MIEUX PROGRAMMER SUR ORIC 1 ET ATMOS

Michel Archambault

SORACOM éditions

informatique

Michel Archambault

MIEUX PROGRAMMER SUR ORIC 1 ET ATMOS

SORACOM éditions

Du même auteur aux
EDITIONS ETSF

- Formation pratique à l'electronique de base
- Construisez vos appareils de mesure
- guide pratique des montages electroniques
- Labo photo - Montages

En préparation aux
Editions Soracom

- Bien programmer sur Amstrad

MIEUX PROGRAMMER SUR ORIC 1 ET ATMOS

Editions SORACOM - 1985
ISBN 2-904032-30-4

AVERTISSEMENT

Cet ouvrage prend la suite du manuel de l'ORIC ATMOS ; il suppose donc que le lecteur connaît déjà les fonctions BASIC de l'appareil. Il reste à savoir s'en servir avec *efficacité* ;après le permis de conduire, il s'agit de devenir un bon conducteur...

Nous mêlons continuellement deux choses. D'une part de bonnes méthodes éprouvées par les programmeurs chevronnés, et d'autre part une multitude de trucs, de tours de mains, qui sont eux très souvent spécifiques aux ORIC, et à l'ATMOS en particulier.

En effet, il n'y a pas deux micro-ordinateurs identiques, chacun a ses points forts et ses points faibles et aucun n'est parfait (même s'il fait quinze fois le prix de l'ATMOS !). Nous allons donc apprendre à nous appuyer sur des qualités afin de contourner des lacunes.

Parlant du perfectionnement de la programmation en BASIC, l'auteur a conscience qu'il s'adresse à des gens qui ont des idées de programmes, idées qui doivent se concrétiser d'une manière SOLIDE et RAPIDEMENT. Si votre idée est bonne, il ne faut plus en faire un programme boîteux truffé de fautes de débutant, mais un vrai LOGICIEL

qui « tienne la route », même s'il est court (ce qui vous incitera à vous lancer dans de plus longs). Ces « bonnes habitudes à prendre » vous amèneront à la qualité professionnelle ; d'ailleurs nous n'hésiterons pas à vous familiariser un peu avec le jargon.

Hormis les utilitaires du chapitre XII, qui complètent le BASIC ORIC, nous n'indiquons que des programmes très courts qui sont en somme des « passages » de démonstrations. Tous ont été testés, et ils sont représentés en 38 caractères par ligne, exactement comme vous le verrez sur votre écran (après avoir fait LIST). Notre principal souci a été de rester clair pour tous, car pour vous la micro-informatique est avant tout un LOISIR, donc pas de « cours magistraux » !

La table des matières risque de vous surprendre un peu, on parle beaucoup de l'esthétique des pages d'écran, des sécurités «anti-plantages » et pas un mot sur le langage machine ni sur la synthèse musicale.... Question de priorité :

Pour le langage machine, il faudrait y consacrer un énorme volume (et il en existe déjà) ; pour que la musique accompagne une image *qui bouge,* il faut du langage machine. Pour faire de la musique sans image (ou fixe), un instrument spécifique est meilleur marché et bien plus pratique qu'un micro-ordinateur... Alors qu'il y tant de belles choses à faire en BASIC !

Nota : Conçu à partir de l'ATMOS, cet ouvrage s'adresse aussi, disons à 90 %, aux possesseurs de l'ORIC 1.

Chapitre I

LA PROGRAMMATION STRUCTUREE

Disons tout de suite de quoi il s'agit, et pourquoi nous avons voulu commencer par cette technique.

Un programme « structuré » ou « modulaire », est pratiquement une suite de GOSUB, suivie d'un END et enfin tout un bloc de sous-programmes, appelés par ces GOSUB et se terminant chacun par un RETURN.

Cette configuration va à l'encontre des courts « programmes fleuves » que vous avez connus lors de votre initiation au BASIC. On appelle un programme « fleuve » lorsqu'il s'exécute comme il se lit, c'est-à-dire dans l'ordre des numéros de lignes. A présent vous voulez concevoir des programmes importants de votre cru, et la tendance naturelle va vous conduire à faire la même chose mais en beaucoup plus long... En ce cas, attendez-vous à bien des misères ! La plupart des programmeurs chevronnés ne s'y risqueraient pas... Pourquoi ?

Parce que l'on est pratiquement sûr d'aboutir à un programme truffé de GOTO vers le haut ou vers le bas, un enchevêtrement quasi inextricable. C'est ce qu'en jargon on appelle un programme « bouts-de-ficelle » ou encore programme « spaghetti ». C'est la faute classique du débutant, même si son idée de départ peut être qualifiée de géniale. Rappelez-vous toujours les deux grandes lois de la programmation (quel que soit le langage utilisé).

1 — UN PROGRAMME NE FONCTIONNE JAMAIS DU PREMIER COUP
2 — UN PROGRAMME PARFAIT N'EST JAMAIS DEFINITIF

(Il ne s'agit pas de boutades mais de réalités chaque jour confirmées).
Alors, comment trouver le « bug » subtil qui bloque ou perturbe un tel type de programme (BUG ou BOGUE = ERREUR) ? Même s'il parvient à fonctionner, comment y ajouter une autre routine ? Comment en recopier une partie pour un autre programme ?

En conclusion, un programme fleuve de 3 000 octets est déjà de la haute voltige, que l'auteur préfère laisser à quelques grands maîtres...

Après ce réquisitoire très partial, faisons l'éloge de la programmation structurée ou modulaire.

A lire les manuels d'initiation au BASIC, on a tendance à croire que l'on ne crée un sous-programme que si ce dernier va être appelé de nombreuses fois par des GOSUB éparpillés dans le programme. Certes, c'est là un cas évident mais en pratique peu fréquent. En fait nous préférons dire que l'on doit créer un sous-programme à chaque fois que l'on a affaire à une suite de lignes BASIC remplissant une fonction bien spéciale. Même s'il ne comporte qu'une seule ligne ! Même s'il ne sert qu'une seule fois ! Exemples : un module de calculs assez complexes, une page d'écran pour le titre, une suite de commandes destinée à l'édition sur imprimante (taille de caractère, tabulations, etc...), une courte séquence musicale, etc...

Voyons les avantages d'une telle pratique :

1) *Chaque module est indépendant,* vous pouvez le refaire entièrement en lui ajoutant une quinzaine de lignes si nécessaire.

2) A force d'améliorer le programme principal, vous décidez qu'il vaut mieux appeler ce module beaucoup plus tard ou beaucoup plus tôt, ou plusieurs fois (au lieu d'une comme prévu au départ). Pas de problème, il suffit, dans le programme principal, de déplacer ou d'ajouter des GOSUB à telle ligne... Et si ce module fait une dizaine de lignes, imaginez le grand chambardement qui s'ensuivrait dans un programme fleuve !

3) Vous venez de faire un programme structuré dans un domaine d'application qui vous est cher, et pour cela vous avez conçu une multitude de sous-programmes assez spécialisés ; soit un bloc qui dépasse les cent lignes, bien plus que le programme principal qui n'en fait qu'une vingtaine. Quelque temps plus tard, vous décidez de faire un autre programme qui touche le même domaine, mais ayant un rôle complètement différent. Voulez-vous parier que la plupart des modules du premier programme vont être utiles pour le second ?
Et vous n'aurez même pas la peine de les recopier ! Chargez l'ancien programme, conservez les modules, effacez le « programme principal », et tapez seulement le nouveau « principal ». Vous aurez certainement à créer de nouveaux modules, mais n'effacez pas ceux qui ne servent pas dans ce programme, car ce nouveau bloc de modules peut vous

être utile pour un futur programme numéro trois... (à moins de manquer de place mémoire).

Vous êtes ainsi en train de vous construire une « boîte à outils » car tous vos modules, une fois appelés par des GOSUB, sont comparables à de nouvelles fonctions BASIC très spécifiques à votre type de problème. Vous devenez ainsi très productif et avec des méthodes bien éprouvées. Se sentir rapide et efficace, ça donne des ailes !

4) Améliorons cela : enregistrez une cassette où il n'y aura uniquement que vos modules (prendre des numéros de lignes supérieurs à 10 000), exemple CSAVE ''PANOPLIE''.

Vous attaquez un nouveau programme, vous commencez par changer PANOPLIE en RAM, ou bien encore commencer à taper le programme principal puis faire :

```
CLOAD ''PANOPLIE'',J
```

et il viendra sa mettre à la suite. Nous verrons un cas très concrêt de « caisse à outils » avec les programmes du chapitre XII.

Quels sont les inconvénients de la programmation structurée ?

— Le risque à combattre est le manque de clarté du listing, car cette profusion de GOSUB dans le programme principal est assez pénible à lire. Pour éviter cela, ne soyez pas avare de REM, car l'ORIC 1 (48 k) et l'ATMOS ont une mémoire très confortable ; alors autant en profiter. Deux méthodes au choix :

• Au tout début du programme, prévoyez plusieurs lignes REM où vous légendez *tous* vos modules, par exemple :

```
50' Légende des GOSUB :
60' 10000 = TITRE
70' 11000 = ENTREES DES DONNEES
80' 12000 = CALCUL DE LA SURFACE
etc...
```

• On peut aussi mettre un REM au bout de la ligne où apparaît un GOSUB, exemple :

```
420 IF A=1 THEN GOSUB 15000:'EDITION
```

Trois recommandations :

La ligne départ d'un sous-programme est un multiple de 1000, et de préférence supérieure à 10000. Cette ligne ne comportera qu'un REM, le rôle de ce module.

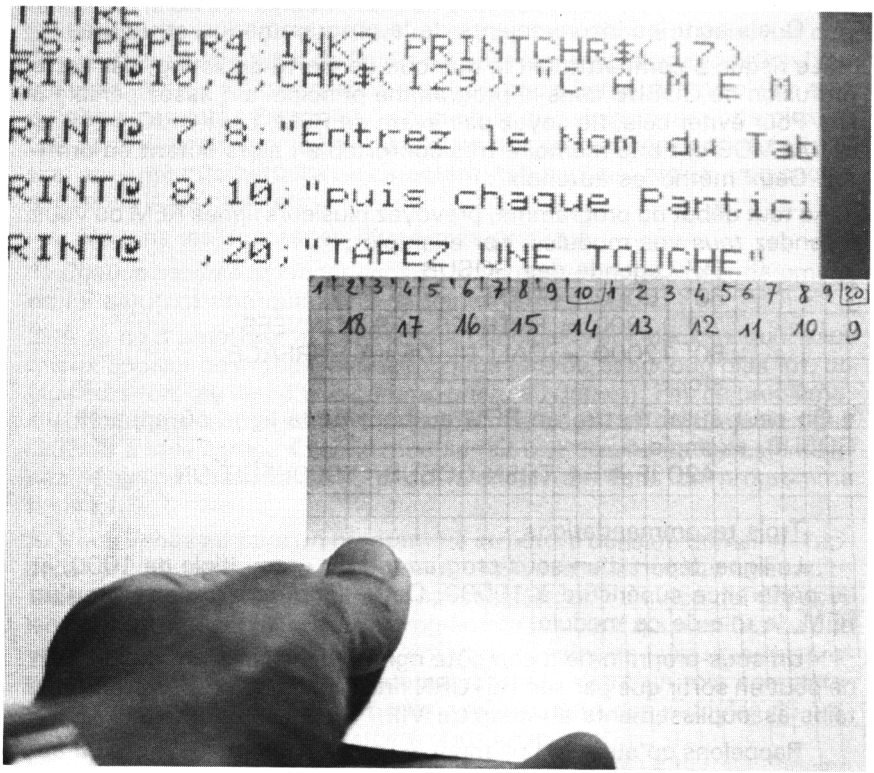
Un sous-programme a son côté rigide : une fois entré dedans, on ne peut en sortir que par son RETURN final. Bien que nous verrons certains assouplissements au chapitre VIII.

Rappelons qu'au sein d'un module, on a parfaitement le droit d'appeler par des GOSUB d'autres sous-programmes. De même, l'ordre, en numéros de lignes des divers sous-programmes n'a aucune importance.

Le temps que perd le micro-ordinateur à aller sur un module et à en revenir est quasi négligeable.

EN RESUME

La nécessité de la programmation structurée en BASIC apparaît à partir d'une certaine longueur du programme. Cette façon de concevoir un programme pourra vous être très utile si, par la suite, vous apprenez d'autres langages tels le FORTH ou le PASCAL où cela et obligatoire et non plus optionnel.



Cette réglette indique la tabulation d'une chaîne pour son centrage.

_ Chapitre II _

ORGANIGRAMME ET VARIABLES

C'est le plan détaillé, le schéma d'un programme. Certains auteurs parlent aussi d'« ordinogramme ». Ne pas confondre avec l'« algorythme » qui est un texte décrivant le ''scénario'' d'un programme.

On dit parfois qu'il faut toujours dessiner l'organigramme avant de taper la première ligne. C'est exagéré. Prenons le cas d'un bricoleur en menuiserie ; si c'est pour une étagère, il n'a pas besoin de plan ; en revanche, s'il veut réaliser une grande armoire, c'est quasi indispensable...

L'organigamme est un type de dessin qui à ses *normes,* tout comme le dessin industriel ou un circuit électrique ; mais heureusement, ces normes sont fort peu nombreuses, donc faciles à retenir.

Pour les problèmes complexes, la conception sur le papier d'un organigramme qui « tienne la route » est un jeu de piste passionnant : on s'aperçoit alors que l'idée de départ est irréalisable, mais qu'on arrive aux buts souhaités par des voies bien différentes, mais que quelques astuces vont simplifier. Après bien des feuilles froissées, on aboutit alors à un circuit correct que les diverses circonstances ne pourront prendre en défaut. A ce stade, on distingue nettement les divers modules qu'il faudra réaliser. Plus de la moitié du travail est déjà fait, il ne reste plus qu'à passer (enfin) au clavier. Cela ira très vite parce que les hésitations vont être très rares. Une anecdote professionnelle va vous le prouver :

Pour ce programme de calculs scientifiques, le cahier des charges remis faisait près de vingt pages. L'élaboration de l'organigramme a pris plus d'une semaine à plein temps ! Chaque soir, la corbeille à papier était bien remplie... A la fin, l'organigramme tenait sur une feuille 40 × 55 cm, et en écrivant petit ! Pour le taper en BASIC, trois jours ont suffit ; il faisait moins de 8 000 octets. Hormis la correction de trois ou quatre étourderies de syntaxe, il a fonctionné du premier coup.

Certes, vous devez penser que ce cas est extrême, mais trois faits sont à retenir de cet exemple : le temps au clavier a été la moitié du temps passé sur le papier. La longueur du programme BASIC semble modeste en regard du problème initial (nombreuses simplifications). En attaquant directement au clavier, il aurait eu peu de chances de réussite.

Tout ce long préambule était destiné à vous motiver, car nous savons qu'un débutant en BASIC n'éprouve vraiment aucun attrait pour ces petits carrés et losanges. Afin que nous parlions la même langue, il est essentiel que nous respections ces quelques normes :

LES NORMES DE DESSIN (voir figure II1)

1) On débute par le haut de la page. Le lancement du programme (RUN) est représenté par un cercle avec un trait tangent.

2) Une étape conditionnelle, c'est-à-dire un IF, est représentée par un losange, ou encore par un hexagone très allongé quand on a beaucoup de choses à inscrire dedans.

3) Le carré ou rectangle s'appelle un « pavé ». C'est le symbole passe-partout, on y inscrit ce qui s'y passe. En fait, il y a quelques petites variantes, pas toujours respectées :
— un pavé avec un plan coupé (trapèze ou coin ''écorné'') signale qu'il s'agit là d'une édition, aussi bien sur imprimante qu'à l'écran. Par exemple, la page d'écran montrant le résultat final, ou donnant des indications ;
— un pavé aux angles arrondis signifie normalement le début du programme ; en fait cette configuration est souvent utilisée pour représenter une étape où on entre des données (INPUT).

3) Losanges et pavés sont reliés par des traits (le ''trajet'' du logiciel). N'oubliez pas les flèches sur ces traits, car certaines remontent (GOTO...).

Un organigramme est fait pour être clair, et tout d'abord avec soi-même. Aussi, soyez très succinct pour vos inscriptions dans un pavé. Par exemple, écrivez « Calcul de la moyenne M », mais pas la formule ! Ne mettez pas des choses différentes dans le même pavé, du genre « Entrée de A et calcul de M », mais faites deux pavés différents reliés par un trait. En effet, qui vous dit que par la suite vous n'allez pas faire venir un trait fléché entre ces deux pavés ?

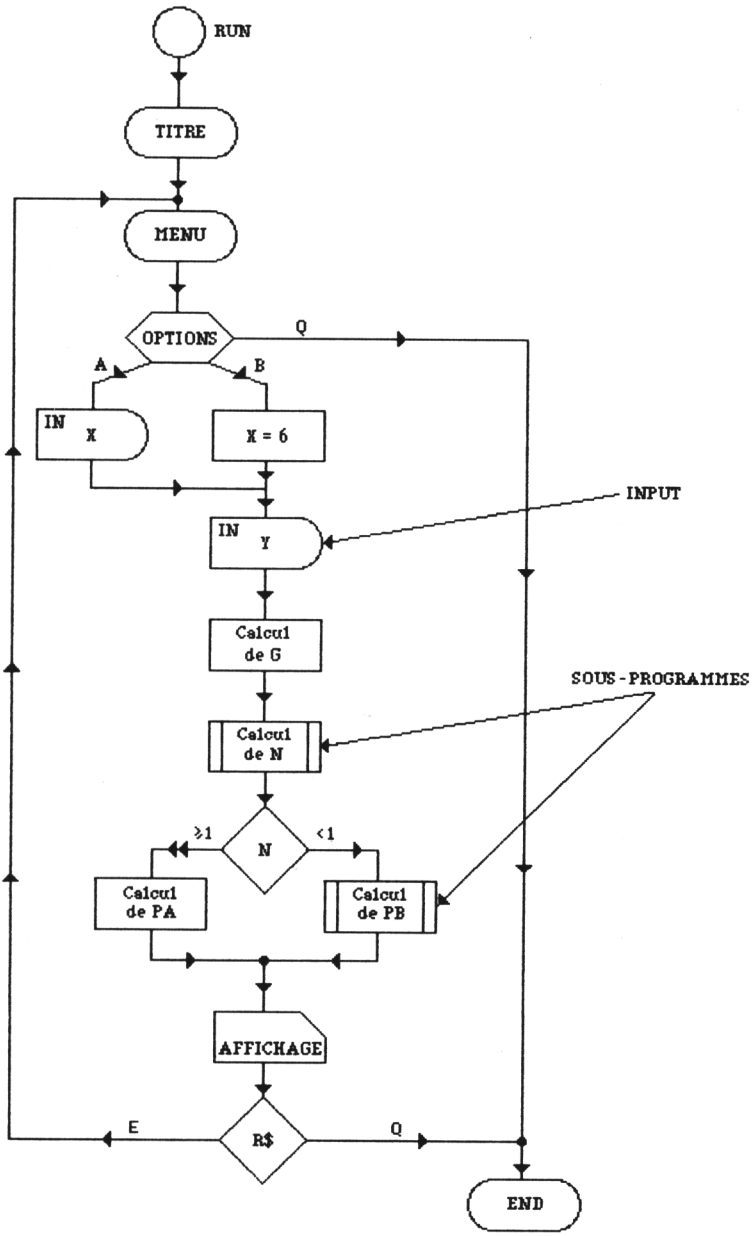

RUN
TITRE
MENU
OPTIONS
Q
A
B
IN X
X = 6
INPUT
IN Y
Calcul de G
SOUS-PROGRAMMES
Calcul de N
≥1
N
<1
Calcul de PA
Calcul de PB
AFFICHAGE
E
R$
Q
END

Figure II 1

Petit conseil pratique : inutile de vous appliquer à faire de jolis traits lors du premier tracé, car il a neuf chances sur dix pour aller à la corbeille... Commencez plutôt avec crayon et gomme.

Vous allez alors vous attaquer au contenu de certains pavés qui vous semblent complexes. Il est possible que vous deviez concevoir un mini organigramme pour quelques-uns, mais sur une feuille séparée. En cas de doute de fonctionnement BASIC concernant certains détails dont vous n'êtes pas sûr, il est conseillé de les vérifier à l'écran, soit en mode direct, ou par quelques lignes (deux ou trois) de mini-programme-test. Il arrive parfois que l'on ait ainsi une surprise de taille (bonne ou mauvaise), pouvant conduire à une modification importante dans l'organigramme principal !

Nous commençons à entrer dans le détail et nous baptisons des variables. Et là, il faut de l'ordre car le programmeur risque ensuite de s'y perdre ; le piège est classique, avec risques de « bugs fantômes » très difficiles à débusquer.

LES NOMS DE VARIABLES

Dans un ORIC, un nom de variables ne peut comporter que *deux* caractères, puis un troisième caractère du genre ($, %) ; c'est d'ailleurs le cas de très nombreux autres micro-ordinateurs.

Ainsi, des variables comme REFERENCE, REVENU et RETARD seront toutes trois tronquées en RE ! De même, PRENOM$ et PRISE$ sont confondues en PR$. Le premier caractère doit être une lettre majuscule, le deuxième caractère étant au choix, une lettre majuscule ou un chiffre, ou absent.

Exemples :

NB, N4, N, AC$, A3$, A$ sont admis.
Nb, nB, 4B, N* sont refusés.

Si de nombreuses variables ont un lien commun, il est pratique de les *indicer*, exemple NB(14), alors que sans indice, vous êtes limité à 10 valeurs, exemple de N0 à N9. L'indiçage des variables a d'autres avantages : l'indice peut être paramétré, c'est-à-dire défini par un calcul, exemple NB(I*2−1). S'il s'agit de *constantes,* à définir en début de programme, il serait fastidieux d'écrire et de les taper une à une avec le signe " = ", surtout s'il y en a une trentaine. On les entre sous forme de variables indicées dans une boucle FOR-NEXT contenant la fonction READ, avec ailleurs des lignes de DATA où les variables sont écrites (sans " " pour les chaînes) à la « queue leu leu » (voir chapitre VII).

Lorsqu'une variable désigne un nombre entier (= non décimal), tel un comptage, une progression de boucle FOR-NEXT, un numéro de ligne, etc..., nous vous demandons d'utiliser une lettre comprise dans l'alphabet entre I et N (I, J, K, L, M, N) ; I et N étant les premières lettres du mot INTEGER (= entier). Il s'agit là d'une convention d'écriture

héritée de langages plus anciens, mais toujours utilisés (FORTRAN notamment).

En deuxième caractère il est déconseillé d'utiliser les lettres I et surtout O qui peuvent être confondues avec 1 et 0.

Sur ORIC, l'instruction LET est inutile. Une variable numérique non encore définie est égale à zéro, une variable chaîne a une longueur nulle. Exemple :

PRINT WX → 0 : PRINT LEN (Z$) → 0

Le suffixe % implique que la valeur de la variable est entière ; exemple, entrons A% = 5.75 puis faisons PRINT A% → 5. Pourvu que ce nombre soit compris entre 32767 et −32767, on a le même effet que si l'on avait écrit A = INT(5.75) (et c'est plus court...). Le grand avantage des nombres entiers définis par % est qu'ils n'occupent que deux octets au lieu de six ; ainsi B = 1 ou D = INT(6.92) ou E = −982785.5752 occupent chacun six octets, alors que F% = 1 ou G% = −16312 n'en prennent que deux chacun ! Mentionnons deux bizarreries typiques aux ORIC : les calculs avec des valeurs entières (%) sont un peu plus longs qu'avec des nombres « réels » à six octets. Si vous tapez FOR I% = 1 TO 1000:NEXT → "Syntax Error", même si on prédéfinit 1 et 1000 comme étant des entiers par A% = 1:B% = 1000. Même refus si on fait FOR I = A% TO B%:NEXT.

Conclusion : Si vous utilisez des variables entières, faites quelques vérifications de fonctionnement...

LA LISTE DES VARIABLES

A mesure que vous concevez votre programme, inscrivez sur une feuille la signification de chaque nouvelle variable introduite. Sinon vous risquez facilement d'utiliser deux fois les mêmes caractères pour deux variables complètement différentes (ne souriez pas, cette étourderie est un grand classique). Le pire c'est que le programme semblera fonctionner, et vous ne vous douterez pas qu'il fournit des résultats faux ! Non seulement vous tenez cette liste à jour, mais pour les longs programmes, inscrivez aussi le ou les numéros de lignes où chaque variable apparaît.

Sur l'ORIC 1 48 k et sur l'ATMOS, on dispose d'une mémoire disponible assez confortable, alors profitez-en pour reporter ces légendes dans le programme sous forme de REM ; ou bien un bloc de lignes REM en début du listing, ou des REM en bout de ligne où telle variable apparaît pour la première fois, et n'oubliez pas de faire de même pour les tableaux DIM.
Une feuille de papier peut s'égarer, pas de REM...

Feuille de l'organigramme, liste des variables, nombreuses instructions REM, vous voyez que l'on n'oublie pas la seconde "loi" (« un programme parfait n'est jamais définitif »). En effet, après de nombreux

mois, il est banal que l'on revienne sur un programme pourtant déjà bien rodé. Motifs : petites améliorations, actualisations, adjonctions de nouvelles options (ou suppression), ou bien encore on s'inspire fortement d'un ancien programme pour en construire un autre traitant un sujet différent. Or, quand on lit un listing « vieux » de trois mois seulement, on se surprend à se demander « mais pourquoi ai-je fait comme ça ? D'où sort ce flag ? Où voulais-je en venir ? » Ce n'est plus du BASIC, c'est du « chinois » !

NE FAITES JAMAIS CONFIANCE EN VOTRE MEMOIRE. Ce serait la troisième loi.

LA MISE AU NET

Le programme était relativement court, de ce fait vous n'aviez pas fait d'organigramme préalable. Alors faites-le maintenant, à chaud. Idem pour la liste des variables.

Le programme était long et vous aviez fait un organigramme. Redessinez-le, mais proprement et sans ratures. Mieux, ajoutez au-dessus de chaque pavé *le numéro de ligne* où il commence. Ce petit raffinement sera par la suite super utile, et peut-être plus tôt que vous ne le pensez ! Pourquoi ? Parce qu'il est bon d'ajouter, çà et là quelques petites sécurités qui éviteront au programme de « planter » suite à un concours de circonstances rares mais possibles, beaucoup de lignes genre IF... THEN GOTO..., et là vous serez heureux de ne pas avoir à chercher la ligne de destination parmi plusieurs centaines...

Si vous disposez d'une imprimante (quasi obligatoire pour les programmes très longs), faites un listing ''final'', et en haut, inscrivez en gros caractères le nom du programme, puis *la date* de ce listing (très important...) ainsi que le nombre d'octets qu'il occupe. Pour cela faites RESET puis PRINT 37629 – FRE(0) (pour l'ATMOS). Pourquoi ? Par sage précaution vous aviez fait plusieurs sauvegardes sur des cassettes différentes. Toutes n'ont peut-être pas été mises à jour ? Quelque temps plus tard, vous pouvez vous demander « est-ce que c'est le bon enregistrement ? » Chargez-le et demandez-lui son nombre d'octets : s'il y a une différence...
Autre cas : c'est le bon programme, mais vous venez de commettre une fausse manœuvre du genre taper un nombre puis la touche RETURN. « Ai-je effacé une ligne ? ». RESET et demande d'octets occupés. Ce nombre sera aussi à inscrire (au crayon) sur la cassette, et tenez-le à jour...

Pourquoi ce RESET préalable ? Parce que si le programme a été lancé, il y a déjà les DIM et des variables qui ont pris leur place ; en faisant RESET, il ne reste que le programme en RAM.

_Chapitre III _

LA NUMEROTATION RATIONNELLE DES LIGNES

Nous venons de voir quelques bonnes habitudes à prendre pour éviter de perdre bêtement du temps en maintes occasions. Il va en être de même avec la numérotation des lignes, domaine qui de prime abord semble bien dérisoire et secondaire, mais qui est en fait une base pour des astuces bien pratiques.

Tout d'abord sachez que le pas entre des lignes successives n'influe pas sur la vitesse d'exécution, et voici pourquoi.

Si dans la RAM on décortique octet par octet le contenu d'une ligne, que trouvons-nous ? Tout d'abord l'adresse (en hexadécimal sur deux octets) de la *ligne suivante,* puis sur deux octets le numéro de la ligne, puis des numéros de codes qui correspondent à ce que contient la ligne, enfin 00 qui signifie fin de la ligne de programme.

Revenons à cette adresse de la ligne suivante, c'est en fait celle qui suit immédiatement celle où se trouve le 00 final. Donc, avant de commencer à exécuter la première ligne, l'ordinateur sait déjà où se trouvera la deuxième ligne, qui commencera bien sûr par l'adresse de la troisième et le numéro de la deuxième ligne, numéro dont il n'a que faire (sauf s'il y a des GOTO ou GOSUB) !

En conclusion, que vous numérotiez vos lignes dans l'ordre 1, 2, 3, 4 ou 100, 500, 1024, 4215 ne changera strictement rien à la vitesse d'exécution du programme.

Par habitude, on numérote avec un pas de 10, mais en faisant un grand saut pour passer à la rubrique suivante, celle-ci commençant par un numéro multiple de 100 ou de 1000.
Exemple :
Le premier stade de votre programme (premier pavé de l'organigramme) va de la ligne 10 à la ligne 90 ; pour le deuxième stade, commencez à 200 ou à 300. Non seulement le listing est plus clair, mais cela vous laisse un espace de sécurité pour ajouter des lignes au premier stade.

Pour les numéros de lignes, vous ne serez jamais en manque puisque vous pouvez aller jusqu'à 63999 ! Alors, de grâce, ne soyez pas radin, "aérez" votre numérotation, c'est gratuit, ça ne gaspille aucune place et, répétons-le, ça ne retarde pas...

QUELQUES NORMES

1) Débutez un programme à la ligne 10 : c'est un REM qui indique le nom du programme et la date de la *dernière modification.* En 20, le nom de l'auteur, etc... Réservez les lignes jusqu'à 50 (environ) pour des REM ultérieurs.

2) Le programme proprement dit commence en 100, mais c'est encore un REM donnant le nom du "pavé".

3) Si vous avez un menu principal auquel le programme retourne souvent, mettez-le en 1000. Ainsi, en cas d'ennui, de fausse manœuvre, vous savez que sur chacun de vos programmes on pourra se tirer d'affaire en tapant CTRL C puis GOTO 1000.

4) Ne commencez vos sous-programmes (GOSUB) qu'à partir de 5000 ou au-delà, et de 1000 en 1000 (pas de GOSUB 6730...). Bien que l'ordre importe peu, il est plus clair de les classer par genres. Exemple série dix mille = pages d'écran, série vingt mille = calculs divers, etc...

5) La première ligne d'un module est son titre en REM, même si après il n'y a qu'une seule ligne.

6) Une dérogation : vous désirez ajouter un sous-programme qui est une variante du module commençant en 12000 : il sera plus clair de le mettre en 12500 qu'en 18000.

7) Pour insérer une ligne entre deux au pas de 10, mettez-la en 5. S'il y a deux lignes à ajouter, mettez-les en 3 et 7. Ne mettez jamais deux numéros qui se suivent.

8) N'utilisez pas les lignes au-delà de 40000 ou 50000 (sur ATMOS), car elles peuvent être utilisées pour ajouter un sous-programme utilitaire chargé par CLOAD" ",J (exemple pratique au chapitre XII).

9) Pour insérer une ligne *provisoire* lors de la mise au point du programme, utilisez un numéro se terminant par 9. Exemple entre 620 et 630 :

629 PRINT I:END

Même en faisant LIST à l'écran, un tel numéro se fait remarquer, et de ce fait, on ne peut oublier de l'effacer après usage. En revanche, si vous aviez utilisé 625....

10) Une ligne de séparation se termine par 99 ; soit une ligne avant le début du module suivant. Exemple :

9999 END:'.....

Ainsi, quelle que soit la longueur du programme principal, on est sûr qu'il ne risque pas d'enchaîner en final sur le premier sous-programme commençant en 10000... Autre cas :

12999'.....

pour séparer les modules commençant en 12000 et 13000 (listing plus lisible).

11) Après la dernière ligne tapée, entrez une ligne REM le signalant. Exemple courant :

63999' ... FIN DE LISTING ...

C'est utile si vous avez une imprimante ou si vous faites publier votre programme dans une revue. On est ainsi certain que la fin n'a pas été coupée...

Une petite explication concernant la recommandation (plutôt que norme) n° 1, à savoir titre et date en ligne 10 : 10 et 1000 (menu) sont les deux seuls numéros de lignes à connaître par cœur (facile ?). On vient de modifier une ou plusieurs lignes, on fait ensuite EDIT 10 pour modifier la date. S'il y a plusieurs personnes autorisées à modifier tel programme, la date sera suivie par les initiales de celui ayant fait la dernière modification.

Le nom figurant en ligne 10 doit être celui que l'on utilise après CSAVE. Si vous voulez un titre plus explicite, mettez-le en REM en ligne 20 ou 30.

D'autre part, en laissant libres les numéros de zéro à neuf, on peut, par la suite, y loger 10 lignes de REM pleines de caractères ''bidon'', soit un espace de 800 octets réservé dans la zone RAM programme pour y placer une routine en langage machine.

EFFACEMENT D'UNE SERIE DE LIGNES

Au BASIC de l'ORIC il manque la fonction DELETE. Heureusement, l'éditeur ORIC (que l'on critique souvent à tort), permet d'aller plus vite qu'en entrant chaque numéro au clavier puis RETURN.

1) Listez à l'écran la zone de programme à effacer (Ex.: LIST 200-300).

2) Amenez le curseur en haut, face au premier numéro.

3) Placez le majeur de la main gauche sur la touche CTRL, l'index gau-

che sur la touche A et l'index droit sur la touche RETURN. La touche CTRL peut rester enfoncée en permanence.

4) Passez en CTRL A les chiffres (seulement) du numéro de ligne, puis RETURN, et ainsi de suite. C'est très rapide !

Deux cas particuliers :

— Une ligne de programme tient sur deux lignes d'écran : quand le curseur arrive sur cette deuxième ligne, tapez RETURN, il vient alors se placer au numéro suivant.

— Vous avez fait une fausse manœuvre, c'est-à-dire que vous avez fait CTRL A sur ce qu'il ne fallait pas. Ne faites surtout pas RETURN ! Mais avec l'index droit tapez sur la touche X, car CTRL X annule ce que vous avez chargé dans le « buffer clavier » (mémoire clavier). Cette action va laisser un signe "\" (« anti-slash ») sur la ligne d'écran ; si c'est une ligne que vous désirez conserver, remonter le curseur par une touche flèche avant de faire CTRL X.

RENUMEROTATION DE LIGNES

Vous avez tellement fait de modifications de lignes qu'une zone du programme n'est vraiment pas belle à voir ; des pas de 1, 2, 3, 5, 20... Heureusement que vous avez laissé de la place avec la rubrique suivante pour les numérotations au pas de 10. Renuméroter une quinzaine de lignes sur l'écran, c'est facile : nouveau numéro et CTRL A sur toute la ligne. Mais attention, les anciens numéros ne se terminant pas par zéro ne vont pas être effacés, d'où des lignes en double exemplaire qu'il faut absolument effacer ! Afin d'éviter cette pagaille, nous vous conseillons d'opérer ainsi :

1) Lister à l'écran la zone de lignes à traiter.

2) Effacez-les comme expliqué ci-dessus par CTRL A sur les numéros (les lignes restent néanmoins écrites sur l'écran...).

3) Remontez le curseur, et en face de chaque ligne tapez le nouveau numéro, puis faites CTRL *sur le reste de la ligne.*

Deux cas particuliers :

a) Il y a des GOTO dans certaines lignes. Après listage à l'écran, écrire sur une feuille la liste des numéros actuels avec, en face, leurs futurs numéros. C'est facile et rapide. Puis renumérotez comme il vient d'être expliqué, mais en passant en CTRL A sur la ligne ; changez au passage les destinations des GOTO en consultant votre liste de conversion.

b) La zone de programe à traiter ne tient pas sur l'écran : même méthode, rédigez sur papier la "table de conversion". Ensuite, attention ! Il faut *commencer par la fin* de la zone. Un exemple : vous devez renuméroter les lignes 300 à 450 qui vont devenir 300 à 520. Faites LIST 382-450 et opérez normalement, puis attaquez la première moitié en faisant LIST 300-380. Et n'oubliez pas les GOTO au passage.

Vous voyez qu'avec méthode et réflexion, on peut se tirer d'affaire rapidement.

Deuxième méthode bien plus rapide et élégante si vous disposez d'un ATMOS et non d'un ORIC 1 ; utiliser un sous-programme qui va faire tout ce travail en moins d'une seconde.

Ce sous-programme utilitaire a été préalablement enregistré sur une cassette à conserver soigneusement. Ses numéros de lignes commencent à 56000, on peut *l'ajouter* au programme en cours en le chargeant par CLOAD '' '',J (décrit au chapitre XII). Ceci fait, tapez (en mode direct) :

GOSUB 56000 (et RETURN)

En bas de l'écran apparaît alors un petit questionnaire qui vous demande la première ligne à traiter, quel numéro elle va prendre, la dernière ligne à traiter (donnez bien une ligne existante !) et le pas désiré. Une seconde de travail. Là non plus, les GOTO et GOSUB ne sont pas corrigés. Vous voulez la liste de conversion ? Tapez GOSUB 56500, et elle apparaît à l'écran. Vous préférez cette liste sur votre imprimante, alors tapez plutôt GOSUB 56600.

Vous trouverez le listing de ce sous-programme RENUM au chapitre XII. Nous nous sommes inspirés du programme publié à la page 73 du manuel de l'ATMOS (où il est expliqué en détail), mais en améliorant sur de nombreux points, notamment en lui faisant charger en DIM les anciens et nouveaux numéros de lignes.

Vous allez pouvoir l'utiliser autant de fois que nécessaire au cours de votre programmation, mais attention à la définition de DIM à son début ! Pour éviter qu'à la deuxième renumérotation l'écran vous réponde ''Redim Array Error'', pensez à tout réinitialiser en faisant RESET avant GOSUB 56000.

Quand votre programme sera terminé, vous pourrez effacer notre RENUM avant d'enregistrer la version finale de votre logiciel.

_ Chapitre IV_

MODIFIONS LE PROGRAMME

Dans ce chapitre, nous allons surtout apprendre à jongler avec l'éditeur ORIC (éditeur = dispositif ROM pour modifier des lignes de programmes). Dans le chapitre précédent, nous en avons eu un avant-goût avec les astuces pour effacer et renuméroter des lignes, mais on peut aller beaucoup plus loin.

Lorsque l'on est habitué à d'autres micro-ordinateurs, il est normal que l'on soit surpris et dérouté par l'éditeur de l'ORIC ; il est vraiment très spécial, en particulier pour insérer des caractères dans un mot ou une ligne. Au début, l'auteur a lui aussi pesté contre l'absence d'une touche INSERT, puis une fois bien rodé, on s'aperçoit que ce mode d'édition est en fait super-puissant car il permet des acrobaties fort pratiques qui sont irréalisables avec des éditeurs dits « classiques ».

Première constation, il serait barbare d'essayer d'apprendre une série de modes opératoire, mieux vaut comprendre comment ça marche, ensuite tout ''coule de source''.

LE PRINCIPE DE L'EDITEUR ORIC

Contrairement aux autres éditeurs qui lisent la « mémoire écran » (la ligne où se trouve le curseur), l'éditeur ORIC ne connaît qu'une seule chose, le BUFFER CLAVIER. Cette mémoire contient au maximum 79

caractères, au 80e (après avoir prévenu par des PING), il se vide complètement. S'il y a moins de 80 caractères, une action sur la touche RETURN va transférer tout son contenu dans la RAM. En cas d'erreur, on peut vider le buffer clavier (annulation) par CTRL X.

La touche DEL (de DELETE = effacer) efface dans le buffer caractère par caractère entré en commençant par le dernier, puis l'avant-dernier, etc...

Les touches de flèches et CTRL ne font rien entrer dans le buffer, ni sortir (sauf CTRL X déjà vu, et bien sûr CTRL A).

Le tout est de se souvenir de ce que l'on vient d'EMPILER dans le buffer avant de faire RETURN. On charge le buffer soit par CTRL A, soit directement par les touches de *caractères* (lettres, chiffres, blanc, signes). A partir de là, tous les coups sont permis. En voici quelques exemples que vous ne soupçonniez peut-être pas.

1) Permutation de deux lignes

CTRL A sur le premier *numéro.* Descendre par flèche, et CTRL A sur le texte de la 2e ligne. Return. CTRL A sur le 2e numéro puis sur le texte de la première. Return.

2) Scinder une ligne en deux lignes

CTRL A sur la 1re instruction. Return. Monter ou descendre ou à gauche par flèche. Taper au clavier le numéro suivant, puis CTRL A sur la 2e moitié de la ligne. Return.

3) ASSEMBLER DEUX LIGNES COURTES EN UNE LONGUE

CTRL A sur la 1re ligne. Taper ":" au clavier. CTRL A sur *seulement le texte* de la 2e ligne et Return.

A noter que nous n'avons parlé que sur deux lignes, qui peuvent être "distantes", et pourquoi pas opérer en « piochage » sur trois lignes ? Car des fois on quitte une ligne en cours d'écriture pour aller (par les flèches) recopier un passage difficile déjà écrit sur une ligne précédente. Autrement dit, outre la copie totale du texte d'une ligne, on fait aussi de la copie partielle ; et l'auteur ne s'en prive pas...

LES TECHNIQUES DE L'INSERTION

On n'appelle une ligne par EDIT que lorsque la ou les insertions sont importantes parce que la ligne d'écran est dégagée. Mais, quand il y a plusieurs lignes voisines à modifier, nous préférons notre méthode "maison" d'édition plein écran.

1) On liste le passage du programme à traiter, par exemple LIST 500-515.

2) Il faut commencer par le haut, c'est évident, mais voilà qu'on arrive

sur une ligne oú l'on doit faire une insertion importante. Arrivé au bon endroit, par CTRL A, on monte d'une ligne par la flèche et là *on tape CTRL N,* la ligne de travail s'efface (on n'en avait plus besoin) et on peut alors y écrire en toute clarté. Après retour par flèches au caractère départ « deuxième partie », on termine la ligne en CTRL A et Return. Donc chaque ligne à modifier va conduire à l'effacement de la ligne d'écran au-dessus. Si cela vous contrarie parce que cette ligne supérieure est en fait le *début* de la ligne de programme que vous êtes en train de traiter, il suffit tout bêtement de monter de *deux* lignes par la flèche, au lieu d'une...

Si le début de l'insertion se situe vers la droite de l'écran, repérez bien l'endroit d'où vous quittez vers le haut, puis déplacez le curseur assez loin à gauche par la flèche.

3) Dans la plupart des cas, il s'agit d'insérer un seul caractère oublié, et le plus souvent un blanc (espace). Pour aller très vite, examinez le bas de votre clavier et vous constaterez que toutes les touches à manœuvrer sont *à la file dans le sens droite-gauche ;* flèche haut, barre d'espacement, flèche bas et flèche gauche, vous pouvez alors poursuivre en CTRL A. On tape ces quatre touches à la volée sans regarder l'écran. Vous avez oublié un caractère quelconque, même manœuvre mais en remplaçant la barre d'espacement par le caractère en question, et toujours à la volée sans regarder l'écran.

Après avor modifié une pleine page d'écran, il est quand même prudent de faire CTRL L (très important) et de relister le passage.

RECUPERATION D'UNE ZONE DE PROGRAMME

Pour votre nouveau programme, vous aimeriez bien récupérer une suite de lignes très fastidieuses à taper (exemple certaines lignes de DATA), figurant dans un autre programme. Chargez cet ancien programme. Lister à l'écran le passage convoité (il faut qu'il tienne sur une page d'écran). Taper NEW. Repasser les lignes en CTRL A (on peut alors modifier les numéros de ces lignes). Le fameux passage est en RAM.

A ce stade, deux orientations possibles : vous complétez ce nouveau programme au clavier, ou bien vous sauvegardez ce passage sur cassette afin de le réinjecter par CLOAD'' '',J sur un programme déjà en RAM (sur ATMOS).

Après tous ces exemples, vous admettrez que l'éditeur par buffer clavier et CTRL A est en fait un outil super puissant pour celui qui a bien compris le principe.

SUBSTITUTIONS DE LIGNES

Vous êtes confronté au problème suivant :
Dans votre programme, les lignes 310, 320, 330 et 340 ont une écriture longue et complexe et ne vous donnent pas entièrement satisfac-

tion (par exemple exécution trop lente). Vous avez une autre idée, mais vous ne savez pas si elle sera meilleure que la première. Comment remplacer ces lignes tout en les gardant en secours ? C'est tout simple. Il suffit de déplacer leurs numéros dans une zone ''tranquille'', par exemple entre l'instruction END et le début des sous-programmes. Lister ces lignes et tapez 9 devant chacune d'elles, et CTRL A. Elles sont recopiées en 9310, 9320, 9330 et 9340 ; pour les remettre en place, ce sera facile, effacement et retour aux anciens numéros.

LES PIEGES DE LA LONGUEUR DE LIGNE PROGRAMME

Lorsque vous tapez une nouvelle ligne de programme contenant l'instruction PRINT, vous utilisez le point d'interrogation. Pour le buffer clavier chaque ''?'' est remplacé par ''PRINT'', soit cinq caractères. Faites le petit essai suivant :
60?:?:? etc... jusqu'à remplir deux lignes d'écran (jusqu'aux deuxième PING) et Return. A présent, tapez EDIT 60. La ligne 60 tient alors six lignes d'écran plus trois caractères ! Et elle est parfaitement exécutable par RUN 60.

Voilà pourquoi il peut vous arriver d'être dans l'impossibilité de repasser en CTRL A une ligne programme devenue bien trop longue suite à de nombreux ''?''. Impossible ? Pas du tout ! En repassant en CTRL A, à chaque fois que le curseur se place sur le ''P'' d'un PRINT, tapez un ''?'' par dessus, puis passez R, I, N et T avec la flèche droite, et reprenez en CTRL A... Le buffer n'a chargé que le ''?'' et vous voilà sauvé.

Mettons les choses au pire. A cette ligne surchargée, vous avez à insérer d'autres caractères. Deux solutions : ou bien vous scindez cette ligne en deux (nous avons vu comment), c'est ce qu'il y a de plus raisonnable, ou bien vous supprimez quelques caractères non indispensables, à savoir les blancs (espaces) et certains '';'' qui se mettent entre les CHR$. Exemple :

```
? CHR$(27);''J'';CHR$(4)
```

(attention, cela ne marche pas toujours...).

LES SAUVEGARDES DE TRAVAIL

De temps à autre vous prenez la très sage précaution de sauver votre programme en cours d'écriture, mais, de grâce, n'écrasez pas l'ancien enregistrement ! Sinon, imaginez une brève coupure de courant en cours d'enregistrement. Que vous resterait-il ? RIEN.

Si votre magnéto-cassette n'a pas de compteur, utilisez alternativement deux cassettes. S'il a un compteur, enregistrez une fois à 10, une fois à 50.

Autre astuce utile : dans l'ordre CSAVE, ne répétez pas le futur nom de votre œuvre, mais plutôt la date et l'heure. Exemple :

CSAVE''27-21H30''

Ainsi, en reprenant votre programmation quelques jours plus tard, vous verrez, en faisant CLOAD '' '', quelle est la version la plus récente (un nom de programme peut commencer par des chiffres).

Autre cas : votre programme fonctionne mais « boîte » un peu, et vous décidez une ou plusieurs importantes modifications. Archivage ! Si vous avez une imprimante, listez ce programme. Enregistrez-le à l'autre bout de la cassette (début face B) et notez-le sur le listing d'imprimante pour ne pas l'oublier. Alors là seulement, vous pourrez commencer le ''charcutage'' du programme. Vous avez fait des boulerversements considérables, mais hélas cette idée n° 2 ne fonctionne guère mieux que l'idée n°1. Archivez de même cette idée n° 2. Alors vient l'idée n° 3, très différente de la précédente, tellement qu'il vaut mieux partir du listing du programme n° 1. Heureusement qu'il avait été archivé ! Si l'idée n° 3 vous donne pleine satisfaction, vous pourrez alors détruire les archives n° 1 et 2. Ce petit scénario n'est pas de la fiction, c'est une histoire banale parce que courante.

La petite morale de toutes ces précautions est que la dernière idée en date n'est pas forcément meilleure que la précédente.

Note : On cogite beaucoup plus efficacement sur un listing d'imprimante que sur des listings à l'écran.
Pour un long travail de retouches à l'écran, commencez par taper ''PAPER2'' (vert), vos yeux vous en remercieront...

_ Chapitre V _

L'ESTHETIQUE DES PAGES D'ECRAN

Il y a deux sortes de programmes : les petits programmes d'essais afin de « voir si ça marche », et les... LOGICIELS, c'est-à-dire quelque chose dont on va se resservir, qu'on risque aussi de montrer à des amis. Pas de débraillé, vos programmes méritent mieux que cela. Un bon programme sans présentation, c'est comme un bon vin dans un gobelet en plastique, ça gâche tout...

Qu'est-ce qu'un programme avec une présentation minimum ? Tout d'abord un titre, et bien centré. Puis des explications (exemples : les règles de ce jeu, le mode opératoire), donc des phrases, elles aussi bien centrées, des lignes espacées et agréablement réparties sur la hauteur de l'écran, et non plus tassées en haut !

Ensuite il y a les couleurs, et pas à tort et à travers : vous avez sans doute déjà remarqué que certaines combinaisons PAPER/INK sont insoutenables, ou illisibles.

Commençons par ce qui semble le plus difficile : le centrage horizontal d'un mot ou d'une phrase. Deux cas se présentent :
— On ne connaît pas la longueur de la chaîne. Il faudra un centrage automatique.
— On connaît la chaîne à afficher (un titre). On mesure sa longueur sur l'écran avec une réglette spéciale (fabriquée par vous), et on inscrit sa position dans la ligne de programme.

LE CENTRAGE AUTOMATIQUE

La longueur de l'écran comprend 40 colonnes, donc la longueur de la marge à gauche est égale à la moitié des colonnes non utilisées par la chaîne. Commençons par le cas général où l'on positionne le départ en largeur et en hauteur. Essayez ce très court programme (pour ORIC-1 utilisez PLOT en ligne 50).

```
10 'CENTRAGE AUTOMATIQUE D'UNE CHAINE
20 CLS
30 PRINT@2,2;:INPUT"MOT OU PHRASE";A$
:CLS
40 A=LEN(A$)
50 PRINT@(40-A)/2,10;A$
60 GOTO 30
90 END
```

Vous constaterez que cette méthode est infaillible. A noter que l'on peut inclure la ligne 40 dans la formule de calcul de la ligne 50. Nous ne l'avons pas fait afin d'être plus clairs.

Autre cas : vous ne voulez pas fixer la position verticale, on va faire précéder notre chaîne par un nombre calculé de blancs, et ce par la fonction SPC. Essayez ceci :

```
100  CENTRAGE PAR SPC CALCULE
110 CLS
120 PRINT:INPUT"MOT OU PHRASE";A$
130 A=LEN(A$)
140 PRINT:PRINT SPC((37-A)/2);A$
150 GOTO 120
190 END
```

La grande différence entre les lignes 50 et 140 est que l'origine est maintenant 37 et non 40, parce que l'affichage d'écran commence sur la colonne n° 2 (la troisième).

D'autre part, vous remarquerez qu'ici la fonction PRINT commence *en début* de ligne et non en cours de ligne comme précédemment, donc il y a *effacement* de ce qui aurait pu se trouver à gauche du mot. Mais il y a encore une autre différence au cas où le PRINT serait suivi d'un attribut de couleur papier, par exemple, ESC Q (bande rouge) : dans

le premier programme, cette bande rouge partirait du premier caractère, alors que dans le dernier programme, la bande colorée partirait en début de ligne !

Note : Si le problème est de centrer une chaîne non pas sur l'écran mais sur une feuille d'imprimante, il est recommandé d'utiliser le positionnement par SPC plutôt que par les instructions spécifiques de votre imprimante ; c'est beaucoup plus fiable...

Enfin, une troisième méthode, c'est par la fonction TAB (pour ATMOS, déconseillée pour ORIC-1).

```
200 'CENTRAGE PAR TAB
210 CLS
220 PRINT:INPUT"MOT OU PHRASE";A$
230 A=LEN(A$)
240 PRINT:PRINT TAB((40-A)/2);A$
250 GOTO 220
290 END
```

C'est le même programme que le précédent, mais en remplaçant SPC par TAB et 37 par 40. On n'y efface pas le début de ligne.

LE CENTRAGE MESURE

C'est de loin le cas le plus fréquent quand on rédige un programme. Rien n'est plus fastidieux que de compter les caractères, soustraire à 40 et diviser par deux. Il est tellement plus simple de se bricoler une petite réglette en cartoline (Bristol) sur laquelle on lira immédiatement le positionnement horizontal ! Il est impensable de la trouver dans le commerce car l'écartement des graduations dépend uniquement de votre téléviseur.
La longueur de cette réglette sera environ égale à la largeur de la fenêtre sur écran. Son bord supérieur est gradué de 1 à 40 (=LEN), et au-dessous l'échelle de tabulation horizontale qui en résulte, de 18 à 2 (=TAB).

Voici comment la fabriquer en quelques instants :
— à mi-hauteur de l'écran, tapez sur une seule ligne la suite de chiffres 1 à 9, 0 à 9, etc... Donc de 1 à 38 ;
— placer l'angle gauche de la réglette sous le 1, puis avec un crayon recopiez chaque chiffre sur la réglette (sans la bouger...). Il est évident que le premier zéro correspond à 10, le deuxième à 20 et le troisième à 30. C'est l'échelle LEN ;
— on n'a plus besoin de l'écran. Sous certaines valeurs de l'échelle LEN, vous allez inscrire les nombres suivants (TAB) :

Haut	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
Bas	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3

Voici maintenant comment s'en servir :
— en tapant votre ligne de programme, il faut laisser deux blancs à l'emplacement du positionnement horizontal. Puis, vous complétez la ligne avec la chaîne à centrer ;
— placer la réglette sous cette chaîne en disposant le chiffre 1 de l'échelle LEN (l'angle supérieur gauche) exactement au-dessous du premier caractère de la chaîne. Sous le dernier caractère, on lit sa longueur LEN, on s'en moque, et au-dessous le positionnement pour son centrage (échelle TAB) ;
— faites RETURN, remontez le curseur et repassez la ligne en CTRL A, mais en remplaçant les deux blancs par la tabulation lue sur la réglette. RETURN; terminé.

Trois cas particuliers :

a) LEN est impair, et vous tombez entre deux valeurs ; prenez la plus petite, c'est arbitraire, mais si vous prenez une fois la plus grande, une fois la plus petite, l'effet s'en ressentira...

b) Cas très fréquent, la chaîne continue sur la ligne du dessous : on utilise enfin l'échelle LEN. Sur la première ligne, repérez cette valeur, par exemple 14. Sur la deuxième ligne, on prend la suite, donc en plaçant LEN = 15 sous le premier caractère. A l'extrémité de la chaîne, on peut alors lire la tabulation à effectuer.

c) L'ordre PRINT est suivi d'un ou plusieurs attributs : il faut savoir que chaque attribut provoque un *décalage* de la chaîne de une case. Vous serez peut-être surpris en essayant ces quelques lignes :

```
300 'DECALAGE PAR LES ATTRIBUTS
310 A$="ORIC-ATMOS"
320 CLS
330 PRINT@15,10;A$
340 PRINT@15,11;CHR$(27)"Q";A$
350 PRINT@15,12;CHR$(27)"Q"CHR$(27)"F
";A$
390 END
```

Donc, à la tabulation lue, il faut soustraire une unité pour attribut.

LA REPARTITION VERTICALE

C'est l'harmonieuse disposition des lignes sur la hauteur de l'écran. Deux règles primaires :
– n'utilisez jamais la ligne n° zéro qui est tout en haut de l'écran, c'est laid et peu lisible ;
– n'entrez pas vos lignes par des PRINT, c'est lourd et souvent aléatoire (SCROLLING), mais positionnez vos lignes par PRINT@ au PLOT. Sauf pour les pages d'imprimante où les lignes seront espacées par des LPRINT.

L'indice de positionnement vertical peut être compris entre 0 et 25, soit 26 lignes. La ligne n° 26 (à ras de la fenêtre) n'est pas accessible en mode programmé.

Le mini-programme qui suit va être très instructif sur de nombreux points. Il commence par repérer les numéros de lignes, il illustre la double hauteur et espacement double, il donne un exemple d'un encerclement et attend la frappe d'une touche en évitant ainsi le « SCROLLING » par le « READY » :

```
 400 'REPARTITION VERTICALE
 410 CLS: FOR V=0 TO 25:PRINT@2,V;V:NE
XT
 420 PRINT@10,7;CHR$(4);CHR$(27)"J";"O
 R I C * A T M O S";CHR$(4)
 430 PRINT@ 6,16;"* C'EST SOUPLE A PRO
GRAMMER *"
 440 L$="**********************************
"
 450 E$="*                                *
"
 460 PRINT@6,14;L$
 470 PRINT@6,15;E$
 480 PRINT@6,17;E$
 485 PRINT@6,18;L$
 490 PRINT@6,18;L$
 500 PRINT@12,24;"Tapez une Touche";:G
ET R$
 590 END
```

LA DOUBLE HAUTEUR (ligne 420)

ESC J n'est pas suffisant, il faut initialiser par CHR$(4) (= CTRL D), et le répéter en final pour l'annuler.
Nous l'avons positionné sur la ligne n° 7, numéro *impair* (c'est obligatoire !), et nous constatons qu'il occupe les lignes 7 et 8 (et non 6 et 7...).

Dans la ligne 420, nous avons laissé un blanc entre chaque lettre du titre car si la hauteur de chaque caractère est doublée, sa largeur reste la même. C'est une question d'esthétique, à moins que vous ne préfériez le style Bernard BUFFET...

L'ENTOURAGE

Nous avons préparé deux chaînes L$ et E$ (lignes 440 et 435) qui ont la même longueur. De même, le texte à encercler a lui aussi la même longueur, mais commence et se termine comme E$.

LA SUPPRESSION DE « READY »

Nous gardons l'écran en suspend par la fonction GET. En ligne 500, le « ; » place le curseur clignotant après le mot « touche ». Appuyez maintenant sur une touche : le mot « READY » se place en n° 25. Tapez une ligne 499 WAIT 500 et refaites RUN. A présent supprimons ce disgracieux curseur clignotant par CTRL Q. Ajoutez donc une ligne

498 PRINT CHR$(17)

et observez les différences.

Dans la pratique, on n'utilisera cette ligne 500 que lorsque l'on désire que la suite soit commandée par l'opérateur ; en 510 on aurait, par exemple, un CLS puis une autre page d'écran.
Pensez à rétablir le curseur clignotant, si nécessaire et tout au moins en fin de programme, par un second CTRL Q.

Il y a une autre méthode, plus radicale, pour supprimer l'affichage du « READY », c'est de faire POKE 26,96. Pour le rétablir, faire POKE 26,76.

LE CHOIX DES COULEURS

Il faut savoir combiner des considérations purement artistiques et d'autres purement techniques : si avec certaines combinaisons PAPER/INK les couleurs « bavent » l'une sur l'autre, rendant ainsi le texte peu lisible, il ne faut accuser ni le micro-ordinateur, ni le téléviseur, mais vous-même. En effet, la reproduction des couleurs sur un écran vidéo est assez spéciale, et vouloir l'ignorer conduit parfois à des déceptions...

LA VIDEO COULEUR

Lorsque l'écran représente du blanc, examinez avec une loupe la surface de celui-ci. On ne voit que des petits rectangles rouges, verts et bleus, également lumineux. Vue de loin, notre rétine fait une « synthèse additive » et n'y voit que du blanc. Le blanc n'est pas une

couleur, mais le mélange « équitable » de toutes les couleurs. Dans les tubes couleur modernes, il s'agirait plutôt de lignes verticales en « pointillés » rouge, vert, bleu, rouge, vert, bleu, etc... (voir figure V1). Remarquez qu'après chaque « triplet » horizontal, il y a un décalage d'une demi-hauteur avec le triplet suivant. Gardez votre loupe et faites un écran rouge (PAPER 1). On observe alors une ligne verticale rouge suivie de *deux lignes noires.* Idem pour les deux autres couleurs « primaires » vert et bleu. Par contre, avec les trois autres couleurs, nous aurons deux lignes colorées pour une seule ligne noire. Résumons-nous :

n°	couleur	combinaisons
0	noir	—
1	rouge	R
2	vert	V
3	jaune	R+V
4	bleu	B
5	magenta	R+B
6	cyan	V+B
7	blanc	R+V+B

Si on fait PAPER6:INK1 (ou l'inverse), nous avons chaque ligne verticale éclairée séparément ou par PAPER ou par INK. Il en résulte une excellente netteté verticale, mais une sorte de flou sur les motifs horizontaux ; flou dû aux décalages verticaux d'un triplet à l'autre.

A présent faisons PAPER6:INK5 (ou l'inverse), les petits rectangles bleus vont être éclairés à la fois par PAPER et INK : la netteté horizontale est améliorée alors que la netteté verticale est un peu moins bonne. C'est un compromis dont l'effet est préférable au précédent.

Faisons maintenant une combinaison PAPER/INK avec deux couleurs primaires seulement, par exemple, PAPER4:INK2. Les lignes de rectangles rouges sont donc toutes en noir, et l'effet n'est pas très joli car nos lettres présentent un liseré noir à gauche et en bas.

Et avec une seule couleur ? Faisons PAPER0:INK2. Les lettres vertes sur fond noir sont nettes, mais ces traits verts *étroits* apparaissent comme piquetés de noir, alors que si on fait l'inverse (PAPER2:INK0), les lettres noires paraissent nettes et homogènes, donc nettes, et le piquetage noir du fond vert n'apparaît plus !

Revenons au fond noir. Avec des lettres en couleurs primaires (B,V,R,), vous aurez des caractères « sales » mais nets. Avec des lettres en couleurs composées (jaune, magenta et cyan), vous aurez des caractères plus « brillants » mais avec un flou sur les traits horizontaux, surtout en magenta !
En revanche, sur fond blanc, toutes les couleurs donnent des caractères nets.

Nous avons voulu montrer l'explication technique de ces quelques nettetés bizarres, et pour vous rassurer, sachez que la presse technique est unanime pour louer l'excellente netteté des couleurs obtenues par l'ORIC-1 et l'ATMOS.

Il y a cinquante-six combinaisons PAPER/INK ($8 \times 7 = 56$), et il est difficile d'indiquer une formule qui ne donnerait que des combinaisons agréables, tant les cas particuliers sont nombreux.

Voyons à présent le côté « artistique ». Il y a des couleurs « chaudes » (jaune, rouge, blanc) et des couleurs « froides » (cyan, bleu, noir), des teintes claires (jaune, cyan, blanc, vert) et des foncées (noir, bleu, rouge, magenta). S'il s'agit de texte, il faut du contraste pour une meilleure lisibilité, soit en opposant chaude/froide ou claire/foncée, ou encore chaude et claire/froide et foncée. En revanche, un tel contraste est pénible, trop « criard » dans des dessins en HIRES où les surfaces en INK sont bien plus importantes. Un exemple : un texte en rouge/magenta est peu lisible, mais mis côte-à-côte, c'est agréable. Essayez ESC Q (bande rouge) et au-dessus ESC U (bande magenta). Même remarque pour vert/jaune).

Note : Le manuel ATMOS, page 313, comporte une erreur dans la légende des broches de la prise RVB : borne n° 4 lire « SYNCHRO » et non « CYAN ».

Pour afficher un attribut en mode direct, par exemple ESC Q, il faut d'abord enfoncer *seule* la touche ESC, et après la lettre (pas simultanément !).

UTILISONS LES ATTRIBUTS

Grâce à eux nous allons pouvoir égayer nos pages d'écran avec plusieurs couleurs de fonds et de lettres simultanément, voire même plusieurs couleurs de lettres sur une même ligne (beaucoup ont écrit que c'était impossible, nous montrerons le contraire) !

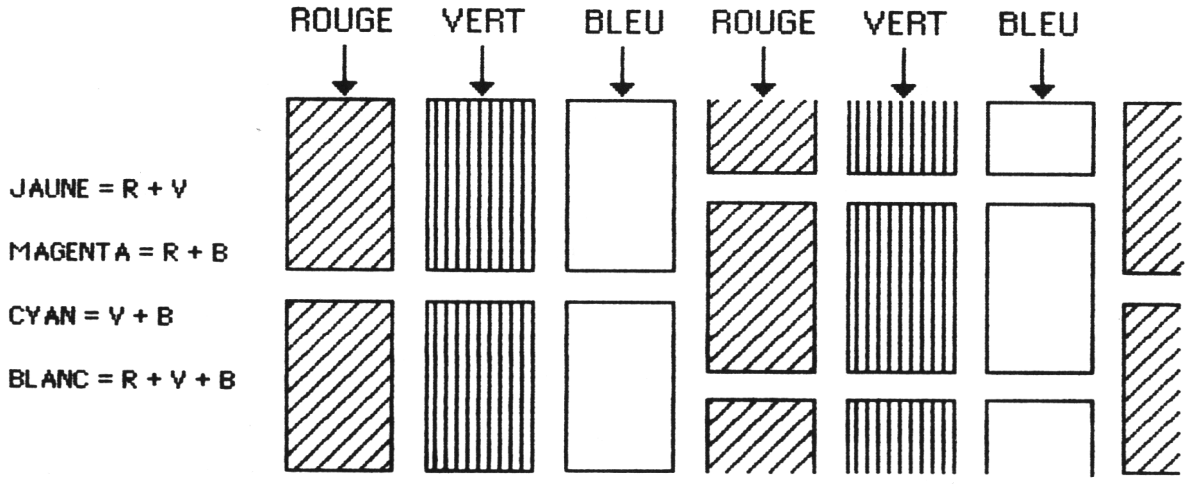
Le manuel d'origine ORIC n'est pas très clair sur les différentes façons de coder ces attributs, aussi nous préférons les résumer dans le tableau V1.

Pour obtenir l'encre rouge sur une seule ligne, nous avons le choix entre trois écritures (à la suite d'un ordre PRINT positionné ou non). CHR$/27)''A''; ou CHR$(27);CHR$(65); ou CHR$ (129);.
Dans ces écritures, 27, 65 et 129 sont respectivement les codes ASCII de ESC, A et attribut bande rouge. Essayez le petit programme suivant, et remarquez l'effet de sa seconde partie (la ligne 650). Nous avons utilisé le troisième mode d'écriture des attributs : avouez qu'il est plus rapide à écrire !

```
600 'ATTRIBUTS DE PAPIERS
610 CLS:PRINT:PRINT
620 FOR I=144 TO 151
630 PRINT CHR$(I):NEXT
640 FOR I=144 TO 151
650 PRINT CHR$(I);:NEXT
690 END
```

CHRS(27)" " (ou CHR$())		EFFET sur la ligne	Code global
@	64	Encre noire	128
A	65	Encre rouge	129
B	66	Encre verte	130
C	67	Encre jaune	131
D	68	Encre bleue	132
E	69	Encre magenta	133
F	70	Encre cyan	134
G	71	Encre blanche	135
H	72	Texte standard (lettres)	136
I	73	Texte 1/2 graphique	137
J	74	Hauteur double en lettres	138
K	75	Hauteur double en 1/2 graphique	139
L	76	Clignotant en lettres	140
M	77	Clignotant en 1/2 graph.	141
N	78	Clignotant haut. double lettres	142
O	79	Clignotant haut. double 1/2 graph.	143
P	80	Papier noir	144
Q	81	Papier rouge	145
R	82	Papier vert	146
S	83	Papier jaune	147
T	84	Papier bleu	148
U	85	Papier magenta	149
V	86	Papier cyan	150
W	87	Papier blanc	151

Tableau V1 : Les trois manières de coder un attribut de ligne.



La surface d'un écran TV couleur vue à la loupe

Figure V 1

Abordons les attributs d'encre. Entrez ce petit programme et regardez l'effet des lignes 740 à 770. Chaque lettre est d'une couleur différente mais elles sont séparées par un espace, lequel est provoqué par chaque attribut.

```
 700 'ATTRIBUTS D'ENCRES
 710 CLS:A$="ORIC C'EST SUPER !"
 720 PRINT:PRINT:FOR I=128 TO 135
 730 PRINT TAB(10);CHR$(I);A$:PRINT:NE
XT
 740 B$="*ATMOS*"
 750 FOR J=1 TO LEN(B$):C$=MID$(B$,J,1
)
 760 PRINT TAB(12);CHR$(127+J);C$;
 770 NEXT
 790 END
```

Et maintenant combinons les attributs, faisons un titre en double hauteur dans une "fenêtre", avec sous-titres en d'autres couleurs (nous reprenons en partie l'exemple des lignes 400 à 500).

```
 800 'TITRE EN FENETRE
 810 CLS:PAPER6
 820 PRINT@10,7;CHR$(4);CHR$(27)"J";"O
 R I C * A T M O S";CHR$(4)
 830 FOR I=6 TO 9
 840 PRINT@9,I;CHR$(147):PRINT@32,I;CH
R$(150)
 850 NEXT
 860 PRINT@7,16;CHR$(132)"C'EST";CHR$(
128)"FACILE";CHR$(129)"A PROGRAMMER"
 870 PRINT@10,22;CHR$(140);CHR$(133)"T
apez une Touche";CHR$(17);
 880 GET R$:PRINT CHR$(17);
 890 END
```

Analysons certaines lignes :
Sur le titre en double hauteur (en lettres noires puisque INK n'est pas spécifié), nous allons superposer trois « segments » jaunes qui vont constituer la fenêtre jaune. On est en fond cyan, donc on tronque l'extrémité de ces bandes jaunes par une bande cyan. Tout est dans la ligne 840, répétée trois fois par la bande FOR NEXT de la ligne 830.

En ligne 860, nous écrivons la phrase « C'est facile à programmer » en trois couleurs de lettres, mais observez bien que nous n'avons pas mis d'espace au début de chaque tronçon puisque l'attribut d'encre va faire ce décalage.
En ligne 870, nous combinons l'attribut d'encre et l'attribut de clignotement, soit deux attributs. Remarquez sa tabulation à 10 alors qu'elle était à 12 dans la ligne 500... Rappelons que le CHR$(17) efface le curseur clignotant.
En ligne 880, après avoir tapé une touche, nous restituons le curseur.

En paramétrant (paramétrer = obtenir par calcul) le numéro de code de l'attribut dans des boucles FOR NEXT, ralenties ou non par des WAIT, on peut obtenir très simplement des effets colorés originaux, voire animés. Voulez-vous vous amuser un peu ? Alors tapez le programme suivant. Vous pourrez en faire un générique pour votre logiciel préféré en l'adaptant à vos besoins, mais, avant de l'écrire, quelques avertissements :

— La ligne 915 est facultative, elle sert à effacer le mot « CAPS » en haut de l'écran : on lui superpose quatre espaces dans la mémoire d'écran.
— Ligne 940 : POKE 26,96 efface le « READY » (rappel).
— Ligne 950 : mettez votre nom dans la couleur qui vous convient (ici magenta).
— Ligne 960 : après avoir tapé une touche quelconque, CALL # 247 ou CALL 583 (pour l'ATMOS) provoque le RESET, ce qui restituera le « CAPS », le curseur clignotant et le « READY », mais sans effacer le programme... Attention ! Pour un ORIC-1, c'est CALL # 22B ou CALL 555.

```
 900 'MOSAIQUE
 910 CLS:PAPER 0:B=4:PRINT CHR$(17)
 915 FOR I=0 TO 3:POKE 48036+I,32:NEXT
 920 FOR J=0 TO 12:GOSUB 1000:NEXT:B=5
 930 FOR J=1 TO 11:GOSUB 1000:NEXT
 940 INK7:PRINT@2,12;CHR$(140)"Mosaiqu
e":POKE 26,96
 950 PRINT@23,25;CHR$(133)"Editions SO
RACOM"
 960 GET R$:CALL 583
1000 'TRACE D'UNE LIGNE
1010 FOR N=144 TO 150
1020 PRINTCHR$(N);SPC(B);:NEXT:RETURN
```

La ligne 920 détermine la moitié supérieure, la ligne 930 la moitié inférieure. On peut modifier le dessin en changeant les valeurs de B.

Chapitre VI

LES DONNEES : ENTREZ SANS PLANTER

Il n'est pas suffisant qu'un programme soit performant, il faut aussi que l'utilisateur soit dans l'impossibilité de le bloquer accidentellement. Ce point est primordial.

Ne dites pas « l'utilisateur, ce sera moi, et je connais bien mon programme ». Aujourd'hui certes, mais cela ne sera plus vrai quelques mois plus tard. Un oubli, une étourderie au clavier et c'est le plantage honteux. Que faire ? RUN et tout recommencer ? Rechercher fébrilement le listing afin de déterminer quel GOTO fera le sauvetage ? Ni l'un, ni l'autre ; il fallait prévoir des SECURITES sur chaque entrée clavier (on dit aussi « saisie de données »).

Une ligne de sécurité s'écrit en moins de quinze secondes, tandis que débloquer un programme peut prendre un bon (?) quart d'heure... L'erreur peut être répétitive, la sécurité est définitive.

Tout logiciel digne de ce nom doit être conçu pour un utilisateur peu intelligent, étourdi et ignorant tout du BASIC.
Heureusement c'est facile, simplement quelques bonne habitudes à prendre.

POUR ENTRER DES NOMBRES

Bannissez à tout jamais cette écriture de débutant :

```
10 'A NE PAS FAIRE
20 PRINT"PRIX ?":INPUT PR
90 END
```

A l'écran vous aurez le "?" et le curseur sous "PRIX ?". Pas très joli... Ce serait déjà mieux si l'entrée (le curseur) était sur la même ligne, et avec un seul "?" au lieu de deux ! Les effets des lignes 110 et 120 sont rigoureusement identiques, mais la seconde manière est plus courte.

```
100 'ENTREE SUR LA MEME LIGNE
110 PRINT"PRIX ";:INPUT PR
120 INPUT"POIDS ";PD
190 END
```

En effet, cette écriture est dangereuse. Pour vous en convaincre, au lieu de taper le nombre 110, vous avez laissé un doigt sur SHIFT et vous avez entré (!). Regardez le massacre ! Sous notre ligne apparaît "?REDO FROM START" (= refaites depuis le depart), et une ligne encore plus bas, le "?" et le curseur vous invitent à entrer une valeur numérique. Le programme n'est pas bloqué, mais vous pouvez dire adieu à l'esthétique de votre belle page d'écran ; surtout si ces deux lignes ont effacé quelque chose !

Pour éviter cette banale mésaventure, il y a deux règles d'or à toujours respecter :

— Pour entrer un nombre, on le demande sous forme de chaîne, puis on le vérifie et on le confirme par la fonction VAL.
— La ligne contenant INPUT doit débuter par un positionnement sur l'écran. Exemple :

```
200 'SECURITE CHIFFRES PAR VAL
210 CLS
220 PRINT @12,10;:INPUT"PRIX ";PR$
230 PR=VAL(PR$)
240 IF PR=0 AND PR$<>"0" THEN 220
290 END
```

Vous constatez que la ligne 240 refusera les lettre ou signes de ponctuations, mais acceptera le chiffre zéro. D'autre part, en cas d'erreur, la ligne 220 se réécrit au même endroit.

Mais il y a mieux : supposons que l'on ait entrée ''ABCDE'', le curseur revient sur le A en laissant derrière lui ''BCDE''. Il faudra donc taper les chiffres sur ces lettres. Avouez que cela ne fait pas très propre. C'est l'occasion d'inclure un CTRL N, qui efface la ligne, et qui s'écrit PRINT CHR$(14).

```
300 'ERREUR EFFACEE
310 CLS
320 PRINT @12,10;CHR$(14);:INPUT"PRIX
";PR$
330 PR=VAL(PR$)
340 IF PR=0 AND PR$<>"0" THEN 320
390 END
```

Donc, quelle que soit la « bêtise » entrée, la zone d'entrée reste propre. Mais attention, le CTRL N efface *toute la ligne,* et non à partir de la 12e colonne. Donc un seul INPUT par ligne. Mais, rassurez-vous, nous verrons une autre façon de nettoyer la zone d'entrée sans effacer le reste de la ligne, et ce dans un autre programme de ce même chapitre (ligne n° 570).

En fait, il faut signaler un cas particulier, mais hélas fréquent, où nous n'avons pas trouvé la parade. Dans le programme 200-290 ou 300-390, essayez de répondre par la touche RETURN. Il se passe le phénomène suivant : le curseur ''?'' passe à gauche de la ligne au-dessous et attend l'entrée numérique ; la ligne 340 est inefficace mais c'est l'équivalent à un ''REDO FROM START''. Vous pouvez toujours essayer d'ajouter une ligne du genre IF PR$ = '''' (chaîne vide) ou CHR$(13), rien n'y fait, c'est un BUG ORIC...

Si cela vous arrive, tapez un blanc (barre d'espacement) et RETURN, le curseur revient alors à sa bonne place.

LES CODES D'INTERVENTIONS

L'écran vous demande d'entrer une suite de valeurs, mais vous êtes seul à connaître leur nombre ; il faut donc prévenir le micro-ordinateur quand vous avez terminé la saisie de données. Par exemple, le code de fin sera la lettre Q, initiale de « quitter ». Voici un petit programme qui calcule l'âge moyen d'un groupe quelconque de personnes :

```
400 'MOYENNE D'AGE
410 CLS:N=0:TA=0:'NOMBRE,TOTAL AGES
415 PRINT@10,22;"POUR QUITTER --> Q"
420 PRINT@ 14,10;CHR$(14);:INPUT"AGE
";AG$
430 IF AG$="Q"THEN 470
440 AG=VAL(AG$):IF AG=0 THEN 420
450 N=N+1:TA=TA+AG
460 GOTO 420
470 CLS:PRINT@5,10;"L'age moyen de ce
s";N;"personnes"
480 PRINT@ 12,12;"est de";INT(TA/N +.
5);"ans."
485 PRINT@10,22;SPC(18)
490 END
```

Analysons :

Ligne 410
Il n'était pas utile d'initialiser N et TA ; c'est pour la clarté.
Ligne 415
On affiche le code au bas de l'écran.
Ligne 430
Q saute la validation et le comptage.
Ligne 440
On refuse l'âge zéro (normal...).
Ligne 450
Les incrémentations de N et TA ne se font qu'après validation de l'entrée.
Ligne 480
Il serait stupide de donner le résultat avec six décimales, on l'arrondit à l'entier *le plus proche* (par le +.5).
Ligne 485
On efface la légende du code pour faire plus propre.

Un autre code d'intervention est la lettre E lorsque l'on se rend compte que l'on a commis une *erreur,* mais on a déjà appuyé sur la touche RETURN. Deux possibilités pour ajouter une telle ligne (en 435) : le code E renvoie en 410 et on doit alors tout recommencer ; ou mieux, on implique alors que N = N – 1 et que TA = TA – AG (qui est toujours en mémoire) et enfin retour en 420. Il faudrait aussi modifier les lignes 415 et 485 pour signaler que l'on dispose aussi de l'option Erreur = E.

Il faut absolument retenir que les codes d'interventions en service doivent être mentionnés (légendes) au bas de l'écran. En effet, si l'opérateur a affaire à plusieurs tableaux de saisie, comment voulez-vous qu'il sache (ou se rappelle) qu'il a la possibilité de corriger ou non une erreur ?

LA SAISIE PLEINE PAGE

Dans l'exemple précédent, les valeurs entrées disparaissent de l'écran, et si la liste est longue, on risque brusquement de ne plus savoir où on en est ! Il est donc souvent souhaitable de conserver un contrôle visuel sur ce qui a déjà été entré. Dans le programme « moyenne d'âge », on pourrait faire afficher les âges entrés en colonne sur un côté de l'écran. Pas très joli mais efficace, car ici le problème était simple puisque c'est toujours le même nom de variable AG. Malheureusement, dans la pratique, ce cas est fort rare, et nous devons entrer sur plusieurs noms de variables ; il faut donc que :

a) les localisations des INPUT se fassent à des endroits différents de l'écran ;
b) après avoir entré une donnée, il faut que le curseur aille se placer automatiquement sur la zone d'entrée suivante.

C'est la « saisie pleine page ». C'est facile avec une quinzaine de questions. Il suffit de les afficher les unes au-dessus des autres ; vous savez le faire. Mais vous devez remplir les cases d'un tableau, par exemple, quatre colonnes sur douze lignes, soit quarante-huit valeurs... De ce fait l'écran devra représenter ce tableau, avec légendes des colonnes et des lignes, et avec bien sûr le déplacement automatique du curseur clignotant à la case suivante.

Le programme qui suit n'est pas seulement une démonstration, c'est un « produit fini ». Non seulement nous vous invitons à le taper au clavier pour en « toucher » le mécanisme, mais nous vous conseillons de l'enregistrer sur cassette, car il y a neuf chances sur dix pour qu'il vous serve personnellement. Vous n'auriez alors qu'à modifier les légendes de colonnes de la ligne 520.

Les douze lignes représentent le mois, les quatre colonnes des « postes » d'entrées ou de sorties, d'argent, d'heures, d'objets, à votre guise. Chaque entrée validée va aller se placer dans un tableau DIM BL (12, 4) ; BL voulant dire « bilan annuel ». Nous allons profiter de l'occasion pour lui faire calculer chaque total ligne que l'on appelle TM (total mois) et chaque total-colonne T, d'où le total annuel global TA. Non seulement ces sous-totaux vont s'afficher sur l'écran, mais nous les mettons aussi en DIM. Où cela ? Mais dans les indices zéro. En effet, rappelons que lorsque l'on définit un tableau DIM BL (12, 4), on dispose en fait des lignes 0 à 12 (soit 13) et des colonnes 0 à 4 (soit 5).

Afin de ne pas alourdir davantage, nous n'avons pas introduit de code d'erreur, mais comme maintenant vous savez faire...

```
500 'PAGE TABLEAU BILAN ANNUEL
510 CLS:PAPER2:DIM BL(12,4)
520 PRINT@2,1;"MOIS   PAUL   JEAN   M
```

```
ARIE  MARC   TOT"
 530 FOR M=1 TO 12
 540 PRINT@2,3+M;M
 550 FOR C=1 TO 4
 560 PRINT@ C*7,M+3;:INPUT P$
 570 P=VAL(P$):IF P=0 AND P$<>"0"THEN
PRINT@ C*7,M+3;SPC(8):GOTO 560
 580 PRINT@ C*7,M+3;"!"
 590 BL(M,C)=P:NEXT:GOSUB 1000:NEXT
 600 'TOTAL ANNUEL PAR PERSONNE
 610 PRINT@2,18;"TOTAL"
 620 FOR C=1 TO 4:FOR M=1 TO 12
 630 T=T+BL(M,C):NEXT:BL(0,C)=T
 640 PRINT@ C*7,18;T:T=0:NEXT
 650 'TOTAL ANNUEL GLOBAL
 660 FOR C=1 TO 4:TA=TA+BL(0,C):NEXT
 670 BL(0,0)=TA
 680 PRINT@ 9,20;"TOTAL ANNUEL:";TA
 690 END
 1000 'TOTAL MENSUEL
 1010 FOR C=1 TO 4:TM=TM+BL(M,C):NEXT
 1020 L=LEN(STR$(TM))-1
 1030 PRINT@39-L,M+3;TM
 1040 BL(M,0)=TM:TM=0
 1050 RETURN
```

Analysons quelques points de détails :

Ligne 520
Une façon de légender des colonnes autrement que par TAB. Avantage : cette chaîne peut s'appeler LE$ si ce programme devrait être appelé par un GOSUB. Ainsi, en changeant LE$ avant ce GOSUB (dans le programme principal), vous pourrez entrer plusieurs bilans avec des quartés de postes complètement différents. De retour au programme principal, vous pourrez archiver chaque bilan sur cassette par la fonction STORE.

Ligne 530
Première boucle FOR NEXT, on traite d'abord par ligne, donc par mois, lequel s'inscrit dans la colonne « MOIS ».

Ligne 550
Seconde boucle FOR NEXT, imbriquée dans la première, on traite chaque colonne.

Ligne 560
C'est le positionnement paramétré de chaque INPUT.

Ligne 570 (fin)
Comme nous ne pouvons effacer une erreur par CTRL N, nous affichons huit blancs à la place.

Ligne 580
Ça, c'est un gadget ! Les ''?'' laissés par les INPUT précédents, ne sont guère esthétiques, aussi sont-ils remplacés après usage par ce petit trait vertical (la touche en haut à droite + SHIFT). D'une ligne à l'autre, ils construisent ainsi des *traits verticaux continus* qui vont séparer les colonnes.

Lignes 1020-1030
En fin de ligne, le total mensuel est calculé et affiché, mais pour éviter de perdre de la place et afin d'avoir les chiffres des unités sur une même colonne, nous paramétrons le positionnement du PRINT par la longueur du nombre (ligne 1030).

L'ATMOS dispose un blanc devant chaque nombre affiché par PRINT. Aussi, si on transforme le nombre 24 en chaîne, par STR$, la longueur LEN de cette chaîne est trois = 1 blanc et ''23''. D'où le '' – 1'' à la fin de la ligne 1020 (à vérifier sur ORIC-1...).

Que de valeurs sur le même écran ! Quelle vue d'ensemble ! Si vous avez une imprimante, c'est le moment de lui commander une copie d'écran (nous verrons comment dans un chapitre ultérieur), avec en plus la possibilité d'archiver ces 65 valeurs (48 entrées, 17 calculées) en commandant STORE BL, ''FRAIS SPORT 84'', par exemple...

Si vous n'avez pas encore une grande habitude des tableaux DIM à plusieurs dimensions, ce sera pour vous l'occasion de « fixer les idées ». Le tableau terminé, interrogez-le en mode direct, en tapant, par exemple :

PRINT BL (7,2), BL (7,0), BL (0,0) et RETURN

Note : Il existe dans le commerce de coûteux logiciels appelés « ...CALC... » qui effectuent des saisies de tableaux. Etant polyvalents, ils sont lourds à définir (nombreuses options demandées). Vous constatez que, lorsque l'on sait se servir du BASIC, on a souvent intérêt à s'en faire un « sur mesures ». Enorme gain en mémoire et vitesse de mise en « opérationnel » plus rapide, puisque sans options. Quant au prix...

LES ENTREES DE CHAINES

Nous allons très souvent faire appel aux fonctions LEFT$, MID$, RIGHT$; et pour vous éviter de revoir le manuel, nous en faisons tout de suite une brève description.

Elles servent à LIRE une partie de la chaîne citée. Les indices commencent à 1 et non à zéro. Exemple : LEFT$(A$,4) donne les quatre premiers caractères en partant de la gauche, tandis que RIGHT$(A$,4) prend les 4 premiers caractères en partant de la droite.

MID$(A$,4,2) va prendre deux caractères à partir du quatrième en partant de la gauche ; ce qu'on appelle les caractères numéro 4 et 5.OK ?

Attention, une lacune souvent gênante dans le BASIC des ORIC : la fonction *MID$ ne peut servir à écrire* dans une chaîne existante ; ainsi, en faisant MID$(A$,4,2)="XXW", vous aurez droit à un "SYNTAX ERROR" (dans la plupart des BASIC, MID$ peut à la fois lire et écrire).

Nous reviendrons plus en détail sur ces fonctions au chapitre IX.

Lorsque l'on entre une chaîne par un INPUT, il est souvent nécessaire de la « formater », c'est-à-dire lui imposer une longueur maxi, mini ou fixe, parce que la suite du programme l'exige, pour détecter des erreurs d'entrées, ou pour obtenir une belle présentation sur une feuille d'imprimante.

Les quatre exemples suivants illustrent les utilisations fréquentes de ces divers formatages de chaînes :

```
1100 'CHAINE MAXI=8
1110 CLS
1120 PRINT@10,10;:INPUT"NOM";NM$
1130 NM$=LEFT$(NM$,8)
1140 PRINT@5,13;NM$,LEN(NM$)
1190 END
1199 '-----------------------------------
1200 'CHAINE IMPOSEE A 8
1210 CLS
1220 PRINT@10,10;:INPUT"NOM";NM$
1230 IF LEN(NM$) <> 8 THEN 1220
1290 END
1299 '-----------------------------------
1300 'CHAINE FORMATEE A 8
1310 CLS
1320 PRINT@10,10;:INPUT"NOM";NM$
1340 NM$=LEFT$(NM$+"        ",8)
1390 END
1399 '-----------------------------------
1400 'CHAINE CALEE A DROITE
1410 CLS:I=2:PRINT@12,23;"QUITTER-->
Q"
```

```
 1420 PRINT@10,1;CHR$(14);:INPUT"QUANT
ITE";NB$
 1425 IF NB$="0" THEN 1490
 1430 NB=VAL(NB$):IF NB=0 AND NB$<>"0"
 THEN 1420
 1440 NB$=RIGHT$("      "+NB$,6)
 1450 PRINT@32,I;NB$:I=I+1:GOTO 1420
 1490 END
```

Dans le premier programme, nous vous avons ajouté la ligne 1140 pour montrer que l'ORIC accepte d'essayer de tronquer à 8 une chaîne de longueur inférieure à 8.

Le dernier programme (1400-) est très important, car les lignes 1440 et 1450 ont le même effet que la fonction BASIC PRINT USING que ne possèdent pas les ORIC. Vous remarquerez que les nombres entrés s'affichent à droite de l'écran parfaitement alignés sur les chiffres des unités. Cette pratique est vivement conseillée pour présenter des colonnes de nombres sur imprimantes. Pour des nombres décimaux, c'est plus compliqué, nous verrons cela dans un chapitre ultérieur.

LA FONCTION GET

La fonction GET diffère de INPUT par les points suivants :

— elle n'attend qu'un seul caractère (pas de "?") ;
— elle ne l'affiche pas à l'écran ;
— il n'est pas utile de faire RETURN ;
— si elle attend un chiffre et que l'on entre une lettre, le programme « plante » avec le message "TYPE MISMATCH ERROR" ; donc sans "REDO FROM START" qui lui ne bloquerait pas le programme.

De ce fait, on n'écrit jamais GET N, c'est trop risqué, mais plutôt GET R$ (R = initiale de réponse).

On l'utilise soit pour interrompre un programme (lecture d'écran), lequel repart dès que l'on frappe une touche quelconque, soit pour répondre à un menu. Exemple pour un programme rassemblant quatre jeux

```
1500 'REPONSE MENU PAR GET
1510 CLS:PRINT@16,5;"M E N U"
1520 PRINT@12,10;"1- FORMULE 1"
1530 PRINT@12,12;"2- MORPION"
1540 PRINT@12,14;"3- TIR LASER"
```

```
 1550 PRINT@12,16;"4- SLALOM"
 1560 PRINT@12,18;"5- QUITTER"
 1570 PRINT@16,24;"REPONSE:";:GET R$
 1580 R=VAL(R$):IF R<1 OR R>5 THEN 157
0
 1590 ON R GOSUB 12000,16000,20000,250
00,1690
 1600 GOTO 1500
 1690 CLS:END
```

Remarquez qu'à la fin d'un jeu on revient au menu.

LA FONCTION KEY$ (KEY = touche de clavier)

KEY$ désigne le caractère se trouvant dans le buffer clavier, il n'a donc aucun pouvoir d'arrêt sauf si on le teste par une condition précise. Son usage classique est dans les jeux d'arcades pour décéler si l'opérateur a tapé une touche (flèches) ; le tout généralement dans une boucle REPEAT/UNTIL. Mais rien n'empêche de l'utiliser dans une boucle FOR NEXT. Exemple :

```
1700 'INTERVENTION PAR KEY$
1710 CLS:PRINT"TAPEZ A":PRINT
1720 FOR I=1 TO 10000
1730 C$=KEY$
1740 IF C$="A"THEN PRINT I
1750 NEXT:PRINT"TERMINE !"
1790 END
```

Chapitre VII

LA BONNE GESTION DES DIM, FICHIERS ET DATA

C'est simple lorsqu'il y a relativement peu de données et que l'on ne s'en sert qu'une fois ; au-delà il y a souvent des mauvaises surprises du genre ''REDIM ARRAY ERROR'', ''OUT OF DATA ERROR''... L'ORIC, comme beaucoup de micro-ordinateurs, ne possède pas de fonction ERASE pour « vidanger » un tableau DIM, ou le RESTORE avec numéro de ligne. Il faut donc bien préparer son affaire avant de programmer. Si vous avez de *l'ordre* dans vos DIM et vos DATA, vous pourrez, en très peu de lignes, gérer et stocker un nombre considérable de données numériques et alphanumériques (= chaînes).

MIEUX CONNAITRE LES DIM

INITIALISATION

C'est en début de programme que l'on « annonce » ses DIM, même si l'on s'en sert que beaucoup plus loin. Mettez-les sur une même ligne ; il ne sera alors pas utile de répéter ''DIM'', il suffit de les séparer par une virgule.

Exemple :

```
30 DIM BL(12,4), AC$(24), RD%(520), Z(5).
```

Comme pour les variables, nous avons droit à deux caractères, plus le suffixe $ ou %.

On peut s'abstenir de DIMensionner un tableau si le nombre n'excède pas 10, mais ne croyez surtout pas faire des économies de mémoire, bien au contraire ! Si vous entrez quelque part P(1) = 124, aussitôt un micro-ordinateur établit lui-même un DIM P(10) ! Si vous n'avez que quatre valeurs, c'est un gaspillage d'environ 40 octets ; il aurait mieux valu annoncer dès le départ DIM P(4).

Une fois annoncé, il n'est plus possible de redimensionner un tableau DIM, ni même de le répéter ! Ainsi, plus loin, si nous redisons DIM AC$(24), c'est le plantage avec code d'erreur. C'est pour ne pas risquer qu'un GOTO provoque un ''REDIM ARRAY ERROR'' (ARRAY = tablau, réseau) que nous placerons nos annonces de DIM en début de programme. Pour « effacer » ces dimensions, il y a deux ordres possibles, RUN ou CLEAR, mais dans les deux cas vous perdez toutes vos variables.

L'ESPACE MEMOIRE

L'emploi des DIM est assez traitre en ce qui concerne leur encombrement mémoire, il faut se méfier et calculer. Ainsi, notre tableau BL(12,4) occupe $(12+1)\times(4+1)\times 6$ octets = 390 octets (+ ce qu'il faut à la fonction DIM).

Dans l'exemple de notre ligne 30, nous avons annoncé DIM RD%(520), soit 521 nombres entiers de deux octets chacun, donc au minimum 1042 octets. Or, si nous n'avions pas utilisé le « % », ce serait le triple !

Méfiez-vous également des tableaux de chaînes, à raison de 1 octet par caractère, ça monte parfois très vite en mémoire...

LES DIM A TROIS DIMENSIONS

Nous savons que DIM Z(5) est une liste banale, que BL(12,4) est un tableau de 13 lignes en 5 colonnes (les indices zéro à ne pas oublier...), mais que représenterait DIM BG(12,4,2) ? C'est tout simplement plusieurs tableaux (12,4) ''superposés'' en quelque sorte en rez-de-chaussée et deux étages. Soyons plus concrets en reprenant notre tableau de bilan annuel BL(12,4) du chapitre précédent : BG(12,4,1) sera le bilan 1984, BG(12,4,2) le bilan 1985, et BG(12,4,0) pourra être au choix le bilan 83, ou le total, la moyenne ou la variation en % entre 84 et 85. Donc trois tableaux (12,4) de structures identiques, mais d'intitulés différents.

LES DIM A PLUSIEURS DIMENSIONS

On a la possibilité d'annoncer DIM BV(12,4,2,3) et puisque BG(12,4,2) représentait une pile de trois tableaux, ce DIM BV représente un ensemble regroupant quatre de ces piles, par exemple, pour des quartés de colonnes différentes.

On peut même aller plus loin ; la cinquième dimension est admise, mais alors là, deux conseils : prenez des notes claires et détaillées pour vous y retrouver... et faites le calcul de l'encombrement mémoire...

LA REUTILISATION D'UN TABLEAU DIM

Vous avez rempli un tableau DIM W(45) et vous voulez le regarnir avec d'autres valeurs. Trois cas se présentent :

- On sait que tout sera renouvelé ; aucun problème, les nouvelles valeurs écraseront les précédentes.
- On sait que l'on ne va pas le remplir complètement, mais on sait d'où à où, par exemple de W(1) à W(27). Ce n'est pas très gênant car on l'exploitera ensuite de 1 à 27.
- Même cas que précédemment, mais on ne veut pas qu'il subsiste des données anciennnes, par exemple parce que l'on va enregistrer en fichier par STORE. Il faut donc le ''laver''avant de le regarnir. Cette mise à zéro est simple et rapide :

```
FOR N=0 TO 45:W(N)=0:NEXT
```

S'il s'agissait d'un tableau de chaînes AC$(24) :

```
FOR N=0 TO 24:AC$(N)='' '':NEXT
```

S'il s'agit d'un tableau à plusieurs dimensions, il faut utiliser des boucles FOR NEXT imbriquées. Vous savez faire.

LE TABLEAU CENTRE DE CALCULS

Si vous avez de nombreux calculs à effectuer sur les données entrées dans un tableau DIM, prévoyez des colonnes et des lignes supplémentaires pour des résultats intermédiaires ou définitifs ; mais, de grâce, ne sortez surtout pas du tableau !

Reprenons les totaux. Nous avions utilisé la ligne et la colonne zéro, mais si nous avions à faire, en plus des calculs de moyennes, des coefficients comparatifs, des statistiques et autres insanités, nous aurions aurions plutôt annoncé dès le départ DIM(15,7). En restant en « vase clos », les calculs les plus fastidieux s'écrivent en une ou deux lignes BASIC dans une boucle FOR NEXT.

Mieux encore : imaginons qu'à mi-calcul on n'ait plus besoin des résultats intermédiaires de la ligne 14 et de la colonne 6 : écrasons-les par des résultats à conserver en cours de calcul, et ainsi de suite. Par cette technique répétée, il nous est arrivé d'éviter de gaspiller un nombre respectable de kilo-octets sur un très gros tableau.

ENREGISTREMENT ET LECTURE DES FICHIERS

Les fonctions STORE et RECALL sont d'une facilité d'emploi... désarmante ! On désigne le tableau, et la ROM se charge de tout, et très vite. Pensez que sur la plupart des micro-ordinateurs il faut décortiquer le tableau par des boucles FOR NEXT, pour enregistrer ou lire chaque élément.

Supposons que vous ayez à la fois des nombres et des chaînes ; vous pouvez soit enregistrer les deux tableaux à la suite, soit transformer vos nombres en chaînes (par STR$), et traiter ainsi un seul tableau chaînes. Ce sont les cas particuliers qui feront la décision.

Vous savez qu'après l'ordre STORE ou RECALL on met le nom du tableau DIM, virgule et un nom de baptême quelconque entre " ". Or, nous avons remarqué un petit BUG de l'ORIC ATMOS concernant le nom de baptême : on ne peut pas le paramétrer.

```
10 'STORE DANGEREUX
20 DIM BX(35)
25 FOR I=1 TO 35:BX(I)=I*2:NEXT
30 AN=84
40 A$="TRUC"+STR$(AN)
50 STORE BX,A$
90 END
```

Si vous essayez ce petit programme et que vous le rechargiez par RECALL, deux éventualités : ou bien l'écran se brouille et vous aurez de la chance si RESET est efficace, ou bien tout semble très correctement se passer à l'enregistrement et la lecture. Mais ne criez pas victoire pour autant ; faites imprimer le contenu de DIM BX(35) ainsi reconstitué : que des zéros....

Nous avons vérifié que le blanc entre « TRUC » et « 84 » n'avait aucune responsabilité : mystère !

Voilà comment nous nous tirons d'affaire : l'index qui sert à identifier le fichier, parmi plusieurs enregistrés à la file, n'est pas dans le titre mais dans le tableau lui-même, par exemple en BX(0.).

```
100 'STORE PARAMETRE
110 DIM BX(35):AN=84
120 FOR I=1 TO 35:BX(I)=I*2:NEXT
130 BX(0)=AN
140 STORE BX,"TRUC"
190 END
```

A la lecture, pour retrouver le "bon" dans une série de fichiers tous appelés « TRUC » :

```
200 'RECALL PARAMETRE
210 DIM BX(35):AN=84
220 REPEAT
230 RECALL BX,"TRUC"
240 UNTIL BX(0)=AN
290 END
```

Il est évident que pour un fichier chaînes, l'index numérique sera mis en chaîne par STR$,

L'UTILISATION DES DATA

Les lignes du DATA sont très faciles à utiliser si l'on a bien compris leur principe de fonctionnement. Ce n'est pas compliqué, loin de là, mais c'est assez spécial.

- Lorsque l'on met des DATA dans un programme, ils vont s'empiler dans une zone à part de la RAM, appelons la « la boîte à DATA ».
- On peut y mélanger des nombres et des chaînes. Chaque élément, séparé par une virgule, est comme une carte mise dans la boîte.
- On lit ces cartes par la fonction READ. Elles sont lues dans le même ordre dans lequel on les a entrées.
- Dès qu'une carte a été lue, elle est mise dans une seconde boîte.
- Avec l'ordre BASIC RESTORE (= recharge), les cartes lues sont remises dans la première boîte ; elles pourront donc être relues.
- Si la boîte n° 1, celle des DATA à lire, a été vidée et que l'on demande READ, le programme plante, avec le message

"OUT OF DATA ERROR"

- Si l'on a besoin de 10 DATA à partir de la 25e, il faudra d'abord "lire à blanc" les 24 premières.
- Le nombre de DATA par ligne n'a aucune importance.
- Les lignes de DATA peuvent être avant ou après les commandes READ.

Le gros avantage des DATA est l'économie d'écriture lorsque l'on tape un programme. L'exemple suivant vous en convaincra : dans la première partie, nous remplissons un tableau DIM avec 24 *constantes* (mois de l'année avec leurs nombres de jours). Sans les DATA il aurait

fallu écrire : M$(1,1) = "janvier" : M$(1,2) = "31" : etc... Quelle corvée ! Nous faisons cela en quelques lignes courtes (lignes 310 à 360). Ce petit programme n'a aucune utilité pratique, mais il rassemble toutes les possibilités d'utilisations des DATA :

```
 300 'DATA EN DIM
 310 DIM M$(12,2)
 320 FOR I=1 TO 12:READ M$(I,1):NEXT
 330 FOR I=1 TO 12:READ M$(I,2):NEXT
 340 DATA JANVIER,FEVRIER,MARS,AVRIL,M
AI,JUIN
 350 DATA JUILLET,AOUT,SEPTEMBRE,OCTOB
RE,NOVEMBRE,DECEMBRE
 360 DATA 31,28,31,30,31,30,31,31,30,3
1,30,31
 400 'RE-UTILISATION DES DATA
 410 'NOMBRE DE JOURS DU SEMESTRE 2
 420 RESTORE
 430 FOR I=1 TO 18:READ A$:NEXT
 440 FOR N=1 TO 6:READ J
 450 T=T+J:NEXT:CLS
 460 PRINT@ 5,10;"LE SECOND SEMESTRE A
";T;"JOURS"
 490 END
```

Dans la première partie, les 24 DATA ont été pris comme chaînes, or, dans la seconde partie (lignes 420 à 460) les DATA de la ligne 330 sont pris, cette fois, comme valeurs numériques. Après le RESTORE, nous lisons à blanc 18 DATA (ligne 430), puis nous prenons la suite comme *nombres* (ligne 440).

L'ordre READ équivaudrait à un « GOSUB DATA », et pour vous le prouver, ajoutez la ligne 339 END.
Faites RESET, puis RUN, et après le « READY » demandez PRINT M$(10,1) et vous aurez « OCTOBRE ». Donc, il suffit que le programme soit en RAM pour que les DATA soient accessibles, comme pour des sous-programmes. Petite curiosité : le bloc des DATA peut être rechargé par RESTORE, RUN ou CLEAR, mais pas par RESET. Exemple : exécutez le programme par RUN. Les DATA sont "vidés". Faites RESET et tapez READ A$: PRINT A$. Réponse « OUT OF DATA ERROR ». Tapez CLEAR, et reposez la question précédante ; réponse « JANVIER ».

CONSEILS DATA PRATIQUES

Sur une même ligne, vous avez vu que l'on pourrait alterner des chaînes et des nombres, donc avec destinations différentes. Pour la *clarté* ne le faites pas ! Faites des lignes ordonnées, voire légendées en haut par un REM, sinon vous serez perdu le jour où vous vous voudrez modifier une seule valeur parmi ces DATA.

Remarquez que dans notre programme, nous avons mis six noms de mois par ligne et les douze nombres ensemble. C'est uniquement pour la lisibilité. Essayez de changer de ligne que lorsque cela correspond à un changement de rubrique ; nous en verrons un exemple concret au chapitre XI.

Vous avez vu que nous avons dû compter 18 DATA à lire à blanc ; ici, c'était facile... alors pour éviter de fastidieux comptages, prenez les dispositions suivantes :

— Ne dispersez pas vos DATA un peu partout dans votre programme.
— En tête de chaque bloc de lignes DATA écrivez une ligne REM pour dire à quoi ce bloc est destiné, et le *nombre total* de DATA qu'il comporte. Il sera plus facile d'additionner 56 + 34 que de compter 90 DATA une par une...

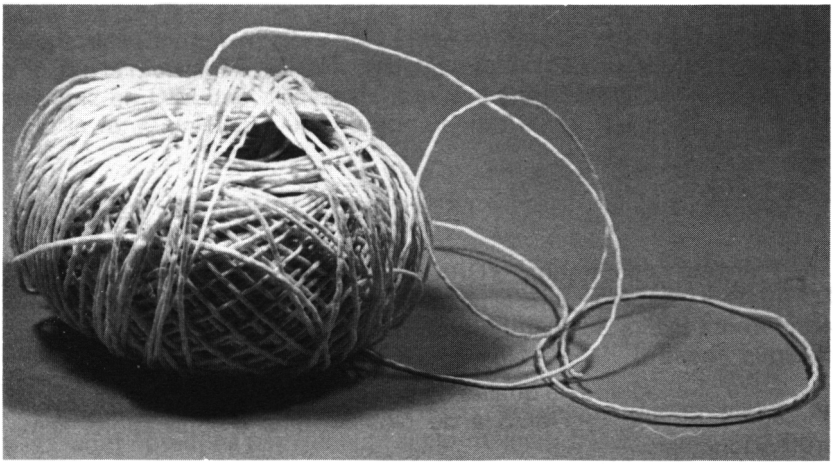
DIVERS

Si vous laissez un blanc entre une virgule et une chaîne, il ne sera pas pris en compte. Si vous tenez à ce blanc, mettez le DATA entre " " :

Exemple :

, MARS, AVRIL, " MAI", JUIN, etc..

Si des lignes DATA comportent chacune huit nombres de 1 ou 2 chiffres, laissez un blanc devant chaque nombre à un chiffre. Ainsi, toutes ces lignes auront au listing la même longueur ; pas pour faire joli, mais afin de vérifier, sans les compter, qu'il y en a bien huit par ligne.



Chapitre VIII

LES BOUCLES ET LEURS PIEGES

Nous disposons de trois sortes de boucles : les FOR NEXT, REPEAT, UNTIL et par GOTO. Chacune a ses avantages et inconvénients, aussi, le choix se fait en fonction du problème à traiter.

LES VITESSES D'EXECUTIONS

Elles sont très différentes ; dans le rapport un à dix ! Nous vous livrons nos chronométrages (au centième de seconde arrondi au dixième) pour les faire compter de 1 à 1000 :

```
10 'DUREES DES BOUCLES A 1000:
20 'FOR NEXT ---> 1.3 s
30 FOR I=1 TO 1000:NEXT
40 '-----------------------
50 'FOR NEXT I---> 1.7 s
60 FOR I=1 TO 1000:NEXT I
70 '-----------------------
80 'REPEAT..UNTIL---> 10.4 s
90 REPEAT:N=N+1:UNTIL N=1000
```

```
100 '-----------------------
110 'COMPTAGE IF+GOTO---> 13.9 s
120 K=K+1:IF K=1000 THEN 140
130 GOTO 120
140 END
```

Première constatation : l'ORIC ATMOS est rapide (un peu plus que l'ORIC-1), il est même dans le peloton de tête en rapidité parmi les micro-ordinateurs 8 bits du marché (son microprocesseur 6502 y est pour quelque chose !).

Note : Sa rapidité, la qualité de son clavier mécanique et sa qualité d'image font que l'ATMOS a de plus en plus d'applications semi-professionnelles.

Deuxième constatation : en ligne 60, l'ajout *inutile* du nom de la variable après NEXT augmente le temps de 30 %. Même si vous avez quatre boucles FOR NEXT imbriquées, il n'a pas besoin qu'on lui indique la variable, il utilisera toujours le bon NEXT sans risque d'erreur.

Troisième constatation : les boucles par incrémentation (comptage) et comparaison de valeurs sont considérablement plus lentes. En fait, ce sont surtout les comparaisons qui retardent, *surtout la fonction IF.* UNTIL fait à peu près la même chose, mais moins lentement.

Une preuve du retard d'exécution apporté par chaque IF dans une boucle : on modifie ainsi la ligne 30.

```
200 'RETARD DU A IF:---> 4.8 s
210 FOR I=1 TO 1000:IF I=0 THEN PING
220 NEXT
290 END
```

Donc, la plus simple des conditions IF (IF I = 0...) prend 3,5 millisecondes ; des comparaisons plus complexes peuvent demander dix, vingt fois plus... En conclusion, quand le nombre de boucles est important (plus de 100), soyez assez prudent, radin même, sur le nombre de conditions IF que vous y logez. Un programme qui se met à ''pédaler dans le vide'' pendant de longues secondes (ou minutes...), est très déroutant pour l'opérateur ; lequel peut s'affoler et taper CTRL C croyant avoir fait une fausse manœuvre !

Il est des cas où ces ''temps morts'' sont inévitables (exemple : les tris alphabétiques), alors prévenez l'opérateur en faisant apparaître à l'écran le message « Un peu de patience... ». Si vous aimez l'humour, préparez en DIM une dizaine de messages différents de ce genre, et c'est la fonction RND qui fera son choix...

LES PIEGES DES BOUCLES FOR NEXT

Le piège classique est celui des valeurs finales en fin de boucle : si en ligne 35 on avait demandé PRINT I, vous auriez 1001 : la boucle a bien compté jusqu'à 1000, puis a *ajouté son STEP* (ici 1 par défaut) mais ne l'a pas exécuté. Rappelez-vous : la valeur finale de la variable est celle de sa dernière valeur de boucle plus la valeur du STEP.

D'autres surprises surviennent quand les valeurs départ, arrivée et STEP sont obtenues par calculs : si par malheur STEP = 0 vous risquez d'attendre la suite longtemps... Si le STEP est négatif alors que la progression est positive, il n'y a pas de boucle, pas de message d'erreur, et le programme se poursuit.

Exemple :

```
FOR I=TO 5 STEP-1:NEXT:PING
```

Après le PING, faites PRINT I ; réponse 1. C'est traitre !

L'ATMOS accepte des valeurs départ, arrivée et STEP décimales :

Faisons :

```
FOR I=1.3 TO 4:PRINT I:NEXT
```

Nous aurons pour I 1,3 ; 2,3 et 3,3.
Donc, il ne ''grogne'' pas si la valeur arrivée est impossible, mais ne la dépasse pas. Même chose évidemment si la valeur d'arrivée est décimale avec départ entier et STEP = 1. En conséquence, si par votre calcul vous pensiez obtenir une valeur d'arrivée égale à 12, mais qu'en fait son résultat est 11.99999, il s'arrêtera à 11... Pensez à arrondir vos calculs à l'entier le plus proche ; vous vous souvenez de « INT(I+.5) » ?

LES SORTIES DE BOUCLES

Imaginons un groupe de 10 lignes où la première commence par FOR et la dernière ne comprend que NEXT ; disons que c'est la ligne 790.

A mi-parcours, vous avez introduit une première condition IF qui, si elle est vérifiée, demande de sauter la suite et de repartir sur un nouvel index de comptage : il serait souvent très dangereux d'écrire IF... THEN NEXT, alors qu'en toute quiétude, vous pouvez écrire IF... THEN 790 (la ligne où se trouve *l'unique* NEXT pour ce FOR).

Deuxième cas : vous introduisez une condition IF qui doit arrêter cette boucle.

Exemple schématique :

```
700 FOR I=1 TO 500
-----------------------
750 IF A=Z THEN I=500:GOTO 790
-----------------------------------------------------
790 NEXT
```

Et la boucle se termine sans ennuis. Désirez-vous mémoriser la valeur de l'index au moment de l'interruption ?

```
750 IF A=Z THEN K=I:I=500:GOTO 790
```

Il est évident qu'il ne faudrait pas sortir d'une boucle par un GOTO vers l'extérieur... Le programme resterait alors dans l'attente d'un NEXT pour revenir au FOR de la ligne 700, ce serait de la roulette russe...

LES PIEGES DES REPEAT UNTIL (UNTIL = jusqu'à ce que)

Le risque est la boucle sans fin (ou presque) parce que la condition UNTIL n'est jamais vérifiée, et ce parce que votre condition est trop précise.

Un exemple :

```
... UNTIL N=12
```

Or ce N est calculé par une division ou une racine carrée, et alors que vous escomptez 12, le calcul donne en fait 11.99999... Donc, UNTIL N = 12 n'est jamais vérifié et la boucle continue. Il aurait suffit d'écrire, par exemple :

```
...UNTIL N+.1>12 ou UNTIL INT(N+0.5)=12
```

Le risque d'obtenir une valeur non entière est dû à une opération arithmétique ou mathématique qui provoque une *réduction* d'un nombre ; nous avons cité la division et la racine carrée, mais il y a aussi les logarithmes. Ce phénomène du « .999... » est d'autant plus traitre qu'il ne se produit qu'avec certaines valeurs, vous avez déjà dû le remarquer avec une calculatrice de poche. Attention ! Même si le calcul de N n'est pas ''dangereux'', il a peut-être utilisé une variable qui elle a été obtenue par division. Soyez donc méfiant ; c'est le genre de programme qui peut fonctionner cent fois de suite et planter la cent-unième sans que l'on comprenne pourquoi... En règle générale, sitôt après avoir tapé le mot « UNTIL », marquez un temps d'arrêt afin d'imaginer tous les cas particuliers, ''vicieux'', qui pourraient passer au travers de votre condition ; disons tous les cas peu probables mais non impossibles. Nuance !

Imaginons que vous avez trouvé un de ces cas rarissimes. Il faut le bloquer mais vous ne trouvez pas la parade dans la condition suivant le mot UNTIL : alors, ajoutez une instruction IF qui va forcer le UNTIL. Imaginons cet exemple suivant.

Vous avez remarqué que dans ce fameux cas extraordinaire, il n'y aurait rien d'écrit à un certain endroit de l'écran : à l'aide du plan mémoire de l'écran (manuel ATMOS page 272), vous déduisez que

l'adresse de cette case est 48740. Une case ''vide'', c'est un espace, et le code ASCII d'un espace, c'est 32 : donc, avant la ligne « UNTIL N = 12 » il suffira d'insérer la ligne :

IF PEEK (48740) = 32 THEN N = 12

De même qu'un bon joueur d'échecs est celui qui prévoit les bons coups de l'adversaire, un bon programme est celui qui prévoit les mauvais coups du hasard.

LES PIEGES DES BOUCLES IF... GOTO

Ces boucles ressemblent fort aux REPEAT UNTIL mais en... négatif.

En effet, toutes deux débutent souvent par une ''incrémentation'' au comptage, du genre « I = I + 1 », mais alors que UNTIL est une ''invitation'' à quitter la boucle, le IF... GOTO est une invitation à la reprendre. Autrement dit le « UNTIL N = 12 » peut être remplacé par :

IF N< >12 THEN GOTO (début de la boucle)

Il est dès lors évident que les pièges précédents s'appliquent à ces types de boucles.

QUELLES BOUCLES CHOISIR ?

Le FOR-NEXT implique un nombre de passages définis, ou du moins maxima (puisque l'on peut les écourter). Nous avons vu qu'elles sont environ dix fois plus rapides, que l'on peut imbriquer avec très peu de risques d'erreur. C'est l'élite, à utiliser en priorité tant que faire se peut.

Les REPEAT-UNTIL losque l'on a aucune idée du nombre de passages qui vont être effectués, l'exemple le plus parlant est celui de la boucle contenant la fonction KEY$. Un autre avantage, sa *clarté de lecture* dans un listing, débute par REPEAT, finit par UNTIL ; pas d'équivoque.

Les IF... GOTO sont les boucles les plus lentes, les listings deviennent souvent très difficiles à décrypter : on ne doit les utiliser que dans des cas si complexes que les deux autres types de boucles sont inutilisables. En effet, puisque les débuts de ces boucles ne sont pas des points de retours systématiques (comme avec FOR ou REPEAT), tous les coups sont permis, mais alors gare aux « labyrinthes BASIC », aux programmes « bouts de ficelles » ou « spaghetti » que nous avons tant décriés au début de cet ouvrage.

Cette solution d'apparente facilité ne doit être réservée que pour des passages d'extrême difficulté ; car c'est du ''pilotage à vue''. Si ça passe tant mieux, si ça ne passe pas, un seul mot de trois lettres pourra sauvegarder vos nerfs : NEW...

LES PIEGES DES INCREMENTS

Paresse et habitude allant de pair, nous avons tous la manie de réutiliser les mêmes noms de variables, surtout pour les comptages. Les bons usages préconisant les lettres de I à N (début de INTEGER ; rappel), on a tendance à trop utiliser les lettres I et N au détriment des autres, et de ce fait, on oublie souvent de les réinitialiser à zéro. Voici un exemple classique :

REPEAT:I=I+1

C'est la première fois que I apparaît, donc I=0 ; très bien.

Mais ultérieurement, on a ajouté en amont un GOSUB vers un sous-programme, contenant lui cette ligne d'apparence anodine, FOR I=1 TO 50...

Donc, notre REPEAT commencera avec valeur I=52. Surprise...

Juste avant l'instruction REPEAT, si on avait logé I=0, il n'y aurait pas eu de problème. Même si une initialisation de variable ne sert apparemment à rien, elle n'a rien de déshonorant : si une modification ultérieure a lieu, vous serez protégé. C'est là le type de la petite erreur bête dont on ne se méfie pas assez. Et ce cas est en fait très fréquent !

Faites très attention à l'endroit où vous placez votre incrémentation : on a trop tendance à la mettre en début de boucle, alors qu'il est souvent plus prudent de la mettre vers la fin. Nous vous en avons donné un exemple au chapitre VI dans le programme « moyenne d'âge » : vous y remarquerez que les incrémentations (ligne 450) ne se font qu'après validation de AG$.

Il est des cas où l'on est vraiment obligé d'incrémenter dès le départ : si en aval intervient une invalidation, pensez, dans le même IF, à décrémenter de −1...

LA TECHNIQUE DES FLAGS

Un FLAG (=drapeau) est une sorte de mouchard, on sait alors si le déroulement du programme est passé par tel endroit, si telle condition IF s'est trouvée vérifiée. Pour plus de clarté, c'est un nom de variable dont la première lettre est F, tel FB, F1, F2 etc... Le plus souvent, on ne leur attribue que deux valeurs, 0 et 1.

On "pose" un FLAG généralement après un THEN ou un ELSE, ou dans un sous-programme appelé par GOSUB.

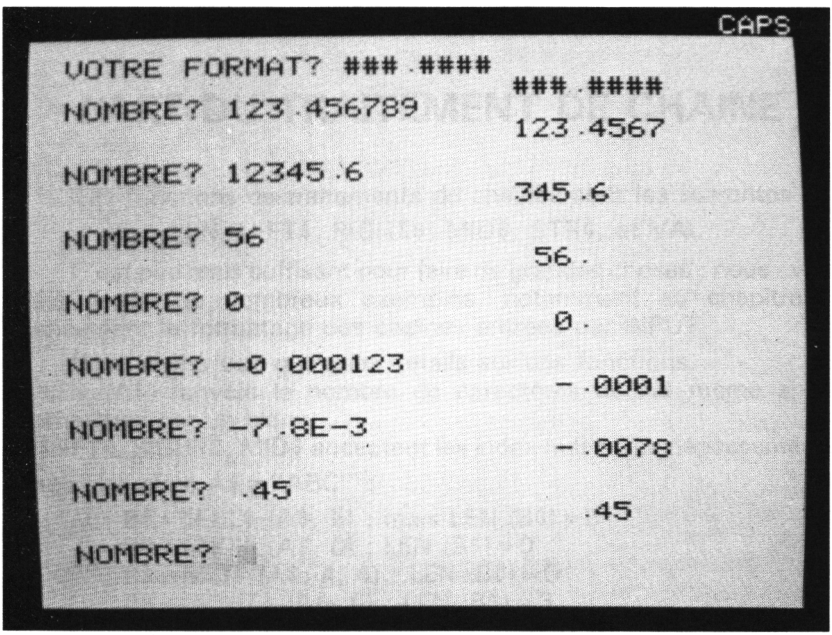
Note : Pour tester un FLAG, vous n'êtes pas obligé sur ORIC d'écrire IF FT=1 ou IF FT>0, vous avez le droit à cette simplicité :

IF FT THEN...

Il est fréquent d'utiliser des FLAGS provisoires lors de la mise au point d'un programme, donc en concurrence ou en complément de la fonction TRON.

On peut aussi incrémenter des FLAGS, c'est-à-dire qu'au lieu du « THEN FT = 1 ». Un exemple : vous vous souvenez des sécurités d'entrées du chapitre VI où une ligne redemandait le même INPUT en cas d'erreur : avant chaque GOTO à la ligne INPUT, ajoutons par exemple, F1 = F1 + 1. Pour une autre entrée de données mettons dans la ligne de sécurité F2 = F2 + 1 etc... Et puis plus loin :

```
 830 'COMPTAGES DES ERREURS
 840 TE=F1+F2+F3
 850 IF TE>=5 AND TE<10 THEN PRINT"DEJ
A";TE;"ERREURS:FAITES ATTENTION !"
 860 IF TE>=10 THEN PRINT"ALLEZ VOUS R
EPOSER UN PEU..."
```



Chapitre IX

L'ART DU TRAITEMENT DE CHAINE

Les fonctions de traitements de chaînes sont les suivantes :

LEN, LEFT$, RIGHT$, MID$, STR$, et VAL

C'est peu mais suffisant pour faire de grandes choses ; nous avons déjà donné de nombreux exemples, notamment au chapitre VI, concernant le formatage des chaînes entrées par INPUT.

Nous allons voir quelques détails sur ces fonctions.

- LEN (A$) renvoie le nombre de caractères de A$ même si A$ commence par un blanc.
- LEFT$, RIGHT$, MID$ acceptent les index nuls et les dépassements.

Exemples avec A$ = "ABC" :

```
B$ = LEFT$ (A$, 5) ; mais LEN (B$) = 3
B$ = LEFT$ (A$, 0) ; LEN (B$) = 0
B$ = MID$ (A$, 4, 4) ; LEN (B$) = 0
B$ = RIGHT$ (A$, 6) ; LEN (B$) = 3
B$ = RIGHT$ (A$, 1.5) ; LEN (B$) = 1 (B$ = C)
```

Par contre, il y aura un message d'erreur avec un index négatif ; c'est la moindre des choses.

Ces fonctions sur l'ORIC sont donc d'une tolérance exceptionnelle. La seule ombre au tableau est que MID$ ne peut servir qu'à lire et non à écrire.

- STR$ (N) considère le nombre N comme il serait représenté à l'écran,

c'est-à-dire avec un blanc devant, mais attention aux nombres négatifs ! Le signe "-" remplace le blanc.

Exemple :

A$ = STR$ (5), LEN (A$) = 2
A$ = STR$ (−5), LEN (A$) = 2 (et non pas 3)

Attention également aux nombres inférieurs à 0.01.

N = 0.009 ; A$ = STR$ (N) ; A$ = "9E−03" ; LEN (A$) = 6 avec N = −.009 ; A$ = "9E−03" et LEN (A$) = 6

• VAL (A$) considère les premiers caractères rencontrés à l'exception des blancs ; ainsi :

A$ = "34ABCD" ; VAL (A$) = 34
A$ = "ABCD34" ; VAL (A$) = 0
A$ = " 34" ; VAL (A$) = 34
A$ = "12,34" ; VAL (A$) = 12
A$ = "48 + 4" ; VAL (A$) = 48
A$ = "−48.56" ; VAL A$ = −48.56

LA CONCATENATION

C'est fusionner plusieurs chaînes bout-à-bout afin d'en faire une plus longue.

A ce sujet, il faut bien faire la différence d'écritures avec une concaténation par l'ordre PRINT.

Exemple :

A$ = "CONC":B$ = "ATENA":C$ = "TION"
PRINT A$;B$;C$ (des points-virgules)

donnera à l'écran CONCATENATION, tandis que l'on peut aussi écrire

T$ = A$ + B$ + C$:PRINT T$ (des signes +)

Notez bien la nuance. Si vous mettez des ; à la place des + et inversement, il y aura « Syntax Error ».

LA SUBSTITUTION

C'est remplacer une partie d'une chaîne par une chaîne plus courte. Supposons que dans la chaîne TX$ = "ABCDEF" nous voulions imposer le mot MO$ = "XY" à partir du troisième caractère de TX$, afin d'obtenir TX$ = "ABXYEF".
Avec l'ORIC, ce n'est pas facile.

Comme cette pratique peut être courante, faisons un sous-programme utilitaire avec un numéro de ligne élevé (58000). A noter que ce module (très court) va remplacer une fonction BASIC que l'ORIC ne possède pas, à savoir le MID$ interactif. Ce petit utilitaire va être le premier d'une longue série.

```
10 'SUBSTITUTION DE CHAINE:DEMONST.
15 CLS
20 INPUT"TX$";TX$
30 INPUT"MO$";MO$
40 INPUT"POSITION K DANS TX$";K
50 GOSUB 58000
60 PRINT"RESULTAT:";TX$
70 PRINT:GOTO 20
90 END

58000 'SUBSTITUTION DE CHAINE
58010 'TX$=CHAINE RECEPTIVE
58020 'MO$=CHAINE IMPOSEE
58030 'K=POSITION DE MO$ DANS TX$
58040 K=K-1
58050 TX$=LEFT$(TX$,K)+MO$+RIGHT$(TX$
,LEN(TX$)-K-LEN(MO$))
58060 RETURN
```

Tout est dans la ligne 58050. Après l'avoir essayée, effacez les lignes 10 à 90 et enregistrez le programme 58000 – 58060 sur une cassette, sous le nom « SUBSTIT ». Puis faites NEW.

LA FONCTION INSTR

Ne la cherchez pas dans le manuel, l'ORIC ne la possède pas, du moins pas encore...

C'est en quelque sorte l'inverse de la précédente car elle renvoie la position K du mot MO$ dans le texte TX$.

Il s'agit d'un très court sous-programme qui va être appelé en maintes occasions, notamment par d'autres sous-programmes utilitaires que nous allons décrire plus loin.

```
51000 'INSTR
51010 'Place K du mot MO$ dans le tex
te TX$
51030 K=0:FOR I=1 TO LEN(TX$):IF MID$
(TX$,I,LEN(MO$))=MO$ THEN K=I
51040 NEXT:RETURN
```

Mo$ peut être un blanc ou commencer par un blanc, mais à la condition qu'il ne soit pas entré par INPUT MO$; en effet, rappelons que si une chaîne entrée par INPUT débute par un blanc, celui-ci n'est pas pris en compte (c'est le cas de presque tous les micro-ordinateurs), mais vous pouvez écrire MO$ = " " ou MO$ = " A".

REPONSE A UN MENU

Ce n'est pas une nouvelle fonction mais un petit utilitaire très pratique et à usage fréquent, il utilise le INSTR précédent.

Nous avions illustré au chapitre VI une page d'écran représentant un menu de divers jeux (lignes 1500-). Nous avions dû légender chaque jeu par un chiffre, et ajouter quatre lignes de BASIC pour proposer, vérifier et enterriner le choix. Faisons mieux :

On pourra légender chaque titre par un *caractère quelconque*, lettre, chiffre, signe de ponctuation, etc..., et ce petit sous-programme se chargera de tout. C'est très utile lorsqu'un programme propose plusieurs menus d'options ; c'est la majorité des cas.

```
 100 'REPONSE MENU PAR MODULE
 110 CLS:PRINT@16,5;"M E N U"
 120 PRINT@12,10;"F-FORMULE 1"
 130 PRINT@12,12;"M-MORPION"
 140 PRINT@12,14;"*-ETOILES"
 150 PRINT@12,16;"S-SLALOM"
 160 PRINT@12,18;"Q-QUITTER"
 200 TX$="FM*SQ":GOSUB 55000
 210 ON K GOSUB 12000,16000,20000,2500
0,290
 290 CLS:END

 55000 'REPONSE A UN MENU
 55010 'Entre TX$ et MO$,renvoie K
 55020 LT=LEN(TX$):TT=15-LT:IF TT<2 TH
EN TT=2
 55030 PRINT@TT,24;"REPONSE(";:FOR I=1
 TO LT-1:PRINTMID$(TX$,I,1)+",";:NEXT
 55040 PRINTMID$(TX$,LT,1)+") ->";:GET
 MO$
 55050 GOSUB 51000:IF K=0 THEN PRINTCH
R$(14):GOTO 55000
 55060 RETURN
```

Remarquez la simplicité d'écrire de la ligne 200 : TX$ rassemble toutes les légendes, *sans espaces ni ponctuation*. C'est le module 55000 qui se charge d'écrire en bas de l'écran "Réponse (F, M, *, S, Q)" avec centrage automatique, le GET, la validation de la réponse, en renvoyant le numéro d'ordre K de l'option choisie. Pour un autre menu proposant OUI, NON et QUITTER, on écrirait simplement TX$ = "ONQ":GOSUB 55000:Le grand confort...

LA FONCTION PRINT USING

Cette grande absente du BASIC ORIC est maintenant disponible. Mais expliquons d'abord le rôle de cette fonction que beaucoup d'entre vous peuvent ignorer :

C'est le formatage imposé pour afficher des nombres. Sans PRINT USING les nombres s'inscrivent à l'écran (ou sur imprimante) en partant de la gauche, et de ce fait dans une colonne de nombres, les chiffres des unités et les points décimaux ne sont pas du tout les uns sous les autres, ce qui n'est guère présentable. Supposons que vous imposiez le format "# # # .# #".

Tous les nombres vont être tronqués avec deux décimales, le point décimal sera toujours au même endroit. En somme, chaque nombre est présenté sous forme de *chaîne de longueur constante*, avec ici le point décimal en quatrième position.

Notre PRINT USING est un sous-programme (GOSUB 52000) qui demande deux renseignements : le format d'écriture PU$ (de PRINT USING) et le nombre NB à traiter. Il retourne la chaîne NB$ formatée selon votre PU$.

Lorsque vous définissez votre PU$, vous pouvez utiliser *n'importe quel caractère* de remplissage, seul le "." est réservé au point décimal, bien que l'on ait le droit de fixer un format pour entiers, genre PU$ = "AAA" (nous avons utilisé le caractère "#" (dièse) car c'est le caractère obligatoire en BASIC MICROSOFT).

Ce sous-programme peut sembler assez long et complexe (il est néanmoins d'une exécution très rapide), car il tient compte de tous les cas particuliers concernant PU$ et NB. L'auteur vous lance même le défit de mettre ce module en défaut...

```
300 'DEMONSTRATION PRINT USING
310 CLS:INPUT"VOTRE FORMAT";PU$
320 PRINT TAB(25);PU$
330 INPUT"NOMBRE";NB
340 GOSUB 52000
350 PRINT TAB(25);NB$
360 GOTO 330
390 END
```

```
 52000 'PRINT USING     ####.##
 52010 'PU$=FORMAT+NB=Nombre:Retourne
NB$ formate
 52030 TX$=PU$:MO$=".":GOSUB 51000:KF=
K
 52040 LT=LEN(PU$):LE=K-1:LD=LT-K
 52050 IF K=0 THEN LE=LT:LD=0:IF ABS(N
B)<1 THEN NB=0
 52060 NB$=STR$(NB):L=LEN(NB$)-1:IF NB
<0 THEN L=L+1
 52065 NB$=RIGHT$(NB$,L)
 52070 IF ABS(NB)<.01 AND NB<>0 THEN G
OSUB 52500:GOTO 52130
 52080 TX$=NB$:GOSUB 51000
 52090 IF K=0 THEN E$=RIGHT$("       "+
NB$,LE):D$=LEFT$("       ",LD):GOTO 521
10
 52100 E$=LEFT$(NB$,K-1):D$=RIGHT$(NB$
,L-K)
 52110 IF KF=0 THEN NB$=RIGHT$("
"+E$,LT):GOTO 52130
 52120 NB$=RIGHT$("       "+E$,LE)+"."+
LEFT$(D$+"       ",LD)
 52130 RETURN
 52500 'FORMATAGE PETITS NOMBRES
 52510 NZ=VAL(RIGHT$(NB$,2))-1
 52520 CH$=STR$(NB*100000)
 52530 CH$=RIGHT$(CH$,LEN(CH$)-1)
 52540 D$=LEFT$("000000",NZ)+CH$
 52550 D$=LEFT$(D$,LD)
 52560 E$=LEFT$("       ",LE)
 52570 IF NB<0 THEN E$=RIGHT$("      -"
,LE)
 52580 NB$=E$+"."+D$
 52590 RETURN
 52999 '---------------
```

Nous vous conseillons de taper ce programme à la suite des autres de ce chapitre. Après avoir effacé les programmes de démonstrations, vous enregistrerez le reste sur une cassette sous le titre « BASIC CHAINE ».

Analysons ce module PRINT USING :
— Il commence par analyser le format PU$; sa longueur totale LT, la position K du point décimal (KF = K), la longueur entière LE et la longueur décimale LD.
— Lignes 52060-52065 : le nombre NB est transformé en NB$, et en éliminant le blanc de gauche, sauf si NB est négatif.
— Ligne 52070 : ORIC transforme les nombres inférieurs à .01 (en valeur absolue) en notation exponentielle ; ainsi .00723 devient 7.23E−03. D'où un détour au sous-module 52500− que nous verrons plus loin.
— Toujours par notre fonction INSTR, on recherche la position du point décimal dans NB$.
— On définit alors la chaîne partie entière E$ de largeur LE, et la chaîne décimale D$ de longueur LD. Il ne reste plus qu'à concaténer E$ + "." + D$.

Le sous-module petits nombres (52500−) : on commence par définir le nombre de zéros NZ qu'il y aura à droite du point décimal. Le nombre NB est multiplié arbitrairement par 100.000 afin de supprimer la notation « scientifique » pour lire les chiffres que l'on va placer après les NZ zéros.

Après concaténations, le "7.23E−03" est transformé en ".00723" formaté selon le PU$, un NB$ prêt à l'emploi.

Ce sous-programme conduit à des résultats spectaculaires, en regard de la simplicité d'écriture, en ce qui concerne les programmes d'éditions (c'est-à-dire sur imprimante) ; en effet, chaque ligne de programme est de la forme (un exemple) :

```
LPRINT NB$ (I);SPC (5);:NEXT
```

ce qui assure une tabulation sûre et impeccable sur le plan de la présentation et de la lisibilité de ce tableau de nombres.

Note : Certains BASIC étendent la fonction PRINT USING au formatage des chaînes ; le caractère "#" est alors remplacé par "&" (= ET commercial ou AMPERSAND). Dans la pratique, il faut avouer que l'on ne s'en sert pratiquement jamais... Nous ne l'avons donc pas développé bien que cela soit possible, et plus simplement que pour les nombres. Si cela vous dit...

Ne vous contentez pas de recopier scrupuleusement les programmes utilitaires de ce chapitre, analysez-les, et ce afin de vous familiariser avec ces triturations de LEN, LEFT$, et d'autres. En effet, vous serez un jour confronté à un problème très personnel, très spécial, qui sera résolu très facilement grâce à ces fonctions de traitement de chaînes.

LE TRI ALPHABETIQUE

Les symboles "<" et ">" sont également applicables aux caractères. En ce cas, un micro-ordinateur considère les codes ASCII de ces caractères. Examinez la table de correspondance de la page 263 du manuel ATMOS : on y remarque l'ordre suivant :

Espace (32), les chiffres 0 à 9 (48 à 57), les lettres majuscules A à Z (65 à 90) et les lettres minuscules a à z (97 à 122). Pour une même lettre, on passe de majuscule à minuscule en ajoutant 32. (ORIC respecte le CODE ASCII international).

On peut donc dire que A<D puisque 65<68. De ce fait, on peut faire du tri alphabétique sur une liste de noms ; si les premières lettres sont identiques, le BASIC considère alors les deuxièmes, etc...

Exemples :

RENAUD<RENE;ANDRE<ANDREE mais
ZOE<accroc; "1000"<ZERO;" 9"<"5"

Pour effectuer un tri alphabétique, on loge en vrac tous les noms dans un tableau DIM ; au cours du classement, ces noms vont changer de place, c'est-à-dire d'index.

La méthode classique est celle dite du "tri à bulles", elle est très lente : elle consiste à comparer les noms deux-à-deux, les permutant si nécessaire, en repassant le tableau de haut en bas de nombreuses fois jusqu'à ce que tout soit dans l'ordre. La "bulle" est une variable relais où l'on place un des deux noms afin de faire la permutation.

L'auteur a imaginé une autre méthode de tri qui est trois fois plus rapide que la classique, sans présenter un quelconque inconvénient par rapport à cette dernière. Nous l'avons baptisée "tri en escalier" :

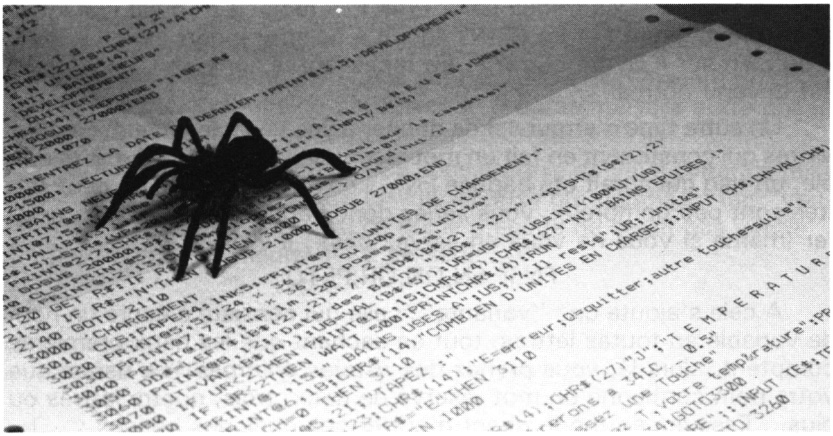
```
400 'DEMONSTRATION DE TRI RAPIDE
410 'REMPLISSAGE DE LA LISTE
420 NB=30:DIM P$(NB):CLS
430 FOR I=1 TO NB:READ P$(I):NEXT
440 DATA Z,Y,X,W,V,U,T,S,R,Q
450 DATA 9,8,7,6,5,4,3,2,1,0
460 DATA P,O,N,M,L,K,D,C,B,A
470 GOSUB 59000:'TRI
480 PRINT:FOR I=1 TO NB:PRINT P$(I);"
";:NEXT:PRINT
490 PRINT@4,10;"CE CLASSEMENT A PRIS
5.5 secondes":END
```

```
59000 'TRI EN ESCALIER;M.ARCHAMBAULT
59010 FOR J=1 TO NB:E$=P$(J)
59020 FOR I=J TO NB
59030 IF P$(I) <= E$ THEN E$=P$(I):N=
I
59040 NEXT
59050 P$(N)=P$(J):P$(J)=E$:NEXT
59090 RETURN
```

Le principe du tri est le suivant : on considère d'abord un "étalon" E$, arbitrairement, celui en haut du tableau, P$ (1). Puis on le compare à tous les autres ; si on rencontre plus petit, c'est ce dernier qui devient E$, on le repère par son index N, et ainsi de suite jusqu'en bas du tableau. Le E$ "vainqueur" est placé à l'index 1 et celui qui était en 1 passe à l'index N (une permutation).

On reprend une deuxième passe mais en partant une "marche" plus bas, E$ = P$ (2). Le plus petit E$ est logé en index 2. On repart depuis l'index 3, et ainsi de suite.

Le tableau à classer était d'une difficulté voulue. Nous l'avons utilisé également pour un test comparatif avec le tri à bulle ; il a fallu 16,3 s contre 5,5 s pour le nôtre, soit un rapport d'environ 3.



Chapitre X

LES TESTS ANTI-BUGS

Il y a deux sortes de bugs (bugs = erreurs dans un programme), les francs qui font planter le programme et avec ou sans message d'erreur ; et les sournois qui font croire que tout se passe bien mais conduisent de *temps à autre* à des résultats erronés (mais non flagrants...) ; de ce fait, on ne les remarque que bien plus tard... Heureusement, ces derniers sont beaucoup plus rares.

Avant d'indiquer les méthodes de dépistages, il est préférable de passer en revue quelques types de bugs classiques et pourtant d'apparence anodine. Donc, nous ne parlerons pas de l'étourderie banale et criante genre caractère oublié, caractère à la place d'un autre, etc...

QUELQUES ERREURS DE SYNTAXE PLUS SUBTILES

On est assez décontenancé lorsque l'écran signale une erreur de syntaxe dans une ligne apparemment correctement écrite. Notre exemple préféré, et authentique, était de la forme :

510 LPRINT AB$

C'est pourtant correct ! Heureusement, le hasard a voulu que l'on s'apercoive que ce programmeur débutant avait tapé sa ligne de la façon suivante :

510 L? AB$

En effet, ''?'' se traduit par « PRINT », mais ''L?'' c'est une lettre quelconque avant l'ordre PRINT, qui n'a aucun rapport avec LPRINT, d'où bien sûr « Syntax Error ». En repassant la ligne en CTRL A tout est devenu normal.

Un autre type d'erreur est de donner à une variable un nom en deux lettres qui constituent en fait un mot réservé au BASIC ORIC, par exemple, un flag qui aurait été baptisé FN... Les mots réservés en deux lettres sont peu nombreux, vous seriez donc impardonnable de les ignorer (même si vous ne vous en servez pas) ; il y a :

FN,IF,LN,ON,OR,PI,TO

A cela s'ajoute une ''variante'', celle qui consiste à écrire un nom de variable en toutes lettres, tout en sachant que les deux premières comptent, alors là, vous prenez des *risques énormes !* Le risque que votre nom contienne un mot réservé de deux, trois, quatre lettres ou plus... Essayez en mode direct d'entrer ces quelques bêtises :

```
NOTE = 45 (NOT)
RENDU$ = ''A'' (END)
CHIFFRE$ = ''9'' (IF et FRE)
```

Par contre, LETTRE$ = ''A'' est accepté, mais attention ! C'est TRE$ qui est égal à ''A''... et faire PRINT LETTRE$ plantera (fonction LET...).

Dans notre premier exemple, NOTE pourra être remplacé sans dommage par NO ou NTE.

Il y a aussi des fautes de frappe difficiles à remarquer sur l'écran et sur un listing d'imprimante parce que certains caractères se ressemblent. A titre d'exemple, tapez :

''K'' et ''k'', ''l'' et ''1''

A noter aussi que certaines imprimantes ne barrent pas le zéro, et pour le distinguer de la lettre ''O'', il faut un œil de lynx.

LA SYNTAXE DES CONDITIONS

Tout d'abord, cette règle super importante dont l'oubli est hélas classique :

• lorsqu'une condition IF n'est pas vérifiée, le déroulement du programme passe à la ligne suivante (la suite après IF est ignorée). Donc, si à la suite d'une ligne comportant IF, vous ajoutez un autre pas de programme, séparé par '':'', celui-ci ne sera exécuté que si la condition IF qui le précède a été vérifiée.

En ce qui concerne AND et OR, il y a le danger de se laisser abuser par le langage courant. Un exemple : vous vous dites ''je veux qu'il y ait le son PING pour deux valeurs de N, à savoir 6 *et* 7'' et vous seriez machinalement tenté d'écrire :

IF N = 6 AND N = 7 THEN PING

Condition irréalisable ! Il fallait mettre OR et non AND. Notre exemple est volontairement très simpliste, mais sachez que pour des cas complexes ce genre de confusion nous guette. AND et OR exigent un choix très réfléchi, car en cas de confusion le programme fera autre chose que prévu et sans donner de message d'erreur !

N'utilisez pas la fonction ELSE sous prétexe de vous éviter de faire une nouvelle ligne. Car ELSE a un rôle *exclusif* qui signifie en fait "vérifier ce qui suit dans le cas où ce qui le précède ne serait pas vérifié". On peut mettre plusieurs ESLE en série.

```
 10 'EFFET EXCLUSIF DE ELSE
 20 CLS
 30 INPUT"N[5]";N
 40 INPUT"P[3]";P
 50 IF N=5 THEN PING ELSE IF P=3 THEN
SHOOT ELSE IF N=P THEN ZAP ELSE FS=1
 60 IF FS AND N+P=12 THEN PRINT"GAGNE
!" ELSE IF FS THEN PRINT"PERDU..."
 70 PRINT:FS=0:GOTO 30
 90 END
```

Ce programme débouche sur cinq "conséquences" :

PING, SHOOT, GAGNE, ! et PERDU

Or, par l'effet exclusif des ELSE, *une seule* conséquence n'est possible, et ce, quelles que soient les valeurs de N et P. Le programme conditionnel est court (deux lignes), mais était-ce le but recherché ? Remarque importante : les ELSE ont aussi introduit des *priorités* , ainsi pour N = 5 et P = 5 il y aura PING, le ZAP de l'égalité ne peut être vérifié ; pour N = 9 et P = 3 on aura SHOOT et non pas GAGNE.

Voilà pourquoi un programme peut parfois avoir des réactions "bizarres", parce que l'on a utilisé ELSE simplement pour enchaîner sur un autre IF. *ELSE est à manier avec précautions.*

Une petite parenthèse : nous ne pouvions tout écrire dans la ligne 50 (les 78 caractères maxi), d'où ce "flag silence" FS afin de poursuivre en ligne 60 ; et sans oublier de le remettre à zéro en ligne 70.

LA LOCALISATION DE ZONE D'ERREUR

Si ce programme plante avec un message d'erreur, l'ORIC indique en plus le numéro de la ligne : l'erreur est donc soit sur cette ligne, soit *en amont*. C'est dès lors assez facile de mieux cerner l'endroit coupable. En revanche, lorsque le fonctionnement est défectueux, mais sans

message d'erreur, il faut d'abord identifier la zone coupable : la méthode la plus élégante consiste à tronçonner le programme çà et là, par des lignes comportant seulement la fonction STOP.

Prendre des lignes se terminant en 9 (pour le repérage) et les noter sur une feuille ou mieux sur le listing d'imprimante.

Exemple :

499 STOP

Puis lancer le programme : il va s'interrompre à chaque STOP rencontré avec le message « BREAK IN 499 ».

Mais jusque là, rien d'anormal ne s'est produit, c'est donc plus loin. En mode direct, tapez CONT et Return (=condition), Apparaît alors « BREAK IN 629 » (un autre STOP que vous aviez mis en 629). Vous constatez une anomalie ; il y a donc un bug entre les lignes 500 et 620. Mais laquelle ? *Interrogez la RAM* sur les valeurs de ses variables en cours. Tapez par exemple en mode direct :

PRINT N,C$,I et Return

ou bien FOR J=1 TO 12:PRINT PR$ (J):NEXT

afin de visualiser le contenu d'un DIM ; ou encore

LIST 500-620

Vous pouvez poursuivre par CONT, mais attention, certaines précautions s'imposent :

POUR QUESTIONNER LA RAM

Vous avez droit en mode direct à la majorité des mots BASIC, mais le danger est d'effacer toutes ces variables en RAM. Trois choses sont interdites : RUN, RESET, et *modifier quoi que ce soit au programme*. Ajouter, effectuer ou modifier une ligne au programme a pour effet de ré-initialiser toutes les variables. Si vous tapiez CONT, vous auriez un message d'erreur (CAN'T CONTINUE ERROR).

En revanche, après un arrêt par STOP (ou par CTRL C), vous avez le droit de modifier la valeur d'une variable, par exemple N=15 et return, et de repartir ensuite soit par CONT ou encore par GOTO 500.

En somme, STOP est un CTRL C programmé avec mise en mémoire du numéro de ligne suivant ; CONT est l'équivalent de GOTO ce numéro de ligne.

Le problème est épineux, vous modifiez à plusieurs reprises le programme, que vous relancez par RUN, et vous trouvez fastidieux de retapez les mêmes questions en mode direct ; alors programmez-les :

629 PRINT N,C$,I,CD:STOP

LE DEPISTAGE DU BUG

Vous avez localisé le groupe de lignes farceuses, mais il s'agit en fait d'un enchevêtrement de GOTO (bien fait pour vous !) dont la plupart sont consécutifs à des IF. Plutôt que d'attraper une migraine, utilisez la fonction TRON (= contraction de TRACE ON), elle est faite pour cela. Elle aussi, mettez-là sur une ligne provisoire dont le numéro se termine par 9. A la fin de la zone dangereuse, n'oubliez pas de programmer une autre ligne "en 9" pour TROFF (= TRACE OFF) ; qui annule TRON.

Grâce à TRON, chaque numéro de ligne sur lequel "passe" le programme apparaît entre crochets. Si une ligne comprend trois instructions différentes séparées par des ":", ce numéro de ligne apparaît trois fois. Cette écriture est assez envahissante à l'écran ; donc programmez le TROFF le plus tôt possible.

Un petit piège : si dans la zone TRON figure un CLS, effacez-le provisoirement ou transformez sa ligne en REM par un apostrophe ('). Sinon ce CLS effacerait ce qu'aurait déjà écrit TRON, vous faisant croire qu'il n'est pas passé par ces lignes...

Les bugs du "troisième degré" : même les indications fournies par TRON ne vous donnent pas la clef du mystère. Prenons l'exemple du petit programme précédent sur l'effet exclusif de ELSE :

Ajoutez la ligne

```
05 TRON    et modifiez la n° 20
20' CLS    , et RUN
```

En exécutant le programme, vous remarquerez que 70 apparaît 3 fois (3 instructions) mais la fameuse ligne 50 n'apparaît qu'une seule fois ; vous n'êtes donc pas renseigné pour autant quant à ce qu'il s'y passe...

Alors les grands moyens : LES FLAGS. C'est l'arme absolue, mais à n'utiliser qu'en dernier recours car elle "défigure" les lignes de programmes, qu'il faudra ensuite remettre dans leur forme initiale.

Le plus souvent cela consiste à insérer un flag après chaque THEN, ou après un ELSE s'il n'est pas suivi d'un THEN.

Exemple :

```
IF... THEN PING : F1 = 1:ELSE...E2 = 1...
```

Il suffira ensuite de questionner ces flags en RAM.

Nous n'avons encore jamais rencontré un "bug rebelle" résistant à ces mouchards.

ERREURS DU "PASSE"

Il s'agit des lignes "mortes", désaffectées, obsolètes, recopiées ailleurs, et que l'on a oublié d'effacer lorsque l'on triturait le programme pour sa mise au point. Dans un petit programme de quinze lignes, cette

étourderie est quasi impossible, mais quand il y a plusieurs centaines de lignes, c'est banal.

Souvent, ces lignes mortes sont sans conséquences pour le bon déroulement du programme, parce qu'elles ne sont jamais lues ou encore parce qu'elles refont ce qui a déjà été fait. Il arrive que l'on s'en aperçoive par hasard, alors que le programme fonctionnait quotidiennement depuis des mois !

On les débusque généralement lorsqu'une fois le programme mis au point, on décide de remettre à jour l'organigramme. A ce titre, on comprend bien le pourquoi des numéros de lignes "en 9"...

LE DEPISTAGE DES BUGS DU FUTUR

Un programme peut s'abîmer accidentellement, un défaut de lecture de la cassette, on a (sans le savoir) tapé un nombre quelconque et Return, et on a ainsi effacé une ligne.
Voici la parade :

Votre programme est bien au point et fonctionne à merveille. Faites RESET puis tapez :

? 37629 – FRE (0) , et Return (ATMOS)

A l'écran apparaît le nombre d'octets qu'occupe votre programme : NOTEZ-LE au crayon sur la cassette, ainsi que sur le dernier listing d'imprimante (avec la date).

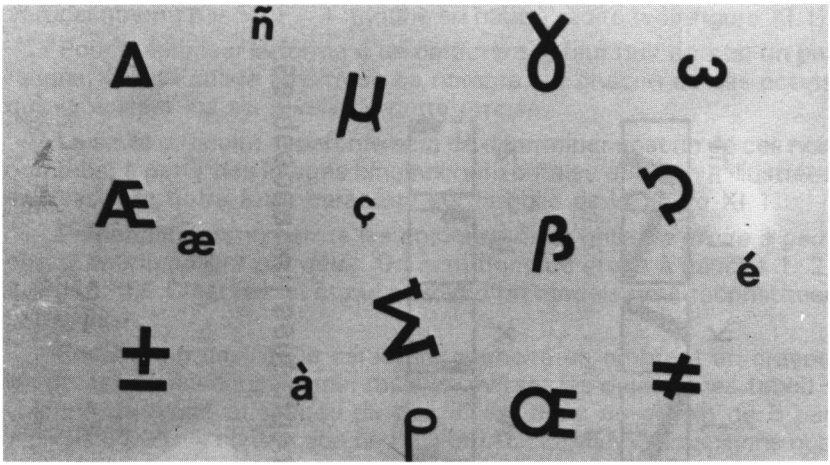
Pour l'ORIC 1, il faudra modifier ce nombre d'octets disponibles à vide, sinon faire

? FRE (0) , et Return

qui indique ce qu'il reste de disponible.

En cas de doute sur un programme que l'on vient de recharger, il suffit de reposer cette même question afin de s'assurer que le nombre affiché est identique à celui noté à l'origine.

Note : La première série de l'ATMOS affiche souvent à tort « ERRORS FOUND » (erreurs trouvées) en fin de chargement par CLOAD. Il est alors rassurant de demander son nombre d'octets.



Chapitre XI

CREONS DE NOUVEAUX CARACTERES

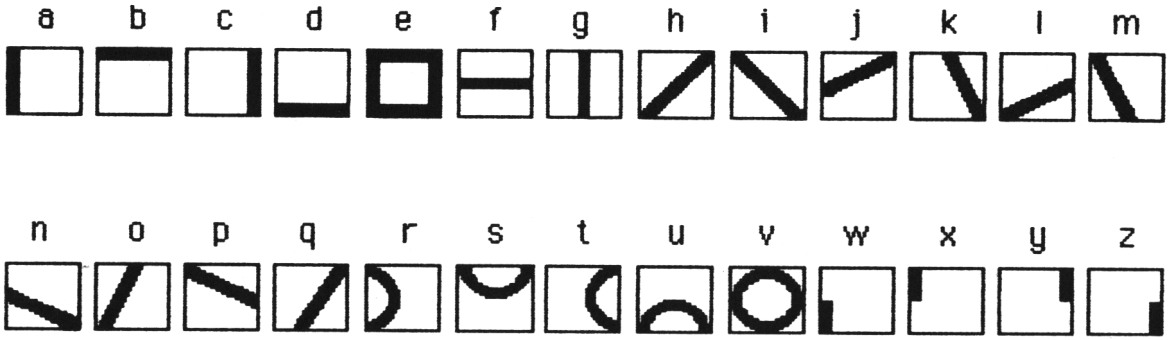
C'est une chose tellement facile et rapide que l'on aurait tendance à en abuser. Mais à une condition : ne pas se référer au manuel d'origine qui complique les choses bien inutilement (plus une erreur dans l'exemple...).

Nous allons d'une part créer les caractères AZERTY accentués, des caractères semi-graphiques afin de pouvoir dessiner en mode TEXT ou des animations rapides en HIRES, mais aussi fabriquer des caractères à la demande telles certaines lettres grecques.

Bien sûr, nous allons vous livrer ces sous-programmes tout-prêts, mais nous tenons surtout à vous apprendre à créer les vôtres, et pour cela, commencez par oublier ce que vous avez lu (ou compris) dans le manuel.

LE PRINCIPE

Chaque case de l'écran est une grille de huit ''points'' de haut sur six ''points'' de large. Ces petits points ou mini-cases s'appellent des ''pixels''. La colonne de droite n'est jamais utilisée sinon les caractères se toucheraient sur une même ligne. De même, la rangée du bas n'est pas utilisée, et ce, afin de séparer les lignes ; deux exceptions, la lettre ''y'' (vous l'avez remarqué avec le y de ''ready'') et le trait



Redéfinition des lettres minuscules en éléments graphiques juxtaposables.
Exemple simple, "vfv" dessine une haltère.

Figure XI 2

vertical obtenu par SHIFT + touche en haut à droite (voir figure XI 1).

Pour mémoriser la forme d'un caractère, il faut huit octets, un par rangée. C'est l'image binaire de ce nombre sur chacun de ces octets qui va éclairer les six pixels de cette rangée.

La seule difficulté apparente sera de déterminer chacun de ces huit nombres, à partir des images binaires horizontales que l'on a illustrées en dessinant notre futur caractère sur la grille de la figure XI 1.

Primordial : on numérote les colonnes de la grille *de droite à gauche et en multipliant par deux*. On écrit donc de droite à gauche 1, 2, 4, 8, 16, 32. C'est facile, et nul besoin d'un modèle pour reconstituer cette grille.

Ensuite, on dessine le caractère souhaité en ombrant au crayon les petites cases nécessaires, mais en évitant les deux zones tabou : colonne de droite et rangée du bas. C'est donc un dessin de 5 par 7 pixels (PIXEL = contraction de PICTURIAL ELEMENT). Imaginons que l'on veuille faire la lettre grecque oméga (symbole de l'Ohm).

A présent, faisons les comptes par rangée en commençant naturellement par le haut :

— Rangée n° 1 : sont noircies les cases 16, 8 et 4 : 16 + 8 + 4 = 28, et d'une.
— Rangée n° 2 : on totalise les cases 32 + 2 = 34.
— Idem pour les rangées n° 3, 4 et 5.
— Rangée n° 6 : ce sont les cases 16 + 4 = 20.
— Rangée n° 7 : on a 32 + 16 + 4 + 2 = 54.
— Rangée n° 8 : elle est vide, donc = 0.

Les huit octets pour faire la lettre oméga sont donc :

28, 34, 34, 34, 34, 20, 54, 0

Maintenant, il faut stocker en mémoire.

LA MISE EN MEMOIRE DES CARACTERES

Le système des ORIC présente une originalité (très pratique), rare chez ses concurrents. A la mise sous tension, sa mémoire résidente ROM est "recopiée" en mémoire vive RAM ; de ce fait, elle est modifiable par des POKE ou DOKE. Mieux, le fabricant indique dans le manuel où trouver chacune des données.

Où loger notre représentation de l'oméga ? A la place d'un caractère dont on ne se sert presque jamais, mais qu'on puisse obtenir par le clavier. Que diriez-vous du "&" ? (= Ampersand ou ET commercial). Où le situer dans la RAM ? Facile : à la page 263 du manuel ATMOS, nous lisons que son code ASCII est 38. D'autre part, nous savons que les caractères de l'ORIC sont définis à partir de l'adresse 46080. Comme il y a 8 octets par caractère et qu'ils sont dans l'ordre des codes ASCII, notre & débute à l'adresse :

$46080 + (38 \times 8) = 46384$ (ATMOS)

On va lui « POKER » nos huit octets de l'oméga :

```
10 'LETTRE OMEGA
20 FOR I=0 TO 7:READ A
30 POKE 46384+I,A:NEXT
40 DATA 28,34,34,34,34,20,54, 0
50 END
```

Faites RUN et tapez ''&'', notre lettre apparaît. Et cette mise en mémoire est solide ! Seule une action sur le bouton *RESET* pourrait remettre en place le caractère d'origine. Vous pouvez faire RUN 50, CLOAD '' '', CLEAR, même *NEW*, notre oméga est toujours à la place du &.

Une petite parenthèse : il y a un caractère graphique qui ne possède pas de touche au clavier, faites PRINT CHR$ (96).

C'est le signe du ''copyright''.

LES CARACTERES AZERTY ACCENTUES

Lorsque vous frappez une touche au clavier, *vous ne commandez pas un caractère mais un code ASCII*, un nombre compris entre 1 et 151 (voir pages 262 à 265 du manuel), lequel va aller chercher à partir de l'adresse 46080 l'image à dessiner à l'écran.

La touche du clavier ORIC marquée ''Q'' équivaut à CHR$ (80) ; 80 = code ASCII de Q, vous ne pouvez pas lui faire commander CHR$ (65), 65 étant le code du A. Donc désolé pour les dactylos, mais le clavier des ORIC restera un QWERTY. En revanche, nous pouvons faire apparaître certains caractères spécifiques aux claviers AZERTY français. Pas n'importe où ! Mais aux codes ASCII de la norme internationale. Pourquoi ?

La plupart des imprimantes possèdent une option ''nationalités de claviers''. Par exemple, on a le choix entre USA, Angleterre, France, Allemagne, Danemark, Suède, Italie et Espagne. Les différences portent sur une dizaine de codes ASCII environ. Il s'agit de caractères et d'accents spéciaux. Ce choix de nationalités s'effectue soit par une batterie de petits commutateurs (''switches''), soit par un ordre spécial à la suite de LPRINT. Supposons que notre imprimante ait été ainsi positionnée ''France''. Non seulement, nos caractères nationaux (à, é, è, ç, ù, § et °) apparaîtront à l'écran, mais aussi sur l'imprimante ! Puisque ce sont les mêmes codes ASCII. En voici la correspondance officielle :

ASCII	FRANCE	QWERTY
64	à	@
92	ç	\
123	é	{
125	è	}
93	§	]
124	ù	|
91	°	[

Il suffira de faire GOSUB 50 000.

```
50000 'AZERTY  (56 DATA)
50030 AD=46592:GOSUB 50300:'@
50040 AD=46816:GOSUB 50300:'\
50050 AD=47064:GOSUB 50300:'{
50060 AD=47080:GOSUB 50300:'}
50070 AD=46824:GOSUB 50300:']
50080 AD=47072:GOSUB 50300:'|
50090 AD=46808:GOSUB 50300:'[
50290 RETURN
50300 FOR I=0 TO 7:READ A
50310 POKE AD+I,A:NEXT:RETURN
50320 DATA 16, 8,28, 2,30,34,30, 0
50330 DATA  0, 0,30,32,32,32,30, 8
50340 DATA  4, 8,28,34,62,32,30, 0
50350 DATA 16, 8,28,34,62,32,30, 0
50360 DATA 30,32,28,34,28, 2,60, 0
50370 DATA 16, 8,34,34,34,38,26, 0
50380 DATA 24,36,24, 0, 0, 0, 0, 0
50999 '---------------
```

Que reste-t-il comme caractères redéfinissables ? Seulement & (38) et £ (95), c'est vraiment peu. Aussi, si vous avez réellement besoin d'autres caractères très spéciaux comme plusieurs lettres grecques, des signes mathématiques, etc., vous pourrez transformer quelques lettres minuscules dont vous êtes sûr de ne pas vous servir (j, k, w). L'adresse départ sera toujours :

LES CARACTERES SEMI-GRAPHIQUES

Pour dessiner, on était obligé de passer en HIRES, d'où l'idée de redéfinir tous les caractères minuscules en petits symboles graphiques élémentaires, ce qui permet de mélanger intimement texte et dessin. On dispose ainsi de vingt-six motifs *juxtaposables*, c'est-à-dire que l'on a utilisé les "pixels tabou", afin d'obtenir des tracés continus dans les sens horizontal et vertical.

Si vous êtes intéressé par ce procédé, enregistrez-le sur une cassette à part ; vous pourrez le joindre à un de vos programmes par CLOAD " ", J (ATMOS).

Par GOSUB 62300, il apparaît à l'écran la légende lettres (ici en majuscules)/motif. Par exemple, après être passé en CTRL T, la lettre E (donc e) donne un carré.

```
62000 'MINISTYLE;M.ARCHAMBAULT/84
62010 'MINUSCULES=SYMBOLES GRAPHIQUES
62020 FOR L=46856 TO 47056 STEP 8
62030 FOR I=0 TO 7
62040 READ A
62050 POKE L+I,A
62060 NEXT:NEXT
62100 ' 208 DATA
62101 DATA32,32,32,32,32,32,32,32
62102 DATA63, 0, 0, 0, 0, 0, 0, 0
62103 DATA 1, 1, 1, 1, 1, 1, 1, 1
62104 DATA 0, 0, 0, 0, 0, 0, 0,63
62105 DATA63,33,33,33,33,33,33,63
62106 DATA 0, 0, 0,63, 0, 0, 0, 0
62107 DATA 8, 8, 8, 8, 8, 8, 8, 8
62108 DATA 1, 2, 2, 4, 8, 8,16,32
62109 DATA32,16,16, 8, 4, 4, 2, 1
62110 DATA 1, 6,24,32, 0, 0, 0, 0
62111 DATA 8, 4, 4, 4, 2, 2, 2, 1
62112 DATA 0, 0, 0, 1, 2,12,16,32
62113 DATA32,32,16,16,16, 8, 8, 8
```

```
 62114 DATA 0, 0, 0,32,16,12, 2, 1
 62115 DATA 8, 8, 8,16,16,16,32,32
 62116 DATA32,24, 6, 1, 0, 0, 0, 0
 62117 DATA 1, 2, 2, 2, 4, 4, 4, 8
 62118 DATA48, 8, 4, 4, 4, 4, 8,48
 62119 DATA33,33,18,12, 0, 0, 0, 0
 62120 DATA 3, 4, 8, 8, 8, 8, 4, 3
 62121 DATA 0, 0, 0, 0,12,18,33,33
 62122 DATA 0,12,18,33,33,33,18,12
 62123 DATA 0, 0, 0, 0,32,32,32,32
 62124 DATA32,32,32,32, 0, 0, 0, 0
 62125 DATA 1, 1, 1, 1, 0, 0, 0, 0
 62126 DATA 0, 0, 0, 0, 1, 1, 1, 1
 62200 RETURN
 62300 'LEGENDE
 62310 CLS
 62320 FOR I=65 TO 77:PRINTCHR$(I);" "
;:NEXT:PRINT
 62330 FOR I=97 TO109:PRINTCHR$(I);" "
;:NEXT:PRINT:PRINT
 62340 FOR I=78 TO90:PRINTCHR$(I);" ";
:NEXT:PRINT
 62350 FOR I=110TO122:PRINTCHR$(I);" "
;:NEXT:PRINT:PRINT
 62390 RETURN
```

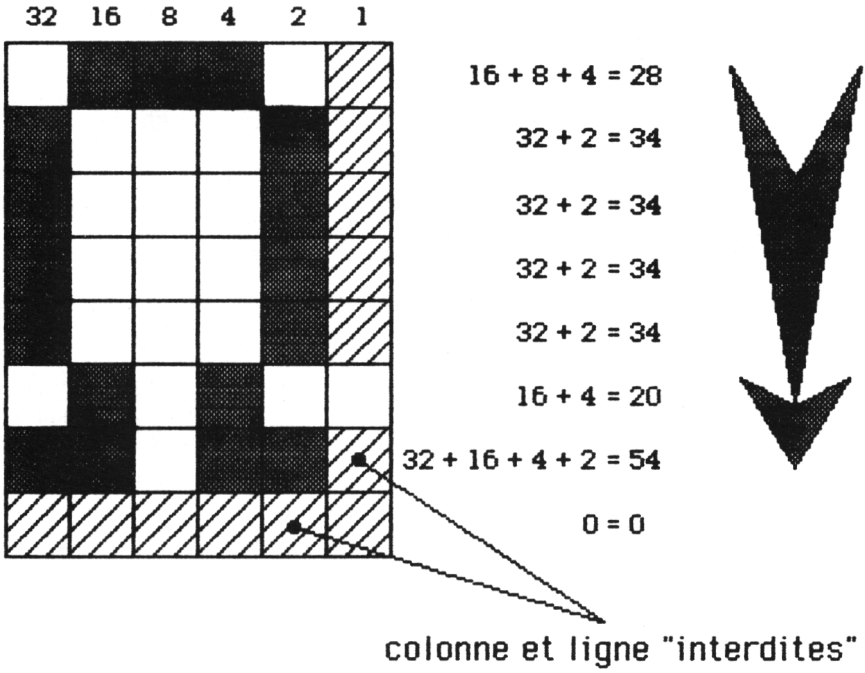
Après avoir fait GOSUB 62000, lancez ce petit programme de démonstration.

```
  100 'CABRIOLET
  105 CLS:PRINT CHR$(17)
  110 C1$="  dd cidd "
  120 C2$=" q ubbb ui"
  130 C3$=" bbsbbbbsb"
  140 FOR I=2 TO 30
  150 PRINT@I,9;C1$:PRINT@I,10;C2$:PRIN
T@I,11;C3$
  160 WAIT 3:NEXT:EXPLODE:PRINT CHR$(17
)
  190 END
```

Mais le grand intérêt de cette méthode réside dans les programmes d'animation graphique. Non seulement la rapidité est telle qu'il faut la freiner par des WAIT, mais comme nous sommes en mode TEXT, nous bénéficions des effets colorés des attributs. Disons que nous sommes ici bien plus proches des programmes en langage machine que du HIRES en BASIC, mais en conservant sa simplicité. Essayez ce petit scénario dont le graphisme n'a rien à envier à certaines cassettes du commerce.

```
 200 'SOUCOUPE
 210 CLS:PRINT CHR$(17):PAPER4:INK2
 220 PRINT@2,18;"A L U N I S S A G E"
 230 PRINT@3,25;"bsbihbsihbsbihihsbihb
ssihbsihihsbs"
 240 FOR I=20 TO 24:PRINT@2,I;CHR$(133
):NEXT:PRINT@2,25;CHR$(135)
 250 S1$="      "
 260 S2$=" dud "
 270 S3$=" pfj "
 280 V=0:FOR H=2 TO 10:GOSUB 400
 290 WAIT10:NEXT
 300 FOR H=10 TO 31:V=H-10:GOSUB 400:W
AIT 10:NEXT:WAIT200
 310 PRINT@3,6;"ter":SHOOT
 320 FOR V=21 TO 5 STEP-1:GOSUB 400:WA
IT 10:NEXT
 330 PRINT@2,6;CHR$(129)
 340 ZAP:FOR I=29 TO 3 STEP-1:PRINT@I,
6;"ff ":WAIT 1:NEXT
 350 PRINT@3,6;"* ":EXPLODE:WAIT 20:PR
INT@3,6;"  ";
 390 PRINT CHR$(17):END
 400 PRINT@H,V;S1$:PRINT@H,V+1;S2$
 410 PRINT@H,V+2;S3$:PRINT@H,V+3;S1$
 430 RETURN
```

Que la ligne 230 ne vous effraie pas, c'est pour dessiner un sol lunaire... La soucoupe volante est définie par seulement six caractères (lignes 260-270). En ces quelques lignes, la soucoupe se déplace horizontalement, puis descend en oblique, change de couleur à l'alunissage. Un temps d'arrêt. Un vaisseau ennemi apparaît, la soucoupe décolle alors à la verticale, s'arrête à l'altitude de l'ennemi ; sa coupole devient rouge, lancement d'un missile qui détruit l'autre vaisseau. Si cela vous dit de continuer...



Les huit octets codifiants ce caractère (lettre OMEGA), sont donc : 28, 34, 34, 34, 34, 20, 54, 0. Les deux bits de gauche de chaque octet ne sont pas pris en compte (valeur maxi 63).

Figure XI 1

Grâce à la fonction CHAR, ces figurines sont transposables en HIRES. Vous serez alors étonné de leur vitesse de traçage ; vitesse qui serait impossible en les redessinant constamment par des DRAW !

Autre utilisation, agrémenter des pages d'écran texte, mots soulignés, encadrés etc...

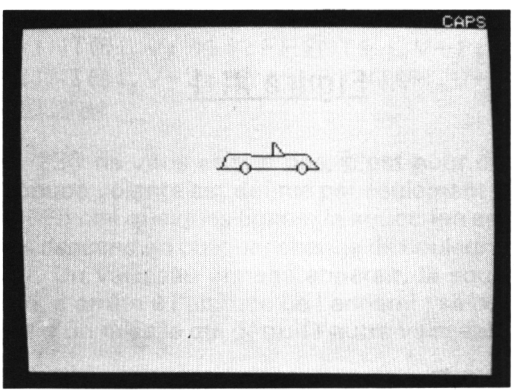
Exemple :

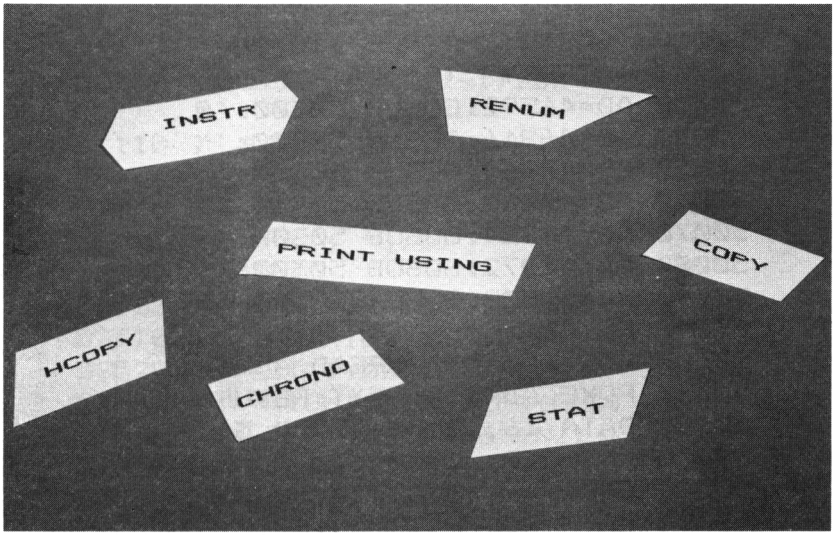
```
 500 'ENCADREMENTS
 510 CLS
 520 PRINT@11, 9;"cbbbbbbbbbbbbba"
 530 PRINT@11,10;"c ENCADREMENT a"
 540 PRINT@11,11;"cddddddddddddda"
 550 PRINT@11,15;"OU MOT SOULIGNE"
 560 PRINT@18,16;"bbbbbbbb"
 580 FOR I=2 TO 37:PRINT@I,1;"e";:NEXT
 590 FOR V=2 TO 24:PRINT@2,V;"e":PRINT
@37,V;"v":NEXT
 600 FOR I=2 TO 37:PRINT@I,25;"v";:NEX
T
 610 PRINT@19,19;:GET R$:CLS
 690 END
```

Deux remarques :

— Rien ne vous empêche de modifier les symboles de la figure XI 2 afin d'obtenir des dessins "pleins", par exemple un disque plein pour la lettre "v".

— Les caractères dits "alternés" obtenus en LORES 1 (ou par attributs de ligne) n'ont aucune application pratique en France : vous pouvez les redéfinir également mais en changeant l'adresse départ qui n'est plus 46080, mais 47104 (ATMOS). Ils seront reproductibles en HIRES si le deuxième paramètre de la fonction CHAR est 1 au lieu de zéro.





Chapitre XII

LES FONCTIONS BASIC MANQUANTES

Nous avons déjà construit les fonctions INSTR, PRINT USING et quelques autres, mais pour que cela soit opérationnel, nous allons grouper ces sous-programmes utilitaires sur trois cassettes différentes :

- le basic chaînes ;
- les aides à la programmation ;
- les aides à l'imprimante.

Leurs numéros de lignes sont tous supérieurs à 50000. Le but est de pouvoir ajouter au programme en cours le contenu de telle cassette, et ce, par CLOAD " ", J (ATMOS).

La première cassette, vous l'avez déjà en majeure partie ; le listing suivant vous permettra de la compléter si nécessaire :

Une fois votre programme terminé et vérifié, vous pourrez alléger l'ensemble en supprimant les modules inutilisés.

```
 49999 END:'----------------------
 50000 'BASIC CHAINES ;10/84
 50020 'AZERTY  (56 DATA)
 50030 AD=46592:GOSUB 50300:'@
 50040 AD=46816:GOSUB 50300:'\
 50050 AD=47064:GOSUB 50300:'{
 50060 AD=47080:GOSUB 50300:'}
 50070 AD=46824:GOSUB 50300:']
 50080 AD=47072:GOSUB 50300:'|
 50090 AD=46808:GOSUB 50300:'[
 50290 RETURN
 50300 FOR I=0 TO 7:READ A
 50310 POKE AD+I,A:NEXT:RETURN
 50320 DATA 16, 8,28, 2,30,34,30, 0:'@
 50330 DATA  0, 0,30,32,32,32,30, 8:'\
 50340 DATA  4, 8,28,34,62,32,30, 0:'{
 50350 DATA 16, 8,28,34,62,32,30, 0:'}
 50360 DATA 30,32,28,34,28, 2,60, 0:']
 50370 DATA 16, 8,34,34,34,38,26, 0:'|
 50380 DATA 24,36,24, 0, 0, 0, 0, 0:'[
 50999 '----------------
 51000 'INSTR
 51010 'Place K du mot MO$ dans le tex
te TX$
 51030 K=0:FOR I=1 TO LEN(TX$):IF MID$
(TX$,I,LEN(MO$))=MO$ THEN K=I
 51040 NEXT:RETURN
 51999 '----------------
 52000 'PRINT USING     ####.##
 52010 'PU$=FORMAT+NB=Nombre:Retourne
NB$ formate
 52030 TX$=PU$:MO$=".":GOSUB 51000:KF=
K
 52040 LT=LEN(PU$):LE=K-1:LD=LT-K
 52050 IF K=0 THEN LE=LT:LD=0:IF ABS(N
B)<1 THEN NB=0
 52060 NB$=STR$(NB):L=LEN(NB$)-1:IF NB
<0 THEN L=L+1
 52065 NB$=RIGHT$(NB$,L)
 52070 IF ABS(NB)<.01 AND NB<>0 THEN G
OSUB 52500:GOTO 52130
 52080 TX$=NB$:GOSUB 51000
 52090 IF K=0 THEN E$=RIGHT$("       "+
```

```
NB$,LE):D$=LEFT$("          ",LD):GOTO 521
10
 52100 E$=LEFT$(NB$,K-1):D$=RIGHT$(NB$
,L-K)
 52110 IF KF=0 THEN NB$=RIGHT$("
"+E$,LT):GOTO 52130
 52120 NB$=RIGHT$("          "+E$,LE)+"."+
LEFT$(D$+"          ",LD)
 52130 RETURN
 52500 'FORMATAGE PETITS NOMBRES
 52510 NZ=VAL(RIGHT$(NB$,2))-1
 52520 CH$=STR$(NB*100000)
 52530 CH$=RIGHT$(CH$,LEN(CH$)-1)
 52540 D$=LEFT$("000000",NZ)+CH$
 52550 D$=LEFT$(D$,LD)
 52560 E$=LEFT$("          ",LE)
 52570 IF NB<0 THEN E$=RIGHT$("      -"
,LE)
 52580 NB$=E$+"."+D$
 52590 RETURN
 52999 '----------------
 55000 'REPONSE A UN MENU
 55010 'Entre TX$ et MO$,renvoie K
 55020 LT=LEN(TX$):TT=15-LT:IF TT<2 TH
EN TT=2
 55030 PRINT@TT,24;"REPONSE(";:FOR I=1
 TO LT-1:PRINTMID$(TX$,I,1)+",";:NEXT
 55040 PRINTMID$(TX$,LT,1)+") ->";:GET
 MO$
 55050 GOSUB 51000:IF K=0 THEN PRINTCH
R$(14):GOTO 55000
 55060 RETURN
 57999 '----------------
 58000 'SUBSTITUTION DE CHAINE
 58010 'TX$=CHAINE RECEPTIVE
 58020 'MO$=CHAINE IMPOSEE
 58030 'K=POSITION DE MO$ DANS TX$
 58040 K=K-1
 58050 TX$=LEFT$(TX$,K)+MO$+RIGHT$(TX$
,LEN(TX$)-K-LEN(MO$))
 58060 RETURN
 63999 '---FIN DE LISTING---
```

L'UTILITAIRE « AIDE A LA PROGRAMMATION »

Il groupe trois modules qui seront inutiles une fois le programme terminé. A effacer si vous avez le temps.

— *Le module CHRONO* est un chronomètre électronique au centième de seconde. On l'utilise pour comparer les durées de deux processus en concurrence. C'est lui que nous avons utilisé pour chronométrer les différentes boucles de 1 à 1000 au début du chapitre VIII. Le top départ, c'est GOSUB 54000, ce qui impose la valeur maxi (65535) au compteur situé à l'adresse 630. Il décompte en centièmes de seconde (piloté par quartz).
Le top fin est obtenu par GOSUB 54100, qui lit la valeur du compteur. Puis calcul en minutes, secondes et centièmes , le tout présenté en ''toutes lettres'' si vous demandez :

PRINT CH$ (CH = CHRONO)

Une réserve : entre deux tops, n'utilisez pas l'instruction WAIT, car elle utilise le même compteur... Durée maxi, 10 min 55 s.

— *Le module RENUM* renumérote les lignes de programme : vous fixez le début, la fin et le pas. Il ne corrige pas les GOTO et GOSUB, mais vous présente une liste des anciens et des nouveaux numéros de lignes afin de faire ces retouches à la main mais sans erreurs.
C'est le programme de la page 73 du manuel ATMOS mais bien amélioré. En tapant GOSUB 56000, le bas de l'écran vous demande la zone de lignes à renuméroter et à quel pas (100 lignes maximum). C'est très rapide. Voulez-vous le tableau de correspondances des lignes ? A l'écran, tapez GOSUB 56500 ; sur imprimante, faites GOSUB 56600.

— *Le module STAT* vous communique le bilan mémoire. Tapez GOSUB 60000 et sur la dernière ligne de l'écran apparaît ''en clair'' le nombre d'octets utilisés par le programme, et combien il en reste de disponible. Pour plus de précision, il est préférable de faire RESET avant de commander STAT.
La valeur fixe 37629 (ligne 60010) correspond à l'ATMOS.

Note : Dans un esprit de standardisation, nous donnons à nos modules le nom de la fonction BASIC qu'ils remplacent : c'était le cas pour PRINT USING, INSTR, c'est ici le cas pour RENUM et STAT.

```
 49999 END: '----------------------
 54000 'CHRONO ->CH$(maxi=10mn 55s)
 54010 DOKE 630,65535: 'DEPART
 54020 RETURN
 54100 TA=DEEK(630): 'ARRET
 54110 TS=(65535-TA)/100
 54120 IF TS<60 THEN CH$=STR$(TS)+"s":
RETURN
```

```
 54130 TM=INT(TS/60):CH$=STR$(TM)+"mn"
+STR$(TS-TM*60)+"s"
 54140 RETURN
 54999 '-----------------
 56000 'RENUM(sans les GOTO,GOSUB)
 56005 DIM L(100,2)
 56010 PRINT@6,24;"RENUM :LIGNE DEPART
 ";:INPUT DB:GOSUB 56200
 56020 PRINT@10,24;"QUI DEVIENT..";:IN
PUT NL:GOSUB 56200
 56030 PRINT@7,24;"JUSQU'A(incluse)";:
INPUT FI:GOSUB 56200
 56040 IF FI>=49999 THEN FI=49999
 56050 PRINT@12,24;"AU PAS DE ";:INPUT
 PA:GOSUB 56200
 56060 W=#501:I=1:REPEAT
 56070 LI=DEEK(W+2):IF LI<DB THEN 5609
0 ELSE L(I,1)=LI:L(I,2)=NL
 56080 DOKE(W+2),NL:NL=NL+PA:I=I+1
 56090 W=DEEK(W): UNTIL DEEK(W+2)>FI:R
ETURN
 56200 PRINT@2,24;CHR$(14):RETURN
 56300 '---
 56500 'CORRESP.LIGNES
 56510 PRINT"ANCIEN. NOUVEL.LIGNES"
 56530 I=1:REPEAT
 56540 PRINT L(I,1),L(I,2):ILOGI+1
 56560 UNTIL L(I,1)=0
 56570 RETURN
 56600 'CORRESP.LIGNES IMPRIM.
 56610 LPRINT"ANCIEN. NOUVEL.LIGNES"
 56620 LPRINT
 56630 I=1:REPEAT
 56640 LPRINT L(I,1),L(I,2):I=I+1
 56660 UNTIL L(I,1)=0
 56670 RETURN
 56999 '-----------------
 60000 'STAT
 60010 ML=FRE(""):MP=37629-FRE(0)
 60020 PRINT@ 4,25;"Pris";MP;";il rest
e";ML;"octets"
 60030 RETURN
 63999 '---FIN DE LISTING---
```

L'UTILITAIRE « AIDE A L'IMPRIMANTE »

Disons tout de suite qu'il ne s'agit pas de l'imprimante ORIC MCP 40, ni de la SEIKOSHA GP 100, mais de l'EPSON RX 80. Pourquoi ? L'auteur estime que l'ORIC ATMOS a la qualité professionnelle (ce serait chose partout admise si le prix de vente et le volume du boîtier étaient doubles...). De ce fait, il est logique de l'équiper d'une imprimante professionnelle, certes plus chère mais d'un rapport qualité/prix aussi exceptionnel que celui de l'ATMOS. Qu'avons-nous en plus ?

La vitesse de 100 caractères par seconde, une qualité d'impression très proche de la ''qualité courrier'', une robustesse légendaire, et surtout une multitude d'options d'impression : plusieurs tailles et formes de lettres, plusieurs ''claviers'', trois épaisseurs de traits, redéfinition des caractères, et d'innombrables gadgets, le tout programmable à partir de n'importe quel micro-ordinateur.

Si vous possédez un autre type d'imprimante, l'auteur ne veut pas que vous vous sentiez lésé ; aussi pour chaque module de cet utilitaire, nous indiquerons d'une part le fonctionnement détaillé et d'autre part, la traduction en clair des ordres BASIC de l'EPSON RX 80. Ainsi, il vous sera facile d'adapter ces sous-programmes à votre imprimante.

```
 49999 END:'----------------------
 51000 'INSTR
 51010 'Place K du mot MO$ dans le tex
te TX$
 51020 'Pas de Blanc au debut de MO$
 51030 K=0:FOR I=1 TO LEN(TX$):IF MID$
(TX$,I,LEN(MO$))=MO$ THEN K=I
 51040 NEXT:RETURN
 51999 '----------------
 53000 'HARD COPY texte
 53030 FOR I=48040 TO 49080 STEP 40
 53040 FOR J=0 TO 39:C=PEEK(I+J):LPRIN
T CHR$(C);:NEXT
 53050 LPRINT CHR$(13):NEXT
 53060 RETURN
 53999 '----------------
 55000 'REPONSE A UN MENU
 55010 'Entre TX$ et MO$,renvoie K
 55020 LT=LEN(TX$):TT=15-LT:IF TT<2 TH
EN TT=2
```

```
 55030 PRINT@TT,24;"REPONSE(";:FOR I=1
 TO LT-1:PRINTMID$(TX$,I,1)+",";:NEXT
 55040 PRINTMID$(TX$,LT,1)+") ->";:GET
 MO$
 55050 GOSUB 51000:IF K=0 THEN PRINTCH
R$(14):GOTO 55000
 55060 RETURN
 55999 '----------------
 57000 'HARD COPY HIRES
 57010 LPRINT CHR$(27)"@";CHR$(27)"A"C
HR$(6);
 57020 DIM Z(200):C=0
 57030 FORAH=40961 TO 40999:I=0
 57035 IF AH>40961 THEN C=-64
 57040 FOR A=AH TO AH+7960 STEP 40
 57050 I=I+1
 57060 Z(I)=PEEK(A):NEXT
 57080 LPRINT CHR$(27)"*"CHR$(4);CHR$(
I);CHR$(0);
 57090 FOR J=I TO 1 STEP-1
 57095 Y=Z(J)+C
 57100 LPRINT CHR$(Y);:NEXT
 57110 LPRINT CHR$(13):NEXT
 57120 LPRINT CHR$(27)"@":RETURN
 60999 '----------------
 61000 'IMPRIMANTE EPSON RX 80(switch
sur Pica)
 61010 CLS:PRINT@10,2;"IMPRIMANTE BRAN
CHEE"
 61020 PRINT@8,13;"(M = MAXI ; Q = QUI
TTER)"
 61030 PRINT@3,10;"NOMBRE DE CARACTERE
S PAR LIGNE:";:INPUT RE$
 61040 IF RE$="Q"THEN CLS:RETURN
 61050 IF RE$="M"THEN LPRINT CHR$(27);
"@";:GOTO 61080
 61060 RE=VAL(RE$):IF RE<4 THEN 61000
 61070 LPRINT CHR$(27);"@";CHR$(27);"Q
";CHR$(RE);
 61080 CLS:PRINT@15,6;"CLAVIER:":PRINT
@14,9;"U = U.S.A":PRINT@14,11;"F = FRA
NCE"
```

```
 61090 PRINT@14,13;"G = GERMANY":PRINT
@14,15;"E = ENGLAND"
 61100 PRINT@10,17;"Q = QUITTER":TX$="
UFGEQ":GOSUB 55000
 61105 IF K=5 THEN CLS:RETURN
 61110 LPRINT CHR$(27);"R";CHR$(K-1);:
CLS
 61120 PRINT@11,2;"FORME DES LETTRES":
PRINT@10,6;"N = NORMALE(pica)"
 61130 PRINT@10,8;"E = ELITE":PRINT@10
,10;"C = CONDENSEE"
 61140 PRINT@10,12;"L = LARGES"
 61150 TX$="NECL":GOSUB 55000
 61160 ON K GOTO 61200,61170,61180,611
90
 61170 LPRINT CHR$(27);"M";:GOTO 61200
 61180 LPRINT CHR$(15);:GOTO 61200
 61190 LPRINT CHR$(27);"W";CHR$(1);
 61200 CLS:PRINT@10,3;"FORCE DE LA FRA
PPE"
 61210 PRINT@13,8;"N = NORMALE":PRINT@
13,10;"R = RENFORCEE":TX$="NR":GOSUB 5
5000
 61220 IF K=2 THEN LPRINT CHR$(27);"E"
;
 61230 CLS:RETURN
 63999 '---FIN DE LISTING---
```

LEGENDE DE CODES D'EDITION UTILISES

Ils sont de la forme LPRINT CHR$ (27)... CHR$ (27) est le code de la touche "ESC".

ESC "@"	: Initialisation de l'imprimante
ESC "A" CHR$ (n)	: Interligne en n/72e de pouce
ESC "*"	: Impression par "bit image" (et non plus par code ASCII)
ESC "Q" CHR$ (n)	: n caractères par ligne
ESC "R" CHR$ (n)	: n de 0 à 10 : sélection entre onze claviers internationnaux
ESC "M"	: Lettres forme « ELITE »
CHR$ (15)	: Lettres forme « CONDENSEE »
ESC "W"	: Lettres double taille
ESC "E"	: Trait renforcé (ou "gras")

LE MODULE « HARD COPY TEXTE » (lignes 53000-)

Il y a copie de l'écran sur l'imprimante. On lit la mémoire d'écran ligne par ligne et on imprime le code ASCII trouvé. C'est banal, mais très rapide.

LE MODULE « HARD COPY HIRES » (lignes 57000-)

C'est assez complexe, mais relativement rapide (environ trois minutes). Le principe est le suivant :

L'écran haute résolution est composé de 200 lignes. Chacune de ces lignes est l'illustration binaire de 40 octets mis bout à bout. Chaque bit = un pixel. Donc $40 \times 6 = 240$ pixels (ou points). On ne peut donc lire dans la mémoire d'écran que des octets, donc des ''tirets'' *horizontaux* de 6 pixels chacun.

Malheureusement toutes les imprimantes à aiguilles (ou à ''matrices'') impriment des images binaires *verticales*. Dans la tête d'impression les huit petites aiguilles sont disposées les unes au-dessus des autres. La parade est simple, il suffit de tourner le dessin d'un quart de tour sur le papier ; ainsi les octets horizontaux de l'écran sont transmis tels quels à la tête d'imprimante, et il ne sera pas très gênant de regarder l'image finale en tournant la feuille de papier. Le bas de l'image se trouve sur ce qu'on appelait le bord gauche du papier.

Nous allons donc explorer la mémoire d'écran par ''bandelettes'' verticales, soit 200 octets que nous mettons en DIM Z (200). Une fois rempli, nous vidons son contenu sur l'imprimante, et ce 40 fois de suite.

La fidélité est telle qu'un point isolé (un centre de cercle) est parfaitement reproduit.

LE MODULE « OPTIONS » (lignes 61000-)

Il a pour rôle de charger la mémoire de l'imprimante avec diverses options. Tout ce qui sera ensuite commandé par LPRINT ou LLIST sera imprimé selon ces modes.

C'est une suite de menus à l'écran, d'où la présence des deux vieilles connaissances, INSTR et REPONSE MENU (51000- et 55000-).

On nous demande successivement :

- le nombre de caractères par ligne,
- la nationalité du clavier,
- la forme des lettres (quatre),
- la frappe, normale ou renforcée.

C'est ultra-rapide car, mis-à-part la première question, on n'a pas à faire RETURN.

LES TROIS CASSETTES D'UTILITAIRES

Faites deux enregistrements par cassette. Il est indispensable d'inscrire sur l'étiquette (du boîtier) le plan de chacune, c'est-à-dire la légende de chaque GOSUB et peut-être quelques détails d'emploi figurant dans les REM. Par exemple, pour la cassette n° 2 (programmation), vous inscrivez :

```
54000   : CHRONO départ
54100   : CHRONO arrêt → CH$
56000   : RENUM
          56500 : PRINT LISTE n° [
          56600 : LPRINT LISTE n°;
60000   : STAT
```

La première fois nous avions logé tous ces modules sur un seul (et long) enregistrement. Ce fut une erreur pour les raisons suivantes :

- Trop long à charger.
- Trop de mémoire occupée inutilement (près de 4 500 octets au total).
- Effacement partiel ou total fastidieux.
- Un programme court devient donc long à charger (car il a tout le bloc en queue).
- Il est impossible d'avoir besoin de seulement la moitié des modules pour un même programme.

Voilà pourquoi nous les avons ensuite fractionnés en trois (c'est un minimum). Ils sont alors très vite chargés et, de ce fait, on les utilise sans hésiter.

Chapitre XIII

HIRES SANS IMPATIENCE

Il serait faux de vous laisser croire qu'on peut tout faire dès que l'on est en écran HIRES, du moins par le BASIC seulement. Les super dessins animés des meilleures cassettes de jeux sont le plus souvent en langage machine. Par le BASIC, on peut s'en approcher, pas l'égaler, surtout en ce qui concerne la vitesse. Vous avez déjà sans doute remarqué que les fonctions DRAW et CIRCLE ne sont pas instantanées...

Dès le départ, il faut faire un choix entre les graphismes en modes TEXT ou HIRES. Voyons les avantages et inconvénients :

— TEXT : Figures simples pouvant se déplacer rapidement (notre programme SOUCOUPE du chapitre XI). Ces figurines peuvent être ''pleines'' mais *petites*.

— HIRES : vu la lenteur du tracé, il est scabreux de faire une animation. Contrairement au mode TEXT, un deuxième tracé n'écrase pas le premier, il s'y *superpose*. En revanche, le tracé de la forme peut être programmé, paramétré par des calculs. Dessiner une petite forme aux contours compliqués demanderait une fastidieuse suite de DRAW. Il n'est guère envisageable de ''remplir'' un motif, sauf s'il s'agit d'un rectangle par la fonction FILL. En HIRES, on peut très facilement dessiner des grands dessins géométriques, des courbes etc..., ce qui serait quasi délirant en mode TEXT.

Un autre avantage de HIRES est qu'il peut reprendre, par la fonction CHAR, des figurines obtenues par juxtaposition de caractères redéfinis, mais les mettre en animation (telle la soucoupe) est déconseillé pour deux raisons :
— écriture lourde,
— scintillement beaucoup plus visible qu'en mode TEXT, car il faut alternativement effacer l'image précédente.

Dans un programme essentiellement graphique, il n'est pas honteux, mais au contraire astucieux, de passer tour à tour du mode TEXT au mode HIRES en fonction de ce que l'on veut tracer.

La haute résolution est un domaine du BASIC franchement à part. Il faut prendre des habitudes nouvelles, puis les astuces de tracé viennent rapidement.

Au début de cet ouvrage, nous avons dit que l'organigramme était obligatoire pour un programme long, et facultatif pour un court. En HIRES point d'organigramme, mais un PLAN de l'ECRAN, même si le dessin est simple !

Second point, il nous faut détruire une "légende". On croit souvent que l'on ne dispose que de deux couleurs PAPER et INK, puisque les attributs de ligne sont réservés au mode TEXT. FAUX, mais ils se programment autrement.

A l'adresse d'un octet de la mémoire écran, il suffit d'imposer un nombre, par POKE, compris entre 0 et 23, aussitôt toute la partie droite de cette ligne subit cet attribut. En voici la liste :

0	encre noire
1	encre rouge
2	encre verte
3	encre jaune
4	encre bleue
5	encre magenta
6	encre cyan
7	encre blanche
8 à 11	pas d'effet
12 à 15	clignotant
16	papier noir
17	papier rouge
18	papier vert
19	papier jaune
20	papier bleu
21	papier magenta
22	papier cyan
23	papier blanc

Essayez ce petit programme, son effet sera plus "parlant" et instructif. Pour mieux les discerner, chaque attribut est appliqué sur cinq lignes consécutives (ligne 150). Remarquez la lenteur de la ligne 130.

Observez bien l'avantage de la ligne 50, il faudra la réinscrire (provisoirement) à chaque fois que l'on met au point un programme en HIRES, car c'est un gain de temps considérable. Observez attentivement les effets de ces deux pages d'écran (INK en haut, PAPER en bas).

```
10 'ATTRIBUTS EN HIRES
20 A=42000:GOSUB 100
30 GET R$
40 A=42009:GOSUB 100
50 GET R$:TEXT:LIST-170
60 END
100 HIRES
110 FOR I=20 TO 60
120 CURSET I,20,1
130 DRAW 179,179,1:NEXT
140 FOR I=0 TO 23
150 FOR N=0 TO 4:A=A+40
160 POKE A,I:NEXT:NEXT
170 RETURN
```

A présent, ajoutez cette ligne 165 :

```
165 POKE 42018,17:POKE 42025,16
```

Le premier POKE donne le départ d'un petit trait rouge (en paper), interrompu par le second POKE qui remet le papier noir. Une sorte de « mini-FILL » mais avec la couleur plus facile à programmer.

VIVE LE CURMOV !

La tentation du débutant est d'abuser de l'instruction CURSET pour redémarrer chaque série du DRAW, après chaque discontinuité du tracé : danger ! Imaginons un tracé géométrique assez complexe qui a nécessité cinq CURSET ; hélas, l'image est bonne mais le dessin est mal centré, beaucoup trop en haut à gauche. Il va vous falloir recalculer chaque CURSET ; d'où cinq risques d'erreurs. Pour éviter de tomber dans cette galère, il aurait suffit de définir seulement le *CURSET de départ* et tous les autres décrochements de tracés par des CURMOV. En modifiant les coordonnées de cet unique CURSET, toute la figure se déplace sur l'écran... N'est-ce pas mieux ainsi ?

LE PLAN D'ECRAN

Rien de plus facile et aussi de plus sûr. Prenez du papier quadrillé 5 × 5 mm, format commercial 21 × 28 cm. Tracez à l'encre un rectangle 240 × 200 mm. C'est l'écran HIRES. Un carreau = 5 ''points'', autrement dit 1 point = 1 mm. Dans le sens horizontal, il ne faudra pas dépasser la valeur 239, et 199 dans le sens vertical (vous avez droit aux valeurs zéro).

Attention aux pièges des couleurs PAPER et INK, en effet :

Sans rien spécifier d'autre que HIRES, vous serez en fond noir et encre blanche ; vous pouvez alors toucher le bord gauche (exemple CURSET 0,15,1).

Après HIRES, vous demandez PAPER 4. C'est un attribut pour les 200 lignes ; de ce fait, vous ne pouvez rien mettre de zéro à 5 sur le bord gauche (6 bits par attribut). Si en plus, vous demandez INK 1, c'est encore 6 points d'attributs. Le CURSET visible le plus à gauche sera donc positionné à 12, exemple CURSET 12,100,1.

Sur ce cadre, faites votre dessin au crayon (pour gommer), en suivant le quadrillage. Ainsi, vos DRAW et vos CURMOV seront presque toujours des multiples de 5 (facilité).

L'autre solution consiste à dessiner sur une feuille quadrillée vierge le motif qu'il faudra centrer. Marquez votre CURSET de départ. Avec un double décimètre, mesurez l'encombrement de votre dessin. Vous savez que son ''axe vertical'' doit se situer vers 240/2 = 120. Votre départ est à 25 mm à gauche de ce milieu (un exemple), donc son positionnement horizontal sera 120 − 25 = 95. Opérez de même avec le positionnement en hauteur.

Partant du point de départ, numérotez vos segments DRAW dans l'ordre.

Grâce à ce dessin sur petits carreaux, vous connaissez déjà le CURSET de départ, et tous les déplacements des DRAW et CURMOV. Il ne reste plus qu'à passer au clavier et tout ira alors très vite et sans erreur.

Dans l'exemple qui va suivre, nous allons mettre tout ça en pratique, en vous montrant en plus quelques petites ''astuces pour fainéants''.

LA PROGRAMMATION HIRES

Nous vous proposons de dessiner trois caricatures identiques de personnages à mines patibulaires.

Sur le papier, nous n'en avions dessiné qu'un ; trois positions pour le CURSET de départ donc trois reproductions. Le personnage du bas est le premier et nous lui avons rajouté un petit détail sur son chapeau afin de vous situer le CURSET de départ.

Nous commençons donc par le chapeau, d'abord le tour de la coiffe dans le sens horaire, soit 6 segments qui nous ramènent au point de départ. De là, nous repartons pour les 3 segments du bord du chapeau.

Première astuce de fainéant, nous nous servons des DATA, et l'instruction DRAW est dans une boucle FOR NEXT.

Puis, nous faisons un CURMOV afin d'attaquer le pourtour du visage, encore dans le sens horaire (13 segments).

Il ne reste plus que les yeux, le nez et la bouche. Etant donné qu'il y a peu de segments entre chaque CURMOV, on écrit chaque DRAW.

Terminé. Seconde astuce de fainéant, les lignes 450 et suivantes : la fonction ON GOTO rend l'écriture bien plus simple et claire qu'avec des IF THEN ELSE.

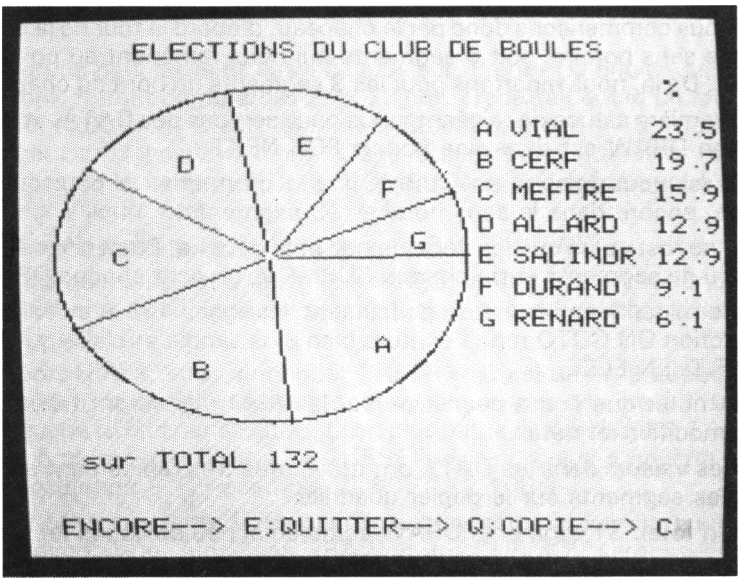
A noter que l'on a pas lésiné sur les REM ; ils seront très utiles pour modifier tel détail.

Les valeurs dans les DATA ont été écrites très rapidement en suivant les segments sur le papier quadrillé.

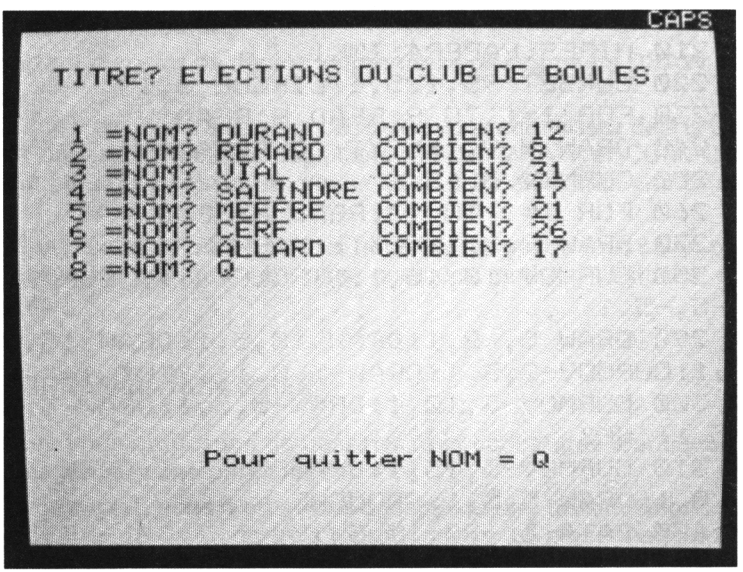
Au total, 31 segments DRAW écrits en si peu de lignes (et si peu de temps).

Entraînez-vous en commençant par quelque chose de simple, par exemple, dessinez au crayon une maison avec porte, fenêtres et cheminée, et entrez tout cela en DATA. Votre rapidité vers la fin vous surprendra.

```
 200 'KESSALTYPES
 210 HIRES:PAPER4:INK1
 220 CURSET 95,125,1:CIRCLE 3,1
 230 FOR I=1 TO 9:READ H:READ V
 240 DRAW H,V,1:NEXT:'CHAPEAU
 250 CURMOV 0,15,1
 260 FOR I=1 TO 13:READ H:READ V
 270 DRAW H,V,1:NEXT:'VISAGE
 280 CURMOV 10,10,1:CIRCLE 3,1: CURMOV
-5,-5,1
 290 DRAW 5,-2,1:DRAW 10,5,1:DRAW 15,0
,1:CURMOV-2,3,1:DRAW-5,0,1:'YEUX
 300 CURMOV-3,10,1:DRAW-5,5,1:DRAW-5,-
5,1:'NEZ
 310 CURMOV-5,15,1:DRAW 5,-5,1:DRAW 10
,0,1:DRAW 5,5,1:'BOUCHE
 400 DATA 5,-20,15,5,15,-5,5,20,-20,5,
-20,-5
 420 DATA -35,15,110,0,-35,-15
```



Le ''Camembert'' tracé et légendé en HIRES.



Entrée des données du programme ''Camembert''.

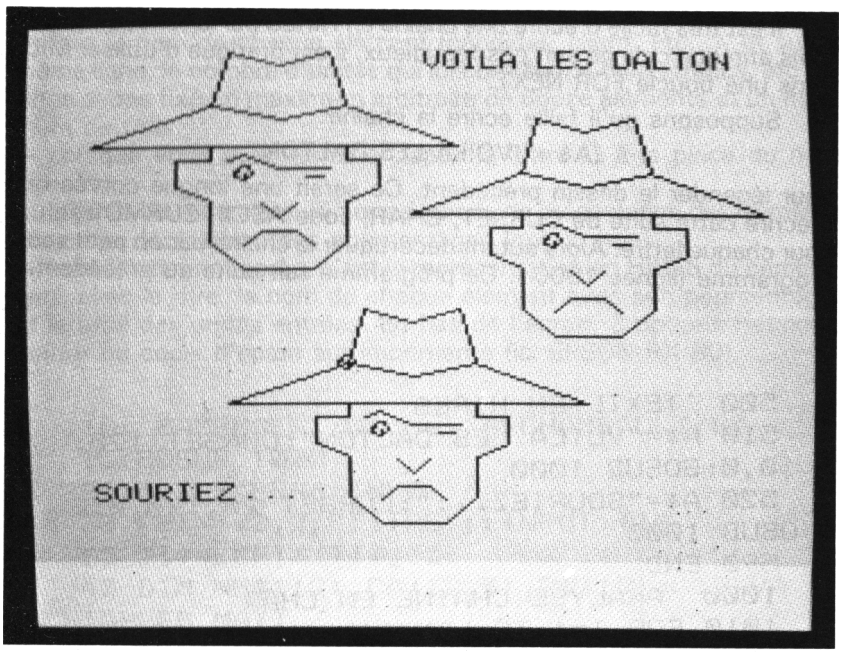


Illustration du programme ''KESSALTYPES''.

```
 430 DATA 0,5,10,0,0,10,-10,5,0,15,-10
,10,-20,0
 440 DATA -10,-10,0,-15,-10,-5,0,-10,1
0,0,0,-5
 450 N=N+1:ON N GOTO 460,470,480
 460 CURSET 55,30,1:RESTORE:GOTO 230
 470 CURSET150,55,1:RESTORE:GOTO 230
 480 '
```

CHAR EN CHARADE

Il est très facile d'écrire une chaîne en HIRES par la fonction CHAR, mais afin que cela ne soit pas fastidieux, il est pratique d'utiliser MID$ dans une boucle FOR NEXT.

Supposons qu'il faille écrire la chaîne

A$ = "VOILA LES DALTON"

pour légender le dessin précédent. Ce serait une longue corvée que d'écrire cette suite de CURSET, CHAR, code ASCII, CURMOV, etc... pour chaque lettre. Alors autant décortiquer la chaîne par un petit sous-programme (lignes 1000-). Ce programme fait suite au précédent.

```
 500 'TEXTE EN HIRES
 510 A$="VOILA LES DALTON":CURSET 120,
10,0:GOSUB 1000
 520 A$="SOURIEZ...":CURSET 20,170,0:G
OSUB 1000
 590 END
 1000 'ANALYSE CHAINE EN CHAR
 1010 FOR I=1 TO LEN(A$)
 1020 CHAR ASC(MID$(A$,I,1)),0,1
 1030 CURMOV 6,0,0:NEXT:RETURN
```

En ligne 30, nous avons mis le CURMOV normal de 6, mais rien n'empêche alors d'augmenter cette valeur afin d'espacer davantage les lettres.

LE DESSIN PARAMÉTRÉ

Voyons des choses un peu plus sérieuses. Nous allons établir un dessin non pas par des DRAW immuables, mais par des calculs ; et pour illustrer cela, nous vous livrons un programme complet qui exécute une représentation des divers pourcentages par des secteurs de cercles.

Ce petit logiciel va utiliser beaucoup de méthodes, d'astuces et de gadgets que nous avons déjà exposés. Nous vous conseillons de le sauver sur une cassette à part.

Voici son "scénario" :

— Vous devez d'abord donner le titre, le nom de ce "camembert", par exemple "chiffre d'affaires 1985", "élections du club", etc...
— Puis le nom de chaque élément (candidat, client, etc...), et sur la même ligne, le nombre d'unités qui lui correspond (voix, francs, etc...). Nous avons fixé un maximum arbitraire de douze éléments et un minimum de trois.
— Lorsque vous avez terminé, tapez la lettre Q à la place du nom suivant.
— L'écran vous demande si vous désirez une présentation des éléments par ordre décroissant des pourcentages (oui ou non).
— Immédiatement après, c'est la représentation en HIRES du camembert, avec le titre, le nom de chaque élément avec son pourcentage, et le total des unités entrées. En bas de l'écran, 3 options : encore, quitter ou copie d'écran sur imprimante (ici EPSON RX 80).

```
 10 'CAMEMBERT/Michel ARCHAMBAULT/84
 15 GOSUB 1000
 20 CLS:PAPER6:INK0
 30 PRINT@2,2;"TITRE";:INPUT TI$
 35 TI$=LEFT$(TI$,35)
 40 DIM NM$(12),CA(12,3),R$(12):'12 VA
RIABLES MAXI
 50 N=N+1:PRINT@2,4+N;N;"=NOM";:INPUT
NM$
 55 IF N>2 AND NM$="Q"THEN N=N-1:GOTO
87
 60 NM$(N)=LEFT$(NM$+"          ",7)
 65 IF N=3 THEN PRINT@10,22;"Pour quit
ter NOM = Q"
 70 PRINT@20,4+N;"COMBIEN";:INPUT NB$:
NB=VAL(NB$)
 75 IF NB=0 AND NB$<>"0" THEN 70 ELSE
CA(N,1)=NB
 80 T=T+CA(N,1):IF N=12 THEN 87
```

```
 85 GOTO 50
 87 PRINT@4,7+N;CHR$(14);"VOULEZ-VOUS
UN CLASSEMENT?(O/N)";:GET R$
 88 IF R$="O" THEN GOSUB 2000 ELSE IF
R$<>"N" THEN 87
 90 FOR I=1 TO N:CA(I,2)=CA(I,1)/T:CA(
I,3)=2*PI*CA(I,2)+CA(I-1,3):NEXT
 95 PAPER6
 100 HIRES:PAPER6:INK0
 110 CURSET 82,99,1:CIRCLE 70,1
 120 FOR I=1 TO N:CURSET 82,99,0
 130 DRAW COS(CA(I,3))*70,SIN(CA(I,3))
*70,1:NEXT
 140 FOR I=N TO 1 STEP-1
 150 AL=(CA(I,3)+CA(I-1,3))/2
 160 CURSET 82,99,0
 170 DRAW COS(AL)*50,SIN(AL)*50,0
 180 CHAR 64+I,0,1
 190 NEXT
 200 TI=LEN(TI$)*6:CURSET(238-TI)/2,10
,0
 210 TX$=TI$:GOSUB 500
 215 CURSET 220,26,0:CHAR 37,0,1
 220 H=155:V=30:FOR I=1 TO N
 225 PC=CA(I,2)*100:PC=INT((PC+.05)/.1
)*.1
 230 R$(I)=CHR$(64+I)+" "+NM$(I)+STR$(
PC):NEXT
 240 FOR I=1 TO N:CURSET H,V+I*13,0
 250 TX$=R$(I):GOSUB 500:NEXT
 260 CURSET 20,180,0:TX$="sur TOTAL"+S
TR$(T):GOSUB 500
 280 PRINT:PRINT"ENCORE--> E;QUITTER--
> Q;COPIE--> C";:GET R$:PRINT:PRINT
 290 IF R$="C"THEN GOSUB 57000:GOTO 28
0
 300 IF R$="E" THEN TEXT:RUN 20
 310 IF R$<>"Q"THEN 280 ELSE TEXT:END
 500 FOR P=1TO LEN(TX$)
 510 CHAR ASC(MID$(TX$,P,1)),0,1:CURMO
V 6,0,0:NEXT:RETURN
 1000 'TITRE
 1010 CLS:PAPER4:INK7:PRINTCHR$(17)
```

```
 1020 PRINT@10,4;CHR$(129);"C A M E M
B E R T"
 1030 PRINT@ 7,8;"Entrez le Nom du Tab
leau,"
 1040 PRINT@ 8,10;"puis chaque Partici
pant"
 1050 PRINT@11,20;"TAPEZ UNE TOUCHE"
 1060 GET R$
 1100 FOR I=25 TO 0 STEP-1:WAIT 5
 1110 PRINT@ I,I;CHR$(27)"R":NEXT
 1120 FOR I=0 TO 25:WAIT 5
 1130 PRINT@0,I;CHR$(27)"R":NEXT
 1140 PRINTCHR$(17)
 1150 RETURN
 2000 'CLASSEMENT
 2010 FOR J=1 TO N:E=CA(J,1)
 2020 FOR I=J TO N
 2030 IF CA(I,1)>=E THEN E=CA(I,1):K=I
:E$=NM$(I)
 2040 NEXT
 2050 CA(K,1)=CA(J,1):CA(J,1)=E
 2060 NM$(K)=NM$(J):NM$(J)=E$
 2070 NEXT:RETURN
 56999 '----------------
 57000 'HARD COPY HIRES/EPSON RX-80
 57010 LPRINT CHR$(27)"@";CHR$(27)"A"C
HR$(6);
 57020 DIM Z(200):C=0
 57030 FORAH=40961 TO 40999:I=0
 57035 IF AH>40961 THEN C=-64
 57040 FOR A=AH TO AH+7960 STEP 40
 57050 I=I+1
 57060 Z(I)=PEEK(A):NEXT
 57080 LPRINT CHR$(27)"*"CHR$(4);CHR$(
I);CHR$(0);
 57090 FOR J=I TO 1 STEP-1
 57095 Y=Z(J)+C
 57100 LPRINT CHR$(Y);:NEXT
 57110 LPRINT CHR$(13):NEXT
 57120 LPRINT CHR$(27)"@":RETURN
 63999 '---FIN DE LISTING---
```

En parcourant ces lignes, vous allez y retrouver de nombreuses vieilles connaissances ; rien ne doit être nouveau pour vous, sauf la façon dont nous traçons les secteurs de cercle. Il s'agit des fonctions mathématiques (trigonométriques) SIN et COS (ligne 130). Nous avions déjà traduit les divers pourcentages en angles exprimés en radians (ligne 90).

A la ligne 170, nous traçons des bissectrices invisibles, afin de disposer la lettre légende (A, B, C, D...) au cœur de chaque secteur du cercle.

Ce programme peut être utilisé à des fins professionnelles. La rapidité du dessin est surprenante.

FILL EN FILE

Programmer la couleur d'un rectangle FILL n'est pas chose simple ! Nous avons fouillé dans cette fonction afin d'obtenir jusqu'à trois couleurs différentes sur un même écran.

Elles dépendent des couleurs PAPER (P) et INK (I). Nous sommes obligés de vous parler des couleurs "complémentaires" (ou "opposées"). Pour les retrouver, tracer ces six lettres (initiales des couleurs) les unes sous les autres.

JMC 3 5 6
BVR 4 2 1

Ainsi, nous avons les complémentarités suivantes :
Jaune/bleu ; magenta/vert et cyan/rouge et bien sûr noir/blanc. En additionnant leurs codes, on obtient *toujours* 7, (3 + 4, 5 + 2, 6 + 1 et 0 + 7). Considérons aussi une "trace", un trait (DRAW) qui traverse le rectangle FILL. Soit K le troisième paramètre d'une instruction FILL.

K	63	191	138	
Rectangle	I	1/I	1/P	1/P
Trace	I	1/I	1/I	1/P

Dans ce tableau 1/I = complémentaire de la couleur INK et 1/P = complémentaire de la couleur PAPER.

```
10 'FILL EN 3 COULEURS
20 HIRES
30 PAPER2
40 INK1
50 CURSET 30,30,0
60 FILL 100,05,63
70 CURSET 80,40,1
```

```
80 FILL 100,05,191
90 CURSET 130,50,0
100 FILL 100,05,192
110 CURSET 180,60,0
120 FILL 100,05,138
130 CURSET 20,80,1:DRAW 210,0,1
140 PRINT"    63        191      192
  138";
190 END
```

Mais savez-vous que FILL peut conduire à de superbes fonds à rayures ?

```
200 'FILL EN RAYURES
210 HIRES
220 PAPER2
230 INK1
240 FOR I=1 TO 12:READ K:GOSUB 300
250 GET R$:NEXT
260 DATA 79,95,122,150,161,163
270 DATA 164,168,170,177,213,224
290 RESTORE:END
300 CURSET 100,50,0
310 FILL 100,7,K
320 PRINTCHR$(14);"        ";K;
330 RETURN
```

Vous aimerez certainement K = 170, et K = 150 vous surprendra.

Dans ces deux petits programmes de démonstration, évitez que INK + PAPER = 7.

Chapitre XIV

LES CALCULS SIMPLIFIES

Le but principal de ce chapitre est de vous éviter de sombrer dans des questions de détails, en vous livrant un éventail de petites formules très utiles et simples. Nous nous sommes bornés aux petits problèmes que l'on rencontre fréquemment. C'est de l'arithmétique simple, uniquement ; et comme il ne s'agit pas d'un cours de mathématique, nous ne ferons aucune démonstration. De l'utilitaire sans plus.

Mais tout d'abord, trois conseils :

1) Si votre formule vous semble trop complexe, n'hésitez pas à la fractionner en deux ou plusieurs formules. Autrement dit, procédez par étape.
2) L'ennemi juré est le risque d'une division par zéro. Veillez-y !
3) Dans une même formule, comptez soigneusement vos parenthèses : le nombre des "(" doit toujours être égal à celui des ")".

LES TRAITEMENTS D'UN NOMBRE NB

1) NOMBRE MAXIMAL DE DECIMALES

NB = INT (NB * A)/A

Pour 1 décimale A = 10
Pour 2 décimales A = 100
Pour 3 décimales A = 1000

2) ARRONDIR A L'ENTIER LE PLUS PROCHE

NB = INT (NB + .5)

3) ARRONDIR A UN MULTIPLE QUELCONQUE

On veut, par exemple, arrondir des nombres de telle sorte que le chiffre des unités soit 5 ou 0, mais la valeur la plus proche.

Exemple :

652 → 650 ; 654 → 655

Dans ce cas, on dit qu'on arrondit à « 5 près le plus proche » ; nous dirons pour cette formule générale à « R près le plus proche »

NB = INT ((NB + R/2)/R) * R

Avec ce petit programme de démonstration, amusez-vous à entrer pour NB et R les valeurs les plus fantaisistes (et décimales), voire négatives. Evitez seulement R = 0.

```
10 'ARRONDIR A R PRES
20 INPUT"R";R
30 INPUT"NOMBRE";NB
40 NB=INT((NB+R/2)/R)*R
50 PRINT"A";R;"PRES------>";NB:PRINT
60 GOTO20
```

Les puristes remarqueront que la formule n° 2 est une simplification de cette formule.

4) PARTIE ENTIERE SI NB PEUT ETRE NEGATIF

PE = INT (ABS (NB)) * SGN (NB)

5) MODULO

(Le modulo est la partie décimale d'un nombre).

MO = ABS (NB) − PE

(PE de la formule précédente).

LES TESTS SUR DES NOMBRES

1) TESTS SUR DES MULTIPLES

Exemple : (multiples de deux = nombres pairs)

IF NB/2 = INT (NB/2) THEN... (pair)

2) COMPARAISON DE DATES

Il suffit d'écrire la date dans l'ordre millésime, mois, jour, et de tout fusionner en un seul nombre, exemple 850702 pour le 2 juillet 85. Dans le petit passage suivant, vous pouvez entrer successivement 2,7 et 1985 ; tout sera formaté en 850702.

```
100 'FORMATAGE DE LA DATE
110 INPUT"JOUR";J$
120 INPUT"MOIS";M$
130 INPUT"ANNEE";A$
140 J$=RIGHT$("0"+J$,2)
150 M$=RIGHT$("0"+M$,2)
160 A$=RIGHT$(A$,2)
170 DA=VAL(A$+M$+J$):PRINT DA
```

Un tel formatage de la date dans un fichier va permettre un classement par âge ou par ancienneté.

3) NOMBRE DE JOURS ENTRE DEUX DATES

La méthode consiste à calculer le nombre de jours depuis le début du siècle (quel qu'il soit), et ce pour chacune des dates, et faire ensuite la différence. Une seule (mais petite) restriction : il faut que les deux dates soient dans lè même siècle. Nous prenons la suite du programme précédent afin d'entrer la date.

```
200 'JOURS DEPUIS LE DEBUT DU SIECLE
210 DATA 31,59,90,120,151,181,212,243
,273,304,334,365
220 DIM M(12):FOR I=1 TO 12:READ M(I)
:NEXT
230 J=VAL(J$):M=VAL(M$):A=VAL(A$):IF
A=0 THEN A=100
240 AP=A-1
250 JS=365*AP+INT(AP/4)+M(M-1)+J
260 IF INT(A/4)=A/4 AND M>2 THEN JS=J
S+1
270 PRINT JS;"JOURS"
290 END
```

En DIM M(12), nous stockons les jours cumulés de mois en mois.

Notre siècle commence en réalité le 1-1-1901 (et non le 1-1-1900 comme on aurait tendance à le croire). Par la ligne 230, l'an 2000 s'écrit dans la formule « 100 ».

Le calcul de la ligne 250 tient compte des années bissextiles antérieures, tandis que la ligne 260 fait cette correction sur l'année de la date, et à partir de mars.

Sur le plan pratique, il faudra mettre les lignes 210 et 220 au début de votre programme, alors que les lignes 230 à 260 seront appelées par un GOSUB.

Pour le 1-1-1901, notre formule donne JS = 1. L'an 2000 sera bissextile, 1900 ne l'était pas, parce que ses deux premiers chiffres (19) ne donnent pas un nombre divisible par 4.

Nos lecteurs historiens ont ainsi tous les éléments pour étendre cette formule dans le passé, avec des dates à cheval sur deux siècles.

INTERVALLE ALEATOIRE

Il s'agit d'un nombre aléatoire entre un minimum P (= petit) et un maximum G (= grand). Afin d'illustrer la formule (ligne 320), voici un petit programme pour ceux qui aiment jouer au loto (1 à 49).

```
 300 '*** L O T O ***
 310 CLS:P=1:G=49:FOR K=1 TO 7
 320 NB=INT(RND(1)*((G-P)+1)+P)
 330 FOR J=1 TO K
 340 IF NB=NB(J) THEN J=K:NEXT:GOTO 32
0
 350 NEXT:NB(K)=NB:NEXT
 360 PRINT@ 11,7;"JOUEZ CES NUMEROS":P
RINT:PRINT
 370 FOR J=1 TO 6:PRINT TAB(17);NB(J):
NEXT:PRINT:PRINT TAB(17);NB(7):PING
 380 PRINT@ 6,24;"ENCORE--> E ; QUITTE
R--> Q";:GET R$:IF R$="E" THEN RUN 300
 390 END
```

Le rôle des lignes 330 et 340 est de vérifier que le nouveau nombre n'ait pas déjà été sorti.

CALCULS DIVERS

1) TRANSFORMATION D'ANGLE DEGRES-RADIANS

Les fonctions trigonométriques de l'ORIC ne concernent que des angles exprimés en radians. Pour obtenir l'angle en radians AR à partir d'un angle en degrés AD :

$$AR = 2*PI*AD/360$$

(Pour des grades, remplacer 360 par 400).

2) FACTORIELLE

La factorielle est le produit des premiers nombres entiers.

Exemple :

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

```
400 'FACTORIELLE
410 INPUT"NOMBRE";NB
420 FACT=1
430 FOR I=1 TO NB:FACT=FACT*I:NEXT
440 PRINT NB;"!=";FACT:PRINT:GOTO 410
490 END
```

Au delà de 33! = 8.68331762 E + 36, c'est l'OVERFLOW ERROR.

3) RACINE CUBIQUE

```
500 'RACINE CUBIQUE
510 INPUT"NOMBRE";NB
520 N=ABS(NB):IF N=0 THEN 510
530 R0=EXP(LN(N)/3)*SGN(NB)
540 PRINT"RACINE CUBIQUE DE";NB;"=";R
0:PRINT:GOTO510
590 END
```

La rapidité de l'ATMOS est ici étonnante. Cela semble aussi instantané que pour une simple addition ! Et pourtant, un logarithme népérien et une exponentielle...

Pour d'autres racines il suffit, ligne 530, de remplacer "3" par tout autre nombre même décimal.

Grâce aux fonctions ABS et SGN, nous pouvons introduire des nombres négatifs ; ce que refusent généralement les calculatrices.

4) INTEGRALES

Nous ne calculons pas, nous mesurons ! C'est l'avantage d'un micro-ordinateur. Après avoir tracé un graphique en haute résolution, nous allons mesurer la surface en *pixels* entre deux points de l'axe des X, X1 ET X2.

Vous remarquerez que le "calcul" n'occupe que très peu de lignes (730 à 760), mais il est par contre assez long, vu les milliers de points (pixels) à compter.

```
 600 'MESURE D'INTEGRALE
 610 '---TRACE DU GRAPHIQUE
 620 CLS:HIRES:PAPER2:INK0
 630 CURSET20,10,0:DRAW 0,180,1:DRAW 2
20,0,1
 640 CURSET20,150,1:DRAW 200,-90,1
 650 A$="X DE 0 A 200":CURSET 80,10,0
 660 FOR I=1TO LEN(A$):C=ASC(MID$(A$,I
,1)):CHAR C,0,1:CURMOV 6,0,0:NEXT
 670 INPUT"X1";X1:X1=X1+20
 680 INPUT"X2";X2:X2=X2+20:PRINT"PATIE
NCE...";
 690 IF X1>220 OR X2>220 OR X1<20 OR X
2<20 OR X1>=X2 THEN 670
 700 CURSET X1,190,1:DRAW 0,5,1
 710 CURSET X2,190,1:DRAW 0,5,1
 720 '---MESURE
 730 FOR X=X1 TO X2:Y=190
 740 REPEAT:Y=Y-1:S=S+1
 750 UNTIL POINT(X,Y)=-1
 760 NEXT
 770 PRINT CHR$(14);"AIRE=";S;"points
(E/Q)";:GET R$
 780 IF R$="E"THEN RUN 600
 790 TEXT:LIST 600-
```

On peut ainsi faire des intégrales sur des courbes dont on ignore la fonction mathématique...

Chapitre XV

LE MAGNETOPHONE SANS ANGOISSE

Sur le plan magnétophone, l'ORIC ATMOS est au ''HIT PARADE'' pour les raisons suivantes :
— l'interface cassette est intégrée d'origine,
— ne nécessite pas un magnéto-cassette spécial à la marque,
— deux vitesses d'enregistrement dont une très rapide et fiable : 2400 bauds, soit environ 240 caractères à la seconde,
— on dispose en ROM de très nombreuses options d'enregistrements et de lectures,
— La fiabilité de ces enregistrements.

Ajoutons aussi la possibilité de remplacer le magnétophone par un lecteur de disquettes (DRIVE) ; nous en parlerons succinctement.

LE PRINCIPE

Il s'agit de transmettre des octets dans un sens ou dans un autre. On ne peut transmettre les 8 bits en une seule fois (il faudrait un magnétophone huit pistes...), alors on les envoie un par un, par trains de huit bits.

Enregistrons un programme par l'ordre CSAVE. Le début de l'enregistrement comporte d'abord des indications générales qui indiquent :
— s'il s'agit d'un programme ou d'un fichier ;

— si il est en BASIC ou en langage machine ;
— le nom de baptème que vous lui avez donné ;
— des indications de longueurs ;
— s'il devra s'en suivre un RUN automatique.

Viennent seulement ensuite, les lignes du programme, et enfin, un signal ''fin de l'enregistrement''.

Par conséquent, s'il manque un petit morceau du début ou de la fin, il serait impossible de charger ce programme par CLOAD (le C de CSAVE ou CLOAD signifie cassette).

Le codage de ces bits en signaux électroniques est très complexe (chaque fabricant a le sien), disons simplement que la *fréquence* de ces pics est très importante. Si votre magnétophone ne tourne pas à la vitesse standard de 4,75 cm/s, il serait incapable de lire des cassettes (du commerce) enregistrées à la vitesse correcte.

L'interface des ORIC tolère des niveaux de signal (amplitude) assez variables ; elle n'est tolérante qu'entre certaines limites mini et maxi. Lorsqu'il y a un problème pour charger un programme, c'est neuf fois sur dix parce que l'amplitude (= ''volume'') est trop forte ou trop faible.

La vitesse normale (rapide) nécessite un magnétophone ayant une ''bande passante'' relativement correcte dans les aigus (8 000 Hertz minimum). Si votre magnétophone est un modèle ''de bazar'', ou si sa tête est très usée, vous devrez utiliser la vitesse lente de 300 bauds (,S) ; c'est sécurisant mais affreusement long...

LE MAGNETOPHONE

Il est de bons magnétophones en musique qui sont quasi inutilisables pour micro-ordinateurs. Cette fameuse question de l'amplitude ! En effet :
— En position lecture, le bouton de volume de certains modèles n'agit que sur la sortie haut-parleur, tandis que le niveau de la prise sortie reste constant. En ce cas, il est généralement trop fort, mais il est alors facile de l'atténuer par le petit montage de la figure XV 1.
— La plupart des magnéto-cassettes n'ont pas de réglage de volume à l'enregistrement, il est automatique (C.A.G. = contrôle automatique de gain), car ce dispositif électronique coûte moins cher qu'un vu-mètre. Hélas, certains modèles ont un CAG qui donne un niveau ''moyen'' relativement bas, trop bas.

Si c'est votre cas, vous pouvez vous tirer d'affaire en utilisant la sortie HP supplémentaire, avec l'atténuateur de la figure XV 2, et en ajustant par le potentiomètre de volume. Cet ajustage devra être fait par tâtonnements successifs en partant d'un niveau très faible. Pas de surcharges sur l'ORIC !
— Le bouton de volume est actif en lecture mais il n'y a pas d'inter pour couper le HP incorporé, et le bruit est insoutenable. Enfoncez dans la prise pour HP supplémentaire une fiche mâle qui le déconnecte. Il est

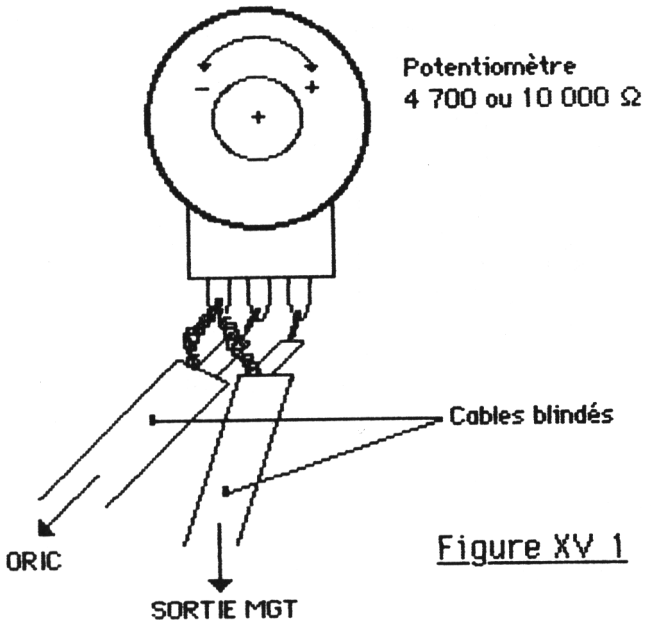
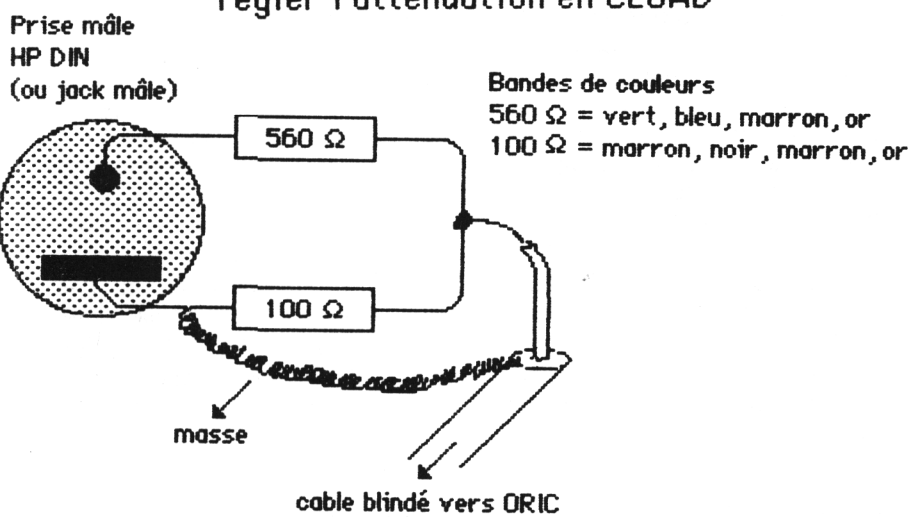


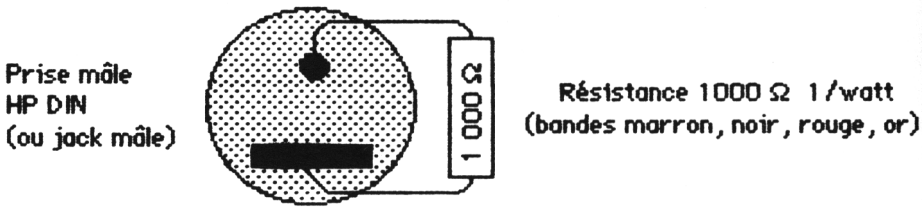
Figure XV 1

Montage d'un potentiomètre pour régler l'atténuation en CLOAD



Atténuateur sur la prise de sortie haut-parleur supplémentaire du magnétophone.

prudent de ''court-circuiter'' ses bornes par une résistance de 680, 820 ou 1 000 ohms. Voir figure XV 3.



Fiche pour couper le HP du magnétophone

Figure XV 3

Rappelons qu'il existe des magnéto-cassettes conçus pour micro-ordinateurs (exemple ''HERMES''). Ils possèdent un commutateur ''musique/micro-ordinateur''.

LES CASSETTES

La qualité normale dite « FERRO » est largement suffisante, mais méfiez-vous des marques ''bidon'' qui sont souvent chargées avec des rebuts de fabrications de grandes marques. En musique et paroles, cela peut suffire, mais si un manque provoque la perte d'un seul bit, le programme sera inchargeable !

Des cassettes C15 et C30 seraient suffisantes, mais il se trouve qu'elles sont actuellement aussi chères que les C60 ! De plus, il y a souvent absence de la marque. Le niveau d'enregistrement-lecture peut varier ; en musique, cela n'a guère d'importance, mais ici... Donc, si un modèle de cassette vous convient en rapport qualité-prix, restez-y fidèle. Mieux, faites en un petit stock d'avance.

Les cassettes C60 constituent un maximum. N'utilisez pas des C90, et surtout pas de C120 ! rembobiner une C60 est chose lente, alors n'utilisez pas la face B (sauf pour des sauvegardes en vitesse lente).

Une cassette de programme demande davantage de soin qu'une cassette de musique. Rembobinez toujours après usage, et gare à la poussière !

LES PRECAUTIONS ANTI-CHEVAUCHEMENTS

Nous avons expliqué pourquoi un enregistrement doit rester ''pur'' du début à la fin. Il lui faut en outre un SILENCE avant le début.

Les premières spires de la bande vont progressivement s'abîmer mécaniquement à l'usage. Si vous avez démarré un enregistrement quelques centimètres après l'amorce, ne vous étonnez pas si au bout d'une dizaine de fois, vous ne parvenez plus à le lire...

Un compteur sur le magnétophone est quasi obligatoire. Vous voulez enregistrer un programme neuf. Compteur à zéro. Départ bande en enregistrement, attendez dix secondes (ce qui peut faire trois au compteur) et là seulement, appuyez sur RETURN pour le CSAVE. Par flèche et CTRLA, repassez l'ordre CSAVE " " à l'écran. Dès que le compteur atteint un multiple de 10 (20 ou 50 par exemple), nouvelle action sur RETURN. Après la fin, encore trois à cinq secondes, STOP, rembobinez.

Cette sauvegarde est suffisante pour la cassette mais pas pour le programme ; pensez à un éventuel accident (ou perte) de cassette. Vous allez également le sauver sur une *cassette d'archivage* (une C60) qui reçoit vos divers programmes à la suite, mais avec des repères compteurs multiples de 10 ou de 20 (tenez son étiquette bien à jour !).

Pour les cassettes non-archivage, évitez d'y mettre plusieurs programmes, c'est vraiment pénible à charger. Une cassette vierge C60 est bon marché, ne soyez pas trop "radin"...

Vous voulez à présent remettre à jour (ré-enregistrer en écrasant l'ancien) un programme qui commence au repère 10. La super grosse bêtise consiste à s'amener en lecture sur 10, STOP, et PLAY en enregistrement + RETURN. S'il démarre seulement un à deux centimètres après le début de l'ancien, vous ne pourrez jamais le lire ! En effet, en CLOAD, L'ORIC est "stimulé" par le début de l'ancien départ, après, il ne reçoit plus ce qu'il attend ; il plante.

Voici comment il faut toujours procéder :
1) Se positionner à environ deux unités de compteur avant le repère départ, par exemple 8 pour 10, 78 pour 80...
2) Démarrer en enregistrement à ce repère en 8.
3) Quand le compteur arrive sur le repère 10 (ou 80...), on déclenche le CSAVE par RETURN

Donc la zone 8-10 est toujours écrasée par du "silence-amorce".

En position CLOAD, on se prépositionnera aussi en 8.

Pour la cassette d'archivage, sitôt l'enregistrement terminé, notez sur l'étiquette le repère compteur (multiple de 10) pour le prochain enregistrement ; avec une marge de sécurité confortable entre chaque programme, car l'un d'entre eux pourrait être "rallongé" par la suite.

LA TELECOMMANDE

Il est dommage que le manuel ATMOS soit si peu loquace sur cette possibilité. Sur la prise "TAPE" des ORIC, les broches n° 6 et 7 sont reliées aux contacts d'un petit relais interne. Celui-ci "colle" lorsque les fonctions CSAVE, CLOAD, STORE et RECALL sont en action. On peut donc relier ces broches par deux fils ordinaires non blindés à la prise

''REMOTE'' du magnétophone. Celui-ci ne tournera que lorsque le relais sera ''collé''. C'est pratique en position lecture, car la mise en route est recommandée alors par la touche RETURN et l'arrêt est automatique. En revanche, il faut être prudent en position enregistrement, car si le magnéto-cassette démarre lentement, le début de l'enregistrement risque d'être raté ou illisible par l'ORIC. Or, nous apprécions de pouvoir programmer des STORE en automatique, aussi faisons-nous un prédémarrage :

```
POKE 770,183:WAIT 150
STORE...
```

Ce POKE fait déjà coller le relais, et nous lui laissons 1,5 seconde de ''lancement'', donc pas le moindre risque. A ce propos, voulez-vous télécommander tout autre chose qu'un magnéto-cassette ? Un moteur, une lampe, etc... ? C'est ultra-simple. Pour faire coller le relais interne, c'est POKE 770,183, et pour le décoller c'est POKE 770,247 (ATMOS). Mais n'y faites pas passer du 200 volts ! Intercalez un relais intermédiaire de puissance ou un triac (voir du même auteur « FORMATION PRATIQUE A L'ELECTRONIQUE MODERNE » ; E.T.S.F).

LES OPTIONS

Prenez la sage habitude de vérifier de temps à autre un enregistrement par la fonction

VERIFY (ATMOS), CLOAD '' '', V

qui ne fait que comparer votre enregistrement au programme encore en RAM. Si des erreurs apparaissent, le niveau n'est pas bon, ou bien il serait grand temps de nettoyer la tête magnétique avec un coton tige imbibé d'alcool (pas d'autres solvants !).

Une autre option d'usage fréquent est :

CLOAD '' '', J

pour ajouter au programme en RAM un autre programme dont les numéros de lignes sont *obligatoirement* plus élevés (sinon les surprises seront sournoises...).

Le démarrage par RUN automatique par

CSAVE '' '', AUTO

présente parfois des ratés en lecture (relancer par RUN). Si tel est votre cas, savez-vous que la face 1 de la cassette de démonstration ATMOS est prévue pour cela ?

A propos de la sauvegarde d'une page d'écran (c'est parfois bien pratique !), nous nous devons de corriger une erreur relevée dans le manuel page 81 :

Pour sauver une page en HIRES, faire

CSAVE ''H1'', A 40960, E 48959

Le démarrage par RUN automatique par

CSAVE " ", AUTO

présente parfois des ratés en lecture (relancer par RUN). Si tel est votre cas, savez-vous que la face 1 de la cassette de démonstration ATMOS est prévue pour cela ?

A propos de la sauvegarde d'une page d'écran (c'est parfois bien pratique !), nous nous devons de corriger une erreur relevée dans le manuel page 81 :

Pour sauver une page en HIRES, faire

CSAVE "H1", A 40960, E 48959

et si vous voulez faire de même avec la petite bande texte inférieure :

CSAVE "T1", A 49000, E 49119

Pour relire ces fichiers d'écran, n'oubliez surtout pas de commander d'abord HIRES.

```
HIRES:CLOAD "H1"
POKE 26,96:CLOAD "T1"
```

Sans ce POKE, le "ready" ferait "scroller" vos trois lignes de texte. Il est amusant de voir l'image HIRES se former à mesure que la cassette se déroule.

QU'APPORTE LE LECTEUR DE DISQUETTES ? (DRIVE)

Essentiellement, la RAPIDITE à l'enregistrement comme à la lecture ; quelques secondes au lieu de plusieurs minutes. Autre avantage, l'accès direct pour lire un fichier ou un programme parmi une vingtaine sur la même face de "DISK". Cela rappelle la différence entre un album 33 tours et sa version cassette si vous désirez écouter l'air numéro quatre...

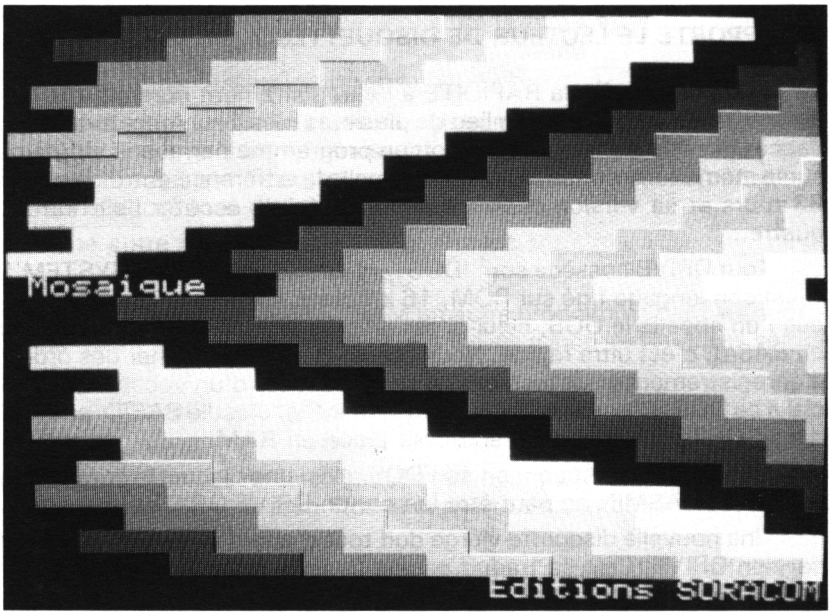
Tout DRIVE possède son "DOS" = "DISK OPERATING SYSTEM". C'est son langage figé sur ROM, 16 k-octets, comme le BASIC ! Lorsque l'on appelle le DOS, celui-ci est recopié en RAM *à l'emplacement du BASIC*, c'est ultra rapide. Nous pouvons alors lui donner des ordres d'enregistrement et de lecture : ce sont des mots d'un vocabulaire spécial à ce DOS. Ces manœuvres étant terminées, c'est le BASIC, en ROM dans l'ORIC, qui revient prendre sa place en RAM.

Chaque fabricant conçoit son DOS, ainsi une disquette enregistrée sur DRIVE JASMIN ne peut être lue par un DRIVE ORIC et vice versa.

Une nouvelle disquette vierge doit tout d'abord être "FORMATEE" par son DRIVE. Cela se traduit par un traçage magnétique des pistes de guidage, qui seront lues pour enregistrer ou lire. Il ne s'agit pas d'une piste en spirale mais de plusieurs pistes concentriques, elles-mêmes divisées en secteurs. Une piste spéciale est le "DIRECTORY", c'est l'équivalent de l'étiquette centrale d'un 33 tours. On n'y trouve le nom de chaque enregistrement, son volume en octets et sa position exacte

(piste/secteur), le catalogue. C'est le DOS qui le gère et le tient à jour automatiquement. On dispose de nouvelles options assez alléchantes, le DRIVE est donc bien plus intéressant que le magnéto-cassette ; un seul inconvénient mais de taille... son prix !

Le super luxe est d'opérer avec deux DRIVES (utilisations professionnelles) ; sur l'un, une disquette protégée contenant les programmes qui s'appellent successivement et mutuellement, et sur l'autre écriture et lecture de fichiers de résultats. Autre utilisation, duplication de disques.



Le dessin par les attributs de lignes ; hélas, ici en noir et blanc

et si vous voulez faire de même avec la petite bande texte inférieure :

CSAVE "T1", A 49000, E 49119

Pour relire ces fichiers d'écran, n'oubliez surtout pas de commander d'abord HIRES.

```
HIRES:CLOAD "H1"
POKE 26,96:CLOAD "T1"
```

Sans ce POKE, le "ready" ferait "scroller" vos trois lignes de texte. Il est amusant de voir l'image HIRES se former à mesure que la cassette se déroule.

QU'APPORTE LE LECTEUR DE DISQUETTES ? (DRIVE)

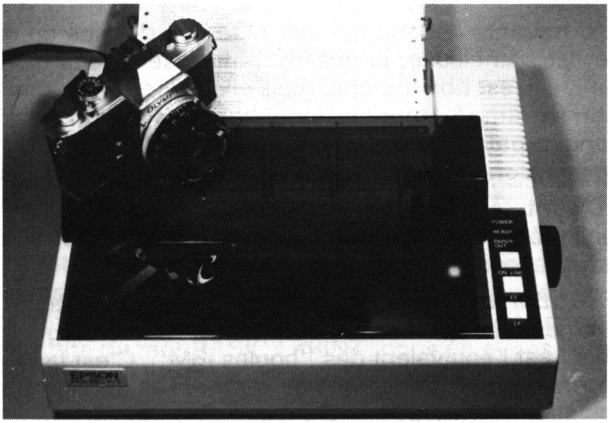
Essentiellement, la RAPIDITE à l'enregistrement comme à la lecture ; quelques secondes au lieu de plusieurs minutes. Autre avantage, l'accès direct pour lire un fichier ou un programme parmi une vingtaine sur la même face de "DISK". Cela rappelle la différence entre un album 33 tours et sa version cassette si vous désirez écouter l'air numéro quatre...

Tout DRIVE possède son "DOS"="DISK OPERATING SYSTEM". C'est son langage figé sur ROM, 16 k-octets, comme le BASIC ! Lorsque l'on appelle le DOS, celui-ci est recopié en RAM *à l'emplacement du BASIC*, c'est ultra rapide. Nous pouvons alors lui donner des ordres d'enregistrement et de lecture : ce sont des mots d'un vocabulaire spécial à ce DOS. Ces manœuvres étant terminées, c'est le BASIC, en ROM dans l'ORIC, qui revient prendre sa place en RAM.

Chaque fabricant conçoit son DOS, ainsi une disquette enregistrée sur DRIVE JASMIN ne peut être lue par un DRIVE ORIC et vice versa.

Une nouvelle disquette vierge doit tout d'abord être "FORMATEE" par son DRIVE. Cela se traduit par un traçage magnétique des pistes de guidage, qui seront lues pour enregistrer ou lire. Il ne s'agit pas d'une piste en spirale mais de plusieurs pistes concentriques, elles-mêmes divisées en secteurs. Une piste spéciale est le "DIRECTORY", c'est l'équivalent de l'étiquette centrale d'un 33 tours. On n'y trouve le nom de chaque enregistrement, son volume en octets et sa position exacte (piste/secteur), le catalogue. C'est le DOS qui le gère et le tient à jour automatiquement. On dispose de nouvelles options assez alléchantes, le DRIVE est donc bien plus intéressant que le magnéto-cassette ; un seul inconvénient mais de taille... son prix !

Le super luxe est d'opérer avec deux DRIVES (utilisations professionnelles) ; sur l'un, une disquette protégée contenant les programmes qui s'appellent successivement et mutuellement, et sur l'autre écriture et lecture de fichiers de résultats. Autre utilisation, duplication de disques.



Chapitre XVI

IMPRIMANTES ET PHOTOS D'ECRAN

Dès que l'on dépasse le stade du débutant et que l'on se lance dans des programmes de plusieurs kilo-octets, une imprimante devient presque indispensable pour ''s'y retrouver'', pour avoir une trace écrite de programmes anciens dans lequel on veut réutiliser un passage, etc...

On peut classer les imprimantes de trois façons.

1) Les standards (cas pour ORIC) et celles spéciales à tel type de micro-ordinateur.

2) Le mode d'impression mécanique (à billes, à aiguilles, thermiques, à marguerites, etc...).

3) Le mode d'entrée du signal, ''parallèle'' au standard CENTRONICS, les plus fréquentes, (cas pour ORIC), ou ''série'' au standard RS-232.

Pour les ORIC, deux bons points : l'adoption du standard CENTRONICS, et surtout le fait que l'interface est incluse sans supplément de prix (le cas est rarissime, hélas). Un mauvais point, son socle de sortie imprimante est un modèle très peu courant. Il est vrai que le socle ''normalisé'' est assez encombrant... Vous avez intérêt à acheter un câble « ORIC CENTRONICS » plutôt que d'essayer de le monter vous-même (fourni pour l'imprimante ORIC).

LES MODES D'IMPRESSIONS

— *A BILLES :* la petite imprimante ORIC utilise quatre stylos billes spéciaux, noir, bleu, rouge et vert. C'est plus qu'une imprimante, c'est aussi

une ''table traçante'', donc qui sait faire des dessins en quatre couleurs. Elle est silencieuse, la qualité, la finesse de ses caractères est stupéfiante, elle est bon marché, mais elle présente par contre un gros défaut, son extrême lenteur ! Lister un long programme est une opération de longue patience.

— *THERMIQUES :* c'est un papier spécial qui bleuit au contact de la chaleur. C'est très silencieux, moyennement rapide, mais d'une lisibilité et d'une présentation vraiment... peu flatteuses. Elles sont de moins en moins utilisées.

— *A MARGUERITES :* ce sont plutôt des machines à écrire automatiques commandées par micro-ordinateur. La ''marguerite'' interchangeable'' est l'équivalent des ''boules IBM''. C'est la super qualité courrier, donc pour le traitement de texte. Elles sont lentes (10 à 20 caractères/seconde) et très onéreuses.

— *A AIGUILLES* (ou ''matrices'') : elles utilisent du papier ordinaire, un ruban encreur sur lequel passe et repasse la tête d'impression. Dans celle-ci, un étagement de huit micro-aiguilles mues par huit électro-aimants dans un plan vertical. Cinq à six frappes horizontalement pour dessiner un caractère, 80 à 160 caractères par seconde, une ligne à l'aller, la ligne suivante au retour ! Une vitesse mécanique ahurissante, et c'est super robuste... et aussi très bruyant. Les marques les plus courantes sont SEIKOSHA et EPSON. Ce type d'imprimante est celui qui s'adapte le mieux aux besoins de la micro-informatique (et de la grosse informatique). On peut leur demander toutes sortes de caractères, même en double hauteur (un passage pour le haut, un autre pour le bas), des lettres minuscules en italiques, des caractères japonais ou définis par le programmeur, mais par contre pour les faire dessiner, c'est lourd et lent, et en une seule couleur, celle du ruban encreur.

LE COTE ELECTRONIQUE

Treize conducteurs relient l'ORIC à l'imprimante, dont huit pour chacun des bits de l'octet à transmettre, d'où le nom de transmission en ''parallèle''.

L'ORIC peut sortir des octets beaucoup plus vite que la machine peut les imprimer, aussi les autres fils instituent un dialogue entre eux, du genre « es-tu prêt à recevoir ? Oui, envoie ! Arrête ! Ça y-est, je les ai imprimés : prêt à les imprimer ? » etc...

Le circuit électronique d'une imprimante est aussi impressionnant que celui d'un micro-ordinateur. Elle a sa ROM qui contient son langage de conversation et d'exécution (aiguilles, mouvement de la tête, avance du papier, etc...), et sa mémoire vive RAM où elle stocke ce que lui envoie l'ORIC, à savoir d'une part les codes ASCII à imprimer sur la ligne en cours et d'autre part, des consignes permanentes reçues dès le départ : style des lettres, taille, etc...

LA MEMOIRE D'UNE IMPRIMANTE

Dès la mise sous tension, la plupart de ces paramètres d'écriture sont définis par défaut ; des conditions standard. De ce fait, il suffit de taper au clavier LLIST... ou LPRINT "..." et ça marche tout seul.

Si ces conditions "par défaut" ne vous conviennent pas, vous avez la possibité d'en modifier un certain nombre en ouvrant le capot de l'imprimante, afin de modifier certains commutateurs miniatures ("switches") ou "cavaliers".

Mais si exceptionnellement, vous voulez en modifier un ou plusieurs, vous le ferez par l'ORIC, ces ordres auront priorité sur les positionnements de ces switches. Ces ordres s'écrivent, en mode direct au programme, à la suite du mot BASIC LPRINT, nous en avons illustré quelques uns au chapitre XII.

Très important : ces consignes d'écriture *resteront en vigueur* tant qu'ils n'auront pas été annulés par un contr'ordre venant du micro-ordinateur, ou bien encore, et c'est bien plus simple, il suffit d'éteindre et de rallumer l'imprimante...

Neuf fois sur dix, un programme d'édition est truffé de tels ordres et l'édition terminée, le papier détaché, ces options d'écritures vont rester en service, et gare aux surprises si à la suite vous voulez lister un programme !

On reconnaît un programmeur consciencieux au fait qu'il *commence et termine* un programme d'édition par un ordre qui vide la mémoire de l'imprimante, c'est ce qu'on appelle *l'initialisation*.

Pour cette raison, lorsque l'on est plusieurs à travailler sur une même imprimante, exemple dans un "club-micro", il est prudent de l'éteindre et de la rallumer avant de s'en servir. Inversement, si vous venez d'imprimer une ligne en caractères géants, videz la mémoire après usage, car si après, on vient faire un listing...

LA MISE EN PAGE

Nous laissons de côté les listings et nous abordons les programmes d'éditions.

Comme pour une lettre ou une page d'écran, une page d'édition doit être bien présentée : titres en plus gras caractères (ou soulignés), et bien centrés. Sauts de lignes pour "aérer", etc...

Pour les centrages, nous vous conseillons d'utiliser la fonction SPC, du genre :

```
LPRINT SPC (29);A$
```

car dans la pratique, c'est souvent moins lourd à programmer que les fonctions de tabulations de l'imprimante.

Si vous utilisez du papier à feuilles prédécoupées (pointillages perforés), il va falloir calculer les sauts de pages, de telle sorte que si au

départ on prépositionne le papier à ras la réglette de découpage, il faut qu'une fois le travail terminé, le papier s'arrête avec une séparation de feuille au même endroit. Disons tout de suite que ce n'est pas facile !

Il faut établir une formule qui donnera le nombre de lignes à sauter (ex. LS) en fin de chaque page; puis écrire :

FOR I=1 TO LS:LPRINT:NEXT

LES PHOTOS D'ECRANS

Même avec très peu de matériel, mais avec de bonnes méthodes, vous obtiendrez tout de suite 100 % de réussite.

1) La recommandation n° 1 va peut-être vous surprendre : *Nettoyez l'écran* ! Pas seulement avec un chiffon, mais avec un coton imbibé d'alcool (ou d'eau de cologne) ; vous serez surpris de la couche de crasse que l'électricité statique du tube a attiré ! Si vous ne nettoyez pas le verre, cette couche se devinera sur les photos, surtout dans les angles avec une perte de netteté.

2) Votre appareil photo est certainement à obturateur rideaux, alors la vitesse la plus rapide sera 1/4 seconde. S'il s'agit d'un obturateur central, vous aurez droit au 1/15 s, pas plus rapide. Si vous ne respectez pas cela, vous aurez des phénomènes stroboscopiques. Des zones plus sombres en diagonales. Par conséquent, un film de 100 ASA constitue un maximum, car pour une brillance d'écran normale, attendez-vous à f:11au 1/4 s, ou mieux encore, f:16 à 1/2 s.

L'appareil sera sur pied, *bien en face de l'écran* (ce n'est pas évident...), c'est-à-dire son axe optique bien perpendiculaire à l'écran. Vous pouvez conserver la focale normale, mais surtout pas de grands angulaires...

Tout est bien réglé ? Armé ? Alors allez *éteindre l'éclairage de la pièce* ! Ce n'est pas pour améliorer le contraste, mais tout simplement pour que l'appareil photo ne capte pas son "propre" reflet dans le verre de l'écran... Dans le viseur, on ne le remarque jamais, mais sur l'épreuve ou sur la diapo, c'est franchement ridicule.

Si vous souhaitez que votre photo paraisse dans la presse, évitez les fonds noirs, les techniques d'encrages risquent de la transformer en "tas de charbon", si c'est de la couleur, de la diapositive exclusivement (type "lumière du jour").

LA TRICHERIE

Comment faire une photo nette au quart de seconde d'un passage animé ?

Il faut arrêter le programme sur le bon endroit. Pas à la main par CTRL C, mais en souillant le programme d'une instruction d'arrêt (celui

mis en RAM, pas celui sur cassette). Pas de STOP, ni de END qui vont afficher cet horrible ''ready'', mais plutôt un WAIT 1000 glissé au bon endroit.

Alors ? Est-ce que tout cela était vraiment difficile ?

CONCLUSION

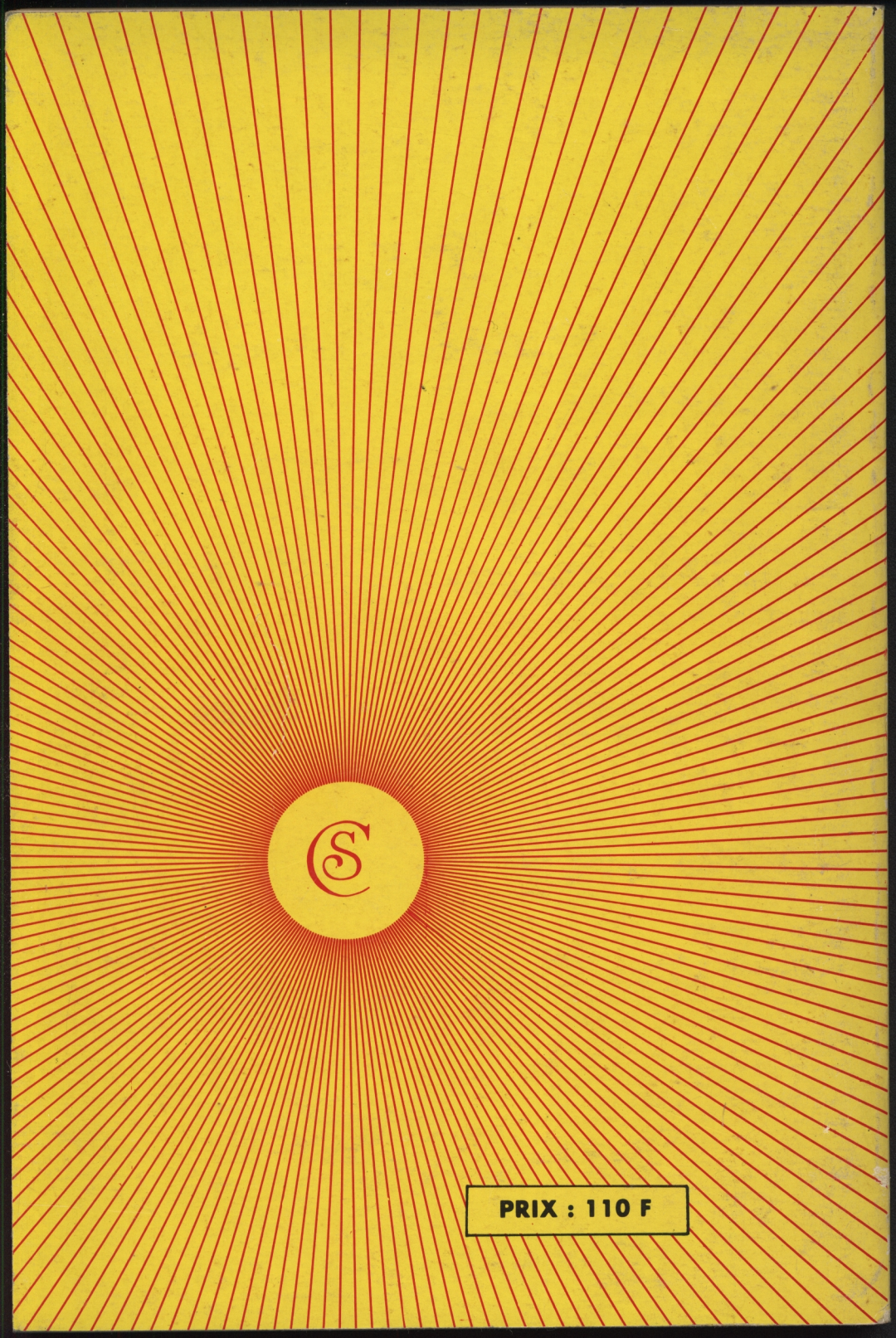
(Puisqu'il en faut toujours une !)

Notre bourrage de crâne est terminé, et peut-être, voyez-vous maintenant votre clavier autrement qu'en noir et rouge. N'ayez aucune crainte, ce clavier a l'air d'être très résistant puisque celui de l'auteur ne donne toujours pas de signe de fatigue. Lancez-vous à fond ; vous avez les armes, il ne manque plus que l'entraînement.

Un conseil, ce sera le dernier, c'est promis : faites attention à vos heures de sommeil en retard, car plus on programme vite, plus on se couche tard.

TABLE DES MATIERES

Composition : Fideptex
Maquette : SORACOM
Impression : JOUVE - Mayenne
N° d'éditeur : 041
N° d'impression : 14213
Dépôt légal : Avril 1985



PRIX : 110 F

MIEUX PROGRAMMER SUR ORIC 1 ET ATMOS

Michel Archambault