ISSN: 0393 - 1439

CHIP SPECIAL

Mensile di
micro
e personal
computer
N° 9 Ottobre 1985
L. 8000

HANNES RÜGHEIMER
CHRISTIAN SPANIK

PEEK POKE

COMMODORE 64

manuale

casa editrice tecniche nuove -spedizione in abbonamento postale-gruppo III/70

Indice

COSA BISOGNA SAPERE DI QUESTO MANUALE:

Prima di tutto:

e' opinione di tutti coloro che hanno un Commodore 64, che e' uno dei piu' ingegnosi strumenti.
Indipendentemente dal fatto che voi abbiate una vasta o limitata conoscenza di questo apparecchio, con questo manuale nelle vostre mani, potete esplorare ed esaminare il campo sconosciuto della memoria del Commodore.
Questo libro deve essere una guida, deve spiegare quali siano le locazioni di utilizzo e come impiegarle pienamente.
Viene spiegato tutto esattamente ed e' presente anche una tabella delle parole-chiave e alcune spiegazioni sui termini tecnici: ma dovete avere in ogni caso una conoscenza del BASIC e una piccola esperienza per quanto riguarda la programmazione.

Inoltre:

il manuale del Commodore 64 viene usato come aiuto durante la programmazione.
Percio' abbiamo tentato di collocare le informazioni in modo che possiate trovarle il piu' velocemente possibile.
Di conseguenza si avranno le informazioni in maniera oggettiva.
Solo cosi' si puo' lavorare: per chi incomincia dovra' fare piu' attenzione ai programmi d'esempio.
Il professionista puo' farne a meno.
Ancora una parola per quanto riguarda la dimensione del manuale:
noi vi avremmo potuto spiegare tutte le 65535 locazioni.
Ma non ne avete bisogno, vi basta conoscerne alcune centinaia, le altre sono 'ambito libero RAM', ed abbiamo argomentato di quelle che sono le piu' importanti per la programmazione.
Infine affermiamo, come autori, di aver scritto questo nostro secondo libro sul Commodore 64, per poter aiutare coloro che iniziano.
Percio' vi ringraziamo, se volete comunicarci qualche miglioria, che ci preoccuperemo di introdurre al piu' presto.

A questo punto vi auguriamo

'BUONE POKEs'

COME SI DEVE USARE IL MANUALE

1.LOCAZIONI

Questo termine indica il numero della locazione in esame.
Con il comando POKE, si scrive un valore nella locazione; con PEEK si legge il contenuto.

2.DENOMINAZIONE

Qui trovate i nomi, lo scopo delle locazioni.

3.VALORE NORMALE

Ogni qual volta fosse necessario, abbiamo registrato presso una locazione il valore, che e' tipico e noto alla normale funzione del computer.
La maggior parte di questi valori corrispondono al contenuto di una locazione dopo aver acceso il computer.
Si capisce da soli che per i registri della tastiera, e i valori casuali, non viene indicato nessun valore normale.

4.DESCRIZIONE

Questa descrizione vi spiega le funzioni piu' importanti delle locazioni, come si devono usare, cosa e' piu' interessante per voi, e altre cose.

5.COMPITO DEI BITs

Se una locazione ha piu' funzioni trovate qui per ogni Bit la singola funzione in un sommario. Se voi avete gia' esperienza con le locazioni, basta dare un'occhiata al sommario, per trovare la funzione desiderata

6.SPIEGAZIONE

Il compito di ciascun bit viene esposto nel dettaglio.
Si parla anche di particolarita' o curiosita'. Questa spiegazione e' utile anche a coloro che iniziano.

7.ESEMPI

Infine ci sono ancora esempi, che devono fornire le spiegazioni. Sotto questo punto trovate anche tutte le 'utilita'',siano in Basic o in Linguaggio macchina, che vi faciliteranno compiti spiacevoli o lunghi.

I TERMINI TECNICI

I Termini tecnici nel testo, che vi vogliamo spiegare piu' dettagliatamente, verranno scritti in corsivo.
Le spiegazioni nel testo vi avrebbero portato troppo lontano. Alla fine del libro trovate percio' l'appendice dei termini tecnici, dove vi potrete trovare una spiegazione dettagliata dei suddetti termini.

L'ASCII E LA TABELLA CODICI VIDEO

Nell'appendice c'e' inoltre il sommario dell'ASCII e il codice video, che vengono usati nel testo, ma sono raccolti globalmente nelle nominate tabelle.

CARATTERI SPECIALI

Per risparmiarvi la ricerca e l'indagine del significato dei segni grafici, abbiamo stampato tali segni come comando CHR$ (esempio: CHR$(147) sta per schermo spento).
La funzione dei tasti sta nelle parentesi.
Per esempio (SHIFT) (RUN/STOP).

BITS e BYTES

Prima di incominciare a descrivere ed utilizzare le istruzioni PEEK e POKE, e' assolutamente necessaria la conoscenza del significato di 'Bit' e 'Byte'.
Il Bit e' la piu' piccola unita' di immagazzinamento delle informazioni. Esso puo' accettare due soli valori o per meglio dire due cifre: 1 e 0, che possono essere paragonate a 'vero' e 'falso'.
All'interno del computer questa distinzione viene realizzata mediante un flusso di corrente d'entrata ed uscita.
Due Bits insieme possono fornire quattro combinazioni : entrambi spenti, accesi, il primo spento ed il secondo acceso, e viceversa.
Per rappresentare attraverso i Bits le cifre, si puo' impiegare il sistema BINARIO.
Ogni cella di memoria del computer e' in grado di memorizzare (peeK ,poKe) un BYTE: un insieme di OTTO bits.
Il piu' grosso numero binario contenibile da un byte di 8 cifre e' 11111111, che equivale al valore decimale 255.
Ad ogni combinazione dei Bits corrisponde un preciso numero decimale.
Cosi' potete rappresentare e SCRIVERE IN MEMORIA con soli 8 Bits un numero compreso tra 0 e 255 (256 sono le combinazioni !).
Per ogni Byte di memoria (RAM/ROM) c'e' un compito ben preciso !
E' bene riportare ora la distinzione tra RAM e ROM.
In memoria RAM i valori vengonno sia letti che scritti (peeK,poKe); in lingua inglese RAM vuole dire appunto 'Random Access Memory'.
Con il termine ROM si specifica un tipo di memoria A SOLA LETTURA (peeK) : la memoria ROM contiene dati non modificabili dal programmatore (Read Only Memory).
Il contenuto della ROM e' invariato anche a computer spento, al contrario della RAM.
Oltre a cio' ci sono ancora altri Bytes, che hanno altre funzioni: ciascun bit di essi ha un compito ben preciso (uno solo!), a seconda che esso sia acceso (1) o spento. E' inportante dunque poter andare ad interferire (accendere o spegnere) , NELL'AMBITO DI UN BYTE ,UN SINGOLO BIT .
Il BASIC ci mette a disposizione un valido strumento : le istruzioni AND e OR.
Entrambe servono ad eseguire operazioni tra due bytes allo scopo di ottenerne un terzo come voluto.
I Bit di egual valore vengono confrontati l'uno con l'altro.

L'operatore AND funziona cosi':

```
Byte 1       11001010  202
Byte 2 AND   10101111  175
             --------
risultato    10001010  138
```

Nel risultato viene inserito un Bit ad UNO se ,i rispettivi bits del primo e del secondo bytes SONO ENTRAMBI 1.
Lo stesso si puo' fare anche in BASIC:

```
PRINT 202 AND 175
```

La TABELLA di funzionamento ('della verita'') e' la seguente:

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1
```

Questa funzione vi serve se occorre agire (solo spegnere) su particolari bits appartenenti ad un byte.
Se ad esempio devono essere isolati i quattro Bits minori (chiamati: Nibble), il compito viene eseguito con AND 00001111 (in BASIC: AND 15): la stessa cosa succede per i maggiori.

Al contrario la funzione OR opera diversamente; ad es:

```
Byte 1       11001010  202
Byte 2 OR    10101111  175
             --------
Risultato    11101111  239
```

Nel risultato viene inserito un Bit ad 1 se ALMENO UNO dei due bits in ingresso e' acceso.

Proviamo con il BASIC:

```
PRINT 202 OR 175
```

Lo schema di fuzionamento e' il seguente:

```
  0 OR 0 = 0
  0 OR 1 = 1
  1 OR 0 = 1
  1 OR 1 = 1
```

Potete usare questo operatore, se desiderate accendete uno o piu' Bits di un byte: se per esempio volete settare (==>1) il Bit 0 (il piu' piccolo Bit) e' sufficente un : OR 00000001 ovvero OR1.

Infine ci sono ancora 2 programmi BASIC che dovrebbero mostrarvi il criterio di conversione decimale-binario.

```
10 B$="" : INPUT "CIFRADECIMALE";D
20 IF (D/2↑R) >=1 THEN R=R+1 : GOTO 20
30 FOR X=R-1 TO 0 STEP -1
40 IF D/(2↑X) >=1 THEN B$=B$+"1" : D=D-2↑X: GOTO 60
50 B$=B$+"0"
60 NEXT X : PRINT B$ : PRINT
70 GOTO 10
```

E nell'altra direzione: Binario ==> Decimale:

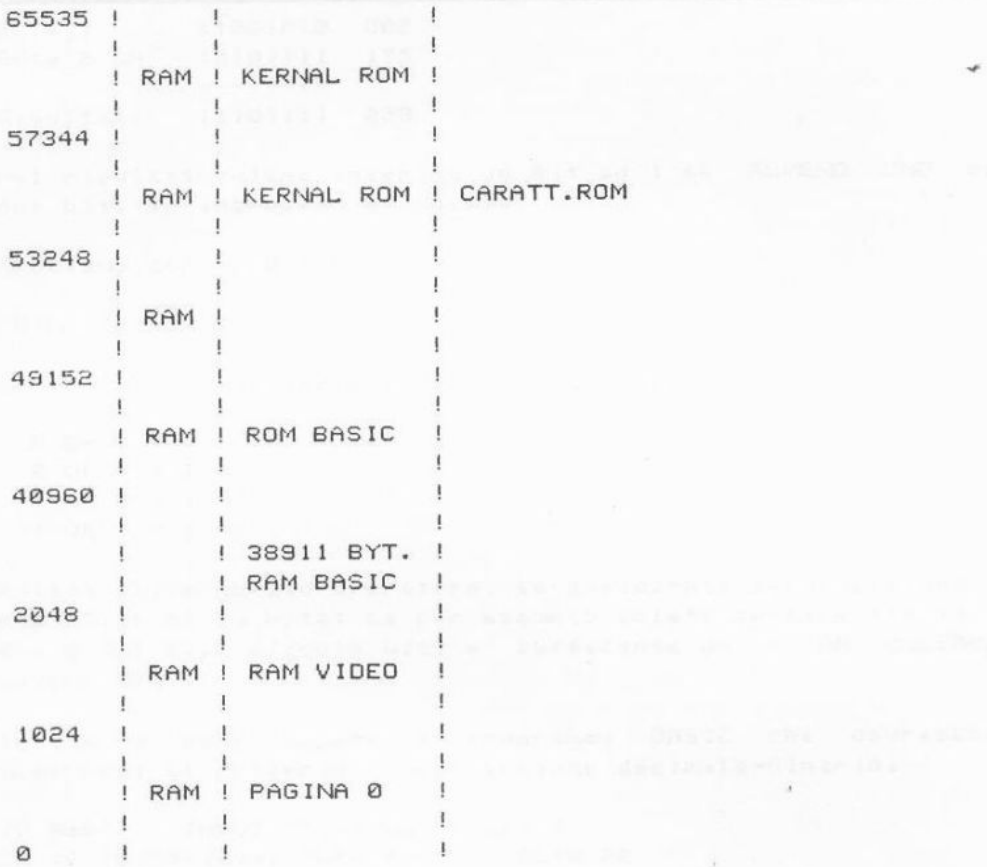
```
10 D=0 : INPUT "CIFRADECIMALE";B$
20 FOR X=1 TO LEN(B$) : IF MID$(B$,X,1)="1" THEN
D=D+2↑(LEN(B$)-X)
30 NEXT
40 PRINT D : PRINT : GOTO 10
```

6510

PAGINA ZERO

LA RIPARTIZIONE DELLA MEMORIA

Il Commodore 64 e' fornito di 64K di RAM, 20K ROM (Read Only Memory) e 4K Input/Output registers.
Tutto insieme fanno ben 88K di memoria ! Un abituale processore ad otto bits, come il 6510, puo' indirizzare al massimo 64 K. Tuttavia per avere l'intervento su tutta la memoria disponibile e' possibile agire attraverso la tecnica di gestione dei banchi (16K) ('memory BanKing'). Ci sono addirittura determinate locazioni che sono occupate da ben tre diversi tipi di memoria ,SELEZIONABILI dall'utente.
Il seguente schema, mostra la memoria e le relative caratteristiche.



Posteriormente all'atto di accensione della macchina, il sistema presenta la cosiddetta 'configurazione base'.
Per lavorare il Commodore 64 ha bisogno dell'interprete BASIC, che si trova in ROM dalla locazione 40960 (esadecimale $A000) fino alla 49152 ($BFFF).
Inoltre e' necessario il normale funzionamento del KERNAL, dalla locazione 57344 fino a 65535 ($E000-$FFFF).
In ultimo restano i cosidetti registri di Input/Output da 53248 fino a 57343 ($D000- $DFFF).
Cosi' succede che voi potete usare dei 65535 bytes (64K) RAM, che gia' sono disponibili, solo 38911 Bytes. C'e' pero' la possibilita' di attivare di nuovo la RAM andata persa.
In questo caso pero' volendo ad esempio disattivare

l'interprete BASIC per sfruttarne la memoria, dovremo ovviamente scrivere i programmi in linguaggi alternativi (es .assembler,LOGO,FORTRAN..)
Una idea potrebbe ancora essere la ricopiatura, byte dopo byte, della ROM dell'interprete nella RAM relativa: potremo in seguito modificare a nostra scelta il BASIC STESSO della macchina .
Senza dubbio deve esserci una conoscenza base nell'Assembler o del linguaggio macchina.
Inoltre e' necessaria una mappa di memoria della ROM, dettagliati documenti sul funzionamento dell'interprete BASIC e KERNAL.
Alla locazione 1 vedrete come si esegue questa commutazione, e poi troverete alcuni esempi.
Noi qui vogliamo descrivere ancora un po' gli ambiti di memoria.
Dalla locazione 0 fino alla 1023 ($0000-$03FF) c'e' la 'PAGINA ZERO' , un importante ambito necessario al processore.
Questo termina con l'inizio della zona (video) RAM da 1024 fino a 2047 ($0400-$07FF).In questo il sistema memorizza i caratteri da visualizzare.
La mappa video puo' essere spostata. Ulteriori informazioni le trovate alla locazione 646 e 53272.
Da 2048 fino a 40959 ($0800 - $9FFF) risiede la memoria dedicata al BASIC.
In questo ambito di memoria sono allocati: i valori dellle variabili, stringhe, vettori ed il programma. Voi avete la possibilita', di delimitare questo ambito DAL BASSO E DALL'ALTO e cosi' riservare posto ad eventuali programmi in linguaggio macchina , oppure al set di caratteri, ai dati relativi agli Sprites,....
L'interprete BASIC risiede dalla locazione 40960 fino a 49152 ($A000-$BFFF).
La memoria compresa 49152 e 53247 ($C000 - $CFFF) non puo' essere utilizzata dal comune programmatore in BASIC.
Dal 53248 fino a 57343 ($D000-DFFF) sono localizzati, a seconda della disposizione dei banchi, i registri I/O, la ROM caratteri o 4K RAM.
Abbastanza in alto nella memoria, da 57344 fino a 65535 ($E000-$FFFF) c'e' il KERNAL.
Nei seguenti capitoli si parla delle locazioni della pagina zero.
Per trovare una logica suddivisione, abbiamo reso facente parte della pagina zero anche la RAM video, ANCHE SE ESSA UFFICIALMENTE NON APPARTIENE ALLA PAGINA ZERO.

LOCAZIONE 1

6510 registro Input/Output

Valore Normale: 55, confronta esempi
Contenuto:

Con questa locazione, si puo' comunicare al 6510, se deve 'vedere' un particolare ambito della memoria come ROM, RAM, ozona per I/O.
Possiamo addirittura, agendo da linguaggio macchina, eliminare sia il BASIC che il KERNAL.

Accanto alla configurazione della memoria la locazione 1 ha ancora un'altra funzione: essa serve per il comando del Datasette.
Ci si puo' chiedere se uno dei tasti del Datasette e' premuto.
Cosi' si puo' per esempio accendere o spegnere la registrazione per poco tempo e tenere il registratore sotto controllo del programma.

Compito dei Bits:

0 BASIC ROM (1)/RAM (0) (40960-49151) ($A000-$BFFF)
1 KERNAL ROM (1)/RAM (0) (57344-65535) ($E000-$FFFF)
2 I/O (1)/carattere ROM (0) (53248-57343) ($D000- $DFFF)
3 Datasette: distribuzione dati
5 Datasette:nessun tasto pigiato (1),tasto pigiato (0).
5 Datasette: motore spento (1),acceso (0)
6,7 non utilizzati, sempre 0

Spiegazione:

Quando il Bit 0 e' acceso, la RAM e' in funzione.
Quando il Bit 0 e' spento anche la RAM non e' attivata.
Quando entrambi i Bits (#0 e #1) sono al valore 0, verrano interrotti in aggiunta anche i Registri I/O: avrete effettivamente 64K RAM, con i quali raramente potete lavorare senza che il registro I/O sia in funzione: il Computer diventerebbe praticamente inutile.
Il Bit 2 e' acceso o spento per la selezione della zona per l' INPUT/OUTPUT o ROM caratteri, dalla locazione 54247 fino alla 57343.
Poiche' normalmente il video-Chip puo' leggere questa memoria dei caratteri, e' possibile modificarla allo scopo di eventuali personalizzazioni.
L' isolamento dei registri I/O e' possiblile pero' solo con determinati accorgimenti e limitazioni, descritti al tema 'set di caratteri'.
Dal Bit-Datasette #3 fino al #5 avete bisogno in realta' solo degli ultimi.
Il Bit #4 e' collegato con un sensore, che notifica al computer, se uno dei tasti Play, Play record, Wind, o Rewind e' premuto: purtroppo pero' non e' possibile accertare quale.

Questi Bit possono anche esser letti in modo sensato. Col Bit #5 potete regolare, se il motore del Datasette riceve corrente o no.
A questo punto il valore 0 non potra' piu' rimanere nella locazione 192 (leggere dettagli alla locazione num.192).
Entrambi gli ultimi Bits sono spenti, cioe' sempre 0.

Esempio:

Il valore normale di questa locazione e' 55, in quanto il Datasette non viene usato. (il valore 39 si ottiene schiacciando il tasto di registrazione).

```
POKE 1,54  : disattiva il BASIC
POKE 1,53  : disattiva BASIC e KERNAL
POKE 1,52  : disattiva BASIC,KERNAL,I/O
POKE 1, PEEK(1) OR 32 : motore off.
```

Il seguente programma ricopia il BASIC nella RAM SOTTOSTANTE.

```
FORX=40960TO49151:POKEX,PEEK(X):NEXTX
```

lo stesso con BASIC e KERNAL:

```
FORX=40960TO49151:POKEX,PEEK(X):NEXTX:FORX=57344TO65535:POKEX
,PEEK(X):NEXTX
```

Entrambi i programmi servono per migliorare la caratteristica del Commodore 64.
Normalmente l'unico sistema di trasferimento, APPARENTEMENTE insensato, consiste prima nel leggere il contenuto di una locazione, e successivamente nel riportarlo immediatamente (poke) nel medesimo indirizzo.
Per il Commodore vale tutto questo: se in un ambito di memoria e' presente la RAM, essendo pero' inserita la ROM, con il comando PEEK leggeremo la RAM solo dopo aver opportunamente agito sulla locazione 1.
Al contrario attraverso l'istruzione POKE si scrive sempre nella RAM.
Cosa questa molto logica in quanto la memoria ROM non puo' essere alterata.
Il programma appena visto usa una PEEK per leggere la ROM, ed una POKE per ricopiare tale valore nella sottostante RAM: tutto questo per l'intera zona di memoria.
Ma rimane un problema: il tempo richiesto per tale operazione e' eccessivo.
Percio' vi abbiamo scritto ancora un programma in linguaggio macchina che ricopia BASIC e KERNAL in meno di 2 secondi.
Poi potete ancora decidere se volete attivare solo il BASIC o anche il KERNAL.
Poiche' il programma e' allocato nel buffer di cassetta, esso viene cancellato dall'uso del Datasette (vedere loc. 828-1019).
Pero' potete spostarlo in qualsiasi altro posto della memoria, alterando semplicemente il valore di A.

```
10 A=828
20 FOR X=A TO A+72
30 READ W : S=S+W : POKE X,W
40 NEXT X
50 IF S= 10394 THEN SYS A : PRINT "OK"
60 DATA 120,162,0,134,98,162,191,134,
   99,160,255,162,55,134,1,177,98,162,52,134,1
70 DATA 145,98,136,192,255,208,239,198,99,165,
   99,201,159,208,229,162,255,134,99
80 DATA 160,255,162,55,134,1,177,98,162,52,
   134,1,145,98,136,192,255,208,239,198
90 DATA 99,165,99,201,223,208,229,162,55,134,1,88,96
```

LOCAZIONE 19

Flag per richiesta (input)

Valore Normale: 0

Contenuto:

Questo flag indica se l'attuale immissione di dati, avviene da consolle o da una memoria di massa (attraverso GET# e INPUT#).
Poiche' se l'input e' rivolto ad un File non appare sul video nessun punto interrogativo, voi potete con POKE19,64 effettuare il disinnesto del segnale '?' anche durante un INPUT da consolle.
Per ristabilire la normale funzione, dovete rispegnere questo Flag dopo l'INPUT.
Esempio di come funzioni L'INPUT senza segnale '?':

```
10 POKE 19,64
20 INPUT "nome": ";A$
30 POKE 19,0
```

LOCAZIONI 43/44

Indicatore dell'inizio del programma BASIC

Valore normale: PEEK(43)=1 ; PEEK(44)=8

Contenuto

I due bytes in questione puntano alla locazione di memoria alla quale ha origine il nostro programma scritto in BASIC. Normalmente esso comincia dalla locazione 2048 ($0800), poiche' i bytes 43 e 44 contengono rispettivamente 0 ed 8

(low ed hi-byte).
Abbiamo ottenuto tale valore attraverso la seguente formula:

```
INIZIO=PEEK(43)+256*PEEK(44)
```

Voi potete modificare questo indice e cosi' spostare l'origine del programma BASIC.
Questo per esempio e' necessario per collocare un particolare set di carateri nell'ambito che va da 2048 fino a 4095 ($0800-$09FF). (Piu' dettagliatamente consultate le spiegazioni circa il 'VIC').
Ma possono esserci in questo ambito di memoria anche un programma in linguaggio macchina oppure una seconda mappa video.
In definitiva, e' possibile avere due o piu' programmi BASIC presenti in memoria, e chiamarli alternativamente.

Riferimenti:

L'interprete attende nella locazione precedente l'inizio del programma BASIC, un BYTE A ZERO.
Se volete assegnare al programma un'altra locazione d'inizio, dovete prima digitare:

```
POKE(LOCAZIONE INIZIALE-1),0:NEW
```

Anche il NEW e' importante: esso si preoccupa che la nuova locazione d'inizio venga comunicat a all'interprete.
In generale, se desiderate alterare la locazione d'inizio di un programma, dovete rispettare le seguenti regole:

1] La memoria BASIC deve essere totalmente RAM.

2] L'ambito di memoria che va da 0 fino al 1023 ($0000-$03FF) e' proibito e dedicato al lavoro del Computer.

3] per poter utilizzare la RAM video (1024-2047 ovvero $0400-$07FF) occorre spostare (vederè loc.53272) la mappa video in altra zona di memoria.

Percio' risulta evidente che un programma BASIC puo' essere allocato da 2048 ($0800) sino a 40959 ($9FFF).

Riferimento:

Le seguenti formule vi forniranno due valori da inserire in 43 e 44 per spostare l'inizio del programma BASIC.

```
HB=INT(inizio/256)
LB=inizio-HB*256
POKE43,LB:POKE44,HB
```

esempio:

se desiderate spostare l'inizio del programma basic ad

esempio alla locazione 4097, occorrera' digitare dunque:

```
POKE43,1
POKE44,16
POKE4096,0
```

E' consigliabile impiegare come punto di partenza del programma Basic, un multiplo di 256 (1 pagina) addizionato al valore 1: i calcoli saranno relativamente piu' semplici.

LOCAZIONI 45/46

Puntatori alla fine del programma basic.

Valore Normale:

```
peeK(45)=3
peeK(46)=8
```

nel caso in cui in memoria non sia presente alcun programma.

Contenuto:

Anche in questo caso vengono usati due bytes come puntatori all' ultima cella componente il programma basic.
L' interprete deve necessariamente disporre di questi due indici poiche' esso, per la memorizzazione delle variabili, agisce in coseguenza della lunghezza dell'attuale programma in memoria.
Alcuni programmi di 'tool' (aiuto alla programmazione) alterano questi valori di fine programma, se dopo un NEW, desideriamo riavere il nostro vecchio listato.
E' importante conoscere la seguente formula:

```
fine programma = peeK(45)+256*peeK(46)
```

Essa fornisce l'indirizzo finale del programma attualmente presente: il tutto e' utile permettendoci di poter sfruttare in altro modo la memoria libera.
In tale caso, anche il tetto massimo della memoria deve essere mutato (vedere 55 e 56).

INDICAZIONI:

Se voi non avete in memoria alcun programma, leggendo i primi due bytes dell'inizio BASIC, troverete un doppio zero.
Questo perche' il sistema operativo nei primi due bytes di una riga Basic, mette l'indirizzo di memoria della successiva.

LOCAZIONI 47/48

Puntatori alla fine delle variabili

Valore Normale:

dipendente di volta in volta dal programma Basic memorizzato.

Contenuto:

Attraverso tale indicatori l'interprete Basic viene a conoscenza di quale sia l'ultima cella di memoria dell'allocamento di variabili.
Solitamente esse vengono poste immediatamente a seguito della fine del programma basic.
Questi puntatori vi danno una indicazione dunque di quanta memoria, alla fine del programma, resti libera per la memorizzazione delle variabili.
Si noti che il comando CLR distrugge tutte le variabili e di conseguenza ripristina il valore delle locazioni interessate all'originario.
Per trovare la cella di memoria puntata si usi la formula:

```
indirizzo = peeK(47)+256*peeK(48)
```

Il discorso vale solo per le variabili numeriche: per le altre (stringhe) si legga la locazione 51/52.

.....

LOCAZIONI 49/50

Puntatori alla fine della memoria disponibile

Valore Normale:

dipendente di volta in volta dal programma basic memorizzato.

Contenuto:

Entrambi i bytes servono all'interprete come puntatori all'ultima cella di memoria utilizzabile (leggere 51 e 52).
Per calcolare il corretto indirizzo si usi la formula:

```
indirizzo = peeK(49)+256*peeK(50)
```

questo valore indica il limite massimo della memoria: tutto cio' che segue e' libero.

INDICAZIONE:

Se voi desiderate, agendo sulle locazioni 43 e 44, disporre in memoria di due differenti programmi Basic, occorre fare attenzione ad una cosa:

caricate in memoria il primo programma, mandatelo in esecuzione, calcolate con 45 e 46 l'indirizzo di fine, ed allocate il secondo almeno una pagina (255 bytes) dopo.
Si puo' procedere cosi' e disporre in memoria di un gran numero di programmi: occorre pero' che l'ultimo, quello piu' in alto', abbia lo spazio necessario alla memorizzazione delle variabili.
Per eseguire poi il programma voluto, digitate le relative POKEs in 43 e 44, e quindi date RUN.
Tutti i programmi memorizzati sfrutteranno la medesima zona di memoria per la memorizzazione delle variabili (leggere 55/56).
Questo comporta un risparmio di memoria ripetto alla condizione in cui ogni programma disponga della proprie variabili.

LOCAZIONI 51/52

Puntatori di fine memoria variabili non numeriche (stringhe)

Valore Normale:

dipendente di volta in volta dal programma Basic memorizzato.

Contenuto:

La memorizzazione delle stringhe avviene in un modo particolare: esse iniziano dal tetto massimo fissato in 55 e 56 e procedono verso la fine del programma Basic (verso il basso).
Se create un eccessivo numero di variabili non numeriche oppure una troppo lunga, incapperete nel messaggio di errore:

OUT OF MEMORY
ERROR

Una formula che fornisca una indicazione del primo byte usato per memorizzare stringhe, potrebbe essere:

```
indirizzo = peeK(51)+256*peeK(52)
```

Pensate cosa potrebbe accadere: voi allocate un programma in liguaggio macchina alla fine del programma basic, e poi create delle stringhe la cui memorizzazione rischia di distruggere il listato in codice macchina.
Per evitare questo si leggano le locazioni 55/56.

LOCAZIONI 55/56

Puntatori limite RAM BASIC.

Valore Normale:

```
peeK(55)=0
peeK(56)=160
```

Contenuto:

All'interprete basic e' comunicato, tramite tali due vettori, quale sia il tetto massimo della memoria adibita al programma Basic.
Normalmente questo valore e' uguale a 40960 dovuto a peeK(55)+256*peeK(56).
Questa informazione e' di grande utilita' per l'interprete BASIC, per conoscere da quale indirizzo esso deve far partire la memorizzazione delle stringhe (leggere loc.51/52).
Facendo puntare questi vettori ad un indirizzo piu' basso del solito, voi garantite che la memoria sovrastante non venga mai alterata: potrete mettervi routines in linguaggio macchina, pagine hi-res, ...
Spostare i puntatori 55 e 56 verso l'alto non ha senso in quanto cadrebbero sull'ambito ROM.
I seguenti comandi causano lo spostamento del tetto massimo di memoria:

```
HB=INDIRIZZO/256
LB=INDIRIZZO-256*HB
POKE55,LB
POKE56,HB
```

Il limite massimo deve superare ovviamente l'inizio del programma Basic (locaz. 43/44).

ESEMPIO

Se volete riservare la memoria da 40485 a 40959 per un programma in linguaggio macchina, dovete dunque scrivere:

```
POKE 55,37
POKE 56,158
```

LOCAZIONI 59/60

Linea numerica per CONT.

Valore Normale:

```
peeK(59)=0
peeK(60)=0
```

nel caso in cui in memoria non sia presente alcun programma di cui e' stata interrotta l'esecuzione

Contenuto:

Se voi interrompete l'esecuzione di un programma mediante il tasto RUN/STOP, potrete riattivarlo tramite il comando CONT, solo se nel frattempo non si sia verificata nessuna condizione di errore.
(attenzione: non incappate nei SYNTAX ERROR in modo diretto)
In questi due vettori in questione, e' posto l'attuale riga in cui il programma si e' arrestato.
Questa linea e' evidenziata anche dal messaggio BREAK IN.... all'arresto dell'esecuzione.

Potrete ottenere il valore esatto con:

```
PRINT PEEK(59)+256*PEEK(60)
```

CONTENUTO

Come potrete riprendere l'esecuzione di un programma dopo un errore, lo troverete alla locazione 61/62.

LOCAZIONI 61/62

Puntatori alla riga per CONT

Valore Normale:

```
peeK(61)=0
peeK(62)=0
```

Nel caso in cui nessun programma sia stato interrotto.

Contenuto:

Tali vettori puntano alla memoria in cui ha inizio la riga del programma, alla quale e' avvenuto l'arresto dell'esecuzione.
Puo' essere interessante agire su questi puntatori tramite il MONITOR.
Il vantaggio lo si comprende quando, a causa di un errore, l'istruzione CONT non puo' piu' essere utilizzata correttamente: entrambi i bytes in 61 e 62 vengono riposti allo 0.
Il conseguente errore sarebbe : CAN'T CONTINUE ERROR.
Per evitare questo, appena interrompete un programma, memorizzate il contenuto di 61 e 61:

```
PRINT PEEK(61);PEEK(62)
```

Se si verifichera' un errore prima del CONT, bastera' riscrivere i valori trovati in 61 e 62 e ridare CONT.

INDICAZIONI:

Se voi avete cambiato il programma nella memoria, ovviamente CONT non dara' piu' esito positivo, ed anzi avra' conseguenze insolite.

LOCAZIONI 63/64

Puntatori riga dell'attuale DATA.

Valore Normale: -

Contenuto:

In questi puntatori e' memorizzato il numero dell'attuale riga alla quale e' stato letto l'ultimo DATA.
E' possibile dunque testare ,da Basic, a quale riga la lettura dei dati e' giunta: il tutto serve molto durante la correzione degli errori di un programma.

```
PRINT PEEK(63)+256*PEEK(64)
```

LOCAZIONE 144

Variabile di stato ST

valore normale: 0

contenuto:

Con peeK (144) si puo' interrogare lo stato del sistema
In questo byte puo' essere letto, in base ai diversi bits, quale errore sia sorto dallo scambio di dati effettuati con cassetta.
Si legge st=0 quando non e' sorto alcun errore.
Il valore letto in 144 puo' essere usato all' interno di un programma per verificare ad esempio se un file e' stato letto completamente.

Ecco un utile quadro:

	cassetta	cassetta	bus dati
1	lett.file	verify/load	
2	-	-	tempo tras corso per scrittura
3	-	-	tempo tras corso per lettura
4	blocco piu' corto	blocco piu' corto	-
8	blocco piu' lungo	blocco piu' lungo	-
16	errore lett ura fatale	errore fata le	-
32	errore somma esaminata	errore somma esaminata	-
64	fine file	-	fine dati
128	fine bande	fine bande	device not present

riferimento:

Per errori multipli si sommano i relativi valori.

LOCAZIONE 152

numero files aperti

valore normale :-

contenuto:

in questa locazione l'interprete Basic registra il numero dei files aperti che deve essere sempre minore di 10, pena il messaggio: 'too many files error'.
Questo messaggio puo' essere evitato, esaminando il contenuto di questa locazione nel programma prima dell'apertura di un file.

riferimento:

questa locazione contiene l'indice per la tabella dei files, nella quale sono archiviate le altre informazioni come numeri dell'apparecchio, indirizzi secondari etc. (vedere locazioni 601/630).
Se voi riducete il contenuto di questa locazione, vengono di conseguenza chiusi gli ultimi files aperti; per provare , potremo servirci di un get# od input #, che ci daranno il messaggio 'file not open error'.
Con sys (65511) potete chiudere contemporaneamente tutti i files aperti: questa chiusura pero' non provvede alla regolare fase di conclusione di un file su disco cassetta.

LOCAZIONE 157

Flag modo programma/modo diretto.

valore normale : 128

contenuto:

Questo flag stabilisce se siamo in modo diretto o programma, e se dunque devono essere stampati, o meno, i messaggi come 'searching', 'loading'...
Il valore 128 consente la scrittura, che viene invece interdetta con POKE 157,0.
Potete dunque mettere, all'interno di un programma, una POKE 157,128 per controllare cosa sta succedendo.
Questa funzione si adatta particolarmente a tests di prova del programma.

riferimento:

dopo un cont-error, questo flag resta in modo programma.

Esempio:

```
poke 157,0 : load '$',8
```

LOCAZIONI 160/162

Orologio software: ti,ti$

Valore normale: -

Contenuto:

Il Commodore 64 ha un orologio realizzato tramite il software.
Questo puo' essere interrogato utilizzando le variabili ti (sessantesimi di secondo) e ti$ (l'ora nel giusto formato hhmmss).
Potete regolare l'ora con ti$="hhmmss": per esempio: 16 h 20 minuti 35 secondi danno ti$='162035'.
L'ora e' memorizzata in 3 bytes: se vogliamo ricavare l'ora tramite PEEK possiamo digitare:

```
PRINT PEEK(160)*2↑16+PEEK(161)*2↑8+PEEK(162)
```

Con PEEK (162) si puo' leggere un valore che viene modificato ogni sessantesimo di secondo: questo puo' essere pou' pratico per la produzione casuale di numeri o simili.

LOCAZIONE 184

numero dell'attuale file

valore normale: -

contenuto:

in questa locazione trovate il numero dell' attuale file: tramite una sola PEEK, il programma puo' sapere con quale periferic a sta dialogando.

LOCAZIONE 185

attuale indirizzo secondario

valore normale: -

contenuto:

Qui viene letto l'indirizzo secondario del file corrente; tuttavia non e' possibile modificare tale valore attraverso una POKE, poiche' dovremmo andare ad interferire anche nella tavola files.

riferimento:

Il sistema operativo esegue un or 96 (or$60) con il valore inserito nella locazione.
Se voi volete sapere qual'e' il vero e proprio valore, dovete applicare la seguente formula:

indirizzo secondario = peeK (185) and (255-96)

per (255-96) potete scrivere anche 159.

LOCAZIONE 186

Attuale periferica in dialogo.

valore normale: -

contenuto

Qui trovate il numero relativo alla periferica in uso corrente.
Questo valore e' valido non solo per i files aperti, ma anche per i comandi LOAD e SAVE.
Grazie a questa locazione voi potete per esempio all'inizio di un programma inserire una peeK (186) per testare se il programma stesso e' stato caricato con il floppy o cassetta.
Anche se voi usate piu' di un floppy, e' possibile ugualmente verificare con quale dischetto state attualmente lavorando.

LOCAZIONE 192

controllo del motore della cassetta

Valore normale: 0 (disinserita)

Contenuto:

Questo flag serve al sistema operativo per chiudere ed avviare il motore del registratore.
Il motore puo' essere usato solo se il valore in questa locazione non e' 0.
(vedere locazione 1).

Esempio:

Questo programma accende e spegne alternativamente il motore della cassette:

```
10 POKE 192,1
20 POKE 1,23 : FOR X=0 TO 500 : NEXT
30 POKE 1,55 : FOR X=0 to 500 : NEXT
40 GOTO 20
```

LOCAZIONE 197

ultimo tasto premuto.

Valore normale : 64 se nessun tasto premuto

Contenuto:

Il questa locazione la routine di scansione tastiera (vedi locazioni 655/656) deposita il valore dell' ultimo tasto premuto.
Al contrario trovate nella locazione 203 l'attuale tasto premuto.
Se voi leggete da BASIC la locazione 197 apparira' dello stesso contenuto della 203, ma solo falsamente entrambe le locazioni vengono spesso qualificate come se fossero uguali.
Il punto e' questo: il sistema si serve di due locazioni per stabilire se un tasto va effettivamente considerato premuto due volte od una sola.
Entra in gioco anche la locazione 650 (funzione di ripetizione).

LOCAZIONE 198

contatore per il buffer della tastiera

Valore Normale: 0 in modo diretto

Contenuto:

Tutti i tasti premuti vengono memorizzati, nel caso in cui non possano essere analizzati subito, in una particolare area della memoria detta 'buffer della tastiera'.
Questo sistema ha molti vantaggi: anche la richiesta che viene fatta mentre il computer e' occupato (per esempio col programma, con calcoli etc.) puo' successivamente essere elaborata, non appena il Computer e' di nuovo libero.

Cio' nonostante il campo di memoria in questione e' limitato: esso contiene 10 bytes e si trova dalla locazione 631 fino alla 640.
Normalmente basta per non avere problemi in questo campo: sorgono le difficolta' allorche' il computer e' occupato a lungo.
Se in questo periodo molti tasti vengono premuti, il computer memorizzera' esclusivamente i primi dieci.
La locazione 198 indica al COMPUTER quanti sono i tasti disposti nel buffer della tastiera.
Con PEEK (198) voi potete leggere quanti sono i tasti premuti, e dopo di che predisporre il vostro programma utilizzando questi tasti.
I tasti gia' memorizzati vengono automaticamente eseguiti, (ed anche riscritti sul video) non appena il programma incontra un errore in INPUT od un END.

Riferimento:

I tasti <SHIFT>, <C=> e <CTRL> rimangono senza effetto sul buffer: essi servono solamente come tasti selettori.
Se voi scrivete in questa locazione con un valore (POKE) potete programmare in anticipo il funzionamento della tastiera. Vi sara' spiegato piu' dettagliatamente nelle locazioni 641/640.

Esempio:

Con questa sequenza potremo far attendere al computer la pressione di un tasto:

```
POKE 198,0 : WAIT 198,1
```

LOCAZIONE 199

Flag per reverse acceso/spento.

Valore Normale: 0 in modo diretto

Contenuto:

Questo Flag comunica all'interprete se il modo RVS e' attivo o no, e se anche le cifre devono essere scritte in reverse.
Con POKE 199,1 potete sostituire il segno <CTRL> <9>, con POKE 199,0 il segno <CTRL> <0>.

Riferimento:

Un <RETURN> spegne il modo reverse.
Dopo una PRINT che non termini con ';' oppure ',' si spegne automaticamente il reverse, se acceso in precedenza.
Percio' il POKE 199,1 deve essere ripetuto davanti ad ogni riga di questo tipo.

Esempio:

```
10 POKE 199,1 : PRINT 'reverse acceso'
20 PRINT "reverse di nuovo spento'
30 POKE 199,1 : PRINT 'ci sono di nuovo'
```

LOCAZIONE 203

Tasto premuto.

Valore normale: 64

Contenuto:

Con PEEK (203) sapremo quale tasto della consolle e' attualmente premuto
Purtoppo i codici che qui possono essere letti, non sono identici alla tabella ASCII, ne' al codice video.
Percio' vi abbiamo fatto qui una lista:

00 <INST/DEL>	20 <C>	40 <+>	60 <SPAZIO>
01 <RETURN>	21 <F>	41 <P>	61
02 <CRSR L/R>	22 <T>	42 <L>	62 <Q>
03 <F7>	23 <X>	43 <->	63 <RUN/STOP>
04 <F1>	24 <7> <'>	44 <.> <>>	64 <NESSUNO>
05 <F3>	25 <Y>	45 <:> <[>	
06 <F5>	26 <G>	46 <@>	
07 <CRSR U/D>	27 <8> <(>	47 <,> <(>	
08 <3> <#>	28 <B>	48 <£>	
09 <W>	29 <H>	49 <*>	
10 <A>	30 <U>	50 <;> <]>	
11 <4> <$>	31 <V>	51 <CLR/HOME>	
12 <Z>	32 <9> <)>	52	
13 <S>	33 <I>	53 <=>	
14 <E>	34 <J>	54 <PI GRECA>	
15	35 <0>	55 <?> </>	
16 <5> <%>	36 <M>	56 <1> <!>	
17 <R>	37 <K>	57 <>	
18 <D>	38 <O>	58	
19 <6> <&>	39 <N>	59 <2> <''>	

In ordine per tasti:

NESSUNO 64	1 56	S..... 13
RETURN 1	2 59	T..... 22
CRSR U/D 7	3 8	U..... 30
CRSR L/R 2	4 11	V..... 31
CLR/HOME 51	5 16	W..... 9
INST/DEL 0	6 19	X..... 23
RUN/STOP 63	7 24	Y..... 25
SPAZIO 60	8 27	Z..... 12
A..... 10	9 32	
B..... 28	0 35	
C..... 20	+ 40	
D..... 18	- 43	
E..... 14	£ 48	
F..... 21	@ 46	
G..... 26	* 49	
H..... 29	PI GRECA .. 54	
I..... 33	: [..... 45	
J..... 34	;] 50	
K..... 37	= 53	
L..... 42	, < 47	
M..... 36	. > 44	
N..... 39	? / 55	
O..... 38	F1 4	
P..... 41	F3 5	
Q..... 62	F5 6	
R..... 17	F7 3	

LOCAZIONE 204

Flag per cursore acceso/spento

Valore normale: 1 modo diretto; 0 programma.

Contenuto:

Potete lasciare lampeggiare, agendo su questo Flag, il cursore anche durante l'esecuzione di un programma.
Il contenuto di questa locazione e' 1, se il cursore e' spento, 0 se il cursore lampeggia.
Questa funzione puo' essere utile per far lampeggiare sul video il cursore che verra' riconosciuto piu' facilmente dall'utente.

Esempio:

Questo piccolo programma aspetta che venga premuto un tasto che poi stampa:

```
10 PRINT'premi un tasto ';
20 POKE204,0:POKE198,0:WAIT198,1:GETA$
30 POKE204,1:PRINTA$
```

LOCAZIONE 207

Flag cursore nella fase lampeggiante

Valore normale: 0 o 1

Contenuto:

Questo Flag vi comunica se il cursore lampeggiante e' acceso o e' spento.
Noi non possiamo saper (peeK) se esso e' acceso o spento, poiche' il cursore viene sempre disabilitato prima che un comando BASIC possa essere interpretato.
Se, come detto alla loc.204, un programma che si serve (poKe204,1) del cursore, successivamente lo disabilita, puo' accadere che il cursore, sebbene non lampeggi, resti ACCESO sullo schermo.
E poiche' da questo momento il sistema di funzionamento non si preoccupa piu' del cursore, rimane un blocco invertito sul video.
Per evitare tutto questo, questo Flag prima del disinnesto deve essere posto a 0.

Esempio:

Il programma gia' visto (loc.204) nella stesura corretta:

```
10 PRINT 'premi un tasto';
20 POKE204,0:POKE198,0:WAIT198,1:GETA$
30 POKE207,0:POKE204,1:PRINTA$
```

LOCAZIONI 211/214

Colonna e linea per Cursore.

Valore Normale: (a seconda dell'attuale posizione del cursore)

Contenuto:

L'attuale colonna del cursore nell'ambito del video e' posta nella locazione 214.
Potete usare entrambe le locazioni, ma poiche' l'immagine video del Commodore 64 ha 40 colonne (da 0 fino a 39) e 25 linee (da 0 fino a 25), non dovete usare una istruzione POKE con nessun valore piu' grande in entrambi gli indirizzi.
Per comunicare le nuove coordinate al sistema di funzionamento, dovete inoltre effettuare una SYS 58640.

Esempio:

Nella ventitreesima colonna della quinta riga deve essere stampato 'Hallo'.

```
POKE211,23:POKE214,5:SYS 58640:PRINT'HALLO'
```

LOCAZIONI 243 e 244

Indicatori dell'attuale posizione del cursore nella RAM colore.

Valore normale: (a seconda dell'attuale posizione del cursore)

Contenuto:

Questi indicatori memorizzano nella sequenza High Byte/Low Byte, la locazione della RAM colore dell'attuale posizione del cursore.
Con la seguente istruzione:

```
PRINT PEEK(243) + 256 * PEEK(244)
```

possiamo calcolare la locazione esatta.
Queste locazioni sono particolarmante sfruttabili se si ricorre alla programmazione in linguaggio macchina.

LOCAZIONI 256-511

Zona 'Stack' riservata al processore.

Valore Normale: -

Contenuto:

Questa zona di memoria contiene le informazioni necessarie al corretto funzionamento del processore 6510.
Da ambiente BASIC occorre usare tale RAM con estrema cautela, cosi' come da linguaggio macchina.

LOCAZIONI 601-610

Tabella numeri files.

Valore Normale: -

Contenuto:

Qui trovate la gia' menzionata tabella files (vedi loc 184).
Nell'ambito che va da 601 fino a 620 vi sono i numeri dei files momentaneamente aperti.
Nei seguenti ambiti sono memorizzati i numeri delle periferiche e gli indirizzi secondari.
La tabella viene creata dinamicamante, ovvero rispettando l'ordine col quale il files vengono aperti (OPEN): nelle locazioni 601/611/621 trovate i dati relativi a primo file aperto, nelle celle 602/612/622 quelli del secondo ....
Il numero totale di files aperti contemporaneamente puo' essere letto con una:

```
PRINT PEEK (152)
```

E' possibile leggere dunque in questa tabella, oltre a quanti files sono aperti, anche i parametri relativi.
Tutto questo puo' rivelarsi utile nel caso, ad esempio, vogliamo comunicare con un apparecchio senza dover necessariamnete aprire UN ALTRO canale per l'invio di dati.

LOCAZIONI 611-620

Tabella File: numeri periferiche.

Contenuto:

Qui sono deposti i numeri delle periferiche con le quali desideriamo comunicare tramite l'apertura dei files.
(controllate la locazione 601/610).

Esempio:

Per selezionare tra la stampante ed il plotter, possiamo agire cosi':

```
10 OPEN 4,4,0
20 POKE 611,6: PRINT#4, 'PLOTTER'
30 POKE 611,4: PTINT#4, 'STAMPANTE'
40 GOTO 20
```

LOCAZIONI 621-630

Tabella Files: locazioni secondarie

Valore normale: -

Contenuto :

Qui sono contenute le locazioni secondarie dei files aperti.
I valori vengono associati dall'interprete con OR 96 (OR $60).
Il numero originario si ottiene attraverso AND 159 (confronta anche 185).
Voi potete agire andando a scrivere un preciso valore in questa tabella allo scopo di evitare le lunghe fasi di chiusura ed apertura files.
(Vedete il gia' menzionato 601/610).

Esempio:

Una stampante COMMODORE stampa in modo grafico o testo, a seconda che rispettivamente il numero logico valga 0 o 7.

```
10 OPEN 4,4,0
20 POKE 621,0 OR 96 : PRINT#4, 'modo grafico'
30 POKE 621,7 OR 96 : PRINT#4, 'modo testo'
40 GOTO 20
```

LOCAZIONI 631-640

Buffer della tastiera

Valore Normale: -

Contenuto:

In questo campo della memoria vengono memorizzati fino a dieci tasti che noi digitiamo quando il computer e' occupato e non puo' eseguirli.
Cosi' nessuna istruzione viene persa, anche se il Computer e' gia' occupato.
Se vengono scritti piu' di 10 caratteri, gli eccessivi non verranno presi in considerazione.
Altre informazioni sulla teoria e pratica del buffer di tastiera le troverete alla locazione 198.
Nelle seguenti righe verranno esposte alcune tecniche di utilizzo del buffer: cosi' potrete effettuare da programma, molte operazioni normalmente consentite solo in modo DIRETTO, come la creazione di nuove righe Basic, o la loro eliminazione.

Riferimenti:

Possiamo da programma tramite POKE, scrivere un qualche comando nel buffer tastiera, servendoci dei codici ASCII; avvertendo poi (poke198,...) l'interprete che alla fine del programma deve eseguire i comandi da noi 'falsamente' digitati.
Ma pensateci su: voi avete solo 10 caratteri a disposizione.
Per istruzioni piu' lunghe farete dunque cosi':
dovete stampare il testo da eseguire sulla sommita' del video, e quindi portare il cursore all'inizio di questa riga e deporre solo un CHR$ (13) (return) nel buffer di tastiera.
Cosi' l'intera riga viene eseguita.
Non dimenticatevi, nel caso fosse necessario, di preoccuparvi che il computer dal modo diretto ritorni di nuovo al programma.
Un GOTO o RUN od un RETURN puo' occuparsi di cio'.

Esempio:

I seguenti piccoli programmi ripartono sempre da soli.
I codici R,U,N, e CHR$(13) vengono 'POKATI' nel buffer, e 4 in 198 (4 immissioni).
Poi il computer viene lasciato libero con END nel modo diretto: ma subito eseguira' RUN e RETURN; e cosi' via.

```
10 POKE, ASC('R') : POKE 632, ASC('U'):POKE 633, ASC('N')
20 POKE 634,13
30 POKE 198,4
40 END
```

Il prossimo esempio mostra come si puo' aggiungere una nuova riga BASIC da un programma.

```
10 PRINT CHR$(147) : PRINT : PRINT '30 REM QUESTA RIGA E'
NUOVA'
20 PRINT 'GOTO 70'
40 PRINT CHR$(19);
50 POKE 631,13 : POKE 632,13 : POKE 198,2
60 END
70 LIST
```

Abbiamo adoperato il codice CHR$ per evitare la sostituzione del segno grafico relativo.
CHR$ (147) sta per il tasto <SHIFT> <CLR/HOME> e CHR$ (19) per <HOME>.
Dopo l'immissione della riga 30 incontrate un GOTO 70: si otterra' la lista del programma modificato.

LOCAZIONE 646

attuale colore dei caratteri.

Valore Normale: 14

Contenuto:

In questo byte si trova il codice del colore dei caratteri che stamperemo sullo schermo.
Una POKE in questa locazione altera il colore di tali caratteri.
A prescindere dal vantaggio di una migliore leggibilita', l'utente ha la possibilita' di selezionare i colori semplicemente attraverso i codici relativi.
Il valore da immettere nella locazione 646 deve essere minore o uguale a 15 (contano solo i primi 4 Bits).
Tutti i valori al di sopra di questo, producono di nuovo i colori specificati dai primi 4 bits del numero.

Riferimento:

A causa del colore di sfondo possono essere letti sempre SOLO 15 colori:

Esempio:

```
10 FOR X=0 TO 15
20 POKE 646,X
30 PRINT 'HALLO, COME STAI ?'
$0 NEXT X
```

LOCAZIONE 648

Puntatore inizio RAM video.

Valore Normale: 4 (leggere loc.1024)

Contenuto:

La zona di memoria nella quale solitamente e' allocata la mappa video, ovvero il testo che leggete sullo schermo televisivo, comincia da 1024 fino al 2047 ($0400-$07FF).
Ma questo ambito puo' essere spostato, agendo contemporaneamente su due locazioni di estrema importanza.
La prima suggerisce al VIC, da dove devono essere letti i dati video: essa e' 53272.
L'altra invece (648) indica al sistema di funzionamento dove i dati da visualizzare debbono essere DEPOSTI.

Nella locazione (648) viene sempre deposta la parte alta (hi-byte) dell'indirizzo video: questo fatto, se decidete di alterare la posizione d'inizio dello screen, vi obbliga a sceglierla tra i multipli di 256.
Il contenuto normale di questa posizione e' 4 (4 * 256 = 1024).
Se volete calcolare il valore da immettere nella locazione 648 potrete seguire questo schema:

valore = INT (locazione inizio/ 256)

Riferimento:

Fate attenzione che la RAM video non coincida con la memoria riservata al programma BASIC, ovvero dalla locazione 2048 fino alla 40959 ($0800-$9FFF);
Si evitino anche i valori da ZERO sino alla terza pagina compreso (pagina zero), o quelli che allochino la RAM video nella ROM.

Il valore contenuto in 648 non viene riportato allo standard (=4) neppure attraverso RUN/STOP+RESTORE.
Nel caso si renda necessario il ripristino dei valori corretti, l'unica suluzione e' digitare poke648,4

Esempio:

Sono riportati in riferimento alla locazione 53272.

LOCAZIONE 649

Grandezza del buffer della tastiera

Valore Normale: 10

Contenuto:

In questa posizione e' stabilito quanto debba essere grande il buffer di tastiera. Il valore normale e' 10 (confrontate 631-640 e 198).
Non dovete aumentare questo valore.
Potete pero' ridurre tale valore, se volete limitare il numero degli eventuali dati in input.

Riferimento:

Con POKE 649,0 si blocca la tastiera: potrete poi riabilitarla SOLO SE IN MODO PROGRAMMA, poiche' in modo diretto non vi sara' ovviamente possibile scrivere nessuna istruzione.

LOCAZIONE 650

Flag per la funzione di ripetizione.

Valore Normale: 0

Contenuto:

Con questo flag potete regolare quali tasti possono avere una funzione automatica di ripetizione.
Potete fare in modo che, mantenendo in pressione un tasto, esso venga successivamente ripetuto ad un ritmo veloce e costante.
Vi sono 3 possibilita':

POKE 650,128 tutti i tasti con REPEAT.

POKE 650,64 nessun tasto ripetuto.

POKE 650,0 situazione standard.

LOCAZIONE 653

Flag per <SHIFT>, <CTRL> e <C=>.

Valore Normale: -

Contenuto:

Questo flag vi dice, se uno o piu' tasti suddetti vengono premuti.
In collegamento con la locazione 203 puo' cosi' essere interrogata l'intera tastiera.

Bit #0 e' per il tasto <SHIFT> fisso.

Bit #1 per <C=>.

Bit #2 per <CTRL>.

Esempio:

Valore decimale in <653>	tasto pigiato
0	nessuno
1	<SHIFT>
2	<C=>
3	<SHIFT> <C=>
4	<CTRL>
5	<SHIFT> <CTRL>
6	<C=> <CTRL>
7	<SHIFT><C=><CTRL>

LOCAZIONI 655/656

Indicatori relativi alla tastiera

Valore normale:

PEEK(655) = 72;
PEEK(656) = 235;
posizione indicata 60232 ($EB48)

Contenuto:

Questi puntatori indicano al sistema operativo l'inizio della routine addetta alla scansione della tastiera: voi potete alterare questi vettori, volendo ad es. bloccare la tastiera o utilizzare i tasti funzione.

Esempio:

Ecco un piccolo prospetto:

PEEK(655)	PEEK(656)	Locazione	Hex.	
71	235	60231	$EB47	Tastiera bloccata
72	235	60232	$EB48	Normale routine

macchina

Il seguente programma in L.M. lavora secondo questo principio: esso fornisce al sistema operativo una routine di scansione tastiere ampliata.
Essa e' rilocabile a piacere: la posizione d'inizio viene fissata mediante la variabile A.
C'e' tuttavia una limitazione: nel campo che va dal 49152 fino al49240 (si trova nel non utilizzato) (piu' avanti troverete spiegazioni sul tema 'Ripartizione della memoria') vengono memorizzati i tasti funzione.
E' necessario non allocare in tale RAM il programma in linguaggio macchina.

Il programma (BASIC) e' una routine d'aiuto alla imposizione dei tasti di funzionamento.
Essi hanno a disposizione 10 caratteri per ciascuno. Ogni posizione viene contrassegnata mediante un punto.
Se inserite un numero inferiore a 10, terminate l'immissione dei dati.
Per assegnare il RETURN ad un comando, servirsi della freccia verso sinistra.
Il programma poi vi domanda qual'e' il tasto, che deve gestire. Selezionatelo mediante una cifra che va da 1 fino ad 8.

```
 10 A=828 : FOR X=A TO A+114 : READ W : S=S+W : POKE X,W :
NEXT X
 20 IF S<>14626 THEN PRINT "ERRORE NEI DATI!!!": STOP
 30 POKE 656, INT(a256) : POKE 655,A-(INT(A/256)*256)
 40 FOR X=0 TO 7 : POKE 49152+X*11,1 : NEXTX
 50 REM ZEILE 40 SETZT FKT-BELEGUNG ZUREUCK !
 60 DATA 165,203,197,197,240,16,201,3,
    240,15,201,4,240,11,201,5,240,7
```

```
 70 DATA 201,6,240,3,76,72,235,174,141,2,
    224,1,208,2,9,16,162,192,134
 80 DATA 255,201,4,208,2,162,0,201,20,
    208,2,162,11,201,5,208,2,162,22
 90 DATA 201,21,208,2,162,33,201,6,208,2,
    162,44,201,22,208,2,162,55,201
 100 DATA 3,208,2,162,66,201,19,208,2,
     162,77,134,254,162,118,134,252,162
 110 DATA 2,134,253,160,0,177,254,133,198,
     168,177,254,145,252,136,208
 120 DATA 249,198,198,76,72,235
```

Ecco il programma per la memorizzazione dei tasti funzione.

```
 10 PRINT ".........";PRINT CHR$(145);
 20 POKE 198,0 : WAIT 198,1 : GET A$
 30 IF A$=CHR$(13) THEN 70
 40 X=X+1 : IF X=11 THE X=X-1 : GOTO 70
 50 F$=F$+A$ : PRINT A$;
 60 GOTO 20
 70 IF X=0 THEN END
 75 PRINT ; PRINT "WELCHE FKT.-TASTE (1-8) ";
 80 POKE 198,0 :WAIT 198,1 : GET A$
 90 IF A$<"1" OR A$>"8" THEN 70
 95 PRINT A$
 100 A=VAL(A$)-1
 110 POKE 49152+A*11,X+1
 120 FOR I=1 TO X
 130 A$ = MID$(F$,I,1)
 140 IF A$=CHR$(95) THEN A$=CHR$(13)
 150 POKE 49152+A*11+I, ASC(A$)
 160 NEXTI
```

LOCAZIONE 657

Selezione commutazione (SHIFT)+(C=)

Valore normale: 0

Contenuto:

Normalmente possiamo commutare con (SHIFT)+(C=) fra modo minuscolo e maiuscolo.
Questa commutazione puo' essere evitata con PRINT CHR$(8) e ripristinata con PRINT CHR$(9).
Entrambi i comandi CHR$ inseriscono un semplice BIT nel registro in esame.
Con POKE 657,125 potete bloccare la commutazione, con POKE 657,0 inserirla.

Riferimento:

La diretta commutazione attraverso la scrittura della locazione 53272 (vedete poi) o attraverso la stampa del CHR$(14) del testo e CHR$(142) per il modo grafica non puo' venire evitata.

LOCAZIONE 678

Flag per PAL/NTSC

Valore normale: 1 (in Europa)

Contenuto:

Poiche' il C-64 e' in funzione sia in America che in Europa con una diversa rete di frequenza (confronta anche 54272/54273) anche i sistemi video (Pal in Europa/NTSC in America) lavorano diversamente: ci sono due diverse versioni del C-64. La versione Europea ha una frequenza al quarzo di 7.734472 Mhz.
La frequenza della versione americana e' 14.31818 Mhz
Poiche' da questa frequenza dipendono anche altri fattori (es. tonalita'), puo' essere utile esaminare con che tipo di macchina lavoriamo:

Circa questo problema ci sono queste informazioni sul Flag:

1 Versione Europea
0 Versione Americana

LOCAZIONI 704-766

inutilizzate

Valore Normale: inutilizzate

Contenuto:

Queste locazioni di memoria in pagina zero sono libere.
Qui potete deporre ad esempio i dati relativi ad uno sprite (leggere chiarimenti al tema 'Sprites')

LOCAZIONI 768-769

Puntatori alla routine di stampa messaggi d' errore.

Valore Normale:

PEEK (768) = 139 e PEEK(769) = 227; posizione puntata 58251 ($E38B).

Contenuto:

In questa locazione e' deposto l'indirizzo iniziale della routine dell'interprete autorizzata alla stampa dei messaggi di errore.
La posizione e' specificata nella sequenza : HighByte (parte alta) seguito dal lowByte (parte bassa).
La routine dei messaggi di errore inizia pertanto all' indirizzo decimale 58251 ($E38B).
Talvolta puo' essere giustamente utile disattivare la routine in questione ; possiamo agire a tale scopo scrivendo:

POKE768,185

Se in seguito si desidera riabilitare il normale funzionamento di stampa degli errori, ovviamente digiteremo:

POKE768,139.

Questa tecnica funziona solo all'interno di un programma.

Riferimento:

Fino a quando il messaggio di errore viene inibito, non potete attendervi nessun SYNTAX ERROR da programmi errati che pertanto prima dovranno essere preventivamente collaudati nel funzionamento globale.

Esempio:

Il seguente programma esamina, se un particolare apparecchi o periferico e' inserito.

```
10 GN=(numero periferica)
20 OPEN 127,GN
30 POKE 768,185
40 PRINT#127 : CLOSE 127
50 POKE 768,139
60 IF ST=-128 THEN PRINT 'L'APPARECCHIO NON E' PRONTO'
```

LOCAZIONI 774/775

Puntatori alla routine di LIST.

Valore normale:

PEEK(774) = 26
PEEK(775) = 167
posizione specificata 42778 ($A71A)

Contenuto:

Questo indicatore contiene come HighByte e Low Byte le locazioni d'inizio della LIST-Routine dell'interprete BASIC. Si offrono dunque valide possibilita' di azione : ad esempio possiamo proteggere un testo BASIC dal LIST con semplici mezzi.
Se entrambi i Byte ad esempio sono puntano ad un'altra routine, verra' eseguita questa ultima, in luogo della vera e propria List-Routine.
Un'altra possibilita' consiste nella modifica dei puntatori 774 e 775 per farli puntare ad una nuova routine che formatti il listato come piu' ci aggrada.

Ecco alcune proposte per la 'alterazione' dei vettori 774 e 775:

PEEK(774)	PEEK(775)	LOCAZIONE	HEX	CONT.
68	166	42564	$A644	NEW
26	167	42778	$A71A	LIS
7	168	43015	$FCE2	SYNTAX ERROR
226	252	64738	$FCE2	Sistem Reset

LOCAZIONI 785-786

USR-locazione

Valore Normale: deve esser e stabilito da chi ne fauso.

Contenuto:

In questi vettori e' memorizzata la locazione d'inizio di un programma in linguaggio macchina che desideriamo mandare in esecuzione ; possiamo agire a tale scopo utilizzando il comando BASIC USR (X).
Con USR(X) abbiamo la possibilita' di trasmettere un parametro al programma in linguaggio macchina, il cui valore poi puo' essere utilizzato.

Riferimento:

Questo riferimento e' dedicato solo ai programmatori in linguaggio macchina e dunque non principianti.
Il programma in linguaggio macchina puo' leggere il valore passatogli in VIRGOLA MOBILE da un accumulatore (1) le cui locazioni di memoria risiedono dalla cella 97 fino alla 102 ($61-$66).
Se la URS-routine deve fornire al programma BASIC (come se fosse una funzione matematica) un rusultato,essa dovra' scriverlo nuovamente in FLOATING POINT (virgola mobile) nello stesso accumulatore 1.

LOCAZIONI 788-789

Vettori per la IRQ-routine.

Valore Normale:

PEEK(788) = 34
PEEK(789) = 234
posizione indicata ($EA31).

Contenuto:

Questi vettori puntano alla routine che il Computer esegue ogni sessantesimo di secondo (IRQ).
Potremo alterate i valori da essi contenuti facendoli puntare ad una nuova routine da noi scritta (solo in linguaggio macchina).
In questo ambito si aprono vaste possibilita': la routine che il computer esegue ogni sessantesimo di secondo puo' girare in parallelo al normale funzionamento di un programma BASIC.
Per esempio possiamo inserire nelle linee dell' IRQ una procedura che, ad intervalli periodici, cambi i colori dello schermo, durante l'esecuzione di un banale programma BASIC.

Riferimento:

Prima di alterare un vettore d'interruzione, e' opportuno digitare:

POKE56334,0

Poiche' il sistema ha bisogno esattamente di un sessantesimo di secondo, per eseguire, le due POKE.
Frattanto avrebbe potuto gia' aver luogo una interruzione, e il computer sarebbe andato in TILT.

Poiche' ad ogni IRQ si provvede anche alla visualizzazione dei caratteri presenti sullo schermo, inibendo gli IRQ per alterarne i vettori (poke..) avremo come effetto immediato l'interruzione della visualizzazione.
Come ultima operazione non dimenticatevi di scrivere:

```
POKE 56334,1
```

per ripristinare il normale funzionamento del sistema (e schermo).

Esempi:

E' importante notare che, ad esempio, agendo opportunamente sui vettori d'interruzione potete mettere 'fuori combattimento' il tasto RUN/STOP.
Si arriva a questo semplicemente con POKE 788,52 all'inizio del programma BASIC.
Attenzione:
questo comando non disattiva tuttavia l'effeto della sequenza RUN/STOP e RESTORE.
Inoltre i vettori dopo ogni <RUN/STOP><RESTORE> verranno deposti di nuovo al loro valore normale.
Se volete disinnescare l'azione di <RUN/STOP><RESTORE>, andate alla locazione 792/793.
Col POKE 788,49 riattiverete nuovamente il tasto RUN/STOP.
Questo piccolo programma in linguaggio macchina modifica circa ogni 4 secondi il colore del quadro.

```
10 FOR X=8192 TO 8212 : READ A:S = S+A
20 POKE X,A : NEXT X
30 DATA 166,162,224,0,224,128,240,3,76,49,
   234,174,32,208,202,142,32,208,76,8,32

40 IF S<> 2620 THEN PRINT 'ERRORE NEI DATA':STOP
50 POKE56334,0:POKE788,0:POKE789,32:POKE56334,1
```

LOCAZIONI 792-793

Vettore per la NMI-Routine

Valore Normale:

```
PEEK(792) = 71
PEEK(793) = 254
```

Contenuto:

Questo vettore punta alla Routine che il sistema richiama alla pressione del tasto <RESTORE>.
Alterando opportunamente tali vettori possiamo ad esempio inibire l' effetto del consueto <RUN/STOP><RESTORE>.

Esempio:

Ecco alcuni possibili valori.

PEEK(792)	PEEK(793)	LOCAZIONE	HEX.	CONT
7	168	43015	$A807	SINTAX ERROR
226	252	64738	$FCE2	RESET
71	254	65095	$FE47	NMI
102	254	65126	$FE66	RUN
193	254	65217	$FEC1	RUN/STOP

LOCAZIONI 828-1019

Buffer del registratore.

Valore Normale= -

Contenuto

Se avete lavorato col Datasette, i dati, che devono essere letti o scritti, vengono memorizzati (a blocchi) in questo ambito, per un particolare tipo di operazione del sistema atto a velocizzare il piu' possibile lo scambio di informazioni attraverso supporto magnetico a nastro.

Se voi non possedete nessun Datasette o non ne fate uso, potete servirvi liberamente di questa zona di memoria per inserirvi programmi in linguaggio macchina o dati per la definizine degli Sprites (leggere informazioni piu' precise alla voce 'SPRITES').

LOCAZIONI 1024-2023

Mappa RAM del video.

Valore Normale: -

Contenuto:

In questa posizione e' collocata normalmente la RAM del video, ovvero l'ambito di memoria in cui il Computer depone i numeri e caratteri che l'utente VEDE sullo schermo televisivo.
La mappa video puo' essere spostata come gia' menzionato alla locazione 648.

Ulteriori informazioni si troveranno in riferimento al registro 53272.
Potete scrivere sulla mappa video, oltre attraverso le consuete PRINT, anche mediante l'istruzione POKE seguita dal codice corrispondente al carattere da visualizzare.

Riferimento:

Dopo aver 'POKATO' un carattere nella mappa video, per poterlo vedere effettivamente occorre agire anche nella mappa colori : Il relativo ambito di memoria si estende da 55296 fino a 56295.
Diventa ancora piu' facile se, stabilito il colore, lo scriviamo nella locazione 646 e digitiamo CHR$(147): l'intera RAM colori viene riempita automaticamente col colore selezionato.

Gli ultimi modelli del Commodore 64 fanno automaticamente un'operazione del tipo di quella descritta.
Per visualizzare un carattere con una POKE nella mappa video, non sara' dunque necessario, per poterlo osservare, doverne mutare il colore.

Esempio:

Questo piccolo programma riempie il video con cuoricini bianchi.

```
10 POKE 646,1 : PRINT CHR$(147) : POKE 53281,3
20 FOR X=1024 TO 2023: POKE X,83 : NEXT
```

LOCAZIONI 2040-2047

Puntatori definizione SPRITES

Valore Normale: -

Contenuto:

In questi 8 BYTEs giacciono i puntatori degli 8 sprites, che indicano a partire da quale locazione comincia la relativa griglia di 64 bytes per la definizione delle figure stesse.
Il valore da assegnare allo sprite numero x puo' essere cosi' attribuito:

```
POKE (2040+numero sprite),locazione inizio definizione/64
```

Nel primo byte c'e' l'indicatore per lo sprite #0, poi seguono i numeri dall 1 fino al 7.
Ulteriori informazioni le troverete nella spiegazione dedicata al VIC.

Riferimento:

Se spostiamo la mappa della RAM video, allora conseguentemente ed automaticamente verra' alterata anche la zona di memoria contenete i puntatori per le definizioni degli sprites.
Se ad esempio il video e' collocato a 2048, i puntatori delle definizioni verranno letti non piu' da 2040 a 2047, ma da 3064 in poi.
Generalmente vale percio':

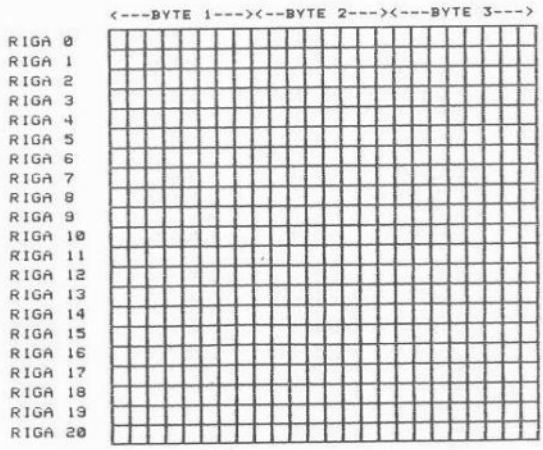
Indicatore definizione sprite #0 = RAM video + 1016

VIC

CHIP DI INTERFACCIA VIDEO

VIC 6566 - IL CHIP VIDEO.

Con il Commodore 64 sussiste la possibilita' di illustrare, muovere simboli e figure indipendentemente da quanto presente sul video.
Il segreto si chiama 'SPRITE'.
Uno sprite e' un blocco di 24 per 21 punti che l'utente puo' accendere o spegnere a proprio piacere.
Uno sprite puo' accettare 1 dei 16 possibili colori, oppure ben 3 contemporaneamente a patto che il numero di punti si riduca a soli 12*21.
Uno sprite puo' essere spostato attraverso l'intero video senza che si alteri nulla eventualmente presente sullo schermo: ne' prima ne' dopo la raffigurazione.
Percio' gli sprites sono compatibili con tutti i possibili sfondi.
Gli sprites possono essere ingranditi sia orrizzontalmente che verticalmente.
Se si 'scontrano' uno sprite e lo sfondo (es: caratteri) potete rivelare il fatto in un registro di collisione e il programma puo' comportarsi di conseguenza, ad esempio lasciando esplodere i veicoli spaziali che si sono urtati o, al campione che sta inciampando contro la barriera, ed assegnare una penalita'.
Il Commodore puo' visualizzare nello stesso istante 8 SPRITES.
Gli sprites sono numerati da #0 fino a #7.
Poiche' uno Sprites consiste di 24*21 punti, gli occorrono ben 63 bytes nei quali memorizzare tramite bits la relativa immagine.
Il calcolo dei valori avviene come segue:
Ciascuno dei 63 Byte viene posto uno vicino all'altro in triplice fila; ogni bits corrisponde ad un punto sul video; abbiamo 21 file di tre bytes per un totale di 63 celle.
Poiche' un Computer puo' calcolate meglio con 64 , viene praticamente aggiunto un byte morto.
Otteniamo dunque il seguente quadro:



All'interno di questo campione potete accendere o meno qualsiasi punto.
I bytes vengono poi calcolati ed inseriti mediante POKE in determinati ambiti della memoria.
Dunque uno sprite abbisogna di un grosso blocco di 64 Byte di RAM: come vedrete il COMMODORE 64 non da' problemi in merito.

Il VIC, ovvero CHIP di INTERFACCIA VIDEO, intende la memoria come ripartita in blocchi da 64 bytes.
Il BLOCCO 0 va dalla locazione 0 sino alla 63, il BLOCCO 1 va dalla 64 fino alla 127, il blocco 2 va dalla 128 fino alla 191.....
Ciascun blocco specifica la figura, l'immagine dello sprite al quale e' assegnato.
I puntatori della definizione degli sprites, sono allocati immediatamente dopo la RAM-video, a partire dalla locazione 2040.
Vediamo quali ambiti di memoria sono a disposizione:

blocco	indirizzi	funzione
11	704-766	
13	832-894	buffer cassetta
14	896-958	''
15	960-1022	''
32	2048-2110	programma BASIC
33	2112-2174	''
63	4032-4094	''
128	8192-8254	''
129	8256-8318	''
255	16320-16382	''

Possiamo dunque inserire i dati nel buffer della cassetta, purche' nessuna operazione su nastro abbia luogo. (vedere 828-1029)
l'ambito di memoria che parte da 2048 e' solitamente occupato da programma BASIC: dovremo dunque spostarlo in alto, tramite le locazioni 43 e 44 (anche 49/50).

RIFERIMENTO:

L'ambito di memoria tra 4096 e 8192, non puo' essere utilizzato per immagazzinare i dati degli sprites, poiche' il VIC, in tale zona non riesce a leggere.
Per poter visualizzare uno sprite sullo schermo, occorre innanzitutto accenderlo, agendo sul registro 53269: ogni bit corrisponde ad un preciso sprite.
Inoltre per ogni sprite deve essere data una precisa coordinata x, ed y.
Esse vanno inserite a coppie a partire dalla locazione 53248.
Occorre ancora precisare per ogni sprite, da vedere, la relativa priorita' con lo sfondo.

GRAFICA HIRES

HIRES in inglese e' una forma contratta di HIGHT RESOLUTION.
E' un particolare tipo visualizzativo in cui possiamo

accendere o spegnere tutti i punti ,ad uno ad uno, dello schermo.
Potremo ottenere ad esempio un tipo di immagine simile a quella riproducenti foto dei giornali.
Il COMMMODORE 64 dispone di una grafica a 320 per 200 punti: ben 64000 puntini in totale.
Per poter visualizzare la grafica in hi-res, il computer abbisogna di ben 8000 bytes.
Questa memoria e' definita come 'BIT-MAP'.
Essa funziona sul principio tipico degli sprites.
Il BIT-MAP e' costituito da 200 rige di 40 bytes ciascuna.
Ogni riga corrisponde ad ogni linea del video.
Questo significa:
il primo byte del BIT-MAP non e' altro che i primi otto puntini sulla sinistra, in alto al teleschermo.
Il secondo byte e' posto esattamente sotto.
Si prosegue cosi' per i primi 8 bytes.
Il discorso riprende dal nono, che e' visualizzato 8 punti piu' a destra del primo, ma sulla stessa linea.

Lo schema spiega questo concetto:

```
<-------------------- 40 BYTES ------------------->
BYTE 0     BYTE 8      ......  BYTE 304   BYTE 312
BYTE 1     BYTE 9      ......  BYTE 304   BYTE 313
BYTE 2     BYTE 10     ......  BYTE 306   BYTE 314
BYTE 3     BYTE 11     ......  BYTE 307   BYTE 315
BYTE 4     BYTE 12     ......  BYTE 308   BYTE 316
BYTE 5     BYTE 13     ......  BYTE 309   BYTE 317
BYTE 6     BYTE 14     ......  BYTE 310   BYTE 318
BYTE 7     BYTE 15     ......  BYTE 311   BYTE 319

...        ...         ......  ...        ...
...        ...         ......  ...        ...
...        ...         ......  ...        ...

BYTE 7680  BYTE 7688   ......  BYTE 7984  BYTE 7992
BYTE 7681  BYTE 7689   ......  BYTE 7985  BYTE 7993
BYTE 7682  BYTE 7690   ......  BYTE 7986  BYTE 7994
BYTE 7683  BYTE 7691   ......  BYTE 7987  BYTE 7995
BYTE 7684  BYTE 7692   ......  BYTE 7988  BYTE 7996
BYTE 7685  BYTE 7693   ......  BYTE 7989  BYTE 7997
BYTE 7686  BYTE 7694   ......  BYTE 7990  BYTE 7998
BYTE 7687  BYTE 7695   ......  BYTE 7991  BYTE 7999
```

Per il BIT-MAP necessitate come gia' accennato, di ben 8000 bytes.
Servono anche 1000 bytes che decidono il colore da dare ai puntini.
La mappa colore non e' tuttavia la solita', da 55296 in poi, bensi', ora e' il video (normalmente da 1024) che funge da mappa-colore.

Per la determinazione dei colori vale la seguente osservazione:
Per ogni blocco da 8 per 8 bits, del BIT-MAP, e' associato un colore nella RAM video.
I bits maggiori, stabiliscono il colore delle righe (e dei punti), i bit minori decidono il colore dello sfondo.
Se entriamo in multicolore, (vedere loc. 53270) avremo una situazione del tipo:
due puntini adiacenti del BIT-MAP vengono visualizzati come uno solo.
Questo punto e', in orizzontale, di dimensioni doppie che in hi-res.
Percio' la risoluzione orizzontale e' la meta': ma la combinazione dei due bits decide pero' uno dei 4 colori possibili.

Possibili combinazioni dei bits:

00 colore da 53281

01 colore dai bits 4-7 del video

10 colore dai bits 0-3 del video

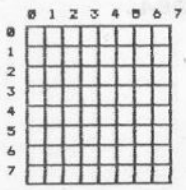
11 bit 0-3 dalla mappa colore 55296

In questo caso entrano in gioco anche i bits 0-3 della mappa colore, cosa in contrasto con il modo HI-RES.

Ci si puo' chiedere la migliore posizione in cui allocare il BIT-MAP: ma normalmente non c'e' molta scelta.
Poiche' il VIC puo' vedere solo blocchi da 16K-BYTES, ci sono 2 possibili posizioni di inizio del BIT-MAP : 0 e 8192.
Il primo caso e' da scartare poiche' e' la zona tipica della pagina zero.
Resta dunque il secondo ambito.
Come agire per la selezione del collocamento del BIT-MAP, lo leggerete alla locazione 53272.
Voi potete anche selezionare quale blocco da 16 K dovra' contenere le informazioni necessarie al VIC.

SET DI CARATTERI

L'immagine dei caratteri che il VIC illustra sullo schermo e' letta in una precisa memoria ROM.
Ogni carattere necessita di 8 file di 8 bits ciascuna.
Poiche' il set del COMMODORE dispone di ben 256 caratteri differenti , la mappa caratteri dovra' essere lunga 256*8 bytes.
Avremo dunque una ROM lunga 4 K-bytes.
Agendo opportunamente sul VIC possiamo spostare il set i caratteri da 4096 a 8192, nella RAM.
Potremo dunque creare dei caratteri personali.
Questo e' il motivo per cui in questo ambito di memoria non possono visualizzarsi ne' gli sprites ne' il BIT-MAP.
Un carattere consiste in un pacchetto di 8 per 8 puntini.



Come spostare il set di caratteri lo vedrete alla locazione 53272.

Se voi volete creare un nuovo set di caratteri e' piu' semplice trasferire con un ciclo quelli della ROM, per poi andarli a modificare.
Il programma che realizza tale trasferimento e' listato alla locazione 53272.
Anche i caratteri possono essere visualizzati in modo multicolore.
Questo modo viene come di consueto inserito agendo sulla locazione 53270.
I due punti vengono nuovamente raggruppati in una combinazione che ne decide il colore.
La regola con cui vengono assegnati i colori ad una particolare combinazione di bits e':

00 da 53280

01 dalla RAM-video , bits 4-7

10 dalla RAM-video , bits 0-3

11 bits 0-3 della mappa colori da 55296

LOCAZIONE 53248

Coordinata x dello sprite #0.

Valore Normale:

0 se lo sprite e' spento.

Contenuto:

Qui viene memorizzata la coordinata x dello sprite numero 0. Quando il valore in tale registro e' minore di 24, allora lo sprite 0 e' sicuramente al di fuori del quadro visibile.

riferimenti:

Il valore massimo scrivibile in tale registro e' 255: come fare se la coordinata x e' maggiore di tale valore, lo trovate alla locazione 53264.

Compito dei bits: -

Esempio:

Poiche' uno sprite per essere visto deve essere prima acceso, trovate gli esempi relativi alla locazione 53264.

LOCAZIONE 53249

Coordinata y dello sprite #0.

Valore Normale:

0 se lo sprite e' spento.

Contenuto:

Qui viene inserita l'ordinata relativa allo sprite numero #0
Se il valore in questo registro e' minore di 30 oppure maggiore di 250, allora il nostro sprite non sara' visualizzabile totalmente.

Compito dei bits:-

Esempio:

Vedere la locazione 53269.

LOCAZIONI 53250-53263

Coordinate x,y degli sprites #1-#7.

Valore Normale:

0 se sono spenti.

Contenuto:

Le coordinate degli sprites da 1 a 7 sono memorizzate nel seguente ordine:

LOCAZIONE	FUNZIONE
53250	x sprite #1
53251	y sprite #1
53252	x sprite #2
53253	y sprite #2
53254	x sprite #3
53255	y sprite #3
53256	x sprite #4
53257	y sprite #4
53258	x sprite #5
53259	y sprite #5
53260	x sprite #6
53261	y sprite #6
53262	x sprite #7
53263	y sprite #7

LOCAZIONE 53264

Bit maggiore ascisse sprites.

Valore Normale: -

Contenuto:

Senza usare tale registro, le ascisse degli sprites potevano andare esclusivamente da 0 a 255, coprendo 2/3 della larghezza del video.
Per poter percorrere la rimanente zona, (da 255 sino a 320), occorre agire su tale registro.

Compito dei bits:

0 coordinata x sprite #0

1 coordinata x sprite #1

2 coordinata x sprite #2

3 coordinata x sprite #3

4 coordinata x sprite #4

5 coordinata x sprite #5

6 coordinata x sprite #6

7 coordinata x sprite #7

Spiegazione:

Se un bit e' acceso, alla ascissa del relativo sprite e' sommato il valore 255.
Se la x dello sprite e' 1, accendendo il corrispondente bits in tale registro, essa saltera' al valore 256.
E' possibile effettuare tale commutazione solo per gli sprites voluti.
Se le coordinate debbono essere calcolate normalmente, allora occorrera' disinserire il bit.

Esempio:

Questo programma muove uno sprite in senso orizzontale per l'intero video.

Memorizzate questo listato su nastro o dischetto : in seguito ne avrete bisogno.

```
10 FORX=832TO894:READW:S=S+W:POKEX,W:NEXTX
20 IFS<>4176THENPRINT'ERRORE IN DATA':STOP
30 POKE2040,13:POKE53269,1:POKE53264,0
40 POKE53249,100
50 FORX=0TO368
60 W=X:IFX>255THENW=X-256:POKE53264,1
70 POKE53248,W
80 NEXTX
90 DATA 0,36,0,0,102,0,0,102,0,0,102,0,
   0,102,0,0,126,0,0,219,0,1,255,128
100 DATA 1,231,128,0,195,0,0,60,0,0,126,0,
    0,219,0,0,219,0,0,219,0,1,219,128
110 DATA 1,126,128,1,126,128,7,66,224,15
    195,240,0,0,0
```

LOCAZIONE 53265

Registro di controllo modo grafico.

Valore Normale: 27

Contenuto:

In questo registro sono raggruppate molte funzioni.

Ad esempio e' possibile selezionare tramite tale registro il modo visualizzativo ad alta risoluzione.
Agendo su tale locazione potrete anche spegnere completamente il video: tale metodo rende piu' veloce l'esecuzione di un programma Basic.
Un'altra funzione e' la selezione delle righe visualizzabili. Voi potete decidere se disporre di 24 oppure 25 righe testo. Poiche' alcuni computer dipongono di 24 righe, questo artificio consente una maggiore trasportabilita' del programma.
Inoltre si puo' realizzare lo SCROLL FINE (SMOOTH SCROLLING): spostare il contenuto dello schermo verso l'alto o basso, molto linearmente.
Per questo occorrera' disporre il video a 24 righe,la 25-esima servira' per far entrare le nuove informazioni sul video.
Questo artificio e' piu' volte utile durante la programmazione di giochi con effetti di movimento verso l'alto o verso il basso.

Compito dei bits:

0-2 scroll fine (0-7), std=3

3 25 righe (1) / 24 righe (0)

4 video acceso (1)/ spento (0)

5 alta risoluzione (1)/ testo (0)

6 sfondo colori est.(1)/ norm.(0)

7 vedere loc.53266.

Spiegazione:

Il bit 0 fino al secondo immagazzinano la posizione di scroll del video.
Essa varia tra un minimo di 0 ed un massimo di 7.
Il valore standard si trova nel mezzo: 3.
I valori maggiori di 3 fanno muovere lo schermo verso l'alto, e viceversa.
Per modificare questi tre bits, utilizzate la seguente formula:

```
POKE 53265,(PEEK(53265)AND248)OR( x )
```

Il bit #3 decide se le righe del video sono 24 oppure 25.
Se esso e' acceso allora il VIC vede 25 righe, in caso contrario ne mostra solo 24.
Il bit #4 decide se il video e' acceso oppure spento: se esso e' a 0 allora il testo sullo schermo scompare, e viceversa.
Poiche' ora il VIC non impiega tempo per la visualizzazione delle immagini, il computer lavora piu' velocemente, anche durante le operazioni su dischetto o cassetta.
Ecco spiegato perche' il DATASSETTE spegne il video ad ogni operazione.

Riferimento:

Questo risparmio di tempo puo' essere molto importante se volete collegare al computer delle periferiche come stampante, drive,....
Il bit #5 inserisce il modo HI-RESOLUTION.
Ne saprete di piu' leggendo a proposito della grafica in alta risoluzione del VIC.
La posizione del BITMAP e della memoria colori e' stabilita dalla locazione 53272.
Il bit #6 attiva lo sfondo a piu' colori.
In questo modo il testo e' visualizzabile con 4 differenti colori di sfondo.
Ogni blocco di 64 caratteri ha un preciso colore selezionabile dall'utente.

codice	tastiera	colore
0-63	normale	da 53281
64-127	SHIFT	da 53282
128-191	RVS	da 53283
192-255	SHIFT+RVS	da 53284

Il bit #7 e' commentato alla locazione 53266.

Riferimento:

Se provate ad inserire il modo HIRES contemporaneamente a quello a sfondo esteso, il VIC va in panne, sino a che non optate per uno solo dei due casi.
Motivo: inserire contemporaneamente tutti e due i metodi non ha molto senso; o lavorate in alta risoluzione oppure con i caratteri.
Per attivare nuovamente il VIC:

```
POKE 53265,27
```

Esempio:

Il seguente breve programma utilizza lo sfondo a piu' colori.

```
10 POKE53281,0:POKE53282,2:POKE53283,5:POKE53284,14
20 POKE53265,91:PRINTCHR$(147)
30 FORX=0TO255:POKE1024+X,X:NEXT
```

Il seguente programma usa invece l'alta risoluzione:

```
10 POKE53265,59:POKE53272,24
20 FORK=8192TO16191:POKEX,0:NEXT
```

Leggere anche la locazione 53272.
L'esecuzione di questo programma, a causa del ciclo FOR..NEXT, e' particolarmente lunga.
Eccovi la versione in linguaggio macchina:

```
10 A=828
20 FORX=ATOA+34
30 READW:S=S+W:POKEX,W
40 NEXTW
50 IDS<>4607THENPRINT'ERRORE IN DATA':STOP
60 DATA 162,32
70 DATA 160,0,134,100,132,99,169,0,170,145,99
80 DATA 200,192,0,208,249,232,224,32,240,11,164,100,200
90 DATA 132,100,160,0,192,0,240,233,96
```

A questo punto due osservazioni:
per primo notiamo che il programma e' fatto in modo da poter essere messo ovunque perche' molto corto; poi occorre ricordare che esso non puo' essere allocato da 8192 fino a 16383.
Se desiderate spostare il BITMAP, agite mutando la cifra 32 nella riga 60.

Il valore e' calcolabile con la seguente formula:

VALORE=(INIZIO)/256

Ricordate che pero' il BITMAP puo' essere allocato in non tutte le zone di memoria (leggere ulteriori informazioni sul VIC).
Confrontate i due programmi sotto , essi hanno un tempo di esecuzione diverso a causa delle differenti condizioni del video con cui lavorano:

```
10 TI$='000000':FORX=1TO1000:NEXT:PRINTTI$
10
POKE53265,11:TI$='000000':FORX=0TO1000:NEXT:POKE53265,27:PRIN
TTI$
```

L'ultimo esempio mostra l'utilizzo dello SCROLL FINE, e lavora con il metodo delle 24 righe testo.

```
10 FORX=23TO16STEP-1
20 POKE53265,X
30 FORY=1TO100:NEXTY
40 NEXTX
```

LOCAZIONE 53266

Registro RASTER.

Valore Normale: -

Contenuto:

In questo registro trovate il numero della riga dello schermo che il pennello elettronico del vostro televisore o monitor, sta percorrendo.
Poiche' questa operazione e' estremamente veloce, da BASIC non ha molto senso interrogare tale registro.
Da linguaggio macchina invece possiamo testare a quale riga dello schermo televisivo e' pervenuta la visualizzazione.
Tramite questo test possiamo fare in modo, ad esempio, di avere sul video piu' modi grafici contemporaneamente (testo ed alta risoluzione ...).
Il valore letto in questo registro varia da 0 a 256, ma le righe del video sono di piu': il bit #7 della locazione 53265 e' la parte piu' significativa dell'indirizzo.

Voi potete anche scrivere in questo registro: se scrivete un valore, la prima volta che il RASTER arrivera' a scandire tale numero di riga, verra' generata una interruzione, se il bit nella locazione #0 e' spento, e ,se il bit #0 di 53274 e' acceso.
(vedere locazioni 788 e 789).
Attraverso una corretta programmazione di queste interruzioni si arriva ai simultanei modi grafici.
Ulteriori note le troverete alla locazione 53274 e 53273.

Esempio:

Questo programma in linguaggio macchina divide il video in due zone: la parte alta in hires, quella bassa in testo.

Noi abbiamo allocato il programma da 679: l'utente potra' modificare a piacere l'indirizzo.

```
10 A=679:FORX=ATOA+49:READW:POKEX,W:S=S+W:NEXT
20 IFS<>5570THENPRINT'ERRORE NEI DATA':STOP
30 POKE56334,0:POKE788,A-INT(A-256)*256):POKE789,INT(A/256)
40 POKE53274,129:POKE56334,1
50 DATA 173,25,208,41,1,240,40,141,25,208,169,
   59,72,169,24,72,162,208
60 DATA 173,18,208,16,10,104,104,169,27,72,169,
   21,72,162,16,142,18,208
70 DATA 104,141,24,208,104,141,17,208,76,188,
   254,76,49,234
```

Riferimento:

Se premete un tasto noterete un leggero tremolio del video.
Questo accade perche' il VIC non lavora in modo ben sincronizzato col processore, che puo' essere gia' impegnato in altre interruzioni.
Infatti quando l'interruzione generata dal VIC comunica il segnale, il processore finisce prima cio' che stava eseguendo.
Il tempo di attesa, genera tali irregolarita' di visualizzazione.

LOCAZIONE 53267

Posizione x della penna ottica.

Valore Normale: -

Contenuto:

In questo registro e' collocato il valore dell'attuale ascissa della penna luminosa.
Quando la fotocellula della penna coincide con il passaggio del fascio elettronico, allora il VIC memorizza in 53267 l'attuale coordinata.

Riferimento:

Poiche' la penna luminosa puo' essere in una ascissa maggiore di 255, occorre un ulteriore bit di controllo.
E' per questo che la penna va usata con il metodo MULTICOLOR.

Riferimento:

Esattamente accade che anche la penna-luce genera periodicamente una interruzione (vedere 53266).
Discuteremo di questo tema alla locazione 53273 e 53274.

LOCAZIONE 53268

Posizione y della penna ottica.

Valore Normale: -

Contenuto:

Qui trovate il valore dell' ordinata riferita all'attuale posizione della penna-luce.
A tale scopo bastano evidentemente 8 bits.
leggere ulteriori informazioni alla locazione 53268.

LOCAZIONE 53269

Flag acceso/spento per sprites.

Valore Normale:

Se nesuno sprite e' acceso.

Contenuto:

Questa posizione di memoria determina se uno sprite e' acceso oppure no.
Ogni bits del byte in questione e' collegato ad uno sprite.
Possiamo dunque accendere ma anche spegnere uno sprite.

Compito dei bits:

0 sprite #0 acceso (1)/ spento (0)

1 sprite #1 acceso (1)/ spento (0)

2 sprite #2 acceso (1)/ spento (0)

3 sprite #3 acceso (1)/ spento (0)

4 sprite #4 acceso (1)/ spento (0)

5 sprite #5 acceso (1)/ spento (0)

6 sprite #6 acceso (1)/ spento (0)

7 sprite #7 acceso (1)/ spento (0)

Spiegazione:

Per accendere un bit, utilizzare la seguente formula:

```
POKE53269,PEEK(53269)OR(2↑NUM.SPRITE)
```

Per spegnere un bit, utilizzare invece:

```
POKE53269,PEEK(53269)AND(255-2↑NUM.SPRITE)
```

Riferimento:

Uno sprite e' dunque visibile se entrambe le coordinate lo centrano sul video, se il puntatore della definizione e' disposto, e se e' acceso il relativo bit in 53269.

Esempio:

Questo programma mostra sul video una intera famiglia di lepri.
Tutti gli otto sprites presenti sullo schermo, hanno la medesima definizione.
Il colore di tutte le immagini e' grigio chiaro.

```
10 FORX=832TO894:READW:S=S+W:POKEX,W:NEXTX
20 IFS<>4176THENPRINT'ERRORE IN DATA':STOP
30 FOR X=0 TO 7 : POKE 2040+X,13 : POKE
   53287+X,15 : POKE 53248+2*X,30*(X+1)
40 POKE 53249+2*X,160 : NEXT : POKE 53269,255
60 DATA 0,36,0,0,102,0,0,102,0,0,102,0,
   0,102,0,0,126,0,0,219,0,1,255,128
70 DATA 1,231,128,0,195,0,0,60,0,0,126,0,
   0,219,0,0,219,0,0,219,0,1,219,128
80 DATA 1,126,128,1,126,128,7,66,224,
   15,195,240,0,0,0
```

Se non l'avete ancora fatto registrate su disco o nastro questo programma, ne avremo ancora bisogno in seguito.

LOCAZIONE 53270

Registro di controllo x.

Valore Normale: 200

Contenuto:

In questo registro si trovano molte funzioni importanti :
lo scroll fine orizzontale, che in linea di massima funziona come quello verticale, la selezione del MULTICOLOR, in cui ben 4 colori possono essere usati per ogni punto dello schermo.
Mediante lo scroll fine possiamo far passare lentamente sullo schermo figure ed immagini disegnate a caratteri, da destra verso sinistra e viceversa.
Assieme al video si muove anche la mappa colore.
Quando si usa lo scroll fine, si lavora in luogo di 40 colonne, con 38.
Il metodo multicolore e' valido sia per l'HIRES che per il normale testo.

Compito dei bits:

0-2	scroll fine (0-7), std=0
3	40 colonne (1)/ 38 colonne (0)
4	multicolore (1)/ normale (0)
5	sempre 0
6-7	sempre 1

Spiegazione:

I bits da 0 a 2 definiscono il movimento orizzontale del video.

Il valore che tali 3 bits possono codificare va dal minimo 0 al massimo 7.
Il valore letto normalmente e' 0.
Per modificare tali bits potete scrivere:

```
POKE 53270,(PEEK(53270)AND248)OR(num.)
```

Il bit #3 stabilisce se il testo e' fornito di 38 oppure 40 colonne.
Il bit #4 attiva il modo multicolore, che restera' sempre attivo sino a quando non lo azzereremo.
Ricordate che con tale sistema visualizzativo ogni combinazione di 2 bits definisce un colore: 00,01,10, e 11 corrispondono ciascuno ad un colore definibile.

METODO MULTICOLORE PER IL TESTO.

Se il colore di un carattere acceso sul video e' minore di 8, sebbene esso sia in multicolore, viene visualizzato normalmente.
I colori da 8 in poi attivano il multicolore.
Il bit #3 della mappa colore, decide dunque se un segno e' illustrabile o meno in modo MULTICOLOR.
Le possibili combinazioni bits-colori sono:

punti	colore
00	da 53281
01	da 53282
10	da 53283
11	da ram 55296 se > 8

METODO MULTICOLORE PER GRAFICA HI-RES.

Anche in alta risoluzione possiamo richiamare il modo multicolore: in ogni gruppo da 8 per 8 bits, possono risiedere contemporaneamente sino a 4 differenti colori.
Essi possono essere scelti tra un totale di ben 16.
Le combinazioni di bits e le relative fonti di colore sono riportate nel seguito:

punti	colore
00	da 53281
01	bits 0,1,2,3 ram video
10	bits 4,5,6,7 ram video
11	bits 0,1,2,3 colore da 55296.

Riferimento:

Notate che durante la visualizzazione in grafica multicolore, il video serve a memorizzare ora i COLORI.

Anche gli sprites possono essere visualizzati in MULTICOLOR: leggere le locazioni 53276,53285 e 53286.

Riferimento:

Se voi provate ad inserire contemporaneamente il multicolor e la visualizzazione a sfondo a piu' colori, il VIC si blocca ed il video si spegne.

Per uscire da tale situazione dovrete prendere un decisione in merito.
Si legga la locazione 53265.

Esempio:

Questo piccolo programma effettua lo scroll-fine orizzontale.

```
10 X=X+1:IFX=7THENX=-X
20 POKE53270,(PEEK(53270)AND240)ORABS(X)
30 GOTO10
```

Questo programmino utilizza il modo MULTICOLORE del testo.

```
10 POKE53281,9:PRINTCHR$(143)CHR$(147)
20 POKE53281,6:POKE53282,5:POKE53283,0
30 FORX=0TO255:POKE1024+X,X:NEXT
40 POKE 53270,216
```

LOCAZIONE 53271

Espansione in y per sprites.

Valore Normale: 0

Contenuto:

Gli sprites possono essere ingranditi indipendentemente l'uno dall'altro in tutte e due le direzioni.
Per espandere uno sprite in verticale (y) agiamo sul registro 53271.
Se il bit in tale locazione, e' acceso, conseguentemente e' raddoppiata verticalmente l'immagine dello sprite relativo.
Voi potete usare tale tecnica per simulare con lo sprite un oggetto al quale noi ci avviciniamo.

Compito dei bits:

0 sprite #0 espanso (1)/ normale (0)

1 sprite #1 espanso (1)/ normale (0)

2 sprite #2 espanso (1)/ normale (0)

3 sprite #3 espanso (1)/ normale (0)

4 sprite #4 espanso (1)/ normale (0)

5 sprite #5 espanso (1)/ normale (0)

6 sprite #6 espanso (1)/ normale (0)

7 sprite #7 espanso (1)/ normale (0)

Spiegazione:

Per espandere uno sprite in altezza, utilizzate la seguente formula:

```
POKE53271,PEEK(53271)OR2↑(NUM. SPRITE)
```

Per avere l'effetto opposto:

```
POKE53271,PEEK(53271)OR2↑(NUM. SPRITE)
```

Esempio:

Vedere la locazione 53277.

LOCAZIONE 53272

Vettore set caratteri e RAM video.

Valore Normale: 21

Contenuto:

Questo byte decide la ripartizione della memoria del VIC.
Questo vettore stabilisce dove e' allocato il video, ed il set dei caratteri; inoltre fissa l'inizio del BIT-MAP.

Cosi' potete disporre come volete della memoria agendo anche sulle locazioni: 43,44,45,46 e 648.

Anche il set di caratteri puo' essere spostato.
Voi potrete dunque creare a piacimento il vostro insieme di simboli e lettere.
Leggere dettagli alla voce 'set di caratteri'.

Compito dei bits:

0 non usato, sempre ad 1

1-3 puntatore inizio caratteri.

4-7 puntatore inizio RAM video.

Spiegazione:

I Bits #1,#2,#3 determinano la posizione si inizio del set di caratteri.
Sono possibili le seguenti combinazioni dei valori:

bits	memoria caratteri	'A'
XXXX000X	0-2047 ($0000-$07ff)	0
XXXX001X	2048-4095 ($0800-$0fff)	2
XXXX010X	4096-6143 ($1000-$17ff)	4
	inutilizzabili.	
XXXX011X	6144-8191 ($1800-$1fff)	6
	inutilizzabili.	
XXXX100X	8192-10239 ($2000-$27ff)	8
XXXX101X	10240-12287 ($2800-$2fff)	10
XXXX110X	12288-14335 ($3000-$37ff)	12
XXXX111X	14336-16383 ($3800-$3fff)	14

Il set di caratteri necessita di 2 K-BYTES.
L' ambito da 0-2047 non e' consigliabile poiche' vi risiede la pagina zero ed il video.
L'ambito da 2048 a 4095 e' solitamente occupato dal BASIC: voi potete trasferire il programma piu' in alto (vedere loc.43 e 44).
Da 4096 sino a 6143, inutilizzabile: il VIC legge la ROM da 53248 in cui ha sede la normale definizione del set di caratteri.
Lo stesso discorso per la memoria da 6144 a 8191.
Se voi lavorate in alta risoluzione, calcolerete allo stesso modo l'inizio del BIT-MAP.
Un blocco di 16K contiene solo due possibili posizioni del BIT-MAP: 0 e 8192.
Non prendete in considerazione la prima ipotesi: e' sede della pagina zero.

Voi potete fissare la locazione di inizio con la seguente formula:

```
POKE53272,(PEEK(53272)AND240)ORA
```

Il valore di A lo ricavate dalla tabella elencata sopra.

I bits #4,#5,#6,#7 stabiliscono le posizioni di inizio della ram-video.
Considerate che in alta risoluzione la mappa video si tramuta in mappa colore.
Notate bene che BIT-MAP e RAM del video, non devono assolutamente sovrapporsi.
Ecco le possibili posizioni dalla mappa video:

bits	memoria video	'B'
0000xxxx	0-1023 ($0000-$03ff)	0
0001xxxx	1024-2047 ($0400-$0bff)	16
0010xxxx	2048-3071 ($0800-$0fff)	32
0011xxxx	3072-4095 ($0c00-$0fff)	48
0100xxxx	4096-5119 ($1000-$13ff)	64
0101xxxx	5120-6143 ($1400-$17ff)	80
0110xxxx	6144-7167 ($1800-$1bff)	96
0111xxxx	7168-8191 ($1c00-$1fff)	112
1000xxxx	8192-9215 ($2000-$23ff)	128

1001xxxx 9216-10239 ($2400-$27ff) 144

1010xxxx 10240-11263 ($2800-$2bff) 160

1011xxxx 11264-12287 ($2c00-$2fff) 176

1100xxxx 12288-13311 ($3000-$33ff) 192

1101xxxx 13312-14335 ($3400-$37ff) 208

1110xxxx 14336-15359 ($3800-$3bff) 224

1111xxxx 15360-16383 ($3c00-$3fff) 240

Il video non puo' iniziare dalla locazione zero: qui e' allocata la pagina zero appunto.
Esso normalmete risiede dalla locazione 1024.
gli indirizzi 4096,5120,6144 e 7168 non possono essere utilizzati: il VIC legge i dati dalla ROM.
Ricordate che se spostate l'inizio del video, dovete aggiornare anche la locazione 648.
Prendete dalla tabella il valore di 'B' ed usate la seguente regola per spostare la mappa video:

```
POKE53272,(PEEK(53272)AND15)OR B
```

Riferimento:

I puntatori di definizione degli sprites vengono spostati se noi ovviamente alteriamo l'indirizzo di partenza della mappa video (leggere locazioni 2040-2047).

Riferimento:

Il VIC puo' indirizzare esclusivamente blocchi da 16 K-bytes.
Normalmente esso gestisce le locazioni 0-16384.
Questo ambito e' ricco di complicazioni: e' sede della pagina zero, del programma BASIC, ...
Voi potete spostare addirittura il blocco di 16K: come si fa lo leggete alla locazione 56576.
Evidentemente le due ultime tabelle non vanno piu' bene: occorre aggiungere al valore di memoria, il valore del banco da 16384 bytes usato.
Fate attenzione tuttavia, che set di caratteri e ram-video, debbono risiedere nello stesso banco da 16K.

Esempio:

Le seguenti righe spostano l'inizio del video alla locazione 3072.
Potete usare la memoria da 1024 sino a 2048 per la

definizione degli sprites.

```
10 POKE53272,53:POKE648,12
20 POKE44,16:POKE4096,0:NEW
```

Il seguente programma altera il set di caratteri:
esso copia la ROM caratteri nella RAM.
Si legga anche la locazione 1.

```
10 Z=2048
20 POKE56334,0:POKE1,51
30 FORX=0TO2047:POKEZ+X,PEEK(53248+X):NEXT
40 POKE1,55:POKE56334,1
50 POKE53272,(PEEK(53272)AND240)OR(Z/1024)
```

Poiche' i tempi di esecuzione di tale programma sono particolarmente lunghi, ne riportiamo la versione realizzata in linguaggio macchina.
Potremo, come al solito, allocare il programma dove ci pare. Dovremo a tale scopo agire sulla variabile A.

```
10 A=828:Z=2048
20 FORX=ATOA+47:READW:S=S+W:POKEX,W:NEXT
30 IFS<>7975THENPRINT'ERRORE IN DATA':STOP
40 POKE255,(Z/256)+7:POKE53272,(PEEK(53272)AND240)OR(Z/1024)
50 SYS A
60 DATA 120,162,0,134,252,134,254,162,215,
   134,253,162,8,134,251,162,51
70 DATA 134,1,160,255,177,252,145,254,136
   192,255,208,247,198,253,198,255
80 DATA 198,251,166,251,224,0,208,233,162,
   55,134,1,88,96
```

LOCAZIONE 53273

Registro delle interruzioni.

Valore Normale: -

Contenuto:

In questo registro puo' essere stabilito se il VIC ha generato una (o piu') interruzioni.

Ci sono 4 fattori che causano una interruzione:

1.
Il raster e' arrivato ad una riga pari a quella scritta in 53266.

2.
E' avvenuta una collisione sprite-sfondo.

3.
E' avvenuta una collisione sprite-sprite.

4.
La penna ottica fa conoscere tramite il foto-diodo la propria posizione.

Se una di queste condizioni e' verificata, si accende il relativo bit nel registro 53273.

Poiche' queste cause possono avvenire piu' volte nell'ambito di un secondo, e' impensabile poter indagare su tale registro, tramite il BASIC.

Potete pero' scrivere un programma in linguaggio macchina che prenda decisioni in merito a quello letto in tale registro indicatore.

Compito dei bits:

0	raster-interruzione.
1	collisione sprite-sfondo.
2	collisione sprite-sprite.
3	segnale dalla penna ottica.
4-6	sempre ad 1, inusati.
7	verifica di interruzione in corso.

Spiegazione:

Se il bit #0 e' acceso vuol dire che il raster e' arrivato a visualizzare la riga specificata nella locazione 53266.
Se il bit #1 e' acceso, e' in atto una collisione tra sprite e lo sfondo.
Se volete sapere quali sprites, leggete la locazione 53279.
Il bit #2 indica un urto tra piu' sprites accesi.
Per ulteriori informazioni leggere la locazione 53278.
Il bit #3 e' acceso se la penna ottica e' inserita ed ha segnalato la nuova posizione.
Il bit #7 viene sempre acceso se e' in corso una interruzione tra le 4 possibilita' descritte sopra.

E' questo un bit di veloce verifica.

LOCAZIONE 53274

lista delle interruzioni.

Valore normale: -

Contenuto:

I singoli Bits di questo registro determinano le attuali richieste di interruzioni del VIC (come descritto in 53273). Questo registro varia il proprio contenuto, praticamente senza sosta, sino a 60 volte al secondo: questo poiche' un sessantesimo di secondo per il microprocessore e' un tempo relativamente lungo.
Voi potete decidere quale delle possibili cause provocano l'interruzione del processore.
All'interno di questa interruzione possono essere esaminati i bits nella locazione 53273 e corrispondentemente essere utilizzati.

TABELLA

bit 0 : Rende possibile interruzione.

bit 1 : interruzione di collisione sprite-sfondo.

bit 2 : interruzione di collisione sprite-sprite.

bit 3 : interruzione per la penna luminosa.

bit 4-7 : non utilizzati.

Spiegazione:

Se il Bit #0 e' inserito, ha luogo subito l'interruzione del processore.

Il Bit #1 rende possibile un'interruzione del processore, non appena avviene una collisione fra lo sprite e lo sfondo.

Il Bit #2 indica un'interruzione del processore, non appena ha luogo una collisione fra 2 o piu' Sprites.

Il Bit #3 rende possibile un'interruzione del processore, non appena incontra un segnale dalla penna-luce.

Non appena una delle interruzioni suddette ha avuto luogo, viene acceso il Bit #7 nella locazione 53273 (vedrete poi).

Riferimento:

Questo Byte segnala solo l'effetto di una interruzione VIC del mondo esterno
(cioe' le altre esterne al computer).
Se la corrispondente interruzione non e' richiesta, l'appartenente Bit e' fissato al valore 0.
Al seguito del generarsi di una interruzione, potremo dunque indagare sulla sorgente che l'ha causata, andando a leggere in tale registro.

LOCAZIONE 53275

Indicatore priorita' sfondo-sprite.

Valore normale: 0

Contenuto:

Questo registro stabilisce per ogni sprite, se la relativa immagine appare davanti o dietro i caratteri presenti sul video.
Voi potete visualizzare ogni sprite dietro il testo ed avanti ad esso, ripetutamente, cosi' da raggiungere un effetto tridimensionale.

Compito dei Bits:

0 Spr.0 davanti (0)/ dietro (1) testo

1 Spr.1 davanti (0)/ dietro (1) testo

2 Spr.2 davanti (0)/ dietro (1) testo

3 Spr.3 davanti (0)/ dietro (1) testo

4 Spr.4 davanti (0)/ dietro (1) testo

5 Spr.5 davanti (0)/ dietro (1) testo

6 Spr.6 davanti (0)/ dietro (1) testo

7 Spr.7 davanti (0)/ dietro (1) testo

Spiegazioni:

Se un Bit e' inserito in questo registro lo sprite e' rappresentato dietro i caratteri.

Usate la seguente formula, per visualizzare lo sprite voluto dietro il testo:

POKE 53275, PEEK (53275) OR 2↑(numero sprite)

Se volete riportarlo alla condizione iniziale, usate la seguente formula:

POKE 53275, PEEK (53275) AND (255 - 2↑(numero sprite))

Esempio :

Questo programma sposta la nostra lepre avanti e dietro al video, alla pressione di un tasto.

```
10 FOR X=832 TO 894 : READ W : S=S+W :POKE X,W: NEXT
20 IF S<>4176 THEN PRINT 'ERRORE NEI DATA!':STOP
30 POKE 2040,13 : POKE 53269,1 : POKE 53287,15
40 FOR X=1TO4: POKE 211,18 : POKE 214, 12+X :SYS 58640
50 PRINT CHR$(144)CHR$(221)CHR$(221)CHR$(221):NEXT : POKE
53248,168 : POKE 53249,164
55 POKE 53275,1
56 GET A$ : IF A$<>"" THEN 56
57 POKE 53275,0 : END
60 DATA 0,36,0,0,102,0,0,102,0,0,102,0,
   0,102,0,0,126,0,0,219,0,1,255,128
70 DATA 1,231,128,0,195,0,0,60,0,0,126,0,
   0,219,0,0,219,0,0,219,0,1,219,128
80 DATA 1,126,128,1,126,128,7,66,224,
   15,195,240,0,0,0
```

LOCAZIONE 53276

Indica se gli sprites sono nel modo multicolore.

Valore normale: 0

Contenuto: in questa locazione puo' essere letto e determinato se uno sprite viene visualizzato nel modo multicolore.
In questo caso la risoluzione orizzontale si dimezza ed e' di 12 punti.
Tuttavia possono essere visualizzati contemporaneamente sino a 3 colori (4 con il colore dello sfondo).
Cosi' possiamo realizzare figure colorate.
Questo modo visualizzativo puo' essere attivato a piacere e relativamente a ciascun sprite.
Cosi' potete lavorare parzialmente con piu' sprites colorati e parzialmente con sprites ad alta risoluzione ma con un solo colore.

Riferimento:

Un effetto rilevante si ottiene sovrapponendo uno sprite visualizzato in modo multicolore ad uno che invece e' disposto in grafica ad un solo colore.

Compito dei Bits:

0 Sprite #0 multicolore (1)/ normale (0)

1 Sprite #1 multicolore (1)/ normale (0)

2 Sprite #2 multicolore (1)/ normale (0)

3 Sprite #3 multicolore (1)/ normale (0)

4 Sprite #4 multicolore (1)/ normale (0)

5 Sprite #5 multicolore (1)/ normale (0)

6 Sprite #6 multicolore (1)/ normale (0)

7 Sprite #7 multicolore (1)/ normale (0)

Spiegazione:

Se un Bit e' acceso in questo registro, l'appartenente sprite e' visualizzato in multicolore.
Seguite la seguente formula, per disporre uno sprite in multicolor:

```
POKE 53276, PEEK(53276) OR 2↑(numero Sprite)
```

Per renderlo di nuovo ad un solo colore, usate questa formula:

```
POKE 53276, PEEK(53276) AND 255 - 2↑(numero Sprite)
```

Nel modo multicolore vengono raggruppati sempre 2 punti adiacenti.
Il colore di un unico 'doppio-punto', dipende dalla corrispettiva combinazione di Bits.
Ulteriori informazioni le troverete nel seguito.

combinazioni 'doppio bit' e colori:

00 colore in 53281

01 colore in 53285

10 colore in 53287+numero sprite

11 colore in 53286

Esempio:

Qui un piccolo esempio per uno sprite multicolore:

```
10 FOR X=832 TO 894 : READ W : S=S+W :POKE X,W : NEXT
20 IF S<>4550 THEN PRINT 'ERRORE NEI DATA!':STOP
30 POKE 53248,160 :  POKE  53249,140  :  POKE  2040,14  :POKE
53269,1 : POKE 53276,1
40 POKE 53285,8 : POKE 53286,1 : POKE 53287,9
50 DATA 0,170,0,2,170,160,2,170,168,10,68,168,
   9,85,104,9,85,88
60 DATA 9,213,216,9,85,88,9,93,88,9,93,88,
   9,93,88,9,85,80
70 DATA 1,85,80,1,127,80,0,85,64,0,21,0,
   0,21,0,2,85,104
80 DATA 10,85,106,42,149,170,42,170,170
```

LOCAZIONE 53277

Espansione orizzontale degli sprites dallo #0 fino al #7

Valore normale: 0

Contenuto:

Gli sprites possono ampliarsi non solo nella direzione verticale come descritto in 53271 ma anche orizzontalmente. Questo ampliamento ha luogo indipendentemente per ogni sprite.
Voi potete anche ampliare solo alcuni sprites verticalmente, altri orizzontalmente, altri in entrambe le direzioni e altri per niente.
Per illustrare, ad esempio, l' avvicinarsi di un oggetto, lo sprite deve essere cambiato da 'piccolo' a 'grande'.

Compiti dei Bits:

0 sprite #0 grande (1) / piccolo (0)

1 sprite #1 grande (1) / piccolo (0)

2 sprite #2 grande (1) / piccolo (0)

3 sprite #3 grande (1) / piccolo (0)

4 sprite #4 grande (1) / piccolo (0)

5 sprite #5 grande (1) / piccolo (0)

6 sprite #6 grande (1) / piccolo (0)

7 sprite #7 grande (1) / piccolo (0)

Spiegazione:

L'ampliamento orizzontale di uno sprite viene attivato di modo che il Bit appartenente venga messo ad 1.
Per ampliare uno sprite orizzontalmente, seguite la seguente formula:

```
POKE 53277, PEEK(53277) OR 2↑(numero Sprite)
```

L'operazione inversa e':

```
POKE 53277, PEEK(53277) AND (255 - 2↑(numero Sprite))
```

Esempio:

Questo programma lavora con la solita lepre.

```
10 FOR X=832 TO 894 : READ W : S=S+W :POKE X,W : NEXT
20 IF S<>4176 THEN PRINT 'ERRORE NEI DATA!' :STOP
30 POKE 2040,13 : POKE 53269,1 : POKE 53287,15
40 POKE 53248,155 : POKE 53249,135
50 POKE 53271,1 : POKE 53277,1
60 DATA 0,36,0,0,102,0,0,102,0,0,102,0,
   0,102,0,0,126,0,0,219,0,1,255,128
70 DATA 1,231,128,0,195,0,0,60,0,0,126,0,
   0,219,0,0,219,0,0,219,0,1,219,128
80 DATA 1,126,128,1,126,128,7,66,224,
   15,195,240,0,0,0
```

LOCAZIONE 53278

Registro collisioni sprite-sprite.

Valore normale: 0 (sprites non usati).

Contenuto:

Questo registro rende possibile la determinazione delle collisioni tra Sprites.
Se queste sono in atto, il programma prendera' conseguenti decisioni.
Un esempio: se due veicoli spaziali collidono il programma provvedera' a farli esplodere.
Contemporaneamente ad una collisione, vengono accesi i Bits nel registro delle interruzioni 53273, e se la locazione 53274 lo permette ha luogo perfino un'interruzione.
Voi non dovete conoscere troppo bene il linguaggio macchina, per usare questo registro .
Anche da BASIC possiamo agire ugualmente, sebbene piu' lentamente.

Compito dei Bits:

0 Sprite 0 sta collidendo.

1 Sprite 1 sta collidendo.

2 Sprite 2 sta collidendo.

3 Sprite 3 sta collidendo.

4 Sprite 4 sta collidendo.

5 Sprite 5 sta collidendo.

6 Sprite 6 sta collidendo.

7 Sprite 7 sta collidendo.

Spiegazioni:

I Bits che appartengono agli Sprites che avevano partecipato alla collisione, sono accesi.
Con questa formula stabilite se uno sprite e' in collisione:

```
A=PEEK(53278)
IF A AND 2↑(Numero Sprite) THEN (...c'era!)
```

Perche' l'utilizzo della variabile A?
A proposito di entrambi i registri di collisione c'e' un importante principio per cui fare attenzione.
I Bits si spengono nel momento in cui vengono letti.
Dopo il primo PEEK (53278), il valore di questo registro e'=0.
Dovete fare attenzione ed esaminare piu' Sprites in collisione: voi dovete prima memorizzare il valore che avete ricevuto dal primo PEEK, affinche' non venga perso.
Quindi voi non dovete leggere nessun nuovo valore fino a quando non siete pronti per il successivo esame.

Riferimento:

Fate attenzione che gli Sprites anche fuori dell'ambito visibile possono collidere.
Non appena gli Sprites vengono accesi, attraverso la locazione 53269, ha inizio l'esame delle collisioni.
Se a questo punto avete ancora a zero tutte le coordinate, il registro di collisione salta al valore 255.
Prima dell'inizio dell'esame, assegnate a ciascuno sprites la propria posizione, quindi, se tutti gli Sprites sono al loro posto, date POKE 53278,0.

Esempio:

2 lepri nella riserva di caccia saltellano esattamente una dietro l'altra. Se litigano, cercano entrambi volentieri la lontananza.

```
 10 FOR X=832 TO 894 : READ W : S=S+W : POKE X,W : NEXT
 20 IF S<>4176 THEN PRINT "ERRORE NEI DATI!" : STOP
 30 POKE 2040,13 : POKE 2041,13 : POKE 53269,3 : POKE
    53287,15 : POKE 53288,15
 40 POKE 53249,140 : POKE 53251,140
 45 X=0 : POKE 53278,0
 50 X=X+2 : POKE 53248, ABS(X) : POKE 53250,255-ABS(X)
 55 IF PEEK(53278)=0THEN50
 56 X=X-1 : POKE 53248, ABS(X) : POKE 53250,255-ABS(X)
 57 IF X>0 THEN 56
 58 END
 60 DATA 0,36,0,0,102,0,0,102,0,0,102,
    0,0,102,0,0,126,0,0,219,0,1,255,128
 70 DATA 1,231,128,0,195,0,0,60,0,0,126,
    0,0,219,0,0,219,0,0,219,0,1,219,128
 80 DATA 1,126,128,1,126,128,7,66,224,
    15,195,240,0,0,0
```

LOCAZIONE 53279

Registro di collisione Sprite-Sfondo

Valore normale: 0

Contenuto:

Questo registro e' per l'esame della collisione tra le singole lepri, pardon!, Sprites, e lo sfondo fisso.
Quando uno Sprite copre il testo o la grafica in alta risoluzione, o viceversa, viene acceso il relativo bit in questa locazione.
Al verificarsi di una collisione sprite-sfondo, se la locazione 53274 lo permette, si genera una interruzione del processore.

Compito dei Bits:

0 Sprite #0 collisione col testo

1 Sprite #1 collisione col testo

2 Sprite #2 collisione col testo

3 Sprite #3 collisione col testo

4 Sprite #4 collisione col testo

5 Sprite #5 collisione col testo

6 Sprite #6 collisione col testo

7 Sprite #7 collisione col testo

Spiegazione:

I Bits degli Sprites che si scontrano con lo sfondo, vengono inseriti.
Essi rimangono inseriti fino a quando vengono letti con PEEK.
Percio' ecco la formula che esamina, se uno Sprite aveva preso parte al contatto:

```
A=PEEK(53279)
IF A AND 2↑(numero Sprite) THEN (...c'era!)
```

Per ulteriori chiarimenti consultate 53278.

Esempio:

La nostra lepre corre per sbaglio contro una parete.
Voi potete usare il dato letto nella locazione 53279.

```
10 FOR X=832 TO 894 : READ W : S=S+W :POKE X,W : NEXT
20 IF S<>4176 THEN PRINT 'ERRORE NEI DATA!' :STOP
30 POKE 2040,13 : POKE 53269,1 : POKE 53287,15
35 FOR X=1 TO 5 : POKE 211,30 : POKE 214,8+X :SYS 58640 :
PRINT CHR$(18)"  ":NEXT
40 POKE 53249,140 : POKE 53279,0
50 X=X+1 : POKE 53248,x : IF PEEK (53279)=0 THEN 50
55 X=X-1 : POKE 53248,X : IF X>0 THEN 55
57 END
60 DATA 0,36,0,0,102,0,0,102,0,0,102,0,
   0,102,0,0,126,0,0,219,0,1,255,128
70 DATA 1,231,128,0,195,0,0,60,0,0,126,0,
   0,219,0,0,219,0,0,219,0,1,219,128
80 DATA 1,126,128,1,126,128,7,66,224,
   15,195,240,0,0,0
```

LOCAZIONE 53280

Colore della cornice del video.

Valore normale:
254 (corrisponde al colore 14)

Contenuto:

In questo registro viene indicato il colore che deve assumere la cornice del video.

Riferimento:

Se selezionate un colore e leggete il registro appena dopo, non meravigliatevi della cifra enorme.
Avviene che tutti i Bits, che non sono occupati dal VIC, hanno sempre il valore 1.
Solo i Bits da 0 fino al 3 sono responsabili del colore.
I Bits maggiori sono tutti ad 1.
Cosi' leggete il codice colore da un valore di 240.
Voi otterrete il vero numero-codice mediante:

```
PRINT PEEK (53280) AND 15
```

Al contrario: dei numeri piu' grossi di 15 vengono utilizzati solo i primi quattro Bits.

Compito dei Bits: -

codici colore:

Ecco un prospetto dei codici colore che possono essere usati con POKE in questo e in tutti i registri dei colori:

numero	colore
0	nero
1	bianco
2	rosso
3	cyan
4	rosa
5	verde
6	blu
7	giallo
8	arancio
9	marrone
10	rosso chiaro
11	grigio scuro
12	grigio medio
13	verde chiaro
14	blu chiaro
15	grigio chiaro.

Esempio:

Questo programma lascia passare sullo sfondo tutti i possibili 16 colori:

```
10 FOR X=0 TO 15 : POKE 53280,X
20 FOR Y=1 TO 500 : NEXT Y,X
```

LOCAZIONE 53281

Colore sfondo

Valore normale:
246 (corrisponde al codice colore 6)

Contenuto:

Questo registro decide il colore dello sfondo.
Per ulteriori informazioni leggete locazione (53280).

Riferimento:

Puo' suonare banale, ma vi si potra' presentare davanti agli occhi una volta: tutti i punti di uno Sprite, che hanno valore 0 (presso l'illustrazione multicolore 00) utilizzano questo stesso colore di fondo.

Compito dei Bits: -

Esempio:

Ecco un programma, che lascia passare tutti i possibili 16 colori.

```
10 FOR X=0 TO 15 : POKE 53281,X
20 FOR Y=1 TO 500 : NEXT Y,X
```

LOCAZIONE 53282

Colore dello sfondo.

Valore normale:
241 (corrisponde al codice colore 1)

Contenuto:

Questo registro viene usato se e' attivo il modo multicolore, per decidere uno degli ampliati colori di sfondo (vedere le locazione 53265 e 53270).

Compito dei Bits: -

Spiegazione:

Questo registro e' importante esclusivamente per l'illustrazione del testo nel metodo multicolore: per la maggior parte dei casi si puo' fare a meno di tale registro.
Il codice colore specificato in tale locazione viene assegnato alla coppia 01.
Se lavoriamo in multicolore vengono accettati con questo colore, i simboli con il codice da 64 fino a 127.

Esempio:

Un esempio per il multicolore del testo lo trovate sotto la locazione 53270, per lo sfondo alla locazione 53265.

LOCAZIONE 53283

Codice colore n.2 per lo sfondo.

Valore normale:
242 (corrisponde al codice colore 2)

Contenuto:

Anche questa locazione e' in collegamento col testo-multicolore (leggere le locazioni 53265 e 53270)

Compito dei Bits: -

Spiegazioni:

Nel metodo multicolore il colore deciso da questa locazione e' assegnato alle coppie di bits 10.
Il colore viene assegnato dal codice 127 fino al 191.

Esempio:

Un esempio di multicolore per i testi lo trovate alla fine della locazione 53270, per lo sfondo con colori ampliati alla locazione 53265.

LOCAZIONE 53284

Sfondo colorato 3

Valore normale:
243 (corrisponde al codice colore 3)

Contenuto:

Questo registro viene usato solo per la condizione di multicolore attiva.

Compito dei Bits: -

Spiegazione:

Il colore qui deposto viene assegnato ai caratteri con il codice video 192 fino al 255-ésimo.

Esempio:

Un esempio per lo sfondo a piu' colori, lo trovate alla locazione 53265.

LOCAZIONE 53285

Colore 1 per Sprites in multicolor.

Valore normale:

244 (corrisponde al codice 4)

Contenuto:

Gli Sprites possono essere visualizzati nel metodo multicolore.
Come cio' funziona lo apprenderete dettagliatamente alla locazione 53276.
In questo registro viene deposto solo 1 dei 3 colori del multicolore.
Percio' tutti i 16 colori sono a disposizione.

Se gli Sprites lavorano nel metodo multicolor, TUTTI hanno 2 colori in comune.
Questi colori vengono deposti nella locazione 53285 e 53286.
Il terzo colore e' individuale e puo' per ogni singolo Sprite essere fissato nell'appartenente registro dei colori dalla locazione 53287.

Compito dei Bits: -

Spiegazione:

Il colore qui deposto e' assegnato alle coppie di punti 01.

Esempio: Un esempio per uno Sprite multicolor lo trovate presso la locazione 53276.

LOCAZIONE 53286

Colore 2 per Sprites in multicolor.

Valore normale:
240 (corrisponde al numero 0)

Contenuto:

Qui e' deposto il secondo colore comune agli sprites in multicolore (leggere la locazione 53285).

Compito dei Bits: -

Spiegazione:

Il colore qui deposto e' assegnato alla coppia-punti 11.

Esempio:

Un esempio per uno Sprite multicolore lo trovate presso la locazione 53276.

LOCAZIONE 53287

registro colore dello sprite 0.

Valore normale:
241 (corrisponde al numero 1)

Contenuto:

In questa posizionne viene deposto il colore, che lo Sprite #0 ha, non appena appare sullo schermo. Voi potete dare ad ogni Sprite un colore qualsiasi dei 16.

Compito dei Bit: -

Spiegazione:

I punti, che sono fissati presso lo Sprite, ricevono questo colore. Se usate gli Sprites multicolor (confronta la locazione 53276 e 53285), puo' essere stabilito questo colore.
In questo caso i punti che ricevono tale colore sono la combinazione '10'.

Riferimento:

Per gli Sprites vale la stessa cosa, che abbiamo gia' detto alla locazione (646): se lo Sprite ha lo stesso colore dello sfondo non lo si puo' vedere.
L'esame delle collisioni funziona pero' normalmente (locazione 53278/53279).

LOCAZIONI 53288-53294

Registro dei colori (sprites).

Valore normale: vedete il sommario

Contenuto:

Per esso vale la stessa cosa, che e' gia' stato spiegato in precedenza (confronta 53287).

.....

LOCAZIONI 53312-54254

Posizione fata morgana del VIC

Contenuto:

Originalmente l'ambito in questione e' libero.
Ma sulla base di una caratteristica della gestione video, si unisce di fatto al VIC: sono qui allocati altri 15 VICs fantasmi (ne e' presente solo uno: solo le locazioni si ripetono).
Si puo' diversamente disporre di tali locazioni riflesse.

POKE 53280+15*64,0

causa la stessa cosa di:

POKE 53280,0

Se voi trovate in un programma una locazione come 54097, non allarmatevi dunque!!
Saprete infatti la risposta, conoscendo le Pokes-fata-morgana, come noi le abbiamo chiamate.

Esempio:

Ancora un piccolo esempio affinche' crediate:

```
 10 FOR X=0 TO 15
 20 POKE 53280 + X*64,X
 30 FOR Y=1 TO 500 : NEXT Y,X
```

.....

SID

DISPOSITIVO DI INTERFACCIA SONORA

SID 6581

SOUND INTERFACE DEVICE

Il Commodore 64 contiene un potente Sound-Synthesizer il SID-Chip.
Questo chip puo' produrre 3 voci completamente indipendenti l'una dall'altra.
Per ogni voce puo' essere programmata una frequenza, una determinata curva d'inviluppo, una determinata forma d'onda e nel caso dell'onda rettangolare, una precisa larghezza.
Queste voci possono essere modulate o sincronizzate: le possibilita' della produzione del suono sono veramente notevoli.

In ultimo e' presente anche un filtro, con il quale possiamo attenuare la voce, in una delimitata fascia di frequenza.
L'intero sistema sonoro e' gravato da un grosso svantăggio: per la produzione di un unico ed elementare tono, dovete scrivere molte POKEs.
I suoni del Commodore 64 sono molto complessi: potete abituarvi a queste funzioni alquanto complicate, facendo ripetuti esperimenti con i registri SID.
Poi procederete al meglio col seguente ordine:
(Vi abbiamo riportato in queste locazioni solo degli esempi della voce 1).

Prima dovete stabilire il volume da 0 fino a 15.
Per questo motivo ci serviamo della locazione 54296, che e' fissa.
Decidete dunque eventualmente anche il desiderato valore

```
POKE 54296,15
```

Selezionate anche la frequenza desiderata, che viene deposta come High Byte/Low Byte nelle locazioni 54273/54274.

```
POKE 54273,20 : POKE 54272,10
```

Stabilite poi la ADSR -curva d'inviluppo-.
Questa curva d'inviluppo stabilisce l'andamento del volume all'interno del tono. Percio' ha 4 valori: ATTACK, DECAY, SUSTAIN, RELEASE.
Il valore Attack, che stabilisce il tempo (da 0 fino a 15), che usa il tono per raggiungere il volume massimo, e il valore Decay, che stabilisce il tempo del decadimento, vengono definiti nella locazione 54277.
Sustain e Release, con valori ancora tra 0 e 15, vanno nella locazione 54278.
Nella locazione 54277 e 54278 e' scritto tutto quello che volete sapere a riguardo di questo argomento.

```
POKE 54277,168 : POKE 54278,252
```

Nel caso esclusivo in cui desideriate avere la forma d'onda rettangolare occorre programmarne la larghezza interna:

```
POKE 54274,20 : POKE 54275,8
```

Voi avete 4 forme d'onda da selezionare: quadrata, a dente di sega, rettangolare, e rumore.
Noi ci siamo decisi per la terza.
Se voi inserite il Bit #0, risuona il tono, e incomincia nuovamente il ciclo ADSR della curva d'inviluppo.

```
POKE 54276,65
```

Voi potete selezionare un tono qualsiasi all'interno di un programma, piu' vantaggioso a seconda dei casi.
Per introdurre il ciclo Release della curva d'inviluppo che e' stabilito per l'abbassamento del tono, il Bit #0 deve essere spento nella locazione 54276.

```
POKE54276,64
```

Ulteriori dettagli e possibilita' sul SID-sound-chips le troverete nelle descrizioni delle seguenti locazioni.

LOCAZIONI 54272-54273

Voce 1: frequenza

Valore normale: il contenuto del registro SID non puo' essere letto con una PEEK.
Tutti i registri SID hanno direttamente il valore dell'inserimento.
Il contenuto di questa locazione e' tale sino alla prossima POKE.

Contenuto:

In entrambe le locazioni la frequenza per la voce 1 viene stabilita come sequenza di High Byte e Low Byte.
Complessivamente 16 Bits sono a disposizione del SID : si hanno ben 65535 differenti frequenze.
Queste frequenze ricoprono un ambito di 8 ottavi.
Se volete programmare della musica, avete bisogno di una tabella, che vi metta a disposizione il valore esatto delle POKEs per ogni frequenza (vedere sotto).
Per creare del rumore il migliore modo e' procedere per tentativi.

Compito dei Bits: -

Spiegazioni:

Se voi selezionate la frequenza desiderata (0 e' il tono piu' basso, 65535 il piu' alto), usate la seguente formula:

```
HB=INT((frequenza)/256)
LB=(frequenza)-HB*256
POKE 54272
POKE 54273,HB
```

Il valore della frequenza non corrisponde al normale indicatore centrale.
Se volete convertire una determinata locazione centrale, potete usare la seguente formula approsimativa:

SID (valore di frequenza)=(frequenza in Hz)*17.0284

Per questo potete anche programmare senza una conoscenza professionale.

Ecco la scala musicale in C maggiore.

Tono	Low Byte	High Byte
C	103	17
#C	112	18
D	137	19
#D	178	20
E	237	21
F	59	23
#F	157	24
G	20	26
#G	160	27
A	69	29
#A	3	31
H	219	32
C	207	34

Riferimento:

Nei manuali del Commodore trovate le tabelle per tutte le 8 ottave.
Ma per precauzione, se trattate con l'edizione americana, sappiate che il SID e' usato come base per una indiretta produzione delle frequenze: tutto questo in Europa e' diverso poiche' le norme televisive e le corrispondenti reti di frequenza (USA 60 Hz,Europa 50 Hz) sono differenti.
Percio' i valori europei non sono d'accordo al 100% con i manuali americani (Confronta anche 678).

Esempio:

Questo programma attiva il SID per la totalita' delle sue frequenze.
Le Pokes nella riga 20 vengono spiegate dalle corrispondenti note.
Con la riga (20a) sentite esattamente tutte le frequenze.
l'esecuzione di questo programma dura ben 17 minuti a causa della lentezza del Basic.

```
10 POKE 54296,15 : POKE 54278,240 : POKE 54276,33
20 FOR X=0 TO 255 : POKE 54273,X : NEXTX
(20a) FOR X=0 TO 255 : FOR Y=0 TO 255 : POKE 54272,Y:POKE
54273,X : NEXT Y,X
```

LOCAZIONI 54274-54275

Voce 1: def. onda rettangolare.

Contenuto:

Se viene scelta la forma d'onda 'rettangolare' (confronta 54276), deve esserne definita la larghezza interna, in piu' del solito.
Essa varia fra lo 0 e 4095.
Voi stabilite la proporzione fra le durate delle due fasi rettangolari dell'onda (vedi l'illustrazione sottostante).
Mediante la selezione di questi valori il risultante tono muta.
Quanto piu' la lunghezza delle due fasi e' simile in durata, tanto piu' il timbro diventa cupo.

Compito dei Bits:

Locazione 54274 (Low Byte):

Tutti i Bits inseriti.

Locazione 54275 (High Nibble):
0 - 3 inseriti
4 - 7 non usati

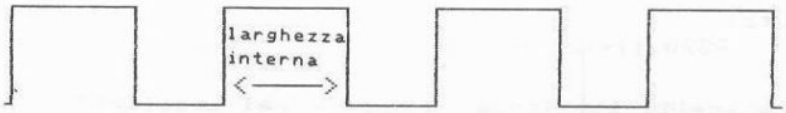
Spiegazioni:

La larghezza interna puo' essere indicata da 0 fino a 4095.
Per l'immagazzinamento di queste cifre si necessita di 12 Bits.
Percio' gli 8 Bits minori vengono deposti nella locazione 54274 e i 4 Bits maggiori (mezzo Byte = un Nibble) nella locazione 54275.
I 4 Bits maggiori (4 fino a 7) non sono utilizzabili.

La formula con la quale e' possibile trovare la larghezza interna e' la seguente:

```
HN=INT((larghezza interna)/256)
Lb=(larghezza interna-HN*256)
POKE 54274,LB
POKE 54275,HN
```

Questo schema dovrebbe spiegare, di quale caratteristica goda l'oscillazione rettangolare, e quale influenza rivesta la larhezza interna:



I valori 0 e 4095 fanno in modo che, nessun suono venga emesso.
Il valore 2048, che e' esattamente nel mezzo, causa una regolare oscillazione del rettangolo, cioe' tutti i rettangoli hanno la stessa larghezza.
Gli altri valori sono messi in modo simmetrico rispetto a questa regolare oscillazione.
Questo vuol dire che un'oscillazione con una larghezza interna di 1536 (2048-512) manda un suono simile, ad una oscillazione con la larghezza interna 2560 (2048+512).

Esempio:

Questo programma lascia aumentare la larghezza da 0 fino a 2048.
Le PoKes nella riga 10 vengono spiegate dalle rispettive locazioni.
Il valore STEP indicato nella riga 20, sale velocemente come la larghezza interna.

```
10 POKE 54296,15 : POKE 54278,240 :POKE 54273,20 :POKE
54276,65
20 FOR X=0 TO 2048 STEP 8
30 H=INT(X/256) : L=X-H*256 : POKE 54274,L :POKE 54275,H
40 NEXT
```

LOCAZIONE 54276

Voce 1: registro di controllo

Contenuto:

Questa locazione comprende molte funzioni di cui la piu' importante e' la scelta della forma d'onda.

Ce ne sono 4 a disposizione, triangolo, dente di sega, rettangolare e fruscio.
Uno dei Bits del registro e' fisso, rispettivamente il tono per iniziare o per smettere.
Le altre due funzioni sono la sincronizzazione e la modulazione circolare.
In questo registro la sincronizzazione puo' selezionare la voce fra 1 e 3.
Per cui succede che: per la voce 1 viene, come di consueto, programmato un tono; per la voce 3 viene accesa la frequenza (vedi locazioni 54286/54287); se solo il tono della voce 1 inizia e nello stesso momento la sincronizzazione e' inserita, la frequenza dei toni dipendenti dalla frequenza della voce 3 terminano.
Anche per la modulazione circolare la frequenza della voce 3 si mischia con un valore della voce 1.
Se la voce 1 e' programmata come onda rettangolare e la modulazione viene attivata, si abbassano i toni dipendenti dalla frequenza della voce 3 (viene 'modulata') se il risultante tono non ha molti sovratoni armonici.
A seconda del numero delle frequenze di modulazione puo' essere provocato un rumore spaziale con una risonanza 'galattica'.
In ultima analisi, in questa locazione e' presente un Bit di controllo con il quale puo' essere disponibile una voce 'muta'.

Compito dei Bits:

0 Gate-Bit: Tono Start(1)/Stop(0)
1 Sincronizzazione voce 1 con 3
accesa(1) / spenta(0)
2 Modulazione circolare voce 1 con
voce 3 acceso(1) / spenta(0)
3 Test-Bit: voce 1 bloccata RESET(1)/normale (0)
4 Forma d'onda triangolo
5 Forma d'onda denti di sega
6 Forma d'onda rettangolo
7 Forma d'onda fruscio

Spiegazioni:

Il Bit 0 serve per l'immagazzinamento dell'ADSR (curva d'inviluppo, confronta con locazione 54277 e 54278).
Se questo Bit e' inserito s'accende il tono.
Esso raggiunge a seconda del valore Attack il suo volume massimo, cade da esso mediante il valore Release ad una velocita' stabilita al volume che e' specificato dal valore del SUSTAIN.
Il tono decade quando il valore-Release e' inserito.
Occorre aggiungere semplicemente Poke 54276,0 per completare il tono.
Tuttavia dei nuovi valori possono essere inseriti nei registri, quando il Bit #0 non e' attivo,

Riferimento:

Se il Bit 0 viene spento, prima che un ciclo sia concluso

incomincia il nuovo ciclo, in termini piu' chiari: se il Bit #0 si spegne (posto a 0) mentre il tono e' ancora in fase Attack o Decay, e non ha ancora raggiunto il Level-Sustain incomincia subito il Release.

Se il Bit #1 e' inserito la sincronizzazione ha luogo fra la voce 1 e la voce 3. Percio' viene usata la frequenza della voce 3 (locazione 54286 e 54287) per modificare la frequenza della voce 1.
Poiche' la voce 1 produce nella maggior parte dei casi il tono principale, che attraverso la sincronizzazione viene reso estraneo, la frequenza nella voce 3 deve essere bassa come nella voce 1.

Riferimento:

Per raggiungere un effetto udibile la frequenza nella voce 3 deve essere 0. Gli altri registri della voce 3 non hanno influenza.
Il Bit #2 inserisce la modulazione circolare della voce 1 con la voce 3.
Il presupposto e' che per la voce 1 sia fissata la forma d'onda rettangolare cioe' che il Bit #4 sia inserito. In questo caso ha luogo una frequenza 'mixata' della voce 1 e 3.

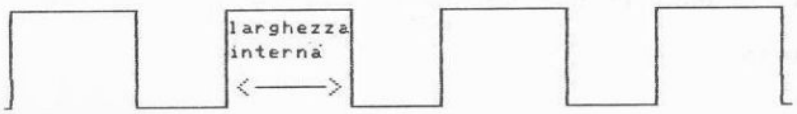
Riferimento:

Se il Bit 3 e' inserito, la voce 1 viene esclusa e nello stesso momento azzerata. Questa funzione puo' diventare necessaria, se la forma d'onda viene scelta contemporaneamente con un'altra forma, poiche' in questo caso la voce puo' essere bloccata.

Bit #4 seleziona la forma d'onda triangolare.

Bit #5 sceglie la forma d'onda a dente di sega.

Bit #6 sceglie la forma d'onda rettangolare. In questo caso deve essere definita anche la larghezza contenuta nei registri 53274 e 53275.



Bit #7 scieglie la forma d'onda rumore. Si tratta di una oscillazione casuale, che puo' essere usata per tutti i rumori: esplosioni,meccanismi del motore .....

Riferimento:

Voi potete a vostra scelta, mischiare le forme d'onda, accendendo piu' di un Bit. Il rumore non va combinato con altre forme d'onda, poiche' il SID puo' bloccarsi. Se voi inoltre selezionate la forma rettangolare con un'altra, dovrete tenere l'oscillazione il piu' bassa possibile poiche' essa si puo' sovrapporre con le altre, alterando il tono.

Esempi:

Questo programma usa lo stesso tono in tutte e quattro le forme d'onda. Le POKE nelle righe 10 e 20 vengono spiegate nelle locazioni corrispondenti. Il valore X informa quale Bit e' inserito.

```
 10 POKE 54296,15 :  POKE 54278,240 : POKE 54273,30
 20 POKE 54275,8
 30 POKE 54276,2↑x+1 : PRINT X
 40 POKE 198,0 : WAIT 198,1 : POKE 54276,2↑X
 50 X=X+1 : IF X=8 THEN X=4
 60 GOTO 30
```

In questo esempio abbiamo fatto attenzione a non mischiare anche il rumore con le altre forme d'onda. Percio' dovete effettuare per sicurezza dopo la scelta di ogni tono un RESET della voce attraverso l'inserimento del Bit #3. Disponete in aggiunta al suddetto esempio le seguenti righe, in questo modo:

```
 20 POKE 54275,4
 30 POKE 54276,16*X+1 : PRINT CHR$(147)

 31 IF X AND 1 THEN PRINT 'TRIANGOLO'
 32 IF X AND 2 THEN PRINT 'DENTE DI SEGA'
 33 IF X AND 4 THEN PRINT 'QUADRATA'
 34 IF X AND 8 THEN PRINT 'RUMORE'
 40 POKE 198,0 : WAIT 198,1 : POKE 54276,16*X+8
 50 X=X+1 : IF X=16 THEN X=0
```

Il seguente programma mostra le possibilita' di sincronizzazione. La forma d'onda scelta e' dente di sega.

```
 10 POKE 54296,15 : POKE 54278,251 : POKE 54276,35
 20 FOR X=0 TO 255
 30 POKE 54273,X : POKE 54287,255-X
 40 NEXT
 50  POKE 542776,32
```

Ecco un'altro piccolo esempio di modulazione. La voce 3 e' regolata su una frequenza molto bassa, mentre la voce 1 percorre lo spettro di frequenza. Se volete fare alcuni esperimenti, dovete modificare il valore nella locazione 54287 (voce 3: frequenza).

```
 10 POKE 54296,15 : POKE 54278,251 : POKE 54276,21 :
    POKE 54287,4
 20 FOR X=0 TO 255
 30 POKE 54273,X
 40 NEXT
 50 POKE 54276,16
```

LOCAZIONE 54277

Voce 1: AttacK/Decay

Contenuto:

In questo registro e nel prossimo, viene stabilita la curva d'inviluppo del tono.
Questa curva d'inviluppo viene scritta mediante 4 determinati valori: AttacK, Decay, Sustain e Release.
In questa locazione vengono decisi entrambi i valori AttacK e Decay.
La curva d'inviluppo stabilisce l'andamento del volume piu' alto del tono.
Il valore AttacK e' il tempo che un tono ha bisogno per raggiungere il suo volume piu' alto.
Decay e' il tempo, necessario che esso impiega per ritornare al suo volume di base.
Il ciclo-AttacK ricomincia non appena il Bit #0 viene attivato nella locazione 54276.

Compito del Bits:

0-3 Valore Decay

4-7 Valore AttacK

Spiegazione:

Per entrambi i valori sono a disposizione i rispettivi 4 Bits.
Il valore AttacK viene memorizzato nei maggiori 4 Bits, il valore Decay nei minori 4 Bits.
Voi potete usare la seguente formula per trovare il valore delle POKEs da dare in questo registro:

POKE 54277, (AttacK)*16 + (Decay)

La seguente tabella informa sui possibili valori ottenibili.

Valore	Attack	Decay
0	0.002 s	0.006 s
1	0.008 s	0.024 s
2	0.016 s	0.048 s
3	0.024 s	0.072 s
4	0.038 s	0.114 s
5	0.056 s	0.168 s
6	0.068 s	0.204 s
7	0.080 s	0.240 s
8	0.100 s	0.300 s
9	0.250 s	0.750 s
10	0.500 s	1.5 s
11	0.800 s	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

Esempio:

Lo troverete alla locazione 53278.

LOCAZIONE 53278

Voce 1: Sustain/Release

Contenuto:

In questo registro vengono deposti i valori Sustain e Release della curva d'inviluppo della voce 1.
Sustain e' il volume di base del tono, che viene raggiunto dopo che il tono ha percorso il ciclo-Decay.
Il tono rimane sul suo Sustain-Level fino a quando il Bit 0 viene spento nella locazione 54276.
Il tempo, che e' necessario, viene determinato mediante il valore Release.

Compito dei Bits:

0-3	Valore Release
4-7	Valore Sustain

Spiegazione:

Per entrambi i parametri sono a disposizione 4 Bits.
Voi potete usare le POKEs in base alla seguente formula:

POKE 54278, (Sustain)*16 + (Release)

Per il Sustain potete indicare un relativo tono, da 0 a 15, dove 0 non si sente e il 15 e' il piu' alto.

Riferimento:

Il volume complessivo per tutte e 3 le voci viene indicato nella locazione 54296.
Per il Release sono possibili i seguenti valori:

Valore	Release	Valore	Release
0	0.006 s	8	0.300 s
1	0.024 s	9	0.750 s
2	0.048 s	10	1.5 s
3	0.072 s	11	2.4 s
4	0.114 s	12	3 s
5	0.168 s	13	9 s
6	0.204 s	14	15 s
7	0.240 s	15	24 s

Esempio:

Il seguente programma suona lo stesso tono, ma chiede il valore per Attack, Decay, Sustain e Release. Cosi' potete fare esperimenti piu' facilmente.

```
10 POKE 54296,15 : POKE 54273,30
20 INPUT 'ATTACK';A
30 INPUT 'DECAY';D
40 INPUT 'SUSTAIN';S
50 INPUT 'RELEASE';R
60 POKE 54277,A*16+D
70 POKE 54278,S*16+R
80 POKE 54276,33
90 FOR X=1 TO 2000 : NEXT
100 POKE 54276,32
110 PRINT : GOTO 20
```

LOCAZIONI 54279-54280

Voce 2: Frequenza

Contenuto:

Questi registri determinano la frequenza nel High Byte/Low Byte.
Tutti gli altri fanno lo stesso che per voce 1.
Ulteriori informazioni alla locazioni 54272/54273.

LOCAZIONI 54281-54282

Voce 2: larghezza interna

Contenuto:

Qui viene indicata la larghezza interna dell' oscillazione della voce 2 se definita in forma rettangolare.

La forma d'onda della voce 2 viene indicata nella locazione 54274/54275.

LOCAZIONE 54283

Voce 2: Registro di controllo

Contenuto:

In questo registro possono essere riposte le forme d'onda; per la scelta ci sono come sempre il triangolo, rettangolo, dente di sega o il fruscio.
Inoltre viene immagazzinato nel Gate-Bit di questo registro, la curva d'inviluppo.
Esiste anche la possibilita', di attivare la sincronizzazione fra la voce 2 e la 1, la modulazione circolare della voce 2 con la voce 1.
Con un test del Bit la voce 2 puo' essere inserita di nuovo.

Compito dei Bits:

0 Gate-Bit: inizio tono(1)/stop(0)
1 Sincronizzazione voce 2 con voce 1 acceso(1)/spento(0)
2 Modulazione circolare voce 2 con voce 1 acceso(1)/spento(0)
3 Test Bit: voce 2 bloccata e RESET(1)/normale(0)
4 Forma d'onda triangolo
5 onda a dente di sega
6 onda rettangolare
7 rumore.

La stessa cosa accade per la voce 1. Ulteriori informazioni alla locazione 54276.

LOCAZIONE 54284

Voce 2: Attack/Decay

Contenuto:

In questo registro vengono indicati i valori Attack e Decay della curva d'inviluppo della voce 2.
Il ciclo Attack incomincia, se il Bit #0 viene attivato nella locazione 54283.
La stessa situazione accade alla voce 1. Ulteriori informazioni le trovate alla locazione 54277.

LOCAZIONE 54285

Voce 2: Sustain/Release

Contenuto:

Questa locazione comprende sia il valore Sustain che il valore Release della curva d'inviluppo della voce 2.
Il tono viene mantenuto sul livello Sustain fino a quando il Bit #0 nella locazione 54283 e' posto a zero.
Poi incomincia il ciclo-Release.
La stessa cosa accade alla voce 1. Ulteriori informazioni si trovano nella locazione 54277.

LOCAZIONI 54286-54287

Voce 3: Frequenza

Contenuto:

Questo registro comprende come sequenza High Byte/Low Byte la frequenza per la voce 3.
La stessa cosa succede per la voce 1.
Ulteriori informazioni le trovate alla locazione 54272/54273.

LOCAZIONI 54288-54289

Voce 3: Larghezza interna

Contenuto:

Qui viene indicata la larghezza interna dell'oscillazione rettangolare della voce 3.
La forma d'onda viene indicata nella locazione 54290.
La stessa cosa accade per la voce 1.
Ulteriori informazioni le trovate alle locazioni 54274/54275.

LOCAZIONE 54290

Voce 3: Registro di controllo

Contenuto:

In questo registro puo' essere posta la forma d'onda; si dispone di onde triangolari, a dente di sega, rettangolari e rumore.
Inoltre viene immagazzinato nel Gate-Bit la curva d'inviluppo nel registro.
Sussiste ancora la possibilita' della scelta della sincronizzazione fra la voce 3 e 2 come fra la modulazione circolare con la voce 3 e 2.

Compito dei Bits:

0 Gate-Bit: Tono start(1)/Stop(0)
1 Sincronizzazione voce 3 con voce 2 acceso(1)/Spento(0)
2 Modulazione circolare voce 3 con voce 2 acceso(1)/spento(0)
3 Test Bit: voce 3 bloccata e RESET(1)/normale(0)
4 Forma : triangolo
5 Forma : dente di sega
6 Forma : rettangolo
7 Forma : fruscio

Tutto questo accade anche per la voce 1.
Ulteriori informazioni le trovate alla locazione 54276.

LOCAZIONE 54291

Voce 3: Attack/Decay

Contenuto:

In questo registro vengono indicati il valore Attack e di Decay relativi alla curva d'inviluppo della voce 3. Il ciclo-Attack comincia, se il Bit #0 della locazione 54290 e' attivato.

La stessa cosa accade per la voce 1.
Ulteriori informazioni le trovate alla locazione 54277.

LOCAZIONE 54292

Voce 3: Sustain/Release

Contenuto:

Questa locazione comprende sia il valore Sustain che Release della curva d'inviluppo della voce 3.
Il tono viene mantenuto nel livello Sustain, fino a che il Bit #0 della locazione 54290 e' posto a zero.
Poi incomincia il ciclo-Release.

La stessa cosa accade per la voce 1.
Ulteriori informazioni le trovate alla locazione 54277.

LOCAZIONI 54293-54294

Frequenza di filtro.

Contenuto:

Col SID sussiste la possibilita' di condurre ognuna delle 3 voci ad azione di filtro, che puo' escludere determinata parte dello spettro di frequenza.
I registri in questione determinano appunto la frequenza-filtro.
Essi variano tra un contenuto pari allo 0 sino a 2047. Nella locazione 54295 puo' essere determinata quale fonte del segnale viene filtrata.
I determinati modi di filtro, che decidono quale zona di frequenze viene filtrata, vengono scelti nella locazione 54296.

Compito dei Bits:

Locazione 54293 (Low Nibble)

0-2 Bits 0-2 frequenza di filtro

3-7 non usato, sempre 0

Locazione 54294 (High Byte):

0 - 7 Bits 3-10 frequenza di filtro.

Spiegazione:

La frequenza di filtro varia fra lo 0 e 2047. Questo valore non corrisponde al valore della frequenza selezionato nella locazione 54272/73, 54279/80 e 54286/87. Tuttavia conoscendo la frequenza di filtro in hertz si puo' arrivare al valore corretto tramite:

(frequenza filtro) = ((Hert)-30)/5.82

Questa formula, di carattere insolito, da i valori esatti da inserire tramite POKEs nei due registri:

```
POKE 54294, INT((freq.filtro)/8) :
POKE 54293, (freq.filtro) AND 7
```

Esempio:

Questo programma filtra a 460 HERTZ. La frequenza del tono, percorre l'intero spettro.

```
10 POKE 54278,240 : POKE 54276,33
20 POKE 54294,64 : POKE 54295,1 :POKE 54296,31
30 FOR X=0 TO 255 : POKE 54273,X :NEXT
40 POKE 54276,32
```

Questo esempio e' esattamente l'opposto: la frequenza del tono si trova a circa 300 Hertz e la frequenza del filtro ripercorre tutto lo spettro:

```
10 POKE 54278,240 : POKE 54276,33
20 POKE 54273,20 : POKE 54295,1 :POKE 54296,79
30 FOR X=0 TO 255 : POKE 54294,X : NEXT
40 POKE 54276,32
```

LOCAZIONE 54295

Registro di controllo del filtro.

Contenuto:

In questo registro sono determinate le 2 piu' importanti funzioni del filtro: qui possono essere scelti quali delle possibili fonti del segnale devono essere filtrate.
Le possibili fonti del segnale sono le 3 voci ed i segnali esterni.
Voi potete inoltre selezionare in questa locazione l'eco del filtro.
Questo eco accentua i componenti della frequenza del filtro attraverso la quale operazione risulta un suono quasi piu' marcato.

Compito del Bit:

0	Voce 1 filtrata
1	Voce 2 filtrata
2	Voce 3 filtrata
3	Segnale esterno filtrato
4-7	Risonanza del filtro

Spiegazione:

Se il Bit #0 e' inserito, viene filtrata la voce 1.
Il Bit #1 decide se la voce 2 viene filtrata.
Il Bit #2 e' fisso per il filtro della voce 3.
Il Bit #3 e' fisso: segnali esterni vengono filtrati.
Quest'ultima entrata e' mixata con le altre voci e depositata di nuovo all'uscita. Se voi inserite un segnale esterno, dovete fare attenzione a questo. Il segnale e' memorizzato sopra il Pin 5 della scatola audio/video. L'impedenza dell'entrata e' dell'ordine di grandezza di 100 KOhm. Il voltaggio deve essere di circa 6 Volt.

Riferimento:

Voi potete condurre ulteriori voci all'azione di filtro.
I Bit #0 - #3 possono essere accesi in una qualsiasi combinazione.

Nei Bits #4 - #7 viene stabilita la risonanza del filtro da un valore tra 0 e 15.

Con la seguente formula potete calcolare le PoKes del valore per questa locazione:

```
POKE 54295, (risonanza)*16 + (voce 1) + 2↑(voce 2) + 4↑(voce
3) + 8↑(esterno)
```

Percio' viene inserito per il valore 'Voce...' e 'esterno' un 1, se ha luogo il filtro.

Esempio:

Questo programma mostra l'uso della risonanza del filtro. Lo stesso tono viene prodotto alternativamente ad una risonanza piena e non piena.

```
10 POKE 54278,240 : POKE 54276,33
20 POKE 54294,64 : POKE 54295,1+16*15*A :POKE 54296,31
30 FOR X=0 TO 255 : POKE 54273,X : NEXT
40 POKE 54276,32
50 FOR Y=0 TO 500 : NEXT : A=1-A : GOTO10
```

LOCAZIONE 54296

Selezione dei filtri e volume

Contenuto:

Questo registro e' molto importante, poiche' qui il volume globale del tono prodotto viene deciso.
Questo volume deve essere maggiore di 0, affinche' i toni possano essere uditi.
Il volume qui deposto e' usato dopo il ciclo d'attacco (confronta 54277/54284/54291) per raggiungere il valore massimo.

Questo indica che tanto piu' il valore qui stabilito e' alto, tanto piu' il tono sara' alto anche durante la fase Sustain.
Anche l'uso del filtro viene scelto in questo registro.
Potete selezionare fra Low-Pass (l'ambito della frequenza al di sotto della quale non c'e' azione di filtro) e High-Pass (l'ambito della frequenza massima al di sopra della quale non c'e' azione di filtro).

Questo uso dei filtri puo' essere anche combinato.
Inoltre c'e' anche la possibilita' di rendere la voce muta in alcuni casi (modulazione circolare, sincronizzazione: locazioni 54276/54283/54290).

Compito dei Bits:

0-3	Tono alto di base
4	Filtro Low-Pass
5	Filtro Band-Pass
6	Filtro High-Pass
7	Voce 3 accesa(0)/spenta(1)

Spiegazione:

Nel Bit 0-3 viene stabilito il volume di base. Esso deve essere maggiore dello 0 affinche' un tono possa essere ascoltato.
Questa decisione e' una costante del volume per tutte e 3 le voci.

Il Bit #4 attiva il filtro Low-Pass.
L'ambito al di sotto della quale frequenza non e' presente azione di filtro.

Il Bit #6 attiva il filtro High-Pass le frequenze sopra tale valore non vengono filtrate.

Riferimento:

Se il Bit #4 e il Bit #6 sono usati contemporaneamente, l' High-Pass e Low-Pass possono decidere la gamma dello spettro di frequenze da udire o filtrare.

Un inserimento del Bit #7 rende muta la voce 3.

Esempio:

Questo programma usa un tono e ne diminuisce il volume. Come effetto simula una campana che suona.

```
10 POKE 54273,30 : POKE 54278,240 : POKE 54276,33
20 FOR X=15 TO 0 STEP -.15 : POKE 54296,X : NEXT
30 GOTO 20
```

L'intero spettro di frequenza scorre attraverso il filtro. In particolare usiamo la forma d'onda RUMORE. Schiacciando un tasto potete cambiare i filtri fra di loro.

```
10 POKE 54278,240 : POKE 54276,129
20 POKE 54294,42  : POKE 54295,1
30 POKE 198,0 : WAIT 198,1
40 A=A+1 : PRINT CHR$(147)
50 IF A=8 THEN A=0
60 IF A AND 1 THEN PRINT 'LOW PASS'
70 IF A AND 2 THEN PRINT 'BAND PASS'
80 IF A AND 4 THEN PRINT 'HIGH PASS'
90 POKE 54296, 15+16*A
100 FOR X=0 TO 255 : POKE 54273,X : NEXT
110 GOTO 30
```

Infine ancora un piccolo programma: un tono sulla voce 3, schiacciando un tasto, viene alternativamente acceso e spento.

```
10 POKE 54292,240 : POKE 54287,30
20 POKE 54290,33 : A=1
30 POKE 198,0 : WAIT 198,1 : A=1-A
40 POKE 54296,15 + A*128
50 GOTO 30
```

LOCAZIONE 54297

A/D trasformatore 1

Valore normale:
255 nel caso di mancanza di collegamenti.

Contenuto:

Nel SID sono costruiti due trasformatori analogico-digitali. Voi potete trasformare un flusso di parole in un valore digitale. Questo trasformatore analogico-digitale puo' essere usato per la lettura delle Paddles.
Questa locazione e' per la paddle #1.

Compito dei Bits: -

Spiegazione:

Con PEEK (54297) potete leggere la momentanea posizione di una o dell'altra resistenza regolabile.
Una resistenza collegata dovrebbe avere valori maggiori di 470 kOhm.

Riferimento:

Molti Paddles del mercato consueto non raggiungono mai il valore massimo 255 poiche' la resistenza non e' abbastanza grande.

Esempio:

C'e' anche la possibilita' di usare piu' di 2 Paddles. Voi potete poi leggere informazioni anche dalla locazione 56320: qui non troverete pero' i valori dei tasti premuti. Il seguente programma in linguaggio macchina vi permette di leggere tutte e 4 le fonti di informazione:

E' necessario digitare SYS 828 (o un'altra locazione d'inizio) e potete leggere nelle seguenti locazioni il valore delle Paddles:

820 Porta 1 Paddle 1

821 Porta 1 Paddle 2

822 Porta 1 joyst.

823 Porta 2 Paddle 1

824 Porta 2 Paddle 2

825 Porta 2 joy

Tutto il resto lo trovate presso la locazione 56320 e 56321

```
10 A=828 : FOR X=A TO A+62 : READ W : S=S+W :POKE X,W : NEXT
20 IF S<> 6792 THEN PRINT 'ERRORE NEI DATA!' :STOP
30 DATA 120,173,1,220,41,12,141,54,3,173,0,
   220,41,12,141,57,3,162,128
40 DATA 142,0,220,160,0,136,208,253,174,25,
   212,142,55,3,174,26,212,142
50 DATA 56,3,162,64,142,0,220,160,0,136,208,
   253,174,25,212,142,52,3
60 DATA 174,26,212,142,53,3,88,96
```

LOCAZIONE 54298

A/D trasformatore 2

Valore normale: 255 nel caso in cui nessuna Paddle sia stata inserita.

Contenuto:

In questa locazione possono essere letti i valori del trasformatore A/D numero2.
Questa locazione e' fissa per il Paddle 2. Ulteriori informazioni le troverete presso 54297.

LOCAZIONE 54299

Voce 3: oscillatore

Valore normale:

Contenuto:

Il contenuto di questo registro dipende dal momentaneo stato della forma d'onda della voce 3.
Viene modificato lo stato digitale di questa forma d'onda.
Voi potete utilizzare questo valore cosi': mediante il trasferimento di questo registro in un'altro, per esempio la frequenza, il filtro di frequenza, o la larghezza interna: voi potrete di seguito raggiungere vari effetti.
Se avete scelto la forma d'onda 'rumore', questo registro illustra perfettamente le oscillazioni.

Compito dei Bits: -

Spiegazione:

Se avete scelto la forma d'onda 'triangolo' nella locazione 54290, il contenuto di questo registro varia dallo 0 fino a 255 e riconta all'indietro fino al valore 0.
La velocita' di questo conteggio e' determinata dalla frequenza in (54287/54288). Piu' e' alta, e piu' veloce e' il conteggio.

Se voi volete leggere questo valore da BASIC, dovete digitare POKE 54286,1 che corrisponde ad una frequenza di circa 0.05 Hertz.
Presso la forma d'onda 'dente di sega' il valore della locazione sale fino a 255, poi retrocede subito a 0 e ricomincia dall'inizio.
La velocita' dipende di nuovo dalla frequenza scelta. Presso la forma d'onda 'rettangolo' il valore in questo registro salta da 0 a 255, dipendentemente dalla larghezza interna e dalla frequenza scelta. La larghezza interna indica la durata di 255.
Presso la forma d'onda 'rumore' in questo registro il valore muta casualmente fra 0 e 255: voi potete usarlo per la produzione di particolari effetti risuonanti.

Riferimento:

Col Bit #7 nella locazione 54297 potete regolare la voce 3 muta.

Esempio:

Questo programma usa la forma d'onda 'triangolo', per la voce 1: l'effetto e' una sorta di sirena:

```
 10 POKE 54296,143 : POKE 54278,240 : POKE 54276,33
 20 POKE 54286,25 : POKE 54290,17
 30 POKE 54273, PEEK(54299) : GOTO 30
```

Sostituite il valore 17 nella riga 20 con altre forme d'onda (33,65,129) e ascoltate cosa succede.
Se volete usare il linguaggio macchina:

```
 10 A=828 : FOR X=A TO A+33 : READ W : S=S+W : POKE X,W :
NEXT
 20 IF S <> 4417 THEN PRINT "ERRORE NEI DATI!" : STOP
 30 SYS A
 40 DATA 169,240,141,6,212,169,15,141,24,212,169,
    33,141,4,212,169,255,141
 50 DATA 11,212,169,17,141,18,212,173,27,212,141,
    1,212,24,144,247
```

Se mutate il valore 255 nella riga 40, la somma esaminata nella riga 20 deve essere naturalmente adattata.

LOCAZIONE 54230

voce 3: curva d'inviluppo, generatore.

Valore normale: -

Contenuto:

In questo registro si puo' leggere l'attuale stato in forma digitale, della voce 3.
Voi potete raggiungere attraverso l'utilizzo di questo registro, nuovi effetti acustici.

Compito dei Bits: -

Spiegazione:

Se la curva d'inviluppo della voce 3 e' nella fase iniziale (ciclo-d'attacco), allora il contenuto di questo registro aumenta fino a 255.
Poi questo valore resta stazionario per un breve tempo e poi retrocede al Sustain-Level.
Successivamente dopo l'inizio del ciclo-Release ritorna di nuovo sullo 0 dove la curva di inviluppo si spegne.

Riferimento:

Per poter leggere questo registro, deve essere programmata nella locazione 54231 e 54232 per la voce 3, una curva d'inviluppo che viene memorizzata nella locazione 54290 attraverso il gate-bit.

Riferimento:

Con il Bit #7 nella locazione 54296 la voce 3 puo' essere regolata come 'muta', utile questo per interessanti effetti acustici.

Esempio:

Questo programma mostra acusticamente lo svolgimento di una curva d'inviluppo con i valori Attack=12, Decay=12, Sustain=2, Release=12

```
10 POKE 54296,15 : POKE 54278,240 : POKE 54276,33
20 POKE 54291,204 : POKE 54292,44
30 POKE 54290,17
40 FOR X=1 TO 300 : POKE 54273, PEEK(54300) : NEXT
50 POKE 54290,16
60 FOR X=1 TO 200 : POKE 54273, PEEK(54300) : NEXT
70 POKE 54276,32
```

.....

LOCAZIONI 54336-55260

'Fata-Morgana' del SID

Contenuto:

Di nuovo accade per il SID quello che avveniva per il VIC: i registri del SID li trovate altre 15 volte nella memoria, ma c'e' solo un solo SID!
Il Commodore si comporta proprio generosamente.

Tutti i 34 Bytes si trovano in un nuovo blocco col registro-SID.
In effetti POKE 54296+15*64,15 produce la stessa cosa di poke 54296,15.

Esempio:

```
10 POKE 55128,15:POKE 55110,240 : POKE 55105,20 :POKE
55108,33
```

.....

LOCAZIONI 55296-56295

Colore-RAM

Valore normale: -

Contenuto:

In questo ambito della memoria, vengono deposti i codici dei colori del testo che appare sul video.
Al contrario della mappa dello schermo, questo ambito non e' mobile (Confronta locazioni 1024-2023).

Compito dei Bits: -

Spiegazione:

Per la memorizzazione dei colori vengono usati solo i 4 Bits minori.
I 4 Bits maggiori non vengono sfruttati. Il codice colore usato lo trovate alla locazione 53270.
Se voi non scrivete su video con PRINT ma con delle POKEs, (confronta loc. 1024-2023), dovete preoccuparvi che le informazioni sui colori arrivino nel modo giusto in questa memoria.
Potete ad esempio riempire la mappa colore con il desiderato codice.

Riferimento:

Per i vecchi computers Commodore dovete usare la POKE con il valore desiderato nella locazione 53281 (sfondo video) e pulire il video con PRINT CHR$(147).

Per la nuova versione invece se si mette il codice-colore desiderato nella locazione 53281 e poi si pulisce il video, non otterrete nessun effetto.

Occorre usare le POKEs del colore per ogni carattere che abbiamo scritto sullo schermo, agendo nella mappa colore appunto.
A tale scopo potete usare la seguente formula:

POKE (locazione carattere presente su video) + 54272,
(codice-colore)

Attenzione: questa formula vale solo, se avete la RAM-video in 1024.
Negli altri casi dovete sostituire la cifra 54272 mediante:

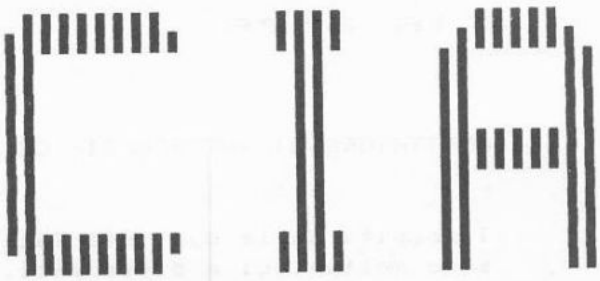
55296-(indirizzo partenza mappa video)

Inoltre nelle locazioni 243/244 della pagina zero trovate l'attuale posizione nella mappa colore, dell'ultimo carattere PRINTato sul video.

Esempio:

Questo programma scrive caratteri e colori sul video:

```
10 FOR X=0 TO 255
20 POKE 1024+X,X : POKE 55296+X,X
30 NEXT
```



ADATTATORE DI INTERFACCIA COMPLESSA

CIA 6526

ADATTATORE DI INTERFACCIA COMPLESSA.

I compiti delle due interfacce installate nel Commodore 64 sono molteplici e differenti.
Il c.i.a.1 si occupa in massima parte della gestione della tastiera.
Questo CHIP e' anche, metaforicamente parlando, un collegamento con tutti i tasti, registrando il comando (la pigiatura di un tasto), chiudendo l'apposito contatto.
Per mezzo di esso vengono analizzate le porte JOYSTICK, PADDLES, e la penna luminosa.
Inoltre il CIA1 e' responsabile della interruzione (IRQ) video ogni sessantesimo di secondo.
Accanto a queste molteplici funzioni, ve ne sono altre che non vengono sfruttate dal C64 ma che possono esserlo via softuare.
A questo genere di operazioni appartiene per esempio l'AM/PM ovvero la possibilita', se programmata, di avere in ogni momento l'ora esatta.
Passando ora al C.I.A.2 notiamo la sua principale caratteristica: gestire gli INPUTS e OUTPUTS.
Esso e' un sistema standard per la trasmissione dei dati a 8 bits sulle varie periferiche.

.....

LOCAZIONE 56320

CIA 1:porta a

Valore Normale:

127,nel caso che nessun joystick o paddles sia premuto.

Contenuto:

Questa e le seguenti locazioni appartengono al CIA1. Questa porta ha in se' moltissime funzioni, esaminiamone alcune.
E' possibile leggere in questa locazione il valore momentaneo del JOYSTICK 2 nelle sue posizioni complete o del PADDLE 2.
Questa porta,inoltre, interagisce con la tastiera (9+1=fuoco).

Riferimento:

Non lasciatevi trarre in inganno: la porta a corrisponde al controlport 2, porta b corrisponde al control port 1. Tutto questo suona strano ma corrisponde al reale funzionamento.

Compito dei BITS:

0 Joystick 2 in alto. (0)

1 Joystick 2 in basso. (0)

2 Joystick 2 a sinistra. (0)

Paddle 1 porta 2 fuoco.

3 Joystick 2 destra. (0)

Paddle 2 porta 2 fuoco. (0)

4 Joystick 2 fuoco. (0)

5 Non usato. (1)

6/7 Scelta Paddle (01)=porta 1

(10)=porta 2

Joystick:
Nella porta A puo' essere esaminato il Joystick che e' inserito nel controlport 2, percio' le 5 direzioni sono riportate in ogni bit di questa locazione.
Se viene chiuso solo il corrispondente contatto il valore di questo bit diventa 0.
Il contenuto normale di questa locazione e' 127 cioe' il bit 7 e' 0.
Se per esempio il Joystick viene spostato verso l'alto il valore che si ricava e' 126.
Con la formula :

IF (PEEK(56320) and 2↑(bit)) = 0 THEN BIT SPENTO.

potete ricavare la direzione del Joystick.

Riferimento:

Se il joystick viene spostato in una direzione diagonale vengono interessati 2 bits.
(Obliquo a sinistra produce: 122=127-2↑0-2↑2)
Il fuoco inoltre ha, in linea di massima, un valore inferiore di 16 a quelli normali.
Fuoco+obliquo+alto+sinistra=106=127-2↑0-2↑4

Paddles:
il valore delle paddles viene letto nella locazione 54297 e 54298.

Riferimento:

Se il bit 2 e' spento viene usato il bottone del fuoco del paddle 1, se il bit 3 e' spento quello del paddle 2. Naturalmente entrambi i tasti del fuoco possono essere pigiati contemporaneamente.
Nella locazione 54297 vi e' un programma di esempio che prende in considerazione questo problema.

Esempio:

Questo programma mette in movimento uno sprite sullo sfondo video per mezzo del joystick. (Vedere 53248,53249...).

```
10 X=128:Y=128:POKE53269,1:POKE2040,13
20 A=PEEK(53260)
30 IF (A AND 1)=0 THEN Y=Y-1
40 IF (A AND 2)=0 THEN Y=Y+1
50 IF (A AND 4)=0 THEN X=X-1
60 IF (A AND 8)=0 THEN X=X+1
70 POKE53248,X:POKE53249,Y
80 GOTO 20
```

LOCAZIONE 56321

CIA 1: PORTA B

Valore Normale:

255, nel caso non sia premuto nessun joystick o paddles.

Contenuto:

C'e' solo la porta B del cia1.
In questa locazione puo' essere letto il Joystick 1 e il tasto del fuoco del Paddle.

Compito dei BIT:

0 Joystick in alto (0)

1 '' in basso (0)

2 '' a sinistra (0)

Paddle 1 porta 1 fuoco

3 Joystick a destra

Paddle 2 porta 1 fuoco

4 Joystick 1 fuoco

5 Non usato (1)

6/7 Timer

Spiegazione:

Richiesta dalla tastiera: il CIA1 e' in collegamento con l'I/O e la tastiera. Si puo' esaminare sul sistema di

funzionamento delle porte A e B quale tasto e' premuto.
Presso la matrice della tastiera ci sono 8 righe e 8 scissioni. La disposizione funziona nel seguente modo:

RIGHE (56321):

0	254	<1>	<£>	<+>	<9>	<7>	<5>	<3>	<DEL>
1	253	< >	<*>	<P>	<I>	<Y>	<R>	<W>	<RET>
2	251	<CTRL>	<;>	<L>	<J>	<G>	<D>	<A>	<CSRS L/R>
3	247	<2>	<CLR>	<->	<0>	<8>	<6>	<4>	<F7>
4	239	<SPACE>	<SHIFT>	<.>	<M>	<B>	<C>	<Z>	<F1>
5	223	<C=>	<=>	<:>	<K>	<H>	<F>	<S>	<F3>
6	191	<Q>	< >	<@>	<O>	<U>	<T>	<E>	<F5>
7	127	<RUN/STOP>	</>	<,>	<N>	<V>	<X>	<SHIFT>	<CRSR U/L>
56320:		254	253	251	247	239	223	191	127
SCISS.:		7	6	5	4	3	2	1	0

Riferimento:

Il tasto restore e' in collegamento diretto col processore.
SHIFT LOCK e' identico allo shift normale.
SHIFT a destra e a sinistra hanno la stessa funzione ma si distinguono tra loro nel rilevamento della tastiera.

L'esame della matrice della tastiera si svolge nel seguente modo:
Nella porta a viene scelta una scissione e il bit conseguente viene impostato a zero. Vengono poi prodotti i valori della tabella 56320.
Se vengono premuti piu' di un tasto contemporaneamente in una scissione vengono spenti piu' BITS.
ES: SHIFT a sinistra = 126.
Il vantaggio di un'unica routine e' che voi potete esaminare ogni combinazione di tasti.

JoistickK: nella porta B puo' essere esaminato il tasto di fuoco dei PADDLES collegati al controlport 1.
Questi Paddles vengono normalmente trasmessi anche dal trasformatore A/B nella locazione 54297 e 54298.
Se il bit 2 e' spento significa che e' premuto il tasto fuoco del paddle 1 al controlport 1.
Se il bit 3 e' spento e' premuto il fuoco del paddle 2.

LOCAZIONE 56322

CIA1 : registro direzione dati PORTA A

Valore normale : 255.

Contenuto:

Questo registro controlla i bits in uscita e in entrata. Percio' vale:
se il bit e' usato in questa locazione e' definito nella porta A come uscita o anche:

0 = entrata 1 = uscita

Spiegazione:

normalmente non c'e' bisogno del registro direzione dati poiche' viene attivato dalle routines del sistema.
Se selezionate un registro di lettura ricevete il valore attuale di ogni bit inndipendentemente dal fatto che esso sia definito come entrata o uscita.

.....

LOCAZIONE 56323

CIA 1: registro direzione dati PORTA B

Valore Normale : 0

Contenuto : Vedi 56322

.....

LOCAZIONI 56324-56325

CIA 1 : Timer A

Valore Normale: 52

Contenuto:

Il CIA ha 2 timer a 16 bits che alla base non hanno nient'altro che contare all'indietro fino a 0 e poi darvi un segnale.
Il timer A viene usato per produrre il segnale di interruzione che deve comparire ogni sessantesimo di secondo.

Poiche' un ciclo passa ogni sessantesimo di secondo il BASIC non e' efficace per gestire questa locazione.

Spiegazione:

Un intervento di lettura produce il valore momentaneo del timer.

Riferimento:

Mediante la scrittura di un valore nel byte alto e' anche possibile stabilire quante interruzioni hanno luogo.
Il valore normale e' nei dintorni di 52 (nel byte alto, il byte basso lavora troppo velocemente).
Valori al di sotto di questa cifra accellerano le interruzioni, al di sopra le rallentano.
L'accellerazione delle interruzioni non ha nulla a che fare con l'accellerazione del lavoro del computer.
In generale non va toccata questa locazione di memoria da Basic.

Riferimento:

Tutte le manipolazioni del timer cambiano il valore di TI$.
Le funzioni del timer vengono memorizzate nella locazione 56334.

Sperimentate come si comporta il vostro calcolatore con:

+ interruzioni : poKe 56325,5

- interruzioni : poKe 56325,200:

.....

LOCAZIONI 56326-56327

CIA 1 : TIMER B

Valore Normale:

Contenuto:

Questo Timer funziona come il timer A.
Viene usato pero' solo per le operazioni sul registratore del C64.

.....

LOCAZIONI 56328-56331

CIA 1 : Orologio ora esatta.

Valore Normale: e' un rilevamento inutile.

Contenuto:

Nel CIA si trova un orologio esatto (1/10 di secondo). Questo orologio funziona con la frequenza NET e percio' e' piu' esatto di TI$. (Vedi 160-162)

Spiegazione:

Questo registro lavora nel formato BCD e quindi non accetta piu' di 4 BITS, da 0 fino a 15.
Questo sistema ha il vantaggio di una lettura piu' veloce.
L'orologio lavora solo nel metodo 12 ore e percio' esiste un flag che indica il PM o l'AM. (1=PM)

Riferimento:

0.00 e' per l'orologio 12.00 AM.

Come si regola questo orologio?
Prima di tutto dovete cambiare l'ora attuale nel formato BCD con la seguente formula:

```
(Dec.-cifra)=INT((BCD-cifra)/16)*10+((BCD-cifra) AND 15)
```

Vi e' un registro per i decimi di secondo, uno per i secondi, i minuti e le ore.L'orologio lavora solo nel modo '12 ore', il Bit #7 nella locazione ore 56331, e' diviso in un flag AM (mattino) e uno PM (pomeriggio). Se il Bit e' acceso, e' pomeriggio.

Riferimento:

Le 0.00 per un orologio (mezzanotte) sono per l'orologio in tempo reale le 12.00 AM. Voi dovreste inserire nel vostro programma una routine, che attivi l'orologio correttamente.

Per un corretto funzionamento dell'orologio e' utile comunicare al CIA la rete di frequenza. Vedi locazione 56334.

Come si regola l'orologio?
Prima di tutto dovete cambiare il valore per le ore, minuti e secondi. Incominciate con adattare le ore. Non appena si modifica il registro delle ore (56331), si blocca automaticamente l'orologio. Esso si arresta, fino a quando non si modifica il registro dei decimi di secondo. Cosi' potete regolare tutti i registri dell'orologio senza che esso continui a modificarsi. Usate le POKE anche per le ore i minuti e i secondi. Per le lettere dell'orologio cominciate

col registro delle ore. Non appena ha luogo una lettura del registro delle ore, viene memorizzato il tempo attuale l'orologio continua a funzionare normalmente ma la locazione 56328-56331 non varia di contenuto.

Riferimento:

Se deve essere letto solo il registro delle ore, deve essere letto come proforma il registro dei decimi di secondo, affinche' venga indicato ancora il tempo esatto. La lettura dei minuti e secondi da soli non interrompe il segnale.

Voi potete (vedi locazione 56335) inserire anche un'allarme.

Esempio:

Questo programma vi aiuta ad impostare l'orologio.

```
 10 INPUT 'TEMPO (HHMMSS);T$
 20 H$ = LEFT$(T$,2)
 30 M$ = MID$(T$,3,2)
 40 S$ = RIGHT$(T$,2)
 50 A = VAL(S$) : GOSUB 100 : S = BC
 60 A = VAL(M$) : GOSUB 100 : M = BC
 70 A = VAL(H$) : IF A>11 THEN PM = 128 : A=A-12
 80 GOSUB 100 : H = BC OR PM
 90 POKE 56334,129 : POKE 56331,H : POKE 56330,M :
    POKE 56329,S : POKE 56328,0 : END
100 BC = INT (A/10) * 16 + (A - INT(A/10)*10)
110 RETURN
```

E con questo programma potete leggere le ore i minuti e i secondi.

```
 10 H = PEEK(56331)AND127
 20 M = PEEK(56330)
 30 S = PEEK(56329)
 40 A = S : GOSUB 100 : S$ = STR$(A)
 50 A = M : GOSUB 100 : M$ = STR$(A)
 60 A = H : GOSUB 100 : H$ = STR$(A)
 70 IF PEEK(56331) AND 128 THEN A = A+12 : H$= STR$(A)
 80 PRINT H$'.'M$':'S$
 90 Z=PEEK(56328): END
100 D = INT (A/16)*10 + (A AND 15) : A=D
110 RETURN
```

LOCAZIONE 56332

CIA 1: serie I/O buffer

Valore normale: 0

Contenuto:

In questo Byte il CIA memorizza 8 Bits, che poi vengono uilizzati dalla porta seriale.

Normalmente non ha nessun senso utilizzare tale locazione di memoria da basic.
La Porta del CIA 1 e' il Pin #5 della USER-PORT.
La porta in 56334 puo' essere programmata in input od output.

LOCAZIONE 56333

CIA 1: registro di controllo interruzioni

Valore normale: -

Contenuto:

Questo registro di controllo gioca per il CIA 1 un ruolo importante:
nel caso che si presenti una interruzione, qui si puo' leggere, di che tipo di interruzione si tratta, e come e' stata provocata.
Mediante la lettura in questa locazione puo' essere testato quale fattore abbia richiesto una interruzzione.
I possibili casi sono:
Timer A raggiunge il valore 0,
Timer B raggiunge lo 0,
l'orologio del tempo esatto ha raggiunto il tempo d'allarme,...

Contenuto dei Bits:

0	Timer A trascorso
1	Timer B trascorso
2	Allarme orologio ora esatta
3	buffer pieno/vuoto
4	Lato negativo sul 'FLAG'
5-6	Non usato
7	SCRITTURA: (1) lasciare interruz. (0) bloccarle LETTURA: ha luogo inter. autorizzata.

Spiegazione:

Se ha luogo un'interruzione mediante una delle possibili fonti autorizzate, il Bit #7 viene acceso.

Riferimento:

Se voi leggete questo registro, vengono spenti tutti i Bits.
Se voi scrivete un valore in questa locazione, il Bit #7 decide cosa accade.
Gli altri Bit decidono, quale causa (quali cause) sono le generatrici.

Se il Bit #7 e' acceso viene permessa la rispettiva interruzione.
Se il Bit #7 e' spento, viene bloccata la rispettiva interruzione.

Un esempio:

la 'normale' interruzione viene prodotta attraverso il Timer A.
Se volete disinserire l'interruzione (vedi anche locazione 788/789), dovete scrivere un valore in questo registro, nel Bit #0.
Con POKE 56333,1 si bloccano le interruzioni.
Se volete permettere di nuovo questa interruzione, agite sul Bit #7:

POKE 56333,129

rende possibili le interruzioni.

Il Bit #0 decide, come gia' detto, del Timer A (56324/56325).

Il Bit #1 e' fisso per il Timer B. Questo Timer pero' non viene normalmente usato (locazione 56326/56327).

Il Bit #3 permette un'interruzione se il registro 5 e' attivo. Le condizioni che causano l'interruzione dipendono dal tipo di funzionamento di questa serie di Porte. Per questo decide il Bit #6 nella locazione 56334.

Il Bit #4 genera un'interruzione, se sussiste un segnale negativo sul Pin Flag del CIA.
Cosi' e' possibile lanciare l'interruzione anche attraverso un contatore esterno.

Esempio:

Questo piccolo programma accende il Timer A e rende il Timer B responsabile di una interruzione:

```
10 POKE 56327,52
20 POKE 56334,0
30 POKE 56333,130
40 POKE 56335,1
```

LOCAZIONE 56334

CIA 1: registro di controllo A

Valore normale: 1

Contenuto:

In questo registro sono riunite piu' funzioni di controllo del CIA.
Agendo sul bit 1, si puo' attivare o spegnere il timer A.
Cosi' c'e' la possibilita', di sospendere un'interruzione.
Il segnale del Timer A e' sul Bit# 6 della Porta B (locazione 56321).
E' anche possibile stabilire, se il segnale emesso ha la forma d'onda tipo 'rettangolo'.

Puo' anche essere stabilito se il Timer A solo una volta o sempre riconta fino allo 0.
Un'ultima funzione e' particolarmente significativa: qui viene messo a punto, se l'orologio lavora con una frequenza di 50 o 60 Hertz.

Compito dei Bits:

```
0    Timer A Start(1) / Stop(0)
1    Timer A Dal Bit 6 Porta B
     (1=si , 0=no)
2    Timer A 0 passivo
            1 impulso
3    Timer A monostabile(0)/
            continuo   (1)
4    Timer A : caricamento
            (1=ja , 0=nein)
5    Timer A :
     segnale di sistema (0)
     segnale CNT(1)
6    Modo per Porta seriale
     1=input
     0=output
7    frequenza:
     0=60 Hz ; 1=50 Hz
```

Spiegazione:

Il Bit #0 decide se il Timer A viene fermato o se continua a contare.
Cosi' potete spegnere le interruzioni con POKE 56334,0 o lasciarle con POKE 56334,1.
Il Bit #1 ed il Bit #2 sono inutili per il C-64: qui puo' essere scelto se il segnale del Timer viene emesso dal 6 Bit della Porta B e in quale forma (rettangolo, impulso del flusso).
Poiche' pero' la Porta B viene usata dal C-64 per la scansione della tastiera, dovete lasciare questo Bit sempre spento.
Il bit #3 deve essere lasciato allo stato attuale per non compromettere il regolare funzionamento della macchina.

Il Bit #4 offre la possibilita', di trasmettere al Timer A il valore memorizzato insieme nel Timer. Poiche' comunque queste cifre per il BASIC vanno troppo svelte, rimane questa funzione senza uso pratico.

Il Bit #5 e' usato per la scelta della frequenza di lavoro del sistema (circa 1 MHz).

Il Bit #6 stabilisce il tipo di movimento della serie delle Porte nella locazione 56332.
Se il Bit e' spento, dispone la porta dati in INPUT, se il Bit e' inserito la porta dati e' in output.
Anche questa Porta non ha normalmente per il C-64 nessuna funzione.

Il Bit #7 e' importante per il funzionamento dell'orologio dell'ora esatta (vedi 56328-56331).
Esso stabilisce, se questo orologio lavora con una frequenza di 50 Hz (Bit acceso) o 60 Hz (Bit spento).
Il valore normale di questi Bits e' 0 poiche' in America la rete di frequenza della corrente alternata e' pari a 60 Hz.
In ITALIA questo valore deve essere posto sempre sull'1, se deve funzionare correttamente:

POKE 56334,129.

Esempio:

Questa POKE vi mostra cosa succede, agendo sulla tastiera:

POKE 56334,7

LOCAZIONE 56335

cia 1: registro di controllo B

Valore normale: 8

Contenuto:

Questo secondo registro di controllo-CIA e' relativo al Timer B.
Il Timer B non e' solitamente utilizzabile da un normale programma Basic.
Viene usato solamente per la scansione del tempo durante le operazioni del registratore.

Compito dei Bits:

0	Come locazione 56334, per timer B
1	''
2	''
3	''
4	''
5-6	selezione modo timer B: 00 = cont. puls. sistema 01 = conteggio segnali CNT 10 = Underflow Timer A 11 = Underflow Timer A con segnale CNT-Signal
7	allarme (1) orologio (0)

Spiegazione:

I Bits #0-#4 sono identici alla corrispondente funzione del Timer A nella locazione 56334.

Il Bit #7 funziona come segue:
se questo Bit e' inserito, il tempo viene memorizzato e usato come tempo d'allarme. Se il tempo attuale e' uguale al tempo d'allarme, viene inserito ad 1 il Bit #2 nella locazione 56333.
Nel caso sia permessa, viene generata anche una interruzione.

Riferimento:

Il tempo d'allarme puo' essere solo scritto; la lettura non e' possibile.

Esempio:

Questo programma stabilisce un tempo d'allarme. Esso e' identico fino alla riga 85 con il programma per fissare l'ora alle locazioni 56328-56331.

```
10 INPUT 'ora (HHMMSS)';T$
20 H$ = LEFT$(T$,2)
30 M$ = MID$(T$,3,2)
40 S$ = RIGHT$(T$,2)
50 A = VAL(S$) : GOSUB 100 : S = BC
60 A = VAL(M$) : GOSUB 100 : M = BC
70 A = VAL(H$) : IF A>11 THEN PM = 128 : A = A-12
80 GOSUB 100 : H = BC OR PM
85 POKE 56335,136
90 POKE 56334,129 : POKE 56331,H : POKE 56330,M :POKE 56329,S
: POKE 56328,0 : END
100 BC = INT (A/10?) * 16 + (A - INT(A/10)*10)
110 RETURN
```

Ecco un ulteriore esempio applicativo di quanto detto fino ad ora.

```
10 POKE 56333,133
20 a = 828 : FOR X=A TO A+14
30 READ W : S = S+W : POKE X,W
40 NEXT X
50 IF S<> 1600 THEN PRINT 'ERRORE NEI DATA!!!' : STOP
60 POKE 788,60 : POKE 789,3
70 DATA 173,13,220,41,4,240,5,162,1,142,32,208,76,49,234
```

LOCAZIONI 56336-56575

Fata-morgana del CIA 1

Contenuto:

Anche per il CIA 1 ci sono le POKEs magiche !
La distanza fra le due identiche locazioni e' questa volta 16.
agendo sulle locazioni fantasma:

```
POKE 56334+15*64,0
```

ha lo stesso effetto che

```
POKE 56334,0.
```

Riferimento:

Sebbene esistano queste locazioni-Fata-Morgana esiste un solo CIA-Chip.

LOCAZIONE 56576

CIA 2: PORTA A

Valore normale: 151

Contenuto:

La Porta A del CIA 2 ha 2 compiti principali: uno per la direzione dati del VIC.
Questo e' anche il compito che rende interessante questa locazione per il programmatore-BASIC.
L'altro e' che questa porta memorizza il BUS seriale.

Compito del Bit:		
	0-1	Scelta locazioni VIC
		11 loc. da 0 a 16383 (Banco 0)
		10 loc. da 16384 a 32767 (Banco 1)
		01 loc. da 32768 a 49151 (Banco 2)
		00 loc. da 49152 a 65535 (Banco 3)
	2	Uscita dati per RS-232 (Userport PIN #L)
	3	Bus seriale: segnale ATN (Userp. PIN#9:Bus ser.PIN#3)
	4	Segnale CLOCK-OUT (Bus ser. PIN #4)
	5	Uscita SERIALE (Bus ser. PIN #5)
	6	Segnale di CLOCK-IN (Bus ser. PIN #4)
	7	Entrata SERIALE (Bus ser. PIN #5)

Contenuto:

I Bit #0 e #1 sono fissi per la ripartizione di memoria del VIC. Il VIC puo' indirizzare 16K. Cosi' potete usare normalmente l'ambito delle locazioni da 0 fino a 16383.

Riferimento:

All'interno di questo ambito devono esserci tutti i dati piu' importanti per il VIC. (Essi sono: schermo RAM, set di caratteri, indicatore Sprite, etc....)

Voi potete scegliere mediante l'azione su questi Bits anche gli altri 3 ambiti 16K per il VIC.

Dovete fare attenzione a questo:

Il banco di memoria 3 (l'ambito va da 49152 a 65535) e il banco della memoria 2 (da 32768 a 49151) sono coperti da ROM. Per il video RAM avete bisogno di un ambito libero RAM (che puo' essere posto nei 4K da 49152 a 53247).
L'ambito ROM si adatta meglio al video Hires, sprites e set di caratteri.

Riferimento:

La posizione del video RAM viene comunicata al sistema di funzionamento della locazione 648.
L'inizio del set di caratteri e' deposto nella locazione 53248 (confronta locazione 1). Per il VIC comincia dalla locazione 4096. Questo 'riflesso' funziona solo nei banchi 0 e 2. Nel primo e nel terzo dovete mettere a disposizione eventualmente,un set di caratteri da solo. Confrontate la locazione 53272.

Cosi' selezionate l'ambito di memoria:

```
POKE 56576,3 (banco di memoria)
```

Riferimento:

La posizione del colore RAM non e' variabile. Si trova sempre nell'ambito da 55296 a 56295.

Gli altri Bit di questa locazione sono fissi per una sola direzione del Bus seriale.

Esempio:

Queste POKEs scelgono come ambito VIC le locazioni da 32768 a 49151. Il video RAM muta la relativa locazione da 1024 sino a 33792.

```
10 POKE 56576,1 : POKE 648,132
20 PRINT CHR$(147)
```

LOCAZIONE 56577

CIA 2: Porta B

Valore normale: 255

Contenuto:

Questa porta ha solo una funzione in relazione con la RS-232, e non viene usata nella versione originale.

Compito del Bit:	Bit	Userport
	0	Pin #C
	1	Pin #D
	2	Pin #E
	3	Pin #F
	4	Pin #G
	5	Pin #H
	6	Pin #I
	7	Pin #K

LOCAZIONE 56578

CIA 2: registro direzione dati Porta A

Valore normale: 63

Contenuto:

La funzione di questo registro e' identica alla locazione 56322 del CIA 1.
Al fine di garantire un corretto funzionamento della macchina il contenuto di tale registro non dovrebbe essere alterato (confronta 56576).

LOCAZIONE: 56579

CIA 2: registro direzione dati Port B

Valore normale: 0

Contenuto:

Questo registro normalmente non e' usato (confronta 56577)

LOCAZIONI 56580-56581

CIA 2: Timer A

Valore normale: -

Contenuto:

Entrambi i Timers sono utili solo in collegamento con la RS-232.
Su questo tipo di funzione troverete chiarimenti alla locazione 56324/56325.

LOCAZIONI 56582-56583

CIA 2: Timer B

Valore normale: -

Contenuto:

Per avere chiarimenti a proposito della presente locazione, leggere alle locazioni 56582 e 56583.

LOCAZIONI 56584-56587

CIA 2: orologio

Valore normale: -

Contenuto:

Nel CIA 2 si trova un secondo orologio che ha lo stesso funzionamento di quello nella locazione (56328-56331).
Cosi' potete lasciar scorrere due orologi insieme.
Nella locazione 56590 viene riposta la frequenza in Hertz che segna il tempo.
Ulteriori informazioni le trovate alle locazioni 56328 e 56331.

LOCAZIONE 56588

CIA 2: serie I/O-buffer

Valore normale: 0

Contenuto:

In questo byte vengono memorizzati i Data che devono essere inseriti nella porta seriale ('serie Port').
Questa locazione non ha nessun utilizzo pratico.
La 'serie Port' e' posta al Pin #7 della User-port.

LOCAZIONE 56589

CIA 2: registro di controllo delle interruzioni

Valore normale: 0

Contenuto:

Questo registro di controllo e' adibito alla memorizzazione delle richieste di interruzione attraverso la RS 232.
Nei casi normali non viene usato.
Il modo di funzionamento di questo registro e' scritto alla locazione 56333.

Compito dei Bits:

0	timer A : registro interruzione.
1	timer B : registro interruzione.
2	allarme
3	porta seriale attivo/passivo.
4	flag indicatore NMI da RS-232.
5-6	non utilizzati
7	flag NMI (1) verificato (0) non verificato

LOCAZIONE 56590

CIA 2: registro di controllo A

Valore normale: 8

Contenuto:

Questo registro immagazzina la funzione del Timer A (locazione 56580 e 56581) e seleziona la frequenza di lavoro (locazioni 56584/56587).
In generale, il metodo di funzionamento di questo registro e' descritto alla locazione 56334.

Compito dei Bits:

0 timer A stop (0), attivo (1)

1 output su pin#6 timer A:
  0 = attivo
  1 = passivo

2 output su pin#6 timer A:
  0 = bistabile
  1 = impulso.

3 timer A:
  1=monostabile
  0=continuo

4 timer A:
  valore caricato: 1=si

5 timer A:
  1= segnale CNT
  0= clock di sistema

6 modo per posta seriale:
  1=input
  0=output

7 frequenza del clock:
  1=50 Hz
  0=60 Hz

LOCAZIONE 56591

CIA 2: registro di controllo B

Valore normale: 8

Contenuto:

Questo registro immagazzina la funzione del Timer B (56582/56583) e contiene la funzione d'allarme dell'orologio (locazione 56584 fino a 56587).
Il criterio generale di funzionamento lo trovate alla locazione 56335.

Compito dei Bits:

0 Vedere registro A di controllo, ora B

1 ''

2 ''

3 ''

4 ''

5-6 Modo del timer B:
00 = cont.clocK di sitema
01 = segnali CNT
10 = underflow timer A
11 = underflow timer A mentre CNT
e' positivo.

7 allarme (1)
orologio (0)

Riferimento:

Poiche' molte funzioni del CIA 2 nei casi normali non sono quasi mai usate, potete invece sfruttarle tramite l'Hardware, agente sulla User-port.
Descrivere nel dettaglio tecnico il tale argomento oltrepassa lo scopo del testo.
Affrontando un simile argomento valicheremmo i confini di un discorso orientato prettamente al software.

Ecco ancora il compito dei PINs della User-port elencati:

```
  sopra 1 2 3 4 5 6 7 8 9 10 11 12

        - - - - - - - - - - - -

  sotto A B C D E F G H I K  L  M
```

Compito dei Bits:

1	Massa	A	Massa
2	+5V	B	FLAG2
3	Reset	C	CIA #2 Port B Bit #0
4	CNT1	D	Bit #1
5	Ser.Port 1	E	Bit #2
6	CNT2	F	Bit #3
7	Ser.Port 2	G	Bit #4
8	PC-Signal	H	Bit #5
9	Ser.ATN	I	Bit #6
10	9 V	K	Bit #7
11	9 V	L	CIA #2 Port A Bit #8
12	Massa	M	Massa

TERMINI TECNICI

CODICE ASCII:

Inglese: codice di standard americano per lo scambio d'informazioni.
Questo codice assegna ad ogni lettera e ad ogni segno della memoria un numero fra 0 e 255.
Questi numeri possono essere usati col comando CHR$(argomento).

PRINT CHR$ (65) stampa per esempio la lettera 'A'.

Inoltre avete bisogno del codice ASCII, per impartire gli ordini nel buffer della tastiera (631-640).
In fondo al testo e' riportata la completa tabella dei codici ASCII.

INTERPRETE BASIC

l'interprete BASIC e' un programma scritto interamente in lunguaggio macchina, il cui compito e' di far eseguire al computer le istruzioni che l'utente digita seguendo lo standard BASIC.

FORMATO BCD

Inglese: binary coded digit.
In questo formato di rappresentazione numerica, si possono scrivere esclusivamente i numeri fra 0 fino a 99.
I 4 Bits maggiori (High Nibble) memorizzano le decine. I 4 Bits minori (Low Nibble) le unita'.

SISTEMA OPERATIVO.

Il sistema operativo e' di vitale importanza per il corretto funzionamento della normale attivita' della macchina.
Esso provvede alla scansione della tastiera, alla visualizzazione delle immagini, alla gestione delle periferiche, ecc...

CIFRE BINARIE

Questo sistema di conteggio e' basato sulle potenze dalla base 2.
Ogni numero e' espresso come somma di termini tutti potenze di due.

Avremo ad esempio:

```
0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9
1010 = 10
1011 = 11
1100 = 12
1101 = 13
1110 = 14
1111 = 15
```

CODICE VIDEO

Questo codice differisce dal codice ASCII.
Esso associa un numero tra 0 e 255 ad ogni tipo di carattere che possiamo visualizzare. (poke)
Per evidenziare ad esempio una lettera con una POKE sulla mappa video, dovremo consultare la tabella riportata alla fine del testo.

Il BIT-MAP

Il 'Bitmap' e' una zona RAM estesa per 8K, nel quale viene stabilita l'immagine di una grafica ad alta risoluzione.
Quando siamo in alta risoluzione, ad ogni bit acceso, del 'BITMAP', corrisponde un puntino luminoso sullo schermo.
Ulteriori precisazioni sono date a proposito del VIC.

FILE

Il termine voleva originariamente indicare una lista di informazioni, una fila di dati scritti.
Oggigiorno con tale termine si indica un insieme piu' o meno ordinato di informazioni, registrate su un supporto magnetico, permanente: un dischetto o anche una cassetta.

CIFRE ESADECIMALI

Questo sistema di numeri e' basato sulla cifra 16. Ad ogni combinazione di potenza di 16 viene assegnato un numero.
Questo sistema di numeri e' particolarmente importante per la programmazione in linguaggio macchina, poiche' in questo sistema la stampa di una grossa cifra ha bisogno di poco spazio.

Le cifre di questo sistema di numeri vanno da 0 a 9 e da A fino a F.

0 = 0	4 = 4	8 = 8	C = 12
1 = 1	5 = 5	9 = 9	D = 13
2 = 2	6 = 6	A = 10	E = 14
3 = 3	7 = 7	B = 11	F = 15

HIGH BYTE/LOW BYTE

Questo principio viene usato per scomporre in due byte i numeri tra 256 e 65535.
Essi, per essere memorizzati, devono essere rappresentati da due bytes, per un totale di 16 bits.
Il byte piu' significativo (il primo) e' chiamato byte alto (High Byte), poiche' esso rappresenta le cifre maggiori; il secondo byte e' detto 'basso' ovvero 'Low Byte'.

Esempio:

1111111001000111 binario = 65095 decim.

11111110 e 01000111 = 254 e 71 decimale

High-Byte e Low-Byte

Potremo ad esempio scrivere, ora:

```
POKE 792,71 : POKE 793,254
```

INTERRUZIONI

Nei casi normali il processore interrompe ogni sessantesimo di secondo il suo lavoro ed esegue il programma in linguaggio macchina le cui locazioni d'inizio sono fissate in entrambi i Bytes 788/789.
Durante queste interruzioni il computer interroga la tastiera e aggiorna l'orologio TI$ software.
Potete proibire le interruzioni per bloccare la tastiera onde accellerare l'esecuzione dei programmi che non necessitano di input.

PENNA LUMINOSA

Con una penna luminosa e' possibile comunicare con il computer tramite video.
Il computer puo' rivelare, attraverso il foto-diodo della penna luminosa, in quale posto del video si trova la penna stessa.
Il programma basic provvedera' poi a prendere determinate decisioni in base al collocamento della penna da parte dell'utente.

MODO MULTICOLORE

In modo multicolore possono essere usati all'interno di una carattere, ovvero un blocco di 8*8 punti, fino a quattro colori diversi.
Rispetto al metodo ALTA RISOLUZIONE, la risoluzione orizzontale e' dimezzata: ora due punti vengono mostrati come uno solo, ma da 4 potenziali colori.
Le coppie di 'due punti' sono: 00,01,10,11.

NIBBLE

Un Nibble e' 'mezzo Byte', ovvero 4 Bits.
Il valore maggiore di un Nibble e' percio' 15 (=1111).
All'interno di un Byte si parla di High Nibble (Bits #7 fino a #4) e di un Low Nibble (Bits #3 fino a #0)

PAGINE DI MEMORIA.

Una 'pagina' e' un ambito di memoria di 256 Bytes.
La memoria Commodore viene suddivisa in ben 256 pagine diverse.
Infatti:

64 K = 256 Bytes * 256 pagine = 65535 Bytes = 64 K).

BUS SERIALE

Al BUS del C-64 vengono collegati gli apparecchi periferici come: Floppy, stampante, Printer-Plotter etc.
Dal BUS vengono comunicati i dati solo in una direzione, ed uno dietro l'altro, bit dopo bit.

Percio' uno scambio seriale di dati e' piu' lento di uno parallelo, in cui si passano piu' bits alla volta.

CODICE MEMORIA

Confronta tabella codice ASCII

TOOL

ToolKit, tradotto: attrezzo
Si parla di ToolKit, se un programma (in linguaggio macchina) serve come strumento di sviluppo per altri programmi.

USERPORT (porta utente)

La Userport e' lo strumento attraverso il quale il nostro computer e' in grado di ricevere informazioni dal mondo esterno.

PAGINA ZERO.

La prima pagina della memoria del C-64 viene usata dal processore per memorizzare i piu' importanti dati, che sono necessari all'esecuzione dell'interprete BASIC e del sistema operativo.
Poiche' l'ambito di memoria da 0 fino a 1023 ha queste funzioni, vengono indicati come pagina zero, anche la 1 , 2 e 3.

CODICI VIDEO

CODICE	CARATTERE
0	@ / @
1	A / a
2	B / b
3	C / c
4	D / d
5	E / e
6	F / f
7	G / g
8	H / h
9	I / i
10	J / j
11	K / k
12	L / l
13	M / m
14	N / n
15	O / o
16	P / p
17	Q / q
18	R / r
19	S / s
20	T / t
21	U / u
22	V / v
23	W / w
24	X / x
25	Y / y
26	Z / z
27	[/ [
28	£ / €
29	] /]
30	↑ / ↑
31	← / ←
32	/
33	! / !
34	" "
35	# / #
36	$ / $
37	% / %
38	& / &
39	' / '
40	(/ (
41	) /)
42	* / *

CODICE	CARATTERE
43	+ / +
44	, / ,
45	- / -
46	. / .
47	/ / /
48	0 / 0
49	1 / 1
50	2 / 2
51	3 / 3
52	4 / 4
53	5 / 5
54	6 / 6
55	7 / 7
56	8 / 8
57	9 / 9
58	: / :
59	; / ;
60	< / <
61	= / =
62	> / >
63	? / ?
64	— / —
65	A / A
66	B / B
67	C / C
68	D / D
69	E / E
70	F / F
71	G / G
72	H / H
73	I / I
74	J / J
75	K / K
76	L / L
77	M / M
78	N / N
79	O / O
80	P / P
81	Q / Q
82	R / R
83	S / S
84	T / T
85	U / U

CODICE	CARATTERE
86	V / V
87	W / W
88	X / X
89	Y / Y
90	Z / Z
91	+ / +
92	▒ / ▒
93	│ / │
94	π / ▚
95	▗ / ▒
96	/
97	▮ / ▮
98	▬ / ▬
99	‾ / ‾
100	_ / _
101	│ / │
102	▒ / ▒
103	│ / │
104	≈ / ≈
105	◤ / ▒
106	│ / │
107	├ / ├
108	▪ / ▪
109	└ / └
110	┐ / ┐
111	─ / ─
112	┌ / ┌
113	┴ / ┴
114	┬ / ┬
115	┤ / ┤
116	│ / │
117	▮ / ▮
118	▮ / ▮
119	‾ / ‾
120	─ / ─
121	▬ / ▬
122	┘ / ┘
123	▪ / ▪
124	▪ / ▪
125	┘ / ┘
126	▪ / ▪
127	▚ / ▚

CODICI ASCII

CARATTERE	CODICE	CARATTERE	CODICE	CARATTERE	CODICE
	0	@	64		128
	1	A	65	ORN	129
	2	B	66		130
<RUN STOP>	3	C	67	LOAD+RUN	131
	4	D	68		132
WHT	5	E	69	F1	133
	6	F	70	F2	134
	7	G	71	F3	135
<SHIFT><C=>AUS	8	H	72	F4	136
<SHIFT><C=>AN	9	I	73	F5	137
	10	J	74	F6	138
	11	K	75	F7	139
	12	L	76	F8	140
<RETURN>	13	M	77	<SHIFT><RETURN>	141
UMSCH. TEXT	14	N	78	UMSCH. GRAFIK	142
	15	O	79		143
	16	P	80	BLK	144
CRSR. DOWN	17	Q	81	CRSR UP	145
RVS ON	18	R	82	RVS OFF	146
HOME	19	S	83	CLR	147
DEL	20	T	84	INST	148
	21	U	85	BRN	149
	22	V	86	L.RED	150
	23	W	87	GRY 1	151
	24	X	88	GRY 2	152
	25	Y	89	L GRN	153
	26	Z	90	L BLU	154
	27	[	91	GRY 3	155
RED	28	£	92	PUR	156
CRSR. RIGHT	29	]	93	CRSR. LEFT	157
GRN	30	↑	94	YEL	158
BLU	31	←	95	CYN	159
SPACE	32	—	96	SPACE	160
!	33	♠	97	▌	161
"	34	▏	98	▀	162
#	35	—	99	▁	163
$	36	—	100	—	164
%	37	—	101	▏	165
&	38	—	102	▒	166
'	39	▏	103	▕	167
(	40	▏	104	▔	168
)	41	╲	105	▝	169
*	42	╲	106	▕	170
+	43	╱	107	├	171
,	44	└	108	▗	172
-	45	╲	109	└	173
.	46	╱	110	┐	174
/	47	┌	111	▂	175
0	48	┐	112	┌	176
1	49	♠	113	┴	177
2	50	—	114	┬	178
3	51	♠	115	┤	179
4	52	▏	116	▏	180
5	53	╱	117	▌	181
6	54	╳	118	▌	182
7	55	○	119	▔	183
8	56	♠	120	▔	184
9	57	▏	121	▀	185
:	58	♠	122	▁	186
;	59	+	123	▖	187
<	60	▒	124	▝	188
=	61	▏	125	▗	189
>	62	π	126	▝	190
?	63	◥	127	▚	191