PRINT@ et PLOT en mode HIRES

par André C.

Les commandes PRINT@ et PLOT sont rejetées en mode HIRES. On obtient le message d'erreur "?DISP TYPE MISMATCH ERROR ". C'est une limitation très fâcheuse pour utiliser la zone TEXT du bas de l'écran ! Et c'est un problème qui me titille depuis des années.

Pour développer un jeu en mode HIRES, j'avais besoin d'exploiter les 3 lignes de la zone TEXT pour y afficher certains paramètres de l'évolution du jeu (messages, score etc.) et il fallait trouver une solution.

Etat des lieux

En mode HIRES, la zone TEXT fonctionne comme l'écran TEXT normal (mais hélas avec quelques limitations). Si les numéros de colonnes restent inchangés (X va de 0 à 39), par contre, les numéros de lignes sont restreints (Y va de 1 à 3). En mode HIRES, on peut observer ces valeurs en suivant le contenu des variables #0268 (numéro de ligne du curseur TEXT) et #0269 (numéro de colonne du curseur TEXT). Mais à quoi cela peut-il servir puisque les commandes PRINT@X,Y; et PLOTX,Y sont prohibées? C'est ce que nous allons voir.

Pour la petite anecdote

Il existe en RAM, entre la zone écran HIRES et la zone écran TEXT, une zone mémoire de 40 octets, inutilisée à l'écran, qui pourrait correspondre à une ligne numéro zéro (une sorte de ligne service inutilisée). Il s'agit peut-être d'un résidu de développement. Notons sur le plan pratique, qu'avec une routine appropriée, il serait possible de scroller la zone TEXT d'une ligne vers le haut (c'est à dire cacher la ligne numéro 1 en vue de la récupérer ultérieurement). Tout autre usage est également à recommander (par exemple stockage de variables ou d'un bout de code en langage machine). Comme en mode TEXT, le scrolling de l'écran vers le haut n'affecte pas cette "ligne 0".

Revenons à nos moutons

Que faire pour afficher sélectivement au point X,Y ? Il n'y a pas 36000 solutions. Soit on arrive à repositionner le curseur en X,Y et le PRINT suivant se fera à partir de ce point. Soit on calcule l'adresse en RAM correspondant à ce point X,Y et on y POKE le code Ascii souhaité.

La zone texte commence en #BF68. La formule est donc AD=#BF40+(#28*Y)+X avec Y de 1 à 3 et X de 0 à 39. Mais le problème n'est pas là. Pour afficher une chaîne de caractère, il faudra découper celle-ci avec la commande MID$ et en POKEr tous les code Ascii un à un. Idem pour les nombres qu'il faut d'abord transformer en chaine de caractères. Vous pouvez imaginer la lenteur de l'affichage !

Reste donc la commande PRINT qui a la particularité d'afficher au curseur. Le problème est élémentaire mon cher Watson, il suffit de repositionner le curseur. En quelque sorte, la commande PRINT@X,Y; est scindée en deux :
1) positionner le curseur en X,Y
2) envoyer un PRINT.

Positionner le curseur sur une ligne Y

Je me suis demandé s'il ne serait pas possible de forcer la variable #0268 avec les valeurs 1, 2 ou 3 à l'aide d'un POKE. Zut, ça ne marche pas : l'affichage commence systématiquement sur la première ligne. Divers autres essais ne se sont pas révélés plus heureux.

A la réflexion, ce n'est pas grave, car il suffit d'un PRINT pour passer à la ligne suivante et comme il n'y a que 3 lignes, c'est vite fait. Cette solution serait bien trop lourde en mode TEXT où il faudrait répéter 26 fois la commande PRINT pour placer le curseur sur la dernière ligne !

Je rappelle en passant que PRINT"" (chaîne vide) ne fait rien sinon passer à la ligne suivante et que PRINT""; ne fait absolument rien (sinon valider des variables du système, mais c'est un autre sujet).

Il est utile de se rappeler que la commande CLS nettoie la zone TEXT et repositionne le curseur au début de la première ligne. Elle peut être utilisée à tout moment, alors que les comman-

des PAPER et INK doivent être placées avant la commande HIRES si on veut affecter la zone TEXT. Sinon, elles affectent la zone HIRES.

Attention, la zone TEXT est scrollée exactement comme en mode TEXT, mais comme ici on ne dispose que de 3 lignes, les infos fichent rapidement le camp vers le haut, notamment lors du retour au Ready. Seule solution pour contrôler ça : Utiliser un";" à la fin des commandes PRINT.

Positionner le curseur sur une colonne X

Et que faut-il faire pour afficher à un endroit spécifique sur la ligne ?

Ecartons tout de suite l'utilisation d'espaces pour atteindre la case ciblée, car ces espaces écrasent les informations situées à gauche de cette case.

On pourrait évidemment remplacer ces espaces par des codes Ascii 9 (déplacement à droite). Mais imaginez-vous en train d'écrire `PRINT CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9)CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9) CHR$(9)"OK"` seulement pour afficher "OK" au milieu de la ligne !

J'ai donc repris l'étude des variables du système.

En page zéro, l'adresse #30 contient la position du curseur sur la ligne TEXT, mais la commande POKE#30,X ne modifie pas l'affichage.

En page 2, la variable #0253 contient le numéro de la première colonne valide (et donc 0 ou 2), mais la commande POKE#0253,X ne modifie pas l'affichage. De toute façon, il me semblait farfelu de toucher à ce paramètre qui gère les 2 colonnes protégées.

Après le #0268 qui n'avait pas marché, il me semblait peu probable que modifier la variable #0269 (numéro de colonne du curseur TEXT de 0 à 39) donne des résultats. Mais j'ai quand même testé.

Divine surprise, un `POKE#0269,10:PRINT"OK"` affiche bien "OK" à la position X=10 de la première ligne de la zone TEXT. Un `POKE#0269,12:PRINT"OK"` envoie le "OK" sur la deuxième ligne. Enfin, avec un `POKE#0269,14:PRINT"OK";` on voit apparaître le hoquet sur la troisième ligne (attention à ne pas oublier le ";" final). Ceci est illustré par la recopie du bas de l'écran ci-dessous et le programme correspondant.


```
100 PAPER3:INK4
110 HIRES
120 POKE#26A,(PEEK(#26A)AND#FE):PRINT
""
130 POKE#269,10:PRINT"OK"
140 POKE#269,12:PRINT"OK"
150 POKE#269,14:PRINT"OK";
160 GET R$
```

Dernier point sensible, cette méthode ne permet pas la mise à jour automatique de l'affichage du curseur TEXT et mes essais laissent traîner des curseurs fantômes… Il est donc impératif de bloquer l'affichage du curseur en plaçant la commande suivante :

`POKE#26A,(PEEK(#26A)AND#FE):PRINT"":' CURSEUR TEXT OFF` dans le programme Basic, juste après la commande HIRES.

NB. Cette commande éteint le curseur quelle que soit son statut antérieur, alors que : `POKE#26A,(PEEK(#26A)OR#01):PRINT"":' CURSEUR TEXT ON` allume le curseur à coup sûr (dans un programme Basic complexe, l'utilisation répétée de la bascule PRINTCHR$(17) c'est-à-dire contrôle + Q est vite difficile à gérer, car on ne sait plus où on en est).

La recopie du bas de l'écran ci-dessous et le programme correspondant donnent un exemple concret de pseudo PRINT@ en mode HIRES !


```
100 PAPER3:INK4
110 HIRES
120 POKE#26A,(PEEK(#26A)AND#FE):PRINT
""
130 PR=1015:OD=2232:LU=4330
140 POKE#269, 8:PRINT"Pierre";
150 POKE#269,17:PRINT"Odette";
160 POKE#269,26:PRINT"Lucien";
170 PRINT
180 POKE#269,8:PRINT PR;
190 POKE#269,17:PRINT OD;
200 POKE#269,26:PRINT LU;
210 GET R$
```

En conclusion, n'importe laquelle des 120 cases de la zone TEXT est adressable individuellement. En fait, pas la 124e, qui n'est accessible qu'avec un POKE#BFDF,code Ascii sous peine de déclencher un scrolling. Coule, je vais pouvoir retourner au développent de mon jeu…