JPC

SEPTEMBRE 1993

NUMERO 89

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau — Editorial 1
Courrier du Coeur 2

HP28

G. Toublanc — Outils pour bidouilleurs 4

HP48

J. Belin — Des bugs sur les HP48G(X) 8
G. Toublanc — Transferts de fichiers sur HP48GX 8
G. Toublanc — Les triangles de Sierpinski 9
G. Toublanc — Compatibilité et HP48 15

HP-71

G. Toublanc — Triangles de Sierpinski sur HP-71 18

HP95

J. Belin — Configuration de QEDIT sur HP95 22

Le coin des codes 28

EDITORIAL

Comme les précédents numéros, ce JPC arrive avec un retard important. Il suffit d'en regarder le sommaire pour en connaitre la raison. En effet, c'est la première fois dans l'histoire du club qu'un journal est entièrement rédigé par deux personnes seulement.

Il y a quelques mois, nous avions pourtant cru à une réelle reprise des contributions. Depuis, certains habitués ont progressivement cessé de participer activement et les nouveaux venus n'ont pas "transformé" un essai qui avait pourtant été bien acceuilli par tous. On ne peut bien sur rien leur reprocher, mais pourquoi personne ne les ont remplacés ?

Certains d'entre vous déclarent être incapables de faire un article, car ils ne sont pas des spécialistes de la programmation. Or, il n'est pas besoin d'être un dieu de la programmation (nous mêmes ne le sommes pas) pour donner son avis ou rendre compte de son expérience sur les programmes que l'on utilise habituellement. Par exemple, les articles tels que le transfert de fichiers sur HP48 ou la configuration de QEDIT parus dans ce JPC peuvent être très bien écrits par un non-programmeur et sont suceptibles d'intéresser le plus grand nombre.

De l'autre côté, nous savons que certains adhérents ont développé des applications spécifiques à leur métier (ou leurs loisirs) mais ne nous les transmettent pas car ils les jugent trop spécifiques. Or, dans tous les cas en notre connaissance, nous savons qu'elles auraient intéressé *au moins* un adhérent. N'hésitez donc pas à nous contacter pour que nous discutions de la possibilité de les publier, ou au moins les rendre disponibles aux autres adhérents.

Certaines autres personnes déclarent être incapables de rédiger un article, sur quelques sujet que ce soit. Pourtant, la plupart de ces déclarations sont incluses dans des lettres d'une ou deux pages *manuscrites*, parfaitement rédigées, et sans aucune rature, provenant de personnes ayant toutes passé un certain "bac français" quelques mois ou quelques années plus tôt ! Pourtant il nous semble bien que la rédaction d'un article est beaucoup moins stressante que la dite épreuve : vous décidez vous même du sujet (qui normalement vous passionne), vous avez beaucoup plus de temps pour le préparer, et surtout vous n'êtes pas noté à la fin !

Enfin, il est toujours arrivé que les adhérents acquièrent de nouvaux accessoires ou livres (mêmes connus) bien avant que les membres du bureau ne les aient en mains. Dans le passé, dès qu'un adhérent entrait en possession d'un objet, il s'empressait de nous prévenir puis (si il était le premier) rédigeait un article de présentation. Aujourd'hui, tout le monde semble penser que ces objets nous tombent automatiquement dans les mains, alors que nous avons toujours eu le plus grand mal à nous faire prêter les nouveautés par les éditeurs ou distributeurs. Il est vrai que nous ne sommes pas diffusés à cent cinquante mille exemplaires...

En résumé, quelque soit votre niveau ou vos centres d'intérêts, il existe toujours un moyen de participer à la vie du club. Si vous avez un doute quant à un sujet d'article ou sur la façon de terminer un article, n'hésitez pas à nous contacter, nous sommes toujours là pour vous conseiller. Et surtout, rapellez vous toujours que si le journal a un peu de mal actuellement à arriver à 32 pages, il dépassait allègrement les 50 il y a quelques années et tout le monde se félicitait de la variété des sujets abordés. Notre objectif est de revenir à cette pagination et cette variété, mais il nous est bien sûr impossible d'y arriver à quelques-uns seulement...

COURRIER DU COEUR

Robert Vertente
8, rue des Marnaudes
93250 VILLEMOMBLE
Tel prof. : (1) 44 86 53 35

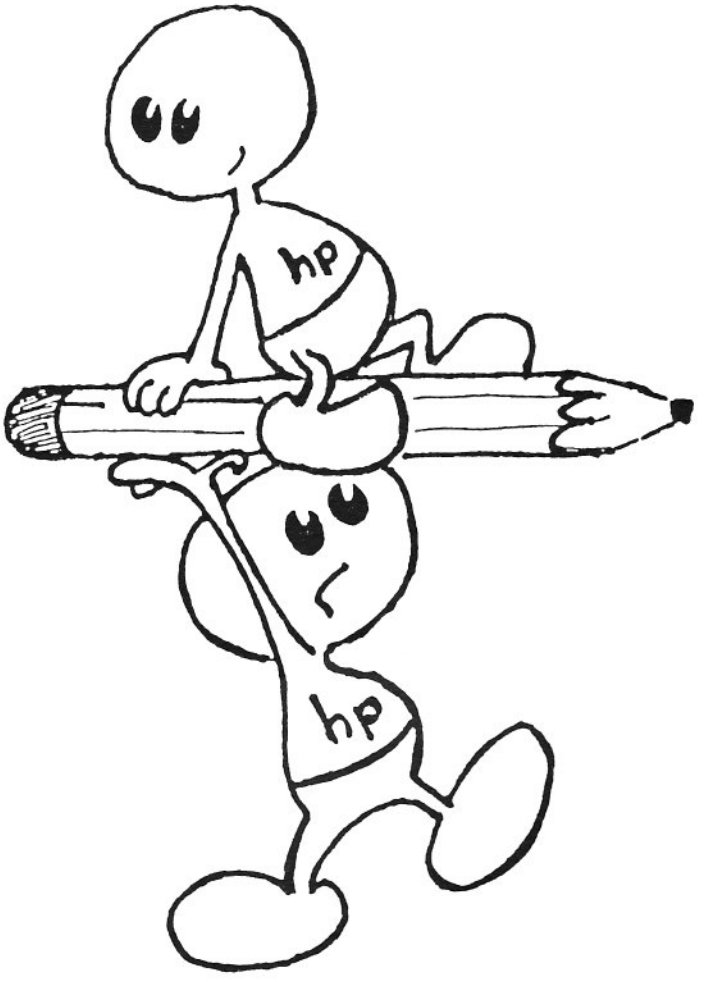
Vend :

HP27S, imprimante IR HP82240B, HP-71, lecteur de cartes avec cartes, module Maths, module HP-IL, lecteur de cassettes, cables, docs. Prix à débattre.

Philippe Guez
56, rue Jean-Jacques Rousseau
75001 Paris
Tel prof. : (1) 42 52 06 18

Cherche :

Modules Ram ou Eprom (32ko/64ko) pour HP-71.



HP28

G. Toublanc — Outils pour bidouilleurs *4*

Le coin des codes *28*

OUTILS POUR BIDOUILLEURS HP-28S

Tout d'abord je dois remercier deux personnes qui se sont manifestées pour formuler l'intérêt qu'elles portent à cette rubrique : Paul Jebeily qui a transformé sa HP-28 et qui, malgré son peu d'ancienneté dans le club, a su apporter déja sa contribution à JPC, puis Jean-François Garnier qui a participé sous plusieurs formes aux activités du club (articles, décompilateurs pour HP-28 et HP-48 recherche des points d'entrées HP-28...). Quant aux autres possesseurs de HP-28 c'est, comme trop souvent dans les clubs, la léthargie, l'inertie et l'indifférence.

Les programmes proposés ici sont des classiques pour tout curieux voulant aller au delà de ce que le constructeur a mis à notre disposition. Sauf le programme ADDR, ils ont déjà été publiés dans le livre *Au centre de la hp28...* et même PEEK et POKE ont vu le jour bien avant dans JPC où les auteurs ont fait leurs premières armes (certains oublient cela). Alors pourquoi une deuxième ou troisième parution ? A cela quatre raisons :
- Bien que la HP-28 ait un âge respectable, certains de ses possesseurs sont de nouveaux venus dans le clan HP et le livre cité n'est plus disponible.
- Les programmes déjà existants sont améliorables du point de vue vitesse et encombrement. On peut même dire que certains ne sont pas des modèles du genre bien qu'ils fonctionnent bien.
- La programmation en *System Rpl* était totalement ignorée. Or celle-ci est économique en mémoire et rend les programmes plus rapides.
- De nouvelles instructions utilisables en langage assembleur n'étaient pas connues des auteurs du livre. Celles-ci donnent des possibilités accrues de programmation.

Même si l'on a la possibilité de transférer sur un PC les programmes HP-28, il est trés agréable d'en avoir le plus possible immédiatement à sa disposition. Par exemple le PEEK du livre cité occupe 162 octets en faisant encore appel à un programme de vérification des arguments (CHK qui lui-même prend 200 octets). Celui présenté ici ne prend que 102,5 octets, est autonome et aussi plus rapide. Les 32 K de Ram de nos machines nous obligent à être plus soucieux des économies que les propriétaires de HP-48SX et GX.

Avertissement: depuis que j'ai relancé la rubrique HP-28 (JPC-84) pour l'assembleur et le *System Rpl* j'ai souvent fourni des commentaires explicatifs pour les différents programmes. Il ne m'est pas possible de donner à chaque fois, pour ce qui a déjà été expliqué, des commentaires aussi fournis. Aussi je vous renvoie aux précédents articles ou même aux programmes pour HP-48 lorsqu'il ne s'agit que d'une adaptation que je signale.

Pour commencer voici un petit programme (inconnu du livre) qui donne l'adresse d'un objet mis sur pile. Avant la partie code les instructions sont en *System Rpl*.

ADDR
cksum #6625h 32 octets

```
ASSEMBLE
        con(5)  #02c67    * DOCOL
        con(5)  #0c3e1    * CK1NOLASTWD
        con(5)  #10c2f    * BINT_0
        con(5)  #04f3d    * TOTEMPOB
        con(5)  #020e5    * SWAP
        con(5)  #02c96    * DOCODE
        rel(5)  fin
        a=dat1  a         * adresse de l'objet
        d1=d1+  5         * drop sur
        d=d+1   a         * cet objet
        c=dat1  a         * adresse de # 0 copié
        cd1ex             * en mémoire
        d1=d1+  10        * saute prolog + len
        govlng  #044b5    * fait dat1=a   a
*                           d1=c
*                           a=dat0   a
*                           d0=d0+   5
*                           pc=(a)
fin     con(5)  #02f90    * SEMI
```

Remarque: on doit pouvoir faire beaucoup mieux en utilisant la routine d'adresse #17520h qui semble être soeur de PUSHhxsLoop pour HP-48. Avis aux amateurs.

On sait qu'en mémoire le code hexa est inversé par rapport à celui que l'on transfère dans les divers registres de calcul. Aussi le très classique petit programme REV nous permet d'inverser l'ordre des caractères d'une chaîne (à la suite d'un PEEK par exemple.

REV
cksum # 417Ch 41 octets

```
ASSEMBLE
        con(5)  #02c67    * DOCOL
        con(5)  #0c3e1    * CK1NOLASTWD
        con(5)  #0c7a5    * Dispatch1
        con(5)  #071c7    * str ou THREE
        con(5)  #02c67    * DOCOL
        con(5)  #04f3d    * TOTEMPOB
        con(5)  #0204a    * DUP
        con(5)  #040c2    * LEN$
```

```
        con(5)  #02c96    * DOCODE
        rel(5)  fin
        gosbvl  #04f27    * POP#
        gosbvl  #05081    * =SAVPTR
        b=a     a         * voir JPC 82
        a=dat1  a         * page 28 pour
        d1=a              * les commen-
        d1=d1+  10        * taires (HP-48)
        cd1ex
        d1=c
        c=c+b   a
        c=c+b   a
        d0=c
        bsrb.f  a
        gonc    decr
loop    d0=d0-  2
        c=dat1  b
        a=dat0  b
        dat1=a  b
        dat0=c  b
        d1=d1+  2
decr    b=b-1   a
        gonc    loop
out     govlng  #125e5    * =GETPTRLOOP
fin     con(5)  #02f90    * SEMI
        con(5)  #02f90    * SEMI
```

Le troisième larron est PEEK qui permet de jouer les indiscrets c'est à dire traduire sous forme de chaîne le code hexadecimal pris à une adresse.

PEEK
cksum # C293h 102.5 octets

syntaxe :
niveau 2 : 1 entier binaire pour l'adresse
niveau 1 : 1 entier binaire pour le nombre
de quartets à prendre

```
ASSEMBLE
        con(5)  #02c67    * DOCOL
        con(5)  #0c3c4    * CK2NOLASTWD
        con(5)  #02c96    * DOCODE
        rel(5)  fin
        c=dat1  a         * adresse nbre de nib
        cd1ex
        rstk=c            * sauve adresse 1er niv.
        a=dat1  a         * prologue
        lchex   02a70     * pour entier binaire
        ?a=c    a         * entier binaire ?
        goyes   ok1
badtype govlng  #0c51b    * erreur: type invalide
ok1     d1=d1+  10        * saute prolog + len
        a=dat1  a         * nombre de nibs à poper
        r2=a              * sauvegardedu nb de nib
        a=c     a         * prologue entier binaire
        c=rstk            * adresse 1er niveau
        d1=c
        d1=d1+  5         * drop nb nib
        d=d+1   a         *
        c=dat1  a         * adresse de l'adresse
        cd1ex             * pour peeker
        rstk=c            * sauvegarde 2e niv.
        c=dat1  a         * prologue
        ?a#c    a         * diff. de ent. binaire?
        goyes   badtype   * erreur
        d1=d1+  10        * saute prolog + len
        c=dat1  a         * adresse où peeker
        r3=c              * sauvegarde
        c=rstk            * adresse 2e niv.
        d1=c
        c=r2              * nb nib
        c=c+c   a         * conversion en quartets
        c=c+con a,10      * ajoute prolog + len
        r4=c              * espace à reserver.
        gosbvl  #1e9ba    * res_str : chaîne reser
        d0=d0-  5         * revient sur len de str
        c=r4              * longueur réservée
        c=c-con a,5       * longueur str + 5
        dat0=c  a         * charge en mémoire
        d0=d0+  5         * revient sur 1er caract
        c=r2              * nb nib
        b=c     a         * compteur
        a=r3              * adresse où peeker
        d1=a
        lchex   39        * valeur ASCII de 9
        gonc    decr      * départ sur compteur
loop    a=c     b         * boucle de conversion
        a=dat1  1         * hexa <-> ascii
        ?c>=a   b
        goyes   inf10
        a=a+con b,7
inf10   dat0=a  b
        d0=d0+  2
        d1=d1+  1
decr    b=b-1   a
        gonc    loop
        a=r0              * adresse chaîne
        govlng  #1f7fd    * GPOveWrALp
*                         1er niveau (ex 2e) avec
*                         cette adresse et retour
fin     con(5)  #02f90    * SEMI
```

Terminons par la dangereuse commande POKE qui nous permet de changer le code en Ram donc à utiliser en connaissance de causes (surtout par ceux qui n'ont pas de solution de sauvegarde sur PC).

Cette commande fait donc l'inverse de PEEK : on lui donne une adresse et la chaîne des codes qui vont modifier la Ram à cette adresse. On doit faire la conversion ascii -> hexa à l'inverse de ce qui a été fait pour PEEK. Pour le reste voir PEEK.

POKE
cksum # 2E8Ch 91.5 octets

syntaxe :

2ème niveau : 1 entier binaire pour l'adresse
où poker
1er niveau : la chaîne de code à poker

```
ASSEMBLE
        con(5)  #02c67    * DOCOL
        con(5)  #0c3c4    * CK2NOLASTWD
        con(5)  #02c96    * DOCODE
        rel(5)  fin
        gosbvl  #05081    * SAVPTR
        c=dat1  a
        cd1ex
        rstk=c            * 1er niv.
        a=dat1  a         * prlogue
        lchex   02a4e     * prologue chaine
        ?a=c    a
        goyes   ok1
badtype govlng  #0c51b
ok1     d1=d1+  5
        a=dat1  a
        a=a-con a,5
        asrb.f  a
        r0=a              * nombre de codes
        d1=d1+  5         * @ 1er caract.
        c=rstk            * adresse 1er niv.
        cd1ex             *
        rstk=c            * adr. 1er car.
        d1=d1+  5         * @ 2ème niv.
        c=dat1  a
        cd1ex
        a=dat1  a         * prologue
        lchex   02a70     * prol. entier bin.
        ?a#c    a
        goyes   badtype
        d1=d1+  10
        c=dat1  a         * adresse où poker
        d0=c              *
        c=rstk
        d1=c              * @ 1er car.
        c=r0              * nbre de codes
        b=c     a
        lchex   39
        gonc    decr
loop    a=dat1  b         * conversion ascii->hexa
        ?a<=c   b
        goyes   inf10
        a=a-con b,7
inf10   dat0=a  1
        d0=d0+  1
        d1=d1+  2
decr    b=b-1   a
        gonc    loop
        govlng  #125e5    * GETPTRLOOP
fin     con(5)  #0211a    * 2DROP
        con(5)  #02f90    * SEMI
```

Nous voici donc en possession des premiers outils qui étendent les possibilités de la machine. Il ne reste plus qu'à les compléter par quelques autres et à passer aux applications.

Pour les projets d'avenir je pense qu'il faudrait s'orienter sur le graphisme (qui avait déja été abordé il y a longtemps dans JPC), à la fois en *User Rpl* et aussi en assembleur. J'ai toujours dans mes cartons un projet pour réaliser `LINE` en assembleur.

Je rappelle que pour programmer en *System Rpl* et en assembleur les compilateurs sur PC pour HP-71 ou HP48 conviennent, il suffit de donner les adresses des points d'entrées. Pour le reste le compilateur se charge des différentes tâches. La documentation de l'assembleur `SASM`, contenu dans le kit de développement HP, est valable pour les HP-28 qui sont équipées du microprocesseur Saturn.

En espérant un réveil de quelques uns.....

Guy Toublanc (276)

HP48

J. Belin	Des bugs sur les HP48G(X)	*8*
G. Toublanc	Transferts de fichiers sur HP48GX	*8*
G. Toublanc	Les triangles de Sierpinski	*9*
G. Toublanc	Compatibilité et HP48	*15*
	Le coin des codes	*28*

DES BUGS SUR LA GX

Au cours de ces derniers jours, nous avons eu connaissance de l'existence de quelques bugs présentes sur les HP48G et GX :

N'affichez pas l'horloge lorsque vous avez dans un port une carte Ram mergée avec la mémoire principale ou une bibliothèque attachée. Il en résulte un plantage de la machine (apparement sans perte des données)...

Si vous pressez EDIT alors que la ligne de commande est active, le menu correspondant apparait. Mais si vous sortez de la ligne de commande en pressant ENTER ou CANCEL, le menu reste affiché au lieu de laisser réapparaitre le menu précédent.

La fonction CHOOSE plante lorsque le premier argument est une liste ET que l'un des cas se présentent :
- Une liste avec moins de deux éléments est sélectionnée.
- Au moins un des autres arguments est une liste vide.
Ce bug présente un risque de corruption de la mémoire.

Notez bien que ces bugs sont présentes sur les premières version des machines. Au moment où vous lirez ces lignes, une version corrigée (code de version P) devrait être disponible chez tous les revendeurs. En cas de doute, utilisez la nouvelle commande VERSION.

Si vous avez connaissance d'autre bugs n'hésitez pas à nous en faire part.

Jacques Belin (123)

TRANSFERTS PLUS RAPIDES SUR HP-48G(X)

Le protocole série KERMIT était le seul à être présent dans la Rom de nos HP-48S(X) et s'il est apprécié par ses qualités de fiabilité on peut lui reprocher sa lenteur de transfert. Les HP-48G(X) nous gratifient d'un deuxième protocole série: *XMODEM* dont l'avantage est un gain de rapidité appréciable car ne faisant pas les vérifications multiples de Kermit. Par exemple :

Transfert d'un fichier binaire de 62282 octets entre HP-48GX et PC sous WINDOWS :

	Kermit	Xmodem
PC → HP-48GX	9'49''	1'16''
HP-48GX → PC	4'36''	1'38''

Donc des temps plus acceptables et aussi une usure moins importante des piles qui est liée aux durées de transfert.

Je ne peux pas donner d'exemple de transfert entre deux HP-48G(X) n'ayant vu qu'un seul de ceux-ci qui n'a pu être chronométré mais m'a semblé rapide.

Je signale que pour la communication série entre deux HP-48 il est possible de réaliser facilement un connecteur reliant deux cables HP (JPC 85 p15). Cela est d'autant plus d'actualité que HP offre gratuitement un cable pour une HP-48G(X) acquise entre le 1/9/93 et le 31/10/93.

Côté PC le protocole XMODEM est pratiquement présent dans la grande majorité des PC puisqu'il est intégré dans WINDOWS. Pour ceux qui ne l'ont jamais utilisé voici quelques indications :

Vous trouverez le sujet traité dans le manuel de WINDOWS au sous-chapître des accessoires : *Terminal* et pour la HP-48: le chapître *Utilisation de Xmodem* page 27-14 du manuel d'utilisation.

Il faut établir la configuration :

1- coté HP-48 faire [flèche cyan (droite)] [I/O] pour activer le menu déroulant TRANSFER. Descendre sur Transfer.. et valider avec [ENTER] puis aller sur TYPE et activer CHOOS pour valider Xmodem. Dans ce type de transfert l'affichage montre les autres paramètres par défaut :

```
port : Wire
baud : 9600        _OVRW
```

On remarquera qu'avec ce protocole il y a moins de paramètres qu'avec Kermit et qu'apparemment il n'y a que le mode binaire.

On peut opérer avec cette fenêtre pour la suite mais je préfère sortir en validant avec ENTER et utiliser tout de suite la méthode directe pour l'avenir.

Faire [flèche mauve (gauche)] [I/O] puis [NXT]. Mettre sur la pile le nom de l'objet que l'on veut transférer.

2- Côté PC en étant sous WINDOWS il faut activer succesivement ACCESSOIRES puis TERMINAL, en étant dans le menu Fichier activer Nouveau.

Passer dans le menu Paramètres et activer la fenêtre Communications. Laisser les paramètres par défaut :

```
vitesse de transmission : 9600 bauds
bits de données         : 8
bit d'arrêt             : 1
parité                  : aucune
```

choisir :

```
contrôle de flux        : aucun
port                    : COM1 ou COM2 suivant
                          la connexion du cable HP
```

Après avoir validé, revenir au menu Fichier pour sauvegarder cette configuration pour un futur retour dans TERMINAL où Nouveau sera remplacé par Ouvrir (fichier). Donc sauvegarder avec Sauver, le fichier aura l'extension TRM (HP48.TRM semble tout indiqué pour ce fichier de configuration).
Passer au menu Transferts et choisir Envoyer un fichier binaire ou Recevoir un fichier binaire.

Le nom du fichier étant donné :

Pour un transfert HP → PC valider sur le PC puis activer XSEND sur la HP-48.

Pour un transfert PC → HP activer XRECV sur la HP-48 puis valider sur le PC.

Il faut attendre quelques instants avant de voir s'effectuer le transfert qui après se fait plus rapidement que par Kermit.

Voici donc des indications assez brutes sur ce deuxième protocole et WINDOWS car je ne suis pas un habitué de ce dernier. Il doit être possible de trouver des astuces pour activer plus rapidement Xmodem à la fois sur PC et HP-48. Faites nous part de vos découvertes dans la rubrique *Trucs, astuces et petits utilitaires* qui est tout indiquée pour cela.

Remarque : dans PCTOOLS 6 on trouve aussi XMODEM comme dans d'autres intégrés (WORKS etc..). Il y a donc de grandes chances que vous ayez ce protocole dans votre PC.

Guy Toublanc (276)

GRAPHISME EVOLUTIF OU LE CHAOS MAITRISE

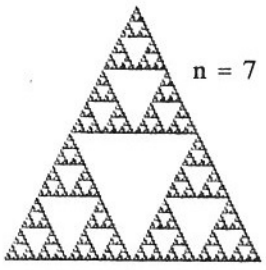
Le graphisme n'a pas eu beaucoup de succès dans les colonnes de JPC puisqu'il faut remonter au n°78 pour trouver l'unique article dans ce domaine (NDLR : en ce qui concerne la 48) et sous la signature de votre serviteur. Faut-il en déduire un manque d'imagination des lecteurs de JPC ?

La construction des triangles de Sierpinski me permet de revenir à la fois sur le graphisme, la transcription d'un programme écrit en *User Rpl* en un programme en langage *System Rpl* et assembleur, et l'optimisation dans ce dernier langage.

1 - User Rpl avant tout

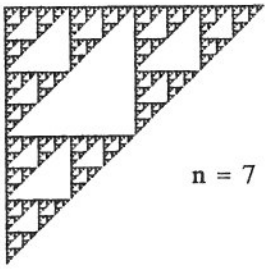
Ce langage, s'il est méprisé par quelques uns (qui ne savent pas ou plus s'en servir), est bien celui auquel on doit avoir recours en premier lieu pour débuter une programmation. En effet il est :
- sécurisant, il ne plante pas la machine
- le débogage est aisé
- la traduction d'une formule est naturelle
- la valeur des résultats intermédiaires peut être obtenue très simplement.

Le domaine du chaos et des fractales est propice à la génération de dessins attactifs. Bien sûr nos HP-48 ne nous permettent pas d'obtenir des effets aussi spectaculaires qu'avec un PC et un moniteur couleurs. En limitant nos ambitions nous pouvons tout de même nous faire plaisir. C'est ce que je vais tenter en volant un programme de notre ami Robert Pulluard (tout de même avec son aimable autorisation). Robert a débuté, dans son bulletin 48sxtant, une rubrique *chaos et fractales* où il expose très simplement, comme à son habitude, quelques thèmes de ce difficile sujet et parmi ceux-ci les *triangles de Sierpinski*. Je vous renvoie au n° 22 de *48sxtant* pour plus de détails et en prenant un raccourci je dirai simplement que ces triangles sont la représentation en noir et blanc du triangle de Pascal où le noir correspond aux valeurs impaires. Sans calculer les valeurs on peut *noircir* en faisant un AND binaire avec les deux coordonnées des points du triangle, un résultat nul se traduisant par du noir.



Ces triangles de Sierpinski possèdent la propriété d'*auto-similarité* c'est-à-dire que si l'on fait un zoom sur une zone on retrouve la même structure que celle qui nous apparaît pour l'ensemble.

Pour simplifier la programmation, Robert construit le triangle rectangle dérivé de l'original et cela avec le programme ci-dessous SIER2 qui affiche le triangle au fur et à mesure de la construction. Il suffit de fournir un nombre entier n.

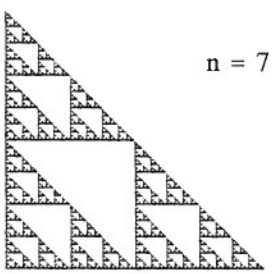


SIER2

```
« FLOOR 2 SWAP ^ 1 - 1 MAX → n
  « n R→B DUP BLANK PICT STO
    { # 0h # 0h } PVIEW
    0 n
    FOR r
      0 n r -
      FOR   s
        r R→B s R→B DUP2
        IF AND R→B
        THEN DROP2
        ELSE 2 →LIST PIXON
        END
      NEXT
    NEXT
    7 FREEZE
  »
»
```

2 - Traduction simple

Nous allons traduire ce programme en *System Rpl* et assembleur. La seule différence sera d'obtenir un triangle avec la pointe en haut, diposition qui nous semble plus natuerelle, pour le reste c'est l'analogie complète : le programme effectue un balayage par colonnes de 1 point.



On se rappellera la structure d'un objet *graphic*: les lignes sont bout à bout et leur longueur réelle est un multiple de 8 (8 bits = 1 octet). Si pour l'affichage les lignes font 131 points il faut savoir qu'elles correspondent à 136 bits soit 34 quartets ou 17 octets.

Pour allumer un point on fera par exemple :

```
cbit=1   n
```

sachant que l'ordre dans un registre est l'inverse de celui en mémoire.

Les instructions de la première ligne ci-dessous vérifient la présence et la nature de l'argument puis les autres sont les homologues de celles en *User Rpl*, mais plusieurs pouvant être groupées en une seule. Puis vient la partie assembleur qui *noircit* les points à l'aide d'une boucle générale et de 4 sous-boucles travaillant chacune sur un bit différent.

```
::
  CK1NOLASTWD CK&DISPATCH0 real
  ::
    %FLOOR %2 SWAP %^ COERCE ONE #MAX
    DUPDUP MAKEGROB GROB>GDISP
    ZEROZERO WINDOW#
CODE
        gosbvl =POP#   * 2^n nombre de points de
        r0=a           * la base et de la hauteur
        gosbvl =SAVPTR
        d0=(5) #01f79 * adresse où CLCD10 se
        a=dat0 a       * se sert de VDISP
        lchex  7055b   * VDISP pour HP-48S..
        ?a=c   a       * version HP-48S..?
        goyes  hp48s
        lchex  806da   * VDISP pour HP-48G..
hp48s   d0=c
        a=dat0 a       * adresse du PICT
        d0=a
        d0=d0+ 5       * @ grandeur totale
        a=dat0 a       * grandeur totale en nibs
        cd0ex
        d0=c
        c=a+c   a      * adresse fin PICT
        d1=c
        d1=d1-  1      * @ fin PICT - 1
        a=a-con a,15   * longueur L du PICT
        d0=d0+ 5       * @ la hauteur h
        c=dat0 a       * nombre de lignes h
        gosbvl =IntDiv * L / h := nibs par ligne
        d=c    a       * nb quartets par ligne
        a=0    a       * compteur colonnes
        c=r0           * hauteur
        c=c-1  a       * compteur lignes
******  boucle pour le balayage vertical  *******
loop    rstk=c         * sauve le compteur lignes
        b=c  x         * compteur lignes boucle0
        d1=d1+ 1       * ligne n : quartet suivant
```

```
        cd1ex
        c=c-d  a       * remonte d'une ligne
        d1=c           * @ début ligne n - 1 nib
        d0=c           * dito    pointeur quartets
*******  boucle pour 1 colonne de 1 bit  *********
********  bit 0 du quartet (en mémoire)  *********
loop0   c=a    x       * n° ligne
        c=c&b  x       * AND sur coordonnées
        ?c#0   x       * si <> 0 on ne noircit
        goyes  nopix0  * pas
        c=dat0 1       * le quartet à noircir
        cbit=1 0       * on le fait avec le bit 0
        dat0=c 1       * stocke en mémoire
nopix0  cd0ex
        c=c-d  a       * adresse ligne précédente
        cd0ex          * pour la boucle 0
        b=b-1  x       * compteur lignes
        gonc   loop0   * encore 1 tour
        cd1ex          * adresse ligne n
        c=c-d  a       * adresse ligne précédente
        d1=c           * @ ligne précédente
        d0=c           * dito   pointeur quartets
        a=a+1  x       * colonne suivante
        c=rstk         * compteur lignes boucle1
        c=c-1  x       * actualiation
        goc    gocout  * si plus de lignes : sort
        rstk=c         * sauve compteur lignes
        b=c    x       * compteur lignes boucle1
*******  boucle pour 1 colonne de 1 bit  *********
********  bit 1 du quartet (en mémoire)  *********
loop1   c=a    x       * n° de colonne
        c=c&b  x       * AND sur coordonnées
        ?c#0   x       * etc...
        goyes  nopix1  * on répète les mêmes
        c=dat0 1       * opérations avec :
        cbit=1 1       * boucle 1 pour bit 1
        dat0=c 1       * boucle 2 pour bit 2
nopix1  cd0ex          * boucle 3 pour bit 3
        c=c-d  a
        cd0ex
        b=b-1  x
        gonc   loop1
        cd1ex
        c=c-d  a
        d1=c
        d0=c
        a=a+1  x
        c=rstk
        c=c-1  x
gocout  goc    out
        rstk=c
        b=c  x
*******  boucle pour 1 colonne de 1 bit  *********
********  bit 2 du quartet (en mémoire)  *********
loop2   c=a    x
        c=c&b  x
        ?c#0   x
        goyes  nopix2
        c=dat0 1
        cbit=1 2
        dat0=c 1
nopix2  cd0ex
        c=c-d  a
        cd0ex
        b=b-1  x
        gonc   loop2
        cd1ex
        c=c-d  a
        d1=c
        d0=c
        a=a+1  x
        c=rstk
        c=c-1  x
        goc    out
        rstk=c
        b=c  x
*******  boucle pour 1 colonne de 1 bit  *********
********  bit 3 du quartet (en mémoire)  *********
loop3   c=a    x
        c=c&b  x
        ?c#0   a
        goyes  nopix3
        c=dat0 1
        cbit=1 3
        dat0=c 1
nopix3  cd0ex
        c=c-d  a
        cd0ex
        b=b-1  x
        gonc   loop3
        a=a+1  x       * actualise compteur colonne
        c=rstk         * compteur lignes boucle
        c=c-1  x       * générale et actualisation
        goc    out     * sort si le compteur était
        goto   loop    * à zéro sinon encore un
out     govlng =GETPTRLOOP * tour de grande boucle
ENDCODE
    SetDAsTemp         ( gèle l'affichage )
  ;
;
```

3 - Un peu d'optimisation

Le programme ci-dessus est la traduction pas-à-pas du programme de Robert. Mais si en *User Rpl* on adresse directement un pixel pour l'allumer, en assembleur nous sommes contraints de faire les transferts registre<->mémoire au moins par quartet c'est-à-dire 4 bits à la fois. Donc il est plus rationnel de travailler à la fois sur les 4 bits d'un quartet pour ne faire qu'un transfert. Mais nous ne *noircirons* plus avec `cbit=1 n` qui serait perdu avec le *AND* suivant. Cette opération se fera en manipulant les status bits (st) qui peuvent s'échanger avec le champ X du registre C. D'autre part au lieu de faire un balayage par colonne cela se fera par ligne, l'avancement

s'effectuant par quartets d'une même ligne. Le résultat sera un programme moins encombrant et plus rapide.

On aura pu remarquer que les opérations se font sur le champ X pour les colonnes et les lignes. Cela se justifie car la Ram maximale avec 2 cartes d'extension ne permet pas d'aller au-delà de n=9 ce qui correspond à 512 points pour la base et la hauteur du triangle. Donc le champ X convient et permet de gagner un peu de rapidité par rapport à l'usage du champ A. Voici la partie du listing modifié. Les labels p0 à p3 ne servent à rien pour l'instant.

```
CODE
        gosbvl =POP#
        ....
        ....
        c=r0
        c=c-1 a
******  boucle pour quartets 'horizontaux'  ******
loop    rstk=c          * sauve compteur lignes
        b=c  x          * compteur colonnes
        cd1ex
        c=c-d  a        * adresse ligne précédente
        d1=c            * @ ligne n - 1 quartet
        d0=c            * dito   pointeur quartet
loop0   clrst           * mise à zéro des status bits
        d0=d0+ 1        * quartet suivant de la ligne
p3      c=a    x        * n° de ligne
        c=c&b  x        * AND sur coordonnées
        ?c#0   x
        goyes  nopix0
        st=1   0        * remplace cbit=1 0
nopix0  b=b-1  x
        goc    load     * si dernière colonne
p2      c=a    x        * sinon bit suivant
        c=c&b  x        * etc...
        ?c#0   x
        goyes  nopix1
        st=1   1
nopix1  b=b-1  x
        goc    load
p1      c=a    x
        c=c&b  x
        ?c#0   x
        goyes  nopix2
        st=1   2
nopix2  b=b-1  x
        goc    load
p0      c=a    x
        c=c&b  x
        ?c#0   x
        goyes  nopix3
        st=1   3
nopix3  b=b-1  x
load    c=st           * c(x) <- status bits
        dat0=c 1       * stocke en mémoire
        gonc   loop0   * si encore 1 colonne
        a=a+1  x       * actualise compteur lignes
        c=rstk         * lignes non traitées
        c=c-1  x       * actualise compteur
        gonc   loop    * si encore ligne à traiter
        govlng =GETPTRLOOP
ENDCODE
```

4 - Revenir aux sources

Les programmes ci-dessus génèrent des triangles rectangles aussi nous allons traiter le problème un peu plus compliqué du tracé de triangles comme celui présenté au tout début. Nous n'avons donc plus de côté vertical et nous devons normalement décaler chaque ligne d'un demi point car la base a le même nombre de points que la hauteur. Ceci oblige donc à décaler d'un point toutes les deux lignes. Chaque ligne n'étant pas de longueur multiple de 4 nous ne pouvons faire le décalage d'un quartet à chaque boucle comme précédemment. Il faut donc comptabiliser ces sauts d'un quartet d'une autre manière. Cela se fera à l'aide du registre P qui nous permettra de nous brancher à un niveau de boucle correspondant au décalage d'un bit toutes les 2 lignes.

Voici la partie du listing modifié par rapport au programme précédent. Pour les labels p0 à p3 voyez le listing ci-dessus.

```
        .......
        .......
        c=c-1 a
        p=   8       * initialisation de P
*******  boucle pour balayage horizontal  *******
loop    rstk=c
        b=c  x          * compteur lignes
        p=p-1
        gonc   idem     * pas de décalage d'1 nib
        p=     7        * sinon P réinitialisé
        d1=d1+ 1        * décalage d'un quartet
idem    cd1ex
        c=c-d  a        * adresse ligne précédente
        d1=c
        d0=c
        d0=d0+ 1        * décalage d'un quartet
        clrst           * mise à zéro des status bits
        ?p=    5
        goyes  p2       * départ de ligne sur bit 1
        ?p=    4
        goyes  p2       * départ de ligne sur bit 1
        ?p=    3
        goyes  p1       * départ de ligne sur bit 2
        ?p=    2
        goyes  p1       * départ de ligne sur bit 2
        ?p=    1
        goyes  p0       * départ de ligne sur bit 3
        ?p=    0
```

```
        goyes  p0      * départ de ligne sur bit 3
        gonc   p3      * départ de ligne sur bit 0
loop0   clrst
     .....
     .....
```

5 - objets inanimés...

Avec les programmes ci-dessus les triangles sont affichés d'office mais on peut désirer créer une animation graphique avec un programme d'animation qui doit activer l'affichage suivant le temps désiré. Je vous propose des modifications aux programmes ci-dessus : les objets graphiques sont générés mais sans être affichés.

Voici les modifications de début et de fin de programme par rapport au listing du paragraphe 3 *un peu d'optimisation*. La création de l'objet graphique se fait maintenant au début de la partie *assembleur* ce qui nous montre une autre façon de programmer.

```
::
  CK1NOLASTWD CK&DISPATCH0 real
  ::
    %FLOOR %2 SWAP %^ COERCE ONE #MAX
CODE
        gosbvl =SAVPTR
        c=dat1 a
        d1=c
        d1=d1+ 5       * @ 2^n
        c=dat1 a       * hauteur et largeur
        r0=c           * pour makegrob
        r1=c           * pour makegrob
        gosbvl =makegrob * création du graphic
        d0=d0- 16
        d0=d0- 4       * @ adresse de l'objet
        cd0ex
        cr3ex          * R3 <- adr graph
        d=c    a       * nb quartets/ligne (de R3)
        d1=d1- 1       * @ fin graph. - 1 quartet
        a=0  a         * initialise n° de ligne
        c=r0           * hauteur
        c=c-1 a        * compteur lignes à traiter
loop    rstk=c         * sauvegarde
        ....
        ....
load    ....
        gonc   loop
        a=r3           * adresse du graph.
        govlng =GPOverWrALp * GETPTR puis
*                        l'adresse de graph.
*                        remplace celle de 2^n
ENDCODE
  ;
;
```

6 - Noir ou blanc

Nous pouvons obtenir des triangles en inversion vidéo avec la modification suivante :

à lintérieur des boucles il suffit de remplacer `?c#0 x` par `?c=0 x`

De là à faire des programmes travaillant avec les 2 modes graphiques il n'y a qu'un pas à franchir :
un entier positif génèrera un triangle normal et un entier négatif génèrera un triangle inversé.

Voici donc les modifications à apporter :

1 - pour les programmes qui à la fois construisent et affichent les objet graphiques :

```
:: CK1NOLASTWD CK&DISPATCH0 real
 :: DUP %0< ITE FALSE TRUE SWAP
    %ABS %FLOOR ..............
    ............. WINDOW# SWAP
CODE
        gosbvl =popflag   * ajout
        a=0    s          *
        goc    posit      *
        a=a+1  s          *
posit   gosbvl =POP#
        r0=a
        gosbvl =SAVPTR
        ...............
        ...............
loop0   clrst
        d0=d0+ 1
        c=a    x
        c=c&b  x
        ?c=0   s          *  ajout qui se répète
        goyes  pos0       *  pour chaque bit
        ?c=0   x          *
        goyes  nopix0     *
        gonc   black0     *
pos0    ?c#0   x
        goyes  nopix0
black0  st=1   0
nopix0  b=b-1  x
        ........
        ........
black3  st=1   3
nopix3  b=b-1  x
load    c=st
        dat0=c 1
        goc    nxtline    * remplace gonc qui
        goto   loop0      * plus possible ici
nxtline a=a+1  x
        c=rstk
        c=c-1  x
        goc    out        * remplace gonc qui
```

```
        goto    loop        * plus possible ici
out     govlng =GETPTRLOOP
```

2 - pour les programmes qui construisent seulement les objets graphiques mais ne les affichent pas :

```
:: CK1NOLASTWD CK&DISPATCH0 real
 :: DUP %0< ITE FALSE TRUE
    SWAP %ABS %FLOOR .....
    ............#MAX SWAP
CODE
        gosbvl =popflag   * ajout
        c=0    s          *
        goc    posit      *
        c=c+1  s          *
posit   gosbvl =SAVPTR
        ........          * puis mêmes ajouts et
        ........          * et modifications que
        ........          * ci-dessus pour les
                          * mêmes raisons
```

Evidemment ces mofifications ralentissent un peu l'exécution des programmes mais cette polyvalence est appréciable.

Conclusion:

Nous avons vu qu'il était possible de suivre pas à pas un programme réalisé en *User Rpl* pour le traduire en assembleur, ce qui ne donnera pas un programme optimisé. Pour obtenir ce dernier il faut tenir compte d'une containte de l'assembleur : la plus petite unité de transfert entre registres de calcul et mémoire est le quartet. Il faut donc modifier la programmation pour réduire le nombre de transferts. De ce point de vue nous aurions pu augmenter encore la rapidité des programmes en travaillant sur les 2 quartets du champ B.

Vous trouverez dans le coin des codes les listes de codes correspondant aux programmes suivants :

`DSIERREC`
construit et affiche des triangles rectangulaires

`DSIERISO`
construit et affiche des triangles isocèles

`GSIERREC`
construit des triangles rectangulaires

`GSIERISO`
construit des triangles isocèles

Tous ces programmes acceptent des entiers positifs ou négatifs et sont compatibles HP-48S. et HP-48G.

Pour terminer voici un petit programme qui n'a pas la prétention de faire du grand spectacle mais qui n'est qu'un modeste exemple de possibilité de réaliser des animations. Il simule une coulée de laves lors d'une éruption volcanique. Il doit être possible de faire beaucoup mieux avec d'autres paramètres. A vous de jouer.

ERUPT

```
« -6 GSIERREC PICT STO
  { # 0d # 0d } PVIEW 2 WAIT
  3 127
  FOR I PICT I R→B DUP 2 / 1 - 2 →LIST
      I 7 MOD GSIERREC GXOR
  NEXT 7 FREEZE
»
```

Il suffit d'appeler `ERUPT` pour voir cette petite animation.

ANIMATE sur HP48G

Sur HP-48G, la commande `ANIMATE ( → PRG → GROB )` devrait donner des possibilités dans ce domaine. Mais peut-être que vous ne pouvez pas le faire car le manuel utilisateur est erroné :

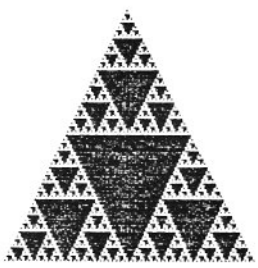
Si celui-ci indique bien qu'il faut fournir au niveau 1 le nombre d'objets graphiques *n* puis du niveau 2 au niveau n+1 les n objets graphiques, pour la 2ième option il faut toujours fournir les n objets graphiques et au niveau 1 une liste :

```
{ n { # x  # y } d r }
```

où :
n est le nombre d'objets à animer
x y sont les coordonnées d'affichage du coin supérieur gauche du graphique
d est le délai d'affichage
r est le nombre de répétitions

Ces précisions m'ont été aimablement communiquées par notre ami Robert Pulluard.

Guy Toublanc (276)



COMPATIBILITE ET HP-48G(X)

Certains possesseurs de HP-48S(X) puis HP-48G(X) ont pu constater que certains programmes écrits en *System Rpl* et assembleur pour HP-48S(X) sont incompatibles avec les HP-48G(X).

A cela il y a deux raisons évidentes :

- L'utilisation de progammes internes dont les points d'entrée ne sont pas supportés par HP qui pour des raisons techniques est obligé de modifier ou d'ajouter des programmes internes.

- Une raison physique puisque la principale RAM interne à subi une translation de #10000h passant de #70000h à #80000h.
Il y à encore d'autres raisons puisque l'on peut constater que des librairies qui fonctionnent bien en port 0 ou 1, ne fonctionnent plus en port 2.

Ayant une HP-48GX depuis peu de temps je n'ai pas pu explorer très largement la ROM.

Il est possible de trouver des correspondances entre les deux types de machines et cela très simplement. Il suffit de désassembler à partir de points d'entrée supportés. Par exemple pour le programme assembleur interne SAVPTR qui sauve les pointeurs.

SAVPTR

```
     HP-48S(X)                HP-48G(X)

0679B CD0EX              0679B CD0EX
0679E D0= 705B0          0679E D0= 8072F
067A5 DAT0=C A           067A5 DAT0=C A
067A8 D0= 70579          067A8 D0= 806F8
067AF CD1EX              067AF CD1EX
067B2 D1=C               067B2 D1=C
067B5 DAT0=C A           067B5 DAT0=C A
067B8 D0= 70574          067B8 D0= 806F3
067BF C=B A              067BF C=B A
067C1 DAT0=C A           067C1 DAT0=C A
067C4 D0= 7066E          067C4 D0= 807ED
067CB C=D A              067CB C=D A
067CD DAT0=C A           067CD DAT0=C A
067D0 RTNCC              067D0 RTNCC
```

On voit donc que si les programmes sont aux mêmes adresses les sauvegardes de pointeurs se font à des adresses décalées de #1017Fh. Mais cette translation n'est pas forcément constante pour tous les programmes internes. Donc il faut rester attentif et toujours faire des contrôles avant de convertir un programme HP-48S(X).

Dans le programme CIRCLE (JPC 78 pages 17-21) on trouve au début du listing assembleur :

```
D0=(5) #70565  * =VDISP
A=DAT0 A
```

pour avoir l'adresse du PICT.

Pour trouver VDISP sur HP-48G(X) on peut procéder de la manière suivante :

Chercher à l'aide du programme FINDMEM de Laurent Grand (JPC-85), par exemple, les adresses où l'on trouve le code "1F56507" dans la Rom HP-48S(X) et correspondant à D0=(5) #70565. Parmi les diverses adresses la première est # 01F77. En se reportant à la liste des points d'entrée HP on voit que le programme CLCD10 est voisin. En désassemblant à l'aide d'un décompilateur, par exemple MON48 de Jean-François Garnier, on voit que CLCD10 utilise VDISP. Si la Rom HP-48G(X) a été transférée sur PC on désassembler celle ci à partir de CLCD10. Sinon en faisant dans la HP-48G(X) un PEEK à l'adresse de CLCD10 on peut obtenir les résultats suivants :

CLCD10

```
     HP-48S(X)                              HP-48G(X)

01F6D 01F22 Primitive                       22F10
01F72 137           CD1EX                   137
01F75 06            RSTK=C                  06
01F77 1FB5507       D1=(5) =VDISP           1FAD608
01F7E 147           C=DAT1 A                147
.........           ........                ...
```

Le désassemblage montre que les deux programmes sont aux mêmes adresses et identiques sauf la 4ième ligne. A droite 1FAD608 correspond à D1=(5) #806AD alors qu'à gauche nous avons D1=(5) #7055B. Cela nous permet de récupérer l'adresse de VDISP pour HP-48G(X) soit # 806DA.

Pour les programmes utilisant VDISP il suffira de remplacer #7055B par #806DA pour obtenir les versions compatibles HP-48G(X).

Il y a une autre solution qui consiste à faire un seul programme compatible HP-48S(X) et HP-48G(X). Il suffit de détecter le type de Rom et d'utiliser le VDISP correspondant. Donc le programme pourra être modifié de la manière suivante :

```
D0=(5) =VDISP
A=DAT0 A
........
```

sera remplacé par:

```
        DO=(5) #01F79
        A=DATO A
        LCHEX  7055B
        ?A=C   A
        GOYES  hp48s
        LCHEX  806DA
hp48s   DO=    C
        A=DATO A
        ........
```

C'est cette procédure que j'ai employé pour les programmes de tracé de triangles de Sierpinski qui sont publiés dans ce même JPC.

Pour CIRCLE nous sauvegarderons C(A) avant la modification ce qui nous donnera :

```
        ............
        GOSUB  conver
        RSTK=C
        DO=(5) #01F79
        ............
        ............
hp48s   DO=    C
        C=RSTK
        A=DATO A
        ........
```

Vous trouverez la liste des codes pour le nouveau programme CIRCLE, compatible G et S, dans *le coin des codes*. Pour ceux qui ont la version S ils pourrons éviter de tout retaper en utilisant le programme ci-dessous qui nécessite →ASC et ASC→ (voir ci-dessous) ou équivalents si compatibilité G.

modicircle
cksum # D3C5h 303.5 octets

```
« 'CIRCLE' RCL
  BYTES DROP # DA13h ≠
  IF THEN # 203h DOERR END
  'CIRCLE' RCL →ASC → S
  « S 1 25 SUB "1D" + S 28 67 SUB +
    "51206" + "1B97F" +
    "10142" + "34B55" +
    "078A2" + "9034A" +
    "D6081" + "3407" +
    S 78 OVER SIZE SUB + ASC→
  »
»
```

Rapide inventaire jpc

Les numéros entre () indiquent la dernière version et les lettres entre (), les initiales des auteurs lorsqu'il y plusieurs versions (LG=Laurent Grand GT=Guy Toublanc)

Programmes	cksum	JPC n°	page	compatibilité HP-48G(X)
ASSCOD	# 105Bh	78	15	oui
→ASC	# 7E07h	78	15	oui
PIL5	# 956Dh	78	16	oui
DEPIL	# 3D18h	78	16	oui
CIRCLE	# DA13h	78	18	voir ci-dessus
RING	# EC29h	78	18	oui
ASC→ (2)	# FC91h	79	11	oui
PEEK	# D7B7h	79	12	oui
XPEEK	# 6EF8h	79	12	non
REV.rpl	# EC5Ch	80	12	oui
PIL	# 2DA2h	80	13	oui
FFACT (LG)	# 5129h	81	9	oui
POKE	# 2647h	81	12	oui
ADDR	# B811h	81	13	oui
EDIV (2)	# EE66h	81	13	oui
FFACTS (GT)	# B80Fh	82	24	oui
FFACTSX (GT)	# 7561h	82	24	oui
TIMED	# 1C05h	82	26	oui
FINDMEM	# ED36h	85	18	oui
RECOVER (2)	# A824h	85	19	non
UNLINE	#	86	18	oui
TOUPPER	# DCBFh	86	22	oui
SYSRCL à PARSE		87	18 19	oui
PROGDEC et ALLDEC		88 88	13 14	non car dépendant de <-RPL-> qui doit être revu
REV (2)	# 9069h	88	15	oui

Il y a donc beaucoup de choses réutilisables. Sauf pour des programmes très pointus il doit être possible, de faire en général une seule version de programmes compatibles S et G, comme pour CIRCLE, DSIERREC et DSIERISO (dans ce JPC). Avec un peu d'astuce cela ne devrait pas diminuer beaucoup la vitesse des programmes polyvalents mais quelle simplicité pour tous et les possesseurs de HP-48S. ne seraient pas abandonnés.

Guy Toublanc (276)

HP-71

G. Toublanc — Triangles de Sierpinski sur HP-71 *18*

Le coin des codes *28*

TRIANGLES DE SIERPINSKI

Les nostalgiques ou les inconditionnels du HP-71, possesseurs de JPCRom (version E), ne doivent pas oublier qu'ils peuvent faire facilement du beau graphisme, soit de façon utilitaire ou pour créer des dessins divertissants. Il aurait été possible d'ailleurs de grossir le nombre des fonctions du module graphique intégré dans JPCRom si le créateur du Lex nous avait laissé les fichiers sources. Or l'ensemble des fichiers qui ont été remis au club est amputé de tout ce qui n'a pas été publié dans JPC concernant JPCRom ce qui rend impossible toute actualisation rationnelle de ce module. C'est une vieille histoire mais ici se présente l'occasion de dire la vérité contrairement à ce qu'ont voulu faire croire les auteurs (et complices) de cette amputation qui par ailleurs ont rendu de très grands services au club ,personne ne peut le nier, mais dont la sortie du club s'est faite avec moins de noblesse en masquant la vérité. Aussi il faut se contenter de ce qui existe de JPCRom, d'ailleurs qui s'est manifesté à ce sujet ?

J'avais donné une suite au graphisme en créant le Lex CERCLEX avec deux mots CIRCLE ET RING (JPC-78) et cela en transposant pour HP-71 des programmes pour HP-48. Je rappelle aussi qu'il est possible de visualiser des fichiers graphiques avec les programmes GLCD71, GLCDCMT et les Lex TRANSLEX et LCDLEX (JPC-71-72), travaux publiés à une époque ou deux numéros ne suffisaient pas pour publier correctement un article et ses programmes.

L'occasion de transposer, un programme pour HP48 en Lex HP-71, se présentant à nouveau, voici du graphisme divertissant: les triangles étranges de Sierpinski. Pour quelques explications reportez vous à l'article correspondant dans la rubrique HP-48. Voici le mode d'emploi du Lex SIERPLEX avec son ordre SIERP :

Celui-ci est fait pour être utilisé avec un bout de programme basic et il ne vérifie que le type GRAPH du fichier qu'on lui donne ainsi que la hauteur maxi.

Il suffit de fournir un nombre entier pour générer un triangle de Sierpinski, le fichier graph sera par exemple de 40960 octets pour n=9, le triangle aura une base de 512 points ce qui est la limite supérieure car pour n=10 cela ferait 1024 points et donc incompatible avec les 640 points d'une ligne de fichier graph.

Pour obtenir le triangle en vidéo inversée il faut fournir un entier négatif.

Donc un petit bout de programme basic qui fait le calcul de la base du triangle en fontion de n, et génère le fichier graph correspondant,puis trace le triangle avec l'ordre SIERP :

```
10 INPUT "n (1-9) ? ";N
20 K = MAX(1,2^ABS(N))
30 GINIT "RASTER",K
40 D = HTD(ADDR$("GRAPHILE")) + 16
50 IF N < 0 THEN D=-D
60 SIERP D
```

Le fichier source (pour les commentaires voir coté HP48) :

```
        LEX     'SIERPLEX'
        ID      #68
        MSG     0
        POLL    0
        ENTRY   SIERe
        CHAR    #D
        KEY     'SIERP'
        TOKEN   254
        ENDTXT

=ARGERR EQU     #0BF19
=EXPEXC EQU     #0F186
=FIXDC  EQU     #05493
=FIXP   EQU     #02A6E
=MFERR  EQU     #09393
=NXTSTM EQU     #08A48
=RNDAHX EQU     #136CB
=eFTYPE EQU     #3F

argerr  GOVLNG =ARGERR
typerr  LC(2)  =eFTYPE
        GOVLNG =MFERR
SIERd   GOVLNG =FIXDC
SIERp   GOVLNG =FIXP

        REL(5) SIERd
        REL(5) SIERp
SIERe   GOSBVL =EXPEXC
        GOSBVL =RNDAHX * conversion réel -> hexa
        GOC    posit   * si nombre positif
        A=-A   A       * pour valeur absolue
posit   D0=A           * @ sur type fichier
        A=DAT0 A
        LCHEX  0E222   * type fichier graph
        ?A#C   A
        GOYES  typerr  * si pas fichier graph
        LC(5)  160  * longueur des lignes en nib.
        D=C    A
        D0=D0+ 16      * @ sur longueur fichier
        A=DAT0 A
        CD0EX
        D0=C
        A=A+C  A   * fin après données graphiques
```

```
        D0=D0+ 5      * @ sur hauteur
        C=DAT0 A
        D0=A
        D0=D0- 1   * position fin ligne n - 1 nib
        A=C    A
        LC(5)  512
        ?A>C   A
        GOYES  argerr * @ si hauteur > 9
        C=A    A  * pour la suite voir HP-48
        A=0    A
        C=C-1  A
        P=   8

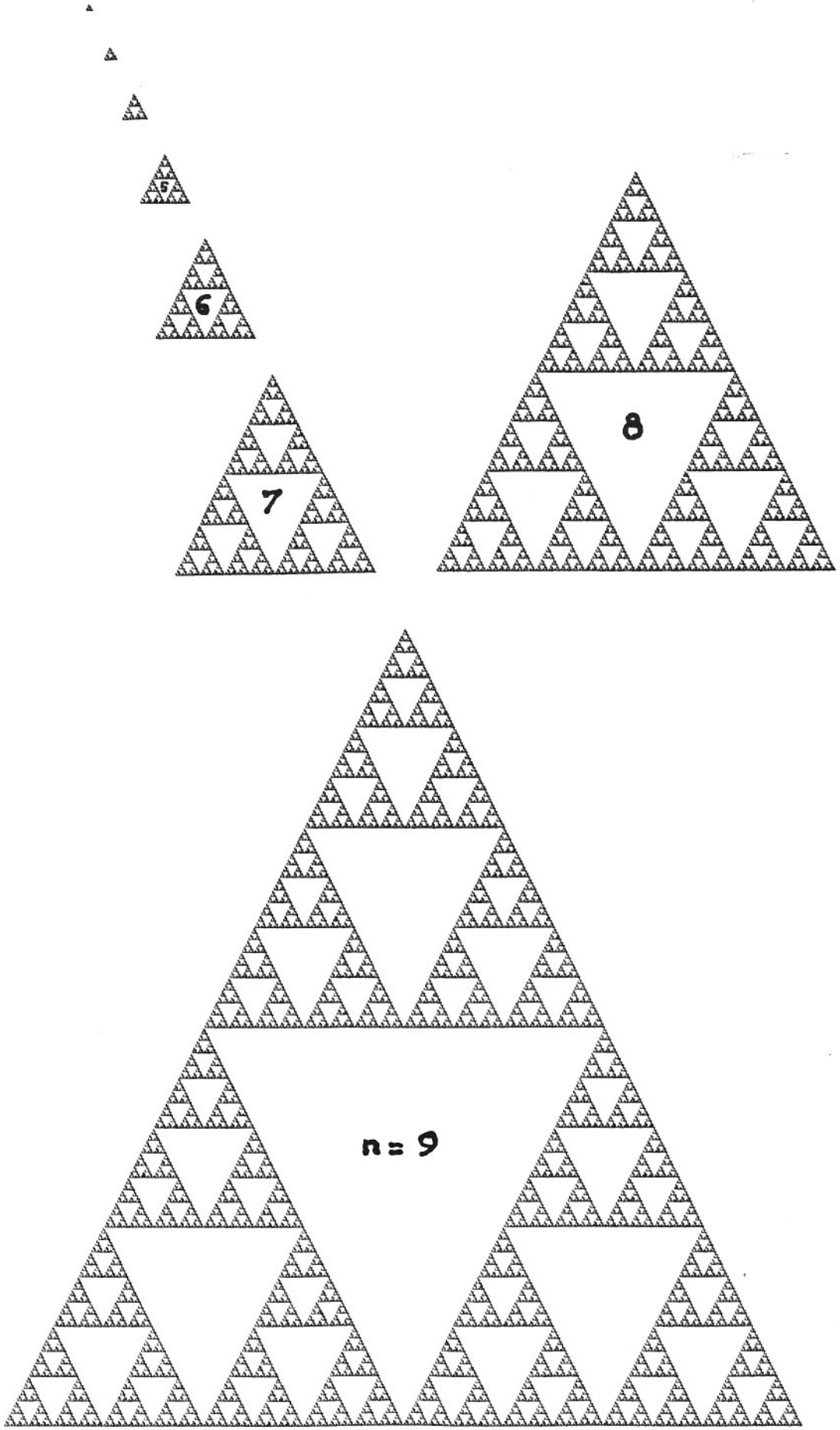
loop    RSTK=C
        B=C    X
        P=P-1
        GONC   idem
        P=     7
        D0=D0+ 1
idem    CD0EX
        C=C-D  a
        D0=C
        D1=C
        D1=D1+ 1
        CLRST
        ?P=    5
        GOYES  p2
        ?P=    4
        GOYES  p2
        ?P=    3
        GOYES  p1
        ?P=    2
        GOYES  p1
        ?P=    1
        GOYES  p0
        ?P=    0
        GOYES  p0
        GONC   p3
loop0   CLRST
        D1=D1+ 1
p3      C=A    X
        C=C&B  X
        ?B#0   S
        GOYES  pos0
        ?C=0   X
        GOYES  nopix0
        GONC   black0
pos0    ?C#0   X
        GOYES  nopix0
black0  ST=1   3
nopix0  B=B-1  X
        GOC    load
p2      C=A    X
        C=C&B  X
        ?B#0   S
        GOYES  pos1
        ?C=0   X
        GOYES  nopix1
        GONC   black1
pos1    ?C#0   X
        GOYES  nopix1
black1  ST=1   2
nopix1  B=B-1  X
        GOC    load
p1      C=A    X
        C=C&B  X
        ?B#0   S
        GOYES  pos2
        ?C=0   X
        GOYES  nopix2
        GONC   black2
pos2    ?C#0   X
        GOYES  nopix2
black2  ST=1   1
nopix2  B=B-1  X
        GOC    load
p0      C=A    X
        C=C&B  X
        ?B#0   S
        GOYES  pos3
        ?C=0   X
        GOYES  nopix3
        GONC   black3
pos3    ?C#0   X
        GOYES  nopix3
black3  ST=1   0
nopix3  B=B-1  X
load    C=ST
        DAT1=C 1
        GOC    nxtline
        GOTO   loop0
nxtline A=A+1  X
        C=RSTK
        C=C-1  X
        GOC    out
        GOTO   loop
out     GOVLNG =NXTSTM

        END
```

Vous trouverez la liste des codes hexa dans *le coin des codes*.

La rubrique HP-71 est toujours ouverte,il suffit de faire un petit effort de participation dans un quelconque langage y compris l'excellent basic du HP-71.

Guy Toublanc (276)


5
6
7
8
n = 9

HP95

J. Belin — Configuration de QEDIT sur HP95 *22*

Le coin des codes *28*

UN EDITEUR POUR LE HP95

De nombreuses personnes regrettent la faible puissance et le manque d'ergonomie de l'éditeur de texte intégré au HP95. Au point de sacrifier un peu de place sur leur disque interne pour installer un autre éditeur.

Les programmes les plus utilisés sur HP95 sont probalement VDE et QEDIT (Ils sont inclus dans notre disquette n°2). Ces deux éditeurs possèdent certaines fonctionnalités communes telles que la possibilité d'ouvrir plusieurs fichiers simultanément ou de diviser l'écran en plusieurs fenêtres.

Cependant, ils possèdent de nombreuses différences qui les rendent complémentaires.

En effet, VDE est plutôt orienté "traitement de texte" et offre des options telles que la justification de paragraphes ou différents styles de caractères. Il possède une option d'installation relative au HP95.

Par contre, QEDIT est un éditeur de texte avancé offrant des possibilités relativement rares telles que la possibilité de reconfigurer totalement le clavier, la manipulation de colonnes de texte ou une aide au tracé de cadres ou de tableaux. Il possède cependant une gestion de paragraphe simplifiée. Une autre caractéristique importante est que QEDIT ne possède pas de limite (si ce n'est la mémoire disponible) dans la taille des fichiers édités.

Suivant le type de travail que vous aurez à éditer (textes ou sources), l'un de ces éditeurs devrait donc vous contenter. Pour ma part, j'ai porté ma préférence sur QEDIT.

Cet article a pour but de vous donner quelques indications vous permettant d'installer au mieux cet éditeur sur le HP95.

Mais, avant de le lire, il est indispensable que vous lisiez (ou au moins de parcouriez !) la documention incluse avec le programme. En effet, cet article n'a pas la prétention de la remplacer, et je ne citerai ici que certaines possibilités ou passerais sous silence certaines parties de l'installation (l'impression par exemple) qui ne posent pas de problème.

A l'origine, le programme est configuré de façon à fonctionner de la même façon que WordStar (voir l'annexe A de la doccumentation pour voir les assignations des commandes sur le clavier). Les utilisateurs d'autres éditeurs tels que celui des languages Borland ne seront donc pas dépaysés. Les autres ont toute latitude pour modifier les fichiers d'installation pour le reconfigurer.

L'exemple présenté aujourd'hui consiste à compléter la configuration initiale de façon à ajouter certaines commandes présentes sur les langages Borland et ajouter certaines commandes spécifiques au HP95. Cependant, le programme présentant un grand nombre d'options de configuration relevant de goûts ou d'habitudes personnelles, je ne donnerai ici qu'un exemple de configuration très incomplet, que vous devrez ensuite étoffer ou modifier selons vos choix personnels.

Avant de commencer, je voudrais signaler que j'ai inclus dans la disquette un exemple de configuration adaptée au HP95, par Jorge Treviño (fichier QED-95.ZIP). Sans être trop critique, je vous conseille d'oublier le fichier de configuration du clavier et de ne garder que le fichier d'aide si vous n'aimez pas la présentation de celui présenté dans cet l'article...

Description du Fichier de définition des touches :

Le fichier QCONFIG.DAT présent sur le fichier ZIP original nous servira de base (Jorge Treviño en a retiré quelques unes importantes de son fichier).

Commençons par ajouter les principales définitions manquantes par rapport aux éditeurs Borland :

```
f1      QuickHelp
f2      SaveFile
f3      EditFile
f5      ZoomWindow
f6      NextWindow
f10     MainMenu

@f3     Exit

^f5     ResizeWindow

^k_i    ShiftRight
^k_u    ShiftLeft
```

Puis ajoutons les accès directs au menu principal :

```
@b      MacroBegin MainMenu 'b'
@e      MacroBegin MainMenu 'e'
@f      MacroBegin MainMenu 'f'
@m      MacroBegin MainMenu 'm'
@o      MacroBegin MainMenu 'o'
@p      MacroBegin MainMenu 'p'
@q      MacroBegin MainMenu 'q'
@s      MacroBegin MainMenu 's'
@w      MacroBegin MainMenu 'w'
```

Afin de marquer un bloc de texte, il est possible, sur PC, d'utiliser les combinaisons de touches Shift+Flèches. Malheureusement, ces combinaisons étant réservées par le HP95, il est nécessaire trouver une autre méthode. Je vous propose donc d'utiliser la commande MarkCharacter et de l'assigner sur une touche proche des flèches, par exemple F9.

```
f9          MarkCharacter
```

Profitez en aussi pour assigner, à coté de cette touche, la commande soeur permettant de créer un bloc multi-colonnes :

```
f8          MarkColumn
```

Pour les mêmes raisons que précédement, il est nécessaire de déplacer les assignations des commandes d'accès au presse papier. Etant assignées aux touches [Shift][INS], [Ctrl][INS] et [Shift][Del], je vous propose de les assigner à la touche placée au dessus de INS sur le HP95, c'est à dire F9. Pour la touche [Shift][Del], nous prendrons [Shift][F10] :

```
#f9         Paste
^f9         Copy
#f10        Cut
```

Dans le même esprit, vous pouvez assigner les commandes de décalement de l'affichage sur des touches proches de celles permettant de décaler la fenêtre virtuelle du HP95 (Alt+Flèches), en conservant l'accès par la touche Alt :

```
@f9         ScreenRight
@f10        ScreenLeft
```

Enfin, puisque nous avons une touche MENU sur le HP95, profitons en :

```
200         MainMenu
```

Le code 200 correspond à cette touche. En effet, il est possible d'assigner des commandes aux touches spéciales du HP95 en spécifiant leur code étendu. En voici les plus importants :

	Normal	Shift	Ctrl	Alt
FILLER	168	164	174	171
COMM	172	n/d	178	175
APPT	176	n/d	182	179
PHONE	180	n/d	186	183
MEMO	184	n/d	190	187
LOTUS	188	n/d	194	191
CALC	192	n/d	198	195
MENU	200	201	202	203

Notez enfin que vous pouvez très bien enregistrer des macros sur ces touches. Cependant, vous pouvez noter à ce sujet qu'il n'est pas possible d'écrire des macros sur des séquences de touches (telles que [Ctrl][K][B]).

Fichier de description de l'écran d'aide

Ce fichier texte (que j'ai appelé QHELP.95) vous permet de définir l'écran qui sera affiché lors de l'appel de la macro QuickHelp (assignée à F1).

Contrairement au fichiers donnés en exemple, j'ai modifié la presentation afin de faire coincider les marquages présents au bas de l'ecran. Les lignes repérées par S, C et A représentent les combinaisons possibles avec les touches Shift, Ctrl et Alt. Ce qui donne pour le fichier QCONFIG.H95 décrit précédement :

```
 QED95  HELP
        SCREEN
                                                                 
                Clo                                 Scr  Sc     A
                Fil                                 Rgt  Lt     
                                                                 
                                Res                 Cpy         C
                                Win                             
                                                                 
                                                    Pas  Cu     S
                                                    te   t      
                                                                 
 He  Sav Lod        Zom Nxt        Mrk  Mrk  Me
 lp  Fil Fil        Win Win        Car  Col  nu
```

La place pour chaque description étant très comptée, vous voyez qu'il faut faire bref ! Je vous conseillerai donc d'etre un peu égoïste et de ne pas hésiter a prendre des abreviations dont vous serez sûr de vous rapeler, même si elles vous sont "personnelles". Vous pouvez même utiliser les caractères semi-graphiques pour créer des petits logos...

Notez que le dessin du cadre et du logo "QED95" a été très grandement facilité par l'utilisation de la macro BoxDraw (que vous devez absolument essayer !).

Configuration du programme

Une fois les fichiers édités, vous devez configurer le programme. Pour cela, il faut modifier les données initialisées dans le programme Q.EXE à l'aide du programme QCONFIG.

Une fois celui-ci lancé, il vous offre un menu permettant de modifier différents types de paramètres (écran, impression...). Il est conseillé de charger en premier (par l'option *KEYS*) les informations stockées dans le fichier de configuration du clavier. En effet, il est tout a fait possible qu'une erreur de syntaxe se soit glissée sur une de ses lignes.

Dans ce cas, le programme s'arrête immédiatement (apres avoir tout de même affiché un message d'erreur) sans sauvegarder les autres modifications...

Une fois ce fichier accepté par le programme, profitez en pour charger l'écran d'aide (par l'option Help Install.

Vous pouvez maintenant modifier les autres options (afin de ne pas surcharger l'article, j'ai abrégé les textes affichés par le programme) :

Advanced options

Pour les questions :

`Test for presence of EGA or VGA adapters ?`
Repondez "N", car nous n'avons pas ce type d'écran sur le HP95. Notez aussi que cela me vous empeche aucunement d'utilser le programme sur un ordinateur équipé de ce type d'écran.

`Test for presence of enhanced keyboard ?`
Répondez "Y". Cela permet d'avoir accès aux touches spéciales du HP95.

`Swap to Expanded memory... ?`
Nous n'avons pas d'EMS, donc la réponse est "N".

`Drive for disk swapping... ?`
A priori il faudrait indiquer l'unité (disque C ou Carte Ram) sur laquelle on est sûr qu'il y ait toujours de la place. Dans le doute, j'ai mis "*".

`Should QEdit change the cursor size ?`
`Do you want a large flashing cursor ?`
La réponse qui pourrait être la plus logique serait "Y" pour la première question et "N" pour la deuxième, qui affiche un curseur de taille diférente suivant le mode de frappe (Insertion ou Overwrite). Malheureusement, si cela marche sur un PC, le HP95 refuse de changer la taille du curseur. D'autre part, il est souvent difficile de localiser le curseur car il est trop fin. Pour corriger ce problème plusieurs solutions existent :
- Si vous n'utilisez pas d'utilitaire permettant de modifier la taille du curseur, il est conseillé de répondre "Y" aux deux questions pour obtenir un gros curseur. En effet, le curseur affiché originellement est très fin et difficilement visible en permanence.
- Si vous utilisez un utilitaire tel que `CTYPE`, permettant de modifier la taille du curseur, il faut répondre "N" aux deux questions car le programme remet un curseur fin et ne restitue pas la bonne taille en fin d'exécution.
- Si vous utilisez `95BUDDY`, et quelques soient les réponses, le curseur sera toujours celui affiché par le driver (normalement trois pixels d'épaisseur), que ce soit en mode Insertion ou en Overwrite.

Note : si vous êtes un utilisateur des éditeurs Borland, et que que vous désirez utiliser le programme sur PC, ne répondez pas "Y" et "N". En effet, `QEDIT` a le petit désavantage d'afficher le curseur de facon inverse à notre habitude : fin en mode "overwrite" et gras en "insertion". D'où de nombreuses fautes de frappe...

`Get num rows on screen from BIOS ?`
Même si le HP95 possède un écran de 16 lignes, son BIOS retourne la valeur 25. Cette option étant donc sans utilité, répondez "N".

`Number of rows on screen... ?`
Ici, il faudrait répondre "16", mais le programme d'instalation n'acceptant que 25 lignes au minimum, nous ne pouvons que mettre cette valeur. Nous corrigerons ce problème un peu plus tard.

`Do you want the return command to split lines... ?`
A moins que vous n'utilisiez un éditeur un peu exotique, vous pouvez répondre "Y".

`Should Cut and Copy use current line if no block... ?`
J'ai répondu "Y", mais cela est un choix personnel.

`Amount of memory to reserve from DOS... ?`
Je n'ai pas testé cette option, et ai répondu "0". N'ayant pas eu de problème, je vous invite à répondre ainsi...

`Code generated by ALT numeric keypad... ?`
Là non plus je n'ai pas fait de test, mais la valeur "0" fonctionne parfaitement.

`Do you want the ESCAPE command to invoke the menus ?`
Ayant assigné la barre des menu sur une autre touche, j'ai répondu "N". De toutes façons, je pense que l'activation des menus par cette touche n'est pas très ergonomique.

`Should the editor restore the starting drive/dir... ?`
J'ai répondu "Y", mais vous êtes libres de faire autrement...

`Should editor change the DOS PROMPT during shell ?`
Repondez "Y", car il est toujours interessant de savoir que vous êtes dans um processus fils de `QEDIT`...

`Should [TabRt]/[TabLt] be allowed to shift a block ?`
Puisqu'on nous le propose, répondons "Y" !

Colors/screen

Comme vous pouvez vous en douter, cette option permet de gérer l'affichage...

`Statusline position... ?`
Je pense qu'elle est plus lisible en haut. J'ai donc dépondu "T".

`Status line fill character... ?`
La réponse par défaut (l'espace) me semble la plus lisible.

`Do you want the display boxed ?`
L'écran du HP95 étant déjà petit... Réponse "N" de rigueur !

`Do you want the original screen restored... ?`
Pourquoi pas ?

`Do you want snow checking ?`
Inutile, cela fonctionne très bien.

`Columns to scroll horizontally for rt/lt commands...`
Cette valeur est utilisée par les commandes `ScreenLeft` et `ScreenRight`. Un valeur de 35 me parait bonne, mais vous pouvez aussi ne décaler que la moitié de l'écran (en spécifiant 20).

Les parametres suivants concernent les couleurs affichées. Le HP95 n'acceptant que deux couleurs (Blanc et Noir), notre choix est limité aux attributs de codes 7 (caractères Noirs sur fond Blanc, pour le HP95 qui est inversé par rapport a un écran vidéo) et 112 (caractères Blancs sur fond noir). Vous pouvez utiliser les valeurs suivantes :

```
Blocked Text:          112
Error msgs:            112
Text Area:             7
Status Line:           112
Cursor Line:           7
Messages:              7
Hi-lited items:        112
EOF line/indicator:    7
Help screen:           112
Box borders:           7
PopUp windows:         112
Menu items:            112
Selected Menu items:   7
```

Notez que le curseur n'est pas apparent lors de l'ouverture de certaines boites de dialogue ou à l'intérieur des blocs sélectionnés. Cela n'est pas la faute du programme, mais celle du HP95 lui même.

`Do you want the del_ch command to join lines... ?`
La réponse "Y" reflete le fonctionnement habituel des éditeurs.

`Should PQuit, Exit & File return to Editfile Prompt ?`
Repondre selon vos convenances personnelles.

`Do you want the editor to start in wordwrap mode ?`
Si vous comptez utiliser le programme pour taper du texte, répondez "Y". Si vous prévoyez de faire autre chose (sources C/assembleur...), répondez "N". De toutes façons, vous pouvez changer facilement de mode en assignant la commande `ToggleWordWrap` à une touche fonction.

`Default right margin for wordwrap... ?`
Une valeur de 38 permet de ne pas créer de scrolling horizontal lorsque on arrive en fin d'écran. Vous pourrez utiliser la commande `SetMargin` pour modifier cette valeur en cours d'édition.

Les autres questions concernent plutôt des choix personnels. A titre indicatif, voici mes réponses :

```
Do you want the editor start in insertmode ? : Y
Do you want the editor start in autoindent mode ? : Y
Do you want the editor use sticky autoindent ? : N
Load wildcarded filespecs from the command line ? : Y
Load wildcarded filespecs from inside the editor ? : Y
Do you want editor start in enter matching mode ? : N
Use CMODE for .c, .cpp... files ? : N
Default number of deleted lines : 30
Remove trailing blanks from edited lines ? : Y
Delete white space after words ? : Y
Terminate files with a Control Z ? : N
Do you want backups of saved files ? : N
Prompt for filename on open window ? : Y
Close window when a file is quit ? : Y
Display End Of File marker ? : N
Insert line blocks ABOVE the cursor line ? : Y
Do you want a beep on warning messages.... : N
Default Find options [IWBG]  : IG
Default Find_Replace options [IWBNG] : IG
Default file extension number 1 : .txt
Default file extension number 2 : .doc
Default file extension number 3 : .c
Default file extension number 4 : .asm
Default file extension number 5 : .h
Default file extension number 6 : .inc
```

Je n'ai pas modifié les autres parties de la configuration (impression et tabulations). Comme les options précédentes, les réponses concernent plutût des convenances personnelles.

Une fois toutes ces options spécifiées, vous pouvez sauver les modifications et sortir du programme.

Cependant, notre travail n'est pas terminé, puisque notre programme fonctionne uniquement en 25 lignes. Occupons nous maintenant du problème.

Conversion du programme en 16 lignes d'affichage

En voyant le processus de configuration, il est aisément compréhensible que toutes les valeurs que nous venont d'entrer seront stockées dans des variables du programme. Il n'y a donc probablement qu'une occurence de chaque valeur dans le code de chaque programme.

Il faut donc pouvoir modifier cette variable qui nous interesse et qui contient la valeur 25 et la remplacer par 16. Cela est tout à fait possible avec un editeur de fichiers binaires tel que celui inclus dans PCTOOLS.

Seul petit problème, lequel des 46000 octets du programme est celui qui nous intéresse ?

Pour le savoir effectuez les manipulations suivantes :
- Copiez le fichier Q.EXE dans Q2.EXE.
- Relancez le programme QCONFIG en spécifiant cette fois-ci Q2.EXE comme programme à configurer.
- Relancez la configurations des options avancées, en ne changeant que la question concernant le nombre de lignes (toute autre valeur que celle initiale convient).
- Sortez du programme de configuration (sans oublier de sauvegarder).
-Enfin, exécutez la commande Dos FC, qui donne la liste des differences entre les fichiers :

```
\dos\fc  q.exe q2.exe
```

Si tout s'est bien passé, une seule différence apparaitra et l'adresse correspondante sera affichée. Si vous utilisez la version diffusée par le club, ce sera A1BE. Notez que cette adresse est conservée quelque soient les modifications de configuration. Vous n'avez donc pas à refaire la manipulation que je viens de décrire si vous utilisez la version 2.15.

Maintenant que nous connaissons l'adresse à modifier, nous pouvons éditer le code du programme pour en modifier le contenu.

Le fait de garder Q.EXE intact a deux avantages : Premièrement, vous gardez ainsi une copie de sécurité, sur laquelle vous pourrez relancer éventuellement la configuration. Deuxiemement, vous pourrez utiliser cette version du programme en plein écran sur votre PC.

La prochaine étape consiste donc à recopier Q.EXE sous un autre nom, QED95.EXE par exemple.

Ensuite deux solutions se présentent à vous, suivant les utilitaires en votre possession et le nombre de reconfigurations que vous aurez à faire :

Solution n°1

Si vous possédez un éditeur de fichiers binaires (tel que celui inclus dans PCTOOLS, lancez le avec QED95.EXE comme paramètre, et en affichant son "dump". Aux environs des neufs dixièmes du code, vous verrez les données suivantes (pour raisons de mise en page, j'ai éliminé certaines colonnes Hexa) :

```
Adresse   --Codes hexa--    Valeur ASCII

  A100    4E ..... 39 8A   NPBSH| TVWXY[ 9
  A110    8B ..... 61 62    1;3@78   klm.ab
  A120    63 ..... 2E 70   cd  .]_ ^"%)*-.p
  A130    71 ..... 00 F0   qref{no  IG
  A140    20 ..... 20 20     QEdit v2.15
  A150    41 ..... 00 43   August, 1991   C
  A160    6F ..... 39 38   opyright (C) 198
  A170    35 ..... 61 72   5-1991 by SemWar
  A180    65 ..... 72 65   e, all rights re
  A190    73 ..... 70 00   served  p p   p
  A1A0    07 ..... 0F 00         p p p
  A1B0    70 ..... 19 00   p . .          v
  A1C0    27 ..... 00 01   ' ^     7 #
  A1D0    01 ..... 01 00
  A1E0    01 ..... 00 00
  A1F0    00 ..... 00 49    (["{)]"}IG    I
```

L'octet qui nous interesse est souligné, en avant dernière position de la 12eme colonne. Cela est verifié par sa valeur : 19h soit 25 en décimal. Il nous suffit donc de la remplacer par 10h (16 en décimal).

N'oubliez pas, le cas échéant de sauvegarder la modification.

Solution n°2

La méthode que je viens de citer fonctionne parfaitement, mais est très vite fastidieuse lorsque on reconfigure le programme un grand nombre de fois. Il est donc préférable de créer un petit programme pour faire le travail à notre place :

```
        DOSSEG
code    segment para 'CODE'
        assume cs:code
        ORG  100h
start:  jmp  strt

POSBYTE EQU  0A1BEh          ; pos. octet à modifier
VALBYTE EQU  16              ; val. octet à modifier
POSCHK  EQU  0A14Ch          ; pos. octet de controle
```

```
VALCHK  EQU  035h            ; valeur octet controle
FNAME   DB   "QED95.EXE",0   ; nom du fichier
msg_ok  DB   "File Converted",13,10,"$"
msg_bad DB   "Bad File",13,10,"$"
buffer  DB   (?)             ; buffer lecture/ecriture

strt:   mov  ax,3d02h        ; ouverture fichier
        mov  dx,offset FNAME
        int  21h
        jc   error           ; si carry -> erreur
        mov  bx,ax
        mov  ax,4200h        ; posit. sur oct. contr.
        xor  cx,cx
        mov  dx,POSCHK
        int  21h
        jc   error
        mov  ah,3Fh          ; lecture octet controle
        mov  cx,1
        mov  dx,offset buffer
        mov  si,dx
        int  21h
        jc   error
        cmp  byte ptr [si],VALCHK
        jne  error           ; mauvaise valeur, erreur
        mov  ax,4200h        ; posit. /octet à modifier
        xor  cx,cx
        mov  dx,POSBYTE
        int  21h
        jc   error
        mov  byte ptr [si],VALBYTE  ; nouvelle valeur
                                    ;  dans buffer
        mov  ah,40h          ; Ecriture nouvelle valeur
        mov  cx,1
        mov  dx,si
        int  21h
        jc   error
        mov  ah,3eh          ; Fermeture du fichier
        int  21h
        jnc  exit
error:  mov  dx,offset msg_bad
        jmp  exit_1
exit:   mov  dx,offset msg_ok
exit_1: mov  ah,09           ; affichage message
        int  21h
        mov  ah,4ch          ; sortie
        int  21h
Code    ends
        END  start           ; That's All Folks !
```

Ce programme place la valeur VALBYTE en position POSBYTE, après avoir vérifié que l'on soit dans le bon programme (et qu'il ne soit pas compressé) en comparant l'octet contenu en position POSCHK à la valeur VALCHK. Ici, il s'agit du "5" de la chaine "QEdit v2.15" affichée dans le dump de la solution n°1.

Vous trouverez les codes Hexa de ce programme dans le *Coin des codes*, en fin de ce journal.

C'est presque fini !

Le programme est maintenant configuré en 16 lignes et est prêt à être transféré sur votre HP95.

Cependant, avant cela, vous pouvez compresser le programme à l'aide de d'un programme tel que PKLITE ou DIET (qui donnent des taux de compressions équivalents). Notez qu'une fois compressé, le programme ne peut plus être modifié par QCONFIG. D'où l'interêt de conserver Q.EXE...

Enfin, pas tout à fait...

Une fois que vous aurez configuré cette premiere version du programme, vous pourrez commencer à vous familiariser avec l'éditeur. Je vous incite à refaire plusieurs fois l'installation en changeant les paramêtres et en essayant les différentes commandes disponibles (elles sont déjà pratiquement toutes assignées sur le clavier). Une fois que vous aurez défini une liste de vos commandes préférés, je vous conseille de les assigner aux touches fonctions, en les regroupant (commandes de blocs, contrôle des fenêtres, accès aux fichiers...) par touches et par type d'accès (Shift, Alt, Ctrl...) de façon a avoir quelque chose de logique et aisément memorisable.

Facile cette dernière étape ? Pas sûr... C'est celle qui m'a posé le plus de problèmes !

Enfin, si vous désirez réduire de quelques octets la taille de l'exécutable, vous pouvez ensuite effacer certaines assignations décrites dans le fichier original (une grande partie des touches accessibles par Alt, par exemple). Attention cependant à ne pas effacer les commandes de base...

Jacques Belin (123)

Note importante pour les utilisateurs de 95BUDDY : Peu avant la mise en page de ce journal, j'ai constaté un problème lorsque ce driver est présent. En effet, si une commande est assignée à la touche [Ctrl][a], cette commande est exécutée lorsque la touche CHAR est pressée. Il convient donc de désactiver la ligne "^a" du fichier de configuration du clavier. N'ayant pas eu le temps de contrôler si d'autres touches posaient des problèmes, je vous conseille d'être prudent.

LE COIN DES CODES

La compilation de certains programmes, tels ceux écrits en assembleur, nécessitent souvent un logiciel que ne possèdent pas tous nos lecteurs. Le *Coin des Codes* permet de résoudre ce problème.

Note importante :

Même si la présentation des listings est identique, le traitement de ceux-ci est différente suivant le programme d'entrée et la machine de destination. Chaque listing est prévu pour être entré sur sa machine de destination. Par exemple, ne tentez pas d'entrer un programme HP48 dans un fichier MS-DOS à l'aide du programme MAKEDOS. Vous obtiendrez un fichier que vous ne pourrez pas transférer dans la HP48.

Programmes HP48 et HP28S

Par rapport aux méthodes habituelles sur HP48, notre méthode effectuant un calcul local du checksum en fin de chaque ligne permet de faciliter la recherche d'erreurs, par rapport à une même recherche dans une chaîne de plusieurs centaines d'octets.

Par mesure de sécurité sauvegardez vos programmes et fichiers, éventuellement verrouillez vos cartes.

Tapez les deux programmes correspondant à votre machine:
- pour HP-48 : ASSCOD48 et INPUT48
- pour HP-28S : ASSCOD28 et INPUT28

(*attention aux parties communes et spécifiques HP-48 ou HP-28S et aux commentaires*)
Ceci avec la plus grande attention car leur mauvais fonctionnement peut entraîner un désordre fatal pour les objets contenus dans votre machine. La commande SPEED (JPC-85 page 12) peut être incluse au début de ASSCOD28. NEWLINE (←) s'obtient sur HP-48 par [flèche bleue] puis [.].

ASSCOD48 996.5 octets
cksum # 5D1Bh

```
« RCLF HEX 64 STWS 1 SF "" 'tmpcod' STO
"nombre d'octets                          @ + NEWLINE
                                          @ NEWLINE
" 17 INPUT48 1 CF STR→ 2 * 16             @ INPUT28
DUP2 / IP 3 ROLLD MOD DUP2 0 > +          @ si HP-28S
3 ROLLD 1 + 4 ROLL # 0h DUP → n r f s o « 1 SWAP
FOR i
 DO "ligne " i 1 - R→B # 1000h + →STR 4 6 SUB + DUP
  "                                       @ NEWLINE
  chaine                                  @ + NEWLINE
  " + "---- ---- ---- ----" n i <
  IF THEN r DUP 4 / IP + SWAP OVER 1 SWAP OVER -
       SUB SWAP 18 + ELSE 38 END 'o' STO +
  "                                       @ NEWLINE
  " + 1 FS?C IF THEN ROT SWAP CLLCD 1 DISP HALT
             ELSE o INPUT48 END DUP       @ INPUT28
  WHILE DUP " " POS DUP                   @ si HP-28S
  REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD
   1 + 25 SUB IF DUP " " == THEN DROP ELSE + END
  END DROP 0 OVER SIZE 1 SWAP
  FOR j OVER j DUP SUB NUM j * + NEXT
  s + DUP # FFFh AND
  "                                       @ NEWLINE
  somme de controle                       @ + NEWLINE
   ---                                    @ + NEWLINE
  #" 6 ROLL SWAP + 34 INPUT48 STR→ ==     @ INPUT28
  IF THEN SWAP DROP 1                     @ si HP-28S
  ELSE DROP2 1000 1 BEEP 1 SF 0 END
 UNTIL END 's' STO
 WHILE DUP " " POS DUP
 REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + 19 SUB +
 END DROP 'tmpcod' DUP RCL ROT + SWAP STO
NEXT f STOF » tmpcod
@ * HP-48 * @@ ********** HP-28S **********
"GROB 8 " @@ # 20204A04F3D02C67h # F80004F02C96040Ch
OVER SIZE @@ # 8DCC05081F804F27h # 313103190F818341h
2 / " "   @@ # 681808AE91B51391h # 45DC061171085168h
+ + SWAP  @@ # F3D3CEAA125E5D8Eh # 2F9004h 28 STWS
+ STR→    @@ #0 OR 64 STWS 1 7 START # 3B82h SYSEVAL
# 4017h   @@ NEXT # 20238h SYSEVAL # 4F3Dh SYSEVAL
SYSEVAL   @@ EVAL
# 56B6h   @@ toutes ces lignes marquées @@ peuvent
SYSEVAL   @@ être remplacées par ASC→ :
DROP      @@ soit pour HP-48 (JPC-79 page 11)
NEWOB     @@ ou pour HP-28S (JPC-85 page 8)
"fin" CLLCD 1 DISP »                      @ HP-28S dito
```

ASSCOD28 1136 octets
cksum # D419h

INPUT48 412.5 octets
cksum # 15C7h

```
« SWAP 1                                  @ partie commune
WHILE OVER CLLCD 1 DISP                   @ HP-48 et HP-28
REPEAT                                    @ puis séquence
 DO UNTIL KEY END                         @ HP-48 ou HP-28
@ ********** HP-48 ********** @@ ****** HP-28S ******
 IF DUP 51 == THEN DROP 0       @@ IF DUP "ENTER" ==
 ELSE IF DUP 55 == THEN DROP    @@ THEN DROP 0
   1 OVER SIZE 1 - SUB          @@ ELSE
  ELSE                          @@  IF DUP "BACK" ==
   CASE DUP 17 < THEN 54 + END  @@  THEN DROP 1 OVER
        DUP 66 < THEN 7 - END   @@    SIZE 1 - SUB
        DUP 76 < THEN 20 - END  @@  ELSE
        DUP 86 < THEN 33 - END  @@ suite voir HP-48
           92 == THEN 48 END    @@
                 46 END CHR     @@
  + DUP SIZE 3 PICK - 2 + 5 MOD NOT @ partie
  1 FC? AND IF THEN " " + END END 1 @ commune
 END END SWAP 60 SUB »                    @ HP-48 et HP-28
```

INPUT28 228 octets
cksum # 9412h

Donc à partir de maintenant pour tout assemblage de chaîne de codes procédez de manière suivante :

1- lancez le programme ASSCOD48 ou ASSCOD28
2- donnez le nombre d'octets (1/2 octet compris) puis validez avec ENTER.
3- tapez chaque ligne de codes, correspondant au numéro de ligne à 3 chiffres, sans les espaces et validez.
4- tapez la somme de contrôle et validez. S'il y a erreur la ligne de codes sera demandée à nouveau après émission d'un BEEP. L'appui sur EDIT fera apparaître la ligne des codes qui pourra être corrigée. Alors relancez avec CONT.
5- après l'affichage de "fin" stockez le programme assemblé dans la variable donnée en tête.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

Programmes HP-71

Mode d'emploi :

1- Lancer le programme : RUN MAKELEX2
2- Entrer la taille du fichier.
3- Entrer les listes de codes puis le checksum. En cas d'erreur corriger la ligne que vous venez d'introduire.
4- Une fois que toutes les lignes sont entrées, éteindre le HP-71, puis le rallumer.

Notez bien que cette MAKELEX2 et les codes Hexa parus dans ce journal sont incompatibles avec MAKELEX, qui a servi dans les JPC antérieurs au numéro 81.

Programmes MS-DOS

Afin d'être utilisé par tous, ce programme est destiné à être écrit en GWBASIC. Il devrait cependant être facile de le convertir pour un autre programme (QBASIC, Turbo BASIC...).

Mode d'emploi :

1- Lancer le programme : GWBASIC MAKEDOS.BAS
2- Entrer le nom du fichier destination.
3- Entrer la taille du fichier.
4- Entrer les listes de codes puis le checksum (en prenant soin d'entrer les codes héxadécimaux en majuscules). En cas d'erreur corriger la ligne, en prenant soin de placer le curseur après le dernier caractère avant de taper sur la touche d'entrée.
5- Une fois que toutes les lignes sont entrées, sortir du GWBASIC en exécutant la commande SYSTEM. Le nouveau programme est immédiatement disponible.

Note : La taille du fichier résultant peut être supérieure d'un octet à ce qui est affiché dans le listing. Cela n'est pas un problème.



MAKELEX2

```
 10 SFLAG -1 @ DIM E$[40],C$[40] @ LC OFF @ PURGE AH @ INPUT "Nb d'octets : ";N
 20 CREATE DATA AH,1,N-4 @ B=HTD(ADDR$("AH")) @ A=B+37 @ N=2*N+37 @ N1=N DIV 16
 30 L9=MOD(N,16)+(MOD(N,16) DIV 5) @ S=0 @ P$="---- ---- ---- ----"
 40 IF MOD(N,16)=0 THEN N1=N1-1 @ L9=LEN(P$)
 50 FOR X=0 TO N1
 60   IF X=N1 THEN P$=P$[1,L9]
 70   C$=P$
 80   M=S @ DISP DTH$(X)[3]; @ INPUT ": ",C$;C$ @ INPUT " sm = ","---";D$
 90   FOR Z=1 TO LEN(C$)
100     IF C$[Z,Z]#" " THEN M=M+(Z-(Z DIV 5))*NUM(C$[Z])
110   NEXT Z
120   IF D$#DTH$(MOD(M,4096))[3] THEN PRINT "Erreur de somme" @ BEEP @ GOTO 80
130   IF X=0 THEN E$=C$
140   IF X=1 THEN E$=E$&C$
150   IF X=2 THEN C$=C$[7]
160   IF X>1 THEN GOSUB 210
170   S=M
180 NEXT X
190 C$=E$ @ A=B @ GOSUB 210 @ CFLAG -1
200 END
210 FOR Z=1 TO LEN(C$)
220   IF C$[Z,Z]#" " THEN POKE DTH$(A),C$[Z,Z] @ A=A+1
230 NEXT Z @ RETURN
```

MAKEDOS.BAS

```
 10 INPUT "Nom du fichier : ",NOM$ : INPUT "Nombre d'octets : ",N : N=N*2
 20 OPEN NOM$ FOR OUTPUT AS #1 : CLOSE #1 : KILL NOM$
 30 OPEN "bin.tmp" FOR OUTPUT AS #1 : S=0 : P$="---- ---- ---- ----"
 40 NLINES=N\16 : LENLAST=(N MOD 16)+((N MOD 16)\5)
 50 IF (N MOD 16)=0 THEN NLINES=NLINES-1 : LENLAST=LEN(P$)
 60 FOR X=0 TO NLINES
 70   IF X=NLINES THEN P$=LEFT$(P$, LENLAST)
 80   C$=P$
 90   X2$="00"+HEX$(X) : PRINT RIGHT$(X2$,3);":";
100   Y=CSRLIN : LOCATE Y,6 : PRINT C$; : LOCATE Y,6 : INPUT "",C$ : Y=Y-1
110   LOCATE Y,27 : PRINT " sm = ---" : LOCATE Y,33 : INPUT "",D$
120   M=S
130   FOR Z=1 TO LEN(C$)
140     IF MID$(C$,Z,1)<>" " THEN M=(M+((Z-(Z\5))*ASC(MID$(C$,Z,1)))) MOD 4096
150   NEXT Z
160   D2$="00"+HEX$(M) : D2$=RIGHT$(D2$,3)
170   IF D2$<>D$ THEN PRINT "Erreur de somme" : BEEP : GOTO 90
180   FOR Z=1 TO LEN(C$) STEP 2
190     IF MID$(C$,Z,1)=" " THEN Z=Z-1 : GOTO 230
200     CH=ASC(MID$(C$,Z,1))-48 : IF CH>9 THEN CH=CH-7
210     CL=ASC(MID$(C$,Z+1,1))-48 : IF CL>9 THEN CL=CL-7
220     PRINT#1,CHR$((16*CH)+CL);
230   NEXT Z
240   S=M
250 NEXT X
260 CLOSE #1 : NAME "bin.tmp" AS NOM$ : END
```

```
ADDR28S                    (HP28)
# 6625h                32 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C0F2 C01D   E80
001: 3F40 5E02 069C 20D1   BBB
002: 0001 4317 4E71 4713   7C8
003: 7179 8D5B 4400 9F20   4F0


REV28S                     (HP28)
# 417Ch                68 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C07   E08
001: C170 76C2 0D3F 40A4   BF2
002: 0202 C040 69C2 0150   7CB
003: 008F 72F4 08F1 8050   4A5
004: D814 3131 1791 3713   014
005: 5C9C 9134 819F 1541   CEA
006: 1811 4F14 A149 14C1   9EE
007: 71CD 5BE8 D5E5 2109   819
008: F200 9F20             FB2


PEEK28S                    (HP28)
# C293h             102.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 04C3 C069 C209   D85
001: B000 1471 3706 1433   8B8
002: 407A 208A 2908 DB15   68A
003: C017 9143 102D A071   308
004: 3517 4E71 4713 7061   EDD
005: 478A 66D1 7914 710B   BFE
006: 0713 511A C681 8F29   9DE
007: 10C8 FAB9 E118 411C   805
008: 818F A414 4164 11AD   58D
009: 5113 1313 1935 D1AE   37A
00A: A15B 09EA 8081 8606   07D
00B: 1481 6117 0CD5 2E11   DBA
00C: 08DD F7F1 09F2 0      204


POKE28S                    (HP28)
# 2E8Ch              91.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 04C3 C069 C20E   E45
001: 9000 8F18 0501 4713   9D4
002: 7061 4334 E4A2 08A2   713
003: 908D B15C 0174 1438   398
004: 18F8 4819 F010 0174   FA6
005: 0713 7061 7414 7137   B40
006: 1433 407A 208A 65C1   8A2
007: 7914 7134 0713 5118   3FF
008: D531 935A 114B 9EA8   2C5
009: 0818 6861 5801 6017   E4E
00A: 1CD5 5E8D 5E52 1A11   C1C
00B: 2009 F20              1FC


DSIERREC                   (HP48)
# F56Bh              229 octets

     0123 4567 89AB CDEF   sm

000: D9D2 02BA 81D9 F819   EC3
001: FF30 D9D2 0881 3083   B47
002: 7A28 DA16 0CA3 018A   94A
003: 3032 2300 09A2 68FA   73F
004: 2ED2 A232 2300 7AA2   44D
005: AEC8 19FF 306C 4269   261
006: BC26 F851 149F 21CF   141
007: 1462 50F4 3223 0CCD   F87
008: 2093 1008 F20A 16AC   DAB
009: 0450 B448 F146 6010   9B1
00A: 08FB 9760 1B97 F101   6FC
00B: 4234 B550 78A2 9034   37C
00C: AD60 8134 1421 3016   E93
00D: 4142 1361 34C2 1351   A34
00E: C081 8F8E 1641 468F   84C
00F: 42F3 0D7D 0118 CE06   65D
010: AB51 37EB 1351 3408   2CA
011: 160A B60E 3594 AA09   0FC
012: 3AD0 5709 3E50 850A   E4A
013: 3D45 6AB6 0E35 94AA   D32
014: 093A D057 093E 5085   A50
015: 1A3D 434A B60E 3594   824
016: AA09 3AD0 5709 3E50   573
017: 852A 3D41 2AB6 0E35   367
018: 94AA 093A D057 093E   17C
019: 5085 3A3D 0915 C046   E8B
01A: 06C6 FB34 07A3 E460   C3F
01B: 6D4F 8D34 150F 2293   956
01C: B213 0B21 30          468


DSIERISO                   (HP48)
# AA4Ah              255 octets

     0123 4567 89AB CDEF   sm

000: D9D2 02BA 81D9 F819   EC3
001: FF30 D9D2 0881 3083   B47
002: 7A28 DA16 0CA3 018A   94A
003: 3032 2300 09A2 68FA   73F
004: 2ED2 A232 2300 7AA2   44D
005: AEC8 19FF 306C 4269   261
006: BC26 F851 149F 21CF   141
007: 1462 50F4 3223 0CCD   F87
008: 20D6 100A C08F 20A1   D1E
009: 6450 B448 F146 6010   92A
00A: 08FB 9760 1B97 F101   675
00B: 4234 B550 78A2 9034   2F5
00C: AD60 8134 1421 3016   E0C
00D: 4142 1361 34C2 1351   9AD
00E: C081 8F8E 1641 468F   7C5
00F: 42F3 0D7D 0118 CE28   614
010: 06AB 50D5 7027 1701   210
011: 37EB 1351 3416 0088   DF8
012: 9554 8940 4893 D589   B71
013: 2858 9157 8900 7570   76C
014: 0816 0AB6 0E35 94AA   616
015: 093A D057 093E 5085   334
016: 0A3D 456A B60E 3594   121
017: AA09 3AD0 5709 3E50   E70
018: 851A 3D43 4AB6 0E35   C83
019: 94AA 093A D057 093E   A98
01A: 5085 2A3D 412A B60E   90C
01B: 3594 AA09 3AD0 5709   69B
01C: 3E50 853A 3D09 15C0   3F9
01D: 4606 C6FB 3407 A3E4   267
01E: 606D 1F20 8D34 150F   FFA
01F: 2293 B213 0B21 30     559


GSIERREC                   (HP48)
# E05Dh              192 octets

     0123 4567 89AB CDEF   sm

000: D9D2 02BA 81D9 F819   EC3
001: FF30 D9D2 0881 3083   B47
002: 7A28 DA16 0CA3 018A   94A
003: 3032 2300 09A2 68FA   73F
004: 2ED2 A232 2300 7AA2   44D
005: AEC8 19FF 306C 4263   201
006: 2230 CCD2 0D01 008F   FA9
007: 20A1 6AC2 450B 468F   E09
008: B976 0147 1351 7414   980
009: 7108 1098 F3B5 1118   63D
00A: F183 1361 2BD7 1C0D   48F
00B: 0118 CE06 AB51 37EB   3B5
00C: 1341 3508 160A B60E   158
00D: 3594 AA09 3AD0 5709   EE7
00E: 3E50 850A 3D45 6AB6   D66
00F: 0E35 94AA 093A D057   B77
010: 093E 5085 1A3D 434A   94D
011: B60E 3594 AA09 3AD0   7BB
012: 5709 3E50 852A 3D41   509
013: 2AB6 0E35 94AA 093A   371
014: D057 093E 5085 3A3D   166
015: 0915 C046 06C6 FB34   F96
016: 07A3 E460 6D4F 1138   D08
017: D276 30B2 130B 2130   8BE
```

```
GSIERISO                    (HP48)
# 897Ch                  218 octets

     0123 4567 89AB CDEF   sm

000: D9D2 02BA 81D9 F819   EC3
001: FF30 D9D2 0881 3083   B47
002: 7A28 DA16 0CA3 018A   94A
003: 3032 2300 09A2 68FA   73F
004: 2ED2 A232 2300 7AA2   44D
005: AEC8 19FF 306C 4263   201
006: 2230 CCD2 0141 008F   F17
007: 20A1 6AC2 450B 468F   D77
008: B976 0147 1351 7414   8EE
009: 7108 1098 F3B5 1118   5AB
00A: F183 1361 2BD7 1C0D   3FD
00B: 0118 CE28 06AB 50D5   259
00C: 7027 1701 37EB 1351   EEB
00D: 3416 0088 9554 8940   B15
00E: 4893 D589 2858 9157   809
00F: 8900 7570 0816 0AB6   514
010: 0E35 94AA 093A D057   325
011: 093E 5085 0A3D 456A   12C
012: B60E 3594 AA09 3AD0   F9A
013: 5709 3E50 851A 3D43   CFD
014: 4AB6 0E35 94AA 093A   B67
015: D057 093E 5085 2A3D   94F
016: 412A B60E 3594 AA09   780
017: 3AD0 5709 3E50 853A   4FB
018: 3D09 15C0 4606 C6FB   3CF
019: 3407 A3E4 606D 1F20   157
01A: 1138 D276 30B2 130B   E24
01B: 2130                  011


CIRCLE                      (HP48)
# 8FB7h                  378 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0FDE 813A 040D   E59
001: 9D20 CCD2 01D2 008F   C53
002: B976 0201 3713 5108   797
003: 1471 3517 4137 1351   2DA
004: 0B75 1206 1B97 F101   F98
005: 4234 B550 78A2 9034   C18
006: AD60 8134 0714 2132   73C
007: 16EA D014 281C C212   46A
008: B135 17F7 0D11 8414   0EA
009: 28AC 6061 B081 CEE1   FB9
00A: 2813 5174 1471 3517   B40
00B: 476A 1440 FA10 A164   861
00C: 142C C81C 81CE 4E48   79F
00D: 1CD6 D718 4D21 0911   41A
00E: 2C43 4300 00E2 0611   F72
00F: 9604 0758 007D A243   C46
010: 0FB0 E119 5E01 2ACE   B66
011: D512 AE9C 6C6C A203   9FC
012: 4600 00C2 0611 9E61   679
013: 0911 2E24 DB11 98A6   48B
014: 6078 3007 208D 3415   0A5
015: 0119 FA10 9623 0119   C9D
016: FA10 9693 0119 FA12   9BC
017: AFA1 0A70 5064 2084   5B8
018: 0724 04BC 7920 11AF   3B3
019: A10A 7230 44C7 6101   FAB
01A: 1912 AFA1 0987 0008   C27
01B: 505E C129 FA12 9164   97A
01C: 5430 7031 1A20 110E   584
01D: A146 8BE0 0C4D BD51   499
01E: 6913 6134 1845 40C2   0CB
01F: CD5B F135 0611 111B   D2B
020: CA14 6184 8BE8 B303   AF6
021: 0E02 A858 1C81 C137   85D
022: C213 5301 550A 06A0   4B4
023: D59F 15B0 94FE 00E0   308
024: E159 0070 1B8C 0E0E   172
025: B8C1 5100 703D 015F   E79
026: FAC7 ABA3 2005 9BEC   E00
027: 0324 009B 270A F202   AE9
028: AD2A B617 D133 EA13   8EA
029: 180D 0D21 5712 3D1D   67D
02A: 71F8 1C00 1337 9201   21C
02B: FC90 0013 37B1 01F6   EFD
02C: 0000 1337 D00D BE99   DAC
02D: 4A00 FA03 A8B5 40C8   BFA
02E: A0E5 AF0D 01B2 130B   996
02F: 2130                  B83


SIERPILX                    (HP-71)
                       212 octets

     0123 4567 89AB CDEF   sm

000: 3594 5425 0594 C485   CDB
001: 802E 0000 0000 0000   6BD
002: CA10 086E FEF0 0000   401
003: F710 0000 0000 0000   DA8
004: 0040 00D9 3594 5425   9CF
005: 05EF 1FF8 D91F B031   89B
006: F38D 3939 08D3 9450   589
007: 8DE6 A202 FFFF 4FFF   7AF
008: F8F6 81F0 8FBC 6314   60C
009: 40F8 1301 4234 222E   23E
00A: 08A6 AB34 0A00 0D71   F4A
00B: 6F14 2136 134C A164   C20
00C: 1461 3018 0DA3 4002   810
00D: 008B 628D 6D0C E280   679
00E: 6AB5 0D57 0271 6013   238
00F: 6EB1 3413 5170 0889   E64
010: 5548 9404 893D 5892   B8F
011: 8589 1578 9007 5700   739
012: 8170 AB60 E359 49A0   4F2
013: 93AD 0570 93E5 0853   1D0
014: A3D4 56AB 60E3 5949   FC3
015: A093 AD05 7093 E508   D0D
016: 52A3 D434 AB60 E359   B2B
017: 49A0 93AD 0570 93E5   8F3
018: 0851 A3D4 12AB 60E3   712
019: 5949 A093 AD05 7093   434
01A: E508 50A3 D091 5D04   157
01B: 606C 6FB3 407A 3E46   F99
01C: 06D1 F8D8 4A80 F      4A3


PATQED95.COM                (MS-DOS)
                       132 octets

     0123 4567 89AB CDEF   sm

000: EB27 5145 4439 352E   CF2
001: 4558 4500 4669 6C65   9F1
002: 2043 6F6E 7665 7274   718
003: 6564 0D0A 2442 6164   33E
004: 2046 696C 650D 0A24   09E
005: 00B8 023D BA02 01CD   F04
006: 2172 418B D8B8 0042   C0D
007: 33C9 BA4C A1CD 2172   A7C
008: 33B4 3FB9 0100 BA28   824
009: 018B F2CD 2172 2580   4FE
00A: 3C35 7520 B800 4233   0B2
00B: C9BA BEA1 CD21 7214   E83
00C: C604 10B4 40B9 0100   A39
00D: 8BD6 CD21 7206 B43E   860
00E: CD21 7305 BA1D 01EB   732
00F: 03BA 0C01 B409 CD21   51F
010: B44C CD21             D3E
```

Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

PPC Paris est le représentant Français de **HEX** (**H**andhelds **E**uropean Clubs EXchange), la fédération des principaux clubs Européens d'utilisateurs de calculateurs HP.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Les informations et programmes parus dans ce journal sont publiées "Tels quels" et ne peuvent en aucun cas engager la responsabilité de Hewlett-Packard ou de PPC Paris. Hewlett-Packard se réserve le droit de ne pas répondre aux questions concernant le sujet de certains articles.

Les programmes publiés peuvent être utilisés librement. Cependant, ils ne peuvent être vendus ou fournis dans un ensemble commercialisé, sous quelque forme que ce soit, sans l'accord écrit de l'auteur ou de PPC Paris.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.

Imprimé par Paris Copie, 4 rue Linné, 75005 Paris