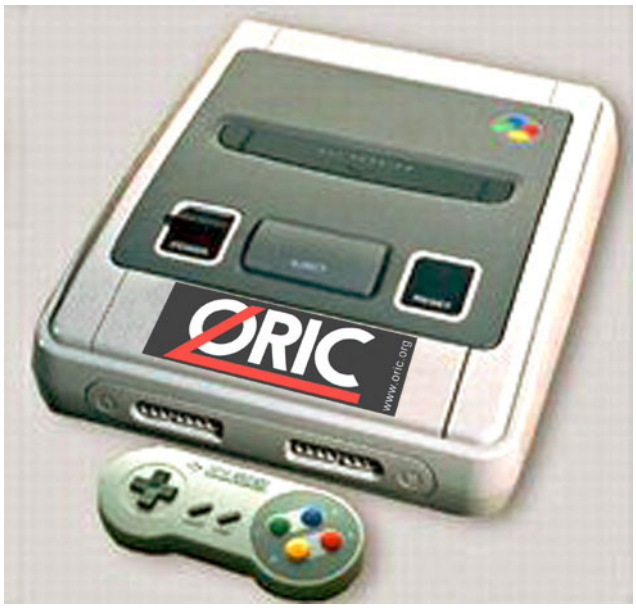
# Mise au point Software pour Cartouches Super-Oric (8)

## Quelques Notions sur les Caractères Redéfinis (1)

*par André C.*




### Un peu de Publicité...

Ce n'est pas sans émotion que je me souviens de mon premier PRINT « TOTO » sur Oric-1. Que d'heures et de jours passés à expérimenter le fonctionnement de cette machine ! Je me souviens surtout de ma fascination pour la redéfinition des caractères, pour les attributs de couleur, clignotement, double hauteur etc. Puis est venu l'Atmos. Je fus choqué par les couleurs rouge et noire du boîtier, mais bientôt conquis par le toucher du clavier. Et dans un deuxième temps par le Microdisc. Enfin est venu le Telestrat avec de nouvelles possibilités. Ce sont ses merveilleuses cartouches qui m'ont le plus séduit, car j'ai tout de suite compris la souplesse du système. Nous avons tous fait l'expérience inoubliable de la prise en main d'un Oric. Avec cette rubrique, mon but est de vous inciter à tenter l'expérience du Super-Oric. Si vous êtes sceptiques, sachez qu'il ne vous en coûtera rien, puisque tous vos essais peuvent se faire avec un émulateur (www.snes9x.com ou www.zsnes.com). Mais le matériel réel n'est pas mal non plus. En effet, pour cette quatrième génération d'Oric, vous aurez dans une même machine, outre la couleur gris-bleu de l'Oric-1, la Rom de l'Atmos, les cartouches du Telestrat, la formidable puissance du 65816 et de nouvelles possibilités ahurissantes !

### Petit rappel des épisodes précédents

Ne manquez pas de garder sous le coude l'article de Fabrice F., dans le Ceo-Mag n° 151 pages 14-17. En voici un petit résumé, limité à la structure générale de notre nouvelle machine. Le Super-Oric, c'est un écran Hires de 256x224 points (largeur x hauteur). Chaque point peut spécifiquement être affiché dans l'une des 256 couleurs de la palette à choisir parmi 32768 possibilités ! A cet écran Hires se superpose un écran Texte de 28 lignes de 32 caractères. Et comble du bonheur, 1024 caractères redéfinissables sont utilisables, chacun étant caractérisé par une matrice de 8x8 points (remarque 28x8=224 et 32x8=256, on retrouve la taille de l'écran Hires). Ici encore la couleur de chaque point est indépendante de celle des points voisins (voir plus loin). La mémoire interne est doublée : Aux 64 ko habituels s'ajoutent encore les 64 autres d'une deuxième banque pour les écrans Texte et Hires, la définition des caractères et celle des 256 couleurs. Le résultat est que la première banque de 64 Ko est débarrassée de tout ça et que, outre les habituels 16 Ko de la Rom, il reste 48 Ko utilisables !

Au cours des épisodes précédents, nous avons vu :
1) La structure d'une cartouche Super-Oric et du fichier émulateur correspondant (n°173 Sept. 2004).
2) La mise en place d'un petit programme Basic et l'utilisation du joypad (n°174 Octobre 2004).
3) La confection et la mise ne place d'une image dans l'écran Hires (n°176 Decembre 2004).
4) A titre d'information, quelques notions sur la structure de l'écran Hires (n°177 Janvier 2005).
5) L'écran Texte et quelques manipulations de base, dont les commandes SCRN(X,Y) et PLOT X,Y,WORD utilisées à la place de PEEK et POKE dans la mémoire de l'écran (n°178 Février 2005).
6 & 7) L'utilisation des 256 couleurs et leur définition parmi 32768 possibilités (n° 179 Mars 2005 et 180 Avril 2005).

### Bref au boulot !

Au menu du jour, une petite initiation à la redéfinition des caractères. Disons tout de suite, que le Super-Oric va vous décoiffer ! Avant d'attaquer la redéfinition des caractères, il est bon

de faire quelques rappels plus précis sur la structure de l'écran texte.

Pour les 3 générations précédentes d'Oric, l'écran était formé d'une suite de 1120 octets localisés en mémoire de #BB80 à #BFDF et correspondant à 28 lignes de 40 caractères. Chaque octet, selon sa valeur représente soit un attribut, soit un code Ascii. Avec cette 4e génération d'Oric, l'écran Texte est formé d'une suite de 1792 octets situé en mémoire de #F000 à F6FF dans la deuxième banque et correspondant à 28 lignes de 32 caractères, mais avec 2 octets pour chaque caractère au lieu d'un seul. Outre le n° Ascii du caractère (de 0 à 1023, c'est à dire de #0 à #3FF, ce qui se code sur 10 bits), ces deux octets contiennent aussi le n° de palette (de 0 à 7, ce qui se code sur 3 bits). Il reste 3 bits qui servent à la priorité (1=devant l'écran Hires et 0=derrière l'écran Hires), le retournement horizontal et le retournement vertical. Le détail de la structure de ces 2 octets a été donné par Fabrice dans l'article déjà cité.

**Définition d'un caractère.**

Définir un caractère, c'est attribuer à un code Ascii (de #0 à #3FF) une suite d'octets indiquant la couleur de chacun de ses 64 points. Cela se fait très simplement avec la commande DEF CHAR n, word0, word1, word2… où n désigne le code Ascii et word représente une paire d'octet. Le nombre de 'words' nécessaire dépend du mode Lores 0 ou Lores1. Avec le mode Lores 0 (mode par défaut), chacun des 64 points d'un caractère sera affiché avec une des 4 couleurs de n°0 à 3 (codage sur 2 bits). Il faudra donc 64x2= 128 bits pour y parvenir, soit 16 octets, soit 8 words. Avec le mode Lores 1, chacun des 64 points d'un caractère sera affiché avec une des 16 couleurs de n°0 à 15 (codage sur 4 bits). Il faudra donc 64x4= 256 bits pour y parvenir, soit 32 octets, soit 16 'words'. Pour aujourd'hui, nous nous en tiendrons au mode par défaut.

Notez bien un point important : La définition de chaque caractère consiste à donner un n° de couleur à chacun de ses 64 points. Ensuite, lors de l'affichage, le numéro de palette à utilisé est indiqué et placé dans la mémoire de l'écran. Ceci veut dire qu'un même caractère peut être affiché avec plusieurs ensembles de couleurs. Nous reviendrons la-dessus pour fabriquer des caractères clignotants 'on/off' et même scintillants en plusieurs couleurs (une superbe nouveauté).

**Structure des données à mettre dans la commande DEF CHAR**

Comment sont organisés les 8 'words' qui caractérisent un caractère en mode Lores 0 ? Vous êtes sans doute habitués à mettre les bits à 0 ou à 1 selon que vous voulez les afficher avec la couleur PAPER ou INK (pour faire bref). Maintenant il faudra mettre chaque point à 00, 01, 10 ou 11, selon la couleur d'affichage souhaitée pour ce point. Je cite Fabrice : " En mode LORES 0, chaque caractère est défini par 8 valeurs (une par ligne de 8 pixels) : L'octet de poids faible contient le plan 0, l'octet de poids fort le plan 1. 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 ".

Cela semble un peu obscur. Reprenons l'exemple de l'Oric classique : Nous avions l'habitude de dessiner une grille de 6x8 points et de dessiner directement le caractère en mettant des 1 dans les cases à définir avec la couleur INK, tandis que les autres restaient à 0 (couleur PAPER). Ça marche de manière simple pour une définition sur un bit. Mais maintenant, les points peuvent prendre 4 valeurs (couleurs de 0 à 3, c'est à dire de 00 à 11 en binaire) ; il faut donc 2 bits au lieu d'un seul. Alors, on superpose deux grilles de 8x8 points. Pour chaque point, dans la première grille dite 'plan 0' on écrira le b0 (bit de poids faible) de la couleur et dans la seconde dite 'plan 1' on écrira le b1 (bit de poids fort). Et ainsi de suite pour les 8 points de la ligne et pour les 8 lignes. Ensuite il suffira de mettre côte à côte les deux grilles, à gauche celle du plan 1 et à droite celle du plan 0. Pour chaque ligne, on lira les 8 bits qui constitueront l'octet de poids fort, puis les 8 bits qui constitueront l'octet de poids faible du premier 'word' de la ligne des données de DEF CHAR. Idem pour la deuxième ligne qui donnera le deuxième 'word' etc. pour avoir les 8 'words' correspondant aux 8 lignes du caractère.

**Voyons ce que cela donne sur un exemple**

Nous allons redéfinir le caractère 'A' du message CAPS affiché en haut à droite de l'écran. Pour cette redéfinition, nous utiliserons 4 couleurs de n° 0 à 3 (00 à 11 en binaire). Pour faire simple, nous nous contenterons de définir les 8 pixels de la première ligne et d'attribuer la couleur n°0 à tous les pixels des autres lignes.

Traçons un tableau de 8 sur 8 cases, dont nous ne voyons ci-dessous que les deux premières lignes, car les lignes suivantes sont identiques à la deuxième.

Dans ces cases, inscrivons en binaire le n° de couleur souhaité. Pour la première ligne (de n°0 comme toujours en informatique) nous répéterons deux fois les quatre n° possibles dans cet exemple (soit 00, 01, 10 et 11 en binaire). Pour la seconde ligne (et toutes les suivantes) nous remplirons toutes les cases avec le n°0 (00 en binaire). Voilà ce que ça donne :

| Lignes | Couleurs | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 00 | 01 | 10 | 11 | 00 | 01 | 10 | 11 |
| 1 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 2 | etc. | | | | | | | |

Traçons maintenant deux tableaux, que nous appellerons ‘Plan 1’ et ‘Plan 0’, toujours de 8x8 cases. Dans le tableau ‘Plan 1’ reportons les bits de poids fort b1 des n° de couleurs et dans le tableau ‘Plan 0’ reportons les bits de poids faible b0. Voilà ce que ça donne :

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

On peut facilement lire les bits par groupe de 4, ce qui donne pour la première ligne : 0011 0011 0101 0101. La valeur hexadécimale correspondant est apparente : #3355. Pour la deuxième ligne (et les 6 suivantes) on obtient #0000. La ligne Basic permettant de définir cette nouvelle lettre ‘A’ sera :

DEFCHAR65,#3355,#0,#0,#0,#0,#0,#0,#0

Par souci de différentiation, on a écrit le code Ascii (65) en décimal et les données de redéfinition en hexadécimal, mais on peut évidemment mettre tout en décimal ou tout en hexadécimal.

**Et les couleurs ?**

Dans la définition de notre caractère, nous avons indiqué quatre couleurs de 0 à 3. Ce n'est pas très parlant. On fait cela dépendra de la palette qui sera utilisée pour les afficher. Nous avons déjà vu :
1) Qu'il est possible de définir 256 couleurs à choisir parmi 32768 définitions, au moyen de la commande DEFINK.
2) Que la couleur n°0 est celle de PAPER et la couleur n°1 est celle de INK. Ces deux couleurs peuvent être redéfinie non seulement à l'aide de DEFINK, mais aussi avec les commandes PAPER et INK.
3) Que la palette de 256 couleurs est utilisée pour l'écran Hires. Cependant il est possible de n'utiliser pour l'écran Hires qu'une partie de ces 256 couleurs (par exemple les 128 dernières couleurs de la palette).
4) Que le début de la palette de 256 couleurs est aussi utilisée pour l'écran Texte. Selon le mode Lores0 ou Lores1, les 32 ou les 128 premières couleurs de cette palette sont alors respectivement utilisables.
5) Par anticipation sur ce que nous allons voir plus tard, disons pendant qu'on y est que 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.

Dans le cas du mode Lores 0 (caractères définis en 4 couleurs), la zone de 32 octets peut être considérée comme divisible en 8 'palettes' de 4 couleurs. Le fait d'utiliser le même mot 'palette' prête un peu à confusion, mais quand on s'exprime dans un contexte 'Lores 0' il faut savoir qu'on dispose de 8 'palettes' (de n°0 à 7) de quatre couleurs.

Par analogie, dans le cas du mode Lores 1 on dispose de 8 'palettes' (de n°0 à n°7) de 16 couleurs chacune.

Complexe ? Pas du tout ! Donnons un exemple concret. Voici le début d'une palette de 256 couleurs :

#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF,
#21F,#3F0,#3C1F,#1A9F,#7C0F,#4210,#300,#6318
etc. Ces 16 définitions correspondent aux 8 couleurs classiques de l'Oric (Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc), suivies de 8 nouvelles couleurs (Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair).

En mode Lores 0, on disposera des palettes suivantes :

Palette n°0 avec les couleurs n°0 (Noir), n°1 (Rouge), n°2 (Vert) et n°3 (Jaune).

Palette n°1 avec les couleurs n°0 (Bleu), n°1 (Magenta), n°2 (Cyan) et n°3 (Blanc).

Palette n°2 avec les couleurs n°0 (Orange), n°1 (Vert Pomme), n°2 (Fushia) et n°3 (Jaune Or).

Palette n°3 avec les couleurs n°0 (Violet), n°1 (Gris Foncé), n°2 (Vert Foncé) et n°3 (Gris Clair).

Palette n°4, Palette n°5, Palette n°6 et Palette n°7 (pas définies dans notre exemple).

En mode Lores 1, on disposera des palettes suivantes :

Palette n°0 avec les couleurs n°0 (Noir), n°1 (Rouge), n°2 (Vert), n°3 (Jaune), n°4 (Bleu), n°5 (Magenta), n°6 (Cyan), n°7 (Blanc), n°8 (Orange), n°9 (Vert Pomme), n°10 (Fushia), n°11 (Jaune Or), n°12 (Violet), n°13 (Gris Foncé), n°14 (Vert Foncé) et n°15 (Gris Clair).

Palettes n°1 à 7 (pas définies dans notre exemple).

**Petit essai pratique**

Bon, tout ça c'est bien joli, mais on voudrait bien les voir ce caractère 'A' que nous avons redéfini et dont les pixels peuvent soit disant avoir des couleurs indépendantes !
Notre premier essai sera le suivant :

```
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 END
```

Nous retrouvons à la ligne 20 les données de redéfinition établies précédemment. Et le ligne 30 redéfinit 8 couleurs, en partant de la couleur n°0 (premier paramètre). Nous avons déjà vu (Ceo-Mag n°179) que les valeurs indiquées correspondent aux 8 couleurs classiques de l'Oric (Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan et Blanc). En fait, nous n'avions besoin que de 4 couleurs pour ce premier exemple. Mais il faut savoir investir pour l'avenir !

**Petite parenthèse importante :** Nous verrons plus tard que la commande DEFINK dépasse les 80 caractères autorisés par le tampon clavier de l'Oric classique. Donc nous sommes amenés à changer de méthode et à préparer nos petits programmes Basic non plus sur un Oric, comme nous l'avons fait jusqu'ici (en fait sous Euphoric), mais avec un éditeur de texte Ascii. Il suffira ensuite de mouliner ce texte Ascii avec l'utilitaire de Fabrice Francès (**Txt2bas.exe**, version du 11.09.2002 de taille 116 530 octets). En effet cet utilitaire accepte jusqu'à 256 caractères par ligne. Donc, après avoir tapé notre programme, sauvons-le sous le nom Lores0a.txt (attention à bien sauver en mode 'texte seul'). Puis moulinons-le avec la commande :

```
txt2bas -v1 lores0a.txt lores0a.tap
```

Note : Cet utilitaire marche correctement dans une fenêtre Dos sous Windows, même avec W2000.

Ensuite procédons comme d'habitude : A l'aide d'un éditeur hexadécimal, mettons le fichier .tap obtenu en mode AUTO (le bit d'offset 7 doit être non nul). Puis concaténons le fichier ORIC2.OBJ (checksum #80E9) de Fabrice suivit de Lores0a.tap et enfin complétons avec des #00 jusqu'à obtenir une taille totale de #40000 octets (256Ko). Note : Le fichier ORIC2.OBJ de Fabrice est la dernière version du Basic 1.1 'Super-Oric'. Dans cette version Fabrice a incorporé le mode Lores 1 et les sons, dont nous aurons bientôt besoin. Mais vous pouvez utiliser une des versions précédentes, car, pour aujourd'hui, nous ne traiterons que du mode Lores 0.

Sauvez votre 'cartouche' Super-Oric' sous le nom Lores0a.swc. Si vous avez précédé comme moi, elle doit avoir une checksum de #A495. Testez votre cartouche avec Snes9X. On peut aussi utiliser Zsnes, mais il faut chercher manuellement la bonne configuration vidéo, sinon la recopie d'écran mangera la première rangée de pixels (qui est justement celle qui nous intéresse aujourd'hui). Sur mon système, je doit proscrire l'utilisation du paramètre 'D' (pour 'Allows 2XSAI, Hires, etc.). Sinon la résolution à utiliser dépend de l'écran de votre PC (il faut rester largement en deçà du maximum).

Voici une recopie d'écran de ce que j'ai obtenu ou plutôt un agrandissement du coin en haut à droite. On voit d'abord que les couleurs PAPER et INK sont bien respectivement 'Noir' et 'Rouge', comme prévu (couleurs n°0 et 1 de la ligne 30 de notre programme). Puis la lettre 'A' de CAPS est réduite à sa première ligne de 8 pixels qui sont bien Noir, Rouge, Vert, Jaune Noir, Rouge, Vert, Jaune comme planifié. Evidemment les deux pixels Noirs ne se distinguent pas du fond, lui aussi Noir.

Deulignes

# Deulignes: Scrollings Basic

*Par Simon G.*

Envie de voir un petit effet rapide ? Voici deux scrollings en Basic. Ok, ils ne font pas tout à fait deux lignes... Mais par contre, ils font bien mal aux yeux :-)

**Scrolling vertical**

```
10 CLS
12 PLOT5,10,«SIMON'S FANTASTIC BASIC SCROLLER»
20 FOR I=0 TO 7
30 POKE 46336+I,255
35 POKE 46336+I+7,0
40 POKE 46335+I,0
50 NEXT
60 GOTO 20
```

**Scrolling horizontal**

```
10 CLS
12 PLOT5,10,«SIMON'S FANTASTIC BASIC SCROLLER»
25 J=1:POKE 46336,1
27 FOR K = 2 TO 6
29 J=J*2
30 POKE 46336,J
34 NEXT
60 GOTO 25
```

Le principe est simple : On redéfinit, en boucle, le caractère espace qui est présent un peu partout sur l'écran...

**In English :** Here are two short scrolling programs, if you're bored and want to hurt your eyes. The idea is simple: a loop redefines constantly the space char.