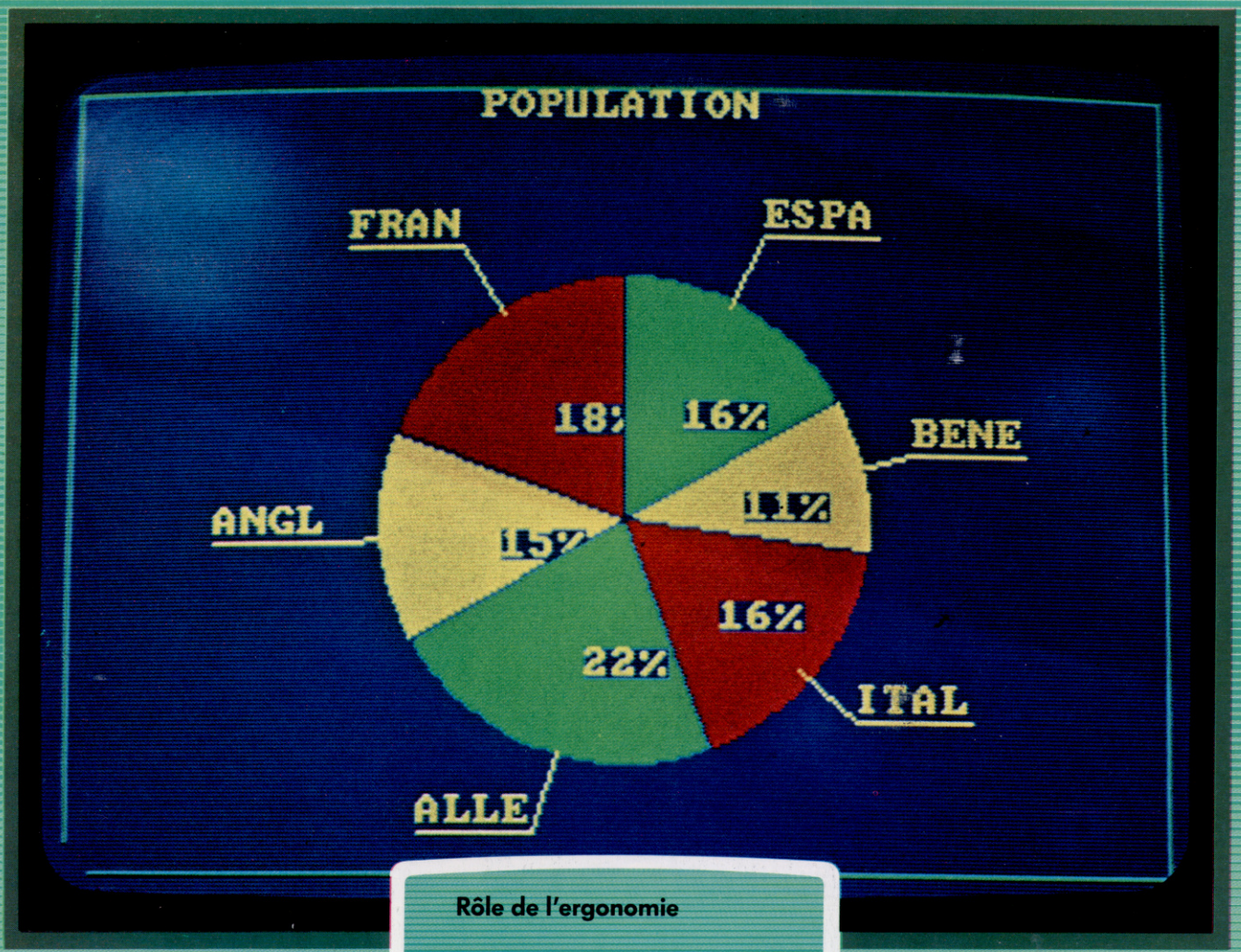
ABC INFORMATIQUE

N° 51

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Rôle de l'ergonomie

Routines générales d'aide

Le Sega SC3000H

Mixage du son

EDITIONS ATLAS

M6062-51-12-F 85FB-3,80FS-$1.95

Prêt à porter

L'ergonomie entend, entre autres, déterminer les principes fondamentaux qui régissent l'interaction homme-machine, afin d'accroître la productivité.

On sait que les sons ou les couleurs peuvent affecter profondément les performances de chacun. C'est pourquoi certains standards téléphoniques diffusent de la musique à l'intention des usagers qui attendent. De la même façon, les moniteurs monochromes sont le plus souvent verts, parce que c'est une couleur reposante; un affichage noir et blanc se révèle très vite fatigant pour la vue.

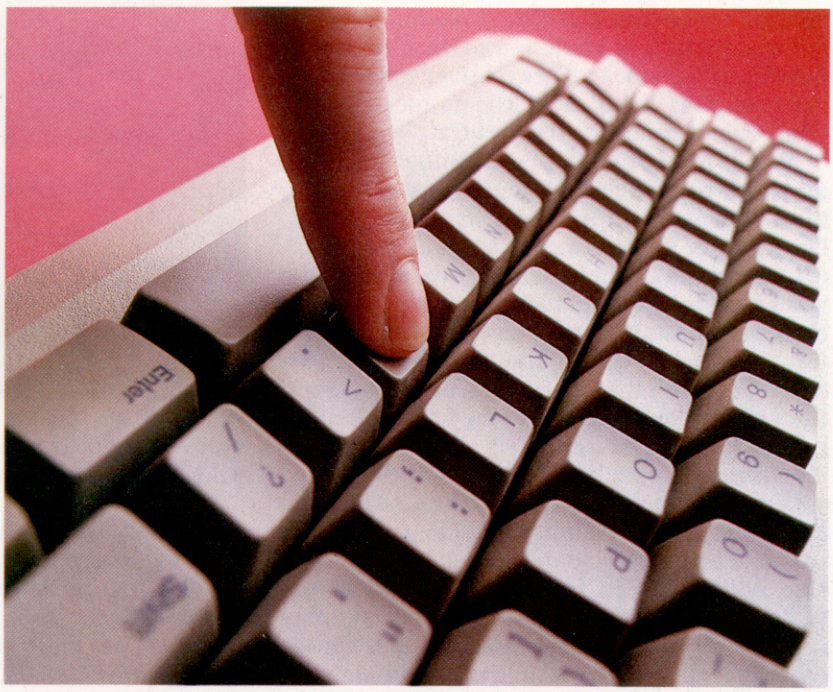
Des combinaisons de couleurs différentes provoquent des réactions très variées : des lettres bleues sur fond jaune paraissent agréables à l'œil, alors qu'un mélange cyan et vert laisse une impression déplaisante.

Le passionné d'informatique peut se demander en quoi de telles considérations le concernent. Mais les ergonomistes ont montré qu'un environnement négatif provoque vite une fatigue oculaire ou des douleurs dorsales, causées par de simples effets de couleurs ou de sons. Une étude restée célèbre permit ainsi d'analyser les médiocres performances d'un groupe d'opérateurs travaillant sur des terminaux installés aux quatre coins d'une pièce; une telle disposition faisait naître un sentiment de stress et d'isolement. Les employés furent donc répartis en postes de travail situés au centre de la même pièce, de façon que chaque poste de travail soit en face des autres. L'ambiance était infiniment plus détendue, et la productivité de chacun augmenta aussitôt.

Il ne faut pas oublier que l'informatisation d'une tâche a parfois pour conséquence de la rendre monotone, routinière et, surtout, peu compliquée : sources importantes d'erreur. L'une des fonctions d'un analyste de systèmes consiste justement à tenir compte très précisément de ce facteur humain. C'est là que l'ergonomie se révèle irremplaçable.

De son point de vue, l'opérateur humain fait partie intégrante de l'ensemble; il a des caractéristiques et des marges de tolérance qui lui sont propres. Les sentiments, les habitudes et les perceptions de chacun contribuent à l'efficacité d'un système, au même titre que le rythme de son horloge interne ou que les capacités de son bus d'adressage. L'humaniste et le financier en conviendront sans peine.

Le programmeur peut, lui aussi, bénéficier d'une étude approfondie des exigences particulières de son travail. La programmation exige une attention constante, un épuisant souci du détail, une logique rigoureuse; mais peu de gens en sont capables naturellement, et beaucoup y sont réfractaires. C'est pourquoi on ne peut jamais garantir qu'un programme quelconque ne comportera aucune erreur.



Au doigt et à l'œil
L'utilisateur rencontre ses premiers problèmes dès qu'il entreprend de se servir du clavier. Celui-ci a été l'objet de bien des améliorations ergonomiques : touches sculptées, clavier plat, affichage par cristaux liquides, pavé numérique. Mais il semble impossible d'en surmonter la faiblesse principale : l'incohérence d'une répartition des lettres de type AZERTY. Ce dispositif, adopté vers la fin du XIXe siècle, était même destiné à ralentir la frappe, dont les machines à écrire d'alors ne pouvaient suivre le rythme ! Mais il est désormais entré dans les mœurs. On voit que l'ergonomie, si elle veut vraiment prendre en compte les facteurs humains, doit aussi apprendre à en accepter les côtés traditionnels. (Cl. Ian McKinnell.)

Pourtant, l'application de l'ergonomie à la création de nouveaux langages de programmation ou de méthodes de création de systèmes est une entreprise fascinante, qui ne peut qu'être bénéfique aux programmeurs et à leurs utilisateurs. Les psychologues ont consacré des années de travail à l'étude des personnes affectées au traitement des données; ils ont déjà des idées précises, bien qu'incomplètes, sur les capacités du cerveau, sur sa vitesse de réaction ou sur les structures de la mémoire. Cela pourrait permettre la mise au point de programmes ou d'ensembles de données d'usage aisé, que tous pourraient utiliser sans effort ou avec un minimum de formation.

Depuis longtemps déjà, l'industrie a eu recours à des méthodes de ce type pour des tâches d'organisation, de sélection ou de perfectionnement. Mais les psychologues insistent sur le fait qu'un système doit être adapté aux capacités de ceux qui l'utilisent, et non l'inverse.

Ces derniers temps, les ergonomistes ont ainsi étudié de près les réactions des usagers à des logiciels complexes : c'est ce qu'on appelle volontiers « interface utilisateur ». En effet, depuis les débuts, tout proches encore, de l'informatique, les seules personnes concernées ont été des pro-

fessionnels, très compétents et motivés, capables de surmonter tous les obstacles, afin d'être suffisamment performants : la maîtrise technique signifiait aussitôt un véritable pouvoir. Mais les choses ont changé, l'utilisateur d'aujourd'hui n'a rien d'un spécialiste, et il supporte assez mal les exigences des machines : s'il lui faut apprendre à manipuler une base de données afin de faire usage d'un distributeur de billets, il y a peu de chances qu'il s'acharne longtemps ! D'ailleurs, le problème ne se situe pas seulement au niveau des usagers occasionnels. Il existe des bases de données qui permettent d'accéder à des informations indispensables à la gestion, à partir d'un simple bureau, et qui stockent ces renseignements dans des mémoires occupant plusieurs mégaoctets à l'intérieur d'un ordinateur central.

Mais elles ne produisent pas toujours les effets attendus, parce que leur emploi suppose la pratique de langages très complexes comme le SQL. On s'efforce donc de mettre au point des procédés permettant de communiquer avec elles par des voies plus directes, qui incluraient le français ou l'anglais. Il faudrait pour cela que les interventions de l'usager soient traduites automatiquement dans le langage du système lui-même : l'opérateur devrait tout au plus préciser ses exigences, tandis que l'appareil lui fournirait des réponses plus détaillées.

Bien des programmes font déjà usage de fonctions de type HELP, qui indiquent à l'usager la conduite à suivre en cas de difficultés. Les recherches poursuivies en intelligence artificielle ont également permis la création de programmes capables de justifier les décisions qu'ils prennent ; on pense qu'il sera possible d'arriver de cette façon à la mise au point de « modules-conseil » capables de venir en aide aux utilisateurs. Ce type de logiciels « intelligents » fait naître des questions très philosophiques : qu'est-ce qu'une bonne explication ? A qui est-elle destinée ? Le programmeur s'intéressera à la structure de sa base de données, à la façon dont celles-là sont mises en œuvre ; l'homme d'affaires sera curieux de savoir comment il peut analyser tel problème commercial. Ce sont des distinctions d'ordres sémantique et linguistique que les ergonomistes et les informaticiens devront prendre en compte dans un avenir très proche.

Une autre difficulté est liée à l'existence de « modèles » mentaux stéréotypés (ainsi les clichés raciaux ou nationaux), dont les gens font usage pour suppléer à leur ignorance. Nous avons tous tendance à croire, par exemple, qu'il suffit de connaître la profession d'un inconnu (chef d'orchestre, militaire, etc.), pour être en mesure de prédire, aussitôt, quelle sera sa personnalité — comportement, opinions, attitudes politiques... Dans le même ordre d'idées, les gens traitent les ordinateurs de façon très naïve, et les croient spontanément plus intelligents et plus doués qu'ils ne le sont. Après quoi, quand ils se heurtent aux réponses très brèves et très impersonnelles que leur adressent la plupart des programmes (surtout en cas d'erreur), ils ont l'impression, tout aussi erronée, d'être traités avec hostilité et brutalité. Ce type de blocage ne peut aller qu'en empirant.

Conditions de travail
On s'accorde généralement à penser que des facteurs physiques très précis (température, bruit, dimensions des locaux, etc.) sont en grande partie responsables de l'efficacité du travail fourni par les employés. Mais la couleur des murs, la lumière, la sensation d'espace (ou d'étouffement) constituent autant d'éléments plus subtils, mais aussi contraignants, surtout en informatique. Un analyste de systèmes ne pourra donc se contenter de choisir le matériel et les logiciels ; il devra aussi se préoccuper des conditions de travail réelles.

Les programmeurs s'efforcent, bien entendu, de faciliter au maximum la tâche de l'utilisateur, mais cela n'aboutit parfois qu'à un échec paradoxal, parce que le programme semble être bien plus intelligent qu'il ne l'est ! Il peut être utile d'émettre des messages du genre BONJOUR ROBERT, JE SUIS LE RESPONSABLE DE LA BASE DE DONNEES ; mais l'usager sera peut-être tenté de répondre de la même façon, ce qui mènera inévitablement à des résultats catastrophiques.

Rendre un logiciel d'usage aisé est une entreprise subtile, qui consiste à tenir compte des réactions humaines, de leur complexité et de leur caractère contradictoire face à l'ordinateur : il ne suffit pas de peindre les lecteurs de disquettes en rose !

Le cas du forth

Les gens peuvent être victimes des idées reçues, comme d'un environnement mal conçu. Le langage FORTH, dans lequel beaucoup voient un successeur du BASIC, fut créé par l'astronome Charles Moore, qui travaillait à l'observatoire de Kitts Peak, en Arizona. Moore cherchait à contrôler les déplacements d'un télescope à l'aide d'un ordinateur utilisant des programmes écrits en FORTRAN. Il se rendit compte que ce langage se révélait peu adapté à des applications de contrôle. Il eut donc l'idée d'écrire le FORTH, qui n'est pas rigidement structuré comme ses prédécesseurs et qui permet à l'utilisateur d'inventer son propre dialecte, et donc de satisfaire les besoins qui lui sont particuliers. Mais cette souplesse d'utilisation se paie d'une assez grande difficulté d'emploi : ce n'est pas parce qu'un système est efficace qu'il est commode à mettre en œuvre !

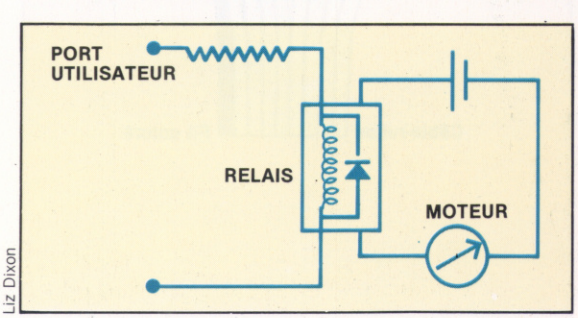
Protégez-nous!

Nous découvrons ici comment produire une sortie à partir du port utilisateur et nous présentons un réseau de commande complet, dont nous expliquerons plus loin la construction.

Dans la plupart des applications de la micro-informatique, le terme « tampon » désigne un endroit de stockage temporaire destiné aux données qui sont transférées d'une partie d'un système à une autre. Cependant, en électronique analogique, le terme sert à décrire un circuit qui protège un dispositif de l'activité d'un autre. Nous devons donc protéger les délicats circuits internes du micro, afin qu'ils ne soient pas endommagés par les composants auxquels nous les connectons.

La puce d'entrée/sortie de votre micro-ordinateur fonctionne à des niveaux de tension de 0 et de + 5 V, et utilise des courants mesurés en milliampères (mA). Nous devons donc veiller à ne pas introduire des tensions supérieures sur l'une des lignes du port utilisateur ni à imposer une intensité plus grande que 30 à 40 mA.

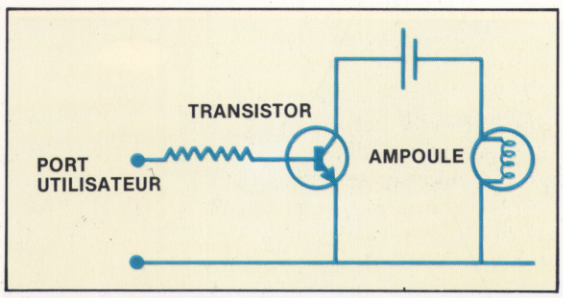
Nous avons déjà montré comment il est possible de changer le contenu du registre de données en mettant en contact des fils dénudés du port utilisateur. Nous avons découvert que ce type de mise à la terre des cellules du registre n'introduisait pas de tensions ni de courants dangereux pour le système; il n'était donc pas nécessaire de protéger les circuits internes du micro. Si, cependant, nous désirons connecter d'autres dispositifs, une protection doit être prévue, et celle-ci peut prendre plusieurs formes. Nous pouvons seulement nous assurer qu'une intensité supérieure à 20 mA ne sera jamais appliquée à l'une des broches du port utilisateur. C'est réalisable en utilisant un relais connecté au port utilisateur pour mettre sous et hors tension le dispositif de sortie et en introduisant une résistance dans le circuit d'alimentation du relais. Si le circuit fonctionne à + 5 V et si nous avons besoin d'une intensité ne dépassant pas 20 mA, nous utilisons alors la loi d'Ohm pour calculer la valeur de la résistance nécessaire à partir de la formule de base, où la tension est égale au produit de l'intensité par la résistance, et en faisant attention aux unités choisies :



Liz Dixon

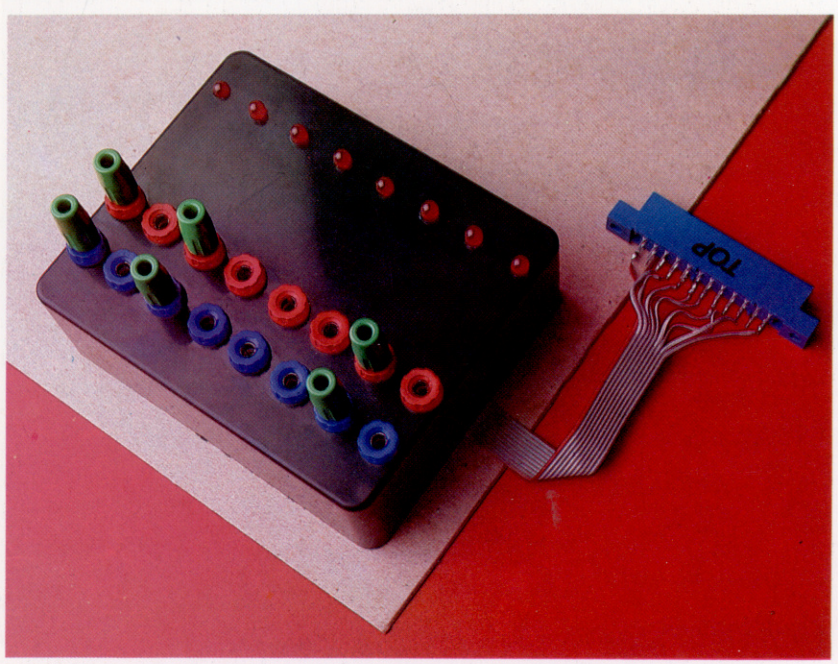
$$V = I \times R$$
$$R = V/I$$
$$R = 5/0{,}02$$
$$R = 250\ \Omega$$

La sortie d'une broche de port utilisateur peut servir à solliciter un interrupteur à transistor pour compléter un circuit externe.

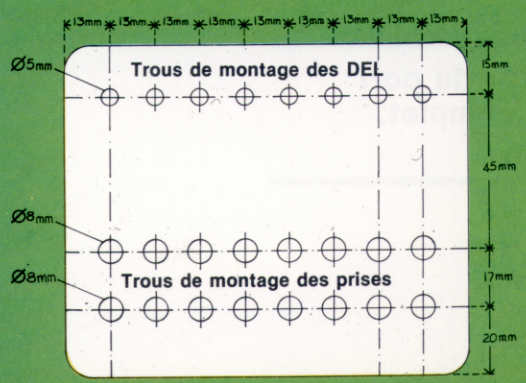


Le boîtier tampon que nous construirons utilise le principe d'interruption à transistor pour protéger le port utilisateur; dans un but pratique, puisque les circuits destinés aux huit lignes sont contenus sur une seule puce.

Dès que le tampon du port utilisateur a été mis en place, nous pouvons ajouter une série de modules à celui-ci et y connecter d'autres dispositifs. Ces modules nous permettront de commander l'interruption des DEL (diodes électroluminescentes), des moteurs à basse et haute tension



L'intermédiaire
Le boîtier tampon est connecté au port utilisateur du Commodore 64 par le connecteur décrit dans le dernier article. Le boîtier protège l'ordinateur contre des niveaux dangereux d'intensité d'entrée/sortie. Les DEL indiquent l'état des lignes de sortie du port utilisateur, et les fiches et prises servent d'interrupteurs sur les lignes d'entrées.



Base du boîtier

Perforation du boîtier

Les DEL et les prises rouges et noires sont montées sur la base du boîtier. Le plastique du boîtier est mou, et on le perfore facilement. Afin d'éviter de l'endommager, couvrez la surface d'un ruban cache, puis marquez l'emplacement des trous. Amorcez les perforations avec une mèche très fine, puis faites des trous d'un diamètre de 4 mm pour les DEL, et de 8 mm pour les prises.

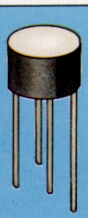
Le nom des pièces

Pour construire l'unité d'interface, vous avez besoin des pièces énumérées ci-après. Nous donnons les numéros de pièces Maplin, mais vous pouvez les obtenir (ou des substituts acceptables) chez tout fournisseur de pièces électroniques. Les dimensions exactes du boîtier ne sont pas limitées, mais notre réalisation correspond exactement au boîtier décrit.

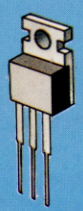
Quantité	Article	N° Maplin
8	Résistance de 4,7 kΩ, 0,4 W	M4K7
8	Résistance 240 Ω, 0,4 W	M23OR
1	Condensateur électrolytique 1 µF	FF01B
1	Condensateur 1 µF	BX76H
8	DEL rouge	WL27E
8	Diode 1N4148	QL80B
1	Redresseur en pont W005	QL375
3	Tampons hex 7407	QX76H
1	Régulateur de tension µA7805UC	QL31J
1	Bande de connexion 36 x 50 trous	FL09K
3	Prise de puces à 14 broches DIL	BL18U
1	Rouleau de 50 g de fil 22 swg	BL14Q
1	Mètre de câble-ruban à 12 voies	XR65V
8	Prises de 4 mm noires	HF69A
8	Prises de 4 mm rouges	HF73Q
8	Fiches de 4 mm noires	HF62S
8	Fiches de 4 mm rouges	HF66W
1	Prise d'alimentation 2,1 mm	RK37S
1	Prise 12 voies	YW30H
1	Boîtier plastique 115 x 95 x 37 mm	LH22Y

Nouvelles pièces

La liste des pièces renferme plusieurs noms de composants déjà décrits, mais les pièces que nous avons illustrées peuvent ne pas être familières. La prise d'alimentation, le redresseur et le régulateur font partie de l'alimentation du boîtier. Notre réalisation nous permettra d'utiliser comme entrée presque tout transformateur de secteur, pourvu que sa sortie soit comprise entre 7 V et 25 V — CA ou CC. La prise à 12 voies sera le port d'entrée/sortie du boîtier, par lequel les données sont reçues en provenance des dispositifs externes, et leur sont transmis.

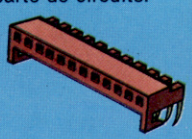


Redresseur en pont
Convertit une plage de tension d'entrée (CA ou CC) en tension CC d'une polarité connue.



Régulateur de tension
Uniformise la sortie du redresseur en pont (qui peut être ondulée) en une tension stable de 5 V CC.

Prise d'alimentation
Accepte une fiche d'alimentation de 2 mm, telle qu'utilisée sur plusieurs ordinateurs, et se monte directement sur la carte de circuits.

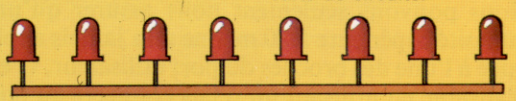


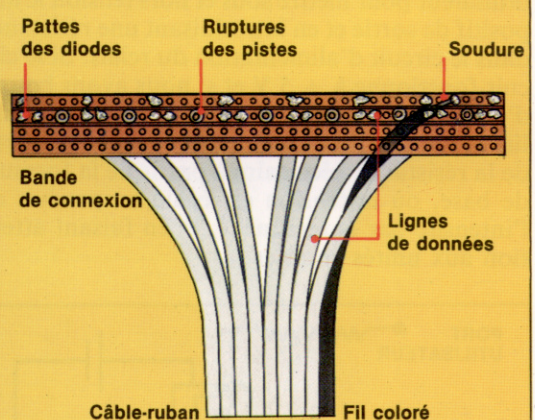
Prise Minicon
Se monte directement sur la carte et permet une connexion facile des câbles externes.

et des relais de secteur. Nous serons, alors, en mesure de commander des dispositifs domestiques tels que des lumières, des enregistreuses, des téléviseurs, etc. Nous pouvons, de plus, y ajouter un convertisseur numérique/analogique, qui nous permettra de piloter des affichages décodés à sept segments. En raison des basses tension et intensité à la sortie du port utilisateur, nous aurons également besoin d'une alimentation externe de 9 V. Chaque module sera connecté à un bus commun, le long duquel les huit lignes de données, une prise de terre et une ligne d'alimentation de 9 V seront acheminées.

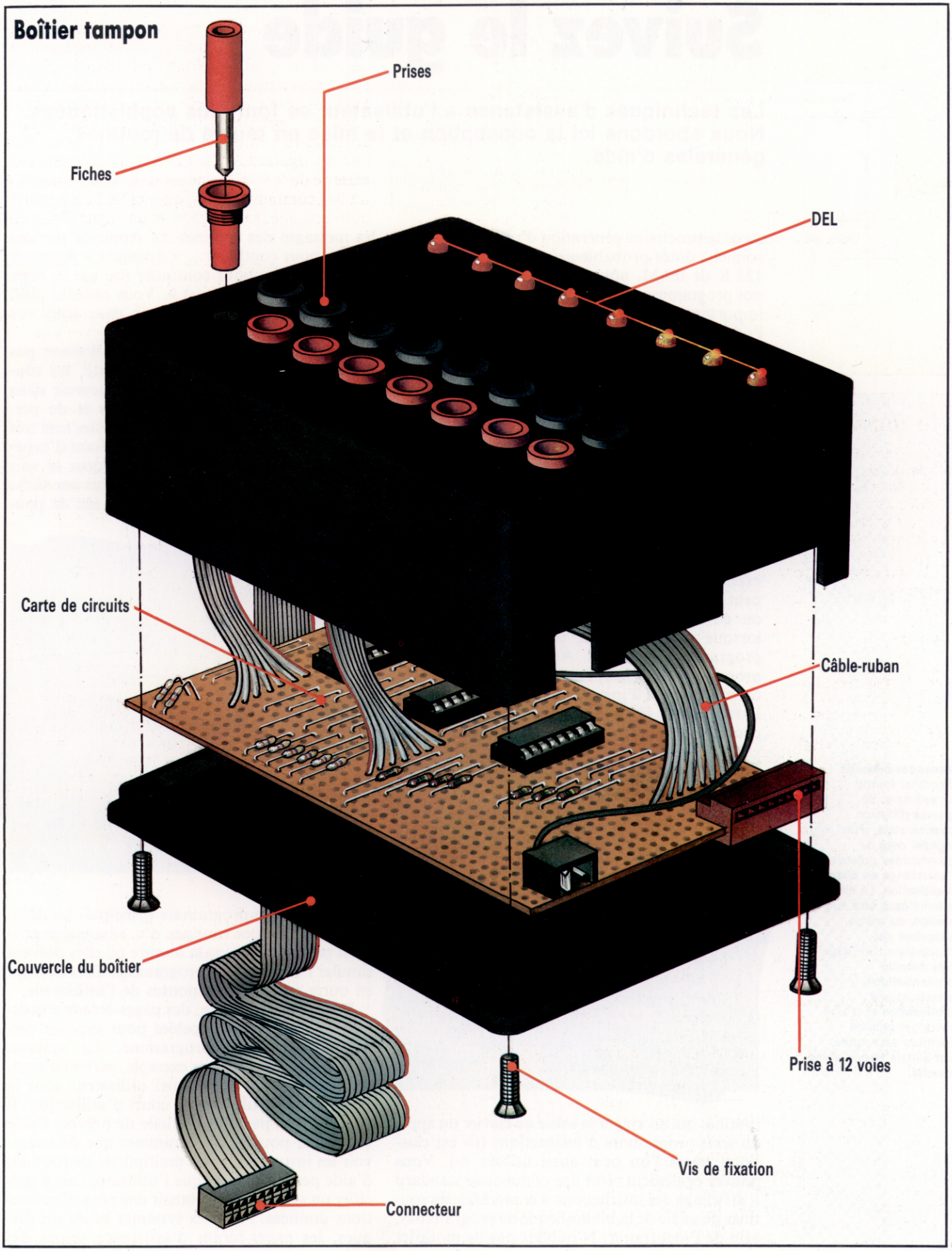
Comment construire l'affichage DEL

Les DEL occupent une bande de connexion d'une largeur de 4 pistes et d'une longueur de 36 trous. Insérez les diodes comme il est illustré, en plaçant les pattes les plus longues sur la piste latérale et en laissant 4 trous entre chacune d'elles. Cela devrait correspondre à l'espacement des perforations du boîtier; si nécessaire, repositionnez les diodes. Soudez les pattes aux pistes de cuivre, en veillant à ne pas répandre de soudure d'une piste à l'autre. Utilisez un multimètre pour vérifier la résistance entre les deux pistes; si elle est nulle, vous avez shunté les deux pistes. Coupez une longueur de 20 cm de câble-ruban à 12 voies et retirez 3 fils, ce qui donne un câble-ruban de 9 fils incluant le fil coloré. Dénudez et torsadez les extrémités des fils. Soudez le fil coloré à la piste latérale de la bande. Soudez maintenant chacun des autres fils le long de l'autre piste, en plaçant chaque fil près d'une patte de diode. Coupez la bande à 7 endroits, afin que chaque broche et que chaque paire de fils soit isolée sur sa propre portion de piste. De nouveau, testez la résistance entre les pistes et entre les coupures.
Posez délicatement les diodes et la carte dans les trous du boîtier, vissez les prises dans leurs trous, puis asseyez-vous et admirez votre travail en attendant le prochain article dans lequel vous apprendrez comment construire la carte de circuits.





Boîtier tampon



Rosalind Buckland

Suivez le guide

Les techniques d'assistance à l'utilisateur se font plus sophistiquées. Nous abordons ici la conception et la mise en œuvre de routines générales d'aide.

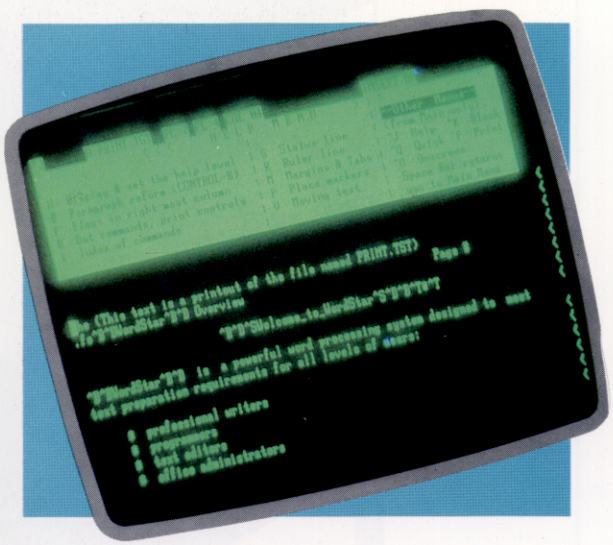
Avec la prochaine génération d'ordinateurs personnels, dotés probablement d'un minimum de 128 K de RAM, nous pourrons disposer, pour nos programmes, d'une quantité de mémoire très importante. Plus d'excuses à l'absence d'aides à l'utilisateur, d'instructions, de messages d'erreurs intelligibles ou d'assistance en cours d'utilisation !

Un programme peut prodiguer trois types d'aides : des instructions, des pages-écrans d'assistance et des notes explicatives. Les instructions peuvent prendre deux formes : soit elles sont fournies en un seul bloc au début du programme, soit, si besoin est, au cours du programme (sous la forme de messages-guides opérateurs pour la saisie, par exemple). Dans l'idéal, les deux devraient être prévus.

Dans leur forme la plus simple, les instructions consistent en une ou plusieurs pages de texte exposant le bon usage du programme. Le texte peut figurer en tant que chaînes de caractères, ou en tant qu'instructions DATA. Il s'affiche, lorsque nécessaire, par un appel à un sous-programme spécifique. Au commencement du programme principal, l'utilisateur se voit demander s'il a besoin d'instructions d'utilisation. Dans l'affirmative, le sous-programme est appelé. Il convient, par ailleurs, d'incorporer aux autres routines qui acceptent la saisie au clavier un appel du sous-programme d'instructions («?» est classique, mais l'on peut aussi utiliser «!»). Vous pouvez également créer une commande standard « affichage des instructions » et modifier les routines de saisie de la bibliothèque de programmes, afin de l'incorporer. N'oubliez pas de modifier les messages des routines : « Appuyez sur une touche pour continuer... » devenant « Appuyez sur une touche pour continuer (ou sur «!» pour accéder aux instructions) ». Vous obtenez ainsi un format standard à utiliser dans tous vos programmes.

Messages-conseils
Wordstar fournit un exemple, de grande diffusion commerciale, d'un logiciel doté de commandes prévoyant l'assistance en cours d'utilisation. Le menu affiché peut être mis en abrégé, ou même supprimé par l'utilisateur. Un fichier très détaillé d'informations, d'instructions d'utilisation et d'aide reste par ailleurs toujours accessible, sur simple frappe d'une touche.

Les instructions d'utilisation ne doivent pas nécessairement consister en du texte, les routines d'instructions pouvant se concevoir dans l'optique de fournir des exemples et de permettre à l'utilisateur de s'entraîner. Elles sont très répandues pour les programmes traitant d'expériences scientifiques. L'utilisateur peut se voir alors demander d'effectuer avec succès une tâche d'un certain niveau de difficulté avant de pouvoir accéder au programme principal. Le développement de ces routines d'« enseignement » n'est pas simple, dans la mesure où elles doivent simuler l'ensemble du programme. Elles devront en outre évaluer les réponses de l'utilisateur.



De manière similaire, des pages-écrans d'assistance peuvent être appelées pour exposer certaines parties du programme. Le système d'exploitation Unix, par exemple, permet d'accéder à la totalité du manuel utilisateur sous la forme d'une seule aide en cours d'utilisation. Il ne vous sera pas plus difficile de prévoir l'aide utilisateur pour vos programmes que de concevoir les instructions. La multiplicité de routines d'aide peut impliquer que l'utilisateur ait à spécifier un numéro pour obtenir une page d'instructions données. Avec les systèmes basés sur disques, les pages-écrans d'assistance seront des

fichiers distincts. Une routine d'aide revient alors à appeler le nom du fichier approprié.

Les routines d'aide et d'instructions peuvent prendre plus d'une page-écran d'informations. Il convient donc que l'utilisateur puisse se déplacer en avant et en arrière dans le texte. Veillez également à ce qu'il soit possible de quitter la routine d'assistance à n'importe quel moment, le programme principal devant être réintégré exactement là où il avait été laissé. Il est, en effet, très fastidieux d'avoir à subir dix pages d'informations une fois que l'on a obtenu le renseignement voulu. La routine d'aide doit positionner un drapeau, qui indique à la routine d'appel qu'elle doit revenir à la dernière instruction, le drapeau lui-même étant alors retiré.

Bonne direction
Les logiciels d'affaires de l'Apricot ACT guident l'utilisateur au travers d'une suite de programmes utilitaires compliqués, à l'aide de son système de menus hiérarchisés. L'aide (HELP) est une option sur chaque menu, qui consiste à expliquer les autres éléments du menu. (Cl. Ian McKinnell.)

L'utilisateur qui aborde un programme pour la première fois n'en connaît pas la structure, et il peut facilement s'égarer. Il est donc important de lui donner des repères pour le guider dans l'utilisation du logiciel. Les menus sont les meilleurs indicateurs du déroulement d'un programme : on peut les comparer à des panneaux indiquant les directions possibles lors d'un croisement de routes. Les systèmes Macintosh et Lisa de chez Apple, utilisent un principe similaire, mais avec des figures-guides.

On retrouve cette démarche sur de nombreux systèmes. Ainsi, le Minitel, distribué par les PTT, permet l'accès à des banques de données qui vous guident à l'aide de menus.

Certaines options sont plus importantes que les autres : elles indiquent les directions principales. Dans un système reposant sur des commandes, une douzaine sont généralement possibles. Cependant, toutes ne sont pas disponibles et ne s'appliquent pas à un moment particulier du programme. Lorsque le nombre d'options est petit, il est utile d'afficher une ligne ou deux de commentaires pour les expliciter. D'autres options, telle QUIT, sont utilisables en tout point du programme. Il est donc préférable de les afficher de manière permanente. UNDO, SAVE, ainsi que d'autres commandes spécifiques à des applications peuvent également être affichées en permanence. Une technique répandue est de programmer ces dernières sous forme de touches de fonction et d'afficher une ligne de message à l'écran pour les détailler. Il est également utile d'indiquer le moyen de sortir du programme. Cela donne confiance aux nouveaux venus, qui s'inquiètent souvent du moyen de retourner au système d'exploitation !

Des systèmes expérimentaux ont été développés dans l'optique de piloter l'utilisateur selon ses performances et d'ajuster en conséquence le niveau d'aide fourni. La commercialisation de ce type de logiciel n'est pas pour demain, mais il est possible de s'inspirer du principe pour atteindre un objectif intermédiaire. Si l'utilisateur donne son nom à chaque utilisation du programme, il est possible de tenir un fichier des utilisateurs et de leur niveau respectif. Ce niveau peut être évalué à partir du nombre d'utilisations à l'actif du sujet ou, s'il s'agit d'un jeu, de son meilleur score. Il est mis à jour à chaque fin d'utilisation du programme. Au fur et à mesure que le « niveau » s'accroît, le type d'aide apporté change, devenant plus bref et moins complet. L'utilisateur pourra aussi choisir le niveau d'aide qu'il désire, comme avec le progiciel de traitement de texte Wordstar.

L'aide incorporée au programme contribue grandement à accroître l'efficacité d'un logiciel. Une fois la routine d'aide créée (comme celle qui est donnée ici), il est facile de la modifier pour qu'elle garde trace du nombre de fois où chaque page d'aide a été demandée. Cela permet d'avoir des indications précises sur les points délicats.

Ressemblance

Nous donnons ici des solutions, qui ne sont évidemment pas les seules possibles, aux exercices de programmation proposés précédemment dans cette rubrique.

Les exercices de programmation proposés précédemment ressemblaient tous à l'exemple du programme « Numéros de téléphone » dans la mesure où ils contrôlent le port utilisateur et comparent son état à une certaine valeur de référence.

1. Alarme contre le vol

```
10 REM VERSION BBC 1.1
20 DATREG = &FE60:DDR=&FE62
30 ?DDR=0:? DATREG=127
40 REPEAT
50 PRINT "TOUT VA BIEN"
60 ALARME=? DATREG
70 UNTIL ALARME<>255
80 PRINT "ALARME"
90 FOR PIN=0 TO 7
100 IF (ALARME AND 2 PIN)=0
THEN PRINT "ALARME SUR
INTERRUPTEUR NO"PIN
110 NEXT
120 END
```

```
10 REM CBM64 VERSION 1.1
20 DDR=56579: DATREG=56577
30 POKE DDR,0: POKE DATREG,255
40 FOR C=0 TO 1 STEP 0
50 PRINT "TOUT VA BIEN"
60 ALARME=PEEK (DATREG)
70 IF ALARME>255 THEN C=1
80 NEXT C
90 PRINT "BRUITS ALARME"
100 FOR PIN=0 TO 7
110 IF (ALARME AND 2 PIN)=0
THEN PRINT ALARME SUR
INTERRUPTEUR #";PIN
120 NEXT:END
READY.
```

2. Compte d'impulsions (i)

```
10 REM VERSION BBC 1.2
20 DATREG=&FE60:DDR=&FE62
30 ?DDR=0:?DATREG=127
40 COMPTE=0: REM INIT COMPTE
50 TEMPS=0: REM INIT TEMPS
60 REPEAT
70 REPEAT
80 UNTIL ?"DATREG<>255
90 COMPTE=COMPTE + 1
100 UNTIL TEMPS > 6000: REM 1 MIN
110 PRINT "IMPULSIONS="COMPTE"
120 END
```

```
10 REM VERSION CBM64 1.2
20 DDR=56579:DATREG=56577
30 POKE DDR,0: REM
40 T=TI: REM INTIT TEMPS
50 IF PEEK(DATREG)=255 THEN 50
60 COMPTE = COMPTE + 1
70 IF (TI-T) 3600 THEN 50
80 PRINT "IMPULSIONS=";COMPTE
90 END
READY.
```

3. Compte d'impulsions (ii)

```
10 REM VERSION BBC 1.3
20 DATREG=&FE60:DDR=&FE62
30 DIM C 8
40 ?DDR=0:TEMPS=0
50 REPEAT
60 FOR N=0 TO 7
70 C(N)=C(N)+NOT(?DATREG AND(2:N))
80 NEXT N
90 UNTIL TEMPS < 6000
100 FOR N=0 TO 7
110 PRINT "LIGNE"N""="C(N)
120 NEXT N
130 END
```

```
10 REM VERSION CBM64 1.3
20 DDR=56579:DATREG=56577
30 POKE DDR,0:T=TI
40 FOR K=0 TO 7
50 C(K)=C(K)+NOT(PEEK(DATREG)AND(2 K))
60 NEXT K
70 IF (TI-T) 3600 THEN 40
80 FOR K=0 TO 7
90 PRINT "LIGNE";K;"="C(K)
100 NEXT K
110 END
```

4. Combinaison

```
10 REM VERSION BBC 1.4
20 DATREG=&FE60:DDR=&FE62:FLAG=1
30 ?DDR=15: REM ENTREE DES LIGNES 0-3
40 DIM C(3)
50 REM CODE=359
60 C(1)=3:C(2)=5:C(3)=9
70 FOR N=1 TO 3
80 A$=GET$: REM FRAPPE TOUCHE
90 NEXT N
100 IF FLAG=1 THEN PRINT "OUVERT"
ELSE PRINT "MAUVAIS COMBINAISON"
110 END
```

```
10 REM VERSION CBM64 1.4
20 DDR=56579:DATREG=56577
30 POKE DDR,15: REM ENTREE DES LIGNES 0-3
40 C(1)=1:C(2)=2:C(3)=4: REM CODE 124
50 FOR K=1 TO 3
55 PRINT"ENTER A DIGIT"
60 GET A$: IF A$="OUVERT" THEN 60:REM ATTENDRE TOUCHE
65 PRINT A$
70 IF PEEK(DATREG)<>C(K) THEN FL=1
80 NEXT K
90 IF FL=1 THEN PRINT"MAUVAISE COMBINAISON": END
100 PRINT
110 END
```

5. Changement de couleur

```
10 REM VERSION BBC 1.5
20 MODE5
30 DATREG=&FE60:DDR=&FE62
40 ?DDR=120:REM TOUTES LES ENTREES SAUF D7
45 ?DATREG=255
50 A$=GET$: REM ATTENDRE TOUCHE
60 SCOL=?REGDON: REM ARRIERE-PLAN>127
70 GCOL0,SCOL:REM CHANGER COULEUR ECRAN
80 CLG: REM EFFACER ECRAN GRAPHIQUE
90 END
```

```
10 REM VERSION CBM64 1.5
20 DDR=56579: DATREG=56577
30 POKE DDR,0:REM TOUTE ENTREE
40 GET A$: IF A$="" THEN 40
50 SCOL=PEEK(DATREG)
60 POKE 53281,SCOL
70 END
```

Doué pour les jeux

Sega est une société japonaise connue pour ses jeux de cafés. Un ordinateur conçu par cette société, le SC3000H, se devait d'être doué pour les jeux...

Bien qu'il ne soit en rien révolutionnaire, ni par sa conception ni par sa fonction, le SC3000H, qui utilise le classique microprocesseur Z80A, est un micro bien conçu, pouvant recevoir de nombreuses extensions et doté de nombreux logiciels. D'aspect agréable, le Sega est léger (1,100 kg). Les touches, en plastique moulé, sont de type machine à écrire avec une course d'environ 1 cm. Dans l'ensemble, sa qualité est bonne pour une machine au prix abordable. Outre les touches standard, Sega dispose d'une touche de fonction, non programmable, utilisée pour saisir des mots clés BASIC, d'une touche graphique pour accéder aux symboles graphiques du clavier, des touches « vide-écran » et « insertion/effacement », et d'un ensemble de quatre touches fléchées, pour les jeux notamment. L'absence de symboles graphiques sur les touches est cependant un gros inconvénient, car elle rend l'utilisation du BASIC plus délicate.

Le SC3000H est équipé, de manière satisfaisante, d'interfaces. Sur le côté gauche, deux ports manettes de jeu, du genre Atari, sont disponibles. Une case destinée à recevoir une cartouche de ROM est accessible sur le côté droit. Enfin, à l'arrière de la machine, des connexions concernent un poste de TV, un port moniteur couleur composite, un port imprimante de type DIN, une interface cassette et un boîtier d'alimentation pour l'adaptateur 9 V. Un commutateur est également fourni en liaison avec le téléviseur, ainsi qu'une cartouche BASIC accompagnée d'un petit manuel d'instructions.

La mise en œuvre du Sega est immédiate, ce qui est préférable, vu le peu de documentation d'accompagnement. Un dépliant de deux pages décrit la connexion de l'ordinateur avec le téléviseur ainsi que l'alimentation. Une diode-témoin verte s'allume lorsque la machine fonctionne. La documentation ne mentionne pas le fait, pourtant important, qu'en l'absence d'une cartouche, soit BASIC, soit de jeu, dans le port de la ROM l'ordinateur ne peut fonctionner. Il n'y a pas d'interpréteur BASIC, et une fois la cartouche BASIC chargée, le SC3000H ne dispose plus que de 515 octets de RAM utilisateur, ce qui est déri-

Le nouveau nippon est arrivé
Destiné au marché des ordinateurs bon marché, le Sega SC3000H est conçu pour concurrencer le Spectrum ou le Commodore 64.
La machine comporte un ensemble complet de périphériques, dont un magnétophone à cassette, des manettes de jeu et une imprimante couleur/traceur.
(Cl. Chris Stevens.)

soire et représente moins que le ZX81 sans extension. C'est un handicap sérieux pour l'utilisateur, qui devra acquérir un module d'extension mémoire s'il veut utiliser la machine pour autre chose que des jeux.

Le manuel du SC3000H peut également être trompeur, dans la mesure où il se réfère au clavier avec touches de fonction du SC3000, et oblige l'utilisateur à consulter le diagramme du clavier pour repérer les symboles graphiques. Le diagramme du clavier montre aussi les mots clés BASIC imprimés sur ou au-dessus de certaines touches. On y accède en maintenant appuyée la touche de fonction et en appuyant, en même temps, sur la touche alphanumérique appropriée. On peut utiliser de la sorte des commandes telles que RUN, LOAD et GOTO, des fonctions mathématiques (ABS, SIN, COS, TAN) et des fonctions de chaînes de caractères telles que LEFT$, RIGHT$ et MID$. Mais il faut toujours se référer au manuel pour trouver les touches appropriées.

Le graphisme du SC3000H se révèle assez étonnant à l'utilisation. L'affichage concerne deux modes, le mode texte avec 24 lignes de 40 colonnes et deux couleurs, et le mode graphique d'une résolution de 256 par 192 pixels pouvant porter sur seize couleurs. La luminosité peut être réglée et rendre jusqu'à deux cent dix nuances. Il est en outre possible de créer jusqu'à trente-deux figures graphiques. Le BASIC du Sega, très semblable au BASIC étendu de Microsoft, utilise les commandes DRAW, COLOR, PAINT et SPRITE pour la programmation graphique.

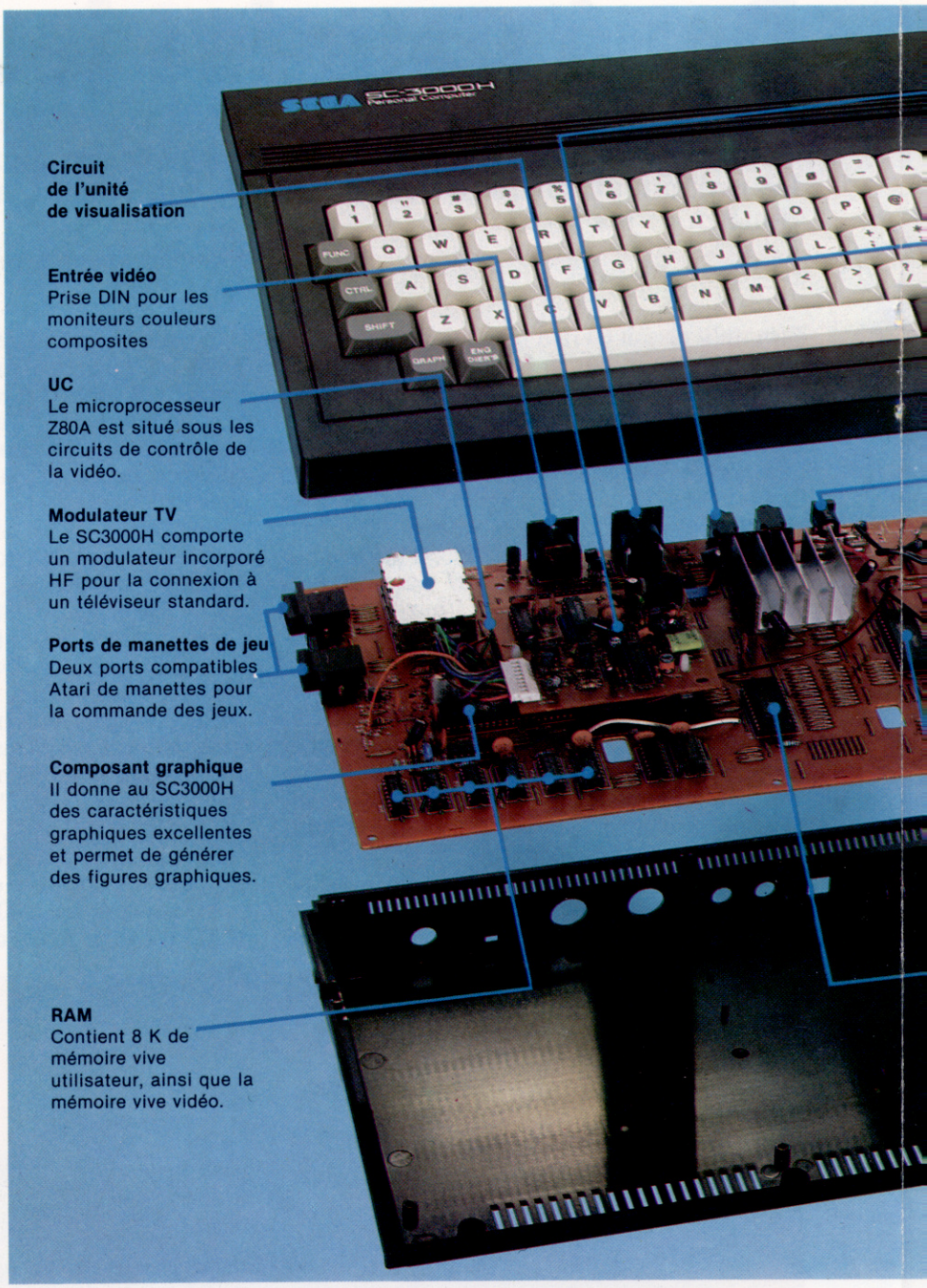
Le SC3000H dispose de six canaux pour le son, que l'on adresse par des commandes POKE ou par les commandes BASIC BEEP et SOUND. Une cartouche musique aide à la composition de mélodies, mais les résultats sont loin des caractéristiques d'un synthétiseur, quoi qu'en dise le manuel...

L'expérience de Sega en matière de jeux de cafés se retrouve dans la qualité de ses logiciels de jeu. Le graphisme est généralement bon, bien qu'il semble que l'affichage simultané de texte et de graphiques soit assez difficile. Visuellement, ils ont le même attrait que les jeux de cafés, et le son de bonne qualité du SC3000H contribue à cette réussite. La touche RESET est utilisée de manière inhabituelle : au lieu de réinitialiser un jeu, elle constitue un interrupteur-bascule pour marquer une pause.

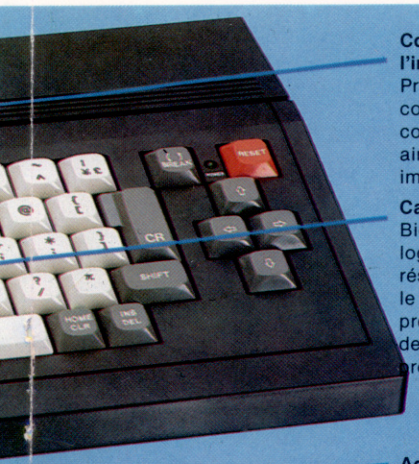
Sega prévoit deux types de manettes de jeu, mais aucune ne permet un déplacement à l'écran maniable et rapide. En revanche, l'ensemble des touches de déplacement du curseur permet de jouer facilement depuis le clavier.

Il existe de nombreux périphériques pour le Sega, dont un magnétophone à cassette, une imprimante couleur/traceur et un module d'extension. Ce dernier fournit 64 K de RAM supplémentaires et comporte un lecteur incorporé de disque.

Sega SC3000H est une machine agréable à utiliser, qui comporte certaines caractéristiques inhabituelles pour son prix. Le fabricant devrait faire un effort quant à la documentation, et le



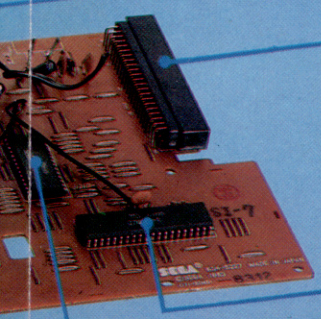
Les manettes de jeu du Sega SC3000H
Deux types de manettes pour deux manières de jouer : tous les deux utilisent un mécanisme à huit positions de contact pour le déplacement de la manette, et sont compatibles avec la connexion standard à 9 broches d'Atari. (Cl. Chris Stevens.)



Connection de l'imprimante
Prise de type DIN pour connexion à l'imprimante couleur/traceur de Sega, ainsi qu'avec les autres imprimantes.

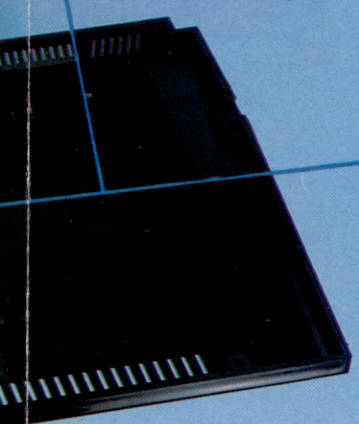
Cassette E/S
Bien que la plupart des logiciels de chez Sega résident sur cartouche, le contrôle cassette est prévu pour la sauvegarde de vos propres programmes.

Adaptateur d'alimentation



Port cartouche
La plupart des logiciels de jeu de Sega sont sur des cartouches de ROM connectées par l'intermédiaire de ce port.

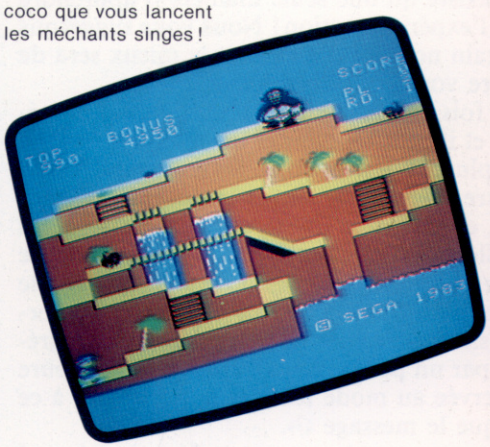
Contrôle des E/S
Il contrôle de nombreuses fonctions d'entrée/sortie.



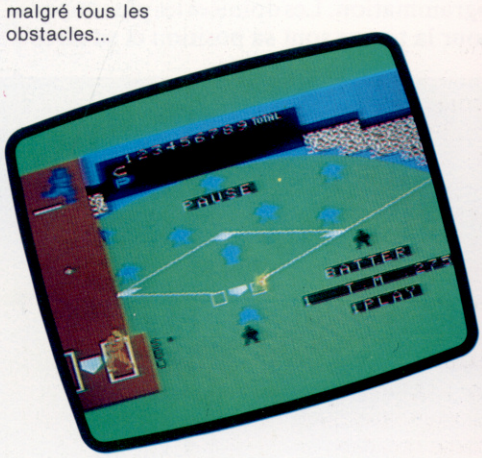
ROM
Comme le SC3000H ne comporte pas de langage BASIC incorporé, la ROM est essentiellement utilisée pour les opérations internes et pour l'affichage.

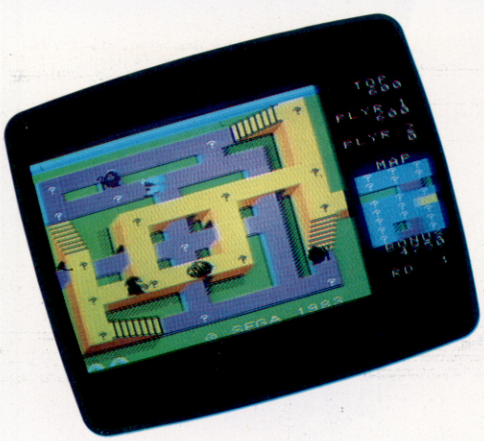
Safari
Chassez les fauves de la jungle avec un fusil qui endort...

Congo Bongo
Pourchassez le gorille à flanc de falaise. Attention aux noix de coco que vous lancent les méchants singes !



Baseball
Marquez des points malgré tous les obstacles...





Le mystère de Sinbad
Un mélange subtil de labyrinthe et d'aventures.

Les logiciels de Sega
Ils concernent surtout les jeux et sont à la hauteur de cette société réputée pour ses jeux de cafés. De nombreux jeux disposent de l'excellent type d'affichage « 3D » et mettent en œuvre des formats innovateurs.

SEGA SC3000H

PRIX
★ ★

DIMENSIONS
353 × 210 × 46 mm.

UC
Z80A, 4 MHz.

MÉMOIRE
8 K RAM extensible de manière interne à 48 K.
18 K ROM, extensible de manière interne à 32 K.
16 K RAM vidéo.

AFFICHAGE
24 lignes de 40 colonnes pour le texte, résolution graphique de 256 × 192 avec figures graphiques, et 16 couleurs en 15 niveaux d'intensité.

INTERFACES
Port cartouche ROM, ports manettes de jeux (2), TV et interface vidéo, audio, cassette et imprimante.

LANGAGES DISPONIBLES
BASIC, LOGO.

CLAVIER
64 touches, de type machine à écrire; touches pour le déplacement du curseur et touches pour graphiques et caractères spéciaux.

DOCUMENTATION
Brochure de deux pages pour l'installation, plus le manuel accompagnant la cartouche BASIC. La brochure est assez complète, mais il n'est dit nulle part qu'une cartouche doit être chargée pour que la machine tourne. Toute la documentation se réfère à la version « touches de fonction » du clavier.

FORCES
Machine agréable dotée de bonnes capacités graphiques et sonores, ainsi que d'un bon BASIC. Pour le prix, c'est un bon micro de jeux pour les débutants.

FAIBLESSES
La documentation est catastrophique. La mémoire de 8 K seulement fait qu'il est pratiquement impossible de programmer sur cette machine. Le clavier ne comporte pas de symboles pour les caractères spéciaux.

clavier devrait être repensé pour indiquer les mots clés et les symboles graphiques. Mais le principal handicap concerne évidemment le manque de mémoire utilisateur : telle qu'elle est, la machine ne peut recevoir que des logiciels de sa marque.

Tortue tournante

Il existe aujourd'hui de nombreuses versions de LOGO. Nous traitons ici des utilisations de la tortue, qui permet de tracer des formes complexes sans difficulté.

Abréviations

De nombreuses commandes LOGO ont des abréviations; voici celles des commandes énumérées ici :

AVANCE	AV
RECULE	RE
DROITE	DR
GAUCHE	GA
LÈVEPLUME	LP
POSEPLUME	PP
AFFICHER	AF
PLEINÉCRAN	PE
ÉCRANTEXTE	ET
ÉCRANÉCLATÉ	EE

La première version de LOGO disponible sur micro-ordinateur fut celle du MIT. Elle est maintenant considérée comme la version standard tournant sur la plupart des machines. Nos programmes d'exemples utilisent tous le LOGO MIT. Lorsque des différences interviennent entre le LOGO MIT et d'autres versions de LOGO, elle sont exposées à la rubrique « Nuances ».

Il n'existe qu'une seule manière d'apprendre LOGO, l'expérimentation! Nous vous suggérons un certain nombre d'essais, et le mieux sera de résoudre vos propres problèmes.

Une fois chargé, LOGO est en mode « immédiat », et il peut recevoir et exécuter des ordres. Sur la plupart des versions, les commandes doivent être passées en lettres majuscules. Tapez DESSINE, et l'écran se divise en deux parties (cela s'appelle « mode éclaté »). La partie supérieure est réservée au mode graphique. Elle occupe le plus de place dans la zone d'affichage. La tortue est située au centre de l'écran et elle est représentée par un petit triangle. La partie inférieure est réservée au mode texte et ne comporte à ce stade que le message «?».

La tortue graphique est un objet avec lequel on communique en lui donnant des ordres. Cette notion d'objet concret facilite la compréhension de la programmation. Les données les plus importantes pour la tortue sont sa position et sa direction; il faut également savoir si le crayon qu'elle porte est baissé ou levé. S'il est levé, la tortue avance sans laisser de trace; s'il est baissé, la tortue trace une ligne en se déplaçant. En tapant DESSINE, vous positionnez la tortue au centre de l'écran, dirigée vers le haut, le crayon baissé.

Donnons maintenant un ordre à la tortue :

```
AVANCE 40
```

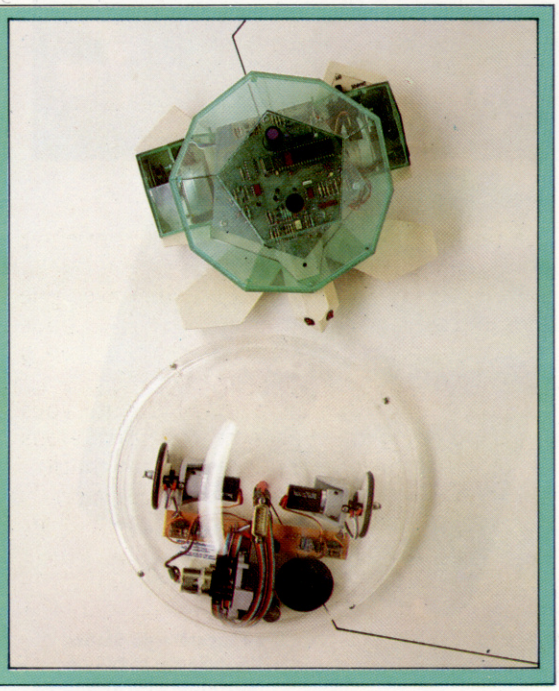
Elle se déplacera de 40 unités vers le haut en traçant une ligne. AVANCE est une commande de la tortue, le nombre 40 est une donnée en entrée. Certaines commandes nécessitent des données, d'autres pas. DESSINE, par exemple, n'est pas accompagné de données.

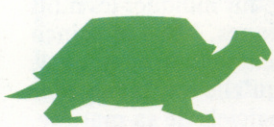
Une deuxième commande de la tortue est RECULE. RECULE 10 dit à la tortue de revenir en arrière de 10 unités. Ainsi, AVANCE et RECULE changent-ils tous les deux la position de la tortue à l'écran (tous les deux nécessitent des données). Par contre, DROITE et GAUCHE ne modifient pas sa position, mais la font simplement tourner; ils changent la direction de la tortue. Ces deux commandes nécessitent un angle compris entre 0 et 360° comme données d'entrée.

Essayez ces commandes, et tracez des formes géométriques simples. Voyez ce qui arrive lorsque vous demandez à la tortue de dépasser les limites de l'écran. Utilisez des nombres négatifs

A la rencontre de la tortue

La tortue est un petit robot destiné à dessiner. Il comporte des roues actionnées par des moteurs progressifs, et un crayon rétractable. On peut lui demander d'avancer, de reculer, d'aller à gauche, à droite et de lever ou baisser son crayon.
Le crayon trace un trait sur la surface. Les premières tortues avaient la carapace très réaliste et bombée, comme celle montrée ici. Elles étaient contrôlées depuis le clavier à l'aide d'un câble parallèle.
Les dernières tortues sont télécommandées. Une version radiocommandée vient de sortir à un prix à peine supérieur au prix normal.
Il existe encore une autre tortue, Valiant Turtle, commandée par infrarouges et un peu plus chère. Dans ce cas, plus de câble! Par extension, le nom « tortue » s'applique aussi au curseur de l'écran LOGO.
Il s'agit le plus souvent de triangles; Atari propose pourtant une vraie petite tortue.
(Cl. Paul Chave.)





en entrée. Pour effacer l'écran, tapez DESSINE. En essayant de nouvelles commandes, vous verrez peut-être la tortue se déplacer dans la partie texte de l'écran. Elle sera alors « derrière » le texte et ne pourra être vue. Utilisez alors les commandes suivantes.

PLEINÉCRAN : la totalité de l'écran est utilisée pour le mode graphique.
ÉCRANTEXTE : supprime la partie graphique et ne laisse que le texte.
ÉCRANÉCLATÉ : renvoie au mode éclaté.
LÈVEPLUME : permet de déplacer la tortue sans qu'elle laisse de trace.
POSEPLUME : la tortue trace une ligne.

Dans les deux sens

Les commandes que nous avons vues jusqu'ici font obéir aveuglément la tortue. Mais vous pouvez aussi utiliser LOGO pour obtenir des informations de la tortue. La direction de la tortue se mesure en degrés. Une direction de 0 indique que la tortue est tournée vers le haut. La direction est mesurée, selon le sens des aiguilles d'une montre, jusqu'à 360°. Pour connaître la direction de la tortue, tapez :

```
AFFICHER DIRECTION
```

La position de la tortue est définie selon un système de coordonnées, dont l'origine est au centre de l'écran (coordonnées « x,y » = « 0,0 »). Vous pouvez connaître à tout instant la position de la tortue en tapant :

```
AFFICHER XCOR AFFICHER YCOR
```

Expérimentez ces commandes en traçant une forme géométrique et en vérifiant, ensuite, la position et la direction de la tortue. Vous utiliserez alors ces données pour la faire revenir au point de départ.

Vous avez peut-être déjà rencontré des messages d'erreur s'affichant dans la zone texte de LOGO. Sinon, faites délibérément une erreur pour voir. Tapez, par exemple :

```
AVANCE50
```

Le message suivant apparaîtra :

```
IL N'EXISTE PAS DE PROCÉDURE APPELÉE AVANCE50
```

La raison est que LOGO exige un espace entre la commande et les données, entre AVANCE et 50, afin d'éviter toute confusion possible avec une commande qui s'appellerait AVANCE50. Vous obtiendrez également un message d'erreur si vous tapez une commande en minuscules.

LOGO est doté d'un éditeur de ligne qui vous permet de corriger le libellé de vos instructions une fois tapées, avant de faire un retour-chariot. Vous utilisez les touches du curseur pour vous déplacer sur la ligne du texte à modifier. Pour insérer des caractères, tapez-les simplement, le texte à droite de l'erreur se déplaçant automatiquement pour faire autant de place que nécessaire. La touche d'effacement supprime les caractères à gauche du curseur. Une fois la ligne corrigée, faites <RC> pour communiquer la nouvelle instruction à LOGO. Si vous avez déjà validé la commande lorsque vous vous rendez compte de la faute de frappe, faites CONTROL-P pour restituer à l'éditeur la dernière ligne tapée. Cette caractéristique est également utilisable pour répéter une commande.

Nous pouvons maintenant passer à quelque chose de plus mathématique : par exemple un carré (quatre côtés égaux et quatre angles égaux de 90°). Vous obtiendrez un carré de la sorte :

```
AV 50 DR 90
AV 50 DR 90
AV 50 DR 90
AV 50 DR 90
```

Vous remarquez que nous avons abrégé les commandes et que nous pouvons faire figurer plus d'une commande par ligne.

Vous objecterez peut-être, une fois le carré dessiné, que ce dernier ressemble davantage à un rectangle qu'à un carré. C'est là un problème technique dû au fait que l'unité horizontale n'est pas égale à l'unité verticale de mesure. Il existe cependant une commande LOGO pour rectifier ce décalage : ASPECT suivi d'un nombre (0,8 par défaut). Cela permet de modifier l'écart d'échelle. Mettez maintenant en pratique les exercices suivants : dessiner un hexagone, un pentagone, divers rectangles, une forme circulaire, un parallélogramme, et toute autre forme...

Vous pouvez simplifier certaines commandes et réduire la frappe nécessaire en tapant RÉPÈTE. Pour dessiner à nouveau le carré, tapez simplement :

```
RÉPÈTE 4[AV 50 DR 90]
```

RÉPÈTE est une commande qui suppose deux entrées. D'abord un nombre indiquant le nombre de répétitions, ensuite la liste des commandes concernées. La liste doit toujours figurer entre crochets. Ainsi, pour notre exemple du carré, il s'agit de répéter quatre fois la séquence «AVANCE 50 DROITE 90». Utilisez maintenant RÉPÈTE pour simplifier les formes que vous avez précédemment élaborées. Essayez enfin de dessiner des formes étoilées que nous vous montrons dans certaines des illustrations ci-contre. Vous vous rendrez compte qu'il s'agit d'un jeu d'enfants !

Nuances de logo

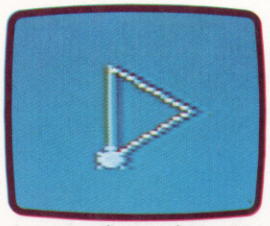
Sur Atari, le curseur-tortue est bien une tortue, et non un triangle ! Pour toutes les versions LCSI : utiliser VIDE-ÉCRAN (abréviation VE) pour commencer a dessiner.

Utiliser CONTROL-Y pour rappeler la dernière ligne (le Spectrum n'a pas cette caractéristique).

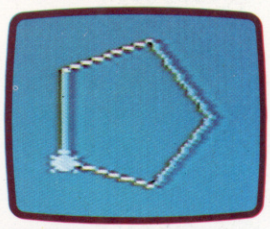
Pour modifier l'aspect (la différence d'échelle) : LCSI Apple – SETCRUNCH suivi de la nouvelle mesure; Atari – SETSCR suivi de la nouvelle mesure; Spectrum SETCRUNCH suivi par les deux valeurs de coordonnées [x,y] (la norme est [100 100]).

Exercices logo

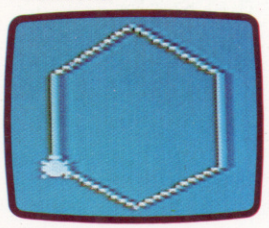
Pouvez-vous écrire les procédures pour créer ces formes ? Ces exemples ont été réalisés par LOGO LCSI sur Atari 600XL.



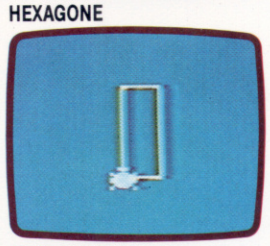
TRIANGLE ÉQUILATÉRAL



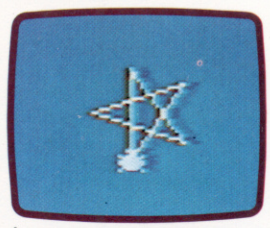
PENTAGONE



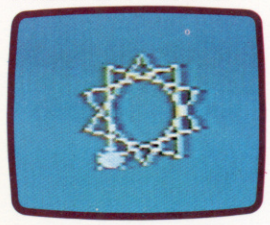
HEXAGONE



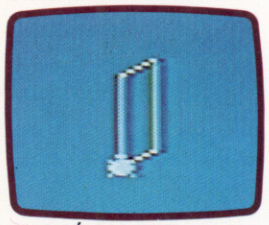
RECTANGLE



ÉTOILE A CINQ BRANCHES



ÉTOILE A TROIS BRANCHES



PARALLÉLOGRAMME

Ian McKinnell

Séquence sonore

Le mixage séquentiel est important sur scène et dans le studio, où une synchronisation précise est vitale pour le mixage du son. Examinons un développement de ce mixage : l'interface MIDI.

Le séquenceur a eu un impact très important sur la création musicale, autant pour les représentations sur scène que dans les studios. Mais il a pour inconvénient de se limiter à un seul synthétiseur. Si un joueur de clavier possède deux synthétiseurs, l'un doté de puissantes possibilités de mixage séquentiel, l'autre produisant un excellent son, il est impossible de combiner ces deux unités. Le problème s'aggrave dans les studios, où plusieurs équipements différents produisent les sons. La fonction d'une interface numérique pour des instruments musicaux est d'établir cette connexion et de confier la commande du système à une machine dotée d'un excellent potentiel de traitement.

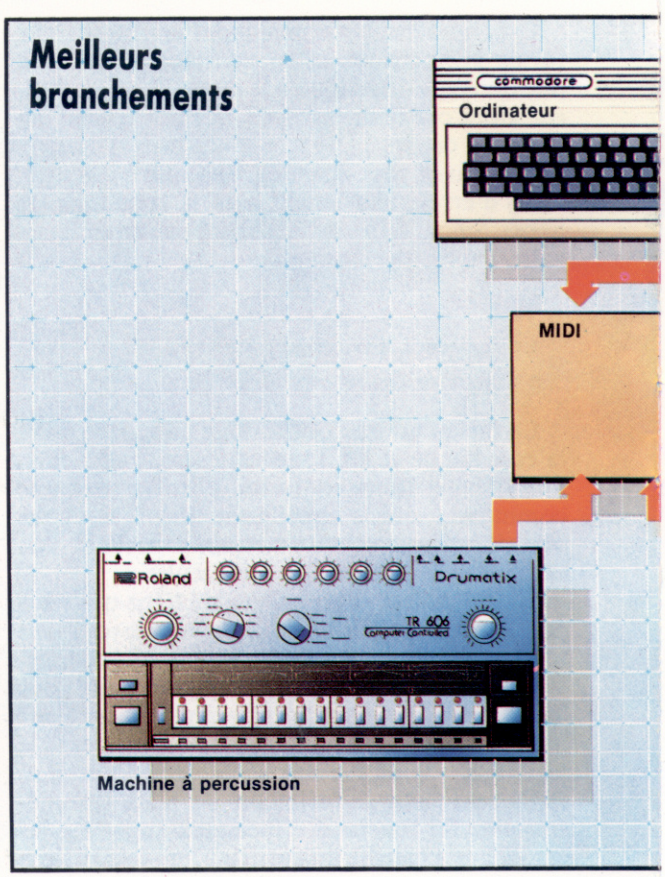
MIDI tente de parvenir à ce résultat sous une forme standardisée, afin de permettre à tout système numérique de contrôler un autre système dans le domaine de la création musicale.

L'interface MIDI (Musical Instrument Digital Interface) fut développée après une série de rencontres entre les principaux fabricants japonais et américains d'instruments numériques. Les spécifications actuelles furent arrêtées en août 1983. Cette conception repose sur un circuit de commande qui transmet des données d'un instrument à l'autre, ou d'un micro-ordinateur à un instrument. MIDI utilise un processeur à 8 bits. Ses concepteurs eurent à choisir entre deux modes de transmission – parallèle ou série. La transmission parallèle entraîne la présence de lignes individuelles, afin que les 8 bits de chaque octet soient envoyés simultanément ; d'où une vitesse de transmission très élevée. Son inconvénient réside dans le coût supplémentaire qu'elle entraîne et dans la présence d'au moins huit fils câblés à un connecteur à 25 broches de type D. La transmission série n'utilise que deux lignes. Sur l'une des lignes, les bits de données sont envoyés un à la fois ; une seconde ligne permet à l'instrument récepteur de signaler à l'instrument maître ou au micro-ordinateur des erreurs de parité : la transmission série est donc plus lente que la transmission parallèle, mais elle offre l'avantage d'utiliser un connecteur plus simple et beaucoup moins coûteux.

L'objectif premier de MIDI était de fournir un circuit de commande qui puisse améliorer la musique électronique et autoriser une forme de contrôle de la tonalité. Son prix devait le situer à la portée de la plupart des propriétaires d'ordinateurs domestiques et des musiciens utilisant des synthétiseurs et des batteries électroniques. Voilà pourquoi la transmission série fut retenue.

MIDI transmet de façon asynchrone, en empruntant les mêmes lignes que l'interface RS232. Lorsque les données sont transmises de façon asynchrone, chaque octet doit être défini pour l'instrument récepteur. Avec MIDI, cela est réalisé par un ACIA (adapteur d'interface de communication asynchrone) Motorola 6850, qui ajoute, à partir de l'instrument maître, 2 bits supplémentaires à chaque octet. Le premier est « 0 », le bit de départ, suivi des 8 bits de données série, le tout se terminant par un « 1 », le bit d'arrêt. Ce mot série à 10 bits est alors transmis à l'instrument récepteur où une seconde puce ACIA le reconvertit en données réelles de 8 bits.

La précieuse puce ACIA est protégée dans le circuit par une opto-isolation. Un opto-isolateur est un dispositif qui utilise des cellules photo-électriques pour permettre à deux circuits électriques non connectés d'échanger des signaux tout en demeurant électriquement isolés ; les incidents de surtension ne peuvent donc pas endommager la puce.



Glissade
Le glissando est facilement obtenu sur un instrument à cordes, en faisant glisser les doigts le long du manche. Une programmation soignée est nécessaire pour reproduire de manière vraisemblable cet effet de glissement.

MIDI présente une différence importante de conception par rapport à l'interface RS232. La vitesse de transmission de données de l'interface RS232 est de 1 920 mots série par seconde, ou 19,2 kilobauds : MIDI n'atteint que la moitié de cette vitesse.

Cela ne compromet pas la compatibilité avec les ordinateurs domestiques, puisque les circuits logiques internes du MIDI règlent une nouvelle vitesse d'horloge de 3 125 mots par seconde (31,25 kilobauds). Cette vitesse est assez élevée pour une transmission série, mais on peut déjà dire qu'elle ne l'est pas encore assez.

MIDI est destiné à servir d'interface avec plus qu'un seul instrument. Lorsque plus d'un instrument reçoit des instructions MIDI, il faut d'abord que les données appropriées soient envoyées au bon instrument, car, dans le cas contraire, il serait possible qu'une batterie finisse par essayer de jouer un morceau musical très doux et, à l'inverse, qu'un synthétiseur polyphonique reproduise — ou tente de le faire — une rythmique à partir du *do* de référence. Les instruments compatibles avec MIDI sont censés avoir un code d'identification numérique. L'un des seize canaux MIDI correspond à ce code, de telle sorte que seul ce canal accepte des données pour cet instrument. La première partie d'une transmission MIDI consiste, donc, en un octet d'état qui comporte ce numéro. Toutes les données qui suivent cette instruction d'acheminement peuvent alors spécifier comment la commande doit être interprétée.

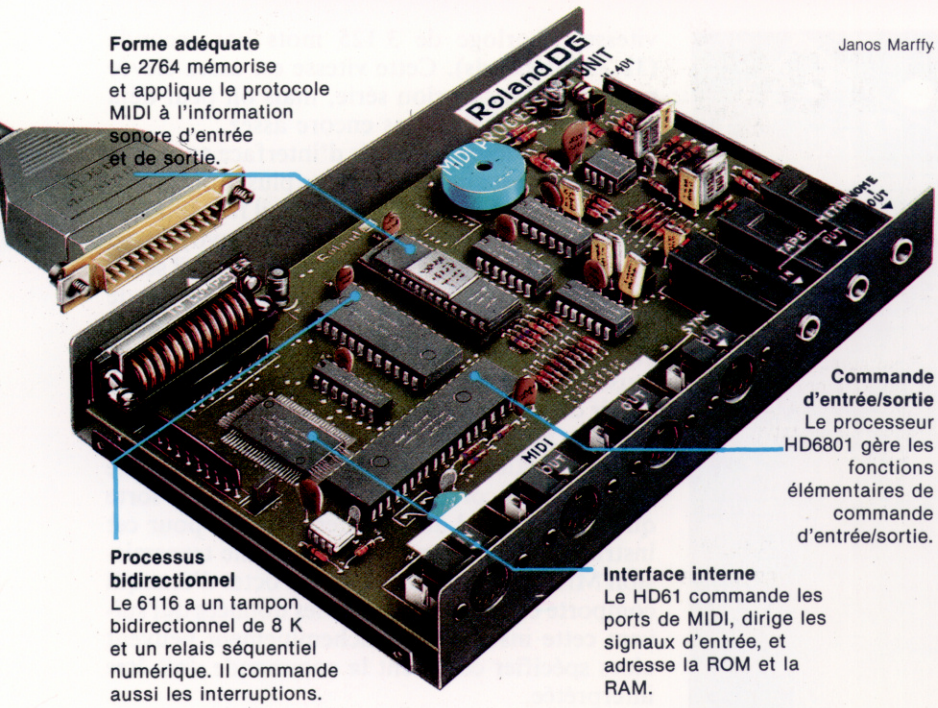
L'unité, 100 × 120 × 45 mm environ, possède deux prises DIN à cinq broches, portant les inscriptions « MIDI IN » et « MIDI OUT ». « MIDI IN » accepte toutes les instructions provenant d'un micro-ordinateur ou d'un synthétiseur maître, et « MIDI OUT » transmet la chaîne de bits modifiée à l'instrument récepteur. Des modèles possèdent aussi « MIDI THRU », une seconde prise de sortie qui transmet simplement le flux de bits initial non modifié à « MIDI IN ». Cela peut être envoyé à une seconde interface. Le câble, qui a une longueur maximale de 15 m, est muni de fiches DIN à cinq broches et se branche sur le panneau arrière d'un micro-ordinateur ou d'un synthétiseur maître.

Kevin Jones

« Do » de référence

Imaginons un nouvel utilisateur de MIDI. Il désire essayer une courte mélodie. Elle commence par le *do* de référence, monte jusqu'au *mi,* puis au *sol,* et ainsi de suite. Le mode d'instruction dépendra du type de logiciel musical qu'il utilise. Il peut utiliser un crayon optique pour tracer les notes sur l'écran. Il peut entrer l'information au clavier de son micro-ordinateur, à l'aide d'un langage de composition musical, en travaillant toujours à l'écran. Quel que soit le mode d'entrée de la musique, la transmission MIDI sera toujours identique : pour démarrer la mélodie, le premier octet — transmis sous la forme d'un mot série doté de 2 bits supplémentaires provenant de la puce ACIA — enverra l'instruction JOUER/SUR CANAL 6; le deuxième octet, DO SOUS le LA DU DIAPASON.

L'instruction minimale produira la note do de référence sur l'instrument récepteur. Et le synthétiseur continuera à émettre cette note, sauf si une instruction limite sa durée, telle que ARRÊTER DE JOUER/SUR CANAL 6, octet un; DO DE RÉFÉRENCE, octet deux; et DURÉE DE X, octet trois. Si cette seconde instruction n'est pas donnée et si le reste de la



Janos Marffy

Forme adéquate
Le 2764 mémorise et applique le protocole MIDI à l'information sonore d'entrée et de sortie.

Commande d'entrée/sortie
Le processeur HD6801 gère les fonctions élémentaires de commande d'entrée/sortie.

Processus bidirectionnel
Le 6116 a un tampon bidirectionnel de 8 K et un relais séquentiel numérique. Il commande aussi les interruptions.

Interface interne
Le HD61 commande les ports de MIDI, dirige les signaux d'entrée, et adresse la ROM et la RAM.

L'interface numérique d'instruments musicaux
L'interface MIDI rend compatible les protocoles d'entrée/sortie de l'ordinateur et des instruments qui y sont connectés — tout comme n'importe quelle interface —, ce qui permet aux instruments d'utiliser la mémoire de l'ordinateur. Elle traite aussi le son numérisé qu'on lui transmet, ajoutant une information de commande et de synchronisation à l'entrée du synthétiseur.

mélodie, *mi, sol,* etc., est entré sans paramètre de durée, toutes les notes de la mélodie continueront à être émises. Le résultat de cette procédure sera un accord soutenu formé des notes de la mélodie.

Mais, ce qui était censé être une mélodie semble devenir un accord. Pour éviter une telle erreur, il suffirait d'exécuter la séquence, mais en envoyant cette fois une instruction MONOPHONIQUE au synthétiseur. Le terme « monophonie » signifie simplement son unique, par opposition à sons multiples (polyphonie). Si le synthétiseur ne peut produire qu'un son à la fois lorsqu'il est réglé en mode MONOPHONIQUE, la séquence, piètrement entrée, peut alors être jouée sous la forme d'une succession de notes — une mélodie, et non un accord.

Supposons, qu'après quelques essais le nouvel utilisateur réussisse à entrer sa mélodie correctement et que le synthétiseur interfacé joue. Le rythme de la mélodie — défini par la séquence des durées — est correct, et la mélodie elle-même reste « dans les temps ». Nous devons souligner que, jusqu'ici, les types d'instructions ont été assez limités. Seuls deux paramètres musicaux ont été définis — la note et la durée.

Le compositeur écoute la mélodie plusieurs fois, puis décide qu'elle n'est pas assez douce — c'est normal avec ce niveau de définition. Au lieu de placer le *do* et le *mi* l'un après l'autre, la note *do* monte alors progressivement jusqu'au *mi*. Le terme glissendo désigne ce type de mouvement qui ressemble à ce que donnerait la mélodie si elle était sifflée par une personne. Dans cette situation, cela pourrait ajouter un élément supplémentaire de vivacité aux performances du synthétiseur. Cette instruction remplace donc maintenant l'instruction initiale du *do,* ajoutant 1 octet supplémentaire.

Si le synthétiseur récepteur ne peut produire de glissandos, il ne peut exécuter cette dernière instruction. Il se peut qu'il exécute le *do* comme s'il recevait l'instruction initiale, ou il peut faire tout autre chose.

Si les instructions d'un utilisateur de MIDI sont destinées à produire une section de musique polyphonique et si le synthétiseur récepteur n'est qu'un instrument monophonique, il fera sans doute une sélection aléatoire à partir de la polyphonie, et produira une exécution monophonique. Pour résumer, utiliser MIDI pour relier un micro-ordinateur à un synthétiseur peu évolué ne transformera pas ce dernier en un synthétiseur performant.

Ces remarques s'appliquent aussi dans l'autre sens. L'instrument récepteur peut être un superbe synthétiseur valant 100 000 F ; si des paramètres musicaux suffisants n'ont pas été définis et si les propres contrôles du synthétiseur n'ont pas été réglés correctement, la musicalité résultante sera comparable à celle que pourrait fournir une calculatrice de poche.

En pratique, cette deuxième situation est facilement corrigée. Le nombre maximal de paramètres est déterminé de manière constante en utilisant les boutons du synthétiseur, et les instructions MIDI doivent s'y intégrer.

Nous avons jusqu'ici traité les caractéristiques de note et de durée, mais MIDI comporte 128 commandes théoriques, couvrant le filtrage, la distorsion, le « bruit blanc » (toutes les fréquences possibles), et le « bruit rose » (fréquences moyennes), ayant des valeurs comprises entre 1 et 128. C'est suffisant pour gérer les paramètres offerts sur la plupart des synthétiseurs ; ces commandes intéresseront probablement les propriétaires de micro-ordinateurs.

Rôle de la vitesse de transmission

C'est ici que la vitesse de transmission de MIDI devient importante. Nous avons vu qu'une commande très simple, concernant une note et ne définissant que deux paramètres, utilisait trois mots d'affilée. A une vitesse de 31,25 kilobauds, cela prend presque une milliseconde. Des accords de six notes sont communs dans plusieurs types de musique : la transmission d'un tel accord prend 5,76 millisecondes. Si nous commençons à peaufiner cet accord à l'aide des commandes MIDI, le temps de transmission devient suffisamment lent pour que l'oreille humaine commence à détecter des changements dans les caractéristiques sonores, causés par ce retard.

Ces changements ne sont apparents que lorsque des sons, et notamment des sons similaires, se produisent simultanément. La musique, malheureusement, est un médium « parallèle » : en tant qu'auditeurs, nous sommes habitués à entendre les sons simultanément.

Il n'est donc pas surprenant que l'on ait critiqué la transmission série de MIDI ; une transmission parallèle aurait été plus appropriée. Reste à voir si les utilisateurs de MIDI sont gênés par ce défaut. Retenons que la conception de l'interface semble être seulement un compromis entre le coût et l'efficacité. L'avenir reste ouvert.

Registre par registre

Après l'explication générale, voyons les registres du 6809 en détail pour déterminer comment ils servent à stocker et à déplacer des données.

Nous avons vu qu'un registre est un emplacement mémoire à l'intérieur du processeur lui-même. Récapitulons.

• **Les registres d'index** servent à modifier les adresses que nous utilisons dans notre programme.

• **Les pointeurs de pile** sont employés par le processeur pour adresser une mémoire de travail, et peuvent également être utilisés par le programmeur pour stocker et retrouver rapidement des données.

• **Le compteur de programme** contient l'adresse de l'instruction suivante et peut être modifié par le programmeur pour transférer le contrôle, ce qui donne l'équivalent en langage d'assemblage d'une instruction GOTO.

• **Les accumulateurs** sont les registres les plus souvent employés, en particulier pour effectuer les fonctions arithmétiques.

• **Le registre de code de condition** contient des drapeaux représentant l'état du processeur (par exemple, si la dernière opération a donné un résultat nul); ces drapeaux peuvent être testés pour faire un choix ou une boucle, ce qui équivaut en langage d'assemblage à la structure IF...THEN.

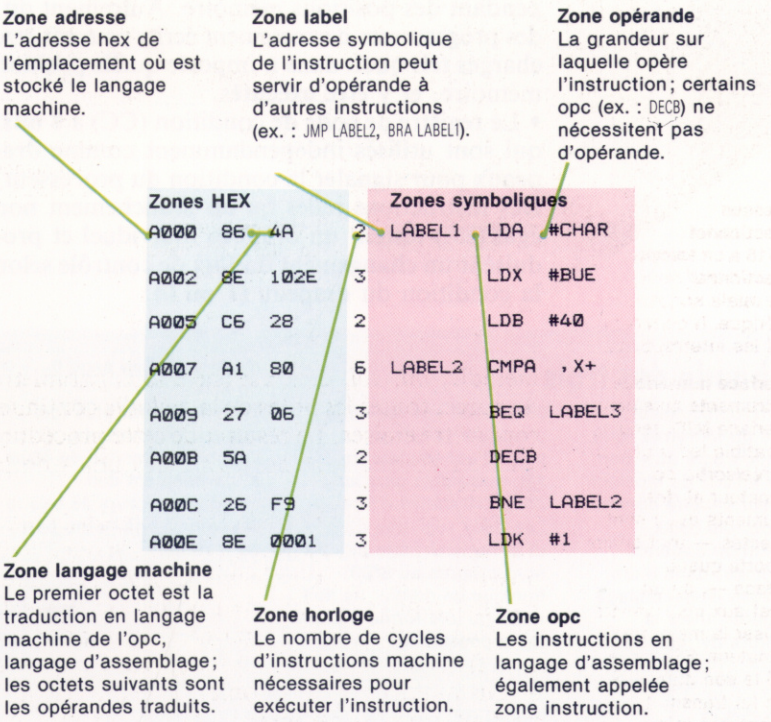
Le processeur 6809 contient tous ces registres. Comme il est dérivé du Motorola 6800 (de même que le 6502), il existe beaucoup de similitudes entre les langages d'assemblage utilisés sur les deux processeurs. Toutefois, ils ne sont pas compatibles. Cependant, il y a beaucoup de programmes 6800 qui sont compatibles en code source — un programme en langage d'assemblage écrit pour le 6800 peut être réassemblé pour le 6809 avec quelques chances de tourner. Mais même ce faible degré de compatibilité n'est pas réalisé avec le 6502 (ou son dérivé, le 6510, qui est utilisé sur le Commodore 64). Néanmoins, les similitudes entre processeurs impliquent que la traduction d'un programme en langage d'assemblage dans une version 6809 n'est pas trop difficile, et peut constituer une bonne introduction au 6809 pour ceux qui connaissent déjà le 6502. Voici les registres du 6809 :

• Deux accumulateurs 8 bits, A et B. Il n'y a pas de différence fonctionnelle entre eux lorsqu'ils servent de registres à 8 bits, de sorte qu'on peut les utiliser indifféremment. Le fait qu'il y en ait deux permet de garder des valeurs dans un accumulateur, tout en travaillant dans l'autre. Il est également possible de traiter ces deux accumulateurs comme un seul accumulateur 16 bits, ce qui permet au processeur d'effectuer des opérations arithmétiques à 16 bits. De ce fait, le 6809 est parfois considéré comme un pseudo-16 bits, mais ce n'est pas vraiment le cas, et il vaut mieux dire qu'il s'agit d'un 8 bits évolué. Lorsque les deux accumulateurs sont utilisés simultanément, on dit qu'ils forment le registre D à 16 bits.

• Il y a deux registres d'index, X et Y. Là encore, il n'existe pas de différence fonctionnelle entre eux, sauf que certaines instructions, utilisant le registre Y, se traduiront en instruction à 2 octets, contrairement à 1 octet pour l'instruction correspondante du registre X; le programme devient un peu plus long et lent. C'est pourquoi il est préférable d'utiliser X là où seul un registre d'index est nécessaire.

• Le 6809 a deux pointeurs de pile, S et U. Le processeur utilise S pour toutes ses opérations de pile. Bien que le programmeur soit libre d'utiliser S s'il le désire, il faut toujours s'assurer que le fonctionnement du processeur n'en est pas affecté; aussi vaut-il mieux utiliser U. La présence de deux pointeurs de pile fait du 6809 un processeur idéal à utiliser avec FORTH.

• L'utilisation des registres X, Y, S et U n'est pas restreinte à leur désignation : S et U peuvent tous deux servir de registres d'index, par exemple, et tous les quatre peuvent être employés pour stocker et manipuler des nombres à 16 bits.



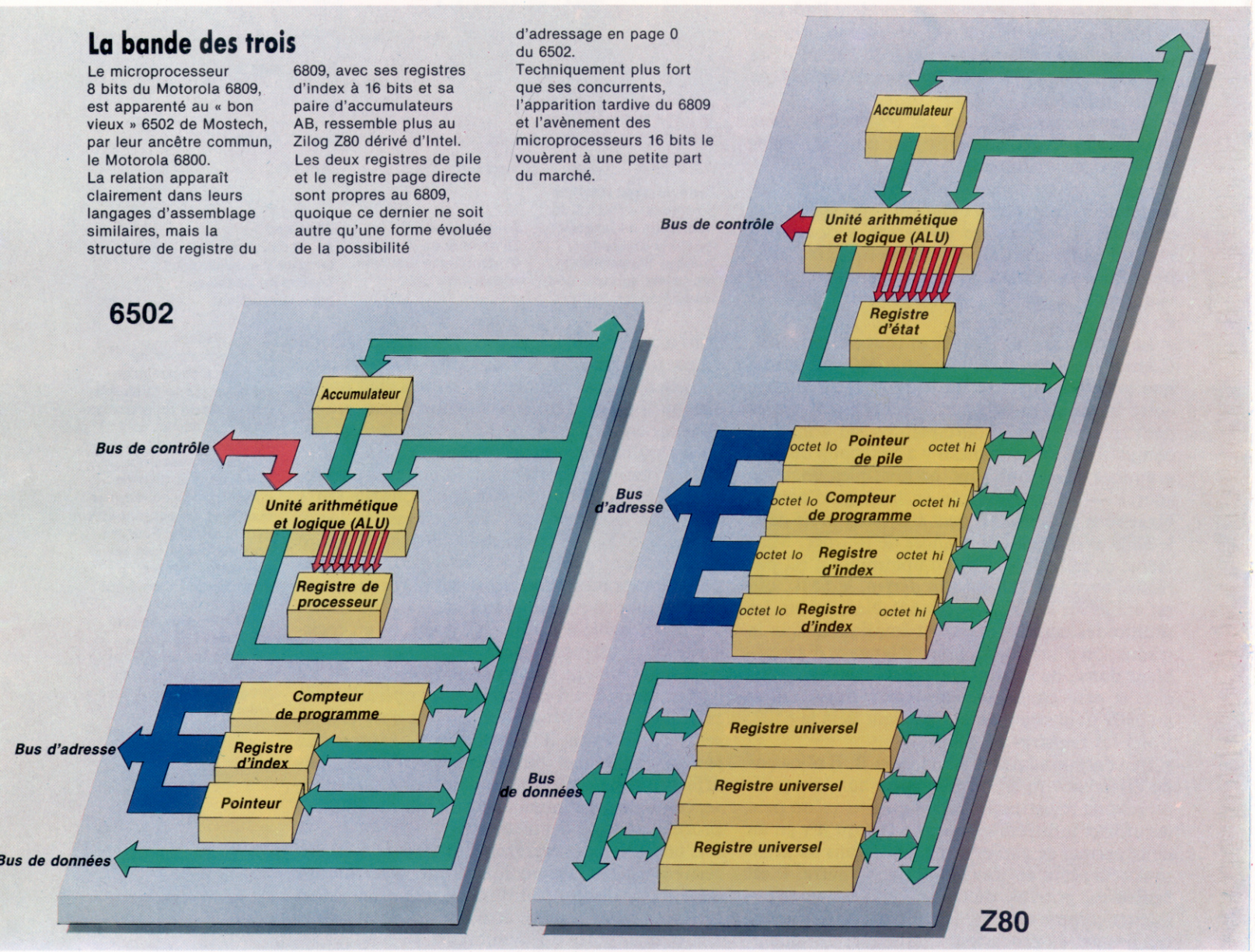
Zones d'étude
Les programmes en langage d'assemblage paraissent très différents des listes de programmes BASIC. Lorsqu'ils sont affichés par un programme assembleur (qui traduit le langage d'assemblage en langage machine), ils peuvent être déconcertants. Pour les comprendre, il faut se concentrer sur les colonnes (ou zones) qui vous intéressent (généralement, ce sont les zones label, opc et opérande) et ignorer le reste. Cet exemple de programme devrait vous y aider.

• Le compteur de programme (CP) est un registre à 16 bits qui est automatiquement ajusté par le processeur, de manière à indiquer l'instruction suivante. Les instructions de saut (JMP) et de branchement (BRA) modifient le contenu du compteur de programme, et le 6809 lui permet de servir de registre d'index. Nous allons examiner de plus près les effets de ce dernier ; mais en utilisant des adresses relatives au contenu du CP au lieu de spécifier une adresse absolue stockée en X et Y, on peut écrire un programme indépendant des positions mémoire. Autrement dit, des programmes correctement écrits peuvent être chargés tels quels dans n'importe quelle position mémoire, et y être exécutés.
• Le registre de code de condition (CC) a 8 bits, qui sont utilisés indépendamment comme drapeaux pour signaler la condition du processeur. Des instructions telles qu'un branchement non égal (BNE) testent un drapeau individuel et produisent un changement du flux de contrôle selon la condition du drapeau (1 ou 0).

Il y a plusieurs instructions qui servent simplement à entrer, sortir et échanger des données entre les différents registres ; pour les besoins des exemples suivants, supposons que nous ayons réservé un certain nombre d'emplacements mémoire en utilisant les directives d'assembleur FCB et FDB. Ce ne sont pas des instructions au processeur. Lorsque l'assembleur traduit le programme en langage d'assemblage en langage machine, il obéit immédiatement à ces directives et réserve les emplacements mémoire désirés pour le programme. Il ne traduira pas les directives en langage machine, car elles ne concernent pas le processeur. Comme elles ressemblent à des codes opérations (opc) de langage d'assemblage (tels que BNE ou JMP), on les appelle parfois pseudo-op ; « directives d'assembleur » serait un nom plus correct, car il explique leur fonction : diriger le fonctionnement du programme assembleur. Si ces directives sont munies d'un label, celui-ci sera traduit par l'assembleur dans l'adresse appropriée. Ainsi, nous obtenons :

La bande des trois

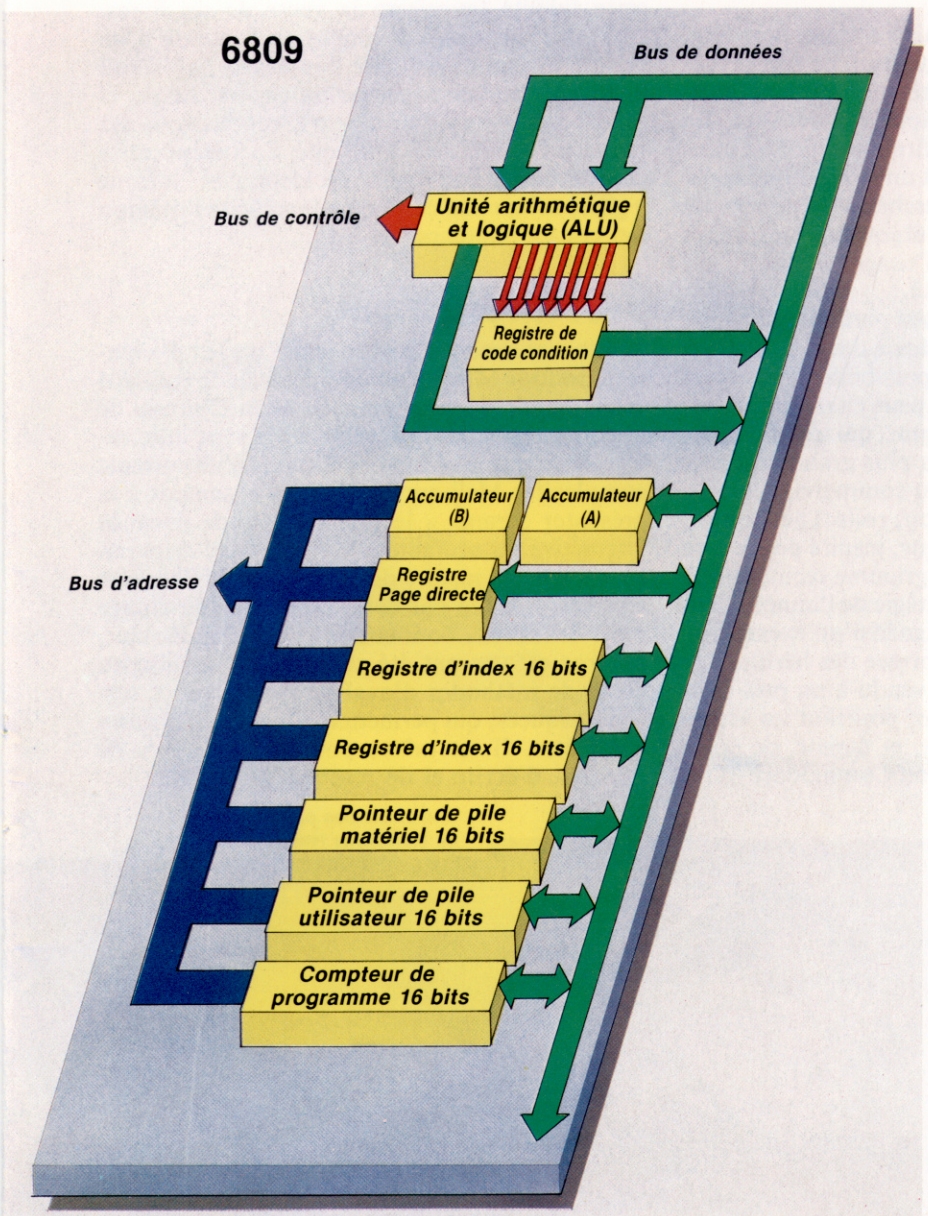
Le microprocesseur 8 bits du Motorola 6809, est apparenté au « bon vieux » 6502 de Mostech, par leur ancêtre commun, le Motorola 6800.
La relation apparaît clairement dans leurs langages d'assemblage similaires, mais la structure de registre du 6809, avec ses registres d'index à 16 bits et sa paire d'accumulateurs AB, ressemble plus au Zilog Z80 dérivé d'Intel.
Les deux registres de pile et le registre page directe sont propres au 6809, quoique ce dernier ne soit autre qu'une forme évoluée de la possibilité d'adressage en page 0 du 6502.
Techniquement plus fort que ses concurrents, l'apparition tardive du 6809 et l'avènement des microprocesseurs 16 bits le vouèrent à une petite part du marché.



NUM1 FCB 0 réserve un seul octet référencé par NUM1, de valeur initiale 0;
NUM 2 FCB 0 analogue au précédent;
NUM3 FDB#A93B réserve 2 octets pour le nombre à 16 bits #A93B (# est souvent utilisé par l'assembleur 6809 pour indiquer que le nombre est en notation hexadécimale).

Les instructions suivantes chargent les valeurs stockées dans différents registres :

LDA NUM1 charge le nombre à 8 bits stocké à l'emplacement mémoire représenté par NUM1 dans l'accumulateur A;
LDB NUM2 comme ci-dessus, charge NUM2 dans l'accumulateur B;
LDX NUM3
LDY NUM3
LDS NUM3
LDU NUM3
LDD NUM3
ces instructions chargent le nombre à 16 bits en NUM3 respectivement dans les registres X, Y, S, U et D.



Kevin Jones

De même, le contenu à 8 ou 16 bits d'un registre peut être stocké dans un emplacement mémoire à l'aide de :

```
STA NUM1
STB NUM2
STX NUM3
STY NUM3
STS NUM3
STU NUM3
STD NUM3
```

Notez que, si l'accumulateur est chargé à partir de NUM1, on peut effectivement copier NUM1 dans l'accumulateur sans le modifier; les opérations de stockage fonctionnent de la même façon.

Les contenus de deux registres peuvent être échangés (pourvu qu'ils soient de la même taille), à l'aide de l'instruction EXG. Par exemple :

EXG A,B échange les contenus de A et B;
EXG X,S échange les contenus de X et S.

Le contenu de l'un peut être transféré dans l'autre — par exemple : TFR Y,U copie le contenu de Y dans U. Les deux registres doivent encore être de la même taille.

Pour écrire un programme qui fasse effectivement quelque chose, introduisons l'instruction ADD, qui additionne le contenu d'un emplacement mémoire à celui de l'un des accumulateurs.

ADDA NUM1 signifie « additionner le contenu de NUM1 dans le registre A, et mettre le résultat de l'addition dans le registre A ».

D'abord, nous additionnons les deux nombres à 8 bits dans NUM1 et NUM2, en remettant le résultat dans NUM1 et en ignorant tout dépassement, si la somme est supérieure à un nombre à 8 bits. Ensuite, nous additionnons à nouveau les contenus des deux emplacements, mais cette fois en obtenant un résultat à 16 bits dans NUM3. Premier exemple :

```
LDA NUM1     copier d'abord le nombre en A;
ADDA NUM2    additionner le second nombre;
STA NUM1     stocker la réponse dans NUM1.
```

Second exemple :

```
LDB NUM1     copier d'abord le nombre en B;
SEX          convertir le nombre à 8 bits en B, dans un nombre à 16 bits en D;
STD NUM3     copier D dans NUM3;
LDB NUM2     copier le second nombre dans B;
SEX          le convertir en un nombre 16 bits dans D;
ADD NUM3     additionner le premier nombre à 16 bits à partir de NUM3 dans D;
STD NUM3     stocker la réponse en NUM3.
```

Image de marque

Melbourne House est surtout connue pour ses excellents jeux d'aventures, tels que The Hobbit ou Mugsy, au graphisme remarquable et à l'intrigue soigneusement agencée.

L'Australien Fred Milgrom fonda Melbourne House en 1977. Le lancement du ZX80 lui avait fait prendre conscience du vaste marché potentiel que représentait l'informatique individuelle. En 1980, la nouvelle compagnie publia ainsi un recueil de cinquante programmes destinés à cet appareil, tandis que sortait le tout premier jeu de Melbourne House, Space Invaders, également prévu pour le ZX80. Ses succès conduisirent l'entreprise à publier une série d'ouvrages consacrés à cette machine.

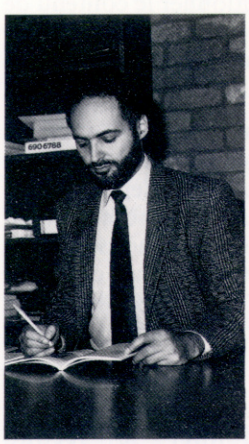
Le ZX81 fut commercialisé l'année suivante, et il cessa aussitôt d'être question de son prédécesseur; les ventes s'effondrèrent, sauf aux États-Unis, ce qui permit d'échapper au désastre. La leçon ne fut pas perdue; la firme comprit les bienfaits de la diversification, et un tout nouvel appareil apparaissant sur le marché se vit pourvu de manuels et de logiciels d'autant plus recherchés que les guides destinés à l'usager n'étaient pas tous de la meilleure qualité...

Melbourne House tira ainsi parti du phénoménal succès du Spectrum, grâce à des programmes qui utilisaient au mieux les possibilités graphiques et sonores de l'engin : on peut citer, par exemple, le jeu d'arcades Penetrator, qui se vendit particulièrement bien. Mais la plus grande réussite de la compagnie, au niveau commercial comme à celui de la programmation, reste The Hobbit, un jeu d'aventures graphique, inspiré de l'œuvre de J.R.R. Tolkien. Le programme reçut ainsi le prix du meilleur jeu de stratégie de l'année. Chaque cassette était accompagnée d'un exemplaire du livre, à la demande expresse des héritiers de l'auteur; le tout fut donc vendu à un prix trois fois supérieur à la normale; pourtant les ventes n'en souffrirent nullement, et depuis, de nombreuses versions, destinées au Commodore 64 ou à d'autres micro-ordinateurs qui se vendent bien, ont fait leur apparition.

Les programmeurs maison sont tous fixés à Melbourne, en Australie. Ils sont répartis en équipes de quatre membres, dont chacune s'occupe d'un aspect particulier du jeu. Celui-ci connaît donc une période de développement plus longue. Mais cette politique exigeante porte ses fruits, car le consommateur sait pouvoir compter sur un produit de qualité. Melbourne House espère que cette fidélité favorisera la vente de ses livres. Ceux-ci sont toujours conçus en fonction d'un certain niveau de maîtrise de l'usager, qu'il soit débutant, bon connaisseur ou expert. Livres et programmes sont, par ailleurs, vendus sous des jaquettes similaires, afin que l'acheteur reste convaincu que leur qualité est identique. La firme y glisse même une carte où elle sollicite l'opinion des consommateurs.

B.D. interactive

Elle a récemment mis en vente un jeu d'aventures intitulé Mugsy, où il faut garder le contrôle d'une bande de gangsters, en plein Chicago de la grande époque. Les graphismes sont admirables, bien que l'intrigue ne soit pas d'une grande complexité; Melbourne House n'hésite pas à la présenter comme « la première bande dessinée interactive sur ordinateur ». Vient aussi de paraître un logiciel inspiré des aventures de Sherlock Holmes, qui n'a pas demandé moins de quinze mois de travail. Tout comme pour The Hobbit, de gros efforts ont été consacrés à la mise au point de méthodes d'analyse des réponses très sophistiquées, qui permettent l'entrée de phrases complètes. Il est même possible, au cours de l'action, d'écrire et de recevoir des lettres!



Alfred Milgrom, codirecteur et principal éditeur *de Melbourne House.*

Philip Mitchell, auteur de The Hobbit et de Sherlock Holmes.

Un jeu, un livre
Melbourne vend avec la cassette de The Hobbit, programme passionnant, un exemplaire du livre de Tolkien — une remarquable idée de marketing.
(Cl. Ian McKinnell.)

Une partie des programmeurs de Melbourne House.