 27 00

ce84: 00 0d 0d 00 00 00 00 00

ce8c: 00 00 00 00 00 08 18 00

ce94: 00 4f 00 00 00 00 00 18

ce9c: 4f 00 00 07 07 00 00 00

cea4: 00 00 00 00 07 06 0a 07

ceac: 06 04 05 08 09 05 47 52

ceb4: 41 50 48 49 43 44 4c 4f

cebc: 41 44 22 44 49 52 45 43

cec4: 54 4f 52 59 0d 53 43 4e

cecc: 43 4c 52 0d 44 53 41 56

ced4: 45 22 52 55 4e 0d 4c 49

cedc: 53 54 0d 4d 4f 4e 49 54

cee4: 4f 52 0d 44 cc 22 2a 0d

ceec: 52 55 4e 0d 48 45 4c 50

cef4: 0d

-----  
cef5 - cfff

Queste tavole contengono i valori per  
schermo a 40 colonne fino a \$CE8C ed i  
valori per 80 colonne fino a \$CEA7.  
Il resto della tavola contiene  
i valori delle stringhe assegnate  
ai tasti funzione

Area libera

## ROUTINE DI RESET

|                |                |                                       |
|----------------|----------------|---------------------------------------|
| e000: a2 ff    | ldx #\$ff      | Valore di iniz. per Stack             |
| e002: 78       | sei            | Disabilita tutti gli interrupt.       |
| e003: 9a       | txs            | Fissa Stack di sistema per iniz.      |
| e004: d8       | cld            | Reset del modo decimale               |
| e005: a9 00    | lda #\$00      | Metti ZERO in A                       |
| e007: 8d 00 ff | sta * \$fff00  | Abilita tutte le ROM di sistema       |
| e00a: a2 0a    | ldx #\$0a      | Prep. ciclo e vis. contatore.         |
| e00c: bd 4b e0 | lda * \$e04b,x | Carica byte per iniz. contatore       |
| e00f: 9d 00 d5 | sta * \$d500,x | Inizializza reg. MMU                  |
| e012: ca       | dex            | Ciclo e vis. cont.-1                  |
| e013: 10 f7    | bpl * \$e00c   | Trasf. 11 valori da tavola            |
| e015: 8d 04 0a | sta \$0a04     | Esegui un clear di NMI                |
| e018: 20 cd e0 | jsr * \$e0cd   | Vai a rou.IRQ,NMI+copy                |
| e01b: 20 f0 e1 | jsr * \$elf0   | Contr.cod. CBM su RAM1                |
| e01e: 20 42 e2 | jsr * \$e242   | Contr. cartr per config. C64          |
| e021: 20 09 e1 | jsr * \$e109   | Vai a Rout.IOINIT                     |
| e024: 20 3d f6 | jsr * \$f63d   | Contr. tast. per SHIFT RUN/STOP       |
| e027: 48       | pha            | Salva A su Stack                      |
| e028: 30 07    | bmi * \$e031   | Fissa bit 7,salta contr.reset         |
| e02a: a9 a5    | lda #\$a5      | Punt.sisst.inizio                     |
| e02c: cd 02 0a | cmp \$0a02     | Contr. per part.(WARM/START)          |
| e02f: f0 03    | beq * \$e034   | Se contr.prec.pos. salta              |
| e031: 20 93 e0 | jsr * \$e093   | Vai a sub.RAMTAS contr.clear di RAM   |
| e034: 20 56 e0 | jsr * \$e056   | Vai a sub.RESTORE: Iniz. I/O          |
| e037: 20 00 c0 | jsr * \$c000   | Vai a CINT per iniz.editor di schermo |
| e03a: 68       | pla            | Carica codice tastiera                |
| e03b: 58       | cli            | Abilita tutti gli interrupt           |
| e03c: 30 03    | bmi * \$e041   | Fissa bit 7.Salta ingresso monitor    |
| e03e: 4c 00 b0 | jmp * \$b000   | Vai a ingresso rout.Monitor           |
| e041: c9 df    | cmp #\$df      | Contr.conf. sist. come C64            |
| e043: f0 03    | beq * \$e048   | Contr. pos. esegui.                   |
| e045: 6c 00 0a | jmp (\$0a00)   | Salta al vett.riprist.sistema         |
| e048: 4c 4b e2 | jmp * \$e24b   | Modo C64:configura come C64           |

## TAVOLA DI INIZIALIZZAZIONE PER MMU

|             |          |                             |        |
|-------------|----------|-----------------------------|--------|
| e04b: 00 00 | (\$0000) | Registro di configurazione  | \$D500 |
| e04d: 00 00 | (\$0000) | Reg. A di preconfigurazione | \$D501 |
| e04f: 00 bf | (\$bf00) | Reg. B di                   | \$D502 |
| e051: 04 00 | (\$0004) | Reg. C di                   | \$D503 |
| e053: 00 01 | (\$0100) | Reg. D di                   | \$D504 |
| e055: 00 a2 | (\$a200) | Reg. di config. MOD0        |        |

## ROUTINE KERNAL RESTOR

|             |           |                                 |
|-------------|-----------|---------------------------------|
| e056: a2 73 | ldx #\$73 | Byte basso tav.vettore kernal   |
| e058: a0 e0 | ldy #\$e0 | Byte alto tav. vettore kernal   |
| e05a: 18    | clc       | Clear di carry per tav. vettori |

## ROUTINE KERNAL VECTOR

|                |              |               |                                 |
|----------------|--------------|---------------|---------------------------------|
| e05b: 86 c3    | stx \$c3     | [ memuss ]    | Byte basso in Pag. zero         |
| e05d: 84 c4    | sty \$c4     | [ memuss+ 1 ] | Byte alto in Pag. zero          |
| e05f: a0 1f    | ldy #\$1f    |               | Fissa a 32 ciclo contatore      |
| e061: b9 14 03 | lda \$0314,y |               | Lett.byte da pag.3 tav. vettori |
| e064: b0 02    | bcs * \$e068 |               | Contr. agg. val. se pos. salta  |
| e066: b1 c3    | lda (\$c3),y | [ memuss ]    | Lett. val. da tavola vettori    |

|                |                         |                                  |
|----------------|-------------------------|----------------------------------|
| e068: 99 14 03 | sta \$0314,y            | Immag. in Pag.3 tavola vettori   |
| e06b: 90 02    | bcc * \$e06f            | Contr. trasf. tavola vett. salta |
| e06d: 91 c3    | sta (\$c3),y [ memuss ] | Copia in pag. indicizzata        |
| e06f: 88       | dey                     | Decrem. di 1 ciclo contatore     |
| e070: 10 ef    | bpl * \$e061            | Ciclo trasf. tavola              |
| e072: 60       | rts                     |                                  |

## TAVOLA DEI VETTORI

|             |          |                             |
|-------------|----------|-----------------------------|
| e073: 65 fa | (\$fa65) | Vettore per ingresso IRQ    |
| e075: 03 b0 | (\$b003) | Vettore ingresso monitor    |
| e077: 40 fa | (\$fa40) | Vettore per ingresso NMI    |
| e079: bd ef | (\$efbd) | Vettore a Subr. Kernal OPEN |
| e07b: 88 f1 | (\$f188) | " " CLOSE                   |
| e07d: 06 f1 | (\$f106) | " " CHKIN                   |
| e07f: 4c f1 | (\$f14c) | " " CKOUT                   |
| e081: 26 f2 | (\$f226) | " " CLRCH                   |
| e083: 06 ef | (\$ef06) | " " BASIN                   |
| e085: 79 ef | (\$ef79) | " " BSOUT                   |
| e087: 6e f6 | (\$f66e) | " " STOP                    |
| e089: eb ee | (\$eeeb) | " " GETIN                   |
| e08b: 22 f2 | (\$f222) | " " CLALL                   |
| e08d: 06 b0 | (\$b006) | " Ingresso Extended Monitor |
| e08f: 6c f2 | (\$f26c) | " a Subr. Kernal LOADSP     |
| e091: 4e f5 | (\$f54e) | " " SAVESP                  |

## ROUTINE KERNAL RAMTAS

|                |                       |   |
|----------------|-----------------------|---|
| e093: a9 00    | lda #\$00             | Inizial. A con \$00                     |
| e095: a8       | tay                   | Trasf. A in Y                           |
| e096: 99 02 00 | sta \$0002,y          | Esegui un clear su tutta Pag. 0         |
| e099: c8       | iny                   | Protez. porte process.da op. prec.      |
| e09a: d0 fa    | bne * \$e096          | Registri \$00 e \$01                    |
| e09c: a0 0b    | ldy #\$0b             | Fissa il puntatore del buffer di cass.  |
| e09e: 84 b3    | sty \$b3 [ tapel+ 1 ] | che si trova in Pag. 0 (\$B2-\$B3) all' |
| e0a0: 85 b2    | sta \$b2 [ tapel ]    | indirizzo di partenza \$0B00            |
| e0a2: a0 0c    | ldy #\$0c             | Esegui come sopra ma per INGRESSO RS    |
| e0a4: 84 c9    | sty \$c9 [ ribuf+ 1 ] | 232 (\$C8-\$C9) per indir.part \$0C00   |
| e0a6: 85 c8    | sta \$c8 [ ribuf ]    | ***                                     |
| e0a8: a0 0d    | ldy #\$0d             | Esegui come sopra ma per USCITA RS      |
| e0aa: 84 cb    | sty \$cb [ robuf+ 1 ] | 232 (\$CA-\$CB) per indir.part \$0D00   |
| e0ac: 85 ca    | sta \$ca [ robuf ]    | ***                                     |
| e0ae: 18       | clc                   | Esegui un clear di carry                |
| e0af: a0 ff    | ldy #\$ff             | Fissa la parte ALTA della memoria       |
| e0b1: a2 00    | ldx #\$00             | nei banchi a \$FF00                     |
| e0b3: 20 6b f7 | jsr * \$f76b          | Vai a subr. MEMTOP                      |
| e0b6: a0 1c    | ldy \$1c              | Fissa la parte BASSA della memoria      |
| e0b8: a2 00    | ldx #\$00             | nei banchi a \$1C00                     |
| e0ba: 20 7a f7 | jsr * \$f77a          | Vai a subr. MEMBOT                      |
| e0bd: a0 40    | ldy \$40              | Inizializza sistema                     |
| e0bf: a2 00    | ldx #\$00             | Fissa il vettore di RESTART ad ind.     |
| e0c1: 8c 01 0a | sty \$0a01            | \$A00-\$A01 con val. \$4000 per punto   |
| e0c4: 8e 00 0a | stx \$0a00            | punto inizio sistema (COLD)             |
| e0c7: a9 a5    | lda \$a5              | Iniz.sistema (COLD-WARM)                |
| e0c9: 8d 02 0a | sta \$0a02            | Puntatore prec. con \$A5                |
| e0cc: 60       | rts                   |   |

## ROUTINE COPIA NMI,IRQ E PAG. 0

|             |           |                              |
|-------------|-----------|------------------------------|
| e0cd: a0 03 | ldy #\$03 | Pred. cont.ciclo per 4 cicli |
|-------------|-----------|------------------------------|

|                |                           |                                       |
|----------------|---------------------------|---------------------------------------|
| e0cf: b9 05 e1 | lda * \$e105,y            | Carica val. da banco RAM              |
| e0d2: 8d 00 ff | sta * \$ff00              | Fissa config. corisp. prec.valore     |
| e0d5: a2 3f    | ldx #\$3f                 | Trasfer. 64 Bytes                     |
| e0d7: bd 05 ff | lda * \$ff05,x            | Carica da ROM NMI+IRQ                 |
| e0da: 9d 05 ff | sta * \$ff05,x            | Copia in RAM                          |
| e0dd: ca       | dex                       | Decr. cont. di 1                      |
| e0de: 10 f7    | bpl * \$e0d7              | Ciclo trasf 64 bytes                  |
| e0e0: a2 05    | ldx #\$05                 | Con questa rout. i vettori di NMI,    |
| e0e2: bd fa ff | lda * \$fffa,x [ nmivec   | RESET e IRQ sono copiati dalla ROM    |
| e0e5: 9d fa ff | sta * \$fffa,x [ nmivec ] | (ZONA KERNAL) nella RAM anzidetta     |
| e0e8: ca       | dex                       | Il ciclo continua fino a quando i     |
| e0e9: 10 f7    | bpl * \$e0e2              | valori dei 3 vett. siano trasf.       |
| e0eb: 88       | dex                       | Decr. cont. di ciclo di 1             |
| e0ec: 10 e1    | bpl * \$e0cf              | Routine di copia vett. in RAM         |
| e0ee: a2 59    | ldx #\$59                 | Trasf. 90 bytes                       |
| e0f0: bd 00 f8 | lda * \$f800,x            | Con questa routine vengono trasferite |
| e0f3: 9d a2 02 | sta \$02a2,x              | in RAM alle pag. 2 e 3 le routines    |
| e0f6: ca       | dex                       | FETCH,STASH,CMPARE,JSRPFAR,JMPFAR     |
| e0f7: 10 f7    | bpl * \$e0f0              | cosi' come sono                       |
| e0f9: a2 0c    | ldx #\$0c                 | Trasf. dei 13 Bytes                   |
| e0fb: bd 5a f8 | lda * \$f85a,x            | Questa routine si comporta            |
| e0fe: 9d f0 03 | sta \$03f0,x              | come quella descritta in prec         |
| el01: ca       | dex                       | per la RAM pero' di indirizzo \$03F0  |
| el02: 10 f7    | bpl * \$e0fb              | ***                                   |
| el04: 60       | rts                       |                                       |

## TAVOLA DEI BANCHI RAM

|             |          |           |
|-------------|----------|-----------|
| el05: 00 40 | (\$4000) | RAM 0 e 1 |
| el07: 80 c0 | (\$c080) | RAM 2 e 3 |

## ROUTINE KERNAL IOINIT

|                |                    |                                      |
|----------------|--------------------|--------------------------------------|
| el09: a9 7f    | lda #\$7f          | Carica valore per interrupt          |
| el0b: 8d 0d dc | sta * \$dc0d       | Iniz. ICR del CIA 1                  |
| el0e: 8d 0d dd | sta * \$dd0d       | Iniz. ICR del CIA 2                  |
| el11: 8d 00 dc | sta * \$dc00       | Iniz. porta A CIA 1                  |
| el14: a9 08    | lda #\$08          | Iniz. timer                          |
| el16: 8d 0e dc | sta * \$dc0e       | CRA del CIA 1 timer A                |
| el19: 8d 0e dd | sta * \$dd0e       | CRA del CIA 2 timer A                |
| el1c: 8d 0f dc | sta * \$dc0f       | CRA del CIA 1 timer B                |
| el1f: 8d 0f dd | sta * \$dd0f       | CRA del CIA 2 timer B                |
| el22: a2 00    | ldx #\$00          | Registro CIA per modo input          |
| el24: 8e 03 dc | stx * \$dc03       | Registro dir. dati B del CIA 1       |
| el27: 8e 03 dd | stx * \$dd03       | Registro dir. dati B del CIA 2       |
| el2a: ca       | dex                | Reg. X per MODO OUTPUT               |
| el2b: 8e 02 dc | stx * \$dc02       | Registro dir. dati A del CIA 1       |
| el2e: a9 07    | lda #\$07          | Fissa video contr. ai 16k di mem.+ba |
| el30: 8d 00 dd | sta * \$dd00       | Seg. ATN su A,clear CIA2             |
| el33: a9 3f    | lda #\$3f          | Fissa bits 0-5 per uscita dati       |
| el35: 8d 02 dd | sta * \$dd02       | Registro dir. dati A del CIA 2       |
| el38: a9 e3    | lda #\$e3          | Iniz. reg.dat. porta proc.           |
| el3a: 85 01    | sta \$01 [ r6510 ] | con valore di default (\$E3)         |
| el3c: a9 2f    | lda #\$2f          | Come sopra ma con valore di          |
| el3e: 85 00    | sta \$00 [ d6510 ] | default \$2F                         |
| el40: a2 ff    | ldx #\$ff          | Iniz. punt PAL/NTSC                  |
| el42: ad 11 d0 | lda * \$d011       | Ciclo di attesa fino a quando MSB di |
| el45: 10 fb    | bpl * \$e142       | linea raster sia fissato.            |
| el47: a9 08    | lda #\$08          | Confr. val PAL/NTSC                  |

|                |                    |  |
|----------------|--------------------|--|
| el49: cd 12 d0 | cmp * \$d012       | Confr. byte basso RASTER                   |
| el4c: 90 06    | bcc * \$el54       | Se inf. a 8 e' versione PAL                |
| el4e: ad 11 d0 | lda * \$d011       | Attendi fino a che MSB della linea         |
| el51: 30 f4    | bmi * \$el47       | RASTER sia messa a 0                       |
| el53: e8       | inx                | Fissa punt. PAL/NTSC a NTSC (\$0)          |
| el54: 8e 03 0a | stx \$0a03         | Immag. in X att. punt PAL/NTSC             |
| el57: a9 00    | lda #\$00          | Iniz. val. per punt.                       |
| el59: 8d 37 0a | sta \$0a37         | Immag. temp. per operaz. banchi            |
| el5c: 8d 39 0a | sta \$0a39         | Immag. temp. per VDC 80 colonne            |
| el5f: 8d 0a 0a | sta \$0a0a         | Idem IRQ cassetta                          |
| el62: 8d 3a 0a | sta \$0a3a         | Iniz. temp. punt. IRQ                      |
| el65: 8d 36 0a | sta \$0a36         | Fissa linea RASTER per RASTER interruttore |
| el68: 85 99    | sta \$99 [ dfltn ] | La perif. standard (ingr) = tast.          |
| el6a: a9 03    | lda #\$03          | Fissa immag. in pag. 0 a 3 per perif       |
| el6c: 85 9a    | sta \$9a [ dflto ] | in uscita (schermo)                        |
| el6e: a2 30    | ldx #\$30          | Trasferisci 49 Bytes                       |
| el70: bd c7 e2 | lda * \$e2c7,x     | Tavola di inizializz per int. VIC          |
| el73: 9d 00 d0 | sta * \$d000,x     | Copia q.s. nei reg. di controllo           |
| el76: ca       | dex                | Dim. ciclo di l                            |
| el77: 10 f7    | bpl * \$el70       | Ciclo trasf. 49 valori                     |
| el79: a2 00    | ldx #\$00          | Fissa ciclo cont. per iniz. VDC            |
| el7b: 20 dc e1 | jsr * \$eldc       | Inizializza reg. VDC                       |
| el7e: ad 00 d6 | lda * \$d600       | Leggi status VDC                           |
| el81: 29 07    | and #\$07          | Contr. per clear bits 0-2                  |
| el83: f0 05    | beq * \$el8a       | Se pos. salta iniz. reg. VDC               |
| el85: a2 3b    | ldx #\$3b          | Sposta punt. a tavola VDC                  |
| el87: 20 dc e1 | jsr * \$eldc       | Inizializza reg. VDC                       |
| el8a: 2c 03 0a | bit \$0a03         | Controllo per vers PAL/NTSC                |
| el8d: 10 05    | bpl * \$el94       | Salta se versione NTSC                     |
| el8f: a2 3e    | ldx #\$3e          | Spostam. punt. alla tavola VDC             |
| el91: 20 dc e1 | jsr * \$eldc       | Iniziali. registri VDC                     |
| el94: ad 04 0a | lda \$0a04         | Controllo punt. status reset NMI           |
| el97: 30 15    | bmi * \$elae       | Se VDC gia' iniz. salta                    |
| el99: 20 27 c0 | jsr * \$c027       | Vai a rout. INIT80                         |
| el9c: a9 80    | lda #\$80          | Bit 7 in A                                 |
| el9e: 0d 04 0a | ora \$0a04         | Esegui OR con st. NMI/VDC                  |
| elal: 8d 04 0a | sta \$0a04         | Scrivilo in FLAG di Status                 |
| ela4: a2 ff    | ldx #\$ff          | Cont. ciclo alto al max. val.              |
| ela6: a0 ff    | ldy #\$ff          | Cont. ciclo basso al min.                  |
| ela8: 88       | dey                | Decr. ciclo cont. basso                    |
| ela9: d0 fd    | bne * \$ela8       | Contr. ciclo cod. basso. Se neg. cont.     |
| elab: ca       | dex                | Decr. ciclo cont. alto                     |
| elac: d0 fa    | bne * \$ela8       | Contr. ciclo alto. Se neg. cont.           |
| elae: a9 00    | lda #\$00          | Iniz. valore per reg. SID                  |
| elb0: a2 18    | ldx #\$18          | Puntatore e ciclo SID                      |
| elb2: 9d 00 d4 | sta * \$d400,x     | Esegui un clear sul SID                    |
| elb5: ca       | dex                | Ciclo e decr. punt                         |
| elb6: 10 fa    | bpl * \$elb2       | Ciclo e contr. canc. 19 reg.               |
| elb8: a2 01    | ldx #\$01          | Carica X con #1                            |
| elba: 8e 1a d0 | stx * \$d01a       | Fissa reg. IRQ                             |
| elbd: ca       | dex                | Decr. fino a 0 X                           |
| elbe: 8e 1c 0a | stx \$0alc         | Esegui un clear punt. seriale              |
| elcl: 8e 0f 0a | stx \$0a0f         | Esegui clear RS-232 NMI                    |
| elc4: ca       | dex                | Fissa X a \$FF (val. alto)                 |
| elc5: 8e 06 dc | stx * \$dc06       | Immetti val. in timer B basso              |
| elc8: 8e 07 dc | stx * \$dc07       | Immetti val. in timer B alto               |
| elcb: a2 11    | ldx #\$11          | Car. cod. per caric. forzato e timer A     |
| elcd: 8e 0f dc | stx * \$dc0f       | Iniz. reg. contr. CIA                      |
| eld0: 20 c3 e5 | jsr * \$e5c3       | Routine di controllo per modo FAST         |
| eld3: 20 d6 e5 | jsr * \$e5d6       | seriale per unita' a dischi                |

|       |          |       |          |                                 |
|-------|----------|-------|----------|---------------------------------|
| eld6: | 20 c3 e5 | jsr * | \$e5c3   | Rif. rout. prec a segnale basso |
| eld9: | 4c 4e e5 | jmp * | \$e54e   | Se contr. pos. esegui RTS       |
| eldc: | bc f8 e2 | ldy * | \$e2f8,x | Carica sel. regist.da tavola    |
| eldf: | 30 0d    | bmi * | \$elee   | Controllo bit 7 ON              |
| el1:  | e8       | inx   |          | Increment. 1 tavola VDC         |
| ele2: | bd f8 e2 | lda * | \$e2f8,x | Carica A con val da tavola      |
| ele5: | e8       | inx   |          | Incr. tavola VDC di 1           |
| ele6: | 8c 00 d6 | sty * | \$d600   | Fissa porta di sel.reg.VDC      |
| ele9: | 8d 01 d6 | sta * | \$d601   | Metti in A reg. prec.           |
| elec: | 10 ee    | bpl * | \$eldc   | Vai al ciclo di inizio          |
| elee: | e8       | inx   |          | Incr. tavola VDC 1              |
| elef: | 60       | rts   |          |                                 |

-----  
CONTROLLO PER CODICE CBM IN RAM

|       |          |          |                    |  |
|-------|----------|----------|--------------------|--|
| elf0: | a2 f5    | ldx #    | \$f5               | Iniz.2 Byte in pag. 0 con i valori     |
| elf2: | a0 ff    | ldy #    | \$ff               | \$C3 (LO) e \$C4 (HI) per l' indirizzo |
| elf4: | 86 c3    | stx \$c3 | [ memuss ]         | iniziale della tavola vettori kernal   |
| elf6: | 84 c4    | sty \$c4 | [ memuss+ 1 ]      | di indirizzo \$FFF5                    |
| elf8: | a9 c3    | lda      | \$c3               | Fissa FETVEC per la inizializzazione   |
| elfa: | 8d aa 02 | sta      | \$02aa             | routine di inizio tav. vettori         |
| elfd: | a0 02    | ldy #    | \$02               | Spostam. per rout. FETCH               |
| elff: | a2 7f    | ldx #    | \$7f               | Cod. di conf(valido solo RAM 1)        |
| e201: | 20 a2 02 | jsr      | \$02a2             | Rout. FETCH                            |
| e204: | d9 c4 e2 | cmp *    | \$e2c4,y           | Controllo codici C B M                 |
| e207: | d0 1b    | bne *    | \$e224             | Se diversi esci                        |
| e209: | 88       | dey      |                    | Esegui ciclo per controllo delle       |
| e20a: | 10 f3    | bpl *    | \$elff             | 3 lettere precedenti                   |
| e20c: | a2 f8    | ldx #    | \$f8               | Iniz. i due byte di pag.0              |
| e20e: | a0 ff    | ldy #    | \$ff               | Punt. a ind. \$C3 (LO) e \$C4 (HI)     |
| e210: | 86 c3    | stx \$c3 | [ memuss ]         | Indirizzi Kernal 128                   |
| e212: | 84 c4    | sty \$c4 | [ memuss+ 1 ]      | Vettore di ind. prec. (\$FFF8)         |
| e214: | a0 01    | ldy #    | \$01               | Codice di configur. per rout. FETCH    |
| e216: | a2 7f    | ldx #    | \$7f               | per solo RAM 1                         |
| e218: | 20 a2 02 | jsr      | \$02a2             | Salta a rout. FETCH                    |
| e21b: | 99 02 00 | sta      | \$0002,y           | Immetti ind.ingr. (LO-HI) in pagina    |
| e21e: | 88       | dey      |                    | zero per \$02-\$03                     |
| e21f: | 10 f5    | bpl *    | \$e216             | Ciclo per trasf. indirizzi             |
| e221: | 6c 02 00 | jmp      | (\$0002)           | Salto indir. via pag. zero             |
| e224: | a9 40    | lda      | \$40               | RAM 1, abilita tutte le ROM            |
| e226: | 8d 00 ff | sta *    | \$ff00             | Fissa la config.                       |
| e229: | a9 24    | lda      | \$24               | Iniz. i 2 bytes del vettore KERNAL     |
| e22b: | a0 e2    | ldy #    | \$e2               | per il modo 128 con il valore          |
| e22d: | 8d f8 ff | sta *    | \$fff8 [ cl28vec ] | di default che viene quindi fissato in |
| e230: | 8c f9 ff | sty *    | \$fff9             | \$E224                                 |
| e233: | a2 03    | ldx #    | \$03               | Cont. ciclo trasf.3 valori             |
| e235: | bd c3 e2 | lda *    | \$e2c3,x           | Carica C B M da tavola                 |
| e238: | 9d f4 ff | sta *    | \$fff4,x           | Copia su vett.spazio RAM banco 1       |
| e23b: | ca       | dex      |                    | Ciclo che controlla trasf. delle       |
| e23c: | d0 f7    | bne *    | \$e235             | 3 lettere viste sopra                  |
| e23e: | 8e 00 ff | stx *    | \$ff00             | RAM 0, abilita tutta la ROM            |
| e241: | 60       | rts      |                    |  |

-----  
CONTROLLO PER EXTRAROM

|       |          |       |        |                                       |
|-------|----------|-------|--------|---------------------------------------|
| e242: | ad 05 d5 | lda * | \$d505 | Leggi reg. MCR di MMU                 |
| e245: | 29 30    | and # | \$30   | Contr. se bit 5 e' stato fissato per  |
| e247: | c9 30    | cmp # | \$30   | esecuzione EXROM                      |
| e249: | f0 20    | beq * | \$e26b | Se pos. nessun cartr. C64 e' presente |
| e24b: | a9 e3    | lda # | \$e3   | Carica i valori del C64 sulla porta   |

|                |                          |           |                                      |
|----------------|--------------------------|-----------|--------------------------------------|
| e24d: 85 01    | sta \$01                 | [ r6510 ] | del proce.(registro dati)            |
| e24f: a9 2f    | lda #\$2f                |           | Come sopra nel reg. di               |
| e251: 85 00    | sta \$00                 | [ d6510 ] | direzione dati                       |
| e253: a2 08    | ldx #\$08                |           | Copia gli 8 bytes                    |
| e255: bd 62 e2 | lda * \$e262,x           |           | Con questa Rout. la ROM orig.        |
| e258: 95 01    | sta \$01,x               | [ r6510 ] | con tutti i sui valori e' copiata    |
| e25a: ca       | dex                      |           | nella pagina 0 perche' le rout. pos. |
| e25b: d0 f8    | bne * \$e255             |           | girare solo qui                      |
| e25d: 8e 30 d0 | stx * \$d030             |           | Metti il clock a 1Mhz                |
| e260: 4c 02 00 | jmp \$0002               |           | Conf. quindi C64                     |
| e263: a9 f7    | lda #\$f7                |           | Scrivi val. iniz. sist. C64          |
| e265: 8d 05 d5 | sta * \$d505             |           | nei reg MCR di MMU                   |
| e268: 6c fc ff | jmp (* \$fffc)[ resvec ] |           | Vai a vettore di RESET C64           |
| e26b: a2 03    | ldx #\$03                |           | Iniz. ciclo e vis. contato.          |
| e26d: 8e c0 0a | stx \$0ac0               |           | per contr. pres. cartridge           |
| e270: a9 00    | lda \$00                 |           | Esegui un clear dei primi 4 Bytes    |
| e272: 9d c1 0a | sta \$0acl,x             |           | del PAT cioe' della tavola di indir. |
| e275: ca       | dex                      |           | fisico della EXP. Cart               |
| e276: 10 fa    | bpl * \$e272             |           | Se \$00 e' inizializ.                |
| e278: 85 9e    | sta \$9e                 | [ ptr1 ]  | Immag.ind.LO per contr. cartridge    |
| e27a: a0 09    | ldy #\$09                |           | Spost. codice Cart.                  |
| e27c: ae c0 0a | ldx \$0ac0               |           | Spost. cont. per contr. cart         |
| e27f: bd bc e2 | lda * \$e2bc,x           |           | Carica ind. HI da tavola             |
| e282: 85 9f    | sta \$9f                 | [ ptr2 ]  | Immetti in pag. 0                    |
| e284: bd c0 e2 | lda * \$e2c0,x           |           | Carica da tav. val.banco per contr.  |
| e287: 85 02    | sta \$02                 | [ bank ]  | Immetti byte banco in pag.0          |
| e289: a6 02    | ldx \$02                 | [ bank ]  | Carica cod. banco da pag. 0          |
| E28b: a9 9e    | lda #\$9e                |           | Metti in A ind. \$9E come VETVEC     |
| e28d: 20 d0 f7 | jsr * \$f7d0             |           | Vai a INDFET                         |
| e290: d9 bd e2 | cmp * \$e2bd,y           |           | Contr. l car., per codice CBM        |
| e293: d0 21    | bne * \$e2b6             |           | Se diverso,pross.ind. banco          |
| e295: 88       | dey                      |           | Continua contr. per cod. CBM         |
| e296: c0 07    | cpy #\$07                |           | Se le tre lett. sono ric.cont.       |
| e298: b0 ef    | bcx * \$e289             |           | Se no ciclo                          |
| e29a: a6 02    | ldx \$02                 | [ bank ]  | Carica cod.banco per contr.          |
| e29c: a9 9e    | lda #\$9e                |           | Metti in A ind.\$9E come FETVEC      |
| e29e: 20 d0 f7 | jsr * \$f7d0             |           | Vai a rout. INDFET                   |
| e2a1: ae c0 0a | ldx \$0ac0               |           | Carica punt. spost. ROM              |
| e2a4: 9d c1 0a | sta \$0acl,x             |           | Immag.tavola ident. espans.          |
| e2a7: c9 01    | cmp #\$01                |           | Controlla espans. indicata           |
| e2a9: d0 0b    | bne * \$e2b6             |           | Se div. vai al pross. contr.         |
| e2ab: a5 9e    | lda \$9e                 | [ ptr1 ]  | Metti in A punt. ind.ingr.           |
| e2ad: a4 9f    | ldy \$9f                 | [ ptr2 ]  | Metti in Y c.s.                      |
| e2af: 85 04    | sta \$04                 | [ pc-lo ] | Ind.ingr LO in PC-LO                 |
| e2b1: 84 03    | sty \$03                 | [ pc-hi ] | Ind.ingr HI in PC-HI                 |
| e2b3: 20 cd 02 | jsr \$02cd               |           | Vai a JSRFAR                         |
| e2b6: ce c0 0a | dec \$0ac0               |           | Decr. di l cont. spost.              |
| e2b9: 10 bf    | bpl * \$e27a             |           | Se diverso da 0 continua             |
| e2bb: 60       | rts                      |           |                                      |

## INDIRIZZI ALTI PER CONTROLLO CARTRIDGE

|                   |                |
|-------------------|----------------|
| e2bc: c0 80 c0 80 | \$C000, \$8000 |
|-------------------|----------------|

## NUMERO BANCHI PER CONTROLLO CARTRIDGE

|                   |
|-------------------|
| e2c0: 04 04 08 08 |
|-------------------|

## CODICI PER INDICAZIONI CARTRIDGE

e2c4: 43 42 4d                      cbm                      C B M

## TAVOLA DI INIZIALIZZAZIONE PER REGISTRI VIC

```
e2c7: 00 00 00 00 00 00 00 00
e2cf: 00 00 00 00 00 00 00 00
e2d7: 00 1b ff 00 00 00 08 00
e2df: 14 ff 01 00 00 00 00 00
e2e7: 0d 0b 01 02 03 01 02 00
e2ef: 01 02 03 04 05 06 07 ff
e2ff: fc
```

## TAVOLA DI INIZIALIZZAZIONE PER REGISTRI VDC

```
e2f8: 00 7e 01 50 02 66 03 49
e300: 04 20 05 00 06 19 07 1d
e308: 08 00 09 07 0a 20 0b 07
e310: 0c 00 0d 00 0e 00 0f 00
e318: 14 08 15 00 17 08 18 20
e320: 19 40 1a f0 1b 00 1c 20
e328: 1d 07 22 7d 23 64 24 05
e330: 16 78 ff 19 47 ff 04 27
e338: 07 20 ff
```

## ROUTINES KERNAL TALK E LISTN

```

e33b: 09 40      ora $F40      Fissa bit 6 per TALK
e33d: 2c        .byte $2c    Vai a $e340
e33e: 09 20      ora $F20      Fissa bit 5 per operaz. di LISTEN
e340: 20 ec e7    jsr * $e7ec    Attendi per fine trasf RS-232
e343: 48        pha        Salva su Stack segn. Talk/Listn
e344: 24 94      bit $94    [ c3p0 ]    Contr. per altro byte in uscita
e346: 10 0a      bpl * $e352    Contr.neg.continua
e348: 38        sec        Fissa il carry per rot.
e349: 66 a3      ror $a3    [ pcntr ]    Fissa Flag per EOI
e34b: 20 8c e3    jsr * $e38c    Uscita byte su bus seriale
e34e: 46 94      lsr $94    [ c3p0 ]    Canc. caratt.nel segn.buffer
e350: 46 a3      lsr $a3    [ pcntr ]    Clear del flag EOI
e352: 68        pla        Riprist.vecchio A
e353: 85 95      sta $95    [ bsour ]    Immag. byte uscita pag. 0
e355: 20 73 e5    jsr * $e573    Disabilita sprite.(1MHz)
e358: 20 57 e5    jsr * $e557    Metti HI uscita dati
e35b: ad 00 dd    lda * $dd00    Contr. se segn. ATN e' fissato
e35e: 29 08      and #$08    Contr. porta A CIA 2
e360: d0 12      bne * $e374    Se contr. neg. salta
e362: 20 d6 e5    jsr * $e5d6    Impulso per modo serial. FAST
e365: a9 ff      lda #$ff    Fis. I/O dati buffer seriale
e367: 8d 0c dc    sta * $dc0c    Immag. trasf. valore HI
e36a: 20 bc e5    jsr * $e5bc    Attesa risposta Bus
e36d: 8a        txa        Immag. X in A
e36e: a2 14      ldx #$14    Fissa cont. ciclo a 20
e370: ca        dex        Decr. cont.ciclo di 1
e371: d0 fd      bne * $e370    Attesa ciclo conto alla rov.
e373: aa        tax        Ripris. cont. prec. X
e374: ad 00 dd    lda * $dd00    Leggi val. porta A CIA 2
e377: 09 08      ora #$08    Fissa in LO segnale di ATN e riscrivi.
e379: 8d 00 dd    sta * $dd00    su porta A del CIA 2
e37c: 20 73 e5    jsr * $e573    Freq. clock 1 MHz. Dis. sprites
e37f: 20 4e e5    jsr * $e54e    Uscita clock basso

```



|                |              |           |                                       |
|----------------|--------------|-----------|---------------------------------------|
| e382: 20 57 e5 | jsr * \$e557 |           | Uscita clock alto                     |
| e385: 8a       | txa          |           | Immag. in A cont. X                   |
| e386: a2 b8    | ldx #\$b8    |           | Metti a 184 cont.ciclo                |
| e388: ca       | dex          |           | Decr. cont. ciclo di 1                |
| e389: d0 fd    | bne * \$e388 |           | Ciclo fino a cont.=0                  |
| e38b: aa       | tax          |           | Riprist. cont. reg. X                 |
| e38c: 20 73 e5 | jsr * \$e573 |           | Freq. clock 1 MHz. Dis. sprite        |
| e38f: 20 57 e5 | jsr * \$e557 |           | Metti HI uscita dati                  |
| e392: 20 69 e5 | jsr * \$e569 |           | Metti in carry bit da bus seriale     |
| e395: 90 03    | bcc * \$e39a |           | Contr.dati non LO.Se pos salta        |
| e397: 4c 28 e4 | jmp * \$e428 |           | Segn. di DEVICE NOT PRESENT           |
| e39a: 2c 0d dc | bit * \$dc0d |           | Contr. reg.interrupt CIA              |
| e39d: 20 45 e5 | jsr * \$e545 |           | Uscita clock alta                     |
| e3a0: 24 a3    | bit \$a3     | [ pcntr ] | Contr.fiss.punt.Pag.0 per EOI         |
| e3a2: 10 0a    | bpl * \$e3ae |           | Se contr. neg. salta                  |
| e3a4: 20 69 e5 | jsr * \$e569 |           | Metti in carry bit da bus seriale     |
| e3a7: 90 fb    | bcc * \$e3a4 |           | Att.fino a segnale data LO            |
| e3a9: 20 69 e5 | jsr * \$e569 |           | Come prec.                            |
| e3ac: b0 fb    | bcs * \$e3a9 |           | Att.fino a segnale data HI            |
| e3ae: ad 00 dd | lda * \$dd00 |           | In questa rout. i dati sono letti     |
| e3b1: cd 00 dd | cmp * \$dd00 |           | dalla porta A del CIA 2               |
| e3b4: d0 f8    | bne * \$e3ae |           | Controllo diff.                       |
| e3b6: 48       | pha          |           | I dati letti imm. in Stack            |
| e3b7: ad 0d dc | lda * \$dc0d |           | Contr. reg. di interrupt              |
| e3ba: 29 08    | and #\$08    |           | Verif. se timer A attivo              |
| e3bc: f0 05    | beq * \$e3c3 |           | Se pos. salta                         |
| e3be: a9 c0    | lda #\$c0    |           | Fissa bits 6 e 7 nei punt. di sist.   |
| e3c0: 8d 1c 0a | sta \$0alc   |           | per modo seriale FAST                 |
| e3c3: 68       | pla          |           | Metti in A dati ril. da STACK         |
| e3c4: 10 e8    | bpl * \$e3ae |           | Bit 7 clear quindi salta a rout.      |
| e3c6: 09 10    | ora #\$10    |           | Metti a 1 bit 4 per uscita Clock su   |
| e3c8: 8d 00 dd | sta * \$dd00 |           | bus seriale e scrivi su porta A       |
| e3cb: 29 08    | and #\$08    |           | Controllo se bit 3 a 1                |
| e3cd: d0 13    | bne * \$e3e2 |           | Se neg. salta                         |
| e3cf: 2c 1c 0a | bit \$0alc   |           | Contr. bit 7                          |
| e3d2: 10 0e    | bpl * \$e3e2 |           | Se bit 7 =0 salta.                    |
| e3d4: 20 d6 e5 | jsr * \$e5d6 |           | Segn. per modo seriale FAST           |
| e3d7: a5 95    | lda \$95     | [ bsour ] | Carica byte immag. in prec e scrivilo |
| e3d9: 8d 0c dc | sta * \$dc0c |           | sul registro di I/O del CIA           |
| e3dc: 20 bc e5 | jsr * \$e5bc |           | Attesa risposta dal bus               |
| e3df: 4c 12 e4 | jmp * \$e412 |           | Byte in uscita su bus seriale         |
| e3e2: a9 08    | lda #\$08    |           | Iniz. contatore per n. di bits da     |
| e3e4: 85 a5    | sta \$a5     | [ cntdn ] | inviare con 8                         |
| e3e6: ad 00 dd | lda * \$dd00 |           | Rout. di lettura dati dalla           |
| e3e9: cd 00 dd | cmp * \$dd00 |           | porta A del CIA 2                     |
| e3ec: d0 f8    | bne * \$e3e6 |           |                                       |
| e3ee: 0a       | asl a        |           | Dati spos.(sin) in flag di carry      |
| e3ef: 90 34    | bcc * \$e425 |           | Uscita dati alti                      |
| e3f1: 66 95    | ror \$95     | [ bsour ] | Prepara bit per uscita                |
| e3f3: b0 05    | bcs * \$e3fa |           | Contr. se bit = 1                     |
| e3f5: 20 60 e5 | jsr * \$e560 |           | Se neg. fai uscire dati LO            |
| e3f8: d0 03    | bne * \$e3fd |           | Salta quindi al Clock uscita HI       |
| e3fa: 20 57 e5 | jsr * \$e557 |           | Uscita dati HI                        |
| e3fd: 20 45 e5 | jsr * \$e545 |           | Uscita Clock HI                       |
| e400: ea       | nop          |           | Nessuna operazione                    |
| e401: ea       | nop          |           | " "                                   |
| e402: ea       | nop          |           | " "                                   |
| e403: ea       | nop          |           | " "                                   |
| e404: ad 00 dd | lda * \$dd00 |           | Leggi la porta A del CIA 2            |
| e407: 29 df    | and #\$df    |           | Eseg.clear dati usc.bus seriale       |

|       |          |       |          |           |  |
|-------|----------|-------|----------|-----------|--|
| e409: | 09 10    | ora   | #\$10    |           | Fissa uscita clock bus seriale         |
| e40b: | 8d 00 dd | sta   | * \$dd00 |           | Scritt.datì porta A                    |
| e40e: | c6 a5    | dec   | \$a5     | [ cntdn ] | Decr.cont.bit di 1                     |
| e410: | d0 d4    | bne   | * \$e3e6 |           | Bit uscita add.Quindi ciclo            |
| e412: | 8a       | txa   |          |           | Copia X in A                           |
| e413: | 48       | pha   |          |           | Immag. X in Stack                      |
| e414: | a2 22    | ldx   | #\$22    |           | Cont.impul Hi a #34                    |
| e416: | 20 69 e5 | jsr   | * \$e569 |           | Metti 1 bit dal bus seriale sul carry  |
| e419: | b0 05    | bcs   | * \$e420 |           | Metti i data HI poi salta              |
| e41b: | 68       | pla   |          |           | Prendi il cont.vecchio reg. X dallo    |
| e41c: | aa       | tax   |          |           | Stack ed esegui un Restore             |
| e41d: | 4c 9f e5 | jmp   | * \$e59f |           | Reset freq. clock                      |
| e420: | ca       | dex   |          |           | Decrem. di 1 contat. data HI           |
| e421: | d0 f3    | bne   | * \$e416 |           | Contr. se 22 imp. HI.Se neg. cont.     |
| e423: | 68       | pla   |          |           | Ripren.cont.vecchio X da Stack e ripr. |
| e424: | aa       | tax   |          |           | contenuti di X                         |
| e425: | a9 03    | lda   | #\$03    |           | Codice per stato sistema               |
| e427: | 2c       | .byte | \$2c 9   |           | Vai a \$e42a                           |
| e428: | a9 80    | lda   | #\$80    |           | Device not present                     |
| e42a: | 48       | pha   |          |           | Immag. su Stack codice Status          |
| e42b: | ad 1c 0a | lda   | \$0alc   |           | Controlla punt.modo seri.FAST          |
| e42e: | 29 7f    | and   | #\$7f    |           | Mask out bit 7                         |
| e430: | 8d 1c 0a | sta   | \$0alc   |           | Scrivi su flag FAST                    |
| e433: | 68       | pla   |          |           | Preleva cod. errore                    |
| e434: | 20 57 f7 | jsr   | * \$f757 |           | Fissa nuovo status del sistema         |
| e437: | 20 9f e5 | jsr   | * \$e59f |           | Esegui Reset freq. clock               |
| e43a: | 18       | clc   |          |           | Clear carry per ind. di OK             |
| e43b: | 4c 35 e5 | jmp   | * \$e535 |           | Disab perif. con UNLSN                 |
| e43e: | 20 73 e5 | jsr   | * \$e573 |           | Metti clock di siste a 1MHz.Dis.sprit  |
| e441: | a9 00    | lda   | #\$00    |           | Esegui un Clear su punt.pag.0 per      |
| e443: | 85 a5    | sta   | \$a5     | [ cntdn ] | indicatore seriale EOI                 |
| e445: | 2c 0d dc | bit   | * \$dc0d |           | Leggi bit 7 del CIA                    |
| e448: | 8a       | txa   |          |           | Immag. in A att. val X                 |
| e449: | 48       | pha   |          |           | per mezzo di A su Stack                |
| e44a: | 20 45 e5 | jsr   | * \$e545 |           | Manda segn. di clock su porta A        |
| e44d: | 20 69 e5 | jsr   | * \$e569 |           | Metti bit da bus seriale su Carry      |
| e450: | 10 fb    | bpl   | * \$e44d |           | Attesa per segn. dati alto             |
| e452: | a2 0d    | ldx   | #\$0d    |           | Iniz.cont.ciclo con #13                |
| e454: | ad 00 dd | lda   | * \$dd00 |           | Leggi dati porta A di CIA 2            |
| e457: | 29 df    | and   | #\$df    |           | Eseg.clear bit 6.Att.imp.bus seriale   |
| e459: | 8d 00 dd | sta   | * \$dd00 |           | Scrivi dati su porta A                 |
| e45c: | ad 00 dd | lda   | * \$dd00 |           | Leggi dati da porta A del CIA 2        |
| e45f: | cd 00 dd | cmp   | * \$dd00 |           | Arrivo di un bit sul bus seriale       |
| e462: | d0 f8    | bne   | * \$e45c |           | alla porta                             |
| e464: | 0a       | asl   | a        |           | Sposta bit dati sul carry              |
| e465: | 10 1d    | bpl   | * \$e484 |           | Preleva un byte dal bus                |
| e467: | ca       | dex   |          |           | Decrementa cont. cicli di 1            |
| e468: | d0 f2    | bne   | * \$e45c |           | Contr.ciclo non 0 quindi salta         |
| e46a: | a5 a5    | lda   | \$a5     | [ cntdn ] | Contr.punt. EOI in Pag. 0              |
| e46c: | d0 0f    | bne   | * \$e47d |           | Se prec #0 EOI rice.Quindi salta       |
| e46e: | 20 60 e5 | jsr   | * \$e560 |           | Manda segn.data LO su bus seriale      |
| e471: | 20 45 e5 | jsr   | * \$e545 |           | Segnale clock alto su bus seriale      |
| e474: | a9 40    | lda   | #\$40    |           | Cod. status linea EOI                  |
| e476: | 20 57 f7 | jsr   | * \$f757 |           | Reset di sistema                       |
| e479: | e6 a5    | inc   | \$a5     | [ cntdn ] | Punt. EOI                              |
| e47b: | d0 d5    | bne   | * \$e452 |           | Metti byte dati su EOI                 |
| e47d: | 68       | pla   |          |           | Ripr cont. X                           |
| e47e: | aa       | tax   |          |           | Usa A da Stack                         |
| e47f: | a9 02    | lda   | #\$02    |           | Cod. status per lettura                |
| e481: | 4c 2a e4 | jmp   | * \$e42a |           | Reset di sistema                       |

```

e484: a2 08      ldx #$08      Fissa cont.per bits di 8 data
e486: ad 0d dc      lda * $dc0d      Leggi reg.contr. interrupt
e489: 29 08      and #$08      Controllo di interrupt per TIMER
e48b: d0 28      bne * $e4b5      CLOCK o BUS. Se pos. salta
e48d: ad 00 dd      lda * $dd00      Leggi dati porta A di CIA 2 ed att.
e490: cd 00 dd      cmp * $dd00      che un bit sia pos. sulla porta
e493: d0 f8      bne * $e48d      ***
e495: 0a          asl a          Sposta bit dati su carry
e496: 10 ee      bpl * $e486      Attendi per dati validi
e498: 66 a4      ror $a4      [ firt ] Bit di dati in immag. bit
e49a: ad 00 dd      lda * $dd00      Leggi porta dati del CIA 2
e49d: cd 00 dd      cmp * $dd00      ed attendi fino a che i dati arr.
e4a0: d0 f8      bne * $e49a      alla porta
e4a2: 0a          asl a          Sposta bit dati sul carry
e4a3: 30 f5      bmi * $e49a      Se neg. attendi
e4a5: ca          dex          Decr.cont.n.bit dati di 1
e4a6: f0 17      beq * $e4bf      Contr. 8 bit di dati.Quindi salta
e4a8: ad 00 dd      lda * $dd00      Simile al prec. per lettura
e4ab: cd 00 dd      cmp * $dd00      dati porta A CIA 2
e4ae: d0 f8      bne * $e4a8      ***
e4b0: 0a          asl a          Spost bit dati nel flag di carry
e4b1: 10 f5      bpl * $e4a8      Salta se bit ricevuto e' 0
e4b3: 30 e3      bmi * $e498      Salta se bit ricevuto e' 1
e4b5: ad 0c dc      lda * $dc0c      Immag. contenuto buffer dati I/O
e4b8: 85 a4      sta $a4      [ firt ] in pag. 0
e4ba: a9 c0      lda #$c0      Fissa bits 6 e 7 nel flag SYS
e4bc: 8d 1c 0a     sta $0alc      Modo seriale FAST
e4bf: 68          pla          Ripristina vecchio cont.X per mezzo
e4c0: aa          tax          dell' A dallo stack
e4c1: 20 60 e5     jsr * $e560      In.segn.dat. LO su bus seriale
e4c4: 24 90      bit $90      [ status ] Contr. STATUS per bit EOI=1
e4c6: 50 03      bvc * $e4cb      Cer. EOF.Se neg. continua
e4c8: 20 38 e5     jsr * $e538      Perif. disabil.Rout UNLSN
e4cb: 20 9f e5     jsr * $e59f      Reset su freq. clock
e4ce: a5 a4      lda $a4      [ firt ] Metti byte dati in A
e4d0: 18          clc          Es. clear di Carry per indic. OK
e4d1: 60          rts

```

## ROUTINE KERNAL SECND

```

e4d2: 85 95      sta $95      [ bsour ] Immag. pag.0 ind. sec.
e4d4: 20 7c e3     jsr * $e37c      Uscita dati con ATN
e4d7: ad 00 dd      lda * $dd00      Leggi dati porta A di CIA 2
e4da: 29 f7      and #$f7      Es. mask out bit 3
e4dc: 8d 00 dd      sta * $dd00      Riprendi segn. ATN e in. su bus seriale

```

## ROUTINE KERNAL TKSA &amp; CIOUT

```

e4e0: 85 95      sta $95      [ bsour ] Immag. ind.sec. in Pag. 0
e4e2: 20 7c e3     jsr * $e37c      Uscita dati con ATN
e4e5: 24 90      bit $90      [ status ] Contr. STATUS per bit EOF = 1
e4e7: 30 4c      bmi * $e535      Ril. EOF. Vai a UNLSN
e4e9: 20 73 e5     jsr * $e573      Metti freq. clock a 1MHz
e4ec: 20 60 e5     jsr * $e560      In. segn.dat. LO su bus seriale
e4ef: 20 d7 e4     jsr * $e4d7      Vai a subr. SECND
e4f2: 20 45 e5     jsr * $e545      In. segn.clock alto su bus seriale
e4f5: ad 00 dd      lda * $dd00      Leggi dati porta A del CIA 2 ed
e4f8: cd 00 dd      cmp * $dd00      attendi fino a quando un bit giunga
e4fb: d0 f8      bne * $e4f5      sulla porta A
e4fd: 0a          asl a          Sposta bit dati su Carry

```

|                |              |           |                                     |
|----------------|--------------|-----------|-------------------------------------|
| e4fe: 30 f5    | bmi * \$e4f5 |           | Attesa per dati HI                  |
| e500: 4c 9f e5 | jmp * \$e59f |           | Reset freq. clock                   |
| e503: 24 94    | bit \$94     | [ c3p0 ]  | Contr.se uscita altro Byte          |
| e505: 30 05    | bmi * \$e50c |           | Se pos. vai a uscita ciclo          |
| e507: 38       | sec          |           | Fissa carry per rotaz.              |
| e508: 66 94    | ror \$94     | [ c3p0 ]  | Fissa flag per byte.bufferiz.       |
| e50a: d0 05    | bne * \$e511 |           | Salta a uscita ciclo                |
| e50c: 48       | pha          |           | Salva il Byte sullo Stack           |
| e50d: 20 8c e3 | jsr * \$e38c |           | Metti Byte buff. in uscita su Stack |
| e510: 68       | pla          |           | Prel. byte da Stack                 |
| e511: 85 95    | sta \$95     | [ bsour ] | Metti in pag 0 quan. immag.         |
| e513: 18       | clc          |           | Esegui Clear del carry              |
| e514: 60       | rts          |           |                                     |

## ROUTINE KERNAL UNTLK &amp; UNLSN

|                |              |  |                                       |
|----------------|--------------|--|---------------------------------------|
| e515: 20 73 e5 | jsr * \$e573 |  | Reset d. freq. clock                  |
| e518: 20 4e e5 | jsr * \$e54e |  | Segnale di clock basso su porta A     |
| e51b: ad 00 dd | lda * \$dd00 |  | Lett. dati porta A su CIA 2           |
| e51e: 09 08    | ora #\$08    |  | Metti bit 3 a questo valore e         |
| e520: 8d 00 dd | sta * \$dd00 |  | Uscita segn.ATN LO su bus             |
| e523: a9 5f    | lda #\$5f    |  | Carica cod. in A per UNTALK           |
| e525: 2c       | .byte \$2c   |  | Vai a \$e528                          |
| e526: a9 3f    | lda #\$3f    |  | Carica codice per UNLSN               |
| e528: 48       | pha          |  | Immagazz. q.s. in stack               |
| e529: ad 1c 0a | lda \$0alc   |  | Punt. di STATUS per seriale modo FAST |
| e52c: 29 7f    | and #\$7f    |  | Mask out del bit 7                    |
| e52e: 8d 1c 0a | sta \$0alc   |  | Riscrivi                              |
| e531: 68       | pla          |  | Ripristina vecchi cont. A             |
| e532: 20 43 e3 | jsr * \$e343 |  | Vai a subr. LISTN                     |
| e535: 20 d7 e4 | jsr * \$e4d7 |  | Esegui reset ATN (HI)                 |
| e538: 8a       | txa          |  | Immag.cont.X in A                     |
| e539: a2 0a    | ldx #\$0a    |  | Ciclo di temp. di 40 microsec.        |
| e53b: ca       | dex          |  | Decr. di 1 cont. ciclo                |
| e53c: d0 fd    | bne * \$e53b |  | Attendi per fine ciclo                |
| e53e: aa       | tax          |  | Ripr. cont. X                         |
| e53f: 20 45 e5 | jsr * \$e545 |  | In.segn. HI di clock su porta A       |
| e542: 4c 57 e5 | jmp * \$e557 |  | Segn. dati alto su porta A            |
| e545: ad 00 dd | lda * \$dd00 |  | Leggi dati porta A di CIA 2           |
| e548: 29 ef    | and #\$ef    |  | Esegui un clear su bit 4 per uscita   |
| e54a: 8d 00 dd | sta * \$dd00 |  | clock su bus ser.e scr. su porta A    |
| e54d: 60       | rts          |  |                                       |

|                |              |  |                                     |
|----------------|--------------|--|-------------------------------------|
| e54e: ad 00 dd | lda * \$dd00 |  | Leggi dati porta A di CIA 2         |
| e551: 09 10    | ora #\$10    |  | Fissa bit 4 per uscita clock su bus |
| e553: 8d 00 dd | sta * \$dd00 |  | seriale e scr. su porta A           |
| e556: 60       | rts          |  |                                     |

|                |              |  |                                      |
|----------------|--------------|--|--------------------------------------|
| e557: ad 00 dd | lda * \$dd00 |  | In queste due subr. vengono eseguite |
| e55a: 29 df    | and #\$df    |  | le operazioni prec, sequenzialmente  |
| e55c: 8d 00 dd | sta * \$dd00 |  | ma per bit 5 anziche 4               |
| e55f: 60       | rts          |  |                                      |

|                |              |      |
|----------------|--------------|------|
| e560: ad 00 dd | lda * \$dd00 | idem |
| e563: 09 20    | ora #\$20    | idem |
| e565: 8d 00 dd | sta * \$dd00 | idem |
| e568: 60       | rts          |      |

## Il Sistema Operativo del Commodore 128

|                |              |                                      |
|----------------|--------------|--------------------------------------|
| e5bc: ad 0d dc | lda * \$dc0d | Carica A reg.CIA contr. interrupt    |
| e5bf: 29 08    | and #\$08    | Attendi fino a clear di bit 4 che e' |
| e5c1: f0 f9    | beq * \$e5bc | SQR                                  |
| e5c3: ad 0e dc | lda * \$dc0e | Leggi reg. contr. A di CIA           |
| e5c6: 29 80    | and #\$80    | Canc. bit 7 per freq. 50 Hz          |
| e5c8: 09 08    | ora #\$08    | Fissa timer su modo TOGGLE           |
| e5ca: 8d 0e dc | sta * \$dc0e | Pai partire il timer                 |
| e5cd: a        |              |                                      |

```

e569: ad 00 dd  lda * $dd00      Leggi dati porta A di CIA 2 ed attendi
e56c: cd 00 dd  cmp * $dd00      fino a quando un bit giunga alla
e56f: d0 f8     bne * $e569      porta A
e571: 0a        asl a           Il bit ricev. in Carry
e572: 60        rts

```

```

e573: 78        sei            Disab. tutti gli interrupt
e574: 2c 3a 0a  bit $0a3a      Contr. immag. interrupt
e577: 30 25     bmi * $e59e      Fissa bit 7
e579: 2c 37 0a  bit $0a37      Controllo freq. clock
e57c: 30 20     bmi * $e59e      Fissa bit 7
e57e: ad 30 d0  lda * $d030      Car. reg. VIC per freq. clock
e581: 8d 37 0a  sta $0a37      Salva q.s.
e584: ad 15 d0  lda * $d015      Abilita reg. VIC per sprites
e587: 8d 38 0a  sta $0a38      Salva q.s.
e58a: a9 00     lda #$00        Iniz. STATUS per funz. 1 MHz
e58c: 8d 15 d0  sta * $d015      Dis. tutti sprites
e58f: 8d 30 d0  sta * $d030      Fissa freq. clock a 1 MHz
e592: ad 38 0a  lda $0a38      Contr. sprites attivi
e595: f0 07     beq * $e59e      Se neg. vai ad ind.
e597: 8a        txa            Immag. X in A
e598: a2 00     ldx #$00        Ciclo di rit di 1.3 millisec
e59a: ca        dex            Decr. di 1 cont. ciclo
e59b: d0 fd     bne * $e59a      Esegui tutto ciclo di rit.
e59d: aa        tax            Riprist. vecchio cont X
e59e: 60        rts

```

```

e59f: 2c 3a 0a  bit $0a3a      Contr. immag. interrupt
e5a2: 30 16     bmi * $e5ba      Fissa bit 7
e5a4: 2c 37 0a  bit $0a37      Contro. freq. di clock immagaz.
e5a7: 10 11     bpl * $e5ba      Se freq. non cambiata, salta
e5a9: ad 38 0a  lda $0a38      Scrivi val. immag. dello sprite
e5ac: 8d 15 d0  sta * $d015      Riabilita reg.
e5af: ad 37 0a  lda $0a37      Carica A con val. di sist. immag.
e5b2: 8d 30 d0  sta * $d030      Ripr. freq. clock
e5b5: a9 00     lda #$00        Esegui clear della freq. clock di sis.
e5b7: 8d 37 0a  sta $0a37      per immag. temp.
e5ba: 58        cli            Abilita tutti gli interrupt
e5bb: 60        rts

```

```

e5bc: ad 0d dc  lda * $dc0d      Carica A reg. CIA contr. interrupt
e5bf: 29 08     and #$08        Attendi fino a clear di bit 4 che e'
e5c1: f0 f9     beq * $e5bc      SQR
e5c3: ad 0e dc  lda * $dc0e      Leggi reg. contr. A di CIA
e5c6: 29 80     and #$80        Canc. bit 7 per freq. 50 Hz
e5c8: 09 08     ora #$08        Fissa timer su modo TOGGLE
e5ca: 8d 0e dc  sta * $dc0e      Fai partire il timer
e5cd: ad 05 d5  lda * $d505      Esegui un MASK out di bit di controllo
e5d0: 29 f7     and #$f7        per modo seriale FAST nel reg. conf.
e5d2: 8d 05 d5  sta * $d505      di MMU
e5d5: 60        rts

```

```

e5d6: ad 05 d5  lda * $d505      Fissa bit di controllo per modo seri.
e5d9: 09 08     ora #$08        FAST nel reg. di conf. modo di

```

|                |              |  |
|----------------|--------------|--|
| e5db: 8d 05 d5 | sta * \$d505 | MMU                                    |
| e5de: a9 7f    | lda #\$7f    | Esegui un clear cod. per interrupt     |
| e5e0: 8d 0d dc | sta * \$dc0d | Interrupt su reg di contr.             |
| e5e3: a9 00    | lda #\$00    | Carica timer A HI del CIA2 con         |
| e5e5: 8d 05 dc | sta * \$dc05 | val. HI #0                             |
| e5e8: a9 04    | lda #\$04    | Come sopra per LO                      |
| e5ea: 8d 04 dc | sta * \$dc04 | per val. LO #4                         |
| e5ed: ad 0e dc | lda * \$dc0e | Leggi reg. contr A di CIA 2            |
| e5f0: 29 80    | and #\$80    | Canc. bit 7 per freq 50 HZ             |
| e5f2: 09 55    | ora #\$55    | Fissa timer per forz. caricam.         |
| e5f4: 8d 0e dc | sta * \$dc0e | Metti in OFF bus seriale.Fai part.tim. |
| e5f7: 2c 0d dc | bit \$dcod   | Let. reg. contr. interrupt             |
| e5fa: 60       | rts          |  |

-----  
ROUTINE KERNAL FSTMOD

|             |              |                                     |
|-------------|--------------|-------------------------------------|
| e5fb: 90 c6 | bcc * \$e5c3 | Att. risp.da bus seriale            |
| e5fd: b0 d7 | bcs * \$e5d6 | Invia imp. FAST su bus seriale      |
| e5ff: a5 b4 | lda \$b4     | N. di bits da inviare               |
| e601: f0 47 | beq * \$e64a | Contr. per Byte compl. trasf.       |
| e603: 30 3f | bmi * \$e644 | Contr.rich. bit di stop             |
| e605: 46 b6 | lsr \$b6     | Sposta bit succ. su Carry           |
| e607: a2 00 | ldx #\$00    | Iniz. reg. X con \$00               |
| e609: 90 01 | bcc * \$e60c | Contr. esec. clear del bit          |
| e60b: ca    | dex          | Se neg. fissa X a \$FF              |
| e60c: 8a    | txa          | Copia in A Bit prec                 |
| e60d: 45 bd | eor \$bd     | Confr. con parita'                  |
| e60f: 85 bd | sta \$bd     | Risalva su pag. zero                |
| e611: c6 b4 | dec \$b4     | Decr. cont. bit di l                |
| e613: f0 06 | beq * \$e61b | Contr. trasf. eseguito su tutti bit |
| e615: 8a    | txa          | Trasf X in A                        |
| e616: 29 04 | and #\$04    | Prel. bit 2                         |
| e618: 85 b5 | sta \$b5     | Trasf. bit 2 in reg. uscita         |
| e61a: 60    | rts          |                                     |

-----  
CONTROLLO PARITA' IN TRASMISSIONE

|                |              |                                  |
|----------------|--------------|----------------------------------|
| e61b: a9 20    | lda #\$20    | Metti bit 5 in A per parita'     |
| e61d: 2c 11 0a | bit \$0a11   | Contr. reg. com. RS-232          |
| e620: f0 14    | beq * \$e636 | Se senza parita' salta           |
| e622: 30 1c    | bmi * \$e640 | Contr. parita' fissata           |
| e624: 70 14    | bvs * \$e63a | Contr. nessuna parita'           |
| e626: a5 bd    | lda \$bd     | Contr. parita' 1                 |
| e628: d0 01    | bne * \$e62b | Contr. neg. salta                |
| e62a: ca       | dex          | Fissa parita' a \$ff             |
| e62b: c6 b4    | dec \$b4     | Fissa cont. bit a \$FF           |
| e62d: ad 10 0a | lda \$0a10   | Metti in A reg contr. RS-232     |
| e630: 10 e3    | bpl * \$e615 | Contr.se rich. 2 bits di stop    |
| e632: c6 b4    | dec \$b4     | Metti a \$FE cont. bit           |
| e634: d0 df    | bne * \$e615 | Se non= 0 calc. i bits di stop   |
| e636: e6 b4    | inc \$b4     | Incr. di l cont. bit             |
| e638: d0 f0    | bne * \$e62a | C.s.                             |
| e63a: a5 bd    | lda \$bd     | Prendi val. di parita' da pag. 0 |
| e63c: f0 ed    | beq * \$e62b | Uscita bit ZERO per 0            |
| e63e: d0 ea    | bne * \$e62a | Ness. 0, allora uscita bit       |
| e640: 70 e9    | bvs * \$e62b | Rout. di uscita 0 bit            |
| e642: 50 e6    | bvc * \$e62a | Come sopra per bit 1             |
| e644: e6 b4    | inc \$b4     | Incr.cont.bit di l               |
| e646: a2 ff    | ldx #\$ff    | Immetti in X cod.per bit di stop |
| e648: d0 cb    | bne * \$e615 | Slato incond.                    |

|                |              |            |  |
|----------------|--------------|------------|--|
| e64a: ad 11 0a | lda \$0a11   |            | Metti in A reg.com. di RS-232          |
| e64d: 4a       | lsr a        |            | Metti bit 0 in carry                   |
| e64e: 90 07    | bcc * \$e657 |            | Salta 3 linne per lett. HANDSHAKE      |
| e650: 2c 01 dd | bit * \$dd01 |            | Leggi porta B di CIA 2                 |
| e653: 10 1d    | bpl * \$e672 |            | Contro.perdita segn.DSR(DATA SET READ) |
| e655: 50 1e    | bvc * \$e675 |            | Contr. perd. segn.CTS (CLEAR TO SEND)  |
| e657: a9 00    | lda #\$00    |            | Es.clear buff.Pag.0 per RS-232         |
| e659: 85 bd    | sta \$bd     | [ roprty ] | Metti parita' a (\$00) ed immag in Pag |
| e65b: 85 b5    | sta \$b5     | [ diff ]   | 0 per bit da inviare                   |
| e65d: ae 15 0a | ldx \$0a15   |            | Copia n. di bits da trasf. in Pag. 0   |
| e660: 86 b4    | stx \$b4     | [ bitts ]  | come contatore                         |
| e662: ac 1a 0a | ldy \$0a1a   |            | Confr. ind. di inizio buffer uscita    |
| e665: cc 1b 0a | cpy \$0a1b   |            | con relativa fine                      |
| e668: f0 13    | beq * \$e67d |            | Se tutti bytes trasf. esegui           |
| e66a: b1 ca    | lda (\$ca),y | [ robuf ]  | Carica in A byte di dati per RS-232    |
| e66c: 85 b6    | sta \$b6     | [ prp ]    | e immagazzina                          |
| e66e: ee 1a 0a | inc \$0a1a   |            | Incr. di 1 inizio buff. uscita dati    |
| e671: 60'      | rts          |            |  |

## FISSAGGIO DI NMI PER RS-232

|                |              |  |                                    |
|----------------|--------------|--|------------------------------------|
| e672: a9 40    | lda #\$40    |  | Carica codice per DATA SET READY   |
| e674: 2c       | .byte \$2c   |  | Vai a \$e677                       |
| e677: 0d 14 0a | ora \$0a14   |  | Confr.con reg di status di RS-232  |
| e67a: 8d 14 0a | sta \$0a14   |  | Immetti in reg. di status          |
| e67d: a9 01    | lda \$01     |  | Carica A con \$01 ed es. clear di  |
| e67f: 8d 0d dd | sta * \$dd0d |  | NMI per il timer A                 |
| e682: 4d 0f 0a | eor \$0a0f   |  | Confr. con NMI di RS-232           |
| e685: 09 80    | ora #\$80    |  | Es. OR su flag per RS-232 ed imme. |
| e687: 8d 0f 0a | sta \$0a0f   |  | il valore in NMI di RS-232         |
| e68a: 8d 0d dd | sta * \$dd0d |  | Ese. i succ. NMI                   |
| e68d: 60       | rts          |  |                                    |

## CALCOLO PER RS-232

|                |              |  |                              |
|----------------|--------------|--|------------------------------|
| e68e: a2 09    | ldx #\$09    |  | Valore di default a 8 bits   |
| e690: a9 20    | lda #\$20    |  | Contr. val. per n. bits      |
| e692: 2c 10 0a | bit \$0a10   |  | Contr. reg. RS-232           |
| e695: f0 01    | beq * \$e698 |  | Contr. se es. clear su bit 5 |
| e697: ca       | dex          |  | Decr. di 1 n. bits dati      |
| e698: 50 02    | bvc * \$e69c |  | Contr. se es. clear su bit 6 |
| e69a: ca       | dex          |  | Decr. di 1 bits dati         |
| e69b: ca       | dex          |  | C.s.                         |
| e69c: 60       | rts          |  |                              |

|             |              |            |                                  |
|-------------|--------------|------------|----------------------------------|
| e69d: a6 a9 | ldx \$a9     | [ rinone ] | Contr. se e' bit di inizio       |
| e69f: d0 33 | bne * \$e6d4 |            | Se contr. neg. salta             |
| e6a1: c6 a8 | dec \$a8     | [ bitci ]  | Decr. cont. bit di 1             |
| e6a3: f0 3a | beq * \$e6df |            | Se tutti bits ric. continua      |
| e6a5: 30 0d | bmi * \$e6b4 |            | Se attend. bits di stop(2) salta |
| e6a7: a5 a7 | lda \$a7     | [ shcnl ]  | Metti in A bit ricevuto          |
| e6a9: 45 ab | eor \$ab     | [ riprty ] | Confr. q.s. con parita'          |
| e6ab: 85 ab | sta \$ab     | [ riprty ] | Immetti val. parita' in pag. 0   |
| e6ad: 46 a7 | lsr \$a7     | [ shcnl ]  | Sposta bit ric. in carry         |
| e6af: 66 aa | ror \$aa     | [ rdflg ]  | Immetti q.s. in buffer ingresso  |
| e6b1: 60    | rts          |            |                                  |

|             |          |           |                      |
|-------------|----------|-----------|----------------------|
| e6b2: c6 a8 | dec \$a8 | [ bitci ] | Decr. di 1 cont. bit |
|-------------|----------|-----------|----------------------|

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| e6b4: a5 a7    | lda \$a7     | [ shcnl ]  | Metti bit di stop in A e controlla    |
| e6b6: f0 6b    | beq * \$e723 |            | che =0. Se contr. pos. salta          |
| e6b8: ad 10 0a | lda \$0a10   |            | Carica in A reg. contr. RS-232        |
| e6bb: 0a       | asl a        |            | Metti su carry n.bits di STOP         |
| e6bc: a9 01    | lda #\$01    |            | Somma val.n.bits di stop              |
| e6be: 65 a8    | adc \$a8     | [ bitci ]  | Somma bits di dati e di stop          |
| e6c0: d0 ef    | bne * \$e6b1 |            | Se non ric. tutti bits stop salta     |
| e6c2: a9 90    | lda #\$90    |            | Metti in A RXD                        |
| e6c4: 8d 0d dd | sta * \$dd0d |            | Abilita NMI                           |
| e6c7: 0d 0f 0a | ora \$0a0f   |            | Confr. con NMI di RS-232              |
| e6ca: 8d 0f 0a | sta \$0a0f   |            | Metti in A stato di NMI di RS-232     |
| e6cd: 85 a9    | sta \$a9     | [ rinone ] | Fissa flag per bit parten.            |
| e6cf: a9 02    | lda #\$02    |            | Metti 2 in A per transmiss.           |
| e6d1: 4c 7f e6 | jmp * \$e67f |            | Esegui clear di NMI(per timer B)      |
| e6d4: a5 a7    | lda \$a7     | [ shcnl ]  | Metti in A val.bit di part.           |
| e6d6: d0 ea    | bne * \$e6c2 |            | Se non=0 salta. Ese. quindi il reset  |
| e6d8: 85 a9    | sta \$a9     | [ rinone ] | bit part. di pag. zero e reset anche  |
| e6da: a9 01    | lda #\$01    |            | del punt. sempre in pag. 0 per uscita |
| e6dc: 85 ab    | sta \$ab     | [ riprty ] | RS-232. Reset su parita' ingr.        |
| e6de: 60       | rts          |            |                                       |

-----  
CONTINUAZIONE ROUTINE RS-232

|                |              |            |  |
|----------------|--------------|------------|--|
| e6df: ac 18 0a | ldy \$0a18   |            | Ind. ini. di RS-232                    |
| e6e2: c8       | iny          |            | Incr. di 1 buffer di input             |
| e6e3: cc 19 0a | cpy \$0a19   |            | Confr. con fine                        |
| e6e6: f0 2a    | beq * \$e712 |            | Fissa status del buffer                |
| e6e8: 8c 18 0a | sty \$0a18   |            | Scrivi buffer ind.                     |
| e6eb: 88       | dey          |            | Decr. di 1                             |
| e6ec: a5 aa    | lda \$aa     | [ rdflg ]  | Carica bit ric. da pag. 0              |
| e6ee: ae 15 0a | ldx \$0a15   |            | Metti in X n. di bits dati             |
| e6f1: e0 09    | cpx #\$09    |            | Contr. ric 1 bit di stop               |
| e6f3: f0 04    | beq * \$e6f9 |            | Se contr. pos. tutto OK                |
| e6f5: 4a       | lsr a        |            | Sposta bits in pos. corretta           |
| e6f6: e8       | inx          |            | Increment. bit di 1                    |
| e6f7: d0 f8    | bne * \$e6f1 |            | Vai a ind.                             |
| e6f9: 91 c8    | sta (\$c8),y | [ ribuf ]  | Scrivi byte(daind. prec)in buffer ingr |
| e6fb: a9 20    | lda #\$20    |            | Contr. val. per contr. parita'         |
| e6fd: 2c 11 0a | bit \$0a11   |            | Contr. reg comando RS-232              |
| e700: f0 b0    | beq * \$e6b2 |            | Trasf se ness. parita'                 |
| e702: 30 ad    | bmi * \$e6b1 |            | Fis. valore di bit per parita'         |
| e704: a5 a7    | lda \$a7     | [ shcnl ]  | Bit di parita'7 ric. in A              |
| e706: 45 ab    | eor \$ab     | [ riprty ] | Confr. con parita' calcolata           |
| e708: f0 03    | beq * \$e70d |            | Se contr. pos. continua                |
| e70a: 70 a5    | bvs * \$e6b1 |            | Contr. parita', se pos. OK             |
| e70c: 2c       | .byte \$2c   |            | Vai a \$e714                           |
| e70d: 50 a2    | bvc * \$e6b1 |            | Controllo parita'                      |
| e70f: a9 01    | lda #\$01    |            | Metti in A cod. err. parita'           |
| e711: 2c       | .byte \$2c   |            | Vai a \$e714                           |
| e712: a9 a4    | lda #\$04    |            | Buffer di input in A                   |
| e714: 2c       | .byte \$2c   |            | Vai a \$e717                           |
| e715: a9 80    | lda #\$80    |            | Se ricev.com. di stop allora in A      |
| e717: 2c       | .byte \$2c   |            | Vai a \$e71a                           |
| e718: a9 02    | lda #\$02    |            | Carica cod.errore in A                 |
| e71a: 0d 14 0a | ora \$0a14   |            | Es. OR cod. con reg.stato di RS-232    |
| e71d: 8d 14 0a | sta \$0a14   |            | Metti q.s. in reg.sta. RS-232          |
| e720: 4c c2 e6 | jmp * \$e6c2 |            | Ricevi succ. byte                      |
| e723: a5 aa    | lda \$aa     | [ rdflg ]  | Metti in A byte ricevuto               |
| e725: d0 f1    | bne * \$e718 |            | Contr. errore                          |
| e727: f0 ec    | beq * \$e715 |            | Ferma com. ricevuto                    |



|       |          |              |           |                                    |
|-------|----------|--------------|-----------|------------------------------------|
| e729: | 85 9a    | sta \$9a     | [ dflto ] | Immetti in pag. 0 n. perif         |
| e72b: | ad 11 0a | lda \$0all   |           | Car. RS-232 (reg. comando)         |
| e72e: | 4a       | lsr a        |           | Sposta bit 0 in carry              |
| e72f: | 90 29    | bcc * \$e75a |           | Salta per 3 linee di HANDSHAKE     |
| e731: | a9 02    | lda #\$02    |           | Car. A cod'contr. DSR              |
| e733: | 2c 01 dd | bit * \$dd01 |           | Lettura porta B di CIA 2           |
| e736: | 10 1d    | bpl * \$e755 |           | Se nessun DSR errore               |
| e738: | d0 20    | bne * \$e75a |           | Nessun segn. di invio dati         |
| e73a: | ad 0f 0a | lda \$0a0f   |           | Metti in A stato NMI di RS-232     |
| e73d: | 29 02    | and #\$02    |           | Contr. ric. dati attivo            |
| e73f: | d0 f9    | bne * \$e73a |           | Attesa per ricez. avvenuta         |
| e741: | 2c 01 dd | bit * \$dd01 |           | Leggi porta B di CIA 2             |
| e744: | 70 fb    | bvs * \$e741 |           | Attendi per segn. di Clear To Send |
| e746: | ad 01 dd | lda * \$dd01 |           | Leggi porta B CIA 2                |
| e749: | 09 02    | ora #\$02    |           | Fissa bit 2 per segnale prec.      |
| e74b: | 8d 01 dd | sta * \$dd01 |           | Scrivi su porta B                  |
| e74e: | 2c 01 dd | bit * \$dd01 |           | Leggi porta B di CIA 2             |
| e751: | 70 07    | bvs * \$e75a |           | Attesa segnale Clear To Send       |
| e753: | 30 f9    | bmi * \$e74e |           | Atti. DATA READY                   |
| e755: | a9 40    | lda #\$40    |           | Codice segn. prec perd.            |
| e757: | 8d 14 0a | sta \$0a14   |           | Scrivi segn. su RS-232             |
| e75a: | 18       | clc          |           | Fissa carry per indic. tutto OK    |
| e75b: | 60       | rts          |           |                                    |

-----  
USCITA SU BUFFER RS-232

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| e75c: | 20 70 e7 | jsr * \$e770 |           | Inizio trasf.                         |
| e75f: | ac 1b 0a | ldy \$0alb   |           | Indic. buffer uscita RS-232           |
| e762: | c8       | iny          |           | Incr. buffer di 1                     |
| e763: | cc 1a 0a | cpy \$0ala   |           | Confr. con inizio buffer uscita       |
| e766: | f0 f4    | beq * \$e75c |           | Se buffer pieno attendi               |
| e768: | 8c 1b 0a | sty \$0alb   |           | Fissa nuovo ind. su buffer uscita     |
| e76b: | 88       | dey          |           | Decrem.punt. buffer di 1              |
| e76c: | a5 9e    | lda \$9e     | [ ptr1 ]  | Byte di uscita in A                   |
| e76e: | 91 ca    | sta (\$ca),y | [ robuf ] | Scrivilo in buffer di uscita          |
| e770: | ad 0f 0a | lda \$0a0f   |           | Copia flag di NMI di RS-232 in A      |
| e773: | 4a       | lsr a        |           | Controlla se bit 0 e' a 1             |
| e774: | b0 1e    | bcs * \$e794 |           | Contr. cont. invio dati               |
| e776: | a9 10    | lda #\$10    |           | Iniz. timer A con \$10                |
| e778: | 8d 0e dd | sta * \$dd0e |           | Carica in A e fallo partire           |
| e77b: | ad 16 0a | lda \$0a16   |           | Rout. per fissare il BAUD RATE, cioe' |
| e77e: | 8d 04 dd | sta * \$dd04 |           | la vel. di trasmissione agli indiri.  |
| e781: | ad 17 0a | lda \$0a17   |           | \$DD04 e                              |
| e784: | 8d 05 dd | sta * \$dd05 |           | \$DD05                                |
| e787: | a9 81    | lda #\$81    |           | Cod. timer A in Underflow di NMI      |
| e789: | 20 7f e6 | jsr * \$e67f |           | Esegui NMI per timer A                |
| e78c: | 20 4a e6 | jsr * \$e64a |           | Contr. CTS+DSR, abil. trasf.          |
| e78f: | a9 11    | lda #\$11    |           | Timer A a \$11                        |
| e791: | 8d 0e dd | sta * \$dd0e |           | Partenza timer                        |
| e794: | 60       | rts          |           |                                       |

-----  
CHKIN DI RS-232

|       |          |              |           |                                  |
|-------|----------|--------------|-----------|----------------------------------|
| e795: | 85 99    | sta \$99     | [ dfltn ] | Immetti n. perif. in pag. 0      |
| e797: | ad 11 0a | lda \$0all   |           | Metti in A reg. com. RS-232      |
| e79a: | 4a       | lsr a        |           | Bit 0 per HANDSHAKE in carry     |
| e79b: | 90 28    | bcc * \$e7c5 |           | Es. 3 linee di HANDSHAKE e cont. |
| e79d: | 29 08    | and #\$08    |           | Contr. per duplex                |
| e79f: | f0 24    | beq * \$e7c5 |           | Pos. full duplex e cont.         |
| e7a1: | a9 02    | lda #\$02    |           | Cod. contr. segn. DSR            |

```

e7a3: 2c 01 dd bit * $dd01
e7a6: 10 ad bpl * $e755
e7a8: f0 22 beq * $e7cc
e7aa: ad 0f 0a lda $0a0f
e7ad: 4a lsr a
e7ae: b0 fa bcs * $e7aa
e7b0: ad 01 dd lda * $dd01
e7b3: 29 fd and #$fd
e7b5: 8d 01 dd sta * $dd01
e7b8: ad 01 dd lda * $dd01
e7bb: 29 04 and #$04
e7bd: f0 f9 beq * $e7b8
e7bf: a9 90 lda #$90
e7c1: 18 clc
e7c2: 4c 7f e6 jmp * $e67f
e7c5: ad 0f 0a lda $0a0f
e7c8: 29 12 and #$12
e7ca: f0 f3 beq * $e7bf
e7cc: 18 clc
e7cd: 60 rts

```

Contr. per DSR di porta B CIA 2  
 Se perd. segnale fissa status esci  
 Contr. segn Ready To Send  
 Metti in A flag di st.di RS-232 NMI  
 Se inviato operazione in corso e po  
 attendi fine trasferimento  
 Lett. porta A di CIA 2. Canc. bit 0  
 Richiesta di invio segnale  
 Segn. di ritorno su porta B  
 Lett. porta B di CIA 2 e controlla  
 pres. segnale DTR  
 Se segnale prec. non pres. attendi  
 Metti in A NMI (mask) per flag  
 Es. clear di carry  
 Abil. NMI di RS-232  
 Metti in A st. NMI di RS-232  
 Contr. se RS-232 non ancora attiva  
 e quindi partenza  
 Es. clear di carry

## GET DA RS-232

```

e7ce: ad 14 0a lda $0a14
e7d1: ac 19 0a ldy $0a19
e7d4: cc 18 0a cpy $0a18
e7d7: f0 0b beq * $e7e4
e7d9: 29 f7 and #$f7
e7db: 8d 14 0a sta $0a14
e7de: b1 c8 lda ($c8),y [ ribuf ]
e7e0: ee 19 0a inc $0a19
e7e3: 60 rts

```

Metti in A byte di status di RS-232  
 Ind. fine buffer input di RS-232  
 Confr. con inizio buffer input  
 Se = buffer pieno e salta  
 Mask out bit 3  
 Clear status di RS-232  
 Lett. 1 byte da buffer RS-232  
 Incr. ind. buffer input di RS-232

## CONTROLLO BUFFER PER RS-232

```

e7e4: 09 08 ora #$08
e7e6: 8d 14 0a sta $0a14
e7e9: a9 00 lda #$00
e7eb: 60 rts

```

Fissa bit 3  
 Metti in A st. di RS-232  
 Car. \$00 come car. letto

## ROUTINE DI ATTESA PER FINE RS-232

```

e7ec: 48 pha
e7ed: ad 0f 0a lda $0a0f
e7f0: f0 11 beq * $e803
e7f2: ad 0f 0a lda $0a0f
e7f5: 29 03 and #$03
e7f7: d0 f9 bne * $e7f2
e7f9: a9 10 lda #$10
e7fb: 8d 0d dd sta * $dd0d
e7fe: a9 00 lda #$00
e800: 8d 0f 0a sta $0a0f
e803: 68 pla
e804: 60 rts

```

Salva su Stack cont. di A  
 Carica flag NMI di RS-232  
 Se non =1 OK e continua  
 Rileggi flag NMI di RS-232  
 BIT 0 per inviare, bit 1 per ricevere  
 Attesa per fine  
 Carica A con \$10  
 Linea di interrupt  
 Flag NMI di RS-232  
 Fissa Status per OK  
 Ripris. cont. A

## ROUTINE NMI PER RS-232

```

e805: 98 tya
e806: 2d 0f 0a and $0a0f
e809: aa tax

```

ICR= reg. controllo interrupt  
 Es. AND con flag NMI di RS-232  
 Immag. ris. in X

|                |                   |                                       |
|----------------|-------------------|---------------------------------------|
| e80a: 29 01    | and #01           | Es. mask dei bits 1-7 e contr. se op. |
| e80c: f0 28    | beq * \$e836      | invio attiva. Se neg. salta           |
| e80e: ad 00 dd | lda * \$dd00      | Carica A con dati porta               |
| e811: 29 fb    | and #fbb          | Es. clear bit 2 (TXD) e poni il bit   |
| e813: 05 b5    | ora \$b5 [ diff ] | da inviare in A                       |
| e815: 8d 00 dd | sta * \$dd00      | Carica A con dati porta               |
| e818: ad 0f 0a | lda \$0a0f        | Copia in A flag NMI di RS-232         |
| e81b: 8d 0d dd | sta * \$dd0d      | Riscrivi q.s. in ICR                  |
| e81e: 8a       | txa               | Trasf. in A ICR/NMI                   |
| e81f: 29 12    | and #12           | Prendi bit 1 e 4                      |
| e821: f0 0d    | beq * \$e830      | Se non = 1 part. byte ricezione       |
| e823: 29 02    | and #02           | Es. AND log. su bit 1. Timer B        |
| e825: f0 06    | beq * \$e82d      | Se non fissato, bit inizio            |
| e827: 20 78 e8 | jsr * \$e878      | Integ. rice. bit                      |
| e82a: 4c 30 e8 | jmp * \$e830      | Iniz. ricev. byte                     |
| e82d: 20 a9 e8 | jsr * \$e8a9      | Prep. ricevz. byte successivo         |
| e830: 20 ff e5 | jsr * \$e5ff      | Inizio ricez. byte                    |
| e833: 4c 49 e8 | jmp * \$e849      | Return da interrupt                   |
| e836: 8a       | txa               | Immetti X in A                        |
| e837: 29 02    | and #02           | Es. AND per ric. dati e contr.        |
| e839: f0 06    | beq * \$e841      | Se contr. neg. salta                  |
| e83b: 20 78 e8 | jsr * \$e878      | L' integr. ric. il bit                |
| e83e: 4c 49 e8 | jmp * \$e849      | Ritorno da fase di Interrupt          |
| e841: 8a       | txa               | Ripris. cont. X                       |
| e842: 29 10    | and #10           | Contr. attesa bit di partenza         |
| e844: f0 03    | beq * \$e849      | Se neg. continua                      |
| e846: 20 a9 e8 | jsr * \$e8a9      | Prep. rice. pross. bit                |
| e849: ad 0f 0a | lda \$0a0f        | Carica A flag NMI di RS-232           |
| e84c: 8d 0d dd | sta * \$dd0d      | Metti in ICR di CIA 2 q.s             |
| e84f: 60       | rts               |                                       |

-----  
 COSTANTI RS-232 PER BUAD RATE. VERS NTSC

|             |          |       |      |
|-------------|----------|-------|------|
| e850: c1 27 | (\$27c1) | 50    | BAUD |
| e852: 3e 1a | (\$1a3e) | 75    | "    |
| e854: c5 11 | (\$11c5) | 110   | "    |
| e856: 74 0e | (\$0e74) | 134.5 | "    |
| e858: ed 0c | (\$0ced) | 150   | "    |
| e85a: 45 06 | (\$0645) | 300   | "    |
| e85c: f0 02 | (\$02f0) | 600   | "    |
| e85e: 46 01 | (\$0146) | 1200  | "    |
| e860: b8 00 | (\$00b8) | 1800  | "    |
| e862: 71 00 | (\$0071) | 2400  | "    |

-----  
 COSTANTI RS-232 PER BAUD RATE. VERS. PAL

|             |          |       |      |
|-------------|----------|-------|------|
| e864: 19 26 | (\$2619) | 50    | BAUD |
| e866: 44 19 | (\$1944) | 75    | "    |
| e868: 1a 11 | (\$111a) | 110   | "    |
| e86a: e8 0d | (\$0de8) | 134.5 | "    |
| e86c: 70 0c | (\$0c70) | 150   | "    |
| e86e: 06 06 | (\$0606) | 300   | "    |
| e870: d1 02 | (\$02d1) | 600   | "    |
| e872: 37 01 | (\$0137) | 1200  | "    |
| e874: ae 00 | (\$00ae) | 1800  | "    |
| e876: 69 00 | (\$0069) | 2400  | "    |

-----  
 INGRESSO ROUTINE NMI PER RS-232

|                |              |                             |
|----------------|--------------|-----------------------------|
| e878: ad 01 dd | lda * \$dd01 | Leggi dati porta B di CIA 2 |
|----------------|--------------|-----------------------------|

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| e87b: | 29 01    | and #01      |           | Prep. bit per ricez. dati             |
| e87d: | 85 a7    | sta \$a7     | [ shcnl ] | Immag.pag 0 flag input di RS-232      |
| e87f: | ad 06 dd | lda * \$dd06 |           | Car. LO di timer B CIA2               |
| e882: | e9 28    | sbc #28      |           | Sottrai da questo 28                  |
| e884: | 6d 16 0a | adc \$0a16   |           | Somma temp.baud rate HI               |
| e887: | 8d 06 dd | sta * \$dd06 |           | Reset timer B                         |
| e88a: | ad 07 dd | lda * \$dd07 |           | Car. HI di timer B CIA 2              |
| e88d: | 6d 17 0a | adc \$0a17   |           | Somma temp.baud rate HI               |
| e890: | 8d 07 dd | sta * \$dd07 |           | Reset timer B                         |
| e893: | a9 11    | lda #11      |           | Scrivi 11 nel reg. di contr.          |
| e895: | 8d 0f dd | sta * \$dd0f |           | del timer B CIA 2 per part.           |
| e898: | ad 0f 0a | lda \$0a0f   |           | Metti in A condiz. NMI di RS-232      |
| e89b: | 8d 0d dd | sta * \$dd0d |           | Fissa reg.contr. interrupt CIA        |
| e89e: | a9 ff    | lda #fff     |           | Val. iniz. per timer B                |
| e8a0: | 8d 06 dd | sta * \$dd06 |           | Timer B LO a valore HI                |
| e8a3: | 8d 07 dd | sta * \$dd07 |           | Timer B HI a valore HI                |
| e8a6: | 4c 9d e6 | jmp * \$e69d |           | Ricez. bit da process.                |
| e8a9: | ad 12 0a | lda \$0a12   |           | Metti in A baud rate uten. di RS-232  |
| e8ac: | 8d 06 dd | sta * \$dd06 |           | Somma q.s. in timer LO                |
| e8af: | ad 13 0a | lda \$0a13   |           | Metti in A baud rate uten. di RS-232  |
| e8b2: | 8d 07 dd | sta * \$dd07 |           | Somma q.s. in timer HI                |
| e8b5: | a9 11    | lda #11      |           | Scrivi 11 nel registro di contr.      |
| e8b7: | 8d 0f dd | sta * \$dd0f |           | Timer partenza                        |
| e8ba: | a9 12    | lda #12      |           | Es. invers. bits 0,1,4 di RS-232 rel. |
| e8bc: | 4d 0f 0a | eor \$0a0f   |           | flag NMI. valore poi riportato nel    |
| e8bf: | 8d 0f 0a | sta \$0a0f   |           | nel flag NMI                          |
| e8c2: | a9 ff    | lda #fff     |           | Val. di iniz. per timer B             |
| e8c4: | 8d 06 dd | sta * \$dd06 |           | Metti timer B LO a valore HI          |
| e8c7: | 8d 07 dd | sta * \$dd07 |           | Idem per B HI                         |
| e8ca: | ae 15 0a | ldx \$0a15   |           | Metti in pag. zero n.bits da inviare  |
| e8cd: | 86 a8    | stx \$a8     | [ bitci ] | cont. dei bits di RS-232              |
| e8cf: | 60       | rts          |           |                                       |

-----  
LETTURA HEADER DA NASTRO

|       |          |              |            |                                     |
|-------|----------|--------------|------------|-------------------------------------|
| e8d0: | a5 93    | lda \$93     | [ verck ]  | Metti in Stack di sistema punt. per |
| e8d2: | 48       | pha          |            | LOAD/VERIFY a mezzo A               |
| e8d3: | 20 f2 e9 | jsr * \$e9f2 |            | Rout. lettura dati da blocco nastro |
| e8d6: | 68       | pla          |            | Prel. flag di LOAD/VERIFY da Stack  |
| e8d7: | 85 93    | sta \$93     | [ verck ]  | e riportalo in pag. 0               |
| e8d9: | b0 3d    | bcs * \$e918 |            | Salta a ind. se rilevi err.         |
| e8db: | a0 00    | ldy #00      |            | Fissa spost. per buffer nastro      |
| e8dd: | b1 b2    | lda (\$b2),y | [ tapel ]  | Car.byte del blocco dati letto      |
| e8df: | c9 05    | cmp #05      |            | Contr. per EOT                      |
| e8e1: | f0 34    | beq * \$e917 |            | Se pos. vai a ind.                  |
| e8e3: | c9 01    | cmp #01      |            | Contr.se header e' di progr. Basic  |
| e8e5: | f0 08    | beq * \$e8ef |            | Se pos. valutalo                    |
| e8e7: | c9 03    | cmp #03      |            | Contr. se header e' di prog. in LM  |
| e8e9: | f0 04    | beq * \$e8ef |            | Se pos. vai ad ind.                 |
| e8eb: | c9 04    | cmp #04      |            | Contr. header e' di blocco dati     |
| e8ed: | d0 e1    | bne * \$e8d0 |            | Se neg. leggi in tipo di header     |
| e8ef: | aa       | tax          |            | immagaz. in X                       |
| e8f0: | 24 9d    | bit \$9d     | [ msgflg ] | Contr. flag di Status della Kern    |
| e8f2: | 10 22    | bpl * \$e916 |            | Contr. mess. non possibile          |
| e8f4: | a0 63    | ldy #63      |            | Spostati su FOUND                   |
| e8f6: | 20 22 f7 | jsr * \$f722 |            | Uscita mess. di controllo           |
| e8f9: | a0 05    | ldy #05      |            | Fissa spost. su iniz. nome file     |
| e8fb: | b1 b2    | lda (\$b2),y | [ tapel ]  | Let. caratt. da buffer nastro       |
| e8fd: | 20 d2 ff | jsr * \$ffd2 | [ bsout ]  | Vai a BSOUT                         |
| e900: | c8       | iny          |            | Incr. punt. di 1                    |

|             |              |             |                                     |
|-------------|--------------|-------------|-------------------------------------|
| e901: c0 15 | cpy # \$15   |             | Contr. max lungh. file per 16 car   |
| e903: d0 f6 | bne * \$e8fb |             | Se non trovato continua             |
| e905: a5 a1 | lda \$a1     | [ time+ 1 ] | Metti in A byte di temp. MIDDLE     |
| e907: 69 02 | adc # \$02   |             | Ciclo di rit. di 8.5 sec            |
| e909: a4 91 | ldy \$91     | [ stkey ]   | Contr. flag stop e tasto C in Pag.0 |
| e90b: c8    | iny          |             | Increm. valore di 1                 |
| e90c: d0 04 | bne * \$e912 |             | Se tasto premuto cont.              |
| e90e: c5 a1 | cmp \$a1     | [ time+ 1 ] | Contr. ciclo rit. di 8.5 sec.       |
| e910: d0 f7 | bne * \$e909 |             | Contr. temp. neg. Cont. attesa      |
| e912: c0 f0 | cpy # \$f0   |             | Contr. tasto spazio prem.           |
| e914: f0 ba | beq * \$e8d0 |             | Contr. pos.leggi Header             |
| e916: 18    | clc          |             | Es. clear di carry (indic.)         |
| e917: 88    | dey          |             | Vecchio val.flag STOP/C             |
| e918: 60    | rts          |             |                                     |

-----  
SCRITTURA BLOCCO DATI SU NASTRO

|                |              |           |                                      |
|----------------|--------------|-----------|--------------------------------------|
| e919: 85 9e    | sta \$9e     | [ ptr1 ]  | Immetti tipo HEADER in Pag.0         |
| e91b: 20 80 e9 | jsr * \$e980 |           | Prendi ind. buffer nastro            |
| e91e: 90 5f    | bcc * \$e97f |           | Se ind. non valido salta             |
| e920: a5 c2    | lda \$c2     | [ stah ]  | Imm. in A Indir. HI di part.         |
| e922: 48       | pha          |           | Salva q.s. su Stack                  |
| e923: a5 c1    | lda \$c1     | [ track ] | Immetti ind. part. LO in A e salva   |
| e925: 48       | pha          |           | su Stack                             |
| e926: a5 af    | lda \$af     | [ eah ]   | Immetti ind. fine HI in A e salva    |
| e928: 48       | pha          |           | su Stack                             |
| e929: a5 ae    | lda \$ae     | [ eal ]   | C.s. per ind. fine LO                |
| e92b: 48       | pha          |           |                                      |
| e92c: a0 bf    | ldy # \$bf   |           | Carica lungh.buffer nastro per cicl  |
| e92e: a9 20    | lda # \$20   |           | Car. A con car. per spazio           |
| e930: 91 b2    | sta (\$b2),y | [ tapel ] | Es. clear su buffer nastro           |
| e932: 88       | dey          |           | Ciclo che continua fino a quando la  |
| e933: d0 fb    | bne * \$e930 |           | intera lungh. in Y non E' clear      |
| e935: a5 9e    | lda \$9e     | [ ptr1 ]  | Carica tipo Header                   |
| e937: 91 b2    | sta (\$b2),y | [ tapel ] | Prima pos. nel buffer nastro         |
| e939: c8       | iny          |           | Incr. di 1 buffer nastro             |
| e93a: a5 c1    | lda \$c1     | [ track ] | Car. ind.part.LO da pag. 0           |
| e93c: 91 b2    | sta (\$b2),y | [ tapel ] | Immetti q.s. in buffer nastro        |
| e93e: c8       | iny          |           | Incr. buffer nastro di 1             |
| e93f: a5 c2    | lda \$c2     |           | Carica ind.part. HI da pag. 0        |
| e941: 91 b2    | sta (\$b2),y | [ tapel ] | Immetti q.s. in buffer nastro        |
| e943: c8       | iny          |           | Incr. di 1 buffer nastro             |
| e944: a5 ae    | lda \$ae     | [ eal ]   | Carica ind.fine LO da Pag. 0         |
| e946: 91 b2    | sta (\$b2),y | [ tapel ] | Immettiq.s. in buffer nastro         |
| e948: c8       | iny          |           | Incr.buffer nastro di 1              |
| e949: a5 af    | lda \$af     | [ eah ]   | Carica ind.fine HI da pag. 0         |
| e94b: 91 b2    | sta (\$b2),y | [ tapel ] | Immetti q.s. in buffer nastro        |
| e94d: c8       | iny          |           | Incr. buffer nastro di 1             |
| e94e: 84 9f    | sty \$9f     | [ ptr2 ]  | Salva q.s. in buffer nastro          |
| e950: a0 00    | ldy # \$00   |           | Es. clear su contat.lungh. nome file |
| e952: 84 9e    | sty \$9e     | [ ptr1 ]  | presente in pag. 0                   |
| e954: a4 9e    | ldy \$9e     | [ ptr1 ]  | Metti cont. per lungh. nome file     |
| e956: c4 b7    | cpy \$b7     | [ fnlen ] | Confronta q.s. con att. lungh.       |
| e958: f0 0d    | beq * \$e967 |           | Contr. tutte lettere in Buffer       |
| e95a: 20 ae f7 | jsr * \$f7ae |           | Lettere da nome file                 |
| e95d: a4 9f    | ldy \$9f     | [ ptr2 ]  | Metti Y di Pag. 0 in buffer nastro   |
| e95f: 91 b2    | sta (\$b2),y | [ tapel ] | Metti in buffer lettere nome file    |
| e961: e6 9e    | inc \$9e     | [ ptr1 ]  | Incr.cont.lungh.nome file di 1       |
| e963: e6 9f    | inc \$9f     | [ ptr2 ]  | Incr. di 1 buffer nastro             |
| e965: d0 ed    | bne * \$e954 |           | Ciclo per succ. lettera              |

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| e967: | 20 87 e9 | jsr * \$e987 |           | Inizio e ind. buffer nastro           |
| e96a: | a9 69    | lda #\$69    |           | Immag. dati di CHECKSUM e blocco di   |
| e96c: | 85 ab    | sta \$ab     | [ ripty ] | Header in pag.zero                    |
| e96e: | 20 1c ea | jsr * \$ealc |           | Scrivi blocco su nastro               |
| e971: | a8       | tya          |           | Salva in Y att.cont. di A             |
| e972: | 68       | pla          |           | Questa serie di istruzioni serve per  |
| e973: | 85 ae    | sta \$ae     | [ eal ]   | caricare in A da Stack indirizzi di   |
| e975: | 68       | pla          |           | inizio e fine ed immag. in pag. 0. La |
| e976: | 85 af    | sta \$af     | [ eah ]   | sequenza partendo da \$E972:          |
| e978: | 68       | pla          |           | Fine ind. HI-fine ind. LO             |
| e979: | 85 c1    | sta \$c1     | [ track ] | Inizio ind. HI                        |
| e97b: | 68       | pla          |           | Inizio ind. LO                        |
| e97c: | 85 c2    | sta \$c2     | [ stah ]  |                                       |
| e97e: | 98       | tya          |           | Salva ancora su A                     |
| e97f: | 60       | rts          |           |                                       |

|       |       |           |              |                               |
|-------|-------|-----------|--------------|-------------------------------|
| e980: | a6 b2 | ldx \$b2  | [ tapel ]    | Partenza buffer nastro in X   |
| e982: | a4 b3 | ldy \$b3  | [ tapel+ 1 ] | Idem (altra parte) in Y       |
| e984: | c0 02 | cpy #\$02 |              | Pag. zero e stack non dispon. |
| e986: | 60    | rts       |              |                               |

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| e987: | 20 80 e9 | jsr * \$e980 |           | Vai a ind. buffer nastro              |
| e98a: | 8a       | txa          |           | Trasf. in A buffer in. nastro LO      |
| e98b: | 85 c1    | sta \$c1     | [ track ] | Immag.q.s in Pag. 0                   |
| e98d: | 18       | clc          |           | Es. clear di carry per somma          |
| e98e: | 69 c0    | adc #\$c0    |           | Ind. di fine=ind.iniz +192            |
| e990: | 85 ae    | sta \$ae     | [ eal ]   | Metti nuovo ind. di fine LO in Pag. 0 |
| e992: | 98       | tya          |           | Inizio buffer nastro HI in A          |
| e993: | 85 c2    | sta \$c2     | [ stah ]  | Immag. Pag.0 ind.ini.I/O HI           |
| e995: | 69 00    | adc #\$00    |           | Poni ind.fine HI=ind.part.HI + il     |
| e997: | 85 af    | sta \$af     | [ eah ]   | carry in Pag. 0 come ind.fine HI      |
| e999: | 60       | rts          |           |                                       |

## RICERCA HEADER NASTRO PER NOME

|       |          |               |           |                                      |
|-------|----------|---------------|-----------|--------------------------------------|
| e99a: | 20 d0 e8 | jsr * \$e8d0  |           | Ricerca pross. header nastro         |
| e99d: | b0 1e    | bcs * \$e9bd  |           | Se EOT trovato return                |
| e99f: | a0 05    | ldy #\$05     |           | Sost.al nome in buffer nastro        |
| e9a1: | 84 9f    | sty \$9f      | [ ptr2 ]  | Immag. in pag. 0                     |
| e9a3: | a0 00    | ldy #\$00     |           | Iniz. contat.per lungh.nome file     |
| e9a5: | 84 9e    | sty \$9e      | [ ptr1 ]  | in pag. 0                            |
| e9a7: | c4 b7    | cpy \$b7      | [ fnlen ] | Confr.lungh.nome                     |
| e9a9: | f0 11    | beq * \$e9bc  |           | Se ident. cont. operaz.              |
| e9ab: | 20 ae f7 | jsr * \$f7ae  |           | Metti car. nome da trov.             |
| e9ae: | a4 9f    | ldy \$9f      | [ ptr2 ]  | Sost. al nome file nel buffer nastro |
| e9b0: | d1 b2    | cmp (\$b2),y  | [ tapel ] | Confronta con car. da trov.          |
| e9b2: | d0 e6    | bne * \$e99a7 |           | Se diverso salta                     |
| e9b4: | e6 9e    | inc \$9e      | [ ptr1 ]  | Incr. di 1 cont. nome file           |
| e9b6: | e6 9f    | inc \$9f      | [ ptr2 ]  | Sposta su buffer nas.nome file +1    |
| e9b8: | a4 9e    | ldy \$9e      | [ ptr1 ]  | Metti in Y cont. lungh. nome file    |
| e9ba: | d0 eb    | bne * \$e9a7  |           | Confronto con caratt. succ.          |
| e9bc: | 18       | clc           |           | Es. clear di carry per indic. di OK  |
| e9bd: | 60       | rts           |           |                                      |

|       |          |              |           |                                    |
|-------|----------|--------------|-----------|------------------------------------|
| e9be: | 20 80 e9 | jsr * \$e980 |           | Prel.ind.buffer nastro             |
| e9c1: | e6 a6    | inc \$a6     | [ bufpt ] | Incr.buffer nastro +1 in Pag. 0    |
| e9c3: | a4 a6    | ldy \$a6     | [ bufpt ] | Confronta con val. di 192 definito |
| e9c5: | c0 c0    | cpy #\$c0    |           | come massimo                       |
| e9c7: | 60       | rts          |           |                                    |

|       |          |       |        |            |                                       |
|-------|----------|-------|--------|------------|---------------------------------------|
| ----- |          |       |        |            |                                       |
| e9c8: | 20 df e9 | jsr * | \$e9df |            | Controllo tasto premuto               |
| e9cb: | f0 1a    | beq * | \$e9e7 |            | Contr. pos. continua                  |
| e9cd: | a0 1b    | ldy   | #\$1b  |            | Sposta in Y PRESS PLAY ON TAPE e      |
| e9cf: | 20 22 f7 | jsr * | \$f722 |            | visual.messaggio                      |
| e9d2: | 20 8f ea | jsr * | \$ea8f |            | Controllo per tasto STOP              |
| e9d5: | 20 df e9 | jsr * | \$e9df |            | Esec. del controllo tasto premuto     |
| e9d8: | d0 f8    | bne * | \$e9d2 |            | Contr. neg. a ciclo ritardo           |
| e9da: | a0 6a    | ldy   | #\$6a  |            | Pos. per messaggio OK                 |
| e9dc: | 4c 22 f7 | jmp * | \$f722 |            | Vis. messaggio di contr.              |
| e9df: | a9 10    | lda   | #\$10  |            | Fissa bit 4 per contr. tasto cass.    |
| e9el: | 24 01    | bit   | \$01   | [ r6510 ]  | Contr. dati porta reg.proc.           |
| e9e3: | d0 02    | bne * | \$e9e7 |            | Se tasto non prem. esci               |
| e9e5: | 24 01    | bit   | \$01   | [ r6510 ]  | Esegui di nuovo contr.                |
| e9e7: | 18       | clc   |        |            | Clear di carry per flag ZERO          |
| e9e8: | 60       | rts   |        |            |                                       |
| ----- |          |       |        |            |                                       |
| e9e9: | 20 df e9 | jsr * | \$e9df |            | Controllo tasto cass. premuto         |
| e9ec: | f0 f9    | beq * | \$e9e7 |            | Se tasto prem. continua               |
| e9ee: | a0 2e    | ldy   | #\$2e  |            | Carica PRESS R & P ON TAPE            |
| e9f0: | d0 dd    | bne * | \$e9cf |            | Contr.rit.nastro e ciclo STOP         |
| e9f2: | a9 00    | lda   | #\$00  |            | Lett. dati da nastro                  |
| e9f4: | 85 90    | sta   | \$90   | [ status ] | Iniz. status                          |
| e9f6: | 85 93    | sta   | \$93   | [ verck ]  | Es. clear su punt. LOAD/VERIFY        |
| e9f8: | 20 87 e9 | jsr * | \$e987 |            | Pre.ind.buffer nastro e ind.fine      |
| e9fb: | 20 c8 e9 | jsr * | \$e9c8 |            | Attesa per tasto prem.su cassetta     |
| e9fe: | b0 1f    | bcs * | \$ealf |            | Contr. STOP premuto                   |
| ea00: | 78       | sei   |        |            | Disab. tutti gli interrupt            |
| ea01: | a9 00    | lda   | #\$00  |            | Iniz. val. per immag. IRQ             |
| ea03: | 85 aa    | sta   | \$aa   | [ rdflg ]  | Immag.byte ing.mod. lettura           |
| ea05: | 85 b4    | sta   | \$b4   | [ bitts ]  | Punt. temp. nastro                    |
| ea07: | 85 b0    | sta   | \$b0   | [ cmp0 ]   | Costante temp. di cassetta            |
| ea09: | 85 9e    | sta   | \$9e   | [ ptr1 ]   | Errore cassetta passo 1               |
| ea0b: | 85 9f    | sta   | \$9f   | [ ptr2 ]   | Errore cassetta passo 2               |
| ea0d: | 85 9c    | sta   | \$9c   | [ dpsw ]   | Flag nastro per byte ric.             |
| ea0f: | a9 90    | lda   | #\$90  |            | Flag IRQ su relat. pin                |
| ea11: | a2 0e    | ldx   | #\$0e  |            | Numero vett. di IRQ                   |
| ea13: | d0 11    | bne * | \$ea26 |            | Scrivi blocco dati su nastro          |
| ea15: | 20 87 e9 | jsr * | \$e987 |            | Carica ind.buffer nastro              |
| ea18: | a9 14    | lda   | #\$14  |            | Fissa lungh. per scrittura            |
| ea1a: | 85 ab    | sta   | \$ab   | [ ripty ]  | Immag. in Pag. 0                      |
| ea1c: | 20 e9 e9 | jsr * | \$e9e9 |            | Attendi per tasto RECORD & PLAY prem. |
| ea1f: | b0 7a    | bcs * | \$ea9b |            | Se STOP premuto vai a ind.            |
| ea21: | 78       | sei   |        |            | Disab. tutti gli interrupt            |
| ea22: | a9 82    | lda   | #\$82  |            | Se timer B in UNDERFLOW genera IRQ    |
| ea24: | a2 08    | ldx   | #\$08  |            | Num. vett. IRQ (\$EE2E)               |
| ea26: | a0 00    | ldy   | #\$00  |            | Fissa mask di Interrupt regi. CIA     |
| ea28: | 8c 1a d0 | sty * | \$d01a |            | Disabilita Interrupt                  |
| ea2b: | 88       | dey   |        |            | Decr. di l reg. Y e fissa il regis.   |
| ea2c: | 8c 19 d0 | sty * | \$d019 |            | di Interrupt (IRR)                    |
| ea2f: | 8d 0d dc | sta * | \$dc0d |            | Reset di IRQ                          |
| ea32: | ad 0e dc | lda * | \$dc0e |            | Car. A reg.contr.CIA timer B          |
| ea35: | 09 19    | ora   | #\$19  |            | Es. un OR e fai partire               |
| ea37: | 8d 0f dc | sta * | \$dc0f |            | Contr. reg. B IRQ su timer B          |
| ea3a: | 29 91    | and   | #\$91  |            | Fissa punt. di confr. tempor. per     |
| ea3c: | 8d 0b 0a | sta   | \$0a0b |            | operazioni su nastro                  |
| ea3f: | 20 ec e7 | jsr * | \$e7ec |            | Attendi per fine trasf. RS-232        |
| ea42: | ad 11 d0 | lda * | \$d011 |            | Copia in A reg. contr. VIC            |
| ea45: | a8       | tay   |        |            | Trasf. q.s. in Y                      |
| ea46: | 29 10    | and   | #\$10  |            | Bit 4 a 1 per schermo attivo          |

|                |              |           |                                       |
|----------------|--------------|-----------|---------------------------------------|
| ea48: 8d 39 0a | sta \$0a39   |           | Immag. val.in mag. temp. VDC          |
| ea4b: 98       | tya          |           | Vecchio val. in A                     |
| ea4c: 29 6f    | and #\$6f    |           | Es.clear bit 8                        |
| ea4e: 8d 11 d0 | sta * \$d011 |           | Disab schermo                         |
| ea51: 20 74 e5 | jsr * \$e574 |           | Metti clock a 1 MHz                   |
| ea54: ad 14 03 | lda \$0314   |           | Ind. vettore LO di IRQ in IRQ         |
| ea57: 8d 09 0a | sta \$0a09   |           | Immag. temp. per operaz.nastro        |
| ea5a: ad 15 03 | lda \$0315   |           | Ind. vettore HI di IRQ in IRQ         |
| ea5d: 8d 0a 0a | sta \$0a0a   |           | Immag. temp. per operaz. nastro       |
| ea60: 20 9b ee | jsr * \$ee9b |           | Reset vettore di IRQ per operaz. nas. |
| ea63: a9 02    | lda #\$02    |           | N.blocchi data da leggere             |
| ea65: 85 be    | sta \$be     | [ fsblk ] | Metti q.s. in pag. 0                  |
| ea67: 20 5a ed | jsr * \$ed5a |           | Iniz. contat. bit seriale di I/O      |
| ea6a: a5 01    | lda \$01     | [ r6510 ] | Attiva motore cassetta mettendo a 1   |
| ea6c: 29 1f    | and #\$1f    |           | il quarto bit del registro dati       |
| ea6e: 85 01    | sta \$01     | [ r6510 ] | della porta del proc.                 |
| ea70: 85 c0    | sta \$c0     | [ casl ]  | Fissa punt. per motore nastro         |
| ea72: a2 ff    | ldx #\$ff    |           | Cont. per ciclo rit. HI               |
| ea74: a0 ff    | ldy #\$ff    |           | Cont. per ciclo rit. LO               |
| ea76: 88       | dey          |           | Decrementa Y e X contando da 65535 a  |
| ea77: d0 fd    | bne * \$ea76 |           | 0 per creare un ritardo necessario    |
| ea79: ca       | dex          |           | per le operazioni su                  |
| ea7a: d0 f8    | bne * \$ea74 |           | nastro                                |
| ea7c: 58       | cli          |           | Abilita Interrupt per I/O nastro      |
| ea7d: ad 0a 0a | lda \$0a0a   |           | Confronta vettore nastro IRQ con      |
| ea80: cd 15 03 | cmp \$0315   |           | norm. punt. HI di IRQ                 |
| ea83: 18       | clc          |           | Es. clear di carry per indic.         |
| ea84: f0 15    | beq * \$ea9b |           | Contr. vettori IRQ                    |
| ea86: 20 8f ea | jsr * \$ea8f |           | Controlla tasto di STOP prem.         |
| ea89: 20 3d f6 | jsr * \$f63d |           | Se contr. pos. fissa Flag             |
| ea8c: 4c 7d ea | jmp * \$ea7d |           | Continua per fine                     |
| ea8f: 20 e1 ff | jsr * \$ffef | [ stop ]  | Contr. tasto STOP                     |
| ea92: 18       | clc          |           | Fissa indic                           |
| ea93: d0 0b    | bne * \$eaa0 |           | Tasto non premuto, RTS                |
| ea95: 20 57 ee | jsr * \$ee57 |           | Motore disatt. fissa norm.IRQ         |
| ea98: 38       | sec          |           | Fissa Carry per errore                |
| ea99: 68       | pla          |           | Metti in A ind.ritorno prel. da       |
| ea9a: 68       | pla          |           | Stack. Quindi esegui clear            |
| ea9b: a9 00    | lda #\$00    |           | Carica in A cod. per interrupt        |
| ea9d: 8d 0a 0a | sta \$0a0a   |           | Fissa per IRQ normale                 |
| aaa0: 60       | rts          |           |                                       |

|             |              |          |                                     |
|-------------|--------------|----------|-------------------------------------|
| eaal: 86 b1 | stx \$b1     | [ temp ] | Immag. X in Pag. 0                  |
| aaa3: a5 b0 | lda \$b0     | [ cmp0 ] | In A costante di temp. per nastro   |
| aaa5: 0a    | asl a        |          | La prec. costante e' moltiplicata   |
| aaa6: 0a    | asl a        |          | per un fattore 4                    |
| aaa7: 18    | clc          |          | Es. clear di carry per somma        |
| aaa8: 65 b0 | adc \$b0     | [ cmp0 ] | Sommaci cost. di temporizz          |
| aaaa: 18    | clc          |          | Clear di carry per somma            |
| eaab: 65 b1 | adc \$b1     | [ temp ] | Somma cont. reg. X ed immetti       |
| eaad: 85 b1 | sta \$b1     | [ temp ] | questo valore in Pag. 0             |
| eaaf: a9 00 | lda #\$00    |          | Carica valore LO per timer A        |
| eabl: 24 b0 | bit \$b0     | [ cmp0 ] | Contr.cost.di temporizz.> 128       |
| eab3: 30 01 | bmi * \$eab6 |          | Se contr. pos. salta                |
| eab5: 2a    | rol a        |          | Il val. prec. per il timer A viene  |
| eab6: 06 b1 | asl \$b1     | [ temp ] | moltip. per 4 con rotaz. del cont.  |
| eab8: 2a    | rol a        |          | di A in unione allo spost. a sinis. |
| eab9: 06 b1 | asl \$b1     | [ temp ] | della costante di tempor. nastro    |
| eabb: 2a    | rol a        |          | ****                                |



```

eabc: aa      tax
eabd: ad 06 dc lda * $dc06
eac0: c9 16   cmp # $16
eac2: 90 f9   bcc * $eabd
eac4: 65 b1   adc $b1 [ temp ]
eac6: 8d 04 dc sta * $dc04
eac9: 8a      txa
eaca: 6d 07 dc adc * $dc07
eacd: 8d 05 dc sta * $dc05
ead0: ad 0b 0a lda $0a0b
ead3: 8d 0e dc sta * $dc0e
ead6: 8d 0d 0a sta $0a0d
ead9: ad 0d dc lda * $dc0d
eadc: 29 10   and # $10
eade: f0 09   beq * $eae9
eae0: a9 ea   lda # $ea
eae2: 48      pha
eae3: a9 e9   lda # $e9
eae5: 48      pha
eae6: 4c c8 ee jmp * $eec8
eae9: 58      cli
eaea: 60      rts

```

Trasf in X val. Hi di Timer  
 Metti in A val CIA l timer B  
 Metti timer B HI a 63755  
 Contr. q.s.se pos.ciclo lett.timer  
 Somma LO per inizializz  
 Metti timer A LO  
 Somma in A val. HI di inizializz.  
 con carry su timer B alto  
 Metti q.s. in timer A HI  
 Copia in A val.iniz da nastro  
 Costan.per part.timer A  
 Reset flag timer A  
 Reg. di contr. interrupt in A  
 Contr.impulso neg.su pin FLAG  
 Se imp.non arriv.attendine arrivo  
 Immetti il contenuto delle locaz. di  
 pag. 0 \$EA e \$E9 su stack consid.  
 come indirizzo di return  
 \*\*\*  
 Simula chiamata di Interrupt  
 Abilita tutti interrupt

```

-----
eaeB: ae 07 dc ldx * $dc07
eaeE: a0 ff   ldy # $ff
eaf0: 98      tya
eaf1: ed 06 dc sbc * $dc06
eaf4: ec 07 dc cpx * $dc07
eaf7: d0 f2   bne * $eaeB
eaf9: 86 b1   stx $b1 [ temp ]
eafb: aa      tax
eafc: 8c 06 dc sty * $dc06
eaff: 8c 07 dc sty * $dc07
eb02: a9 19   lda # $19
eb04: 8d 0f dc sta * $dc0f
eb07: ad 0d dc lda * $dc0d
eb0a: 8d 0c 0a sta $0a0c
eb0d: 98      tya
eb0e: e5 b1   sbc $b1 [ temp ]
eb10: 86 b1   stx $b1 [ temp ]
eb12: 4a      lsr a
eb13: 66 b1   ror $b1 [ temp ]
eb15: 4a      lsr a
eb16: 66 b1   ror $b1 [ temp ]
eb18: a5 b0   lda $b0 [ cmp0 ]
eb1a: 18      clc
eb1b: 69 3c   adc # $3c
eb1d: c5 b1   cmp $b1 [ temp ]
eb1f: b0 4a   bcs * $eb6b
eb21: a6 9c   ldx $9c [ dpsw ]
eb23: f0 03   beq * $eb28
eb25: 4c 1f   jmp * $ec1f
eb28: a6 a3   ldx $a3 [ pcntr ]
eb2a: 30 1b   bmi * $eb47
eb2c: a2 00   ldx # $00
eb2e: 69 30   adc # $30
eb30: 65 b0   adc $b0 [ cmp0 ]
eb32: c5 b1   cmp $b1 [ temp ]
eb34: b0 1c   bcs * $eb52

```

Metti in X timer B di CIA l  
 Iniz. reg Y con val. HI  
 Trasf.q.s. in A per sottraz.  
 Dim. timer B di #255  
 Contr. decr.timer B HI  
 Se contr. pos.eseg.confir. temp.  
 Immetti timer B HI in Pag. 0  
 Temp. Lo fino arrivo ult. segn in X  
 Timer B LO  
 Timer B HI  
 Abilita modo funz. timer B  
 Fai partire timer B  
 Metti in A ICR  
 Immag. per nastro q.s.  
 Iniz. A con #255  
 Sottrai 255 da timer B HI  
 Metti in X di pag. 0 tempo trasc.  
 Il valore immag. in A per tempo  
 trascorso e' diviso per un fattore  
 4  
  
 Prendi cost. di temp. da pag.0  
 Es.clear di carry per somma  
 Somma 60 a cost. di tempor.  
 Confr. ultimo segn.con tem.  
 Se magg. nessuna inf. e salta  
 Contr.se quanto ric e' un byte  
 Se no salta  
 Cont. rout.ricev. byte  
 Contr. se byte e' letto per int.  
 Se pos. esegui valut. codice per imp  
 corto in X (0)  
 Fissa A per lett. impulso  
 Sommaci cost. di temporizz.  
 L' impulso ricev. e' di tipo cor.  
 Se si salta a imp. lungo

|       |          |     |          |            |  |
|-------|----------|-----|----------|------------|--|
| eb36: | e8       | inx |          |            | Metti in X codice per imp.lungo        |
| eb37: | 69 26    | adc | #\$26    |            | Fissa A per lett. imp.                 |
| eb39: | 65 b0    | adc | \$b0     | [ cmp0 ]   | Sommaci cost. di temporizz.            |
| eb3b: | c5 b1    | cmp | \$b1     | [ temp ]   | Contr. per imp. lungo                  |
| eb3d: | b0 17    | bcs | * \$eb56 |            | Se pos. salta altre durate             |
| eb3f: | 69 2c    | adc | #\$2c    |            | Controlla se atteso impulso e' di tipo |
| eb41: | 65 b0    | adc | \$b0     | [ cmp0 ]   | piu' lungo                             |
| eb43: | c5 b1    | cmp | \$b1     | [ temp ]   | Se contr. pos.e' un byte di header     |
| eb45: | 90 03    | bcc | * \$eb4a |            | Se contr. neg. salta                   |
| eb47: | 4c cf eb | jmp | * \$ebcf |            | Ricez. byte                            |
| eb4a: | a5 b4    | lda | \$b4     | [ bitts ]  | Contr. se timer A e' abilit.           |
| eb4c: | f0 1d    | beq | * \$eb6b |            | Se no salta                            |
| eb4e: | 85 a8    | sta | \$a8     | [ bitci ]  | Fissa puntatore per READ ERROR         |
| eb50: | d0 19    | bne | * \$eb6b |            | Salta a lett. interrupt timer          |
| eb52: | e6 a9    | inc | \$a9     | [ rinone ] | Incr. di 1 punt.lungh.impulsi          |
| eb54: | b0 02    | bcs | * \$eb58 |            | Salta se var. decrem.                  |
| eb56: | c6 a9    | dec | \$a9     | [ rinone ] | Decr di 1 punt.lungh.imp.              |
| eb58: | 38       | sec |          |            | Fissa il carry per sottraz. da         |
| eb59: | e9 13    | sbc | #\$13    |            | valore letto 19                        |
| eb5b: | e5 b1    | sbc | \$b1     | [ temp ]   | Sottrai tempo trasc.                   |
| eb5d: | 65 92    | adc | \$92     | [ svxt ]   | Somma per temporizz. in pag. 0         |
| eb5f: | 85 92    | sta | \$92     | [ svxt ]   | Correz. flag e immag.                  |

-----

|       |          |       |          |           |                                      |
|-------|----------|-------|----------|-----------|--------------------------------------|
| eb61: | a5 a4    | lda   | \$a4     | [ firt ]  | Inverti il flag di Pag.0 per la      |
| eb63: | 49 01    | eor   | #\$01    |           | ricezione di entrambi i tipi di imp. |
| eb65: | 85 a4    | sta   | \$a4     | [ firt ]  | Immag. in pag. 0 entr. gli imp.      |
| eb67: | f0 2b    | beq   | * \$eb94 |           | ricevuti. Quindi salta               |
| eb69: | 86 c5    | stx   | \$c5     | [ data ]  | Immag. segn. ricev. in Pag. 0        |
| eb6b: | a5 b4    | lda   | \$b4     | [ bitts ] | Contr. se timer A e' abilitato       |
| eb6d: | f0 22    | beq   | * \$eb91 |           | Se contr. neg. term. interrupt       |
| eb6f: | ad 0c 0a | lda   | \$0a0c   |           | Metti cont. di ICR in A              |
| eb72: | 29 01    | and   | #\$01    |           | Contr. se era un Interrupt timer A   |
| eb74: | d0 05    | bne   | * \$eb7b |           | Se si salta a ind.                   |
| eb76: | ad 0d 0a | lda   | \$0a0d   |           | Contr. se timer A abilit.            |
| eb79: | d0 16    | bne   | * \$eb91 |           | Se contr.neg. termina interrupt      |
| eb7b: | a9 00    | lda   | #\$00    |           | Es. clear in pag. 0 per cont. di     |
| eb7d: | 85 a4    | sta   | \$a4     | [ firt ]  | impulso (val.L0)                     |
| eb7f: | 8d 0d 0a | sta   | \$0a0d   |           | Fissa punt. per timeout di timer A   |
| eb82: | a5 a3    | lda   | \$a3     | [ pctr ]  | Contr. che byte sia compl. letto     |
| eb84: | 10 30    | bpl   | * \$ebb6 |           | Se neg. salta ad ind.                |
| eb86: | 30 bf    | bmi   | * \$eb47 |           | Se pos. procedi di conseguenza       |
| eb88: | a2 a6    | ldx   | #\$a6    |           | Car. val. iniz. per timer A          |
| eb8a: | 20 a1 ea | jsr   | * \$eaal |           | Prep. nastro per lettura             |
| eb8d: | a5 9b    | lda   | \$9b     | [ prty ]  | Byte di parita in Pag. 0 in A        |
| eb8f: | d0 b9    | bne   | * \$eb4a |           | Se <> 0 errore di parita'            |
| eb91: | 4c 33 ff | jmp   | * \$ff33 |           | Vai a kernal Interrupt               |
| eb94: | a5 92    | lda   | \$92     | [ svxt ]  | Metti in A punt. correz. temporizz.  |
| eb96: | f0 07    | beq   | * \$eb9f |           | Es. clear su Flag evai ad ind.       |
| eb98: | 30 03    | bmi   | * \$eb9d |           | Contr. per >0.                       |
| eb9a: | c6 b0    | dec   | \$b0     | [ cmp0 ]  | Decr.dil costan.temporizz.in pag.0   |
| eb9c: | 2c       | .byte | \$2c     |           | Vai a \$eb9f                         |
| eb9d: | e6 b0    | inc   | \$d0     |           | Incrementa costante di tempor.       |
| eb9f: | a9 00    | lda   | #\$00    |           | Metti in A punt.cost.di temporizz.   |
| ebal: | 85 92    | sta   | \$92     | [ svxt ]  | Cancella correzione (L0)             |
| eba3: | e4 c5    | cpx   | \$c5     | [ data ]  | Confronta impulso ricevuto con       |
| eba5: | d0 0f    | bne   | * \$ebb6 |           | quello atteso. Se <> Ok e salta      |
| eba7: | 8a       | txa   |          |           | Contr. se imp. ricev. e' corto       |
| eba8: | d0 a0    | bne   | * \$eb4a |           | Se neg. errore                       |

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| ebaa: a5 a9    | lda \$a9     | [ rinone ] | Cambia in A punt.lungh.impulso        |
| ebac: 30 bd    | bmi * \$eb6b |            | Contr.se val.neg.Quindi salta         |
| ebae: c9 10    | cmp #\$10    |            | Contr.ricez.per 16 impulsi corti      |
| ebb0: 90 b9    | bcc * \$eb6b |            | Se neg. errore                        |
| ebb2: 85 96    | sta \$96     | [ syno ]   | Se pos. e' un flag di EOB             |
| ebb4: b0 b5    | bcs * \$eb6b |            | Salto incond.                         |
| ebb6: 8a       | txa          |            | Trasf bit ricevuto in A               |
| ebb7: 45 9b    | eor \$9b     | [ prty ]   | Confronta con parita' nastro          |
| ebb9: 85 9b    | sta \$9b     | [ prty ]   | Immag. in parita' nastro              |
| ebbb: a5 b4    | lda \$b4     | [ bitts ]  | Contr. abilit. timer A                |
| ebbd: f0 d2    | beq * \$eb91 |            | Se disabil. fine interrupt            |
| ebbf: c6 a3    | dec \$a3     | [ pcntr ]  | Decr.di 1 cont.immag.bit in Pag. 0    |
| ebc1: 30 c5    | bmi * \$eb88 |            | Contr.ricez. bit di parita'.Pos.salta |
| ebc3: 46 c5    | lsr \$c5     | [ data ]   | Contr. neg.mett. bit letto in Pag.0   |
| ebc5: 66 bf    | ror \$bf     | [ drive ]  | per dati nastro                       |
| ebc7: a2 da    | ldx #\$da    |            | Valore di inizializz.per timer A      |
| ebc9: 20 a1 ea | jsr * \$eaal |            | Predisp.sincronizz.cassetta           |
| ebcc: 4c 33 ff | jmp * \$ff33 |            | Vai a rout. di IRQ                    |
| ebcf: a5 96    | lda \$96     | [ syno ]   | Contr. ricez. EOB                     |
| ebd1: f0 04    | beq * \$ebd7 |            | Se neg.vai a lett. timer              |
| ebd3: a5 b4    | lda \$b4     | [ bitts ]  | Contr. abilit. timer A                |
| ebd5: f0 07    | beq * \$ebde |            | Se neg. vai contr.cont. bit           |
| ebd7: a5 a3    | lda \$a3     | [ pcntr ]  | Contr.se cont.bit in Pag.0 in negati. |
| ebd9: 30 03    | bmi * \$ebde |            | Se contr.pos.attendi per byte Header  |
| ebdb: 4c 56 eb | jmp * \$eb56 |            | Elab.impulso lungo.                   |
| ebde: 46 b1    | lsr \$b1     | [ temp ]   | Dim. tempo di elab.fino ad incontr.   |
| ebe0: a9 93    | lda #\$93    |            | ultimo dato negativo e quindi         |
| ebe2: 38       | sec          |            | sottrai questo valore                 |
| ebe3: e5 b1    | sbc \$b1     | [ temp ]   | dalla costante 147                    |
| ebe5: 65 b0    | adc \$b0     | [ cmp0 ]   | Somma cost. di temporizz. in Pag.0    |
| ebe7: 0a       | asl a        |            | Raddoppia questo valore               |
| ebe8: aa       | tax          |            | Trasf. in X.Iniz.val.per timer A      |
| ebe9: 20 a1 ea | jsr * \$eaal |            | Prepara sincron. di cassetta          |
| ebec: e6 9c    | inc \$9c     | [ dpsw ]   | Incr.punt. in pag.0 per byte ricev.   |
| ebef: a5 b4    | lda \$b4     | [ bitts ]  | Controlla abilitaz.timer A            |
| ebf0: d0 11    | bne * \$ec03 |            | Se abili. salta                       |
| ebf2: a5 96    | lda \$96     | [ syno ]   | Contr. ricez. EOB                     |
| ebf4: f0 26    | beq * \$ec1c |            | Se neg. vai a normale rout.IRQ        |
| ebf6: 85 a8    | sta \$a8     | [ bitci ]  | Visualizz. per lettura errore         |
| ebf8: a9 00    | lda #\$00    |            | Prepara immag. per EOB                |
| ebfa: 85 96    | sta \$96     | [ syno ]   | Cont.q.s.                             |
| ebfc: a9 81    | lda #\$81    |            | Val.codice abilitaz.timer A           |
| ebfe: 8d 0d dc | sta * \$dc0d |            | Abilita Interrupt per timer A         |
| ec01: 85 b4    | sta \$b4     | [ bitts ]  | Fissa flag in Pag.0 timer A           |
| ec03: a5 96    | lda \$96     | [ syno ]   | Copia A di Pag. 0 per ricez. EOB      |
| ec05: 85 b5    | sta \$b5     | [ diff ]   | Iniz. flag per EOB                    |
| ec07: f0 09    | beq * \$ec12 |            | Se nessun EOB salta                   |
| ec09: a9 00    | lda #\$00    |            | Codice di contr.per disab.timer A     |
| ec0b: 85 b4    | sta \$b4     | [ bitts ]  | Immetti punt. in pag.0                |
| ec0d: a9 01    | lda #\$01    |            | Cod. di contr.Disab. timer A          |
| ec0f: 8d 0d dc | sta * \$dc0d |            | Interrupt in reg. contr. CIA          |
| ec12: a5 bf    | lda \$bf     | [ drive ]  | Esegui lett. Pag.0                    |
| ec14: 85 bd    | sta \$bd     | [ roppty ] | Immag. in pag. 0 per lett. byte       |
| ec16: a5 a8    | lda \$a8     | [ bitci ]  | Unisci punt. in pag.0 per lett.err.   |
| ec18: 05 a9    | ora \$a9     | [ rinone ] | con punt. impulsi                     |
| ec1a: 85 b6    | sta \$b6     | [ prp ]    | Metti in err.cod.byte                 |
| ec1c: 4c 33 ff | jmp * \$ff33 |            | Vai a IRQ                             |
| ec1f: 20 5a ed | jsr * \$ed5a |            | Fissa cont. bit per uscita seriale    |
| ec22: 85 9c    | sta \$9c     | [ dpsw ]   | Reset punt.per Byte ricev.            |
| ec24: a2 da    | ldx #\$da    |            | Valore di inizializz. per timer A     |

|       |    |    |    |     |          |            |   |
|-------|----|----|----|-----|----------|------------|---|
| ec26: | 20 | al | ea | jsr | * \$eaal |            | Prep. sincr. di cassetta                    |
| ec29: | a5 | be |    | lda | \$be     | [ fsblk ]  | Contr.n. blocchi riman.=0                   |
| ec2b: | f0 | 02 |    | beq | * \$ec2f |            | Se pos. salta                               |
| ec2d: | 85 | a7 |    | sta | \$a7     | [ shcnl ]  | Ese. reset su n.blocchi da legg.            |
| ec2f: | a9 | 0f |    | lda | #\$0f    |            | Mask out val.cont.prima di lettura          |
| ec31: | 24 | aa |    | bit | \$aa     | [ rdflg ]  | Contr. punt.lettura da nastro               |
| ec33: | 10 | 17 |    | bpl | * \$ec4c |            | Se tutti car. ricev. fine                   |
| ec35: | a5 | b5 |    | lda | \$b5     | [ diff ]   | Contr.ricez. EOB                            |
| ec37: | d0 | 0c |    | bne | * \$ec45 |            | Sepos. salta                                |
| ec39: | a6 | be |    | ldx | \$be     | [ fsblk ]  | Contr. se n. restanti blocchi da leggere =1 |
| ec3b: | ca |    |    | dex |          |            |   |
| ec3c: | d0 | 0b |    | bne | * \$ec49 |            | Se neg. vai a IRQ                           |
| ec3e: | a9 | 08 |    | lda | #\$08    |            | Metti in A bit 3 per LONG BLOCK             |
| ec40: | 20 | 57 | f7 | jsr | * \$f757 |            | Reset sui punt. di sistema                  |
| ec43: | d0 | 04 |    | bne | * \$ec49 |            | Salto inc. a rout. IRQ                      |
| ec45: | a9 | 00 |    | lda | #\$00    |            | Metti punt.Pag.0 in lettura nastro          |
| ec47: | 85 | aa |    | sta | \$aa     | [ rdflg ]  | Fissane la scansione                        |
| ec49: | 4c | 33 | ff | jmp | * \$ff33 |            | Vai a IRQ normale                           |
| ec4c: | 70 | 31 |    | bvs | * \$ec7f |            | Punt. lett. nastro in READ                  |
| ec4e: | d0 | 18 |    | bne | * \$ec68 |            | Idem ma in COUNT                            |
| ec50: | a5 | b5 |    | lda | \$b5     | [ diff ]   | Contr. ricez. EOB                           |
| ec52: | d0 | f5 |    | bne | * \$ec49 |            | Se pos. vai a IRQ normale                   |
| ec54: | a5 | b6 |    | lda | \$b6     | [ prp ]    | Contr. errore lett. byte                    |
| ec56: | d0 | f1 |    | bne | * \$ec49 |            | Se errore vai a IRQ                         |
| ec58: | a5 | a7 |    | lda | \$a7     | [ shcnl ]  | Metti in A n.blocchi da leggere             |
| ec5a: | 4a |    |    | lsr | a        |            | Metti bit 0 in carry                        |
| ec5b: | a5 | bd |    | lda | \$bd     | [ roppty ] | Metti in A byte letto da Pag. 0             |
| ec5d: | 30 | 03 |    | bmi | * \$ec62 |            | Se e' un COUNT BYTE salta                   |
| ec5f: | 90 | 18 |    | bcc | * \$ec79 |            | Se e' letto piu' di un blocco salta         |
| ec61: | 18 |    |    | clc |          |            | Clear di carry(che e' punt.)                |
| ec62: | b0 | 15 |    | bcs | * \$ec79 |            | Salta se e' letto solo un blocco            |
| ec64: | 29 | 0f |    | and | #\$0f    |            | Mask out nibble super. (bits 4-7)           |
| ec66: | 85 | aa |    | sta | \$aa     | [ rdflg ]  | Immag. in A il valore                       |
| ec68: | c6 | aa |    | dec | \$aa     | [ rdflg ]  | Contr. ricez., tutti i byte di sincr.       |
| ec6a: | d0 | dd |    | bne | * \$ec49 |            | Se neg.IRQ normale                          |
| ec6c: | a9 | 40 |    | lda | #\$40    |            | Metti bit 6 in A e metti a READ punt        |
| ec6e: | 85 | aa |    | sta | \$aa     | [ rdflg ]  | lettura nastro di Pag.0                     |
| ec70: | 20 | 51 | ed | jsr | * \$ed51 |            | Copia indir. partenza di I/O                |
| ec73: | a9 | 00 |    | lda | \$00     |            | Metti a 0 punt. di pag. 0 per lett.         |
| ec75: | 85 | ab |    | sta | \$ab     | [ rippty ] | CHECKSUM                                    |
| ec77: | f0 | d0 |    | beq | * \$ec49 |            | Vai a IRQ normale                           |
| ec79: | a9 | 80 |    | lda | #\$80    |            | Metti bit 7 in A e metti punt. lett.        |
| ec7b: | 85 | aa |    | sta | \$aa     | [ rdflg ]  | nastro di Pag. 0 a fine                     |
| ec7d: | d0 | ca |    | bne | * \$ec49 |            | Vai a IRQ normale                           |
| ec7f: | a5 | b5 |    | lda | \$b5     | [ diff ]   | Contr. se EOB                               |
| ec81: | f0 | 0a |    | beq | * \$ec8d |            | Se neg. salta                               |
| ec83: | a9 | 04 |    | lda | #\$04    |            | Metti bit 2 in A per blocco corto           |
| ec85: | 20 | 57 | f7 | jsr | * \$f757 |            | Azzera i punt. di sistema                   |
| ec88: | a9 | 00 |    | lda | #\$00    |            | Metti cod. lett. punt. a 0                  |
| ec8a: | 4c | 0c | ed | jmp | * \$ed0c |            | Salto assol. a ind.                         |
| ec8d: | 20 | b7 | ee | jsr | * \$eeb7 |            | Contr.rintracc. fine                        |
| ec90: | 90 | 03 |    | bcc | * \$ec95 |            | Se neg. continua normalmente                |
| ec92: | 4c | 0a | ed | jmp | * \$ed0a |            | Let. fine di un blocco                      |
| ec95: | a6 | a7 |    | ldx | \$a7     | [ shcnl ]  | Contr. se n. blocchi da leggere             |
| ec97: | ca |    |    | dex |          |            | =1  |
| ec98: | f0 | 2e |    | beq | * \$ec8c |            | Se pos. salta                               |
| ec9a: | a5 | 93 |    | lda | \$93     | [ verck ]  | Contr. per marker di Verify                 |
| ec9c: | f0 | 0d |    | beq | * \$ecab |            | Se q.s.non fissato salta                    |
| ec9e: | a0 | 00 |    | ldy | #\$00    |            | Metti a 0 Y per confronto                   |
| eca0: | 20 | cc | f7 | jsr | * \$f7cc |            | Rout. per LSV                               |

|                |              |            |  |
|----------------|--------------|------------|--|
| eca3: c5 bd    | cmp \$bd     | [ roppty ] | Confr.con byte letto                   |
| eca5: f0 04    | beq * \$ecab |            | Se entr.= allora OK e salta            |
| eca7: a9 01    | lda #\$01    |            | Codice per err. lett. carattere        |
| eca9: 85 b6    | sta \$b6     | [ prp ]    | Punt. in Pag. 0 per nastro             |
| ecab: a5 b6    | lda \$b6     | [ prp ]    | Contr. punt. prec. per ril.errore      |
| ecad: f0 4c    | beq * \$ecfb |            | Se nessun errore,salta                 |
| ecaf: a2 3d    | ldx #\$3d    |            | Controllo rilev err. 31 durante la     |
| ecbl: e4 9e    | cpx \$9e     | [ ptrl ]   | fase di lettura                        |
| ecb3: 90 3f    | bcc * \$ecf4 |            | Se pos.salta perche' non corregg.      |
| ecb5: a6 9e    | ldx \$9e     | [ ptrl ]   | Spos.in Stack per lett. errore         |
| ecb7: a5 ad    | lda \$ad     | [ sah ]    | Poni LO ind.byte di errore             |
| ecb9: 9d 01 01 | sta \$0101,x |            | Immag. q.s. in stack                   |
| ecbc: a5 ac    | lda \$ac     | [ sal ]    | Esegui q.s. ma per                     |
| ecbe: 9d 00 01 | sta \$0100,x |            | byte HI                                |
| eccl: e8       | inx          |            | Increment.punt. ind. errore e cont.    |
| ecc2: e8       | inx          |            | n.errore di 2                          |
| ecc3: 86 9e    | stx \$9e     | [ ptrl ]   | Immag. in cont. errori                 |
| ecc5: 4c fb ec | jmp * \$ecfb |            | Se nessun errore cont.                 |
| ecc8: a6 9f    | ldx \$9f     | [ ptr2 ]   | Contr. per correz. di tutti gli err.   |
| ecca: e4 9e    | cpx \$9e     | [ ptrl ]   | letti                                  |
| eccc: f0 37    | beq * \$ed05 |            | Se pos. continua                       |
| ecce: a5 ac    | lda \$ac     | [ sal ]    | Metti a val. LO att. ind. byte         |
| ecd0: dd 00 01 | cmp \$0100,x |            | Confr. con ind.err. LO                 |
| ecd3: d0 30    | bne * \$ed05 |            | Se diverso salta                       |
| ecd5: a5 ad    | lda \$ad     | [ sah ]    | Esegui quanto sopra ma per             |
| ecd7: dd 01 01 | cmp \$0101,x |            | byte HI                                |
| ecda: d0 29    | bne * \$ed05 |            | Se <> salta                            |
| ecdc: e6 9f    | inc \$9f     | [ ptr2 ]   | Increment cont. correz errori in Pag.0 |
| ecde: e6 9f    | inc \$9f     | [ ptr2 ]   | per un passo 2x2                       |
| ece0: a5 93    | lda \$93     | [ verck ]  | Controllo per verify marker            |
| ece2: f0 0c    | beq * \$ecf0 |            | Se neg. fissa marker                   |
| ece4: a0 00    | ldy #\$00    |            | Spost. per routine                     |
| ece6: 20 cc f7 | jsr * \$f7cc |            | Rout.per chiam. LVS                    |
| ece9: c5 bd    | cmp \$bd     | [ roppty ] | Contr.byte letto =byte in memoria      |
| ecfb: f0 18    | beq * \$ed05 |            | Se pos. salta                          |
| eced: c8       | iny          |            | Incrempuntatore                        |
| ecce: 84 b6    | sty \$b6     | [ prp ]    | Immetti in punt.err. pag. 0            |
| ecf0: a5 b6    | lda \$b6     | [ prp ]    | Contr.rilev. errore                    |
| ecf2: f0 07    | beq * \$ecfb |            | Se nessun errore salta                 |
| ecf4: a9 10    | lda #\$10    |            | Bit 4 lett. errori non corretti        |
| ecf6: 20 57 f7 | jsr * \$f757 |            | Ese. azz.punt. di sistema              |
| ecf9: d0 0a    | bne * \$ed05 |            | Salto incond. a ind.                   |
| ecfb: a5 93    | lda \$93     | [ verck ]  | Controlla verify marker                |
| ecfd: d0 06    | bne * \$ed05 |            | Se pos,. salta                         |
| ecff: a8       | tay          |            | Metti a 0 punt. spostam.               |
| ed00: a5 bd    | lda \$bd     | [ roppty ] | Metti byte in A                        |
| ed02: 20 bc f7 | jsr * \$f7bc |            | Vai a rout. STASH                      |
| ed05: 20 cl ee | jsr * \$eecl |            | Incr ind. di inizio I/O                |
| ed08: d0 44    | bne * \$ed4e |            | Vai a IRQ.                             |
| ed0a: a9 80    | lda #\$80    |            | Metti a fine cod.punt.lettura          |
| ed0c: 85 aa    | sta \$aa     | [ rdflg ]  | Metti in A punt.lett.nastro            |
| ed0e: 78       | sei          |            | Disabilita tutti interrupt             |
| ed0f: a2 01    | ldx #\$01    |            | Cari.val. per interrupt timer A        |
| ed11: 8e 0d dc | stx * \$dc0d |            | Disabilita ICR                         |
| ed14: ae 0d dc | ldx * \$dc0d |            | Es. reset puntat. interrupt            |
| ed17: a6 be    | ldx \$be     | [ fsblk ]  | Contr. seil n. di blocchi che resta    |
| ed19: ca       | dex          |            | da proces. =0                          |
| ed1a: 30 02    | bmi * \$edle |            | Se = 0 salta                           |
| ed1c: 86 be    | stx \$be     | [ fsblk ]  | Immag. nuovo n.in Pag.0                |
| ed1e: c6 a7    | dec \$a7     | [ shcnl ]  | Decr. di 1 cont. blocchi in Pag.0      |

|                |              |            |   |
|----------------|--------------|------------|---|
| ed20: f0 08    | beg * \$ed2a |            | Contr. se cont. sopra =0. Quindi sal.   |
| ed22: a5 9e    | lda \$9e     | [ ptrl ]   | Contr. pres. errore in passo 1          |
| ed24: d0 28    | bne * \$ed4e |            | Se pres. salta                          |
| ed26: 85 be    | sta \$be     | [ fsblk ]  | Contr. n. blocchi da proces=0           |
| ed28: f0 24    | beg * \$ed4e |            | Se pos. vai a IRQ                       |
| ed2a: 20 57 ee | jsr * \$ee57 |            | Vai a rout. fine nastro I/O             |
| ed2d: 20 51 ed | jsr * \$ed51 |            | Copia ind. part. in punt.               |
| ed30: a0 00    | ldy #\$00    |            | Metti a 0 punt. in Pag. 0 per CHECKSUM  |
| ed32: 84 ab    | sty \$ab     | [ riprty ] | Metti cont. spost. a 0                  |
| ed34: 20 cc f7 | jsr * \$f7cc |            | Vai a rout. FETCH                       |
| ed37: 45 ab    | eor \$ab     | [ riprty ] | Opera su byte in mem. e CHECKSUM e      |
| ed39: 85 ab    | sta \$ab     | [ riprty ] | immag. in punt. di CHECKSUM             |
| ed3b: 20 c1 ee | jsr * \$eecl |            | Increment. ind. iniziale I/O            |
| ed3e: 20 b7 ee | jsr * \$eeb7 |            | Contr. ritrov. ind. di fine             |
| ed41: 90 f1    | bcc * \$ed34 |            | Se nessun ind. fine cont.               |
| ed43: a5 ab    | lda \$ab     | [ riprty ] | Confronta CHECKSUM generato con         |
| ed45: 45 bd    | eor \$bd     | [ roprty ] | quello letto                            |
| ed47: f0 05    | beg * \$ed4e |            | Se = allora OK e continua               |
| ed49: a9 20    | lda #\$20    |            | Fissa bit 5 per errore di CHECKSUM      |
| ed4b: 20 57 f7 | jsr * \$f757 |            | Esegui reset punt. di sistema           |
| ed4e: 4c 33 ff | jmp * \$ff33 |            | Vai a rout. IRQ normale                 |
| ed51: a5 c2    | lda \$c2     | [ stah ]   | Copia ind. iniz. I/O val. HI            |
| ed53: 85 ad    | sta \$ad     | [ sah ]    | Immag. in A val. HI                     |
| ed55: a5 c1    | lda \$c1     | [ track ]  | Copia ind. I/O LO                       |
| ed57: 85 ac    | sta \$ac     | [ sal ]    | Immag. in A val. LO                     |
| ed59: 60       | rts          |            |   |
| -----          |              |            |   |
| ed5a: a9 08    | lda #\$08    |            | Iniz. trasfer. contatore per 8 bits     |
| ed5c: 85 a3    | sta \$a3     | [ pcntr ]  | in Pag. 0                               |
| ed5e: a9 00    | lda #\$00    |            | Azzera byte HI dei 2 byte               |
| ed60: 85 a4    | sta \$a4     | [ firt ]   | Metti cont. in Pag. 0 per uscita a \$00 |
| ed62: 85 a8    | sta \$a8     | [ bitci ]  | Azzera flag lett. errori nastro         |
| ed64: 85 9b    | sta \$9b     | [ prty ]   | Inizializ. parita' per nastro           |
| ed66: 85 a9    | sta \$a9     | [ rinone ] | Inizializ. flag zero                    |
| ed68: 60       | rts          |            |   |
| -----          |              |            |   |
| ed69: a5 bd    | lda \$bd     | [ roprty ] | Metti in A bit per uscita da Pag. 0     |
| ed6b: 4a       | lsr a        |            | Idem ma in Carry                        |
| ed6c: a9 60    | lda #\$60    |            | Fissa temp. per bit 0                   |
| ed6e: 90 02    | bcc * \$ed72 |            | Fissa temp. per uscita                  |
| ed70: a9 b0    | lda #\$b0    |            | Fissa temp. per bit 1                   |
| ed72: a2 00    | ldx #\$00    |            | Metti val. LO per temp. byte HI         |
| ed74: 8d 06 dc | sta * \$dc06 |            | Temp. B di CIA1 Byte LO                 |
| ed77: 8e 07 dc | stx * \$dc07 |            | Temp. B di CIA1 Byte HI                 |
| ed7a: ad 0d dc | lda * \$dc0d |            | Es. clear su flag di interrupt          |
| ed7d: a9 19    | lda \$19     |            | Carica timer B e inizia                 |
| ed7f: 8d 0f dc | sta * \$dc0f |            | Car. reg. di contr. CIA                 |
| ed82: a5 01    | lda \$01     | [ r6510 ]  | Invers. val. per uscita bit             |
| ed84: 49 08    | eor #\$08    |            | Invers. su porta process.               |
| ed86: 85 01    | sta \$01     | [ r6510 ]  | Riportati su porta prec. ancora         |
| ed88: 29 08    | and #\$08    |            | Salva att. segnale                      |
| ed8a: 60       | rts          |            |   |
| -----          |              |            |   |
| ed8b: 38       | sec          |            | Fissa Carry per rotaz.                  |
| ed8c: 66 b6    | ror \$b6     | [ prp ]    | Esegui q.s.                             |
| ed8e: 30 3c    | bmi * \$edcc |            | Ritorno da interrupt                    |
| ed90: a5 a8    | lda \$a8     | [ bitci ]  | Contr. scritt. byte di impulso          |

|       |       |        |                 |  |
|-------|-------|--------|-----------------|--|
| ed92: | d0 12 | bne    | * \$eda6        | Se contr. pos. scrivi byte             |
| ed94: | a9 10 | lda    | #\$10           | Metti in A val. LO di byte freq.       |
| ed96: | a2 01 | ldx    | #\$01           | Metti in X val. HI byte freq.          |
| ed98: | 20 74 | ed jsr | * \$ed74        | Scrivi su nastro byte imp.             |
| ed9b: | d0 2f | bne    | * \$edcc        | Se e' la prima onda mez.vai a IRQ      |
| ed9d: | e6 a8 | inc    | \$a8 [ bitci ]  | Fissa punt. per impul.scritto          |
| ed9f: | a5 b6 | lda    | \$b6 [ prp ]    | Contr. punt. blocco scrittura          |
| edal: | 10 29 | bpl    | * \$edcc        | Se contr.pos.a IRQ                     |
| eda3: | 4c 1b | ee jmp | * \$eelb        | Se blocco termin. cont. scritt.        |
| eda6: | a5 a9 | lda    | \$a9 [ rinone ] | Contr.per scritt. impul. lungo         |
| eda8: | d0 09 | bne    | * \$edb3        | Se pos. vai a impul.lungo              |
| edaa: | 20 70 | ed jsr | * \$ed70        | Scrivi impul.prec.su nastro            |
| edad: | d0 1d | bne    | * \$edcc        | Controllo onda pos. a IRQ              |
| edaf: | e6 a9 | inc    | \$a9 [ rinone ] | Fissa punt. per scrittura              |
| edbl: | d0 19 | bne    | * \$edcc        | Vai a IRQ                              |
| edb3: | 20 69 | ed jsr | * \$ed69        | Scrivi un bit su nastro                |
| edb6: | d0 14 | bne    | * \$edcc        | A IRQ                                  |
| edb8: | a5 a4 | lda    | \$a4 [ firt ]   | Inverti punt.bit imp. in pag.0         |
| edba: | 49 01 | eor    | #\$01           | esegui                                 |
| edbc: | 85 a4 | sta    | \$a4 [ firt ]   | e salva di nuovo                       |
| edbe: | f0 0f | beq    | * \$edcf        | Se 0 scrivi entrambi gli impulsi       |
| edc0: | a5 bd | lda    | \$bd [ roprty ] | Es.invers. in pag.0 del bit 0          |
| edc2: | 49 01 | eor    | #\$01           | sposta immagazz. ed esegui op.         |
| edc4: | 85 bd | sta    | \$bd [ roprty ] | Salva ancora su A                      |
| edc6: | 29 01 | and    | #\$01           | Canc. bit att.ed esegui un AND con     |
| edc8: | 45 9b | eor    | \$9b [ prty ]   | bit di parita' del Byte                |
| edca: | 85 9b | sta    | \$9b [ prty ]   | Immaga. nel flag di parita'            |
| edcc: | 4c 33 | ff jmp | * \$fff33       | Vai a IRQ                              |
| edcf: | 46 bd | lsr    | \$bd [ roprty ] | Es. spost. su bit e decrem. di 1       |
| edd1: | c6 a3 | dec    | \$a3 [ pcntr ]  | il cont. in pag.0 dei bit              |
| edd3: | a5 a3 | lda    | \$a3 [ pcntr ]  | Controlla se ril. segn di fine         |
| edd5: | f0 3b | bec    | * \$eel2        | Se pos. genera bit di parita'          |
| edd7: | 10 f3 | bpl    | * \$edcc        | Se neg. vai a IRQ                      |
| edd9: | 20 5a | ed jsr | * \$ed5a        | Contatore bit per uscita seriale       |
| eddc: | 58    | cli    |                 | Abilita tutti gli interrupt            |
| eddd: | a5 a5 | lda    | \$a5 [ cntdn ]  | Contr. scritt.bytes di sincron.        |
| eddf: | f0 12 | beq    | * \$edf3        | Se sono stati scritti salta            |
| edel: | a2 00 | ldx    | #\$00           | Metti a 0 immag. di CHECKSUM per eseg  |
| ede3: | 86 c5 | stx    | \$c5 [ data ]   | lettura buffer                         |
| ede5: | c6 a5 | dec    | \$a5 [ cntdn ]  | Decr.cont. sincron. di 1               |
| ede7: | a6 be | ldx    | \$be [ fsblk ]  | Contr. se il primo blocco e' gia'      |
| ede9: | e0 02 | cpx    | #\$02           | stato letto                            |
| edeb: | d0 02 | bne    | * \$edef        | Se neg. salta                          |
| eded: | 09 80 | ora    | #\$80           | Bit 7 in byte di sincronizz.           |
| edef: | 85 bd | sta    | \$bd [ roprty ] | e in pag. 0 con spost. bit prer immag. |
| edf1: | d0 d9 | bne    | * \$edcc        | Vai a IRQ                              |
| edf3: | 20 b7 | ee jsr | * \$eeb7        | Contr.rilev indir. di fine             |
| edf6: | 90 0a | bcc    | * \$ee02        | Se non trovato cont. a scrivere        |
| edf8: | d0 91 | bne    | * \$ed8b        | Fissa punt. per blocco scritto         |
| edfa: | e6 ad | inc    | \$ad [ sah ]    | Incr. di 1 att. ind. byte              |
| edfc: | a5 c5 | lda    | \$c5 [ data ]   | Carica buffer di CHECKSUM da pag.0     |
| edfe: | 85 bd | sta    | \$bd [ roprty ] | Immag. q.s. in zona immag. bit         |
| ee00: | b0 ca | bcs    | * \$edcc        | Vai a normale IRQ                      |
| ee02: | a0 00 | ldy    | #\$00           | Metti a 0 punt. spostamento            |
| ee04: | 20 c8 | f7 jsr | * \$f7cc        | Vai a rout. FETCH                      |
| ee07: | 85 bd | sta    | \$bd [ roprty ] | Imma. caratt. in zona immag. bit       |
| ee09: | 45 c5 | eor    | \$c5 [ data ]   | Esegui EOR con CHEKSUM immag. in prec. |
| ee0b: | 85 c5 | sta    | \$c5 [ data ]   | e immag. ancora                        |
| ee0d: | 20 c1 | ee jsr | * \$eecl        | Incr. ind. di partenza I/O             |
| eel0: | d0 ba | bne    | * \$edcc        | Vai a normale IRQ                      |

|                |               |            |  |
|----------------|---------------|------------|--|
| ee12: a5 9b    | lda \$9b      | [ prty ]   | Inverti bit di parita' del byte        |
| ee14: 49 01    | eor #\$01     |            | di pag. 0 e copialo nell' A            |
| ee16: 85 bd    | sta \$bd      | [ roppty ] |  |
| ee18: 4c 33 ff | jmp * \$fff33 |            | Vai a IRQ                              |
| ee1b: c6 be    | dec \$be      | [ fsblk ]  | Contr. per scritt. tutti i bits        |
| ee1d: d0 03    | bne * \$ee22  |            | Se contr. neg. salta                   |
| ee1f: 20 b0 ee | jsr * \$eeb0  |            | Disattiva mot. cassetta                |
| ee22: a9 50    | lda #\$50     |            | Iniz. contatore in pag. 0 per          |
| ee24: 85 a7    | sta \$a7      | [ shcnl ]  | imp. piu' corto                        |
| ee26: a2 08    | ldx #\$08     |            | Esegui spost. per IRQ 1 (scrittura)    |
| ee28: 78       | sei           |            | Disabilita tutti gli interrupt         |
| ee29: 20 9b ee | jsr * \$ee9b  |            | Fissa i vettori di IRQ                 |
| ee2c: d0 ea    | bne * \$ee18  |            | Vai a normale IRQ                      |
| ee2e: a9 78    | lda #\$78     |            | Metti in A codice per imp. HEADER      |
| ee30: 20 72 ed | jsr * \$ed72  |            | Scrivi q.s                             |
| ee33: d0 e3    | bne * \$ee18  |            | Se e' primo mezzo segn. vai a IRQ      |
| ee35: c6 a7    | dec \$a7      | [ shcnl ]  | Decr. cont. di HEADER di 1             |
| ee37: d0 df    | bne * \$ee18  |            | Se non ril. fine vai a IRQ             |
| ee39: 20 5a ed | jsr * \$ed5a  |            | Fissa cont. bit per uscita seriale     |
| ee3c: c6 ab    | dec \$ab      | [ rippty ] | Decr.durata imp.corto                  |
| ee3e: 10 d8    | bpl * \$ee18  |            | Se non e' inc.segn di fine vai a IRQ   |
| ee40: a2 0a    | ldx #\$0a     |            | Spostam. per IRQ 2                     |
| ee42: 20 9b ee | jsr * \$ee9b  |            | Fissa vettore di IRQ                   |
| ee45: 58       | cli           |            | Abilita tutti gli interrupt            |
| ee46: e6 ab    | inc \$ab      | [ rippty ] | Incr. lungh. impulso                   |
| ee48: a5 be    | lda \$be      | [ fsblk ]  | Controlla per scritt. tutti blocchi    |
| ee4a: f0 49    | beq * \$ee95  |            | Se pos. salta                          |
| ee4c: 20 51 ed | jsr * \$ed51  |            | Copia ind. di fine I/O                 |
| ee4f: a2 09    | ldx #\$09     |            | Esegui un reset sul cont. di pag.0     |
| ee51: 86 a5    | stx \$a5      | [ cntdn ]  | per sincronizz. e resetta punt.        |
| ee53: 86 b6    | stx \$b6      | [ prp ]    | blocchi scritti                        |
| ee55: d0 82    | bne * \$edd9  |            | Salata a ind.                          |
| ee57: 08       | php           |            | Salva stato del proc. su Stack         |
| ee58: 78       | sei           |            | Disabilita tutti gli interrupt         |
| ee59: ad 11 d0 | lda * \$d011  |            | Metti in A cont.reg.contr. VIC         |
| ee5c: 0d 39 0a | ora \$0a39    |            | Es. OR con punt. tempor. VDC           |
| ee5f: 29 7f    | and #\$7f     |            | Disatt. schermo                        |
| ee61: 8d 11 d0 | sta * \$d011  |            | Scrivi valore in reg. VIC              |
| ee64: 2c 3a 0a | bit \$0a3a    |            | Contr. immag. IRQ                      |
| ee67: 30 16    | bmi * \$ee7f  |            | Fissa bit 7 e salta a ind.             |
| ee69: 2c 37 0a | bit \$0a37    |            | Contr. immag. freq. clock              |
| ee6c: 10 11    | bpl * \$ee7f  |            | Se bit 7 a 0 nessun aggiornam.         |
| ee6e: ad 38 0a | lda \$0a38    |            | Carica status per sprites              |
| ee71: 8d 15 d0 | sta * \$d015  |            | Fissa reg. visual. sprites             |
| ee74: ad 37 0a | lda \$0a37    |            | Metti in A freq.clock salvato in prec. |
| ee77: 8d 30 d0 | sta * \$d030  |            | e riporta il sistema al val. prec.     |
| ee7a: a9 00    | lda #\$00     |            | Metti 0 in A per es.clear di           |
| ee7c: 8d 37 0a | sta \$0a37    |            | frequen. clock di sistema              |
| ee7f: 20 b0 ee | jsr * \$eeb0  |            | Disattiva motore                       |
| ee82: 20 b8 e1 | jsr * \$elb8  |            | Metti temporiz. e i 2 CIA in modo sta  |
| ee85: ad 0a 0a | lda \$0a0a    |            | Contr. se vett.di interrupt e' stand.  |
| ee88: f0 09    | beq * \$ee93  |            | Se positivo esci                       |
| ee8a: 8d 15 03 | sta \$0315    |            | Metti a modo stand.vett. HI di IRQ     |
| ee8d: ad 09 0a | lda \$0a09    |            | Carica ind. LO di IRQ                  |
| ee90: 8d 14 03 | sta \$0314    |            | Metti in modo stand.vett.LO di IRQ     |
| ee93: 28       | plp           |            | Riprist. status proec.                 |
| ee94: 60       | rts           |            |  |

-----

ee95: 20 57 ee jsr \* \$ee57

Termine operaz.di registr.



```

ee98: 4c 33 ff jmp * $ff33
ee9b: bd a0 ee lda * $eea0,x
ee9e: 8d 14 03 sta $0314
eeal: bd a1 ee lda * $eeal,x
eea4: 8d 15 03 sta $0315
eea7: 60 rts

```

Vai a IRQ. Vengono fiss.i vett.IRQ  
 IRQ indirizzo LO  
 Copia q.s in vett.di sistema IRQ  
 IRQ Indirizzo HI  
 Copia q.s.in vett.di sistema IRQ

## TAVOLA VETTORI DI IRQ

```

eea8: 2e ee ($ee2e)
eeaa: 90 ed ($ed90)
eeac: 65 fa ($fa65)
eeae: eb ea ($eaeB)

```

IRQ 1:scrivi HEADER su nastro  
 IRQ 2:scrivi BUFFER su nastro  
 IRQ per lettura tastiera  
 IRQ per lettura da nastro

```

eeb0: a5 01 lda $01 [ r6510 ]
eeb2: 09 20 ora #$20
eeb4: 85 01 sta $01 [ r6510 ]
eeb6: 60 rts

```

Car. A per disab. motore  
 Fissa bit 5  
 Disabil. motore cassetta

```

eeb7: 38 sec
eeb8: a5 ac lda $ac [ sal ]
eeba: e5 ae sbc $ae [ eal ]
eebc: a5 ad lda $ad [ sah ]
eebe: e5 af sbc $af [ eah ]
eec0: 60 rts

```

Predisp. carry per sottraz.  
 Metti in A ind.part.LO di I/O  
 Sottrai ind. finale corrisp  
 Metti in A ind.part.HI di I/O  
 Sottrai ind. finale corrisp

```

eec1: e6 ac inc $ac [ sal ]
eec3: d0 02 bne * $eec7
eec5: e6 ad inc $ad [ sah ]
eec7: 60 rts

```

Incr.di 1 ind.part.LO di I/O  
 Se per val.LO nessun Overflow vai ind.  
 Incr.c.s.ma per val. HI

```

eec8: 08 php
eec9: 68 pla
eeea: 29 ef and #$ef
eecc: 48 pha
eedc: 4c 17 ff jmp * $ff17
eed0: a5 01 lda $01 [ r6510 ]
eed2: 29 10 and #$10
eed4: f0 0a beq * $eee0
eed6: a0 00 ldy #$00
eed8: 84 c0 sty $c0 [ cas1 ]
eeda: a5 01 lda $01 [ r6510 ]
eedc: 09 20 ora #$20
eede: d0 08 bne * $eee8
eee0: a5 c0 lda $c0 [ cas1 ]
eee2: d0 06 bne * $eeea
eee4: a5 01 lda $01 [ r6510 ]
eee6: 29 df and #$df
eee8: 85 01 sta $01 [ r6510 ]
eeea: 60 rts

```

Salva su stack status del proc.  
 Copia q.s. in A  
 Esegui un clear sul flag di break  
 Reins. status su Stack  
 Vai a KERNAL IRQ  
 Contr. tasto prem. di cassetta  
 Esegui un AND per tasto prem.  
 Se nessun tasto prem. esci  
 Indicat. cass.  
 Eseg.reset flag nastro (OFF)in pag.0  
 Metti in A reg.datì porta process.  
 Fissa bit per disab. motore  
 Es. salto incond.  
 Contr.flag mot.nastro in pag.0  
 Se mot. attivo vai ad ind.  
 Metti in A reg.datì porta process.  
 Es.clear del bit per abilit. motore  
 Riscr. in porta del process.

## ROUTINE KERNAL GETIN

```

eeeb: a5 99 lda $99 [ dfltn ]
eeed: d0 0a bne * $eef9
eeef: a5 d0 lda $d0 [ ndx ]
eef1: 05 d1 ora $d1 [ kyndx ]
eef3: f0 0f beq * $ef04
eef5: 78 sei

```

Carica A con att.perif. in ingresso  
 Se non e' la tast. continua  
 Metti in A n. car. in buffer tastiera  
 Eseg. OR con punt. tasto funz.  
 Se nessun carattere esci  
 Disabilita tutti gli interrupt

```

eef6: 4c 06 c0 jmp * $c006
eef9: c9 02 cmp #$02
eefb: d0 18 bne * $ef15
eefd: 84 97 sty $97 [ xsav ]
eeff: 20 ce e7 jsr * $e7ce
ef02: a4 97 ldy $97 [ xsav ]
ef04: 18 clc
ef05: 60 rts

```

Prendi un car. da buffer di tastiera  
 Contr. se RS-232 e' perif di input  
 Se non e' RS-232 vai a BASIN  
 Immag. att. cont. di reg. Y  
 Rout. GETIN di RS-232  
 Riprist. cont. di Y  
 Clear di carry come marker

---

 ROUTINE KERNAL BASIN

```

ef06: a5 99 lda $99 [ dfltn ]
ef08: d0 0b bne * $ef15
ef0a: a5 ec lda $ec [ pntr ]
ef0c: 85 e9 sta $e9 [ lstp ]
ef0e: a5 eb lda $eb [ tblx ]
ef10: 85 e8 sta $e8 [ lsexp ]
ef12: 4c 09 c0 jmp * $c009
ef15: c9 03 cmp #$03
ef17: d0 09 bne * $ef22
ef19: 85 d6 sta $d6 [ crsm ]
ef1b: a5 e7 lda $e7 [ scrt ]
ef1d: 85 ea sta $ea [ indx ]
ef1f: 4c 09 c0 jmp * $c009
ef22: b0 38 bcs * $ef5c
ef24: c9 02 cmp #$02
ef26: f0 3f beq * $ef67
ef28: 86 97 stx $97 [ xsav ]
ef2a: 20 48 ef jsr * $ef48
ef2d: b0 16 bcs * $ef45
ef2f: 48 pha
ef30: 20 48 ef jsr * $ef48
ef33: b0 0d bcs * $ef42
ef35: d0 05 bne * $ef3c
ef37: a9 40 lda #$40
ef39: 20 57 f7 jsr * $f757
ef3c: c6 a6 dec $a6 [ bufpt ]
ef3e: a6 97 ldx $97 [ xsav ]
ef40: 68 pla
ef41: 60 rts

```

Metti in A att.perif. in input  
 Se non e' la tastiera continua  
 Metti att.colocurs. in A  
 Immag. in Pag.0 inizio col.input  
 Metti att.linea cursore in A  
 Immag. in pag.0 inizio linea input  
 Prel. carat. da schermo  
 Contr.se perif in input e' lo scher.  
 Se non e' lo schermo continua  
 Immag. in pag.0 punt. per input(get)  
 Metti in A bordo destro finestra  
 Immag.in pag.0 q.s.per fine linea in.  
 Prel.car. da schermo  
 Contr.se n.perif.>3.prel.car.da bus  
 Contr.se n.perif in input=2  
 Se contr. pos.prel.car.da RS-232  
 Salva att. cont. di X  
 Leggi un car. da nastro  
 Esci da rout. lett. cassetta  
 Salva A su Stack  
 Leggi car. da cassetta  
 Se rilev. errore salta  
 Contr.per ult.caratt.letto da nastro  
 Metti EOF in A  
 Fissa di conseg. il b. di STATUS  
 Decr.di 1 punt.buffer nastro  
 Riprist. cont.reg. X  
 Ripris. A con dati da Stack

```

ef42: aa tax
ef43: 68 pla
ef44: 8a txa
ef45: a6 97 ldx $97 [ xsav ]
ef47: 60 rts

```

Mettti in X n.errore  
 Prel. caratt.da Stack  
 Metti n.err. in A  
 Riprist. cont. reg. X

```

ef48: 20 be e9 jsr * $e9be
ef4b: d0 0b bne * $ef58
ef4d: 20 f2 e9 jsr * $e9f2
ef50: b0 09 bcs * $ef5b
ef52: a9 00 lda #$00
ef54: 85 a6 sta $a6 [ bufpt ]
ef56: f0 f0 beq * $ef48
ef58: b1 b2 lda ($b2),y[ tapel ]
ef5a: 18 clc
ef5b: 60 rts

```

Incrementa punt.buffer nastro e poi  
 leggi cara. da nastro  
 Leggi pross. blocco da cassetta  
 Contr. tasto premuto  
 Carica A con \$00  
 Immag. in A punt.buffer nastro  
 Vai al pross.caratt.  
 Leggi cara. da buffer  
 Clear di carry per indic. di OK

```

ef5c: a5 90 lda $90 [ status ]
ef5e: d0 03 bne * $ef63

```

Metti in A STATUS del sistema  
 Contr. se q.s. e' ok

|                     |                        |  |
|---------------------|------------------------|--|
| ef60: 4c 3e e4      | jmp * \$e43e           | Vai a Rout.ACPTR                       |
| ef63: a9 0d         | lda #\$0d              | Metti in A cod. per CR                 |
| ef65: 18            | clc                    | Clear di carry per indic. di OK        |
| ef66: 60            | rts                    |  |
| -----               |                        |  |
| ef67: 20 fd ee      | jsr * \$eefd           | Leggi un byte da RS-232                |
| ef6a: b0 f9         | bcs * \$ef65           | Se ril. errore esci                    |
| ef6c: c9 00         | cmp #\$00              | Contr. se car. letto e' byte 0         |
| ef6e: d0 f6         | bne * \$ef66           | Se contr.neg. OK e esci                |
| ef70: ad 14 0a      | lda \$0a14             | Metti in A STATUS RS-232               |
| ef73: 29 60         | and #\$60              | Contr. pres. DSR                       |
| ef75: d0 ec         | bne * \$ef63           | Se pos. ripo. codice CR                |
| ef77: f0 ee         | beq * \$ef67           | Se neg. leggi di nuovo                 |
| ef79: 48            | pha                    | Immag. car. in uscita                  |
| ef7a: a5 9a         | lda \$9a [ dflto ]     | Metti in A att. perif in uscita        |
| ef7c: c9 03         | cmp #\$03              | Contr. se q.s. e' 3(per schermo)       |
| ef7e: d0 04         | bne * \$ef84           | Se neg.vai a uscita schermo            |
| ef80: 68            | pla                    | Metti il car. in uscita nella rout.    |
| ef81: 4c 0c c0      | jmp * \$c00c           | di uscita car.di schermo               |
| ef84: 90 04         | bcc * \$ef8a           | Uscita su RS-232 o cassetta            |
| ef86: 68            | pla                    | Prel. car. da Stack                    |
| ef87: 4c 03 e5      | jmp * \$e503           | Vai a BSOUT per uscita seriale         |
| ef8a: 4a            | lsr a                  | Contr. se RS-232 o cassetta            |
| ef8b: 68            | pla                    | Prel. car. da Stack                    |
| ef8c: 85 9e         | sta \$9e [ ptr1 ]      | Immag. q.s. in pag. 0                  |
| ef8e: 8a            | txa                    | Metti in A att. cont. di X             |
| ef8f: 48            | pha                    | Metti q.s. su Stack per mezzo di A     |
| ef90: 98            | tya                    | Trasf. su Stack att. contenuto di Y    |
| ef91: 48            | pha                    | per mezzo di A                         |
| ef92: 90 23         | bcc * \$efb7           | Vai a uscita RS-232                    |
| ef94: 20 be e9      | jsr * \$e9be           | Incr. punt. buffer nastro              |
| ef97: d0 0e         | bne * \$efa7           | Contr. se buffer non pieno e metti.car |
| ef99: 20 15 ea      | jsr * \$ea15           | Scrivi cont. buffer su nastro          |
| ef9c: b0 0e         | bcs * \$efac           | Contr. per STOP premuto                |
| ef9e: a9 02         | lda #\$02              | Fissa byte di contr.per blocco dati    |
| efa0: a0 00         | ldy #\$00              | Sposta lett. su buffer di nastro       |
| efa2: 91 b2         | sta (\$b2),y [ tapel ] | Scrivi byte di controllo su buffer     |
| efa4: c8            | iny                    | Incr.spostam. in buffer nastro e       |
| efa5: 84 a6         | sty \$a6 [ bufpt ]     | immag. in Y di pag. 0                  |
| efa7: a5 9e         | lda \$9e [ ptr1 ]      | Prel. car. uscita da pag. 0            |
| efa9: 91 b2         | sta (\$b2),y [ tapel ] | Scrivilo in buffer di uscita           |
| efab: 18            | clc                    | Clear di carry per indic. di OK        |
| efac: 68            | pla                    | Riprist. val. da Stack                 |
| efad: a8            | tay                    | Ripristina cont. di Y                  |
| efae: 68            | pla                    | Ripristina, prel. da Stack i cont.     |
| efaf: aa            | tax                    | di X                                   |
| efb0: a5 9e         | lda \$9e [ ptr1 ]      | Metti car. in uscita                   |
| efb2: 90 02         | bcc * \$efb6           | Se tutto e' OK vai ad ind.             |
| efb4: a9 00         | lda #\$00              | Fissa flag per tasto STOP premuto      |
| efb6: 60            | rts                    |  |
| -----               |                        |  |
| ROUTINE KERNAL OPEN |                        |  |
| efb7: 20 5f e7      | jsr * \$e75f           | Scrivi car. in buffer RS-232           |
| efba: 4c ab ef      | jmp * \$efab           | Contr. status Stack e return           |
| efbd: a6 b8         | ldx \$b8 [ 1a ]        | Metti n. di file logico in X           |
| efbf: 20 02 f2      | jsr * \$f202           | Cerca LFN in tavola relativa           |
| efc2: f0 2f         | beq * \$eff3           | Se rilev. LFN uscita errore            |
| efc4: a6 98         | ldx \$98 [ 1dtnnd ]    | Metti in X n. di files aperti          |
| efc6: e0 0a         | cpx #\$0a              | Contr.che non piu' di 10 files aper.   |

|                |     |          |           |  |
|----------------|-----|----------|-----------|--|
| efc8: b0 26    | bc  | * \$eff0 |           | Se piu' di 10 errore                   |
| efca: e6 98    | inc | \$98     | [ ldtnd ] | Incr. di 1 n.files aperti              |
| efcc: a5 b8    | lda | \$b8     | [ la ]    | Metti in A n. di file logico           |
| efce: 9d 62 03 | sta | \$0362,x |           | Immag. in A LFN                        |
| efd1: a5 b9    | lda | \$b9     | [ sa ]    | Metti in A ind. secondario             |
| efd3: 09 60    | ora | #\$60    |           | Fissa per PRINT, INPUT e GET           |
| efd5: 85 b9    | sta | \$b9     | [ sa ]    | Metti in A ancora                      |
| efd7: 9d 76 03 | sta | \$0376,x |           | Inserisci indir. sec in tavola relat   |
| efda: a5 ba    | lda | \$ba     | [ fa ]    | Metti in A indir.perif.                |
| efdc: 9d 6c 03 | sta | \$036c,x |           | Metti GA in tavola relativa            |
| efdf: f0 0d    | beq | * \$efee |           | Controllo se tastiera                  |
| efel: c9 02    | cmp | #\$02    |           | Contr. se RS-232 e' sel. come per.     |
| efe3: f0 5b    | beq | * \$f040 |           | Se pos. vai a RS-232                   |
| efe5: 90 0f    | bcc | * \$eff6 |           | Contr. per val. min. di 2.=OPEN nastro |
| efe7: c9 03    | cmp | #\$03    |           | Contr. se schermo selez.come perif.    |
| efe9: f0 03    | beq | * \$efee |           | Se pos. vai ad ind.                    |
| efeb: 20 cb f0 | jsr | * \$f0cb |           | Apertura file su bus seriale           |
| efee: 18       | clc |          |           | Clear di carry per tutto OK            |
| efef: 60       | rts |          |           |  |
| -----          |     |          |           |  |
| eff0: 4c 7c f6 | jmp | * \$f67c |           | Errore di I/O n.1.N. eccessivo filese  |
| eff3: 4c 7f f6 | jmp | * \$f67f |           | Errore di I/O n.2.File gia' aperto     |
| eff6: 20 80 e9 | jsr | * \$e980 |           | Vai a ind.iniz.buffer nastro           |
| eff9: b0 03    | bcs | * \$efee |           | Se carry a 1,ind. valido               |
| effb: 4c 94 f6 | jmp | * \$f694 |           | Errore di I/O n.9.N.perif.illegale     |
| effe: a5 b9    | lda | \$b9     | [ sa ]    | Metti in A ind. second.                |
| -----          |     |          |           |  |

|       |          |     |          |           |                                       |
|-------|----------|-----|----------|-----------|---------------------------------------|
| f000: | 29 0f    | and | #\$0f    |           | Maschera il Nibble alto (4-7)         |
| f002: | d0 1f    | bne | * \$f023 |           | Se non zero attesa Record/Play        |
| f004: | 20 c8 e9 | jsr | * \$e9c8 |           | Attesa per tasto su registratore      |
| f007: | b0 36    | bcs | * \$f03f |           | Errore se Carry=1, RTS                |
| f009: | 20 0f f5 | jsr | * \$f50f |           | Messaggio 'SEARCHING FOR'             |
| f00c: | a5 b7    | lda | \$b7     | [ fnlen ] | Lunghezza del nome file               |
| f00e: | f0 0a    | beq | * \$f01a |           | Salta se nessun nome                  |
| f010: | 20 9a e9 | jsr | * \$e99a |           | Cerca il nome nell'header del nastro  |
| f013: | 90 18    | bcc | * \$f02d |           | Continua se non trovato               |
| f015: | f0 28    | beq | * \$f03f |           | Ritorna se il Carry e' a 1            |
| f017: | 4c 85 f6 | jmp | * \$f685 |           | Errore di I/O n.4 (File not found)    |
| f01a: | 20 d0 e8 | jsr | * \$e8d0 |           | Cerca la prossima testata sul nastro  |
| f01d: | 90 0e    | bcc | * \$f02d |           | Continua se trovata                   |
| f01f: | f0 1e    | beq | * \$f03f |           | ***                                   |
| f021: | b0 f4    | bcs | * \$f017 |           | ***                                   |
| f023: | 20 e9 e9 | jsr | * \$e9e9 |           | Attesa Per tasti Record/Play          |
| f026: | b0 17    | bcs | * \$f03f |           | Tasto stop premuto, fine.             |
| f028: | a9 04    | lda | #\$04    |           | Contr. data-code della testata in Acc |
| f02a: | 20 19 e9 | jsr | * \$e919 |           | Scriva testata nel nastro             |
| f02d: | a9 bf    | lda | #\$bf    |           | Punt. fine buffer in Acc.             |
| f02f: | a4 b9    | ldy | \$b9     | [ sa ]    | Carica il reg. Y con l'ind. secondar  |
| f031: | c0 60    | cpy | #\$60    |           | Codice SA per Print, Input o get ?    |
| f033: | f0 07    | beq | * \$f03c |           | Se si, setta Punt. e ritorna          |
| f035: | a0 00    | ldy | #\$00    |           |                                       |
| f037: | a9 02    | lda | #\$02    |           | Byte di controllo per blocco dati.    |
| f039: | 91 b2    | sta | (\$b2),y | [ tapel ] | Scriva il Byte nel Buff. di cassetta  |
| f03b: | 98       | tya |          |           | Trasferisce il Reg. Y in Acc.         |
| f03c: | 85 a6    | sta | \$a6     | [ bufpt ] | Setta il Punt. in pag. zero del Buff  |
| f03e: | 18       | clc |          |           | Ripulisce il carry per OK             |
| f03f: | 60       | rts |          |           | Ritorno dalla Subroutine              |

-----  
Apertura Dell' RS232

|       |          |     |            |           |  |
|-------|----------|-----|------------|-----------|--|
| f040: | 20 b0 f0 | jsr | * \$f0b0   |           | Resetta i CIA                          |
| f043: | 8c 14 0a | sty | \$0a14     |           | Resetta il Byte di stato in Pag. Zero  |
| f046: | c4 b7    | cpy | \$b7       | [ fnlen ] | Confronta con la lungh. del nome file  |
| f048: | f0 0b    | beq | * \$f055   |           | Se uguale, calcola i data bits         |
| f04a: | 20 ae f7 | jsr | * \$f7ae   |           | Preleva 1 Byte per il Registro RS232   |
| f04d: | 99 10 0a | sta | \$0a10,y   |           | Inizializza il Registro di Controllo   |
| f050: | c8       | iny |            |           | Il Registro di Comando e               |
| f051: | c0 04    | cpy | #\$04      |           | La velocita (Baud rate)                |
| f053: | d0 f1    | bne | * \$f046   |           | Ciclo per settare tutti i 4 valori     |
| f055: | 20 8e e6 | jsr | * \$e68e   |           | Calcola il numero dei data bits        |
| f058: | 8e 15 0a | stx | \$0a15     |           | Immagazzina il num. dei bits da inviar |
| f05b: | ad 10 0a | lda | \$0a10     |           | Carica il Registro di Controllo RS232  |
| f05e: | 29 0f    | and | #\$0f      |           | Isola i bits del Baud Rate             |
| f060: | f0 1c    | beq | * \$f07e   |           | Determina il valore                    |
| f062: | 0a       | asl | a          |           | Moltiplica per 2                       |
| f063: | aa       | tax |            |           | Copia il contenuto nel reg. X          |
| f064: | ad 03 0a | lda | \$0a03     |           | Carica il puntatore PAL/NTSC           |
| f067: | d0 09    | bne | * \$f072   |           | Se non NTSC (USA), esce                |
| f069: | bc 4f e8 | ldy | * \$e84f,x |           | Costante di tempo per NTSC (alto)      |
| f06c: | bd 4e e8 | lda | * \$e84e,x |           | Costante di tempo per NTSC (basso)     |
| f06f: | 4c 78 f0 | jmp | * \$f078   |           | Salva costanti NTSC                    |
| f072: | bc 63 e8 | ldy | * \$e863,x |           | Costante di tempo per PAL (alto)       |
| f075: | bd 62 e8 | lda | * \$e862,x |           | Costante di tempo per PAL (basso)      |
| f078: | 8c 13 0a | sty | \$0a13     |           | Salva la parte alta del Baud rate      |
| f07b: | 8d 12 0a | sta | \$0a12     |           | Salva la parte bassa del Baud rate     |
| f07e: | ad 12 0a | lda | \$0a12     |           | Ricarica la parte bassa del Baud rate  |
| f081: | 0a       | asl | a          |           | Moltiplica per 2                       |

|       |          |     |          |                                       |
|-------|----------|-----|----------|---------------------------------------|
| f082: | aa       |     | tax      | Trasferisce detto valore nel reg. X   |
| f083: | ad 13 0a | lda | \$0a13   | Carica la parte alta Del Baud rate    |
| f086: | 2a       | rol | a        | moltiplica per 2                      |
| f087: | a8       | tay |          | Trsferisce il valore nel reg. Y       |
| f088: | 8a       | txa |          | Mette in Acc. la parte bassa del baud |
| f089: | 69 c8    | adc | #\$c8    | Somma il valore decimale 200          |
| f08b: | 8d 16 0a | sta | \$0a16   | Salva il valore del baud rate (trasm) |
| f08e: | 98       | tya |          | Carica l'acc. co nparte alta del Baud |
| f08f: | 69 00    | adc | #\$00    | Somma zero                            |
| f091: | 8d 17 0a | sta | \$0a17   | Salva il valore del baud rate (trasm) |
| f094: | ad 11 0a | lda | \$0a11   | Carica il val. del Regis. di Comando  |
| f097: | 4a       | lsr | a        | Cerca se modo 3 line.                 |
| f098: | 90 09    | bcc | * \$f0a3 | Se si salta controllo DSR             |
| f09a: | ad 01 dd | lda | * \$dd01 | Guarda se persente DSR                |
| f09d: | 0a       | asl | a        | Segnale DSR mancante                  |
| f09e: | b0 03    | bcs | * \$f0a3 | Se no salta                           |
| f0a0: | 20 55 e7 | jsr | * \$e755 | Setta lo status per DSR               |
| f0a3: | ad 18 0a | lda | \$0a18   | Mette l'inizio del buffer input RS232 |
| f0a6: | 8d 19 0a | sta | \$0a19   | Uguale alla fine del buff. di input   |
| f0a9: | ad 1b 0a | lda | \$0a1b   | Mette l'inizio del buff. di uscita    |
| f0ac: | 8d 1a 0a | sta | \$0a1a   | Uguale alla fine del buff. di uscita  |
| f0af: | 60       | rts |          | Ritornc dalla Subroutine              |

## ----- Resetta i CIA per RS232

|       |          |     |          |  |
|-------|----------|-----|----------|--|
| f0b0: | a9 7f    | lda | #\$7f    | Azzera gli Interrupts                  |
| f0b2: | 8d 0d dd | sta | * \$dd0d | nel CIA                                |
| f0b5: | a9 06    | lda | #\$06    | Predispone i byt 1 e 2 per uscita      |
| f0b7: | 8d 03 dd | sta | * \$dd03 | Scrive nel Reg. direzione dati(portaB) |
| f0ba: | 8d 01 dd | sta | * \$dd01 | Scrive nel Reg. di uscita (portaB)     |
| f0bd: | a9 04    | lda | #\$04    | predispone il reg. di uscita           |
| f0bf: | 0d 00 dd | ora | * \$dd00 | Della porta A come output. Per abilit. |
| f0c2: | 8d 00 dd | sta | * \$dd00 | Il Segnale di trasmissione (TXD)       |
| f0c5: | a0 00    | ldy | #\$00    | Carica il reg. Y con zero e            |
| f0c7: | 8c 0f 0a | sty | \$0a0f   | Cancella il flag di Rs232 NMI          |
| f0ca: | 60       | rts |          | Ritorna dalla Subroutine               |

## ----- Apertura del file sul bus seriale

|       |       |     |          |           |  |
|-------|-------|-----|----------|-----------|--|
| f0cb: | a5 b9 | lda | \$b9     | [ sa ]    | Mette il val. dell'ind. sec. nell'Acc. |
| f0cd: | 30 04 | bmi | * \$f0d3 |           | Esce se il byt 7 e' settato per CLOSE  |
| f0cf: | a4 b7 | ldy | \$b7     | [ fnlen ] | Carica la lunghezza del nome file      |
| f0d1: | d0 02 | bne | * \$f0d5 |           | Continua se non e' zero                |
| f0d3: | 18    | clc |          |           | Cancella il Carry per OK               |
| f0d4: | 60    | rts |          |           | Ritorna dalla subroutine               |

## ----- Invia il nome file sul Bus Seriale

|       |          |     |          |            |  |
|-------|----------|-----|----------|------------|--|
| f0d5: | a9 00    | lda | #\$00    |            | Azzera il Byte di stato                |
| f0d7: | 85 90    | sta | \$90     | [ status ] |  |
| f0d9: | a5 ba    | lda | \$ba     | [ fa ]     | Carica l'Acc. con il numero del Device |
| f0db: | 20 3e e3 | jsr | * \$e33e |            | Attesa per fine del Trsferimento RS232 |
| f0de: | 24 90    | bit | \$90     | [ status ] | Test del Byte di status per EOF        |
| f0e0: | 30 0b    | bmi | * \$f0ed |            | Se EOF, errore di uscita               |
| f0e2: | a5 b9    | lda | \$b9     | [ sa ]     | Carica l'Acc. con l'ind. Secondario    |
| f0e4: | 09 f0    | ora | #\$f0    |            |  |
| f0e6: | 20 d2 e4 | jsr | * \$e4d2 |            | Rout. SECND: SA per LISTEN             |
| f0e9: | a5 90    | lda | \$90     | [ status ] | Carica lo status in Acc.               |
| f0eb: | 10 05    | bpl | * \$f0f2 |            | Se OK continua.                        |
| f0ed: | 68       | pla |          |            | Toglie L'indirizzo RTS dallo STACK     |

|                |     |          |           |  |
|----------------|-----|----------|-----------|--|
| f0ee: 68       | pla |          |           | Toglie l'indirizzo RTS dallo STACK     |
| f0ef: 4c 88 f6 | jmp | * \$f688 |           | Errore di I/O n.5 (Device not Present) |
| f0f2: a5 b7    | lda | \$b7     | [ fnlen ] | Carica la lunghezza del nome file      |
| f0f4: f0 0d    | beq | * \$f103 |           | Esce se nessun nome                    |
| f0f6: a0 00    | ldy | #\$00    |           | Visual. primo carattere del nome file  |
| f0f8: 20 ae f7 | jsr | * \$f7ae |           | Legge 1 carattere del nome file        |
| f0fb: 20 03 e5 | jsr | * \$e503 |           | Kernal CROUT: Invia byte su bus serial |
| f0fe: c8       | iny |          |           | Incrementa il puntatore per visualizz  |
| f0ff: c4 b7    | cpy | \$b7     | [ fnlen ] | Controllo se visualizzati tutti        |
| f101: d0 f5    | bne | * \$f0f8 |           | Se no continua a inviare               |
| f103: 4c b0 f5 | jmp | * \$f5b0 |           | Salta a UNLSN su bus seriale e esce.   |

-----  
Routine Kernal CHKIN.  
Predispone il canale di input

|                |     |          |           |  |
|----------------|-----|----------|-----------|--|
| f106: 20 02 f2 | jsr | * \$f202 |           | Ricerca LFN nella tavola               |
| f109: d0 3e    | bne | * \$f149 |           | errore di I/O n.3 (file not found)     |
| f10b: 20 12 f2 | jsr | * \$f212 |           | Resetta LFN,DA,SA                      |
| f10e: f0 13    | beq | * \$f123 |           | Se DA=0, allora setta come standard    |
| f110: c9 03    | cmp | #\$03    |           | Da=3 (schermo) ?                       |
| f112: f0 0f    | beq | * \$f123 |           | Se si, setta lo schermo come standard  |
| f114: b0 11    | bcs | * \$f127 |           | Se maggiore di 3, sceglie bus seriale  |
| f116: c9 02    | cmp | #\$02    |           | Controlla se e' stata scelta l'RS232   |
| f118: d0 03    | bne | * \$f11d |           | Se no allora e' la cassetta            |
| f11a: 4c 95 e7 | jmp | * \$e795 |           | Setta l'RS232 come input               |
| f11d: a6 b9    | ldx | \$b9     | [ sa ]    | Carica X con l'ind. secondario         |
| f11f: e0 60    | cpx | #\$60    |           | Controlla se l'ind. second. e' = 0     |
| f121: d0 20    | bne | * \$f143 |           | Errore di I/O n.6 (Not input file)     |
| f123: 85 99    | sta | \$99     | [ dfltn ] | Mette in Pag. zero per stand.input.dev |
| f125: 18       | clc |          |           | Azzera il Carry per OK                 |
| f126: 60       | rts |          |           | Ritorna dalla subroutine               |

|                |     |          |            |  |
|----------------|-----|----------|------------|--|
| f127: aa       | tax |          |            | Trasferisce L'ind. del device in X       |
| f128: 20 3b e3 | jsr | * \$e33b |            | Rout. TALK: com. su bus seriale          |
| f12b: 24 90    | bit | \$90     | [ status ] | Test del byt di status                   |
| f12d: 30 11    | bmi | * \$f140 |            | Se bit 7 =1, allora (Device not Present) |
| f12f: a5 b9    | lda | \$b9     | [ sa ]     | Carica l'indirizzo secondario nell'Acc   |
| f131: 10 05    | bpl | * \$f138 |            | Invia l'ind. Secondario per TALK         |
| f133: 20 e9 e4 | jsr | * \$e4e9 |            | Attesa per il segnale di clock           |
| f136: 10 03    | bpl | * \$f13b |            | Salta all'uscita dell'ind.Sec di TALK    |
| f138: 20 e0 e4 | jsr | * \$e4e0 |            | Vai a rout. TKSA per invio Ind.Sec.      |
| f13b: 8a       | txa |          |            | Trasferisce l'ind. del device da A a X   |
| f13c: 24 90    | bit | \$90     | [ status ] | Test del bit di status per EOF           |
| f13e: 10 e3    | bpl | * \$f123 |            | Se tutto OK, fissa perif. di input       |
| f140: 4c 88 f6 | jmp | * \$f688 |            | Errore di I/O n.5 (device not present)   |
| f143: 4c 8b f6 | jmp | * \$f68b |            | Errore di I/O n.6 (Not input file)       |
| f146: 4c 8e f6 | jmp | * \$f68e |            | Errore di I/O n.7 (Not output file)      |
| f149: 4c 82 f6 | jmp | * \$f682 |            | errore di I/O n.3 (File not open)        |

-----  
Routine Kernal CKOUT

|                |     |          |  |                                       |
|----------------|-----|----------|--|---------------------------------------|
| f14c: 20 02 f2 | jsr | * \$f202 |  | Cerca per LFN nella tavola per LFN    |
| f14f: d0 f8    | bne | * \$f149 |  | Errore di I/O n.3 (File not open)     |
| f151: 20 12 f2 | jsr | * \$f212 |  | Resetta LFN,DA,SA                     |
| f154: f0 f0    | beq | * \$f146 |  | Errore di I/O n.7 (Not output file)   |
| f156: c9 03    | cmp | #\$03    |  | Controlla con DA 3 per schermo        |
| f158: f0 0f    | beq | * \$f169 |  | Se posit.mettilo come uscita standard |
| f15a: b0 11    | bcs | * \$f16d |  | Se DA >3, esegui val. seriale         |

|                |              |           |                                       |
|----------------|--------------|-----------|---------------------------------------|
| fl5c: c9 02    | cmp #02      |           | Contr. se e' selezionata l'RS-232     |
| fl5e: d0 03    | bne * \$f163 |           | Se neg. salta ad indirizzo            |
| fl60: 4c 29 e7 | jmp * \$e729 |           | Uscita RS-232                         |
| fl63: a6 b9    | ldx \$b9     | [ sa ]    | Metti indirizzo second.in X           |
| fl65: e0 60    | cpx #060     |           | Contr. se indir. sec. = 0             |
| fl67: f0 dd    | beq * \$f146 |           | Errore di I/O n.7 (Not output file)   |
| fl69: 85 9a    | sta \$9a     | [ dflto ] | Immag.in Pag.0 per perif.stand.uscita |
| fl6b: 18       | clc          |           | Esegui un clear di carry per OK       |
| fl6c: 60       | rts          |           |                                       |

|                |              |            |                                       |
|----------------|--------------|------------|---------------------------------------|
| fl6d: aa       | tax          |            | Trasf.in X indir.perif.per LISTN      |
| fl6e: 20 3e e3 | jsr * \$e33e |            | Vai a rout. LISTN                     |
| fl71: 24 90    | bit \$90     | [ status ] | Contr.STATUS per fissare bit EOF      |
| fl73: 30 cb    | bmi * \$f140 |            | Errore di I/O n. 5 (device not pres)  |
| fl75: a5 b9    | lda \$b9     | [ sa ]     | Carica in A indirizzo second.         |
| fl77: 10 05    | bpl * \$f17e |            | Esegui un clear su bit di OPEN/CLOSE  |
| fl79: 20 d7 e4 | jsr * \$e4d7 |            | Resetta il segnale di ATN             |
| fl7c: d0 03    | bne * \$f181 |            | Vai a uscita di indir. secondario     |
| fl7e: 20 d2 e4 | jsr * \$e4d2 |            | Trasf. in A indirizzo perif.          |
| fl81: 8a       | txa          |            | Controlla bit EOF sia settato         |
| fl82: 24 90    | bit \$90     | [ status ] | Errore di I/O n.5(Device not present) |
| fl84: 10 e3    | bpl * \$f169 |            | Se tutto OK vai a RTS                 |
| fl86: 30 b8    | bmi * \$f140 |            | Errore di I/O n.5(Device not present) |

## ROUTINE KERNAL CLOSE

|                |              |          |  |
|----------------|--------------|----------|--|
| fl88: 66 92    | ror \$92     | [ svxt ] | Esegui rotazione carry                 |
| fl8a: 20 07 f2 | jsr * \$f207 |          | Ricerca LFN nella relativa tavola      |
| fl8d: d0 dc    | bne * \$f16b |          | Se non trovato allora Oke return       |
| fl8f: 20 12 f2 | jsr * \$f212 |          | Vai a tavola corr. LFN,DA,SA           |
| fl92: 8a       | txa          |          | Tavola spostam.puntatore               |
| fl93: 48       | pha          |          | Salva su Stack                         |
| fl94: a5 ba    | lda \$ba     | [ fa ]   | Carica in A indirizzo perif.           |
| fl96: f0 4c    | beq * \$fle4 |          | Contr. se tastiera e' perif.indiriz    |
| fl98: c9 03    | cmp #03      |          | Controlla se la periferica indirizz.   |
| fl9a: f0 48    | beq * \$fle4 |          | era lo schermo(33). Se si salta ad ind |
| fl9c: b0 31    | bcs * \$flcf |          | Contr.se e' una perif. sul BUS seria.  |
| fl9e: c9 02    | cmp #02      |          | E' una RS-232                          |
| fla0: d0 07    | bne * \$fla9 |          | Se contr. neg. chiudi                  |
| fla2: 68       | pla          |          | Spostati sulla tavola                  |
| fla3: 20 e5 f1 | jsr * \$fle5 |          | Cancella file di ingresso da tavola    |
| fla6: 4c b0 f0 | jmp * \$f0b0 |          | Es. reset dei CIA e vai a RTS          |
| fla9: a5 b9    | lda \$b9     | [ sa ]   | Carica in A indir. secondario          |
| flab: 29 0f    | and #0f      |          | Esegui un mask out dei nublles super.  |
| flad: f0 35    | beq * \$fle4 |          | Cancella file ingr. da tavola          |
| flaf: 20 80 e9 | jsr * \$e980 |          | Prendi ind.buffer nastro e             |
| flb2: a9 00    | lda #00      |          | Fissa marker per chiusura              |
| flb4: 38       | sec          |          | Fissa il carry per contr.              |
| flb5: 20 8c ef | jsr * \$ef8c |          | Scrivi un caratt. nel buffer           |
| flb8: 20 15 ea | jsr * \$eal5 |          | Scrivi buffer su nastro                |
| flbb: 90 04    | bcc * \$flcl |          | Tutto OK,continua con chius.nastro     |
| flbd: 68       | pla          |          | Riporta uscita carattere               |
| flbe: a9 00    | lda #00      |          | Rimpiazza con CHR\$(0)                 |
| flc0: 60       | rts          |          |  |

|             |          |        |                               |
|-------------|----------|--------|-------------------------------|
| flcl: a5 b9 | lda \$b9 | [ sa ] | Carica in A indir. secondario |
|-------------|----------|--------|-------------------------------|



|       |          |     |          |           |                                       |
|-------|----------|-----|----------|-----------|---------------------------------------|
| flc3: | c9 62    | cmp | #\$62    |           | Contr.nibble basso di SA = 2          |
| flc5: | d0 1d    | bne | * \$fle4 |           | Canc. ingr.file da tavola             |
| flc7: | a9 05    | lda | #\$05    |           | Fissa byte di control.per EOT         |
| flc9: | 20 19 e9 | jsr | * \$e919 |           | Scrivi blocco dati su nastro          |
| flcc: | 4c e4 f1 | jmp | * \$fle4 |           | Cancella ingr.file da tavola          |
| flcf: | 24 92    | bit | \$92     | [ svxt ]  | Contr.cost. di tempo per nastro       |
| fld1: | 10 0e    | bpl | * \$fle1 |           | Se inf. di 128, invia chiusura        |
| fld3: | a5 ba    | lda | \$ba     | [ fa ]    | Metti in A Indirizzo perif.           |
| fld5: | c9 08    | cmp | #\$08    |           | Contr. se e' il drive                 |
| fld7: | 90 08    | bcc | * \$fle1 |           | Se neg. salta a chiusura disco        |
| fld9: | a5 b9    | lda | \$b9     | [ sa ]    | Metti in A indirizzo second.          |
| fldb: | 29 0f    | and | #\$0f    |           | Esegui un mask out nibble super.      |
| fldd: | c9 0f    | cmp | #\$0f    |           | Era aperto il canale com.?            |
| fldf: | f0 03    | beq | * \$fle4 |           | Se pos.cancella ingr.file da tavola   |
| fle1: | 20 9e f5 | jsr | * \$f59e |           | Invia comando CLOSE a periferica      |
| fle4: | 68       | pla |          |           | Esegui spostam. da tavola             |
| fle5: | aa       | tax |          |           | Copia q.s. da A a X                   |
| fle6: | c6 98    | dec | \$98     | [ ldtnd ] | Decr.n. di file aperti                |
| fle8: | e4 98    | cpx | \$98     | [ ldtnd ] | Contr.la tavola di ingresso per ved.  |
| flea: | f0 14    | beq | * \$f200 |           | se viene trovato ultimo ingr.         |
| flec: | a4 98    | ldy | \$98     | [ ldtnd ] | Carica n. di files aperti per spost.  |
| flee: | b9 62 03 | lda | \$0362,y |           | Carica in A ultimo ingresso da tavola |
| flf1: | 9d 62 03 | sta | \$0362,x |           | LFN e copia su pos. libera            |
| flf4: | b9 6c 03 | lda | \$036c,y |           | Metti ult. ingresso in tavola DA      |
| flf7: | 9d 6c 03 | sta | \$036c,x |           | Copia su posizioni libera             |
| flfa: | b9 76 03 | lda | \$0376,y |           | Metti ult. ingresso in tavola SA      |
| flfd: | 9d 76 03 | sta | \$0376,x |           | Copia su posizioni libere             |
| f200: | 18       | clc |          |           | Esegui un clear di carry per OK       |
| f201: | 60       | rts |          |           |                                       |

|       |          |     |          |            |                                    |
|-------|----------|-----|----------|------------|------------------------------------|
| f202: | a9 00    | lda | #\$00    |            | Clear il byte di Status e fissa l' |
| f204: | 85 90    | sta | \$90     | [ status ] | indicatore per tutto OK            |
| f206: | 8a       | txa |          |            | Sposta in A target per LFN         |
| f207: | a6 98    | ldx | \$98     | [ ldtnd ]  | Metti in X n. di files aperti      |
| f209: | ca       | dex |          |            | Decrementa di 1                    |
| f20a: | 30 05    | bmi | * \$f211 |            | Contr. se confronti negativi       |
| f20c: | dd 62 03 | cmp | \$0362,x |            | Confronta con Byte da tavola LFN   |
| f20f: | d0 f8    | bne | * \$f209 |            | Se diverso esegui pross. confronto |
| f211: | 60       | rts |          |            |                                    |

|       |          |     |          |        |                                      |
|-------|----------|-----|----------|--------|--------------------------------------|
| f212: | bd 62 03 | lda | \$0362,x |        | Metti in A n.di file logico dichiar. |
| f215: | 85 b8    | sta | \$b8     | [ la ] | da X in Pag. 0 per LFN               |
| f217: | bd 76 03 | lda | \$0376,x |        | Idem per indirizzo secondario        |
| f21a: | 85 b9    | sta | \$b9     | [ sa ] |                                      |
| f21c: | bd 6c 03 | lda | \$036c,x |        | Idem per indirizzo periferica        |
| f21f: | 85 ba    | sta | \$ba     | [ fa ] |                                      |
| f221: | 60       | rts |          |        |                                      |

## ROUTINE KERNAL CLALL

|       |       |     |       |           |                                  |
|-------|-------|-----|-------|-----------|----------------------------------|
| f222: | a9 00 | lda | #\$00 |           | Carica A con 0 ed immagazzina in |
| f224: | 85 98 | sta | \$98  | [ ldtnd ] | pagina 0 per n.di files aperti   |

## ROUTINE KERNAL CLRCH

|       |          |              |           |                                      |
|-------|----------|--------------|-----------|--------------------------------------|
| f226: | a2 03    | ldx #\$03    |           | Carica codice per perif. schermo     |
| f228: | e4 9a    | cpx \$9a     | [ dflto ] | Confr. con attuale perif in uscita   |
| f22a: | b0 03    | bcs * \$f22f |           | nella routine CLRCH sul bus seriale  |
| f22c: | 20 26 e5 | jsr * \$e526 |           | Vai a routine UNLSN:comandi su bus   |
| f22f: | e4 99    | cpx \$99     | [ dfltn ] | Confronta con attuale perif input    |
| f231: | b0 03    | bcs * \$f236 |           | nella routine CLRCH su bus seriale   |
| f233: | 20 15 e5 | jsr * \$e515 |           | Vai a rout. UNTLK:comando su bus ser |
| f236: | 86 9a    | stx \$9a     | [ dflto ] | Metti schermo come periferica uscita |
| f238: | a9 00    | lda #\$00    |           | Metti la tastiera come periferica    |
| f23a: | 85 99    | sta \$99     | [ dfltn ] | standard di input                    |
| f23c: | 60       | rts          |           |                                      |

|       |          |              |           |                                       |
|-------|----------|--------------|-----------|---------------------------------------|
| f23d: | 85 ba    | sta \$ba     | [ fa ]    | Immag. in pag. 0 per att.ind.perifer. |
| f23f: | c5 9a    | cmp \$9a     | [ dflto ] | Confronta con att. perif in uscita    |
| f241: | d0 05    | bne * \$f248 |           | Se diverso confr.con perif.in ingres. |
| f243: | a9 03    | lda #\$03    |           | Carica A con ind. perif.per schermo e |
| f245: | 85 9a    | sta \$9a     | [ dflto ] | mettilo come perif.in uscita          |
| f247: | 2c c5 99 | bit \$99c5   |           | ***                                   |
| f24a: | d0 04    | bne * \$f250 |           | Se diverso ricerca in tavola DA       |
| f24c: | a9 00    | lda #\$00    |           | Carica A con codice per tastiera      |
| f24e: | 85 99    | sta \$99     | [ dfltn ] | e mettila come perif.di input         |
| f250: | a5 ba    | lda \$ba     | [ fa ]    | Metti in A indirizzo periferica       |
| f252: | a6 98    | ldx \$98     | [ ldtn ]  | Metti in X n. di files aperti         |
| f254: | ca       | dex          |           | Decrementa di 1                       |
| f255: | 30 0d    | bmi * \$f264 |           | Se tutti i confr. neg, esci           |
| f257: | dd 6c 03 | cmp \$036c,x |           | Confr.con tavola per ind. perif.      |
| f25a: | d0 f8    | bne * \$f254 |           | Se non trovato confronta ancora       |
| f25c: | bd 62 03 | lda \$0362,x |           | Carica LFN per DA corrispond          |
| f25f: | 20 c3 ff | jsr * \$ffc3 | [ close ] | Vai a CLOSE                           |
| f262: | 90 ec    | bcc * \$f250 |           | Controlla per carry clear.            |
| f264: | 60       | rts          |           |                                       |

## ROUTINE KERNAL LOAD

|       |          |              |               |                                       |
|-------|----------|--------------|---------------|---------------------------------------|
| f265: | 86 c3    | stx \$c3     | [ memuss ]    | Immetti ind.iniz.di part.LO in pag.0  |
| f267: | 84 c4    | sty \$c4     | [ memuss+ 1 ] | Come sopra ma HI                      |
| f269: | 6c 30 03 | jmp (\$0330) |               | Vettore punta a LOADSP                |
| f26c: | 85 93    | sta \$93     | [ verck ]     | Immag. in pag. 0 flag, LOAD/VERIFY    |
| f26e: | a9 00    | lda #\$00    |               | Carica A con 0                        |
| f270: | 85 90    | sta \$90     | [ status ]    | Fissa lo status per tutto OK          |
| f272: | a5 ba    | lda \$ba     | [ fa ]        | Metti in A ind. perif.                |
| f274: | c9 04    | cmp #\$04    |               | Controlla per ind.perif valido        |
| f276: | b0 03    | bcs * \$f27b |               | Se indir.perif >4 allora OK           |
| f278: | 4c 26 f3 | jmp * \$f326 |               | Controlla per cassetta                |
| f27b: | ad 1c 0a | lda \$0alc   |               | Leggi punt. di sistema per modo seria |
| f27e: | 29 be    | and #\$be    |               | FAST ed elimina bit 6 che 1=FAST      |
| f280: | 8d 1c 0a | sta \$0alc   |               | 0=slow                                |
| f283: | a6 b9    | ldx \$b9     | [ sa ]        | Metti ind.second. in reg.X            |
| f285: | 86 9e    | stx \$9e     | [ ptr1 ]      | e immagag. q.s. in pag.0              |
| f287: | a4 b7    | ldy \$b7     | [ fnlen ]     | Metti lungh. nome file                |
| f289: | d0 03    | bne * \$f28e |               | Se non e' 0, vai a mess. errore       |
| f28b: | 4c 1a f3 | jmp * \$f31a |               | Errore di I/O n.8(Missing filename)   |
| f28e: | 84 9f    | sty \$9f     | [ ptr2 ]      | Immag. lungh.nome file                |
| f290: | 20 0f f5 | jsr * \$f50f |               | Messaggio "SEARCHING FOR"             |
| f293: | 20 a1 f3 | jsr * \$f3a1 |               | Contr. nome file e modo seriale FAST  |
| f296: | b0 03    | bcs * \$f29b |               | Contr. carry.Se fissato OK            |
| f298: | 4c 9b f3 | jmp * \$f39b |               | Fissa indir.fine caricam. e RTS       |
| f29b: | a4 9f    | ldy \$9f     | [ ptr2 ]      | Metti lunghezza nome file in Y e in   |

|                |              |               |                                       |
|----------------|--------------|---------------|---------------------------------------|
| f29d: 84 b7    | sty \$b7     | [ fnlen ]     | pagina 0 per lungh.nome file          |
| f29f: a9 60    | lda #\$60    |               | Metti Sa = 0                          |
| f2a1: 85 b9    | sta \$b9     | [ sa ]        | Immag. in pag. 0 per ind.sec.         |
| f2a3: 20 cb f0 | jsr * \$f0cb |               | Invia comando TALK a bus seriale      |
| f2a6: a5 ba    | lda \$ba     | [ fa ]        | Metti in A indirizzo periferica       |
| f2a8: 20 3b e3 | jsr * \$e33b |               | Vai a rout.TALK                       |
| f2ab: a5 b9    | lda \$b9     | [ sa ]        | Metti in A indirizzo secondario       |
| f2ad: 20 e0 e4 | jsr * \$e4e0 |               | Routine TKSA                          |
| f2b0: 20 3e e4 | jsr * \$e43e |               | Preleva un byte dal bus seriale       |
| f2b3: 85 ae    | sta \$ae     | [ eal ]       | Immetti ind.iniz.in pag.0             |
| f2b5: 20 3e e4 | jsr * \$e43e |               | Preleva un byte dal bus seriale       |
| f2b8: 85 af    | sta \$af     | [ eah ]       | Immag. ind.iniz.in Pag. 0             |
| f2ba: a5 90    | lda \$90     | [ status ]    | Carica STATUS in A                    |
| f2bc: 4a       | lsr a        |               | Spostam. a destra del bit di TIMEOUT  |
| f2bd: 4a       | lsr a        |               | Come sopra ma sul Carry               |
| f2be: b0 57    | bcs * \$f317 |               | Timeout per lettura                   |
| f2c0: a5 9e    | lda \$9e     | [ ptrl ]      | Metti in A indir.secondario           |
| f2c2: d0 08    | bne * \$f2cc |               | Se diverso da 0 salta a ind.          |
| f2c4: a5 c3    | lda \$c3     | [ memuss ]    | Copia indirizzo iniziale dato da X    |
| f2c6: 85 ae    | sta \$ae     | [ eal ]       | Y per il comando LOAD da \$C3 e \$C4  |
| f2c8: a5 c4    | lda \$c4     | [ memuss+ 1 ] | a \$AE,\$AF                           |
| f2ca: 85 af    | sta \$af     | [ eah ]       |                                       |
| f2cc: 20 33 f5 | jsr * \$f533 |               | Visualizza messaggio di controllo     |
| f2cf: a9 fd    | lda #\$fd    |               | Carica bit di timeout                 |
| f2d1: 25 90    | and \$90     | [ status ]    | da Status e riscrivi di nuovo         |
| f2d3: 85 90    | sta \$90     | [ status ]    | si STATUS                             |
| f2d5: 20 el ff | jsr * \$ffef | [ stop ]      | Vai a rout. kernal STOP               |
| f2d8: f0 49    | beq * \$f323 |               | Controllo tasto di STOP               |
| f2da: 20 3e e4 | jsr * \$e43e |               | Vai a ACPTR                           |
| f2dd: aa       | tax          |               | Trasf. cont. in X                     |
| f2de: a5 90    | lda \$90     | [ status ]    | Carica in A STATUS                    |
| f2e0: 4a       | lsr a        |               | Elimina il bit di lettura TIMEOUT     |
| f2e1: 4a       | lsr a        |               | dal byte di status                    |
| f2e2: b0 eb    | bcs * \$f2cf |               | Contr.timeout, se e' in t.out leggi   |
| f2e4: 8a       | txa          |               | Ripristina cont.precede.in A          |
| f2e5: a4 93    | ldy \$93     | [ verck ]     | Contr.punt.load/verify                |
| f2e7: f0 12    | beq * \$f2fb |               | Contr. in pag.0                       |
| f2e9: 85 bd    | sta \$bd     | [ ropriy ]    | Immag.inpag.0 buffer di parita'       |
| f2eb: a0 00    | ldy #\$00    |               | Sposta punt.per FETCH                 |
| f2ed: 20 c9 f7 | jsr * \$f7c9 |               | Vai a FETCH per operazioni LVS        |
| f2f0: c5 bd    | cmp \$bd     | [ ropriy ]    | Confronta con buffer di parita' Pag.0 |
| f2f2: f0 0a    | beq * \$f2fe |               | Se = allora OK e salta                |
| f2f4: a9 10    | lda \$10     |               | Carica A se diverso                   |
| f2f6: 20 57 f7 | jsr * \$f757 |               | Vai a rout. Kernal STATUS             |
| f2f9: d0 03    | bne * \$f2fe |               | Se STATUS non e' OK salta             |
| f2fb: 20 bf f7 | jsr * \$f7bf |               | Vai a rout. INDSTA via Pag. 0         |
| f2fe: e6 ae    | inc \$ae     | [ eal ]       | Incr. punt. byte basso di 1           |
| f300: d0 08    | bne * \$f30a |               | Contr.per overflow. se neg.salta      |
| f302: e6 af    | inc \$af     | [ eah ]       | Incr. punt. byte alto di 1            |
| f304: a5 af    | lda \$af     | [ eah ]       | Contr. se byte alto punta a interva.  |
| f306: c9 ff    | cmp #\$ff    |               | \$FF00. Se contr.positivo salta       |
| f308: f0 16    | beq * \$f320 |               | a uscita errore                       |
| f30a: 24 90    | bit \$90     | [ status ]    | Controllo di STATUS per bit EOF       |
| f30c: 50 c1    | bvc * \$f2cf |               | Se EOF non settato continua           |
| f30e: 20 15 e5 | jsr * \$e515 |               | Vai a rout. UNTLK                     |
| f311: 20 9e f5 | jsr * \$f59e |               | Invia segnale di Unlist su bus seria. |
| f314: 4c 9b f3 | jmp * \$f39b |               | Esegui un clear di carry              |
| f317: 4c 85 f6 | jmp * \$f685 |               | Errore di I/O n.4 (file not found)    |
| f31a: 4c 91 f6 | jmp * \$f691 |               | Errore di I/O n.8 (missing filename)  |
| f31d: 4c 94 f6 | jmp * \$f694 |               | Errore di I/O n.9 (illegal device n)  |

|                |       |          |              |  |
|----------------|-------|----------|--------------|--|
| f320: 4c 97 f6 | jmp * | \$f697   |              | Errore di I/O n. 10                    |
| f323: 4c b5 f5 | jmp * | \$f5b5   |              | Interruzione di routine LOAD           |
| f326: c9 01    | cmp   | #\$01    |              | E' un caricamento da cassetta?         |
| f328: d0 f3    | bne * | \$f31d   |              | Se no e' un errore n.9                 |
| f32a: 20 80 e9 | jsr * | \$e980   |              | Contr.indir. buffer nastro             |
| f32d: 90 ee    | bcc * | \$f31d   |              | Indirizzo buffer nastro illegale       |
| f32f: 20 c8 e9 | jsr * | \$e9c8   |              | Attendi per tasto premuto su cass.     |
| f332: b0 6c    | bcs * | \$f3a0   |              | Tasto di STOP premuto                  |
| f334: 20 0f f5 | jsr * | \$f50f   |              | Uscita nome file                       |
| f337: a5 b7    | lda   | \$b7     | [ fnlen ]    | Immag. lunghezza nome file in Pag.0    |
| f339: f0 09    | beq * | \$f344   |              | Lunghezza =0,vai a ricerca nome        |
| f33b: 20 9a e9 | jsr * | \$e99a   |              | Ricerca per testata nastro             |
| f33e: 90 0b    | bcc * | \$f34b   |              | Se ok continua                         |
| f340: f0 5e    | beq * | \$f3a0   |              | Contr. tasto Stop premuto              |
| f342: b0 d3    | bcs * | \$f317   |              | Errore di I/O n.4(file not found)      |
| f344: 20 d0 e8 | jsr * | \$e8d0   |              | Leggi HEADER da nastro                 |
| f347: f0 57    | beq * | \$f3a0   |              | Controllo tasto di STOP                |
| f349: b0 cc    | bcs * | \$f317   |              | Errore di I/O n.4 (file not found)     |
| f34b: 38       | sec   |          |              | Fissa per ricerca errore               |
| f34c: a5 90    | lda   | \$90     | [ status ]   | Carica in A STATUS                     |
| f34e: 29 10    | and   | #\$10    |              | Cancella bit 4                         |
| f350: d0 4e    | bne * | \$f3a0   |              | Metti a 1 bit 4                        |
| f352: e0 01    | cpx   | #\$01    |              | Contr. codice header                   |
| f354: f0 11    | beq * | \$f367   |              | Se q.s. =1 e' un progr. basic          |
| f356: e0 03    | cpx   | #\$03    |              | Controlla se e' un codice 3 (progr.LM) |
| f358: d0 dd    | bne * | \$f337   |              | Se non e' ne'1 ne' 3 continua ric.     |
| f35a: a0 01    | ldy   | #\$01    |              | Carica buffer cassetta                 |
| f35c: b1 b2    | lda   | (\$b2),y | [ tapel ]    | Prendi ind.inizio LO da Buffer         |
| f35e: 85 c3    | sta   | \$c3     | [ memuss ]   | Copialo su punt.LO indirizzo caricam.  |
| f360: c8       | iny   |          |              | Buffer di cassetta +1                  |
| f361: b1 b2    | lda   | (\$b2),y | [ tapel ]    | Prendi ind.inizio HI da Buffer         |
| f363: 85 c4    | sta   | \$c4     | [ memuss+ 1] | Copialo su punt.HI indirizzo caricam.  |
| f365: b0 04    | bcs * | \$f36b   |              | Salto incondizionato per prog. LM      |
| f367: a5 b9    | lda   | \$b9     | [ sa ]       | Carica ind. sec. in A                  |
| f369: d0 ef    | bne * | \$f35a   |              | Se e' = 0 append                       |
| f36b: a0 03    | ldy   | #\$03    |              | Sposta. su buffer cassetta             |
| f36d: b1 b2    | lda   | (\$b2),y | [ tapel ]    | Prendi ind.fine LO da Buffer           |
| f36f: a0 01    | ldy   | #\$01    |              | Spost. su Buffer di cassetta           |
| f371: f1 b2    | sbc   | (\$b2),y | [ tapel ]    | Sottrai ind.iniz.LO da fine            |
| f373: aa       | tax   |          |              | Trasf. in X                            |
| f374: a0 04    | ldy   | #\$04    |              | Spostam. su buffer cassetta            |
| f376: b1 b2    | lda   | (\$b2),y | [ tapel ]    | Prendi fine ind. HI da Buffer          |
| f378: a0 02    | ldy   | #\$02    |              | Buffer cassetta spostam.               |
| f37a: f1 b2    | sbc   | (\$b2),y | [ tapel ]    | Sottrai ind.iniz.HI da fine            |
| f37c: a8       | tay   |          |              | Trasf.valore HI in Y                   |
| f37d: 18       | clc   |          |              | Esegui un clear di carry per somma     |
| f37e: 8a       | txa   |          |              | Lungh. progr. in A                     |
| f37f: 65 c3    | adc   | \$c3     | [ memuss ]   | Somma ind.iniz.mem a lunghezza progr.  |
| f381: 85 ae    | sta   | \$ae     | [ eal ]      | Immag.per fine ind. basso              |
| f383: 98       | tya   |          |              | Trasf.lungh.progr. in A                |
| f384: 65 c4    | adc   | \$c4     | [ memuss+ 1] | Somma indir.part. memoria a lungh.pro  |
| f386: 85 af    | sta   | \$af     | [ eah ]      | Immetti in punt. per fine ind.HI       |
| f388: c9 ff    | cmp   | #\$ff    |              | Contr.se ind. di fine in \$FF00        |
| f38a: f0 94    | beq * | \$f320   |              | Se si errore I/O n.0                   |
| f38c: a5 c3    | lda   | \$c3     | [ memuss ]   | Copia indir.part.memoria LO in         |
| f38e: 85 c1    | sta   | \$c1     | [ track ]    | Pagina 0 e carica punt. LO             |
| f390: a5 c4    | lda   | \$c4     | [ memuss+ 1] | Come sopra per puntatore HI            |
| f392: 85 c2    | sta   | \$c2     | [ stah ]     |  |
| f394: 20 33 f5 | jsr * | \$f533   |              | Visualizza LOADING/VERIFYING           |
| f397: 20 fb e9 | jsr * | \$e9fb   |              | Carica programma da nastro             |

|                |                |           |  |  |
|----------------|----------------|-----------|--|--|
| f39a: 24       | .byte \$24     |           |  | Vai a \$F39C                           |
| f39b: 18       | clc            |           |  | Esegui un clear di Carry per indic.    |
| f39c: a6 ae    | ldx \$ae       | [ eal ]   |  | Metti in X ind. di fine LO             |
| f39e: a4 af    | ldy \$af       | [ eah ]   |  | Metti in Y ind. di fine HI             |
| f3a0: 60       | rts            |           |  |  |
| -----          |                |           |  |  |
| f3a1: a0 00    | ldy #\$00      |           |  | Fissa per routine FETCH                |
| f3a3: 20 ae f7 | jsr * \$f7ae   |           |  | Vai a indirizzo indicato               |
| f3a6: c9 24    | cmp #\$24      |           |  | Contr.se primo car.e' \$               |
| f3a8: f0 f6    | beq * \$f3a0   |           |  | Se positivo return                     |
| f3aa: a6 ba    | ldx \$ba       | [ fa ]    |  | Carica indir. perif in X               |
| f3ac: a0 0f    | ldy #\$0f      |           |  | Metti ind.secondario a 15              |
| f3ae: a9 00    | lda #\$00      |           |  | Metti file logico a 0                  |
| f3b0: 20 38 f7 | jsr * \$f738   |           |  | Vai a rout.SETLFS per fiss.par.file    |
| f3b3: 85 b7    | sta \$b7       | [ fnlen ] |  | Metti =0 lunghezza nome file           |
| f3b5: 20 c0 ff | jsr * \$ffc0   | [ open ]  |  | Vai a OPEN                             |
| f3b8: a6 b8    | ldx \$b8       | [ la ]    |  | Metti in X n. di file logico           |
| f3ba: 20 c9 ff | jsr * \$ffc9   | [ ckout ] |  | Vai a rout. CKOUT per fiss.canale usc  |
| f3bd: 90 08    | bcc * \$f3c7   |           |  | Se nessun errore continua              |
| f3bf: 20 8c f4 | jsr * \$f48c   |           |  | Richiudi file logico                   |
| f3c2: 68       | pla            |           |  | Sposta ind. RTS da Stack               |
| f3c3: 68       | pla            |           |  | Sposta ind. RTS da Stack               |
| f3c4: 4c 88 f6 | jmp * \$f688   |           |  | Errore di I/O n.5 (device not present) |
| f3c7: a0 03    | ldy #\$03      |           |  | Contatore ciclo                        |
| f3c9: b9 0b f5 | lda * \$f50b,y |           |  | Invia stringhe com. a disco            |
| f3cc: 20 d2 ff | jsr * \$ffd2   | [ bsout ] |  | Vai a rout.BSOUT                       |
| f3cf: 88       | dey            |           |  | Decr. ciclo di 1                       |
| f3d0: d0 f7    | bne * \$f3c9   |           |  | Ciclo a U0:invia un CHR\$(#) a disco   |
| f3d2: 20 ae f7 | jsr * \$f7ae   |           |  | Carattere per nome file                |
| f3d5: 20 d2 ff | jsr * \$ffd2   | [ bsout ] |  | Vai a routine BSOUT                    |
| f3d8: c8       | iny            |           |  | Incr.cont. nome file                   |
| f3d9: c4 9f    | cpy \$9f       | [ ptr2 ]  |  | Confronta con lunghezza nome file      |
| f3db: d0 f5    | bne * \$f3d2   |           |  | Se non trovato succ. carattere         |
| f3dd: 20 cc ff | jsr * \$ffcc   | [ clrch ] |  | Vai a CLRCH                            |
| f3e0: 2c 1c 0a | bit \$0alc     |           |  | Contr. punt. modo seriale FAST         |
| f3e3: 70 05    | bvs * \$f3ea   |           |  | Contr.possibilita' trasferim.          |
| f3e5: 20 8c f4 | jsr * \$f48c   |           |  | Richiudi file logico                   |
| f3e8: 38       | sec            |           |  | Fissa indicatore per OK                |
| f3e9: 60       | rts            |           |  |  |
| -----          |                |           |  |  |
| f3ea: a5 9f    | lda \$9f       | [ ptr2 ]  |  | Immag.temp.per lunghezza nome file     |
| f3ec: 85 b7    | sta \$b7       | [ fnlen ] |  | Puntat. in Pag.0 per lungh.nome file   |
| f3ee: 78       | sei            |           |  | Disabilita gli interrupt               |
| f3ef: 20 45 e5 | jsr * \$e545   |           |  | Invia segn.HI clock su bus seriale     |
| f3f2: 20 c3 e5 | jsr * \$e5c3   |           |  | Attendi risposta da bus seriale        |
| f3f5: 2c 0d dc | bit * \$dc0d   |           |  | Es.clear su flag IRQ di CIA            |
| f3f8: 20 03 f5 | jsr * \$f503   |           |  | Inverti segnale clock LO/HI            |
| f3fb: 20 ba f4 | jsr * \$f4ba   |           |  | Preleva byteda bus                     |
| f3fe: c9 02    | cmp #\$02      |           |  | Contr. visualizz. "File not Found"     |
| f400: d0 08    | bne * \$f40a   |           |  | Se neg. vaiad indirizzo                |
| f402: 20 8c f4 | jsr * \$f48c   |           |  | Segn. clock su bus seriale e chius.fi  |
| f405: 68       | pla            |           |  | Togli dallo Stack e trasf in A i 2     |
| f406: 68       | pla            |           |  | bytes di indirizzo RTS                 |
| f407: 4c 85 f6 | jmp * \$f685   |           |  | Errore di I/O n.4 (File not found)     |
| f40a: 48       | pha            |           |  | Trasf. Status su stack                 |
| f40b: c9 1f    | cmp \$1f       |           |  | Contr. se e' indicat.per ult.blocco    |
| f40d: d0 0b    | bne * \$f41a   |           |  | Se neg. salta                          |
| f40f: 20 03 f5 | jsr * \$f503   |           |  | Inverti segnale di clock LO/HI         |
| f412: 20 ba f4 | jsr * \$f4ba   |           |  | Preleva byte da bus seriale            |
| f415: 85 a5    | sta \$a5       | [ cntdn ] |  | Immag.byte per cont. numero cicli      |

|       |    |    |    |     |      |        |               |  |
|-------|----|----|----|-----|------|--------|---------------|--|
| f417: | 4c | 21 | f4 | jmp | *    | \$f421 |               | Fissa ind. di load                     |
| f41a: | c9 | 02 |    | cmp | #    | \$02   |               | Controllo Status per trasferim         |
| f41c: | 90 | 03 |    | bcc | *    | \$f421 |               | Se contr. \$01 OK                      |
| f41e: | 68 |    |    | pla |      |        |               | Cancella Status immaga.                |
| f41f: | b0 | 77 |    | bcs | *    | \$f498 |               | Vai a "LOAD ERROR" ed esci             |
| f421: | 20 | 33 | f5 | jsr | *    | \$f533 |               | Visualizza LOADING/VERIFYING           |
| f424: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segnale di clock LO/HI         |
| f427: | 20 | ba | f4 | jsr | *    | \$f4ba |               | Indirizzo di caricamento LO            |
| f42a: | 85 | ae |    | sta | \$ae |        | [ eal ]       | Salva su A punt.ind.basso              |
| f42c: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segnale di clock LO/HI         |
| f42f: | 20 | ba | f4 | jsr | *    | \$f4ba |               | Prendi byte da bus,carica ind. HI      |
| f432: | 85 | af |    | sta | \$af |        | [ eah ]       | Immag. in A punt. indirizzo            |
| f434: | a6 | 9e |    | ldx | \$9e |        | [ ptrl ]      | Carica e contr. ind. sec.immagaz.      |
| f436: | d0 | 08 |    | bne | *    | \$f440 |               | Se <> 0,allora carica progr.           |
| f438: | a5 | c3 |    | lda | \$c3 |        | [ memuss ]    | Indirizzo di LOAD LO in A              |
| f43a: | a6 | c4 |    | ldx | \$c4 |        | [ memuss+ 1 ] | Indirizzo di LOAD HI in X              |
| f43c: | 85 | ae |    | sta | \$ae |        | [ eal ]       | Immag. ind. LC in punt. LO             |
| f43e: | 86 | af |    | stx | \$af |        | [ eah ]       | C.S. per ind. HI                       |
| f440: | a5 | ae |    | lda | \$ae |        | [ eal ]       | Metti in A punt. ind. LO               |
| f442: | a6 | af |    | ldx | \$af |        | [ eah ]       | Metti in X punt. HI                    |
| f444: | 85 | ac |    | sta | \$ac |        | [ sal ]       | Fissa A come punt.indirizzo caricam.   |
| f446: | 86 | ad |    | stx | \$ad |        | [ sah ]       | C.s. con reg. X                        |
| f448: | 68 |    |    | pla |      |        |               | Preleva da Stack Status per trasf.     |
| f449: | c9 | 1f |    | cmp | #    | \$1f   |               | Contr. se status punta a ult. blocco   |
| f44b: | f0 | 32 |    | beq | *    | \$f47f |               | Se pos.vai a blocco lungh.standard     |
| f44d: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segn. di Clock LO/HI           |
| f450: | a9 | fc |    | lda | #    | \$fc   |               | Metti cont.data per primo blocco del   |
| f452: | 85 | a5 |    | sta | \$a5 |        | [ cntdn ]     | file da leggere a 252                  |
| f454: | 20 | 3d | f6 | jsr | *    | \$f63d |               | Controlla per Shift RUN/STOP           |
| f457: | 20 | el | ff | jsr | *    | \$ffel | [ stop ]      | Vai a rout. STOP                       |
| f45a: | f0 | 4a |    | beq | *    | \$f4a6 |               | Controllo tasto STOP.Se=0 esci         |
| f45c: | 20 | c5 | f4 | jsr | *    | \$f4c5 |               | Leggi blocco da disco ed esegui        |
| f45f: | b0 | 51 |    | bcs | *    | \$f4b2 |               | Rilev.errore in ind.memoria quindi RTS |
| f461: | 20 | ba | f4 | jsr | *    | \$f4ba |               | Byte da bus seriale                    |
| f464: | c9 | 02 |    | cmp | #    | \$02   |               | Controlla stus di transf.              |
| f466: | 90 | 06 |    | bcc | *    | \$f46e |               | Se e' un codice \$01 OK                |
| f468: | c9 | 1f |    | cmp | #    | \$1f   |               | Contr.se questo e' per ultimo blocco   |
| f46a: | f0 | 0b |    | beq | *    | \$f477 |               | Se pos. leggi ultimo blocco            |
| f46c: | d0 | 2a |    | bne | *    | \$f498 |               | Vai a "LOAD ERROR" ed esci             |
| f46e: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segnale di clock LO/HI         |
| f471: | a9 | fe |    | lda | #    | \$fe   |               | Fissa il contatore data per normale    |
| f473: | 85 | a5 |    | sta | \$a5 |        | [ cntdn ]     | lettura di 254 bytes                   |
| f475: | d0 | dd |    | bne | *    | \$f454 |               | Salto incond.a rout. di lettura        |
| f477: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segnale di clock LO/HI         |
| f47a: | 20 | ba | f4 | jsr | *    | \$f4ba |               | Byte da bus seriale (n. blocco-byte)   |
| f47d: | 85 | a5 |    | sta | \$a5 |        | [ cntdn ]     | Metti in A byte n.cont. ciclo          |
| f47f: | 20 | 03 | f5 | jsr | *    | \$f503 |               | Inverti segnale di clock LO/HI         |
| f482: | 20 | c5 | f4 | jsr | *    | \$f4c5 |               | Lettura blocco da disco e esegui       |
| f485: | b0 | 2b |    | bcs | *    | \$f4b2 |               | Rilev. errore in ind. memoria,Vai RTS  |
| f487: | a9 | 40 |    | lda | #    | \$40   |               | metti in A codice per EOF              |
| f489: | 20 | 57 | f7 | jsr | *    | \$f757 |               | Vai a rout. kernal SETMSG              |
| f48c: | 20 | 45 | e5 | jsr | *    | \$e545 |               | Invia segn. di clock alto su bus ser.  |
| f48f: | 58 |    |    | cli |      |        |               | Abilita tutti gli interrupt            |
| f490: | a5 | b8 |    | lda | \$b8 |        | [ la ]        | Metti n.di file logico in A            |
| f492: | 38 |    |    | sec |      |        |               | Fissa il carry per rout. CLOSE         |
| f493: | 20 | c3 | ff | jsr | *    | \$ffc3 | [ close ]     | Vai a rout.Kernal CLOSE                |
| f496: | 18 |    |    | clc |      |        |               | Es.clear di carry per OK               |
| f497: | 60 |    |    | rts |      |        |               |  |

|                |                      |                                       |
|----------------|----------------------|---------------------------------------|
| f498: a9 02    | lda #\$02            | Errore di Timeout durante lettura     |
| f49a: 20 57 f7 | jsr * \$f757         | Vai a rout. kernal SETMSG             |
| f49d: 20 8c f4 | jsr * \$f48c         | Invia segn.clock HI su bus e close    |
| f4a0: 68       | pla                  | Cancella dallo Stack indirizzo di     |
| f4a1: 68       | pla                  | ritorno per RTS                       |
| f4a2: a9 29    | lda #\$29            | N. errore per errore Basic LOAD       |
| f4a4: 38       | sec                  | Fissa segnale per rilev.errore        |
| f4a5: 60       | rts                  |                                       |
| -----          |                      |                                       |
| f4a6: 20 8c f4 | jsr * \$f48c         | Segnale clock HI su bus e Close file  |
| f4a9: a9 00    | lda #\$00            | Poni a n.0 puntatore in Pag. 0        |
| f4ab: 85 b9    | sta \$b9 [ sa ]      | per att. indirizzo secondario         |
| f4ad: 68       | pla                  | Cancella dallo Stack indirizzo di     |
| f4ae: 68       | pla                  | ritorno per RTS                       |
| f4af: 4c b5 f5 | jmp * \$f5b5         | Vai a rout. di STOP                   |
| f4b2: 20 8c f4 | jsr * \$f48c         | Invia segn.clock HI su bus e Close    |
| f4b5: 68       | pla                  | Cancella dallo stack indirizzo di     |
| f4b6: 68       | pla                  | ritorno per RTS                       |
| f4b7: 4c 97 f6 | jmp * \$f697         | Salta ad uscita errore I/O n.10       |
| f4ba: a9 08    | lda #\$08            | Fissa bit di controllo per interrupt  |
| f4bc: 2c 0d dc | bit * \$dc0d         | Leggi reg. controllo interrupt        |
| f4bf: f0 fb    | beq * \$f4bc         | Attesa per interrupt su bus seriale   |
| f4c1: ad 0c dc | lda * \$dc0c         | leggi da bus dati buffer di CIA       |
| f4c4: 60       | rts                  |                                       |
| -----          |                      |                                       |
| f4c5: a9 08    | lda #\$08            | Fissa bit contr.per bus interrupt     |
| f4c7: 2c 0d dc | bit * \$dc0d         | Leggi reg.controllo interrupt         |
| f4ca: f0 fb    | beq * \$f4c7         | Attesa per interrupt su bus seriale   |
| f4cc: ac 0c dc | ldy * \$dc0c         | Leggi buffer dati CIA da bus seriale  |
| f4cf: ad 00 dd | lda * \$dd00         | Leggi porta dati A di CIA 2           |
| f4d2: 49 10    | eor #\$10            | Di conseguenza inverti il segnale di  |
| f4d4: 8d 00 dd | sta * \$dd00         | clock e scrivi su porta A             |
| f4d7: 98       | tya                  | Trasf. buffer dati in A               |
| f4d8: a4 93    | ldy \$93 [ verck ]   | Controlla punt. per LOAD/VERIFY in Pa |
| f4da: f0 12    | beq * \$f4ee         | 0 per val. \$00.Se pos. e' rout.LOAD  |
| f4dc: 85 bd    | sta \$bd [ roperty ] | Immag. byte di data per operaz.VERIFY |
| f4de: a0 00    | ldy \$00             | Posiz. punt. per rout. FETCH          |
| f4e0: 20 c9 f7 | jsr * \$f7c9         | Vai a FETCH                           |
| f4e3: c5 bd    | cmp \$bd [ roperty ] | Confronta byte dati con memoria       |
| f4e5: f0 0a    | beq * \$f4f1         | Se = allora tutto OK e continua       |
| f4e7: a9 10    | lda #\$10            | Se <> fissa segnale di errore         |
| f4e9: 20 57 f7 | jsr * \$f757         | Vai a rout. STATUS                    |
| f4ec: d0 03    | bne * \$f4f1         | Salta rout. STASH per LOAD            |
| f4ee: 20 bf f7 | jsr * \$f7bf         | Vai a STASH per Load,Save,Verify      |
| f4f1: e6 ae    | inc \$ae [ eal ]     | Increment di 1 LO per operaz. di I/O  |
| f4f3: d0 08    | bne * \$f4fd         | Se nessun overflow vai a indir.       |
| f4f5: e6 af    | inc \$af [ eah ]     | Increment. valore HI di ind. I/O      |
| f4f7: a5 af    | lda \$af [ eah ]     | Controlla se il valore HI di I/O pun. |
| f4f9: c9 ff    | cmp \$fff            | alla tavola di vettori del sistema    |
| f4fb: f0 05    | beq * \$f502         | Se pos. vai a RTS                     |
| f4fd: c6 a5    | dec \$a5 [ cntdn ]   | Decr. di 1 contatore byte dati        |
| f4ff: d0 c4    | bne * \$f4c5         | Ciclo lettura tutti i bytes           |
| f501: 18       | clc                  | Clear di carry per tutto OK           |
| f502: 60       | rts                  |                                       |
| -----          |                      |                                       |
| f503: ad 00 dd | lda * \$dd00         | Leggi porta dati A del CIA 2          |
| f506: 49 10    | eor #\$10            | Inverti segnale di clock              |

f508: 8d 00 dd sta \* \$dd00  
f50b: 60 rts

Riscrivilo sulla porta A

Sequenza di controllo al disco.  
Invia un CHR\$(31)

f50c: 1f 30 55 Ou

f50f: a5 9d lda \$9d [ msgflg ]  
f511: 10 1f bpl \* \$f532  
f513: a0 0c ldy #\$0c  
f515: 20 22 f7 jsr \* \$f722  
f518: a5 b7 lda \$b7 [ fnlen ]  
f51a: f0 16 beq \* \$f532  
f51c: a0 17 ldy #\$17  
f51e: 20 22 f7 jsr \* \$f722  
f521: a4 b7 ldy \$b7 [ fnlen ]  
f523: f0 0d beq \* \$f532  
f525: a0 00 ldy #\$00  
f527: 20 ae f7 jsr \* \$f7ae  
f52a: 20 d2 ff jsr \* \$ffd2 [bsout ]  
f52d: c8 iny  
f52e: c4 b7 cpy \$b7 [ fnlen ]  
f530: d0 f5 bne \* \$f527  
f532: 60 rts

E' consentito un messaggio di contr.  
Se neg. return  
Posizionati per ricerca  
Uscita messaggio controllo sistema  
Metti in A lungh. nome file  
Contr. se lungh. = 0 poi return  
Posizionati per testo FOR  
Uscita messaggio controllo sistema  
Metti in Y lungh.att.nome file  
Se lunghezza = 0 salta  
Iniz. spost. sul nome del file  
Prendi un byte del nome file  
Vai a rout. BSOUT  
Incr. per inizio nome file  
Confronta con lungh. nome file  
Se <> vai al prossimo carattere

f533: a0 49 ldy #\$49  
f535: a5 93 lda \$93 [ verck ]  
f537: f0 02 beq \* \$f53b  
f539: a0 59 ldy #\$59  
f53b: 4c 1e f7 jmp \* \$f71e

Posizio. per testo LOADING  
Preleva segnal. Load-Verify  
Se e' un load (0), visualizza  
Posiz. su VERIFY  
Messaggio di uscita controllo sist.

#### ROUTINE KERNAL SAVESP

f53e: 86 ae stx \$ae [ eal ]  
f540: 84 af sty \$af [ eah ]  
f542: aa tax  
f543: b5 00 lda \$00,x [ d6510 ]  
f545: 85 c1 sta \$c1 [ track ]  
f547: b5 01 lda \$01,x [ r6510 ]  
f549: 85 c2 sta \$c2 [ stah ]  
f54b: 6c 32 03 jmp (\$0332)  
f54e: a5 ba lda \$ba [ fa ]  
f550: c9 01 cmp #\$01  
f552: f0 74 beq \* \$f5c8  
f554: c9 04 cmp #\$04  
f556: b0 09 bcs \* \$f561  
f558: 4c 94 f6 jmp \* \$f694  
f55b: 4c 91 f6 jmp \* \$f691  
f55e: 4c 85 f6 jmp \* \$f685  
f561: a4 b7 ldy \$b7 [ fnlen ]  
f563: f0 f6 beq \* \$f55b  
f565: a9 61 lda \$61  
f567: 85 b9 sta \$b9 [ sa ]  
f569: 20 cb f0 jsr \* \$f0cb  
f56c: 20 bc f5 jsr \* \$f5bc  
f56f: a5 ba lda \$ba [ fa ]  
f571: 20 3e e3 jsr \* \$e33e  
f574: a5 b9 lda \$b9 [ sa ]  
f576: 20 d2 e4 jsr \* \$e4d2

Immag. in X ind. LO di Immagazzinam.  
C.s. su Y ind. HI  
Trasf in X ind.in Pag. 0 di Imm.  
Metti in A indir. per valore LO  
immaq. q.s  
Come sopra per valore HI  
C.S.  
Vai a SAVESP  
Metti in A indir.perif.  
Controlla se perif inusc.e'cassetta  
Se pos. rout. di save su cassetta  
Contr. se ind.perif <4  
Se neg. vai a messaggio di errore  
Errore di I/O n.9(Illegal device)  
Errore di I/O n.8(Missing filename)  
Errore di I/O n.4(File not found)  
Metti lungh.nome file in Y  
Contr.se lungh.=0. errore di I/O 8  
Metti in A ind.secondario per operaz.  
di Print?Write  
Contr. lungh. e indir. secondario  
Se abilitati vis. SAVING  
Metti in A indir.periferica  
Vai a rout. LISTN  
Metti in A indir. secondario  
Vai a SECND



|       |          |     |          |          |  |
|-------|----------|-----|----------|----------|--|
| f579: | a0 00    | ldy | #\$00    |          | Metti =0 reg. Y per piazz.             |
| f57b: | 20 51 ed | jsr | * \$ed51 |          | Copia ind.iniz. da C1,C2 a AD,DC       |
| f57e: | 20 03 e5 | jsr | * \$e503 |          | Vai a rout. CIOUT                      |
| f581: | a5 ad    | lda | \$ad     | [ sah ]  | Immag. ind. di inizio val. HI          |
| f583: | 20 03 e5 | jsr | * \$e503 |          | Vai a CIOUT                            |
| f586: | 20 b7 ee | jsr | * \$eeb7 |          | Sottrazione:Ind.iniz.-ind.finale       |
| f589: | b0 10    | bcs | * \$f59b |          | Se rilev. indirizzo finale esci        |
| f58b: | 20 cc f7 | jsr | * \$f7cc |          | Metti in FETVEC ind.inizio             |
| f58e: | 20 03 e5 | jsr | * \$e503 |          | Vai a CIUOT                            |
| f591: | 20 e1 ff | jsr | * \$ffe1 | [ stop ] | Vai a rout. STOP                       |
| f594: | f0 1f    | beq | * \$f5b5 |          | Se tasto STOP premuto, inter. SAVESP   |
| f596: | 20 c1 ee | jsr | * \$eecl |          | Increment.ind.iniziale di 1            |
| f599: | d0 eb    | bne | * \$f586 |          | Contr. se overflow in byte HI          |
| f59b: | 20 26 e5 | jsr | * \$e526 |          | Vai a UNLSN                            |
| f59e: | 24 b9    | bit | \$b9     | [ sa ]   | Controlla Bit 7 di Ind. sec.           |
| f5a0: | 30 11    | bmi | * \$f5b3 |          | Se bit 7 fissato, salta                |
| f5a2: | a5 ba    | lda | \$ba     | [ fa ]   | Carica in A ind. periferica            |
| f5a4: | 20 3e e3 | jsr | * \$e33e |          | Vai a LISTN                            |
| f5a7: | a5 b9    | lda | \$b9     | [ sa ]   | Carica ind. sec. in A                  |
| f5a9: | 29 ef    | and | * \$ef   |          | Esegui AND su nibble LO di ind. second |
| f5ab: | 09 e0    | ora | * \$e0   |          | Invia un CLOSE a periferica            |
| f5ad: | 20 d2 e4 | jsr | * \$e4d2 |          | Vai a SECND                            |
| f5b0: | 20 26 e5 | jsr | * \$e526 |          | Vai a UNLSN                            |
| f5b3: | 18       | clc |          |          | Ese. un clear di carry per OK          |
| f5b4: | 60       | rts |          |          |  |

## ROUTINE KERNAL SAVESP

|       |          |     |          |  |                                     |
|-------|----------|-----|----------|--|-------------------------------------|
| f5b5: | 20 9e f5 | jsr | * \$f59e |  | Chiudi canale di scritt. su perif   |
| f5b8: | a9 00    | lda | #\$00    |  | Carica A con \$00 come segnalat.    |
| f5ba: | 38       | sec |          |  | Fissa il carry per indic.break/err. |
| f5bb: | 60       | rts |          |  |                                     |

|       |          |     |          |            |                                       |
|-------|----------|-----|----------|------------|---------------------------------------|
| f5bc: | a5 9d    | lda | \$9d     | [ msgflg ] | Contr. se invio mess. consentito      |
| f5be: | 10 37    | bpl | * \$f5f7 |            | Se neg. return                        |
| f5c0: | a0 51    | ldy | * \$51   |            | Piazz. per operaz. di SAVING in Y     |
| f5c2: | 20 22 f7 | jsr | * \$f722 |            | Vis. messaggio di SAVING              |
| f5c5: | 4c 21 f5 | jmp | * \$f521 |            | Vis. nome del file                    |
| f5c8: | 20 80 e9 | jsr | * \$e980 |            | Punt. buffer di cass. in X e Y        |
| f5cb: | 90 8b    | bcc | * \$f558 |            | Se pag.0,1 non abilit.errore di I/O 9 |
| f5cd: | 20 e9 e9 | jsr | * \$e9e9 |            | Attesa per tasto PLAY                 |
| f5d0: | b0 25    | bcs | * \$f5f7 |            | Contr. tasto STOP                     |
| f5d2: | 20 bc f5 | jsr | * \$f5bc |            | Se tutto OK, vis. "SAVING"            |
| f5d5: | a2 03    | ldx | * \$03   |            | Metti tipo HEADER 3, cioe' prog.LM    |
| f5d7: | a5 b9    | lda | \$b9     | [ sa ]     | Carica ind. sec. in A                 |
| f5d9: | 29 01    | and | * \$01   |            | Contr. se bit 0=1                     |
| f5db: | d0 02    | bne | * \$f5df |            | Se pos. progr. in LM                  |
| f5dd: | a2 01    | ldx | * \$01   |            | Metti tipo HEADER 1, progr.Basic      |
| f5df: | 8a       | txa |          |            | Copia tipo Header in A                |
| f5e0: | 20 19 e9 | jsr | * \$e919 |            | Scrivi HEADER su nastro               |
| f5e3: | b0 12    | bcs | * \$f5f7 |            | Contr. tasto STOP premuto             |
| f5e5: | 20 18 ea | jsr | * \$ea18 |            | Salva progr. su cassetta              |
| f5e8: | b0 0d    | bcs | * \$f5f7 |            | Esci se tasto STOP premuto            |
| f5ea: | a5 b9    | lda | \$b9     | [ sa ]     | Metti in A indirizzo secondario       |
| f5ec: | 29 02    | and | * \$02   |            | Contr. se bit 1 =1                    |
| f5ee: | f0 06    | beq | * \$f5f6 |            | Se neg. allora OK ed esci             |
| f5f0: | a9 05    | lda | * \$05   |            | Metti codice per EOT in A             |
| f5f2: | 20 19 e9 | jsr | * \$e919 |            | Scrivi blocco su nastro               |
| f5f5: | 24 18    | bit | \$18     | [ tempot ] | Metti indic. per tutto OK             |

f5f7: 60           rts

-----  
ROUTINE KERNAL UDTIM

Valori riferiti a clock interno

|                |              |             |  |
|----------------|--------------|-------------|--|
| f5f8: e6 a2    | inc \$a2     | [ time+ 2 ] | Incr.byte LO del clock di sistema      |
| f5fa: d0 06    | bne * \$f602 |             | Se nessun overflow esegui correz.      |
| f5fc: e6 a1    | inc \$a1     | [ time+ 1 ] | Incr.byte MI del clock di sistema      |
| f5fe: d0 02    | bne * \$f602 |             | Se nessun overflow esegui correz.      |
| f600: e6 a0    | inc \$a0     | [ time ]    | Incr.byte HI del clock di sistema      |
| f602: 38       | sec          |             | Fissa il carry per sottrazione         |
| f603: a5 a2    | lda \$a2     | [ time+ 2 ] | In questa routine vengono controll.    |
| f605: e9 01    | sbc #\$01    |             | i giusti valori attraverso sottraz.    |
| f607: a5 a1    | lda \$a1     | [ time+ 1 ] | per vedere se il clock di sistema      |
| f609: e9 1a    | sbc #\$1a    |             | interno e messo a 24.00.00 nei bytes   |
| f60b: a5 a0    | lda \$a0     | [ time ]    | \$A0-\$A1-\$A2                         |
| f60d: e9 4f    | sbc #\$4f    |             | In questo caso i tre bytes devono      |
| f60f: 90 08    | bcc * \$f619 |             | essere reinizializzati                 |
| f611: a2 00    | ldx #\$00    |             | Nelle seguenti istruzioni vengono      |
| f613: 86 a0    | stx \$a0     | [ time ]    | Immagazzinati i tre bytes di cui       |
| f615: 86 a1    | stx \$a1     | [ time+ 1 ] | sopra. HI, MI, LO                      |
| f617: 86 a2    | stx \$a2     | [ time+ 2 ] |  |
| f619: ad 1d 0a | lda \$0ald   |             | Contr. immag. temp. clock LO           |
| f61c: d0 0b    | bne * \$f629 |             | Se <>0 allora solo LO -1               |
| f61e: ad 1e 0a | lda \$0ale   |             | Contr. immag. temp. clock MI           |
| f621: d0 03    | bne * \$f626 |             | Se <>0 allora solo MI-1                |
| f623: ce 1f 0a | dec \$0alf   |             | Clock HI -1                            |
| f626: ce 1e 0a | dec \$0ale   |             | Clock MI -1                            |
| f629: ce 1d 0a | dec \$0ald   |             | Clock LO -1                            |
| f62c: 2c 03 0a | bit \$0a03   |             | Contr. punt. PAL/NTSC                  |
| f62f: 10 0c    | bpl * \$f63d |             | Se "PLUS" allora NTSC                  |
| f631: ce 36 0a | dec \$0a36   |             | Decr.punt. linea raster -1             |
| f634: 10 07    | bpl * \$f63d |             | Contr. se =0, se no vai a ind.         |
| f636: a9 05    | lda #\$05    |             | Il punt. di sistema per linea raster   |
| f638: 8d 36 0a | sta \$0a36   |             | e' inizializz. con val. 5              |
| f63b: d0 bb    | bne * \$f5f8 |             | Salto incondizionato a UDTIM           |
| f63d: ad 01 dc | lda * \$dc01 |             | Lettura della porta B per controllo    |
| f640: cd 01 dc | cmp * \$dc01 |             | matrice di tastiera                    |
| f643: d0 f8    | bne * \$f63d |             | Esegui ciclo di attesa                 |
| f645: aa       | tax          |             | Trasferisci codice tastiera in X e     |
| f646: 30 13    | bmi * \$f65b |             | salta a ind. se tasto di RUN/STOP atti |
| f648: a2 bd    | ldx #\$bd    |             | Metti il bit map per riga in porta A   |
| f64a: 8e 00 dc | stx * \$dc00 |             | per la selez.matrice di linea          |
| f64d: ae 01 dc | ldx * \$dc01 |             | C.s. per matrice colonna porta B       |
| f650: ec 01 dc | cpx * \$dc01 |             | Lettura ed attesa                      |
| f653: d0 f8    | bne * \$f64d |             |  |
| f655: 8d 00 dc | sta * \$dc00 |             | Immag. in A per linea matrice porta A  |
| f658: e8       | inx          |             | Incrementa valore di 1                 |
| f659: d0 02    | bne * \$f65d |             | Se nessun tasto SHIFT premuto, salta   |
| f65b: 85 91    | sta \$91     | [ stkey ]   | Immag. in A punt.RVS                   |
| f65d: 60       | rts          |             |  |

-----  
ROUTINE KERNAL RDTIM

|             |          |             |                                     |
|-------------|----------|-------------|-------------------------------------|
| f65e: 78    | sei      |             | Disabilita tutti gli interrupt      |
| f65f: a5 a2 | lda \$a2 | [ time+ 2 ] | Immag.nei reg. di pag. 0 i byte per |
| f661: a6 a1 | ldx \$a1 | [ time+ 1 ] | clock di sistema rispettivamente    |
| f663: a4 a0 | ldy \$a0 | [ time ]    | LO-MI-HI                            |

-----  
ROUTINE KERNAL SETTIM

|             |          |             |                                       |
|-------------|----------|-------------|---------------------------------------|
| f665: 78    | sei      |             | Disabilita tutti gli interrupt        |
| f666: 85 a2 | sta \$a2 | [ time+ 2 ] | Vedi la routine precedente. La diffe  |
| f668: 86 a1 | stx \$a1 | [ time+ 1 ] | fra le due rout.e' che la prima legge |
| f66a: 84 a0 | sty \$a0 | [ time ]    | mentre questa immagazzina i valori    |
| f66c: 58    | cli      |             | Abilita tutti gli interrupt           |
| f66d: 60    | rts      |             |                                       |

---

|                |              |           |                                       |
|----------------|--------------|-----------|---------------------------------------|
| f66e: a5 91    | lda \$91     | [ stkey ] | Predisponi immag.per floag di stop    |
| f670: c9 7f    | cmp #\$7f    |           | Controllo per tasto STOP premuto      |
| f672: d0 07    | bne * \$f67b |           | Se neg.,return con flag di uguale a 0 |
| f674: 08       | php          |           | Salva il flag                         |
| f675: 20 cc ff | jsr * \$ffcc | [ clrch ] | Vai a CLRCH                           |
| f678: 85 d0    | sta \$d0     | [ ndx ]   | Eseg.Clear su punt.buffer tastiera    |
| f67a: 28       | plp          |           | Metti in Stack flag di uguale         |
| f67b: 60       | rts          |           |                                       |

## TAVOLA DEI MESSAGGI DI ERRORE

|                |            |
|----------------|------------|
| f67c: a9 01    | lda #\$01  |
| f67e: 2c a9 02 | bit \$02a9 |
| f681: 2c a9 03 | bit \$03a9 |
| f684: 2c a9 04 | bit \$04a9 |
| f687: 2c a9 05 | bit \$05a9 |
| f68a: 2c a9 06 | bit \$06a9 |
| f68d: 2c a9 07 | bit \$07a9 |
| f690: 2c a9 08 | bit \$08a9 |
| f693: 2c a9 09 | bit \$09a9 |
| f696: 2c a9 10 | bit \$10a9 |

|                |              |            |  |
|----------------|--------------|------------|--|
| f699: 48       | pha          |            | Immagazzina codice errore su Stack     |
| f69a: 20 cc ff | jsr * \$ffcc | [ clrch ]  | Vai a CLRCH                            |
| f69d: a0 00    | ldy #\$00    |            | Posiz.per messag.di errore I/O         |
| f69f: 24 9d    | bit \$9d     | [ msgflg ] | Controlla se messaggi di sistema abili |
| f6a1: 50 0a    | bvc * \$f6ad |            | Se non cons. esci                      |
| f6a3: 20 22 f7 | jsr * \$f722 |            | Vai a rout.di mess.uscita              |
| f6a6: 68       | pla          |            | Sposta in A n.codice errore            |
| f6a7: 48       | pha          |            | e mettilo sullo Stack                  |
| f6a8: 09 30    | ora #\$30    |            | Riporta val.ASCII di codice errore     |
| f6aa: 20 d2 ff | jsr * \$ffd2 | [ bsout ]  | Vai a rout.BSOUT                       |
| f6ad: 68       | pla          |            | Cancella da Stack codice errore        |
| f6ae: 38       | sec          |            | Fissa il flag di carry come segnal.    |
| f6af: 60       | rts          |            |  |

## TAVOLA DEI MESSAGGI DI CONTROLLO E DI SISTEMA

|                               |               |
|-------------------------------|---------------|
| f6b0: 0d 49 2f 4f 20 45 52 52 | <cr> i/o err  |
| f6b8: 4f 52 20 a3 0d 53 45 41 | or <cr> sea   |
| f6c0: 52 43 48 49 4e 47 a0 46 | rching f      |
| c8: 4f 52 a0 0d 50 52 45 53   | or <cr> pres  |
| r6d0: 53 20 50 4c 41 59 20 4f | s play o      |
| f6d8: 4e 20 54 41 50 c5 50 52 | n tapEpr      |
| f6e0: 45 53 53 20 52 45 43 4f | ess reco      |
| f6e8: 52 44 20 26 20 50 4c 41 | rd & pla      |
| f6f0: 59 20 4f 4e 20 54 41 50 | y on tap      |
| f6f8: c5 0d 4c 4f 41 44 49 4e | E <cr> loadin |
| f700: c7 0d 53 41 56 49 4e 47 | G <cr> saving |
| f708: a0 0d 56 45 52 49 46 59 | <cr> verify   |
| f710: 49 4e c7 0d 46 4f 55 4e | inG <cr> foun |

f718: 44 a0 0d 4f 4b 8d            d <cr> ok

```

f71e: 24 9d      bit $9d      [ msgflg ]  Contr. se consent uscite messaggi
f720: 10 0d      bpl * $f72f    Se neg. esci
f722: b9 b0 f6   lda * $f6b0,y  Leggi byte da tavola messaggi
f725: 08         php          Immagazzina suStack
f726: 29 7f      and #$7f      Es. mask out su bit 7, nessun car.RVS
f728: 20 d2 ff   jsr * $ffd2 [ bsout ]  Vai a rout.BSOUT
f72b: c8         iny          Incr. di 1 per vis.pross.carattere
f72c: 28         plp          Preleva carattere da Stack
f72d: 10 f3      bpl * $f722    Contr.se bit 7 e' messo come segnal.
f72f: 18         clc          Ese. un clear di carry
f730: 60         rts

```

#### ROUTINE KERNAL SETNAM

Fissa i parametri per nome del file

```

f731: 85 b7      sta $b7      [ fnlen ]  Metti in A byte per lungh.nome file
f733: 86 bb      stx $bb      [ fnadr ]  Metti in X byte per indir.LO nome file
f735: 84 bc      sty $bc      [ fnadr+ 1 ] Metti in Y byte per indir.HI nome file
f737: 60         rts

```

#### ROUTINE KERNAL SETLFS

Fissa i parametri per file logico

```

f738: 85 b8      sta $b8      [ la ]     Metti in A byte per n.file logico
f73a: 86 ba      stx $ba      [ fa ]     Metti in X byte per ind. periferica
f73c: 84 b9      sty $b9      [ sa ]     Metti in Y byte per ind. secondario
f73e: 60         rts

```

#### ROUTINE KERNAL SETBNK

Fissa i banchi di memoria

```

f73f: 85 c6      sta $c6      [ ba ]     Metti in A n.att.banco per LSV
f741: 86 c7      stx $c7      [ fnbank ] Metti in X n.att.banco per nome file
f743: 60         rts

```

#### ROUTINE KERNAL READST

Leggi lo STATUS del sistema

```

f744: a5 ba      lda $ba      [ fa ]     Metti in A indirizzo periferica
f746: c9 02      cmp #$02      Contr.se e' indirizzata la RS-232
f748: d0 0b      bne * $f755    Se neg. metti normale Status
f74a: ad 14 0a   lda $0a14     Esegui quanto sopra ma per RS-232
f74d: 48         pha          Immagazzina sullo Stack
f74e: a9 00      lda #$00      Carica A con $00 su RS-232
f750: 8d 14 0a   sta $0a14     Prendi lo Status come tutto OK
f753: 68         pla          Preleva status di RS-232 da Stack
f754: 60         rts

```

```

f755: a5 90      lda $90      [ status ]  Metti lo Status in A
f757: 05 90      ora $90      [ status ]  Esegui un OR fra A e Status sistema
f759: 85 90      sta $90      [ status ]  Immag. in A per Status
f75b: 60         rts

```

#### ROUTINE KERNAL SETMSG

Abilita messaggi di controllo sistema

```

f75c: 85 9d      sta $9d      [ msgflg ]  Immag.byte per mess.contr.sistema
f75e: 60         rts

```

#### ROUTINE KERNAL SETTMO

f75f: 8d 0e 0a sta \$0a0e  
f762: 60 rts

Immag. A nel Flag di timeout IEEE

-----  
ROUTINE KERNAL MEMTOP

f763: 90 06 bcc \* \$f76b  
f765: ae 07 0a ldx \$0a07  
f768: ac 08 0a ldy \$0a08  
f76b: 8e 07 0a stx \$0a07  
f76e: 8c 08 0a sty \$0a08  
f771: 60 rts

Contr.se carry a 1 allora leggi  
Questa routine serve per mettere nei  
registri X e Y rispettivamente gli  
indirizzi LO e HI della fine della  
memoria RAM

-----  
ROUTINE KERNAL MEMBOT

f772: 90 06 bcc \* \$f77a  
f774: ae 05 0a ldx \$0a05  
f777: ac 06 0a ldy \$0a06  
f77a: 8e 05 0a stx \$0a05  
f77d: 8c 06 0a sty \$0a06  
f780: 60 rts

Stessa routine della precedente che  
pero' immag. rispettivamente in X e Y  
gli indirizzi LO e HI di inizio  
RAM

-----  
ROUTINE KERNAL IOBASE

f781: a2 00 ldx #\$00  
f783: a0 d0 ldy #\$d0  
f785: 60 rts

Metti in X ind.LO di I/O  
Metti in Y ind.HI di I/O

-----  
ROUTINE KERNAL LKUPSA

f786: 98 tya  
f787: a6 98 ldx \$98 [ 1dnd ]  
f789: ca dex  
f78a: 30 0f bmi \* \$f79b  
f78c: dd 76 03 cmp \$0376,x  
f78f: d0 f8 bne \* \$f789  
f791: 20 12 f2 jsr \* \$f212  
f794: aa tax  
f795: a5 b8 lda \$b8 [ 1a ]  
f797: a4 b9 ldy \$b9 [ sa ]  
f799: 18 clc  
f79a: 60 rts  
f79b: 38 sec  
f79c: 60 rts

Metti in A l' indirizzo secondario  
Metti in X n.di files aperti  
Decrementa di 1  
Se confr.fra indice e q.s negativo es  
Confr. con byte HI da tav. SA  
Se non trovato esegui altro confr.  
Preleva LFN,DA,SA da tavola corris.  
Copia DA rilev. in X  
Metti in A n. di file logico  
Metti ind. secondario in Y  
Clear di carry

Fissa carry

-----  
ROUTINE KERNAL LKUPLA

f79d: aa tax  
f79e: 20 02 f2 jsr \* \$f202  
f7a1: f0 ee beq \* \$f791  
f7a3: d0 f6 bne \* \$f79b

Immag. valore LFN di ricerca in X  
Metti OK lo status, ricerca tav. LFN  
Tavola trovata.Aggiorna pag.0 ed esci  
Se invece non trovato esci con errore

-----  
ROUTINE KERNAL DMA-CALL

f7a5: bd f0 f7 lda \* \$f7f0,x  
f7a8: 29 fe and #\$fe  
f7aa: aa tax  
f7ab: 4c f0 03 jmp \$03f0

Metti su A val. di X indicizzato  
Es. un AND per mask-out bit 0  
Trasf.val.di conf.sureg.X  
Vaia a banco LO di rout.DMA

-----  
ROUTINE KERNAL FETCH

|                |                     |                                      |
|----------------|---------------------|--------------------------------------|
| f7ae: 8e 35 0a | stx \$0a35          | Immag. valori di X                   |
| f7b1: a6 c7    | ldx \$c7 [ fnbank ] | Carica n.banco per attuale nome file |
| f7b3: a9 bb    | lda #\$bb           | Metti in A \$BB per rout. FETVEC     |
| f7b5: 20 d0 f7 | jsr * \$f7d0        | Vai a INDFET                         |
| f7b8: ae 35 0a | ldx \$0a35          | Ripristina cont.reg. X               |
| f7bb: 60       | rts                 |                                      |

## ROUTINE KERNAL STASH

|                |                 |                                       |
|----------------|-----------------|---------------------------------------|
| f7bc: a2 ac    | ldx #\$ac       | Punt. a ind.1(LO) di LSV di I/O       |
| f7be: 2c       | .byte \$2c      | Vai a \$F7C1                          |
| f7bf: a2 ae    | ldx #\$ae       | Punt. a ind.2(LO) di LSV di I/O       |
| f7c1: 8e b9 02 | stx \$02b9      | Metti in STATVEC cont.reg. X          |
| f7c4: a6 c6    | ldx \$c6 [ ba ] | Metti in X n.banco attuale LSV        |
| f7c6: 4c da f7 | jmp * \$f7da    | Vai a rout. INDSTA                    |
| f7c9: a9 ae    | lda #\$ae       | Punt. x ind.1(LO) di LSV              |
| f7cb: 2c       | .byte \$2c      | Vai a \$F7CE                          |
| f7cc: a9 ac    | lda #\$ac       | Punt.per ind.2(LO) di LSV             |
| f7ce: a6 c6    | ldx \$c6 [ ba ] | Metti in X n.banco attuale LSV        |
| f7d0: 8d aa 02 | sta \$02aa      | Metti cont. di A in FETVEC            |
| f7d3: bd f0 f7 | lda * \$f7f0,x  | Metti in A valore determ.di configur. |
| f7d6: aa       | tax             | da tavola e trasf. in X               |
| f7d7: 4c a2 02 | jmp \$02a2      | Vai a FETCH                           |
| f7da: 48       | pha             | Immag. in A cont.per coma. STA        |
| f7db: bd f0 f7 | lda * \$f7f0,x  | Metti in A val.configurazione dato da |
| f7de: aa       | tax             | X ril.da tavola.Trasf.quindi in X     |
| f7df: 68       | pla             | Metti cont. in A per confronto        |
| f7e0: 4c af 02 | jmp \$02af      | Vai a rout. di CMPARE                 |
| f7e3: 48       | pha             | Metti in X val.config.                |
| f7e4: bd f0 f7 | lda * \$f7f0,x  | Fissa val. determ. di config per X    |
| f7e7: aa       | tax             | dalla tavola                          |
| f7e8: 68       | pla             | A per confronto                       |
| f7e9: 4c be 02 | jmp \$02be      | Via a rout. CMPARE                    |
| f7ec: bd f0 f7 | lda * \$f7f0,x  | Carica X con config. definita         |
| f7ef: 60       | rts             |                                       |

|                   |             |
|-------------------|-------------|
| f7f0: 3f 7f bf ff | 16 56 96 d6 |
| f7f8: 2a 6a aa ea | 06 0a 01 00 |

Tavola di configurazione per operazi  
ALL FAR

|                |                         |                                       |
|----------------|-------------------------|---------------------------------------|
| f800: ad 00 ff | lda * \$ff00            | Salva in A att.val. di configurazione |
| f803: 8e 00 ff | stx * \$ff00            | Metti in X nuovo val. config.         |
| f806: aa       | tax                     | Trasf. vecchio val.in X               |
| f807: b1 ff    | lda (\$ff),y [ lofbuf ] | Metti in A valore di Y                |
| f809: 8e 00 ff | stx * \$ff00            | Ripristina vecchia configurazione     |
| f80c: 60       | rts                     |                                       |

|                |                         |                                      |
|----------------|-------------------------|--------------------------------------|
| f80d: 48       | pha                     | Metti cont. A su Stack               |
| f80e: ad 00 ff | lda * \$ff00            | Metti in A att. val. di configuraz.  |
| f811: 8e 00 ff | stx * \$ff00            | Fissa in X nuovo val. di config.     |
| f814: aa       | tax                     | Trasferisci in X vecchio valore conf |
| f815: 68       | pla                     | Ripristina su Stack valore config.   |
| f816: 91 ff    | sta (\$ff),y [ lofbuf ] | Metti su val. di X                   |
| f818: 8e 00 ff | stx * \$ff00            | Ripristina vecchio val. config.      |
| f81b: 60       | rts                     |                                      |

|          |     |
|----------|-----|
| f81c: 48 | pha |
|----------|-----|

Salva su Stack valore per CMPARE

```

f81d: ad 00 ff lda * $ff00
f820: 8e 00 ff stx * $ff00
f823: aa tax
f824: 68 pla
f825: d1 ff cmp ($ff),y [ lofbuf ]
f827: 8e 00 ff stx * $ff00
f82a: 60 rts

```

Salva su A vecchia config.  
 Fissa via X nuovo val. di config.  
 Trasfer. vecchio valore in X  
 Riprendi valore di CMPARE  
 Confronta con val. di Y  
 Ripristina vecchia config.

```

f82b: 20 e3 02 jsr $02e3
f82e: 85 06 sta $06 [ a-reg ]
f830: 86 07 stx $07 [ x-reg ]
f832: 84 08 sty $08 [ y-reg ]
f834: 08 php
f835: 68 pla
f836: 85 05 sta $05 [ s-reg ]
f838: ba tsx
f839: 86 09 stx $09 [ charac ]
f83b: a9 00 lda #$00
f83d: 8d 00 ff sta * $ff00
f840: 60 rts

```

Vai a rout. JMPFAR  
 Salva su A valori rout. prec.  
 Idem per X  
 Idem per Y  
 Salva status del processore su Stack  
 Immetti in A status  
 Salva in Pagina zero  
 Trasf. in X stack  
 Salva in pag. zero punt. immagazz.  
 Metti \$00 in registro di config  
 Abilita tutti i sistemi di ROM

```

f841: a2 00 ldx #$00
f843: b5 03 lda $03,x [ pc-hi ]
f845: 48 pha
f846: e8 inx
f847: e0 03 cpx #$03
f849: 90 f8 bcc * $f843
f84b: a6 02 ldx $02 [ bank ]
f84d: 20 6b ff jsr * $ff6b [ getcgf ]
f850: 8d 00 ff sta * $ff00
f853: a5 06 lda $06 [ a-reg ]
f855: a6 07 ldx $07 [ x-reg ]
f857: a4 08 ldy $08 [ y-reg ]
f859: 40 rti
f85a: ae 00 ff ldx * $ff00
f85d: 8c 01 df sty * $df01
f860: 8d 00 ff sta * $ff00
f863: 8e 00 ff stx * $ff00
f866: 60 rts

```

In questo ciclo, il valore immesso in pagina 0 per i bytes \$03-\$04-\$05 per il contatore di programma e per lo status del proc. vengono immagazz. sullo Stack. Confronto con cont. X alla fine della routine.  
 Carica punt. banco per spos. config.  
 Vai a routine GETCFG  
 Fissa registri di configuraz.  
 Caricamento di tutti i registri in pagina zero A,X,Y

Vai a indirizzo cont. programmi  
 Metti reg. di config. in X  
 Fissa controller DMA  
 Registro di config. in A  
 Registro di config. in X

# ROUTINE KERNAL PHOENIX

```

f867: 78 sei
f868: a2 03 ldx #$03
f86a: 8e c0 0a stx $0ac0
f86d: ae c0 0a ldx $0ac0
f870: bd c1 0a lda $0acl,x
f873: f0 11 beq * $f886
f875: a0 00 ldy #$00
f877: bd bc e2 lda * $e2bc,x
f87a: 85 03 sta $03 [ pc-hi ]
f87c: 84 04 sty $04 [ pc-lo ]
f87e: bd c0 e2 lda * $e2c0,x
f881: 85 02 sta $02 [ bank ]
f883: 20 cd 02 jsr $02cd
f886: ce c0 0a dec $0ac0
f889: 10 e2 bpl * $f86d
f88b: 58 cli

```

Disabilita tutti gli interrupt  
 Inizial. banchi e puntatori  
 per cartridge esterno in banco 3  
 Metti punt. spostam. in X  
 Contr. tavola ID per spazio cartr.  
 Contr. se tavola ingresso =0  
 Fissa ind. di ingresso LO a \$00  
 Metti in A indir. HI da tavola  
 Metti A in ind.HI in PC HI  
 Metti Y in ind. LO in PC LO  
 Carica A con valore banco da Bank tab.  
 Immag.val. di cui sopra in Pag.0  
 Vai a JSRFAR  
 Decrem.punt. spostam. di 1  
 Controlla le 4 aree di cartridge  
 Abilita tutti gli interrupt

|       |          |                |           |  |
|-------|----------|----------------|-----------|--|
| f88c: | a2 08    | ldx #\$08      |           | Indirizzo periferica per BOOT          |
| f88e: | a9 30    | lda #\$30      |           | Metti (0) in A                         |
| f890: | 85 bf    | sta \$bf       | [ drive ] | Byte di pag. 0 per buffer seriale      |
| f892: | 86 ba    | stx \$ba       | [ fa ]    | Fissa indirizzo perif. x disco 8       |
| f894: | 8a       | txa            |           | Copia indirizzo perif. in A            |
| f895: | 20 3d f2 | jsr * \$f23d   |           | Fissa lo standard per perif. I/O       |
| f898: | a2 00    | ldx #\$00      |           | Inizializza lungh.cont.per BOOT        |
| f89a: | 86 9f    | stx \$9f       | [ ptr2 ]  | Carica nome file con 0                 |
| f89c: | 86 c2    | stx \$c2       | [ stah ]  | Fissa n. settore per BOOT              |
| f89e: | e8       | inx            |           | Incrementa cont. di 1                  |
| f89f: | 86 c1    | stx \$c1       | [ track ] | Fissa n. traccia per BOOT              |
| f8a1: | c8       | iny            |           | Incrementa registro di ciclo Y         |
| f8a2: | d0 fd    | bne * \$f8a1   |           | Esegui ciclo 256 volte fino a reg=0    |
| f8a4: | e8       | inx            |           | Idem per X                             |
| f8a5: | d0 fa    | bne * \$f8a1   |           | Idem per X                             |
| f8a7: | a2 0c    | ldx #\$0c      |           | Spostam. puntatore per DOS             |
| f8a9: | bd 08 fa | lda * \$fa08,x |           | Carica caratt. per com. BOOT del DOS   |
| f8ac: | 9d 00 01 | sta \$0100,x   |           | Copia q.s. in buffer DOS               |
| f8af: | ca       | dex            |           | Decrementa punt. di 1                  |
| f8b0: | 10 f7    | bpl * \$f8a9   |           | Ciclo di trasf di 11 caratteri         |
| f8b2: | a5 bf    | lda \$bf       | [ drive ] | Prendi da A n. drive da Pag 0          |
| f8b4: | 8d 06 01 | sta \$0106     |           | Immetti A in buffer di DOS             |
| f8b7: | a9 00    | lda #\$00      |           | Chiam.n.banco per attuale LVS          |
| f8b9: | a2 0f    | ldx #\$0f      |           | N. banco per attuale nome file         |
| f8bb: | 20 3f f7 | jsr * \$f73f   |           | Vai a rout. SETBANK                    |
| f8be: | a9 01    | lda #\$01      |           | Fissa =1 lunghezza nome file           |
| f8c0: | a2 15    | ldx #\$15      |           | Fissa ind. LO di nome file             |
| f8c2: | a0 fa    | ldy \$fa       |           | Fissa ind. HI di nome file             |
| f8c4: | 20 31 f7 | jsr * \$f731   |           | Vai a rout. SETNAM                     |
| f8c7: | a9 00    | lda #\$00      |           | Poni in A n. di file logico            |
| f8c9: | a0 0f    | ldy #\$0f      |           | Poni in Y indir. secondario            |
| f8cb: | a6 ba    | ldx \$ba       | [ fa ]    | Poni in X indirizzo periferica         |
| f8cd: | 20 38 f7 | jsr * \$f738   |           | Vai a rout. SETLFS                     |
| f8d0: | 20 c0 ff | jsr * \$ffc0   | [ open ]  | Vai a rout. OPEN                       |
| f8d3: | b0 16    | bcs * \$f8eb   |           | Contr. rilev. errore.Se pos.fine BOOT  |
| f8d5: | a9 01    | lda #\$01      |           | Lunghezza nome file =1                 |
| f8d7: | a2 16    | ldx #\$16      |           | Indir. LO nome file =FA16              |
| f8d9: | a0 fa    | ldy \$fa       |           | Indir. HI nome file = #                |
| f8db: | 20 31 f7 | jsr * \$f731   |           | Vai a rout. SETNAM                     |
| f8de: | a9 0d    | lda #\$0d      |           | N. file logico in A                    |
| f8e0: | a8       | tay            |           | Poni q.s. come indir. secondario       |
| f8e1: | a6 ba    | ldx \$ba       | [ fa ]    | N. di periferica in X                  |
| f8e3: | 20 38 f7 | jsr * \$f738   |           | Vai a SETLFS                           |
| f8e6: | 20 c0 ff | jsr * \$ffc0   | [ open ]  | Vai a OPEN                             |
| f8e9: | 90 03    | bcc * \$f8ee   |           | Esegui un clear globale e continua     |
| f8eb: | 4c 8b f9 | jmp * \$f98b   |           | Inizializza disco                      |
| f8ee: | a9 00    | lda #\$00      |           | Iniz. II byte di pag. 0                |
| f8f0: | a0 0b    | ldy #\$0b      |           | Poni il punt di indir. inizio a        |
| f8f2: | 85 ac    | sta \$ac       | [ sal ]   | \$0b00                                 |
| f8f4: | 84 ad    | sty \$ad       | [ sah ]   | ***                                    |
| f8f6: | 20 d5 f9 | jsr * \$f9d5   |           | Carica inizio in Buffer cassetta       |
| f8f9: | a2 00    | ldx #\$00      |           | Esegui un clear di ciclo               |
| f8fb: | bd 00 0b | lda \$0b00,x   |           | Controlla che i primi 3 bytes del sett |
| f8fe: | dd c4 e2 | cmp * \$e2c4,x |           | iniziale disco letti nel buffer di     |
| f901: | d0 e8    | bne * \$f8eb   |           | cassetta siano codice di Autostart     |
| f903: | e8       | inx            |           | C-B-M                                  |
| f904: | e0 03    | cpx #\$03      |           | Se pos. allora e' un programma di      |
| f906: | 90 f3    | bcc * \$f8fb   |           | caricamento                            |
| f908: | 20 17 fa | jsr * \$fa17   |           | Vai a PRIMM                            |



f90b: 0d 42 4f 4f 54 49 4e 47 <cr>  
 f913: 20 00 booting

Costanti KERNAL per messaggio di caricamento

# FISSA I PUNTATORI

f915: bd 00 0b lda \$0b00,x  
 f918: 95 a9 sta \$a9,x [ rinone ]  
 f91a: e8 inx  
 f91b: e0 07 cpx #\$07  
 f91d: 90 f6 bcc \* \$f915  
 f91f: bd 00 0b lda \$0b00,x  
 f922: f0 06 beq \* \$f92a  
 f924: 20 d2 ff jsr \* \$ffd2[ bsout ]  
 f927: e8 inx  
 f928: d0 f5 bne \* \$f91f  
 f92a: 86 9e stx \$9e [ ptr1 ]  
 f92c: 20 17 fa jsr \* \$fal7

Poni a \$0B03 indirizzi dei puntatori di LOAD e inizializza pag. 0  
 \*\*\*  
 Confronto puntatori  
 Ciclo per caricamento punt.  
 Prendi caratt. out. da buffer di cass  
 Contr.per val. \$00  
 Vai a rout. BSOUT  
 Increm. per buffer cassetta  
 Salto incondiz. a car. output  
 Immagazzina in buffer di cassetta  
 Vai a routine PRIMM

# COSTANTI PER MESSAGGIO DI BOOT

f92f: 2e 2e 2e ...

f932: 0d 00 a5 ora \$a500  
 f935: ae 85 c6 ldx \* \$c685  
 f938: a5 af lda \$af [ eah ]  
 f93a: f0 09 beq \* \$f945  
 f93c: c6 af dec \$af [ eah ]  
 f93e: 20 b3 f9 jsr \* \$f9b3  
 f941: e6 ad inc \$ad [ sah ]  
 f943: d0 f3 bne \* \$f938  
 f945: 20 8b f9 jsr \* \$f98b  
 f948: a6 9e ldx \$9e [ ptr1 ]  
 f94a: 2c .byte \$2c  
 f94b: e6 9f inx \$9f  
 f94d: e8 inx  
 f94e: bd 00 0b lda \$0b00,x  
 f951: d0 f8 bne \* \$f94b  
 f953: e8 inx  
 f954: 86 04 stx \$04 [ pc-lo ]  
 f956: a6 9e ldx \$9e [ ptr1 ]  
 f958: a9 3a lda #\$3a  
 f95a: 9d 00 0b sta \$0b00,x  
 f95d: ca dex  
 f95e: a5 bf lda \$bf [ drive ]  
 f960: 9d 00 0b sta \$0b00,x  
 f963: 86 9e stx \$9e [ ptr1 ]  
 f965: a6 9f ldx \$9f [ ptr2 ]  
 f967: f0 15 beq \* \$f97e  
 f969: e8 inx  
 f96a: e8 inx  
 f96b: 8a txa  
 f96c: a6 9e ldx \$9e [ ptr1 ]  
 f96e: a0 0b ldy #\$0b  
 f970: 20 31 f7 jsr \* \$f731  
 f973: a9 00 lda #\$00  
 f975: aa tax  
 f976: 20 3f f7 jsr \* \$f73f  
 f979: a9 00 lda #\$00

Punt.sett.BOOT in banco  
 Copia punt. per STASH  
 Contatore per n.blocchi di BOOT  
 Contr. se tutti blocchi BOOT letti  
 Decrem. blocchi di BOOT -1  
 Carica da disco pross.traccia/sett  
 Increm ind. di caric. HI di 1  
 Leggi prossimo blocco  
 Inizializza disco per BOOT  
 Spost. a buffer di cassetta  
 Vai a \$F94D  
 Incr.lungh.cont.nome file  
 Fissa per caratt. dopo 0  
 Carattere dopo nome file  
 Se non=0 continua lett.  
 Fissa per caratt. dopo 0  
 Immettilo in PC LO  
 Posiz. su punt. prima nome file  
 Rimpiazza 0 con (:)  
 Metti q.s. prima del nome file  
 Decrem car. prima di (:)  
 Carattere ASCII del drive  
 Metti (0:xxxx) del nome file  
 Salva indirizzo LO nome file  
 Lunghezza nome file  
 Se nessun nome vai a indir.  
 Incrementa di 2 puntatore nome file  
 \*\*\*  
 Copia in A lunghezza nome file  
 Indirizzo LO nome file  
 Fissa ind. HI nome file  
 Vai a SETNAM  
 Iniz. A e X con \$00  
 Trasf per routine SETBANK  
 Vai a rout. SETBANK  
 Fissa A come marker di LOAD

|                |                    |                                       |
|----------------|--------------------|---------------------------------------|
| f97b: 20 69 f2 | jsr * \$f269       | Vai a vettore kernal LOAD             |
| f97e: a9 0b    | lda #\$0b          | Fissa pag. 0 per PC HI                |
| f980: 85 03    | sta \$03 [ pc-hi ] | Q.s. in \$0B                          |
| f982: a9 0f    | lda #\$0f          | Fissa il punt. di pag. 0 al valore di |
| f984: 85 02    | sta \$02 [ bank ]  | \$0f (banco)                          |
| f986: 20 cd 02 | jsr \$02cd         | Vai a rout. JSRFAIR                   |
| f989: 18       | clc                | Esegui un clear di carry              |
| f98a: 60       | rts                |                                       |

-----  
INIZIALIZZA FLOPPY PER BOOT

|                |                        |                              |
|----------------|------------------------|------------------------------|
| f98b: 08       | php                    | Salva STATUS su Stack        |
| f98c: 48       | pha                    | Salva A su Stack             |
| f98d: 20 cc ff | jsr * \$ffcc [ clrch ] | Vai a CLRCH                  |
| f990: a9 0d    | lda #\$0d              | Chiudi n. di file logico     |
| f992: 18       | clc                    | Clear di carry per OK        |
| f993: 20 c3 ff | jsr * \$ffc3 [ close ] | Vai a rout. CLOSE            |
| f996: a2 00    | ldx #\$00              | Fissa per output file logico |
| f998: 20 c9 ff | jsr * \$ffc9 [ ckout ] | Vai a rout. CKOUT            |
| f99b: b0 0a    | bcs * \$f9a7           | Se ril. errore chiudi        |
| f99d: a9 55    | lda #\$55              | Metti in A carattere U       |
| f99f: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a BSOUT                  |
| f9a2: a9 49    | lda #\$49              | I in A                       |
| f9a4: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a rout. BSOUT            |
| f9a7: 20 cc ff | jsr * \$ffcc [ clrch ] | Vai a rout. CLRCH            |
| f9aa: a9 00    | lda #\$00              | Chiudi n. di file logico     |
| f9ac: 38       | sec                    | Fissa carry per OK           |
| f9ad: 20 c3 ff | jsr * \$ffc3 [ close ] | Vai a rout. CLOSE            |
| f9b0: 68       | pla                    | Metti in A cont. di Stack    |
| f9b1: 28       | plp                    | Ripristina Status            |
| f9b2: 60       | rts                    |                              |

-----  
RESET DI TRACCIA E SETTORE

|                |                        |                                   |
|----------------|------------------------|-----------------------------------|
| f9b3: a6 c2    | ldx \$c2 [ stah ]      | N. di settore da pag. 0           |
| f9b5: e8       | inx                    | Incrementa settore di 1           |
| f9b6: e0 15    | cpx #\$15              | Contr. di validita' per settore   |
| f9b8: 90 04    | bcc * \$f9be           | OK se n.settore <21               |
| f9ba: a2 00    | ldx #\$00              | Carica val. per n.settore =0      |
| f9bc: e6 c1    | inc \$c1 [ track ]     | Incrementa n.traccia              |
| f9be: 86 c2    | stx \$c2 [ stah ]      | N. settore = a pag. 0             |
| f9c0: 8a       | txa                    | Copia n. sett. in A e converti il |
| f9c1: 20 fb f9 | jsr * \$f9fb           | settore in 2-byte ascii           |
| f9c4: 8d 00 01 | sta \$0100             | Metti n. settoreLO in buffer DOS  |
| f9c7: 8e 01 01 | stx \$0101             | N. settore HI in buffer DOS       |
| f9ca: a5 c1    | lda \$c1 [ track ]     | Metti n. traccia in A             |
| f9cc: 20 fb f9 | jsr * \$f9fb           | Converti traccia in 2 Bytes ASCII |
| f9cf: 8d 03 01 | sta \$0103             | N. traccia LO in buffer DOS       |
| f9d2: 8e 04 01 | stx \$0104             | N. traccia HI in buffer DOS       |
| f9d5: a2 00    | ldx #\$00              | Fissa file logico n.0 da CKOUT    |
| f9d7: 20 c9 ff | jsr * \$ffc9 [ ckout ] | Vai a rout. CKOUT                 |
| f9da: a2 0c    | ldx #\$0c              | Preleva 13 car. da buffer DOS     |
| f9dc: bd 00 01 | lda \$0100,x           | 1 carattere da buffer output      |
| f9df: 20 d2 ff | jsr * \$ffd2 [ bsout ] | Vai a rout. BSOUT                 |
| f9e2: ca       | dex                    | Decrementa cont.buffer DOS per 13 |
| f9e3: 10 f7    | bpl * \$f9dc           | volte per uscita caratteri        |
| f9e5: 20 cc ff | jsr * \$ffcc [ clrch ] | Vai a rout. CLRCH                 |
| f9e8: a2 0d    | ldx #\$0d              | Fissa file logico per INPUT       |
| f9ea: 20 c6 ff | jsr * \$ffc6 [ chkin ] | Vai a CHKIN                       |
| f9ed: a0 00    | ldy #\$00              | Posiz. a val. 0 per STASH         |

|                |                        |                                 |
|----------------|------------------------|---------------------------------|
| f9ef: 20 cf ff | jsr * \$ffcf [ basin ] | Vaia a rout. BASIN              |
| f9f2: 20 bc f7 | jsr * \$f7bc           | Vai a STASH                     |
| f9f5: c8       | iny                    | Incrementa punt. STASH          |
| f9f6: d0 f7    | bne * \$f9ef           | Esegui q.s. per 256 bytes letti |
| f9f8: 4c cc ff | jmp * \$ffcc [ clrch ] | Vai a rout. CLRCH               |
| f9fb: a2 30    | ldx #\$30              | Valore ASCII per 0 in X         |
| f9fd: 38       | sec                    | Fissa Carry per sottrazione     |
| f9fe: e9 0a    | sbc #\$0a              | Sottrai 10 (in dec) da A        |
| fa00: 90 03    | bcc * \$fa05           | Esegui un clear di carry        |
| fa02: e8       | inx                    | Incrementa car. ASCII di 1      |
| fa03: b0 f9    | bcs * \$f9fe           | Salto incondizionato a \$f9FE   |
| fa05: 69 3a    | adc #\$3a              | Crea ASCII LO                   |
| fa06: 60       | rts                    |                                 |

-----  
COSTANTI KERNAL PER BOOT-LOAD

fa08: 30 30 20 31 30 20 30 20  
 fa10: 33 31 3a 31 55 49 23

-----  
ROUTINE KERNAL PRIMM

|                |                         |   |
|----------------|-------------------------|---|
| fa17: 48       | pha                     | Metti conten. di A su Stack             |
| fa18: 8a       | txa                     | Salva cont. att. reg. X sullo Stack     |
| fa19: 48       | pha                     | tramite Accumulatore                    |
| fala: 98       | tya                     | Salva attuale cont. reg. Y sullo        |
| falb: 48       | pha                     | Stack a mezzo Accumulatore              |
| falc: a0 00    | ldy #\$00               | Carica spost. puntatore con \$00        |
| fale: ba       | tsx                     | Trasf. Stack in X                       |
| falf: fe 04 01 | inc \$0104,x            | Incr.byte LO di ind. RTS in Stack       |
| fa22: d0 03    | bne * \$fa27            | Contr.overflow, se nessun overfl. salta |
| fa24: fe 05 01 | inc \$0105,x            | Incr.byte HI di ind. RTS in Stack       |
| fa27: bd 04 01 | lda \$0104,x            | Metti byte LO di ind. RTS di Stack      |
| fa2a: 85 ce    | sta \$ce [ imparm ]     | in pag. 0                               |
| fa2c: bd 05 01 | lda \$0105,x            | Come sopra per ind. HI                  |
| fa2f: 85 cf    | sta \$cf [ imparm+1 ]   | ***                                     |
| fa31: b1 ce    | lda (\$ce),y [ imparm ] | Byte da ind. RTS + reg. Y in A          |
| fa33: f0 05    | beq * \$fa3a            | Contr. se \$00                          |
| fa35: 20 d2 ff | jsr * \$ffd2 [ bsout ]  | Vai a rout. kernal BSOUT                |
| fa38: 90 e4    | bcc * \$fale            | Se nessun errore prossimo carattere     |
| fa3a: 68       | pla                     | Prendi un byte da Stack e ripristina    |
| fa3b: a8       | tay                     | vecchi contenuti di Y                   |
| fa3c: 68       | pla                     | Come sopra per X                        |
| fa3d: aa       | tax                     | ***                                     |
| fa3e: 68       | pla                     | Ripristina vecchi cont. A               |
| fa3f: 60       | rts                     |   |

-----  
ROUTINE NMI

|                |                       |                                      |
|----------------|-----------------------|--------------------------------------|
| fa40: d8       | cld                   | Resetta modo decimale                |
| fa41: a9 7f    | lda #\$7f             | Fissa segnalatore NMI                |
| fa43: 8d 0d dd | sta * \$dd0d          | Esegui un clear su NMI               |
| fa46: ac 0d dd | ldy * \$dd0d          | Lettura e clear dei Flags            |
| fa49: 30 14    | bmi * \$fa5f          | Controlla se RS-232 e' attiva        |
| fa4b: 20 3d f6 | jsr * \$f63d          | Leggi RUN/STOP in modo SHIFT         |
| fa4e: 20 e1 ff | jsr * \$ffef [ stop ] | Vai a routine per STOP. Contr.tasto  |
| fa51: d0 0c    | bne * \$fa5f          | Se tasto STOP non premuto, vai a I/O |
| fa53: 20 56 e0 | jsr * \$e056          | Vettori standard per I/O + interrupt |
| fa56: 20 09 e1 | jsr * \$e109          | Inizializza I/O                      |
| fa59: 20 00 c0 | jsr * \$c000          | Inizializza I/O e clear di schermo   |

|                |              |                                 |
|----------------|--------------|---------------------------------|
| fa5c: 6c 00 0a | jmp (\$0a00) | Ingresso WARM START del basic   |
| fa5f: 20 05 e8 | jsr * \$e805 | Vai a rout. NMI per RS-232      |
| fa62: 4c 33 ff | jmp * \$ff33 | Ritorna a rout. di chiamata IRQ |

-----  
ROUTINE IRQ

|                |              |                                    |
|----------------|--------------|------------------------------------|
| fa65: d8       | cld          | Esegui un reset del modo decimale  |
| fa66: 20 24 c0 | jsr * \$c024 | Ingresso a editor routine IRQ      |
| fa69: 90 12    | bcc * \$fa7d | Uscita di IRQ per raster interrupt |
| fa6b: 20 f8 f5 | jsr * \$f5f8 | Vai a UDTIM                        |
| fa6e: 20 d0 ee | jsr * \$eed0 | Controllo nastro tastiera          |
| fa71: ad 0d dc | lda * \$dc0d | Carica reg. contr. interrupt CIA   |
| fa74: ad 04 0a | lda \$0a04   | Carica punt. di status NMI/reset   |
| fa77: 4a       | lsr a        | Controlla se bit 0 e' meso a =0    |
| fa78: 90 03    | bcc * \$fa7d | Se pos. ritorna a IRQ              |
| fa7a: 20 06 40 | jsr \$4006   | Ingresso Basic per IRQ             |
| fa7d: 4c 33 ff | jmp * \$ff33 | Vai a rout. di chiamata per IRQ    |

-----  
TAVOLA DI DECODIFICA TASTIERA SET DI CARATTERI ASCII

```

fa80: 14 0d 1d 88 85 86 87 11
fa88: 33 57 41 34 5a 53 45 01
fa90: 35 52 44 36 43 46 54 58
fa98: 37 59 47 38 42 48 55 56
faa0: 39 49 4a 30 4d 4b 4f 4e
faa8: 2b 50 4c 2d 2e 3a 40 2c
fab0: 5c 2a 3b 13 01 3d 5e 2f
fab8: 31 5f 04 32 20 02 51 03
fac0: 84 38 35 09 32 34 37 31
fac8: 1b 2b 2d 0a 0d 36 39 33
fad0: 08 30 2e 91 11 9d 1d ff
fad8: ff

```

-----  
TAVOLA DI DECODIFICA TASTIERA SET DI CARATTERI ASCII + SHIFT

```

fad9: 94 8d 9d 8c 89 8a 8b 91
fael: 23 d7 c1 24 da d3 c5 01
fae9: 25 d2 c4 26 c3 c6 d4 d8
faf1: 27 d9 c7 28 c2 c8 d5 d6
faf9: 29 c9 ca 30 cd cb cf ce
fb01: db d0 cc dd 3e 5b ba 3c
fb09: a9 c0 5d 93 01 3d de 3f
fb11: 21 5f 04 22 a0 02 d1 83
fb19: 84 38 35 18 32 34 37 31
fb21: 1b 2b 2d 0a 8d 36 39 33
fb29: 08 30 2e 91 11 9d 1d ff
fb31: ff

```

-----  
TAVOLA DI DECODIFICA TASTIERA SET DI CARATTERI ASCII + TASTO COMMODORE

```

fb32: 94 8d 9d 8c 89 8a 8b 91
fb3a: 96 b3 b0 97 ad ae b1 01
fb42: 98 b2 ac 99 bc bb a3 bd
fb4a: 9a b7 a5 9b bf b4 b8 be
fb52: 29 a2 b5 30 a7 a1 b9 aa
fb5a: a6 af b6 dc 3e 5b a4 3c
fb62: a8 df 5d 93 01 3d de 3f

```

```
fb6a: 81 5f 04 95 a0 02 ab 03
fb72: 84 38 35 18 32 34 37 31
fb7a: 1b 2b 2d 0a 8d 36 39 33
fb82: 08 30 2e 91 11 9d 1d ff
fb8a: ff
```

---

TAVOLA DI DECODIFICA TASTIERA SET DI CARATTERI ASCII + CTRL

```
fb8b: ff ff ff ff ff ff ff ff
fb93: 1c 17 01 9f 1a 13 05 ff
fb9b: 9c 12 04 1e 03 06 14 18
fba3: 1f 19 07 9e 02 08 15 16
fbab: 12 09 0a 92 0d 0b 0f 0e
fbb3: ff 10 0c ff ff 1b 00 ff
fbbb: 1c ff 1d ff ff 1f 1e ff
fbc3: 90 06 ff 05 ff ff 11 ff
fbc3: 84 38 35 18 32 34 37 31
fbd3: 1b 2b 2d 0a 8d 36 39 33
fbdb: 08 30 2e 91 11 9d 1d ff
fbe3: ff
```

---

TAVOLA DI DECODIFICA TASTIERA SET DI CARATTERI ASCII + ALT

```
fbe4: 14 0d 1d 88 85 86 87 11
fbec: 33 d7 c1 34 da d3 c5 01
fbf4: 35 d2 c4 36 c3 c6 d4 d8
fbfc: 37 d9 c7 38 c2 c8 d5 d6
fc04: 39 c9 ca 30 cd cb cf ce
fc0c: 2b d0 cc 2d 2e 3a 40 2c
fc14: 5c 2a 3b 13 01 3d 5e 2f
fc1c: 31 5f 04 32 20 02 51 03
fc24: 84 38 35 09 32 34 37 31
fc2c: 1b 2b 2d 0a 0d 36 39 33
fc34: 08 30 2e 91 11 9d 1d ff
fc3c: ff
```

---

ZONA LIBERA DA \$FC3D A \$FEFF

---

```
ff00: 00 3f 7f 01 41
```

---

ROUTINE KERNAL NMI

```
ff05: 78      sei
ff06: 48      pha
ff07: 8a      txa
ff08: 48      pha
ff09: 98      tya
ff0a: 48      pha
ff0b: ad 00 ff lda * $ff00
ff0e: 48      pha
ff0f: a9 00   lda #$00
ff11: 8d 00 ff sta * $ff00
ff14: 6c 18 03 jmp ($0318)
```

```
Disabilita tutti interrupt
Immag. cont. di A su Stack
Trasf. cont. reg. X sullo Stack
tramite l' Accumulatore
Come sopra ma per il registro Y
***
Metti in A registri configurazione
Immag. registri config. su Stack
Metti $00 nei reg. di config.
Abilita tutti i sistemi ROM
Vai a vettore diNMI ($FA40)
```

---

ROUTINE KERNAL IRQ

```

ff17: 48      pha      Immag. cont. di A su Stack
ff18: 8a      txa      Trasf. contenuti reg. X su Stack
ff19: 48      pha      tramite l' Accumulatore
ff1a: 98      tya      Come sopra ma per registro Y
ff1b: 48      pha      ***
ff1c: ad 00 ff lda * $ff00 Metti in A registri configurazione
ff1f: 48      pha      Immag. registri config. su Stack
ff20: a9 00   lda #$00   Metti $00 nei reg. di config.
ff22: 8d 00 ff sta * $ff00 Abilita le ROM di sistema
ff25: ba      tsx      Metti cont. di Stack su reg. X
ff26: bd 05 01 lda $0105,x Carica in A byte di Status di CPU
ff29: 29 10   and #$10   Esegui un AND log. con bit di break
ff2b: f0 03   beq * $ff30 Se nessun break continua normal.
ff2d: 6c 16 03 jmp ($0316) Salta al vettore di BRK ($B003)
ff30: 6c 14 03 jmp ($0314) Salta al vettore di IRQ ($FA65)
ff33: 68      pla      Prel. val. prec. di config. da Stack
ff34: 8d 00 ff sta * $ff00 Ripristina config. selezionata prec.
ff37: 68      pla      Prel. un Byte dallo Stack e ripristina
ff38: a8      tay      cont. prec. di Y
ff39: 68      pla      Come sopra per registro X
ff3a: aa      tax      ***
ff3b: 68      pla      Riprist. cont. prec. di A
ff3c: 40      rti      *****

```

## ROUTINE KERNAL DI RESET

```

ff3d: a9 00   lda #$00   Metti $00 nei reg. di configurazione
ff3f: 8d 00 ff sta * $ff00 Abilita tutte le ROM di sistema
ff42: 4c 00 e0 jmp * $e000   Questa zona di memoria contiene i
ff45: ff ff 4c xxx $4cff   puntatori a FSTMOD, EAINIT, C64-MODE
ff48: fb e5 4c xxx $4ce5   DMA-CALL
ff4b: 3d f2 4c and $4cf2,x
ff4e: 4b e2 4c xxx $4ce2
ff51: a5 f7   lda $f7   [ locks ]

```

```

ff53: 4c 90 f8 jmp * $f890   Puntatore a BOOT-CALL
ff56: 4c 67 f8 jmp * $f867   Puntatore a PHOENIX
ff59: 4c 9d f7 jmp * $f79d   Puntatore a LUKUPLA
ff5c: 4c 86 f7 jmp * $f786   Puntatore a LKUPSA
ff5f: 4c 2a c0 jmp * $c02a   Puntatore a SWAPPER
ff62: 4c 27 c0 jmp * $c027   Puntatore a DLCHR
ff65: 4c 21 c0 jmp * $c021   Puntatore a PFKEY
ff68: 4c 3f f7 jmp * $f73f   Puntatore a SETBANK
ff6b: 4c ec f7 jmp * $f7ec   Puntatore a GETCFG
ff6e: 4c cd 02 jmp * $02cd   Puntatore a JSRFAR
ff71: 4c e3 02 jmp * $02e3   Puntatore a JMPFAR
ff74: 4c d0 f7 jmp * $f7d0   Routine INDFET
ff77: 4c da f7 jmp * $f7da   Routine INDSTA
ff7a: 4c e3 f7 jmp * $f7e3   Routine INDCMP
ff7d: 4c 17 fa jmp * $fal7   Puntatore a PRIMM
ff80: 00      brk
ff81: 4c 00 c0 jmp * $c000   Puntatore a CINT
ff84: 4c 09 e1 jmp * $e109   Puntatore a IOINIT
ff87: 4c 93 e0 jmp * $e093   Puntatore a RAMTAS
ff8a: 4c 56 e0 jmp * $e056   Puntatore a RESTORE
ff8d: 4c 5b e0 jmp * $e05b   Puntatore a VECTOR
ff90: 4c 5c f7 jmp * $f75c   Puntatore a SETMSG
ff93: 4c d2 e4 jmp * $e4d2   Routine SECND
ff96: 4c e0 e4 jmp * $e4e0   Routine TKSA

```

|                |              |        |            |                                  |
|----------------|--------------|--------|------------|----------------------------------|
| ff99: 4c 63 f7 | jmp *        | \$f763 |            | Puntatore a MEMTOP               |
| ff9c: 4c 72 f7 | jmp *        | \$f772 |            | Puntatore a MEMBOT               |
| ff9f: 4c 12 c0 | jmp *        | \$c012 |            | Puntatore a KEY                  |
| ffa2: 4c 5f f7 | jmp *        | \$f75f |            | Puntatore a SETTMO               |
| ffa5: 4c 3e e4 | jmp *        | \$e43e |            | Puntatore a ACPTR                |
| ffa8: 4c 03 e5 | jmp *        | \$e503 |            | Puntatore a CIOUT                |
| ffab: 4c 15 e5 | jmp *        | \$e515 |            | Routine UNTLK                    |
| ffae: 4c 26 e5 | jmp *        | \$e526 |            | Routine UNLSN                    |
| ffb1: 4c 3e e3 | jmp *        | \$e33e |            | Routine LISTN                    |
| ffb4: 4c 3b e3 | jmp *        | \$e33b |            | Routine TALK                     |
| ffb7: 4c 44 f7 | jmp *        | \$f744 |            | Routine READST                   |
| ffba: 4c 38 f7 | jmp *        | \$f738 |            | Routine SETLFS                   |
| ffbd: 4c 31 f7 | jmp *        | \$f731 |            | Routine SETNAM                   |
| ffc0: 6c 1a 03 | jmp (\$031a) |        |            | Vettore a routine OPEN (\$EFBD)  |
| ffc3: 6c 1c 03 | jmp (\$031c) |        |            | Vettore a routine CLOSE (\$F188) |
| ffc6: 6c 1e 03 | jmp (\$031e) |        |            | Vettore a routine CHKIN (\$F106) |
| ffc9: 6c 20 03 | jmp (\$0320) |        |            | Vettore a routine CKOUT (\$F14C) |
| ffcc: 6c 22 03 | jmp (\$0322) |        |            | Vettore a routine CLRCH (\$F226) |
| ffcf: 6c 24 03 | jmp (\$0324) |        |            | Vettore a routine BASIN (\$EF06) |
| ffd2: 6c 26 03 | jmp (\$0326) |        |            | Vettore a routine BSOUT (\$EF79) |
| ffd5: 4c 65 f2 | jmp *        | \$f265 |            | Routine LOADSP                   |
| ffd8: 4c 3e f5 | jmp *        | \$f53e |            | Routine SAVESP                   |
| ffdb: 4c 65 f6 | jmp *        | \$f665 |            | Puntatore a SETTIM               |
| ffde: 4c 5e f6 | jmp *        | \$f65e |            | Puntatore a RDTIM                |
| ffel: 6c 28 03 | jmp (\$0328) |        |            | Vettore a routine STOP (\$F66E)  |
| ffe4: 6c 2a 03 | jmp (\$032a) |        |            | Vettore a routine GETIN (\$EEEE) |
| ffe7: 6c 2c 03 | jmp (\$032c) |        |            | Vettore a routine CLALL (\$F222) |
| ffea: 4c f8 f5 | jmp *        | \$f5f8 |            | Routine UDTIM                    |
| ffed: 4c 0f c0 | jmp *        | \$c00f |            | Puntatore a SCRORG               |
| fff0: 4c 18 c0 | jmp *        | \$c018 |            | Puntatore a PLOT                 |
| fff3: 4c 81 f7 | jmp *        | \$f781 |            | Puntatore a IOBASE               |
| fff6: ff ff 24 | xxx          | \$24ff |            | Vettore modo 128                 |
| fff9: e2 05    | xxx          | \$05   | [ s-reg ]  | Vettore NMI                      |
| fffb: ff 3d ff | xxx *        | \$ff3d |            | Vettore di reset                 |
| fffe: 17 ff    | xxx          | \$ff   | [ lofbuf ] | Vettore di IRQ                   |

-----

## **LA PAGINA ZERO**

L'indirizzamento assoluto e' espresso in termini di byte di ordine alto e ordine basso. Il byte alto e' chiamato anche di indirizzamento perche' si riferisce all'indirizzamento di una pagina di memoria. Per esempio l'indirizzo \$1365 e' in pagina \$16 mentre l'indirizzamento di \$0345 e' in pagina \$03.

C'e' tuttavia un modo di indirizzamento chiamato INDIRIZZAMENTO IN PAGINA ZERO in cui il byte alto e' sempre di 00 perche' indirizza appunto in pagina 0.

Cio' consente un tempo operativo del microprocessore molto piu' veloce ed un'occupazione di memoria inferiore proprio perche' adopera solo un byte invece di 2.

La pagina zero, che va da \$0000 a \$00FF e' impiegata dal Sistema Operativo nella maniera piu' completa possibile per le variabili di sistema. Tuttavia per il C128 non e' sufficiente a contenere le informazioni necessarie e percio' si e' ritenuto necessario commentare anche altre pagine.

La pagina 0 (e seguenti) offre molte possibilita' alle eventuali manipolazioni del programmatore, anche perche' e' strettamente collegata alle corrispondenti routines KERNAL.

Oltre agli indirizzi esa, decimale ed alla spiegazione e' riportato anche la LABEL che evidenzia il collegamento con lo stesso S.O. visto nelle pagine precedenti.



| <u>Indirizzi</u> | <u>Label</u> |  |
|------------------|--------------|--|
| 0000: 0000       | d6510        | 6510 direzione dati                        |
| 0001: 0001       | r6510        | 6510 registro di uscita                    |
| 0002: 0002       | bank         | Locazione per numero del banco             |
| 0003: 0003       | pc-hi        | Loc. di salv. Program Counter alto         |
| 0004: 0004       | pc-lo        | ' Program Counter byte basso               |
| 0005: 0005       | s-reg        | ' Registro di Stato                        |
| 0006: 0006       | a-reg        | ' Accumulatore                             |
| 0007: 0007       | x-reg        | ' Registro X                               |
| 0008: 0008       | y-reg        | ' Registro Y                               |
| 0009: 0009       | charac       | ' Puntatore dello Stack                    |
| 000a: 0010       | endchr       | Flag di ricerca apici a fine stringa       |
| 000b: 0011       | trmpos       | Colonna video dopo ultimo TAB              |
| 000c: 0012       | verchk       | Flag disco: 0=LOAD, 1=VERIFY               |
| 000d: 0013       | count        | Puntatore Buffer Ingresso: Num.indice      |
| 000e: 0014       | domask       | Segno di tangente                          |
| 000f: 0015       | valtyp       | Flag categ. dati: \$FF=STRINGA \$0=NUM.    |
| 0010: 0016       | intflag      | ' :\$80=INTERO \$0=virgola mob             |
| 0011: 0017       | garbfl       | ' per GARBAGE COLLECTION                   |
| 0012: 0018       | subflg       | Puntatore per funzione utente (FN)         |
| 0013: 0019       | inpflg       | Fl. input: \$00=INPUT: \$40=get: \$98=read |
| 0014: 0020       | domask       | Flag per segno TAN/confronto risult.       |
| 0015: 0021       | channl       | Flag per periferiche attive in input       |
| 0016: 0022-0023  | linnum       | Numero di linea, valore intero .           |
| 0018: 0024       | tempot       | Punt. tempor. allo stack di stringa        |
| 0019: 0025-0026  | lastpt       | Indirizzo dell'ultima stringa              |
| 001b: 0027-0029  | tempst       | 3 bytes per stringhe temporanee.           |
| 001e: 0030-0032  | tempst+3     | 3 bytes per stringhe temporanee            |
| 0021: 0033-0035  | tempst+6     | 3 bytes per stringhe temporanee            |
| 0024: 0036-0037  | hindex1      | Puntatore per help 1                       |
| 0026: 0038-0039  | hindex2      | Puntatore per help 2                       |
| 0028: 0040-0044  | resho        | Ris.in virgola mobile di una multip.       |
| 002d: 0045-0046  | txttab       | Puntatore inizio BASIC                     |
| 002f: 0047-0048  | varbab       | Puntatore inizio VARIABILI                 |
| 0031: 0049-0050  | arytab       | Puntatore inizio MATRICI                   |
| 0033: 0051-0052  | strend       | Puntatore fine MATRICI +1                  |
| 0035: 0053-0054  | fretop       | Puntatore inizio STRINGHE                  |
| 0037: 0055-0056  | frespc       | Punt. di aiuto per immagaz. stringhe       |
| 0039: 0057-0058  | max-mem      | Puntatore fine memoria, Var.Banco 1        |
| 003b: 0059-0060  | curlin       | Numero della linea BASIC attuale.          |
| 003d: 0061-0062  | txtptr       | Punt. testo BASIC per CHRGET, CHRGOT       |
| 003f: 0063-0064  | form         | Punt. per PRINT USING.                     |
| 0041: 0065-0066  | datlin       | Numero di line del DATA attuale.           |
| 0043: 0067-0068  | dataptr      | Punt. dell'indiriz. del DATA attuale       |
| 0045: 0069-0070  | inpptr       | Vettore per la routine di INPUT.           |
| 0047: 0071-0072  | varnam       | Nome dell'attuale variabile                |
| 0049: 0073-0074  | fdecpt       | Punt. all'indir. dell'attuale variab       |
| 004b: 0075-0076  | andmsk       | Mask per AND/punt.LIST/punt FOR next       |
| 004d: 0077-0078  | vartxt       | Immagaz. temp. per punt. programma         |
| 004f: 0079       | oppmask      | Mask per comparazione >:2,=:4,<:8          |
| 0050: 0080-0081  | grbpnt       | Punt. var per funz. FN                     |
| 0052: 0082-0084  | dscpnt       | Punt. descr. var LIST-string compar.       |
| 0055: 0085       | helper       | Flag aiuto: \$xx=HELP, \$xx=LIST           |
| 0056: 0086-0087  | jmptr        | Vettore di salto per funzione calc.        |
| 0058: 0088       | oldov        | ***  |
| 0059: 0089       | tempfl       | Area per operazioni INSTRING               |
| 005a: 0090-0091  | arypnt       | Punt. trasf. blocchi, inizial. DIM         |

| <u>Indirizzi</u> | <u>Label</u> |                                      |
|------------------|--------------|--------------------------------------|
| 005c: 0092-0093  | hightr       | Punt. trasferimento blocchi          |
| 005e: 0094       | tempf2       | Punt. temp 2 per operaz. occas.inVM  |
| 005f: 0095-0096  | pdec         | N.pos. dec. per conversione          |
| 0061: 0097       | lowtr        | Punt. dec per lettura numeri         |
| 0062: 0098       | expsgn       | Esponente del segno                  |
| 0063: 0099       | fac          | FAC accumulatore 1 : esponente       |
| 0064: 0100-0103  | rightflag    | FAC accumulatore 1 : mantissa        |
| 0068: 0104       | facsgn       | FAC ' 1 : segno                      |
| 0069: 0105       | degree       | FAC valutazione polinomi             |
| 006a: 0106       | argexp       | FAC 2 esponente                      |
| 006b: 0107-0110  | argho        | FAC 2 mantissa                       |
| 006f: 0111       | argsgn       | FAC 2 segno                          |
| 0070: 0112       | strngl       | Flag risult. tra acc 1 e acc 2       |
| 0071: 0113       | facov        | Virgola mobile accumulatore 1        |
| 0072: 0114-0115  | strng2       | Puntatore al buffer della cassetta   |
| 0074: 0116-0117  | autinc       | Valore di offset per AUTO \$00=off   |
| 0076: 0118       | mvdf1g       | Flag per Hires:1=Iniz. Basic +10K.   |
| 0077: 0119       | noze         | Contatore n. sprites                 |
| 0078: 0120       | hulp         | Contatore di aiuto                   |
| 0079: 0121       | syntmp       | Locaz. temp. per lettura indiretta.  |
| 007a: 0122-0124  | dsdec        | Descrizione della Variab. DS\$       |
| 007d: 0125-0126  | tos          | Fine dello stack durante il run.     |
| 007f: 0127       | runmod       | Flag :modo RUNo DIRETTO              |
| 0080: 0128       | parsts       | DOS parser status word               |
| 0081: 0129       | parstx       | ***                                  |
| 0082: 0130       | oldstk       | ***                                  |
| 0083: 0131       | colsel       | Colore corrente in modo grafico      |
| 0084: 0132       | multicolor1  | Modo multicolore : colore 1          |
| 0085: 0133       | multicolor2  | Modo multicolore: colore 2           |
| 0086: 0134       | foreground   | Colore dello sfondo                  |
| 0087: 0135-0136  | scalex       | Fattore di scala nella direz. X      |
| 0089: 0137-0138  | scaley       | Fattore di scala nella direz. Y      |
| 008b: 0139       | stopnb       | Termina Disegno se diverso colore.   |
| 008c: 0140-0141  | grapnt       | Punt. di indirizzo per routine graf. |
| 008e: 0142       | vtempl       | Loc. temp. per rout. grafiche.       |
| 008f: 0143       | vtemp2       | loc. temp. per routine grafiche      |
| 0090: 0144       | status       | Byte di stato per operazioni di I/O  |
| 0091: 0145       | stkey        | Flag di stop :Tasto STOP,tasto RVS   |
| 0092: 0146       | svxt         | Costante di tempo per cassetta.      |
| 0093: 0147       | verck        | Flag di load: \$00=LOAD,\$01=VERIFY  |
| 0094: 0148       | c3p0         | Flag seriale carattere nel buffer    |
| 0095: 0149       | bsour        | Carattere nel buffer per bus seriale |
| 0096: 0150       | syno         | Sincroniz. per cassetta              |
| 0097: 0151       | xsav         | Indirizzo temporaneo dati            |
| 0098: 0152       | ldtnd        | Indice per file :num. di files aper. |
| 0099: 0153       | df1for       | Standard per ingresso dati:(0=tast)  |
| 009a: 0154       | df1to        | Standard uscita dati:(3=schermo)     |
| 009b: 0155       | prty         | carattere di parita del nastro       |
| 009c: 0156       | dpsw         | Flag: ricevuto byte dal nastro       |
| 009d: 0157       | msgflg       | Flag di messaggio dal Kernel         |
| 009e: 0158       | ptr1         | Errore logico da nastro:passo 1      |
| 009f: 0159       | ptr2         | Errore logico da nastro: passo 2     |
| 00a0: 0160-0162  | time         | Orologio in tempo reale da 24 ore.   |
| 00a3: 0163-0164  | pcntr        | Locaz. temporanea per bua seriale    |
| 00a5: 0165       | cntdn        | Contat. per bit nastro in Save       |
| 00a6: 0166       | bufpt        | Puntatore al buffer cassetta         |

| <u>Indirizzi</u> | <u>Label</u> |                                      |
|------------------|--------------|--------------------------------------|
| 00a7: 0167       | shcnl        | RS232 bits di ingresso               |
| 00a8: 0168       | bitci        | RS232 contatore bit in ingresso      |
| 00a9: 0169       | rinone       | RS232 flag di inizio per bit         |
| 00aa: 0170       | rdflg        | RS232 byte del buffer ingresso dati  |
| 00ab: 0171       | riprty       | Rs232 parita di ingresso.            |
| 00ac: 0172-0173  | sal          | Punt. scroll schermo/buff.cassetta   |
| 00ae: 0174-0175  | eal          | Punt. fine prog./fine cassetta.      |
| 00b0: 0176-0177  | cmp0         | Costante di tempo per cassetta       |
| 00b2: 0178-0179  | tapel        | Punt. inizio del buffer cassetta     |
| 00b4: 0180       | bitts        | RS232 prossimo bit per scroll        |
| 00b5: 0181       | diff         | RS232 prossimo bit da inviare        |
| 00b6: 0182       | prp          | RS232 byte del buffer da inviare     |
| 00b7: 0183       | fnlen        | Lunghezza del nome file corrente     |
| 00b8: 0184       | la           | Numero del file (LFN)                |
| 00b9: 0185       | sa           | Indirizzo secondario attuale         |
| 00ba: 0186       | fa           | numero della periferica attuale      |
| 00bb: 0187-0188  | fnadr        | Punt. indirizzo del nome file corre. |
| 00bd: 0189       | roprty       | Rs232 parita di uscita               |
| 00be: 0190       | fsblk        | Num. deirimanenti blocchi di R/W     |
| 00bf: 0191       | drive        | Buffer per porta seriale             |
| 00c0: 0192       | casl         | Flag. motore della cassetta          |
| 00c1: 0193       | track        | Indirizzo inizio in/out traccia      |
| 00c2: 0194       | stah         | Indirizzo inizio in/out settore      |
| 00c3: 0195-0196  | memuss       | Temp. di Load per cassetta           |
| 00c5: 0197       | data         | Cassetta: read/write data            |
| 00c6: 0198       | ba           | Num. banco per LOAD/SAVE/VERIFY      |
| 00c7: 0199       | fnbank       | Num. del banco del nome file attua.  |
| 00c8: 0200-0201  | ribuf        | RS232 puntatore buffer ingresso      |
| 00ca: 0202-0203  | robuf        | RS232 puntatore buffer uscita        |
| 00cc: 0204-0205  | keytab       | Punt. decodifica tavola tastiera.    |
| 00ce: 0206-0207  | imparm       | Punt. posizione della stringa        |
| 00d0: 0208       | ndx          | indice alla coda del buffer tastiera |
| 00d1: 0209       | kyndx        | Flag chiamata tasti funzione         |
| 00d2: 0210       | keyidx       | Indice stringa tasto funzione        |
| 00d3: 0211       | shflag       | Flag shift:Shift=1,C=2,Ctrl=4,old=8  |
| 00d4: 0212       | sfdx         | Flag per tasto premuto               |
| 00d5: 0213       | ltsx         | Flag del tasto attuale:0=nessuno     |
| 00d6: 0214       | crsm         | Flag per INPUT o GET                 |
| 00d7: 0215       | mode         | Flag per 40/80 colonne               |
| 00d8: 0216       | graphm       | Flag per schermo testo o grafico     |
| 00d9: 0217       | charen       | Punt. al set di caratteri            |
| 00da: 0218       | sedsal       | Puntatore per MOVLIN (basso)         |
| 00db: 0219       | keylen       | Puntatore per MOVLIN (alto)          |
| 00dc: 0220       | keynum       | Numero del tasto funzione.           |
| 00dd: 0221       | keynxt       | Lunghezza stringa attuale tasto funz |
| 00de: 0222       | keybnk       | Banco per tasto funzione             |
| 00df: 0223       | keytmp       | Lunghezza stringa tasto funzione 1   |
| 00e0: 0224-0225  | pnt          | Punt. alla linea di schermo (testo)  |
| 00e2: 0226-0227  | user         | Punt. linea di schermo(attribute)    |
| 00e4: 0228       | scbot        | Bordo inferiore della finestra       |
| 00e5: 0229       | sctop        | Bordo superiore della finestra sche. |
| 00e6: 0230       | scfl         | Bordo sinistro della finestra        |
| 00e7: 0231       | scrt         | Boedo destro della finestra          |
| 00e8: 0232       | lsexp        | Colonna dell'attuale input: inizio   |
| 00e9: 0233       | lstp         | Linea dell'attuale input : inizio    |
| 00ea: 0234       | indx         | Linea dell'attuale input : fine      |

| <u>Indirizzi</u> | <u>Label</u> |                                      |
|------------------|--------------|--------------------------------------|
| 00eb: 0235       | tblx         | linea attuale del cursore            |
| 00ec: 0236       | pntr         | Colonna attuale del cursore          |
| 00ed: 0237       | lines        | Numero massimo delle linee schermo   |
| 00ee: 0238       | columns      | Numero massimo di colonne schermo    |
| 00ef: 0239       | datax        | Attuale carattere da visualizzare    |
| 00f0: 0240       | lstchr       | Precedente carattere stampato        |
| 00f1: 0241       | color        | Colore sotto il cursore              |
| 00f2: 0242       | tcolor       | Salvataggio colore per INSERT/DELETE |
| 00f3: 0243       | rsv          | Flag attivazione modo reverse        |
| 00f4: 0244       | qtsw         | Flag attivazione virgolette          |
| 00f5: 0245       | insrt        | Flag attivazione modo INSERT         |
| 00f6: 0246       | insflg       | Flag attivazione modo auto insert    |
| 00f7: 0247       | locks        | Disabilitazione (c=)(shift) ctrl-s   |
| 00f9: 0248       | beeper       | Disab. scrool schermo                |
| 00fa: 0249-0254  | frekzp       | Area libera per l'utente             |
| 00ff: 0255       | lofbuf       |                                      |

-----

## PAGINA UNO

```

0100 0256 - 0271 Area di 16 Byte per i nomi dei dati
0110 0272 Contatore ciclo per DOS
0111 0273 Lunghezza primo nome file per DOS
0112 0274 Numero periferica per DOS, prima unita' dischi
0113 0275 - 0276 Indirizzo DOS, primo nome file LO/HI
0115 0277 Lunghezza secondo nome file per DOS
0116 0278 Numero periferica per DOS, seconda unita' dischi
0117 0279 - 0280 Indirizzo DOS, secondo nome file LO/HI
0119 0281 - 0282 Indirizzi partenza per BLOAD/BSAVE LO/HI
011b 0283 - 0284 Indirizzi di fine per BSAVE LO/HI
011d 0285 Indirizzi logici DOS
011e 0286 Indirizzi fisici DOS
011f 0287 Indirizzi secondari DOS
0120 0288 Lunghezza record sempre per DOS
0121 0289 Numero banco per DOS
0122 0290 - 0291 Zona di immagazz. per ID disco (2Bytes)
0124 0292 Flag per controllo ID
-----

```

La zona seguente di memoria e' riservata per PRINT USING

```

0125 0293 Puntatore a numero inizio
0126 0294 Puntatore al numero di fine
0127 0295 Flag per dollaro ($)
0128 0296 Flag per virgola (,)
0129 0297 Contatore del numero di comandi
012a 0298 Segno dell' esponente
012b 0299 Puntatore dell' esponente
012c 0300 Contatore per n.posizioni intere
012d 0301 Flag per allineamento dopo il punto
012e 0302 Contatore posizione campi prima del punto
012f 0303 Come sopra ma PRIMA del punto decimale
0130 0304 Flag per segno (+/-)
0131 0305 Flag per campo esponente
0132 0306 Interruttore per comando
0133 0307 Contatore per caratteri nel campo
0134 0308 Numero dei segni
0135 0309 Flag per spazio o asterisco
0136 0310 Puntatore per inizio campo
0137 0311 Puntatore per lunghezza formato
0138 0312 Puntatore per fine campo
-----

```

```

0139 0313 - 0510 Stack di sistema punto di fine
01ff 0511 Stack di sistema punto iniziale
0200 0512 Buffer ingresso monitor e BASIC
02a2 0674 - 0686 Routine FETCH
02af 0687 - 0701 Routine STASH
02be 0702 - 0716 Routine CMPARE
02cd 0717 - 0738 Routine JSRFAR
02e3 0739 Routine JMPFAR
02fc Disabilita tutti gli interrupt
02fd Salta al vettore funzione cartridge
-----

```

## TAVOLA DI VETTORI

|       |             |        |  |
|-------|-------------|--------|--|
| 0300  | 0768        | \$4d3f | Routine di errore (per errore X)   |
| 0302  | 0770        | \$4dc6 | Lettura esecuzione di linea Basic  |
| 0304  | 0772        | \$430d | Conversione codice interprete  |
| 0306  | 0774        | \$5151 | Conversione in codici testo  |
| 0308  | 0776        | \$4aa2 | Esecuzione parola chiave   |
| 030a  | 0778        | \$78da | Valutazione di espressioni   |
| 030c  | 0780        | \$4321 | Conversione routine di escape  |
| 030e  | 0782        | \$51cd | Lista di escape  |
| 0310  | 0784        | \$4ba9 | Esecuzione dell'escape   |
| 0312  | 0786        | \$ffff | Vettore di interruzione TIME   |
| 0314  | 0788        | \$fa65 | Per routine IRQ  |
| 0316  | 0790        | \$b003 | Ingresso monitor   |
| 0318  | 0792        | \$fa40 | Routine NMI  |
| 031a  | 0794        | \$efbd | Routine OPEN   |
| 031c  | 0796        | \$f188 | Routine CLOSE  |
| 031e  | 0798        | \$f106 | Routine CHKIN  |
| 0320  | 0800        | \$f14c | Routine CKOUT  |
| 0322  | 0802        | \$f226 | Routine CLRCH  |
| 0324  | 0804        | \$ef06 | Routine BASIN  |
| 0326  | 0806        | \$ef79 | Routine BSOUT  |
| 0328  | 0808        | \$f66e | Routine STOP   |
| 032a  | 0810        | \$eeeb | Routine GETIN  |
| 032c  | 0812        | \$f222 | Routine CLALL  |
| 032e  | 0814        | \$b006 | Ingresso EXMON   |
| 0330  | 0816        | \$f2sc | Routine LOAD   |
| 0332  | 0818        | \$f54e | Routine SAVE   |
| 0334  | 0820        | \$c7b9 | Uscita carattere con CTRL  |
| 0336  | 0822        | \$c805 | Uscita carattere con SHIFT   |
| 0338  | 0824        | \$c9c1 | Uscita carattere con ESCAPE  |
| 033a  | 0826        | \$c5e1 | Lettura di tastiera  |
| 033c  | 0828        | \$c6ad | Immagazzinamento di tastiera   |
| 033e  | 0830        | \$fa80 | Prima tavola di decodifica tastiera  |
| 0340  | 0832        | \$fad9 | Seconda " " "  |
| 0342  | 0834        | \$fb32 | Terza " " "  |
| 0344  | 0836        | \$fb8b | Quarta " " "   |
| 0346  | 0838        | \$fa80 | Quinta " " "   |
| 0348  | 0840        | \$fbe4 | Sesta " " "  |
| 034a  | 0842 - 0851 |        | Buffer IRQ di tastiera   |
| 0354  | 0852 - 0861 |        | Tavola di bit map per tabulatori   |
| 035e  | 0862 - 0865 |        | Tavola di bit map per overflow di linea  |
| 0362  | 0866 - 0875 |        | Tavola dei numeri di files logici  |
| 036c  | 0876 - 0885 |        | Tavola indirizzi periferiche   |
| 0376  | 0886 - 0895 |        | Tavola indirizzi secondari   |
| ----- |             |        |  |
| 0380  | 0896 - 0901 |        | Routine CHRGET   |
| 0386  | 0902 - 0911 |        | Routine CHRGET   |
| 0390  | 0912 - 0926 |        | Routine QNUM   |
|       |             |        |  |
| 039f  | 0927 - 0938 |        | Caricamento da un banco via PCRA e PCRC. Questa zona, come le successive ha il suo contenuto originale in ROM. L'indirizzo e' \$4294 |
|       |             |        |  |
| 03ab  | 0939 - 0950 |        | Come sopra ma via PCRB e PCRD<br>Indirizzo ROM \$42a4  |

03b7 0951 - 0959 Caricamento da un banco via PCRA e PCRC dell' indirizzo  
dato da pag.0 indice 1  
Indirizzo ROM \$42b0

03c0 0960 - 0968 Come sopra ma per indice 2  
Indirizzo ROM \$42b9

03c9 0969 - 0977 Come sopra ma via PCRA e PCRC sempre da pag.0. Il puntatore  
e' della routine CHRGET.  
Indirizzo ROM \$42c2

03d2 0978 - 0980 Costanti numeriche per il Basic (da ROM)

03d5 0981 Zona per SYS,POKE,PEEK

03d6 0982 - 0985 Immagazz. temporaneo per INSTRING

03da 0986 Puntatori per stringhe e convers. numeri

03db 0987 - 0990 Zona di immagazz. per operazioni SSHAPE

03df 0991 Segnalatore di overflow per MARK1

03e0 0992 Immag. temporaneo per controllo Sprite 1

03e1 0993 Come sopra per Sprite n. 2

03e2 0994 - 0995 Zona colori

03e4 0996 - 1007 Area libera

03f0 1008 - 1020 Routine DMA

03fd 1021 - 1023 Area libera

0400 1024 - 2047 Immagazzinamento di schermo

0800 2048 - 2559 512 Bytes per immagazzinamento Basic

## **SUCCESSIVE LOCAZIONI**

Pur rivestendo una certa importanza le locazioni di memoria successive all'indirizzo decimale 2559 (\$09ff) e le locazioni che contengono il Basic non rientrano nello scopo di questo volume. Tuttavia ci sono delle locazioni di cui molti lettori non disconosceranno l'utilita'. Vediamole con un breve commento.

### GESTIONE DELL' RS 232

|       |           |                          |
|-------|-----------|--------------------------|
| 0a0f: | 2575      | Registro di stato NMI    |
| 0a10: | 2576      | Registro di controllo    |
| 0a11: | 2577      | Registro di comando      |
| 0a12: | 2578-2579 | Per baud rate            |
| 0a14: | 2580      | Registro di stato        |
| 0a15: | 2581      | N. bits da inviare       |
| 0a16: | 2582-2583 | Per baud rate(full bit)  |
| 0a18: | 2584      | Ind.inizio BUFFER input  |
| 0a19: | 2585      | Idem per fine            |
| 0a1a: | 2586      | Ind.inizio BUFFER output |
| 0a1b: | 2587      | Idem per fine            |

### BUFFER DI RS-232

Sono riservate due aree di cui una per i dati in ingresso e l'altra per quelli in uscita.

0c00-0cff    3072-3327    Buffer di input per RS-232



0d00-0dff    3328-3583    Buffer di output per RS-232

#### VARIABILI DI SCHERMO

Da \$0a40 a \$0a59 e' un' area per l' immagazzinamento delle variabili di schermo. Corrisponde ai contenuti di pagina zero a partire dall' indirizzo \$00e0.

#### BUFFER DI CASSETTA

E' riservata un' area da \$0b00 a \$0bff (decimale 2816-3071).

#### GRAFICA

E' riservata un' area per la definizione degli Sprites da \$0e00 a \$0eff (3584-4095)

Altra area per la grafica e' da \$1100 a \$1221. Di particolare importanza e' che all' interno di questa area alcune posizioni possono assumere valori completamente differenti e differenti significati.

#### MUSICA

All' immagazzinamento per i puntatori musicali e' riservata un' area da \$1222 (4642) a \$1274 (4725).  
Le variabili del SID saranno invece da \$1281 (4737) a \$12ff (4863).

### INTERRUPT

Vediamo in dettaglio l' area per l' immagazzinamento dei puntatori di interrupt:

1276: 4726-4728 Immagazzinamento generale  
1279: 4729-4731 Immag. indirizzi LO  
127c: 4732-4734 Idem per HI

Si tratta come si puo' notare agevolmente di gruppi di 3 Bytes.

127f: 4735            INTVAL  
1280: 4736            COLTYP

### ALTRE ZONE

1000: 4096-4105 Area per i tasti di funzione programmabile. Tavola delle lunghezze.

100a: 4106-4351 Come sopra, ma per utilizzata per le funzioni stringa.

1300: 4864-6143 RAM non utilizzata

1800: 6144-7167 Area riservata per applicazioni dei tasti funzione

1c00: 7167-8191 Area per la matrice video numero 2

2000: 8192-16383 Area per VIC bit map

## **LE ROUTINES KERNAL**

Non vogliamo qui tenere una lezione sul Linguaggio Macchina per il quale si rimanda all'apposito manuale scritto per il C64 ma che potra' rivelarsi utilissimo anche in questo caso, tuttavia le routines Kernal, anche se presentate nel corso delle pagine precedenti completamente disassemblate, meritano, per la loro estrema importanza un trattamento particolare.

Per questo motivo le ripresentiamo per esteso cercando di ampliarne le descrizioni.

La conoscenza degli indirizzi e dei significati di queste routine che si riferiscono essenzialmente al controllo dei dati e delle informazioni in ingresso ed in uscita permette infatti, oltre che una migliore risposta in termini di programmazione e di efficienza complessiva, anche la possibilita' di effettuare variazioni al sistema stesso per manipolazioni di dati.

Cio' che consentira' di essere preparati ANCHE a tutte le modifiche che il costruttore o l'utente vorra' apportare.

Anche per chi opera con scioltezza in Linguaggio Macchina e' molto piu' semplice limitarsi a delle modifiche che non il dover riscrivere delle routines e poi provarle.

Di norma a queste routines si accede attravrso un' istruzione di JSR. Vediamo come si comportano in linea generale.

- 1) All'accensione per prima cosa viene

resettato lo Stack Pointer ed azzerato il modo decimale.

#### NOTA

Vedi di fare attenzione a quanto verra' detto nella parte relativa allo Z-80 per tutta la serie di operazioni compiute a questo proposito. Qui vogliamo solo dare un' idea dello schema generale.

2) Per seconda cosa viene controllata la presenza di un eventuale ROM dotata di AUTOSTART sioe' di partenza automatica. Se viene incontrata la routine di autostart sul cartridge, ed e' bene ricordare che non tutti i cartridges sono dotati di questa routine, allora il controllo viene trasferito al programma presente nella ROM stessa, mentre in caso contrario viene eseguita, o meglio proseguita, la procedura di inizializzazione.

Ricordiamo che di solito con le informazioni contenute nel cartridge viene modificata solo in piccola parte il Sistema Operativo.

3) Successivamente le routines Kernal provvedono ad inizializzare tutte le periferiche di Input ed Output come pure il bus seriale.

4) Viene eseguito un controllo della RAM fissando i puntatori di inizio e fine memoria.

Viene quindi inizializzata la pagina ZERO e

UNO della memoria con gli opportuni valori e fissata la zona del buffer di cassetta.

5) Vengono eseguite altre funzioni fra le quali quelle che ci interessano particolarmente sono l' impostazione ai valori normali di procedimento, in altre parole la loro inizializzazione sia dei valori delle tavole che dei valori dei salti indiretti.

E' importante notare che qui sono riportate solo le KERNAL del computer COMMODORE C128, mentre per quelle relative al COMMODORE C64 si rimanda ai volumi:

IL SISTEMA OPERATIVO DEL C 64

GUIDA AL COMMODORE 64

## **DESCRIZIONE DELLE ROUTINES KERNAL**

**NOME            : C64MODE**

**FUNZIONE    :** inserisce il modo C64

**INDIRIZZO:** \$FF4D - 65357

**DESCRIZIONE :** Saltando a questo indirizzo, cioè chiamando in funzione questa routine si passa dal modo 128, che è quello di DEFAULT al modo 64. La frequenza viene ridotta ad 1 MHz e la MMU chiude tutti i registri di accesso a quel modo in modo tale che non si possa tornare indietro. Non ci sono parametri di ingresso o di uscita perché non esiste possibilità di ritornare indietro.

**NOME            : DMA-CALL**

**FUNZIONE    :** inizializzazione della RAM esterna

**INDIRIZZO:** \$FF50 - 65360

**DESCRIZIONE:** Per avere un accesso diretto alla memoria verso un'espansione RAM esterna è necessario per prima cosa chiamare in funzione questa routine.

Nel registro X deve essere indicata la nuova configurazione del sistema per la gestione completa della memoria.

NOME            : **BOOTCALL**

FUNZIONE : esegue il BOOT da disco

INDIRIZZO: \$FF53 - 65363

DESCRIZIONE : Facendo entrare in funzione questa routine si esegue il BOOT del disco. Si carica l' indirizzo che e' nel drive. Accade lo stesso come quando si accende l' apparecchio.

Se la routine non trova un BOOTFILE allora il controllo viene restituito all' unita' centrale.

Nel registro X e' inserito l' indirizzo della periferica collegata.

NOME            : **PHOENIX**

FUNZIONE : partenza a freddo

INDIRIZZO: \$FF56 - 65366

DESCRIZIONE: Partenza a freddo del modo 128. Se viene rilevata la presenza di un cartridge di espansione nella relativa porta allora questa assume il controllo. Dovrebbe esserci l' AUTOSTART. In caso contrario il controllo passa all' unita' a disco. I valori assegnati ai tabulatori, ai tasti funzione ecc. vengono resettati.

NOME            : **LKUPLA**

FUNZIONE : Ricerca il FILENUMBER

INDIRIZZO: \$FF59 - 65369

DESCRIZIONE: Questa routine effettua la ricerca dei parametri di un file basandosi sull' indirizzo logico immagazzinato nell'

Accumulatore. LKUPLA esegue iun CLEAR della variabile di STATUS ed in rapporto ai risultati che ottiene dall' esame del registro restituisce un errore se non si trova un indirizzo logico (LA) nell' accumulatore. Oppure immettera' il primo indirizzo nel registro X e l' indirizzo secondario nel registro Y. Deve essere chiamata prima di un OPEN con un JSR.

NOME            : **LKUPSA**

FUNZIONE : Ricerca i parametri di un file

INDIRIZZO: \$FF5C - 65372

DESCRIZIONE: Ricerca i parametri di un file basandosi sul valore dell' indirizzo secondario immagazzinato nel registro Y. Per il resto e' abbastanza simile come comportamento alla routine precedente.

NOME            : **SWAPPER**

FUNZIONE : passa da 40 a 80 colonne

INDIRIZZO: \$FF5F - 65375

DESCRIZIONE Questa routine inverte il modo 40/80 colonne. Inoltre modifica le informazioni in Pagina zero per lo schermo attivo e lo schermo passivo. La memoria che va E0 fino a FA viene scambiata con la memoria da 0A40 fino a 0A5A. Non e' richiesto nessun parametro di input.

NOME            : **DLCHR**

FUNZIONE : copia il CHARRROM



INDIRIZZO: \$FF62 - 65378

DESCRIZIONE Attivando il tasto ASCII-DIN il set di caratteri viene di nuovo copiato nel VDC-RAM perche' il controllo delle 80 colonne prende le informazioni per i caratteri non da ROM. Puo' essere utili nella grafica perche' qui il set dei caratteri che si trova nel VDC RAM viene sovrascritto. Questa routine si copia nel VDC RAM il set di caratteri che e' stato selezionato con il tasto ASCII-DIN. Non sono necessari ne parametri di input ne' di OUTPUT.

NOME            : **PFKEY**

FUNZIONE : Ridefinizione dei tasti funzione

INDIRIZZO: \$FF65 - 65381

DESCRIZIONE Con questa routine si possono assegnare i valori ai tasti funzione. Nell' accumulatore si trova l' indirizzo della Pagina ZERO che fa da puntatore sul testo del TASTO FUNZIONE. Nel registro X si trova il numero del tasto funzione quindi da 1 a 12. Nel registro Y si trova la lunghezza della stringa. Si chiamera' in funzione questa routine che inserisce la stringa nella tabella.

NOME            : **SETBANK**

FUNZIONE : definisce il banco di memoria per le operazioni disco

INDIRIZZO: \$FF68 - 65384

DESCRIZIONE Questa routine deve essere

chiamata prima di ogni comando LOAD, SAVE e VERIFY ed anche prima di OPEN. L' indice di configurazione del nome del file viene consegnato nel Registro Y e l' indice di configurazione del banco di memoria su cui si lavora viene consegnato nell' Accumulatore. Il registro Y viene memorizzato nella pagina zero all' indirizzo \$C6 e l' Accumulatore in \$C7.

**NOME :GETCONF**

**FUNZIONE :** prende il Byte di configurazione

**INDIRIZZO:** \$FF6B - 65387

**DESCRIZIONE** Normalmente esiste una tabella di 16 Bytes di configurazione che e' sufficiente per la definizione delle varie configurazioni. Questa tabella si trova nell' indirizzo \$F7F0. La routine consegna al Registro X l' indice di configurazione e riceve nell' Accumulatore il Byte di configurazione che normalmente si scrive nel registro di configurazione che si trova a \$FF00

**NOME :JSRFAR**

**FUNZIONE :** salto in subroutine su qualsiasi BANK

**INDIRIZZO:** \$FF6E - 65390

**DESCRIZIONE** Questa routine ha lo scopo di poter saltare in un qualsiasi sottoprogramma in una qualsiasi configurazione. I parametri sono in Pagina Zero dagli indirizzi da \$02 fino a \$09. Al terminare della routine la vecchia configurazione viene ristabilita.

**NOME : JMPFAR**

**FUNZIONE :** salta in qualsiasi banco

**INDIRIZZO:** \$FF71 - 65393

**DESCRIZIONE** Anche con questa routine i parametri vengono consegnati alla pagina zero agli indirizzi da \$02 a \$09. Tuttavia JMPFAR non e' la chiamata di un sottoprogramma, ma solo un salto in qualsiasi BANK. JMPFAR quindi riunisce in se lo switch del byte di configurazione ed il salto. Poiche' qui non avviene un ritorno non ci saranno dei parametri che vengono restituiti.

**NOME : INDFET**

**FUNZIONE :** prende un Byte di qualsiasi Bank

**INDIRIZZO:** \$FF74 - 65396

**DESCRIZIONE** Questa routine che si trova soprattutto nella Pagina Zero da la possibilita' di leggere un qualsiasi indirizzo della memoria in qualsiasi configurazione senza dover cambiare in modo notevole la configurazione attuale. Per far cio' e' necessario per prima cosa definire in un indirizzo in Pagina Zero il puntatore alla memoria che si vuole leggere.

Nell' Accumulatore viene poi consegnato questo indirizzo della Pagina Zero. Nel registro X viene consegnato l' indice di configurazione e nel registro Y viene consegnato l'offset riguardo al puntatore della pagina zero.

NOME            : **INDSTA**

FUNZIONE    : memorizza l' Accumulatore in qualsiasi Bank.

INDIRIZZO: \$FF77 - 65399

DESCRIZIONE Come avviene nella Routine INDFET cosi questa routine carica nella memoria il contenuto dell' Accumulatore in qualsiasi configurazione della memoria. Quindi anche i parametri devono essere consegnati nell' Accumulatore, nel registro X e nel registro Y. Tuttavia il Byte che deve essere memorizzato finisce nell' Accumulatore. L' indirizzo di Pagina Zero nel quale viene memorizzato il puntatore deve essere definito nell' indirizzo \$02B9.

NOME            : **INDCMP**

FUNZIONE    : Confronta l' Accumulatore con la memoria di un Bank qualsiasi.

INDIRIZZO: \$FF7A - 65402

DESCRIZIONE Questa routine confronta l' Accumulatore con qualsiasi indirizzo di memoria in qualsiasi banco. Come avviene per la routine INDSTA anche qui e' necessario comunicare l' indirizzo del puntatore in Pagina Zero. Cio' viene fatto nell' indirizzo \$02C8. Nell' Accumulatore viene consegnato il Byte che deve essere confrontato, nel registro X viene consegnato l' indice di configurazione e nel registro Y l' OFFSET. Dopo che e' stata chiamata questa routine il risultato di questo confronto e cioe' lo STATUS BYTE del processor si trova nell' indirizzo \$05.

NOME            : **PRIMM**

FUNZIONE : inserisce un testo

INDIRIZZO: \$FF7D - 65402

DESCRIZIONE Questa routine e' molto comoda perche' e' molto facile da usare. Infatti non deve essere consegnato alcun parametro. Tutti i Bytes che seguono dopo aver chiamato questa routine vengono consegnati al device di output sul BSOUT.

Come segno che siamo arrivati alla fine viene usato un Byte zero. Il programa viene poi continuato subito dopo il Byte zero. L'unico svantaggio di questa routine e' che quando si va a disassemblare il programma questi diventa poco chiaro e confuso da leggere.

NOME            : **ACPTR**

FUNZIONE : riceve dati dal bus seriale

INDIRIZZO: \$FFA5 - 65445

DESCRIZIONE : Questa e' la routine che si usa quando si desidera ricevere informazioni da una periferica attraverso il BUS seriale, per esempio da disco. Questa routine riceve un Byte di dati dal Bus usando un HANDSHAKING pieno ed il dato e' riportato in Accumulatore. La routine TALK deve essere chiamata in funzione prima di ordinare alla periferica di inviare dati sul Bus.

Se la periferica in ingresso necessita di un comando secondario, questo deve essere inviato usando la routine TKSA prima di ACPTR. Se ci sono errori saranno riportati nella PAROLA di STATO (STATUS WORD) il cui contenuto potra' essere letto dalla routine

READST.

NOME            : **CHKIN**

FUNZIONE : apre un canale di input

INDIRIZZO: \$FFC6 - 65478

DESCRIZIONE: Un qualsiasi File logico che sia stato aperto per mezzo della routine OPEN puo' essere definito come un canale di Input per mezzo di questa routine. Naturalmente la periferica sul canale deve essere una periferica in input, perche' altrimenti avremo un errore e la CHKIN non avra' effetto. Se si stanno ricevendo dati da una qualsiasi altra parte che non sia la tastiera, questa routine (OPEN) deve essere chiamata prima di usare sia le routine CHRIN che GETIN per l' Input dei dati. Se si desidera usare l' Input da tastiera, e nessun altro canale di input e' aperto, allora la chiamata a questa Routine e alla routine OPEN non e' necessaria. Quando questa routine e' utilizzata con una periferica sul bus Seriale, essa invia automaticamente l' indirizzo di chiamata ( e l' indirizzo secondario se questo e' specificato in OPEN) sul Bus.

NOME            : **CHKOUT**

FUNZIONE : Apre un canale per OUTPUT

INDIRIZZO: \$FFC9 - 65481

DESCRIZIONE: Un qualsiasi numero di File logico che sia stato creato dalla routine OPEN puo' essere definito come un canale di

OUTPUT. Percio' la periferica deve essere una periferica in OUTPUT cioe' in uscita perche' in caso contrario avremo una segnalazione di errore. Questa routine (CHKOUT) deve essere messa in funzione prima che un qualsiasi dato sia inviato ad una periferica ( naturalmente in uscita) a meno che non si desideri usare lo schermo in funzione di periferica in uscita. Quando e' usata per aprire un canale per una periferica sul Bus seriale, questa nostra routine inviera' automaticamente l' indirizzo di LISTEN specificato dalla routine OPEN e l' indirizzo secondario se esiste.

NOME            : **CHRIN**

FUNZIONE : Riceve un carattere da un canale di Input

INDIRIZZO: \$FFCF - 65487

DESCRIZIONE: Questa routine riceve un byte di dati da un canale gia' selezionato per mezzo della routine CHKIN come canale in INPUT. Se CHKIN non e' stata usata per definire un diverso canale di input allora i dati saranno attesi da tastiera. Il Byte di dati e' caricato in accumulatore ed il canale resta aperto.

L' ingresso da tastiera e' manipolato in maniera particolare. Per prima cosa e' attivato il cursore che lampeggera' fino alla digitazione di un ritorno carrello da tastiera ( cioe' fino a quando non sia premuto il RETURN). Tutti i caratteri della linea ( max 88) sono immagazzinati nel BASIC INPUT BUFFER. Questi caratteri sono recuperati ad uno ad uno per mezzo di tanti

salta a questa routine      quanti sono questi caratteri.

Quando viene incontrato il ritorno carrello l'intera linea e' stata manipolata.

NOME            : **CHROUT**

FUNZIONE:    Uscita di un carattere

INDIRIZZO:   \$FFD2 - 65490

DESCRIZIONE: Questa routine fa uscire un carattere su un canale gia' aperto. E' necessario usare le routines OPEN e CHKOUT per fissare un canale di uscita prima di chiamare questa routine. Nel caso che queste chiamate siano omesse i dati saranno inviati alla periferica base in uscita, cioe' la numero 3 il video. I Byte che devono uscire , cioe' che sono in Output sono caricati nell' Accumulatore, viene chiamata la routine CHROUT e successivamente i dati sono inviati alla periferica selezionata, mentre il canale viene lasciato aperto.

NOME            : **CIOUT**

FUNZIONE :    Trasmette un Byte sul bus seriale

INDIRIZZO:   \$FFA8 - 65448

DESCRIZIONE: Questa routine e' utilizzata per inviare informazioni a periferiche collegate al bus seriale. Percio' la messa in funzione di questa routine avra' come conseguenza l'immissione di un byte di dati sul bus seriale usando un HANDSHAKING seriale pieno. Prima di chiamare questa routine, deve essere chiamata la routine LISTEN che ordina' alla



periferica sul BUS seriale di tenersi pronta a ricevere i dati. (Se alla periferica necessita un indirizzo secondario questo deve essere inviato attraverso l' utilizzo della routine SECOND che vedremo in seguito). La periferica deve essere in ascolto o sara' generato, attraverso la parola di stato , un errore di fuori tempo (TIMEOUT).

NOME            : **CINT**

FUNZIONE : Inizializza l' editor di schermo e l' integrato 6567

INDIRIZZO: \$FF81 - 65409

DESCRIZIONE: Questa routine abilita l' integrato 6567 (VIDEO CONTROLLER) nel C128 per le normali operazioni. Viene inizializzato anche il KERNAL SCREEN EDITOR Dovrebbe essere chiamata in funzione da un catridge.

NOME            : **CLALL**

FUNZIONE : Chiude tutti i Files

INDIRIZZO: \$FFE7 - 65511

DESCRIZIONE: Questa routine serve per chiudere tutti i files aperti. Quando entra in funzione questa routine i puntatori della tavola dei file aperti sono resettati, chiudendo cosi' tutti i files. Anche la routine CLRCHN viene chiamata per resettare tutti i canali di I/O.

NOME            : **CLOSE**

FUNZIONE : Chiude un file logico

INDIRIZZO: \$FFC3 - 65475

DESCRIZIONE: Questa routine e' utilizzata per chiudere un file logico dopo che tutte le operazioni di I/O sullo stesso file sono state eseguite. La routine e' chiamata dopo che l' accumulatore e' stato caricato con il numero di file logico che deve essere chiuso. Naturalmnete questo sara' lo stesso numero usato quando il file era stato aperto con OPEN.

NOME            : **CLRCHN**

FUNZIONE : Pulisci i canali di I/O

INDIRIZZO: \$FFCC - 65484

DESCRIZIONE: Questa routine e' utilizzata per eseguire il CLEAR di tutti i canali aperti e ripristinare gli stessi canali ai loro valori originari. La periferica normale di INPUT e' 0 (cioe' la tastiera) , mentre la periferica normale di OUTPUT e' 3 ( cioe' il video). Se uno dei canali di comunicazione e' su una porta seriale, viene inviato per prima cosa un segnale di UNTALK per eseguire la pulizia del canale di Input o un segnale di UNLISTEN per la pulizia del canale di Output. Non eseguendo la chiamata a questa routine e quindi lasciando gli ascoltatori ( LISTENERS ) attivi sul bus seriale, diverse periferiche possono ricevere gli stessi dati dal C128 allo stesso tempo. Un sistema per utilizzare questa particolarita' potrebbe essere quello di mettere la stampante in TALK e il disco in

LISTEN per consentire la stampa diretta di un file disco. La routine CLRCHN entra automaticamente in funzione dopo l'esecuzione di CLALL.

NOME            : **GETIN**

FUNZIONE : Riceve un carattere da periferica.

INDIRIZZO: \$FFE4 - 65508

DESCRIZIONE: Se il carattere e' da tastiera, questa routine prende un carattere dalla coda di tastiera (KEYBOARD QUEUE) e lo riporta nell' Accumulatore come valore ASCII. e il buffer ( cioe' la coda di tastiera che e' appunto un buffer) e' vuota allora il carattere caricato nell' Accumulatore sara' zero. I caratteri sono immessi nella coda di tastiera utilizzando sia una parte HARDWARE (INTERRUPT DRIVEN KEYBOARD) sia la routine di scnsione della tastiera SCNKEY. Il buffer puo' contenere al massimo 10 caratteri per cui se e' pieno gli altri carateri che si tentera' di immettere saranno ignorati fino a quando almeno un carattere non sia rimosso dalla coda.

Se il canale invece di essere la tastiera e' l' RS-232 allora viene usato solo il registro A e viene riportato un solo carattere e sara' necessario utilizzare READST per il controllo di validita'.

Se il canale invece e' seriale, cassetta o schermo e' chiamata la routine BASIN.

NOME            : **IOBASE**

FUNZIONE : Definisce la pagina di memoria I/O

INDIRIZZO: \$FFF3 - 65523

DESCRIZIONE: Questa routine fissa i registri X e Y all' indirizzo della sezione di memoria che definisce dove sono localizzate le periferiche I/O. Questo indirizzo puo' essere utilizzato come linea di deviazione (OFFSET) per accedere alla memoria disegnata per le periferiche I/O. La linea di deviazione e' il numero di locazioni dall' inizio della pagina sulla quale si desidera che i registri I/O siano immessi.

NOME            : **IOINIT**

FUNZIONE : Inizializza periferiche I/O

INDIRIZZO: \$FF84 - 65412

DESCRIZIONE: Questa routine inizializza tutte le periferiche di I/O e le routines. E' normalmente chiamata come parte di una procedura di inizializzazione di un programma su cartridge

NOME            : **LISTEN**

FUNZIONE : Invia un comando di ascolto ad una periferica sul Bus seriale

INDIRIZZO: \$FFB1 - 65457

DESCRIZIONE: Questa routine ordina ad una periferica sul Bus seriale di ricevere dati. L' Accumulatore deve essere caricato con un numero compreso fra 0 e 31 prima di chiamare questa routine.

LISTEN eseguirà un OR logico sul numero bit per bit per convertirlo in un indirizzo di ascolto e poi trasmetterà questo dato come comando sul bus seriale.

La periferica specificata si metterà allora in modo di ascolto e sarà pronta per ricevere informazioni.

NOME            : **LOAD**

FUNZIONE : Carica RAM da una periferica

INDIRIZZO: \$FFD5 - 65493

DESCRIZIONE: Questa routine carica Bytes di dati da una qualsiasi periferica in INPUT

direttamente entro la memoria del computer. Può anche essere usata per una operazione di verifica che avviene confrontando i dati presenti sulla periferica con quelli in memoria e lasciando i dati in memoria inalterati.

L' Accumulatore deve essere messo a 0 per un' operazione di LOAD o messo a 1 per un' operazione di verifica.

Se la periferica in Input e' aperta con un' indirizzo secondario di 0, allora sara' ignorata la testata (HEADER) dell' informazione.

In questo caso i registri X e Y devono contenere l' indirizzo di partenza per LOAD. Se la periferica e' collegata con un' indirizzo secondario 1 0 2, allora i dati saranno caricati in memoria con partenza dall' indirizzo specificato dalla testata.

Questa routine inoltre riporta l' indirizzo della piu' alta locazione di RAM caricata.

Prima di chiamare questa routine e' necessario chiamare le routines SETLFS e SETNAM.

#### NOTA

Non si puo' eseguire il LOAD da Tastiera (0), RS-232 (2) o schermo (3).

NOME            : **MEMBOT**

FUNZIONE : Fissa la parte piu' bassa della memoria

INDIRIZZO: \$FF9C - 65436

DESCRIZIONE: Questa routine e' usata per fissare la parte piu' bassa della memoria. Se il bit di Carry dell' Accumulatore e' a 1 quando viene chiamata questa routine, allora un puntatore che indica il Byte piu' basso della RAM e' riportato in X e Y.

NOME            : **MEMTOP**

FUNZIONE : Fissa la parte alta della memoria

INDIRIZZO: \$FF99 - 65433

DESCRIZIONE: Questa routine e' utilizzata per fissare il punto massimo della memoria RAM. Il funzionamento e' simile alla routine precedente (MEMBOT).

Infatti anche in questo caso quando si utilizza questa Routine con il bit di Carry dell' Accumulatore a 1, il puntatore alla fine della memoria RAM e' caricato nei registri X e Y.

Quando invece la si utilizza con il bit di carry a 0, allora il contenuto dei registri X e Y e' caricato nel puntatore al massimo della memoria.

NOME            : **OPEN**

FUNZIONE : Apre un file logico

INDIRIZZO: \$FFC0 - 65472

DESCRIZIONE: Questa routine e' utilizzata per

eseguire la funzione di apertura di un File logico. Non appena il file logico e' stato fissato questi puo' essere utilizzato per operazioni di I/O.

Molte delle Routines del sistema operativo fanno uso di OPEN.

Non sono necessari argomenti o operandi ma prima di utilizzare questa routine sara' necessario metterne in funzione altre due cioe' la SETLFS e la SETNAM.

**NOME : PLOT**

**FUNZIONE :** Legge e fissa la posizione del cursore

**INDIRIZZO:** \$FFF0 - 65520

**DESCRIZIONE:** Quando si salta a questa routine con il carry dell' accumulatore a 1, allora la posizione attuale del cursore , nelle sue coordinate X e Y sara' caricata nel registri Y e X dove Y sara' il numero di colonna del cursore ( da 0 a 79) e X il numero di riga occupato dal cursore ( da 0 a 24).Se invece il carry e a 0 allora verranno letti i valori dei registri X e Y e il cursore posizionato a quei valori.

**NOME : RAMTAS**

**FUNZIONE :** Controlla le RAM, fissa aree per buffer nastro e schermo

**INDIRIZZO:** \$FF87 - 65415

**DESCRIZIONE:** Questa routine e' utilizzata per controllare la memoria RAM e fissare i puntatori della memoria sia in alto che in



basso. Esegue anche il clear delle locazioni \$0000 fino a \$0101 e da \$0200 a \$03FF. Normalmente questa routine e' chiamata come parte di un processo di inizializzazione di un Cartridge.

NOME            : **RDTIM**

FUNZIONE : Legge l'orologio in tempo reale

INDIRIZZO: \$FFDE - 65502

DESCRIZIONE: Questa routine e' utilizzata per leggere il clock o orologio di sistema. La risoluzione del clock, cioe' il tempo minimo e' di 1/60mo di secondo.

Il risultato della lettura di questa routine e' di 3 Bytes che sono riportati rispettivamente nell' Accumulatore , nel registro X e Y. Operando con questi tre registri e, come vedremo poi con la routine SETTIM e' possibile leggere e variare il contenuto dell' orologio del sistema.

NOME            : **READST**

FUNZIONE : Legge lo STATUS WORD

INDIRIZZO: \$FFB7 - 65463

DESCRIZIONE: Questa routine riporta lo stato attuale delle periferiche in I/O nell' accumulatore. E' utilizzata di norma dopo ogni colloquio con le periferiche e riporta le informazioni sullo stato delle periferiche stesse o eventuali errori incontrati durante operazioni di I/O.

NOME            : **RESTOR**

FUNZIONE : Reintegra i vettori di sistema.

INDIRIZZO: \$FF8A - 65418

DESCRIZIONE: Questa routine reintegra i valori mancanti dei vettori di tutto il sistema usati sia nelle KERNAL che nel BASIC come routines e come interrupts.

NOME            : **SAVE**

FUNZIONE : Salva la memoria RAM su periferica

INDIRIZZO: \$FFD8 - 65496

DESCRIZIONE: Questa routine e' utilizzata per eseguire l' operazione di SAVE di una parte di memoria. La memoria e' salvata da un' indirizzo indiretto in pagina 0 specificato dall' Accumulatore a un' indirizzo immagazzinato nei registri X e Y.

Sara' quindi inviato ad un File logico su una periferica.

Le routines SETLFS e SETNAM devono essere utilizzate prima di accedere a questa routine.

Tuttavia non e' necessario dare un nome al file che si desidera salvare su cassetta, mentre e' necessario per qualsiasi altra periferica.

NOME            : **SCNKEY**

FUNZIONE : Esegue la scansione di tastiera

INDIRIZZO: \$FF9F - 65439

DESCRIZIONE: Questa routine esegue la scansione (cioe' la lettura) della tastiera e

controlla se ci sono tasti premuti.  
E' la stessa routine chiamata per mezzo della manipolazione di Interrupt.  
Se un tasto e' premuto, allora il suo valore ASCII e' immesso nella coda di tastiera.

NOME            : **SECOND**

FUNZIONE : Invia un indirizzo secondario per la funzione di ascolto ( LISTEN)

INDIRIZZO: \$FF93 - 65427

DESCRIZIONE: Questa routine e' utilizzata per inviare un indirizzo secondario ad una periferica in I/O dopo che e' stata effettuata una chiamata alla routine LISTEN e quindi e' stato ordinato alla periferica di porsi in ascolto. Questa routine non puo' essere usata per inviare un indirizzo secondario dopo un salto alla routine TALK (chiamata). Normalmente un indirizzo secondario e' usato per comunicare il tipo di informazione che si desidera inviare alla periferica.

NOME            : **SETLFS**

FUNZIONE : Fissa il file logico ( in maniera completa)

INDIRIZZO: \$FFBA - 65466

DESCRIZIONE: Questa routine fissa il numero di file logico, l' indirizzo della periferica e l' indirizzo secondario per le altre routines. Il numero di file logico e' usato dal sistema come chiave di riferimento alla tavola creata dalla routine OPEN file.

L' indirizzo della periferica puo' essere un numero dell' intervallo da 0 a 31.

NOME            : **SETMSG**

FUNZIONE : Controllo dei messaggi di sistema  
in uscita

INDIRIZZO: \$FF90 - 65420

DESCRIZIONE: Questa routine controlla la stampa di errore ed i messaggi di controllo delle Kernal routines. Sia la stampa dei messaggi di errore come la stampa dei messaggi di controllo possono essere selezionate, cioe' scelte, fissando l' accumulatore quando viene chiamata la routine.

NOME            : **SETNAM**

FUNZIONE : Fissa il nome del file

INDIRIZZO: \$FFBD - 65496

DESCRIZIONE: Questa routine e' utilizzata per fissare il nome del file per le routine di OPEN, SAVE e LOAD. L' Accumulatore deve essere caricato con la lunghezza del nome del file. I registri X e Y devono essere caricati con l' indirizzo del nome del file secondo il formato 6510 cioe' prima il byte basso e poi il byte alto. L' indirizzo puo' essere un qualsiasi proponibile indirizzo di memoria del sistema dove sia appunto immagazzinata una stringa di caratteri che e' il nome del file. Se non si desidera nessun nome, allora l' Accumulatore deve essere messo a 0 che rappresentera' un file di lunghezza zero. In

questo caso i registri X e Y possono essere fissati ad un qualsiasi indirizzo di memoria.

NOME            : **SETTIM**

FUNZIONE    : Fissa i valori del clock di sistema

INDIRIZZO: \$FFDB - 65499

DESCRIZIONE: L' orologio di sistema e' mantenuto da una routine di interrupt che lo aggiorna ogni sessantesimo di secondo (un "JIFFY" o ciclo ).

Il sistema di clock occupa 3 Bytes che da una capacita' di contare fino a 5.184.000 cicli ( o JIFFY) per un totale di 24 ore dopo di che l' orologio torna a zero.

NOME            : **SETTMO**

FUNZIONE    : Fissa il flag di fuori tempo ( TIME-OUT) sul bus IEEE

INDIRIZZO: \$FFA2 - 65442

DESCRIZIONE: Questa routine fissa il flag di Fuori tempo per la IEEE. Quando questo flag e' messo a 1 il computer attendera' una risposta da una periferica sulla IEEE per 64 millisecondi.

Se la periferica non rispondera' al segnale DAV (cioe' DATA ADDRESS VALID) entro questo tempo allora il CBM64 riconoscerà una condizione di errore ed abbandonerà la sequenza di HANDSHAKE.

Quando questa routine e' chiamata ed il bit 7 dell' accumulatore contiene uno 0 allora il TIMEOUT e' abilitato, mentre un 1 nello

stesso bit dell' accumulatore lo disabilita.

NOME            : **STOP**

FUNZIONE : Controlla se il tasto di STOP e' premuto.

INDIRIZZO: \$FFE1 - 65505

DESCRIZIONE: Se il tasto di STOP era premuto durante la chiamata alla routine UDTIM, la chiamata a questa routine mette a 1 il flag Z. Per di piu' i canali saranno resettati per mancanza di valori, mentre tutti gli altri flags rimarranno immutati. Se il tasto di STOP non era premuto allora l' Accumulatore conterra' un Byte che rappresenta l' ultima riga della scansione di tastiera . L' utente con questo metodo puo' anche controllare alcuni altri tasti.

NOME            : **TALK**

FUNZIONE : Comando ad una periferica sul BUS seriale di TALK.

INDIRIZZO: \$FFB4 - 65460

DESCRIZIONE: Per utilizzare questa routine per prima cosa l' Accumulatore deve essere caricato con un numero di periferica fra 0 e 31. Quando e' chiamata questa routine, allora viene eseguito un OR logico bit per bit per convertire il numero della periferica in un indirizzo di chiamata. Quindi questi dati saranno trasmessi come comando sul Bus seriale.

**NOME :TKSA**

**FUNZIONE :** Invia un indirizzo secondario ad una periferica dopo la routine TALK

**INDIRIZZO:** \$FF96 - 65430

**DESCRIZIONE:** Questa routine trasmette un indirizzo secondario ad una periferica in attesa di TALK sul bus seriale. Prima di chiamarla deve esserci un numero fra 0 e 31 nell' accumulatore. La routine invia questo numero come un comando di indirizzo secondario sul bus seriale. TKSA puo' essere messa in funzione dopo la chiamata a TALK mentre non operera' dopo una routine o un comando di LISTEN.

**NOME :UDTIM**

**FUNZIONE :** Incrementa l' orologio del sistema

**INDIRIZZO:** \$FFEA - 65514

**DESCRIZIONE:** Questa routine incrementa l' orologio del sistema. Normalmente questa routine e' chiamata dalla normale routine KERNAL di interrupt ogni sessantesimo di secondo. Se l' utente si programma da se gli interrupt questa routine DEVE essere chiamata per incrementare il temporizzatore. Per di piu', se il tasto STOP e' funzionante, cioe' non e' stato disabilitato, deve essere chiamata anche la routine di STOP che abbiamo visto prima.

NOME            : **UNLSN**

FUNZIONE : Invia un comando di UNLISTEN

INDIRIZZO: \$FFAE - 65454

DESCRIZIONE: Questa routine ordina a tutti le periferiche sul bus seriale di fermare la ricezione dei dati dal computer

In altre parole chiamando questa routine viene inviato un comando di UNLISTEN sul bus seriale. Cio' naturalmente avra' effetto solo sulle periferiche alle quali era stato in precedenza inviato un comando di LISTEN.

NOME            : **UNTLK**

FUNZIONE : Invia un comando di UNTALK

INDIRIZZO: \$FFAB - 65451

DESCRIZIONE: Come la precedente solo che invia un messaggio di UNTALK. Naturalmente anche in questo caso avremo una disabilitazione delle periferiche dal bus seriale .

NOME            : **VECTOR**

FUNZIONE : Manipola i vettori su RAM

INDIRIZZO: \$FF8D - 65421

DESCRIZIONE: Questa routine manipola tutti i sistemi di indirizzi di salto vettorizzati immagazzinati in RAM. Chiamando questa routine con il bit di carry dell' accumulatore a 1, l' attuale contenuto dei vettori della RAM viene immagazzinato in una lista a cui puntano X e Y.

Chiamando invece questa routine con lo stesso



bit a 0, una lista dell' utente indirizzata dal contenuto dei registri X e Y e' trasferita nel sistema dei vettori RAM

NOME            : **CLRWIN**  
FUNZIONE : cancella una finestra  
INDIRIZZO: \$C142 - 49474

DESCRIZIONE Se non e' definita alcuna finestra verra' eseguita una pulizia su tutto lo schermo. Se invece questa e' definita la cancellazione avviene entro i limiti della finestra.

NOME            : **CURHOM**  
FUNZIONE : porta il cursore nella posizione HOME all' interno della finestra  
INDIRIZZO: \$C150 - 49482

DESCRIZIONE Il Cursore viene posizionato nell' angolo superiore sinistro della finestra. Se non e' definita alcuna finestra allora il cursore si porta nella parte superiore sinistra dello schermo che e' la posizione 0,0.

NOME            : **GETLIN**  
FUNZIONE : prende una riga di input  
INDIRIZZO: \$C258 - 49752

DESCRIZIONE Dalla tastiera si prendono tanti Bytes che verranno poi scritti sullo schermo nella posizione attuale del cursore fino a che si preme il tasto RETURN.

**NOME            :BSOUT SCNR**

**FUNZIONE** : Output di un Byte sullo schermo attuale.

**INDIRIZZO**: \$C72D - 50989

**DESCRIZIONE**      questa routine e' la continuazione della routine BSOUT di indirizzo FFD2 pero' si risparmia alcune richieste prima di arrivare allo schermo per cui e' piu' veloce.

Il Byte viene consegnato nell' accumulatore e scritto sullo schermo attivo in quel momento nella posizione attuale del cursore.

**NOME            :CLQIR**

**FUNZIONE** : cancella QUOTE, INS e RVS MODE

**INDIRIZZO**: \$C77D - 51069

**DESCRIZIONE** Vengono cancellati i FLAG per le virgolette, gli Insert e Reverse .

Questa routine lavora un po' piu' veloce di quanto avvenga con BSOUT.

## **LE ALTRE ROUTINES**

Vediamo ora alcune altre routine interessanti

\$C854 - 51284    Cursore a destra nella finestra

\$C85A - 51290    Cursore in basso nella finestra

\$C867 - 51303    Cursore in alto nella finestra  
\$C875 - 51317    Cursore a sinistra nella  
finestra  
\$C880 - 51328    Inserisce il II set di  
caratteri  
\$C8BF - 51391    Disabilita il modo reverse  
\$C8C1 - 51393    Abilita il modo reverse  
\$C8C7 - 51399    Inserisce il modo  
sottolineatura  
\$C8CE - 51406    Disabilita la sottolineatura  
\$C91B - 51483    Cancella il carattere a  
sinistra del cursore  
\$C93D - 51517    Cancella il carattere sotto il  
cursore  
\$C94F - 51535    Funzione TABULATORE  
\$C980 - 51584    Disabilita tutti i tabulatori  
\$C98E - 51598    Fa suonare il campanello  
(BELL)  
\$CA14 - 51732    Il cursore definisce la  
posizione alto sinistra della finestra  
\$CA16 - 51734    Il cursore definisce la  
posizione alto a destra della finestra  
\$CA24 - 51748    Definisce lo schermo per la  
finestra  
\$CA52 - 51794    Cancella la riga attuale  
\$CA76 - 51830    Cancella dalla posizione del  
cursore fino al termine della riga.  
\$CA8B - 51851    Cancella dall'inizio riga  
fino all'attuale posizione del cursore  
\$CA9F - 51871    Cancella dall'attuale  
posizione del cursore fino alla fine dello  
schermo.  
\$CABC - 51900    Scroll in alto  
\$CAF2 - 51954    Inserisci il cursore come  
BLOCCO  
\$CAFE - 51966    Inserisci il cursore come  
LINEA

\$CB0B - 51979    Disabilita il Flash del  
cursore  
\$CB21 - 52001    Abilita il Flash del cursore  
\$CB3F - 52031    Reverse dello schermo a 80  
caratteri  
\$CB48 - 52040    Schermo a 80 caratteri in modo  
normale  
\$CC27 - 52263    Esegue uno Space al posto del  
cursore  
\$CC2F - 52271    Preleva il carattere in  
Accumulatore e lo mette nella posizione  
attuale del cursore.  
\$CC4A - 52298    Sulla posizione attuale del  
cursore viene visualizzato il carattere che  
si trova nell' Accumulatore, nel colore del  
registro X alla colonna Y, senza muovere il  
cursore.  
\$CC6A - 52330    Fissa la posizione del cursore  
\$CD2C - 52524    Passa nei modi 40 e 80  
colonne.

## **MEMORY MANAGEMENT UNIT**

### Introduzione

L' unita' di controllo e manipolazione della memoria , MEMORY MANAGEMENT UNIT o MMU, ha il compito di controllare e gestire l' intera memoria del C 128.

Si tratta infatti di una gestione molto complessa perche' il costruttore, desiderando utilizzare ancora processori ad 8 bit come l' 8502 , derivazione del vecchio e glorioso oltre che diffusissimo 6502, e l' altrettanto diffuso Z80, che sono pero' capaci di gestire solo 64K Bytes per volta, ma volendo altresì disporre di una quantita' maggiore di memoria, appunto 128 K e' dovuto ricorrere ad una unita' specifica per la gestione della memoria.

Inoltre si e' resa necessaria sfruttare la compatibilita' completa con il C64 e controllarne quindi tutte le operazioni sia del funzionamento in modo 128 che in C/PM sotto Z80.

Si puo' anzi affermare che quest' ultimo debba essere considerato del tutto indipendente e separato dagli standard abituali COMMODORE, in altre parole, e non per parafrasare la pubblicita' che in questo caso e' vera, ancora un' altro computer.

Vediamo ora le funzioni, in modo schematico,

di questo integrato:

- 1) Controllo del TAB, cioè del Bus di Traslazione indirizzi (TA8-TA15).
- 2) Emissione dei segnali di controllo sia per la selezione del tipo di computer, cioè quelli funzionanti con l'8500 e quello con lo Z80, sia per la scelta del modo di funzionamento appunto C64, C128 e Z80.
- 3) Controllo delle linee CAS per le operazioni di Bank-Switching della RAM.

E' essenziale per una corretta comprensione del MEMORY MANAGEMENT UNIT capire come funziona l'intero sistema della memoria del C 128 che e' appunto controllato dal MMU.

La MMU ha un totale di 11 registri che hanno un indirizzo iniziale \$D500. Ora poiche' i registri del gruppo I/O, cioè quelli che controllano l'ingresso ed uscita dati non sono sempre attivi parte di questi registri sono copiati a \$FF00 fino a \$FF05.

Questi registri controllano il funzionamento degli indirizzi del microprocessore per un totale di 64K bytes di spazio indirizzabile fino ad un MEGA Bytes di RAM e fino a 32K bytes di ROM esterna indirizzabile dal C128. Come potremo vedere ci saranno 4 gruppi di configurazioni disponibili sui registri di preconfigurazione che potranno essere caricati semplicemente nei registri di (normale) configurazione senza attivare per questo il gruppo di controllo I/O.

Cio' evidentemente consente non solo un risparmio di tempo ma anche una grandissima facilita' e flessibilita' di programmazione.

## **ORGANIZZAZIONE DELLA MEMORIA**

Per la comprensione del funzionamento del MMU e' importante vedere l'organizzazione della memoria dell' intero C128 che e' appunto controllata attraverso i registri dell' MMU stesso.

### MAPPA DEI REGISTRI DELLA MMU

|        |                               |     |
|--------|-------------------------------|-----|
| \$D500 | Registro di configurazione    |     |
| \$D501 | Registro di preconfigurazione | A   |
| \$D502 | "                             | " B |
| \$D503 | "                             | " C |
| \$D504 | "                             | " D |
| \$D505 | " di modo                     |     |
| \$D506 | " di RAM                      |     |
| \$D507 | Puntatore LO Pagina 0         |     |
| \$D508 | " HI Pagina 0                 |     |
| \$D509 | " LO Pagina 1                 |     |
| \$D50A | " HI Pagina 1                 |     |
| \$D50B | Registro versione             |     |

## **LA ROM DEL C128**

La mappa di memoria e' importante anche dal punto di vista del mantenimento della compatibilita' con il C64.

Infatti tutti i cosi' detti modi di funzionamento C64 sono compatibili con il COMMODORE 64 poiche', quando questo modo di funzionamento e' selezionato, il C128 diventa un C64 a tutti gli effetti.

Il modo C128 e' controllato da un sistema di reset a sua volta controllato da un bit del Registro di configurazione del MMU.

Sempre nel modo C128 il MMU si pone come mappa di memoria a \$FF00 e nello spazio di I/O a partire da \$D500. L'impiego dei registri MMU di indirizzo \$FF00 consente la manipolazione della memoria senza avere attivato i blocchi di I/O allo stesso tempo e cio' con una perdita minima della RAM vicina o per meglio dire adiacente. Inoltre la MMU viene completamente rimossa dalla mappa di memoria nel modo C64 anche se viene ancora impiegata dai circuiti interni.

Quando il computer funziona in modo C64 le ROM sia esterne che interne funzionano esattamente allo stesso modo che nel normale C64.

Sia il Basic che le routines Kernal forniscono i dati necessari al funzionamento del C64 relativamente alla parte ROM.

Da rilevare che questa ROM duplica alcune delle locazioni ROM utilizzate nel modo C128 in cui sono presenti ed operative circa 48K bytes di Sistema Operativo.

L'esatto ammontare di spazio viene fissato



dal controllo software cosa che consente di accedere velocemente ai banchi utilizzando quella parte di RAM che non viene impiegata altrimenti.

Le cosi' dette ROM esterne presenti in memoria sono quelle impiegate in modo C64 e sono soggette alle regole di MEMORY MAPPING proprie del C64. Vedi per questo il funzionamento dei cartridges sulle porte di espansione.

Le ROM esterne per il C128 hanno regole simili a quelle impiegate con il TED per l' utilizzo dei banchi di memoria.

Ad esempio vengono osservate (dal sistema naturalmente) all' inizializzazione del sistema per vedere se sono presenti ed in caso positivo se hanno codici di priorita'.

Questo sistema consente una notevole flessibilita' certamente superiore al sistema di sostituzione hardware perche' le ROM che contengono il Basic e le routines Kernal possono essere messe da parte o saltata per un particolare programma, rimesse in funzione per un' altro, selezionate sia per l' intero che per una parte di esse.

Questo sistema di manipolazione a banchi e' attuata scrivendo dei valori nei registri di configurazione sistema presenti nelle locazioni \$D500 o \$FF00 della MMU.

## **ORGANIZZAZIONE DELLA MEMORIA RAM**

La memoria RAM presente attualmente nel

sistema e' composta da 128 K di memoria posta una accanto all'altra.

La RAM viene gestita e quindi si accede ad essa per banchi di 64K che e' il massimo che i due processori (8500 e Z80) ad 8 bit riescono a gestire. Nelle figure relative alla disposizione della memoria l'area mostrata come RAM rappresenta cio' che il microprocessore DOVREBBE vedere se tutta la parte ROM fosse disabilitata. Vediamo ora alcuni accenni sugli effetti della tecnica di manipolazione dei banchi (BANK SWITCHING).

Il banco di memoria o semplicemente banco, in uso in un dato momento e' in funzione dei valori immagazzinati nel registro di configurazione del MMU che vedremo dopo.

Un immagazzinamento diretto su questi registri ha effetto immediato. Un immissione indiretta, utilizzando valori di configurazione programmati, puo' essere effettuato andando a scrivere su uno dei registri di caricamento indiretti noti anche come LCR o LOAD CONFIGURATION REGISTER localizzabili nella zona di memoria di indirizzo \$FF00.

La scrittura indiretta attraverso i PCR o PreConfiguration Register consentono al programmatore di avere a disposizione ben 4 diverse configurazioni preprogrammate che permette la personalizzazione di ogni banco in qualsiasi momento.

Ad esempio il Banco n. 1 che e' un DATA Bank, puo' essere selezionato solo come banco RAM, senza avere cioe' ne' ROM ne' I/O abilitati, mentre il Bank 0 potra' contenere la ROM di sistema e gli I/O abilitati.

Inoltre la lettura di un qualsiasi LCR

riporterà il valore dei corrispondenti PCR. Quando si tratta con più banchi di memoria contemporaneamente può avvenire che si desideri utilizzare il banco 1 come RAM pura ma si desideri nello stesso tempo conservare i valori di sistema come Stack, Pagina zero, Schermo, ecc.

La MMU mette a disposizione un' area di memoria chiamata RAM comune o COMMON RAM.

La COMMON RAM è programmabile sia come dimensione che come contenuti di partenza e di arrivo. La dimensione viene fissata dai bit 0 e 1 dell' RCR (RAM CONFIGURATION REGISTER). Se il valore di questi due bit è 0 allora l' area di COMMON RAM sarà di 1K, mentre i valori di 1, 2 e 3 daranno dimensioni rispettivamente di 4, 8 e 16 K di COMMON RAM.

Infine se il bit 2 è a 1 allora la partenza della COMMON RAM è dalla parte bassa della memoria per l' ampiezza decisa in precedenza, mentre se è =1 il bit 3 la partenza dei dati da salvare è dalla parte alta della memoria.

Comunque in tutti i casi la COMMON RAM è fisicamente nel banco 0.

La Pagina 0 e la pagina UNO possono essere locate o rilocate indipendentemente da RCR.

Quando il processore va a cercare un indirizzo in pagina zero o in pagina uno e non lo trova, la MMU somma all' indirizzo alto del processore (H/A) il contenuto pari di P0 o di P1 rispettivamente ed immette questi nuovi indirizzi nel BUS includendo i bits di indirizzamento esteso A16 ed A17. Il procedimento di BANKING fa sì che si possa raggiungere così i nuovi indirizzi.

Nello stesso tempo i contenuti di P0 e P1 sono sottoposti ad un comparatore digitale ed viene messa in funzione una sostituzione rovesciata se gli indirizzi dati dal processore eseguono una ricerca inutile, cioe' non trovano nulla.

Per gli accessi al VIC due bits del MMU sostituiscono le linee A16 ed A17.

## **I REGISTRI DI CONFIGURAZIONE**

I registri di configurazione (CR) controllano le configurazioni ROM, RAM e di I/O del modo C128. Sono di indirizzo \$D500 nello spazio di I/O e \$FF00 nello spazio di sistema.

Nel modo C128 il bit 0 controlla se sono necessari spazi di I/O (\$D000-\$DFFF) o un accesso RAM/ROM siano necessari.

Un bit basso (LO) selezionera' un I/O, uno alto (HI) abiliterà alcune parti di accessi RAM/ROM, la natura dei quali e' controllata da altri bits in questo registro.

Il valore di questo bit e' immagazzinato in un preregistro fino a quando non venga rilevato un fronte di discesa del CLOCK per prevenire variazioni in presenza di valori instabili.

L' I/O del modo 64, la linea hardware guidata da questo bit e' messa alta.

Da notare che quando non esiste spazio di I/O l' accesso ROM/RAM e' controllato dai bits definiti di ROM configurazione alta che

vedremo fra poco. Questi bits sono messi a 0..

Quando il bit di I/O e' alto i registri MMU da \$D500 a \$D503 avranno il loro peso, mentre quando il bit e' basso questi registri spariscono dalla mappa di memoria. I registri MMU da \$FF00 a FF04 sono sempre disponibili nel modo 128.

La linea hardware I/OSE tiene sempre conto della polarita' di questo bit quando siamo in modo 128.

Sempre in modo 128 il bit 1 controlla l'accesso del processore allo spazio basso della ROM (\$4000-\$7FFF). Se il bit e' alto l'area appare come RAM e viene generato al banco di RAM selezionato dagli altri bit di questo stesso registro, un accesso RAM CAS. Se il bit e' basso allora la ROM e' piazzata nell'area.

Questo bit inoltre influenza le linee di status della memoria MS0 e MS1 che vengono decodificate dal PLA per generare la selezione degli integrati ROM.

La selezione delle ROM a questo punto mette basse entrambe le linee di status della memoria quando gli indirizzi selezionati dal processore non sono rilevati nel dichiarato intervallo.

Questi bit sono poi messi a 0 per includere la ROM bassa del basic nel 128.

I bits 2 e 3, in modo 128, determinano il tipo di memoria che sara' presente dagli indirizzi \$8000 a \$8FFF. Se entrambi sono a 0, cioe' bassi, questa zona sara' riservata alla ROM di sistema.

Con solo il bit 2 alto ci sara' la ROM di

funzione interna. Con il bit 3 alto da solo sarà dedicato a ROM di funzione esterna. Con entrambi i bits alti la zona è dedicata alla RAM con generazione di CAS. Questi due bits inoltre hanno effetti sulle linee di accesso hardware alla memoria. Infatti MS0 riflette lo stato del bit 3 e MS1 quello del bit 2. Entrambi questi bit sono messi bassi al reset con il Basic alto.

I bits 4 e 5 determinano i contenuti del blocco alto di memoria (\$C000-\$FFFF) e solo in modo C128 mentre non hanno nessun effetto in modo C64.

Il funzionamento di questi bit e le conseguenze del fatto che siano alti o bassi sono del tutto identiche alla precedente coppia appena vista.

Da notare che il bit di configurazione I/O, quando è fissato per questo spazio, lascia l'area da \$D000 a \$DFFF disposta come spazio di I/O relativamente ai valori di questi bits.

Le locazioni da \$D000 a \$DFFF se non sono state fissate per lo spazio di I/O contengono la ROM carattere. Stesso discorso come sopra per MS0 e MS1 rispondenti relativamente a bit 5 e bit 4.

Ricordiamo che i bit 6 e 7 controllano la selezione dei banchi RAM.

Una parola sui registri di PRECONFIGURAZIONE. Come abbiamo visto nello schema iniziale questi sono di indirizzo \$D501-2-3-4 e non vengono usati in modo C64.

In poche parole si tratta di registri della MMU che possono essere programmati in

anticipo, di qui il nome, con una data configurazione di memoria, poi i valori saranno posti nei registri di configurazione veri e propri. I valori iniziali di questi registri possono essere letti anche nelle locazioni di memoria \$FF01-2-3-4.

## **I REGISTRI DI MODO**

Si tratta di una sola locazione di memoria di indirizzo \$D505 che con i vari valori che assume indica al computer in che modo si sta operando cioè in modo 128, 64 o CPM con lo Z80.

Vediamo come si opera sui singoli bit mettendoli a zero o a uno e che risultati se ne ottiene.

Quando il bit 0 è a 1 è attivato lo Z80, quando è messo a zero l'8502. Da notare che questo bit è sempre a 0 quando si usa il modo 64.

I bits 1 e 2 non sono utilizzati mentre il bit 3 può essere usato come bit di uscita per il buffer del bus seriale in modo FAST. Oppure per il segnale di abilitazione del disco.

I bits 4 e 5 non sono attivi nel modo 128. Servono invece nel modo 64 per controllare, essendo bidirezionali, se le linee relative alla porta dei cardtridges sono attive. Se queste due linee sono attive allora il controllo passa appunto ai cartridges inseriti. Fare attenzione alle notevoli

possibilita' di simulazione attiva di queste linee per la protezione e sprotezione dei programmi.

Se il bit 6 contiene 1 sara' attivo il modo 64 se 0 il modo 128. Ricordiamo che all'accensione o qualora si dia o si simuli un RESET verra' attivato sempre il modo 128, a meno che non si operi con i tasti come indicato nella guida e come si puo' vedere sia nel corso del presente manuale che sulla guida.

Infine il bit 7 indica se siamo in 40 (=1) o in 80 (=0) colonne.

## **I PUNTATORI PER LE PAGINE**

I puntatori per le pagine sono 4 registri che consentono un'allocazione indipendente delle pagine 0 e 1 quando si opera sotto entrambi i processori.

Per la rilocalizzazione della pagina zero avremo a disposizione il puntatore HI (POH) di pagina zero ed il puntatore di LO (POL) della stessa pagina.

I bits 0 e 3 del registro POH corrisponde agli indirizzi di trasferimento TA16 e TA19 rispettivamente per ogni ingresso di pagina 0 e cioe' \$0000-\$00FF.

In un sistema a 128K il bit 0 controlla la generazione di CAS0 e CAS1 a seconda di che sia esso alto o basso mentre i bit 1-23 sono ignorati.

In un sistema a 256K i bits 0 e 1 vengono



direttamente trasferiti su TA16 e TA17 mentre i bits 2 e 3 sono ignorati. Invece in un sistema da 1Mega i bits 2 e 3 divengono i bits di estensione indirizzi TA18 e TA19, mentre i restanti vanno a zero.

I bits di ROM e dei blocchi di I/O definiscono quali pagine vengono fisicamente identificate come pagina zero per determinare gli accessi di tutta la pagina.

Una operazione di scrittura sui registri di POH resta immagazzinata fino ad un'operazione analoga su POL.

I bits da 0 a 7 di POL corrispondono agli indirizzi di trasferimento da TA8 a TA15 per un qualsiasi accesso a pagina 0 fino ad un'eventuale modifica.

In questo caso per modifica si intende naturalmente una rilocazione della pagina zero. Un qualsiasi accesso alla pagina zero, dopo il rilocamento viene riportato alla funzione originale.

Un'operazione di scrittura su questi registri attiva il trasferimento di pagina 0 che viene attivato al successivo ciclo del segnale di clock (LO).

POL e' di indirizzo \$D507 nello spazio di I/O, il registro POH e' di indirizzo \$D508.

I puntatori di pagina 1 (PlH) e (PlL) fanno esattamente le stesse cose di quelli visti per la pagina zero. Gli indirizzi sono PlL a \$D509 e PlH a \$D50A.

Da notare che entrambi i registri pari sono messi a zero da un'operazione di RESET abilitando cosi' le reali pagine zero e uno ed i relativi accessi.

## **I REGISTRI DI SISTEMA**

Questi sono registri di versione sistema di indirizzo \$D50B nel blocco di I/O.

Questo e' un registro a sola lettura che riporta un codice contenente la versione del MMU oltre che le possibilita' e le dimensioni del sistema di memoria.

I bits da 0 a 3 contengono il numero di versione MMU del sistema. Non sono molto interessanti almeno a livello attuale mentre potranno essere validi in futuro.

I bits da 4 a 7 danno informazioni sulla memoria totale disponibile nella configurazione.

Questo computer, e pare che negli USA sia stato fatto, si puo' espandere fino ad 1 Mega-byte di memoria. Il contenuto standard di questi registri e' di 2 BANCHI di 64 K con un valore di \$20

# APPENDICI



## **LE APPENDICI**

Abbiamo ritenuto importante per i lettori allegare a questo volume una serie di appendici che potrebbero rivelarsi di utilita' sia per una migliore comprensione del Sistema Operativo stesso sia per quello che ci e' sembrato un necessario approfondimento.

Non vogliamo che marginalmente entrare nel campo dell' Hardware al quale probabilmente sara' dedicato un apposito volume, ma e' necessario cercare di mettere bene in luce gli stretti rapporti che collegano la parte SOFTWARE del Sistema Operativo di un computer con i componenti che sono utilizzati.

Cercheremo quindi di mettere in luce il rapporto di funzionalita' e di interdipendenza che lega i principali componenti. Della MMU abbiamo gia' approfonditamente parlato, mentre del microprocessore 8502 cuore del computer stesso non e' necessario dare nessuna informazione perche' queste riguardano essenzialmente l' HARDWARE.

Tratteremo quindi dei seguenti integrati e delle loro funzioni:

- \*\*VIC II
- \*\*CIA
- \*\*SID
- \*\*8563 VDC

La parte relativa poi allo Z-80 sara' trattata solo con riferimento alle routines di BOOT del sistema e non come CPM perche'

anche questo richiede un manuale a parte.  
Di ogni integrato daremo le caratteristiche principali e la mappa dei registri spiegandone brevemente il significato.

## **L' INTEGRATO VIC II**

Il nome di questo integrato VIDEO INTERFACE CONTROLLER deriva da uno dei primi integrati impiegato dalla Commodore sul vecchio VIC-20. Si differenzia da quello presente sul CBM64, pur mantenendone una piena compatibilita', essenzialmente per il numero dei registri e di conseguenza per le possibilita' che ne derivano.

A grandi linee le sue caratteristiche sono le seguenti:

- Gestione della RAM video
- Generatore di caratteri
- Manipolazione completamente autonoma di 16 K di RAM dinamica
- Gestione dell' INTERRUPT
- Gestione del RASTER
- Visualizzazione e controllo SPRITES
- Gestione del multicolore in modo TESTO
- Generazione e controllo del segnale standard NTSC
- Gestione modo MULTICOLOR 160x200 pixel con 4 colori
- Gestione dei 16 colori

Gestione del modo alta risoluzione 320x200 pixel  
Gestione e controllo della penna luminosa

## **I REGISTRI DEL VIC II**

Vediamone in dettaglio i registri di controllo che sono 49 con indirizzo di partenza \$D000.

REGISTRI DA 0 A 15 Questi registri gestiscono le coordinate, rispettivamente, X e Y degli Sprites dal numero 0 allo Sprite numero 7. Da notare che, per X, il bit 9 o bit di overflow si rileva nel registro 16 sempre del VIC II, mentre per Y non e' necessario il bit di overflow.

REGISTRO 16 Contiene come detto il bit di overflow degli sprites o Most Significant Bit (MSB)

REGISTRO 17 Registro di controllo linee

REGISTRO 18 Controlla il numero di linea al quale dovrebbe essere generato un IRQ raster. Il nono bit della linea di raster e' nel bit 7 del registro 17.

REGISTRO 19 Parte X della posizione di schermo dove viene rilevato il raggio.

REGISTRO 20 Come sopra per Y

REGISTRO 21 E' dedicato al controllo dell' attivazione degli sprites. I bit da 0 a 7 si riferiscono ai relativi sprites che sono attivi quando hanno valore 1.

REGISTRO 22 Registro di controllo 2 per le colonne di schermo.

REGISTRO 23 Come il 21 per l' espansione degli sprites sull' asse X

REGISTRO 24 Contiene nei bits 1-4 gli indirizzi per il generatore di caratteri RAM. Nei bits 4-7 per la RAM video

REGISTRO 25 Indica quale registro genera un INTERRUPT secondo la seguente tabella:

|       |   |                                  |    |
|-------|---|----------------------------------|----|
| Bit 0 | = | registro                         | 18 |
| Bit 1 | = | "                                | 31 |
| Bit 2 | = | "                                | 30 |
| Bit 3 | = | dal penna luminosa su piedino LP |    |

REGISTRO 26 Come il precedente ma per INTERRUPT MASK

REGISTRO 27 Registro di prioritá sullo sfondo per i singoli sprites.

REGISTRO 28 Controllo sui singoli sprites se devono essere in modo multicolore

REGISTRO 29 Come registro 23 ma per l' asse Y

REGISTRO 30 Controlla la collisione fra sprites



REGISTRO 31 Controllo di collisione fra sprites e fondo

REGISTRO 32 Fissa i colori del bordo

REGISTRI DA 33 A 36 Controllano i colori di fondo

REGISTRI 37 E 38 Controllo degli sprites in multicolor

REGISTRI DA 39 A 46 Ognuno di questi registri controlla il colore del singolo Sprite

REGISTRO 47 I bits da 0 a 3 controllano la scansione di tastiera. Gli altri bits sono inutilizzati

REGISTRO 48 Il bit 0 di questo registro determina se si opera a 1 o 2 MHz.

## **IL SOUND INTERFACE DEVICE**

Il SID abbreviazione di Sound Interface Device e' un potente sintetizzatore polifonico che, per quanto abbia costituito uno dei punti di forza del vecchio CBM 64, non e' stato certamente sfruttato per intero nelle sue capacita'. Sul C128 e' riportato pari pari.

Vediamone le caratteristiche principali:

Uscita a 3 voci completamente indipendenti e

liberamente programmabili  
4 tipi di forme d'onda assegnabile e quindi  
programmabile per ogni voce  
3 filtri  
Generatore di inviluppo per ogni voce  
2 ring modulator  
Opzione alternativa per sorgenti sonore  
esterne  
Controllo di due convertitori analogico  
digitali

Gli indirizzi del SID sono posti a partire da  
\$D400 (54272)

## **I REGISTRI SID**

REGISTRO 0 Byte basso dell' oscillatore per  
la voce 1

REGISTRO 1 Byte alto o superiore dell'  
oscillatore per la stessa voce

REGISTRO 2 Controlla l' ampiezza d' impulso  
per la voce 1 (LSB)

REGISTRO 3 Come sopra per MSB. Di questi due  
registri che provvedono al controllo dell'  
onda quadra dei generatori solo i bits 0-3  
sono utilizzati.

REGISTRO 4 Controllo per la voce 1 secondo i  
seguenti valori immagazzinati nei singoli  
bits:

**\*\*Bit 0 (KEY)** controlla il generatore di inviluppo. Quando passa da valore 0 a valore 1 il VOLUME della voce 1 si incrementa da 0 al massimo, contenuto nel REGISTRO 24 con un tempo di ATTACK e di DECAY specificati nel REGISTRO 5 e con un livello di SOSTEGNO di registro 6.

**\*\*Bit 1 (SYNC)** controlla la sincronizzazione fra registro 1 e 3

**\*\*Bit 2 (RING)** controllo per forma a tringolo dell' oscillatore 1 (WAVEFORM)

**\*\*Bit 3 (TEST)** controlla che, in funzione della presenza di altre forme d' onda generate, sia disabilitata il rumore di fondo (NOISE)

**\*\*Bit 4 (TRIA)** seleziona la forma d' onda detta a TRIANGOLO

**\*\*Bit 5 (SAWT)** come sopra per la forma SAWTOOH

**\*\*Bit 6 (PULS)** seleziona la forma d' onda quadra. E' in relazione con i contenuti dei registri 2 e 3

**\*\*Bit 7 (NOI)** seleziona il rumore di fondo o NOISE

NOTA

E' importante notare che e' possibile, con riferimento ai registri 4-7, selezionare forme d' onda diverse allo stesso tempo.

REGISTRO 5 Con i bits 0-3 si determina il tempo necessario perche' il volume si porti dal massimo livello a quello di sostegno (SUSTAIN). E' possibile una scelta di tempi da 6 millisecondi a 24 secondi per questo

tempo.

Con i bits da 4 a 7 invece si seleziona il tempo per portarsi da 0 al massimo livello di volume. L'intervallo va da 2 millisecondi a 8 secondi

REGISTRO 6 Con i bits 0-3 si determina il tempo entro il quale si passa dal livello di sostegno a zero relativamente al volume. L'intervallo di tempo selezionabile va da 6 millisecondi a 24 secondi.

I bits da 4 a 7 specificano invece il volume che deve essere mantenuto (LIVELLO DI SOSTEGNO) dopo che e' stato raggiunto il livello di sostegno e prima che sia abbandonato quel volume.

REGISTRI DA 7 A 13 Questi registri controllano la voce 2 allo stesso modo che i registri visti da 0 a 6 eseguono per la voce 1.

REGISTRI DA 14 A 20 Questi registri controllano la voce 3 allo stesso modo che i registri visti da 0 a 6 eseguono per la voce 1.

REGISTRO 21 Controllo della frequenza di filtraggio. Byte basso

REGISTRO 22 Controllo della frequenza di filtraggio. Byte alto

REGISTRO 23 Controllo di risonanza dei filtri. I bits da 0 a 2 controllano rispettivamente le voci 1-2-3.

Il bit 3 e' rivolto verso il controllo

esterno.

I bits da 4 a 7 controllano invece la frequenza di risonanza dei filtri

REGISTRO 24 controlla il volume e i filtri:

\*\*Bits 0-3 Controllo del volume totale

\*\*Bit 4 Attiva il filtro passabasso

\*\*Bit 5 Attiva il filtro passabanda

\*\*Bit 6 Attiva il filtro passalto

Il bit 7 controlla la voce 3 quando questa e' impiegata come principale rispetto alle altre.

REGISTRI 25 E 26 Controllano rispettivamente i convertitori analogico digitali 1 e 2.

Ricordiamo che un convertitore di questo tipo serve appunto per convertire un segnale analogico, ad esempio un voltaggio, in uno digitale. Come vediamo il SID ne contiene due con un voltaggio di riferimento di 2.5 volts. Vedi HARDWARE per maggiori approfondimenti e, soprattutto per le applicazioni.

REGISTRO 27 Generatore di rumore bianco per la voce 3

REGISTRO 28 Generatore di inviluppo per la voce 3.

### **L' INTEGRATO 8563 VDC**

Al Commodore 128 possono essere collegati 2 Monitor, di cui uno normale che sara' controllato dal VIC II visto in precedenza e

che consente quindi una piena compatibilità con il CBM 64 ed uno ad 80 colonne controllato appunto da questo nuovo integrato che costituisce pertanto una delle maggiori caratteristiche differenziali rispetto al vecchio 64.

L'8563 controlla dunque l'uscita ad 80 colonne, la controlla anche in un monitor RGB a colori e può visualizzare grafici in alta risoluzione su 640x200 punti.

Ne vedremo ora i registri anche se, a nostro modesto parere, per illustrare questo componente ci vorrebbe realmente un manuale dedicato allo scopo.

## **I REGISTRI DEL 8563**

REGISTRO 0 Controlla il numero di caratteri per linea che possono essere un massimo di 126.

REGISTRO 1 Gestisce il numero di caratteri per linea. Il valore di default è 80

REGISTRO 2 Controllo di sincronizzazione sul bordo sinistro linea

REGISTRO 3 Sincronizzazione orizzontale del carattere sullo schermo

REGISTRO 4 Controlla il numero di linee verticali.

REGISTRO 5 Serve per controllare il registro

precedente

REGISTRO 6 Contiene il numero di caratteri rappresentabili.

REGISTRO 7 Definisce il bordo superiore dello schermo. Se il contenuto di questo registro e' incrementato lo schermo si muove verso l'alto. Se e' decrementato verso il basso.

REGISTRO 8 Determina il modo di interlace

REGISTRO 9 Determina il numero di linee raster per carattere in verticale

REGISTRO 10 Determinano il modo di lampeggiamento del cursore.

REGISTRO 11 Controlla la linea alla quale il cursore si fermerà'.

REGISTRO 12 Controlla l'indirizzo di inizio alto della RAM video che e' controllata appunto dal 8563.

Ricordiamo che la RAM video inizia dall'indirizzo \$0000 ma nella speciale memoria del VDC.

REGISTRO 13 Come sopra per Byte basso. Esiste quindi una corrispondenza univoca e necessaria fra questi due registri.

REGISTRO 14 Controlla la posizione del cursore (Byte HI)

REGISTRO 15 Controlla la posizione del cursore (Byte LO)

REGISTRI 16 E 17 Controllano rispettivamente gli indirizzi e quindi le posizioni verticali ed orizzontali della penna luminosa. Questi registri possono essere solo letti, mentre l'aggiustaggio deve essere fatto usando i raster.

REGISTRO 18 Controlla l'aggiornamento dell'indirizzo di memoria della RAM controllata da questo integrato. Indirizzo HI

REGISTRO 19 Come sopra ma per indirizzo LO

REGISTRO 20 Controlla il byte HI dell'indirizzo di partenza memoria.

REGISTRO 21 Opera in connessione con il precedente ma per l'indirizzo LO.

REGISTRO 22 Determina il numero di linee visualizzate orizzontalmente. Definisce anche la larghezza del carattere.

REGISTRO 23 Determina il numero di linee verticali. Definisce l'altezza del singolo carattere.

REGISTRO 24 Controlla lo scorrimento lento verticale.

Il bit 6 e' anche il bit di REVERSE che consente di eseguire appunto il reverse dell'intero schermo.

REGISTRO 25 Come sopra per l'asse x.  
In particolare pero':



- \*\*Bit 7 abilita il modo testo
- \*\*Bit 6 contiene informazioni sul colore del carattere
- \*\*Bit 5 controlla il modo semi-grafico
- \*\*Bit 4 controlla l' ampiezza del carattere visualizzato
- \*\*Bit 0-3 controllano lo SMOOTH scrolling.

#### NOTA

L' uso e l' impiego di questi due registri e' particolarmente complesso ed va al di la' degli obbiettivi di questo volume.

REGISTRO 26 Controlla i colori ed i modi dell' interno e dell' esterno dello schermo.

REGISTRO 27 Controlla il numero di Bytes che devono essere aggiunti alla RAM video per ogni colonna.

Se si desidera ridefinire un carattere in ampiezza i valori contenuti in questo registro devono essere manipolati via SOFTWARE.

REGISTRO 28 I bits 5-7 determinano la base del generatore di caratteri.

Il bit 4 definisce il tipo di RAM impiegata (1=4164: 0=4416).

Gli altri bit non sono al momento utilizzati.

REGISTRO 29 Controlla ed indica la sottolineatura di una data linea.

Puo' anche essere usato per la sovrالinea.

REGISTRO 30 Contiene il numero di caratteri che servono per aggiornare l' indirizzo di

COPY del contenuto RAM del VDC quando si eseguono degli spostamenti.

REGISTRO 31 Contiene i dati che devono essere scritti in una locazione di memoria.

Se viene letta una locazione di memoria il relativo contenuto appare in questo registro.

REGISTRO 32 Definisce il l' indirizzo di partenza del blocco di memoria che` deve essere copiato. Parte HI.

REGISTRO 33 Come sopra, ma per la parte LO dell' indirizzo.

REGISTRO 34 Riporta il numero di caratteri dall' inizio della linea che deve essere visualizzata.

REGISTRO 35 Come il precedente

REGISTRO 36 Specificano il refresh della memoria del VDC.

Il valore normale e' 245. qualsiasi modifica deve essere fatta anche in funzione del tipo di RAM impiegata e comunque con molta accuratezza per evitare perdite di dati di cui si potrebbe accorgersi solo con molto ritardo.

### **COMPLEX INTERFACE ADAPTER O CIA**

Come dice il nome stesso si tratta di interfacce verso il mondo esterno.  
Vediamone le caratteristiche:

Manipolazione dei registri di 8 bit per uscite seriali di I/O  
Controllo dei due timers di 16 bit completamente indipendente  
Controllo dell' orologio con temporizzatore programmabile  
Controllo delle 16 linee di Input/Output sepratamente programmabili  
Controllo dell' HANDSHAKE per I/O su 8 o 16 bit.

A differenza di quanto scritto fino a questo momento per gli altri integrati qui daremo anche il tipo di accesso per ogni registro oltre alla sigla attribuita dal costruttore per ogni singolo registro.

## **DESCRIZIONE DEI REGISTRI CIA**

REGISTRO 0    PRA  
Accesso: lettura scrittura

Controlla la porta A corrispondente ai pin PA0-PA7

REGISTRO 1    PRB  
Accesso: lettura scrittura

Controlla la porta B corrispondente ai pin PB0-PB7

REGISTRO 2    DDRA  
Accesso: lettura scrittura

Determina la direzione dei dati per i corrispondenti bits della porta A.  
Abbreviazione di DATA DIRECTION REGISTER A

REGISTRO 3    DDRB  
Accesso: lettura scrittura

Determina la direzione dei dati per i corrispondenti bits della porta B.  
Abbreviazione di DATA DIRECTION REGISTER B

REGISTRO 4    TA LO  
Accesso: lettura scrittura

Quando e' in lettura restituisce l' attuale stato del byte di ordine LO del temporizzatore A.

Quando e' in scrittura viene caricato con il byte LO del valore da cui il timer deve essere riportato a zero. In pratica e' un contatore per il timer.

TA LO sta per TIMER A LOW BYTE

REGISTRO 5    TA HI  
Accesso: lettura scrittura

Quando e' in lettura restituisce l' attuale stato del byte di ordine HI del temporizzatore A.

Quando e' in scrittura viene caricato con il byte LO del valore da cui il timer deve essere riportato a zero. In pratica e' un contatore per il timer.

TA HI sta per TIMER A HIGH BYTE

REGISTRO 6    TB LO  
Accesso: lettura scrittura

Quando e' in lettura restituisce l' attuale stato del byte di ordine LO del temporizzatore B.

Quando e' in scrittura viene caricato con il byte LO del valore da cui il timer deve essere riportato a zero. In pratica e' un contatore per il timer.

TB LO sta per TIMER B LOW BYTE

REGISTRO 7      TB HI

Accesso: lettura scrittura

Quando e' in lettura restituisce l' attuale stato del byte di ordine HI del temporizzatore B.

Quando e' in scrittura viene caricato con il byte HI del valore da cui il timer deve essere riportato a zero. In pratica e' un contatore per il timer.

TB HI sta per TIMER B HIGH BYTE

REGISTRO 8      TOD 10

Accesso: lettura scrittura

Controllo del clock per i decimi di secondo

REGISTRO 9      TOD SEC

Accesso: lettura

Controllo clock per secondi

REGISTRO 10      TOD MIN

Accesso: lettura

Controllo clock per minuti. E' previsto in scrittura per controllo registro 8.

REGISTRO 11    TOD HR

Accesso: lettura

Controllo clock per ore.

Anche questo puo' essere operativo in scrittura per controllo registro 8.

REGISTRO 12 SDR

Accesso: lettura scrittura

Registro seriale di dati.    Il colloquio avviene tramite il pin SP.

REGISTRO 13    ICR

Accesso: lettura scrittura

In lettura i bit da 0 a 7 controllano rispettivamente i timeout dei timer A e B, il clock dell'allarme, l' SDR del precedente registro, il segnale sul PIN FLAG.

Mentre i bits 5 e 6 sono sempre =0 il bit 7 controlla l' INTERNAL MASK.

In fase di scrittura e' del tutto uguale, cioe' ha le stesse funzioni opposte a quelle della lettura.

Per il BIT 7 invece ha la funzione di fissare il valore dei bits di mask che saranno a 0 quando questi e' =0 e quindi a 1 quando il bit 7 viene posto a 1.

L' abbreviazione ICR sta per INTERRUPT CONTROL REGISTER.

REGISTRO 14    CRA

Accesso: Lettura scrittura

E' il registro di controllo A. I singoli bits

necessitano di una spiegazione piu approfondita.

**\*\*Bit 0** se 1 Timer A start se 0 stop  
**\*\*Bit 1** se 1 segnale di TIMEOUT per timer A su pin PB6  
**\*\*Bit 2** se 1 ogni TIMEOUT su timer A inverte PB6. Se a 0 allora ogni TIMEOUT su PB6 invia un segnale alto, sempre su PB6 per la durata del clock di sistema.  
**\*\*Bit 3** se 1 il Timer A esegue un conto alla rovescia fino a zero e si ferma. Se a 0 esegue un conto alla rovescia e continua.  
**\*\*Bit 4** se 1 esegue un caricamento assoluto delvalore iniziale sul timer A. Deve essere appunto posto = 1 (normalmente e7 infatti =0) per ogni caricamento assoluto.  
Ha funzione di STROBE.  
**\*\*Bit 5** determina il source del trigger  
**\*\*Bit 6** se 1 allora SP e' in uscita. Se 0 allora SP e' in ingresso  
**\*\*Bit 7** controlla il clock del RT  
Se 1 e' 50 Hz. Se 0 e' 60 Hz

REGISTRO 15    CRB  
Accesso: lettura scrittura

Stesse funzioni viste nel registro precedente per B.  
Sara' interessato in questo caso il pin PB7 invece del PB6

Di seguito l'elenco delle parole chiave del Basic 7.0, cioe' i comandi, delle relative abbreviazioni con le quali possono essere scritte ed infine dei valori TOKENIZZATI con i quali vengono gestiti dall'interprete.

| <u>COMANDO</u> | <u>ABBREVIAZIONE</u> |
|----------------|----------------------|
| ABS            | A shift B            |
| APPEND         | A shift P            |
| ASC            | A shift S            |
| ATN            | A shift T            |
| AUTO           | A shift U            |
| BACKUP         | BA shift C           |
| BANK           | B shift A            |
| BEGIN          | B shift E            |
| BEND           | BE shift N           |
| BLOAD          | B shift L            |
| BOOT           | B shift O            |
| BOX            | nessuna              |
| BSAVE          | B shift S            |
| BUMP           | B shift U            |
| CATALOG        | C shift A            |
| CHAR           | CH shift A           |
| CHR\$          | C shift H            |
| CIRCLE         | C shift I            |
| CLOSE          | CL shift O           |
| CLR            | C shift L            |
| CMD            | C shift M            |
| COLLECT        | COLL shift E         |
| COLINT         | nessuna              |
| COLLISION      | COL shift L          |
| COLOR          | COL shift O          |
| CONCAT         | C shift O            |



| <u>COMANDO</u> | <u>ABBREVIAZIONE</u> |
|----------------|----------------------|
| CONT           | nessuna              |
| COPY           | CO shift P           |
| COS            | nessuna              |
| DATA           | D shift A            |
| DEC            | nessuna              |
| DCLEAR         | DCL shift E          |
| DCLOSE         | D shift C            |
| DEF FN         | nessuna              |
| DELETE         | DE shift L           |
| DIM            | D shift I            |
| DIRECTORY      | DI shift R           |
| DLOAD          | D shift L            |
| DO             | nessuna              |
| DOPEN          | D shift O            |
| DRAW           | D shift R            |
| DSAVE          | D shift S            |
| DVERIFY        | D shift V            |
| EL             | nessuna              |
| END            | nessuna              |
| ENVELOPE       | E shift N            |
| ER             | nessuna              |
| ERR\$          | E shift R            |
| EXIT           | EX shift I           |
| EXP            | E shift X            |
| FAST           | nessuna              |
| FETCH          | F shift E            |
| FILTER         | F shift I            |
| FOR            | F shift O            |
| FRE            | F shift R            |
| FNXX           | nessuna              |
| GET            | G shift E            |
| GETKEY         | GETK shift E         |
| GET#           | nessuna              |
| GOSUB          | GO shift S           |

| <u>COMANDO</u>   | <u>ABBREVIAZIONE</u> |
|------------------|----------------------|
| GO64             | nessuna              |
| GOTO             | G shift O            |
| GRAPHIC          | G shift R            |
| GSHAPE           | G shift S            |
| HEADER           | HE shift A           |
| HELP             |                      |
| HEX\$            | H shift E            |
| IF...GOTO        | nessuna              |
| IF...THEN...ELSE | nessuna              |
| INPUT            | nessuna              |
| INPUT #          | I shift N            |
| INSTR            | IN shift S           |
| INT              | nessuna              |
| JOY              | J shift O            |
| KEY              | K shift E            |
| LEFT\$           | LE shift F           |
| LEN              | nessuna              |
| LET              | L shift E            |
| LIST             | L shift I            |
| LOAD             | L shift O            |
| LOCATE           | LO shift C           |
| LOG              | nessuna              |
| LOOP             | LO shift O           |
| MID\$            | M shift I            |
| MONITOR          | MO shift N           |
| MOVESHAPE        | nessuna              |
| MOVSPR           | M shift O            |
| NEW              | nessuna              |
| NEXT             | N shift E            |
| ON...GOSUB       | ON...GO shift S      |
| ON...GOTO        | ON...G shift O       |
| OPEN             | O shift P            |
| PAINT            | P shift A            |
| PEEK             | PE shift E           |
| PEN              | P shift E            |

| <u>COMANDO</u> | <u>ABBREVIAZIONE</u> |
|----------------|----------------------|
| PI             | nessuna              |
| PLAY           | P shift L            |
| POKE           | PO shift K           |
| POS            | nessuna              |
| POT            | P shift O            |
| PRINT          | ?                    |
| PRINT#         | P shift R            |
| PRINT USING    | ?US shift I          |
| PUDEF          | P shift U            |
| RBUMP          | RB shift U           |
| RCLR           | R shift C            |
| RDOT           | R shift D            |
| READ           | RE shift A           |
| RECORD         | R shift E            |
| REM            | nessuna              |
| RENAME         | RE shift N           |
| RENUMBER       | REN shift U          |
| RESTORE        | RE shift S           |
| RESUME         | RES shift U          |
| RETURN         | RE shift T           |
| RGR            | R shift G            |
| RIGHT\$        | R shift I            |
| RLUM           | nessuna              |
| RND            | R shift N            |
| RREG           | R shift R            |
| RSPCOLOR       | RSP shift C          |
| RSPPOS         | R shift S            |
| RSPR           | nessuna              |
| RSPRITE        | RSP shift R          |
| RUN            | R shift U            |
| RWINDOW        | R shift W            |
| SAVE           | S shift A            |
| SCALE          | SC shift A           |
| SCNCLR         | S shift C            |

| <u>COMANDO</u> | <u>ABBREVIAZIONE</u> |
|----------------|----------------------|
| SCRATCH        | SC shift R           |
| SGN            | S shift G            |
| SIN            | S shift I            |
| SLEEP          | S shift L            |
| SLOW           | nessuna              |
| SOUND          | S shift O            |
| SPC(           | nessuna              |
| SPRCOLOR       | SPR shift C          |
| SPREDEF        | SPR shift D          |
| SPRITE         | S shift P            |
| SPRSAY         | SPR shift S          |
| SQR            | S shift Q            |
| SSHAPE         | S shift S            |
| STASH          | S shift T            |
| STATUS         | nessuna              |
| STEP           | ST shift E           |
| STOP           | ST shift O           |
| STR\$          | ST shift R           |
| SWAP           | S shift W            |
| SYS            | nessuna              |
| TAB(           | T shift A            |
| TAN            | nessuna              |
| TEMPO          | T shift E            |
| TI             | nessuna              |
| TI\$           | nessuna              |
| TO             | nessuna              |
| TRAP           | T shift R            |
| TROFF          | TRO shift F          |
| TRON           | TR shift O           |
| UNTIL          | U shift N            |
| USR            | U shift S            |
| VAL            | nessuna              |
| VERIFY         | V shift E            |
| VOL            | V shift O            |

| <u>COMANDO</u> | <u>ABBREVIAZIONE</u> |
|----------------|----------------------|
| WAIT           | W shift A            |
| WHILE          | W shift H            |
| WIDTH          | WI shift D           |
| WINDOW         | W shift I            |
| XOR            | X shift O            |



|          |           |           |           |
|----------|-----------|-----------|-----------|
| LEFT\$   | \$C8      |           |           |
| RIGHT\$  | \$C9      |           |           |
| MID\$    | \$CA      |           |           |
| GO       | \$CB      |           |           |
| RGR      | \$CC      |           |           |
| RCLR     | \$CD      |           |           |
| POT      | \$CE \$02 |           |           |
| BUMP     | \$CE \$03 |           |           |
| PEN      | \$CE \$04 |           |           |
| RSPPPOS  | \$CE \$05 |           |           |
| RSPRITE  | \$CE \$06 |           |           |
| RSPCOLOR | \$CE \$07 |           |           |
| XOR      | \$CE \$08 | DRAW      | \$E5      |
| RWINDOW  | \$CE \$09 | LOCATE    | \$E6      |
| POINTER  | \$CE \$0A | COLOR     | \$E7      |
| JOY      | \$CF      | SCNCLR    | \$E8      |
| RDOT     | \$D0      | SCALE     | \$E9      |
| DEC      | \$D1      | HELP      | \$EA      |
| HEX\$    | \$D2      | DO        | \$EB      |
| ERR\$    | \$D3      | LOOP      | \$EC      |
| INSTR    | \$D4      | EXIT      | \$ED      |
| ELSE     | \$D5      | DIRECTORY | \$EE      |
| RESUME   | \$D6      | DSAVE     | \$EF      |
| TRAP     | \$D7      | DLOAD     | \$F0      |
| TRON     | \$D8      | HEADER    | \$F1      |
| TROFF    | \$D9      | SCRATCH   | \$F2      |
| SOUND    | \$DA      | COLLECT   | \$F3      |
| VOL      | \$DB      | COPY      | \$F4      |
| AUTO     | \$DC      | RENAME    | \$F5      |
| PUDEF    | \$DD      | BACKUP    | \$F6      |
| GRAPHIC  | \$DE      | DELETE    | \$F7      |
| PAINT    | \$DF      | RENUMBER  | \$F8      |
| CHAR     | \$E0      | KEY       | \$F9      |
| BOX      | \$E1      | MONITOR   | \$FA      |
| CIRCLE   | \$E2      | USING     | \$FB      |
| GSHAPE   | \$E3      | UNTIL     | \$FC      |
| SSHAPE   | \$E4      | WHILE     | \$FD      |
|          |           | BANK      | \$FE \$02 |
|          |           | FILTER    | \$FE \$03 |
|          |           | PLAY      | \$FE \$04 |
|          |           | TEMPO     | \$FE \$05 |
|          |           | MOVSPR    | \$FE \$06 |
|          |           | SPRITE    | \$FE \$07 |
|          |           | SPRCOLOR  | \$FE \$08 |
|          |           | RREG      | \$FE \$09 |
|          |           | ENVELOPE  | \$FE \$0A |
|          |           | SLEEP     | \$FE \$0B |
|          |           | CATALOG   | \$FE \$0C |
|          |           | DOPEN     | \$FE \$0D |
|          |           | APPEND    | \$FE \$0E |
|          |           | DCLOSE    | \$FE \$0F |
|          |           | BSAVE     | \$FE \$10 |
|          |           | BLOAD     | \$FE \$11 |
|          |           | RECORD    | \$FE \$12 |
|          |           | CONCAT    | \$FE \$13 |
|          |           | DVERIFY   | \$FE \$14 |
|          |           | DCLEAR    | \$FE \$15 |
|          |           | SPRSV     | \$FE \$16 |
|          |           | COLLISION | \$FE \$17 |
|          |           | BEGIN     | \$FE \$18 |
|          |           | BEND      | \$FE \$19 |
|          |           | WINDOW    | \$FE \$1A |
|          |           | BOOT      | \$FE \$1B |
|          |           | WIDTH     | \$FE \$1C |
|          |           | SPRDEF    | \$FE \$1D |
|          |           | QUIT      | \$FE \$1E |
|          |           | STASH     | \$FE \$1F |
|          |           | FETCH     | \$FE \$21 |
|          |           | STASH     | \$FE \$23 |
|          |           | OFF       | \$FE \$24 |
|          |           | FAST      | \$FE \$25 |
|          |           | SLOW      | \$FE \$26 |

La tabella precedente mostra i valori sonori dei registri di quattro ottave di note. I valori sonori sono usati come secondo parametro del comando SOUND. Per utilizzare la prima nota nella tabella (A - valore sonoro dei registri 7) utilizzare il 7 come secondo numero dopo il comando SOUND - SOUND 1,7,30.

Per trovare i valori sonori dei registri per frequenze non comprese nella tabella utilizzare la formula seguente:

$$\text{VALORE SONORO DEI REGISTRI} = 1024 - (111860.781/\text{FREQUENZA})$$

Sia la tabella dei valori sonori dei registri sia la formula sono applicati a televisori NTSC, il sistema standard utilizzato negli Stati Uniti e in Canada. In paesi che utilizzano il sistema PAL, calcolare i valori sonori dei registri con l'aiuto della formula seguente:

$$\text{VALORE SONORO DEI REGISTRI} = 1024 - (111840.45/\text{FREQUENZA})$$

#### **TAVOLA DELLE NOTE MUSICALI**

| NOTA  | VALORI SONORI<br>DEI REGISTRI | FREQUENZA<br>EFFETTIVA (HZ) |
|-------|-------------------------------|-----------------------------|
| A la  | 7                             | 110                         |
| B si  | 118                           | 123,5                       |
| C do  | 169                           | 130,8                       |
| D re  | 262                           | 146,8                       |
| E mi  | 345                           | 164,7                       |
| F fa  | 383                           | 174,5                       |
| G sol | 453                           | 195,9                       |
| A la  | 516                           | 220,2                       |
| B si  | 571                           | 246,9                       |
| C do  | 596                           | 261,4                       |
| D re  | 643                           | 293,6                       |
| E mi  | 685                           | 330                         |
| F fa  | 704                           | 349,6                       |
| G sol | 739                           | 392,5                       |
| A la  | 770                           | 440,4                       |
| B si  | 798                           | 494,9                       |
| C do  | 810                           | 522,7                       |
| D re  | 834                           | 588,7                       |
| E mi  | 854                           | 658                         |
| F fa  | 864                           | 699                         |
| G sol | 881                           | 782,2                       |
| A la  | 897                           | 880,7                       |
| B si  | 911                           | 989,9                       |
| C do  | 917                           | 1045                        |
| D re  | 929                           | 1177                        |
| E mi  | 939                           | 1316                        |
| F fa  | 944                           | 1398                        |
| G sol | 953                           | 1575                        |

LO Z 80

Il fatto che sia presente su questo computer ANCHE un microprocessore Z-80, corredato dal relativo software (ma solo per chi dispone del disco), permette di fargirare programmi scritti sotto CPM che e' un sistema operativo di vecchia data, ma sotto il quale i programmi scritti si contano a migliaia.

Sul C128 e' presente un integrato ROM che contiene circa 4K di codice Z80. In questo manuale non presentiamo il disassemblato completo di queste locazioni di memoria ma solo la prima parte cioe' quella relativa al BOOT. E cio' perche' le informazioni contenute nella parte restante dell'integrato non possono essere variate ed anche perche' non hanno realmente molto a che vedere con il CPM stesso. All'accensione o RESET i valori di questa ROM di indirizzo \$D000 sono riportati a inizio pagina zero (\$0000). Il computer eseguirà quindi tutta una serie di controlli come quello sui tasti, sulla presenza o meno del cartridge ecc. Dopo viene abilitato lo Z-80 e va a cercare se esiste il disco CPM. In caso affermativo parte sotto CPM altrimenti sotto 8502.

**LA ROM DELLO Z 80**

|       |          |     |               |                                      |
|-------|----------|-----|---------------|--------------------------------------|
| 0000: | 3E 3E    | LD  | A,\$3E        | Bytes di configurazione nei registri |
| 0002: | 32 00 FF | LD  | (\$FF00),A    | di configurazione per Partenza (COLD |
| 0005: | C3 3B 00 | JP  | \$003B        | START                                |
| 0008: | 31 77 3C | LD  | SP,\$3C77     | Routine RTS80                        |
| 000B: | 3E 3F    | LD  | A,\$3F        | ***                                  |
| 000D: | C3 8C 01 | JP  | \$018C        | ***                                  |
| 0010: | E1       | POP | HL            | Riporta indirizzo dallo stack        |
| 0011: | 6E       | LD  | L,(HL)        | Byte basso di indirizzo di ritorno   |
| 0012: | C3 20 00 | JP  | \$0020        | Salto a routine RST 20               |
| 0015: | 00       | NOP |               | Riempimento di Bytes                 |
| 0016: | 00       | NOP |               | ***                                  |
| 0017: | 00       | NOP |               | ***                                  |
| 0018: | E1       | POP | HL            | Come in precedenza ma con salto      |
| 0019: | 6E       | LD  | L,(HL)        | a RST 28                             |
| 001A: | C3 28 00 | JP  | \$0028        | ***                                  |
| 001D: | 00       | NOP |               | Vedi sopra                           |
| 001E: | 00       | NOP |               | ***                                  |
| 001F: | 00       | NOP |               | ***                                  |
| 0020: | 3A 0F FD | LD  | A,(\$FDOF)    | Routine 20                           |
| 0023: | A7       | AND | A             | ***                                  |
| 0024: | 28 02    | JR  | Z,\$+4>\$0028 | ***                                  |
| 0026: | 2C       | INC | L             | ***                                  |
| 0027: | 2C       | INC | L             | ***                                  |
| 0028: | 26 01    | LD  | H,\$01        | Routine 28                           |
| 002A: | 7E       | LD  | A,(HL)        |                                      |



```

002B: 23      INC HL
002C: 66      LD H,(HL)
002D: 6F      LD L,A
002E: E9      JP (HL)
002F: 00      NOP

0030: 30 35    JR NC,$+55>$0067      Routine 30
0032: 2F      CPL
0033: 31 32 2F LD SP,$2F32
0036: 38 35    JR C,$+55>$006D

0038: C3 FD FD JP $FDFD              Routine 38

003B: 01 2F D0 LD BC,$D02F          Vai a reg. 47 del VIC II
003E: 11 FC FF LD DE,$FFFC          Scrittura di $ff su tastiera
0041: ED 51    OUT (C),D             Nessuna estensione
0043: 03      INC BC                 Registro 48
0044: ED 59    OUT (C),E             Fissa a 1 MHz
0046: 01 05 D5 LD BC,$D505          Registro di configurazione modo
0049: 3E B0    LD A,$B0              Controllo per EXMON e porta giochi
004B: ED 79    OUT (C),A             Abilita modo 128
004D: ED 78    IN A,(C)              Registro configurazione modo
004F: 2F      CPL                  Rileggi
0050: E6 30    AND $30               Controllo per EXROM e giochi
0052: 28 05    JR Z,$+7>$0059        Se negativo nessun cartridge
-----

0054: 3E F1    LD A,$F1              Abilita l'8502 e seleziona il modo
0056: ED 79    OUT (C),A              operativo 64
0058: C7      RST $00                Esegui cold start
0059: 01 0F DC LD BC,$DC0F          Seleziona CRB in CIA 1
005C: 3E 08    LD A,$08              Esegui uno stop
005E: ED 79    OUT (C),A              Come sopra per Timer B
0060: 0D      DEC C                  ***
0061: ED 79    OUT (C),A              ***
0063: 0E 03    LD C,$03              Fissa tutti i bits del registro direz
0065: AF      XOR A                  dati della porta B come Input
0066: ED 79    OUT (C),A              ***
0068: 0D      DEC C                  Come sopra per la porta A ma in
0069: 3D      DEC A                  output
006A: ED 79    OUT (C),A              ***
006C: 0D      DEC C                  Decrementa BC per puntare su porta
006D: 0D      DEC C                  A
006E: 3E 7F    LD A,$7F              Invia $7F alla porta A
0070: ED 79    OU (C),A              ***
0072: 03      INC BC                 Puntatore sulla porta B come input
0073: ED 78    IN A,(C)              e lettura
0075: E6 20    AND $20               Esegui Mask out del tasto CBM
0077: 01 05 D5 LD BC,$D505          Puntatore per modo config. registri
007A: 28 D8     JR Z,$-38>$0054       Controllo tasto CBM
007C: 21 B4 0F LD HL,$0FB4          Questa routine serve per caricare i
007F: 01 0A D5 LD BC,$D50A          registri di MMU con i valori presenti
0082: 16 0B     LD D,$0B              all' ind. $0faa
0084: 7E      LD A,(HL)
0085: ED 79    OUT (C),A
0087: 2B      DEC HL
0088: 0D      DEC C
0089: 15      DEC D
008A: 20 F8     JR NZ,$-6 >$0084      Fine del ciclo precedente
008C: 21 1A 0D LD HL,$0D1A          Copia l' area da $0d1a

```

|       |          |      |             |                                     |
|-------|----------|------|-------------|-------------------------------------|
| 008F: | 11 00 11 | LD   | DE,\$1100   | a \$1100                            |
| 0092: | 01 08 00 | LD   | BD,\$0008   | Copia 8 bytes                       |
| 0095: | ED B0    | LDIR |             |                                     |
| 0097: | 21 E5 0E | LD   | HL,\$0EE5   | Copia 1' area da \$0ee5             |
| 009A: | 11 D0 FF | LD   | DE,\$FFD0   | a \$ffd0                            |
| 009D: | 01 1F 00 | LD   | BC,\$001F   | ***                                 |
| 00A0: | ED B0    | LDIR |             | Copia 31 Bytes                      |
| 00A2: | 21 00 11 | LD   | HL,\$1100   | Vettore di salto a \$1100           |
| 00A5: | 22 FA FF | LD   | (\$FFFA),HL | Copia i vettori di salto in tutti e |
| 00A8: | 22 FC FF | LD   | (\$FFFC),HL | 4 gli indirizzi                     |
| 00AB: | 22 FE FF | LD   | (\$FFFE),HL |                                     |
| 00AE: | 22 DD FF | LD   | (\$FFDD),HL |                                     |
| 00B1: | C3 E0 FF | JP   | \$FFE0      | Vai a controllare lo Z80            |
| ----- |          |      |             |                                     |
| 0EE5: | 78       | SEI  |             | Disabilita gli interrupts           |
| 0EE6: | A9 3E    | LDA  | #\$3E       | Indice di configurazione            |
| 0EE8: | 8D 00 FF | STA  | \$FF00      | Fissa il precedente                 |
| 0EEB: | A9 B0    | LDA  | #\$B0       | Abilita lo Z80                      |
| 0EED: | 8D 05 D5 | STA  | \$D505      | Scrivi q.s. sul reg. config. modo   |
| 0EF0: | EA       | NOP  |             | Ritardo                             |
| 0EF1: | 4C 00 30 | JMP  | \$3000      | Continua                            |
| 0EF4: | EA       | NOP  |             | Ritardo                             |
| ----- |          |      |             |                                     |
| 0EF5: | F3       | DI   |             | Disabilita gli interrupt            |
| 0EF6: | 3E 3E    | LDA  | #\$3E       | Carica 17 indice di configurazione  |
| 0EF8: | 32 00 FF | STA  | \$FF00      | e mettilo nel registro              |
| 0EFB: | 01 05 D5 | LD   | BC,\$D505   | Registro di config.                 |
| 0EFE: | 3E B1    | LD   | A,\$B1      | Abilita 8502                        |
| 0F00: | ED 79    | OUT  | (C),A       | Registro di config. modo            |
| 0F02: | 00       | NOP  |             | Ritardo per ciclo                   |
| 0F03: | CF       | RST  | \$08        | Continua                            |

## INDICE

|                               |        |
|-------------------------------|--------|
| Introduzione                  | pag. 1 |
| Il Sistema operativo          | " 8    |
| La pagina ZERO                | " 130  |
| Il disassemblato di Pag.0     | " 131  |
| La pagina UNO e le altre      | " 135  |
| Tavole di vettori             | " 136  |
| Successive locazioni          | " 138  |
| Gestione dell' RS-232         | " 138  |
| Variabili di schermo          | " 139  |
| Il buffer di cassetta         | " 139  |
| La grafica                    | " 139  |
| La musica                     | " 139  |
| Gli interrupt                 | " 140  |
| Le routines KERNAL            | " 141  |
| Descrizione delle routines    | " 144  |
| <u>MEMORY MANAGEMENT UNIT</u> | " 175  |
| Organizzazione della memoria  | " 177  |
| Mappa dei registri            | " 177  |
| La ROM del 128                | " 178  |
| Organizzazione RAM            | " 179  |
| I registri di configurazione  | " 182  |
| I registri di modo            | " 184  |
| I puntatori per le pagine     | " 186  |
| I registri di sistema         | " 188  |

APPENDICI

pag. 189

|                            |   |     |
|----------------------------|---|-----|
| L' integrato VIC II        | " | 192 |
| I registri del VIC II      | " | 193 |
| Il SID                     | " | 195 |
| I registri del SID         | " | 196 |
| L' integrato 8563 VDC      | " | 199 |
| I registri del 8563        | " | 200 |
| Il CIA                     | " | 204 |
| I registri del CIA         | " | 205 |
| Le parole chiave del Basic | " | 210 |
| I TOKEN                    | " | 215 |
| Le note musicali           | " | 217 |
| Frequenze e valori         | " | 218 |

LO Z 80

|                  |   |     |
|------------------|---|-----|
| Lo z80           | " | 218 |
| La ROM dello Z80 | " | 218 |

CARTOLINA DA RISPEDIRE A EVM

☐ Desidero ricevere GRATUITAMENTE il VS. Catalogo.

\* ☐ Desidero ricevere lo speciale CATALOGO MANUALI EVM per acquisti a prezzi scontati.

Sono in possesso di un computer.....  
con le seguenti periferiche.....  
.....  
.....

SEGNALAZIONE DI ERRORI.....  
.....  
.....

\* Riservato ESCLUSIVAMENTE ai possessori di questo manuale.

COGNOME.....NOME.....  
VIA.....CAP.....  
CITTA'.....







# IL SISTEMA OPERATIVO DEL **commodore 128**

Per la prima volta è disponibile in ITALIANO un manuale che rivela i segreti del C 128.

Per utilizzare **COMPLETAMENTE** le capacità operative di un computer è **INDISPENSABILE** conoscere il Sistema Operativo, cuore stesso di tutto il funzionamento.

La notevole documentazione contenuta in questo manuale consente a **TUTTI** di operare per ottenere il meglio del C 128. I contenuti:

- Il Sistema Operativo con commenti in italiano e con le LABEL di riferimento.
- Le pagine 0-1-2-3-4 disassemblate e commentate.
- Il BOOT dello Z-80 (per CPM) disassemblato e commentato.
- Capitolo sulla Memory Management Unit per il controllo del sistema e della memoria che è espandibile fino a 1 Mega byte.
- I registri e le funzioni degli integrati SID, CIA, 8563 VCD, VIC II.
- Tavole di riferimento.

L. 38.000

(iva compresa)



**NEW EVM COMPUTER**  
**via degl'Innocenti 2 Figline Valdarno**  
**Tel.055-958382/958383**





# IL SISTEMA OPERATIVO DEL CONDORE 128