La touche FUNCT

2ème partie : Expérimentons un peu

par André C.

Le manuel de l'Atmos est très discret sur la touche FUNCT. En fait il n'en parle pas! Aucune commande n'est prévue, ni pour détecter un appui sur FUNCT, ni pour lui associer une action.

Touche	Valeur
Aucune	#38
CTRL	#A2
SHIFTg	#A4
SHIFTd	#A7
FUNCT	#A5

Toute détection d'un appui sur la touche FUNCT repose sur le contenu de la variable #209 en page 2 de la Ram. Pour utiliser la touche FUNC, il faut donc connaître l'existence de cette variable #209. La valeur présente à cette adresse est modifiée si l'on appuie sur l'une des 4 touches de fonctions selon le tableau ci-dessus.

Pour vous en convaincre, essayez le programme suivant:

```
10 PRINT HEX$(PEEK(#209))
20 GOTO 10
```

Testez les différentes touches de votre clavier. Seules les 4 touches de fonction ci-dessus sont prises en compte (voir la note* ci-après). Vous observerez que la boucle est très rapide et que le contenu de la variable #209 est fugace. Dans un programme plus gros, il y a toutes les malchances qu'un appui très bref ne soit pas détecté. C'est le point faible de ce PEEK(#209). Toutefois un appui plus long règle l'affaire.

Note*: Les claviers PC étant plus complexes (par exemple, il y a deux touches CTRL), les émulateurs répondent aussi de façon plus complexe. Le tableau ci-dessus montre ce qu'on a sous Euphoric. Aucune autre touche ne donne de réponses.

Touche	Valeur	Remarque
Aucune	#38	
CTRLg	#A2	
CTRLd	#A0	
SHIFTg	#A4	
SHIFTd	#A7	
ALT	#A5	Qui émule la touche FUNCT
+	#A5	Idem, ça c'est curieux!
AltGr	#A6	Sans équivalent sur Atmos

a) Exemple de détection en Basic.

Dans un programme Basic ça donne quelque chose comme ça:

```
10 IF PEEK(#209)=#A5 THEN PING
20 GOTO 10
```

Le PING peut être remplacé par un GOTO, GOSUB,CALL, etc.

On ne peut donc rien imaginer de plus simple!

b) Exemple de détection en langage machine.

Dans un programme en langage machine, le listing est plus long (ci-dessous), mais l'exécution est encore plus rapide.

```
9801 48        PHA       ; Sauve l'accumulateur
9802 AD 09 02  LDA 0209  ; Lit la variable #209
9805 C9 A5     CMP #A5   ; Est-ce FUNCT?
9807 D0 03     BNE 9802  ; Non, on re-teste
9809 20 9F FA  JSR FA9F  ; Oui, on exécute un PING
980C 68        PLA       ; Restaure l'accumulateur
980D 60        RTS       : Retour à l'appelant
```

C'est le principe général. Notez que le BNE 9802 ne peut pas être remplacé par un BNE 980C car la routine est tellement rapide qu'on en sortirait avant d'avoir eu le temps d'appuyer sur quoi que ce soit!

Vous allez me dire "Oui, mais la routine est bloquée, elle attend indéfiniment un appui sur la touche FUNCT!". Il y a des solutions à ça selon ce qu'on veut faire. Par exemple limiter la boucle de test à un nombre de tours donné puis sortir. Autre possibilité: tester un appui sur une autre touche de fonction, par exemple CTRL:

```
9801 48        PHA       ; Sauve l'accumulateur
9802 AD 09 02  LDA 0209  ; Lit la variable #209
9805 C9 A5     CMP #A5   ; Est-ce FUNCT?
9807 D0 05     BNE 980E  ; Non, on continue au test suivant
9809 20 9F FA  JSR FA9F  ; Oui, on exécute un PING
980C 68        PLA       ; Restaure l'accumulateur
980D 60        RTS       : Retour à l'appelant
980E C9 A2     CMP #A2   ; Est-ce CTRL?
9810 D0 F0     BNE 9802  ; Non, on re-teste
9812 F0 F8     BEQ 980C  ; Oui on sort
```

c) Exemple d'application pratique

Il y en a évidemment des milliers. Prenons par exemple le cas où il faut mettre au point un jeu en mode texte avec beaucoup de caractères redéfinis. Lors du débogage, quand vous testez le programme, la redéfinition des caractères est mise en place. Mais il y a un petit problème pour corriger le listing car il est illisible! Ce qui serait pratique, c'est que le retour au Ready régénère automatiquement les caractères normaux, lorsqu'on presse la touche FUNCT mais que les caractères restent redéfinis lorsqu'on presse la touche CTRL. Après correction du listing, quand on relance le programme, les caractères redéfinis sont remis en place et ainsi de suite.

Cette astuce n'est pas bien difficile à réaliser: Un chargeur Basic détourne l'affichage du Ready vers une routine qui teste un appui sur FUNCT ou sur CTRL. Soit le petit Basic suivant:

```
10 FOR I=1 to 19: READ V:POKE#9800+I,V:NEXT
20 DOKE#1B,#9801
30 DATA #48, #AD, #09, #02, #C9, #A5, #D0, #05, #20, #D0, #F8
40 DATA #68, #60, #C9, #A2, #D0, #F0, #F0, #F8
```

Il suffit de lancer ce petit Basic qui met en place notre routine et détourne le vecteur Ready vers cette routine. Après exécution de ce petit Basic, il tente de retourner au Ready, mais se retrouve dans la routine #9801. Il faut donc presser CTRL pour en sortir. Mettez alors en place le programme Basic en cours de mise au point et lancez-le. Pour corriger le listing, sortez du programme (CTRL+C) et pressez FUNCT. Cette fois le listing est lisible et vous pouvez l'éditer.

Le listing de la petite routine correspondant au chargeur Basic est identique à celui donné au paragraphe b, sauf que le PING (JSR FA9F) est remplacé par un JSR F8D0, qui est la routine chargée de régénérer les caractères.

J'ai intitulé cette partie "Exemple d'application pratique". Oui, bon, ce n'est pas aussi simple car la routine #F8D0 en Rom chamboule les registres du processeur 6502 et le retour n'est pas cool! Donc en plus, il va falloir sauvegarder les registres au début et les restaurer à la fin de notre petite routine. En outre, la routine appelée en Rom régénère les caractères du jeu 0 (le jeu normal), puis enchaîne sur la routine #F816 qui régénère les caractères du jeu 1 (le jeu graphique) et termine avec la routine #F75A qui affiche CAPS. Or il est parfaitement inutile de régénérer le jeu 1 et pas plus d'afficher CAPS. En outre, le "Ready" ayant été détourné, il

serait bien de l'afficher. Il va donc falloir affiner. Voici finalement le listing qui va bien. Tous les registres sont sauvegardés, ce qui assure l'universalité de la routine:

```
; Cette routine FUNCT2 régénère les caractères du jeu 0
; (jeu normal). Elle est activée lors du retour au Ready
; par un détournement du vecteur en #1B/1C (DOKE#1B,#9801).
; Un appui sur la touche FUNCT valide la régénération tandis
; qu'un appui sur la touche CTRL conserve les caractères
; redéfinis, affiche Ready et rend la main. Pour régénérer le
; jeu 1 (graphique) il faudrait ajouter un JSR F816.
      org $9801  ; Adresse d'implantation du sous-programme
; Sauvegarde des registres
      PHP        ; Sauve l'indicateur d'état P
      PHA        ; Sauve l'accumulateur A sur la pile
      TXA        ; Transfère le registre X dans l'accumulateur A
      PHA        ; et le sauve sur la pile
      TYA        ; Transfère le registre Y dans l'accumulateur A
      PHA        ; et le sauve sur la pile
      TSX        ; Transfère le pointeur de pile S dans registre X
      TXA        ; puis dans l'accumulateur A
      PHA        ; et le sauve sur la pile
test1
      LDA $0209  ; Lit la variable #209
      CMP #$A5   ; La touche FUNCT a t-elle été pressée?
      BNE test2  ; Non, on continue au test suivant
      LDX #$05   ; Oui, on régénère les caractères
      JSR $F8D0  ; du jeu normal
fin
; Restauration des registres
      PLA        ; Récupère le pointeur de pile S,
      TAX        ; le transfère dans le registre X
      TXS        ; puis dans le pointeur de pile
      PLA        ; Récupère le registre Y de la pile
      TAY        ; et le restaure
      PLA        ; Récupère le registre X de la pile
      TAX        ; et le restaure
      PLA        ; Restaure l'accumulateur A
      PLP        ; Restaure l'indicateur d'état
      JSR $CCB0  ; Affiche Ready
      RTS        ; et rend la main
test2
      CMP #$A2   ; Est-ce CTRL?
      BNE test1  ; Non, on re-teste
      BEQ fin    ; Oui on sort
done
```

Conclusion

Une fois compris le principe, vous pouvez utiliser cette astuce pour détecter un appui sur la touche FUNCT (et/ou sur l'une des 3 autres touches de contrôle) et lancer l'application de votre choix.

Comme d'habitude, le fichier source .asm ci-dessus et le fichier .tap correspondant seront dans l'archive FUNCT2.zip accompagnant cet article.

A vos claviers!