di Pierluigi Panunzi

Il Mouse

sesta parte

Questa è l'ultima puntata della serie di articoli riguardanti la gestione di un mouse da parte di programmi scritti in Assembler, Pascal oppure in Basic: parleremo perciò delle ultime funzioni rimaste da analizzare. Ancora una volta, per avere sempre ben chiara la situazione, riportiamo in figura 1 l'elenco delle funzioni di gestione del mouse attivabili tramite la chiamata all'INT 33H, una volta caricato il driver del mouse, fornito dalla casa costruttrice del mouse stesso

Le funzioni 0DH e 0EH: Enable & Disable light pen emulation

Eccoci dunque a parlare di un oggetto, la «light pen» (penna ottica), che praticamente non ha avuto diffusione né tantomeno è stato considerato dai produttori di software, al pari, tanto per fare un esempio, del fantomatico «lettore di cassette» citato più volte nella letteratura IBM, ma mai (per fortuna...) effettivamente usato.

Tornando alla light pen, c'è da dire che le premesse erano buone, dal momento che sia la scheda grafica CGA che la EGA possedevano un connettorino a 5 contatti dove poter collegare la penna ottica; inoltre nel BASICA prima e nel GWBASIC poi (rispettivamente il Basic ufficiale di casa IBM e che girava solo grazie alle ROM presenti nella piastra madre ed il Basic dei compatibili) sono implementate alcune funzioni (**ON PEN GOSUB** e **PEN (...)**) che gestiscono appunto la penna ottica.

Data poi la sua scarsa popolarità, la penna ottica non è stata nemmeno presa in considerazione da compilatori di altri linguaggi ad alto livello, primo fra tutti il Turbo Pascal che la ignora completamente.

Poi, con l'avvento della VGA (sulla piasta madre dei PS/2 oppure come scheda per i compatibili) e delle successive Super VGA, il famoso connettorino è completamente scomparso, calando così il sipario verso un componente del personal computer che tutto sommato meritava un po' più di gloria, al pari dei ben più diffusi mouse o tavolette grafiche.

Comunque, più per dovere di cronaca che per effettiva utilizzabilità, parliamo del fatto che abbiamo la possibilità di simulare il comportamento della pen-

AX = 0H	Mouse Reset
AX = 1H	Cursor Enable
AX = 2H	Cursor Disable
AX = 3H	Get Mouse Position and Button Status
AX = 4H	Set Mouse Position
AX = 5H	Get Button Press Information
AX = 6H	Get Button Release Information
AX = 7H	Set min & max horizontal position
AX = 8H	Set min & max vertical position
AX = 9H	Set Graphic Cursor Block
AX = 0AH	Set Text Cursor
AX = 0BH	Read Motion Counters
AX = 0CH	Set User-defined subroutine
AX = 0DH	Enable Light Pen Emulation
AX = 0EH	Disable Light Pen Emulation
AX = 0FH	Set Mickey/Pixel Ratio
AX = 10H	Window Conditional Off
AX = 12H	Set Large Graphic Cursor
AX = 13H	Set Speed Threshold

Tabella 1 - Come di consueto riportiamo una tabella riassuntiva di tutte le funzioni gestibili da un driver del mouse.

Figura 2 - Queste due funzioni servono per gestire il mouse come se fosse quell'oscuro oggetto chiamato «light pen», oggetto molto utile e ben conosciuto, ma che praticamente ha avuto una diffusione ed utilizzazione quasi nulla.

AX = 0DH	Enable Light Pen Emulation
INPUT	-
OUTPUT	-

AX = 0EH	Disable Light Pen Emulation
INPUT	-
OUTPUT	-

na ottica per mezzo del mouse, il che in sé non sarebbe nemmeno tanto intelligente, dal momento che entrambi gli oggetti servono come elementi di puntamento e perciò è a dir poco strano simularne uno con l'altro: il tutto, pensiamo, solo per poter finalmente utilizzare le routine ON PEN GOSUB e PEN (...) del Basic.

Ecco che dunque, attivando la funzione **0DH** del gestore del mouse, quest'ultimo viene visto (solo dal BASICA e dal GWBASIC, e da ora non lo diremo più) come se fosse una penna ottica, che viene attivata (e ciò corrisponde alla pressione del pulsantino in genere posto su di essa per attivarla) premendo contemporaneamente i due pulsanti del mouse (destro e sinistro per i mouse a tre pulsanti).

Ecco dunque la prima scomodità: premere i due pulsanti quando sarebbe bastato richiedere la pressione di uno dei due, magari a scelta, per ottenere la stessa funzione più agevolmente. Ciò fa pensare tra l'altro ad un tentativo di scoraggiare l'utente all'utilizzazione della funzione... (N.d.r. ci sono riusciti benissimo!).

Attivando invece la funzione **0EH** si ottiene la disabilitazione dell'emulazione della penna ottica e da quel momento in poi il Basic farà riferimento alla vera light pen ogni volta che si utilizzano le funzioni citate.

Detto questo non resta che presentare un programmino (forzatamente, se vogliamo) scritto in Basic, che gestisce una pseudo penna ottica «da tavolo»: non ci siamo sprecati più di tanto, proprio per lasciare al lettore la possibilità di adattare il tutto alle proprie esigenze.

In figura 3 vediamo dunque questo programmino che, dopo aver inizializzato l'interfacciamento verso il driver del mouse, resetta il tutto con la funzione **0** e poi, invece di chiamare la funzione **1** per mostrare, in grafica, il cursore, abilita l'emulazione della penna ottica con la funzione **13** e subito ne predispone la gestione con la chiamata ON PEN GOSUB 200 che farà dunque chiamare la subroutine posta a 200 ogni volta che la penna è attivata (premuti i due pulsanti del mouse).

```
 10 DEFINT M
 20 DEF SEG=0:MSEG=256*PEEK(51*4+3)+PEEK(51*4+2)
 30 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)+2
 40 SCREEN 9
 50 DEF SEG=MSEG
 60 M1=0:CALL MOUSE(M1,M2,M3,M4)
 70 M1=13:CALL MOUSE(M1,M2,M3,M4)
 80 PEN ON:ON PEN GOSUB 200
 90 WHILE INKEY$="":WEND
100 M1=14:CALL MOUSE(M1,M2,M3,M4)
110 END
200 LOCATE 1,1: PRINT PEN(1),PEN(2):RETURN
```

Figura 3 - Programmino di esempio di simulazione di una penna ottica per mezzo del mouse.

Figura 4 - Questa funzione consente di spegnere il cursore relativo al mouse, quando questo passa attraverso una zona rettangolare dello schermo.

AX = 10H	Window Conditional Off
INPUT	CX = window x0 DX = window y0 SI = window x1 DI = window y1
OUTPUT	-

La subroutine non fa altro che scrivere in alto a sinistra le due coordinate correnti della pseudo-penna: non dimentichiamoci però di abilitare il Basic a questo tipo di gestione per mezzo della chiamata PEN ON.

Con questo caliamo anche noi il sipario e passiamo alle altre funzioni...

La funzione 10H: Window Conditional Off

Questa funzione, facendo riferimento alla figura 4, consente di cancellare il cursore (freccia o chi per essa) se per caso viene mosso all'interno di una zona rettangolare specificata, in occasione dell'aggiornamento della zona stessa da parte del programma e cioè in tutti quei casi in cui la presenza del cursore provocherebbe dei malfunzionamenti e soprattutto rallenterebbe l'aggiornamento dell'area critica.

Possiamo fare un esempio: supponiamo di avere un programma di grafica che, a seguito di un comando attivabile tramite mouse (clickando ad esempio su di un'icona) debba aggiornare una ben determinata area dello schermo, andando a testare i colori già presenti nell'area stessa.

Ora se inavvertitamente o volutamente passiamo il cursore in quell'area, potrebbe succedere che il pro-

```
uses graph,crt,dos;

const xx = 50;
      yy = 50;

var gd,gm,x,y : integer;
    reg : registers;
    ch : char;

procedure mouse(ax : word);
begin
 reg.AX := ax;
 intr($33,reg);
end;

begin
 clrscr;
 gd := detect;
 initgraph(gd,gm,'');
 mouse(0);
 mouse(1);
 setcolor(blue);
 rectangle(5,5,xx - 5,yy - 5);
 repeat
  reg.CX := 0;                    {*}
  reg.DX := 0;                    {*}
  reg.SI := xx;                   {*}
  reg.DI := yy;                   {*}
  mouse(16);                      {*}
  for x := 0 to xx do
   for y := 0 to yy do
    putpixel(x,y,getpixel(x,y) + 1);
  mouse(1);                       {*}
  delay(300);
 until keypressed;
 ch := readkey;
 mouse(0);
end.
```

*Figura 5 - Programmino di esempio, questa volta in Turbo Pascal, di test della funzione **10H.** Per le istituzioni marcate con «{*}» si veda il testo: in breve escludono la funzione 10H ed i suoi benefici effetti.*

AX = 12H	Set Large Graphic Cursor
INPUT	BH = cursor width (words) BL = cursor x hot spot CH = cursor height (pixels) CL = cursor y hot spot ES:DX = pointer to mask
OUTPUT	-

*Figura 6 - La funzione **12H** consente di definire un cursore grafico di dimensioni orizzontali e verticali a piacere, oltre i valori standard (funzione 9) di 16x16 pixel.*

gramma (che supponiamo ci metta un po' di tempo a fare i suoi calcoli durante l'aggiornamento grafico) legga i pixel corrispondenti al cursore come appartenenti all'area stessa e che perciò li vada ad aggiornare erroneamente.

Considerato che il cursore si sta muovendo, ecco che in genere si otterrebbe una scia colorata che segue i movimenti del cursore.

Invece, fornendo alla funzione 10H le coordinate x-y dei vertici in alto a sinistra e in basso a destra del rettangolo da salvaguardare ed attivando tale funzione, otterremo lo spegnimento automatico del cursore, nel caso che questo attraversi l'area, durante l'aggiornamento dell'area stessa.

Al termine di tale aggiornamento bisogna rendere di nuovo visibile il cursore con la chiamata alla funzione **1** ed in tal modo potremo stare tranquilli.

```
const buf : array[0..35] of word =
        ($f7df,$c3ff,   {screen mask}
         $f39f,$00ff,
         $e10f,$01ff,
         $e00e,$03ff,
         $c006,$07ff,
         $c006,$03ff,
         $8003,$01ff,
         $8003,$00ff,
         $0001,$c3ff,

         $0820,$3c00,   {cursor mask}
         $0c60,$ff00,
         $1ef0,$fe00,
         $1ff1,$fc00,
         $3ff9,$f800,
         $3ff9,$fc00,
         $7ffc,$fe00,
         $7ffc,$ff00,
         $fffe,$3c00);
```

```
with reg do
 begin
  BH := 2;
  BL := 0;
  CH := 9;
  CL := 0;
  ES := seg(buf[0]);
  DX := ofs(buf[0]);
 end;
mouse(18);
```

Figura 7 - Due parti da aggiungere al programma di figura 5 per vedere come funziona la gestione di un «Large Graphic Cursor». Si veda il testo per sapere dove inserire queste due parti nel programma stesso.

In figura 5 vediamo un esempio in Turbo Pascal: in esso vogliamo cambiare continuamente colore all'interno di un'area dove abbiamo inizialmente tracciato un rettangolo blu: per fare questo utilizziamo apposta un procedimento alquanto lento che prevede la scansione pixel a pixel dell'area desiderata con modifica del colore di ogni singolo pixel incrementando di 1 il colore corrente.

Nel tutto abbiamo immesso il mouse, come elemento di disturbo: con il programma così com'è non succede niente se spostiamo il mouse nella zona critica (anzi abbiamo aggiunto quel "delay(300)" per poter effettivamente vedere di tanto in tanto il cursore nella zona).

Invece commentando le istruzioni marcate con l'asterisco e cioè disabilitando la funzione in esame e la successiva (oramai inutile) riaccensione del cursore, otterremo... ma lasciamo al lettore la possibilità di provare da sé gli effetti, mutevoli a seconda dello spostamento del cursore, e magari per valori più piccoli di **xx** e **yy** che comportano una velocità maggiore di aggiornamento dell'area.

La funzione 12H: Set Large Graphic Cursor

Eccoci dunque all'ultima funzione da analizzare: abbiamo già detto che si tratta di una funzione non prevista nello standard della Microsoft, ma presente in molti mouse «compatibili».

```
0000100000100000 0011110000000000
0000110001100000 1111111100000000
0001111011110000 1111111000000000
0001111111110001 1111110000000000
0011111111111001 1111100000000000
0011111111111001 1111110000000000
0111111111111100 1111111000000000
0111111111111100 1111111100000000
1111111111111110 0011110000000000
```

*Figura 8 - Questa è la rappresentazione schematica del **logo** di MC, come sequenza di 2 word orizzontali per 9 verticali: si tratta della **cursor mask**, mentre la **screen mask** non è altro che la complementare di questa (0 al posto di 1 e viceversa).*

Si tratta della funzione che consente di crearsi un cursore più grande di 16x16 pixel, laddove ciò occorresse: rimandiamo per alcuni dettagli al numero 88 di MCmicrocomputer, nel quale abbiamo descritto la funzione 9 («Set Graphic Cursor»).

In particolare, un cursore grande avrà una dimensione orizzontale pari ad un multiplo «n» prefissato di 16 pixel (quelli che non ci servono poi saranno posti a 0 nelle maschere), mentre per quanto riguarda la dimensione verticale, può essere fissata a piacere.

Nella figura 6, vediamo che bisogna perciò porre:

— in **BH** il valore «n» del multiplo di cui sopra (in pratica il numero di word che compongono orizzontalmente il cursore);

— in **CH** il numero di pixel in verticale;

— in **BL** e **CL** rispettivamente le coordinate x ed y del cosiddetto «hot spot» del quale abbiamo appunto parlato nella puntata citata in precedenza, coordinate relative all'angolo in alto a sinistra del rettangolo che costituisce il cursore;

— nella coppia **ES:DX** l'indirizzo completo (segment:offset) della zona di memoria che contiene, in successione, la **screen mask** e la **cursor mask**, formate ognuna da **BH**x**CH** word (cioè «n» x «bit-in-verticale» word).

Più di tutto varrà l'esempio, per il quale basta aggiungere al programma di figura 5 le due parti indicate in figura 7: in particolare la definizione del vettore **buffer** (una costante) va messa ovviamente in testa al programma, mentre il blocco di istruzioni va posto nel programma principale subito dopo l'inizializzazione del mouse con **mouse(0)** e prima delle sua accensione **mouse(1)**.

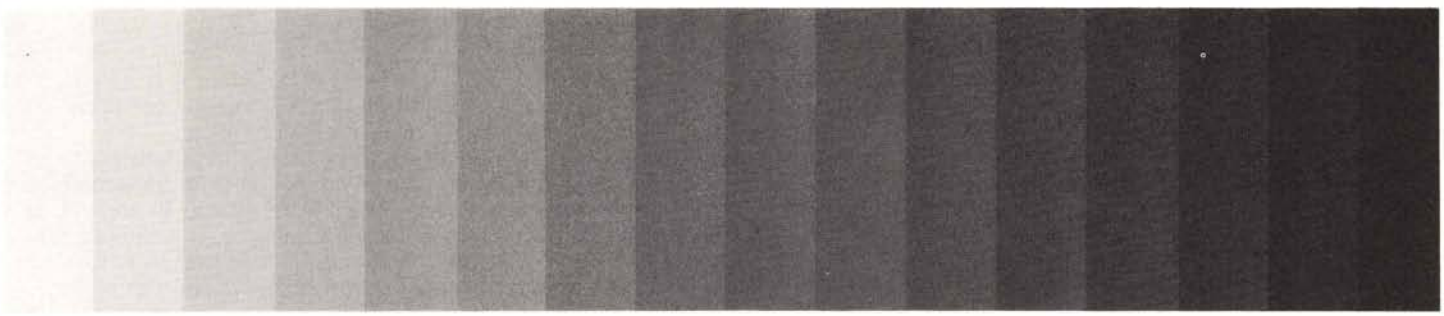
In questo caso il cursore assumerà l'aspetto del **logo** di MCmicrocomputer (vedasi la figura 8, dove è rappresentata la sola cursor mask in quanto la screen mask è esattamente la negata), mentre il programma funzionerà esattamente come prima.

Prima di concludere dunque la lunga serie di puntate relative alla gestione del mouse, diciamo che, visto che quest'ultima funzione non rientra nello standard Microsoft, non è detto che il programma funzioni per tutti i tipi di mouse: quasi sicuramente non dovrebbe funzionare laddove si usino mouse della Microsoft!

A risentirci dunque sul prossimo numero.

MC

E' GRIGIA.



LA SUPERIORITA' DI UN MONITOR CORNERSTONE E' BASATA SU 16 TONALITA' DI GRIGIO.

Nel Desk Top Publishing, nelle applicazioni Cad-Cam e nella grafica la qualità a video riveste sempre maggior importanza. Cornerstone ed EIS propongono al mercato italiano una linea di monitor monocromatici dalle prestazioni eccezionali: i due modelli base da 15'' e 19'' nelle varie versioni consentono di ottenere fino a 16 tonalità di grigio con definizioni da 768 x 1008 PIXEL - per il modello da 15'' single page XL e da 1600 x 1280 PIXEL - per la versione da 19'' dual page. Dotati di schermo antiriflesso, i monitor Cornerstone presentano driver per i più diffusi pacchetti DTP e sono anche compatibili con software per schede Hercules; sono altresì compatibili con i più diffusi computer sia di classe AT sia Micro Channel®. Tutte queste caratteristiche innovative, rendono i monitor Cornerstone prodotti unici, per affidabilità e qualità delle prestazioni. Distribuito in Italia da EIS - Via Fieno 8 - 20123 MILANO - Tel. (02) 80.99.61.

CORNERSTONE TECHNOLOGY

EIS Editrice Italiana Software

SOLUZIONI SEMPLICI A PROBLEMI COMPLESSI.