La Rom Basic 0.01 à nu

Ce qui suit est un pseudo-fichier source de la Rom Basic 0.01 (extraite de la Rom Super-Oric : Base1.bin, présente dans le toute première cartouche Super-Oric : Demo Graphique). Pour obtenir un fichier source opérationnel, il faudrait retirer les adresses et les codes Hexadécimaux (deux premières colonnes) et le plus gros des commentaires. Attention, lorsqu'un commentaire court sur plusieurs lignes, seule la première ligne comporte un point-virgule (indiquant une remarque). Si vous voulez conserver ces commentaires longs, il faudra ajouter des points-virgules au début de chaque ligne du vrai fichier source.

Ce pseudo-fichier source a été élaboré en s'inspirant du fichier source de la Rom Basic 1.46 et en y reportant les différences entre la Rom v1.1 (Basic 11B.rom) et les #4000 octets de la partie Basic récupérée dans la Rom Super-Oric Base1.bin (ou v0.01). Quelques modifications sont dues à des déplacements de routines, donc de la valeur des labels. Il n'y a donc rien à faire, l'assembleur s'en chargera.

Cette version 0.01 a une valeur historique énorme, car elle représente une des toutes premières tentatives d'adaptation de la Rom Basic 1.1 à la Snes pour tirer le meilleur parti de celle-ci, en s'écartant au minimum de l'esprit Oric. Le fait que la commande SHOW apparaisse dans le premier article sur le Super-Oric et dans le listing de la démo graphique (tous deux fournis par Fabrice), alors que la Rom Base1.bin utilise REFRESH, laisse penser que cette Rom n'est peut-être pas la toute première. Mais c'est la première dont nous disposons. En outre, c'est la seule qui affiche correctement les images Hires...

Le fichier proprement dit:

```
; ******************************************************************************
; Dump de la ROM 1.1b pour pouvoir patcher par-dessus. La Rom Basic11B.rom sert de base de travail
pour élaborer une version compatible Snes.

                        org $C000       ; Début de la Rom Basic
        db      ; etc... copier ici tous les datas comme dans le fichier source 1.46
; La Rom Basic est bien sûr conçue pour l'espace mémoire $C000 à $FFFF. On note toutefois que pour
la version 1.46 (numéro de 4 caractères) Fabrice a dû modifier le message de copyright, car 1.1 ne
comporte que 3 caractères. Or, dans Base1.bin, le copyright est encore inchangé (1.1). Il faut donc
revenir à la version originale et rétablir les lignes suivantes:

                        org $EDA0
        db $44,$45,$44,$20,$42,$41,$53,$49,$43,$20,$56,$31,$2e,$31,$0d,$0a ; DED BASIC V1.1--
        db $60,$20,$31,$39,$38,$33,$20,$54,$41,$4e,$47,$45,$52,$49,$4e,$45 ; © 1983 TANGERINE
        db $0d,$0a,$00,$00,$a2,$00,$a0,$00,$c4,$10,$d0,$04,$e4,$11,$f0,$0f ; --.. etc.

; ******************************************************************************
; Version: copiée depuis oric2.s

;                       org $EDAB       ; Dans le copyright
; EDAB 30 2E 30 30      string "0.00"   ; sur la v1.46
; Exemple de modification, présente sur la v1.46, qu'il suffit d'annuler pour revenir à 1.1

; ******************************************************************************
; Modifications dans la table des adresses des commandes et fonctions

                        org $C046       ; RELEASE
C046 E6 EB              db $E6, $EB     ; Adr RELEASE $EC0C-1 devient $EBE7-1 (commande GRAB)
; L'adresse de la commande RELEASE est remplacée par celle de la commande GRAB. Avec le nouvel
espace mémoire de la Snes, GRAB n'a plus de raison d'être et peut donc être supprimé. Idem pour la
commande RELEASE (qui ici devient REFRESH). Mais Fabrice réutilise la place devenue inutile pour
implanter le code de la routine de rafraîchissement de la Vram. Selon toute vraisemblance (à
vérifier), on peut utiliser aussi bien RELEASE (c'est-à-dire REFRESH), qui pointe sur GRAB, que GRAB
pour rafraîchir la Vram.

                        org $C072       ; CLOAD CSAVE DEF
C072 D9 E6              db $D9, $E6     ; Adr CLOAD $E85B-1 devient $E6DA-1
C074 08 E9              db $08, $E9     ; Adr CSAVE $E909-1 reste $E909-1 (pas de modification)
C076 23 F8              db $23, $F8     ; Adr DEL $D4BA-1 devient $F824-1

                        org $C098       ; &()
```

```
C098 87 F2            db $87, $F2    ; Adr &() $02FB vecteur de &() devient $F287 joyread

; Ne sont concernés ci-dessus que les commandes et fonctions dont le nom a été modifié. Mais
d'autres commandes voient aussi leur adresse d'exécution modifiée, le code ayant parfois été
légèrement déplacé. C'est le cas par exemple en $C076 de DEF  = $F824 (au lieu de $D4BA). Les
programmes qui appelleraient directement DEF à l'ancienne adresse auraient donc un problème. Tout
aussi grave est le cas de FILL, PAPER et INK. En effet, dans la table d'adresse (sise de $C006 à
$C0E9), l'adresse indiquée pour les commandes "son", plus PAPER et INK, est celle d'une seule et
même routine "Interprétation son" (sise en $EAFC), qui ensuite oriente vers l'adresse réelle de la
routine appropriée et c'est cette adresse qui est modifiée. De même, dans cette table, l'adresse des
commandes Hires (dont la commande FILL) est toujours celle de la routine "Interprétation graphique"
(sise en $EAF0), qui gère l'orientation appropriée. On observe ainsi des changements d'adresse en
$EAD1 pour FILL = $F21E (au lieu de $F268), en $EAD5 pour PAPER = $F1C8 (ex $F204) et en $EAD7 pour
INK = $F1F6 (ex $F210). Cela reste sans incidence au niveau de la compatibilité, tant qu'on utilise
l'interpréteur Basic, mais tout appel direct à ces commandes à leur ancienne adresse est voué à
l'échec.

; ****************************************************************************
; Modifications dans la table des noms des commandes et fonctions

                      org $C173      ; RELEASE devient REFRESH
C173 524546524553C8   db $52, $45, $46, $52, $45, $53, $C8 ; REFRES.H

                      org $C1DB      ; CLOAD devient LOAD, CSAVE devient SAVE et & devient JOY
; Les deux octets gagnés sur CSAVE et CLOAD équilibrent les deux octets perdus sur JOY. La table des
noms garde donc la même longueur, mais il faut tout décaler!
C1DB 4C 4F 41 C4      db $4C, $4F, $41, $C4         ; LOA.D
C1DF 53 41 56 C5      db $53, $41, $56, $C5         ; SAV.E
C1E3 44 45 C6         db $44, $45, $C6              ; DE.F
C1E6 50 4F 4B C5      db $50, $4F, $4B, $C5         ; POK.E
C1EA 50 52 49 4E D4   db $50, $52, $49, $4E, $D4    ; PRIN.T
C1EF 43 4F 4E D4      db $43, $4F, $4E, $D4         ; CON.T
C1F3 4C 49 53 D4      db $4C, $49, $53, $D4         ; LIS.T
C1F7 43 4C 45 41 D2   db $43, $4C, $45, $41, $D2    ; CLEA.R
C1FC 47 45 D4         db $47, $45, $D4              ; GE.T
C1FF 43 41 4C CC      db $43, $41, $4C, $CC         ; CAL.L
C203 A1               db $A1                        ; .!
C204 4E 45 D7         db $4E, $45, $D7              ; NE.W
C207 54 41 42 A8      db $54, $41, $42, $A8         ; TAB.(
C20B 54 CF            db $54, $CF                   ; T.O
C20D 46 CE            db $46, $CE                   ; F.N
C20F 53 50 43 A8      db $53, $50, $43, $A8         ; SPC.(
C213 C0               db $C0                        ; .@
C214 41 55 54 CF      db $41, $55, $54, $CF         ; AUT.O
C218 45 4C 53 C5      db $45, $4C, $53, $C5         ; ELS.E
C21C 54 48 45 CE      db $54, $48, $45, $CE         ; THE.N
C220 4E 4F D4         db $4E, $4F, $D4              ; NO.T
C223 53 54 45 D0      db $53, $54, $45, $D0         ; STE.P
C227 AB               db $AB                        ; .+
C228 AD               db $AD                        ; .-
C229 AA               db $AA                        ; .*
C22A AF               db $AF                        ; ./
C22B DE               db $DE                        ; .^
C22C 41 4E C4         db $41, $4E, $C4              ; AN.D
C22F 4F D2            db $4F, $D2                   ; O.R
C231 BE               db $BE                        ; .>
C232 BD               db $BD                        ; .=
C233 BC               db $BC                        ; .<
C234 53 47 CE         db $53, $47, $CE              ; SG.N
C237 49 4E D4         db $49, $4E, $D4              ; IN.T
C23A 41 42 D3         db $41, $42, $D3              ; AB.S
C23D 55 53 D2         db $55, $53, $D2              ; US.R
C240 46 52 C5         db $46, $52, $C5              ; FR.E
C243 50 4F D3         db $50, $4F, $D3              ; PO.S
C246 48 45 58 A4      db $48, $45, $58, $A4         ; HEX.$
C24A 4A 4F D9         db $4A, $4F, $D9              ; JO.Y

; ****************************************************************************
; Patches pour pouvoir exécuter la ROM en mode natif 65816 (avec A,X,Y sur 8 bits) -> il faut garder
1 dans le poids fort de l'accumulateur, et utiliser TCS au lieu de TXS. Les mnémoniques 65816
n'étant pas compris par l'assembleur utilisé, il faut entrer le code sous forme d'octets (db = data
byte).
                      cpu "65C"
; Probablement une commande pour l'assembleur Frankenstein, afin qu'il accepte certains mnémoniques
supplémentaires. En fait, le code de plx, phx, ply et phy qui sont des mnémoniques 65C02 & 65816,
est quand même entré sous forme d'octets.

                      org $C72C      ; Dans la commande CLEAR
C72C FA               db $FA         ; plx Pull Index Register X (mnémonique 65C02 & 65816)
C72D A9 FE            lda #$FE
C72F 1B               db $1B         ; tcs Transfer 16-bit Accumulator to Stack Pointer (65816)
```

```
C730 DA              db $DA         ; phx Push Index Register X (mnémonique 65C02 & 65816)

                     org $C865      ; Dans la commande FOR
C865 1B              db $1B         ; tcs

                     org $CA1B      ; Dans la commande RETURN
CA1B E8              inx

                     org $CA2B      ; Dans la commande RETURN
CA2B 8A              txa
CA2C 1B              db $1B         ; tcs
CA2D 68              pla
CA2E C0 0C           cpy #$0C
CA30 F0 18           beq $CA4A
CA32 85 A8           sta $A8
CA34 68              pla
CA35 85 A9           sta $A9
CA37 FA              db $FA         ; plx
CA38 7A              db $7A         ; ply Pull Index Register Y (mnémonique 65C02 & 65816)
CA39 20 11 D9        jsr $D911

                     org $CEAE      ; Dans la commande NEXT
CEAE 8A              txa
CEAF 1B              db $1B         ; tcs

                     org $CEF5      ; Dans la commande NEXT
CEF5 1B              db $1B         ; tcs

                     org $DAA8      ; Dans la commande UNTIL
DAA8 E8              inx
DAA9 C9 8B           cmp #$8B
DAAB F0 09           beq $DAB6

                     org $DAB6      ; Dans la commande UNTIL
DAB6 8A              txa
DAB7 1B              db $1B         ; tcs
DAB8 C0 10           cpy #$10
DABA F0 0A           beq $DAC6
DABC 20 E8 00        jsr $00E8
DABF 20 17 CF        jsr $CF17

                     org $ECCD      ; Dans "Reset Basic (Cold Start Basic)"
ECCD A9 FF           lda #$FF
ECCF 85 A9           sta $A9
ECD1 1B              db $1B         ; tcs

                     org $F88F      ; Dans "Reset System (Cold Start)"
F88F A9 FF           lda #$FF
F891 1B              db $1B         ; tcs

; ****************************************************************************
; Remplacement de la commande CLOAD pour lecture sur cartouche (banques $01, $02 et $03), patch pour
lecture du premier fichier au Cold Reset Basic et patch pour lire des fichiers en banque $7F (banque
secondaire).
                     org $E4F4      ; Dans "Charger un programme" ($E4E0-$E50A)
E4F4 97 33           db $97,$33     ; sta [$33],y DP long remplaçant sta ($33),y en page 0
; Les mnémoniques 65816 utilisant un adressage long indexé indirect en Direct Page sont indiqués par
[] au lieu de (). Dans l'Oric classique, à base de 6502, tout était inclus dans l'espace mémoire de
64 Ko (adresses de $0000 à $FFFF). Cet espace peut être vu comme un ensemble de pages.  Une page,
c'est #FF octets, il y a donc 256 pages dont l'adresse va de $00 (la fameuse page 0) à $FF (255). La
Rom (de $C000 à $FFFF) pouvait par exemple utiliser un pointeur de travail situé en page 0 en
utilisant un mode d'adressage spécifique au 6502, le mode "zéro-page".
; La "Rom Basic", et donc le code de la commande CLOAD, est exécuté dans la banque $7E, mais les
fichiers sont lus dans la mémoire flash, c'est à dire en banque $01, $02 ou $03 et copiés dans la
banque $7E (programmes) ou $7F (caractères, palettes, écrans Text ou Hires). Selon le mode
d'adressage, les registres DP (Direct Page), DBR (Data Bank Register) et PBR (Program Bank Register)
sont mis à contribution. Ici avec les [], il s'agit du mode "DP long, indirectement indexé par Y".
Cela signifie que l'adresse n'est pas dans la même banque que le programme ($7E), mais dans la
banque dont le n° a été  écrit dans le registre DP probablement $7F (à vérifier dans le contexte de
l'exécution). L'espace d'adressage de la Snes, qui va de $000000 à $FFFFFF peut s'écrire sous la
forme $abcdef où ab est le n° de banque (de $00 à $FF), cdef (de $0000 à $FFFF, comme dans l'Oric
classique) est l'adressage à l'intérieur d'une banque, ef (de $00 à $FF) est à l'intérieur d'une
page (256 octets) et enfin l'adressage abcd est le n° de page (sur deux octets). Dans l'Oric
classique (6502), il n'y a qu'un cas où on utilise un mode d'adressage qui ressemble au mode DP,
c'est l'adressage en page zéro. Dans la partie "code 65816", la DP est initialisée en fonction des
registres Snes à initialiser.

                     org $E4F9      ; Dans "Charger un programme"
E4F9 D7 33           db $D7,$33     ; cmp [$33],y

                     org $E6C9      ; Dans "Prendre un octet sur la K7", routine readbyte
```

```
E6C9 E6 00            inc 0
E6CB D0 0A            bne readok
E6CD E6 01            inc 1
E6CF D0 06            bne readok
E6D1 A9 80            lda #$80
E6D3 85 01            sta 1
E6D5 E6 02            inc 2
              readok
E6D7 A7 00            db $A7,0       ; lda [0]
E6D9 60               rts

                      org $E6DA      ; Dans "Prendre un octet sur la K7"
                                     ; C'est l'adresse de la nouvelle commande CLOAD
E6DA C9 80            cmp #$80       ; Teste A (début des tokens Basic en #80)
E6DC B0 03            bcs suite1     ; Branche si au-dessus (= si c'est un token Basic)
E6DE 4C 5B E8         jmp $E85B      ; Sinon ancienne entrée de la commande CLOAD
              suite1                 ; Derrière le CLOAD il y a un token Basic
; Ce peut être TEXT, HIRES, INK ou CHAR suivit d'une virgule. Seule la présence de la virgule est
analysée, car ce sont les adresses qui vont déterminer où le fichier est chargé en banque $7F. Il
devrait donc être possible de mettre n'importe quel token!
E6E1 20 E2 00         jsr $00E2      ; Lit le caractère suivant dans A et procède
; à une analyse sommaire: Z=1 si A = #00 ou ':', C=0 si A = chiffre de 0 à 9
E6E4 20 65 d0         jsr $D065      ; Demande une ',' non = erreur, oui = lit le caractère suivant
E6E7 08               php            ; Empile P, l'indicateur d'états
E6E8 A9 7F            lda #$7F       ; n°banque #7F (banque secondaire)
E6EA 20 B4 E7         jsr $E7B4      ; Reprise de l'analyse de syntaxe de CLOAD normale
E6ED 4C 5F E8         jmp $E85F      ; Reprise de la commande CLOAD normale
; Le code principal de cette commande (de $E85B à $E908) n'est pas modifié. Cependant, elle fait
appel directement ou indirectement à des sous-programmes dont certains ont été modifiés:
; $E4E0-$E50A "Charger le programme", voir ci-dessus (deux occurrences)
; $E7B2 "Syntaxe CLOAD et CSAVE", voir ci-dessous
; $E76A "Configurer le VIA"
; $E57D "Afficher 'Searching'"
; $E59B "Afficher 'Loading'"
; $E93D "Reconfigurer le VIA"
; $E651 "Vérifier pas d'erreur de parité"
; $E5F5-$E606 "Effacer la ligne 0", voir ci-dessous

                      org $E735      ; Dans "Trouver la bande amorce"
                                     ; Routine search synchro
E735 4C 4D E7         jmp $E74D
            bootstrap
E738 A9 22            lda #$22       ; guillemet
E73A 85 36            sta $36
E73C 64 37            stz $37
E73E 64 39            stz $39
E740 A9 36            lda #$36
E742 85 E9            sta $E9
E744 64 EA            stz $EA
E746 20 5B E8         jsr $E85B      ; CLOAD
E749 4C C1 C8         jmp $C8C1

                      org $E76A      ; Dans "Configurer le VIA pour travail K7"
E76A A9 FF            lda #$FF
E76C 85 00            sta 0
E76E 85 01            sta 1
E770 64 02            stz 2
E772 60               rts

                      org $E7B2      ; Dans "Syntaxe CLOAD et CSAVE"
E7B2 A9 7E            lda #$7E
E7B4 85  35           sta $35
E7B6 EA               nop
E7B7 9C AD 02         stz $02AD
E7BA 9C AE 02         stz $02AE
E7BD 9C 5B 02         stz $025B
E7C0 9C 5A 02         stz $025A
E7C3 9C 5C 02         stz $025C
E7C6 9C 5D 02         stz $025D
E7C9 9C B1 02         stz $02B1

; ****************************************************************************
; La commande RELEASE devenue REFRESH a été re-dirigée vers l'ancienne commande GRAB. GRAB n'a plus
de sens dans le Super-Oric. REFRESH et GRAB font la même chose: Envoyer l'écran Hires dans la VRAM
(Ram Vidéo).
                      org $EBE7      ; Dans la commande GRAB
EBE7 22 04 80 80      db $22, $04, $80, $80 ; jsr.l $808004
                                            ; hires_transfer = DMA Transfert Hires -> VRAM
EBEA 60               rts
; Illustration d'un jsr dans le code 65816 de la banque $80, lequel se termine par un RTL. C'est, je
crois, le seul endroit où le Basic exécute une routine 65816 située hors de la banque principale
```

($7E). Un exemple à suivre en cas de besoin, car il y a encore de la place disponible, notamment dans la banque $00 et sa copie $80.

```
; ******************************************************************************
; Affichages sur la première ligne

                        org $E5E1 ; Dans "Afficher nom du programme et son type" ($E5B6-$E5E4)
E5E1 E8                 inx
E5E2 4C 65 F8           jmp $F865

                        org $E5F5       ; Dans "Effacer la ligne 0"
E5F5 48                 pha
E5F6 A2 37              ldx #37         ; Nombre d'octets à effacer (2 octets par caractère),
; soit 28 caractères. C'est la  longueur maximale des chaînes affichées
E5F8 A9 20              lda #$20        ; 2 octets par caract. -> $2020 pour espace avec priorité
                statclear
E5FA CA                 dex             ; 1er octet
E5FB 9F 00 F0 7F        db $9F,$00,$F0,$7F    ; sta.l $7FF000,x Début écran Text en banque 7F
E5FF CA                 dex             ; 2e octet
E600 10 F8              bpl statclear   ;
E602 68                 pla
E603 60                 rts

                        org $F869       ; Dans "Ecrire sur la ligne 0"
F869 A0 00              ldy #0
                statwrite
F86B B1 0C              lda ($0C),y
F86D F0 09              beq endstatus
F86F 9F 00 F0 7F        db $9F,$00,$F0,$7F    ; sta.l $7FF000,x
F873 E8                 inx
F874 E8                 inx
F875 C8                 iny
F876 D0 F3              bne statwrite
                endstatus
F878 60                 rts

                        org $F76A       ; Dans "Ajuster CAPS"
F76A A2 36              ldx #54

; ******************************************************************************
; Patches pour affichage texte en banque $7F : On fait passer les pointeurs texte et graphique sur
24 bits. Du coup le pointeur texte est décalé de $12 en $13,  et il faut libérer les variables $14
et $15
                        org $FA88       ; Dans "Envoyer 14 paramètres au PSG 8912"
FA88 86 0E              stx $0E         ; libération des variables $14/$15
FA8A 84 0F              sty $0F
FA8C A0 00              ldy #0
FA8E B1 0E              lda ($0E),y

                        org $F5EC       ; Dans table des déplacements pour l'exécution des CTRL
F5EC 35                 db $35          ; LF est décalé en F664

                        org $F62F       ; Dans "Traiter les caractères de contrôle"
F62F CE 69 02           dec $0269
F632 CE 69 02           dec $0269
F635 10 42              bpl $F679
F637 EA                 nop
F638 EA                 nop
F639 A9 3E              lda #62

                        org $F649       ; Dans "Traiter caractères de contrôle, curseur haut"
F649 A5 13              lda $13
F64B E9 40              sbc #64
F64D 85 13              sta $13
F64F B0 02              bcs *+4
F651 C6 14              dec $14
F653 4C FE F6           jmp $F6FE

                        org $F656       ; Dans "Traiter caractères de contrôle, curseur droit"
F656 AD 69 02           lda $0269
F659 69 01              adc #1          ; C vaut 1
F65B 8D 69 02           sta $0269
F65E 0A                 asl a
F65F 10 18              bpl $F679
F661 20 0D F7           jsr $F70D

                        org $F664       ; Dans "Traiter caractères de contrôle, curseur bas (LF)"
F664 AD 68 02           lda $0268       ; LF est maintenant en F664 au lieu de F663
F667 CD 7E 02           cmp $027E
F66A F0 10              beq $F67C
F66C EE 68 02           inc $0268
```

```
F66F A5 13            lda $13
F671 69 40            adc #64
F673 85 13            sta $13
F675 90 02            bcc *+4
F677 E6 14            inc $14

                      org $F69B      ; Dans "Traiter caractères contrôle, effacer écran (FF)"
F69B 85 13            sta $13

                      org $F6A0      ; Dans "Traiter caractères contrôle, effacer écran (FF)"
F6A0 85 14            sta $14

                      org $F6A6      ; Dans "Traiter caractères contrôle, effacer écran (FF)"
F6A6 A5 13            lda $13
F6A8 69 40            adc #64
F6AA 85 13            sta $13
F6AC 90 02            bcc *+4
F6AE E6 14            inc $14

                      org $F6BE      ; Dans "Traiter caractères contrôle, curseur Home (RS)"
F6BE 85 13            sta $13

                      org $F6C3      ; Dans "Traiter caractères contrôle, curseur Home (RS)"
F6C3 85 14            sta $14

                      org $F6CB      ; Dans "Traiter caractères contrôle, retour chariot (CR)"
F6CB EA               nop
F6CC EA               nop
F6CD EA               nop

                      org $F70D      ; Dans "Placer curseur au début de la ligne"
F70D 9C 69 02         stz $0269
F710 60               rts
              deek24
F711 B7 00            db $B7,$00     ; lda [0],y
F713 AA               tax
F714 C8               iny
F715 B7 00            db $B7,$00     ; lda [0],y
F717 C8               iny
F718 60               rts
; En $F719 subsiste un ancien #$60 (rts)

                      org $F71A      ; Dans "Effacer ligne courante"
F71A A0 3F            ldy #63
              clrloop
F71C A9 20            lda #$20       ; priority bit à 1
F71E 97 13            db $97,$13     ; sta [$13],y
F720 88               dey
F721 97 13            db $97,$13     ; sta [$13],y
F723 88               dey
F724 10 F6            bpl clrloop
F726 60               rts
; Le reste de l'ancienne routine, de $F727 à $F730 (10 octets) est inchangé. La routine "AY=40*A"
qui suit (de $F731 à $F759) pourrait être récupérée.

                      org $F7E8      ; Dans "Afficher A selon $12-$13 et $269"
F7E8 97 13            db $97,$13     ; sta [$13],y

                      org $F7F3      ; Dans "Afficher A selon $12-$13 et $269"
F7F3 69 40            adc #64

                      org $F7F8      ; Dans "Afficher A selon $12-$13 et $269"
F7F8 97 13            db $97,$13     ; sta [$13],y

                      org $F80C      ; Dans "Effacer ou éteindre le curseur"
F80C B7 13            db $B7,$13     ; lda [$13],y

                      org $F813      ; Dans "Effacer ou éteindre le curseur"
F813 97 13            db $97,$13     ; sta [$13],y

                      org $EDC4      ; Dans "Déplacer un bloc vers le haut"
; Routine de recopie de bloc mémoire en banque $7F (utilisée par le scroll) avec ($0C) : source,
($0E) : destination et $10/$11 : nombre d'octets à transférer (+1 page si non-multiple de 256).
              LEDC4
EDC4 8B               db $8B         ; phb
EDC5 A9 7F            lda #$7F
EDC7 48               pha
EDC8 AB               db $AB         ; plb
EDC9 A0 00            ldy #0
              moveloop
EDCB B1 0C            lda ($0C),y
```

```
EDCD 91 0E            sta ($0E),y
EDCF C8               iny
EDD0 D0 04            bne *+6
EDD2 E6 0D            inc $0D
EDD4 E6 0F            inc $0F
EDD6 C6 10            dec $10
EDD8 D0 F1            bne moveloop
EDDA C6 11            dec $11
EDDC D0 ED            bne moveloop
EDDE AB               db $AB          ; plb
EDDF 60               rts             ; reste 5 octets libres

; ******************************************************************************
; Patches des routines PRINT @, PLOT et SCRN

                      org $CC5E       ; Dans "Traiter @"
CC5E 4C 66 CC         jmp $CC66

                      org $CC69       ; Dans "Traiter @"
CC69 E0 20            cpx #32

                      org $CC8A       ; Dans "Traiter @"
CC8A 0A               asl a
CC8B 8D 69 02         sta $0269
CC8E 8A               txa
CC8F 8D 68 02         sta $0268
CC92 20 0C DA         jsr calc_ligne
CC95 EA               nop
CC96 A4 01            ldy 1
CC98 85 13            sta $13
CC9A 84 14            sty $14

                      org $DA0C       ; Dans "Calculer adresse de la ligne A (mode TEXT)"
               calc_ligne             ; la routine x40 en F731 est récupérable
DA0C 4A               lsr a
DA0D 6A               ror a
DA0E 48               pha
DA0F 09 F0            ora #$F0
DA11 85 01            sta 1
DA13 A9 7F            lda #$7F
DA15 85 02            sta 2
DA17 68               pla
DA18 6A               ror a
DA19 29 C0            and #$C0
DA1B 85 00            sta 0
DA1D 60               rts

                      org $DA22       ; Dans "Récupérer 2 coordonnées pour PLOT et SCRN"
DA22 20 C8 D8         jsr $D8C8       ; Prendre une coordonnée
DA25 E0 20            cpx #32
DA27 B0 F6            bcs $DA1F
DA29 DA               db $DA          ; phx
DA2A 20 65 D0         jsr $D065       ; Demander une virgule
DA2D 20 C8 D8         jsr $D8C8       ; Prendre une coordonnée
DA30 E0 1C            cpx #28
DA32 B0 EB            bcs $DA1F
DA34 8A               txa
DA35 20 0C DA         jsr calc_ligne
DA38 68               pla
DA39 0A               asl a           ; Doubler la coordonnée horizontale
DA3A 05 00            ora 0
DA3C 85 00            sta 0
DA3E 60               rts

                      org $DA48       ; Dans la fonction SCRN
DA48 A0 00            ldy #0          ; Récupération d'un mot pour SCRN
DA4A 20 11 F7         jsr deek24
DA4D 9B               db $9B          ; txy
DA4E 4C 40 DF         jmp $DF40       ; YA -> ACC1 (non signé)

                      org $DA5C       ; Patch de la commande PLOT
DA5C 10 17            bpl numvalue
DA5E 20 D0 D7         jsr $D7D0       ; Récupère un descripteur sur la chaîne
DA61 AA               tax             ; Longueur dans X
DA62 F0 10            beq endplot
DA64 A0 00            ldy #0
               plotloop
DA66 B1 91            lda ($91),y
DA68 97 00            db $97,$00      ; sta [0],y
DA6A C8               iny
DA6B E6 00            inc 0
```

```
DA6D D0 02            bne *+4
DA6F E6 01            inc 1
DA71 CA               dex
DA72 D0 F2            bne plotloop
              endplot
DA74 60               rts
              numvalue
DA75 20 22 D9         jsr $D922       ; ACC1 -> YA
DA78 BB               db $BB          ; tyx
DA79 A0 00            ldy #0
              doke24
DA7B C8               iny
DA7C 97 00            db $97,$00      ; sta [0],y
DA7E 88               dey
DA7F 8A               txa
DA80 97 00            db $97,$00      ; sta [0],y
DA82 C8               iny
DA83 C8               iny
DA84 60               rts

; ****************************************************************************
; Patch du reset Basic (Cold Start Basic)

                      org $ED83       ; Dans "Reset Basic"
ED83 4C 38 E7         jmp bootstrap   ; jmp $E738 routine remplaçant "Trouver la bande amorce".
; L'ancien jmp $C4A8 allait à la routine "Afficher READY" puis à l'interpréteur Basic.

; ****************************************************************************
; Initialisation du mode texte

                      org $EC2F       ; Dans la commande TEXT
EC2F 20 C9 F9         jsr $F9C9       ; Permet de récupérer la routine en F967

2D1E                  org $ED1E       ; Dans "Reset Basic"
ED1E A9 20            lda #$20        ; 32 octets par ligne

                      org $F8D0       ; Dans "Configurer le système"
F8D0 A2 06            ldx #6
F8D2 20 82 F9         jsr LF982       ; Pas de génération du jeu alterné
F8D5 EA               nop
F8D6 EA               nop
F8D7 EA               nop

                      org $F90F       ; Dans "Initialiser couleur écran"
F90F A9 23            lda #$23        ; Protection des colonnes OFF
F911 8D 6A 02         sta $026A

; ****************************************************************************
; Configurer en mode Text

                      org $F9C9               ; Dans "Configurer en mode Text"
F9C9 A9 02            lda #%00000010          ; Enable BG2
F9CB 8F 2C 21 00      db $8F,$2C,$21,$00      ; sta.l $00212C main screen
F9CF 9C 1F 02         stz $021F               ; Flag HIRES off
F9D2 A2 06            ldx #6
              inittextparams
F9D4 BD F4 F9         lda scrollparams,x
F9D7 9D 78 02         sta $0278,x
F9DA CA               dex
F9DB 10 F7            bpl inittextparams
F9DD A9 7F            lda #$7F
F9DF 85 15            sta $15
F9E1 A9 F0            lda #$F0
F9E3 85 14            sta $14
F9E5 A9 00            lda #0
F9E7 85 13            sta $13
F9E9 20 1A F7         jsr $F71A               ; Effacement ligne statut
F9EC A2 0C            ldx #$0C                ; Effacement écran
F9EE 20 38 02         jsr $0238
F9F1 4C 5A F7         jmp $F75A               ; Affiche CAPS
              scrollparams
F9F4 80 F0 40 F0      dw $F080,$F040 ; Adresses 2e et 1e ligne
F9F8 80 07            dw $0780       ; nombre d'octets (une page de + car la 1e non complète)
F9FA 1B               db 27          ; 27 lignes (sans compter la ligne de statut)
; Libre jusqu'en $FA13 inclus (fin d'ancienne routine + "envoyer un attribut de mode d'écran".

; ****************************************************************************
; Initialisation du mode Hires

              graph_ink equ $026D
```

```
; Commande HIRES
                    org $EC39       ; Dans la commande Hires
EC39 EA             nop
EC3A EA             nop
; Dans le code de la commande Hires (inchangé par ailleurs), le flag GRAB est testé et en cas de
réponse positive, un branchement (ici désactivé par les deux nop) à "Disp Type Mismatch error" était
fait. GRAB n'a plus de sens avec le Super-Oric, d'où ce patch. Le code de HIRES fait appel la
routine "Créer l'écran Hires", qui, elle, a été modifiée comme suit:

                    org $F920       ; Dans "Créer l'écran Hires"
; (déborde dans code qui suit). Cette routine valide les modes BG1 et BG2, efface l'écran, force le
flag à Hires, fixe les coordonnées du curseur à X = 0 et Y = 0, initialise l'adresse de l'écran
Hires à $1000, met à zéro l'écran Hires de $1000 à $EFFF) et à #$FF les flags "dirty" correspondants
(de $7FFF10 à $7FFFEF) (= pages à rafraîchir). L'écran Hires est situé en banque $7F de $1000 à
$EFFF, soit de la page $10 à la page $EF. A chaque page correspond un drapeau "dirty" situé de
$7FFF10 à $7FFFEF. Lorsqu'une page est propre (n'a pas besoin d'être recopiée en Vram) ce drapeau
est à #$FF. Lorsqu'une commande Hires modifie (écrit ou efface) la partie de l'écran correspondant à
cette page, le drapeau correspondant est mis à #$00, indiquant qu'elle est "dirty" et donc qu'il
faudra la recopier dans la Vram, lors du prochain "blanking".
F920 48             pha
F921 A9 03          lda #%00000011          ; enable BG2+BG1
F923 8F 2C 21 00    db $8F,$2C,$21,$00      ; sta.l $00212C8f main screen
; $212C = registre "TM" = écran de premier plan. b0 et b1 à 1 activent BG1 et BG2.
F927 AD 1F 02       lda $021F
F92A F0 06          beq LF932               ; beq $F932
F92C 20 5B F9       jsr LF95B
F92F 20 E7 EB       jsr $EBE7               ; Ancienne commande GRAB, c'est à dire REFRESH
                LF932
F932 A2 0C          ldx #$0C        ; Code Form Feed pour effacer l'écran
F934 20 38 02       jsr $0238       ; jsr $F77C afficher un caractère
F937 A9 01          lda #$01        ; force le flag en Hires
F939 8D 1F 02       sta $021F       ; Flag 0=Text 1=Hires
F93C 8D 6D 02       sta $026D       ; Valeur graph_ink $026D = #$01 = couleur n°1 = blanc
F93F 9C 19 02       stz $0219       ; Coordonnée horizontale curseur Hires X = $00
F942 9C 1A 02       stz $021A       ; Coordonnée verticale curseur Hires Y = $00
F945 64 10          stz $10         ; Adresse de la ligne du curseur Hires LL
F947 A9 10          lda #$10
F949 85 11          sta $11         ; Adresse de la ligne du curseur Hires HH
; Soit $1000 au lieu de $A000
F94B A9 7F          lda #$7F        ; b7 = 0
F94D 85 12          sta $12         ; Anciennement adresse de la ligne du curseur Text LL,
; mais utilisé dans le Super-Oric v001 comme flag (à identifier).
F94F A9 80          lda #$80
F951 8D 15 02       sta $0215       ; Motif du sextet du curseur Hires
F954 A9 FF          lda #$FF
F956 8D 13 02       sta $0213       ; Registre Pattern
F959 68             pla
F95A 60             RTS

                LF95B
F95B A9 00          lda #$00
F95D 8F 00 10 7F    db $8F, $00, $10, $7F ; sta.l $7F1000
F961 A2 05          ldx #$05
                LF963
F963 BD 6E F9       lda $F96E,X
F966 95 0C          sta $0C,X
F968 CA             dex
F969 10 F8          bpl LF963               ; bpl $F963
F96B 4C C4 ED       jmp $EDC4               ; Déplacer un bloc vers le haut
; $0C-$0D bas du bloc à décaler $0E-$0F adresse cible $10-$11 nombre d'octets du bloc. En sortie, XY
contient la longueur du bloc.
F96E 00 10 01 10 FF E0db $00 $10 $01 $10 $FF $E0 ; Data
F974 EA             NOP
F975 20 C9 F9       jsr $F9C9               ; placer les paramètres de l'écran Text
F978 A9 01          lda #$01
F97A 0D 6A 02       ora $026A
F97D 8D 6A 02       sta L026A               ; et remettre le curseur
F980 68             pla
F981 60             rts

                    org $F982       ; Dans "Décaler une zone d'après table"
; Cette routine est appelée de $F8D0 avec X = #06
                LF982
F982 A0 05          ldy #$05        ; Pour 5 paramètres, index de 5 à 1
                LF984
F984 B9 A4 F9       lda LF9A4,Y     ; Lire paramètre dans table "Remplir écran Hires"
F987 99 0D 00       sta $000D,Y     ; Copie la table dans zone de travail de $0E à $12 avec:
; $0E-$0F adresse initiale = $FC78, adresse du 1er caractère (espace) de code Ascii #20, $10-11
adresse cible = $0200, adresse du code Ascii #20 en banque $7F et $12 nombre de caractères à
déplacer = #$7F (127).
F98A 88             dey             ; Paramètre précédent
```

```
F98B D0 F7            bne LF984       ; Reprendre boucle pour copier les 5 octets de la table
              LF98D                   ; Au premier tour, on entre dans cette boucle avec
; Y = #$00 puis Y est incrémenté 2 fois à chaque tour de boucle jusqu'à #(1)00.
F98D B2 0E            db $B2, $0E     ; lda ($0E),Y lit à partir de $FC78 en banque $7E
; A chaque tour, lit en $FC78, $FC7B, $FC7E etc. de 3 en 3 (voir plus loin)
F98F 97 10            db $97, $10     ; sta [$10],Y copie à partir de $0200 en banque $7F
; Adressage "Direct Page Indirect Long Indexed, Y" avec n° de banque $7F dans le registre DP. A
chaque tour, écrit en $0200, $0202, $0204 etc. de 2 en 2.
F991 C8               iny             ; Octet suivant
F992 A9 00            lda #$00        ; Intercale un #$00, car caractères définis sur 2 octets
F994 97 10            db $97, $10     ; sta [$10],Y en $0201, $0203, $0205 etc.
F996 E6 0E            inc $0E         ; Ces 3 lignes incrémentent l'adresse en $0E-$0F qui
F998 D0 02            bne LF99C       ; passe en $FC79, $FC7A etc. tandis que l'index Y
F99A E6 0F            inc $0F         ; augmente de 2 en 2. On lit donc un octet sur 3.
; A mon avis, ça ne peut pas marcher, il faudrait supprimer l'incrémentation de $0E-$0F et le INY en
$F99C. Mais comme ça marche, c'est moi qui pige rien à rien.
              LF99C
F99C C8               iny             ; De #$00, #$02, #$04... #$(1)00
F99D D0 EE            bne LF98D       ; Reboucle donc 127 fois
F99F E6 11            inc $11         ; Page suivante
F9A1 CA               dex             ; Je ne pige pas le rôle de ce décompte
F9A2 D0 E9            bne LF98D       ; Reboucle de X = 5 à X = 1
              LF9A4
F9A4 60               rts             ;
                                      ; Table "Remplir écran Hires"
F9A5 78 FC 00 02 7F   db $78, $FC, $00, $02, $7F    ; copiés en $0E-$0F, $10-11 et $12
; Les autres tables: caractères Rom->Ram, caractères Text->Hires, Hires-> Text n'existent plus

; ****************************************************************************
; Patches des routines graphiques

                      org $F024       ; Dans "Afficher un point"
F024 2C 12 02         bit $0212       ; Affichage du point suivant FB code
F027 30 1F            bmi FB23        ; bmi $F048 (fb23 est un label)
F029 A9 00            lda #0
F02B 50 03            bvc LF030       ; bvc *+5 = bvc $F030
              setdot
F02D AD 6D 02         lda graph_ink
              LF030
F030 85 2F            sta $2F
F032 A0 07            ldy #7
              loopfb01
F034 AD 15 02         lda $0215
F037 06 2F            asl $2F
F039 B0 05            bcs bit1
F03B 49 FF            eor #$FF
F03D 37 10            db $37,$10      ; and [$10],y
F03F 2C               db $2C          ; Saute l'instruction qui suit
              bit1
F040 17 10            db $17,$10      ; ora [$10],y
F042 97 10            db $97,$10      ; sta [$10],y
F044 88               dey
F045 10 ED            bpl loopfb01
F047 60               rts

              fb23
F048 70 0C            bvs fb3         ; bvs $F056
F04A A0 07            ldy #$07        ;
              loopfb2
F04C AD 15 02         lda $0215       ;
F04F 57 10            db $57,$10      ; eor [$10],y
F051 97 10            db $97,$10      ; sta [$10],y
F053 88               dey             ;
F054 10 F6            bpl loopfb2     ; bpl $F04C
              fb3
F056 60               rts

              getdotcolor
F057 A0 07            ldy #$07
              getdotloop
F059 B7 10            db $B7,$10      ; lda [$10],y
F05B 2D 15 02         and $0215
F05E 18               clc
F05F 69 FF            adc #$FF
F061 26 2F            rol $2F         ; Résultat dans $2F
F063 88               dey
F064 10 F3            bpl getdotloop  ; bpl $F059
              comparecolor
F066 A5 2F            lda $2F
F068 CD E5 02         cmp $02E5       ; Compare avec la couleur de bordure
F06B 60               rts
```

```
                setcursor
                        org $F06C       ; Params X,Y
F06C 48                 pha
F06D 98                 tya
F06E 18                 clc
F06F 69 10              adc #$10
F071 85 11              sta $11
F073 8A                 txa
F074 29 F8              and #$F8
F076 85 10              sta $10
F078 8A                 txa
F079 29 07              and #7
F07B AA                 tax
F07C BD 84 F0           lda power2,x    ; lda $F084,X
F07F 8D 15 02           sta $0215
F082 68                 pla
F083 60                 rts

                power2
F084 80 40 20 10        db $80, $40, $20, $10 ; Tableau de data
F089 08 04 02 01        db $08, $04, $02, $01

                go_down
F08C E6 11              inc $11
F08E A5 11              lda $11
F090 C9 F0              cmp #$F0
F092 60                 rts
F093 11 60              ora ($60),Y     ; Code inutile

                horizdir equ $021B
                go_up
F095 C6 11              dec $11
F097 A5 11              lda $11
F099 C9 0F              cmp #$0F
F09B 60                 rts

                horizmove
F09C 2C 1B 02           bit horizdir    ; bit $21B
F09F 30 11              bmi gauche
F0A1 4E 15 02           lsr $215        ; (???droite reste en $F0A1???)
F0A4 90 09              bcc droite_ok   ; bcc $F0AF
F0A6 6E 15 02           ror $215
F0A9 A5 10              lda $10
F0AB 69 08              adc #$08
F0AD 85 10              sta $10
                droite_ok
F0AF 60                 rts
F0B0 11 60              ora ($60),Y     ; ???
                gauche
F0B2 0E 15 02           asl $215
F0B5 90 09              bcc gauche_ok   ; bcc $F0C0
F0B7 A5 10              lda $10
F0B9 E9 08              sbc #$08
F0BB 85 10              sta $10
F0BD 2E 15 02           rol $215
                gauche_ok
F0C0 60                 rts
```

; Le reste de la commande "Déplacer vers la gauche" (de $F0C1 à $F0C7) présente dans Basic11B.rom est conservé inchangé, mais n'est pas utilisé. Les commandes suivantes "CURSET" (de $F0C8 à F0FC), "CURMOV" (de $F0FD à $F10F), "DRAW" (de $F110 à $F11C) et "PATTERN" (de $F11D à $F12C) sont également inchangée entre les deux versions: Basic11B.rom et v001. Curieusement la commande suivante "CHAR" (de $F12D à $F170), devenue inutile avec la superposition des écrans TEXT et HIRES est conservée et même adaptée au Super-Oric en $F152, en $F159 et $F169 (voir plus loin les patchs correspondants). La routine "Calculer l'adresse d'un caractère" (de $F171 à $F19A) est également inchangée. Par contre, la routine "Afficher un caractère" (de $F19B à $F1C7) a été patchée en $F1BD (voir un peu plus loin). Le bloc de code de $F1C8 à $F2A6, correspondant à l'ancienne routine "Evaluer la couleur d'un point", ainsi qu'aux anciennes commandes PAPER, INK, FILL et sous-programmes associés, a été complètement réécrit, pour les nouvelles commandes PAPER, INK, FILL et joyread.

```
                        org $EC48       ; Dans la fonction POINT
EC48 20 03 CF           jsr $CF03       ; Correction bug

                        org $EC67       ; Dans la fonction POINT
EC67 20 03 CF           jsr $CF03

                        org $EC83       ; A la fin de la fonction POINT
EC83 20 57 F0           jsr getdotcolor ; Anciennement jsr $F1C8.
```

; Du coup, la routine $F1C8 est disponible jusqu'à $F1FF. L'emplacement de cette routine $F1C8

```
"Evaluer couleur d'un point" a donc été réutilisé pour coder PAPER, INK, FILL, JOY.
EC86 4E E0 02         lsr $02E0      ; Erreur d'intervalle ?
EC89 B0 0E            bcs $EC99
EC8B A8               tay
EC8C 20 B6 D4         jsr $D4B6      ; Y -> ACC1
EC8F 4C 5F D0         jmp $D05F

                      org $EEEE      ; Dans "Afficher un point"
EEEE 20 6C F0         jsr setcursor

                      org $F169      ; Patch de la commande CHAR
F169 20 6C F0         jsr setcursor

                      org $F2C5      ; Dans la commande FILL
F2C5 20 6C F0         jsr setcursor

                      org $F3D4      ; Dans la commande CIRCLE, calcul et affichage
F3D4 20 6C F0         jsr setcursor

                      org $EF67      ; Dans "Tracer une droite selon l'axe des Y"
EF67 20 8C F0         jsr go_down

                      org $EFA2      ; Dans "Tracer une droite selon l'axe des X"
EFA2 20 8C F0         jsr go_down

                      org $F1BD      ; Dans la routine "Afficher un caractère"
F1BD 20 8C F0         jsr go_down

                      org $F466      ; Dans routine "Calculer nouvelle position horizontale"
F466 20 8C F0         jsr go_down

                      org $F0D7      ; Patch dans la routine CURSET
F0D7 20 DF F0         jsr verifycoord ; jsr $F0DF
F0DA B0 30            bcs $F10C
F0DC 4C E8 EE         jmp $EEE8
               verifycoord
F0DF 38               sec
F0E0 AD E2 02         lda $02E2
F0E3 D0 15            bne endverif
F0E5 AE E1 02         ldx $02E1
F0E8 AD E4 02         lda $02E4
F0EB D0 0D            bne endverif
F0ED AC E3 02         ldy $02E3
F0F0 C0 E0            cpy #224
F0F2 B0 06            bcs endverif
F0F4 8E 19 02         stx $0219
F0F7 8C 1A 02         sty $021A
               endverif
F0FA 60               rts
; Note: De $F0FB à F0FC, restent deux anciens octets de la fin de commande CURSET,  probablement
sans signification dans la Rom Basic du Super-Oric:
; F0FB 02             db $02
; F0FC 60             rts

                      org $F152      ; Patch situé dans la commande CHAR
F152 C9 FB            cmp #256-5     ; largeur écran Hires 256 pixels au lieu de 240

                      org $F159      ; Patch situé dans la commande CHAR
F159 C9 D9            cmp #224-7     ; hauteur écran Hires 224 pixels au lieu de 200

                      org $F325      ; Dans la commande DRAW, vérification des paramètres
F325 AD 01 02         lda $0201
F328 38               sec
F329 D0 31            bne $F35C
F32B EA               nop
F32C EA               nop
F32D EA               nop

                      org $F342      ; Dans la commande DRAW, vérification des paramètres
F342 A9 E0            lda #224

                      org $F396      ; Dans la commande CIRCLE
F396 EA               nop
F397 EA               nop

                      org $F3A6      ; Dans la commande CIRCLE
F3A6 C9 E0            cmp #224

; ****************************************************************************
; la commande WAIT
                      org    $D958   ; Dans la commande WAIT
```

```
D958 20 53 E8         jsr     $E853
D95B AA               tax
D95C 4C C9 EE         jmp waitvbl     ; jsr $EEC9

                      org     $EEC9   ; Dans "Attendre qu'un timer s'annule"
              waitvbl
EEC9 CB               db $CB          ; Mnémonique 65816  wai = Wait for Interrupt (1 octet)
EECA C0 02            cpy #2
EECC B0 04            bcs dec_lsb
EECE 8A               txa
EECF F0 05            beq endwait
EED1 CA               dex
              dec_lsb
EED2 88               dey
EED3 88               dey
EED4 80 F3            bra     waitvbl
              endwait
EED6 60               rts

; ******************************************************************************
; PAPER, INK, FILL, JOY             ; Il faut savoir que dans la table d'adresse ($C006 à $C0E9),
l'adresse des commandes "son", plus PAPER et INK, est celle d'une seule routine "Interprétation son"
(sise en $EAFC), qui ensuite oriente vers la routine appropriée. Idem l'adresse des commandes Hires
(dont FILL) est celle d'une seule routine "Interprétation graphique" (sise en $EAF0), qui ensuite
oriente vers la routine appropriée.

                      org $EAD1       ; Table d'adresses
EAD1 1D F2            dw fill-1       ; FILL = $F21E (au lieu de $F268)

                      org $EAD5       ; Table d'adresses
EAD5 C7 F1 F5 F1      dw paper-1,ink-1      ; PAPER = $F1C8 (ex $F204) INK = $F1F6 (ex $F210)

                      org $F1C8       ; Dans ex routine "Evaluer couleur d'un point"
; Le bloc de code de $F1C8 à F2A6, correspondant à l'ancienne routine "Evaluer couleur d'un point",
ainsi qu'aux anciennes commandes DRAW, INK, FILL et sous-programmes associés, a été complètement
réécrit, pour les nouvelles commandes PAPER, INK, FILL et JOY.

              paper
F1C8 A2 00            ldx #0
              ink2
F1CA AD E2 02         lda $02E2
F1CD D0 3B            bne returnerror       ; $F20A
F1CF AD E1 02         lda $02E1
F1D2 C9 08            cmp #8
F1D4 B0 34            bcs returnerror
F1D6 AD C0 02         lda $02C0
F1D9 4A               lsr a
F1DA AD E1 02         lda $02E1
F1DD B0 03            bcs *+5               ;  bcs $f1e2
F1DF 9D 6B 02         sta $026B,x
F1E2 0A               asl a
F1E3 A8               tay
F1E4 8A               txa
F1E5 0A               asl a
F1E6 AA               tax
F1E7 B9 0E F2         lda rgbtable,y
F1EA 9F 00 F8 7F      db $9F,$00,$F8,$7F    ; sta.l $7FF800,x
F1EE B9 0F F2         lda rgbtable+1,y
F1F1 9F 01 F8 7F      db $9F,$01,$F8,$7F    ; sta.l $7FF801,x
F1F5 60               rts
              ink
F1F6 A2 01            ldx #1
F1F8 AD C0 02         lda $02C0
F1FB 4A               lsr a
F1FC 90 CC            bcc ink2              ; ink2 pour le mode texte
F1FE AD E2 02         lda $02E2
F201 D0 07            bne returnerror
F203 AD E1 02         lda $02E1
F206 8D 6D 02         sta graph_ink
F209 60               rts
              returnerror
F20A EE E0 02         inc $02E0
F20D 60               rts
              rgbtable
F20E 00 00 1F 00 E0 03        dw $0000,$001F,$03E0  ; Noir, rouge, vert
F214 FF 03 00 7C 1F 7C        dw $03FF,$7C00,$7C1F  ; jaune, bleu, magenta
F21A E0 7F FF 7F              dw $7FE0,$7FFF        ; cyan, blanc
              fill
F21E AD E6 02         lda $02E6
F221 D0 E7            bne returnerror
F223 20 DF F0         jsr verifycoord
```

```
F226 B0 E2            bcs returnerror
F228 20 6C F0         jsr setcursor          ; $F06C
F22B 20 57 F0         jsr getdotcolor        ; $F057
F22E F0 56            beq endfill            ; $F286
F230 38               sec
F231 6E 1B 02         ror horizdir           ; $021B
F234 20 4D F2         jsr fill_one_dir       ; $F24D
F237 4E 1B 02         lsr horizdir
F23A AE 19 02         ldx $0219
F23D AC 1A 02         ldy $021A
F240 20 6C F0         jsr setcursor
F243 20 9C F0         jsr horizmove
F246 F0 3E            beq endfill
F248 20 57 F0         jsr getdotcolor
F24B F0 39            beq endfill
                   fill_one_dir              ; $F24D
                   search_highest            ; $F24D
F24D 20 95 F0         jsr go_up              ; $F095
F250 F0 05            beq save_highest       ; $f257
F252 20 57 F0         jsr getdotcolor
F255 D0 F6            bne search_highest     ; $F24D
                   save_highest              ; $f257
F257 20 8C F0         jsr go_down
F25A A5 11            lda $11
F25C 48               pha
                   fill_down                 ; $f25D
F25D 20 2D F0         jsr setdot
F260 20 8C F0         jsr go_down
F263 F0 05            beq save_lowest        ; $f26A
F265 20 57 F0         jsr getdotcolor
F268 D0 F3            bne fill_down
                   save_lowest
F26A 20 5D F3         jsr $F35D              ; Sauve curseur dans $216/217/218
F26D 68               pla
F26E 85 11            sta $11
F270 20 9C F0         jsr horizmove
F273 F0 11            beq endfill
                   searchfirstdown           ; $f275
F275 20 57 F0         jsr getdotcolor
F278 D0 05            bne test_too_low       ; $f27F
F27A 20 8C F0         jsr go_down
F27D D0 F6            bne searchfirstdown
                   test_too_low              ; $f27F
F27F A5 11            lda $11
F281 CD 17 02         cmp $0217
F284 90 C7            bcc fill_one_dir
                   endfill
F286 60               rts                    ;
                                             ; Reste 96 octets libres !!
                   joyread
F287 20 22 D9         jsr $D922              ; ACC1->YA
F28A AA               tax
F28B D0 17            bne illquant2
F28D 98               tya
F28E C9 04            cmp #4
F290 B0 12            bcs illquant2
F292 0A               asl a
F293 AA               tax
F294 BF 19 42 00      db $BF,$19,$42,$00     ; lda.l $004219,x
F298 A8               tay
F299 BF 18 42 00      db $BF,$18,$42,$00     ; lda.l $004218,x
F29D 4A               lsr a
F29E 4A               lsr a
F29F 4A               lsr a
F2A0 4A               lsr a
F2A1 4C 40 DF         jmp $DF40
                   illquant2
F2A4 4C 36 D3         jmp $D336      ; Le reste de l'ancienne commande FILL est inchangé.

                      org $F2C5       ; Dans la fin du code inchangé de la commande FILL
F2C5 20 6C F0         jsr setcursor  ; jsr $F06C

                      org $F325       ; Dans la commande DRAW, vérification des paramètres
F325 AD 01 02         lda $0201
F328 38               sec
F329 D0 31            bne $F35C
F32B EA               nop
F32C EA               nop
F32D EA               nop

; ******************************************************************************
```

```
; La définition de couleurs et de caractères

                        org $C076               ; DEF Dans la table d'adresse des commandes
C076 23 F8              dw defcommand-1         ; $F824-1 au lieu de $D4BA-1

                        org $F816       ; A la place de "générer les caractères alternés"
                getbytevalue            ; renvoie dans A et Y la valeur du paramètre (doit être <256)
F816 20 E2 00           jsr $00E2
F819 20 53 E8           jsr $E853
F81C AA                 tax
F81D D0 02              bne illquant
F81F 98                 tya
F820 60                 rts
                illquant
F821 4C 36 D3           jmp $D336       ; Illegal quantity error

                defcommand              ; C'est l'adresse de la nouvelle commande DEF
F824 C9 B2              cmp #$B2
F826 F0 07              beq def_ink
F828 C9 B0              cmp #$B0
F82A F0 0E              beq def_char
F82C 4C BA D4           jmp $D4BA
                def_ink
F82F 20 16 F8           jsr getbytevalue        ; jsr $F816
F832 0A                 asl a
F833 AA                 tax
F834 A9 00              lda #0
F836 69 F8              adc #$F8
F838 80 0D              db $80, $0D     ; bra doke_values code 65816 caché
                def_char
F83A 20 16 F8           jsr getbytevalue
F83D 0A                 asl A
F83E 0A                 asl A
F83F 0A                 asl A
F840 0A                 asl A
F841 AA                 tax
F842 98                 tya
F843 4A                 lsr A
F844 4A                 lsr A
F845 4A                 lsr A
F846 4A                 lsr A
                doke_values
F847 A0 00              ldy #0
F849 86 00              stx 0
F84B 85 01              sta 1
F84D A9 7F              lda #$7F
F84F 85 02              sta 2
                loop_doke
F851 5A                 phy
F852 20 65 D0           jsr $D065       ; Demande une virgule
F855 20 53 E8           jsr $E853       ; Récupère une valeur 16 bits
F858 BB                 db $BB          ; tyx
F859 7A                 ply
F85A 20 7B DA           jsr doke24      ; jsr $DA7B
F85D 20 E8 00           jsr $00E8
F860 C9 2C              cmp #$2C        ; Virgule ?
F862 F0 ED              beq loop_doke
F864 60                 rts
```

; Note sur le rafraîchissement de la Vram. Je m'interroge sur la fonctionnalité du système des drapeaux "dirty" dans cette Rom Basic1.bin. Après avoir chargé une image Hires avec HIRES:LOADINK,"PALETTE":LOADHIRES,"IMAGE", l'utilisateur doit ajouter un REFRESH à son programme pour voir le résultat. REFRESH appelle la routine $EBE7, qui effectue un jsr.l $808004 (hires_transfer -> VRAM). Cette dernière, située dans la banque $80, recopie en bloc les#E000 octets de l'écran Hires dans la Vram. Dans le code de la Rom Basic, et plus précisément dans la routine "Créer l'écran Hires", en $F920, l'écran Hires est mis à zéro de $1000 à $EFFF et les flags "dirty" correspondants sont mis à #$FF, de $7FFF10 à $7FFFEF. A chaque page de l'écran Hires correspond un drapeau "dirty". Lorsqu'une page est propre (n'a pas besoin d'être recopiée en Vram) ce drapeau est à #$FF. En théorie, lorsqu'une commande Hires modifie (écrit ou efface) la partie de l'écran correspondant à cette page, le drapeau correspondant est mis à #$00, indiquant qu'elle est "dirty" et donc qu'il faudra la recopier dans la Vram, lors du prochain "blanking". Or, je n'ai pas trouvé de mise à "dirty" de ces drapeaux, dans le code des diverses commandes Hires (DRAW, FILL etc.). Par contre dans le programme "Demo graphique" de Fabrice, il y a un RELEASE après chaque commande Hires (DRAW, FILL etc.). Le rafraichissement de la Vram n'est donc pas automatique au vu des flags "dirty". Au contraire, il faut faire appel à la commande "hires_transfer -> VRAM". Les drapeaux "dirty" jouent-ils un rôle dans cette Rom Basic1.bin et lequel ? Sont-ils seulement l'ébauche d'un système alternatif destiné à remplacer la routine de rafraîchissement complet de l'écran Hires, lequelle entraîne un "blanking" de durée trop importante et par suite un effet de scintillement. La Rom Super-Oric suivante (v0.02 ou Base2.bin) apportera une réponse partielle à ce problème.