VIC da zero +64

a cura di Tommaso Pantuso

Flashtape per C64 (seconda parte)

di Alessandro Guida e Gianni Itta

Eccoci con la seconda parte dell'articolo in cui viene descritto come velocizzare il registratore a cassette del Commodore 64.

Prima di lasciarvi alla lettura, vogliamo informarvi su alcuni eventi riguardanti il flashtape. Nel corso del mese ci siamo accorti che il programma di velocizzazione, di cui abbiamo cominciato a pubblicare il disassemblato sin dalla volta scorsa, poteva essere migliorato.

Ci siamo quindi messi all'opera ed il risultato è stato una versione con prestazioni superiori rispetto alla release originale. Naturalmente, le migliorie introdotte hanno portato ad una modifica del programma e quindi, per semplicità, abbiamo preferito riproporvi il listato nella sua completezza. In altre parole, la parte pubblicata la volta scorsa non va tenuta in considerazione.

Per quanto riguarda la prima parte dell'articolo, essa non presenta imprecisioni concettuali. L'unica importante differenza su cui vogliamo soffermarci riguarda la codifica dei bit 0 ed 1 da parte del Flashtape che ora è rappresentata da forme d'onda simmetriche, a differenza di quanto avveniva prima. Ciò porta sostanzialmente alla modifica della figura 4 riportata nell'articolo scorso che questa volta assumerà l'aspetto di due onde quadre simmetriche, rispettivamente con semiperiodo di 80 microsecondi (bit 0) e 135 microsecondi (bit 1). Per il resto, a parte qualche valore di benchmark, i concetti già esposti sono corretti.

T.P.

Il listato Assembly

Definiti tutti gli aspetti del nostro velocizzatore possiamo passare ad analizzarne il listato in Assembly.

Cominciamo dal fondo! L'ultima parte del listato presenta delle istruzioni memorizzate in pagina due a partire da $02A7. Queste sono delle routine che interfacciano il programma principale di flashsave con il Basic. Ciò è necessario poiché la maggior parte del codice risiede nella RAM da $E000 in poi e richiede quindi la disabilitazione di tutta la ROM per poter essere letto ed eseguito. È necessaria, perciò, una gestione del passaggio da RAM a ROM e viceversa, esterna a questa zona di memoria.

In figura 1 vengono riportati i registri utilizzati per il colloquio con il registratore e per la gestione della memoria.

L'inizio della routine in $02A7 effettua il controllo del numero di periferica. Se si tratta del registratore (#1) abilita la RAM in $E000 e passa al controllo della sintassi del comando memorizzato dal sistema operativo nel buffer di input a partire da $0200.

La nuova routine di save permette la sintassi originale:

```
SAVE «filename»[,1]
```

che, però, non consente la registrazione di programmi più lunghi di 38K a meno di ometterne il nome.

Abbiamo comunque la possibilità di un'altra sintassi:

```
SAVE: filename
SAVE: [filename],R
SAVE: [filename],S indirizzo
```

Il primo comando è uguale al SAVE tradizionale ma ci permette di dare un nome anche ai programmi maggiori di 38K. Nel ricaricare il programma da nastro al termine del Load avremo il READY.

La seconda sintassi permette l'esecuzione automatica del programma con un RUN.

L'ultima possibilità è quella di avere l'esecuzione automatica con un SYS all'indirizzo specificato. Ad esempio, «SAVE:TEST,S49152» salva il programma col nome TEST in modo che dopo il LOAD venga eseguito con un SYS49152. Inutile aggiungere che questa opzione si utilizza di solito con i programmi in linguaggio macchina. È anche possibile specificare solo il tipo di autostart omettendo il nome del programma.

L'analisi della sintassi procede controllando la presenza del nome, nel qual caso ne è calcolata la lunghezza che viene conservata in NAMLEN ($b7); il valore #$00 indica l'assenza del nome. Viene poi verificata la richiesta di autostart ed aggiornato di conseguenza il flag SRFLAG. Questo contiene per default #$00, ossia la mancanza dell'autostart (il load termina con il READY).

In caso contrario viene posto a #$01 per avere il RUN o a #$80 per il SYS. In caso di SYS viene letto l'indirizzo e conservato in SYSADR. Se viene riscontrata qualche inesattezza nella sintassi il programma ritorna alla pagina 2 per riabilitare la ROM e segnalare l'errore.

A questo punto ($E06C) comincia la registrazione della testata standard: la gestione del tasto PLAY sul registratore e del messaggio «SAVING...» avviene attraverso delle routine del sistema operativo. Per questo motivo sarà nostra cura ricopiare il sistema operativo sulla RAM prima di copiarvici le routine del flashtape.

Si prosegue creando nel buffer cassetta l'header da registrare prima del programma. Dapprima viene ripulito il buffer poi vengono ricopiati i cinque codici iniziali seguiti dal nome. Dalla locazione $0350 alla $0353 vengono conservati gli indirizzi di inizio e fine del programma da registrare prelevandoli da puntatori in pagina zero. Questa coppia di indirizzi viene utilizzata dalla routine di flashload per caricare il programma in formato veloce.

Poiché il programma verrà ricaricato nella stessa zona di memoria da cui era stato salvato, se si vuole registrare un programma che non parta da $0801 sarà sufficiente variare i puntatori $2B,$2C (inizio RAM Basic) e $2D, $2E (fine programma) con dei POKE. Viene poi copiata nel buffer (da $0354) la routine di Flashload. Essa è predisposta per l'esecuzione del RUN automatico. Se il flag SRFLAG indica la richiesta del SYS, verrà aggiunta una istruzione di JMP con l'indirizzo richiesto; in mancanza di autostart

Questo programma è disponibile su cassette presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 159.

viene inserito un JMP a $A52A dove si effettua il relink del programma ed il warm start.

Con queste modifiche è completata la costruzione dell'header che viene registrato in forma tradizionale richiamando due routine del KERNAL (il S.O.). Subito dopo viene forzata la registrazione, come programma appartenente alla testata appena scritta, dei due byte conservati in $E0FB che contengono l'entry-point della routine di load veloce.

La registrazione prosegue con il programma in formato flash. Come già visto è necessario attendere circa 0.3 secondi, poi inviare 255 byte di valore #$FF, uno #$00 per sincronismo e, quindi, tutto il programma un byte dopo l'altro.

La routine di scrittura di un byte richiama la routine di bitout (trasmissione di un bit), comunicando per mezzo del carry lo stato del bit da scrivere. Qui viene chiamata due volte la routine di timing, che sfruttando il timer B di uno dei due CIA del 64 (impostato su 60 microsecondi) temporizza l'alternarsi dello stato positivo e negativo in uscita per la generazione del segnale. In pratica il timer è un contatore hardware che viene decrementato ad ogni ciclo di clock (1 micros.) e quando raggiunge lo zero setta il bit 1 (underflow) del registro IRQ del CIA. Poiché all'inizio del flashsave vengono disabilitate tutte le sorgenti di interrupt, non viene prodotta alcuna interruzione sul microprocessore, ma possiamo ugualmente controllare tale registro per sapere quando è trascorso il tempo impostato.

```
1 REM ***********************************
2 REM *                                 *
3 REM * PROGRAMMA PER LA GENERAZIONE    *
4 REM *        DEL  FLASHTAPE           *
5 REM *                                 *
6 REM * PRIMA DI CARICARLO IN MEMORIA   *
7 REM * DIGITARE:                       *
8 REM *                                 *
9 REM *   POKE44,16:POKE4096,0:NEW      *
10 REM*                                 *
11 REM***********************************
12 REM
13 PRINT"□                ATTENDERE...."
15 RESTORE:IN=2049
20 READA$:IFA$="***"THEN70
30 TT=VAL("&"+A$):POKEIN,TT
40 FORC=1TO7:READA:POKEIN+C,A:TT=TT+A:NEXT
50 READA:IFA<>(TT AND255)THENPRINT"ERRORE NEI
   DATA ALLA LINEA"PEEK(64)*256+PEEK(63):END
60 IN=IN+8:GOTO20
70 HB=INT(IN/256):LB=IN-HB*256
80 PRINT "□":POKE43,1:POKE44,8:POKE45,"LB":POKE46,"HB":CLR"
90 POKE631,19:POKE632,13:POKE198,2
100 END
50000 DATA012,008,010,000,158,032,050,048,062
50010 DATA054,052,000,000,000,000,000,169,019
50020 DATA224,133,254,160,000,132,253,177,053
50030 DATA253,145,253,200,208,249,230,254,000
50040 DATA208,245,169,224,133,254,169,160,026
50050 DATA133,251,169,008,133,252,177,251,094
50060 DATA145,253,230,251,208,002,230,252,035
50070 DATA230,253,208,002,230,254,165,254,060
50080 DATA201,226,208,234,165,253,201,068,020
50090 DATA208,228,169,167,133,253,169,002,049
50100 DATA133,254,177,251,145,253,200,192,069
50110 DATA056,208,247,169,167,141,050,003,017
50120 DATA169,002,141,051,003,160,000,185,199
50130 DATA116,008,208,001,096,032,210,255,158
50140 DATA200,208,244,147,042,032,070,076,251
50150 DATA065,083,072,084,065,080,069,032,038
50160 DATA086,049,046,050,032,042,032,040,121
50170 DATA067,041,049,057,056,053,032,066,165
50180 DATA089,032,065,076,069,088,032,083,022
50190 DATA079,070,084,042,013,000,000,169,201
50200 DATA000,141,225,002,169,042,141,226,178
50210 DATA002,169,165,141,227,002,032,121,091
50220 DATA000,170,240,087,201,058,208,048,244
50230 DATA032,115,000,240,043,160,000,132,210
50240 DATA183,201,044,240,038,165,122,133,102
50250 DATA187,165,123,133,188,032,115,000,175
50260 DATA201,000,240,004,201,044,208,245,119
50270 DATA056,165,122,229,187,133,183,032,083
50280 DATA121,000,201,000,240,037,208,003,042
50290 DATA076,195,002,032,115,000,201,083,192
50300 DATA240,007,201,082,208,242,169,001,126
50310 DATA044,169,128,141,225,002,032,204,177
50320 DATA002,165,020,141,226,002,165,021,230
50330 DATA141,227,002,032,056,248,032,143,113
50340 DATA246,169,060,133,178,169,003,133,067
50350 DATA179,160,191,169,032,145,178,136,166
50360 DATA208,251,162,000,189,246,224,145,145
50370 DATA178,232,200,192,005,208,245,132,112
50380 DATA159,160,000,132,158,164,158,196,103
50390 DATA183,240,012,177,187,164,159,145,243
50400 DATA178,230,158,230,159,208,238,160,025
50410 DATA003,185,043,000,153,080,003,136,091
50420 DATA016,247,160,168,185,156,225,153,030
50430 DATA083,003,136,208,247,173,225,002,053
50440 DATA201,001,240,017,169,076,141,240,061
50450 DATA003,173,226,002,141,241,003,173,194
50460 DATA227,002,141,242,003,032,215,247,085
50470 DATA169,105,133,171,032,107,248,169,110
50480 DATA251,133,193,169,224,133,194,169,186
50490 DATA253,133,174,169,224,133,175,032,013
50500 DATA103,248,076,253,224,003,002,003,144
50510 DATA004,003,129,003,120,169,127,141,184
50520 DATA013,221,173,013,221,173,017,208,015
50530 DATA041,239,141,017,208,165,001,041,085
50540 DATA031,133,001,162,060,142,006,221,244
50545 REM                    D1↑            C1↑
50550 DATA032,138,225,032,121,225,160,255,164
50560 DATA169,255,032,083,225,136,208,248,076
50570 DATA160,007,024,032,100,225,136,016,188
50580 DATA249,160,000,177,172,032,083,225,074
50590 DATA032,107,003,144,244,032,148,225,167
50600 DATA165,001,041,247,009,032,133,001,117
50610 DATA088,024,169,000,141,160,002,076,148
50620 DATA185,002,133,164,169,008,133,163,189
50630 DATA024,032,100,225,006,164,198,163,144
50640 DATA016,247,096,032,107,225,032,107,094
50650 DATA225,096,169,002,044,013,221,240,242
50660 DATA251,144,005,162,011,202,208,253,212
50670 DATA165,001,073,008,133,001,169,000,038
50680 DATA141,007,221,169,025,141,015,221,172
50690 DATA096,162,003,189,080,003,149,172,086
50700 DATA202,016,248,160,000,136,208,253,199
50710 DATA202,208,248,096,169,016,044,013,228
50720 DATA220,240,251,173,013,221,162,000,000
50725 REM                          D2↑ C2↑
50730 DATA142,007,221,162,025,142,015,221,167
50740 DATA074,074,096,230,172,208,002,230,062
50750 DATA173,166,172,164,173,228,174,208,178
50760 DATA006,196,175,208,002,056,036,024,191
50770 DATA096,120,173,017,208,041,239,141,011
50780 DATA017,208,165,001,041,031,133,001,085
50790 DATA162,003,189,080,003,149,172,202,192
50800 DATA016,248,160,000,136,208,253,202,199
50810 DATA208,250,160,255,169,016,044,013,091
50820 DATA220,240,251,136,208,246,162,230,157
50825 REM                          D3↑ C3↑
50830 DATA142,006,221,169,001,072,032,084,215
50840 DATA003,104,176,247,042,201,000,208,213
50850 DATA244,169,008,133,163,032,084,003,068
50860 DATA038,164,198,163,016,247,165,164,131
50870 DATA160,000,145,172,032,107,003,144,251
50880 DATA232,169,131,141,002,003,169,164,243
50890 DATA141,003,003,169,000,141,160,002,107
50900 DATA032,147,252,088,134,045,132,046,108
50910 DATA169,167,072,169,233,072,169,000,027
50920 DATA076,113,168,165,186,201,001,240,126
50930 DATA003,076,237,245,165,001,041,253,250
50940 DATA133,001,076,000,224,165,001,009,097
50950 DATA002,133,001,024,076,147,252,165,032
50960 DATA001,009,002,133,001,076,008,175,149
50970 DATA165,001,009,002,133,001,032,115,202
50980 DATA000,032,107,169,165,001,041,253,000
50990 DATA133,001,096,056,044,048,054,050,226
51000 DATA ***
```

```
0000 0000                ;**********************************
0000 0000                ;*                                *
0000 0000                ;*          FLASHTAPE             *
0000 0000                ;*                                *
0000 0000                ;*   VELOCIZZATORE PER NASTRO      *
0000 0000                ;*                                *
0000 0000                ;*                                *
0000 0000                ;*   ALESSANDRO GUIDA             *
0000 0000                ;*      GIANNI ITTA               *
0000 0000                ;**********************************
0000 0000                ;
0000 0000                ;
0000 0000                        ISAVE=$0332     ;VETTORE SAVE
0000 0000                        FSAVE=$02A7     ;ENTRY FLASHSAVE
0000 0000                        SPNT=$FB        ;PUNTATORE PER TRASFER.
0000 0000                        EPNT=$FD
0000 0000                        SRFLAG=$02E1    ;SYS=80,RUN=01,READY=00
0000 0000                        SVSADR=$02E2
0000 0000                        NAMLEN=$B7      ;LUNGHEZZA NOME
0000 0000                        NAMADR=$BB      ;PUNTATORE NOME
0000 0000                ;
0000 0000                        CHRGET=$0073
0000 0000                        CHRGOT=$0079
0000 0000                ;
0000 0000                ;
0000 0000                        *=$0801
0001                     ;
0001                     ;INIZIO PROGRAMMA CON LINEA BASIC
0001                     ;
0801 0C                          .BYTE $0C,$08   ;LINK
0802 08
0803 0A                          .BYTE $0A,$00   ;NUMERO LINEA
0804 00
0805 9E                          .BYTE $9E,$20   ;CODICE "SYS 2064"
0806 20
0807 32 30                       .BYTE "2064"
0809 00                          .BYTE $00,$00   ;FINE LINEA BASIC
080A 00
080B 00                          .BYTE $00,$00,$00
080D 00
080E 00
080F 00
0810                     ;
0810                     ;COPIA KERNAL IN RAM
0810                     ;
0810 A9 E0                       LDA #$E0        ;PUNTATORE A $E000
0812 85 FE                       STA EPNT+1
0814 A0 00                       LDY #$00
0816 84 FD                       STY EPNT
0818 B1 FD               LOOP1   LDA (EPNT),Y    ;LEGGE IN ROM
081A 91 FD                       STA (EPNT),Y    ;SCRIVE IN RAM
081C C8                          INY
081D D0 F9                       BNE LOOP1
081F E6 FE                       INC EPNT+1
0821 D0 F5                       BNE LOOP1
0823                     ;
0823                     ;COPIA ROUTINE FLASHTAPE
0823                     ;
0823 A9 E0                       LDA #$E0
0825 85 FE                       STA EPNT+1      ;PUNTATORE A $E000
0827 A9 A0                       LDA #$A0
0829 85 FB                       STA SPNT        ;PUNTATORE DA $A000
082B A9 00                       LDA #$00
082D 85 FC                       STA SPNT+1
082F B1 FB               LOOP2   LDA (SPNT),Y    ;LEGGE ROUTINE
0831 91 FD                       STA (EPNT),Y    ;COPIA IN $E000
0833 E6 FB                       INC SPNT        ;AGGIORNA INDIRIZZI
0835 D0 02                       BNE NOC1
0837 E6 FC                       INC SPNT+1
0839 E6 FD               NOC1    INC EPNT
083B D0 02                       BNE NOC2
083D E6 FE                       INC EPNT+1
083F A5 FE               NOC2    LDA EPNT+1      ;CONTROLLA SE HA FINITO
0841 C9 E2                       CMP #$E2
0843 D0 EA                       BNE LOOP2
0845 A5 FD                       LDA EPNT
0847 C9 44                       CMP #$44
0849 D0 E4                       BNE LOOP2
084B                     ;
084B                     ;COPIA ROUTINE DI INTERFACCIA
084B                     ;
084B A9 A7                       LDA #$A7
084D 85 FD                       STA EPNT        ;PUNTATORE A $02A7
084F A9 02                       LDA #$02
0851 85 FE                       STA EPNT+1
0853 B1 FB               LOOP3   LDA (SPNT),Y    ;LEGGE ROUTINE INTERF. '
0855 91 FD                       STA (EPNT),Y    ;COPIA IN $02A7
0857 C8                          INY
0858 C0 38                       CPY #$38
085A D0 F7                       BNE LOOP3
085C                     ;
085C                     ;AGGIORNA VETTORE SAVE
085C                     ;
085C A9 A7                       LDA #<FSAVE     ;LBYTE
085E 8D 32 03                    STA ISAVE
0861 A9 02                       LDA #>FSAVE     ;HBYTE
0863 8D 33 03                    STA ISAVE+1
0866                     ;
0866                     ;SCRIVE TESTATA E TERMINA
0866                     ;
0866 A0 00                       LDY #$00
0868 B9 74 08            LOOP4   LDA TEXT,Y      ;LEGGE MESSAGGIO
086B D0 01                       BNE OUTPUT
086D 60                          RTS             ;TORNA AL BASIC
086E 20 D2 FF            OUTPUT  JSR $FFD2       ;OUTPUT SU SCHERMO
0871 C8                          INY
0872 D0 F4                       BNE LOOP4
0874                     ;
0874 93                  TEXT    .BYTE $93       ;PULISCE SCHERMO
0875 2A 20                       .BYTE "* FLASHTAPE V1.2 *"
0887 20 20                       .BYTE "  (C)1985 BY ALEX SOFT*"
089D 00                          .BYTE $00,$00,$00
089E 00
089F 00
08A0                     ;
08A0                     ;
08A0                     ;INIZIO ROUTINE DI SAVE IN FORMATO FLASH
08A0                     ;
08A0                     ;
08A0                             *=$E000
E000                     ;
E000                     ;
E000                     ;CONTROLLO PARAMETRI SAVE
E000                     ;
E000 A9 00               BEGIN   LDA #$00
E002 8D E1 02                    STA SRFLAG      ;DEFAULT READY
E005 A9 2A                       LDA #$2A
E007 8D E2 02                    STA SVSADR      ;PREPARA INDIRIZZO PER
E00A A9 A5                       LDA #$A5        ; RELINK E WARM START.
E00C 8D E3 02                    STA SVSADR+1
E00F 20 79 00                    JSR CHRGOT      ;SE NON VI SONO ALTRI
E012 AA                          TAX             ; MANCANO PARAMETRI
E013 F0 57                       BEQ HEADER
E015 C9 3A                       CMP #":"        ;PRIMO CARATTERE = ":" ?
E017 D0 30                       BNE ERRLK       ;NO-SALTA A SINTAX ERROR

E019 20 73 00                    JSR CHRGET      ;DOPO I DUE PUNTI DEVE
E01C F0 28                       BEQ ERRLK       ; SEGUIRE IL NOME
E01E A0 00                       LDY #$00
E020 84 B7                       STY NAMLEN      ;AZZERA NAMLEN
E022 C9 2C                       CMP #","        ;SALTA IL NOME
E024 F0 26                       BEQ SRMODE
E026 A5 7A                       LDA $7A         ;AGGIORNA PUNTATORI
E028 85 BB                       STA NAMADR      ; ALL'INIZIO DEL NOME
E02A A5 7B                       LDA $7B
E02C 85 BC                       STA NAMADR+1
E02E 20 73 00            LENGHT  JSR CHRGET      ;LEGGE FILENAME
E031 C9 00                       CMP #$00
E033 F0 04                       BEQ DONE
E035 C9 2C                       CMP #","
E037 D0 F5                       BNE LENGHT
E039 38                  DONE    SEC
E03A A5 7A                       LDA $7A
E03C E5 BB                       SBC NAMADR      ;CALCOLA LUNGHEZZA NOME
E03E 85 B7                       STA NAMLEN
E040 20 79 00                    JSR CHRGOT
E043 C9 00                       CMP #$00        ;FINE LINEA BASIC ?
E045 F0 25                       BEQ HEADER      ;SI
E047 D0 03                       BNE SRMODE      ;NO-ANALIZZA MODO
E049 4C C3 02            ERRLK   JMP ERROR
E04C                     ;
E04C                     ;ANALIZZA RICHIESTA DI READY/SYS/RUN
E04C                     ;
E04C 20 73 00            SRMODE  JSR CHRGET
E04F C9 53                       CMP #"S"
E051 F0 07                       BEQ SYS
E053 C9 52                       CMP #"R"
E055 D0 F2                       BNE ERRLK
E057 A9 01                       LDA #$01        ;FLAG = RUN
E059 2C                          .BYTE $2C       ;SALTA ISTRUZIONE SUCC.
E05A A9 80               SYS     LDA #$80        ;FLAG = SYS
E05C 8D E1 02                    STA SRFLAG
E05F 20 CC 02                    JSR VALGET      ;LEGGE INDIRIZZO 'SYS'
E062 A5 14                       LDA $14         ;IN $14,$15
E064 8D E2 02                    STA SVSADR
E067 A5 15                       LDA $15
E069 8D E3 02                    STA SVSADR+1
E06C                     ;
E06C                     ;INIZIO REGISTRAZIONE IN FORMATO NORMALE
E06C                     ; DELL'HEADER CONTENTE LA ROUTINE DI LOAD
E06C                     ;
E06C 20 38 F8            HEADER  JSR $F838       ;GESTIONE MESSAGGI E
E06F                             ; E TASTO PLAY
E06F 20 8F F6                    JSR $F68F       ;MESSAGGIO SAVING ....
E072                     ;
E072 A9 3C                       LDA #$3C        ;CARICA $033C NEL
E074 85 B2                       STA $B2         ; NEL PUNTATORE $B2,$B3
E076 A9 03                       LDA #$03
E078 85 B3                       STA $B3
E07A                     ;
E07A A0 BF                       LDY #$BF        ;RIFULISCE BUFFER
E07C A9 20                       LDA #$20
E07E 91 B2               CLEAR   STA ($B2),Y
E080 88                          DEY
E081 D0 FB                       BNE CLEAR
E083                     ;
E083 A2 00                       LDX #$00
E085 BD F6 E0            HCODE   LDA CODE,X      ;COPIA I CODICI INIZIALI
E088 91 B2                       STA ($B2),Y     ; DELL'HEADER NECESSARI
E08A E8                          INX             ; PER L'AUTOSTART
E08B C8                          INY
E08C C0 05                       CPY #$05
E08E D0 F5                       BNE HCODE
E090 84 9F                       STY $9F         ;SALVA PUNTATORE BUFFER
E092 A0 00                       LDY #$00
E094 84 9E                       STY $9E         ;PUNTATORE NOME
E096 A4 9E               NAMSET  LDY $9E         ;COPIA IL NOME DEL FILE
E098 C4 B7                       CPY NAMLEN      ; NEL BUFFER DI CASSETTA
E09A F0 0C                       BEQ NAMEND      ;FINE NOME
E09C B1 BB                       LDA (NAMADR),Y
E09E A4 9F                       LDY $9F
E0A0 91 B2                       STA ($B2),Y
E0A2 E6 9E                       INC $9E
E0A4 E6 9F                       INC $9F
E0A6 D0 EE                       BNE NAMSET
E0A8                     ;
E0A8                     ;COPIA ROUTINE DI FLASHLOAD NEL BUFFER CASSETTA
E0A8                     ; I PRIMI 4 BYTE SONO L'INDIRIZZO DI PARTENZA E FINE
E0A8                     ; DEL PROGRAMMA DA REGISTRARE.
E0A8                     ;
E0A8 A0 03               NAMEND  LDY #$03
E0AA B9 2B 00            IOADR   LDA $002B,Y     ;COPIA INDIRIZZI INIZIO
E0AD 99 50 03                    STA $0350,Y     ; FINE IN $0350 E $0352
E0B0 88                          DEY
E0B1 10 F7                       BPL IOADR
E0B3                     ;
E0B3 A0 A8                       LDY #$A8
E0B5 B9 9C E1            FLCODE  LDA COPY,Y      ;COPIA ROUTINE DI LOAD
E0B8 99 53 03                    STA $0353,Y
E0BB 88                          DEY
E0BC D0 F7                       BNE FLCODE
E0BE AD E1 02                    LDA SRFLAG      ;AGGIORNA TERMINE ROUT.
E0C1 C9 01                       CMP #$01        ; IN BASE AL FLAG DI
E0C3 F0 11                       BEQ IOBUF       ; SYS/RUN/READY
E0C5 A9 4C                       LDA #$4C
E0C7 8D F0 03                    STA FINE
E0CA AD E2 02                    LDA SVSADR
E0CD 8D F1 03                    STA FINE+1
E0D0 AD E3 02                    LDA SVSADR+1
E0D3 8D F2 03                    STA FINE+2
E0D6 20 D7 F7            IOBUF   JSR $F7D7       ;IMPOSTA INDIRIZZO BUF.
E0D9 A9 69                       LDA #$69        ;105 SYNC
E0DB 85 AB                       STA $AB
E0DD 20 6B F8                    JSR $F86B       ;SCRIVE BLOCCO HEADER
E0E0                     ;
E0E0 A9 FB                       LDA #<LOADAD    ;AGGIORNA INDIRIZZI I/O
E0E2 85 C1                       STA $C1         ;PER PUNTARE AI DUE BYTE
E0E4 A9 E0                       LDA #>LOADAD    ;CHE PROVOCANO L'AUTOST.
E0E6 85 C2                       STA $C2         ;DELLA ROUTINE DI LOAD
E0E8 A9 FD                       LDA #<LOADAD+2
E0EA 85 AE                       STA $AE         ;LBYTE INDIRIZZO FINE
E0EC A9 E0                       LDA #>LOADAD
E0EE 85 AF                       STA $AF         ;HBYTE
E0F0                     ;
E0F0 20 67 F8                    JSR $F867       ;REGISTRA I DUE BYTE
E0F3 4C FD E0                    JMP FLSAVE
E0F6                     ;
E0F6 03                  CODE    .BYTE $03       ;CODICE PER LOAD NON
E0F7                             ; NON RILOCATO
E0F7 02                          .BYTE $02,$03   ;INDIRIZZO INIZIO LOAD
E0F9 03
E0F9 04                          .BYTE $04,$03   ;INDIRIZZO DI FINE LOAD
E0FA 03
E0FB                     ;
E0FB 01 03               LOADAD  .WORD LOAD      ;INDIRIZZO PARTENZA
E0FD                             ; ROUTINE LOAD VELOCE.
E0FD                     ;
E0FD                     ;ENTRY POINT SAVE VELOCE
E0FD                     ;
E0FD 78                  FLSAVE  SEI             ;DISABILITA IRQ
```

```
00261  E0FE  A9 7F        LDA ##7F        ;DISABILITA NMI
00262  E100  8D 0D DD     STA #DD0D
00263  E103  AD 0D DD     LDA #DD0D       ;RESET U/F FLAG
00264  E106  AD 11 D0     LDA #D011       ;DISABILITA VIDEO
00265  E109  29 EF        AND ##EF
00266  E10B  8D 11 D0     STA #D011
00267  E10E  A5 01        LDA #01         ;AVVIA MOTORE
00268  E110  29 1F        AND ##1F        ;CON SEGNALE OFF
00269  E112  85 01        STA #01
00270  E114  A2 3C        LDX ##3C        ;SETTA TIMER X 60 US
00271  E116  8E 06 DD     STX #DD06       ; (VEDI ARTICOLO)
00272  E119  20 8A E1     JSR ADRLIM      ;LEGGE LIMITI MEMORIA
00273  E11C                               ; DA REGISTRARE.
00274  E11C  20 79 E1     JSR ZERO        ;SEGNALE ON E ATTIVA TIMER
00275  E11F                ;
00276  E11F                ;INVIA 255 BYTE #FF PER LASCIARE REGOLARIZZARE LA
00277  E11F                ;VELOCITA' DEL NASTRO
00278  E11F                ;
00279  E11F  A0 FF        LDY ##FF        ;NUMERO BYTE
00280  E121  A9 FF        HEAD  LDA ##FF  ;BYTE DA INVIARE
00281  E123  20 53 E1     JSR BYTOUT      ;INVIA BYTE
00282  E126  88           DEY             ;SE NON HA FINITO
00283  E127  D0 F8        BNE HEAD
00284  E129                ;
00285  E129                ;INVIA BYTE DI SINCRONISMO (##00)
00286  E129                ;
00287  E129  A0 07        LDY ##07
00288  E12B  18           SYNCBY CLC      ;BIT = 0
00289  E12C  20 64 E1     JSR BITOUT      ;INVIA IL BIT
00290  E12F  88           DEY
00291  E130  10 F9        BPL SYNCBY
00292  E132                ;
00293  E132                ;REGISTRAZIONE MEMORIA
00294  E132                ;
00295  E132  A0 00        NEXTBY LDY ##00
00296  E134  B1 AC        LDA (#AC),Y     ;LEGGE CARATTERE
00297  E136  20 53 E1     JSR BYTOUT      ;E LO INVIA SU NASTRO
00298  E139  20 6B 03     JSR BUMPAD      ;AGGIORNA INDIRIZZO
00299  E13C  90 F4        BCC NEXTBY      ;SE NON HA FINITO RIPETE
00300  E13E  20 94 E1     JSR DELAY       ;ATTENDE 0.3 SEC
00301  E141  A5 01        LDA #01         ;FERMA MOTORE
00302  E143  29 F7        AND ##F7        ;E SEGNALE OFF
00303  E145  09 20        ORA ##20
00304  E147  85 01        STA #01
00305  E149  58           CLI
00306  E14A  18           CLC
00307  E14B  A9 00        LDA ##00
00308  E14D  8D A0 02     STA #02A0
00309  E150  4C B9 02     JMP SAVEND      ;RIABILITA SCHERMO
00310  E153                               ; E TERMINA
00311  E153                ;
00312  E153                ;ROUTINE DI OUTPUT DI UN BYTE
00313  E153                ;
00314  E153  85 A4        BYTOUT STA #A4  ;CONSERVA BYTE DA TRASM.
00315  E155  A9 08        LDA ##08        ;8-BIT(+1)DA INVIARE
00316  E157  85 A3        STA #A3
00317  E159  18           CLC             ;PRIMO BIT DI SINCRON.
00318  E15A  20 64 E1     NEXTBI JSR BITOUT ;INVIA BIT
00319  E15D  06 A4        ASL #A4         ;PORTA NEL CARRY IL BIT
00320  E15F                               ; SUCCESSIVO
00321  E15F  C6 A3        DEC #A3         ;CONTROLLA SE HA FINITO
00322  E161  10 F7        BPL NEXTBI      ;ANCORA
00323  E163  60           RTS
00324  E164                ;
00325  E164                ;ROUTINE DI INVIO DI UN BIT AL NASTRO
00326  E164                ;
00327  E164  20 6B E1     BITOUT JSR TIMING ;TEMPORIZZAZIONE SEGNALE
00328  E167  20 6B E1     JSR TIMING
00329  E16A  60           RTS
00330  E16B                ;
00331  E16B  A9 02        TIMING LDA ##02 ;ATTENDE TIMER U/F
00332  E16D  2C 0D DD     NREADY BIT #DD0D
00333  E170  F0 FB        BEQ NREADY
00334  E172  90 05        BCC ZERO
00335  E174  A2 0B        LDX ##0B
00336  E176  CA           DELTA DEX       ;DELTA PARI A 55US
00337  E177  D0 FD        BNE DELTA
00338  E179  A5 01        ZERO  LDA #01
00339  E17B  49 08        EOR ##08        ;INVERTE SEGNALE
00340  E17D  85 01        STA #01
00341  E17F  A9 00        LDA ##00
00342  E181  8D 07 DD     STA #DD07
00343  E184  A9 19        LDA ##19        ;RIAVVIA TIMER
00344  E186  8D 0F DD     STA #DD0F
00345  E189  60           RTS
00346  E18A                ;
00347  E18A  A2 03        ADRLIM LDX ##03 ;RILEVA I LIMITI DEL
00348  E18C  BD 50 03     L2    LDA #0350,X ; PROGRAMMA DA REGISTR.
00349  E18F  95 AC        STA #AC,X       ; DALLA TESTATA
00350  E191  CA           DEX
00351  E192  10 F8        BPL L2
00352  E194  A0 00        DELAY LDY ##00  ;ATTENDE CIRCA 0.3 SEC.
00353  E196  88           L3    DEY
00354  E197  D0 FD        BNE L3
00355  E199  CA           DEX
00356  E19A  D0 F8        BNE DELAY
00357  E19C  60           COPY  RTS
00358  E19D                ;
00359  E19D                ;ROUTINE DI LOAD VELOCE MEMORIZZATA NELLA TESTATA
00360  E19D                ; (HEADER) DEL PROGRAMMA. NEL BUFFER DI CASSETTA
00361  E19D                ; (#033C).
00362  E19D                ;
00363  E19D                     *=*+0354
00364  0354                ;
00365  0354                ;ROUTINE DI BITIN (INPUT DI UN BIT)
00366  0354                ;
00367  0354  A9 10        BITIN  LDA ##10
00368  0356  2C 0D DC     WAIT   BIT #DC0D ;ATTENDE SEGNALE DI SYNC
00369  0359  F0 FB        BEQ WAIT
00370  035B  AD 0D DD     LDA #DD0D       ;RILEVA STATO TIMER
00371  035E  A2 00        LDX ##00        ;RIIMPOSTA PARTE ALTA TIMER
00372  0360  8E 07 DD     STX #DD07       ; (VEDI ARTICOLO)
00373  0363  A2 19        LDX ##19
00374  0365  8E 0F DD     STX #DD0F       ;RIAVVIA TIMER
00375  0368  4A           LSR A
00376  0369  4A           LSR A           ;RIPORTA BIT NEL CARRY
00377  036A  60           RTS
00378  036B                ;
00379  036B                ;ROUTINE INCREMENTO INDIRIZZO LOAD
00380  036B                ;
00381  036B  E6 AC        BUMPAD INC #AC
00382  036D  D0 02        BNE L4
00383  036F  E6 AD        INC #AD
00384  0371  A6 AC        L4    LDX #AC
00385  0373  A4 AD        LDY #AD         ;SE FINE CARRY=1
00386  0375  E4 AE        CPX #AE
00387  0377  D0 06        BNE NOC
00388  0379  C4 AF        CPY #AF
00389  037B  D0 02        BNE NOC
00390  037D  38           SEC
00391  037E  24           .BYTE #24       ;SALTA ISTRUZIONE SUCCESS.
00392  037F  18           NOC   CLC
00393  0380  60           RTS
00394  0381                ;
00395  0381                ;ENTRY POINT ROUTINE LOAD VELOCE
00396  0381                ;
00397  0381  78           LOAD   SEI
00398  0382  AD 11 D0     LDA #D011       ;DISABILITA VIDEO
00399  0385  29 EF        AND ##EF
00400  0387  8D 11 D0     STA #D011
00401  038A  A5 01        LDA #01         ;AVVIA MOTORE
00402  038C  29 1F        AND ##1F
00403  038E  85 01        STA #01
00404  0390  A2 03        LDX ##03        ;RILEVA I LIMITI DEL
00405  0392                               ; PROGRAMMA DA CARICARE
00406  0392  BD 50 03     L9    LDA #0350,X ;DALLA TESTATA
00407  0395  95 AC        STA #AC,X
00408  0397  CA           DEX
00409  0398  10 F8        BPL L9
00410  039A  A0 00        LDY ##00        ;ATTENDE CIRCA 0.3 SEC.
00411  039C  88           L10   DEY
00412  039D  D0 FD        BNE L10
00413  039F  CA           DEX
00414  03A0  D0 FA        BNE L10
00415  03A2                ;
00416  03A2                ;RICEVE 255 SEGNALI PER PERMETTERE LA STABILIZZAZIONE
00417  03A2                ;DELLA VELOCITA' DEL NASTRO
00418  03A2                ;
00419  03A2  A0 FF        LDY ##FF
00420  03A4  A9 10        L6    LDA ##10
00421  03A6  2C 0D DC     WAIT2  BIT #DC0D ;RILEVA SEGNALE
00422  03A9  F0 FB        BEQ WAIT2
00423  03AB  88           DEY
00424  03AC  D0 F6        BNE L6
00425  03AE                ;
00426  03AE                ;RILEVA BYTE DI SINCRONIZZAZIONE (##00)
00427  03AE                ;
00428  03AE  A2 E6        LDX ##E6        ;IMPOSTA PARTE BASSA TIMER SU
00429  03B0  8E 06 DD     STX #DD06       ;230 US. (VEDI ARTICOLO)
00430  03B3  A9 01        NOSYNC LDA ##01 ;FLAG SYNC NON RICEVUTO
00431  03B5  48           SYNC   PHA
00432  03B6  20 54 03     JSR BITIN       ;RICEVE BIT
00433  03B9  68           PLA             ;RECUPERA BYTE IN
00434  03BA                               ; RICEZIONE
00435  03BA  B0 F7        BCS NOSYNC      ;SE BIT=1 NON E' IL
00436  03BC                               ; BYTE DI SINCRONISMO
00437  03BC  2A           ROL A           ;CONSERVA BIT RICEVUTO
00438  03BD  C9 00        CMP ##00        ;SE BYTE=#00 FINE
00439  03BF  D0 F4        BNE SYNC
00440  03C1                ;
00441  03C1                ;LETTURA DATI DAL NASTRO
00442  03C1                ;
00443  03C1  A9 08        BYTEIN LDA ##08 ;8 BIT (+1) DA RICEVERE
00444  03C3  85 A3        STA #A3
00445  03C5  20 54 03     L1    JSR BITIN ;RICEVE BIT
00446  03C8  26 A4        ROL #A4         ;BIT IN #A4
00447  03CA  C6 A3        DEC #A3         ;-1 BIT DA RICEVERE
00448  03CC  10 F7        BPL L1
00449  03CE  A5 A4        LDA #A4         ;BYTE RIC. IN (A)
00450  03D0                ;
00451  03D0  A0 00        LDY ##00
00452  03D2  91 AC        STA (#AC),Y     ;MEMORIZZA BYTE
00453  03D4  20 6B 03     JSR BUMPAD      ;AGGIORNA INDIRIZZO
00454  03D7  90 E8        BCC BYTEIN      ;RIPETE SE C=0
00455  03D9  A9 83        LDA ##83
00456  03DB  8D 02 03     STA #0302
00457  03DE  A9 A4        LDA ##A4
00458  03E0  8D 03 03     STA #0303
00459  03E3  A9 00        LDA ##00
00460  03E5  8D A0 02     STA #02A0
00461  03E8  20 93 FC     JSR #FC93       ;FERMA MOTORE E ABILITA
00462  03EB                               ; SCHERMO.
00463  03EB  58           CLI
00464  03EC  86 2D        STX #2D         ;AGGIORNA FINE PROGRAMMA
00465  03EE  84 2E        STY #2E
00466  03F0  A9 A7        FINE   LDA ##A7 ;FINE PREDISPOSTA
00467  03F2  48           PHA             ; PER IL RUN AUTOMATICO
00468  03F3  A9 E9        LDA ##E9
00469  03F5  48           PHA
00470  03F6  A9 00        LDA ##00
00471  03F8  4C 71 A8     JMP #A871
00472  03FB                ;
00473  03FB                ;INIZIO ROUTINE SAVE VELOCE IN #02A7
00474  03FB                ;
00475  03FB                     *=#02A7
00476  02A7                ;
00477  02A7  A5 BA        SAVEIN LDA #BA  ;SE PERIFERICA DIVERSA
00478  02A9  C9 01        CMP ##01        ;DA 1 (REGISTRATORE)
00479  02AB  F0 03        BEQ OK          ;VA AL LOAD NORMALE
00480  02AD  4C ED F5     JMP #F5ED
00481  02B0  A5 01        OK     LDA #01  ;ABILITA RAM IN #E000
00482  02B2  29 FD        AND ##FD
00483  02B4  85 01        STA #01
00484  02B6  4C 00 E0     JMP BEGIN       ;CONTROLLO SINTASSI E
00485  02B9                               ; INIZIO SAVE.
00486  02B9  A5 01        SAVEND LDA #01  ;RIABILITA ROM IN #E00
00487  02BB  09 02        ORA ##02
00488  02BD  85 01        STA #01
00489  02BF  18           CLC
00490  02C0  4C 93 FC     JMP #FC93       ;RIABILITA SCHERMO E
00491  02C3                               ; TORNA AL BASIC
00492  02C3                ;
00493  02C3  A5 01        ERROR  LDA #01
00494  02C5  09 02        ORA ##02
00495  02C7  85 01        STA #01
00496  02C9  4C 08 AF     JMP #AF08       ;SYNTAX ERROR
00497  02CC  A5 01        VALGET LDA #01
00498  02CE  09 02        ORA ##02
00499  02D0  85 01        STA #01
00500  02D2  20 73 00     JSR CHRGET
00501  02D5  20 6B A9     JSR #A96B
00502  02D8  A5 01        LDA #01
00503  02DA  29 FD        AND ##FD
00504  02DC  85 01        STA #01
00505  02DE  60           RTS
00506  02DF                ;
00507  02DF                .END
```

```
SYMBOL TABLE
SYMBOL VALUE
 ADRLIM   E18A    BEGIN   E000    BITIN   0354    BITOUT  E164
 BUMPAD   036B    BYTEIN  03C1    BYTOUT  E153    CHRGET  0073
 CHRGOT   0079    CLEAR   E07E    CODE    E0F6    COPY    E19C
 DELAY    E194    DELTA   E176    DONE    E039    EPNT    00FD
 ERRLK    E049    ERROR   02C3    FINE    03F0    FLCODE  E085
 FLSAVE   E0FD    FSAVE   02A7    HCODE   E085    HEAD    E121
 HERDER   E06C    IOADR   E0AA    IOBUF   E006    ISAVE   0332
 L1       03C5    L10     039C    L2      E18C    L3      E196
 L4       0371    L6      03A4    L9      0392    LENGHT  E02E
 LOAD     0381    LORDAD  E0FB    LOOP1   0818    LOOP2   082F
 LOOP3    0853    LOOP4   0868    NAMADR  00BB    NAMEND  E0A9
 NAMLEN   00B7    NAMSET  E096    NEXTBI  E15A    NEXTBY  E132
 NOC      037F    NOC1    0839    NOC2    083F    NOSYNC  03B3
 NREADY   E16D    OK      02B0    OUTPUT  086E    SAVEIN  02A7
 SAVEND   02B9    SPNT    00FB    SRFLAG  02E1    SRMODE  E04C
 SYNC     03B5    SYNCBY  E12B    SYS     E05A    SYSADR  02E2
 TEXT     0874    TIMING  E16B    VALGET  02CC    WAIT    0356
 WAIT2    03A6    ZERO    E179
END OF ASSEMBLY
```

Scaduto il tempo si testa il carry e, se è alto (bit da trasmettere = 1), si aggiunge un ulteriore ritardo di 55 microsecondi.

Va tenuto presente che la durata effettiva dei segnali è di circa 20 microsecondi superiore a quella impostata nel timer a causa del tempo necessario al microprocessore per rilevare l'underflow del timer ed invertire il segnale in uscita.

Nota: Sul listato in Assembler si può vedere che in effetti la linea di uscita verso il registratore viene manipolata a rovescio. In altre parole, quando parliamo di segnale alto, in realtà si trova a zero mentre è a 1 quando il segnale dovrebbe essere basso. Ciò è dovuto al fatto che, per motivi hardware, il segnale tra la registrazione e la riproduzione viene invertito costringendoci, in fase di scrittura, ad operare esattamente al contrario di come dovremmo.

Il save si conclude con la riabilitazione di tutte le sorgenti di interrupt ed il ritorno al Basic.

Segue la routine di save quella per il load, che verrà copiata ad ogni salvataggio nell'header della registrazione.

La fase di load inizia con la lettura della testata e dei due byte seguenti da parte del sistema operativo del 64. Finito il caricamento di questa prima parte viene automaticamente eseguita la routine di flashload a partire da $0381.

Dopo avere disabilitato l'IRQ viene rimesso in movimento il motore e letti i limiti di memoria da caricare da nastro. Comincia poi la ricerca del byte di sincronismo che segna l'inizio dei dati.

Vengono contati 255 segnali in ingresso senza controllarne la durata, al solo scopo di permettere al nastro di stabilizzarsi in velocità. Inizia quindi il test dei bit in ingresso in attesa di otto bit consecutivi di valore 0 (vedi fig. 2). L'ottavo «0» segna l'inizio dei bit appartenenti al programma.

La ricezione di un bit dal nastro è gestita dalla routine di «bitin». In riproduzione il registratore è collegato al piedino FLAG del CIA #1 (Complex Interface Adapter) del C64. Un fronte negativo (passaggio dallo stato alto a quello basso) del segnale su di questa entrata provoca il settaggio del bit 4 del registro delle richieste di interrupt ($DC0D) del CIA. Non possiamo quindi sapere lo stato del segnale in arrivo dal registratore, ma verremo inequivocabilmente informati ogni volta che vi sarà un fronte discendente.

Il flag viene azzerato dalla lettura dello stesso registro.

Per determinare la durata di un segnale in arrivo utilizzeremo il timer b del CIA #2. Il timer verrà avviato ad ogni fronte negativo rilevato, con un

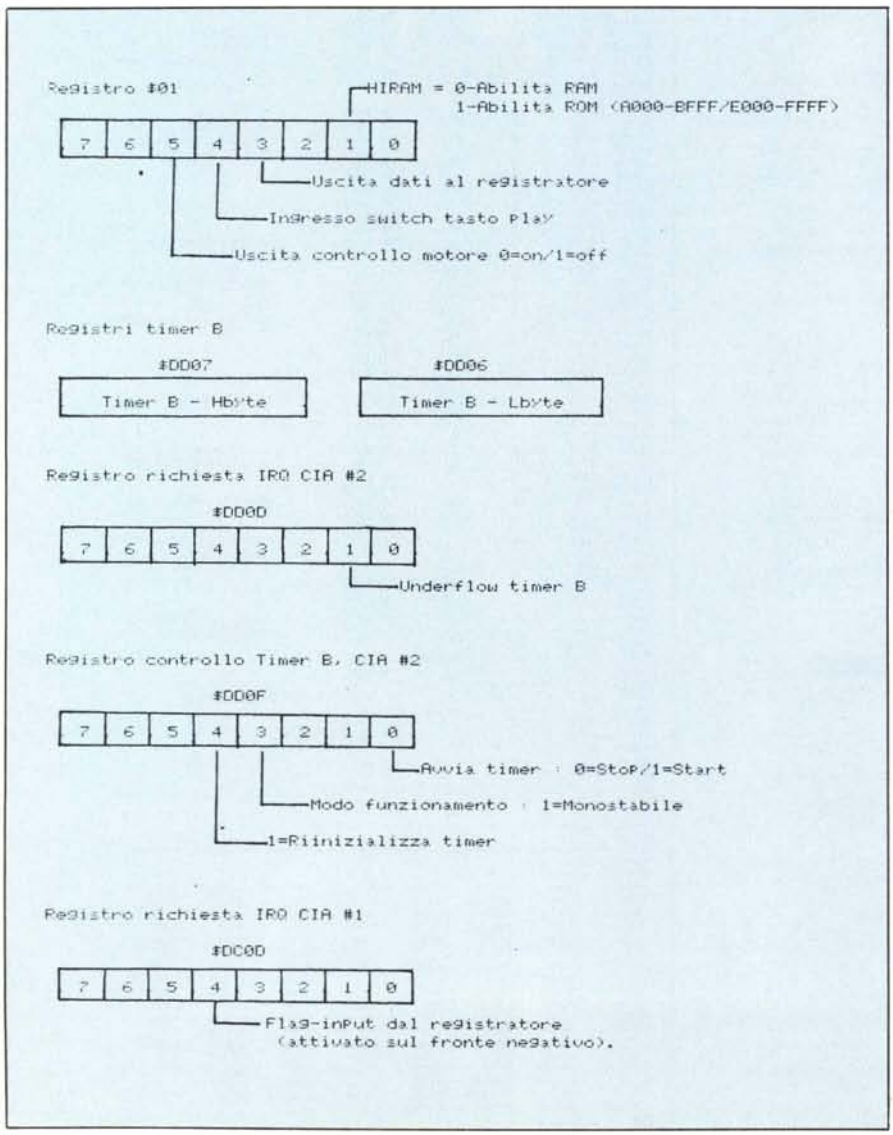
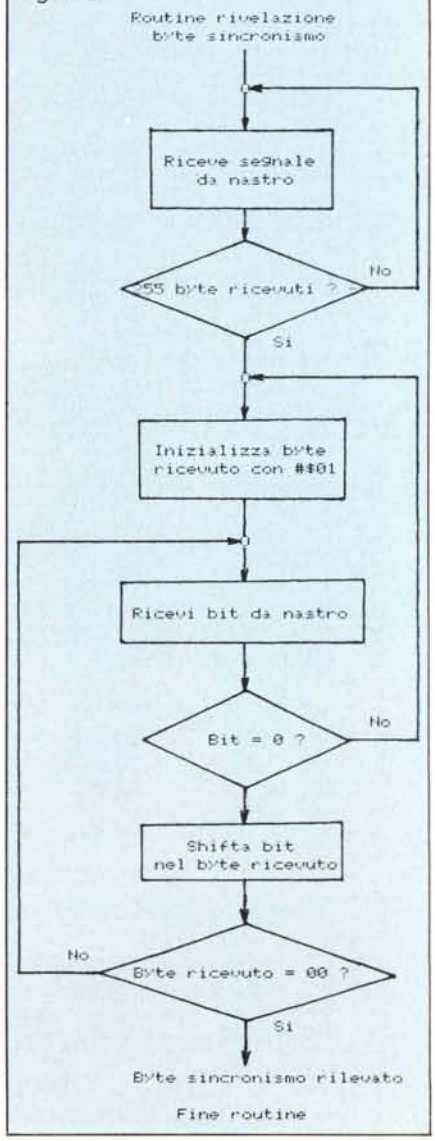


Figura 1 - Registri utilizzati dal programma di questo articolo.

Figura 2



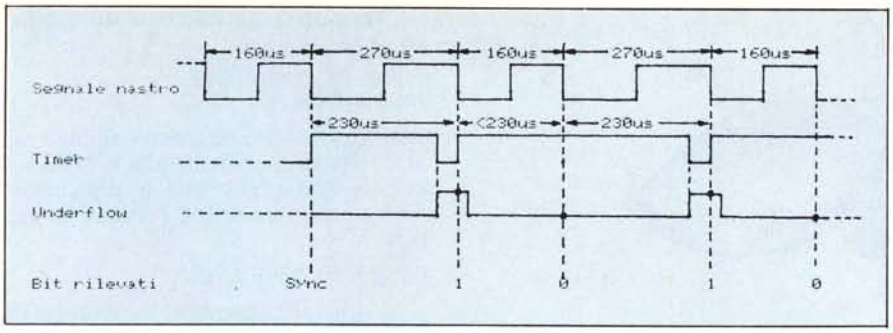


Figura 3 - Esempio processo di ricezione di un byte.

tempo superiore alla durata del bit di valore 0 (160 microsecondi) ma inferiore all'uno (270 microsecondi). Sperimentalmente abbiamo trovato migliore il valore di 230 microsecondi. All'arrivo del bit seguente viene controllato se il timer ha raggiunto lo zero (settando il flag di underflow) o no. Nel primo caso la durata del segnale sarà stata superiore a quella del timer (si tratta, quindi, di un segnale lungo, ossia un bit = 1), nel secondo il segnale avrà avuto una durata inferiore a quella impostata nel timer. Pertanto, lo stato del flag di underflow riflette la durata del segnale: 1-segnale lungo, 0-segnale corto: indica cioè il valore del bit ricevuto. Comunque, prima dell'analisi dello stato dell'underflow viene riavviato il timer per la verifica del bit seguente.

All'uscita della routine lo stato del flag suddetto viene riportato nel carry per essere poi immesso nel byte in ricezione. Ricordiamo infine che ogni byte è composto da nove bit di cui il primo serve solo a fornire il fronte negativo necessario per la valutazione del primo bit e dei seguenti.

Realizzazione del programma

Il listato in Assembly presenta tre campi di indirizzo. La prima parte è posta in $E000, la seconda (il flashload) in $0354, e la terza (le routine di interfacciamento) in $02A7.

Quest'ultime saranno effettivamente allocate in pagina 2 mentre, la routine di load, sarà accodata al flashsave a partire da $E198. Essa verrà copiata nel buffer di cassetta solo durante la fase di save. È stato necessario effettuarne l'assemblaggio nella zona di memoria corretta affinché alle istruzioni di JMP e JSR venissero assegnati gli indirizzi esatti.

Chiarito ciò, risulta comunque preferibile conservare il programma in altra forma. I codici generati dall'Assembler li accoderemo ad una piccola routine che si occuperà di ricopiare la ROM in $E000 in RAM (come detto nel corso dell'articolo), di ricopiare le due parti del flashtape in $E000 e $02A7, ed infine di cambiare gli indirizzi del vettore di SAVE per puntare alla nuova routine.

Per caricare la routine in memoria potete servirvi di un monitor per linguaggio macchina e del codice oggetto riportato sul listato Assembly a lato degli indirizzi. Salvate infine il tutto su disco o nastro a partire da $0801. Il programma così registrato sarà eseguibile con un semplice RUN.

In altenativa potrete utilizzare il listato «generatore» in Basic. Prima di cominciarne la digitazione è, però, indispensabile battere in modo diretto la linea riportata nei REM dello stesso listato, affinché l'inizio della RAM dedicata ai programmi in Basic venga spostato da $0801 a $1001.

Il programma è dotato di un controllo di checksum per ogni linea di dati per cui le possibilità di errore sono minimizzate. Se comunque nutrite il sospetto di aver commesso qualche errore, salvate il programma in questa forma prima di dare il Run. Lanciandone l'esecuzione infatti verrà generata in memoria la versione definitiva del Flashtape, ed il sorgente in Basic cancellato.

A questo punto il programma può essere salvato con un semplice SAVE su disco o nastro.

Prima di concludere dobbiamo sottolineare un aspetto comune a tutti i velocizzatori per nastro.

Per ottenere prestazioni eccezionali, questi programmi lavorano ai limiti delle capacità fisiche del registratore e dei nastri comuni. Pertanto, affinché funzionino a dovere, è necessario che il registratore abbia le testine ben allineate e le cassette siano di buona qualità.

In ogni caso, il flashtape permette a chi dovesse avere delle difficoltà con il proprio registratore di approntare delle versioni leggermente meno veloci.

In tabella 1 sono riportati altri valori di temporizzazione ed i dati da modificare nei listati.

Conclusioni

La stesura di questo programma ha richiesto diverso tempo, per poter creare un prodotto davvero affidabile. Poiché non sempre i calcoli su carta garantiscono la riuscita di un progetto, abbiamo passato lunghe ore in laboratorio ad osservare all'oscilloscopio i trabballanti segnali in uscita dal registratore. Ed ancora più tempo, abbiamo trascorso a verificare il programma su i tipi di registratori più disparati; originali e più o meno compatibili. Infine abbiamo cercato non solo di darvi il programma funzionante, ma anche di condensare in questo articolo, nella maniera più semplice possibile, tutte le spiegazioni circa il funzionamento del registratore Commodore e del nostro velocizzatore.

	1.versione	2.versione	3.versione
Timing save (bit 0)	60 micros.	78 micros.	112 micros.
Timing load	230 "	264 "	325 "
Timer-L save ($E115)	#$ 3C	#$ 4E	#$ 70
Timer-H load ($E1A8)	#$ 00	#$ 01	#$ 01
Timer-L load ($E1F8)	#$ E6	#$ 08	#$ 45
DATA D1/C1 linea 50540	060/244	078/006	112/040
" D2/C2 " 50720	000/000	001/001	001/001
" D3/C3 " 50820	230/157	008/191	69/252

Tabella 1 - Temporizzazioni possibili per variare la velocità del programma flashtape. Per ogni versione vi sono dati da sostituire nel listato Assembly o in quello Basic.

Ci sia permesso, in conclusione di ringraziare la ditta GR-ELEKTRO-SUD di Francavilla F. per averci messo a disposizione per tanto tempo e con tanta pazienza il suo laboratorio.

Per chi volesse approfondire la conoscenza del Commodore-64 e delle sue periferiche, ecco un elenco dei testi e manuali che abbiamo consultato:

Il S.O. del CBM64 - EVM
C64 Programmer's Ref. Guide
Commodore
La programmazione del 6502
Jackson
Macro-Assembler - Commodore
Impariamo il L.M. con il C64
Personal Software
Applicazioni del 6502 - Jackson