Mise au point Software pour Cartouches Super-Oric (9)

Quelques Notions sur les Caractères Redéfinis (2)

par André C.

Lors du dernier épisode, nous avons redéfini et affiché le 'A' de CAPS en 4 couleurs : Noir, Rouge, Vert et Jaune. Voici la suite de cette aventure…

Un autre palette ?

Nous avons défini, d'une part un caractère en 4 couleurs et d'autre part les 2 premières 'palettes' du mode Lores 1. Nous allons faire la démonstration que l'on peut afficher notre 'A' avec d'autres couleurs, celles de la 'palette' n°1 (la deuxième, œuf corse).

Nous allons donc afficher un deuxième 'A' sous celui de CAPS, en utilisant la palette n°1, c'est à dire 'poker' dans l'écran non seulement le code Ascii de 'A', mais aussi d'autres paramètres, dont le numéro de palette. Pour ce faire, il faut utiliser la commande PLOT selon les indications de Fabrice (cf. le fameux article déjà cité). Rappel de la structure du code à mettre dans l'écran :

V	H	P	Palette			n° du caractère (code Ascii)									
0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1

De droite à gauche on a le code Ascii (ici #41, on peut aller de 0 à 1023, puisque 10 bits sont disponibles), le numéro de 'palette' (ici 1, on peut aller de 0 à 7, puisque 3 bits sont disponibles), le bit de priorité (ici 1 pour être devant l'écran Hires, ce qui n'est pas important dans notre exemple, puisqu'il n'y a pas d'écran Hires), le bit H de retournement horizontal et enfin le bit V de retournement vertical (tous deux à zéro, car on veut afficher normalement le caractère). Cela donne en tout dans notre exemple #2441. Ajoutons donc une nouvelle ligne dans notre programme :

```
10 LORES 0:CLS
20 DEFCHAR65,#3355,#0,#0,#0,#0,#0,#0,#0
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF
40 PLOT 29,1,#2441
50 END
```

Et sauvons sous Lores0b.txt. Fabriquons Lores0b.tap (pensez à ajouter le mode AUTO), puis la cartouche Lores0b.swc. Voici ce qu'on obtient :

Les couleurs de la première ligne de pixel du deuxième caractère 'A' sont maintenant : Noir, Magenta, Cyan, Blanc Noir, Magenta, Cyan, Blanc. Tiens Tiens ! Bizarre Bizarre ! On aurait dû avoir Bleu, Magenta, Cyan, Blanc Bleu, Magenta, Cyan, Blanc. En fait, la couleur n°0 de chaque 'palette' Lores a un statut spécial. Elle est 'transparente'. Il n'existe pas de documentation officielle sur la Snes. Les développeurs accrédités par Nintendo en ont probablement obtenu une, mais sa diffusion n'est pas publique. Notez que nous avons la même situation avec l'Oric, puisque la documentation 'constructeur' est réduite au minimum et que tout ce qu'on sait de l'Oric est dû à l'acharnement de passionnés. La documentation 'parallèle' sur la Snes n'est pas très loquace sur la 'transparence' de la couleur n°0 de chaque 'palette' Lores. On pourrait comprendre que dans notre essai par exemple, les pixels Bleus sont devenus transparents et laissent voir ce qu'il y a derrière à savoir le Noir de PAPER ou le Noir d'un écran Hires potentiel non défini. Mais il n'en est rien ! J'ai fait de nombreux essais, tant avec Snes9x, qu'avec Zsnes, qu'avec la console Snes réelle et la résultat est le même dans tous les cas : La couleur n°0 des 'palettes' n°1 à 7 est systématiquement

remplacée par la couleur n°0 de la 'palette' n°0, c'est à dire la couleur PAPER. Fabrice, que j'ai consulté à ce sujet est d'accord et se demande si c'est délibéré ou si c'est une bogue de la Snes. Conclusion : En mode Lores 0, les quatre couleurs promises se réduisent à trois (en plus de la couleur du fond PAPER). De même en mode Lores 1, les 16 couleurs promises se réduisent à 15 (ce qui est proportionnellement moins grave).

Utilisation de deux couleurs seulement

La citation de Fabrice, que nous avons indiquée précédemment, comporte aussi cette phrase : "Si vous n'utilisez que deux couleurs dans votre dessin de caractère, vous pouvez vous contenter d'utiliser le plan 0 et donc fournir 8 octets par caractère". Voyons voir ça et modifions le programme précédent pour dessiner un damier Noir et Rouge, selon le dessin ci-dessous :

Lignes	Couleurs							
0	00	01	00	01	00	01	00	01
1	01	00	01	00	01	00	01	00
2	etc.							

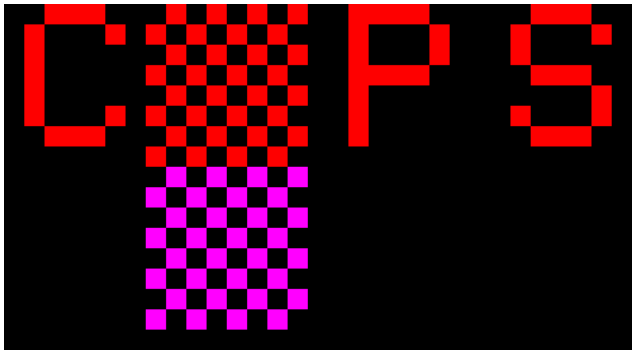
Séparons les plans 0 et 1 selon le tableau suivant :

Lignes	Plan 1 = les bits de poids fort b1								Plan 0 = Les bits de poids faible b0							
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
2	etc.															

On récupère les données de définitions de 'A', soit :
#55,#AA,#55,#AA,#55,#AA,#55,#AA
Que l'on place dans le programme :

```
10 LORES 0:CLS
20 DEFCHAR#41,#55,#AA,#55,#AA,#55,#AA,#55,#AA
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF
40 PLOT 29,1,#2441
50 END
```

Sauvons ce programme sous Lores0c.txt et fabriquons la cartouche Lores0c.swc comme indiqué précédemment. Voici ce qu'on obtient avec l'émulateur Snes9x :



La 'A' de CAPS est remplacé par un damier de 8x8 noir et rouge, tandis que le même caractère, est affiché juste en dessous avec la palette n°1 en Noir et Magenta. Ce dernier résultat est conforme à ce que nous avons déjà observé à propos de la couleur n°0 des 'palettes' Lores. On pourrait d'ailleurs en conclure qu'il serait plus simple et plus clair de définir toutes les couleurs n°0 de toutes les 'palettes' Lores de la même manière, c'est à dire avec la couleur PAPER. MAIS il y a deux paramètres à considérer :

1) La commande PAPER ne met à jour que la couleur n°0 de la première 'palette'.
2) Plusieurs modes utilisent la même palette de 256 couleurs : Lores 0, Lores1 et Hires. Une couleur 'inaccessible' en Lores 0 le devient en Lores 1 et en Hires !

Y a t-il plusieurs manière de disposer les données dans la commande DEFCHAR ?

Avec ce dernier exemple, nous venons de voir que lorsqu'on n'utilise que les couleurs n°0 et 1, les données de DEFCHAR se réduisent des octets. C'était d'ailleurs déjà le cas dans les exemples précédents où un grand nombre de lignes étaient restées de la couleur PAPER et pour lesquelles les données se réduisaient à des #0.

Reprenons le cas de l'exemple précédent. Les données étaient en fait des 'words' dont l'octet de poids fort est nul. C'est donc avec logique que nous les avons disposées sous forme de 8 valeurs indépendantes (un valeur par ligne de 8 pixel). Faisons maintenant la preuve du bien fondé de cette disposition, en regroupant ces 8 octets sous forme de 4 'words'. Cela donne le programme suivant :

```
10 LORES 0:CLS
20 DEFCHAR#41,#55AA,#55AA,#55AA,#55AA
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF
40 PLOT 29,1,#2441
50 END
```

Essayons de prédire le résultant en procédant à l'inverse de ce que nous avons fait jusqu'ici, c'est à dire en recomposant le caractère ligne par ligne à partir des octets de données. #55AA s'écrit en binaire :

Lignes	Plan 1 = les bits de poids fort b1								Plan 0 = Les bits de poids faible b0							
0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
1	etc.															

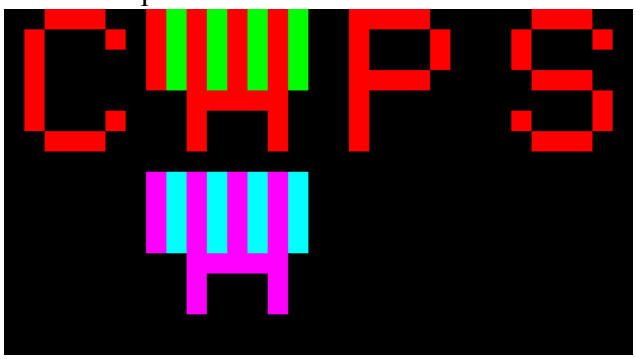
Ensuite on ré-assembles les 2 bits de chacun des 8 pixels, soit :

Lignes	Couleurs							
0	01	01	01	01	10	10	10	10
1	etc.							

Cela correspond à une alternance de pixels de couleur n°1, c'est à dire Rouges et de pixels de couleur n°2, c'est à dire Verts. On peut donc prédire que notre 'A' sera redéfini ainsi : 4 lignes de pixels Rouge, Vert, Rouge etc. et 4 lignes de la couleur PAPER. *[suite ci-contre]*

Comme vous le voyez la commande DEFCHAR à incorporé les 4 'words' dans la définition des 4 premières lignes de 'A', sans toucher aux 4 suivantes. Et c'est bien logique, car pour obtenir les 4 lignes de la couleur PAPER, nous aurions dû encore ajouter 4 fois #0. Mais cela débouche sur une découverte : Il est possible de redéfinir partiellement un caractère. Certes cette redéfinition partielle s'opère obligatoirement par le haut, mais c'est à garder dans un coin de notre mémoire.

Vérification faite, ce n'est pas tout à fait ce qu'on obtient, après avoir incorporé le programme Lores0d.tap dans la cartouche Lores0d.swc :



Pendant que nous y sommes, modifions notre dernier exemple en inversant les octets pour former les 'words'. Cela donne le programme Lores0e.tap suivant :

```
10 LORES 0:CLS
20 DEFCHAR#41,#AA55,#AA55,#AA55,#AA55:
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF
40 PLOT 29,1,#2441
50 END
```

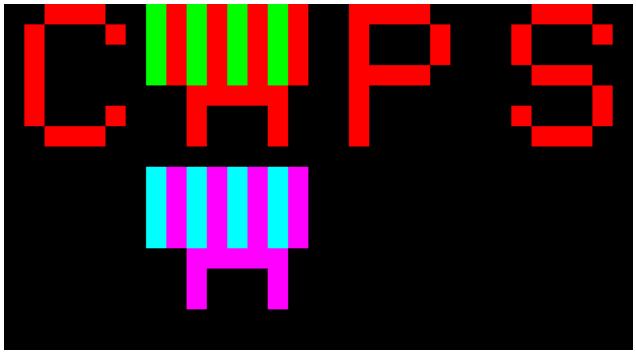
Essayons cette fois de prédire correctement le résultat. Pour les plans 1 et 0 ont a donc :

Lignes	Plan 1 = les bits de poids fort b1								Plan 0 = Les bits de poids faible b0							
0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1
1	etc.															

Ce qui donne pour les pixels :

Lignes	Couleurs							
0	10	10	10	10	01	01	01	01
1	etc.							

Cette fois on devrait obtenir 4 lignes de pixels alternés Vert, Rouge, Vert, etc. et 4 lignes non redéfinies (avec les janbes de l'ancien 'A'). Gagné ! C'est bien ce que montre la cartouche Lores0e.swc :



à suivre...