Le compilateur Basic de Ray MacLaughlin (3)

Le cas des commandes Sedoric (fin)

par André C.

Les commandes Sedoric ne sont pas acceptées par le Compiler. Nous avons vu que la méthode classique qui consiste à placer des commandes dans le tampon clavier ne marche plus avec un Basic compilé. En effet, le programme Basic a été transformé en programme en langage machine et l'interpréteur Basic et donc le TXTPTR ne sont plus d'actualité.
Une autre possibilité serait de faire appel à un sous-programme en langage machine capable d'émuler/remplacer une commande Sedoric. En effet, le Compiler de Ray gère la commande CALL sans aucun problème, comme n'importe quelle commande Basic. La commande LOAD serait notamment bien utile et c'est elle que je vais tenter de réaliser aujourd'hui.

La Ram overlay et le Compiler de Ray

J'ai, sans aucun doute, été trop optimiste en m'attaquant directement à la commande LOAD. La première chose à faire aurait été de vérifier si le Compiler de Ray est compatible avec un aller et retour sur la Ram overlay (où se trouvent les commandes Sedoric).
Rappelons que cette Ram overlay est située de #C000 à #FFFF, aux même adresses donc que la ROM. Cela fonctionne comme une banque permutable qui bascule entre Ram overlay (utilisée par le DOS) et Rom (utilisée par le Basic). Faire des aller/retour entre la Rom et la Ram overlay ne pose aucun problème et heureusement car, en présence d'un lecteur de disquette, cela se produit très fréquemment. Encore faut-il vérifier que ça marche aussi après compilation du programme Basic. Normalement, cela devrait marcher, mais j'ai déjà eu de mauvaises surprises...
Voici un exemple simple, qui consiste à aller lire un octet de la Ram overlay, retourner au Basic et afficher la valeur de cet octet. La figure 2 montre le sous-programme en langage machine utilisé.
L'assemblage de ce programme source OA.ASM produit le fichier OA.TAP, chargeur Basic qui met en place le code lui-même. Je sauve ce code sous le nom OA.BIN (#9801 à #980D) et le place sur une disquette de test COMPI3.DSK.
Il me faut aussi un petit programme Basic (que je sauve sur la disquette de test sous le nom OA.BAS) pour mettre ce sous-programme en œuvre. En voici le listing :

```
10 CALL#9801
20 V=PEEK(#980D)
30 IF V=#00 THEN PRINT"ORIC-1"
40 IF V=#80 THEN PRINT"ATMOS"
```

Figure 1

Je charge OA.BIN en mémoire, puis OA.BAS que je lance avec un RUN.
La figure 3 (page suivante) montre le résultat : La machine que j'ai utilisée était un Atmos. Je suis bien content de l'apprendre !
Minute de vérité maintenant, je mouline AO.BAS avec le Compiler de Ray. J'obtiens AO.COM que je teste après avoir mis en place le sous-programme AO.BIN : ça marche, j'obtiens le même résultat que précédemment (figure 4, page suivante).

Conclusion : Il est possible d'accéder à la Ram overlay à partir d'un programme Basic compilé.

```
; Sous-programme OA.asm en langage machine pour lire en Ram
; overlay si la machine utilisée est un Oric-1 ou un Atmos.
; L'adresse $C024 contient #00 si la machine est équipée d'une
; Rom 1.0 ou contient #80 si la Rom est une version 1.1
     org $9801       ; Adresse d'implantation du sous-programme
     JSR $04F2       ; Passage sur la Ram overlay
     LDA $C024       ; Lit le drapeau de version de Rom
     STA drapeau     ; Et le sauve à l'adresse drapeau
     JMP $04F2       ; Retour sur la Rom
drapeau              ; Adresse de stockage du résultat ($980D)
     db $00
```

Figure 2

Figure 3

Figure 4

Test d'une commande Sedoric sans argument

L'étape suivante avant de se lancer dans CLOAD est de faire un test avec une commande simple. Il existe très peu de commandes Sedoric sans aucune paramètre. J'ai choisi la commande SYS, car elle marche en mode programme (surprenant non?) et n'interfère pas avec le programme Basic (comme le ferait OLD par exemple). Soit le programme en langage machine SY.ASM (figure 6). Après assemblage, j'obtiens SY.TAP, le chargeur BASIC qui met le code en place. Je sauve ce code sous le nom SY.BIN (de #9801 à #9809) sur ma disquette de test. Et voici le petit programme Basic que j'ai utilisé pour tester ce sous-programme (figure 5).

Je sauve ce programme sous le nom SY.BAS sur ma disquette de test. Un petit RUN après avoir placé en mémoire SY.BIN et SY.BAS...

Ça marche (voir figure 7, page suivante). Puisque ça se présente bien, je mouline ce petit programme Basic avec le Compiler de Ray. Après avoir chargé le sous-programme SY.BIN en mémoire, je lance le Basic compilé SY.COM (qui est en AUTO) (figure 8, page suivante) : Bingo ! Ça marche à l'identique !

Conclusion : Il est possible d'utiliser d'exploiter des commandes Sedoric sans paramètre, à partir d'un programme Basic compilé.

Quelques infos sur la commande Sedoric LOAD

Avant d'aborder l'étape suivante, à savoir le test d'une commande Sedoric avec argument (et l'élue de mon cœur est bien sûr la commande LOAD), il est prudent de regarder comment se présente cette commande.

Le code de la commande LOAD est situé dans la Ram overlay à l'adresse #DFF7.

Il comporte deux parties principales :

1. De #DFF7 à #E051, analyse de syntaxe et saisie des paramètres. Les paramètres qui suivent le mot-clé LOAD sont mis en place dans les variables correspondantes de la Ram overlay.
2. De #E052 à #E0AE, chargement du fichier en fonction des variables de la Ram overlay qui ont été mises à jour.

La première phase utilise TXTPTR or il est prudent d'y renoncer pour cause d'incompatibilité avec le Compiler. Il faut donc que mon sous-programme gère les paramètres de LOAD de manière autonome. Heureusement, ce n'est pas bien difficile.

La structure des paramètres est la suivante :

```
10 PRINT"VOICI LA CONFIGURATION"
20 CALL#9801
30 PRINT"AU REVOIR"
```

Figure 5

```
; Sous-programme en langage machine lançant la commande SYS
; pour exécution avec un CALL à partir d'un programme Basic
     org $9801      ; Adresse d'implantation du sous-programme
     JSR $04F2      ; Passage sur la Ram overlay
     JSR $F15A      ; Exécute la commande SYS
     JMP $04F2      ; Retour sur la Rom
fin
```

Figure 6

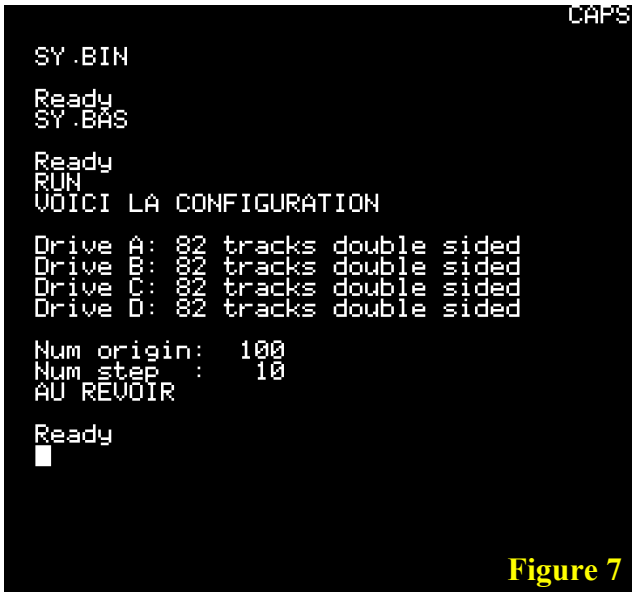

Figure 7

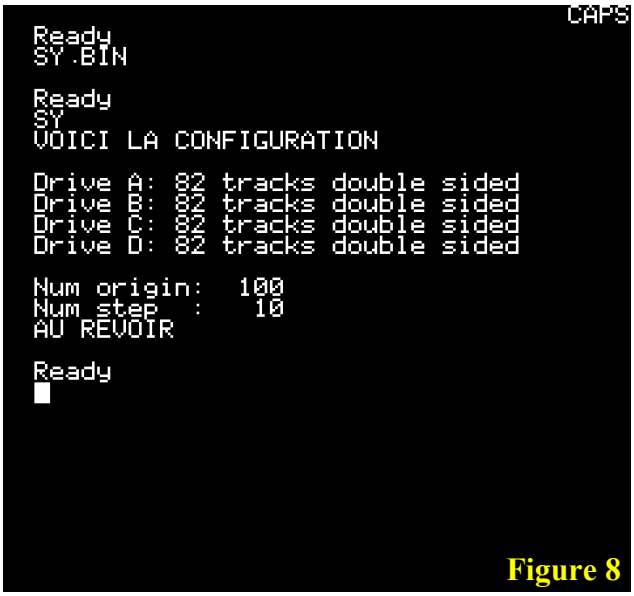

Figure 8

#C028	DRIVE	n° de drive pour LOAD/SAVE (#00 par défaut pour le drive A).
#C029-#C031	BUFNOM	Le nom (complété à 9 caractères avec des espaces).
#C032-#C034	BUFNOM'	L'extension (complétée à 3 caractères avec des espaces).
#C035-#C036	PSDESP	Les coordonnées Piste/Secteur du DEScripteur Principal.
#C037-#C038	NSTOTP	Le nombre de Secteurs TOTaux + Prot/unprot.
#C04D	SALO0	",V" ou ",N" (b6=1 si ",V" et b7=1 si ",N", valeur par défaut #00).
#C04E	VSALO1	",A" ou ",J" (b6=1 si ",A" et b7=1 si ",J", valeur par défaut #00).
#C052	DESALO	Si ",A" (adresse de début de fichier).

Notes en passant :

1. *Normalement, vous n'aurez à prendre en compte que les variables DRIVE et BUFNOM. Je vous indique les autres uniquement au cas où vous voudriez passer à des expériences plus complexes.*
2. *Vous pouvez remarquer que les 16 valeurs de #C029 à #C038 correspondent exactement à une ligne de catalogue dans le directory (sans importance pour mon propos ici).*

Commande Sedoric avec arguments : LOAD

Dans la grande majorité des cas, à l'intérieur d'un programme, nous utilisons la commande LOAD pour charger un fichier de la forme "nnnnnnn.eee" à sa place normale. Dans le cas général, les paramètres à gérer sont donc très simples et se réduisent au nom du fichier à charger et éventuellement au lecteur à utiliser. Il suffit tout simplement de copier "nnnnnnneee" (sans le

```
      org $9801       ; Adresse d'implantation du sous-programme
      JSR $04F2       ; Passage sur la Ram overlay
      LDA $C009       ; Lit le n° de drive par défaut
      STA $C028       ; et le copie devant BUFNOM
      LDX #$00        ; Initialise l'index de lecture/écriture
boucle
      LDA chaine,X    ; Lit les 12 octets de la chaîne
      BEQ suite       ; Si l'octet lu est #$00, c'est fini
      STA $C029,X     ; Sinon le copie dans BUFNOM
      INX             ; Vise l'octet suivant
      BNE boucle      ; et reboucle
suite
      JSR $E052       ; Exécution de la commande Sedoric LOAD
      JMP $04F2       ; Retour sur la Rom
chaine
      string "EXEMPLE01SCR"
      db $00
fin
```

Figure 9

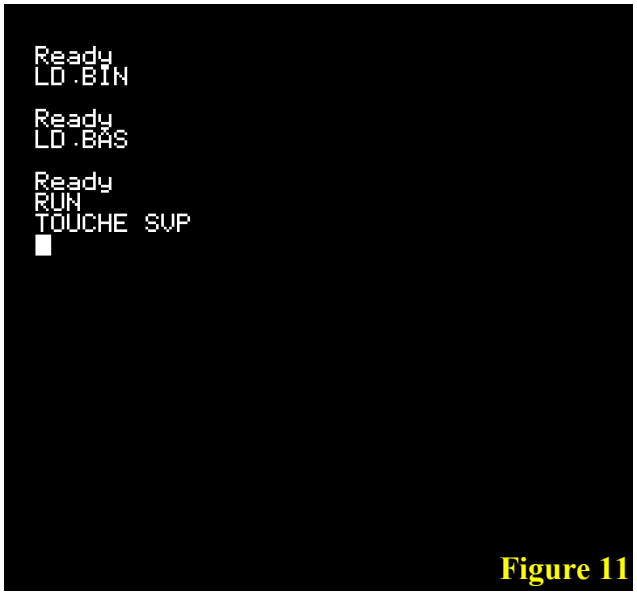

Figure 11

Figure 12

point) dans BUFNOM (de #C029 à #C034) et éventuellement le n° de lecteur (de #00 à #03) à l'adresse #C028, juste devant BUFNOM. C'est ce que fait le programme source LD.ASM de la figure 9 (page précédente).
Vous aurez remarqué que le lecteur à utiliser est le lecteur par défaut. Il n'était donc pas nécessaire de le mentionner. Il suffit de modifier le code pour désigner tout autre lecteur (de #00 à #03).
L'assemblage de ce fichier source produit le fichier LD.TAP, chargeur Basic qui met en place le code lui-même, code que je sauve sous le nom LD.BIN (#de 9801 à #9829) et que je place sur la disquette de test, ainsi que l'écran EXEMPLE01.SCR.
Il me faut aussi un programme Basic pour tester ce sous-programme. La figure 10 montre celui que j'ai utilisé.
Je sauve ce petit programme Basic sous le nom LD.BAS, pour compilation ultérieure. Je tente un petit RUN… Ouf ! Ça marche exactement comme espéré (figure 11), dévoilant le fichier EXEMPLE01.SCR (en l'occurrence, l'écran d'aide de CDA.COM, mon programme de vérification de disquette) (figure 12).
Je mouline LD.BAS avec le Compiler de Ray. J'obtiens LD.COM que je copie sur ma disquette de test COMPI3.DSK, qui contient déjà le fichier EXEMPLE01.SCR et le sous-programme LD.BIN. Je boote avec cette disquette de test, charge en mémoire LD.BIN et LD.COM (qui est en AUTO). Le Basic compilé se lance, réclame une touche (figure 13, page suivante) et affiche l'écran (figure 12). C'est gagné !

```
10 PRINT"TOUCHE SVP":GET R$
20 CALL#9801
30 PING:GET R$
```
Figure 10

Conclusion : Le Compiler de Ray peut accepter toutes les commandes de Sedoric et même les nombreuses routines présentes dans la Ram overlay et dont la liste se trouve dans le livre "Sedoric 3,0 à nu", pages 587 à 610. Seule contrainte : Il faut installer manuellement les paramètres nécessaires dans les bonnes variables. Mais tous ces paramètres et toutes ces variables sont décrits dans l'ouvrage cité ci-dessus, téléchargeable sur le site du CEO :
https://www.oric.org/ftp/ceo/ceomag/downloads/ebooks/sedoric3anu3ed.pdf

Sur la disquette qui accompagne cet article, vous trouverez le fichier Compiler-3.zip avec tous les programmes utilisés dans cet article.

Bon amusement avec votre Oric ! C'est vraiment une belle machine qui permet de passer d'agréables moments !

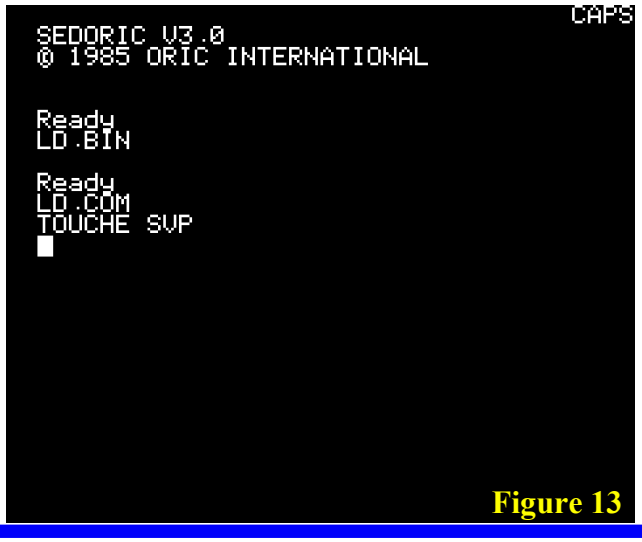

Figure 13