Manipulation des fichiers «composites» Basic + Langage machine

Par André C.

Je ne sais pas si vous êtes comme moi, mais j'ai du mal à me rappeler certaines syntaxes ou certaines procédures que je n'utilise pas souvent. C'est ce qui vient de m'arriver avec un fichier mixte Basic + LM. Je veux parler ici d'un cas de protection, malgré tout assez fréquent, que vous avez probablement déjà rencontré.

Il s'agit d'un fichier comportant un début Basic, immédiatement suivit par une zone comportant dans le meilleur des cas un programme en langage machine (par exemple des sous-programmes destinés à accélérer certaines tâches répétitives, trop lentes en Basic).

Mais hélas, cette zone peut aussi être inextricable: mélanges de bouts de codes, de data, de résidus de mise au point, en un mot, de pièges anti-copie.

Un des problèmes est que si on modifie la longueur de la partie Basic, ne serait-ce qu'en ajoutant un espace, le LM se plante en beauté (si j'ose dire). Effet il est rarement "relogeable", c'est même voulu.

Un autre problème est que si on utilise SAVE ou CSAVE sans indiquer les paramètres de début et de fin de programme, ces commandes considèrent qu'il s'agit d'un programme Basic et ne sauvent que la partie Basic.

Si on indique l'adresse de début et surtout l'adresse de fin (,A#0501,E#**ffff**), alors ces deux commandes sauvent bien l'ensemble (c'est déjà ça), mais avec le l'indicateur "bloc de données" au lieu de l'indicateur "Basic" que nous aurions voulu. En fait, il y a deux procédures différences pour manipuler ce type de fichier, selon que l'on utilise les commandes CSAVE/CLOAD ou SAVE/LOAD.

A) CSAVE/CLOAD

Disons tout d'abord que CLOAD ne pose aucun problème. CLOAD"nom", charge le fichier selon son entête cassette (voir Ceo-Mag n°153, page 17).

En résumé, pour une entête "Sedoric" (NB. la bande amorce est réduite à 4 octets #16), on a (en hexadécimal): 16 16 16 16 24 00 00 **tt aa ffff dddd** 00 nom_de_fichier 00 suivit du programme proprement dit.

Dans cette syntaxe, **tt** est le drapeau "type de fichier" (00 si Basic, 80 si LM ou bloc de données, 40 si tableau), **aa** est le drapeau "auto" (différent de zéro si Auto, sinon Stop), **ffff** est l'adresse de fin et **dddd** est l'adresse de début.

CLOAD"nom" charge en mémoire de **dddd** à **ffff**, ajuste le flag Basic selon le drapeau **tt** et lance l'exécution si le drapeau **aa** est différent de zéro.

Donc, dans le cas qui nous occupe, il suffit de disposer d'un fichier "composite" avec les bonnes adresses et les bons drapeaux. CLOAD, qui prend en charge tous les cas de figure, la chargera correctement.

Pour CSAVE, c'est plus compliqué.

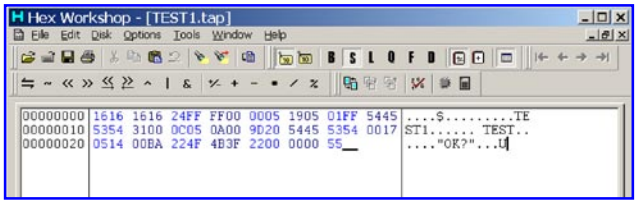
D'une part CSAVE"nom"[,AUTO] sauve un fichier de type Basic en utilisant les pointeurs Basic (adresse de début en **#9A-#9B** et adresse de fin en **#9C-#9D**). Attention, ces pointeurs sont automatiquement mis à jour dès qu'on touche au programme Basic.

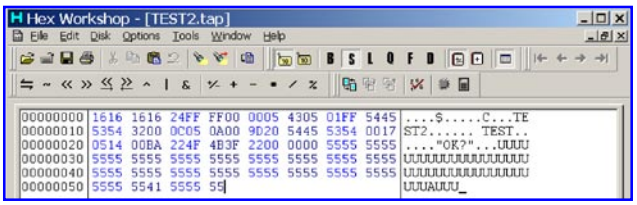
D'autre part, CSAVE"nom",A#**dddd**,#**ffff**[,AUTO] sauve un fichier de type bloc de données en utilisant les adresses indiquées. L'entête de ce fichier est fabriquée avec ces mêmes données: **tt, aa, dddd** et **ffff**. Le drapeau **tt** est automatique mis à #80 (bloc de données).

Bon, tout ça c'est bien beau, mais comment fabriquer un fichier "composite" ? Il y a un truc simple à savoir: il faut tromper le système en plaçant une fausse adresse de fin en #9C-#9D, juste avant de faire CSAVE"nom"[,AUTO]. La "fausse" adresse à indiquer est en fait celle de la fin de la partie bloc de donnés. Cette commande

```
 37631 BYTES FREE

Ready
PRINT HEX$(DEEK(#9A))
#501

Ready
PRINT HEX$(DEEK(#9C))
#503

Ready
10 REM TEST
20 PRINT"OK?"

PRINT HEX$(DEEK(#9A))
#501

Ready
PRINT HEX$(DEEK(#9C))
#519

Ready
CSAVE"TEST1"

Ready
```

```
Ready
PRINT HEX$(DEEK(#9C))
#503

Ready
10 REM TEST
20 PRINT"OK?"

PRINT HEX$(DEEK(#9A))
#501

Ready
PRINT HEX$(DEEK(#9C))
#519

Ready
CSAVE"TEST1"

Ready
DOKE#9C,#543

CSAVE"TEST2"

Ready
```





sauvera alors un fichier "composite" Basic + bloc de données en lui affectant un drapeau "Basic". Démonstration: Les quatre figures ci-dessus parlent d'elles-mêmes.

B) SAVE/LOAD

La commande LOAD"nom",V affiche les adresses de début et de fin du fichier et son type (voir manuel Sedoric page 100). Cette commande lit ces données dans un secteur de la disquette nommé descripteur placé devant les secteurs contenant le fichier proprement dit. Une sorte d'entête en quelque sorte.

Note: Comme vous le savez, en pratique (sauf si le nom de fichier contient un mot-clé) le mot "LOAD" est implicite et peut être omis.

Comme la commande CLOAD, la commande LOAD ne pose pas de problème. Elle gère correctement toutes les situations. Un fichier "composite" sera donc chargé selon le descripteur. Les pointeurs et drapeaux seront mis à jour en fonction.

Mais attention, comme avec CLOAD, le fichier "composite" chargé par LOAD doit être correct, sinon, on aura une erreur, voire un plantage.

Par contre, comment sauver un fichier "composite" ? D'abord, si l'on veut sauver l'ensemble Basic + bloc de données, il faut impérativement utiliser la syntaxe SAVE"nom",A#**dddd**,#**ffff**[,AUTO].

Ensuite, si on veut modifier son type il faut intervenir, non sur le fichier lui-même, mais sur son descripteur avec un éditeur hexadécimal.

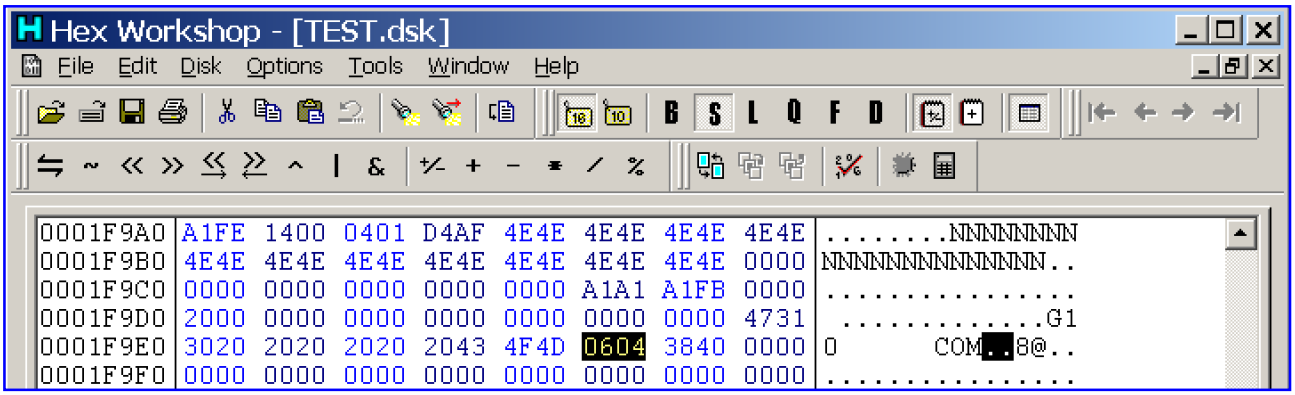
Si votre disquette contient de nombreux fichiers, je vous conseille de créer une disquette vierge et d'y copier le fichier en question. Vous y retrouverez plus facilement vos petits.

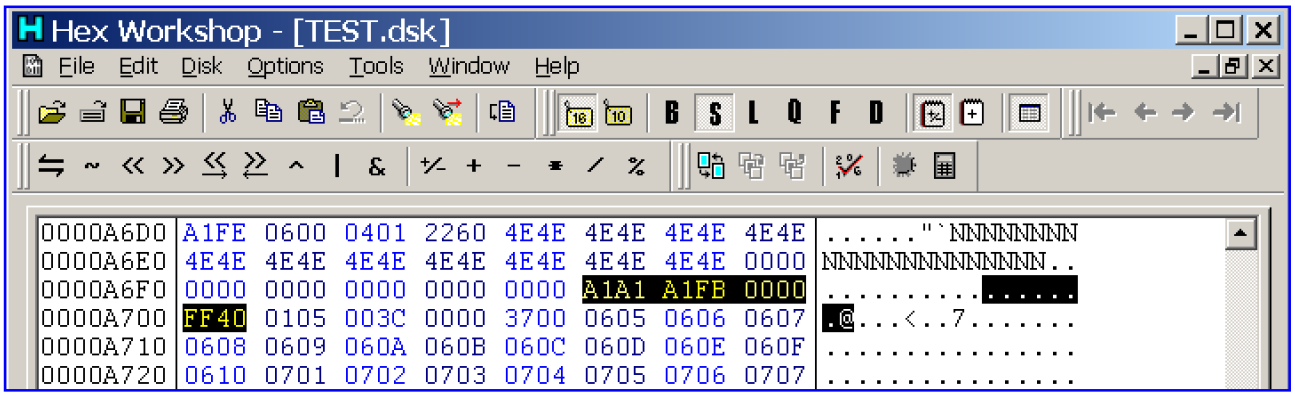
Pour illustrer mon propos j'ai pris l'exemple d'un fichier G10.COM situé sur la disquette TEST.DSK (c'est l'unique fichier de la disquette).

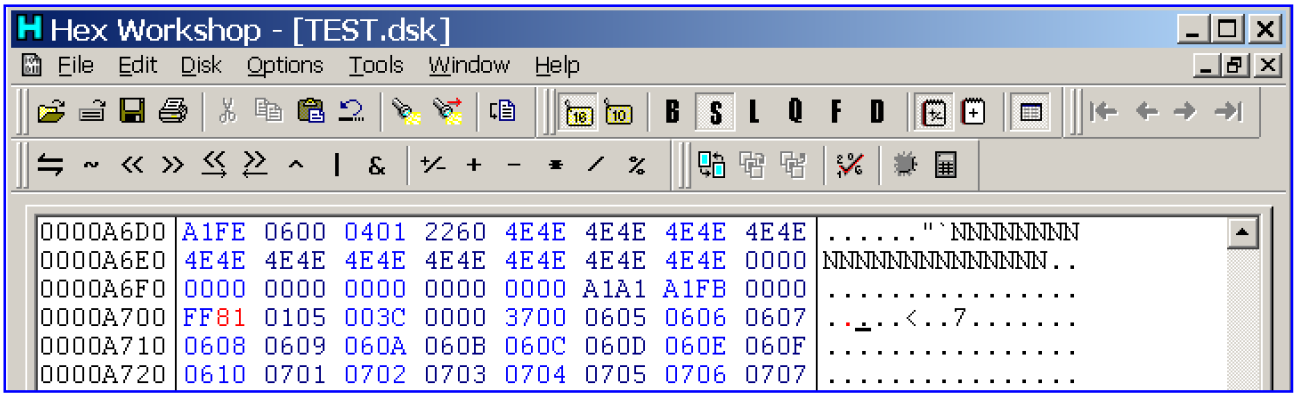
Ce pourrait être une création personnelle (si j'étais vicieux), mais dans cet exemple particulier, il a été importé avec un CLOAD"G10", suivit d'un SAVE"G10",A#0501,E#3C00.

Ce fichier est composé d'une partie Basic de #0501 à #1309 et d'une partie "poubelle" de #130A à #3C00, qui contient quelques bribes de langage machine et de plusieurs paquets de data perdus au milieu d'octets non significatifs.

Il faut ouvrir le fichier "dsk" dans l'éditeur et à l'aide de la fonction "recherche", trouver le premier secteur de directory en cherchant la chaîne hexadécimale A1 FE 14 00 04, soit octet de synchronisation (#A1), flag de début de secteur (#FE), numéro de piste (#14), numéro de face (#00) et numéro de secteur (#04).







On y voit immédiatement (première figure ci-dessus) le nom du fichier (9 octets, ici G10), suivi de son extension (3 octets, ici COM), suivit du numéro de piste **pp** (ici **#06** en surbrillance), suivit du numéro de secteur **ss** (ici **04** en surbrillance). Il faut alors trouver le secteur descripteur de ce fichier en cherchant la chaîne hexadécimale A1 FE **pp** 00 **ss**, ici A1 FE **06** 00 **04**.

On arrive alors dans le descripteur (2e figure ci-dessus). Les 2 premiers octets après **#FB** sont l'adresse du descripteur suivant (**00 00** s'il y a un seul descripteur, ce qui est le cas ici). Ensuite vient l'octet **#FF**, facile à voir, et tout de suite après, le type du fichier: 80 si Basic (81 si Auto), 40 si bloc de données (41 si Auto) etc. Ici on peut voir **40** pour type de fichier. On remplace par **81** (Basic Auto) et on sauve. Un LOAD"G10",V montre le résultat (figure ci-contre).
Mission accomplie.

En conclusion.

Je pense que vous disposez maintenant de toutes les informations nécessaires pour gérer les fichiers "composites". Je suis entré dans les détails afin que cet article puisse être utilisé par tout un chacun. Si toutefois j'avais oublié quelque chose ou si vous aviez un problème, n'hésitez pas à me contacter via le CEO ou mon site Internet http://andre.cheramy.net/oric.htm