VIC da zero +64

di Tommaso Pantuso

Un po' di grafica con il 64

(II parte)

Abbiamo visto cos'è il bit map mode e come si possa accedere a questo tipo di configurazione del sistema per entrare in modo grafico. Oggi continuiamo il discorso aperto la volta scorsa spiegando come sia possibile, in modo grafico, gestire il singolo pixel della mappa di bit per ottenere dei disegni sullo schermo.

Il punto della situazione

Come visto, agendo su alcuni registri del Vic-II, è possibile abilitare il 64 a lavorare in modo grafico.

Praticamente, con delle operazioni di Peek e Poke (supponiamo di trovarci in ambiente Basic), forziamo il sistema in maniera tale da configurarsi con un'area di memoria da 8000 byte (64000 bit) che fungerà da memoria grafica, ed un'area da 1000 byte nella quale si definirà il colore dei punti che verranno visualizzati sullo schermo. Per chiarezza espositiva ripeteremo alcune nozioni già espresse, anche per rendere l'argomento comprensibile a chi non ha letto la puntata precedente.

Intanto, ciascuno dei 64000 bit che fanno parte dell'area grafica avrà una propria immagine sullo schermo nei seguenti termini. Se il bit viene posto ad 1 nella mappa di bit, allora sul teleschermo vedremo un punto illuminato mentre, se lo stesso bit assume valore 0, il punto sullo schermo sarà spento. L'insieme dei punti spenti formerà lo sfondo su cui i pixel accesi andranno a comporre il disegno.

Ora, sia per i pixel accesi che per quelli spenti è possibile definire un colore. La discriminazione tra i colori si ottiene gestendo i punti dell'area grafica a blocchi in un'area di memoria da 1000 byte. Cosa significa? È semplice.

Cominciamo dall'area grafica prendendo i suoi 64000 punti e suddividiamoli idealmente in blocchi da 64 bit: otteniamo un totale di 1000 blocchi. Andiamo ora nell'area da 1000 byte in cui definiremo il colore e creiamo una corrispondenza tra ciascun byte di quest'area e ciascun blocco dell'area grafica: ad ogni byte facciamo corrispondere un blocco, ciascuno in una posizione diversa della mappa, fino a coprire con la corrispondenza così creata, tutta l'area grafica.

La relazione byte-blocco è la seguente: nel byte il numero contenuto nei quattro bit più significativi (quelli più a sinistra) conterrà il codice del colore dei punti accesi nel corrispondente blocco, mentre il numero contenuto nei quattro bit meno significativi costituirà il codice del colore dei punti spenti. Il range di questi codici varia, per ovvi motivi, da 1 a 15 decimale cioè da 0000 a 1111 binario. È evidente quindi, come fatto fondamentale, che in questo modo non può essere controllato il colore dei singoli bit ma soltanto dei gruppi. Per definire quindi un solo colore per il disegno ed uno per lo sfondo non ci resta quindi che memorizzare nei 1000 byte dell'area di colore lo stesso numero.

Individuiamo un punto nella mappa di bit

Le zone di Ram che ci interessano sono: quella di 8000 byte posta a parti-

re dalla locazione 8192 decimale, la mappa grafica, e quella da 1000 byte posta a partire dalla locazione 1024 decimale.

Cominciamo col ricordare che, anche per quanto riguarda la grafica, nella mappa potremo lavorare sempre agendo su un byte, configurando opportunamente in esso i vari bit accendendo e spegnendo quelli desiderati, non potendo accedere semplicemente al singolo bit. In altre parole, quando andremo ad accendere un punto sullo schermo, lo faremo agendo su una locazione ad 8 bit che lo contiene la quale controlla necessariamente 8 dei punti che si trovano nella mappa di bit.

Si tratta allora per prima cosa di individuare, tra i 64000 possibili, il punto esatto che vogliamo illuminare. Quindi, agendo sulla locazione ad 8 bit che lo contiene, modificare solo quel punto senza influenzare gli altri. Se ci seguirete con un po' di attenzione,vedremo insieme come ciò sia possibile servendosi di un semplice algoritmo.

I punti della mappa grafica, possono essere immaginati suddivisi come illustrato nella figura 1. L'aspetto di tale area in questa figura è simile ad uno schermo in modo testo, nel quale troviamo 40 macro-colonne (da 0 a 39) e 25 macro-righe (da 0 a 24). Se agissimo in modo testo, su ogni riga potrebbero essere contenuti 40 caratteri, ciascuno dei quali definito, come ormai dovreste sapere, da un insieme di 8 byte. Prendendo tutti i byte allineati su una stessa fila è facile calcolare che, su una «fila di byte» sono contenuti 320 bit mentre, facendo il calcolo per colonne, ricaveremo che su una «colonna di byte» ci sono 200 bit.

Ogni singolo punto potrà allora essere individuato immaginandolo in un piano cartesiano di 320×200 punti. Supponiamo di voler individuare, in questo piano, il punto P di coordinate (X,Y).

Per prima cosa sarà necessario intercettare, tra gli 8000 byte della mappa grafica — identificabili con le locazioni variabili da 8192 a 16191 — il

```
                    MACRO    MACRO    MACRO    ..   MACRO
                    COLONNA  COLONNA  COLONNA  ..   COLONNA
                      0        1        2            39

                    76543210 76543210 76543210       76543210
        byte 0      oooooooo oooooooo oooooooo  ..   oooooooo  byte 312
        byte 1      oooooooo oooooooo oooooooo  ..   oooooooo  byte 313
        byte 2      oooooooo oooooooo oooooooo  ..   oooooooo  byte 314
MACRO   byte 3      oooooooo oooooooo oooooooo  ..   oooooooo  byte 315
RIGA    byte 4      oooooooo oooooooo oooooooo  ..   oooooooo  byte 316
 0      byte 5      oooooooo oooooooo oooooooo  ..   oooooooo  byte 317
        byte 6      oooooooo oooooooo oooooooo  ..   oooooooo  byte 318
        byte 7      oooooooo oooooooo oooooooo  ..   oooooooo  byte 319

        byte 320    oooooooo oooooooo oooooooo  ..   oooooooo  byte 632
        byte 321    oooooooo oooooooo oooooooo  ..   oooooooo  byte 633
        byte 322    oooooooo oooooooo oooooooo  ..   oooooooo  byte 634
MACRO   byte 323    oooooooo oooooooo oooooooo  ..   oooooooo  byte 635
RIGA    byte 324    oooooooo oooooooo oooooooo  ..   oooooooo  byte 636
 1      byte 325    oooooooo oooooooo oooooooo  ..   oooooooo  byte 637
        byte 326    oooooooo oooooooo oooooooo  ..   oooooooo  byte 638
        byte 327    oooooooo oooooooo oooooooo  ..   oooooooo  byte 639

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

        byte 7680   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7992
        byte 7681   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7993
        byte 7682   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7994
MACRO   byte 7683   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7995
RIGA    byte 7684   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7996
 24     byte 7685   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7997
        byte 7686   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7998
        byte 7687   oooooooo oooooooo oooooooo  ..   oooooooo  byte 7999
```

Figura 1 - Schematizzazione della mappa di bit.

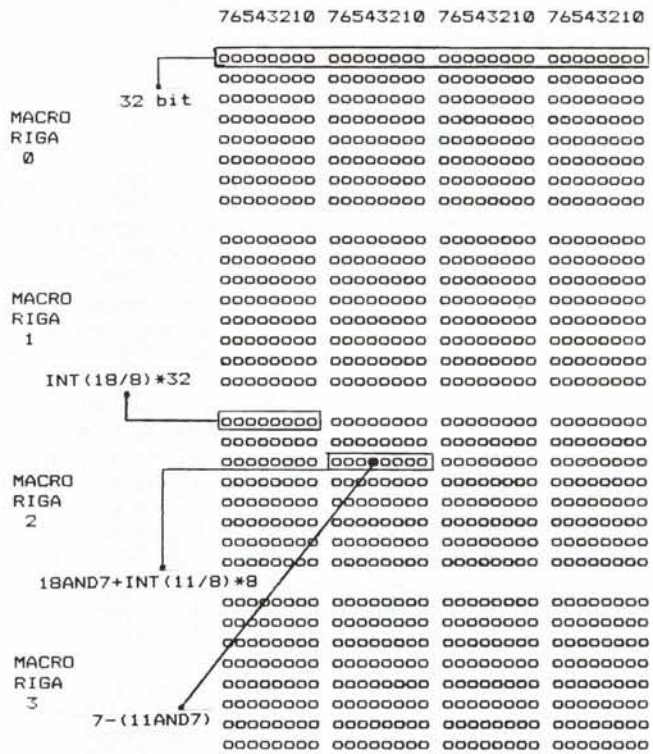

Figura 2 - Procedimento d'intercettazione del punto di coordinate (11,18) sulla mini-mappa di bit.

Listato 1

```
1 REMARK - ROUTINE PER IL TRACCIAMENTO
2 REMARK - DI UNA SINUSOIDE
3 REMARK
10 FORC=1024TO2023:POKEC,16:NEXT
20 REM POSIZIONAMENTO MEMORIA GRAFICA (8K)
30 POKE53272,PEEK(53272)OR8
40 REM SPOSTAMENTO DEL BIT MAP
50 POKE53265,PEEK(53265)OR32
60 REM PULIZIA MEMORIA GRAFICA
70 FORK=8192TO8192*2:POKEK,0:NEXT
80 REM FUNZIONE
90 FORX=0TO319
100 Y=90+80*SIN(X/10)
110 REM INDIVIDUAZIONE PUNTO
120 BYTE=8192+(INT(Y/8)*320)+(INT(X/8)*8)+(YAND7)
130 BIT= 7-(XAND7)
140 REM PLOTTAGGIO
150 POKEBYTE,(PEEK(BYTE)OR(2↑BIT))
160 NEXT
170 GETA$:IFA$=""THEN170
180 POKE53265,27
190 POKE53272,21
```

numero di byte da modificare. Ciò sarà fatto localizzando prima la riga elementare (una di quelle composte da 320 pixel) e poi scorrendo su di essa, di 8 byte per volta, fino a fermarci in quello che effettivamente contiene il punto. Fatto ciò, all'interno del byte, dovremo individuare la posizione esatta del bit e modificarlo.

Il tutto può essere ricavato con le relazioni seguenti che, nel prossimo paragrafo, illustreremo con un esempio.

Il numero del byte da 8192 a 16191, considerando sempre il punto (X,Y), si ricava con

BYTE = 8192 + INT(Y/8) × 320 + Y AND 7 + INT(X/8) × 8

dove 320 è il numero di pixel per linea elementare mentre, il bit all'interno del byte, è individuato dalla relazione BIT = 7 − (X AND 7).

Una volta individuato il bit all'interno della mappa grafica, l'accensione del pixel corrispondente sullo schermo si ottiene semplicemente con:

PIXEL = POKE BYTE, (PEEK(BYTE) OR (2 ↑ BIT)

Questo è tutto, ed in teoria si tratta di un procedimento abbastanza semplice ma siamo sicuri che ci sono buone probabilità che il significato delle varie operazioni non sia chiaro a tutti. È per questo motivo che proseguiamo con un esempio che dovrebbe fugare ogni dubbio.

Un esempio

Intanto cerchiamo di metterci in condizione di seguire meglio il ragionamento che faremo portandoci in uno spazio di azione più ristretto: invece di considerare uno spazio grafico di 64000 punti, cioè 8000 byte a loro volta divisi in gruppi da 8 byte, consideriamo un piano più piccolo, riproducibile comodamente su carta come illustrato nella figura 2. Il piano che useremo per l'esempio lo immaginiamo composto da 1024 punti, (32 × 32) sempre suddiviso in gruppi da 8 byte (64 pixel): è chiaro che ora ogni linea elementare è composta da 32 punti, lo stesso per le colonne.

Ancora, per rendere più evidente il legame con il caso generale, supponiamo che questa mini-mappa di bit abbia inizio alla locazione 8192 (e termini alla 8319).

Dopo queste premesse possiamo quindi partire supponendo di voler individuare, all'interno della mappa, il punto di coordinate

(11,18)

disegnato in nero sulla nostra figura.

Per prima cosa individueremo la macro-riga (da 0 a 3) che contiene il punto e, per far ciò, partendo dall'ordinata di valore 18, calcoleremo

MACRO-RIGA = INT(18/8) = 2.

Essendo ogni linea elementare composta da 32 pixel, moltiplicando il valore appena ricavato per 32 ci posizioneremo all'inizio del blocco da 8 byte, il primo della macro-riga, cioè individueremo automaticamente il valore della prima delle sue locazioni. Nel nostro caso, facendo i conti sulla figura, tale locazione è la 64 esima e, infatti, lo stesso numero otterremo con:

PRIMO BYTE MACRO-RIGA = MACRO-RIGA × 32 = 64.

Da questa posizione possiamo continuare a muoverci individuando, all'interno del blocco 8 × 8 a cui siamo giunti, di quanti bit verso il basso (da 0 a 7) sia spostato, partendo dall'inizio del blocco stesso, quello di ordinata 18. Ciò si ottiene con un calcolo di resti. In altre parole, con le operazioni fatte ci siamo posizionati sul primo byte del blocco perché abbiamo considerato il valore intero della divisione tra 18 e 8 tralasciando il resto. È proprio questo resto che ora terremo in considerazione calcolandolo con un'operazione di And. In generale infatti, per calcolare il resto di un numero che stiamo dividendo per una potenza di 2, basta eseguire l'And tra il numero e il suo divisore diminuito di 1. Nel nostro caso, il resto di 18/8 (che è 2), può essere calcolato con:

RESTO Y = 18 AND 7.

Questa operazione può essere visualizzata in maniera più immediata con la seguente schematizzazione:

```
  10010 (18 in binario)
  01111 (7  in binario)
AND 00010 (2  in binario).
```

Naturalmente, volendo, il resto può essere anche ricavato con un'operazione del tipo: 18 − INT(18/8) × 8.

Tornando a noi, abbiamo ricavato che, all'interno del blocco che abbiamo individuato, il pixel che cerchiamo è nella linea elementare che passa per il terzo byte (byte 2). Ora non ci resta che spostarci in orizzontale di un certo

Listato 2

```
1 REMARK - LA ROUTINE LM CARICATA CON
2 REMARK - QUESTO PROGRAMMINO PULISCE
3 REMARK - LA PAGINA GRAFICA MOLTO PIU'
4 REMARK - RAPIDAMENTE DI UN PROGRAMMA
5 REMARK - IN BASIC
6 REMARK
10 DATA169,0,133,251,169,32,133,252,162
20 DATA32,160,0,152,145,251,200,208,251
30 DATA202,240,4,230,252,208,244,96
40 FORN=49152TO49177:READA:POKEN,A:NEXT
50 POKE53272,PEEK(53272)OR8
60 POKE53265,PEEK(53265)OR32
70 GETA$:IFA$=""THEN70
80 SYS 49152
```

Listato 3

```
10 REMARK - QUI OLTRE A PULIRE LA PA-
20 REMARK - GINA GRAFICA SI DEFINISCE
30 REMARK - ANCHE IL COLORE DEL DISE-
40 REMARK - GNO E DELLO SFONDO
50 REMARK
200 DATA169,0,133,251,169,32,133,252,162
210 DATA32,160,0,152,145,251,200,208,251
220 DATA202,240,4,230,252,208,244,96
230 FORN=49152TO49177:READA:POKEN,A:NEXT
240 POKE53272,PEEK(53272)OR8
250 POKE53265,PEEK(53265)OR32
260 GETA$:IFA$=""THEN260
270 SYS 49152
280 DATA169,0,133,254,169,4,133,255,162
290 DATA4,160,0,152,145,254,200,208,251
300 DATA202,240,4,230,255,208,244,96
310 FORN=49178TO49203:READA:POKEN,A:NEXT
320 GETA$:IFA$=""THEN320
330 SYS 49178
```

Listato 4

```
1 REMARK - TRACCIAMENTO DI SINUSOIDE
2 REMARK - CON PULIZIA DELLA PAGINA
3 REMARK - GRAFICA E DEFINIZIONE DEL
4 REMARK - COLORE IN LM
5 REMARK
10 REM LETTURA DATI
20 FORN=49152TO49203:READA:POKEN,A:NEXT
30 REM POSIZIONAMENTO MEMORIA GRAFICA (8K)
40 POKE53272,PEEK(53272)OR8
50 REM SPOSTAMENTO DEL BIT MAP
60 POKE53265,PEEK(53265)OR32
70 REM PULIZIA MEMORIA GRAFICA
80 SYS 49152
90 REM POSIZIONAMENTO DEL COLORE
100 SYS 49178
110 REM FUNZIONE
120 FORX=0TO319
130 Y=99+99*SIN(X/10)
140 REM INDIVIDUAZIONE PUNTO
150 BYTE=8192+(INT(Y/8)*320)+(INT(X/8)*8)+(YAND7)
160 BIT= 7-(XAND7)
170 REM PLOTTAGGIO
180 POKEBYTE,(PEEK(BYTE)OR(2↑BIT))
190 NEXT
200 GETA$:IFA$=""THEN200
210 REM RITORNO IN MODO TESTO
220 POKE53265,27
230 POKE53272,21
240 PRINT"□"
250 REM ROUTINE PULIZIA MEMORIA GRAFICA
260 REM
270 DATA169,0,133,251,169,32,133,252,162
280 DATA32,160,0,152,145,251,200,208,251
290 DATA202,240,4,230,252,208,244,96
300 REM ROUTINE POSIZIONAMENTO COLORE
310 REM
320 DATA169,253,133,254,169,3,133,255,162
330 DATA4,160,3,152,145,254,200,208,251
340 DATA202,240,4,230,255,208,244,96
```

numero di posti fino a trovare, partendo dal primo byte della macro-riga, il byte che racchiude il punto. Dato che ogni byte è un gruppo di 8 bit, ci sposteremo di 8 posizioni per volta servendoci come base del valore 11 dell'ascissa:

SPOSTAMENTO = INT(11/8) × 8 = 8.

Mettendo insieme tutti i risultati ottenuti fin'ora e tenendo presente di aver supposto che anche in questo esempio la mappa video inizia alla locazione 8192, otterremo la relazione:

BYTE = 8192 + PRIMO BYTE MACRO-RIGA + RESTO Y + SPOSTAMENTO

con cui individueremo in quale byte è contenuto il punto.

Non ci rimane ora che individuare, nel byte la posizione esatta del bit che vogliamo accendere. Per far ciò considereremo ancora un resto, esattamente quello della divisione tra l'ascissa 11 ed 8. Con questa divisione avevamo individuato di quanti byte bisognava spostarsi dall'inizio della macro-riga per individuare il byte contenente il punto. Il resto di questa divisione ci fornisce automaticamente dopo quanti bit dall'inizio del byte troveremo il punto. Anche in questo caso utilizzeremo un'operazione di And logico solo che, essendo ogni bit del byte numerato da 0 a 7 partendo però da sinistra, la posizione del bit ci verrà restituita con:

BIT = 7 – (11 AND 7).

Nel nostro caso BIT = 4 quindi ,una volta individuato il byte, il bit interessato viene acceso — supponendo di utilizzare il Basic — da:

P = POKE BYTE, (PEEK (BYTE) OR (2^BIT).

L'operazione di Or logico si utilizza per agire solo sul bit individuato senza influenzare gli altri. Infatti, supponendo che in BYTE sia contenuta una configurazione qualsiasi, avremo:

BYTE	xxxxxxxx
2^4	00100000
AND	xx1xxxxx

Rifacendo questi ragionamenti per la mappa reale, quella 320 × 200, ritroveremo le relazioni date nel paragrafo precedente.

Alcuni programmi d'esempio

Se per iniziare volete fare qualche prova, potete utilizzare i programmini dimostrativi che troverete in queste pagine.

Il primo, con le righe da 90 a 170, traccia una sinusoide utilizzando l'algoritmo studiato nell'articolo. La riga 10 definisce il colore dei pixel accesi e spenti mentre la 70 effettua la pulizia della mappa grafica. In bit map mode si passa con le righe 30 e 50 mentre si ritorna in modo testo con la 180 e la 190.

Il listato 2 riporta una routine in linguaggio macchina con cui viene pulita la pagina grafica in pochi istanti e può essere utilizzata al posto della linea 70 del listato 1.

Nel listato 3 viene riportata, in aggiunta alla routine di pulizia della pagina grafica, anche una di definizione del colore.

Entrambe le ultime routine vengono impiegate nel listato 4 il quale ottiene lo stesso effetto del listato 1 ma con un notevole risparmio di tempo rispetto a quest'ultimo essendo state velocizzate le operazioni preliminari più noiose.

Naturalmente anche il plottaggio dei punti sarebbe molto più veloce se fatto in LM ma per ora preferiamo tralasciare questo argomento. Il nostro scopo era quello di rendere più chiaro un argomento spesso non molto approfondito sui testi dedicati al 64 e speriamo di esserci riusciti.

MC