Conversion de l'Hyper-Basic vers le Basic 1.0/1.1

par André C.

Pour quoi faire ?

C'est bien l'idée la plus farfelue qui me soit jamais venue et dont je me suis mordu les doigts par la suite. Pourquoi convertir des programmes Hyper-Basic en Basic 1.0/1.1? Pas loin d'un millier de programmes Oric tournent déjà sur Oric-1 ou Atmos alors qu'il y a bien moins de programmes en Hyper-Basic et encore moins qui restent sans équivalent pour Oric-1/Atmos. Alors quel est le besoin? Ben ! Le challenge œuf corse !

Etat de la question

Autant vous dire tout de suite que c'est presque mission impossible. En effet, les possibilités de l'Hyper-Basic sont bien plus étendues que celle du Basic 1.x (même en ajoutant les commandes Sedoric pour comparer de manière plus équitable). Ce n'est pas tant le nombre des mots-clés (211 pour Telestrat au lieu de 202 pour Basic 1.x + Sedoric) qui fait la différence, que surtout les énormes possibilités des commandes de l'Hyper-Basic.

Au chapitre des possibilités par exemple, si vous cherchez à faire de la télématique en ajoutant une carte RS232 à votre Oric-1/Atmos, vous aurez à ré-écrire une trentaine de commandes spécifiques…

Des différences profondes entre les systèmes

Pour commencer, l'adresse d'implantation des programmes Basic est différente. En fait, la carte mémoire du Telestrat est très différente de celle de l'Oric-1/Atmos. Les adresses mémoires sont complètement différentes, surtout les pages système 0, 2, 3, 4, 5 et 6 (tiens, 3 nouvelles pages système !) et il faudra courir après des équivalences pas toujours évidentes (si elles existent) en absence de documentation sur ces pages système.

Autre exemple, le codage des touches du clavier utilisé par l'Hyper-Basic est différent de celui en vigueur avec Oric-1/Atmos. Allons bon ! Il ne manquait plus que ça ! Par exemple, les touches gauche, bas, espace, haut et droite, sont respectivement codées #AC, #B4, #84, #9C et #BC (manuel Sedoric, page 104) alors qu'avec l'Hyper-Basic on a respectivement #85, #86, #80, #83 et #87. La détection des touches pressées va devoir être adaptée…

Heureusement certaines adresses n'ont pas bougé, comme par exemple celles de l'écran ou celles des jeux de caractères. C'est déjà ça !

Vous avez dit "quelle syntaxe ?"

Pour les commandes qui sont présentes dans les deux systèmes, la syntaxe est souvent différente. De quoi s'emmêler les crayons (Ah ! Les vieilles habitudes!).

Voici un exemple frappant : Pour l'Hyper-Basic, on a "PRINT@Y,X;..." et pour le Basic 1.x on a "PRINT@X,Y;…". Je suppose que Fabrice Broche à conduit une guerre d'optimisation du code pour économiser le moindre octet, mais quand même, Il fallait le faire !

Les modifications sont parfois plus positives : Alors que le POKE du Basic 1.x permet seulement de placer des valeurs de 0 à 255 en mémoire (y compris dans l'écran), celui de l'Hyper-Basic permet aussi de placer des chaînes de caractères ! Quand je vous disais que les possibilités de l'Hyper-Basic sont très étendues ! Pour émuler ça avec le Basic 1.x, il faut utiliser une boucle FOR… NEXT, découper les chaînes en caractères individuels à convertir en Ascii avant de les POKEr (sans oublier le calcul de l'adresse où POKEr).

Autre amélioration : Le PLOT et le PRINT@ de l'Hyper-Basic permettent d'afficher sur la ligne service, ce qui n'est pas le cas pour le PLOT et le PRINT@ du BASIC 1.x. Il s'en suit une différence dans les ordonnées : Y va de 0 à 27 pour l'Hyper-Basic et de 0 à 26 pour le Basic 1.x : Ça va faire pas mal d'ajustements pour afficher dans l'écran proprement dit (penser à modifier aussi les ordonnées calculées) ! Pour afficher sur la ligne service, il faudra remplacer les PLOT et les PRINT@ de l'Hyper-Basic par un découpage des chaînes, comme évoqué ci-dessus et POKEr les caractères un à un.

Beaucoup des nouvelles caractéristiques de l'Hyper-Basic sont facilement transposables en Basic 1.x :

- Le système de labels des sous-programmes est très pratique, mais il faut y renoncer et revenir au GOSUB n°_de_ligne.
- WHILE… WEND peut être émulé avec REPEAT… UNTIL en ajoutant un test IF au début.
- COUNT… UNCOUNT peut être ramené à une boucle FOR… TO NEXT.

- Les commandes KEY SET… KEY OFF de Sedoric peuvent remplacer CLI… SEI, etc.

Attention, l'Hyper-Basic autorise des noms de variables de 256 caractères (si, si, si… !). La lecture du programme en devient plus évidente. Exemple, le nom de variable GESTIONCHIEN est plus parlant que GC, son équivalent en Basic 1.x (limité à 2 caractères significatifs maxi). L'adaptation se complique lorsque vous avez 5 ou 6 variables qui commencent par les 2 mêmes lettres… Mais bon, c'est faisable.

Nouvelles commandes

Outre la classique commande RESET (redémarrage "à froid", c'est-à-dire comme à l'allumage), l'Hyper-Basic offre une nouvelle commande, NMI, qui effectue un redémarrage "à chaud", c'est-à-dire sans perte du contenu de la mémoire (programme et variables sont conservées). Pour transposer ça, il n'y a guère qu'un CALL#C471 (Atmos) ou CALL#C475 (Oric-1).

Outre les commandes télématiques, le Telestrat offre beaucoup d'autres commandes originales. Ces nouvelles commandes sont bien pratiques et certaines peuvent être réécrites en Basic 1.x. Exemples :

- CURSOR SET équivaut à : POKE#26A,(PEEK(#26A)OR#01) et
- CURSOR OFF équivaut à : POKE#26A,(PEEK(#26A)AND#FE).
- MIDDLE$ sera un peu plus compliqué à émuler. Cette fonction très pratique centre l'affichage d'une chaîne dans un espace donné, par exemple la largeur de l'écran. C'est faisable avec les commandes de manipulation de chaînes du Basic 1.x, mais évidemment beaucoup plus lourd.

Certaines commandes sont quasiment inutiles :

- UP$ (chaîne) convertit en majuscules les caractères d'une chaîne.
- LO$ (chaîne) convertit en minuscules les caractères d'une chaîne.
- LOB$ (chaîne) idem, mais la première lettre reste en majuscule.
- SPC$(n) génère une chaîne de n espaces.

Si besoin, on peut facilement les émuler tout ça. D'autres commandes ont simplement changé de nom :

- TRON TROFF devient TRACE SET, TRACE OFF.
- LPR SET LPR OFF devient PR SET PR OFF

etc. (était-ce bien nécessaire de changer?).

Mais je ne vais pas faire ici la revue des différences, cela serait très barbant et ne présenterait pas beaucoup d'intérêt.

Et pour la fin, le plus grave sans doute...

Contrairement au Basic 1.x, qui est un langage interprété, l'Hyper-Basic est un langage semi-compilé. Ne me demandez pas comment ça marche, mais le résultat est BEAUCOUP plus rapide (voir l'excellent article de Fabrice F. dans le CEO-mag de mars 2018). Nous avons également vu plus haut que nombre de commandes Hyper-Basic (donc code langage machine en Ram overlay) doivent être remplacées par des acrobaties en Basic 1.x (donc moins réactives).

En conséquence, un jeu d'arcade en Hyper-Basic sera bien plus rapide que son équivalent en Basic 1.x. Pour peu qu'il soit bien écrit et bien optimisé, vous n'arriverez pas à le transposer en Basic 1.x ou alors il faudra accepter qu'il se traîne un peu ! Par contre, un jeu conçu directement en Basic 1.x utilisera de manière optimale les moyens du Basic 1.x et avec un peu d'imagination, cela peut donner un résultat équivalent. En tout état de cause, ne désespérez pas du Basic 1.x, il a largement démontré ses possibilités !

La perte de réactivité est incontournable. Mais c'est seulement vrai si on reste en pur Basic. Il est possible de corriger ce problème, soit en utilisant un compilateur Basic, soit en utilisant des sous-programmes en langage machine pour les parties les plus consommatrices de temps. La commande CALL est faite pour ça.

Conclusion

Vous l'avez déjà compris, la conclusion de tout ça, c'est qu'il n'est PAS RENTABLE de transposer un programme Hyper-Basic en Basic 1.x. Si je me permets d'écrire ça, c'est qu'à titre d'exercice, j'ai adapté le Pac-Girl de Dominique Lamy pour qu'il puisse être exécuté sur un Atmos. Ce fut une expérience instructive, mais pas facile ! J'y suis bien sûr arrivé, mais que de travail pour un résultat plus ou moins convainquant !

La solution la plus simple et la plus rapide, si vous tenez absolument à faire tourner un programme Hyper-Basic sur un Oric-1 ou un Atmos, consiste à le décortiquer pour voir ce qu'il fait et comment il le fait, puis à le récrire directement en Basic 1.x en utilisant au mieux les possibilités de ce Basic, sa "philosophie", ses petites combines et une bonne dose d'imagination. Ne pas s'acharner à lui faire faire quelque chose qu'il peine à exécuter et remplacer ce quelque chose par un équivalent plus adapté.

De tout temps et quel que soit le domaine, la limitation des moyens conduit toujours à une plus grande créativité...