ABC INFORMATIQUE

N° 94

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

D'un coup d'œil

L'Apricot est une pêche

Plus PC que moi, tu meurs

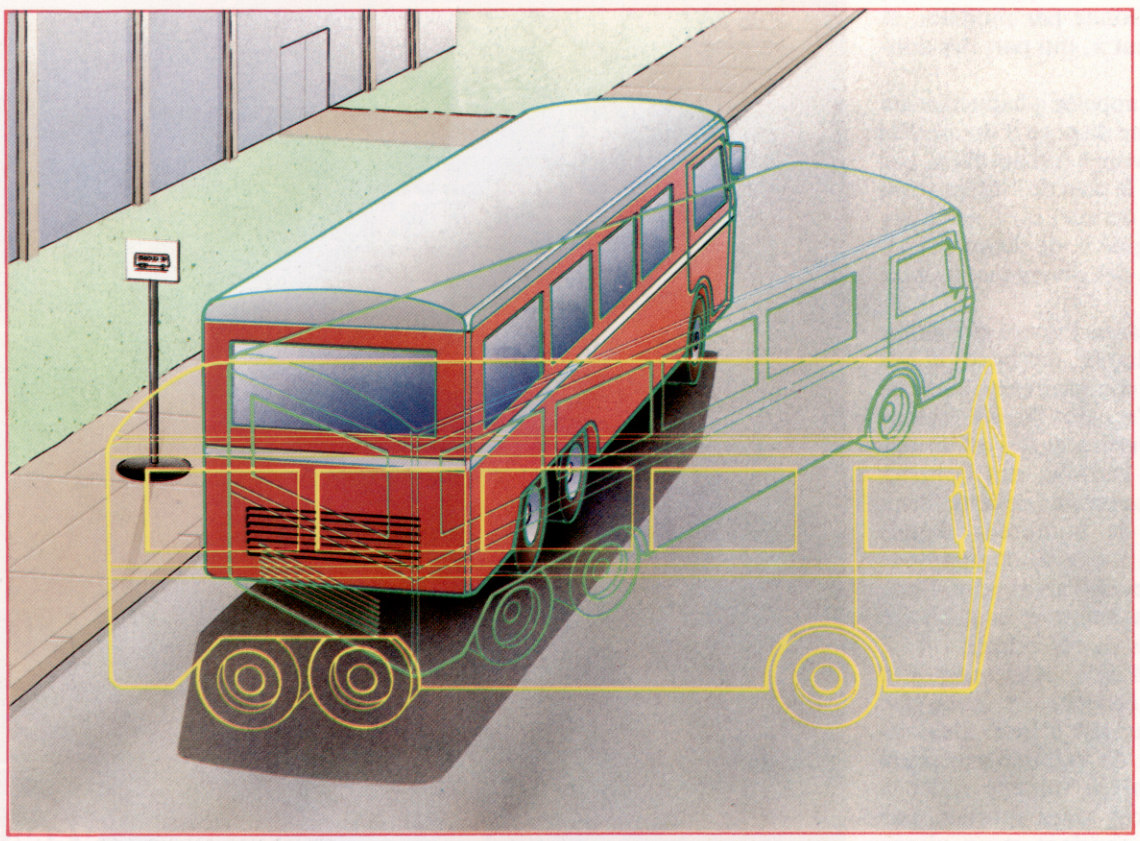
Interaction stellaire

EDITIONS ATLAS

M 6062-94-12,50 F 90 FB - 3,80 FS - $ 1.95

D'un coup d'œil

« Voir » et « comprendre », deux aspects des choses difficiles à envisager séparément. Mais c'est nécessaire pour qu'un ordinateur « comprenne » ce qu'il sera amené à « voir ».



D'accord, pas d'accord
Certains systèmes de reconnaissance de formes utilisent une méthode dite « de haut en bas ». Il s'agit d'analyser une scène ou une structure quelconque, en se mettant à la recherche d'un objet particulier. Pour cela, une représentation simplifiée de celui-ci (réduite à ses contours principaux) est projetée sur l'écran selon divers angles, jusqu'à ce que l'une des images ainsi créées concorde avec l'image réelle. Quand celle-ci est tridimensionnelle (comme ici), il faut tenir compte du fait que les « modèles » de l'autobus peuvent être très nombreux et que par conséquent les opérations de concordance prendront un certain temps.
(Cl. Kevin Jones.)

La vision est un processus extrêmement complexe : les rayons lumineux qui viennent frapper notre rétine sont convertis en signaux électriques et transmis, par l'intermédiaire du nerf optique, jusqu'au cortex, qui se charge de les analyser. C'est cette chaîne d'événements que les physiologistes cherchent à mieux connaître, tandis que les roboticiens s'efforcent de la reproduire.

La perception visuelle joue un rôle si important dans la reconnaissance de notre environnement que nous sommes souvent amenés à dire « je vois » au lieu de « je comprends ». Cette « compréhension » est précisément le problème essentiel de tout système de vision informatisée. L'ordinateur acquiert son information grâce à une caméra, ou à un dispositif du même ordre, mais il lui reste à traiter ces données brutes : ce n'est pas leur collecte, mais leur interprétation qui pose des difficultés.

Cette tâche peut d'ailleurs être décomposée en trois étapes successives :

1. *Traitement de l'image.* Celle-ci doit être la plus « claire » possible, sans distorsions ni flou. C'est un travail relativement simple.

2. *Reconnaissance des formes.* Il faut identifier certains objets, ou certaines caractéristiques (ou leur absence). Voilà qui est déjà plus complexe.

3. *Compréhension des images.* A partir des éléments dont on dispose, il faut pouvoir être en mesure de savoir réellement ce qui se passe dans le monde extérieur. La besogne devient très compliquée...

A dire vrai, on n'est pas encore parvenu à maîtriser pleinement la troisième étape du processus, mais les deux premières ont d'ores et déjà fait l'objet de recherches approfondies, et sont à l'heure actuelle assez bien comprises.

Reconnaissance de formes

Le principe de base consiste à classifier les images relatives à un ensemble de formes limité (ainsi les lettres de l'alphabet pour les systèmes optiques de reconnaissance de caractères). Il faut pour cela examiner des fragments (de petits groupes de pixels) de l'image numérisée. Le système peut alors reconnaître et classer certaines formes,

selon la présence, ou l'absence, de traits distinctifs particuliers. Cela ne va pas sans « distorsions » dans l'analyse : nous savons très bien ce qu'est une ligne en diagonale, mais, pour l'ordinateur, c'est une notion inexistante; il se borne à effectuer des calculs — toujours arbitraires — sur une image. Quelles que soient ses performances, un système de ce genre met en œuvre une notion de la « vision » extrêmement simplifiée.

Historiquement, les recherches sur la vision informatisée (la compréhension des images) ont suivi deux voies divergentes : l'approche « de haut en bas », qui fonctionne par modèles, et l'approche « de bas en haut », qui part des données elles-mêmes.

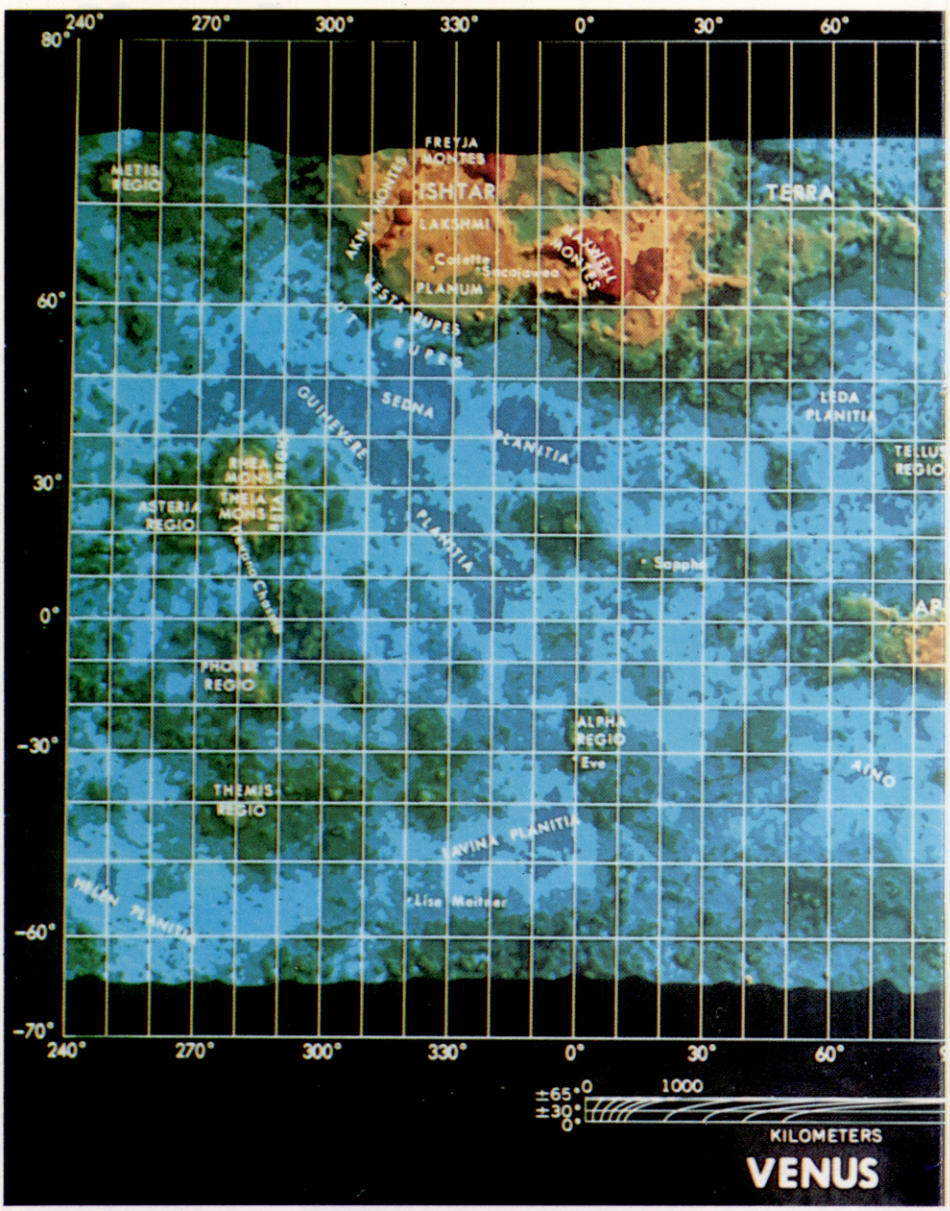
La première est parfois appelée « hallucination contrôlée ». Le programme dispose d'un modèle interne de ce qu'il doit « voir » (un autobus, par exemple); il le projette sur le plan de l'image selon plusieurs angles de prise de vue, qu'il compare ensuite à ce qu'il a « imaginé », de façon à repérer des concordances (ou des divergences) entre modèle et réalité.

La seconde méthode analyse l'image elle-même de façon à identifier des lignes, des bords et des contours. Cela permet de construire une sorte de description simplifiée, qu'on appelle conventionnellement « premier croquis », et qui est en fait une représentation, très stylisée, des données elles-mêmes. L'idée de base est que de cette façon (en faisant abstraction de nombreux détails secondaires), il sera plus facile de comparer le « croquis » à des exemples choisis au sein d'une certaine catégorie d'objets conservés en mémoire et qui servent de modèles de référence.

Le système Wisard (*Wilkie, Alexander and Stonham's Recognition Device;* cet acronyme signifie « sorcier » en anglais) d'Igor Aleksander constitue de ce point de vue une exception intéressante. On peut même lui apprendre à reconnaître des différences assez subtiles, par exemple entre un visage souriant et un visage renfrogné.

Il opère de surcroît en temps réel (25 images par seconde), et a d'abord été employé commercialement pour suivre des chocolats se déplaçant sur une chaîne de fabrication. Peut-être est-ce le premier exemple d'une nouvelle génération de machines réellement capables non seulement de « voir », mais aussi de s'adapter à leur environnement et d'améliorer sans cesse leurs performances.

Le principe de fonctionnement de Wisard est le suivant : au sein de la RAM dont il dispose, il définit des groupes de 8 pixels (les « octuples ») qui jouent le rôle de « détecteurs » de caractéristiques particulières. Chaque octuple peut prendre 256 états différents (de 0 à 255), selon le statut de chacun des pixels qu'il contient. Au cours de la phase d'apprentissage, un 1 est stocké en RAM, à l'adresse spécifiée par l'état de l'octuple lorsque telle ou telle image est présente. Plus tard, au moment de la reconnaissance des formes, la présence d'un 1 à l'adresse RAM en question indique que l'image « enseignée » au système est de nouveau présente.



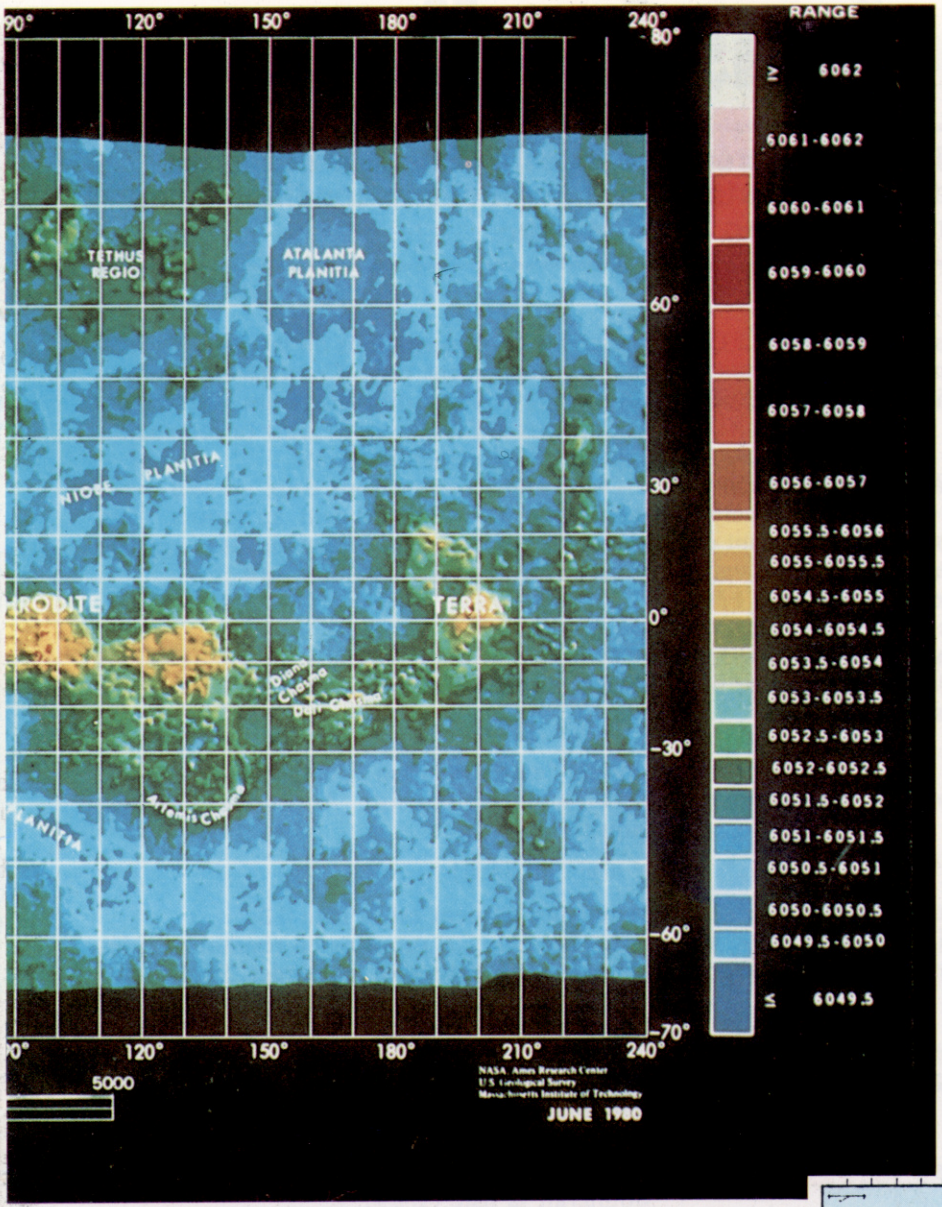
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	1	0	0
0	1	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	0	0	0	1	0

0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0

La différence entre R et A

Le monde réel contient fort peu de données dépourvues d'ambiguïté. Voici deux images numérisées des lettres A et R. Les deux ensembles ne sont pas très « lisibles », car ils sont affectés de distorsions et de « blancs ».

Une donnée « parasite » reste un problème très difficile pour tous les systèmes d'intelligence artificielle. Êtes-vous capable de faire la différence entre les deux lettres? (Cl. Caroline Clayton.)

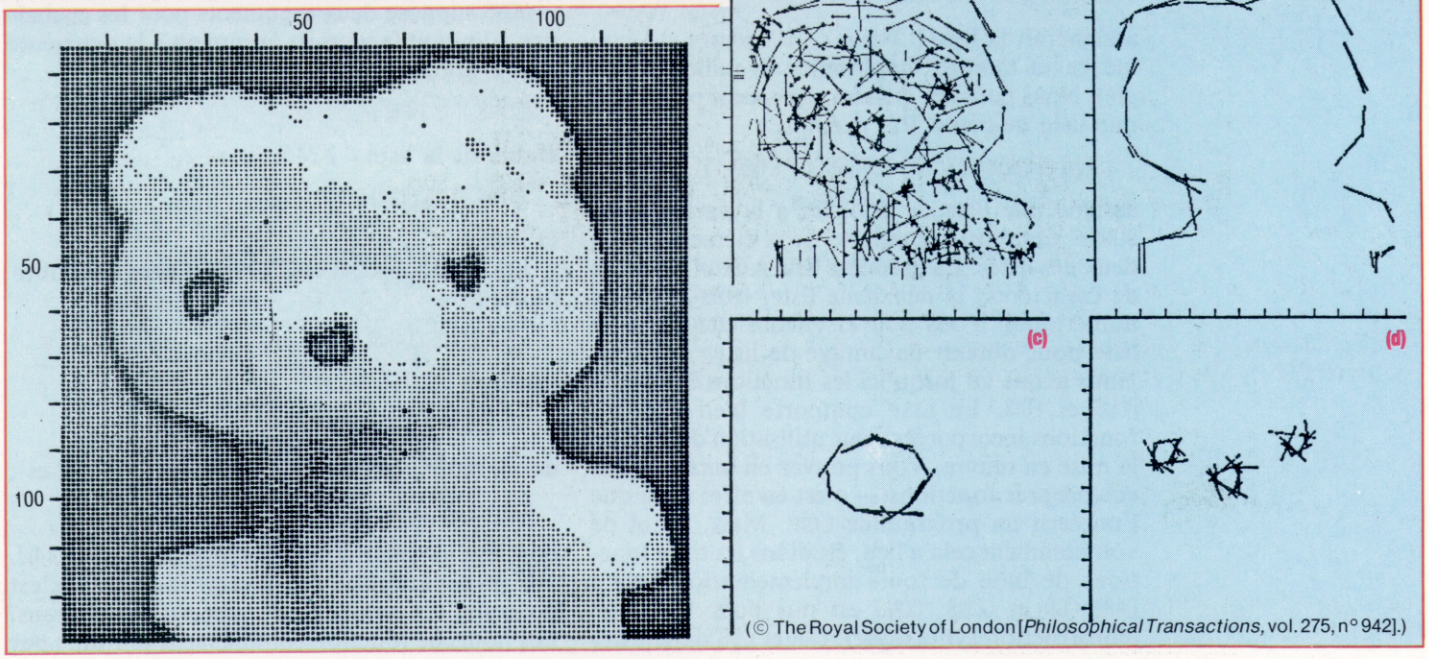


Traitement d'image

Voici un exemple de traitement d'image. Ce cliché transmis par une sonde spatiale a été analysé par ordinateur afin d'accentuer les frontières entre zones d'altitudes différentes. D'où l'emploi de ce qu'on appelle les « fausses couleurs ». Notre image — la surface de Vénus vue au radar — resterait invisible dans des conditions « normales » : Vénus est perpétuellement entourée d'épais nuages, que seuls peuvent percer certaines longueurs d'onde auxquelles notre œil est insensible : infrarouge, ultraviolet, ondes radio. De telles images (parfois fort belles) ont beaucoup apporté à l'astronomie. Plus près de nous, le sonar permet d'obtenir une « image acoustique » du fond de l'océan, sans qu'il soit besoin de recourir aux radiations électromagnétiques (lumière visible, ultraviolet, infrarouge, rayons X, etc.) quelles qu'elles soient. Le traitement d'image implique également le « nettoyage » des images distordues ou brouillées. Les algorithmes indispensables sont aujourd'hui d'emploi si fréquent qu'on ne peut considérer qu'ils appartiennent au domaine de l'intelligence artificielle.

Croquis primaires

Le traitement d'image recourt parfois à une méthode qui ne fait pas usage de connaissances préalables relatives à la signification des structures analysées. Un croquis « primaire » (*a*) détermine des limites et des formes fondamentales (ici à partir de l'image d'un ours en peluche), en comparant des zones adjacentes. De nouvelles analyses à partir de ce croquis (de *b* à *d*) permettent d'extraire de nouveaux groupes facilitant la reconnaissance de l'image de base.



(© The Royal Society of London [*Philosophical Transactions*, vol. 275, n° 942].)

Listes dans les listes

Nous montrons ici comment créer des listes à l'intérieur des listes et mettons en évidence l'importance des instructions conditionnelles et des fonctions utilisateurs, en langage lisp.

Nous avons déjà vu comment demander à LISP d'évaluer une liste comme liste d'éléments de données plutôt que comme une fonction. Aussi :

```
(SETQ X '(A B C D E F) )
```

assigne la liste (A B C D E F) à la variable X; chaque élément est une chaîne à un seul caractère. Voyons ce qui se passe lorsque D, E et F sont des chaînes à un seul caractère, et A, B, C des variables prenant respectivement les valeurs 2, 4 et 8. Nous voulons dans le cas présent assigner à X la liste (2 4 8 D E F). Pour ce faire, nous introduisons une nouvelle fonction : LIST. Elle crée une liste avec ses arguments. Ainsi :

```
(SETQ X (LIST 'A 'B 'C 'D 'E 'F) )
```

serait la même expression que précédemment, mais :

```
(SETQ X LIST A B C 'D 'E 'F)
```

affecterait de manière exacte la liste (2 4 8 D E F).

Ici, en omettant les guillemets simples, A, B et C seront évalués. De manière très semblable :

```
(SETQ X (LIST 1 2 4 '(PLUS 4 4) ) )
```

assigne la liste (1 2 4 (PLUS 4 4)) à X, liste dont le quatrième élément est lui-même une liste (PLUS 4 4). En d'autres termes, PLUS n'a pas été évalué car nous avons fait précéder l'expression d'un guillemet simple. Néanmoins :

```
(SETQ X (LIST 1 2 4 (PLUS 4 4) ) )
```

assignerait la liste (1 2 4 8) à X, PLUS ayant été évalué (ce qui montre l'importance du guillemet simple). Nous pouvons étendre ce concept pour créer une liste de listes. Par exemple :

```
(SETQ PERSONNE '( (JOE BLOGGS) (17 1 1960) ) )
```

assigne une liste de données à la variable PERSONNE. Cette liste comporte deux éléments, tous deux des listes. La première liste a deux éléments de caractères; la deuxième liste, trois éléments numériques. Vous pouvez encore étendre cette liste pour obtenir davantage de listes de listes. Nous avons vu jusqu'ici les fonctions SETQ, LIST, PLUS et FOIS. Le LISP comporte bien d'autres fonctions incorporées; leur utilisation dépend de la mise en œuvre. Vous pouvez en outre définir vos propres fonctions — c'est en effet ainsi que l'on écrit un programme LISP. Mais, avant de voir comment cela a lieu, étudions les trois fonctions de base de toute implémentation LISP : CAR, CDR et CONS. CONS est mis pour CONS*truct* (construction syntaxique). CAR et CDR datent des premières implémentations du langage. Ils sont mis respectivement pour *Contents of Address Register* (« contenu du registre des adresses »), et *Contents of Decrement Register* (« contenu du registre de décrémentation »).

CAR et CDR acceptent tous deux un seul argument, une liste d'un élément au moins. Cela exclut la structure spécifique de liste appelée « liste vide » ou NIL, que l'on écrit :

```
()
```

La fonction CAR retourne le premier élément de son argument de liste. Ainsi :

```
(CAR '(1 2 3 4 5) )
```

donne l'entier 1. Rien n'empêche le résultat d'être lui-même une liste. Ainsi, dans l'expression :

```
(CAR '( 1 2) (3 4) 5) )
```

le premier élément est la liste (1 2).

La fonction CDR retourne en résultat tous les éléments de la liste, sauf le premier. En d'autres termes, elle restitue les éléments restant après une opération CAR. Aussi :

```
(CDR '(1 2 3 4 5) )
```

donnerait la liste (2 3 4 5), et :

```
(CDR '( (1 2) (3 4) 5) )
```

donnerait la liste ((3 4) 5).

CONS suppose deux arguments pour les enchaîner, ajoutant le premier argument à la deuxième liste d'arguments. Ainsi :

```
(CONS 1' (2 3 4 5) )
```

établirait la liste (1 2 3 4 5), et :

```
(CONS '(1 2) '( (3 4) 5) )
```

la liste ((1 2) (3 4) 5).

Les fonctions CAR et CDR seront souvent imbriquées :

```
(CDR '( (1 2) (3 4) 5) )
```

donnera ((3 4) 5), et :

```
(CAR (CDR '( (1 2) (3 4) 5 ) ) )
```

donnera (3 4), qui correspond à (CAR '((3 4) 5)), et :

```
(CAR (CAR (CDR '( (1 2) (3 4) 5) ) ) )
```

donnera la valeur 3, qui correspond à (CAR '(3 4)).

Cela peut rapidement devenir fastidieux, c'est pourquoi LISP permet l'utilisation d'abréviations. Les noms de fonction commencent toujours par

un C et se terminent par un R, mais peuvent comporter toute combinaison de A (pour CAR), et de D (pour CDR). Avec le LISP d'Acornsoft, cette combinaison peut comprendre jusqu'à trois lettres imbriquées. Ainsi, nous pourrions écrire la dernière expression de la sorte :

```
(CAAR (CDR '( (1 2) (3 4) 5) ) )
```

ou :

```
(CAR (CADR '( (1 2) (3 4) 5) ) )
```

ou même plus simplement :

```
(CAADR '(1 2) (3 4) 5) )
```

Toutes ces formations ont pour résultat 3.

Définition de fonctions

Comme nous l'avons déjà vu, les programmes LISP sont constitués de fonctions définies par les utilisateurs. Toutes les expressions LISP sont des listes de fonctions, aussi nous utilisons tout naturellement des fonctions pour définir des fonctions. Dans le cas présent, prenons la fonction DEFUN qui suppose trois arguments de la forme :

```
(DEFUN a (b) (c) )
```

où a est un nom de fonction, b la liste des paramètres (comme pour le PASCAL...) et c la structure de liste comportant le corps principal de la fonction.

Nous pouvons maintenant définir notre toute première fonction. Supposons que nous ayons souvent à multiplier des nombres par 8. Nous pourrions simplement écrire :

```
(FOIS 8 N)
```

à chaque fois, N étant le nombre en question. Définissons plutôt une fonction qui le fera à notre place :

```
(DEFUN FOIS8 (N) (FOIS 8 N) )
```

Ici, la fonction FOIS8 a N pour argument, et utilise la fonction standard FOIS pour multiplier le nombre par 8 et retourner une valeur.

Aussi l'expression :

```
(FOIS8 11)
```

donnerait le résultat 88.

Avant de poursuivre avec les fonctions, étudions un concept important de programmation LISP : la condition. Son rôle peut être illustré en imaginant le BASIC sans instructions IF...THEN !

L'instruction conditionnelle LISP condition a la forme d'une fonction :

```
(COND (Condition1 Expression1)
      (Condition2 Expression2)
          :          :
      (ConditionX ExpressionX))
```

Remarquez tout d'abord que cette fonction porte sur plusieurs lignes. Ce qui est tout à fait normal avec LISP, qui reconnaît la fin d'une expression à la parenthèse finale.

```
(DEFUN ÉGAL (A B) (COND
  ( (EG A B) V)
  ( (OU (ATOM A) (ATOM B) ) RIEN)
  ( (ÉGAL (VOITURE A) (VOITURE B) ) (ÉGAL (CDR A) (CDR B) ) ) (V RIEN) ) )
(DEFUN TROUVER_ VOITURE (X L) (COND
  ( (NUL L) '(PAS DE NUMÉRO) )
  ( (ÉGAL X (CDAR L ) ) (VOITURE L) )
  (V (TROUVER_VOITURE X (CDR L) ) ) ) )

(SETQ VOITURE '( (VOITURE VINCENT 7524 LML 75)  (PARTOT 1213 XL 92)
                 (ALTU 1415 XU 12)              (COSSE  3624  MNO
                                                83))
(PRINTC (TROUVER_VOITURE '(ALTU 1415 XU 12) VOITURES) )
```

Procédure de recherche
Ce simple programme de restitution de données illustre l'utilisation des fonctions CDR et EQ. L'utilisateur entre un numéro d'immatriculation, et, si ce dernier figure à la base de données, on obtient une liste de deux éléments : le numéro d'immatriculation et le nom du propriétaire de la voiture.

La fonction COND ressemble à une instruction PASCAL CASE OF. La condition 1 est évaluée : si elle est vraie, l'expression 1 est utilisée pour obtenir un résultat. Sinon, on passe à la condition 2, et ainsi de suite. Certaines versions LISP (dont Acornsoft LISP) supposent qu'au moins une des conditions (appelées prédicats) soit vraie. La syntaxe suivante vient facilement à bout de cette difficulté :

```
(COND (Condition1 Expression1)
      (Condition2 Expression2)
          :          :
(V                  Expression finale))
```

V est mis pour vrai, et l'expression finale est toujours évaluée. Avec LISP :

```
V = vrai = non zéro
```

et :

```
F = faux = zéro = ()
```

La dernière expression représente la liste vide. Nous pouvons définir maintenant une fonction plus complexe.

ABS donne, avec la plupart des versions du BASIC, la valeur absolue d'un nombre, la valeur positive de son argument (si l'argument est positif, il n'est pas modifié; s'il est négatif, il est inversé).

C'est une fonction essentielle. Avec LISP, l'équivalent sera :

```
(DEFUN ABS (X)
  (corps de la fonction) )
```

X est l'argument de l'entier. En utilisant notre nouvelle fonction de condition, nous pouvons écrire la fonction complète ABS :

```
(DEFUN ABS (X)
  (COND ( (MOINSP X) (MOINS X) )
        ( V          X         ) ) )
```

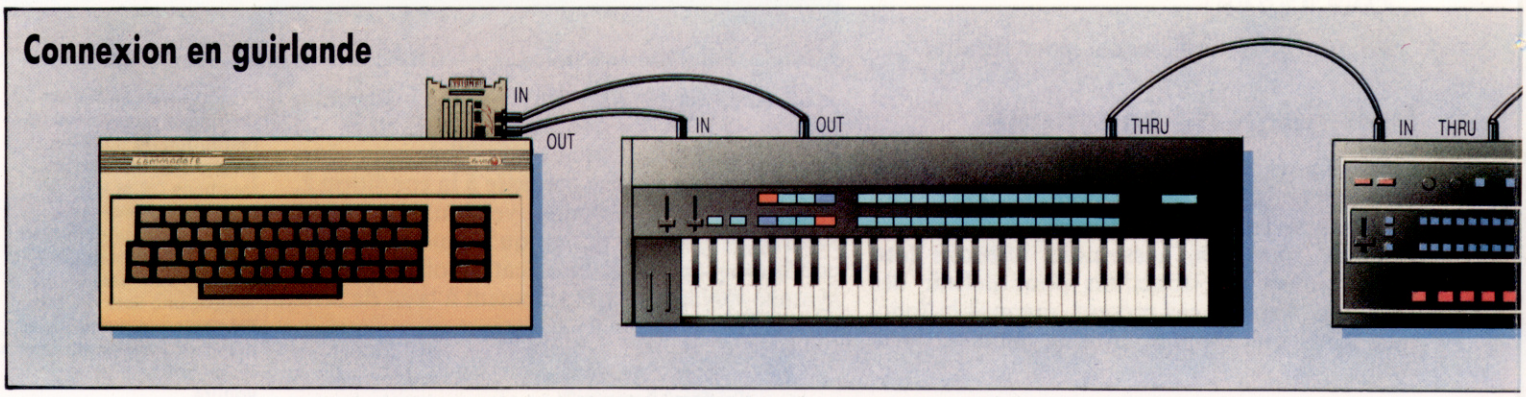
Nous avons utilisé deux nouvelles fonctions. MOINSP est une fonction de test qui retourne la valeur quand son argument est négatif. Dans le cas contraire, elle donne la valeur faux. Si la condition donne le résultat V (vrai), la fonction MOINS affecte le signe « moins » à son argument. Dans le cas contraire, la condition V (qui est toujours vrai) donnera la valeur X telle quelle.

Fonctions tests
Voici quelques autres fonctions tests fréquemment utilisées avec LISP :

(OR arg1 arg2...)	donne vrai lorsqu'au moins un de ses arguments est vrai.
(ONEP arg)	donne vrai si arg = 1.
(ZEROP arg)	donne vrai si arg = 0.
(LISTP arg)	donne vrai si arg est une liste, et faux si arg est un atome.
(ATOM arg)	donne vrai quand arg est un atome, et faux quand l'argument est une liste.
(LESSP arg1 arg2)	donne vrai quand arg1 < arg2.
(GREATERP arg1 arg2)	donne vrai lorsque arg1 > arg2.
(EQ arg1 arg2)	donne vrai si arg1 = arg2.

Règles de composition

Après avoir construit l'interface MIDI, nous attirons l'attention sur un autre aspect tout aussi important du logiciel qui nous permettra de jouer de la musique « en temps réel ».



Enchaînement...
La méthode la plus répandue pour connecter des instruments compatibles MIDI consiste à les enchaîner. La plupart des instruments ont des prises MIDI THRU (en plus de MIDI IN et MIDI OUT). Ces prises permettent de former un système de type bus, dans lequel les instruments sont connectés au bus mais répondent uniquement aux messages envoyés sur des canaux particuliers.

Un « événement » musical est transmis sous la forme d'un groupe d'octets nommé « message ». La plupart des messages ont une longueur qui varie entre un et trois octets, sauf les messages « systèmes exclusifs », qui peuvent être composés d'un nombre indéterminé d'octets. Chaque message commence par un octet dont le bit le plus significatif (BPS) est égal à 1, suivi des autres dont les BPS sont égaux à 0. Un octet dont le BPS est 1 est dit octet d'état, les autres sont des octets de données. Les messages MIDI se répartissent en deux catégories de base, les *canaux* et les *systèmes*.

C'est la méthode de chaînage utilisée dans les petits systèmes qui crée le besoin des messages canaux. Dans ceux-ci, chaque unité reçoit toutes les données envoyées par l'unité maîtresse de transmission.

Les messages canaux sont utilisés pour transmettre l'information qui n'est pas nécessairement destinée à toutes les unités du système. La plupart des récepteurs MIDI peuvent donc être affectés à un numéro de canal, de 1 à 16.

Les messages de canaux ont des octets d'état ayant des valeurs comprises entre $80 et $EF inclusivement, et ont un ou deux octets de données. Le numéro de canal est codé sur les quatre bits les moins significatifs (les quatre chiffres hex les moins significatifs) de l'octet d'état avec le canal 1 représenté par $0 et le canal 16 par $F. Les trois autres bits déterminent le type de message qui suit.

Une des caractéristiques spéciales des messages canaux est que l'octet d'état n'est pas envoyé s'il est le même que le précédent. En fait, l'octet d'état en cours demeure en vigueur jusqu'à ce qu'un autre soit reçu. Mais il y a exception, lorsqu'un système de message en temps réel interrompt temporairement l'état en cours.

C'est particulièrement pratique pour transmettre de nombreux messages *note on/off* consécutifs, puisqu'un message *note on* ayant une vitesse de zéro (intensité de la frappe d'une touche) est équivalent à un message *note off*. En utilisant le même état *(note on)* pour une série de messages de *note on/off,* il est possible d'économiser de l'espace mémoire et du temps de transmission.

L'*affectation de voix* dans les synthétiseurs est le processus qui permet de diriger un message *note* (venant de MIDI ou du clavier de l'instrument) vers l'une des voix du synthétiseur pouvant alors produire la note. Ainsi, un synthétiseur polyphonique à six notes est dit avoir six voix.

Messages de mode MIDI

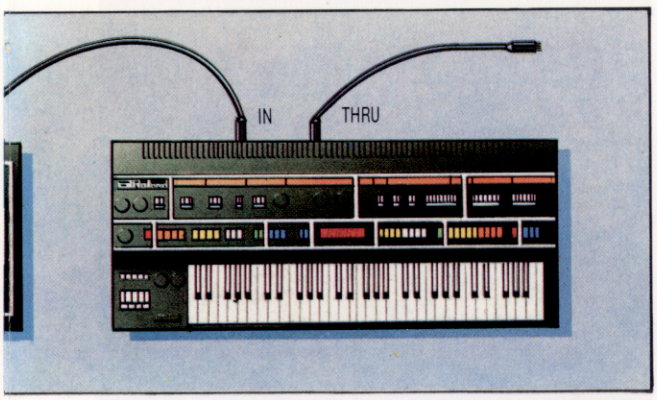
Le *mode omni* est le mode le plus simple par lequel le récepteur répond à tous les messages canaux quel que soit le numéro de canal codé dans les messages. Quand le *mode omni* est mis hors fonction, l'unité ne répond qu'aux messages envoyés sur les canaux MIDI spécifiques.

La plupart des synthétiseurs polyphoniques ont un nombre limité de voix (généralement six ou huit), ce qui signifie qu'une action doit être prise quand une note est demandée alors que toutes les voix sont déjà affectées à des messages de note précédents.

Généralement, les voix sont affectées par rotation stricte, de façon que la note la plus ancienne soit mise hors fonction et laisse disponible la voix pour la nouvelle note. Ce mode de fonctionnement est sélectionné par le message *polymode on.* Un message *mono mode on* dirige le récepteur de façon à affecter chacune des voix « monophoniquement » à l'un des canaux MIDI formant un groupe consécutif en commençant par le canal d'origine (de base) du synthétiseur. Le message

mode mono est envoyé sur le canal de base, et son second octet de données spécifie le nombre total de canaux requis.

Si le mode omni est en fonction, un message mono dirige simplement le récepteur afin d'affecter une voix de façon monophonique aux messages MIDI sur tous les canaux. Les *messages système* ne sont pas codés avec des numéros de canaux dans leurs octets d'état. L'octet d'état est donc reçu par toutes les unités du système. Les messages système sont répartis en trois catégories : commun, temps réel et exclusif.



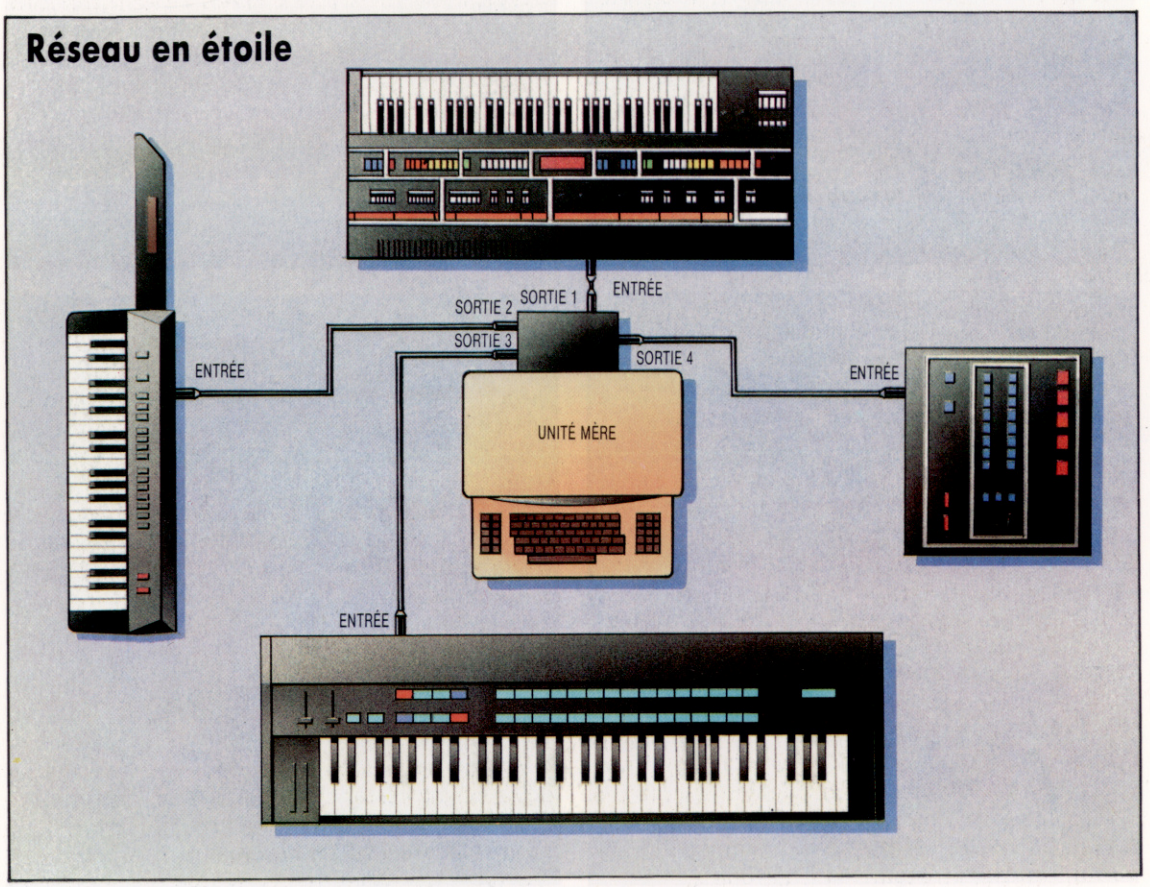
Les messages *système communs* ont des bits d'état compris entre $F1 et $F7. Ils consistent en un octet d'état, suivi de 0, 1 ou 2 octets de données.

Les messages *système en temps réel* sont destinés à toutes les unités du système. Leurs octets d'état sont compris entre $F8 et $FF et sont utilisés principalement par des batteries et des séquenceurs pour synchroniser leurs propres séquences internes à l'horloge de l'émetteur principal. La plupart des synthétiseurs ignorent ces messages, sauf s'ils possèdent une forme quelconque de séquenceur ayant un potentiel de synchronisation MIDI.

Les messages système en temps réel diffèrent des autres par le fait qu'ils consistent uniquement en un octet d'état, sans octet de données. Les messages système peuvent donc être envoyés un à la fois, même s'ils interrompent des messages d'autres types.

Les messages *système exclusifs* commencent par l'octet d'état $F0, suivi par un nombre d'octets de données, et se terminent par l'état fin-exclusif ($F7) ou par tout autre octet d'état. Le premier octet de données est un code d'identification du fabricant (ID). S'il ne correspond pas à l'ID de l'unité réceptrice, le reste du message est ignoré.

Les messages servent à transférer des données entre des instruments de type similaire, et seraient non significatifs pour d'autres unités. Le principal type de données transférées consiste en données de « programme de correction » destinées aux synthétiseurs. (Une telle transmission de programme de correction ne doit pas être confondue avec des messages de canaux $Cx, qui choisissent l'un des programmes de correction déjà stockés dans la mémoire du récepteur.) Cependant, le type de données transmises par les messages système exclusifs est choisi par les fabricants, pourvu qu'ils affectent un code ID valide.



Performance
Les premières applications de MIDI ne permettaient pas la sélection de canaux ; les synthétiseurs essayaient de répondre aux messages sur tous les canaux — ce qui rendait le système inutilisable lorsque les instruments étaient enchaînés. Ces premières unités MIDI devaient être connectées dans un réseau en étoile autour d'une unité mère, qui intégrait les circuits permettant à chaque canal d'être obtenu à partir d'une prise distincte. (Cl. Kevin Jones.)

Messages de voix de canaux

État	Données 1	Données 2	Description	Notes
$8x	p	v	Note off p = hauteur v = vitesse off	1,2
$9x	p	v	*a*) v > 0 Note on p = hauteur v = vitesse	1,2
			b) v = 0 Note off p = hauteur	1,2
$Ax	p	pr	Polyphonique après touche p = hauteur pr = pression	1,3a
$Bx	n	c	*a*) n < $7A Changement de contrôle n = numéro de contrôleur c = valeur de contrôleur	4
			b) n = $7A contrôle local on (c = $7F) off (c = $0)	5
			c) n = $7B, c = 0 Toutes notes off	6
			d) n = $7C, c = 0 Mode omni off (toutes notes off)	6
			e) n = $7D, c = 0 Mode omni on (toutes notes off)	6
			f) n = $7E Mode mono on (toutes notes off) c = nombre de canaux	6,7
			g) n = $7F, c = 0 Mode poly on (toutes notes off)	6
$Cx	pp	»	Sélection de programme de correction pp = numéro de programme	
$Dx	pr	»	Canal pression pr = valeur de pression	3b
$Ex	l	m	Changement de hauteur l = 7 BMS m = 7 BPS	8

Remarques

Sauf exception, tous les octets de données doivent avoir une valeur comprise entre 0 et $7F. Le numéro de canal est représenté par x, le chiffre hex le moins significatif de l'octet d'état (x = 0 spécifie le canal 1 et x = $F spécifie le canal 16).
1. p est la hauteur de la note en demi-tons. Le *do* de référence est le numéro de hauteur $3C, et ainsi tous les *do* sont des multiples de $0C (décimal 12). Les 88 notes de piano standard vont de $15 à $6C.
2. Les valeurs de vitesse vont de $01 à $7F. Les claviers sans capteurs de vitesse en utilisent une implicite de $40. Un message *note on* avec une vitesse de 0 est équivalent à un message *note off* avec une vitesse implicite de $40. C'est généralement la méthode utilisée par des claviers plus simples, qui n'ont aucun capteur de vitesse et tirent ainsi avantage des messages canaux.
3. « Après-touche » est la pression exercée sur une touche après qu'elle a été abaissée. Cette valeur est généralement aussi utilisée pour introduire des effets de modulation sans avoir à se servir d'un dispositif de modulation. Il existe deux types de capteurs de pression donnant deux types de messages différents.
a) Un « après-touche » individuel ou polyphonique nécessite des capteurs de pression pour chaque touche et n'affecte que la note correspondant à celle qui a été pressée. Par conséquent, un numéro de hauteur doit être envoyé en plus de la valeur de la pression. (Sa mise en œuvre nécessite également des circuits de modulation distincts pour chaque voix.)
b) Une pression de canal est produite par un seul capteur de pression, lequel est affecté par toutes les touches de façon égale. La pression transmise correspond à la pression maximale (toutes les touches maintenues abaissées en même temps). L'effet résultant est appliqué simultanément à toutes les voix.
4. Ces messages sont envoyés par des contrôleurs séparés du clavier. Les numéros qui vont de $0 à $1F sont des contrôleurs continus qui ont des valeurs (c) allant de $0 à $7F. Ceux-ci sont équivalents à des manches à balai à potentiomètre ou à d'autres dispositifs musicaux analogiques. Les contrôleurs numéros $20 à $3F sont facultativement utilisés pour envoyer 7 bits moins significatifs aux contrôleurs $0 à $1F, si une résolution très élevée est requise. Les contrôleurs numéros $40 à $5F sont des contrôleurs de commutateurs (comme des pédales de maintien, interrupteurs on/off, etc.), c étant à 0 pour off et à 1 pour on. Ceux-ci sont équivalents aux manches à balai à commutateurs, plus répandus. Les contrôleurs numéros $60 à $79 sont non définis, et ceux allant de $7A à $7F sont réservés à des messages de mode canal. Les numéros n'ont pas à être affectés à des contrôleurs physiques particuliers, sauf la roue de modulation qui est généralement affectée au contrôleur numéro $1.
5. Un contrôle local est facultativement utilisé pour fractionner la liaison interne entre les dispositifs d'entrée d'une unité (généralement un clavier et ses contrôleurs associés) et ses circuits de génération sonore, de telle sorte que le clavier (par exemple) n'envoie que des données à MIDI OUT et que les circuits de génération sonore ne répondent qu'aux données reçues dans MIDI IN.
6. La mise en œuvre de *toutes notes off* est facultative, et ces messages ne doivent pas être utilisés à la place des commandes *note off* isolées pour mettre de nombreuses notes hors fonction. Le fonctionnement de tout cela n'est pas très clair et rend les commandes inutiles, alors qu'elles auraient pu servir à mettre hors fonction les notes toujours actives lors d'une fin de séquence prématurée.
7. Le troisième octet du message de mode mono spécifie le nombre total de canaux nécessaires. Si c'est zéro, le nombre de canaux égale celui des voies disponibles au niveau du récepteur.
8. La roue de hauteur diffère des autres contrôleurs en ce qu'elle peut avoir des valeurs positives et négatives. La position centrale est envoyée comme l = $0 et m = 40. De nombreux récepteurs n'ont que 7 bits de résolution et ne répondent qu'à m, ignorant la valeur de l.

Messages système

État	Données 1	Données 2	Description	Notes
$F0	Tout nombre		Système exclusif	1
$F1			Non défini	
$F2	l	m	Pointeur de position de chanson l = 7 BMS m = 7 BPS	2
$F3	s		Sélection de chanson s = nombre de chansons	
$F4			Non défini	
$F5			Non défini	
$F6			Demande de ton	3
$F7			Fin de système exclusif	1
$F8	—	—	Horloge de synchronisation	4
$F9			Non défini	
$FA	—	—	Départ	4
$FB	—	—	Continuer	4
$FC	—	—	Fin	4
$FD			Non défini	
$FE	—	—	Détection active	5
$FF	—	—	Remise à zéro système	6

Remarques

Les messages $F8 à $FF sont des messages système en temps réel et peuvent être envoyés en tout temps (même durant la transmission d'autres messages).
1. Les messages système exclusifs peuvent avoir un nombre quelconque d'octets de données terminés par « fin de message exclusif » ($F7) ou par tout autre octet d'état.
2. Ce message sert à prérégler de façon arbitraire le pointeur de position de chanson, lequel est un registre interne contenant le nombre de battements (1 battement = 6 horloges MIDI) qui se sont produits depuis le début de la séquence (chanson).
3. Cela sert à demander aux synthétiseurs de régler leurs oscillateurs.
4. Ces messages servent à synchroniser les unités de séquence maîtresses et asservies. L'horloge système est réglée à 1/24 d'un quart de note. La commande « continue » diffère de « start » en ce qu'elle redémarre une séquence depuis le pointeur où elle s'était arrêtée plutôt qu'au début.
5. La détection active est facultativement utilisée à titre de message factice lorsque MIDI n'est occupé à aucune autre tâche. Si elle est utilisée, elle est envoyée de telle sorte qu'il n'y ait jamais plus de 300 ms sans activité. Un récepteur devrait agir normalement s'il ne reçoit jamais de détection active ; sinon, si aucune activité n'est présente pendant plus de 300 ms, il doit mettre ses voix hors fonction et revenir en fonctionnement normal.
6. Ce message sert à initialiser globalement le système dans l'état de mise sous tension. Il doit être envoyé automatiquement lors de la mise sous tension afin d'éviter que deux unités se remettent mutuellement à zéro de façon répétitive.
Tout message non listé ci-dessus ne doit pas être envoyé et doit être ignoré par les récepteurs. Une attention particulière doit être accordée à la suppression de transmissions parasites lors de la mise sous tension.

Une petite pêche

Même si l'Apricot F1e est d'abord une machine de gestion, il peut très bien réussir sur le marché de l'éducation et auprès des petites entreprises.

La composition de base du F1e est connue des personnes ayant déjà utilisé un Apricot. L'ensemble comprend le clavier F1, l'ordinateur lui-même avec une unité de disquette à simple face et un moniteur « écran vert » de 8 pouces. Comme pour l'Apricot portable, le constructeur a utilisé sa technologie infrarouge éliminant la présence des câbles. Le clavier et la souris commandent l'ordinateur par l'intermédiaire des rayons infrarouges, détectés par des capteurs situés à l'avant du F1e. Suivant la tendance actuelle (lancée par le Macintosh), la machine occupe peu d'espace sur la surface du bureau; le F1e est remarquablement étroit, puisqu'il ne mesure que 200 mm de largeur; mais il faut noter que la longueur de la machine contredit un peu cet effort (425 mm).

Le clavier adopte la disposition standard Apricot — les touches sont plates et ressemblent à celles du Sinclair QL. Le clavier ne se prête pas réellement à la frappe rapide et n'est donc pas particulièrement indiqué pour le traitement de texte. Les touches elles-mêmes, qui semblent bien adaptées aux applications de gestion, laissent planer un doute quant à leur fiabilité à long terme. Comme le portable Apricot, quatre boutons sont placés au haut du clavier : Reset pour les démarrages à froid, Repeat Rate qui vous permet de définir le rythme de répétition des caractères lorsqu'une touche est maintenue appuyée, Set Time et Keyboard Lock.

L'ordinateur possède une unité de disquette Sony de $3\frac{1}{2}$ pouces intégrée; cette disquette devient de plus en plus répandue chez les fabricants et elle explique l'utilisation rapide qu'en a fait ACT. Alors que d'autres sociétés ont des difficultés à transférer leur base logicielle sur des disquettes $3\frac{1}{2}$ pouces, tout le logiciel Apricot est déjà compatible. Les utilisateurs désirant acheter le F1e n'ont donc aucun souci à se faire au sujet de la disponibilité des logiciels. De plus, les disquettes simple face, simple densité du F1e peuvent contenir un maximum de 315 K d'information, plus que leurs équivalents en format $5\frac{1}{4}$ pouces.

Une série de DEL apparaît également sur le panneau avant, indiquant l'état de l'alimentation, du verrouillage des majuscules, du défilement et de l'unité de disquette. Les détecteurs de faisceaux infrarouges sont situés juste au-dessous.

Un bus d'extension à 60 voies est placé sur le côté droit de l'ordinateur; ce connecteur permet de brancher une série de cartes d'extension, ou une unité de disquette supplémentaire. Cet emplacement permet de porter la capacité du F1e à celle de l'Apricot en ajoutant le système d'extension MSD, qui permet d'utiliser 10 mégaoctets sur disque dur.

Le panneau arrière, qui comporte peu d'interfaces, est suffisamment bien conçu pour permettre la connexion des périphériques les plus répandus. A l'extrême gauche, on reconnaît le connecteur D à vingt-cinq broches de l'interface série RS232. Ce port est flanqué de deux prises de moniteur. La première est un adaptateur à neuf broches qui fournit un signal RVB pour les moniteurs couleur (bien que les moniteurs

Belle apparence
Le F1e a aussi belle apparence que ses cousins. Avec l'unité de disquette intégrée et le système d'exploitation MS-DOS, le F1e est un réel concurrent sur les marchés de l'éducation et des utilisations domestiques. (Cl. Chris Stevens.)

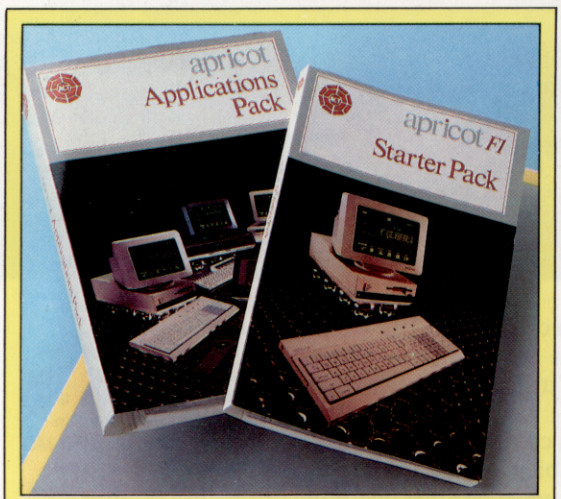
monochromes ACT se branchent eux aussi dans cette prise). Sur la droite, une prise vidéo composite est destinée à d'autres types de moniteurs.

Le prix de base du F1e n'inclut pas le moniteur. L'ordinateur ne possède pas d'adaptateur HF qui permettrait de le brancher à un téléviseur standard ; cependant, il est possible de se procurer cet adaptateur. Mais, comme la version de base du F1e ne comporte pas de moniteur et que plusieurs moniteurs ont leur propre entrée d'alimentation, les prises écran ne fournissent pas automatiquement les lignes d'alimentation, et l'ordinateur lui-même ne peut fournir une telle source. Il est donc nécessaire d'utiliser une alimentation externe de 17 V pour pouvoir utiliser l'un des moniteurs ACT.

Il apparaît assez illogique et contradictoire que le constructeur ACT ait décidé de faire disparaître les câbles du clavier et de la souris tout en continuant à utiliser une alimentation externe pour ses moniteurs. Cela est particulièrement vrai puisque d'autres modèles de la gamme ACT ne présentent pas un tel dispositif. L'alimentation peut bien sûr être placée de façon à ne pas occuper de place sur le plan de travail, mais il s'agit quand même ici d'une erreur au niveau de la conception de la machine.

A l'intérieur de la machine, une autre fente d'extension permet l'installation d'autres cartes. La carte principale du F1e, comme celles des autres ordinateurs ACT, est bien conçue et est entièrement protégée de l'unité de disque et du transformateur d'alimentation (qui peut générer des températures assez dangereuses pour les circuits) par un boîtier métallique qui sert de dissipateur thermique.

Le F1e est construit autour du processus 16 bits 8086 et tourne sous le système d'exploitation MS-DOS, très répandu. Les ordinateurs Apricot ont cependant leur propre système de gestion d'icô-



Super logiciel
Les programmes fournis avec la machine sont les suivants : un programme de traitement de texte, un carnet électronique et un tableur. Bien que les deux premiers ne soient pas des plus évolués, le tableur offre de nombreuses fonctions, tel le découpage, qui ne sont généralement offertes que par des tableurs coûtant assez cher.

Puce Z80 SIO
Cette pouce fournit les signaux d'horloge permettant la synchronisation des interventions de la puce d'E/S série.

ROM du F1
Ces ROM contiennent l'amorçage, l'autodiagnostic et les routines BIOS du système.

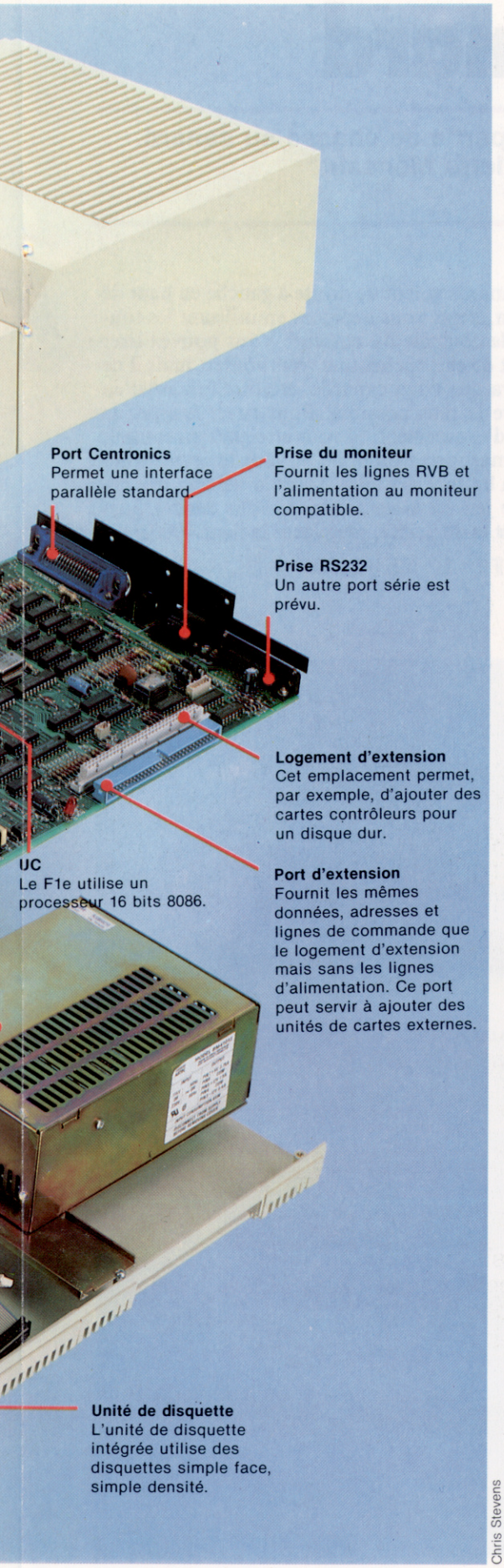
Puces RAM
Le F1e a 256 K de RAM.

Contrôleur de l'unité de disquette
Cette puce logique construite sur demande commande les interventions de l'unité de disquette.

Bloc d'alimentation
Le bloc d'alimentation comprend un transformateur de secteur et une prise de 17 V (courant alternatif) où doit être fournie la source d'alimentation du moniteur.

Voyants indicateurs
Ces voyants indiquent les conditions de mise sous tension, de défilement et d'accès disquette.

Récepteur infrarouge
Le clavier sans fil du F1e utilise une transmission infrarouge pour communiquer avec l'unité principale.



Chris Stevens

nes, ce qui signifie que vous n'aurez que très rarement accès directement au système d'exploitation. Le système de gestion de l'Apricot se nomme Activity, et c'est à ce niveau que les utilisateurs commandent leur machine.

Activity est un programme orienté vers les objets. Cela signifie qu'au lieu d'émettre une série de commandes à l'ordinateur, comme LOAD pour charger un fichier en mémoire, vous n'avez qu'à indiquer (en sélectionnant généralement une icône sur l'écran) quelle fonction vous désirez exécuter.

La plupart de ces systèmes dépendent de l'utilisation d'une souris qui permet de déplacer le curseur sur l'écran; mais, sur le système F1e, le clavier numérique peut être utilisé. Les chiffres 1 à 9 sont disposés sur trois rangées de trois touches. Il est donc possible d'utiliser huit touches du clavier numérique, en excluant la touche centrale, pour définir huit directions de déplacement du curseur. Il suffit donc d'appuyer sur la touche 8 pour diriger le curseur vers le haut. En appuyant sur la touche 3, le curseur se déplace diagonalement vers le bas et vers la droite. Pour sélectionner une icône, vous devez appuyer sur la touche ENTER du clavier numérique.

Parmi la série de programmes contenus sur la disquette système Activity, un programme « tutoriel » vous permet de découvrir l'ensemble du produit. Il explique également l'utilisation des icônes et des éditeurs de polices de caractères (qui vous permettent de charger vos propres jeux de caractères) et le configurateur du système, grâce auquel vous pouvez installer presque toutes les unités périphériques que vous souhaitez.

Logiciels associés

Comme vous pouviez le prévoir, cette machine à usage professionnel est livrée avec plusieurs programmes. SuperWriter est un programme de traitement de texte fondé sur WordStar, mais sans fonctions de mise en page évoluées. SuperPlanner est un « carnet électronique » qui vous permet d'organiser vos activités. Ce programme comporte également un carnet d'adresses, un calendrier, un agenda et un petit système de classement. Bien que ressemblant un peu à un programme de base de données, SuperPlanner ne comporte pas les fonctions de recherche et d'extraction offertes par un véritable programme de ce type. Le dernier programme inclus dans cet ensemble est SuperCalc. Il s'agit d'un tableur pour des applications financières et comptables. Ce programme est sans doute le plus complet de la série. Il comporte de nombreuses commandes permettant de définir des fenêtres, de justifier des textes, etc. Ce programme est un véritable logiciel professionnel.

En réduisant le prix du F1e, ACT annonce sa volonté d'attaquer le marché éducatif. Cette volonté est confirmée par le fait que, en plus de réduire le prix du F1e, ACT a annoncé un nouveau produit nommé B-TRAN, qui permet à l'ordinateur d'exécuter presque tous les programmes écrits en BBC BASIC.

Apricot F1e

PRIX
*** (moniteur non compris).

DIMENSIONS
425 × 200 × 105 mm.

UC
Processeur Intel 8086 fonctionnant à 4,7 MHz.

MÉMOIRE
256 K standard.

ÉCRAN
Résolution de texte 132 × 50 ou 80 × 25 caractères, ou résolution graphique maximale de 800 × 400 points.

INTERFACES
Port RS232, interface Centronics et prises vidéo RVB et composite.

UNITÉS DE DISQUETTES
Une unité de 315 K de $3\frac{1}{2}$ pouces.

SYSTÈME D'EXPLOITATION
MS-DOS, CP/M-86 et CP/M.

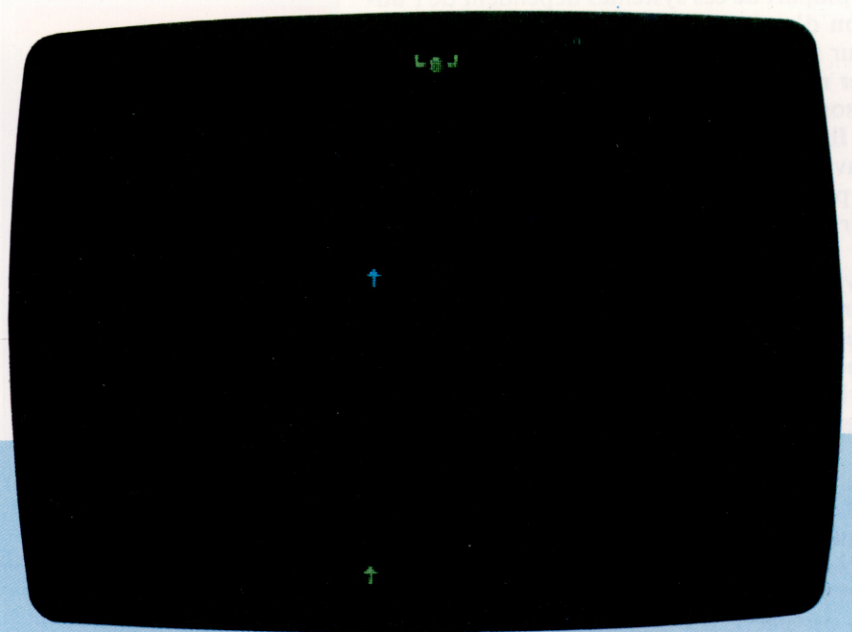
DOCUMENTATION
Le manuel des applications est complet et cohérent, bien que le manuel pour débutants ne fournisse que très peu de détails concernant la machine elle-même.

FORCES
A ce prix, le F1e semble être une bonne affaire et pourrait remporter un réel succès au niveau de l'enseignement et des petites entreprises.

FAIBLESSES
Le clavier plat n'a pas la qualité de celui de l'ACT Apricot. Bien qu'il possède un processeur rapide, le F1e n'est pas aussi rapide que certaines machines comparables.

Chasse au canard

Votre micro-ordinateur vous invite à une partie de chasse au canard. Laissez-vous tenter par ce jeu écrit par Pierre Monsaut pour le Commodore 64.



Les canards volent de droite à gauche en haut de l'écran. Vous vous déplacez en utilisant les touches de contrôle du curseur. Vous pouvez tirer autant de cartouches que vous voulez, mais il ne passera que vingt canards, et votre but est d'en abattre le plus possible. Pour tirer, frappez la barre d'espacement. Si vous atteignez un canard, vous marquerez un point et vous le verrez tomber en battant des ailes. Dans la version présentée, le jeu est assez facile. Si vous désirez augmenter la difficulté, remplacez la ligne 4000 par :

```
4000 IF T - C = 1025 THEN 5000
```

```
5 REM  xxxxxxxxxxxxxxxxxxxx
10 REM x CHASSE AU CANARD x
15 REM xxxxxxxxxxxxxxxxxxxx
20 GOSUB 1000
100 X=-X
110 Y=X+0.5
120 C=C-1
130 IF C<0 THEN GOSUB 3000
140 IF NC=0 THEN 4500
150 PRINT CL$;TAB(C);CC$(INT(Y))
160 PJ=PJ+D1
170 IF PJ<PN THEN PJ=PN
180 IF PJ>PM THEN PJ=PM
190 POKE P1,CR
200 POKE PJ,CJ
210 POKE PJ+M,JC
220 P1=PJ
230 IF ABS(T-PJ)>2 THEN T=T-80:GOTO 250
240 T=PJ
250 IF T<1064 THEN 4000
260 POKE T,CJ
270 POKE T+M,CJ
280 IF T<>PJ THEN POKE T+80,CR
290 GET X$
300 D1=2*((X$=G$)-(X$=D$))
310 IF X$<>" " OR T<>PJ THEN 100
330 T=PJ-80
340 GOTO 100
1000 DIM CC$(1)
1010 CJ=30:JC=5:PN=1985:PM=2022
1030 CC$(0)=CHR$(173)+CHR$(113)+CHR$(189
)+" "
1040 CC$(1)=CHR$(176)+CHR$(113)+CHR$(174
)+" "
1050 PJ=2004:NC=20:T=PJ:X=0.5:C=37
1100 M=54272:CL$=CHR$(19):CR=32:P1=PJ
1130 D$=CHR$(29):G$=CHR$(17)
2000 PRINT CHR$(147);CHR$(30);
2010 POKE 53280,0:POKE 53281,0
2040 RETURN
3000 PRINT CL$;"    ";
3020 FOR I=1 TO 200:NEXT I
3040 C=37:NC=NC-1
3060 RETURN
4000 IF ABS((T-1024)-(C+1))<=1 THEN 5000
4010 POKE T+80,CR
4020 T=PJ
4030 IF NC<>0 THEN 260
4500 PRINT CHR$(147)
4510 FOR I=1 TO 10
4520 PRINT:GET X$
4540 NEXT I
4550 PRINT TAB(13)"SCORE : ";S
4560 FOR I=1 TO 10
4570 PRINT
4580 NEXT I
4590 PRINT TAB(13)"UNE AUTRE ?"
4600 GET X$
4610 IF X$="" THEN 4600
4620 IF X$<>"N" THEN RUN
4630 END
5000 S=S+1
5010 D=C
5020 C=37
5030 PRINT CL$;
5040 FOR I=1 TO 22
5050 X=-X
5060 Y=X+0.5
5070 PRINT TAB(D);"   "
5080 PRINT TAB(D);CC$(INT(Y))
5090 PRINT CHR$(145);
5100 FOR J=1 TO 50
5110 NEXT J
5120 NEXT I
5130 NC=NC-1
5140 FOR I=1 TO 500:NEXT I
5160 PRINT CHR$(147);
5170 IF NC=0 THEN 4500
5180 GOTO 100
```

Plus PC que moi...

Véritable 16 bits, doté du processeur 80186 travaillant à 8 MHz, le Goupil G4 associe le MS-DOS au gwbasic, et il est entièrement compatible aux niveaux logiciel et matériel.

Goupil/S.M.T.

Carte compacte
Le Goupil G4 se présente sous la forme d'une console regroupant, autour de sa carte mère très compacte, le ou les lecteurs, la carte d'affichage et un bus d'extension.

Le Goupil G4, mini-ordinateur français haut de gamme, n'est pas particulièrement destiné à une utilisation domestique. Son microprocesseur Intel 80186 a ses bus de données interne et externe sur 16 bits. Il peut donc traiter de manière interne et communiquer (transmettre et recevoir des informations) par des voies de données de 16 bits. C'est en cela qu'il est un vrai 16 bits. Le jeu d'instructions du microprocesseur Intel 80186 est optimisé et occupe vingt fois moins de composants que son prédécesseur le 8086, pour une efficacité accrue. Sa vitesse de travail de 8 MHz lui permet en effet une vitesse d'exécution élevée.

Cette puissance est confirmée par une grande capacité d'adressage mémoire, de 256 K. La mémoire figure sur la carte mère et est directement extensible à 512 K, sans adjonction de carte. La carte mère est disposée à plat, en fond de bloc de console. Celle-ci est compacte du fait même de cette disposition à plat des cartes. Outre la mémoire centrale, la carte mère comprend le microprocesseur bien sûr, et deux ports d'entrée/sortie de types série RS232C et parallèle Centronics.

La console contient en standard deux cartes (une troisième pouvant venir s'insérer pour le contrôleur d'un disque dur de 10 mégaoctets) : l'une est la mère; l'autre, destinée à l'affichage, permet la gestion directe d'un moniteur couleurs, sans ajout de carte spécifique; on économise ainsi

L'avenir : la dimension réseau

Du 8 bits Flex au 16 bits MS/DOS, la société française constructeur de micro-ordinateurs S.M.T. Goupil a suivi l'évolution du marché de la micro-informatique. Société créée en 1979 par des anciens de la D.G.T. (Direction générale des télécommunications), la S.M.T. a su passer de l'époque du « fer à souder » à celle de la haute technologie incorporant les principaux standards mondiaux. C'est du reste l'évolution elle-même de la micro, marquée au départ par une nébuleuse de systèmes d'exploitation (Flex, CP/M 80, CP/M 86, USED, Prologue), et de microprocesseurs (6809 Motorola, Z80 Zilog, 8088 Intel), pour aboutir à l'adoption, imposée par l'entrée en lice de l'IBM PC, de standards de fait : le processeur 80186 d'Intel et le système d'exploitation MS-DOS. Si la spécificité de la machine y a perdu, l'outil en est devenu plus universel et d'avant-garde (ouverture vers des systèmes de communication d'entreprise). C'est aussi la fin du « rêve » du « micro-ordinateur pour tous », les applications devenant de plus en plus professionnelles. L'ordinateur n'est pas devenu un objet de consommation courante, même s'il est sous-jacent dans la vie quotidienne et risque de revenir en force par le biais des télécommunications.
La S.M.T. vient donc de sortir un nouvel ordinateur, point de départ d'une nouvelle génération : le Goupil G4.

une extension. Cette carte vidéo unique interprète donc intégralement les logiciels graphiques en monochrome ou en couleurs. Les modes possibles sont de 25 lignes par 80 colonnes dans une page de 720 × 350 pixels (mode mixant texte, graphique et couleurs avec le mode texte haute définition), ou de 640 × 200 pixels (graphique monochrome haute définition), 320 × 200 pixels (mode graphique couleurs en quatre couleurs dans une palette de seize).

La console comporte également la mémoire morte de 16 K contenant le BIOS (gestion des entrées/sorties) et la mémoire de masse. Les lecteurs (trois versions possibles : un lecteur 5 pouces, deux lecteurs 5 pouces, un lecteur 5 pouces et un disque dur) sont en effet inclus dans la console, ce qui permet un gain de place appréciable et contribue à l'aspect compact du G4. Enfin, la console intègre un bus d'extension situé à l'arrière et relié par un câble plat à la carte mère. Il permet en version de base l'adjonction directe de deux cartes optionnelles au format standard des compatibles.

Car le Goupil G4 est entièrement (matériel et logiciel) compatible IBM PC. Plus précisément, cela signifie que :

— le contrôleur disque de la carte mère doit gérer le même format disque que l'IBM PC (compatibilité de support);

— le microprocesseur doit être fonctionnellement identique (le 80186 l'est vis-à-vis du 8088 utilisé par l'IBM PC) [compatibilité de processeur];

Goupil/S.M.T.

Omninet et MS/NET
Le Goupil G4 peut se constituer en réseau, par Goupilnet. Premier réseau local en France, il réunit les standards matériel Omninet (Microsoft) et logiciel MS/NET intégré à MS/DOS 3.1. Il marque une étape dans la création de réseaux locaux réunissant des terminaux compatibles de faible coût.

— le système d'exploitation doit être le même (c'est bien le cas avec le MS/DOS);
— la structure du bus doit être identique pour recevoir les cartes IBM et compatibles;
— les codes de caractères doivent s'afficher de manière identique (compatibilité du clavier);
— l'interface vidéo doit être la même, le moniteur devant être adressé de manière identique;
— la compatibilité des entrées/sorties doit être assurée pour permettre l'adjonction de tous les périphériques de l'IBM PC.

Le clavier du G4 est donc compatible, ce qui signifie qu'il a dû suivre celui de l'IBM PC. Néanmoins, plus compact (les 10 touches de fonctions sont placées sur une seule ligne tout en haut), il est doté de 56 touches clavier machine à écrire avec touches de contrôle, et d'un bloc numérique servant aussi au déplacement du curseur. Il est à encodage positionnel, ce qui autorise sa reprogrammation complète et permet l'utilisation de toutes les banques de caractères et des langages symboliques tels que l'APL. Le clavier existe en six versions nationales.

L'écran est de 12 pouces de diagonale, version monochrome ou couleurs. Il affiche à la fois du texte et du graphique. Comme nous l'avons vu, la carte vidéo unique interprète intégralement les logiciels graphiques en monochrome (quatre niveaux de gris) ou en couleurs.

Les extensions sont pour une bonne part intégrées au Goupil G4, extension de la mémoire sur la carte mère (512 K), que l'on peut étendre, par le bus d'extension, à 640 K. Ce dernier peut recevoir au maximum deux cartes. Parmi les cartes particulièrement intéressantes :
— une carte extension mémoire 128 K;
— une carte multiprotocole asynchrone/synchrone pour connexion de Goupil G4 à de gros systèmes (Bull, IBM, etc.);
— une carte réseau local Goupilnet.

Le réseau local Goupilnet réunit les deux standards du marché :
— MS/NET intégré dans la version 3.1 de MS/DOS assure la compatibilité et le fonctionnement des logiciels du marché. C'est la couche logiciel et d'exploitation des programmes de service;
— Omninet, couche physique de Goupilnet, est transparent à l'utilisateur. C'est le réseau le plus utilisé au monde (plus de cent vingt mille terminaux connectés).

Pour être le premier réseau local en France, Goupilnet a mis au point le protocole de transport ISO. Jusqu'à soixante-quatre postes peuvent être connectés; deux terminaux peuvent être distants de 300 m, et le réseau peut atteindre 1 200 m au maximum (utilisation de répéteurs actifs). Le partage des ressources se fait sur une paire torsadée à une vitesse de 1 Mg/b/s. Étant conçu sur des standards logiciel et matériel, Goupilnet permet l'utilisation des principaux terminaux du marché. Une horloge et un calendrier internes sauvegardés, mais aussi l'alimentation 130 V à découpage, autorisent ces extensions.

Le logiciel venant avec le G4 suit la même tendance à l'intégration que le matériel. Goupil G4 est livré en standard avec le système d'exploitation MS-DOS version 2.11, et le langage GWBASIC, tous les deux de Microsoft. L'originalité de la politique logiciel de la S.M.T. tient à la fourniture en standard du programme MS-WIN, toujours de Microsoft. WINDOWS est un programme intégrateur à fenêtres et à icônes. Il permet d'utiliser de manière simple et très lisible certains grands logiciels d'application. Il comporte en outre ses propres utilitaires (bloc-notes, Paint [programme graphique], traitement de texte, module de télécommunications, utilitaires divers [calculette, horloge...]).

Nouvelle orientation du marché de la micro-informatique qui adopte les principaux standards de fait (IBM pour le matériel et Microsoft pour le logiciel), le G4 n'en est pas moins original et pionnier : par sa puissance, son haut degré d'intégration, son environnement graphique (MS-WIN), son réseau local et sa polyvalence due finalement à la compatibilité.

Goupil/S.M.T.

Dernier cri
Véritable 16 bits compatible, doté en standard de 256 K, de ports série RS 232C et Centronics, il permet la gestion directe d'un moniteur couleurs. Livré avec le GWBASIC et le logiciel intégrateur MS-WINDOWS, il utilise les derniers développements de la micro pour un usage professionnel.

Planifier l'action

Nous étudions les procédures des générateurs de programmes Sycero et The Last One. En insistant sur la préorganisation, ces systèmes vous aident à générer des programmes basic concis.

Tout programme destiné à permettre à un néophyte de générer des applications doit satisfaire à un certain nombre de conditions très strictes :

— il doit encourager une approche arborescente nécessaire à la planification du projet et au développement de chacun de ses points ;

— il doit fonctionner par menus, avec messages indiquant les divers choix pour guider l'utilisateur ;

— il doit piéger les erreurs au fur et à mesure de leur apparition et permettre des corrections ou deuxièmes choix ;

— le code résultant doit être transférable, indépendamment de l'ordinateur sur lequel il a été généré ;

— il doit être à même de fournir d'amples informations sur le processus de génération des programmes afin qu'un utilisateur puisse apporter ultérieurement des modifications et améliorations.

Néanmoins, dans la mesure où un générateur de programmes est un outil qui concerne également les utilisateurs professionnels, il doit pouvoir répondre aussi à leur attente. Un programme de niveau trop bas serait un handicap pour les utilisateurs expérimentés ; il retarderait le traitement avec des mises en garde et des dépistages d'erreurs. En outre, une utilisation professionnelle suppose des routines incorporées, des traitements BASIC spécifiques par exemple. En ce sens, les générateurs TLO et Sycero — plus récents — sont très performants.

Des menus d'information sont mis en œuvre dans les deux cas, et disponibles d'un bout à l'autre du processus de génération des programmes. Sycero a des menus d'aide concernant les mouvements du curseur et le graphisme. Les commandes d'édition de texte sont à tout moment disponibles par les touches CTRL et H. Des deux systèmes, TLO est le plus arborescent. Il procède en mettant au point un organigramme, suite de REMarques indiquant ce que doit faire chaque partie du programme (Branchement sur l'option 4 du menu par exemple) ; ces remarques peuvent être incorporées si nécessaire dans le programme BASIC final.

Les écrans de saisie et d'affichage se définissent lors du codage, le code résultant étant au format ASCII. L'utilisateur quitte alors le générateur, charge (LOAD) le programme généré, et le resauvegarde (SAVE) en code binaire. Avec Sycero, la définition du fichier et le dessin des écrans sont les premières choses à faire. Viennent ensuite le traitement des variables d'entrée (qui adaptent les messages d'aide et d'erreur selon la saisie de l'utilisateur), les définitions des rapports et quelques autres options. Les divers modules sont ensuite liés juste avant le codage.

Bien que cette approche soit moins conviviale pour l'utilisateur, elle présente l'avantage d'obliger à une définition très méticuleuse du projet. Faute de quoi le code ne serait pas exécutable. En outre, si le générateur dépiste des erreurs lors du codage, il stoppe le processus et les indique à l'utilisateur. Puis vient enfin la sauvegarde sous la forme d'un fichier binaire exploitable dans l'environnement de Sycero.

Les deux générateurs produisent un code entièrement transférable. En fait, si le programme TLO est suffisamment court, il est certainement possible d'exécuter la version ASCII sur une machine au standard MSX (pour autant qu'il reste dans les limites de la mémoire du micro). Les deux programmes sont néanmoins assez volumineux, du fait des routines très sophistiquées de dépistage d'erreurs qu'ils mettent en place dans les programmes générés.

Certification du disque de travail

Comme pour tous les logiciels MS-DOS, TLO et Sycero doivent être adaptés aux matériels (configurés) par programmes INSTALL utilisables sur doubles lecteurs ou disques durs. TLO doit également certifier un disque de travail, processus semblable au formatage. Le disque de travail peut être soit la totalité d'une disquette, soit un sous-répertoire sur disque dur (ce dernier est certifié lors de son installation). La certification est vitale au bon fonctionnement du générateur. En effet, si le programme ne peut détecter une zone sur le disque de travail lors de modifications de l'organigramme, il ne peut localiser ce dernier. Ce qui suppose alors d'avoir à remonter au tout début, ou d'avoir à manipuler directement du code BASIC. Comme toujours, vous pouvez vous prémunir contre ce genre d'accidents en faisant de fréquentes copies de sauvegarde du disque utilisé.

Après installation, l'utilisateur de TLO se voit proposer un menu d'écran de huit options :

Création de programme	Enquête
Modification de programme	Certification d'un nouveau disque
Modification de fichier	Reprise du codage
Définition de fichier	Retour au BASIC

Vous choisissez la première option et répondez à la question : votre programme a-t-il besoin de fichiers ?

DIY Kits
The Last One (TLO) et Sycero permettent à l'utilisateur de générer des programmes BASIC exécutables indépendamment de l'ordinateur hôte. Cependant, la définition précise des besoins de l'utilisateur suppose une préparation très poussée si l'on veut pleinement tirer profit de ces deux générateurs d'applications.

Si vous répondez oui, vous devez définir les fichiers avec leurs zones et leurs types de zones (alphanumérique, numérique ou de données). L'écran affiche ensuite le menu de création d'organigramme. La dernière version de TLO propose vingt options numérotées de 1 à 12 et de 14 à 21 (les auteurs de TLO sont superstitieux) :

Lister l'organigramme	Effacement
Modifier l'organigramme	Attribution des pointeurs de fichier
Coder le programme	Lecture de fichier
Fusionner des organigrammes	Écriture de fichier
Avorter l'exécution	Recherche ou tri sur fichier
Saisie au clavier	Fusion
Afficher des données	Vérification d'enregistrement
Branchements	Effacement de fichier
Calculs	Fonctions de base de données
Fonctions spéciales	Multifonctions

Vous développez l'organigramme à l'aide de ces fonctions. Par exemple, pour un fichier de noms et d'adresses :

Ian McKinnell

1 . . Branchement sur le menu à trois options
2 . . Positionner le pointeur sur la fin du fichier adresses
3 . . Entrée au clavier destinée au fichier adresses
4 . . Écrire des données au fichier adresses
5 . . Demander <Avez-vous fini ?>. Branchement si « non »
6 . . Branchement inconditionnel direct
7 . . Branchement sur le menu à trois options
8 . . Positionner le pointeur en début du fichier adresses
9 . . Recherche au clavier sur le fichier adresses
10 . . Afficher des données du fichier adresses
11 . . Demander <Autre recherche ?>. Branchement si « oui »
12 . . Branchement direct inconditionnel
13 . . Tri sur le fichier adresses
14 . . Positionner le pointeur au début du fichier adresses
15 . . Lire des données du fichier adresses
16 . . Afficher des données du fichier adresses
17 . . Branchement direct inconditionnel
18 . . Terminer

A B C D E F G H I J K L

Le codage peut alors commencer. Jusqu'ici, vous ne pouviez quitter le programme sans tout perdre. Quand vous avez commencé le codage, en revanche, vous pouvez sortir du programme et le réintégrer par l'option reprise du codage. La destination des branchements est précisée — l'option n° 2 (écrire les données) sera par exemple choisie au premier menu, ou encore l'option n° 7 (lire les données), ou n° 18 (terminer). Les écrans sont enfin dessinés.

Il est conseillé de sauvegarder les écrans, dans la mesure où ils sont susceptibles d'être réutilisés dans d'autres programmes, ou modifiés. Rien, cependant, dans le logiciel TLO et son manuel, n'est dit à ce sujet. Avec Sycero, la sauvegarde des écrans (SAVE) est automatique.

Menu d'ouverture de Sycero

Le menu d'ouverture de Sycero propose treize options, utilisées plus ou moins dans l'ordre où elles figurent :

Configuration du système
Initialisation
Définition de la zone fichier du système
Définition d'écran
Traitement d'écran
Définition des rapports
Traitement des rapports
Définition du programme
Génération d'un programme
Création d'un fichier de données réelles
Exécution d'un programme généré
Utilitaires
Fin de session

La plus grande partie du travail est faite lorsque vous en êtes à l'option génération du programme. Le processus qui suit est assez simple et ne demande que peu d'interventions, à moins que le générateur ne détecte des erreurs. Ce dernier ne peut néanmoins deviner vos intentions, et il est donc impératif de lui donner des instructions très précises. La principale différence entre les deux générateurs d'applications tient à leur approche. Comme le manuel de Sycero le dit, « la meilleure manière de développer une application est d'adopter une démarche arborescente. Commencez par définir les grandes lignes, et développez ensuite les détails en allant du général au particulier.

Parce qu'il commence par la génération de l'organigramme, TLO est plus fidèle à cette démarche classique d'arborescence, même s'il encourage ainsi de manière paradoxale la planification de but en blanc de l'organigramme. Mais Sycero est lui-même inutilisable si l'on ne fait pas de même, bien qu'il autorise des modifications en cours de processus.

Sycero et TLO fournissent tous les deux un moyen pratique de générer des programmes en BASIC susceptibles d'être exploités par eux-mêmes sur la machine hôte. Malgré leurs limites, ils peuvent se révéler utiles pour élaborer des utilitaires de bases de données ou des programmes d'évaluation de jeux de questions-réponses.

Lignes à crochets

L'Interface 1 apporte au Spectrum Sinclair l'accès à quelques routines en langage machine qui peuvent nous servir dans nos programmes. Voici quelques-unes de ces routines.

L'Interface 1 contient 8 K de ROM, qui occupent la première partie de la mémoire. Elle fournit les programmes nécessaires pour commander les divers dispositifs supplémentaires (tels les Microdrive), ainsi que pour étendre l'interpréteur BASIC afin d'accepter des commandes telles que CAT et FORMAT. Cette ROM est usuellement appelée *ROM fantôme*.

Il existe différentes versions de cette ROM, et il y a des différences majeures entre la première version et celle qui est couramment produite (version 2). La version 1 était assez peu efficace pour les Microdrive, et plusieurs des opérations supposées disponibles n'y étaient pas.

Les versions ultérieures de la ROM sont plus efficaces dans le maniement des Microdrive : elles nous allouent plus d'espace sur une cartouche Microdrive. Les bogues ont également été éliminées, et quelques nouvelles possibilités d'utilisation avec des imprimantes série incorporées.

Ces différences de ROM sont généralement « transparentes » à l'utilisateur, pourvu que l'on accède à l'Interface à l'aide de BASIC ou des techniques de programmation classiques en langage machine. Toutefois, si l'on tente d'appeler directement les routines ROM fantôme, il peut se produire des problèmes, car différentes routines sont situées à différentes adresses dans les diverses versions de la ROM. Toute adresse spécifique de ROM fantôme que nous donnons sera valable pour la ROM version 1. Comment la ROM fantôme occupe-t-elle les huit premiers Koctets d'espace mémoire lorsque la ROM BASIC se trouve aussi dans cette zone de la mappe mémoire ? Chaque fois que le Z80 essaie de lire une instruction à partir des adresses &08 ou &1708 avec l'Interface 1 connectée, la ROM BASIC est momentanément « dépaginée » et la ROM fantôme prend la relève pour les opérations à venir. Il y a une routine dans la ROM fantôme pour resélectionner la ROM BASIC principale.

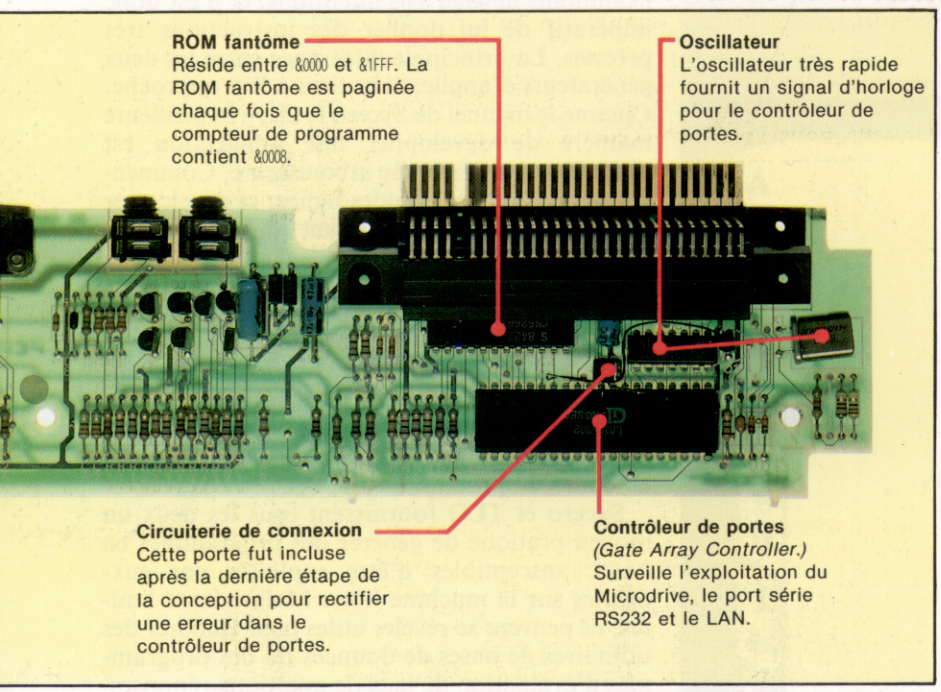
Avant tout, l'Interface 1 doit être initialisée. Cela se fait quand :

1. Une commande NEW est émise avec l'Interface 1 connectée.
2. La ROM fantôme est paginée pour la première fois.

Le résultat pratique de ce processus d'initialisation est qu'un nouveau groupe de variables système est fixé. Celles-ci occupent la RAM à partir de la fin des anciennes variables système jusqu'au début de la zone canaux — plus de 50 octets d'espace. Cela, ainsi que l'établissement de la mappe Microdrive et d'autres informations nécessaires aux divers appareils supportés par l'Interface 1, est responsable du fait que, avec cette interface connectée, le début du texte programme BASIC est déplacé vers le haut de la mémoire. C'est pourquoi il est recommandé de stocker le langage machine dans des instructions REM en ligne 1, une fois que l'interface est connectée (parce que la ligne 1 du programme n'est plus à une position constante en mémoire).

Comme nous l'avons déjà mentionné, l'Interface 1 nous procure aussi des commandes BASIC supplémentaires. En réalité, nous pouvons même y ajouter les nôtres, comme nous le verrons plus tard. Toutefois, à ce stade, nous nous contenterons d'un aperçu de commandes comme CAT et FORMAT. Sur le Spectrum non étendu, celles-ci échapperaient aux différentes vérifications effectuées par l'interpréteur BASIC, et elles engendreraient donc normalement une erreur. Cela se fait à l'aide de l'instruction RST &08, qui donne accès à l'adresse &08. Cependant, quand l'Interface 1 est connectée, cette instruction paginera la ROM fantôme, ce qui donnera à la partie de BASIC en cause une « seconde chance » pour voir s'il s'agit d'une commande qu'elle peut interpréter correctement. Si oui, cela fera l'affaire. Des commandes (comme la suivante, qui sauvegarde le programme appelé Fred sur Microdrive) fonctionnent de manière analogue.

Travaux internes
Voici les principales composantes du circuit imprimé de l'Interface 1. Il fournit Microdrive, l'interface série et les possibilités LAN mais rend aussi possible l'addition de commandes BASIC supplémentaires définies par l'utilisateur. Nous traiterons ultérieurement des commandes BASIC définies par l'utilisateur.



```
SAVE *«m»;1;«Fred»
```

La partie * de la commande engendre une condition erronée, faisant ainsi passer le contrôle à l'adresse &08 — et donc dans la ROM fantôme, où a lieu l'interprétation correcte de l'instruction.

Voyons maintenant les routines supplémentaires en langage machine fournies par l'Interface 1. Nous examinerons ce qu'on pourrait appeler les routines d'« intérêt général » : routines ROM fantôme qui ne sont pas concernées par le contrôle du Microdrive, l'interface série ou les facilités LAN.

Codes crochets

Le premier problème est d'appeler ces routines ; la ROM fantôme n'est active qu'après un appel à l'une ou à l'autre des adresses que nous avons mentionnées. Nous résolvons cela en faisant un appel à l'adresse &08 à l'aide des instructions suivantes :

```
RST   &08
DEFB  nn
```

où nn est une valeur comprise entre &1B et &32 (les valeurs extérieures à cet intervalle engendrent des messages d'erreur). Ces valeurs de nn sont appelées codes crochets, et chacune d'elles appelle une routine ROM fantôme différente. Les effets des codes crochets sont demeurés inchangés dans les différentes versions de la ROM fantôme. Avant de détailler ces appels, il faut faire quelques remarques.

1. Les opérations de code crochet ont tendance à altérer tous les registres ; il est donc nécessaire de préserver ceux que vous allez utiliser après l'appel. De plus, il est recommandé de sauvegarder le registre HL, puisque cela servira pour le retour au BASIC.

2. Certaines des routines code crochet invalident les interruptions. Revalidez-les si vous n'êtes pas certain.

3. Quand on utilise un appel code crochet, la valeur &5C3A doit se trouver dans la paire de registres IY.

4. Avant d'utiliser un code crochet, il est recommandé de s'assurer que les variables système de l'Interface 1 ont été fixées. Il y a, en fait, un code crochet pour faire cette tâche à votre place, et nous allons voir cela tout d'abord.

• *Code crochet 49* (&31). Il est responsable de la création, ou de l'insertion, des variables système de l'Interface 1. Si vous n'êtes pas sûr qu'elles ont été créées, vous devez émettre l'appel suivant avant de tenter d'utiliser l'un ou l'autre des codes crochets :

```
RST   8
DEFB  49
```

Une fois qu'une routine code crochet a été exécutée par la ROM fantôme, alors le contrôle repasse à la ROM principale.

La ROM fantôme nous fournit aussi quelques routines qui existent déjà dans la ROM BASIC sous une forme moins pratique. Elles manient les entrées et sorties écran et clavier.

• *Code crochet 27* (&1B). Il attend qu'un caractère soit tapé au clavier, puis met le code caractère de la touche appuyée dans le registre A. (Notez que rien n'est affiché à l'écran.)

• *Code crochet 32* (&20). Il examine le clavier à l'instant où il est appelé et indique — *via* l'état du drapeau C — si une touche est appuyée ou non. La routine n'attend pas qu'une touche soit appuyée. Si une touche l'est, alors le drapeau C est mis à 1 ; si rien n'est tapé, alors il est mis à 0.

Il est vital que les interruptions soient validées avant d'appeler les codes crochets 27 et 32, puisque l'inspection du clavier sur le Spectrum est commandée par interruptions.

• *Code crochet 28* (&1C). Il est responsable de l'impression du caractère (indiqué par le code ASCII contenu dans le registre A à l'entrée de cette routine) au flux 2, qui est normalement la partie supérieure de l'écran de télévision. L'utilisation de cette routine et du code crochet 27 est montrée dans la partie de code suivante. Si vous entrez ce programme, notez que c'est une boucle permanente, et il vous faudra donc éteindre la machine pour en sortir.

```
      ;call routine in Shadow Rom - use with care!
218888        ld   hl,ADDRESS          ;insert address of routine
22ED5C        ld   (23789),hl          ;put address in system variable
CF            rst  #08
32            defb #32                 ;do it
```

• *Code crochet 31* (&1F). Il est analogue au code crochet 28, mais écrit le caractère (avec code ASCII dans le registre A) vers le flux 3 — habituellement l'imprimante ZX.

• *Code crochet 50* (&32). C'est le dernier code crochet d'intérêt général que nous examinerons ici. Il est curieusement appelé *Sinclair Research Use Only* dans les spécifications. Cela nous permet d'appeler une routine ROM fantôme à une adresse particulière à partir de la ROM principale, plutôt qu'en utilisant les codes crochets. A cause des différences entre versions de la ROM fantôme, tous les programmes qui utilisent cet appel « se planteront » s'ils sont utilisés sur une autre version.

L'adresse de la routine ROM fantôme à laquelle nous voulons accéder est placée dans deux adresses dans les variables système produites par l'Interface 1. L'adresse de la variable concernée est 23789 et 23790. L'appel code crochet est alors fait. La partie de code suivante fera cela :

```
      ;wait for char, print to stream 2
      ;Hook Codes #1B and #1C
CF    start: rst  #08                ;set up Interface 1...
31           defb #31                ;...system variables
FB    loop:  ei                      ;ei - just in case
CF           rst  #08                ;wait for a char...
1B           defb #1B                ;...from the Keyboard
CF           rst  #08                ;char now in A reg...
1C           defb #1C                ;...so print it
18F9         jr   loop               ;round again ad infinitum
```

Ce n'est pas un appel vraiment utile, à moins que vous sachiez ce que font les diverses routines ROM fantôme et avec quelle version de la ROM fantôme vous travaillez. Nous examinerons cet appel en détail quand nous en viendrons à la manière d'utiliser l'Interface 1 pour ajouter de nouvelles instructions au BASIC.

Interaction stellaire

Le marché britannique étant avant tout tourné vers les possesseurs de cassettes, les programmeurs se sont vu distancer par les Américains, dont les logiciels sur disquettes sont plus complets.

Une imagination fertile
Douglas Adams conçut d'abord The Hitch Hiker's Guide to the Galaxy pour la radio, avant d'en tirer quatre livres qui ont connu un succès phénoménal dans les pays anglo-saxons.
(Cl. Jill Furmanovsky/ Time Out.)

Les jeux d'aventures nécessitent la mise en œuvre d'un nombre considérable de données; les descriptions de lieux, les messages d'écran et les sous-programmes de commande bénéficient donc, avec la disquette, d'un support idéal, puisque de cette façon ils peuvent être chargés en mémoire chaque fois qu'on en a besoin. Le problème est qu'en Grande-Bretagne les amateurs d'informatique se limitent généralement aux cassettes, plus lentes mais bien moins chères. Pour les programmeurs britanniques, c'est devenu un problème.

En effet, ces jeux d'aventures, étant vendus sur cassette, sont chargés en RAM une fois pour toutes, ce qui limite leur ampleur : en particulier, l'interaction entre personnages, grande consommatrice d'espace mémoire, et l'étendue du vocabulaire ont toujours dû être plus ou moins sacrifiés, en dépit de réussites isolées qui mettent en œuvre des techniques de programmation très ingénieuses.

La situation aux États-Unis est très différente : l'usager, même s'il est novice en la matière, fait presque toujours l'acquisition d'un lecteur de disquette, d'abord parce que tous les micro-ordinateurs en sont pourvus, et ensuite parce qu'il dispose d'un budget supérieur. Un tel marché était donc tout disposé à accueillir des jeux d'aventures plus compliqués, que certains n'hésitent pas à appeler « fiction interactive ». On doit à la firme américaine Infocom quelques-unes des plus grandes réussites en ce domaine.

The Hitch Hiker's Guide to the Galaxy, récemment paru, est digne de ses prédécesseurs. Ce programme est le résultat d'une collaboration entre les programmeurs d'Infocom et Douglas Adams, l'auteur de l'œuvre qui a servi de point de départ. L'ensemble est d'ores et déjà disponible pour de nombreux appareils (Apple, Atari, Commodore 64). Il vous suffira d'un lecteur de disquette et de beaucoup de patience!

L'intrigue, complexe et délirante à souhait, narre les aventures du malheureux Arthur Dent, véritable anti-héros arraché à sa petite vie paisible et lancé dans l'espace intergalactique à la suite d'une rencontre inopinée avec le vaisseau stellaire *Vogon*. Il s'agissait au départ d'un feuilleton radio.

Le principal point fort du jeu est sans doute son programme d'analyse syntaxique, chargé d'analyser toutes les injonctions tapées par l'usager au clavier. Il peut par exemple faire la différence entre adjectifs, adverbes et prépositions, sans se limiter, comme c'est souvent le cas, aux noms et aux verbes. Le vocabulaire est particulièrement riche — près de deux mille mots —, et les commandes peuvent être très diversifiées, puisqu'il est possible de poser directement des questions aux autres personnages, ou d'accomplir plusieurs actions en même temps.

Hitch Hiker's Guide to the Galaxy : pour les ordinateurs Apple, Apricot, Atari et IBM.
Une version destinée au Commodore 64 est disponible depuis peu.

Genre : aventure textuelle (en anglais).

Manche à balai : inutile.

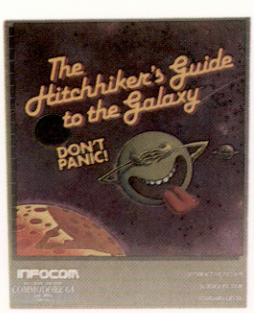
Format : disquette.

Bien entendu, l'ordinateur ne « comprend » pas toujours très bien ce qu'on lui dit. Il est pourtant en mesure, neuf fois sur dix, de donner une réponse acceptable, très supérieure, en tout cas, aux mornes Je ne comprends pas ou Vous ne pouvez pas faire cela.

Ce logiciel recourt par ailleurs à des techniques de programmation classiques, comme l'emploi de « conteneurs » (objets qui en contiennent d'autres) ou d'« objets globaux » (murs, planchers) présents en de très nombreux lieux, tandis que les « véhicules » constituent un type particulier de conteneurs : ce sont eux qui permettent au héros de passer d'une planète à l'autre. Il faut remarquer cependant que cette richesse d'options n'est pas vraiment due à une programmation très élaborée, mais tout simplement aux vastes possibilités de stockage offertes par une disquette.

Lorsqu'on désassemble un jeu d'aventures d'Infocom, on se rend compte en effet que l'intrigue et l'ambiance ont fait l'objet de soins tout particuliers, mais que la structure du programme elle-même reste très classique. Les programmeurs britanniques font remarquer de façon venimeuse que les Américains préfèrent accroître la taille du programme plutôt que de recourir à des techniques un tant soit peu sophistiquées. Peut-être prendront-ils leur revanche une fois le marché britannique équipé de lecteurs de disquette...

The Hitch Hiker's Guide to the Galaxy est en tout cas un programme passionnant, où l'on fera la connaissance de Ford Prefect, de Zaphod Beeblebrox et de nombreux extraterrestes. Les amateurs d'humour absurde seront à la fête. Prenez garde, toutefois : les bars de la galaxie semblent très mal fréquentés...



Les maîtres de l'aventure
Hitch Hiker's Guide to the Galaxy est le dernier titre de jeux d'aventures édités par Infocom, dont le premier succès fut la trilogie de l'Empire Zork.