ISSN 0760-6354

Micro et Robots

16 F
N° 7
Mai 84

LE MAGAZINE DE LA MACHINE INTELLIGENTE

Le nouveau Bob-XA

EXCLUSIF: Le 1er salon mondial de la robotique personnelle aux U.S.A

BANCS D'ESSAIS
Avant-première: EXL 100
Epson HX 20
Tandy PC4

COMPARATIF
3 tables à digitaliser

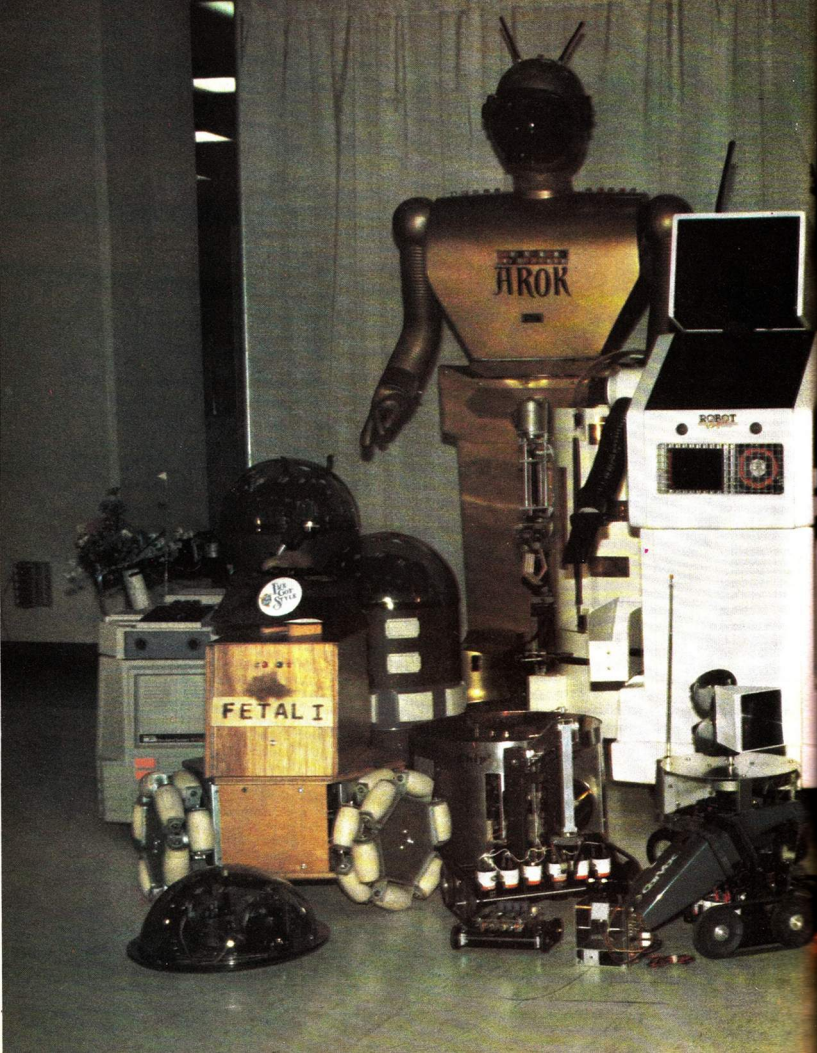
RECHERCHE
Toulouse: la robotique rose!

TECHNOLOGIE
La voix synthétique

REALISATIONS
Le Cybernoïd
Un synthétiseur vocal

Belgique : 130 F.B.
Suisse : 5,60 F.S.
Canada : 2,25 $.

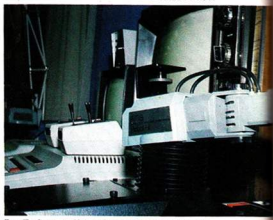
T2351-07-16,00 F


JAROK
ROBOT
FETAL I

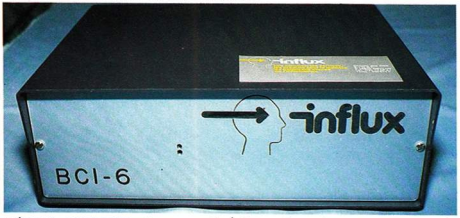
Définir un robot c'est plus difficile que de le reconnaître!

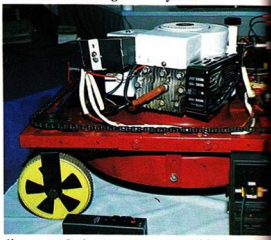
L'un des plus évolués, le Bob/XA d'Androbot.

Nolan Bushnell fondateur d'Androbot.



Le Robot-1 d'Analog Micro Systems.

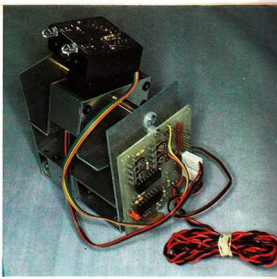




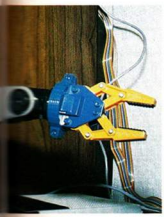
raissent courageux et, pour le moins, sérieusement motivés (l'argent, s'il reste un élément moteur indéniable — nous sommes aux Etats-Unis — n'explique pas tout) car leurs produits sont chers (plusieurs millers de dollars), difficiles à mettre en œuvre et souvent capricieux, d'une mobilité douteuse et d'une utilisé quasi nulle si l'on excepte le potentiel pédagogique qu'ils véhiculent... Quant au plan affectif, leur surdité actuelle même aux plus creux des discours humains, engendre pas mal de frustrations dans les rapports censés s'établir entre l'homme à sa création. Les robots actuels ne s'inscrivent-ils, alors, que dans une perspective de récréation ? Nous ne sommes pas très loin de le penser, dans l'attente de ce que les prospectives les plus autorisées laissent à imaginer pour l'avenir proche. A cet égard, la comparaison avec l'histoire du micro-ordinateur reste tentante pour tout le monde : en serait-on aujourd'hui, avec les robots personnels, là où en en était il n'y a pas même 10 ans avec les micro-ordinateurs ? A supposer que l'histoire puisse se répéter ainsi, il ne faudrait cependant pas sous-estimer les difficultés inhérentes à toute machine destinée à se confronter mécaniquement avec la matière sans parler, même, des problèmes traités par l'Intelligence Artificielle à cette occasion et lors des rapports réciproques Homme-Machine. Car il semble exclu d'ouvrir un marché grand public en fournissant avec le robot un manuel d'utilisation de 300 pages !

A Albuquerque on en était donc là

Le RB 5X de RB Robot en action.

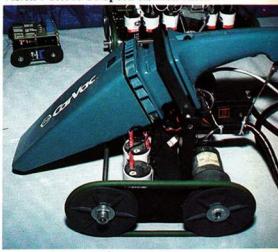


Vision I Stereo de Spectron Instrument.

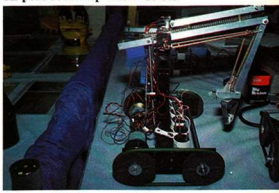


Un robot tondeuse à gazon !

Sur le principe du Fetal 1, un fauteuil «3 roues» motrices fabriqué par International Texas Industries.



Un petit robot aspirateur de S.I.

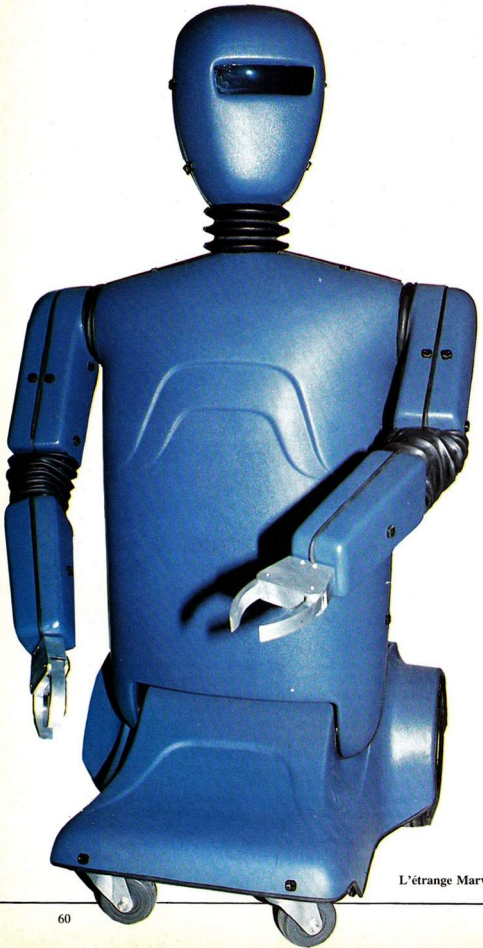


Robot mobile 4 axes de S.I.

de ces réflexions et à faire le point des ressources matérielles et logicielles. Du côté matériel, tout existe pour construire le robot domestique jusqu'aux systèmes de vision artificielle de moins en moins coûteux et de plus en plus performants : le seul problème à résoudre reste celui des grandes séries qui feront chuter les prix. Du côté logiciel, les choses sont beaucoup plus complexes et de l'ordre, de toute manière, des concepts les plus abstraits, du choix des modèles. Et il ne fait guère de doute que l'expansion rapide d'une industrie du robot domestique (au sens le plus large du terme) profitera d'abord à tous ceux qui en auront été les initiateurs et ensuite à la recherche même en Intelligence Artificielle dont les produits «immatériels» constituent la clef et le fond du problème. Pour que les robots, tout compte fait, ne soient pas nos nouveaux assistés !

Analog Micro Systems

Les systèmes de vision artificielle se démocratisent et si l'on n'en voulait qu'une preuve, elle nous serait apportée par Analog Micro Systems avec son système MicronEye vendu à moins de 300$. Un système qui se compose d'une caméra (équipée d'une optique 16 mm/F1,6), d'un trépied, d'une carte d'interface, du soft et du manuel. Le cœur du système n'est autre qu'une Ram photosensibilisée de 64K. La résolution atteint 128 × 256 pixels pour une cadence de 2-5 images par seconde en éclairage normal. A la cadence maximale de 15 images par seconde la résolution



L'étrange Marvin IV.

est encore de 64 × 128. L'information délivrée par Micron est, bien entendu, de type numérique (noir = 0, blanc = 1) mais un programme existe permettant de la traiter en échelle de gris grâce à un balayage multiple à différents temps d'exposition. La caméra a été prévue pour transmettre à une vitesse choisie entre 300 et 153600 bauds. Cette société commercialise en outre un bras 6 axes (le Robot-1 à 289$) et différents logiciels et interfaces.

Androbot

De tous les robots personnels présentés, le B.O.B./XA fabriqué par Androbot nous apparut le plus intéressant et le plus accompli tant dans ses possibilités de navigation que dans ses possibilités de programmation. L'informatique de bord est constituée autour d'un 8088 entouré de 64K de Ram (extensible à 256K) et de 128 K de Rom (8 cartouches). Comme Topo, que nous ons présenté dans notre nº 5, il parle : soit par l'intermédiaire de phrases composées au clavier, soit en puisant dans un vocabulaire pré-enregistré. On a pu voir, sur l'aire de démonstration, le B.O.B./XA se déjouer des obstacles (et du démonstrateur en particulier), reconnaître à haute voix certains objets à distance respectable (par analyse d'un code optique), saisir et déplacer des charges relativement importants (8 kg) grâce, uniquement, à des senseurs intégrés dans un bras un peu particulier (voir photo). Ajoutons que ce robot peut recevoir de nombreuses extensions (la reconnaissance vocale lui donnerait un atout essentiel) et qu'il peut être piloté par un IBM PC ou un Apple IIe. A n'en pas douter, on tient avec ce B.O.B./XA un objet qui ne manque pas d'attrait et rompt avec la dangereuse voie que d'autres semblent suivre, celle du robot-gadget. Complet, le système atteint cependant 4000 $...

Influx

Seule société française ayant eu le courage de se déplacer à Albuquerque, Influx y présentait son unité de

LA PROGRAMM

LE BASIC (V)

Nous arrivons aujourd'hui au dernier volet de présentation des ordres Basic standard avec la description des commandes et instructions diverses.

Les assignations de valeurs

Ces instructions sont au nombre de quatre sauf sur certains Basic élémentaires, tels celui du ZX 81 par exemple, où les trois dernières instructions que nous allons voir font (cruellement) défaut. Ces instructions sont LET, DATA, READ et RESTORE.

LET est cité dans ce paragraphe pour mémoire et pour être logique; en effet, nous avons déjà eu à l'employer dans le courant de cet exposé et, de plus, son utilisation est facultative avec bien des interpréteurs. LET sert à fixer la valeur d'une variable sous la forme : LET A = N ce qui signifie donner à A la valeur N. De très nombreux interpréteurs admettent que l'on écrive tout simplement A = N mais attention, dans ce cas le signe égal n'a pas son sens mathématique; il signifie bel et bien donner à A la valeur N et non A = N qui, mathématiquement parlant, voudrait dire A égal N, c'est-à-dire A et N identiques.

Cela étant précisé, passons à des instructions plus intéressantes avec READ, DATA et RESTORE qui forment un tout indispensable. DATA permet de définir une suite de valeurs qui peuvent être numériques ou chaînes de caractères; ces deux types de valeurs pouvant être mélangés dans le même DATA. Ainsi, par exemple, l'on peut écrire :

DATA 10, 34, «BONJOUR»,3, «MICRO»,256.

L'instruction READ fonctionne conjointement à, au moins, un DATA. READ doit être suivi par plusieurs noms de variables (ou par un nom de variable qui évolue au fur et à mesure du déroulement du programme) et lors de l'exécution du READ, la première variable qui suit se voit affecter la première valeur trouvée dans le premier DATA du programme; la deuxième variable se voit affecter la valeur qui suit la précédente dans le DATA précité et ainsi de suite. Ce fonctionnement nécessite quelques précisions supplémentaires. Les DATA peuvent être écrits en plusieurs endroits d'un programme, lors de l'exécution, toutes les valeurs qui y sont contenues, sont mises bout à bout dans l'ordre numérique des lignes d'instructions.

Ainsi : 100 DATA 1,243 avec plus loin 230 DATA «BONJOUR»,26 est équivalent à un seul DATA contenant dans cet ordre 1,243,«BONJOUR»,26.

Les instructions READ lisent les valeurs contenues dans le DATA global ainsi constitué. De ce fait, il peut y avoir plus de DATA que de valeurs suivant un READ, les DATA surnuméraires seront tout simplement inutilisées; par contre il faut impérativement que le nombre de données suivant le total des DATA soit au moins égal au nombre de variables suivant un READ sinon, il y aura génération d'une erreur car l'interpréteur sera incapa-

ATION

ble d'affecter des valeurs à certaines variables.
Comme nous l'avons dit, les valeurs qui suivent les DATA peuvent être numériques ou chaînes de caractères ; il faut cependant respecter un point important : les variables qui vont suivre le READ devant lire ces DATA doivent être du même type que ce que vous avez mis derrière le DATA. Voici un petit exemple.

```
100 DATA 2, «BONJOUR»,345
110 READ A,A$,B
```

est parfaitement correct et affectera à A la valeur 2, à la chaîne A$ la valeur «BONJOUR» et à B la valeur 345. Une ligne telle que :

```
110 READ A,B,C,
```

produirait une erreur car B (variable numérique) se verrait affecter la chaîne de caractère «BONJOUR» ce qui est impossible.
Cela donne généralement le message «data type mismatch» ce qui, en bon Français, veut dire type de donnée incorrect (plus exactement non adapté).
Pour revenir à ce que nous disions tout à l'heure, un READ n'est pas forcément suivi par une longue liste visible de noms de variables ; il est ainsi possible de faire :

```
100 DATA 12,32,45,456,etc.
200 FOR I = 1 TO N
210 READ A
... exploitation de la valeur de A lue...
300 NEXT I
```

qui affectera, au fur et à mesure de l'évolution de la variable de boucle I, les valeurs 12 puis 32 puis 45, etc. à A. Il faut seulement veiller à ce que N soit inférieur ou égal au nombre de données qui suivent le DATA à moins que vous ne souhaitiez sortir de la boucle par un ON ERROR GOTO comme nous le verrons plus avant dans ces articles.
L'instruction RESTORE est un complément à l'instruction READ. En effet, alors que READ ne peut lire les données qui suivent un DATA que dans un ordre séquentiel, RESTORE permet de revenir au début d'une liste de données de deux façons. RESTORE utilisé seul a pour effet de remettre à 0 le pointeur de lecture des DATA que comporte l'interpréteur, ce qui signifie que le READ qui suivra un RESTORE ira lire la première donnée du premier DATA du programme, exactement comme si aucun READ n'avait eu lieu au préalable.
Si RESTORE est suivi par un numéro de ligne, son action ne portera que sur les DATA dont les numéros de ligne sont supérieurs ou égaux à ce dernier et les READ suivant commenceront leur lecture à la première donnée du premier DATA de numéro de ligne immédiatement supérieur ou égal au numéro de ligne qui suit le RESTORE (ouf !).
Voici quelques exemples simples précisant tout cela :

```
100 DATA 10,20,30,40
110 READ A,B,C,
120 READ D
130 PRINT D
```

fera imprimer 40 puisque D se sera vu affecter la quatrième donnée après le DATA.
Si maintenant nous écrivons :

```
100 DATA 10,20,30,40
110 READ A,B,C
120 RESTORE
130 READ D
140 PRINT D
```

nous verrons afficher 10 car le RESTORE aura ramené le pointeur de données à 0 et le READ qui suit commencera alors sa lecture par la première donnée du premier DATA.
Si maintenant nous écrivons :

```
100 DATA 10,20,30,40
110 READ A,B,C
120 DATA 50,60,70,80
130 READ D,E,F
140 RESTORE 120
150 READ G
160 PRINT G
```

nous verrons imprimer 50 car le RESTORE n'aura agit que sur les DATA de numéros de lignes supérieurs ou égaux à 120.
Attention, si vous voulez essayer ces quelques exemples, l'instruction RESTORE simple fonctionne sur tous les interpréteurs capables de faire des READ et des DATA mais le RESTORE NNN ne fonctionne pas sur tous les interpréteurs.

Les dimensionnements de variables

Nous avons dit, dans un des premiers articles de cette série, qu'il était possible de travailler sur des variables indicées à une ou deux dimensions ; variables indicées que l'on peut aussi appeler des tableaux, les indices servant à évoluer à l'intérieur de ceux-ci.
De telles variables doivent être définies avant toute utilisation afin que le Basic sache quelle place leur réserver en mémoire ; en effet, alors que pour une variable numérique normale le problème ne se pose pas (le codage de celle-ci se faisant sur un nombre fixe d'octets), pour une variable indicée, la place à réserver dépend de la valeur maximum de l'indice.
Si l'on veut constituer un tableau à deux dimensions avec des indices pouvant aller jusqu'à 10 et 10, il faudra réserver en mémoire 100 fois la taille d'une variable élémentaire ! Pour ce faire existe l'instruction DIM qui définit les valeurs maximum des indices de toute variable indicée de la façon suivante : DIM A(10,10) signifie que la variable indicée A(I,J) pourra voir ses indices I et J aller jusqu'à 10. Pour une variable à une seule dimension, on écrirait DIM A(50) pour un indice I de A(I) allant jusqu'à 50 par exemple.
Cette instruction de dimensionnement doit impérativement être présente dans un programme pour toute variable indicée ; elle doit, de plus, apparaître avant toute utilisation de la dite variable. Par ailleurs, il est interdit de dimensionner plusieurs fois de suite (même avec des dimensions identiques), une même variable. On peut citer à titre d'exemple une cause d'erreur très

fréquente lorsqu'on débute et lorsqu'on veut réaliser un programme bouclé sur lui-même en décrivant les lignes suivantes :

10 DIM A(10,10)

... programme proprement dit se terminant par un...

100 GOTO 10 pour faire recommencer celui-ci au début. Cela provoque inévitablement une erreur de redimensionnement. La solution consiste à boucler le programme sur la ligne qui suit le dimensionnement. Si plusieurs dimensionnements apparaissent dans le même programme en des endroits différents, une bonne pratique consiste à les grouper tous ensemble sur une ou plusieurs lignes situées tout au début du programme, ce sera plus rationnel et plus lisible.

Précisons que, pour certains Basic, cette instruction DIM permet aussi de définir la longueur maximum des chaînes de caractères.

Les instructions diverses

Les plus célèbres sont certainement PEEK et POKE qui permettent d'accéder directement à la mémoire de la machine ou à ses circuits d'entrées/sorties.

La fonction POKE ADRESSE, DONNEE, place la donnée spécifiée à l'adresse mémoire indiquée. Cette adresse mémoire peut être celle du registre d'un circuit de sortie ce qui permet ainsi d'envoyer la donnée vers une interface quelconque. La donnée est codée sur 8 bits sauf sur certaines machines disposant du DOKE qui admet une donnée sur 16 bits. L'adresse et la donnée sont exprimées en décimal sauf indications contraires admises par la machine (hexadécimal en faisant précéder adresse et/ou donnée par le symbole prévu pour ce faire).

La fonction PEEK est «l'inverse» de POKE et permet de lire un octet à une adresse spécifiée. Si A = POKE (ADRESSE), A se verra affecter la valeur du mot de 8 bits se trouvant à l'adresse spécifiée. Comme pour PEEK, on commence à voir apparaître DEEK qui peut lire un mot de 16 bits à une adresse définie. Toujours comme pour POKE, un PEEK peut être effectué dans le registre d'un circuit d'entrées/sorties, par exemple pour lire une donnée en provenance d'une carte d'interface (voir par exemple notre article «la crise du port» dans le numéro 5 de *Micro et Robots*.

D'autres instructions diverses existent en Basic; nous allons les citer dans un ordre qui se veut arbitraire :

— REM permet de placer des commentaires dans un listing; tout ce qui suit un REM est considéré comme du commentaire et est ignoré par l'interpréteur. Attention à la fâcheuse manie qui consiste à faire des REM du style roman feuilleton qui gaspillent une place mémoire considérable (1 caractère dans un REM = 1 octet mémoire).

— DEF FNX (Y) permet de définir une fonction et d'appeler ensuite celle-ci par le nom que vous lui aurez donné lors de cette définition. X est le «nom» de la fonction et Y est la variable de définition proprement dite de la fonction. Voici un exemple d'utilisation :

10 DEF FNA(X) = 2* X + 1
20 LET X = 4
30 LET Y = FNA(X) affectera à Y la valeur 9 (2 ×4 + 1).

La syntaxe d'emploi de cette instruction peut varier légèrement selon les interpréteurs et un coup d'œil au manuel de celui que vous employez est conseillé.

— POS fournit une valeur numérique qui est la position de la tête d'impression du terminal (ou du curseur du terminal vidéo) sur la ligne courante.

— TAB(N) permet de déplacer le curseur ou la tête d'impression de N positions. Cette instruction ne doit être utilisée que lors d'un PRINT de la façon suivante : PRINT TAB(10), «BONJOUR» qui fera imprimer BONJOUR à partir de la dixième position de la ligne courante.

Les cas particuliers

Dans un langage parfaitement normalisé, ce paragraphe ne devrait pas exister; ce n'est malheureusement pas le cas pour le Basic (revoir si nécessaire nos premiers articles) où de nombreux mots clés ont été ajoutés pour répondre aux possibilités sans cesse croissantes des micro-ordinateurs domestiques. Ces instructions supplémentaires concernent essentiellement la gestion des graphiques et des sons et l'on trouve ainsi des DRAW pour tracer des traits, des MOVE pour se déplacer sans tracer, des CIRCLE pour tracer des cercles, des COLOR pour définir des couleurs de tracé, des SOUND pour définir des sons, etc.

Aucune normalisation n'existant à ce niveau, nous n'allons pas parler de ces instructions ici, d'autant que leur emploi est relativement simple et peut être assimilé en lisant simplement le manuel de la machine concernée (et en faisant quelques essais vu la clarté de certaines notices). Notre but est en effet de vous montrer avec quelques exemples comment résoudre un certain nombre de problèmes avec les instructions standard, ce qui ne vous empêche pas, par la suite, d'adapter nos exemples aux possibilités de votre machine pour leur donner un aspect plus attractif.

Nous n'allons pas parler non plus des instructions USER ou CALL qui permettent d'appeler un programme en langage machine; en effet, nous n'en aurons pas besoin dans ce qui suit et, de plus, si la méthode d'appel est à peu près normalisée, les sous-programmes que l'on peut appeler ou écrire dépendent à 99% de la machine sur laquelle vous travaillez, il est donc impossible de donner des généralités sur ce sujet.

Conclusion

Nous en resterons là pour aujourd'hui, notre prochain article étant consacré à des exemples concrets de programmes utilisant toutes les notions vues jusqu'à présent. Ces exemples seront, bien sûr, commentés point par point vous permettant de comprendre le pourquoi des solutions adoptées.

C. Tavernier

A lire

LE COMPACT DISC

Après une gestation longue de plus de dix ans, et consacrée tant aux recherches techniques qu'à la définition, dont on ne peut que se féliciter, d'un standard mondial unique, le disque audionumérique est entré, maintenant, dans sa phase opérationnelle. Ses caractéristiques (qualité de la restitution sonore, durée de vie presque infinie), le destinent à supplanter un jour l'actuel microsillon, avec lequel il cohabitera d'abord.

En Europe comme aux Etats-Unis, le disque audio numérique n'existe que sous la forme du «Compact Disc», à lecture sans contact par faisceau Laser. Signalons pourtant que JVC a étudié, pour le marché japonais, un lecteur numérique capacitif (AHD, dérivé du VHD pour la vidéo) exigeant un contact entre capteur et disque. C'est à la première version, la seule intéressant pratiquement les usagers français, que J.C. Hanus et Ch. Pannel consacrent leur livre.

Celui-ci, après un bref historique du Compact-Disc, analyse les limites, notamment en matière de dynamique et de bande passante, de l'enregistrement analogique sur disque.

Un chapitre expose, ensuite, le principe de l'échantillonnage, du codage, en montrant la nécessité et les caractéristiques du filtrage associé. Le chapitre suivant, consacré à la structure du disque et aux procédés de lecture, passionnera les lecteurs de Micro et Robots : la technique fait appel, en effet, à des asservissements gérant à la fois la poursuite de la piste formée par la spirale de micro-cuvettes porteuse de l'information codée en binaire, et la focalisation du faisceau Laser sur la surface de lecture, avec des précisions de l'ordre du micromètre.

Les procédés de conversion numérique/analogique (un chapitre) précédent l'étude des problèmes d'insertion du Compact Disc dans la chaîne Hi-Fi. Là, les auteurs dissèquent les améliorations qu'on peut en attendre (augmentation de la dynamique, en particulier), compte tenu des performances des autres maillons de la chaîne.

Les diverses fonctions dévolues au lecteur, conduisent naturellement l'examen des perspectives d'avenir : celles-ci englobent une extension du traitement numérique du signal jusque dans le pré-amplificateur (gain en bruit et en distorsion), un élargissement du procédé stéréophonique (une nouvelle tétraphonie par exemple ?), et l'enregistrement d'informations supplémentaires, comme des textes ou des images, qu'il serait possible de visualiser sur un écran de télévision.

La rédaction d'un livre comme celui de J.C. Hanus et de Ch. Pannel constitue une œuvre de vulgarisation particulièrement délicate, compte-tenu de la disparité des centres d'intérêt et de la formation scientifique du lectorat visé. Les auteurs ont su magistralement franchir l'écueil. Ils y parviennent, notamment, en scindant leur exposé en deux parties. Dans les deux premiers tiers du livre, on trouvera, souvent éclairées par des analogies simples et très parlantes, des notions accessibles à tous. Le dernier tiers rassemble des annexes où une formulation mathématique satisfaira ceux qui veulent en savoir plus. Une bibliographie détaillée complète l'ensemble.

Tous les lecteurs de Micro et Robots connaissent J.C. Hanus, rédacteur en chef de la revue. Ch. Pannel, chargé de fonctions analogues dans d'autres publications, et ancien universitaire, s'intéresse depuis longtemps aux problèmes de la reproduction sonore. C'est dire tout ce que peut apporter leur livre, fruit de nombreuses années de réflexion.

R. Rateau

Service lecteur : cerclez 37.

TRAITEMENT DE TEXTE

Entre la photocomposition et la production de textes par machines à écrire conventionnelles, il existe un moyen terme que livre le traitement de texte. Un grand nombre de micro-ordinateurs actuels permettent ces traitements de texte qui offrent de nombreux avantages dont les corrections, l'insertion de nouveaux textes, etc. Hal Glatzer dans son livre «L'introduction au traitement de texte» fait le tour de cette question qui revêt une importance croissante pour toutes les entreprises, artisans et particuliers. Des choix clairs sont proposés et de nombreux exemples sont décrits.

Service lecteur : cerclez 38.