LE BASIC COMPILER DE RAY McLAUGHLIN

par André C.

Voici une petite merveille Oricienne, qui est largement méconnue et quasiment inutilisée. Pourtant c'est un joyau qui vous sera très utile, aucun doute là-dessus. Je vous invite à y consacrer quelques instants, cela devrait être un investissement très rentable pour vous.

Un compilateur Basic est un logiciel capable de convertir un programme Basic en programme Langage Machine (LM).

Il y a bien longtemps, le CEO-mag (n°42, octobre 1993, page 6) publiait un banc d'essai des compilateurs Basic. Cet article était repris de celui de Nicholas Haworth, paru dans OUM. Excellent petit article. Deux logiciels étaient en lice :

1) LM Plus d'Isosoft.

2) BASIC COMPILER de Ray McLaughlin.

Ce choix limité est un peu dommage, car il y en a d'autres : Le Basic Compiler de Oasis (éditeur Anglais) et encore au moins un dont j'ai oublié le nom. Mais c'est vrai que ce n'est pas facile à trouver.

En fait, il ne s'agit pas vraiment d'un banc d'essai, mais d'une page de pub. Le logiciel de Ray est largement plébiscité et c'est mérité. En outre cet article donne plein d'informations sur ce qu'est un compilateur Basic, ses possibilités et ses limitations. Je vous conseille donc de vous y reporter.

J'ai eu la curiosité d'essayer ce fameux BASIC COMPILER. Pas de bol, il n'était pas disponible sur oric.org (c'est corrigé). J'ai donc fais un peu d'archéologie dans mes archives et je l'ai retrouvé. Je l'ai peut-être eu à l'époque où Yann L., Ray et moi avons travaillé au livre "Sedoric à nu".

Bref, en piste avec la configuration recommandée : Atmos avec Sedoric (sous Euphoric pour avoir de belles recopies d'écran).

COMPILER.DSK est une disquette Sedoric 2.1 "customisée". La fig. 1 vous en montre le directory. Au boot, un écran-titre affiche un menu (fig. 2). Au vu des 3 premières options, on comprend déjà que l'opération s'effectue en 3 étapes :

1) Compilation du programme Basic, qui produit plusieurs fichiers source en assembleur.

2) Assemblage de ces fichiers source, qui produit plusieurs fichiers en code machine.

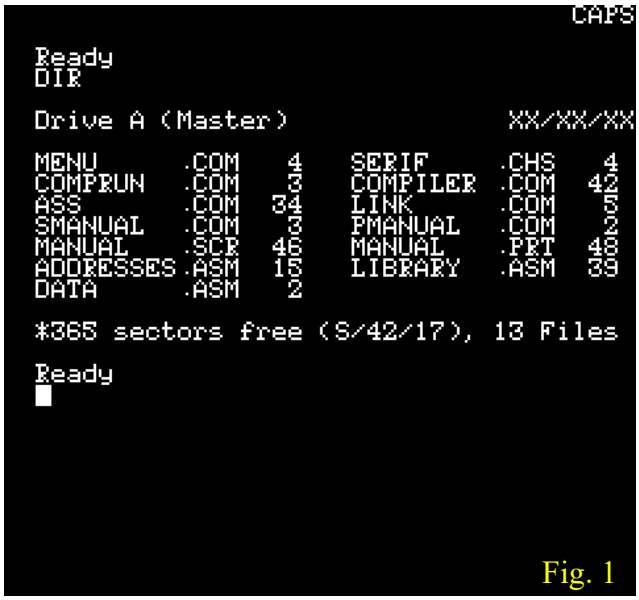

Fig. 1

3) Linkage, qui établit les liens entre les différents morceaux de code machine.

Les options 4 et 5 proposent un mode d'emploi (à l'écran ou à imprimer). Ce n'est pas de refus.

L'affichage à l'écran sous forme de texte qui défile n'est pas très pratique, car le texte, en anglais, est délicat à analyser. J'opte donc pour l'impression et obtiens quatre pages. Au premier abord, cela ne semble pas simple, car je ne connais pas de nombreux termes anglais utilisés (literal, scalar, etc.). Seule solution : se jeter à l'eau. Mais j'accumule échec sur échec. Ça devrait pourtant

Fig. 2

Fig. 3

Fig. 4

bien marcher. Je dois être un peu c.. J'essaie de me tranquilliser en me disant que c'est seulement mon anglais qui flanche ! Le fait est que les indications pour comprendre ce qui cloche sont quasi inexistantes, voir erronées (du genre : FILE NOT FOUND ERROR alors qu'elle y est, INVALID FILE NAME ERROR alors que ce n'est pas le cas, voire THIS FILE WILL RUN AUTOMATICALLY, alors que son exécution ne produit rien !).

Finalement j'y suis arrivé et je vous livre une méthode qui va bien, même si ce n'est pas la seule et si j'ai écarté d'autres possibilités. Il faut respecter quelques conditions élémentaires et un ordre strict dans la procédure, il faut notamment rebooter entre chaque phase. Notez qu'il faut penser à faire un DEL"CODE?.BIN" entre chaque essai pour effacer les traces résiduelles de l'essai précédent.

LA PROCEDURE QUI VA BIEN

1) Disposer d'un programme Basic complètement finalisé (par exemple NOM.BAS).
2) Booter, choisir l'option 1 (COMPILER) et taper COMPILE"NOM.BAS".
3) Booter, choisir l'option 2 (ASSEMBLER), taper A (drive utilisé) et espace en fin de procédure.
4) Le système reboote, taper A (drive utilisé) puis un nom pour sauvegarder le programme.
5) Tester le programme obtenu.

Comme vous le voyez, c'est on ne peut plus simple. Il faut bien sûr indiquer au COMPILER le nom du programme Basic à convertir et au LINKER le nom du programme LM à sauvegarder. Entre temps la procédure se déroule de façon quasi transparente en travaillant sur des fichiers prénommés.

PREMIERE PETITE EXPERIENCE

Soit le programme Basic (basique?) suivant :

```
10 FOR I=1 TO 4:PRINT I:NEXT
20 PING
```

Ce programme permet de voir ce qui se passe à l'écran et d'avoir un témoin auditif. Tapez-le, vérifiez qu'il fonctionne correctement et sauvegardez-le sous F.BAS (ou autre nom à votre guise) sur une disquette Sedoric normale (en AUTO ou en STOP, c'est sans importance). Evitez de le taper directement sur la disquette COMPILER.DSK (il sera impossible de corriger les erreurs, car la commande LIST a été neutralisée). Certes le petit Basic proposé ici est simpliste, mais en conditions réelles, il faudra tester très soigneusement votre programme. En effet, pendant la compilation, les erreurs résiduelles n'afficheront pas de message d'erreur Basic, mais planterons le processus de conversion ou pire le code LM produit ne fera rien au lancement et ce sans explication.

1) Bootez avec la disquette COMPILER.DSK. Au menu, choisissez l'option 6 - BASIC pour retourner au Ready et copiez sur la disquette COMPILER.DSK le programme Basic préparé.
2) Bootez, choisissez l'option 1 - COMPILER. Un écran titre s'affiche (fig. 3). Tapez COMPILE"F.BAS". Le programme COMPILER produit une série de fichier .ASM (programmes source pour l'assembleur) (fig. 4) et affiche le nombre d'erreur, qui doit être de zéro pour pouvoir continuer (sinon, revoir le programme Basic à la recherche d'erreurs ou de commandes interdites, voir le paragraphe LIMITATIONS).
3) Bootez, choisissez l'option 2 - ASSEMBLER.

Fig. 5

Fig. 6

Un écran titre s'affiche (fig. 5) et le programme demande quel est le drive source. Répondre A (tout court). Le programme assemble une série de fichiers .ASM, produit leurs homologues en .BIN (fig. 6), indique le nombre d'erreurs, affiche l'adresse de fin du code LM et enfin invite à retourner au menu. Le système reboote sur le menu.

4) Choisissez l'option 3 - LINKER. Indiquez le drive où se trouvent les fichiers .BIN. Dans notre configuration, il faut taper A (+ RETURN cette fois !). Le LINKER traite les différents fichiers .BIN et affiche leur adresse en mémoire, puis il réclame un nom pour sauver le programme LM produit et affiche de manière optimiste "The file will run automatically. You may run it now by entering A-F.COM" (le fichier démarrera automatiquement. Vous pouvez le lancer en tapant A-F.COM) (fig. 7).

5) Tester le programme LM obtenu. C'est le moment de vérité. Avec notre exemple simpliste, vous aurez la joie de constater que le programme fait ce qui était prévu (fig. 7).

Avec un programme plus compliqué, il se peut que vous soyez désappointé s'il ne se passe rien ou si le résultat n'est pas celui que vous vouliez. Vous êtes alors le bec dans l'eau sans savoir quoi faire et ce n'est pas complètement de votre faute.

En effet, l'encadrement de la procédure laisse à désirer. Le BASIC COMPILER n'a rien vu venir ou, en tout cas, n'a rien dit. Il faut reprendre avec plus de soin. En premier lieu, revoir le listing Basic. En absence de commande LIST, il faut rebooter avec une disquette Sedoric normale. Lorsque votre programme Basic sera bien propre

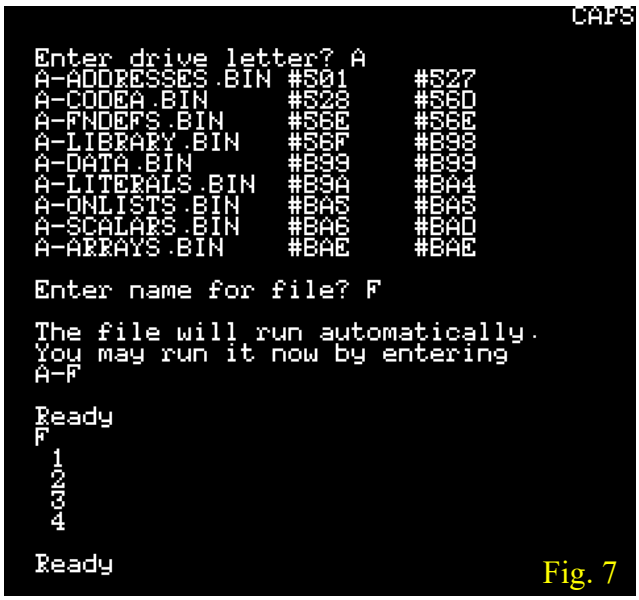

Fig. 7

et si vous suivez pas à pas la procédure que je vous ai indiquée, il ne fait aucun doute que finalement cela marchera (je parle d'expérience).

TAILLE DU PROGRAMME LM OBTENU

Vous serez probablement surpris par la taille du fichier produit. Notre petit programme Basic comportait 41 octets. Son homologue LM, qui fait le même job et pas plus, en comporte 1709. Cela vient du fait que le Basic doit interpréter le programme, analyser la syntaxe, rechercher les bonnes routines, envoyer les paramètres etc. Il fait appel à des routines en Rom et derrière ces 41 octets, il y a des centaines d'octets de code en Rom. Au contraire le programme LM est exécuté directement, "bêtement" si je puis dire. Le plus curieux dans cette histoire est que l'exécution du code LM, dont la taille est en apparence supérieure, est beaucoup plus rapide (200% selon l'article cité).

DEUXIEME PETITE EXPERIENCE

Pour apprécier cette différence de rapidité, nous allons voir ce qui se passe avec un programme qui met beaucoup plus de temps à s'exécuter. Soit le programme Basic ECRAN.BAS suivant :

```
100 FOR AS=33 TO 38
110 AD=#BB80
120 FOR Y=0 TO 27
130 FOR X=2 TO 39
140 POKE AD+X+Y*40,AS
150 NEXT X
160 NEXT Y
170 NEXT AS
```

Ce programme couvre complètement l'écran texte successivement avec les caractères ! (Ascii 33), puis " (Ascii 34), puis # (35), puis $ (36), puis % (37) et enfin avec & (38). Un bon exemple de démo stupide ! L'exécution de cette version Basic prend 89 secondes. ECRAN.COM, la version LM obtenue avec BASIC COMPILER, pédale beaucoup plus vite et ne met que 30 secondes. Elle est donc 3 fois plus rapide ! C'est un résultat spectaculaire, voire inespéré.

RELOCATION DU CODE LM OBTENU

Le code LM produit est stocké d'office en mémoire à la place du programme Basic (à partir de #0501 donc). C'est sans importance s'il doit tourner de manière autonome (un petit jeu par exemple). Mais cela pose problème si l'on veut en faire une routine LM appelable avec un CALL. Ce sera le cas, soit si le programme Basic est trop gros pour être mouliné en entier par BASIC COMPILER, ou soit si vous désirez garder la structure générale en Basic et n'utiliser du code LM que pour les sous-programmes trop consommateurs de temps. Il faudra donc reloger cette routine en mémoire ailleurs qu'en #0501, ce qui implique d'en adapter les adresses internes. Certains moniteurs peuvent effectuer ce travail et possède une commande pour déplacer et adapter du code LM. C'est le cas, par exemple, du Moniteur d'André Chénière, qui est d'ailleurs lui-même auto-relogeable. Il s'agit d'un programme très compact qui pourra cohabiter avec le code à reloger. Dans la disquette accompagnant cet article, vous trouverez un pack "BasicCompiler.zip" contenant les fichiers Compiler.dsk et CheniereAndre.zip.

Dans sa forme native, ce moniteur occupe la zone #AC00-#B44C (vérifiable en tapant MONAC1,V). Le code LM, lui, est situé à partir de #0501 et n'a aucune chance d'atteindre le début

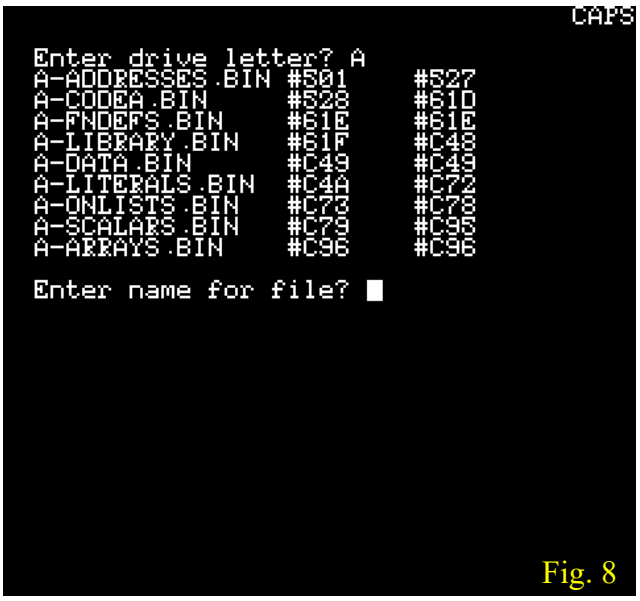

Fig. 8

du moniteur (vérifiable pour le 2e petit exemple testé ci-dessus, en tapant ECRAN.COM,V ce qui donne #0501-#0C96). Ce sera donc largement compatible dans tous les cas.

La syntaxe du moniteur est simple :

Ldddd pour désassembler (L comme List) (où dddd est l'adresse hexadécimale de début).

Mdddd-ffff>nnnn pour déplacer (M comme Move) (adresse de début, adresse de fin et nouvelle adresse).

Rdddd-ffff pour reloger (R comme Relocate) (adresses hexadécimales de début et de fin).

<**RETURN**> pour sortir du moniteur et retourner au Ready du Basic.

LA PROCEDURE QUI VA BIEN POUR RELOGER LE PROGRAMME LM

1) Déterminer la zone occupée par le programme LM à déplacer et notez les adresses de début et de fin.

2) Dans cette zone, déterminez ce qui revient au code LM et ce qui est des data. Il faut ici rendre hommage à Ray qui a prévu le coup et groupé le code en début de zone et les data en fin de zone. Le dernier écran du LINKER (fig. 8) indique les zones occupées par chacun des fichiers linkés : Les quatre premiers (ADDRESSES. BIN, CODEA.BIN, FNDEFS.BIN et LIBRARY.BIN) correspondent à du code LM. Les cinq derniers (DATA.BIN, LITERALS.BIN, ONLISTS.BIN, SCALARS.BIN et ARRAY.BIN) sont des data. Par curiosité, notez que DATA.BIN correspond aux commandes DATA du programme Basic, LITERALS.BIN correspond aux constantes (numériques et chaînes),

ONLISTS.BIN aux n° des lignes qui ont un rôle d'étiquettes, SCALARS.BIN aux variables (numériques et chaînes) et ARRAY.BIN aux tableaux.

Dans notre exemple le code s'étend de #0501 à #0C48. On peut vérifier ça avec la commande Ldddd du moniteur. Dans notre exemple il faut taper L0501 puis L0C49. On trouve du code assembleur cohérent de #0501 à #0C48 et "n'importe quoi" dans le reste de ECRAN.COM.

3) Charger ECRAN.COM,N puis MONAC1. Le moniteur se lance. Déplacer ECRAN.COM avec la commande Mdddd-ffff>nnnn. Dans notre exemple il faut taper M0501-0C96>2501.

4) Reloger les adresses internes de la ou des parties codes avec la commande Rdddd-ffff. Notez que la commande R doit être tapée immédiatement après l'exécution de la commande M sans rien toucher entre les deux. Dans notre exemple il faut taper R2501-2C48.

5) Retour au Basic en pressant <RETURN>.

6) Sauvegardez le nouveau programme LM. Dans notre exemple il faut taper SAVE"ECRAN2",A#2501,E#2C96. Notez au passage l'intérêt de choisir une nouvelle adresse nnnn qui soit homologue de l'adresse d'origine. Cela facilite le calcul de la nouvelle adresse de fin.

7) Tester le nouveau programme en LM. Dans notre exemple, ECRAN2.COM fonctionne exactement comme ECRAN.COM (et toujours aussi rapidement !). Simple non ?

Si la relocation échoue, il y a une autre procédure possible, qui consiste à édite le fichier ADDRESSES.ASM (voir la notice du BASIC COMPILER).

LIMITATIONS

Bon ! Tout ça c'est bien beau, mais il y a quand même quelques limitations.

1) Les commandes LIST et LLIST ont été neutralisées sans doute pour éviter de lister par inadvertance dans le code LM situé à partir de #0501. Il est donc impossible de voir le programme Basic que l'on a en mémoire. C'est un peu pénible de travailler en aveugle, surtout en cas de problème.

2) La taille du programme Basic compilable est imitée par la mémoire disponible. Selon Nicholas Haworth, cette taille peut aller jusqu'à 10 à 15 Ko. Mais ça fait déjà pas mal. 12 Ko, par exemple c'est la taille d'un programme Basic qui s'étendrait de #0501 à #3501 en mémoire et occuperait 48 secteurs sur la disquette.

3) Certaines commandes ne peuvent plus être utilisées : EDIT, STORE, RECALL, CLOAD, CSAVE, TRON, TROFF et CONT. Mais cela n'a rien de critique.

4) Tous les tableaux doivent être déclarés en début de programme (pas de dimensions par défaut). Les variables ne peuvent être utilisées pour dimensionner les tableaux, ni pour les n° de ligne après GOTO, GOSUB, THEN, ELSE, etc. Cela peut être un peu gênant.

5) La commande CTRL+C n'est plus gérée, sauf pour les entrées de GET et INPUT.

Au chapitre des bonnes nouvelles, le compilateur accepte que la commande CALL soit suivie d'une liste de paramètres, mais le programme doit avoir été conçu pour gérer ces paramètres. La commande RESTORE de Sedoric est correctement compilée. D'ailleurs, presque toutes les commandes Sedoric sont acceptées. Pour les commandes Sedoric associées à des variables réservées, il faut déclarer ces variables avant d'utiliser ces commandes. Pour avoir des informations plus précises, reportez-vous au manuel de BASIC COMPILER présent sur la disquette. Mon anglais rustique ne me permet pas de tout comprendre et je risquerais de vous induire en erreur.

CONCLUSION

Si vous n'avez pas perçu mon enthousiasme, c'est sans espoir pour ma prose. Il faut que je me recycle dans une autre activité ! Voilà un outil simple à manipuler qui peut grandement accélérer l'exécution de vos programmes Basic. C'est le moment de vous lancer dans l'écriture d'un nouveau jeu, écriture qui sera quand même plus facile en Basic qu'en assembleur, en tous cas pour la plupart des Oriciens.

Bon amusement et à bientôt.

PS. Vous trouverez le Basic Compiler de Ray McLaughlin et le Moniteur d'André Chenière sur la disquette qui accompagne ce CEO-mag, sur le site oric.org < https://www.oric.org/software/> et sur la page : http://andre.cheramy.net/telechargement/Programmes/choix.htm. Pour télécharger sur oric.org, pensez à vous munir de vos login et mdp (les mêmes que pour télécharger le CEO-mag).