Mise au point Software pour Cartouches Super-Oric (11)

L'affichage en vidéo inverse

par André C.

Je rappelle tout d'abord que point n'est besoin de disposer d'un Super-Oric réél pour tester les exemple donnés dans cette série d'articles. Un émulateur de console Snes fait également l'affaire. Il y en existe deux, en perpétuelle évolution, qui rivalisent pour émuler la console Snes au plus près: Le Snes9x <www.snes9x.com> et le Zsnes <www.zsnes.com>.

Dans le Banc d'Essai, intitulé « Le Clavier Super-Oric de Fabrice F. », paru dans le dernier numéro, j'ai rapporté quelques-unes de mes expérimentations avec le clavier du Super-Oric. J'ai notamment raconté comment il m'a fallut un certain temps pour comprendre que le curseur n'est pas matérialisé par un bloc en vidéo inverse comme avec les Oric traditionnels, mais par un bloc clignotant.

L'Oric classique ne comporte que 128 caractères (et même en fait seulement 96, puisque les Ascii de 0 à 31 sont réservés aux attributs vidéo), le Super-Oric s'offre le luxe de 1024 caractères, tous utilisables, puisque les attributs vidéos ont été supprimés (ce qui permet d'avoir un écran sans 'trous').

L'Oric classique utilise le b7 (le bit de poids le plus fort) des caractères pour afficher ceux-ci en vidéo normale (si b7=0) ou en vidéo inverse (si b7=1). Autrement dit les Ascii de 32 à 127 sont affichés en vidéo normale et les Ascii de 128 à 255 correspondent donc en fait aux mêmes caractères du jeu de base affichés en vidéo inverse. Notez le beau gâchis pour les valeurs de 128 à 151 qui correspondent aux mêmes attributs vidéo que pour les valeurs de 0 à 32).

Il n'en va pas de même avec le Super-Oric, pour lequel ces caractères de 128 à 255 sont tous utilisables de manière autonome. Le Super-Oric offre royalement la possibilité de redéfinir des caractères vraiment indépendants de 0 à 1023 !

Au boot les 96 caractères Ascii de 32 à 127 du Super-Oric sont pré-définis de manière à émuler le jeu de base de l'Oric classique. Avec la seule différence que la matrice de chaque caractère est maintenant de 8x8 pixels au lieu de 6x8 dans l'Oric classique. Fabrice a collé le bloc de 6x6 pixels dans le bloc de 8x8 pixels en alignant à droite. Il a donc simplement ajouté deux colonnes vides (c'est à dire de la couleur du fond) à gauche. Par compatibilité avec l'Oric classique, ces caractères de bases sont définis avec les couleurs n°0 et n°1 du Super-Oric, qui correspondent donc respectivement à PAPER et INK.

Bon, trêve de discours, la vidéo inverse n'est pas présente en standard dans le Super-Oric et si vous en avez besoin, il faut l'y ajouter. C'est ce que nous allons faire aujourd'hui.

LES DIFFERENTES POSSIBILITES.

A) La première idée qui vient à l'esprit est de faire comme dans l'Oric classique, mais en redéfinissant un deuxième jeu de 96 caractères de valeur Ascii + 128.

- Avantage: on peut gérer par programmation l'utilisation de la vidéo inverse en jonglant avec le b7.
- Inconvénient: cela utilise encore 96 caractères.

Bon d'accord il en reste encore 1024-(96x2)=832. Mais la redéfinition des caractères est un exercice assez lourd (voir les mags 181 de Mai, 182 de Juin et 183-184 de Juillet-Août 2005).

B) La seconde idée est de ne pas redéfinir les caractères, mais de les afficher avec une autre palette.

-Avantage: il est plus simple de redéfinir 4 ou 16 couleurs (selon le mode LORES0 ou LORES1) que 96 caractères. Autre avantage, on peut aller plus loin dans les effets, puisqu'on peut même créer 8 palettes. Ainsi les caractères n'oscilleront pas seulement entre vidéo normale et vidéo inverse, mais seront affichables selon 8 possibilités (bonjour les fondus enchaînés par exemple).

-Inconvénient: suite à une bogue de la console SNES, la couleurs n°0 de toutes les palettes est identique à la couleur n°0 de la palette n°0 (PAPER). Autrement dit, il est inutile de chercher à utiliser la couleur n°0 des palettes n°1 à 7, elle sera de toute manière écrasée par la couleur PAPER.

Alors, notre vidéo inverse par gestion des palettes est foutue ? Non, mais il va falloir redéfinir le jeu de caractères de base en utilisant non pas les couleurs n°0 (PAPER) et 1 (INK), mais n°X et n°1. X sera donc une pseudo couleur PAPER, que vous avez intérêt à définir de la même manière que la couleur n°0 (identique à PAPER), sinon vos caractères apparaîtront sur un petit fond carré de couleur différente. En pratique, cela réduit notre gamme de couleurs à 3 ou 15 (selon le mode LORES0 ou LORES1). Le mode LORES0 (3 couleurs utilisables de manière indépendante) suffira à résoudre la majorité des problèmes courants. Rappelons que les 8x8=64 pixels de chaque caractère seront affichables dans l'une de ces 3 couleurs et ceci de manière indépendante. C'est quand même plus souple que dans l'Oric classique (2 couleurs + la contrainte des attributs vidéo en série qui crée des 'trous' dans l'écran).

Vous me direz que finalement le changement de palette n'est pas si avantageux, puisque de toute façon il faut redéfinir un jeu de caractères. Nous allons voir plus loin avec un exemple concret qu'il y a un truc simplificateur!

Outre la définition de nouveaux caractères et/ou d'un nouvelle palette, ces deux premières méthodes impliquent forcément d'écrire dans l'écran. Chaque caractère est défini sur deux octets (16 bits), comme cela a déjà été décrit par Fabrice Francès (mag n°151 pages 14 à 17) ou dans un de mes articles (mag n°182 pages 8 à 10). Les bits b0 à b9 contiennent le code Ascii (de 0 à 1023). Les bits b10 à b12 contiennent le n° de palette (de 0 à 7). Les 3 bits de poids forts b12, b14 et b15 contiennent respectivement les drapeaux de priorité, retournement horizontal et retournement vertical. Il faudra donc intervenir soit sur le b7, soit sur le b10.

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

Si nous utilisons la première méthode (inversion du b7), il faudra :

1) Lire la valeur 16 bits présente dans la case ciblée, à l'aide de la commande SCRN(X,Y) avec X compris entre 0 et 31 et Y compris entre 0 et 27.
2) Inverser le b7, comme dans l'Oric classique, sans toucher aux autres bits.
3) Ecrire le résultat à la même place, à l'aide de la commande PLOT,X,Y,'WORD'

Si nous utilisons la seconde méthode (changement de palette), il faudra :

1) Lire la valeur 16 bit comme ci-dessus.
2) Modifier le n° de palette, c'est à dire les bits b10 à b12. Par exemple, pour passer de la palette n°0 (000) à la palette n°1 (001) et réciproquement, il suffira d'inverser b10. Si on envisage d'utiliser plus de 2 palettes (pour des effets spéciaux), il faudra aussi modifier b11 et peut-être b12.
3) Ecrire le résultat comme ci-dessus.

C) La troisième possibilité serait de redéfinir en bloc la palette n°0. Dans l'Oric classique, on peut ainsi permuter les valeurs de PAPER et INK.

-Avantage : c'est simple car il n'y a pas de caractères, ni de palette à définir. Pour permuter PAPER et INK il suffit de copier PAPER dans une variable, puis INK dans PAPER et enfin la variable dans INK.

-Inconvénient : tout l'écran est affecté (c'est rédhibitoire dans la plupart des cas).

EXEMPLE DE REDEFINITION DE CARACTERE

Pour plus de facilité, nous allons nous contenter de redéfinir un seul caractère, la lettre 'A', que nous allons dessiner en couleur n°3 sur fond couleur n°2 au lieu d'une lettre A en couleur n°1 (INK) sur fond couleur n°0 (PAPER). Les 96 caractères définis par défaut dans le Super-Oric sont exactement ceux de l'Oric classique. Comme la matrice des caractères Super-Oric comporte du coté gauche deux colonnes de plus que celle de l'Oric classique (8x8 au lieu de 6x8) et que Fabrice Francès a laissé ces deux colonnes en couleur de fond, les caractères Super-Oric ont la même définition que les caractères classiques.

Par exemple la lettre 'A' est définie dans les deux cas (Oric ou Super-Oric) par les huit octets #08 #14 #22 #22 #3E #22 #22 #00 (un octet par ligne, soit 8 octets). Reportez-vous si besoin au mag 181 pages 9 à 12 qui explique comment les caractères sont redéfinis.

En fait en LORES0 chaque ligne n'est pas définie par un octet (8 bits) mais par un 'Word' (16 bits). Mais il se trouve que dans le cas présent, les couleurs de pixel utilisées sont la couleur n°0 (00) et n°1 (01) dont le bit de poids fort est à zéro dans les deux cas. Or ces bits de poids forts sont répercutés dans l'octet de poids fort du 'Word'. En fait les 8 'Words' de définition sont #0008 #0014 #0022 #0022 #003E #0022 #0022 #0000 ce qui est bel et bien la même chose que #8 #14 #22 #22 #3E #22 #22 #0.

Oups, de chance pour nous! Le 'A' dessiné en couleur n°3 (11) sur fond de couleur n°2 (10), est de forme identique, sauf que les bits de poids forts des couleurs sont toujours à '1', donc les octets de poids forts des 'Words' seront toujours à #FF. Ce qui donnent les 8 'Words' suivants: #FF08 #FF14 #FF22 #FF22 #FF3E #FF22 #FF22 #FF00.

Vous voyez que pour redéfinir l'ensemble du jeu de caractère normal Oric en couleurs n°3 sur fond de couleur n°2, il suffit de prendre les octets de définitions dans la Rom (Oric ou Super-Oric de #FC78 à #FF77) ou dans la Ram (Oric de #B500 à #B7FF) et d'ajouter #FF devant chaque octet. On trouve la définition de chacun de ces 96 caractères dans tous les bons livres, par exemple dans 'L'Oric à Nu' de Fabrice Broche, pages 384 à 389.

EXEMPLE DE COULEURS INVERSES

Dans le système RVB, chaque couleur est caractérisée par les 3 composantes Rouge, Vert et Bleue, chacune définie par un octet (8 bits) et pouvant donc prendre une valeur de 0 à 255. Pour obtenir la couleur inverse, il faut inverser tous les bits de ces 3 composantes.

En pratique on peut utiliser le truc suivant. Soit donc la couleur définie par les 3 octets r, v et b. La couleur inverse sera définie par les valeurs 255-r, 255-v et 255-b. Ainsi le cyan est défini par les 3 octets 0, 255 et 255. Son inverse est défini par 255, 0 et 0 (Rouge). Ou encore, l'Orange (255, 127, 0) a pour inverse une sorte de Bleu clair (0,128,255).

De manière analogue, les couleurs classiques de Oric sont caractérisées par les 3 composantes Bleu, Vert et Rouge (observez l'ordre qui est différent) chacune définie par un bit: un bit Bleu, un bit Vert et un bit Rouge, ce qui permet des valeurs de 0 à 7 (000 à 111). On peut vérifier que Noir=000, Rouge=001, Vert=010, Jaune=011, Bleu=100, Magenta=101, Cyan=110 et Blanc=111. L'inverse d'une couleur Oric est obtenu en inversant chacun des 3 bits.

En pratique on peut utiliser l'opération suivante: n° de couleur inverse = 7 - n° de couleur normale. Ainsi le Blanc (7=111) et le noir (0=000) sont inverses. Idem pour les couples rouge/Cyan (1=001/6=110), vert/magenta (2=010/5=101) et jaune/bleu (3=011/4=100).

Même système pour le super Oric. Chaque couleur est caractérisée par les 3 composantes Bleu, Vert et Rouge (même ordre que pour l'Oric), chacune définie sur 5 bits. La couleur inverse est obtenue par inversion des 15 bits. Par exemple, pour une couleur définie par le 'Word' #695A (0110 1001 0101 1010) on obtiendra la couleur complémentaire en inversant les 15 bits significatifs (le bit de poids le plus fort ne sert à rien), ce qui donne 0001 0110 1010 0101 soit #16A5.

Le mag n°183 page 12 donne quelques exemples de couleurs inverses. Pour nos essais d'aujourd'hui, choisissons par exemple la première des paires indiquées, soit Orange (#021F) et Bleu foncé (#7DE0).

EXEMPLE PAR GESTION DU b7

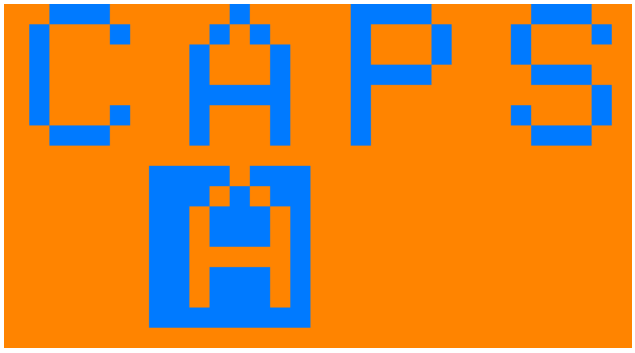
Nous allons donc modifier la lettre A (Ascii 65=#41) et son homologue en vidéo inverse (Ascii 65+128= 193=#C1). Dans un premier temps, pour simplifier le programme, nous afficherons un 'A' en vidéo inverse juste en dessous du 'A' normal de CAPS. Pour voir les voir tous les 2 en même temps.

```
10 LORES 0:CLS
20 DEF CHAR #C1,#FF08,#FF14,#FF22,#FF22
,#FF3E,#FF22,#FF22,#FF00:'A+180
30 DEF INK0,#21F,#7DE0,#7DE0,#21F
:'Orange,Vert,Vert,Orange
40 C=SCRN(29,0):C=C OR #80:PLOT29,1,C
50 END
```

Le OR #80 (00001000 00000000 en binaire) de la ligne 40 force le b7 à 1, ce qui permet d'obtenir le code Ascii du caractère en vidéo inverse.

Après avoir sauvé ce programme au format texte de base (vidinv01.txt), il suffit de le convertir en fichier 'tap' à l'aide de Text2bas.exe (j'ai utilisé la version du 11/09/2002 à 11:07 de 116 530 octets) à l'aide de la commande 'txt2bas -v1 vidinv01.txt vidinv01.bas' dans une fenêtre DOS. Ensuite, à l'aide d'un éditeur hexadécimal, placer une valeur différente de zéro à la place du bit d'offset 7 de ce fichier 'tap' afin de mettre ce programme en exécution AUTO. Puis coller ce 'tap' à la fin de la Rom Super-Oric. J'ai utilisé le fichier Oric3.bin de CRC-32 = 7FE53347, que vous trouverez dans la disquette trimestrielle au format PC de septembre 2005. Enfin complétez avec des #00 jusqu'à obtenir un fichier de #20000 octets (pour une éventuelle utilisation dans une mémoire flash).

Il ne vous reste plus qu'à tester. La figure ci-dessous montre ce que j'ai obtenu dans le coin en haut à droite de l'écran avec l'émulateur Snes9x.

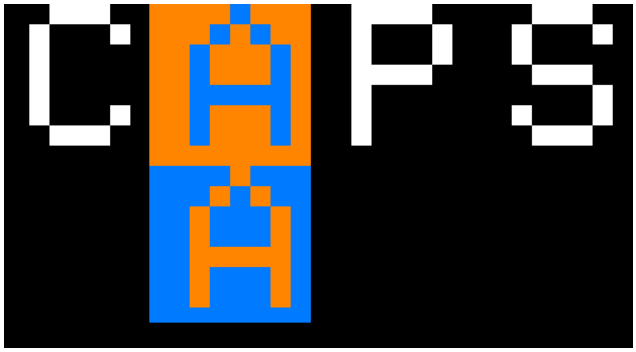


EXEMPLE PAR CHANGEMENT DE PALETTE

Comme ci-dessus, pour simplifier le programme, nous afficherons un 'A' en vidéo inverse juste en dessous du 'A' de CAPS.

```
10 LORES 0:CLS
20 DEF CHAR #C1,#FF08,#FF14,#FF22,#FF22,
#FF3E,#FF22,#FF22,#FF00:'A+180
30 DEF INK2,#21F,#7DE0:'Orange,Vert
couleurs 2 et 3 de la palette 0
40 DEF INK6,#7DE0,#21F:'Vert,Orange
couleurs 2 et 3 de la palette 1
50 PLOT29,0,#20C1:'Remplace le A de CAPS
60 PLOT29,1,#24C1:'En dessous du A de
CAPS
70 END
```

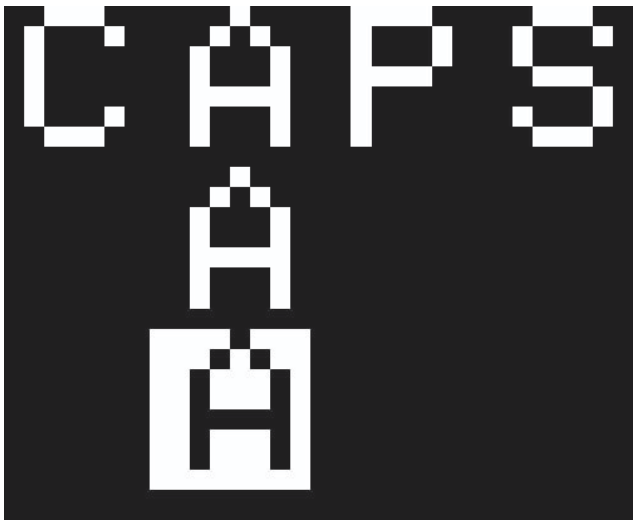
Les couleurs n°0 et n°1 de la palette n°0 (PAPER noir et INK blanc) restent celle définie par défaut lors du boot. Les couleurs n°0 et 1 de la palette n°1 ne sont pas définies (mais cela n'a aucune importance). On procède comme précédemment et on obtient les fichiers vidinv02.txt, cidinv02.tap et vidinv02.swc. Le résultat, obtenu avec l'émulateur de SNES est visible ci-dessous. Le message CAPS est en blanc (INK) sur noir (PAPER) sauf le A qui est en Bleu sur Orange. Le 2e A est en Orange sur Bleu.



Pas convaincus? Essayons le programme suivant (vidinv03.txt):

```
10 LORES 0:CLS
20 DEF CHAR #C1,#FF08,#FF14,#FF22,#FF22,
#FF3E,#FF22,#FF22,#FF00:'A+180
30 DEF INK2,#0,#7FFF:'Noir et Blanc,
couleurs 2 et 3 de la palette 0
40 DEF INK6,#7FFF,#0:'Blanc et Noir
couleurs 2 et 3 de la palette 1
50 PLOT29,1,#20C1:'En dessous du A de
CAPS
60 PLOT29,2,#24C1:'Encore en dessous
70 END
```

Cette fois nous affichons un pseudo A (Ascii #C1) en couleurs 2 et 3 de la palette n°0 en dessous du A de CAPS. Il est identique au A normal. Puis encore en dessous le même pseudo A (Ascii #C1) en couleurs 2 et 3 de la palette n°1 (figure ci-dessous).



PERFECTIONNEMENTS

Pour changer le b7 d'un caractère affiché à l'écran, on peut utiliser les commandes:

```
C=SCRN(X,Y):IFCAND#80THENPLOT
X,Y,CAND#3F7FELSEPLOTX,Y,COR#80
```

Pour changer la palette d'un caractère affiché à l'écran, on peut utiliser les commandes:

```
C=SCRN(X,Y):IFCAND#400THENPLOT
X,Y,CAND#3BFFELSEPLOTX,Y,COR#400
```

Quelques explications:

1) Test IF. Si le résultat du AND est différent de zéro, THEN sera exécuté. Si le résultat du AND est nul, ELSE sera exécuté.

2) On ne peut comprendre le fonctionnement des opérateurs AND et OR que si on écrit les valeurs traitées en binaire. En effet ces opérateurs travaille bit par bit. Voici leurs tables de vérité:

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

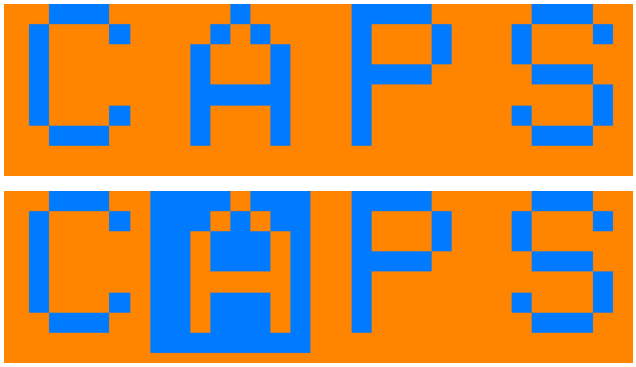
'C AND #0' donnerait un résultat nul à tous les coups quelque soit la valeur des 8 bits de C. 'C AND #80' ne donnera un résultat nul que si le b7 est nul, sinon il donnera #80 comme résultat et THEN sera exécuté. Idem pour 'C AND #400' avec le b10.

3) Pour forcer spécifiquement un bit à 'zéro' sans affecter les autres bits, il faut utiliser AND et un 'masque' dans lequel ce bit est à 'zéro' et tous les autres sont à 'un'. Pour forcer spécifiquement un bit à 'un' il faut utiliser OR et un 'masque' dans lequel ce bit est à 'un' et tous les autres à 'zéro'.

Notre premier programme montrant le basculement de caractère entre vidéo normale et vidéo inverse par basculement de b7 se transforme et devient (vidinv04.txt):

```
10 LORES 0:CLS
20 DEF CHAR #C1,#FF08,#FF14,#FF22,#FF22
,#FF3E,#FF22,#FF22,#FF00:'A+180
30 DEF INK0,#21F,#7DE0,#7DE0,#21F
:'Orange,Vert,Vert,Orange
40 C=SCRN(29,0):IFCAND#80THENPLOT
29,0,CAND#3F7FELSEPLOT29,0,COR#80
50 WAIT 100: GOTO 40
```

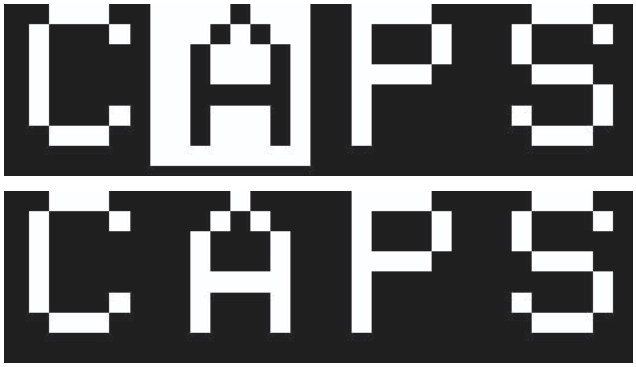
Le A de CAPS alterne régulièrement en vidéo normale et en vidéo inverse. La figure ci-dessous montre ces deux états.



De même, le programme montrant le changement de palette devient (vidinv05.txt):

```
10 LORES 0:CLS
20 DEF CHAR #C1,#FF08,#FF14,#FF22,#FF22
,#FF3E,#FF22,#FF22,#FF00:'A+180
30 DEF INK2,#0,#7FFF:'Noir et Blanc,
couleurs 2 et 3 de la palette 0
40 DEF INK6,#7FFF,#0:'Blanc et Noir
couleurs 2 et 3 de la palette 1
50 PLOT29,0,#20C1:'Remplace le A en
couleur 0 et 1 par notre A en couleur 2
et 3
60 C=SCRN(29,0):IFCAND#400THENPLOT
29,0,CAND#3BFFELSEPLOT29,0,COR#400
70 WAIT 100: GOTO 50
```

Ici encore, le A de CAPS alterne régulièrement en vidéo normale et en vidéo inverse. La figure ci-dessous montre ces deux états.



Voilà, vous savez tout sur la vidéo inverse et le Super-Oric.

A suivre pour de nouvelles aventures…