La commande USER de Sedoric

par André C.

Comment je suis tombé dedans…

Il y a quelques temps, Jérôme D. nous a contacté pour nous signaler une différence incompréhensible entre le code de Sedoric 3.0, tel qu'on peut le lire dans "Sedoric à nu" et le code trouvé dans Sedoric 4.0 et Stratoric 4.0. Après enquête, Jérôme a bel et bien mis le doigt sur une anomalie. Une paire d'octets #00 a accidentellement atterrit au milieu de la commande USER. Un nouveau kit a été diffusé. Les fichiers corrigés sont datés du 01/07/2017, ceci afin d'éviter une confusion avec la version précédente.

La commande USER

Sedoric est plein de commandes géniales, par exemple SEEK, CHANGE, TKEN, UNTKEN, STRUN, etc. et… USER. La mésaventure de Sedoric 4.0 m'a replongé dans cette commande absolument géniale et je ne peux m'empêcher d'attirer votre attention dessus, car le manuel n'explique pas assez clairement son mode d'emploi, de plus la syntaxe décrite comporte une erreur (il manque une virgule), ce qui n'a pas été pour encourager les utilisateurs de Sedoric...

USER permet d'appeler des routines en "Langage Machine" situées n'importe où, même en Ram Overlay (contrairement à CALL), avec passage et retour de paramètres. En standard, les registres A, X, Y et P peuvent être initialisés à l'appel de la routine et récupérés en sortie. Comme avec CALL, tout autre paramètre peut être également utilisé, mais cela demande une analyse, par la routine elle-même, de ce qui suit la commande et d'afficher ou sauvegarder les paramètres désirés avant le retour. Mais c'est un autre problème, éventuellement à revoir une autre fois. En résumé, contrairement à CALL, la commande USER permet :

1) de travailler directement en RAM overlay
2) d'affecter directement les registres du 6502.

Syntaxe : Deux formes sont utilisables.

```
USER n°_de_routine,DEF adresse (,O)
USER n°_de_routine (,A valeur)(,X valeur)(,Y valeur)(,P valeur)
```

Dans les 2 cas le numéro de la routine indiqué est un chiffre de 0 à 3 ou une expression numérique de valeur 0 à 3 (on peut imaginer une redirection de genre ON… GOTO…). Il est donc possible, avec la commande USER, d'utiliser 4 sous-programmes en "Langage Machine" au maximum.

La première forme sert à définir l'adresse d'exécution de la routine. Le paramètre ",O" sert à indiquer que cette routine se trouve en RAM overlay ("O" pour Overlay). Il est donc possible d'utiliser des routines de Sedoric et, si on est un peu connaisseur, de remplacer une routine Sedoric par une routine perso ou d'en écrire une dans une zone libre de la RAM overlay et enfin d'exécuter cette routine, ce qu'un CALL ne peut pas faire. Aucun paramètre autre que ",O" n'est possible après DEF.

La seconde forme exécute la routine préalablement définie et, optionnellement, permet d'initialiser, les registres A, X, Y et P avant exécution. Chacune de ces valeurs, qui peuvent être n'importe quelle expression numérique de 0 à 255, doit bien entendu être compatible avec l'usage que l'on veut en faire. Au retour les variables réservées RA, RX, RY et RP contiennent les valeurs des registres A, X, Y et P. Cette commande est donc très puissante comme nous allons le voir avec quelques exemples.

Exécution d'une routine en Ram avec examen des registres en sortie

Nous utiliserons l'exemple devenu traditionnel qui consiste à implanter en #9801 un message suivit d'un petit programme permettant de l'afficher sur la ligne service. Soit le chargeur Basic suivant :

```
100 DATA #53,#61,#6C,#75,#74,#20
110 DATA #6C,#65,#73,#20
120 DATA #67,#61,#72,#73,#20,#21,#00
130 DATA #A0,#00,#B9,#01,#98,#F0,#06
140 DATA #99,#82,#BB,#C8,#D0,#F5,#60
150 FOR I=#9801 TO #981F
160 READ V:POKE I,V
170 NEXT
180 USER0,DEF#9812
190 USER0,A22,X22,Y22
200 PRINTRA,RX,RY,RP
```

Dont je vous rappelle l'explication :

- Les lignes 100 à 120 contiennent les codes Ascii de la chaîne de caractères du message terminé par zéro :

```
9801 53 61 6C 75 74 20 6C 65 73 20 67 61 72 73 20 21 00 "Salut les gars !"
```

- Les lignes 130 et 140 contiennent le code du petit programme en langage machine.

```
9812 A0 00     LDY #00      mettre la valeur zéro dans le registre Y
9814 B9 01 98  LDA 9801,Y   lire l'octet présent à l'adresse 9801+Y
9817 F0 06     BEQ 981F     si c'est 0, fini, on termine en 981F
9819 99 82 BB  STA BB82,Y   sinon on copie l'octet en BB82+Y
981C C8        INY          Y=Y+1 pour indexer l'octet suivant
981D D0 F5     BNE 9814     reboucle tant que Y ne repasse pas à zéro
981F 60        RTS          retourne au point d'appel
```

- Les lignes 150 à 170 chargent les DATA en RAM.
- La ligne 180 défini l'adresse d'exécution #9812 pour la routine USER n°0.
- La ligne 190 charge les registres A, X et Y du processeur avec la valeur 22 (c'est pour voir ce qui va changer après exécution, cela n'a aucune incidence sur le déroulement du programme qui n'utilise pas ces informations en entrée) et lance la routine USER n°0.
- La ligne 200 affiche l'état des registres après exécution.

La figure n°1, ci-contre, montre d'une part que le message a bien été affiché sur la ligne service et d'autre part que A=0 (le dernier octet lu était le zéro de fin de message), que X=22 (non affecté par la routine) et enfin que Y=16 (16 caractères ont été chargés et affichés sur la ligne service).

Je vous accorde que cet exemple ne présente pas grand intérêt en soit, mais son but est de montrer comment fonctionne la commande Sedoric USER et les possibilités qu'il y a à pouvoir faire passer des valeurs pour les registres du processeur en entrée et à récupérer la valeur de ces registres en sortie. Cela peut être précieux tant pour paramétrer la routine que pour la déboguer en cas de problème.

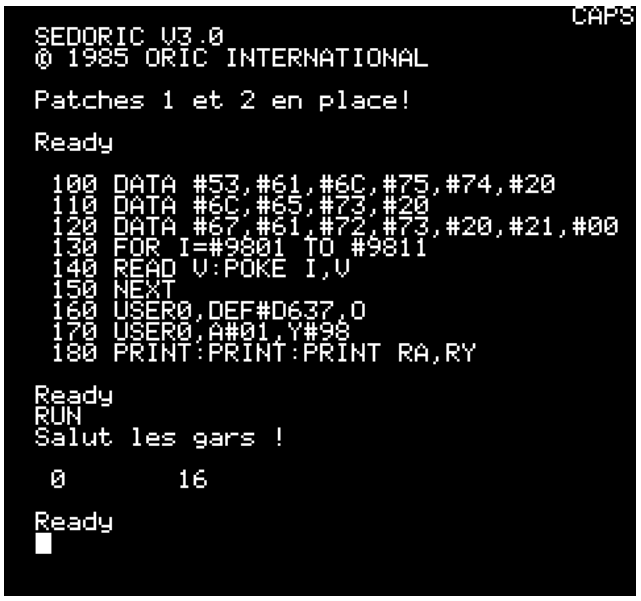
Exécution d'une routine en Ram Overlay avec passage de paramètres en entrée et examen des registres en sortie

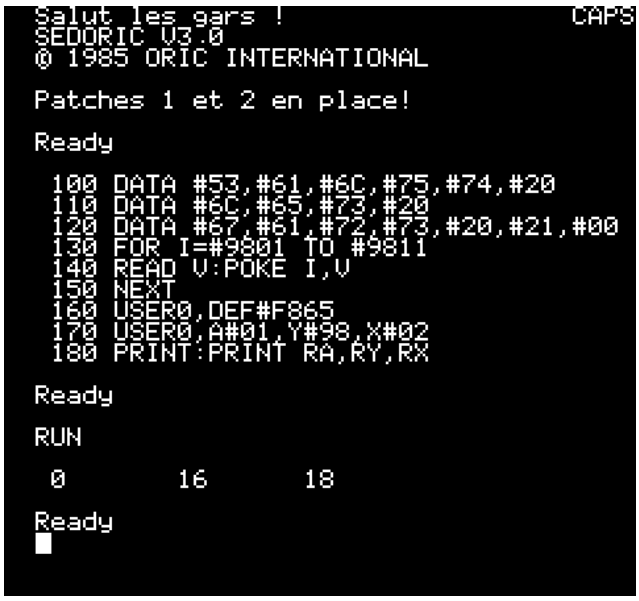
Nous allons reprendre une partie du chargeur Basic précédent, qui devient :

```
100 DATA #53,#61,#6C,#75,#74,#20
110 DATA #6C,#65,#73,#20
120 DATA #67,#61,#72,#73,#20,#21,#00
130 FOR I=#9801 TO #9811
140 READ V:POKE I,V
150 NEXT
160 USER0,DEF#D637,O
170 USER0,A#01,Y#98
180 PRINT:PRINT:PRINT RA,RY
```

Dont voici l'explication :

- Les lignes 100 à 120 contiennent les mêmes codes Ascii de la chaîne de caractères du message terminé par zéro.
- Les lignes 130 à 150 chargent les DATA en RAM.
- La ligne 160 défini l'adresse d'exécution #D637 en Ram Overlay pour la routine USER n°0. Il s'agit de la routine nommée XAFSTR, qui affiche une chaîne située à l'adresse définie par les registres A et Y (A pour l'octet de poids faible et Y pour l'octet de poids fort) et terminée par zéro.
- La ligne 170 charge les registres A et Y du processeur avec l'adresse de la chaîne à afficher et lance la routine USER n°0.
- La ligne 180 affiche l'état des 2 registres après exécution.





La figure n°2, ci-dessus à gauche, montre d'une part que cette fois-ci le message a été affiché au curseur et d'autre part, qu'en sortie, A=0 (le dernier octet lu était le zéro de fin de message) et enfin que Y=16 (16 caractères ont été chargés et affichés).

Exécution d'une routine en Rom avec passage de paramètres en entrée et examen des registres en sortie

Dans la Rom il existe de nombreuses routines réutilisables dans vos programmes. Si certaines ne demandent pas de paramètres et peuvent être appelée avec la commande CALL du Basic, d'autres, et ce sont les plus intéressantes, demandent à ce que certains registres du processeur soient initialisés.

La commande USER de Sedoric permet de faire ça de manière très simple. Pour rester dans les essais d'affichage, nous allons utiliser le sous-programme situé en #F865 (#F82F pour l'Oric-1), qui permet d'afficher une chaîne sur la ligne service en mode Text. En entrée, les registres A et Y pointent sur la chaîne à afficher, qui doit être terminée par #00 et le registre X pointe sur l'ordonnée où doit commencer l'affichage.

Nous allons adapter le chargeur Basic précédent, qui devient :

```
100 DATA #53,#61,#6C,#75,#74,#20
110 DATA #6C,#65,#73,#20
120 DATA #67,#61,#72,#73,#20,#21,#00
130 FOR I=#9801 TO #9811
140 READ V:POKE I,V
150 NEXT
160 USER0,DEF#F865
170 USER0,A#01,Y#98,X#02
180 PRINT:PRINT RA,RY,RX
```

Dont voici l'explication :

- Les lignes 100 à 120 contiennent toujours les mêmes codes Ascii de la chaîne de caractères du message "Salut les gars !" terminée par zéro.
- Les lignes 130 à 150 chargent les DATA en RAM.
- La ligne 160 défini l'adresse d'exécution #F865 en Rom pour la routine USER n°0.
- La ligne 170 charge les registres A et Y du processeur avec l'adresse de la chaîne à afficher (A pour l'octet de poids faible et Y pour l'octet de poids fort), charge le registre X avec l'ordonnée d'affichage de ce message et lance la routine USER n°0.
- La ligne 180 affiche l'état des 3 registres après exécution. La figure n°3 montre d'une part que le message a bien été affiché sur la ligne service et d'autre part que A=0 (le dernier octet lu était le zéro de fin de message), que X=18 (ordonnée du dernier caractère affiché) et enfin que Y=16 (16 caractères ont été chargés et affichés sur la ligne service).

Conclusion

Comme je l'ai déjà indiqué le n° de routine peut être utilisé dans le cas d'un choix entre 4 possibilités (un peu comme la commande ON… GOSUB…).

Mais l'imagination est reine au pays de l'Oric et il ne fait aucun doute que vous saurez trouver d'autres utilisations à cette commande. A vous de jouer !