Ralp, un utilitaire pour le ZX 81

RALP
de P. DEMEL
Permet l'adressage de sous-programmes « USER » inclus dans des lignes REM sans en connaître l'implantation mémoire.
Langage : Basic + code Z 80
Ordinateur : ZX 81 (16 K-octets)

Le stockage des routines en langage machine dans des instructions REM du Basic est pratique courante sur un ZX 81. Cependant, pour y accéder, il faut en connaître l'adresse mémoire, ce qui limite l'emploi de cette méthode. Ralp supprime cette contrainte en autorisant l'accès à une routine par le numéro de la ligne REM la contenant, et ce, pour l'implanter, la modifier ou l'exécuter.

Ecrire des routines en langage machine s'avère impératif pour accélérer l'exécution d'un programme Basic. L'instruction REM est souvent employée pour leur stockage. Sur un ZX 81, c'est d'ailleurs le procédé le plus pratique. Cependant, pour y faire référence, il faut connaître et exprimer dans le programme l'adresse de cette routine. Cette contrainte en limite pratiquement l'implantation à la première ou à la dernière ligne d'un programme, à moins de se livrer à de fastidieux calculs. En effet, l'accès, pour exécution, est facile dans le premier cas (USER 16514), et encore acceptable dans le second (USER [PEEK 16396 + 256 * PEEK 16397 – Long. routine])

Le sous-programme « Ralp » permet d'appeler une routine en langage **machine** par le **numéro de ligne** du programme Basic qui la contient, exactement comme pour un sous-programme Basic.

Les avantages de ce procédé sont nombreux. Tout d'abord, il est inutile de connaître explicitement l'adresse de la routine à exécuter, celle-ci étant calculée par le sous-programme. Ensuite, il est possible d'introduire les routines où l'on veut au cours de l'implantation (ou des modifications) du programme principal.

Notons aussi qu'il n'est pas nécessaire de retoucher les adresses d'accès aux routines (dans la mesure où celles-ci ne changent pas de numéro de ligne) en cas de modification du programme principal. Attention, cependant : les routines contenues dans les instructions REM peuvent adresser directement un des octets les constituant ou être adressées par d'autres routines, ou même des PEEK depuis le programme Basic. Dans ces cas, toute modification du programme déplacera les routines en mémoire et, par là-même, l'adresse des routines.

Ce sous-programme utilise essentiellement une routine contenue dans la ROM du ZX 81, à l'adresse 2520 (09D8h). Son rôle est de rechercher l'adresse-mémoire d'un numéro de ligne du programme Basic.

Mode d'emploi

Le sous-programme « Ralp » doit être placé tout à fait à la fin du programme principal.

Il peut être sauvegardé (label : « Ralp ») et chargé avant l'introduction de tout programme nouveau devant contenir des routines en langage machine.

Le programme de démonstration de la **figure 1** utilise trois fois la routine « Ralp » . Il va nous permettre d'en expliquer le fonctionnement. L'objet de ce programme est de faire exécuter les routines en code machine, implantées dans des instructions REM, aux lignes (choisies au hasard) 85, 210 et 220. Il consiste en un affichage de trois groupes de deux caractères, avec identification de la routine exécutée. Nous en vérifions l'exécution à l'écran (**cf. fig. 2**).

Programme de démonstration

Les lignes 25 et 30 initialisent les variables d'exécution (IP) et d'attribution (AU) de « Ralp ». Les accès aux routines des lignes 85, 210 et 220 se font respectivement par des groupes de deux lignes 30/40, 100/110 et 300/310.

```
 10 REM "DEMONSTRATION"
 20 PRINT "FONCTIONNEMENT RALP"
 25 LET IP=0
 30 LET AU=9952
 40 LET N=85
 50 GOSUB AU
 70 PRINT
 80 PRINT "RESULTAT 1ER USER"
 85 REM 1234567
 90 FOR I=1 TO 50
 95 NEXT I
100 LET N=210
110 GOSUB AU
120 PRINT
130 PRINT "RESULTAT 2EME USER"
140 GOTO 300
210 REM 1234567
220 REM 1234567
300 LET N=220
310 GOSUB AU
320 PRINT
330 PRINT "RESULTAT 3EME USER"
340 STOP
```

Fig. 1. – Liste du programme de démonstration.

```
FONCTIONNEMENT RALP
*0
RESULTAT 1ER USER
1:
RESULTAT 2EME USER
2:
RESULTAT 3EME USER
```

Fig. 2. – Exécution des trois routines appelées via RALP.

Les couples ont la forme :
LET N = « numéro de ligne »
GOSUB AU
et sont équivalentes à une instruction du type :
GOSUB « USR de la ligne n°... »

Entrée d'une routine en code machine

Dans le programme de démonstration, les lignes 85, 210 et 220 ont sept positions réservées, après l'instruction REM, afin de contenir des routines en code machine. Pour introduire dans ces lignes des octets significatifs, il suffit, après l'entrée du programme principal, de faire pour chaque routine :
LET IP = 1
LET N = « n°... de ligne «REM» » (ici : 85, 210 ou 220)
GOTO 9952

Le programme place le caractère « 1: » en haut à gauche de l'écran et nous invite à entrer une chaîne de caractères.

Il faut alors taper le code (décimal) du premier octet de la routine (par exemple, 62).

62 s'inscrit derrière « 1: », et « 2: » apparaît. Le deuxième octet doit être fourni, et ainsi de suite...

Pour faciliter le contrôle du

```
9950 REM "RALP"[RECH.ADR.M  LIGNE
 PG.]
9952 LET RM=-13+PEEK 16396+256*P
EEK 16397
9954 LET HM=INT (RM/256)
9956 LET LM=RM-256*HM
9962 POKE RM+9,LM
9964 POKE RM+10,HM
9966 POKE 16394,N-256*INT (N/256
)
9968 POKE 16395,INT (N/256)
9970 RAND USR RM+2
9972 LET N=5+PEEK RM+256*PEEK (R
M+1)
9974 IF IP=1 THEN GOTO 9980
9976 RAND USR N
9978 RETURN
9980 LET P=-2+PEEK (N-3)+256*PEE
K (N-2)
9982 FOR I=1 TO P
9984 PRINT I;": ";
9986 INPUT J$
9988 IF J$="" THEN GOTO 9994
9990 IF J$<"A" THEN GOTO 9993
9991 LET I=I-1
9992 GOTO 9984
9993 POKE N-1+I,VAL J$
9994 PRINT PEEK (N-1+I),
9995 NEXT I
9996 LET IP=0
9997 PRINT
9998 LIST PEEK 16394+256*PEEK 16
395
9999 REM    E▒RNDLN **▒6  TAN
```

Fig. 3. – Liste du sous-programme Ralp.

ROUTINE PRINCIPALE (9999) : 12 octets après REM

	1	2	3	4	5	6	7	8	9	10	11	12
9999 REM			E	▒	RND	LN	**	▒	6			TAN
Code (décimal)	ADR.ligne		42	10	64	205	216	9	34	ADR. RM		201

Le 1er octet est à l'adresse RM

Fig. 4. – Détail de la routine principale de Ralp, ligne (9999).

Liste Assembleur	Signification
LD HL, (16394)	Chargement de HL avec le numéro de ligne déposé dans E-PPC
CALL 2520	Appel de la routine ROM du ZX 81 « recherche de l'adresse d'une ligne »
LD (RM), HL	Transfert de l'adresse restituée par la routine en RM, RM+1
RET	Retour au sous-programme Ralp

Fig. 5. – Liste Assembleur de la routine de recherche d'adresse.

```
 1:  62                2:  8
 3:  215               4:  62
 5:  6                 6:  215
 7:  201
 210 REM Y▒NOT Y▒NOT TAN
 220 REM 1234567
 300 LET N=220
 310 GOSUB AU
 320 PRINT
```

Fig. 6. – Liste générée après création de la routine dans la ligne (210).

```
 80 PRINT "RESULTAT 1ER USER"
 85 REM Y*NOT Y0NOT TAN
 90 FOR I=1 TO 50
 95 NEXT I
100 LET N=210
110 GOSUB AU
120 PRINT
130 PRINT "RESULTAT 2EME USER"
140 GOTO 300
210 REM Y▒NOT Y▒NOT TAN
220 REM Y▒NOT Y▒NOT TAN
300 LET N=220
```

Fig. 7. – Liste du programme de démonstration après création des trois routines en langage machine.

Liste des variables

- RM Adresse du sous-programme de recherche de la routine en langage machine.
- N Numéro de ligne contenant la routine, puis adresse de cette routine.
- IP Aiguillage de fonction. La valeur 0 provoque l'exécution des lignes (9974 à 9978) qui assurent l'appel à la routine en langage machine. La valeur 1 entraîne l'édition de la routine – lignes (9980) à (9988) – (**fig. 6 et 7**).
- P Nombre d'octets réservés dans l'instruction REM pour la routine.
- I Numéro courant de l'octet en cours d'édition.
- J$ Valeur lue au clavier pour l'éditeur. Toute valeur numérique provoque l'entrée en mémoire du code décimal dans l'octet I. Un caractère alphabétique équivaut à un retour d'un octet en arrière, tandis qu'une chaîne vide correspond à l'avance d'un octet.

travail, l'écran affichera, à la suite du dernier octet, le listing du programme principal à partir du numéro de la ligne qui vient d'être traitée.

Afin de faciliter le chargement, il est possible de se déplacer en avant et en arrière du numéro d'octet pointé, dans l'espace réservé.

Le retour en arrière ← s'effectue en entrant un caractère **alphabétique** au lieu d'un nombre.

Le saut en avant → se fait simplement sans rien entrer : les octets « sautés » ne sont pas modifiés. Par exemple, au cours du chargement de la ligne 85, les cinq premiers octets sont chargés ; l'écran affiche « 6: », la machine attend donc le sixième code.

Pour modifier le troisième, il suffit d'entrer une lettre puis de taper « New Line » et recommencer l'opération jusqu'à l'apparition de « 3: » ; le troisième octet peut être modifié. L'écran affiche alors « 4: ».

Pour revenir au sixième octet, il faut appuyer sur « New Line » jusqu'à l'apparition de « 6: » sur l'écran.

Dans la description qui suit, nous avons représenté les numéros des lignes du sous-programme « Ralp » (**fig. 3**) entre parenthèses – (nnn) – et les variables employées sont encadrées – VAR –.

Recherche de l'adresse du sous-programme (9952 à 9972 + 9999)

En premier lieu, le sous-programme Ralp recherche l'adresse mémoire de l'instruction REM contenant la routine à traiter. Ce traitement est effectué par la routine en langage machine stockée dans la ligne (9999) (**fig. 4**).

Pour cette partie du traitement, la variable N est fournie au sous-programme Ralp chargé avec le numéro de la ligne recherchée. Ce numéro est déposé dans la variable système E-PPC aux adresses 16394/16395. La variable RM contient l'adresse mémoire du premier octet suivant le code de l'instruction REM, ligne (9999).

La routine débute à l'adresse RM+2 et la liste des instructions le composant est détaillée **figure 5.**

Ensuite la variable N est actualisée avec l'adresse de la routine cherchée. ■