INITIATION AU FORTH - VOLET #3
LES STRUCTURES DE CONTRÔLES zFORTH

Olivier SINGLA

INTRODUCTION

Continuons notre étude du langage FORTH ! Avec ce 3ème volet, nous allons nous attaquer à un morceau de choix : les différentes structures de contrôles offertes dans la version de FORTH diffusée par le CNDP, pour les micro-ordinateurs fonctionnant sous le système d'exploitation EDEN ("CP/M like... "). Nous nous bornerons aux structures de contrôles les plus utilisées (Il y en a plus d'une dizaine dans zFORTH...). Nous examinerons, au travers d'une d'elles, les mécanismes internes mis en jeu. Mais de façon générale, on s'intéressera plutôt au protocole d'utilisation.

Il faut bien avoir à l'esprit qu'une structure de contrôle FORTH "typique" est composé d'un certain nombre de MOTS qui agissent au moment de la compilation: ce sont des mots dits immédiats. Le système-FORTH (par le biais du mot INTERPRET) les exécutera, même en état-compilation, ou lieu de compiler ainsi leur adresse-code (CFA) au sein de la définition dans laquelle ils sont invoqués.

Attention ! Notez que le terme "compiler" a en FORTH une signification différente de celle que vous lui assignez habituellement (en particulier ceux qui ont pratiqué le PASCAL ou le C...). Il ne s'agit pas ici d'implanter en mémoire un code-objet, une séquence de code-machine du microprocesseur donc. "Compiler" en FORTH désignera l'action de déposer une valeur dans une cellule du dictionnaire. Ainsi, en "mode compilation", chaque mot référencé (à quelques exceptions prés...) engendrera l'action de compiler un pointeur sur le code correspondant à ce mot.

1. LA STRUCTURE DE «IF-[ELSE]-END»

On peut exprimer son fonctionnement par :

- «si "condition remplie"
- alors faire un groupe d'instructions
- sinon en faire éventuellement un autre»

format : <flag déposé sur la piles IF

[instructions 1]

[ELSE

[instructions 2}]

ENDIF

Définition des mots IF ELSE et ENDIF

:IF	\ (f--) à l'exécution et (-- adr 2) lors de la compilation
COMPILE OBRANCH	\ compile un pointeur sur OBRANCH dans la prochaine cellule libre
HERE	\ empile l'adresse du sommet actuel du dictionnaire (-- adr)
0 ,	\ compile un offset bidon dans la prochaine cellule
2	\ empile 2, marqueur de cette structure de contrôle (-- adr 2)
; IMMEDIATE	\ fin de la définition, celle-ci étant Immédiate
ELSE	\ (adr1 2 -- adr2 2) lors de la compilation exclusivement
2 ?PAIRS	\ teste l'égalité à 2 (y a bien eu IF avant...?), si * de 2 -> mess. d'erreur !
COMPILE BRANCH	\ compile un pointeur sur BRANCH dans la prochaine cellule libre
HERE	\ empile l'adresse du sommet actuel du dictionnaire (adr 1- adr 1 adr2)
0 ,	\ compile un offset bidon dans la prochaine cellule
SWAP	\ (adr 1 - adr 1 adr2) devient (adr 1 -- adr2 adr 1)
2	\ empile 2 (adr2 adr 1 - adr2 adr 1 2)
[COMPILE] ENDIF	\ utilise ENDIF pour résoudre le branchement conditionnel en adr 1
2	\ empile 2 (adr2 - adr2 2)
IMMEDIATE	\ fin de la définition, celle-ci étant Immédiate
ENDIF	\ (adr 2 --) lors de la compilation exclusivement
?COMP	\ génère une erreur si le système n'est pas en mode compilation

2 ?PAIRS	\ teste l'égalité à 2 (y a bien eu IF avant...?), si * de 2 -> mess. d'erreur !
HERE	\ empile l'adresse du sommet actuel du dictionnaire (adr - adr adr2)
OVER	\ (adr - adr adr2 adr)
-	\ calcule l'offset du branchement relatif (adr - adr adr2-adr)
SWAP	1 \ Compile l'offset à l'adresse adr
; IMMEDIATE	\ fin de la définition, celle-ci étant Immédiate

Ces 3 définitions, étant Immédiates, sont donc exécutées lors de la compilation. Comme elles compilent un certain nombre de choses dans le dictionnaire, on peut les assimiler, grossièrement, à des macros-instructions.

Voyons maintenant les définitions référencées dans IF, ELSE et ENDIF, et que nous n'avons pas encore vues.

IMMEDIATE : Positionne le bit de précédence de la dernière définition dans le dictionnaire. Celle-ci devient alors immédiate, exécutés même lors du mode compilation. On peut néanmoins forcer leur compilation par le mot suivant...

[COMPILE] : Permet de forcer la compilation du mot Immédiat dont le nom suit.

COMPILE : A ne pas confondre avec [COMPILE). Avec COMPILE, l'action de compilation d'une définition ne joue pas lors de la compilation, mais est différée à l'exécution de la définition contenant "COMPILE xxx".

HERE : Laisse sur la pile la valeur actuelle de l'adresse du sommet du dictionnaire. Protocole: (n ---). Compile la valeur 16-bits n au sommet du dictionnaire (le prochaine cellule de libre donc), et met à jour celui-ci en conséquence (incrément de 2).

OBRANCH : C'est un mot utilisable seulement en "run time", qui est donc compilé. Il permet d'effectuer un branchement inconditionnel sur la cellule dont l'adresse est calculée comme suit : somme du pointeur d'interprétation (IP=adresse actuelle) + l'offset qui suit BRANCH dans le dictionnaire. BRANCH est compilé, entre autres, par ELSE , REPEAT et MAIN.

OBRANCH : Protocole: (f ---). Idem que BRANCH, mais le branchement n'est effectué que si le flag f absorbé sur la pile est faux (valeur nulle donc). OBRANCH est utilisé, entre outres, par IF , UNTIL et WHILE.

Exemple de définition utilisant "IF...ELSE...ENDIF"

```
: touche
KEY DUP
"A "Z2 [WITHIN] OVER "a "z [WITHIN] OR
CA SWAP EMIT
IF
." est une lettre"
ELSE
."n'est pas une lettre"
ENDIF ;'
```

2. LA STRUCTURE DE CONTRÔLE «BEGIN...UNTIL»

On peut exprimer son fonctionnement par :

«répéter {instructions} jusqu'à ce qu'une condition soit vraie»

format : BEGIN {I*nstructions]* [flag déposé sur la pile) UNTIL

Les instructions situées dans la corps de la boucle doivent laisser un flag sur la pile. UNTIL absorbe alors ce flag, et reboucle si la condition n'est pas vraie (c'est-à-dire si le flag est faux).

Exemple d'utilisation. Soit l'algorithme suivant :

«Jusqu'à ce que l'on appuie sur {RETURN} répéter

‖lire une touche, et afficher le caractère correspondant.»

La programmation pourrait être la suivante :

```
ECH01
BEGIN                \ début de la boucle indéfinie
   KEY               \ lire une touche et empiler son code ASCII
   DUP EMIT          \ le dupliquer, puis afficher le caractère correspondant
   ^M                \ empiler le flag vrai (f=1) si caractère=RETURN
UNTIL ;              \ fin de boucle, puis fin de définition
```

Remarque : Dans la définition donnée ici pair ECHO1, le caractère (RETURN) est affiché (passage en début de ligne suivante) avant que la fin de boucle ne soit détectée. A titre d'exercice, re-écrivez cette définition de telle manière que le (RETURN) ne soit plus généré.

Note : Les instructions situées dans le corps de la boucle «BEGIN... UNTIL» sont exécutées au moins une fois.

3. LA STRUCTURE DE CONTRÔLE «BEGIN...WHILE ...REPEAT»

On peut exprimer son fonctionnement par :

«répéter {instructions} tant qu'une condition est vraie»

format : BEGIN

{instructions I}

[flag déposé sur la pile]

WHILE

{instructions 2}

REPEAT

Le groupe d'instructions 1 laissent un flag sur la pile: si celui-ci est faux (f=0), une sortie de la boucle est alors effectuée (branchement après REPEAT), sinon le groupe d'instructions 2 est exécuté, puis un branchement sur le début de boucle (juste après BEGIN) est alors effectué.

Exemple d'utilisation. Soit l'algorithme suivant:

«Tant que non-appui sur la toucha {RETURN} répéter

‖ lire une touche, et afficher le caractère correspondant.»

La programmation pourrait être la suivante :

```
ECH02
BEGIN          \ début de la boucle indéfinie
KEY DUP        \ lire une touche et dupliquer son code ASCII
^M < >         \ empiler un flag indiquant si appui sur (RETURN)
1WHILE         \ sortie de boucle si ce flag est faux (f=0 )
EMIT           \ afficher le caractère correspondant
REPERT         \ fin de la boucle ==> on revient au début de la boucle
DROP ;         \ «dropper» la valeur 13, code ASCII de (RETURN)
```

4. LA STRUCTURE DE CONTRÔLE «CASE ... ENDCASE»

Cette structure de contrôle permet une structuration efficace, en prévoyant des instructions (des mots) à faire exécuter suivent un certain nombre de ces déterminés (en fait, suivant l'égalité entre une valeur de référence et un jeu d'autres valeurs).

format :

[valeur de *référence* sur la pile] CASE
(valeur 1 sur la pile]OF [mots exécutés si égalité] ENDOF
(valeur 2 sur la pile]OF [mots exécutés si égalité]ENDOF

(valeur n sur la pile]OF [mots exécutés si égalité]ENDOF
ENDCASE

Notes :

Une valeur doit être déposée sur la pile avant l'exécution du mot «CASE» (c'est la valeur de référence), et également avant chaque «OF» (valeur de comparaison).

ENDOF effectue un branchement après «ENDCASE» : donc, en cas d'exécution d'une clause «OF... ENDOF» , celle-ci sera la seule effectuée.

La clause «OTHERWISE» est implicite : les instructions situées dans le corps de la structure «CASE... ENDCASE », mais non comprises dans une clause «OF... ENDOF», sont exécutées inconditionnellement.

Exemple d'utilisation. Soit l'algorithme suivant:

«Lire une touche...

- afficher "un" si on a frappé sur "1"
- afficher "deux" si en a frappé sur "2"
- afficher "ni un, ni deux !" si an a frappé ni sur "l", ni sur "2"»

La programmation pourrait être la suivante :

```
: ESSAI
KEY CASE
"1 OF ."un"          ENDOF
"2 OF . "deux"       ENDOF ,
"ni un, ni deux !"
ENDCASE ;
```

5. LA STRUCTURE DE CONTRÔLE «DO...((NOT)WHEN)(+)LOOP»

Cette structure de contrôle permet de répéter un groupe d'instructions en faisant varier la valeur d'un index entre 2 limites données, avec un Incrément implicite (1) ou spécifié. On peut, optionnellement, utiliser un critère de fin de boucle prématurée. Les valeurs manipulées (index, valeurs de départ et finale) sont signées. Plusieurs possibilités sont offertes :

5.1 «DO...LOOP»

format : [valeur limite+1] [valeur de départ] DO

{instructions}

LOOP

Dans cette option, le pas est, de manière implicite, fixée à la valeur 1.

Exemple ; soit la définition suivante...

: COMPT

10 0 DO | LOOP ;

affichera : 0 1 2 3 4 5 6 7 8 9

Le mot «|» empile la valeur de l'index de la dernière boucle rencontrée (le plus interne).

ATTENTION : La boucle est effectuée au moins une fois... Ainsi :

COMPT2 1 3 DO |, LOOP ;

donnera : 3

(utiliser alors la clause «WHEN» ou «NOTWHEN»...)

NOTE: Les mots «J» et «K» empilent la valeur des index des boucles extérieures.

5.2 «DO...+LOOP»

On utilisera «+LOOP» dans le cas d'un pas différent de 1. Le pas, spécifié, doit être alors empilé avant «+LOOP».

Exemple 1 :

```
COMPT3
30 20 DO | . 2 +LOOP ;
```

donnera: 20 22 24 26 28

(remarquer que la valeur 29 au lieu de 30 produirait le même résultat).

Exemple 2.

```
COMPT4
20 30 D0 | . -2 +LOOP ;
```

donnera: 30 28 26 24 22

La valeur limite de l'index à empiler est donc égale à la valeur réelle de la limite...

+1 (ou le pas) si le pas est positif (cas de «LOOP»)

-1 (ou le pas) si le pas est négatif.

5-3 les clauses «WHEN> et «NOTWHEN»

Ces clauses permettent d'intégrer à une boucle répétitive un critère de continuation si un flag est vrai («WHEN») ou faux («NOTWHEN»).

Exemple 1 :

Afficher la suite des nombres de 1 à 1000, mais s'arrêter si on frappe sur une touche...

```
AFF1
1001 1 D0              \ aller de 1 à 1000
PAUSE? 0-              \ flag vrai si appui sur <ESC>, sinon flag faux
WHEN                   \ continuer uniquement si le flag est vrai I .
LOOP ;
```

exemple 2 : idem...

```
AFF2
1001 1 DO PAUSE? NOTWHEN I . LOOP ;
```

6. LA STRUCTURE DE CONTRÔLE «MAKE...STILL»

Cette structure de contrôle fonctionne de la manière suivante :

[&] MAKE

[clause]

STILL

La {clause} spécifiée sera exécutée {u} fois, sauf si {u} est égale à la valeur 0. Cette structure de contrôle peut être imbriquée. Elle ne peut s'utiliser qu'en mode compilation.

Exemple :

```
RECT ( n 1 n2 --)              \ rectangle plein de n 1 lignes et n2 colonnes
SWAP MAKE
    CA DUP MAKE "* EMIT STILL
STILL DROP CA ;
```

L'index de boucle étant sauvé sur la pile de retour, veiller donc à ne pas altérer sa valeur avant le mot «STILL».

Dans le prochain article, nous aborderons un aspect très important du langage FORTH: nous parlerons de la "machine virtuelle FORTH". Le fonctionnement Interne sera décrit à la fois dans le langage d'un processeur fictif, et en assembleur 6809. Quant au FORTH MOS dans le cadre d'un fonctionnement en NanoRéseau, II est toujours en cours de développement. A ce sujet, si vous voulez tester la préversion, envoyez-moi une disquette vierge, une enveloppe auto-adressée suffisamment affranchie.

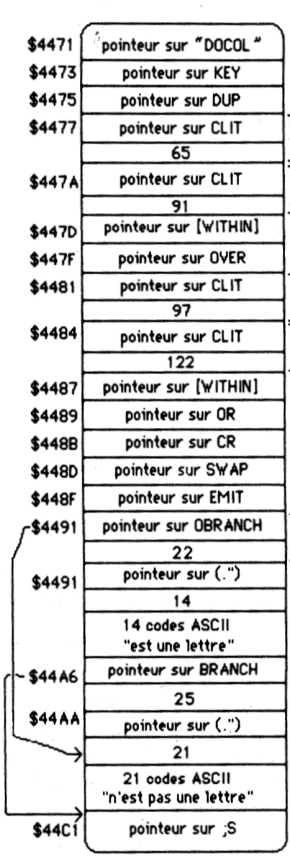
Olivier SINGLA - 18 av. Mayenne - 82000 MONTAUBAN - tel: (63) 66-10-90 (Si vous tombez sur un sifflement strident, ne vous étonnez pas outre mesure, mais connectez votre Minitel).

Olivier SINGLA

Code généré par le "compilateur" FORTH pour la définition suivante...

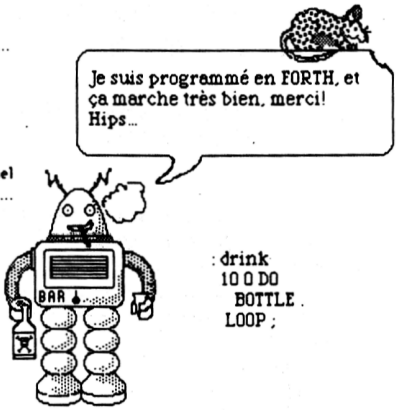
```
: touche
 KEY DUP
 "A "Z [WITHIN] OVER "a "z [WITHIN] OR
 CR SWAP EMIT
 IF
  ." est une lettre"
 ELSE
  ." n'est pas une lettre"
 ENDIF ;
```

"pointeur" doit être pris ici au sens de CFA de la définition (CFA = Code Field Adresse, soit adresse du champ code)



DOCOL, c'est l'interpréteur des mots écrits en FORTH (se charge d'empiler l'adresse de retour sur la pile de ... retour, puis donne la main à la definition invoquée).

;S fait exactement la chose inverse: depile depuis la pile de retour, et rend la main à la définition appelante dont le CFA vient d'être dépilé.




```
: drink
 100 DO
  BOTTLE .
 LOOP ;
```