JPC

JUILLET-AOUT 1992

NUMERO 78

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau	Editorial	*1*
J. Belin	Bruits et rumeurs	*2*
J. Belin	Un disque dur pour les portables	*2*
	Courrier du coeur	*2*

HP-48

A. Aoufi	Les cartes du HP-48 (Acte II)	*4*
A. Aoufi	Banc d'essai de la *SmartRom*	*4*
C. Vaillant	Jeu : Reflexion	*6*
J.F. Garnier	Calendrier perpétuel	*10*
M. Weil	Probabilités	*12*
R. Le Bris	Correctif à DESASS	*14*
G. Toublanc	Assembleur : Les outils du début	*14*
G. Toublanc	Tracé de cercles	*17*
G. Toublanc	Présentation du Coin des Codes HP-48	*21*

HP-71

G. Toublanc	Tracé de cercles	*24*

HP-95

J. Belin	Les logiciels du HP-95	*28*
J. Belin	Le dévelopement de logiciels sur HP-95	*29*
J. Belin	Voyage au centre de la mémoire, le retour	*34*

MS-DOS

J. Belin	Utilitaires de conversion pour ThinkJet	*36*
J. Belin	Une bogue dans LIFCP	*39*

EDITORIAL

Tout d'abord, excusez nous pour le léger retard dans la parution de ce nouveau *JPC*. Retard dû à certains problèmes techniques qui auraient nuit à la qualité d'impression du journal.

Une bonne nouvelle pour les utilisateurs de HP-48, nous débutons ce mois-ci la distribution de ce que nous appellons les *IDS HP-48*. Bien que cela ne soit pas comparable avec ce qui a été fait auparavant pour le HP-71, c'est l'outil indispensable à toutes les personnes désirant mieux connaitre le fonctionnement de leur machine ou programmer en assembleur. Voir les details dans les pages centrales.

Une autre bonne nouvelle provient d'un de nos anciens adhérents italiens, Flavio Casetta. Il a créé depuis peu une société, *Smart Technology*, spécialisée dans les cartes ou périphériques pour calculateurs HP. Ses premiers produits sont la *SmartRom*, que nous vous présentons dans ce *JPC*, et un convertisseur série-parallèle (fourni avec des drivers pour HP-48) que nous vous présenterons dans le prochain numéro. Nous ne pouvons pas actuellement donner leur prix français, car il recherche encore un distributeur pour notre pays, mais sachez qu'il est prêt à faire d'importantes réductions aux membres de PPC Paris...

Autre chose, nous vous avons déjà parlé de la fête qu'organisera à Londres notre confrère anglais HPCC, pour ses dix ans, les 19 et 20 septembre. Les informations que nous avons indiquent qu'elle aura un retentissement exceptionnel, avec la présence, par exemple, de plusieurs membres américains (dont peut-être Bill Wickes). Nous serons bien sûr présents, et étudions actuellement différents moyens pour organiser le déplacement d'un petit groupe. Si vous voulez vous joindre à nous, contactez nous *d'urgence* pour obtenir les informations pratiques et le formulaire d'inscription qu'ils nous ont transmis.

Ceci nous ammène bien sur à l'autre réunion exceptionnelle, celle que nous allons organiser, pour fêter nous aussi nos dix ans. Etalée sur deux jours, elle sera en partie différente de ce qui a déjà été fait, (afin de ne pas faire double emploi avec la réunion de Londres). Il y aura donc très probablement un tournoi de machines. Le jeu sur lequel il sera basé n'est pas encore totalement fixé, mais ce sera un jeu inédit ou en tous cas pas encore porté sur ordinateur. En plus de ce tournoi, il y aura des rencontres, des présentations de matériels et bien sûr notre Assemblée Générale. La fête et le tournoi sont bien sûr ouverts à tous les utilisateurs de calculateurs HP, même si ils ne sont pas adhérents à PPC Paris. Elle aura lieu en Janvier, probablement le deuxième week-end. Les personnes résidant en province ou à l'étranger peuvent nous contacter dès aujourd'hui afin de discuter de la date définitive. D'autre part nous avons déjà desoin d'avoir une première estimation du nombre de personnnes présentes, afin de pouvoir réserver la salle la plus appropriée à l'affluence que nous espérons nombreuse.

Parlons un peu du journal maintenant. Il semble que notre appel du mois dernier concernant le nombre d'articles a porté ses fruits, comme l'indique la taille de ce *JPC*. Nous remercions vivement les adhérents qui ont envoyé des articles, et invitons les autres à faire de même.

En attendant, nous vous souhaitons de bonnes vacances !

Le bureau.

CRIS ET RUGISSEMENTS

Après le *Jaguar*, Hewlett-Packard semble avoir adopté les noms de félins pour le nom de ses ordinateurs de poche tournant sous MS-DOS. En effet, il semblerait que certaines personnes auraient entendu le feulement d'un *Cougar* du côté de Corvallis. Il serait très proche du HP-95, mais serait équipé d'un écran 80×25, à la norme CGA et sortirait de sa tanière vers Avril 1993. Ensuite, un *Lion* vous mangera peut-être dans la main, ceci sous toute reserve car on a entendu le terme *portable*, et non *palmtop*, à son sujet. Suivront-ils le calendrier chinois en sortant un *Tigre* en 1998 ?

Jacques Belin (123)

UN DROLE D'OISEAU

Le mois dernier, dans l'éditorial, nous vous avons annoncé que HP travaillait sur le plus petit disque dur au monde, au format 1,3 pouces. Depuis, de nouvelles informations sur ce produit ont été rendues officielles.

Répondant au doux nom de *KittyHawk*, sa conception a été confiée à plusieurs sociétés, dont Citizen qui a été bien entendu chargée de la partie mécanique, digne de la meilleure horlogerie.

Ses dimensions sont : 10,5×36.5×50.8 mm, soit une surface inférieure à la moitié d'une carte de crédit, ou autrement dit, quasiment la taille d'un *Post it* ref. 653 neuf, pour ceux qui lisent leur *JPC* au bureau ! Son poids est de 32 grammes.

Il est alimenté avec une tension de 5 Volts, contre 12 pour les disques durs classiques. Ce qui devrait simplifier son implantation dans les ordinateurs de poche, alimentés par des batteries de faible capacité.

Il existe en 2 versions : 14 et 20 Mo (formatés), avec un temps d'accès de 18 ms et un taux de transfert de 900 Ko/s.

Il a été conçu pour résister (en fonctionnement) à des chocs correspondant à une accélération de 100 G, soit une chute de 1 mètre sur une surface dure. Ceci grace à un circuit de protection actif (du même type que ceux déclenchant l'*Air Bag* sur certaines voitures), mettant le disque dans un état spécifique dès la détection d'un choc. Hewlett Packard garantit un MTBF de 300000 heures, le double de certains disques durs de grande marque.

Il sera distribué avec deux type d'interfaces : soit en IDE, qu'on trouve sur la plupart des compatibles actuels, soit au standard PCMCIA 2.0 qui est le futur standard des cartes mémoires ou d'interfaces pour les portables.

Il sera disponible à la fin de cette année, au prix de 250$ (pour les OEM).

En ce qui nous concerne, nous l'attendons bien sûr dans les prochaines générations d'ordinateur de poche. Malheureusement, si sa taille lui permettrait d'être inclus dans un HP-95, à la place de la carte mémoire, il possède le défaut de consomer 1,7 Watts (en écriture). Ce qui est un record, mais toujours bien plus que les quelques milliwats de ces mêmes cartes. Il faudra donc attendre encore un peu, à moins que vous acceptiez de transporter des batteries plus lourdes que la machine !

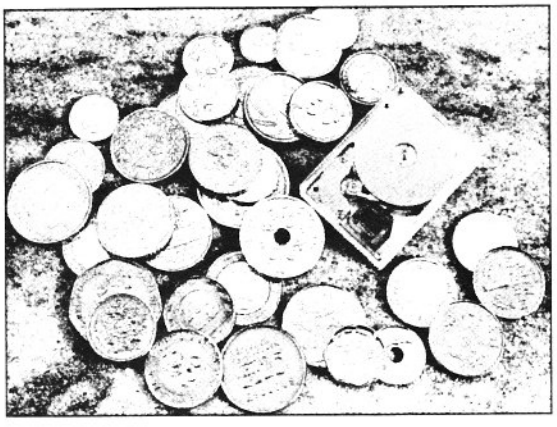
Jacques Belin (123)

COURRIER DU COEUR

Jean REIBEL
9, sq Victor Flemming
92350 LE PLESSIS ROBINSON
Tel : (1) 46 31 46 11

Vend :

Module Ram 64ko (Front Port), module Ram 64Ko (Port lecteur de cartes), module Eprom 64Ko (Front Port) : 600 F chacun ou 1500 F les 3.



HP-48

A. Aoufi	Les cartes du HP-48 (Acte II)	*4*
A. Aoufi	Banc d'essai de la *SmartRom*	*4*
C. Vaillant	Jeu : Reflexion	*6*
J.F. Garnier	Calendrier perpétuel	*10*
M. Weil	Probabilités	*12*
R. Le Bris	Correctif à DESASS	*14*
G. Toublanc	Assembleur : Les outils du début	*14*
G. Toublanc	Tracé de cercles	*17*
G. Toublanc	Présentation du Coin des Codes HP-48	*21*

LES CARTES DU HP-48SX

Nous continuons ce mois-ci le tour d'horizon thématique des différentes cartes d'application pour la HP-48SX.

Electronique : Solid State Electronics Pac de *Sparcom* 100$, dispose d'une bibliothèque de 200 équations pour étudier entre autres transistors,diodes. circuits, de visualiser la réponse de circuits linéaires. il s'agit de l'adaptation d'un logiciel de simulation développé à l'université de Californie.

Finance : Financial Functions Library de *TDS* 80$, permet d'effectuer tous les calculs financiers classiques.

Jeux : Casino 48 Application Card de *Preston Brown Labs* 57$, restitue tous les types de jeux disponibles dans un casino avec un graphisme très détaillé. Entre le Blackjack, la Roulette et le Poker le choix sera difficile.

Utilitaires : SmartRom de *Smart Technology* 170$, propose un ensemble d'outils tels que la manipulation de matrices symboliques. Voir l'étude dans ce numéro. **HP-48 MLDL Rom Card** de *Jan Brittenson* 90$, possède un ensemble de commandes utiles pour le développeur professionnel en langage machine sur 48 que pour l'étudiant. Dispose d'un désassembleur et d'un débuggeur.

Asdin AOUFI (562)

BANC D'ESSAI DE LA CARTE SMARTROM

J'ai pu tester la carte SmartROM, de la société Italienne Smart Technology qui nous l'a aimablement prêtée. Cette carte se compose d'un ensemble d'utilitaires pour la manipulation de matrices symboliques, la gestion de la pile et des divers objets, etc...

Une boite cartonnée bleue contient le manuel d'utilisation, la carte d'application et une disquette au format 3 pouces 1/2. Curieusement il n'y a pas de disquette au format 5 pouces 1/4 !. J'espère que les acheteurs de la carte disposent d'un PC lisant les 2 types de disquettes contrairement au mien. Cela dit un rapide coup d'oeil à la documentation indique que celle-ci est de bonne facture et n'a rien à envier à celles de Hewlett-Packard.

Avant d'étudier les caractéristiques de la carte, précisons qu'il existe un manuel de référence complé-mentaire sur les fonctions cachées représentant 40% du code implémenté en ROM. Nous n'en disposions pas au moment du test. Nous avons aussi regardé les différentes fonctions supplémentaires contenues sur la disquette et qui incluent entre autre la transformée de fourier ou l'analyse du langage RPL. La plupart fonctionnent indépendamment de la carte.

Cette disquette propose 23 fonctions dont certaines très intéressantes. Citons dans le désordre : le dessin de barres avec motifs ombrés, le calendrier mensuel, le remplacement d'une donnée par une autre dans tous les objets du répertoire, le changement du numéro d'identification d'une librairie, la transformée de Fourier d'un tableau de complexes, la conversion chaine -> objet , minuscules -> majuscules et réciproquement, la création d'un environnement pour la saisie de matrices formelles et leur affichage, le dessin de camembert.

Voyons maintenant les fonctions permettant une analyse fine d'un programme écrit en RPL. On peut synthétiser un programme en langage RPL interne en convertissant une liste d'entiers binaires en adresse et inversement, réduire la taille d'un code en compactant tous les SYSEVAL rendant illisible le code ainsi obtenu, supprimer les délimiteurs d'un programme, déplacer un répertoire, trier son contenu. L'ensemble de ces outils est très utile pour disséquer les programmes et manipuler facilement les matrices formelles.

L'installation de la carte ne pose pas de problème. Il est même possible d'installer automatiquement la dite carte à l'aide de la commande INSTALL présente sur la disquette d'accompagnement. Nous allons décrire les diverses fonctions présentes et ce de manière thématique. Le manuel (en Anglais) est très bien fait en reprenant le style du *HP-48 Programmer's Reference Manual*. Il y est décrit par le détail les 93 fonctions de la carte.

Notons un point commun de la plupart des fonctions : elles opérent soit sur un nombre de niveaux déterminé soit sur un nombre quelconque de niveaux explicité par l'utilisateur.

Pile : 17 fonctions.
Duplication, effacement, échange sur les N niveaux de la pile, rotation sur 3 niveaux et sur X, Y, Z, T, comme sur la HP-41C. Existence d'un Meta-Object *MARK* (en fait un nom global défini dans l'appendice B par son adresse hexadécimale) pour les

manipulations de la pile. Très pratique pour les rotations. On élargit ainsi l'éventail des fonctions de manipulations de la pile.

Vérification d'Arguments : 3 fonctions.
Test de présence d'un type d'objet donné dans une liste donnée et renvoit d'un indicateur entier 0 ou 1.

Manipulation de listes : 15 fonctions.
Application d'opérateurs monadiques et dyadiques sur des listes d'objets. Recherche et substitution d'un objet spécifié dans une liste ou un programme donné. Décomposition de listes en ses composants.

Manipulation de chaines : 14 fonctions.
Recherche chaîne, présence d'une chaîne, substitution d'une chaine dans une chaine donnée, inversion, position d'une chaine.

Manipulation de Méta-Objets : 14 fonctions.
Copie, effacement, modification, déplacement de l'adresse en RAM,inversion, duplication, accés, édition, tri dans l'ordre alphabétique et inverse, le tout sur n niveaux de la pile.

Gestion des types d'objets : 13 fonctions.
Conversion entier en binaire, objet en chaine; adresse ROM d'un objet donné ou son adresse en RAM si créé par utilisateur, extraction du bit de poids fort pour entiers binaires. Ventile les instructions d'un programme à chaque niveau de la pile, reconstruit un tel programme: Très original. On peut ainsi travailler dans la pile sur chaque élément du programme.

Manipulation de matrices symboliques (MS) : 22 fonctions.
Création de matrices formelles sous formes de listes imbriquées, édition avec le *Symbolic Matrix Writer*, assemblage d'une matrice n*m à partir des n*m expressions algébriques constitutives, décomposition de la liste en ses composantes; addition, soustraction, multiplication, entre deux MS ou une MS et une expression algébrique, inversion formelle, déterminant formel, transposée; extraction de sous-matrices, de lignes, de colonnes, de coefficients. Curieusement on ne peut semble-t-il ajouter simplement une colonne ou une ligne à une matrice donnée ! On peut tester la cohérence des dimensions, sa forme (carrée ?, symbolique ?). Cet ensemble d'outils est manifestement le plus utile et le plus puissant que j'ai trouvé sur cette carte pour mon usage personnel.

Edition de programme : 3 fonctions.
Recherche d'une instruction, substitution d'une instruction par une série d'autres instructions , décomposition et assemblage d'un programme donné en utilisant la pile de manière intensive. Il est patent que l'édition d'un gros programme soit "très lente", d'où l'astuce de passer par la pile et d'y naviguer.

I/O : 1 fonction.
Attente indéfinie par KEYWAIT.

Utilitaires : 2 fonctions.
Gére CST pour une utilisation optimale des fonctions de la SmartROM dans les 6 domaines principaux appelés: *STACK* (pour la pile), *$&L*, *META* (pour les méta-objets), *SYMB* (pour les matrices symboliques), *TYPES* (pour gérer les types d'objets manipulés), *MISC* (pour les fonctions diverses). On peut ainsi via un menu arborescent mieux manipuler les diverses fonctions de la carte.

Version ROM : 2 fonctions.
Informations et numéro de version.

Six appendices complètent la documentation : Garantie, Objets, Le Microprocesseur, Appels systèmes, messages d'erreurs et commandes cachées.

Citons tout d'abord un rappel des différents types d'objets manipulés par la HP-48SX ainsi que leur structure. Un rappel de la structure interne du microprocesseur Saturn, de ses registres, et des opcodes - ? , un expert en langage machine saura de quoi il s'agit !-, au nombre de 490. Une liste de 1301 points d'entrée du système complète la documentation, il faut noter que cette liste a été établie par Smart Technology elle-même à l'aide d'outils propres. Six messages d'erreurs spécifiques à la SmartROM sont indiqués, ainsi que leur numéro d'erreur. Terminons par les commandes cachées, celles-ci au nombre de 162 possèdent un numéro d'XLIB inférieur à 255, ceci suggère une prochaine version de la carte avec de nouvelles fonctions cachées.

La SmartROM est une carte très puissante par la multitude des utilitaires intégrés qu'elle propose, sa force principale provient d'un cohérence très nette des fonctions intégrées entre elles. Un souci du détail a été soigné jusque dans les messages d'erreurs. La documentation est remarquable bien que quelques coquilles d'impression subsistent, des exemples plus difficiles seraient souhaitables au niveau des fonctions cachées et de certaines autres pour bien les comprendre. Une référence persistante est faite au langage RPL, sur la manipulation des objets. Une indication sur la manière d'acquérir ces connaissances serait judicieuse (n'étant pas moi-même un spécialiste).

En conclusion, nous disposons d'un outil sérieux, de bonne qualité, puissant surtout au niveau du traitement matriciel. Une étude soutenue serait

nécessaire pour localiser les bugs et caractéristiques subtiles mais celle-ci semble requérir des connaissances en système dont je ne dispose pas actuellement.

On se surprend à souhaiter de nouvelles fonctions au niveau du calcul matriciel et de l'étude du langage RPL.

Une nouvelle version est-elle en étude ?

Asdin AOUFI (562)

Dernière minute : Nous avons reçu, quelques jours avant l'impression du journal, le manuel décrivant les fonctions cachées, sous forme de fichier ASCII. Nous n'avons pas eu le temps de l'étudier, mais sachez qu'il occupe plus de 140000 octets, soit environ 180 pages ! D'autre part, une nouvelle version est effectivement annoncée, intègrant un assembleur évolué, nommé ESA.

---

5 RANGS DE REFLEXION

REFLEXION est un jeu de réflexion (eh oui) qu'on lance par ESSAI. Au Début programme demande le niveau : on appuiera sur le chiffre correspondant; puis on demande si le joueur veut jouer en premier. Appuyer sur 1 revient à dire oui. Appuyer sur une autre touche ordonne à l'ordinateur de jouer en premier.

Dans les deux cas une matrice s'affiche. C'est un carré d'ordre 5*5. On a donc :

```
L   1 [ x x x x x ]
I   2 [ x x x x x ]
G   3 [ x x x x x ]
N   4 [ x x x x x ]
E   5 [ x x x x x ]
        1 2 3 4 5
      C O L O N N E
```

Où x est un chiffre compris entre 1 et 9.

Le but du jeu est de prendre chacun son tour un chiffre. Celui ci est alors éliminé du tableau et est remplacé par un 0 (zéro). Ces chiffres s'ajoutent aux scores du joueur (J) et de la machine (M) et c'est celui qui a le plus grand score qui gagne.

Mais la difficulté réside dans le fait que l'on est obligé de choisir dans la Ligne ou la Colonne de l'adversaire. Un pavé rappelle d'ailleurs ou l'adversaire vient de jouer.

Pour choisir le chiffre que l'on veut jouer on se sert de la barre de menu qui s'affiche momentanément.

On a la barre de menu suivante : 1 2 3 4 5 cons pour choisir la LIGNE. La touche cons signifie que l'on demande Conseil à la machine. Puis pour la colonne s'affiche le même type de menu : 1 2 3 4 5 fin. La touche fin signifie que le joueur ne peut plus jouer et qu'il abandonne.

Une fois le choix fait la machine réfléchit et affiche son choix pendant 3 secondes. Son score augmente, le chiffre est éliminé (0) et un pavé s'allume à sa place pour rapeller au joueur ou il doit jouer.

Quand la machine ne peut plus jouer, elle abandonne.

Exemple :

Vous décidez de jouer en premier. La machine attends le choix de la ligne par appui sur la barre de menu. Vous choisissez 1 (par exemple). Puis la machine demande la colonne : prenez 5.

Vous prenez le chiffre correspondant à l'intersection des deux, votre score augmente, la machine réfléchit et affiche le nombre de coups envisagés (56 maximum pour le niveau 1, 2350 maximum pour le niveau 2). Puis elle affiche son coup, augmente son score , et c'est de nouveau à vous de jouer.

Les deux chiffres ont été éliminés et la machine a joué soit dans la même ligne que vous soit dans la même colonne. De même vous devrez jouer dans la même ligne que la machine ou dans la même colonne. C'est un contrainte mais c'est tout là l'intérêt du jeu.

Remarque : au niveau 1 la machine répond au bout de 30 secondes, au niveau 2 au bout d' 1/2 heure et elle joue (paradoxalement) moins bien.

Remarque 2 : la machine ne vérifie pas la validité du coup du joueur. En conséquence soyez beau joueur et répondez dans la même ligne ou la même colonne que vient de jouer la machine. Evidemment au premier coup vous pouvez jouer ou vous voulez.

Remarque 3 : il se peut très bien que l'on ne puisse plus jouer malgré qu'il reste des chiffres dans le tableau. Cela tient au fait que dans la colonne ou la ligne ou l'on doit jouer il n'y a plus rien (que des zéros).

```
REFLEXION
  DIR
    ESSAI
      «
        ERASE
        CLEAR
        0 'EXPLO'  STO
        0 'SCOREJ' STO
        0 'SCOREM' STO
        { 1 -1 } 'CHOIX1' STO
        1 5
        FOR I
          1 5
          FOR J
            TABLEAU I J 2 →LIST
            RAND 9 * FLOOR 1 +
            PUT 'TABLEAU' STO
          NEXT
        NEXT
        "NIVEAU DE JEU (1,2)"
        1 DISP "" 2 DISP
        DO
          UNTIL KEY
        END
        81 -
        2 * 'NIVEAU' STO
        "VOULEZ VOUS JOUER EN 1"
        1 DISP
        DO
          UNTIL KEY
        END
        IF 82 ≥
        THEN
          DEBUT
        END
        1 13
        FOR I
          TABLEAU 1 DISP
          CARRE "" 6 DISP
          { 1 2 3 4 5 "CONS" }
          TMENU
          DO
            "CHOISISSEZ LA LIGNE"
            7 DISP -1 WAIT →STR
            2 2 SUB OBJ→ DUP
            DUP
            IF 6 ==
            THEN
              CONSEIL
            END
          UNTIL 6 ≥
          END
          { 1 2 3 4 5 "FIN" }
          TMENU
          "CHOISISSEZ LA COLONNE"
          7 DISP -1 WAIT →STR
          2 2 SUB OBJ→ DUP
          IF 6 ==
          THEN
            FIN
          END
          2
          →LIST 'CHOIX1' STO
          CHOIX1 DUP TABLEAU
          SWAP GET 'SCOREJ' STO+
          DUP TABLEAU
          SWAP 0 PUT
          'TABLEAU' STO
          TABLEAU 1 DISP
          CARRE "" 6 DISP
          "JE REFLECHIS :::"
          7 DISP NIV12 TOTAL
          DEPTH
          IF 0 >
          THEN
            1000 1 BEEP 2 GET
            DUP →STR
            "JE JOUE : " SWAP +
            7 DISP
            3 WAIT
            'CHOIX1' STO CHOIX1
            DUP TABLEAU SWAP
            GET 'SCOREM' STO+
            TABLEAU SWAP 0 PUT
            'TABLEAU' STO
            0 'EXPLO' STO
          ELSE
            "" 6 DISP
            1000 1 BEEP
            "JE NE PEUX PAS JOUER"
            7 DISP 3 WAIT
            26 'I' STO
          END
        NEXT
        CLLCD "JOUEUR : "
        SCOREJ →STR +
        3 DISP "MACHINE : "
        SCOREM →STR +
        5 DISP 3 WAIT
        2 MENU
      »

CONSEIL
  «
    2000 .5 BEEP
    "CONSEIL EN COURS :::"
    7 DISP CHOIX1 → A
    «
      A NIV12
      TOTAL 2000 .5 BEEP
      DEPTH
      IF 0 >
      THEN
        2 GET →STR
        "JE JOUERAI : "
        SWAP +
```

```
      7 DISP 3 WAIT
    ELSE
      2000 .5 BEEP
    END
    "" 6 DISP
    0 'EXPLO' STO
    A 'CHOIX1' STO
  »
  6
»

CARRE
«
  GROB 25 7 FF00401001 00C810010045100100
            421001004010 11004010E0004010
  GROB 6 7 F3F3F3F3F3F3F3
  → SCOJM CARRE
  «
    CHOIX1
    LIST→ DROP 12 * 6 +
    R→B SWAP 1 - 8 *
    R→B 2 →LIST LCD→
    SWAP CARRE REPL
    { # 64h # 0h }
    SCOJM REPL
    { # 62h # Ah }
    SCOREJ 2 →GROB REPL
    { # 74h # Ah }
    SCOREM 2 →GROB REPL
    →LCD
  »
»

DEBUT
«
  CLLCD TABLEAU
  1 DISP
  "JE REFLECHIS :::"
  7 DISP 1 WAIT 9 → A
  «
    1 10
    FOR B
      1 5
      FOR I
        1 5
        FOR J
          TABLEAU I J 2
          →LIST GET
          IF A ==
          THEN 1000 1 BEEP
            I J 2 →LIST
            'CHOIX1' STO CHOIX1
            →STR "JE JOUE : "
            SWAP + 7 DISP 3
            WAIT CHOIX1 DUP
            TABLEAU SWAP GET
            'SCOREM' STO+
            TABLEAU SWAP 0 PUT
            'TABLEAU' STO 5 'I' STO
            5 'J' STO 10 'B' STO
          END
        NEXT
      NEXT
      A 1 - 'A' STO
    NEXT
  »
»

FIN
«
  CLLCD
  CLEAR "JOUEUR : "
  SCOREJ →STR + 3
  DISP "MACHINE : "
  SCOREM →STR + 5
  DISP 3 WAIT 2 MENU
  0 DOERR
»

LETOTALNIV4
«
  CHOIX22
  CHOIX2 CHOIX1
  1 4
  FOR I
    TABLEAU SWAP GET 4
    ROLL
  NEXT
  + NEG + + EXPLORATION
»

LETOTALNIV2
«
  TABLEAU
  SWAP GET TABLEAU
  CHOIX1 GET -
  EXPLORATION
»

TOTALBIS
«
  IFERR
    WHILE DEPTH 1 ≥
    REPEAT
      MAX
    END
  THEN
  END
  CHOIX1 2 →LIST
»

TOTAL
«
  DEPTH 1 - DUP
  IF 0 >
  THEN
```

```
      1 SWAP
      FOR I
        DUP2 1 GET SWAP 1 GET
        IF <
        THEN
          SWAP DROP
        ELSE
          DROP
        END
      NEXT
    ELSE
      DROP
    END
  »

NIV2
  «
    'CHOIX2' STO
    1 5
    FOR I
      CHOIX2 2 I PUT → A
      «
        IF A CHOIX12 ≥ A CHOIX1
         ≥ A CHOIX22 ≥ A
         CHOIX2 ≥ AND AND AND
        THEN
          A LETOTALNIV4
        END
      »
    NEXT
    1 5
    FOR I
      CHOIX2 1 I PUT → A
      «
        IF A CHOIX12 ≥ A CHOIX1
         ≥ A CHOIX22 ≥ A
         CHOIX2 ≥ AND AND AND
        THEN
          A LETOTALNIV4
        END
      »
    NEXT
  »

NIV22
  «
    'CHOIX22' STO
    1 5
    FOR I
      CHOIX22 2 I PUT → A
      «
        TABLEAU A GET
        IF 0 ≥
        THEN
          IF A CHOIX12 ≥ A
           CHOIX1 ≥ A CHOIX22
           ≥ AND AND
          THEN
            A NIV2
          END
        END
      »
    NEXT
    1 5
    FOR I
      CHOIX22 1 I PUT → A
      «
        TABLEAU A GET
        IF 0 ≥
        THEN
          IF A CHOIX12 ≥ A
           CHOIX1 ≥ A CHOIX22
           ≥ AND AND
          THEN
            A NIV2
          END
        END
      »
    NEXT
  »

NIV1
  «
    'CHOIX1' STO
    1 5
    FOR I
      CHOIX1 2 I PUT → A
      «
        IF A CHOIX12 ≥ A CHOIX1 ≥ AND
        THEN
          A
          IF NIVEAU 2 ==
           TABLEAU A GET 0 ==
           OR
          THEN
            LETOTALNIV2
          ELSE
            NIV22
          END
        END
      »
    NEXT
    1 5
    FOR I
      CHOIX1 1 I PUT → A
      «
        IF A
         CHOIX12 ≥ A CHOIX1
         ≥ AND
        THEN
          A
          IF NIVEAU 2 ==
           TABLEAU A GET 0 ==
           OR
          THEN
            LETOTALNIV2
```

```
        ELSE
          NIV22
        END
      END
    »
  NEXT
»

NIV12
«
  'CHOIX12' STO
  1 5
  FOR I
    CHOIX12 2 I PUT → A
    «
      TABLEAU A GET
      IF 0 ≥
      THEN
        IF A CHOIX12 ≥
        THEN
          A NIV1 TOTALBIS
        END
      END
    »
  NEXT
  1 5
  FOR I
    CHOIX12 1 I PUT → A
    «
      TABLEAU A GET
      IF 0 ≥
      THEN
        IF A CHOIX12 ≥
        THEN
          A NIV1 TOTALBIS
        END
      END
    »
  NEXT
»
SCOREM 50
SCOREJ 54
NIVEAU 2
CHOIX2
{ 4 5 }
CHOIX22
{ 3 5 }
CHOIX1
{ 5 3 }
CHOIX12
{ 5 5 }
TABLEAU
[[ 2 0 0 0 0 ]
 [ 4 1 0 2 3 ]
 [ 3 2 0 0 0 ]
 [ 5 0 0 0 3 ]
 [ 0 0 0 0 0 ]]
EXPLORATION
  «
    1 'EXPLO' STO+
    "                "
    EXPLO →STR + 6 DISP
  »
 EXPLO 0
END
END
```

Christophe Vaillant (523)

LE MOIS LE PLUS LONG

Je vous propose un calendrier perpétuel pour la HP48.

Le fonctionnement est en très simple : après introduction des programmes, lancez le par CALEND. Le mois courant apparait :

```
      Juin 1992
Di Lu Ma Me Je Ve Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

La ligne de menu affiche :
MO- MO+ AN- AN+ ↑↓ FIN

MO+ incrémente le mois, MO- le décrémente.
AN+ incrémente l'année, AN- la décrémente.

La touche ↑↓ sert dans le cas ou le mois est affiché sur plus de 5 lignes (cas du mois d'août 92 par exemple). L'appui sur cette touche permet de supprimer le ligne d'en-tête, et d'afficher les jours de la dernière semaine.

La touche FIN termine le programe. Utilisez cette touche plutôt que la touche ON, car certains flags ont été modifiés.

Organisation du programme :

Le programme est constitué du module principal CALEND, et fait appel à la fonction DOW. Cette fonction, ainsi que ADOW fournie en prime, ont été écrites pour la circonstance, mais sont utilisables par ailleurs :

DOW (Day Of Week) :

retourne le numéro du jour de la semaine correspondant à la date placée sur la pile.
0: dimanche,
1: lundi,
2: mardi
3: mercredi,
4: jeudi,
5: vendredi,
6: samedi.

Ex: 14.071992 DOW donne 2

ADOW (Alpha Day Of Week)
retourne une chaine donnant le nom du jour de la semaine correspondant à la date placée sur la pile.

Ex: 25.121992 ADOW donne "Vendredi".

```
«
  1.01199 SWAP DDAYS
  1 + 7 MOD
»
'DOW' STO

« DOW
  { "Dimanche"
    "Lundi" "Mardi"
    "Mercredi" "Jeudi"
    "Vendredi" "Samedi"
  } SWAP 1 + GET
»
'ADOW' STO

«
  0 0 0 0 0 0 → a m i j n k
  «
  RCLMENU
  DATE FP 100 * DUP IP 'm' STO
  FP 10000 * 'a' STO
  DO
    "    "
    {
      "Janvier" "Fevrier" "Mars" "Avril"
      "Mai" "Juin" "Juillet" "Aout" "Septembre"
      "Octobre" "Novembre" "Decembre"
    }
    m GET + "  " +
    a →STR +
    1 DISP 1 FREEZE
    "Di Lu Ma Me Je Ve Sa" 2 DISP 2 FREEZE
    a 10000 / m + 100 / 1 + DOW 7 SWAP - 3 *
    'i' STO
    CASE
      { 4 6 9 11 } m POS
      THEN
        30
      END
      m 2 ==
      THEN
        IF a 4 MOD
        THEN
          28
        ELSE
          29
        END
      END
      31
  END
  7 + 3 * 'n' STO
  "                    1  2  3  4  5  6  7  8
   9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
   24 25 26 27 28 29 30 31"
  IF k
  THEN
    2
  ELSE
    3
  END
  'j' STO
  DO
    DUP
    i DUP 21 + n MIN 2 -
    SUB j DISP
    'i' 21 STO+
    'j' 1 STO+
    UNTIL i n > j 7 > OR
  END
  DROP
  2 FREEZE
  { "MO-" "MO+" "AN-" "AN+" "↑↓" "FIN" }
  TMENU
  -1 WAIT
  CASE
    DUP 11.1 ==
    THEN
      'm' 1 STO-
      IF m 1 <
      THEN
        12 'm' STO 'a' 1 STO-
      END
    END
    DUP 12.1 ==
    THEN
      1 'm' STO+
      IF m 12 >
      THEN
        1 'm' STO 'a' 1 STO+
      END
    END
    DUP 13.1 ==
    THEN
      'a' 1 STO-
    END
```

```
      DUP 14.1 ==
      THEN
        'a' 1 STO+
      END
    END
    IF DUP 15.1 ==
    THEN
      1 k -
    ELSE
      0
    END
    'k' STO
    UNTIL 16.1 ==
  END
  MENU
  »
»
'CALEND' STO
```

Jean-François GARNIER (242)

Note de l'éditeur : La suite de nombres 1 2 3 ... 31 présente dans le programme CALEND fait partie d'une seule et unique chaine de caractères.

AU NOM DE LA LOI

Les programmes qui suivent, servent essentiellement à calculer les lois de Probabilités suivantes :

- Binomiale
- Chi2
- Fisher-Snedecor
- Géométrique
- Normale
- Poisson
- Student

et ceci de manière DIRECTE et INVERSE. Il n'y a plus qu'à mettre les tables papiers à la poubelle.

Pour cela il faut d'abord créer un répertoire père PROBA où l'on implantera les programmes suivants (qui n'ont pas de propriétaire, mais que l'on trouvera par exemple dans le livre de J.M. FERRARD) (Merci Monsieur Ferrard). Certaines de ces fonctions utilisent les précédentes.

Descriptions :

CNP calcul des fameux coefficients combinatoires (c'est COMB corrigé) . La syntaxe est : n p -> CNP.

```
«
  IFERR COMB
  THEN DROP2
  END
»
'CNP' STO
```

BNP calcule la Probailité *P(Y=k)*, où *Y* est une variable aléatoire (v.a.) binomiale de loi B(n,p). Syntaxe : n p k -> BNP.

```
« → n p k
  « n k CNP
    IF DUP
    THEN p k ^ 1 p - n k - ^ * *
    END
  »
»
'BNP' STO
```

FBNP calcule la fonction de répartition d'une v.a. Binomiale *B(n,p)*. La syntaxe sera n p k -> FNBP.

```
« → n p k
  «
    IF n 2 / k <
    THEN 1 n 1 p -
      n k - 1 - FBNP -
    ELSE 0 0 k
      FOR j n p j BNP +
      NEXT k 0 ≥ *
    END
  »
»
'FBNP' STO
```

GEO calcule la Probabilité *P(Y=x)*, où *Y* est une v.a. Géométrique de paramètre *p*. Syntaxe : p x -> GEO.

```
«
  → para X 'para*(1-para)^(X-1)*(X>0)'
»
'GEO' STO
```

FGEO calculera la fonction de répartition d'une loi géométrique de paramètre *p*. Et sa syntaxe est : p -> FGEO.

```
«
  → para X '(1-(1- para)^X)*(X>0)'
»
'FGEO' STO
```

POIS calcule la Probabilité *P(Y=k)*, où *Y* est une v.a. de Poisson *P(p)* de paramètre *p*. Syntaxe : `p k -> POIS`.

```
«
  → para X 'EXP(-para)*para^X/X!'
»
'POIS' STO
```

FPOIS calcule la fonction de répartition d'une loi de Poisson de paramètre *p*. Et sa syntaxe est : `p -> FPOIS`.

```
«
  → para  X  'Σ(j=0,X,POIS(para,j))'
»
'POIS' STO
```

Bien entendu on peut ajouter d'autres lois...

Et voici maintenant les programmes permettant de calculer et d'inverser les différentes lois de probabilités.

On les mettra dans le même répertoire, ou mieux dans un répertoire fils, on implantera les équations suivantes. Ces équations dénommées `XXXX.EQ` seront montées à l'écran, et utilisée par les fontions `STEQ` et `SOLVR` de la HP48.

NOTATION : dans la suite si *Loi(prob, truc1, truc2,..,x)* est la Loi de Probabilité d'une v.a. donnée *y*, alors *truc1*, *truc2*,.. sont des paramètres spécifiques à la loi, et *prob*, *x* sont des nombres tels que :

$$Probabilité(y \leq x) = prob$$

Et les équations suivantes permettent, en se donnant tous les paramètres (sauf un) d'en déduire celui ci.

IBNP.EQ permettra de résoudre de manière *directe* et *inverse* la loi de probabilité Binomiale *B(N,PARA)*. Parmi les 4 nombres `PROB N PARA` ou `X` il faut entrer les valeurs de 3 paramètres et résoudre (par le shift orange) le 4ième paramètre.

```
«
  prob N para X  FBNP -
»
'IBNP.EQ' STO
```

ICHI2.EQ fera de même pour la loi du Chi2 à N degré de liberté. Ici même procédure: parmi `PROB N X` entrer 2 pramètres et résoudre pour le 3ième.

```
«
  1 prob - N X UTPC  -
»
'ICHI2.EQ' STO
```

IFISH.EQ fera de même pour la loi de FISHER SNEDECOR à N1 et N2 degrés de liberté. Parmi `PROB N1 N2` et `X` entrer 3 paramètres et résoudre le 4ième.

```
«
  1 prob - N1 N2 X  UTPF -
»
'IFISH.EQ' STO
```

IGEO.EQ fera de même pour la loi Géométrique (PARA). Parmi `PROB PARA` et `X`, entrer 2 valeurs et résoudre pour la 3ième.

```
«
  prob para X FGEO -
»
'IGEO.EQ' STO
```

INOR.EQ fera de même pour la loi Normale ou Gaussienne *N(moyenne, écart type)*. Par défault la moyenne (resp: écart type) vaut 0 (resp.: 1).

```
«
  0 'moy' STO 1
  'var' STO 1 prob -
  moy var X UTPN -
»
'INOR.EQ' STO
```

IPOIS.EQ fera de même pour la loi de Poisson.

```
«
  prob  para X FPOIS   -
»
'IPOIS.EQ' STO
```

ISTUD.EQ fera de même pour la loi de Student de degré N.

```
«
  1 prob - N X UTPT   -
»
'ISTUD.EQ' STO
```

Michel Weil (179)

UN VOYAGEUR EGARE

Le livre *Voyage au centre de la HP-48* de Paul Courbis et Sébastien Lalande contient un programme de désassemblage DESASS (page 244). Ce programme donne un GET ERROR systématique sur << M N >> avec argument 0.

Sans entrer dans le détail, pour y remédier, modifier la routine A1 comme suit :

Entre →STR et POS, insérer 1 DUP SUB.

Roger Le Bris (535)

ASSEMBLEUR

Les outils du début

JPC avait pris un très bon départ dans la programmation en assembleur sur HP-28 grâce aux articles de Paul Courbis et Sébastien Lalande. Ces auteurs s'étant arrêtés de publier dans *JPC*, il n'y pas eu de véritable continuation car, si mes souvenirs sont exacts, les seuls programmes en assembleur publiés depuis et avant le n° 76, sont ceux relatifs à la factorisation et au graphisme et écrits par votre serviteur. C'est avec une grande satisfaction que j'ai vu réapparaître ce type de programmation sous la plume de Jean-Louis Attenoux dans *JPC 76*.

Il faut reconnaître que le manque de possibilités de sauvegarde des programmes d'une HP-28 n'incite pas à la création en ce domaine car le travail peut être réduit à néant par un Memory Lost.

Les HP-48, avec leurs cartes Ram qui peuvent devenir Rom ou avec leurs possibilités de sauvegarde sur mémoire de masse, rendent la vie beaucoup plus facile aux programmeurs.

D'autre part il existe déjà des outils de développement ou des réalisations (librairies) mais :

- les auteurs de celles-ci n'ont pas, en général, fourni de fichiers sources, ce qui est frustrant.

- la documentation par HP sur les points d'entrées supportés est incomplète. On est très loin des IDS pour HP-71 avec les 2000 points d'entrées pour 64K de Basic et 16K d'HPIL soit 80K de Rom seulement et une documentation exceptionnellement riche. Pour 256K de Rom HP-48 il n'y a que 2300 points d'entrées supportés dont une partie n'est pas documentée. Aussi n'est-t-il pas étonnant de voir, ici et là des créateurs de librairies utiliser des points d'entrées non supportés.

Devant cette situation le rôle pédagogique d'un club tel que le nôtre est important vis-à-vis des programmeurs potentiels.

En ce qui concerne la programmation en assembleur il faut distinguer :
- la programmation RPL qui fait appel à des adresses s'affichant sous forme d'*External* lors d'une édition de programme.
- la programmation avec les mnémoniques de Saturn telle celle qui se pratique pour réaliser des fichiers Lex ou Bin pour la HP-71 et où l'on travaille directement avec les registres.

Pourquoi faire de l'assembleur ?

Si la Rom de nos HP-48 est très riche et que l'on peut très bien faire de bons programmes, lisibles pour tous, avec les commandes du manuel de base, la programmation en assembleur non seulement étend les possibilités de la machine mais permet un gain de place mémoire et de rapidité souvent très appréciable.

Côté RPL par exemple :

Au lieu de faire :

```
SWAP 4 PICK
```

que la machine compilera sous forme de 3 adresses qui occuperons 3 fois 5 = 15 quartets en mémoire.

En RPL on appellera la routine

```
SWAP4PICKP
```

C'est-à-dire une seule adresse d'où trois fois moins d'occupation mémoire et un gain de rapidité.

En effet dans la première solution chaque commande vérifiera d'abord les conditions d'entrée avant l'exécution :
SWAP : il y a au moins deux éléments sur la pile.
4 PICK : quatre éléments au moins sont sur la pile.

Alors que dans la deuxième solution aucune vérification n'est effectuée.

Or, dans un programme les paramètres sont entrés une seule fois et donc leur validité ne devrait être vérifiée qu'une seule fois si le programme est bien conçu.

Les projets

Les possibilités de sauvegarde des programmes HP-48 nous permettent d'envisager de constituer une programmathèque comme celà à été fait pour les HP-71 et HP-75.
Si un certain nombre de programmes reconnus utiles paraissent dans les colonnes de JPC, cela nous permettrait de réaliser une librairie donnant les mêmes avantages que JPC Rom pour la HP-71.

Les outils du début

Ceux qui possèdent le livre *Voyage au centre de la HP48* emploient déjà les outils que je vous propose ce mois-ci, mais j'utilise souvent des instructions RPL de plus bas niveau ce qui se traduira à l'affichage par des *External* au lieu d'avoir des commandes en clair comme SWAP DROP etc... D'où parfois un gain de rapidité ainsi que d'encombrement mémoire par quelques améliorations.

Ces quelques programmes serviront d'exemples (et non de modèles) pour ceux qui n'ont pas encore programmé en assembleur.

Il nous faut d'abord un programme pour assembler les chaînes de codes hexa. Ce sera ASCOD :

```
ASSCOD

 # 105Bh
 138 octets

« DUP TYPE 2 ≠
  IF
  THEN # 202h DOERR
  ELSE "GROB 8 "
     OVER SIZE 2 / " " +
     + SWAP + STR→
     # 4017h SYSEVAL
     # 56B6h SYSEVAL
     DROP NEWOB
  END
»
```

Puis voici notre premier programme en RPL et Assembleur. On trouvera en parallèle les adresses et les mnémoniques HP pour la partie RPL afin que ceux qui ne possèdent pas encore de compilateur puissent se servir de ces adresses. Ce programme remplace ASSCOD pour sa rapidité.

```
ASC→

        con(5)  #02D9D          * DOCOL
        con(5)  #18F9D          * CK&DISPATCH0
        con(5)  #0400D          * THREE
        con(5)  #02D9D          * DOCOL
        con(5)  #627BB          * DUPLEN$
        con(5)  #055DF          * NULL$
        con(5)  #032C2          * OVER
        con(5)  #6261A          * #5-
        con(5)  #6261A          * #5-
        con(5)  #61C1C          * EXPAND
        con(5)  #03223          * SWAP
        con(5)  #02DCC          * DOCODE
deb     con(5)  (fin)-(deb)
        gosbvl  #06641          * POP#
        gosbvl  #0679B          * SAVPTR
        a=a-1   a
        b=a     a               * compteur
        a=dat1  a               * adresse mémoire
        d0=a                    * réservée
        d1=d1+  5
        a=dat1  a               * adresse chaîne
        d1=a                    * de codes
        d1=d1+  10              * début des codes
        lchex   39
loop                            * conversion
        a=dat1  2               * ASCII -> HEXA
        ?a<=c   b               *
        goyes   inf10           *
        a=a-7   b               * (nouvelle
                                * instruction)
inf10   dat0=a  1               *
        d1=d1+  2               *
        d0=d0+  1               *
        b=b-1   a               *
        gonc    loop            *
        govlng #05143           * GETPTRLOOP
fin     con(5) #60F9B           * SWAPDROP
        con(5) #0312B           * SEMI
        con(5) #0312B           * SEMI
```

Le programme →ASC fait l'inverse de ASC→, il transforme un objet en une chaîne de codes hexa. Il vous permettra de voir comment un objet est codé en mémoire. Par exemple:

```
     « SWAP »     →ASC
donnera  "47A20DDBF1B21303"

→ASC

        con(5)  #02D9D          * DOCOL
        con(5)  #1592D          * CK1NoBlame
        con(5)  #03188          * DUP
```

```
        con(5)  #05902          * OSIZE
        con(5)  #055DF          * NULL$
        con(5)  #032C2          * OVER
        con(5)  #03E6F          * #2*
        con(5)  #61C1C          * EXPAND
        con(5)  #03223          * SWAP
        con(5)  #02DCC          * DOCOL
deb     con(5)  (fin)-(deb)
        gosbvl  #06641          * POP#
        gosbvl  #0679B          * SAVPTR
        a=a-1   a
        b=a     a               * compteur
        a=dat1  a               * adresse mémoire
        d0=a                    * réservée
        d0=d0+  10              * début des données
        d1=d1+  5
        a=dat1  a               * adresse objet
        d1=a
        lchex   39
loop    a=c     b               * conversion
        a=dat1  1               * HEXA -> ASCII
        ?a<=c   b               *
        goyes   inf10           *
        a=a+7   b               * (nouvelle
inf10   dat0=a  2               * instruction)
        d1=d1+  1               *
        d0=d0+  2               *
        b=b-1   a               *
        gonc    loop
        govlng  #05143          * GETPTRLOOP
fin     con(5)  #60F9B          * SWAPDROP
        con(5)  #0312B          * SEMI
```

Les deux programmes suivants sont deux petits utilitaires qui permettent un contrôle des codes entrés avant de les assembler. En effet la moindre erreur de frappe risque de réduire à néant votre travail. PIL5 place des changements de ligne dans la chaîne de façon à pouvoir l'éditer par lignes de 5 caractères et à en faire un contrôle visuel aisé.

PIL5

```
        con(5)  #02D9D          * DOCOL
        con(5)  #18F9D          * CK&DISPATCH0
        con(5)  #0400D          * THREE
        con(5)  #02D9D          * DOCOL
        con(5)  #627BB          * DUPLEN$
        con(5)  #62CB9          * DUPDUP
        con(5)  #04021          * FIVE
        con(5)  #03EF7          * #/
        con(5)  #60F9B          * SWAPDROP
        con(5)  #03DBC          * #+
        con(5)  #03E6F          * #2*
        con(5)  #62D59          * NULL$SWAP
        con(5)  #61C1C          * EXPAND
        con(5)  #03223          * SWAP
        con(5)  #02DCC          * DOCODE
deb     con(5)  (fin)-(deb)
        a=0     w
        gosbvl  #06641          * POP#
        gosbvl  #0679B          * SAVPTR
        c=dat1  a               * adresse mémoire
        d0=c                    * réservée
        rstk=c
        d0=d0+  10              * début données
        d1=d1+  5
        c=dat1  a               * adresse chaîne
        d1=c                    * de codes
        d1=d1+  10              * début des codes
        c=0     w
        lchex   5
        gosbvl  #65807          * idiv
        lchex   0A
        d=c     a
        gonc    decr
loop                            * transfert et
        c=dat1  10              * ajout de newline
        dat0=c  10              *
        d1=d1+  10              *
        d0=d0+  10              *
        c=d     b               *
        dat0=c  2               *
        d0=d0+  2               *
decr    a=a-1   a               *
        gonc    loop            *
        ?b=0    a               *
        goyes   out
        c=b     a
        c=c+c   a
        c=c-1   a
        p=c     0
        c=dat1  wp
        dat0=c  wp
out     gosbvl  #067D2          * GETPTR
        d1=d1+  5
        c=rstk
        dat1=c  a
        d1=d1-  5
        govlng  #05149          *
fin     con(5)  #03244          * DROP
        con(5)  #0312B          * SEMI
        con(5)  #0312B          * SEMI
```

DEPIL fait l'inverse: il restitue la chaîne d'origine.

DEPIL

```
        con(5)  #02D9D          * DOCOL
        con(5)  #18F9D          * CK&DISPATCH0
        con(5)  #0400D          * THREE
        con(5)  #02D9D          * DOCOL
        con(5)  #627BB          * DUPLEN$
        con(5)  #02DCC          * DOCODE
deb     con(5)  (fin)-(deb)
        gosbvl  #06641          * POP#
```

```
        gosbvl  #0679B          * SAVPTR
        b=a     a               * compteur
        a=dat1  a               * adresse chaine
        d0=a
        d0=d0+  10              * début des codes
        cd0ex
        rstk=c                  * sauve l'adresse
        d1=c                    * début transfert
        d0=c
        lchex   0A              * newline
        gonc    decr
loop
        a=dat0  2
        ?a=c    b               * ?newline
        goyes   nwl
        dat1=a  2               * caractère
        d1=d1+  2               * conservé
nwl     d0=d0+  2
decr    b=b-1   a
        gonc    loop
        ad1ex                   * adresse après
                                * dernier caract.
        gosbvl #067D2           * GETPTR
        c=0     w
        c=c+1   a
        r1=c                    * début sub
        c=rstk                  * adresse 1er car.
        c=a-c   a
        csrb                    * nombre caractères
        r0=c                    * fin sub
        st=0    1
        st=0    10
        govlng #05756           * sub et retour RPL
fin
        con(5) #0312B           * SEMI
        con(5) #0312B           * SEMI
```

Vous trouverez dans la rubrique *Le coin des codes HP-48* les chaînes de codes à assembler ainsi que les consignes pour utiliser sans risques ces programmes. Le mois prochain quelques outils supplémentaires vous seront livrés.

Bons débuts en assembleur.

Guy TOUBLANC (276)

CERCLES ET ANIMATION

La commande ARC permet de tracer des cercles mais l'exécution est très lente et il vaut mieux renoncer à faire de l'animation graphique: par exemple pour tracer un cercle de 31 pixels de rayon ma machine me demande 16,6 secondes!!

Aussi plusieurs programmeurs ont apporté une solution en se servant de l'algorithme de *Bresenhamm* beaucoup plus rapide car les calculs se font uniquement avec des soustractions et additions sur les entiers relatifs. Dans l'ouvrage *voyage au centre de la HP48* on trouve l'un de ces programmes, réalisé en assembleur, donc très rapide et qui permet aussi d'effacer des cercles ouvrant ainsi les portes de l'animation graphique.

Ce programme présente quelques faiblesses :
- il se modifie lui-même ce qui ne permet pas de l'utiliser dans une carte Rom.
- aucune vérification n'est faite de la validité des paramètres.
- il utilise le champ A du registre R4 pour les sauvegardes alors que ce champ est réservé pour gérer les interruptions.

D'autre part seule la liste des codes hexa est fournie, donc pas de fichier source pour voir la programmation. Aussi je vous livre un programme sans les faiblesses citées et avec les avantages suivants:
- gain d'occupation mémoire :
  363,5 octets contre 473 octets
- gain de rapidité, par rapport à la commande ARC pour tracer un cercle de 31 pixels de rayon :
  175 fois plus rapide contre 114 fois

Circle

Syntaxe :

```
niveau 2 : le rayon    = 1 réel          r
niveau 1 : le centre c = 1 complexe ( Xc , Yc )
```

Les arguments restent sur la pile.

r positif : traçage d'un cercle de points
r négatif : effaçage d'un cercle de points

A titre d'exemple voici un programme d'animation graphique:

```
DEMO1

 # 615Bh
 197.5 octets

«
 # 131d # 64d
 PDIM ( # 0d # 0d )
 PVIEW
```

```
-31 32
FOR i i ABS i                    @ allume
   0 R→C CIRCLE DROP2 3          @ les
STEP                             @ points
-31 32
FOR i i ABS                      @ éteints
   NEG i 0 R→C CIRCLE            @ les
   DROP2 3                       @ points
STEP                             @
»
```

Il suffit de lancer DEMO1

Ring

Voici maintenant un programme permettant de tracer ou d'effacer un anneau de points.

Syntaxe :

```
niveau 3 : grand rayon R =  1 réel          R
niveau 2 : petit rayon r =  1 réel          r
niveau 1 : le centre    c =  1 complexe ( Xc , Yc )
```

Restent sur la pile R ou r incrémentés et le centre.

R positif : traçage d'un anneau de points
R négatif : effaçage d'un anneau de points

Le programme :

RING

FA28h
79.5 octets

```
«
  3 ROLLD OVER
  0 >
  IF
  THEN SWAP
  ELSE NEG
  END OVER -
  ABS ROT SWAP 0 SWAP
  START CIRCLE
     SWAP 1 + SWAP
  NEXT
»
```

La version en RPL : moins d'occupation mémoire et légèrement plus rapide.

EC29h
59.5 octets

```
CON(5)  #02D9D       * DOCOL
 CON(5)  #6308D       * UNROTOVER
 CON(5)  #2A799       * %0>
 CON(5)  #61AD8       * ITE
 CON(5)  #03223       * SWAP
 CON(5)  #02D9D       * DOCOL
 CON(5)  #2A386       * %-1
 CON(5)  #2A9BC       * %*
 CON(5)  #0312B       * SEMI
 CON(5)  #032C2       * OVER
 CON(5)  #2A981       * %-
 CON(5)  #18CD7       * %ABSCOERCE
 CON(5)  #60EE7       * ROTSWAP
 CON(5)  #03DEF       * #1+
 CON(5)  #073DB       * #1+_ONE_DO
 CON(5)  #02E48       * DOIDNT
 CON(2)  6
 NIBASC  'CIRCLE'
 CON(5)  #03223       * SWAP
 CON(5)  #50262       * %1+
 CON(5)  #03223       * SWAP
 CON(5)  #07334       * LOOP
 CON(5)  #0312B       * SEMI
```

Vous trouverez la liste des codes à assembler dans la rubrique *Le coin des codes HP48*.

A titre d'exemple voici un deuxième programme d'animation graphique:

DEMO2

35F
215 octets

```
«
  # 131d # 64d
  PDIM { # 0d # 0d }
  PVIEW -31 32
   FOR i i ABS
      DUP 2 - i 0 R→C
      ANNEAU DROP2 4
   STEP -31 32
   FOR i i ABS
      NEG DUP ABS 2 - i 0
      R→C ANNEAU DROP2 4
   STEP
 »
```

Il suffit de lancer DEMO2. Le principe est le même que dans DEMO1 : une boucle allume les points d'anneaux au lieu de cercles et l'autre boucle les éteint.

Le programme CIRCLE :

```
CON(5) #02D9D        * DOCOL
CON(5) #18EDF        * CK2&Dispatch
CON(5) #040A3        * EIGHTEEN
CON(5) #02D9D        * DOCOL
CON(5) #02DCC        * DOCODE
```

```
DEBUT   CON(5) (FIN)-(DEBUT)
        GOSBVL #0679B       * SAVPTR
        P=     0
        CD1EX
        D1=C
        R0=C                * sauve pointeur c
        C=DAT1 A            * adresse centre c
        D1=C
        D1=D1+ 5            * passe le prologue
        CD1EX
        D1=C
        R3=C                * sauve adresse Xc
        GOSUB  conver       * conversion en hexa
        D0=(5) #70565
        A=DAT0 A            * adresse de PICT
        AD0EX
        D0=D0+ 15           * pointe largeur
        A=0    M
        A=DAT0 A            * largeur de PICT
        ASRB                * centre en largeur
        C=A+C  A            * pointe centre hor.
        CR3EX               * sauvegarde
        D1=C                * pointe Xc
        D1=D1+ 16           * pointe Yc
        GOSUB  conver       * conversion en hexa
        D0=D0- 5            * @ hauteur PICT
        A=DAT0 A            * hauteur
        ?A#0   A            * hauteur <> 0 ?
        GOYES  hnonul       * alors on continue
        GOTO   OUT          * sinon sortie
hnonul  ASRB                * demi-hauteur
        C=A-C  A            * pointe centre ver.
        CR0EX               * sauvegarde
        D1=C                * pointe 1er niveau
        D1=D1+ 5            * pointe 2e niveau
        C=DAT1 A            * adresse rayon r
        D1=C
        D1=D1+ 5            * passe le prologue
        GOSUB  conver       * conversion hexa r
        GOC    rposit       * saut si r > 0
        C=-C   A            * r < 0
rposit  R2=C                * sauvegarde r
        D0=D0+ 5            * passe la hauteur
        A=DAT0 A            * largeur de PICT
        A=A-1  A            * conversion en
        ASRB                * octets
        ASRB                *
        A=A+1  A            *
        A=A+1  A            *
        ASRB                *
        C=A    A            *
        D=C    A            * sauvegarde
        D0=D0- 5            * @ hauteur PICT
        C=0    A            * départ X = 0
        R1=C                * sauve X
        A=R2                * départ Y = r
        A=A+A  A
        LC(5)  3
        C=C-A  A            * d = 3 - 2*r
        RSTK=C              * sauvegarde
        C=R1                * C(A):=X
        GOTO   start        * départ tracé
                            * maintenant boucle
                            * de tracé
loop    GOSUB  plot         * tracé de 8 points
        C=RSTK              * d
        A=C    A
        P=     4
        LCHEX F
        C=A-C  P            * pour le signe de d
        C=R1                * X
        GONC   dnegat       * d < 0
        CR2EX               * Y
        C=C-1  A            * Y - 1
        B=C    A            * sauve Y
        CR2EX               * sauve Y C:=X
        C=C-B  A            * X := X - Y
dnegat  C=C+C  A
        C=C+C  A            * 4X
        A=A+C  A            * 4X + d
        P=     0
        LC(5)  6
        C=A+C  A            * d := 4X + d + 6
        RSTK=C              * sauve d
        C=R1                * X
        C=C+1  A            * X := X + 1
        R1=C                * sauve X
start   A=R2                * Y
        C=C-A  A            * X - Y
        GOC    loop         * X < Y alors loop
        C=R1                * X
        ?C#A   A            * X <> Y ?
        GOYES  out          * alors sortie
        GOSUB  plot         * encore tracé
out     C=RSTK              * désempile d
OUT     P=     0
        GOVLNG #05143       * GETPTRLOOP
NEGYY   C=R1                * ces différents
        C=-C   A            * segments de
        R1=C                * programme
        GOTO   NEGY         * effectuent
XYEXX   C=R1                * les permutations
        C=-C   A            * entre X et Y et
        R1=C                * les changements
        GOTO   XYEX         * de signe pour
NOXX    C=R1                * obtenir par
        C=-C   A            * symétrie
        CR2EX               * 8 points avec
        C=-C   A            * un couple X Y
        R2=C
        GOSUB  plot2
        GOTO   XYEX
plot    ST=0   0            * marque 1er tour
loop2   GOSUB  plot1
        GOC    NEGYY
        GOSUB  LXX
```

```
NEGY   C=R2              * Y
       C=-C   A          * Y := Y
       R2=C              * sauve Y
       GOSUB  plot2
       GOC    XYEXX
       GOSUB  LXX
XYEX   C=R1              * X
       CR2EX             * Y := X
       C=-C   A          * X := -Y
       R1=C              * sauve X
       ?ST=1  0          * 2e tour ?
       RTNYES            * alors retour
       ST=1   0          * marque 2e tour
       GONC   loop2
LXX    CR1EX             * encore symétrie
       C=-C   A
       CR1EX
       D0=D0+ 5
       GONC   LX
NOX    C=RSTK
       RTNCC
plot1  C=R2              * Y
plot2  P=     0
       A=R0              * Yc
       A=A-C  A
       C=DAT0 A
       ?C<=A  A          * dépassement ?
       RTNYES            * alors retour
       A=A+A  A
       C=D    A
       B=C    A
       D0=D0+ 10
       CD0EX
       D0=C
       D0=D0- 5
       GONC   start4
loop4  C=C+A  A
start4 B=B-1  A
       GONC   loop4
LX     D1=C
       RSTK=C            * sauve pointeur
       A=R1              * calcul pour
       C=R3              * pixon ou pixoff
       A=A+C  A
       C=DAT0 A
       D0=D0- 5
       ?C<=A  A
       GOYES  NOX
       LC(1)  3
       C=A&C  P
       B=C    P
       ASRB
       ASRB
       CD1EX
       C=C+A  A
       D1=C
       LCHEX  1
       GONC   start2
```

```
loop3
       C=C+C  P
start2 B=B-1  P
       GONC   loop3
LRPOS  A=DAT1 1
       ?D#0   S
       GOYES  pixoff
       A=A!C  P          * allume un pixel
       DAT1=A 1
       C=RSTK
       RTN
pixoff A=-A-1 P          * éteint un pixel
       A=A!C  P
       A=-A-1 P
       DAT1=A P
       C=RSTK
       RTNCC
conver                   * teste la validité
       A=0    A          * des paramètres
       C=DAT1 16         * pour la conversion
       D=C    S          * en hexa
       A=C    X
       LCHEX  500
       ?C<=A  X
       GOYES  nul
       LCHEX  004
       ?C>A   X
       GOYES  accept
nul    C=0    W
       RTNSC
accept C=0    M          * début conversion
       C=A    X          * décimal flottant
       D1=D1+ 14         * -> hexa
       AD1EX
       A=A-C  A
       D1=A
       P=C    0
       C=0    A
       C=DAT1 WP
       P=     3
       B=0    A
       D=C    A
       D1=(5) #00C18
       AD1EX
       GOSUB  subcnv
       D1=(5) #0009C
       AD1EX
       GOSUB  subcnv
       D1=(5) #00006
       AD1EX
       GOSUB  subcnv
       C=D    A
       C=C-B  A
       ?C=0   S          * arg. positif ?
       RTNYES            * alors retour
       C=-C   A          * complément
       RTNCC
subcnv                   * sous-routine
```

```
        C=D    P             * de conversion
        GONC   start3
 loop5
        B=B+A  A
 start3 C=C-1  P
        GONC   loop5
        P=P-1
        RTN
FIN     CON(5) #0312B        * SEMI
        CON(5) #0312B        * SEMI
```

Vous trouverez la liste des codes à assembler dans la rubrique *Le coin des codes HP48*.

Bon amusement et bonnes animations. J'espère que le club recevra en retour le fruit de votre imagination.

Guy TOUBLANC (276)

LE COIN DES CODES HP48

Comme il faut un début à tout, je lance cette rubrique à l'instar du COIN des LHEX pour HP71.

Ceux qui n'ont pas encore de programme assembleur pourront procéder de la manière suivante:

- par mesure de sécurité sauvegardez vos programmes et fichiers.
- éventuellement verrouillez vos cartes RAM pour devenir ROMs.

Mettez votre machine en mode HEX pour obtenir les mêmes résultats que ceux précisés ici.

- tapez le programme ASSCOD puis la chaîne de codes hexa ASC→ sans espaces. L'encombrement mémoire et le checksum sont donnés pour les programmes mis sur la pile.

ASSCOD

105Bh
138 octets

```
« DUP TYPE 2 ≠
  IF
  THEN # 202h DOERR
  ELSE "GROB 8 "
    OVER SIZE 2 / " " +
    + SWAP + STR→
    # 4017h SYSEVAL
    # 56B6h SYSEVAL
    DROP NEWOB
  END
»
```

ASC→

1AEFh
162 octets pour la chaîne de codes

DF19h
78.5 octets pour la chaîne assemblée

```
D9D20 D9F81 D0040 D9D20 BB726 FD550 2C230 A1626
A1626 C1C16 32230 CCD20 25000 8F146 608FB 9760C
CD814 31301 74143 13117 93193 15B19 EA808 18E86
15801 71160 CD54E 8D341 50B9F 06B21 30B21 30
```

Cette chaîne de codes étant sur la pile tapez ASSCOD vous obtiendrez sur la pile:

```
External <3h>
 External ""
 External External
 External External
```

que vous stockerez dans ASC→, ce programme vous permettra d'assembler très rapidement les chaînes de codes hexa, remplacera ASSCOD et donc sera à utiliser pour l'assemblage des chaînes suivantes.

L'utilitaire suivant vous permettra de relire les codes avant assemblage car il est très sage de faire cette relecture. Il regroupera les codes par lignes de 5 caractères.

PIL5

93A6h
239 octets pour la chaîne de codes

956Dh
117 octets pour la chaîne assemblée

```
D9D20 D9F81 D0040 D9D20 BB726 9BC26 12040 7FE30
B9F06 CBD30 F6E30 95D26 C1C16 32230 CCD20 09000
AF08F 14660 8FB97 60147 13406 16917 41471 35179
AF230 58F70 85631 A0D75 A115F 915C9 17916 9AEB1
5C116 1CC55 E8A94 1D9C6 CE80D 01571 15418 F2D76
01740 71451 C48D9 41504 4230B 2130B 2130
```

Pour restituer la chaîne à assembler voici le programme complémentaire à PIL5.

DEPIL

AC61h
158 octets pour la chaîne de codes

3D18h
76.5 octets pour la chaîne assemblée

```
D9D20 D9F81 D0040 D9D20 BB726 CCD20 17000 8F146
608FB 9760D 81431 30169 13606 13513 431A0 55115
A1962 90159 11711 61CD5 AE133 8F2D7 60AF2 E6109
07EE8 1E108 84184 A8D65 750B2 130B2 130
```

Donc à partir de maintenant pour tout assemblage de chaîne de codes procédez de manière suivante :
1- tapez la chaîne de codes sans espaces
2- faire DUP BYTES
3- contrôlez le checksum et le nombre d'octets
4- si ce n'est pas bon faire PIL5 puis éditez la chaîne pour la relire et modifier les codes. Puis faire ENTER DEPIL et revenir à l'étape 2.
5- la chaîne contrôlée étant sur la pile faire ASCv et stockez le programme assemblé dans la variable donnée en tête de liste de codes. Après avoir rappelé le programme sur la pile puis tapé BYTES vous devriez obtenir le checksum et le nombre d'octets indiqués pour la chaîne assemblée.

Guy TOUBLANC (276)

Ce mois-ci vous trouverez les listes de codes des programmes en assembleur décrits dans les pages précédentes:

```
→ASC CIRCLE RING
```

```
→ASC
```

3632h
150 octets pour la chaîne de codes

7E07h
72.5 octets pour la chaîne assemblée

```
D9D20 D2951 88130 20950 FD550 2C230 F6E30 C1C16
32230 CCD20 55000 8F146 608FB 9760C CD814 31301
69174 14313 13193 AEA15 B09EA 80818 E0615 81170
161CD 51E8D 34150 B9F06 B2130
```

```
CIRCLE
```

5B93h
732 octets pour la chaîne de codes

DA13h
363.5 octets pour la chaîne assemblée

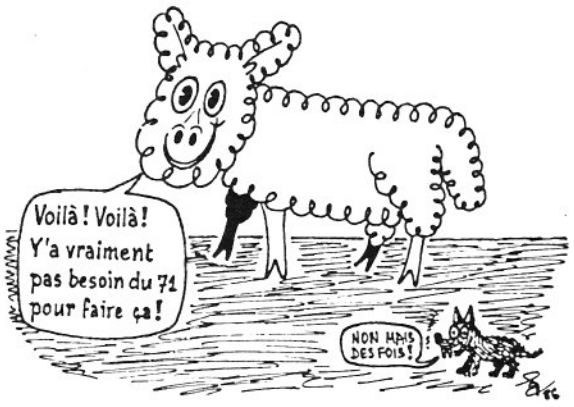
```
D9D20 FDE81 3A040 D9D20 CCD20 4B200 8FB97 60201
37135 10814 71351 74137 13510 B78F1 1B565 07142
13216 EAD01 4281C C212B 13517 F70D1 18414 28AC6
061B0 81CEE 12813 51741 47135 17476 A1440 FA10A
16414 2CC81 C81CE 4E481 CD6D7 184D2 10911 2C434
30000 E2061 19604 07580 07DA2 430FB 0E119 5E012
ACED5 12AE9 C6C6C A2034 60000 C2061 19E61 09112
E24DB 1198A 66078 30072 08D34 15011 9FA10 96230
119FA 10969 30119 FA12A FA10A 70506 42084 07240
4BC79 2011A FA10A 72304 4C761 01191 2AFA1 09870
00850 5EC12 9FA12 91645 43070 311A2 0110E A1468
BE00C 4DBD5 16913 61341 84540 C2CD5 BF135 06111
11BCA 14618 48BE8 B3030 E02A8 581C8 1C137 C2135
30155 0A06A 0D59F 15B09 4FE00 E0E15 90070 1B8C0
E0EB8 C1510 0703D 015FF AC7AB A3200 59BEC 03240
09B27 0AF20 2AD2A B617D 133EA 13180 D0D21 57123
D1D71 F81C0 01337 9201F C9000 1337B 101F6 00001
337D0 0DBE9 94A00 FA03A 8B540 C8A0E 5AF0D 01B21
30B21 30
```

```
RING
```

66D4h
124 octets pour la chaîne de codes

EC29h
59.5 octets pour la chaîne assemblée

```
D9D20 D8036 997A2 8DA16 32230 D9D20 683A2 CB9A2
B2130 2C230 189A2 7DC81 7EE06 FED30 BD370 84E20
60349 42534 C4543 22302 62053 22304 3370B 2130
```



HP-71

G. Toublanc	Tracé de cercles	*24*

ENCERCLEMENT DE TITAN

On sait que nos HP-71 sont capables de faire dans le confort du graphisme grâce au lex GRAPHIC de Pierre David, intégré dans la version E de JPCRom. A l'origine la visualisation des graphiques ne se faisait que sur papier, mais j'avais montré (*JPC 71-72*) que l'on pouvait très bien afficher tout fichier graphique sur l'écran LCD de CMT ou même sur l'écran de nos HP-71.

Si les commandes de traçage de lignes et de cadres sont bien présentes, il n'en est pas de même pour les arcs et cercles pour lesquels il faut avoir recours à des programmes basic d'exécution lente.

Pour remédier à cela je vous propose un lex avec de nouvelles commandes pour tracer des cercles et des anneaux, et aussi pour les effacer. Tout cela se fait rapidement grâce à un algorithme où les calculs se font uniquement avec des soustractions et additions sur les entiers relatifs. Le source du lex n'est pas commenté car je l'ai déjà fait dans la rubrique HP48.

Circle

Syntaxe : `CIRCLE(D, Xc, Yc, R)`

`D` : adresse (décimale) du début des données graphiques dans le fichier graphique.
`Xc` : abscisse du centre du cercle.
`Yc` : ordonnée du centre du cercle.
`R` : rayon du cercle.
si `R` positif : allume les pixels.
si `R` négatif : éteint les pixels.

Exemple :

```
10 INPUT "file ? ","graphile";F$
 - nom du fichier graphique
20 INPUT "Xc,Yc,R ? ";X,Y,R
 - coordonnées du centre puis rayon
30 A=HTD(ADDR$(F$))
 - adressse du fichier
   vérifie en même temps si le fichier existe
40 IF PEEK$(A+16,4)#"222E" THEN BEEP @ END
 - vérifie si c'est un fichier graphique
    sinon stoppe
50 D=A+42
 - adresse départ du graphique
60 CIRCLE D,X,Y,R
 - tracé du cercle
```

Ring

Syntaxe : `RING D,Xc,Yc,R,r`

`D` : adresse (décimale) du début des données graphiques dans le fichier graphique.
`Xc` : abscisse du centre de l'anneau.
`Yc` : ordonnée du centre de l'anneau.
`R` : rayon extérieur de l'anneau.
`r` : rayon intérieur de l'anneau.
si `R` positif : allume les pixels.
si `R` négatif : éteint les pixels.

Exemple :

```
10 INPUT "file ? ","graphile";F$
 - nom du fichier graphique
20 INPUT "Xc,Yc,R,r ? ";X,Y,R,r
 - coordonnées du centre de l'anneau
   grand rayon de l'anneau
   petit rayon de l'anneau
30 A=HTD(ADDR$(F$))
 - adressse du fichier
   vérifie en même temps si le fichier existe
40 IF PEEK$(A+16,4)#"222E" THEN BEEP @ END
 - vérifie si c'est un fichier graphique
    sinon stoppe
50 D=A+42
 - adresse départ du graphique
60 RING D,X,Y,R,r
 - tracé de l'anneau
```

Remarques :

Si vous ne possédez pas la version E de JPCRom il faut remplacer la ligne 40 par:

```
IF PEEK$(DTH$(A+16,4))#"222E" THEN BEEP @ END
```

Le progamme :

```
        LEX     'CERCLEX'
        ID      #68
        MSG     0
        POLL    0
        ENTRY   CIRC
        CHAR    #D
        ENTRY   RING
        CHAR    #D
        KEY     'CIRCLE'
        TOKEN   24
        KEY     'RING'
        TOKEN   25
        ENDTXT
ARGERR  EQU     #0BF19
CSLW5   EQU     #0ED3D
COMCK   EQU     #036CD
DROPDC  EQU     #05470
EXPEX-  EQU     #0F178
MSPARe  EQU     #02E5C
NUMCK   EQU     #0369D
NXTSTM  EQU     #08A48
RESPTR  EQU     #03172
```

```
RNDAHX  EQU     #136CB
popa    GOSBVL  RNDAHX
        ST=0    2
        GOC     pos
        ST=1    2
        A=-A    A
 pos    P=      4
        C=0     W
        C=A     A
        D1=D1+  16
        ?A#0    P
        GOYES   argerr
        ?ST=0   2
        RTNYES
        C=-C    A
        RTNCC
argerr  GOVLNG  ARGERR
RINd
CIRd    GOVLNG  DROPDC
RINp    LC(1)   5
        GOTO    L40
CIRp    LC(1)   4
L40     C=C-1   P
        R2=C
        GOSBVL  NUMCK
        GOSUB   resptr
        C=R2
        ?C=0    P
        GOYES   resptr
        GOSBVL  COMCK
        GONC    mspare
        C=R2
        GOC     L40
resptr  GOVLNG  RESPTR
mspare  GOVLNG  MSPARe
        REL(5)  RINd
        REL(5)  RINp
RING    GOSBVL  =EXPEX-
        ST=0    3
        ST=0    1
        GOSUB   popa
        GOC     POS2
        C=-C    A
        ST=1    1
POS2    RSTK=C
        GOTO    CIRIN
        REL(5)  CIRd
        REL(5)  CIRp
CIR     GOSBVL  =EXPEX-
        ST=0    1
        ST=1    3
CIRIN   GOSUB   popa
        GOC     POS
        ST=1    1
        C=-C    A
POS     R2=C
        ?ST=0   3
        GOYES   ann
        RSTK=C
ann     A=C     A
        C=RSTK
        ?A>=C   A
        GOYES   NORM
        R2=C
        ACEX    A
NORM    C=A-C   A
        GOSBVL  =CSLW5
        D=C     M
        C=A     A
        RSTK=C
        GOSUB   popa
        R0=C
        GOSUB   popa
        R3=C
        GOSBVL  =RNDAHX
        D0=A
        D0=D0-  5
        C=DAT0  A
        D0=D0+  5
        D=C     A
        A=R0
        C=C-A   A
        C=C-1   A
        R0=C
LOOPA   C=0     A
        R1=C
        A=R2
        A=A+A   A
        LC(1)   3
        C=C-A   A
        RSTK=C
        C=R1
        GOTO    LCCC
L3064A  GOSUB   L306B9
        C=RSTK
        A=C     A
        P=      4
        LCHEX   F
        C=A-C   P
        C=R1
        GONC    L30677
        CR2EX
        C=C-1   A
        B=C     A
        CR2EX
        C=C-B   A
L30677  C=C+C   A
        C=C+C   A
        A=A+C   A
        P=      0
        LC(5)   6
        C=A+C   A
        RSTK=C
        C=R1
        C=C+1   A
        R1=C
```

```
LCCC    A=R2
        C=C-A   A
        GOC     L3064A
        C=R1
        ?C#A    A
        GOYES   L30560
        GOSUB   L306B9
L30560  C=RSTK
        C=RSTK
        C=C-1   A
        RSTK=C
        R2=C
        C=D     A
        D=0     A
        D=D-1   M
        D=C     A
        GOC     OUT
        GOTO    LOOPA
OUT     C=RSTK
        GOVLNG  =NXTSTM
NEGYY   C=R1
        C=-C    A
        R1=C
        GOTO    NEGY
XYEXX   C=R1
        C=-C    A
        R1=C
        GOTO    XYEX
L306B9  ST=0    0
L306BC  GOSUB   L30705
        GOC     NEGYY
        GOSUB   LXX
NEGY    C=R2
        C=-C    A
        R2=C
        GOSUB   L30705
        GOC     XYEXX
        GOSUB   LXX
XYEX    C=R1
        CR2EX
        C=-C    A
        R1=C
        ?ST=1   0
        RTNYES
        ST=1    0
        GONC    L306BC
LXX     CR1EX
        C=-C    A
        CR1EX
        GOTO    LX
NOX     C=RSTK
        RTNCC
L30705  A=R2
L30706  C=R0
        C=C-A   A
        ?C>=D   A
        RTNYES
        CSL     A
        A=C     A
        A=A+A   A
        A=A+A   A
        C=A+C   A
        C=C+C   A
        ADOEX
        DO=A
        C=C+A   A
LX      D1=C
        RSTK=C
        A=R1
        C=R3
        A=A+C   A
        LC(5)   640
        ?C<=A   A
        GOYES   NOX
        LC(1)   3
        B=C     P
        B=B&A   P
        B=C-B   P
        ASRB
        ASRB
        CD1EX
        C=C+A   A
        D1=C
        LCHEX   1
        GONC    L3078E
LGOC    C=C+C   P
L3078E  B=B-1   P
        GONC    LGOC
        A=DAT1  1
        ?ST#0   1
        GOYES   L307A9
        A=A!C   P
        DAT1=A  1
        C=RSTK
        RTN
L307A9  A=-A-1  P
        A=A!C   P
        A=-A-1  P
        DAT1=A  P
        C=RSTK
        RTNCC
```

Guy TOUBLANC (276)



HP-95

J. Belin	Les logiciels du HP-95	*28*
J. Belin	Le dévelopement de logiciels sur HP-95	*29*
J. Belin	Voyage au centre de la mémoire, le retour	*34*

LES LOGICIELS DU HP-95

Le mois dernier, nous vous avons présenté les différentes cartes contenant des logiciels pour le HP-95. Ce mois ci, nous continuons avec les logiciels disponibles sur disquette. Il ne s'agit que des logiciels commerciaux et non ceux appartenant au domaine public. Comme le mois précédent, lorsque les prix sont indiqués en Francs, cela veut dire que le logiciel est disponible chez certains distributeurs parisiens.

95 Utility Pack de Richard Harvey

Ce package contient 36 utilitaires spécialement développés pour le HP-95, et optimisés afin d'économiser la place sur votre carte mémoire. On y trouvera des utilitaires permettant de récupérer des informations sur la configuration, l'état des batteries, visualiser des fichiers PCX sur l'écran du HP-95, choisir l'aspect du curseur, commander l'interface série, afficher, filtrer ou crypter des fichiers... Et même jouer de la musique ! Prix : 28$.

Curve Fit de *MacroMed*

Après acquisition de coordonées *x,y* (jusqu'à 100), ce logiciel permet d'interpoler une courbe puis d'exécuter certains calculs sur celle-ci : coefficient d'interpolation, aire, séries de Fourier... Affiche les courbes sur l'écran du HP-95. Prix : 170$. Existe aussi en Rom : 235$.

Swift!BASIC de *Sparcom*

Compatible *System Manager*, ce logiciel comporte un Basic structuré couplé à un éditeur contrôlant la syntaxe des lignes entrées. Il est possible de passer d'un programme à l'autre par un simple appui sur une touche pré-définie. Tous les programmes actifs partagent le même run-time. Prix 160$.

UTIL FORTH de *EMS*

Sous ce nom, nous pouvons trouver un ensemble de programmes basés autour d'un compilateur Forth. Le premier module contient le compilateur lui même, qui est compatible *System Manager*, un assembleur 8086, un décompilateur, un désassembleur et plusieurs fichiers sources (fonctions graphiques, jeux...). Prix 65$. Les autres modules sont un Métacompilateur, permettant de créer des exécutables de très petites taille (Prix 25$), un éditeur Hexadécimal (Prix 30$) et un ensemble de fichiers sources comprenant entre autres l'assembleur. Il est possible de trouver en France un ensemble composé du langage et du métacompilateur pour environ 1000F.

muLISP de *Soft Warehouse*

Interpreteur Lisp spécialement adapté aux contraintes du HP-95 (écran, mémoire...). Prix : 400$.

QAPALM de *QA Systems*

Permet de créer de générer des applications orientées Bases de Données. La création de l'application et des écrans se fait sur PC, avant transfert sur le HP-95. Prix 445$.

DATA BASE de *DYNaXX*

Logiciel de gestion de bases de données, comportant un possibilité d'exportation/importation vers DBASE IV. Prix 1600 F.

CONTACT 95 de *DYNaXX*

Logiciel de gestion de contacts, intégré à l'agenda du HP-95, avec accès à un Bloc Notes. Le logiciel est en outre livré avec une version tournant sur PC, avec Windows 3. Carte mémoire de 1 Mo requise. Prix 1600 F

ACT! de *Contact Software Internationnal*

Plus destiné aux représentants, ce logiciel permet de créer une base de données contenant les coordonnées complètes (Nom, coordonnées, autres renseignements...) des clients, puis de gérer les rendez-vous, autres activités ainsi que les dépenses. Il permet aussi d'echanger des "cartes de visites" entre deux HP-95. Cable RS-232 inclus. Prix 90$. Il existe aussi une version pour PC.

95 Money Manager de *Autobyte*

Logiciel de gestion de compte bancaire (Chequiers, cartes de crédits...). Version pour PC incluse. Prix : 55$.

Le monde de la finance au bout des doigts de *BRUCKERT Diffusion*

Composé de formules destinées au SOLVER et de feuilles de calcul pour Lotus, ce logiciel composé de trois modules est principalement réservé aux financiers. Le premier module (référencé *n° 2*) est orienté sur tout ce qui est Crédit, Crédit-bail, Leasing, ammortissement... Le second module (ref. *n°3*) traite principalement des actions, de l'epargne et du placement. Le troisieme (ref. *n° 4*) traite des problèmes d'analyse financière. Chaque module est

disponible pour 1000F. Les trois modules sont vendus ensembles pour 2500F (ref. *n° 1*).

Mechanical Engineering Equation Library de Paul Harmon.

D'une taille d'environ 60 ko, ce package contient environ 160 equations utilisables avec le calculateur, ainsi que des feuiles de calcul 1-2-3 contenant des formules de conversions. Parmi les domaines abordés, on peut trouver des calculs d'engrenages, de poutres, résistance de matériaux, mécanique des fluides, thermodynamique, calculs de moteurs électriques... Prix : 30$.

FX-7 Flight Pak de *PARAGON TECHNOLOGIES*

Permet de préparer un vol, en s'aidant de la base de données contenant plus de 34000 coordonnées (aéroports, balises...), en Amériques du Nord et Centrale. Connecté à un modem (non fourni), il permet aussi de lire la base de données de la FAA, donnant des informations sur la météo. Prix : 186$, comprenant 6 mois de mises à jour gratuites de la base de donnée.

Smart Rounds de *MacroMed*

Destiné au milieu médical, ce logiciel gère une base de données concernant les patients. Permet aussi de déterminer environ 700 diagnostiques, par description ou par code ICD-9. Prix : 460$.

Jacques Belin (123)

GRANDS LANGAGES POUR PETITE MEMOIRE

Le but de cet article est de vous fournir quelques renseignements pratiques sur les logiciels de développement, dans le cadre d'un environnement tel que celui du HP-95 qui présente une très faible capacité de stockage de masse.

Bien que n'étant pas spécifiquement destiné aux utilisateurs du HP-71, il contient cependant quelques indications qui devraient leur permettre de migrer plus facilement sur le HP-95. Nous étudions cependant la possibilité de passer dans les prochains mois des articles exposant de façon détaillée les différences entre l'assembleur du Saturn et celui du 8086, par exemple, ou le Basic du HP-71 et les Basics tournant sur IBM-PC.

Quel langage choisir?

Un des choix principaux que le programmeur devra fatalement effectuer est celui du langage lui même. Il n'est pas obligatoire qu'un programmeur occasionnel ou utilisant principalement des routines de haut niveau s'attaque tout de suite à langage aussi complexe que le C. Il pourra en fait parfaitement trouver son bonheur dans le Basic le plus courant. Les utilisateurs du HP-71 utilisant les fonctions structurées de JPC Rom pouront facilement trouver un Basic évolué, comme le *TURBO BASIC* de Borland ou le *QUICK BASIC* de Microsoft. Il existe aussi un Basic (*Swift!BASIC*) spécialement adapté à l'environnement du HP-95.

Toujours pour les utilisateurs du HP-71, il existe aussi différents implémentations du langage Forth sur PC, dont au moins un adapté au HP-95 (*UTIL Forth*). Même si ces différentes versions peuvent présenter des différences notables en ce qui concerne le vocabulaire ou variables standard, ce langage présente l'avantage d'avoir une syntaxe de base standard, comme par exemple la définition des mots ou les accès mémoire.

Pour développer des applications plus puissantes, il faudra un jour ou l'autre passer à d'autres langages, comme le Pascal ou le C. Ce premier langage, bien connu dans les milieux étudiants, permet de développer des applications plus importantes, utilisant toutes les finesses de la programmation structurée. Si l'on veut aller encore plus loin, ou plus bas à l'intérieur du système d'exploitation, le C représente la puissance à l'état pur, qu'il vous faudra cependant apprendre à maitriser. C'est aussi l'environnement de développement le plus gourmand en ressources disques.

Faut il prendre un Interpréteur ou un Compilateur ?

La question de la vitesse d'exécution étant mise à part, ce choix peut être déterminant dans le cas d'une machine possédant peu de mémoire. En effet, il faut penser qu'un compilateur génère un fichier exécutable de grande taille (par rapport au source du programme), car il comporte les primitives des fonctions qui sont utilisées dans le programme. Si on a deux programmes relativement semblables, on aura donc un gaspillage, dû à un double stockage de ces primitives sur le disque.

Par contre, dans le cas d'un langage interprété, il existe un seul programme exécutable de grande taille car contenant l'ensemble des fonctions, et plusieurs fichiers sources, de plus petite taille.

Ceci expliqué, il devient compréhensible qu'il existe un seuil dans le nombre de programmes où la somme des tailles des programmes compilés devient supérieure à la somme de la taille des sources à laquelle on ajoute la taille de l'interpréteur. Suivant la taille des programmes, ce nombre se situe souvent entre 4 et 10.

Il est donc conseillé, dans le cas où l'on dispose de très peu de mémoire, mais où l'on utilise un grand nombre de petis programmes, de se servir d'un interpréteur. Ceci bien entendu au détriment de la vitesse d'exécution.

Il existe cepandant une solution intermédiaire, donnée par exemple par les langages Borland ou le *Quick Basic*. Ces compilateurs peuvent se comporter comme des interpréteurs, dans le sens où ils peuvent compiler un programme en langage machine et l'exécuter immédiatement, sans avoir à créer un fichier .EXE ou .COM. La principale contrainte est que l'on doit recompiler le programme à chaque fois que l'on veut l'utiliser. De plus, le compilateur étant chargé en mémoire vive pendant l'exécution du programme, c'est autant de mémoire perdue pour celui-ci.

Faut-il développer sur le PC ou sur le HP-95 ?

Il semble que la plupart des utilisateurs de HP-95 possèdent déjà un IBM-PC.

Dans ce cas la meilleure solution est de garder l'environnement de développement sur le PC et de ne transférer que les exécutables sur le HP-95.

Cette organisation permet d'économiser la mémoire du HP-95, puisqu'on n'y met pas de compilateur, mais surtout d'utiliser à plein les capacités de certains environnements de développement. A savoir : éditeur plein écran, choix dans les modèles de librairies, analyseurs de performances...

Il est même possible, avec certains logiciels (le *Turbo Debugger* par exemple), de controler l'exécution pas-à-pas d'un programme tournant sur le HP-95, à partir du debugger chargé sur le PC, les deux étant reliés par la liaison série. Ce qui permet principalement de tester l'utilisation des fonctions spécifiques de la Rom du HP-95, chose qu'il impossible de faire si l'on teste le programme sur le PC. La seule petite contrainte est que l'on doit charger sur le HP-95 un petit résident d'environ 20 ko.

Si l'on ne possède pas de PC, ou si l'on doit continuer le développement loin de sa base, il est cependant possible d'installer son environnement de développement sur le HP-95. Les techniques de développement deviennent alors les mêmes que pour le développement normal sur PC. Le seul problème est dû à la taille de cet environnement, qui peut en version complète dépasser le taille des cartes mémoires disponibles actuellement.

De plus, pour certains compilateurs, il est nécessaire d'ajouter un éditeur de texte afin de modifier les fichiers sources. En effet, si le HP-95 en possède un tout à fait convenable, il devient très vite contraignant, en phase de test, d'éditer le programme avec l'éditeur intégré, fermer l'éditeur, passer sous le Filler, lancer le COMMAND.COM, compiler le programme, l'exécuter, (s'apercevoir qu'il y a une erreur), sortir du DOS par `exit`, relancer l'éditeur, et ainsi de suite. Il est donc beaucoup plus agréable d'avoir un éditeur de texte directement accessible à partir du DOS.

Une solution consiste à utiliser les compilateurs livrés avec un environnement intégré permettant de lancer la compilation et l'exécution directement à partir de l'éditeur (*Turbo C...*). Le confort d'utilisation est évident, mais il faut penser au fait qu'une grande partie cet environnement reste en mémoire pendant la phase de compilation. Ceci peut limiter la taille du programme à compiler, surtout si la taille ou le nombre des fichiers ouverts simultanément sont importants.

Ce qui implique, dans le cas du HP-95 équipé de 512 Ko de mémoire, qu'il faudra configurer l'espace mémoire (par rapport à la taille du disque C) de façon la plus large possible.

Et si nous parlions Chiffres ?

Après avoir choisi un langage, il est temps maintenant de savoir quels sont les fichiers vraiment utiles que l'on va mettre dans son HP-95. En effet, connaissant la faible contenance des cartes mémoires actuelles, il est illusoire d'espérer installer la totalité d'un système de développement. A titre d'exemple, le *TURBO C++* occupe à lui seul plus de 4 Mo, si on y inclue les librairies corespondant à tous les modèles mémoire... Ceci sans compter les autres utilitaires de développement (*Turbo Assembler*, *Turbo Debugger*...).

J'ai donc essayé de déterminer, pour chaque logiciel, quels étaient les fichiers indispensables à leur bon fonctionnement. Certains logiciels étant inclus en totalité dans un seul exécutable, il n'y a aucun problème. Par contre, d'autres logiciels sont divisés en plusieurs types de fichiers. Dans ce cas, mon but

était de déterminer quelle était la configuration minimale permettant de développer à peu près n'importe quel logiciel, adapté au HP-95.

J'ai laissé de côté, pour les logiciels permettant de faire ce choix, les options permettant de générer du code adapté au 8087 (non inclus dans la machine) ou la gestion des modes graphiques, incompatibles actuellement avec le mode utilisé par le HP-95. Je n'ai pas inclus non plus certains éléments, tels que les fichiers d'aides intégrés à certains langages. Pour plus de détails, se reporter au langage correspondant.

Les tailles mentionnées correspondent à l'encombrement *réel* des fichiers, tenant compte de la dimension des clusters de 512 octets. Vous aurez donc ainsi une indication précise de la place qui sera occupée dans votre carte mémoire. Pour les personnes utilisant un logiciel de compression des fichiers exécutables, j'ai noté la valeur obtenue avec PKLITE. D'autres logiciels de ce type peuvent donner des valeurs différentes, mais qui devraient être relativement proches.

Comme vous pourrez le voir, les versions successives d'un même logiciel ont toujours tendance à voir leur taille augmenter, souvent de façon fort conséquente. J'ai donc inclus les encombrements nécessaires à chaque version, pour vous permettre de faire votre choix, en fonction de la taille de votre carte mémoire.

Notons enfin que cette liste ne se veut pas exhaustive, elle comprend seulement les logiciels que j'ai pu avoir à disposition. Si certains logiciels ne sont pas mentionnés, ce n'est que parce que je n'ai pas pu les tester. Il est donc fort possible que j'ai laissé ainsi de côté certains logiciels (ou versions de logiciels), alors qu'ils auraient pû être plus appropriés que ceux que j'ai cité. D'autre part, cette liste ne comprend que certains langages : Assembleur, Basic, Pascal, Forth et C. J'ai laissé arbitrairement de côté certains autres logiciels (Fortran, Prolog, Lisp...) car je pense qu'ils seront moins utilisés, alors que le Forth aura, je pense, la préférence des anciens utilisateurs de HP-71. Je n'ai pas cité non plus les logiciels spécialement développés pour le HP-95 (*UTIL FORTH*, *Swift!BASIC*...) car, à moins qu'un comparatif détaillé ne le contredise, il sera conseillé de les utiliser à la place des logiciels de ce panorama, car ils tiennent mieux compte des possibilités spécifiques du HP-95.

1- Les Assembleurs

D'une structure simple, ces logiciels sont divisés classiquement en deux exécutables : un assembleur, générant un fichier objet à partir d'un fichier source, et un "linker" générant le fichier exécutable à partir d'un ou plusieurs fichiers objets.

Microsoft Assembler (version 4.00) :

Fichiers : MASM.EXE LINK.EXE
Taille : 130048 octets (79360 avec PKLITE)

Note : LINK.EXE est aussi utilisé par le C Microsoft. Si vous désirez instaler les deux logiciels en même temps, vous pouvez gagner un peu de place en ne copiant qu'une seule fois ce fichier. Ceci n'est pas valable pour le Turbo C, car le linker est inclus dans les fichiers TC.EXE et TCC.EXE.

Turbo Assembler :

Version 1 :

Fichiers : TASM.EXE TLINK.EXE
Taille : 120320 octets (79872 avec PKLITE)

pour la version 2, nous avons les mêmes fichiers que précédement, mais ceux-ci possèdent une taille légèrement supérieure :

Taille : 159744 octets (106496 avec PKLITE)

2- Les Basics.

GWBASIC

Livré en standard avec MS-DOS, ce logiciel n'est cependant pas inclus dans la Rom du HP-95. Pourtant, il semble fonctionner parfaitement sur cette machine. Je dis semble, car il est possible (bien que cela ne soit pas trouvé), que ce logiciel, livré rapelons-le en standard avec le DOS, utilise certaines structures secrètes ou possiblités inhérentes de chaque version du système d'exploitation. Il est donc conseillé d'utiliser une version du programme correspondant au Dos 3.2.

Je vous donne cependant les données correspondant à la version 3.23 (fournie avec le DOS 4.01) que j'ai testé sur le HP-95 :

Fichier : GWBASIC.EXE
Taille : 80960 octets (65024 avec PKLITE)

Note : Je n'ai pas testé le BASICA livré avec certaines versions du DOS, sur le HP-95. Sachant qu'il pose des problèmes sur de nombreuses machines (plantage "dur" par exemple), je conseille la plus extrême prudence quant à son chargement sur le HP-95.

Fichier : GWBASIC.EXE
Taille : 80960 octets (65024 avec PKLITE)

QBASIC :

Remplaçant le GWBASIC dans la version 5 de MS-DOS, cet interpréteur comporte un environnement intégré similaire au QUICK BASIC. Il s'agit ici d'un Basic structuré, qui aura sûrement plus les faveurs des utilsateurs du HP-71 possédant JPC Rom et ses fonctions de programmation structurée. N'ayant pu tester ce programme que sur un IBM-PC chargé avec un DOS 3.10, et non sur le HP-95, je confirme qu'il fonctionne correctement, avec les réserves exprimées pour le GWBASIC.

Fichier : QBASIC.EXE
Taille : 255871 octets (193367 avec PKLITE)

Turbo Basic

Apparu il y a quelques années déjà, ce programme ne comporte pas les réserves précédente. De plus, il s'agit d'un compilateur, ce qui ne gache rien.

Fichiers : TB.EXE TBCONFIG.TB
Taille : 205824 octets (130560 avec PKLITE)

Note : Afin de gagner de la place, je n'ai pas inclus le fichier d'aide en ligne TBHELP.TBH, permettant d'obtenir des informations sur les fonctions et l'utilisation du compilateur, après une pression de la touche F1. Si vous désirez cette aide, cela vous en coûtera environ 41000 octets en plus.

3- Le Forth

Il existe un Forth (*UTIL FORTH*, spécialement étudié pour le HP-95. Ne l'ayant pas eu entre les mains, je n'en parlerais pas aujourd'hui.

MVPFORTH

Seul logiciel du domaine public de ce panorama, c'est aussi celui qui prendra le moins de place dans la mémoire de votre machine. Il s'agit d'un Forth tout a fait classique, qui devrait suffire pour toutes les applications standards.

Fichier : MVPFORTH.COM
Taille : 28032 octets (18294 avec PKLITE)

4- Le Pascal

Je ne sais pas si beaucoup d'utilisateurs utiliseront ce langage, qui est de plus en plus souvent supplanté par le C, mais je vous donne tout de même quelques indications.

Turbo Pascal version 3.01 :

Fichiers : TURBO.COM TURBO.MSG
Taille : 41472 octets (34304 avec PKLITE)

Fait intéressant, vous pouvez noter que ce compilateur possède une taille largement inférieure au *GWBASIC*.

5- Les compilateurs C

N'envisagez ce langage que si vous possédez une carte mémoire de 1, voire 2 Mo. Mais à ce prix là vous aurez la possibilité de développer à peu près n'importe quel type de logiciel.

Pour tous ces compilateurs, j'ai retenu la configuration suivante :
- Génération du modèle de mémoire "Small".
- Bibliothèque d'émulation du 8087.
- Pas de Librairie graphique.

Turbo C.

Ce logiciel existe en de nombreuses déclinaisons : en C standard ou C++, avec l'environnement intégré ou en mode ligne de commande. Vous pourrez constater que suivant la configuration ou la version, la taille nécessaire augmente d'un rapport de 1 à 3. Notez que pour inclure de l'assembleur "en ligne", à partir de la version 2, vous devrez ajouter TASM.EXE à votre environnement.

Comme pour le Turbo Basic, le fichier d'aide en ligne TCHELP.TCH n'a pas été intégré.

Turbo C version 1 et 2

- En configuration ligne de commande :

fichiers : TCC.EXE TCCONFIG.EXE

directory : \INCLUDE et \INCLUDE.SYS
Tous fichiers, excepté GRAPHICS.H (version 2)

directory : \INCLUDE\SYS
tous fichiers

directory : \LIB
CS.LIB COS.OBJ MATHS.LIB EMU.LIB

Version 1 :
Taille : 343552 octets (256512 avec PKLITE)
Version 2 :
Taille : 398848 octets (308224 avec PKLITE)

- En Configuration Editeur intégré :

Les mêmes fichiers que précédement. Seul TC.EXE remplace TCC.EXE.

Version 1 :
Taille : 405504 octets (295424 avec PKLITE)
Version 2 :
Taille : 508928 octets (381440 avec PKLITE)

Turbo C++ version 1.01.

- En utilisation C classique, en mode ligne de commande :

directory : \BIN
TCC.EXE TURBOC.CFG TCDEF.DPR TCDEF.DSK TCCONFIG.TC

directory : \LIB
CS.LIB EMU.LIB MATHS.LIB COS.OBJ WILDARGS.OBJ

directory : \INCLUDE

ALLOC.H	ASSERT.H	BIOS.H	CONIO.H	CTYPE.H
DIR.H	DOS.H	ERRNO.H	FCNTL.H	FLOAT.H
IO.H	LIMITS.H	LOCALE.H	MATH.H	MEM.H
PROCESS.H	SETJMP.H	SIGNAL.H	STDARG.H	STDDEF.H
STDIO.H	STDLIB.H	STRING.H	TIME.H	VALUES.H

directory : \INCLUDE\SYS
Tous fichiers.

Taille : 759296 octets (711680 avec PKLITE)

- En mode C classique, avec l'environement integré :

Ce sont les mêmes fichiers, seul TC.EXE remplace TCC.EXE.

Taille : 1181184 (1104384 avec PKLITE)

Note : Pour fonctionner correctement, avec cette version de l'environnement intégré, il est nécessaire de posséder un HP-95 version 1 Mo, car si on peut charger l'environnement intégré sur la version 512 Ko, on aura de grandes chances d'obtenir un message Out of memory lorsque la compilation sera lancée. Si cela arrive, fermez le maximum de fichiers sources, cela devrait libérer suffisement de mémoire.

- En utilisation mode C++, en mode ligne de commande :

directory : \BIN
TCC.EXE TURBOC.CFG TCDEF.DPR TCDEF.DSK TCCONFIG.TC

directory : \LIB
CS.LIB EMU.LIB MATHS.LIB OLDSTRMS.LIB COS.OBJ WILDARGS.OBJ

directory : \INCLUDE
tous fichiers excepté GRAPHICS.H

directory : \INCLUDE\SYS
Tous fichiers

directory : \CLASSLIB\INCLUDE
tous fichiers

directory : \CLASSLIB\LIB
TCLASSS.LIB

Taille : 1036800 octets (989184 avec PKLITE)

- En utilisation C++, avec l'environement integré :

Ce sont les mêmes fichiers, seul TC.EXE remplace TCC.EXE.

Taille : 1458224 octets (1381576 avec PKLITE)

Vous pouvez noter que pour le Turbo C++, les taux de compression des fichiers TCC.EXE et TC.EXE sont très faibles, de l'ordre de 8 ou 10%, contre 30 à 50% pour la plupart des exécutables.

Note pour les version 2 et C++ : Si un driver BGI, gérant l'affichage graphique devait voir le jour pour le HP-95, il faudrait ajouter les fichiers GRAPHICS.H et GRAPHICS.LIB. coût : environ 42000 octets plus la taille du driver.

Microsoft C (version 5.10)

Comme pour le *TURBO C*, il existe deux configurations pour ce langage : En mode ligne de commande ou avec l'environnement integré. Contrairement au logiciel précédent, ce langage est divisé en plusieurs programmes, dont un Linker qui peut aussi servir après l'utilisation de l'assembleur.

- Version ligne de commande :

directory : \BIN
C1.EXE C2.EXE C3.EXE CL.EXE LINK.EXE
C1.ERR C23.ERR CL.ERR

directory : \LIB
SLIBCE.LIB SLIBFP.LIB EM.LIB LIBH.LIB SVARSTCK.OBJ
SETARGV.OBJ

directory : \INCLUDE et \INCLUDE.SYS
Tous fichiers, sauf GRAPH.H.

Taille : 842752 octets (630784 avec PKLITE)

- Version Environnement Intégré (*Quick C*):

Contrairement aux autres compilateurs, le modèle de mémoire MEDIUM est imposé pour ce programme.

directory : \BIN
QC.EXE C1.ERR C23.ERR LINK.EXE

directory : \LIB
MLIBCE.LIB MLIBFP.LIB MVARSTCK.OBJ QUICKLIB.OBJ
EM.LIB LIBH.LIB SETARGV.OBJ

directory : \INCLUDE et \INCLUDE\SYS
Tous fichiers, sauf GRAPH.H.

Total : 701952 octets (550912 avec (PKLITE)



Comme vous le voyez, cette liste est suffisement vaste pour que vous puissiez y trouver un langage et une configuration qui devrait vous convenir. C'est à vous de choisir maintenant votre logiciel à partir des éléments que je vous ai donné, mais en tenant compte d'un paramêtre qui n'a rien a voir avec la technique, c'est celui du prix de ces logiciels. Sachez seulement qu'il a une bizarre tendance à être proportionnel aux tailles mentionnées dans cet article...

Jacques Belin (123)



VOYAGE AU CENTRE DE LA MEMOIRE : LE RETOUR

Dans l'article *Voyage au centre de la mémoire*, paru le mois dernier, une erreur de frappe s'est glissée dans la réference de l'adresse de la copie de la RAM Vidéo. En effet, il faut lire 0100:0000 au lieu de 1000:0000.

Jacques Belin (123)

MS-DOS

J. Belin	Utilitaires de conversion pour ThinkJet	*36*
J. Belin	Une bogue dans LIFCP	*39*

DU GOTHIQUE AU ROMAN8

La carte HP-LINK, procurant une interface HP-IL à l'IBM PC, permet de rediriger les commandes d'impression vers une imprimante connectée à la boucle, ou l'accès au lecteur 9114, pour lire des disquettes MS-DOS. Ceci est possible si vous avez déclaré un des drivers, livrés avec la carte, dans votre fichier CONFIG.SYS.

Or ces drivers présentent quelques inconvénients, en ce qui concerne la partie impression, parmi lesquels on peut noter le fait que la redirection ne s'effectue pas avec certains logiciels évolués comme Windows ou qu'il est impossible d'effectuer des impressions d'écran. Enfin les caractères accentués de l'IBM, ne sont pas reconnus par la ThinkJet, qui travaille avec le jeu de caractères Roman8.

Le présent article a donc pour but de résoudre ce dernier problème et de vous faire comprendre les raisons du non-fonctionnement sur les logiciels évolués. En ce qui concerne les impressions d'écran, le problème devrait être résolu pour un prochain numéro.

Commençons donc par essayer de comprendre le fonctionnement du driver, face aux logiciels et aux fonctions prévues par le BIOS.

Pour envoyer des caractères à l'imprimante, ou à tout autre périphérique standard, la méthode consiste à charger les registres du microprocesseur avec les données appropriées et à exécuter une interruption spécialement prévue à cet effet, qui va se charger d'exécuter les fonctions du plus bas niveau. Sur un IBM-PC tournant sous MS-DOS, ces fonctions sont accessibles à deux niveaux différents : grâce à une interruption BIOS ou par différentes fonctions accessibles grâce à l'interruption 21 du DOS. L'interruption BIOS, portant le numéro 17 (en hexadécimal), accédant au plus bas niveau, permet d'imprimer sur les périphériques reconnus sous les nom LPT1 à LPT3. Les fonctions DOS permettent quant à elles d'effectuer les mêmes sorties, mais à un niveau plus haut, permettant de s'affranchir des contraintes matérielles et par exemple rediriger la sortie de l'imprimante vers n'importe quel port série ou parallèle.

Pour cela MS-DOS crée, dès son chargement, un driver de périphérique, nommé PRN, en même temps que d'autre drivers qui se chargent par exemple de la liaison de l'ordinateur avec le clavier ou l'ecran.

Ce périphérique est activé à chaque fois que l'on effectue, par exemple, la commande :

```
DIR >prn
```

qui redirige les résultats de l'instruction `DIR` vers l'imprimante.

Il est bien sûr possible de l'activer, à partir d'un programme. Par exemple, le flux standard `stdprn` du C permet d'effectuer les sorties sur le périphérique PRN.

Parmi les drivers fournis avec la carte HP-LINK, le driver HPIL_PRN.SYS est conçu pour remplacer le driver d'origine. Une fois chargé, tous les appels destinés à PRN sont donc pris en charge par ce driver et transmis sur la boucle, à la première imprimante s'y trouvant.

Or si les redirections ou les requètes d'impression provenant des programmes utilisant les accès à `stdprn` sont correctement envoyés sur l'HP-IL, certains programmes ne fonctionnent pas. Une observation rapide du vecteur de l'interruption 17, qui contient l'adresse de la première instruction devant être exécutée à la suite de l'appel de cette interruption, indique que le vecteur pointe toujours à l'intérieur du BIOS, et n'a donc pas été redirigé vers le driver. Ceci implique que tous les programmes imprimant les données par un appel à l'interruption 17 ignoreront automatiquement la présence du driver PRN, qu'il soit HPIL ou non.

Il serait tentant d'intercepter l'interruption 17 pour la rediriger vers une interruption DOS reconnaissant le driver, mais cela est fortement déconseillé, car il y a un grand risque de plantage dû à une tentative de réentrance vers le DOS.

Le cas du traitement de l'interruption 17 ne semblant pas résolvable, il est donc laissé de côté. Il est par contre possible de créer un programme résident qui interceptera les données envoyées par les programmes qui utilisent le flux `stdprn`. En effet, les bibliothèques de ces programmes transmettent ces données au DOS en utilisant des interruptions. Il est donc possible d'intercepter "au vol" les données jointes à ces interruptions, les convertir et les laisser repartir *quasiment* comme elles étaient arrivées. Le fonctionnement détaillé de cet utilitaire sera expliqué plus bas.

Or, une fois cet utilitaire développé, une autre mauvaise surprise nous attend. En effet, si les conversions sont bien effectuées à partir des programmes, il est impossible de convertir

directement un fichier en effectuant une commande du type :

```
type file.txt >prn
```

De la même façon, la commande PRINT de MS-DOS permet bien d'imprimer le fichier sur la ThinkJet, mais sans effectuer les conversions.

Ceci est probablement dû au fait que les accès au périphériques se font, pour ces fonctions, par un accès direct au driver, sans utiliser les appels d'interruptions.

Pour palier à ce problème, la solution serait d'intervenir à l'intérieur du driver, qui contient en son début une structure de données équivalente à la table des vecteurs d'interruption située au début de la mémoire vive. L'initialisation et le fonctionnement du nouveau résident seraient quasiment identique à celui expliqué plus bas. Il existe cependant un problème, dû à l'architecture INTEL, qui est que seul une partie du pointeur (l'offset) est stockée dans cette structure. Ceci veut dire que résident devra être placé dans la limite du segment occupé par le driver, c'est à dire moins de 64Ko après le début de celui-ci. Cela semble un petite contrainte, mais il n'est absolument pas garanti (surtout avec le DOS 5 qui peut charger des résidents en mémoire haute) que le DOS charge le résident dans cette limite.

Si nous ne pouvons pas utiliser de méthode automatique pour imprimer correctement des fichiers à partir de la ligne de commande, il reste une solution (que j'utilise depuis déjà longtemps) qui consiste à rediriger la sortie des utilitaires vers un filtre avant de la rediriger vers le périphérique prn. Cette méthode sera expliquée à la fin de cet article, avec l'utilitaire correspondant.

Les tables de conversions

Avant de décrire ces utilitaires, il est nécessaire de choisir les jeux de caractères qui vont nous servir de base de travail pour la conversion. En ce qui concerne le jeu qui sera utilisé en sortie, cela ne pose pas de problème, puisque le Roman8 ne possède pas de variantes. Par contre, il est possible de charger plusieurs jeux de caractères nationaux sur l'IBM PC, grâce à certains paramètres du fichier CONFIG.SYS. Le jeu PC-8 (chargé par défaut) est le plus courament utilisé, car il comprend la totalité des caractères semi-graphiques. Or, il existe un jeu de caractères, nommé PC-850, qui est beaucoup plus proche du Roman8. En effet, on peut trouver, à la place de certains caractères semi-graphiques secondaires, des caractères accentués qui existent aussi en Roman8. Les caractères communs au PC-8 et au PC-850 se situent à la même place, à l'exception du *cent* (¢) et du *Yen* (¥).

Les tables utilisées dans les utilitaires suivants effectuent donc la conversion du PC-850 vers le Roman8. Les caractères PC-850 qui n'existent pas dans l'autre jeu seront convertis en un point d'interrogation ?. De plus certains caractères semi-graphiques qui n'ont pas d'équivalent direct en Roman8 seront convertis en caractères qui leur sont le plus proche. Cela ne devrait pas poser de problème, à l'exception du fait que la conversion n'est pas totalement inversible.

LES UTILITAIRES

Conversion par un utilitaire résident

La première méthode de conversion, qui est la plus transparente pour les programmes et l'utilisateur consiste à installer un programme résident, qui va intercepter les sorties destinées à l'imprimante afin de les convertir.

Cet utilitaire fonctionne correctement avec tous les programmes utilisant les flux standards pour les sorties vers l'imprimante. Pour savoir s'il fonctionnera avec l'option d'impression de votre éditeur favori, il vous suffit d'essayer d'imprimer un fichier. Si l'impression fonctionne (sans afficher les caractères accentués bien sûr !) l'utilitaire devrait assumer sa tâche sans problème.

Comme nous l'avons vu précédemment, le driver ne gérant pas l'interruption du BIOS, il faudra travailler sur les fonctions de l'interruption du DOS (numéro 21). Ces fonctions permettent d'écrire sur le flux stdprn, à savoir la numéro 5, dédiée à l'imprimante et la numéro 44, écrivant dans un flux spécifié par un handle (en ce qui nous concerne, le numéro 4, affecté de façon standard à stdprn). Enfin une troisième méthode d'écriture est interceptée, qui consiste à accéder directement au driver de périphérique par l'intermédiaire d'une fonction IOCTL. La gestion de ces trois méthodes d'écritures devrait permettre de tenir compte du plus grand nombre de logiciels possible.

Pratiquement, le logiciel fonctionne de la façon suivante :

Lancement : TJR8

Au premier appel, le programme lit le vecteur de l'interruption 21, situé dans la table placée dans les addresses les plus basses de la mémoire vive. Ce vecteur, contient l'adresse de la première instruction devant être exécutée à la suite de l'appel de

l'interruption. Ceci fait le programme lit les deux octets situées juste avant cette adresse. Si ces deux octets correspondent à une valeur prédéfinie, cela veut dire que le résident est déjà chargé. Dans ce cas on en avertit l'utilisateur et on termine le programme de façon classique. Si le résident n'est pas déjà chargé, on stocke le vecteur dans la zone des données du programme et on le remplace par l'adresse de la première instruction de traitement de notre programme. Une fois ceci fait, et après affichage de quelques lignes de message, le programme retourne au DOS en restant résident, mais en prenant soin de laisser le système d'exploitation réutiliser à sa convenance la portion du programme qui a servi à installer la partie résidente, ainsi que le texte des messages.

Nous avons donc maintenant en mémoire uniquement la partie résidente du programme. Si à partir de ce moment une interruption 21 est exécutée, elle activera le code de notre programme. Sa première tâche consistera à déterminer si l'appel de l'interruption le concerne, en testant les valeurs des registres du microprocesseur afin d'y trouver les codes standards des différentes méthodes d'écriture vers l'imprimante. Si ces codes ne correspondent pas, le programme effectue un saut vers la routine qui était précédemment pointée par le vecteur d'interruption, avant l'installation de notre programme. Cet ancien vecteur est, rapellons le, stocké dans la zone de données du programme.

Par contre, si les fonctions interceptées correspondent à celles qui nous intéressent, nous pouvons effectuer la conversion des caractères. Ceci se fait par deux méthodes, suivant la fonction qui a été interceptée. Dans le cas de la fonction 5, un seul caractère est traité par appel de l'interruption. Si le code de ce caractère est supérieur à 127, il est remplacé par son correspondant dans la table de conversion, puis le programme est transféré à l'ancienne interruption par la même méthode que celle précédement expliquée. Il faudra prendre soin de restaurer les registres utilisés, afin d'éviter un plantagedu système. Dans le cas des fonctions Handle et IOCTL, le déroulement est le même, excepté que l'on traitera plusieurs caractères au cours du même appel d'interruption.

Si par bonheur vous possédez un programme fonctionnant parfaitement avec ces utilitaires, mais reconnaissant en plus la ThinkJet en temps qu'imprimante graphique, il vous faudra malheureusement désactiver le résident pour ne pas envoyer de mauvaises données pendant l'impression des graphiques. Pour cela, effectuez la commande suivante :

```
tjr8 U
```

Ceci lancera une deuxième fois le programme. Nous aurons donc à ce moment en mémoire la partie résidente du premier programme *plus*, plus loin, la partie complète du programme que nous venons de lancer. La partie complète va d'abord lire le vecteur d'interruption et chercher les deux octets de signature indiquant que le programme est effectivement chargé. Si les octets sont trouvés, on replacera d'abord l'ancien vecteur d'interruption dans la table des vecteurs, puis on dira au DOS de libérer la mémoire utilisée par la partie résidente. Enfin, la deuxième partie du programme se terminera de façon classique.

Cette méthode d'installation et de désinstallation de résident, relativement simple à mettre en oeuvre représente un petit inconvénient. En effet il n'est pas possible de le décharger si un autre résident utilisant l'interruption 21 a été installé entre-temps, car le deuxième programme ne trouvera pas les octets de signature de notre programme, mais ceux de l'autre résident.

Conversion par un filtre

Le résident précédement décrit ne permettant pas d'intercepter les redirections vers `prn`, il faut passer par une méthode plus simple, qui consiste à intercaler un filtre entre la sortie de la commande précédente et la redirection.

Ce filtre ne fait que lire les caractères dans son flux d'entrée (qui est lié au flux de sortie de la commande précédente), les convertit suivant un tableau de correspondance, puis les renvoie sur son flux de sortie (qui est lié aux flux d'entrée de la commande suivante).

Appelé `ibmtor8`, ce filtre peut s'utiliser de la façon suivante pour imprimer un fichier:

```
type file.txt | ibmtor8 >prn
```

Il comprend une option, `-t`, qui peut convertir les tabulations en espaces.

Pour ma part, je l'ai inclus dans un fichier batch, PRINT.BAT, qui se charge aussi d'initialiser la ThinkJet avec l'option de sauts de perforations, en tenant compte de la longueur de la feuille et le nombre de ligne de textes. Après l'impression, la commande `FF` permet d'éjecter la feuille grâce à un Form Feed.

En voici le Listing :

```
initjet 66 60
ibmtor8 <%1 >prn
ff
```

Vous trouverez les sources de ces commandes après celui de TJR8. J'y ai aussi inclus le fichier r8_pc850.h, qui contient la table de conversion inverse à celle qui est utilisée par ibmtor8. En remplaçant les deux occurences de pc850_r8 dans ce listing, vous obtiendrez la commande inverse que vous pourrez appeler, par exemple, r8toibm.

Jacques Belin (123)

COPIE NON CONFORME

Je vous ai présenté, dans un dernier *JPC*, une série d'utilitaires permettant de lire des disquettes LIF sur un IBM PC.

Or, je viens seulement de découvrir une bogue dans la commande LIFCP. En effet, lorsque l'on veut remettre sur une disquette LIF un fichier précédemment stocké sur MS-DOS avec l'option -b, la copie s'effectue correctement, si ce n'est un petit détail...

En effet, si on veut (ce qui paraît logique) lire ultérieurement ce fichier, on récupèrera des données appartenant à d'autres fichiers. Pire, si l'on effectue une autre copie sur ce même fichier, les données correspondantes auront de grandes chances d'écraser une partie de ces autres fichiers.

Ceci est dû, à une particularité des fichiers MS-DOS créés par la copie d'un fichier LIF à l'aide de l'option -b. En effet, ceux-ci contiennent la copie *exacte et intégrale* de l'entrée du directory contenant le fichier LIF. Ce qui veut dire qu'il contient aussi le numéro du premier secteur contenant les données sur la disquette LIF originale. Or, lorsque l'on réutilise cette option pour recopier le fichier sur une autre disquette LIF, les données du fichier sont bien placées à une place convenable (non utilisée par un autre fichier), mais cette position n'est pas mise à jour dans la nouvelle entrée de directory (qui est elle aussi correctement placée). La nouvelle disquette contient donc maintenant une entrée de directory contenant une position de fichier correspondant à une autre disquette, entraînant les conséquences citées plus haut.

Pour corriger cette grossière erreur, dans le source de LIFCP.C, il suffit de stocker, avant l'écriture de l'entrée du directory sur la disquette, la valeur first_sect issue de la fonction best_room (cherchant un emplacement pour le fichier sur la disquette cible) dans l'entrée de directory référencée par dir. La meilleure méthode consiste à placer une instruction build_dir, réutilisant la majeure partie des données issues de l'instruction break_dir utilisée peu avant.

Cette instruction aura la forme suivante :

```
build_dir(&dir, lif_name, private, secure, type,
          first_sect, long_file, &gdh, imp1, imp2);
```

et sera placée à la fin de la partie de la copie en mode binaire de fichiers MS/DOS->LIF, à la place de la commande memcpy(&dir,lif_name,10), qui devient inutile. Dans le *JPC 76*, vous trouverez cette instruction dans les toutes dernières lignes de la deuxième colonne de la page 28.

D'autre part, et indépendamment de cette bogue, il est conseillé de remplacer les tables de conversions de caractères IBM<->Roman8 par un accès aux fichiers r8_pc850.h et pc850_r8.h présentés par ailleurs dans ce journal. Pour cela il suffit d'appeler ces fichiers à l'aide d'un #include et de remplacer la table par une ligne du type :

```
unsigned char *ibm_r8=pc850_r8;
```

Cette ligne permettra d'éviter (si on est paresseux !) d'avoir à rechercher dans le source la ligne effectuant la conversion et de remplacer ibm_r8 par pc850_r8 !

Cette modification permet d'utiliser les tables correspondant au jeu de caractères PC-850, qui comporte plus de caractères communs avec le jeu Roman-8. De plus, l'utilisation de ces tables sous forme de fichier ".h" permet de simplifier la mise à jour et la recompilation des programmes utilisant ces tables, si celles-ci doivent être modifiées.

Jacques Belin (123)



```
;*****************************************************************************
;*****************************************************************************
;**                                T J R 8                                  **
;**  Conversion en Roman8 des caractères imprimés sur ThinkJet Par HP-LINK  **
;**                                                                         **
;**    Création le 10/06/92              Auteur : Jacques Belin             **
;**                                   (d'après PRCO de Michael Tischer)     **
;**    Dernière modif.: 03/07/92                                            **
;**        Modification de la désinstallation : Ajout de l'option "U"       **
;*****************************************************************************
;*****************************************************************************

;== Ici commence le programme véritable ================================

code      segment para 'CODE'     ;Définition du segment de CODE

          org  100h

          assume cs:code, ds:code, es:code, ss:code

start:    jmp   prcoini           ;La première instruction exécutée

;== Données (restent en mémoire) ======================================

ancint    equ  this dword         ;Ancien vecteur d'interruption 21h
ofsancint dw   (?)                ;Adresse d'offset vecteur d'interruption 21h
segancint dw   (?)                ;Adresse de segment vecteur d'interr. 21h

          ;-- Table de conversion PC850->Roman8 ----------------------

codetab   label byte
          db   180                ;code 128
          db   207                ;code 129
          db   197                ;code 130
          db   192                ;code 131
          db   204                ;code 132
          db   200                ;code 133
          db   212                ;code 134
          db   181                ;code 135
          db   193                ;code 136
          db   205                ;code 137
          db   201                ;code 138
          db   221                ;code 139
          db   209                ;code 140
          db   217                ;code 141
          db   216                ;code 142
          db   208                ;code 143
          db   220                ;code 144
          db   215                ;code 145
          db   211                ;code 146
          db   194                ;code 147
          db   206                ;code 148
          db   202                ;code 149
          db   195                ;code 150
          db   203                ;code 151
          db   239                ;code 152
          db   218                ;code 153
          db   219                ;code 154
          db   214                ;code 155
          db   187                ;code 156
          db   210                ;code 157
          db   '?'                ;code 158
          db   190                ;code 159
          db   196                ;code 160
          db   213                ;code 161
          db   198                ;code 162
          db   199                ;code 163
          db   183                ;code 164
          db   182                ;code 165
          db   249                ;code 166
          db   250                ;code 167
          db   185                ;code 168
          db   '?'                ;code 169
          db   '?'                ;code 170
          db   248                ;code 171
          db   247                ;code 172
          db   184                ;code 173
          db   251                ;code 174
          db   253                ;code 175
          db   '?'                ;code 176
          db   127                ;code 177
          db   '?'                ;code 178
          db   '|'                ;code 179
          db   '|'                ;code 180
          db   224                ;code 181
          db   162                ;code 182
          db   161                ;code 183
          db   '?'                ;code 184
          db   '?'                ;code 185
          db   '?'                ;code 186
          db   '?'                ;code 187
          db   '?'                ;code 188
          db   191                ;code 189
          db   188                ;code 190
          db   '?'                ;code 191
          db   '?'                ;code 192
          db   '?'                ;code 193
          db   '?'                ;code 194
          db   '|'                ;code 195
          db   246                ;code 196
          db   '+'                ;code 197
          db   226                ;code 198
          db   225                ;code 199
          db   '?'                ;code 200
          db   '?'                ;code 201
          db   '?'                ;code 202
          db   '?'                ;code 203
          db   '?'                ;code 204
          db   '='                ;code 205
          db   '?'                ;code 206
          db   186                ;code 207
          db   228                ;code 208
          db   227                ;code 209
          db   164                ;code 210
          db   165                ;code 211
          db   163                ;code 212
          db   '?'                ;code 213
          db   229                ;code 214
          db   166                ;code 215
          db   167                ;code 216
          db   '?'                ;code 217
          db   '?'                ;code 218
          db   '?'                ;code 219
          db   '?'                ;code 220
          db   124                ;code 221
          db   230                ;code 222
          db   '?'                ;code 223
          db   231                ;code 224
          db   222                ;code 225
          db   223                ;code 226
          db   232                ;code 227
          db   234                ;code 228
```

```
        db   233                ;code 229
        db   243                ;code 230
        db   241                ;code 231
        db   240                ;code 232
        db   237                ;code 233
        db   174                ;code 234
        db   173                ;code 235
        db   178                ;code 236
        db   177                ;code 237
        db   176                ;code 238
        db   168                ;code 239
        db   '-'                ;code 240
        db   254                ;code 241
        db   ' '                ;code 242
        db   245                ;code 243
        db   244                ;code 244
        db   189                ;code 245
        db   '?'                ;code 246
        db   '?'                ;code 247
        db   179                ;code 248
        db   171                ;code 249
        db   242                ;code 250
        db   '?'                ;code 251
        db   '?'                ;code 252
        db   '?'                ;code 253
        db   252                ;code 254
        db   160                ;code 255

;== Voici la nouvelle interruption imprimante (reste en mémoire) =======

newpri  proc far

        jmp  short newpri_1

        db   "TJ"               ;Identification du programme

newpri_1: cmp  ah,05h           ;Imprimer caractère (Fonction 5) ?
        je   dos_imp            ;OUI --> routine fonction 5

        cmp  ah,40h             ;fonction Handle ? (Fonction 40) ?
        jne  aint               ;NON --> Appeler ancienne interruption
        cmp  bx,4               ;Handle 4 (stdprn) ?
        jne  aint               ;NON --> Appeler ancienne interruption

        ;Traitement de la fonction d'ecriture sur stdprn par handle

        cmp  cx,0               ;0 octets à ecrire ?
        je   aint               ;OUI --> Appeler ancienne interruption


        pushf                   ;Tous les registres modifiés par la
        push ax                 ;suite doivent être sauvegardés
        push bx
        push cx
        push dx
        push es
        push di
        push bp
        push si

        push cs                 ;Ranger CS sur la pile
        pop  es                 ;Retirer de la pile pour DS

        mov  bp,dx              ;transfert de l'offset buffer dans bp
        mov  si,cx              ;transfert nbre d'octets à ecrire dans si
        xor  bh,bh              ;on efface l'octet de poids fort

loop1 :  mov  bl,ds:[bp+si-1]   ;lecture prochain caractère du buffer
        cmp  bl,127             ;si code inferieur à 127
        jbe  no_conv            ;pas de conversion

        sub  bl,128             ;calcul de l'indice du tableau
        mov  al,es:[bx+(offset codetab)]  ;code converti dans al
        mov  ds:[bp+si-1],al    ;réécriture dans buffer

no_conv : dec  si               ;accès caractère précédent dans budder
        jnz  loop1              ;si si<>0, on boucle

        pop  si                 ;Retirer registres sauvegardés
        pop  bp
        pop  di
        pop  es
        pop  dx
        pop  cx
        pop  bx
        pop  ax
        popf

        jmp  aint               ;Vers l'ancienne routine imprimante

        ;traitement de l'interruption par fonction 5 du DOS

dos_imp :
        cmp  dl,127             ;si code inferieur à 127
        jbe  aint               ;pas de conversion

        pushf                   ;Tous les registres modifiés par la
        push ax                 ;suite doivent être sauvegardés
        push bx
        push si
        push ds

        push cs                 ;Ranger CS sur la pile
        pop  ds                 ;Retirer de la pile pour DS

        mov  bl,dl              ;transfert code caractère dans bl
        sub  bl,128             ;soustraire 128 pour obtenir index table
        xor  bh,bh              ;effacer octet de poids fort
        mov  dl,[bx+(offset codetab)] ;nouveau caractère dans dl

        pop  ds                 ;Retirer registres sauvegardés
        pop  si
        pop  bx
        pop  ax
        popf

aint:   jmp  cs:[ancint]        ;Vers l'ancienne routine imprimante

newpri  endp

instend  equ  this byte         ;Jusqu'à cette cellule de mémoire
                                ;tout doit rester résident

;== Données (peuvent être effacées par le DOS) ========================

installm  db   13,10,"TJR8 est installé.",13,10,"$"
already   db   13,10,"TJR8 est déjà installé.",13,10,"$"
to_des    db   "Pour le désactiver :  TJR8 U",13,10,"$"

elimine   db   "TJR8 est désactivé",13,10,"$"
```

```
;== Programme (peut être effacé par le DOS) ============================
;-- Routine de lancement et initialisation ----------------------------

prcoini   label near

          mov  si,128             ;place sur début paramètre dans PSP
          cmp  byte ptr [si],0    ;Y a t-il des paramètres ?
          je   install            ;non, on installe
          inc  si                 ;décalage de "si" sur le premier paramètre
          inc  si
          cmp  byte ptr [si],"U"  ;désinstalation demandée ?
          jne  install            ;non on va l'installer

          mov  ax,3521h           ;Lire contenu vecteur d'interruption 21h
          int  21h                ;Appeler fonction DOS
          cmp  word ptr es:[bx+2],"JT" ;Tester si programme TJR8
          jne  not_inst           ;Pas de TJRR8 installé --> not_inst

          ;-- Désactivation de TJR8 ---------------------------------------

          mov  dx,es:ofsancint    ;Adresse d'offset de l'interruption 21h
          mov  ax,es:segancint    ;Adresse de segment de l'interruption 21h
          mov  ds,ax              ;dans DS
          mov  ax,2521h           ;Replacer contenu vecteur d'interruption 21h
          int  21h                ;sur ancienne routine

          mov  bx,es              ;Ranger adresse de segment du programme
          mov  es,es:[2Ch]        ;Retirer du PSP adr. seg. de l'environnement
          mov  ah,49h             ;Libérer à nouveau mémoire
          int  21h                ;d'ancien environnent
          mov  es,bx              ;Libérer à nouveau la mémoire
          mov  ah,49h             ;de l'ancienne TJR8
          int  21h                ;à travers la fonction 49h du DOS

          push cs                 ;Sauver CS sur la pile
          pop  ds                 ;Ramener dans DS

not_inst: mov  dx,offset elimine  ;Message : programme éliminé
          mov  ah,9               ;Numéro de fonction pour Ecrire chaine
          int  21h                ;Appeler fonction DOS

          mov  ax,4C00h           ;Terminer programme correctement
          int  21h                ;Appeler fonction pour terminer progr.

          ;-- Installation de TJR8 ----------------------------------

install   label near

          mov  ax,3521h           ;Lire contenu vecteur d'interruption 21h
          int  21h                ;Appeler fonction DOS
          cmp  word ptr es:[bx+2],"JT" ;Tester si programme TJR8
          je   already_inst       ;TJRR8 déjà installé --> already_inst

          mov  ax,3521h           ;Lire contenu du vecteur d'interruption 21h
          int  21h                ;Appeler fonction DOS
          mov  segancint,es       ;Ranger adresses de segment et d'offset
          mov  ofsancint,bx       ;Ranger vecteur d'interruption 21h

          mov  dx,offset newpri   ;Adresse d'offset nouvelle routine d'interr.
          mov  ax,2521h           ;Contenu du vecteur d'interruption 21h
          int  21h                ;sur routine utilisateur
          mov  dx,offset installm ;Message : programme installé
          mov  ah,9               ;Numéro de fonction pour Sortir chaine
          int  21h                ;Appeler fonction DOS
          mov  dx,offset to_des   ;Message : comment désinstaller
          mov  ah,9               ;Numéro de fonction pour Sortir chaine
          int  21h                ;Appeler fonction DOS

          ;-- Seuls le PSP, la nouvelle routine d'interruption et les
          ;-- données correspondantes doivent rester résidentes. -------

          mov  dx,offset instend  ;Calculer nombre de paragraphes (unités de 16
          mov  cl,4               ;octets) dont doit disposer le programme
          shr  dx,cl
          inc  dx
          mov  ax,3100h           ;Terminer programme par code de fin 0 (o.k)
          int  21h                ;mais rester résident

          ;-- TJR8 est déjà installé : on affiche message et on sort -------

already_inst:
          mov  dx,offset already  ;Message : programme déjà installé
          mov  ah,9               ;Numéro de fonction pour Sortir chaine
          int  21h                ;Appeler fonction DOS
          mov  dx,offset to_des   ;Message : comment désinstaller
          mov  ah,9               ;Numéro de fonction pour Sortir chaine
          int  21h                ;Appeler fonction DOS

          mov  ax,4C00h           ;Terminer programme correctement
          int  21h                ;Appeler fonction pour terminer progr.

;== Fin ================================================================

code      ends                    ;Fin du segment de CODE
          end  start
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                IBMTOR8 (filtre de conversion IBM-Roman8)                **/
/**                                                                         **/
/**   création : 08/03/90                        Auteur : JB               **/
/**                                                                         **/
/**   07/06/92 : passage table de conversion en ".h"                        **/
/**                                                                         **/
/*****************************************************************************/
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "getopt.c"
#include "pc850_r8.h"

/*****************************************************************************/
/** Table de conversion                                                     **/
/*****************************************************************************/
unsigned char  *tab=pc850_r8;


void main(int argc, char *argv[])
  {
   unsigned char ci, co;
   int conv_tabs = 0;
   char c;
   int errflag=0;

   while((c=getopt(argc, argv, "t")) != EOF)
     {
      switch(c)                                   /* selection des options */
        {
         case 't' : conv_tabs = 1;
```

```
                    break;

        case '?' : errflag++;
                   break;
      }
    }

  if(optind != argc)
     errflag++;

  if(errflag)                                   /* sortie en cas d'erreur */
    {
     fprintf(stderr, "usage : ibmtor8 [-t] <infile >outfile\n");
     exit(-1);
    }

  do
    {
     ci = (unsigned char) getc(stdin);
     switch((char) ci)
       {
        case '\t' : if(conv_tabs)
                       fputs("        ", stdout);
                    else
                       putc((int) ci, stdout);
                    break;

        case EOF  : break;

        default   : co = (ci > 127) ? tab[ci-128] : ci;
                    putc((int) co, stdout);
                    break;
       }
    }
  while(ci != ((unsigned char)EOF));
 }
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                          PC850_R8.H                                     **/
/**                 Table de conversion Roman8-PC850                        **/
/**                                                                         **/
/**    Création : 07/06/92              Auteur : JB                         **/
/*****************************************************************************/
/*****************************************************************************/

     /* table de conversion de caractères IBM vers Roman8 (à partir du 128e)*/
unsigned char pc850_r8[]={180, 207, 197, 192, 204, 200, 212, 181, /* 128-135 */
                          193, 205, 201, 221, 209, 217, 216, 208, /* 136-143 */
                          220, 215, 211, 194, 206, 202, 195, 203, /* 144-151 */
                          239, 218, 219, 214, 187, 210, '?', 190, /* 152-159 */
                          196, 213, 198, 199, 183, 182, 249, 250, /* 160-167 */
                          185, '?', '?', 248, 247, 184, 251, 253, /* 168-175 */
                          '?', 127, '?', '|', '?', 224, 162, 161, /* 176-183 */
                          '?', '?', '?', '?', '?', 191, 188, '?', /* 184-191 */
                          '?', '?', '?', '?', 246, '?', 226, 225, /* 192-199 */
                          '?', '?', '?', '?', '?', '=', '?', 186, /* 200-207 */
                          228, 227, 164, 165, 163, '?', 229, 166, /* 208-215 */
                          167, '?', '?', '?', '?', '|', 230, '?', /* 216-223 */
                          231, 222, 223, 232, 234, 233, 243, 241, /* 224-231 */
                          240, 237, 174, 173, 178, 177, 176, 168, /* 232-239 */
                          '-', 254, ' ', 245, 244, 189, '?', '?', /* 240-247 */
                          179, 171, 242, '?', '?', '?', 252, 160};/* 248-255 */
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                          R8_PC850.H                                     **/
/**                 Table de conversion Roman8-PC850                        **/
/**                                                                         **/
/**    Création : 07/06/92              Auteur : JB                         **/
/*****************************************************************************/
/*****************************************************************************/

     /* table de conversion de caractères Roman8 vers IBM (à partir du 128e) */
unsigned char r8_pc850[]={'?', '?', '?', '?', '?', '?', '?', '?', /* 128-137 */
                          '?', '?', '?', '?', '?', '?', '?', '?', /* 136-143 */
                          '?', '?', '?', '?', '?', '?', '?', '?', /* 144-151 */
                          '?', '?', '?', '?', '?', '?', '?', '?', /* 152-159 */
                          255, 183, 182, 212, 210, 211, 215, 216, /* 160-167 */
                          239, '?', '?', 249, '?', 235, 234, 156, /* 168-175 */
                          238, 237, 236, 248, 128, 135, 165, 164, /* 176-183 */
                          173, 168, 207, 156, 190, 245, 159, 189, /* 184-191 */
                          131, 136, 147, 150, 160, 130, 162, 163, /* 192-199 */
                          133, 138, 149, 151, 132, 137, 148, 154, /* 200-207 */
                          143, 140, 157, 146, 134, 161, 155, 145, /* 208-215 */
                          142, 141, 153, 154, 144, 139, 225, 226, /* 216-223 */
                          181, 199, 198, 209, 208, 214, 222, 224, /* 224-231 */
                          227, 229, 228, '?', '?', 233, '?', 152, /* 232-239 */
                          232, 231, 250, 230, 244, 243, 196, 172, /* 240-247 */
                          171, 166, 167, 174, 254, 175, 241, 255};/* 248-255 */
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                          INITJET.C                                      **/
/**              Initialisation de la ThinkJet / HP-LINK                    **/
/**                                                                         **/
/**   Dernière modification : 25/04/90                                      **/
/**       Création                                                          **/
/**                                                                         **/
/*****************************************************************************/
/*****************************************************************************/
#include <stdio.h>

cdecl main(int argc, char *argv[]);

main(int argc, char *argv[])
    {
     char pl[4], tl[4];

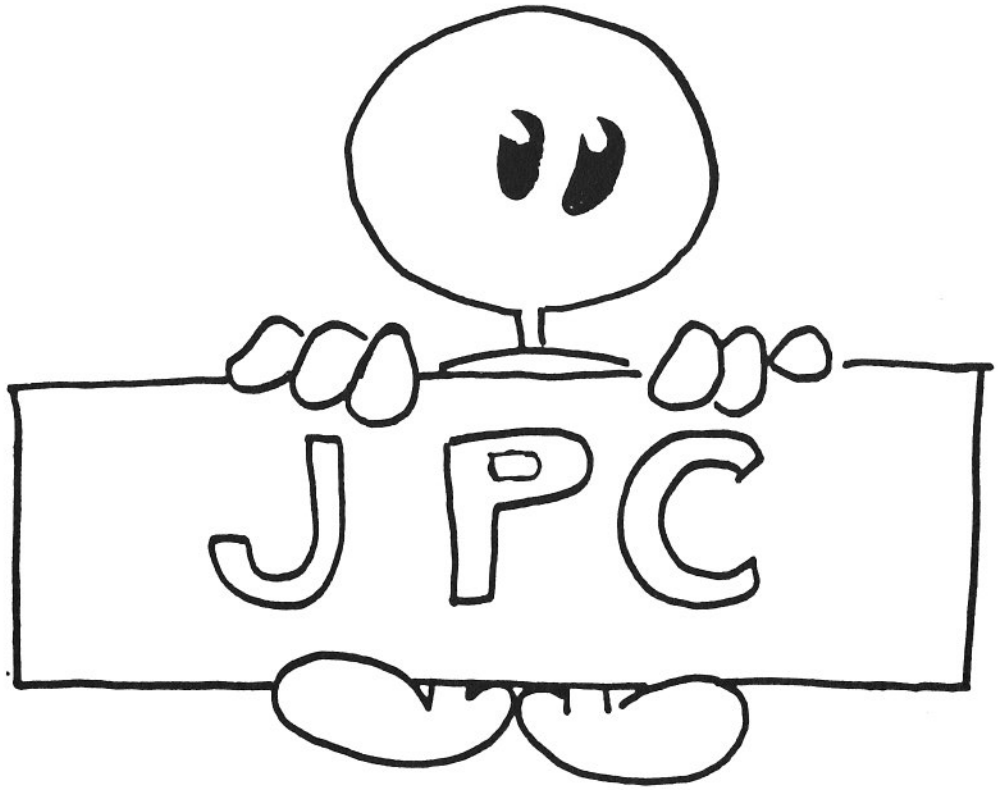
     switch(argc)                              /* suivant le nombre d'arguments */
       {
        case 1 : strcpy(pl, "72");      /* pas d'arguments : page 12" imposée */
                 strcpy(tl, "66");
                 break;

        case 2 : strcpy(pl, argv[1]);  /* 1 argument : nombre de lignes = 66 */
                 strcpy(tl, "66");
                 break;

        case 3 : strcpy(pl, argv[1]);                          /* format libre */
                 strcpy(tl, argv[2]);
                 break;
```

```
        default : break;
       }
     fprintf(stdprn, "%c&l%sp%sf1L", 27, pl, tl); /* initialisation ThinkJet */
    }
```

```
;*****************************************************************************/
;*****************************************************************************/
;**                FF.ASM (Envoi d'un Form-Feed sur une imprimante)         **/
;**                                                                         **/
;**   création : 12/05/92                      Auteur : JB                 **/
;*****************************************************************************/
;*****************************************************************************/
code    segment para 'CODE'
        assume cs:Code
start:  mov     dl,12     ;code du caractère FF
        mov     ah,05h    ;fonction d'écriture vers l'imprimante
        int     21h       ;impression du caractère
        mov     ah,4ch    ;fonction de fin de programme du DOS
        int     21h       ;appel du DOS et sortie
Code    ends
        END     start
```



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

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.