

# Frais kilométriques *Hp 48 & 49*

*Voici la méthode officielle 1999 qui calcule les frais occasionnés par vos déplacements en voiture.*

Votre boulot consiste de temps à autre à prendre le volant pour un rendez-vous d'affaire, seulement vous êtes obligé de prendre votre véhicule. Voici comment votre patron doit vous rembourser vos frais kilométriques : tout tient dans un tableau en fonction de la puissance fiscale de votre voiture, mais aussi en fonction du nombre de kilomètres parcourus (ici, **d** représente la distance en kilomètres).

Puissance Fiscale	Jusqu'à 5000 kms	De 5001 à 20000 kms	A partir de 20001 kms
3 CV et moins	d x 2,130	(d x 1,260) + 4351	d x 1,478
4 CV	d x 2,568	(d x 1,426) + 5711	d x 1,712
5 CV	d x 2,854	(d x 1,562) + 6461	d x 1,885
6 CV	d x 2,976	(d x 1,657) + 6597	d x 1,987
7 CV	d x 3,109	(d x 1,734) + 6877	d x 2,078
8 CV	d x 3,363	(d x 1,868) + 7477	d x 2,242
9 CV	d x 3,444	(d x 1,949) + 7477	d x 2,323
10 CV	d x 3,638	(d x 2,076) + 7812	d x 2,467
11 CV	d x 3,709	(d x 2,161) + 7742	d x 2,548
12 CV	d x 3,985	(d x 2,299) + 8432	d x 2,721
13 CV et plus	d x 4,056	(d x 2,370) + 8432	d x 2,792

Avant de poursuivre, il est vivement conseillé de créer un répertoire pour y mettre l'ensemble des fichiers qui va suivre. On procède de cette manière : ' KMS. DI R' CRDI R suivi d'un appui sur la touche [ENTER] pour valider le tout. Il faut ensuite entrer dans ce nouveau répertoire.

Tout d'abord, on entre successivement toutes les valeurs du tableau dans différentes listes : la première colonne (Jusqu'à 5000 kms) sera stocké dans ' KM1' , la seconde dans ' KM2A' et ' KM2B' puisque la formule présente deux paramètres, et la troisième colonne dans ' KM3' . Voici comment les stocker :

'KM1' (120,5 octets - CRC # F52Bh)

```
{ 2. 13 2. 568 2. 854 2. 976 3. 109 3. 363 3. 444
3. 638 3. 7093. 985 4. 056 }
```

'KM2A' (120,5 octets - CRC # 2E2Ah)

```
{ 1. 26 1. 426 1. 562 1. 657 1. 734 1. 868 1. 949
2. 076 2. 1612. 299 2. 37 }
```

'KM2B' (120,5 octets - CRC # DDDdh)

```
{ 4351. 5711. 6461. 6597. 6877. 7477. 7477.
7812. 7742. 8432. 8432. }
```

'KM3' (120,5 octets - CRC # 27A7h)

```
{ 1. 478 1. 712 1. 885 1. 987 2. 078 2. 242 2. 323
2. 467 2. 548 2. 721 2. 792 }
```

Passons maintenant au programme qui gère ces quatre listes pour donner les frais réels de votre trajet. Pour l'utiliser, il suffit de poser sur la pile et dans l'ordre la puissance fiscale et la distance en kilomètres, ceci avant de lancer le programme.

'KMS' (325,5 octets - CRC # F8FDh)

```
<< IF OVER 3. < THEN 3. SWAP ROT DROP END
IF OVER 13. > THEN 13. SWAP ROT DROP END
SWAP 2. - SWAP `` CV KM
<< IF KM 5000. % THEN KM1 CV GET KM * END
IF KM 5001. $ KM 20000. % AND THEN
KM2A CV GET KM * KM2B CV GET + END
IF KM 20001. $ THEN KM3 CV GET KM * END
"Fr$" ``TAG >>
```

Voici un petit exemple d'un véhicule de 6 CV qui a parcouru 17500 kilomètres, le résultat est instantané (avec les captures avant et après exécution) :



Philippe Pamart

# Quelle mémoire ! *Hp 49*

*La Hp 49 est dotée d'une mémoire fantastique pour une calculatrice de poche, voici un programme qui donne son état.*

Comme il l'a déjà été dit dans un supplément gratuit de Team Palmtops, la Hp 49 dispose d'une mémoire assez imposante, ce qui laisse énormément de marge à l'utilisateur pour travailler. Mais a-t-on toujours assez de mémoire ? En fait, la Hp 49 possède trois ports : le port 0 de 256 Ko qui se partage la Ram avec HOME, le port 1 de 256 Ko également, et le port 2 avec une capacité de 1 Mo de Ram flash (l'avantage de cette Ram flash, c'est qu'on ne perd aucune donnée quand la

machine subit un sévère plantage) ! Dans le répertoire racine HOME, on stocke tous les programmes et objets courants, tandis que dans les ports 0 et 1 on stocke les bibliothèques (library) de programmes. La partie Ram flash quant à elle sert à réaliser des sauvegardes.

Si l'on veut connaître la valeur exacte en Ko de l'espace mémoire libre qu'il nous reste dans chacun des ports, il suffit d'entrer dans le Filer (par un appui sur [Files]) pour obtenir par exemple cet écran :



Dans cette histoire, on ne connaît jamais la mémoire totale de la machine de chacun des ports. Réalisons donc un petit programme qui donne un état graphique de la mémoire par une barre d'état par port.

'MEMOIRE' (436 octets - CRC # 57ECh)

```

« ERASE { # 0h # 0h } PVIEW
  0 2 FOR I 1 PVARs NIP NEXT 3 "LIST
  { 262128 262128 1111736 } / 100 * R"B
  PICT { # 5h# 5h }
  "Mémoire libre" 2 "GROB REPL
  0 2 FOR I
    PICT 5 20 10 I * + 2 "LIST R"B
    "Port " I + 1 "GROB REPL
    30 21 10 I * + 2 "LIST R"B
    DUP { 100 3 } ADD BOX
    30 22 10 I * + 2 "LIST R"B
    DUP PICK3 I 1 + GET 1 2 "LIST ADD BOX
  NEXT DROP 7 FREEZE »

```

Dans le programme, on calcule d'abord le pourcentage de mémoire libre par port sachant que la mémoire totale par port est : 262128 octets pour le port 0, autant pour le port 1 et 1111736 pour le port 2. Ensuite, on fait une boucle qui affichera successivement les ports avec respectivement leur barre d'état de la mémoire. Les barres d'état sont réalisées avec de simples boîtes. Et on finit par un 7 FREEZE pour geler l'écran final.



Ici, le port 0 est quasiment vide, le port 1 est rempli à un peu plus de la moitié, alors que le port 2 dispose encore de toute sa mémoire. Il est bien évidemment possible d'afficher le pourcentage de mémoire libre sous chaque barre d'état : libre à vous de le réaliser puisque peu de modifications sont nécessaires.

Philippe Pamart

## Le compte est bon *Hp 48 & 49*

*"Faire ses comptes vite fait bien fait", telle est votre devise ! Voici un petit programme sans prétention qui s'occupe de cette tâche fastidieuse...*

Faire ses comptes n'est pas toujours une entreprise très passionnante surtout quand il faut tout faire à la main. Par contre, enregistrer ses recettes et ses dépenses au fur et à mesure élimine tous les calculs, mais aussi permet de gagner un temps considérable.

Le programme qui va suivre permet de gérer un compte courant de manière très simple : ajouter les recettes et soustraire les dépenses. Que faut-il pour gérer un compte ? Un programme qui permet de saisir ses recettes/dépenses, un programme qui va visualiser l'ensemble des transactions, et un programme qui va remettre à zéro l'état des transactions tout en gardant le solde du compte. Avant de commencer, créons un répertoire dans lequel les variables et programmes seront confortablement installés. Commençons par un programme qui gère l'ensemble :

'CPT' (178 octets - CRC # 9A98h)

```

« DO CLLCD ERASE
  IF "Comptes" { { "Nouvel objet" NEW. CPT }
  { "Voir comptes" VIEW. CPT } { "Efface..."
  PG. CPT } { "* Quitter *" KILL } } 1. CHOOSE
  THEN EVAL END UNTIL 0. END »

```

Si l'on désire saisir une nouvelle transaction on lancera le programme NEW. CPT, pour voir l'état de son compte ce sera VIEW. CPT, et enfin pour effacer l'état d'un compte tout en gardant le solde on utilisera PG. CPT. Une quatrième option

permet de quitter le logiciel si toutes les opérations sont effectuées. Voici à quoi ressemble le menu déroulant de notre logiciel



Passons maintenant à la phase de saisie des objets des différentes dépenses occasionnées. Il faut tout d'abord prévoir deux variables globales :

- SOLDE. CPT dans laquelle sera stockée la valeur courante de votre compte : cette variable aura la forme d'un réel,
- ETAT. CPT dans laquelle seront stockées toutes les opérations effectuées sur votre compte : cette variable aura la forme d'une liste.

Au démarrage du logiciel, la variable SOLDE. CPT doit contenir une valeur. Si votre compte commence avec un solde de 2000 francs, il suffit de stocker cette valeur dans la variable par un simple 2000 ' SOLDE. CPT' STO suivi d'un appui sur la touche [ENTER]. Ensuite, il faut aussi que ETAT. CPT contienne une liste vide au démarrage qu'on entre de cette manière : { } ' ETAT. CPT' STO toujours suivi d'un appui sur [ENTER].

Le programme de saisie aura pour mission de demander plusieurs choses avant que celles-ci soient ajoutées à la liste ' ETAT. CPT' et que le solde de votre compte soit modifié : la date de la transaction, l'objet de la transaction et enfin le montant de la transaction. Si la transaction est une dépense, alors la valeur du montant sera négative sinon elle sera positive. Cette saisie se fera dans une simple liste assistée bien sûr par un programme. Voici la tête de la liste :

```
{ Date "Objet" Montant }
```

La date est sous la forme JJ. MM où JJ est le jour et MM le mois de la transaction. Ensuite, l'objet est sous forme d'une chaîne de 16 caractères sinon

celle-ci sera tronquée. Et enfin le montant est un réel qui doit être compris entre -99999.99 et 999999.99 francs sous peine d'être lui aussi coupé à l'affichage (à moins de gagner le gros lot à la loterie, on sort quasiment jamais de cette fourchette). Pour concrétiser, voici deux petits exemples :

{ 23.10 "Pain" -5.6 } : le 23 octobre, vous avez acheté un pain qui vous a soulagé de 5 francs et 60 centimes,

{ 8.04 "Bingo" 50 } : le 8 avril, vous avez gagné 50 francs au Bingo.

Voici néanmoins le programme qui permet de vous faciliter la tâche :

#### 'NEW.CPT' (197 octets - CRC # 11D5h)

```
« ETAT.CPT *** Comptes ***
{date objet montant} {" INPUT STR
SOLDE.CPT OVER 3. GET + ' SOLDE.CPT' STO
1. LIST SWAP + ' ETAT.CPT' STO
" Nouveau solde : " SOLDE.CPT + " FF" +
MSGBOX »
```

A titre d'exemple voici les deux exemples sus-cités dans deux captures :



Après validation, le nouveau solde est affiché et on retourne au menu principal.

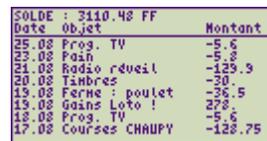
Si ensuite vous désirez visualiser toutes vos transactions, il suffit de créer un petit viewer qui s'en chargera. Dans ce genre de viewer, on demande à ce qu'il affiche le solde de votre compte, mais aussi le détails de vos dépenses et recettes. L'affichage ne se fera si et seulement si la liste ETAT.CPT est vide, cas dont on tiendra bien évidemment compte. On donnera aussi la possibilité à l'utilisateur de naviguer dans ses comptes avec les touches bas et haut. Enfin l'ensemble de l'affichage sera effectué dans l'environnement graphique PICTURE.

#### 'VIEW.CPT' (950 octets - CRC # F7E2h)

```
« IF ETAT.CPT SIZE 0. SAME THEN
" Pas d'objet" MSGBOX
ELSE ERASE { # 0h # 0h } PVIEW
PICT DUP { # 0h # 0h } " SOLDE : " SOLDE.CPT +
" FF" + 1. GROB REPL
{ # 0h # 6h } "Date Objet (13 espaces) Montant"
1. GROB REPL { # 0h # Ch } { # 83h # Ch }
LINE 1. 1. N K
« DO IF K 1. SAME THEN
PICT { # 0h # Dh } # 83h # 34h BLANK
REPL ETAT.CPT N DUP 7. + SUB
1. OVER SIZE FOR I DUP I GET 1.
« STR 1. GROB » DOLI ST
PICT # 0h # Eh I 1. - 6. * +
2. LIST PICK3 1. GET REPL
PICT # 18h # Eh I 1. - 6. * +
```

```
2. LIST PICK3 2. GET REPL
PICT # 60h # Eh I 1. - 6. * +
2. LIST ROT 3. GET REPL
NEXT DROP 0. ' K' STO END
0. WAIT
IF DUP 35.1 SAME
N 8. + ETAT.CPT SIZE % AND
THEN N 8. + ' N' STO 1. ' K' STO END
IF DUP 25.1 SAME N 8. - 0. > AND
THEN N 8. - ' N' STO 1. ' K' STO END
IF 45.1 SAME THEN 2. ' K' STO END
UNTIL K 2. SAME END »
END »
```

Si à partir du menu principal on sélectionne "Voir comptes", on atterrit sur la première capture et suite à un appui sur la touche [BAS] on arrive sur la seconde capture. On peut revenir sur le premier écran en appuyant sur [HAUT]. Enfin, pour revenir au menu principal, on appui sur [BACKSPACE].

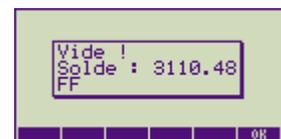


Reste à réaliser un petit programme qui efface l'état d'un compte tout en gardant le solde, c'est à dire qui efface tout ce qui se trouve dans ' ETAT.CPT '. Dans ce programme, on ajoute un boîte de confirmation au cas où l'option aurait été sélectionnée par erreur. Le programme d'effacement :

#### 'PG.CPT' (188,5 octets - CRC # D2CCh)

```
« IF "Etes-vous sûr ?"
{ { "Oui" « { } ' ETAT.CPT' STO "Vi de !" » }
{ "Non" "Ouf !!!" } }
2. CHOOSE
THEN EVAL "
Sol de : " + SOLDE.CPT + " FF" + MSGBOX END »
```

Par défaut, l'option "Non" est sélectionnée afin d'éviter une erreur supplémentaire ! Et quel que soit le choix de l'utilisateur, le programme ouvre une boîte qui renseigne une nouvelle fois sur le solde du compte. Voici, deux captures (pendant et après) :



Pour finir, les quarante-huitards changeront la commande PICK3 en 3 PICK dans les programmes concernés. Le reste doit fonctionner !