Petit divertissement soft

Aujourd'hui, tripotons le TXTPTR !

par André C.

De quoi s'agit-il ?

On appelle TXTPTR (TeXT PoinTeR) l'adresse où l'interpréteur doit lire la prochaine commande dans le programme Basic (ou dans le tampon clavier en mode immédiat). TXTPTR (situé en page zéro, en #E9-#EA) est une sorte de doigt, qui suit pas à pas la lecture du programme.
Chacune des commandes est lue (ainsi que ses éventuels arguments), jusqu'au ":" suivant ou jusqu'au #00 qui marque la fin de la ligne Basic. La commande est alors interprétée et exécutée. Et on passe à la suivante, etc.
Est-il possible de leurrer le système en altérant le TXTPTR ? Certes, c'est un peu risqué, car on aura inévitablement une "?SYNTAX ERROR", voire un plantage, si le TXTPTR est mal positionné. Il faut donc savoir exactement ce que l'on fait.
Avec le débogueur d'Euphoric (F11 pour y entrer, puis F11 pour en sortir), il est aisé de savoir où se trouvent ces ":" et ces #00.
Il suffit d'examiner le programme Basic, situé à partir de #0501. Tapez D0500 pour afficher la mémoire à partir de #0500 (D pour Dump), (chiffres de la rangée du haut svp).

Chaque ligne Basic se caractérise par une paire d'octets contenant l'adresse de la ligne suivante, puis une paire d'octets contenant le n° de la ligne, puis une ou plusieurs commandes séparées par ":" et enfin par #00 qui marque la fin de la ligne.
Avec un peu d'attention, il est facile de relever l'adresse de ces ":" et #00. Il est alors possible de "bousculer" un peu l'interpréteur en tripotant le TXTPTR… On peut ainsi réaliser des GOTO ultrarapides, voire même des choses plus compliquées, imprévues par l'orthodoxie Basic !

Travaux pratiques

Soit le mini-programme Basic :

```
1 CLS:PRINT@10,10;"BONJOUR !"
2 GET R$
3 PRINT@10,10;"AU REVOIR"
4 GET R$:END
5 PRINT@10,10;"COUCOU ! "
6 GET R$
7 DOKE#E9,#0506
```

Figure 1

```
1                    MEMORY DUMP
0500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Remarquez, au milieu de ce listing, le END qui a pour but d'arrêter l'exécution et de tenir la fin en réserve.
Les n° de lignes sont tous inférieurs à 10 et seront donc identiques en décimal et en hexadécimal. Le dump n'en sera que plus facile à lire !
Avant de taper ce programme, regardons sous Euphoric la zone de Ram qui sera utilisée pour le stocker : F11 puis D0500 (chiffres de la rangée du haut svp). Sur la figure 1, on voit que cette zone

Figure 2

```
1                    MEMORY DUMP
0500: 00 1B 05 01 00 94 3A BA C6 31 30 2C 31 30 3B 22   .....▌:▒F10,10;"
0510: 42 4F 4E 4A 4F 55 52 20 21 22 00 24 05 02 00 BE   BONJOUR !".$...▐
0520: 20 52 24 00 3C 05 03 00 BA C6 31 30 2C 31 30 3B    R$.<...▒F10,10;
0530: 22 41 55 20 52 45 56 4F 49 52 22 00 47 05 04 00   "AU REVOIR".G...
0540: BE 20 52 24 3A 80 00 5F 05 05 00 BA C6 31 30 2C   ▐ R$:▌._...▒F10,
0550: 31 30 3B 22 43 4F 55 43 4F 55 20 21 20 22 00 68   10;"COUCOU ! ".h
0560: 05 06 00 BE 20 52 24 00 77 05 07 00 8A 23 45 39   ...▐ R$.w...▌#E9
0570: 2C 23 30 35 30 36 00 00 00 52 00 00 00 00 00 00   ,#0506...R......
0580: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
```

est remplie de #00, y compris l'adresse #0500 (qui doit toujours rester à zéro). Pressez F11 pour sortir du débogueur. Après avoir tapé le mini-programme, F11 pour retourner voir ce que ça donne (touches page_up et page_down pour naviguer dans le dump). Sur la figure 2, on voit que :
La 1ère ligne Basic commence en #0501 par une paire d'octet qui indiquent l'adresse de la ligne suivante, ici #051B. Les 2 octets suivants indiquent le n° de ligne, ici 0001. L'octet suivant est #94, qui est le code (token) de CLS. Puis voici un #3A (situé en #0506), qui est le code Ascii de ":" séparant la 1ère instruction (CLS) de la seconde (PRINT@10,10;"BONJOUR !").
On peut facilement suivre pas à pas le codage de cette instruction, en s'aidant du dump Ascii se trouvant à droite, jusqu'au guillemet final. Ce guillemet final est suivit d'un #00 (situé en #051A), qui marque la fin de la ligne Basic. Avec un peu de patience, on peut trouver toutes les adresses des ":" séparant les commandes et des #00 de fin de ligne Basic, soit :

```
1 CLS:#0506PRINT@10,10;"BONJOUR !"#051A
2 GET R$#0523
3 PRINT@10,10;"AU REVOIR"#053B
4 GET R$:#0544END#0546
5 PRINT@10,10;"COUCOU ! "#055E
6 GET R$#0567
7 DOKE#E9,#0506#0576
```

On observe en #0576 les 3 zéros qui marquent la fin du programme Basic. Ces 3 zéros sont en général suivis d'un octet dénué de signification, ici #52. Puis la Ram est vierge (ou garde les traces de manipulations précédentes).
Si on lance ce programme, il efface l'écran, puis affiche BONJOUR !, attend que l'on presse une touche, puis affiche AU REVOIR, attend une touche, puis rend la main. Cela ne fait pas beaucoup !
Corsons l'affaire en intervenant sur la valeur de TXTPTR.
Le DOKE de la ligne 7 force le TXTPTR à l'adresse #0506 (le ":" après le CLS). Mais pour qu'il soit exécuté, il faut supprimer la commande END, en fait, la remplacer par REM, qui offre

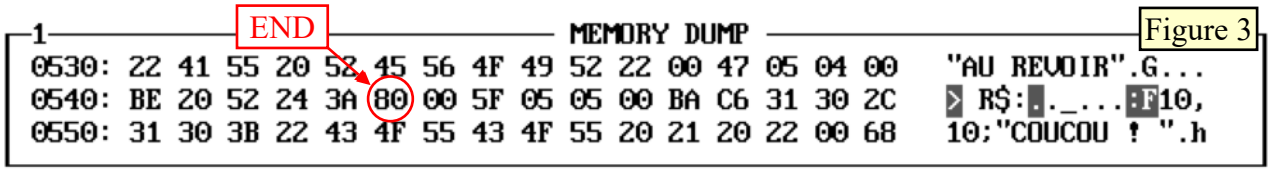

Figure 3

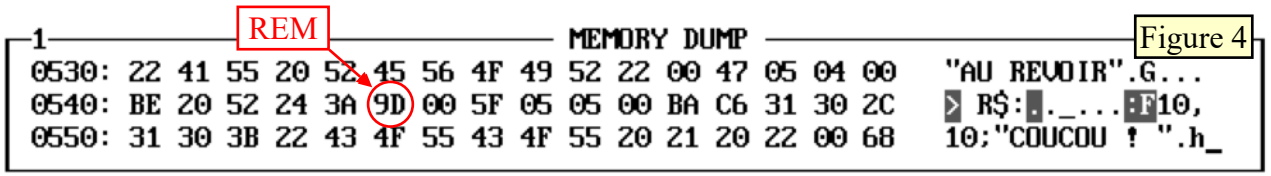

Figure 4

l'avantage d'avoir la même longueur, aussi bien dans le texte du programme (mais cela importe peu, en fait) que surtout dans le codage du programme en Ram : Un seul octet est modifié le #80 de END est remplacé par le #9D de REM (voir les figures 3 et 4 ci-dessus).
Si on lance ce programme, il efface l'écran, puis affiche BONJOUR !, attend que l'on presse une touche, puis affiche AU REVOIR, attend une touche, affiche COUCOU !, attend une touche et reboucle au début (en sautant le CLS), affiche BONJOUR !, attend une touche, affiche AU REVOIR, attend une touche etc., sans fin.
Vous pouvez expérimenter en remplaçant le #0506 de la ligne 7 par n'importe laquelle des adresses indiquées en rouge et pointant sur un ":" ou un #00 de fin de ligne Basic, ça marchera à coup sûr. Vous pouvez aussi voir ce que ça donne quand on commet une erreur dans l'adresse imposée à TXTPTR et essayer de comprendre le plantage qui suivra inévitablement !
Vous devez penser "Tout ça pour ça ! Mais ça ne sert à rien !"
Patience, un prochain article illustrera le pouvoir fabuleux de tripotage de TXTPTR !

à suivre…