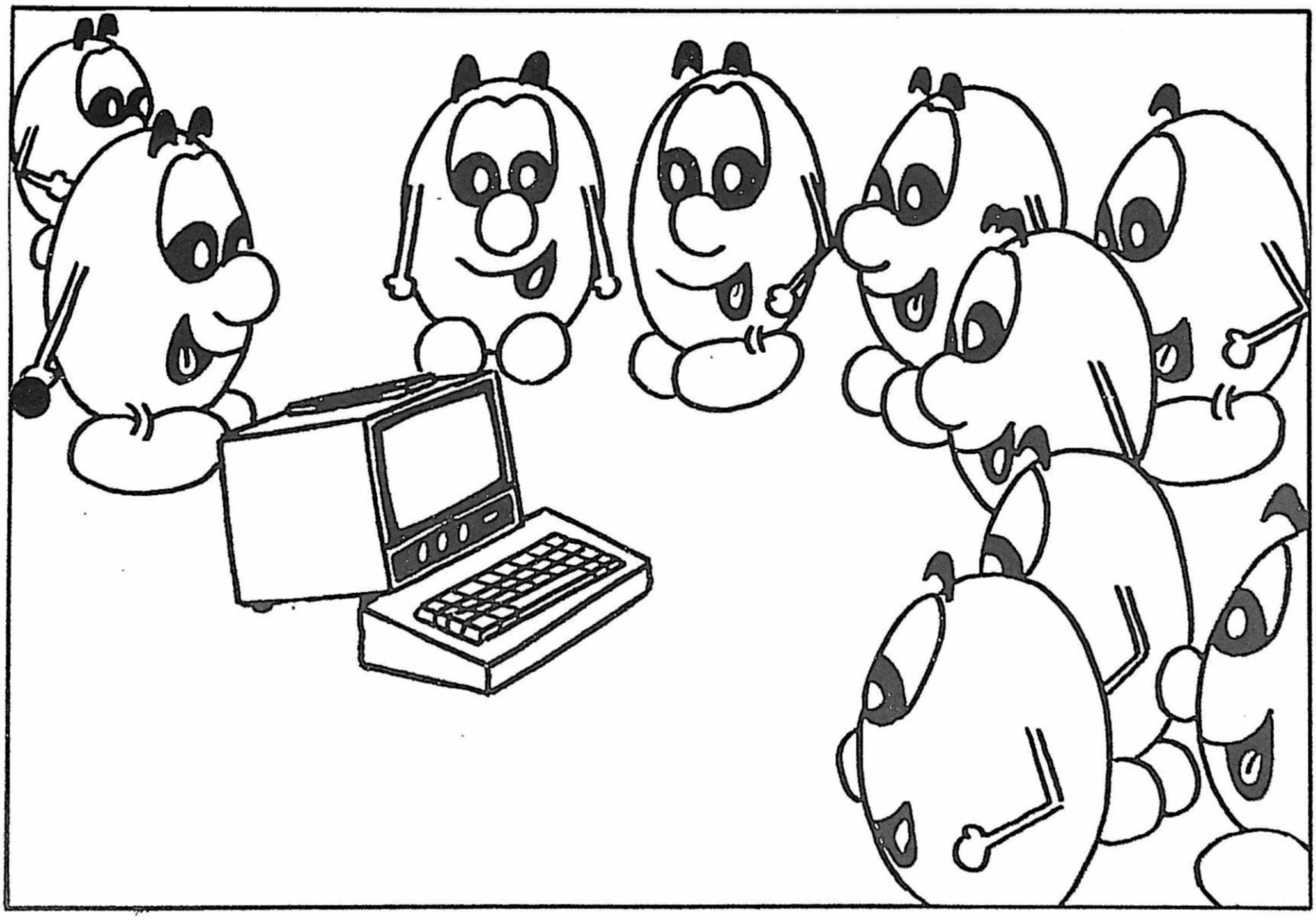
LA PROGRAMMATION DU SMAKY II

Daniel Roux

Février 1979



epsitec-system sa

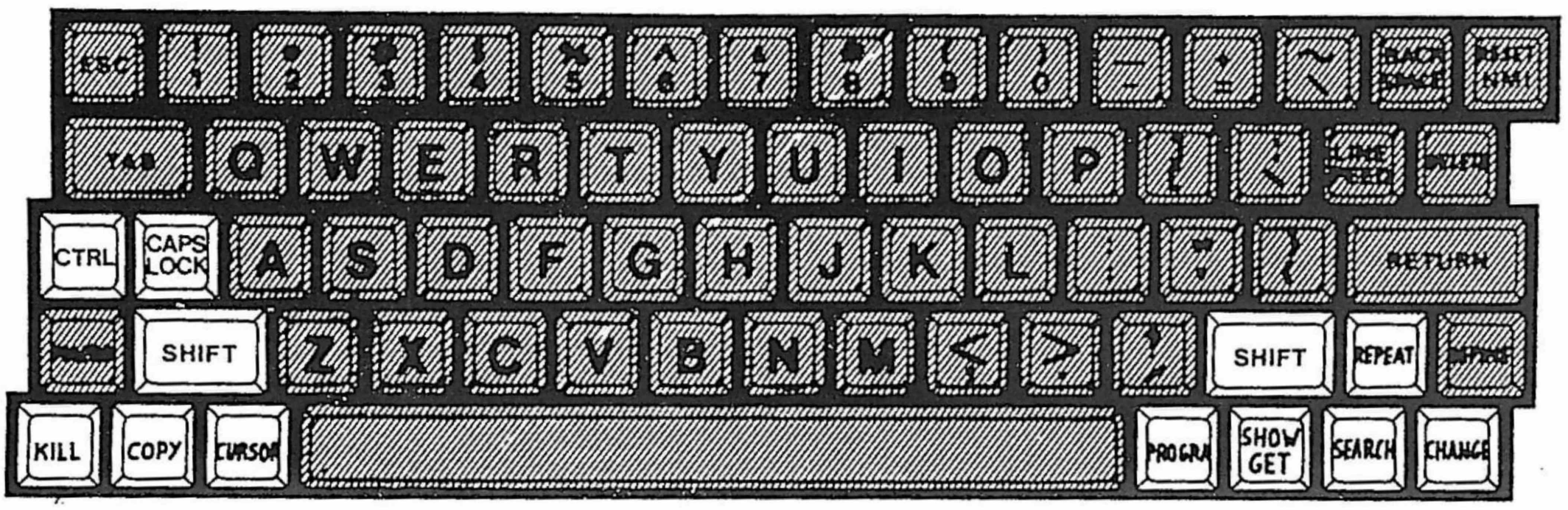
LA PROGRAMMATION DU SMAKY6 2ème partie

4. Programmation en octal avec le moniteur
   4.1 Programmation en assembleur avec SMILE

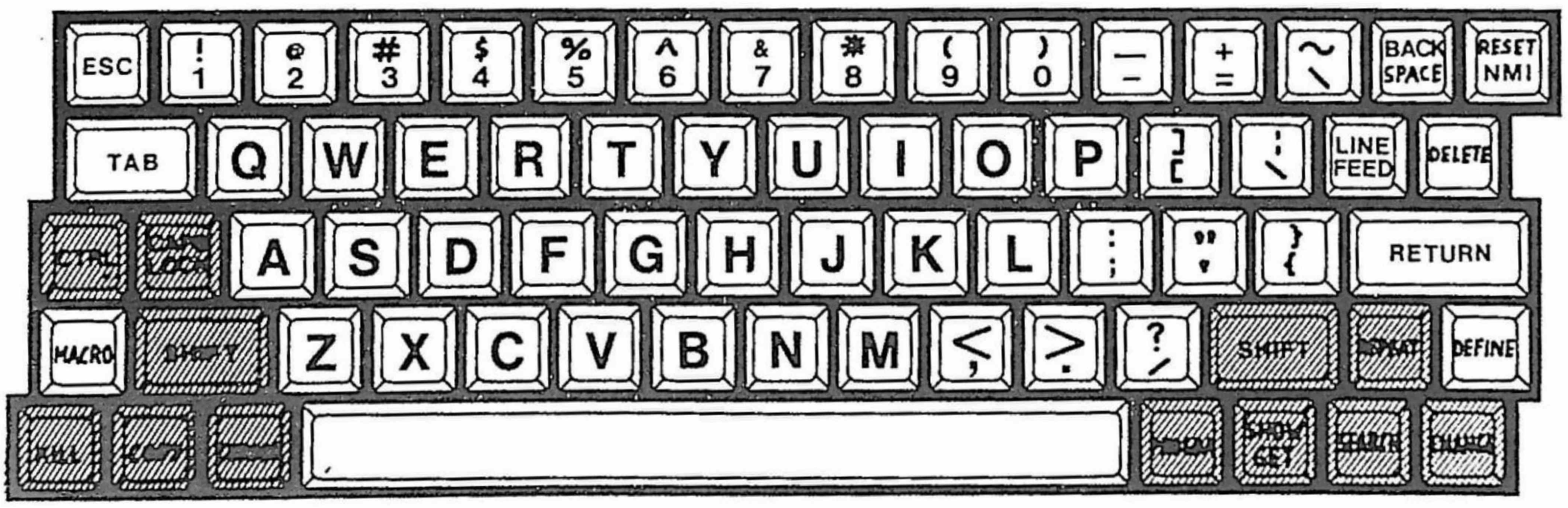
4. PROGRAMMATION EN OCTAL AVEC LE MONITEUR

Le SMAKY a un clavier de 69 touches réparties en deux catégories:

1) Les touches à action immédiate

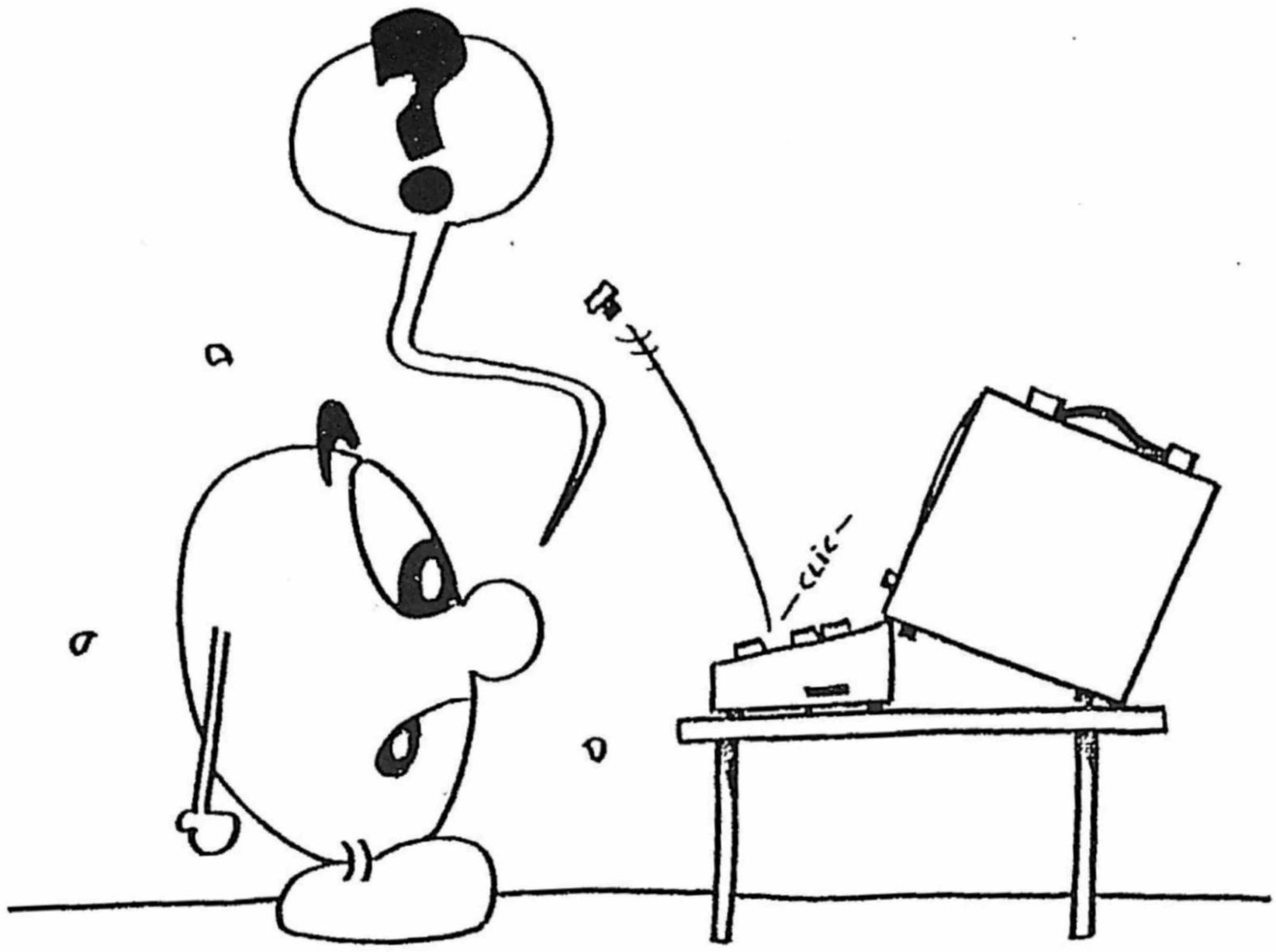


2) Les touches qui n'ont une action que si elles sont associées à une touche de la catégorie 1.



Par exemple, pour faire CTRL A, appuyez sur [CTRL] puis, en maintenant cette touche pressée, appuyez sur [A]. Relachez rapidement les deux touches, car, après environ 1 seconde, l'ordre que l'on vient de taper se répéte.
Par la suite on notera cette opération [CTRL][A].

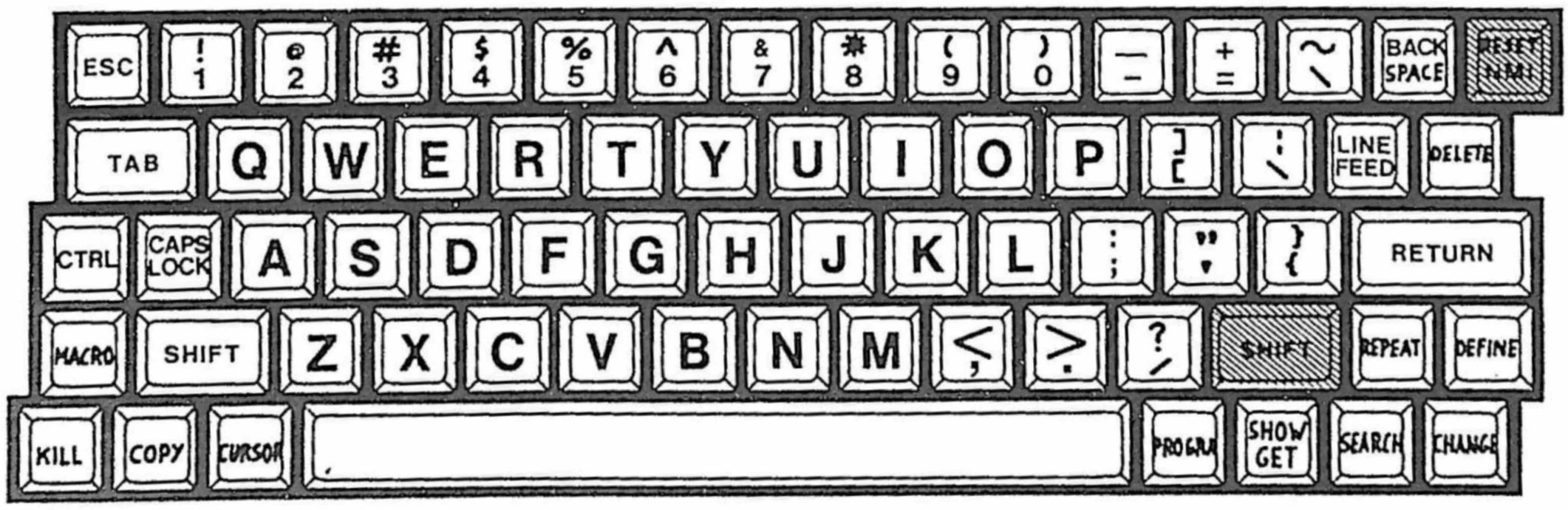
Les deux carrés qui se touchent indiquent, que, lorsque [A] est pressé, [CTRL] doit l'être aussi. Au contraire, [A][B] veut dire qu'il faut appuyer sur [B] après avoir relâché [A] .



Ecrivons maintenant un petit programme en octal avec notre SMAKY.
Pour cela nous allons nous servie du programme moniteur, qui est dans une mémoire morte.

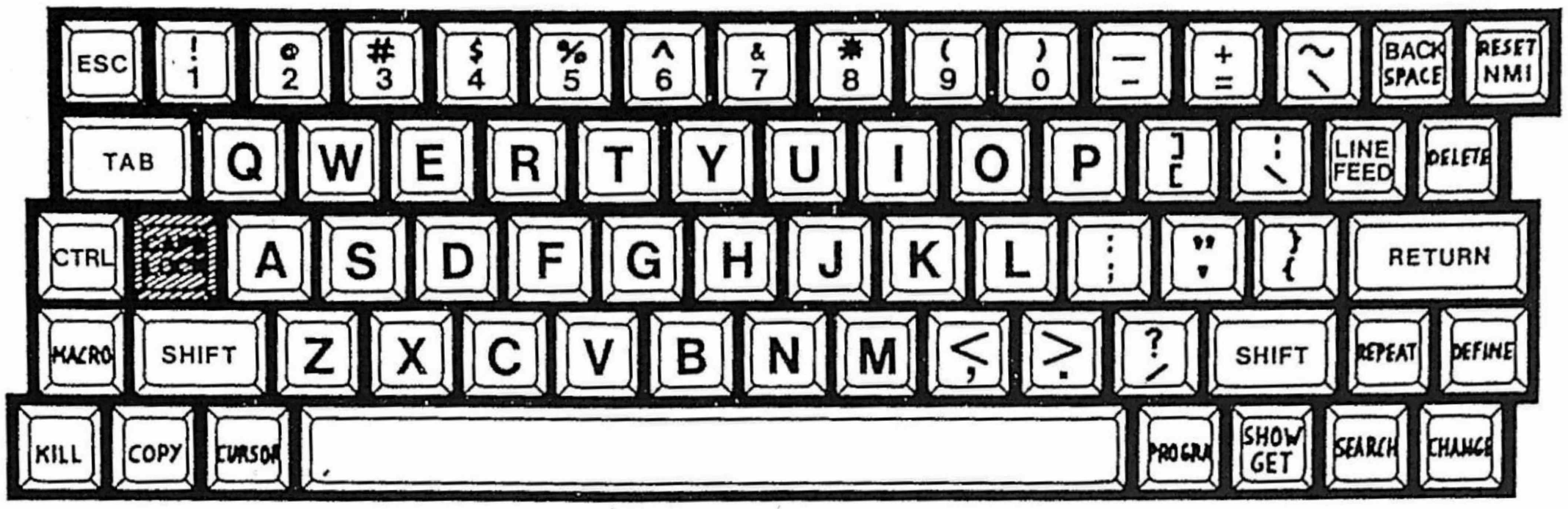
1) Enclenchez le SMAKY à l'aide de l'interrupteur situé sur l'alimentation.

2) Faire [SHIFT][RESET NMI]



Le message SMAKY6 SYS 1-3 * 32k apparaît sur l'écran

3) Vérifier que la touche [CAPS LOCK] est enfoncée



4) Tapez la séquence 

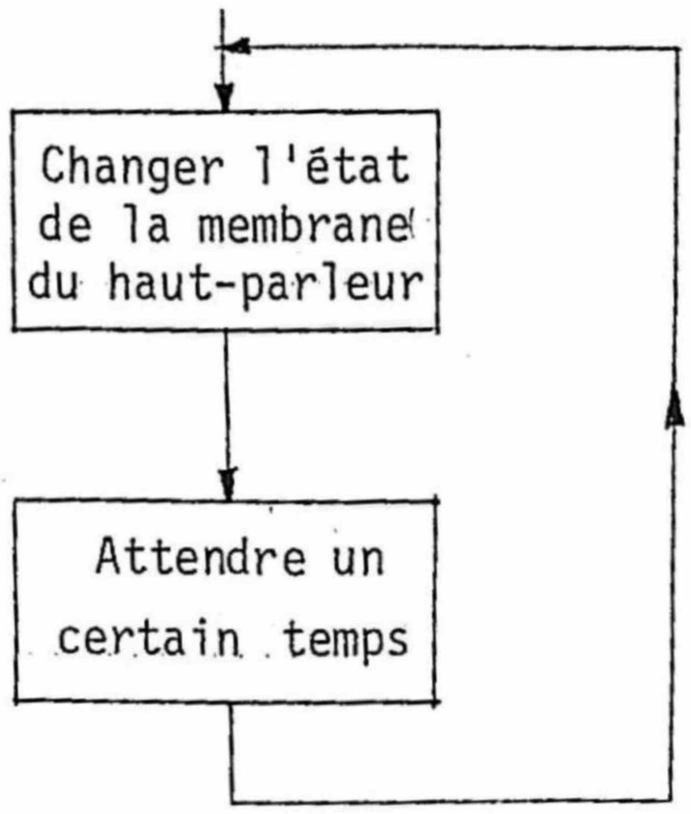
qui, par la suite, sera notée 

Le texte MON 1-3 apparaît sur l'écran. Vous êtes prêts pour travailler avec le moniteur.

La membrane du haut-parleur ne peut avoir que deux positions qui sont:
- . en repos
- . attirée.

Pour commander le haut-parleur, une seule instruction est possible: celle de changement d'état de la membrane.

L'algorithme de notre programme sera donc:



et il s'écrit en octal:

adresse	contenu	
40400	323	} change l'état du haut-parleur
40401	3	
40402	20	} constante de temps
40403	376	
40404	30	} saut à l'adresse 40400
40405	372	

Remarque: l'adresse 40400 correspond au 1er caractère de la cinquième ligne sur l'écran. L'adresse 40401 correspond au deuxième caractère de la cinquième ligne, etc. Ainsi, nous verrons notre petit programme sous forme de six caractères bizarres sur l'écran.

L'instruction 323 suivie d'un 3 fait partie de toute une famille d'instructions qui peuvent s'écrire de la façon suivante:

323
n

Cette instruction transfère le contenu de l'accumulateur A du processeur dans le registre de l'interface ayant pour numéro n. Pour le SMAKY, le haut-parleur correspond au numéro 3.

Lorsque nous effectuons l'instruction | 323 / 3 |, nous ne connaissons pas le contenu de l'accumulateur A, mais cela n'a pas d'importance. En effet, l'interface pour le haut-parleur est construit de telle façon que, quelle que soit la valeur de A, l'instruction | 323 / 3 | change la position de la membrane.

L'instruction 20 suivie de 376 a pour effet d'introduire un délai entre deux changements d'état de la membrane du haut parleur, afin de rendre audible la fréquence de commutation.

L'instruction 30 suivie d'un 374 fait partie de toute une famille d'instructions qui peuvent s'écrire:

30
$\ell'-2$

Cette instruction effectue un saut d'une longueur égale à ℓ'. Si ℓ' est positif, le saut se fait en avant et si ℓ' est négatif, le saut se fait en arrière.

Dans notre cas, il faut reculer de 4, donc $\ell' = -4$.

On peut alors se poser la question suivante: comment représenter un nombre négatif avec 8 bits ? La méthode utilisée par les microprocesseurs est celle du "complément à deux".

Exemple:

+5	→	005	et en binaire	0 0 0 0 0 1 0 1
+2	→	002		0 0 0 0 0 0 1 0
+1	→	001		0 0 0 0 0 0 0 1
0	→	000		0 0 0 0 0 0 0 0
-1	→	377		1 1 1 1 1 1 1 1
-2	→	376		1 1 1 1 1 1 1 0
-3	→	375		1 1 1 1 1 1 0 1
-4	→	374		1 1 1 1 1 1 0 0

Donc, pour notre instruction, il faut prendre la valeur ℓ' -2, soit 374 -2, ce qui fait bien 372.

Pour écrire ce petit programme en mémoire vive à l'adresse 40400, il faut faire les manipulations suivantes:

40400 — ? /

Il apparaît sur l'écran:

323 — > .

S

3 — > .

J

20 — > .

H

376 — > .

÷

30 — > .

X

372 — > .

Z

Note: par la suite ? / sera noté (OPEN)

et > . sera noté (NEXT)

Et pour exécuter notre petit programme qui est maintenant en mémoire vive, il faut faire:

40400 — G

Note: par la suite G sera noté (GO).

On entend maintenant un son.

Pour arrêter l'exécution de ce petit programme et reprendre le contrôle avec le moniteur, presser sur RESET NMI

Cette méthode d'écriture reste très proche de celle qui consisterait à écrire les programmes en binaire. C'est pourquoi nous allons étudier maintenant une méthode beaucoup plus puissante appelée ASSEMBLEUR.

4.1 PROGRAMMATION EN ASSEMBLEUR AVEC SMILE

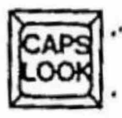
Nous allons récrire le même petit programme que celui expliqué au chapitre 3.1 mais avec un outil beaucoup plus puissant, l' ASSEMBLEUR.

Pour cela nous nous servons du programme SMILE qui se place en mêmoire vive. Il nous faudra donc l'introduire avec un MICROLERU et une bande papier.

Pour charger une bande papier:

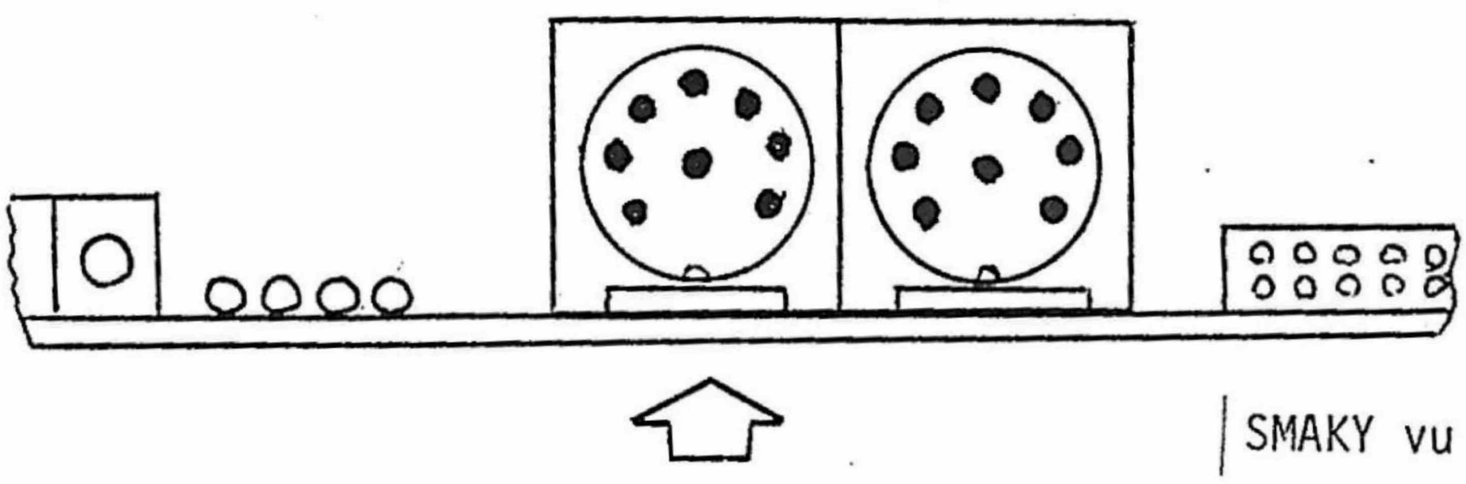
1) Enclenchez le SMAKY

2) Faire 

3) Vérifiez que la touche 

est enfoncée.

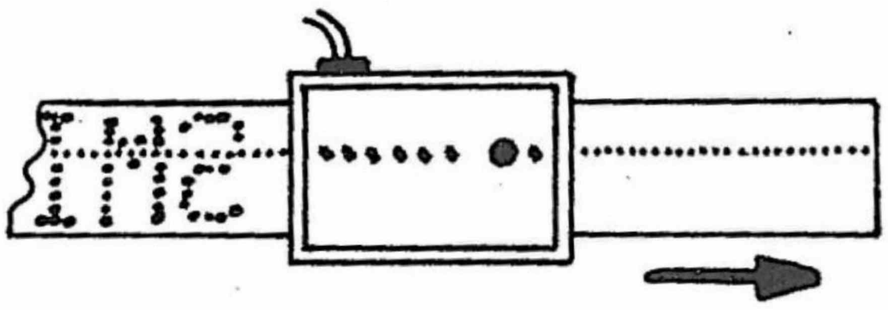
4) Branchez le MICROLERU dans la prise de l'USART 4. La lampe rouge sur le MICROLERU doit s'allumer.



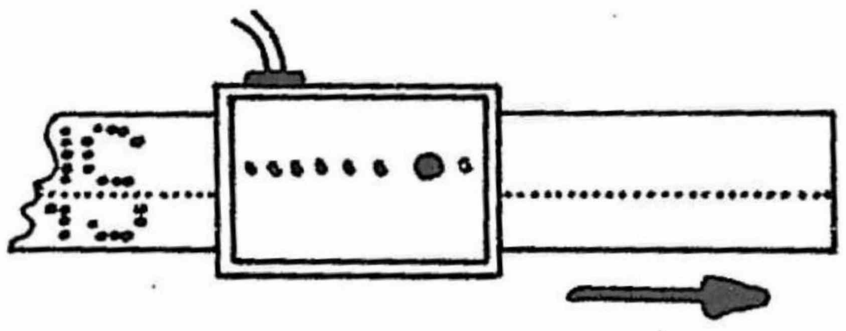
SMAKY vu de derrière

MICROLERU

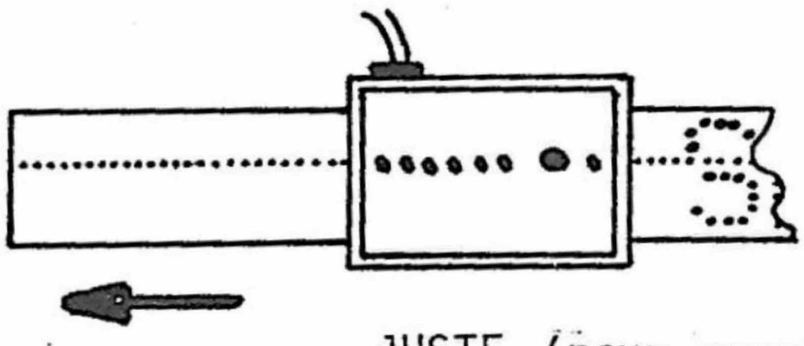
5) Introduisez le début de la bande dans le MICROLERU



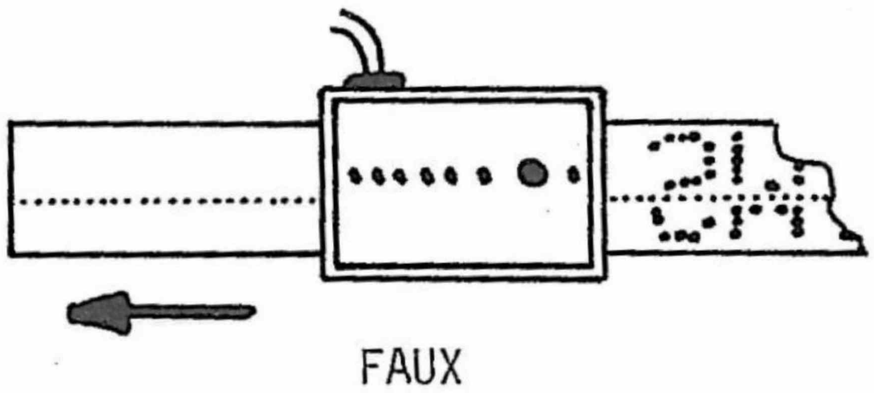
JUSTE (pour droitiers)



FAUX

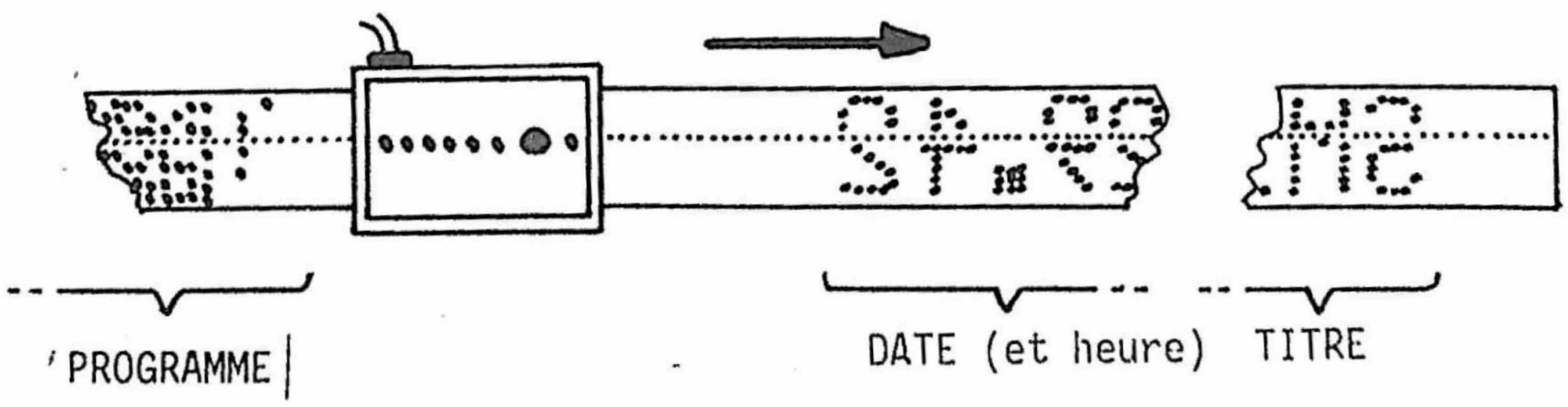


JUSTE (pour gauchers)



FAUX

6) Tirez la bande jusqu'avant le début du programme



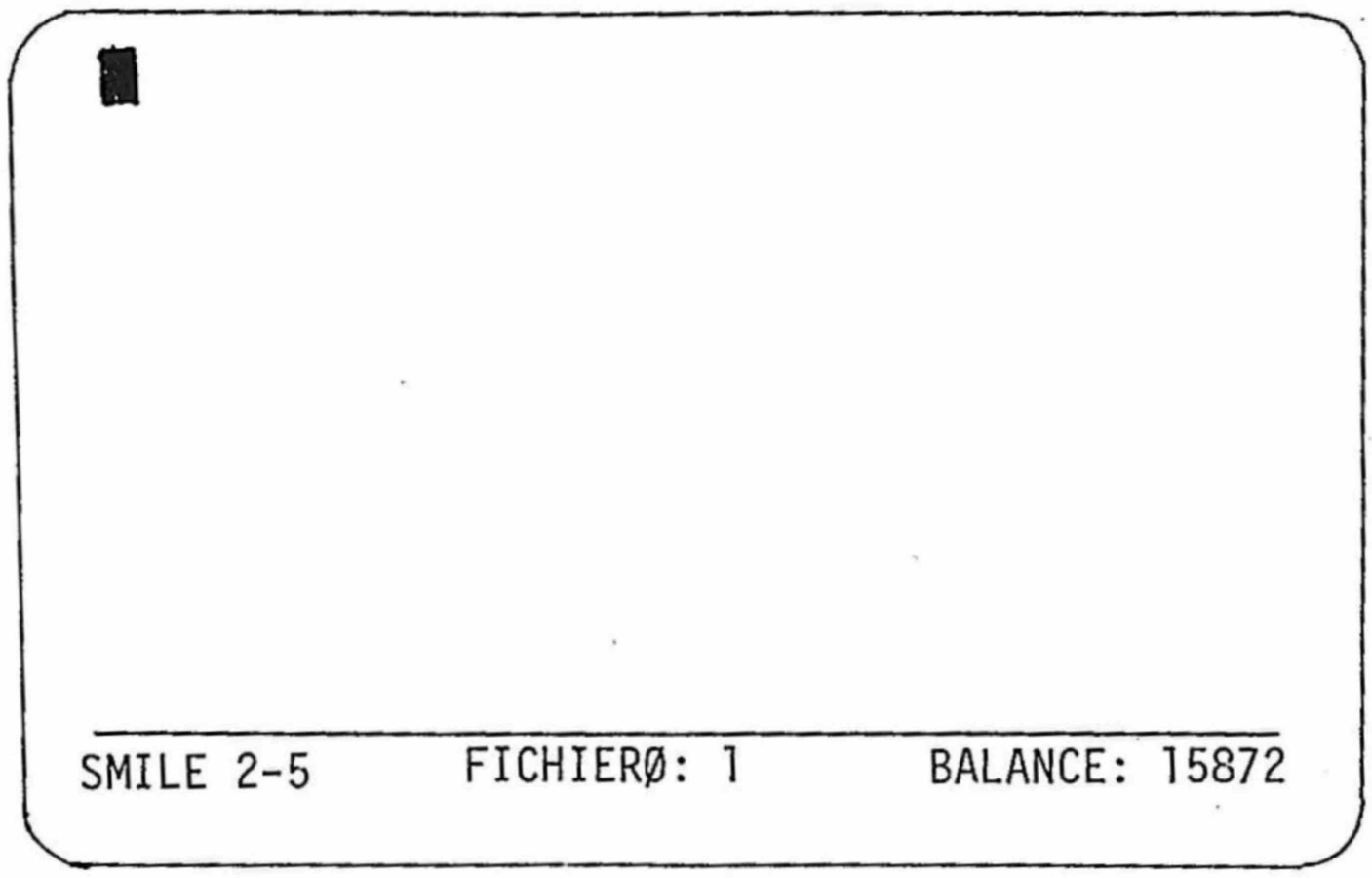
PROGRAMME | DATE (et heure) TITRE

7) Tapez [LOAD♪]

8) Tirez la bande (pas trop vite) jusqu'à la fin. A chaque perforation, le haut-parleur fait un tac. Sinon, vérifier les prises et recommencer au point 4. Il est possible de s'arrêter, mais il faut faire attention de ne pas reculer.

En cas d'erreur (plus de son dans le haut-parleur), tirez la bande en arrière d'environ 50 cm, refaire [LOAD♪] puis continuez à tirer.

Lorsque vous êtes arrivés à la fin de la bande, l'écran affiche:



Vous êtes maintenant prêt à travailler avec SMILE.

Notre petit programme s'écrit maintenant:

```
HP      = 3

TAC:  | LOAD    $HP,A
LOOP: | DECJ,NE B,LOOP
      | JUMP    TAC
```

Pour écrire la première ligne, faire: [H] [P] [TAB] [+ =] [# 3] [RETURN]

Par la suite, on notera [HP→=3↵] avec →=[TAB] et ↵=[RETURN]

Pour écrire la deuxième ligne, qui est vide, il suffit de faire: [↵]

Et pour la suite:

[TAC:→LOAD→$HP,A↵]

[LOOP:→DECJ,NE→B,LOOP↵]

[→JUMP→TAC↵]

REMARQUE: Pour le :, il faut faire [SHIFT] [: ;]

Pour le $, il faut faire [SHIFT] [$ 4]

En cas de faute de frappe, une pression sur la touche [DELETE] efface le dernier caractère, en remontant dans le texte.

Ne pas oublier, après la dernière ligne (JUMP TAC) de mettre un [RETURN] .

Il faut maintenant transformer ce texte en un code compréhensible par le processeur. Cette opération s'appelle ASSEMBLER.

Pour assembler notre petit programme, tapez:

[PROG] [Z]

SMILE a mis maintenant quelque part en mémoire vive, les six bytes (323, 3, 20, 376, 30, 372) que nous avions tapé avec le moniteur au chapitre 4.

Essayons notre petit programme en tapant:

[PROG] [V]

Le processeur exécute maintenant ce programme et nous entendons le son produit dans le haut-parleur. Pour arrêter l'exécution de notre petit programme et retourner dans SMILE, il faut faire [RESET NMI] .

EXPLICATION DU PROGRAMME

- L'instruction `LOAD $HP,A` correspond aux deux bytes | 323 / 3 | et indique qu'il y a transfert (LOAD) de l'accumulateur A dans le périphérique haut-parleur. HP a été défini précédemment comme valant 3 (HP = 3) et $ indique qu'il s'agit d'un périphérique.

L'instruction LOAD HP,A voudrait dire "transfert de l'accumulateur A dans la position mémoire ayant pour valeur HP (3 dans notre cas)", ce qui n'aurait pas de sens pour notre application.

- L'instruction `LOOP: DECJ,NE B,LOOP` correspond aux deux bytes | 20 / 376 | et a pour effet d'introduire un délai. Cette instruction sera expliquée un peu plus loin.

- Et finalement, l'instruction `JUMP TAC` qui correspond aux deux bytes | 30 / 372 |, a pour but d'effectuer un saut (JUMP) à l'adresse TAC. Cette adresse a été définie par TAC: juste avant LOAD $HP,A (TAC: LOAD $HP,A)

```
          HP  =   3

323       TAC:    LOAD      $HP,A
 3

 20       LOOP:   DECJ,NE   B,LOOP
376

 30                JUMP      TAC:
372
```

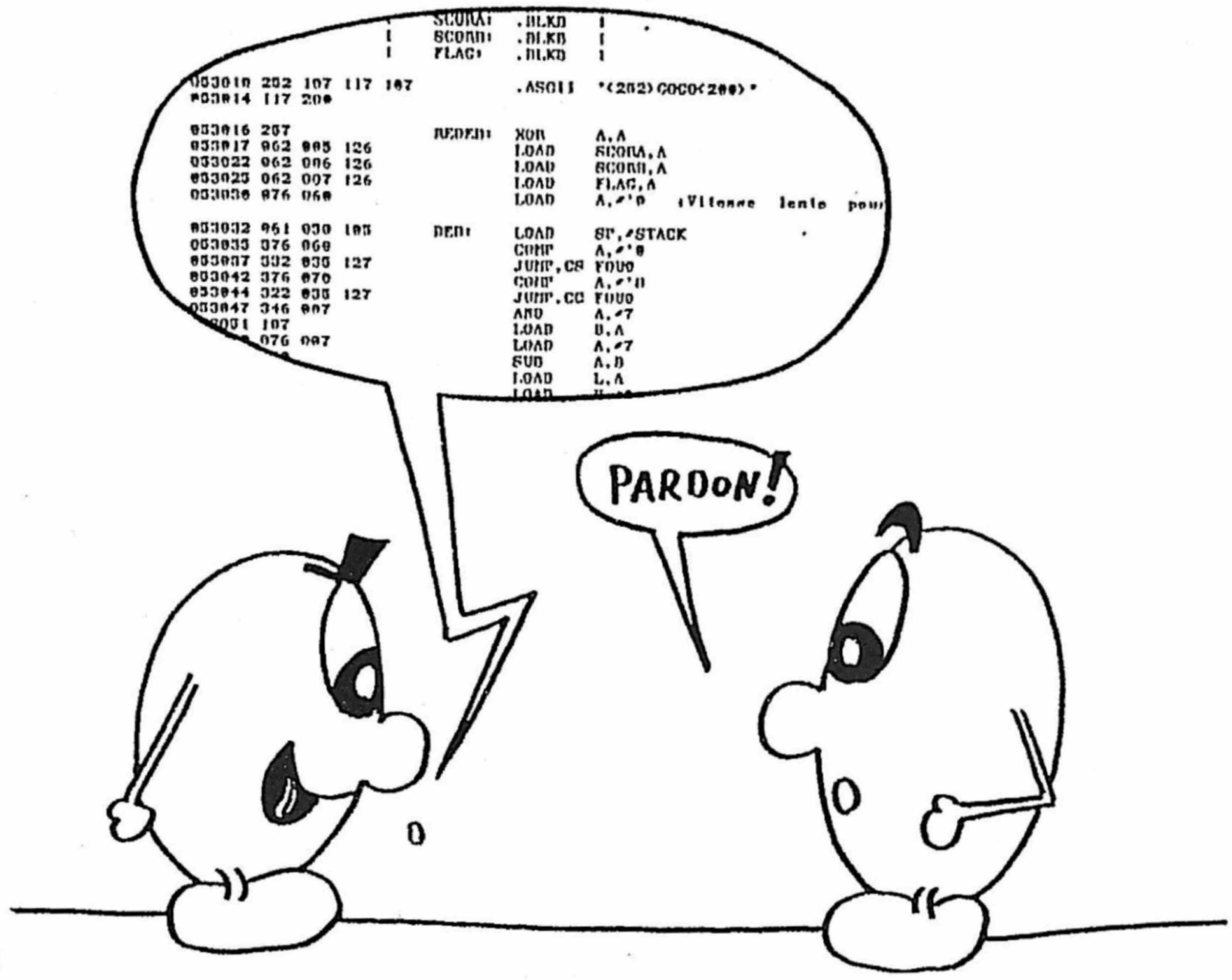
Essayons maintenant de changer la fréquence du son précédent. Pour cela, il faut comprendre l'instruction `LOOP: DECJ,NE B,LOOP`.

Cette instruction est en fait une association de deux instructions qui peuvent s'écrire:

```
LOOP:  DEC       B
       JUMP,NE   LOOP
```

- L'instruction DEC B veut dire que le contenu du registre B (qui est un registre du processeur, tout comme A) est décrémenté (DEC), c'est à dire qu'on lui soustrait 1. Par exemple, s'il valait 127 avant que l'instruction DEC B ne soit effectuée, il vaudra 126 après.

- L'instruction JUMP,NE LOOP effectue un saut (JUMP) à l'adresse LOOP, mais seulement si B est différent de zéro (NE).



EXEMPLE:

Si le registre B vaut 12 et que nous exécutons l'instruction DECJ,NE B,LOOP le registre B va donc d'abord être décrémenté (il vaudra alors 11). Puis comme B est différent de zéro, B va être encore décrémenté et vaudra 10. Et ainsi de suite jusqu'à ce que B atteigne zéro.

Nous pouvons constater deux choses:

1) Plus B est grand avant d'exécuter DECJ,NE B,LOOP et plus le temps d'exécution de cette instruction sera long.

2) Quelle que soit la valeur de B avant l'exécution de DECJ,NE B,LOOP il est certain que B vaudra zéro après cette instruction.

Dans notre exemple, nous n'avons jamais initialisé le registre B. Donc, la première fois que l'instruction DECJ,NE B,LOOP est exécutée, nous ne pouvons pas savoir le temps que cela prendra.

Mais, avant le deuxième passage, B vaudra zéro et le temps d'exécution sera maximal, car, en décrémentant zéro, on trouve

377, 376, 375, 374, 373, ... 3, 2, 1, 0

Pour pouvoir choisir la fréquence du son, il suffit d'initialiser B avant l'instruction DECJ,NE B,LOOP.

Le programme devient alors:

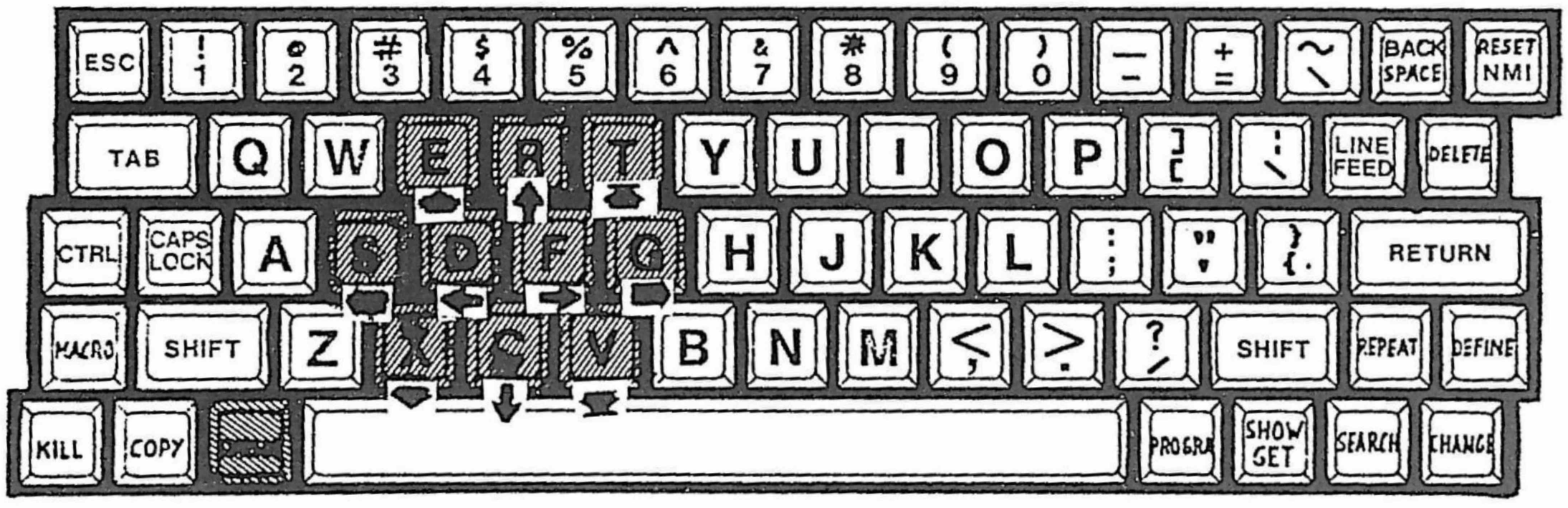
```
        HP =     3
        DUREE=   100

TAC:    LOAD     $HP,A
        LOAD     B,# DUREE
LOOP:   DECJ,NE  B,LOOP
        JUMP     TAC
```

L'instruction LOAD B,# DUREE veut dire que la valeur DUREE est transférée (LOAD) dans le registre B. Le signe # indique que B est chargé par la valeur DUREE, et non pas par le contenu d'une position mémoire d'adresse DUREE (ce qui s'écrirait: LOAD B,DUREE).

Pour pouvoir essayer, il faut rajouter deux lignes dans notre programme.

Vous voyez sur l'écran, que le pointeur (carré blanc clignotant) est tout en haut à gauche sur le H de HP = 3. IL est possible de le déplacer n'importe où dans notre programme grâce à la touche CURSOR et d'une touche de déplacement

CURSOR F	avance le pointeur d'un caractère	→
CURSOR D	recule le pointeur d'un caractère	←
CURSOR C	descend le pointeur d'une ligne	↓
CURSOR R	remonte le curseur d'une ligne	↑
CURSOR X	descend le pointeur de 4 lignes	⇓
CURSOR E	remonte le pointeur de 4 lignes	⇑
CURSOR G	avance le pointeur d'un mot	⇒
CURSOR S	recule le pointeur d'un mot	⇐
CURSOR V	place le pointeur à la fin du texte	⇩
CURSOR T	place le pointeur au début du texte	⇧



Il vous faut faire:

CURSOR C — ♪DUREE→=1ØØ

CURSOR C — CURSOR C — ♪→LOAD→B,# DUREE

Ensuite, PROGRA Z pour assembler, puis PROGRA V pour essayer, et finalement RESET NMI pour arrêter.

On remarque que le son est plus aigu.

EXERCICE:

Changer la valeur de DUREE et essayez.
Pour cela, placez le pointeur à la fin de la ligne DUREE = 100, faire trois fois [DELETE] pour effacer le 100, puis tapez une nouvelle valeur (par exemple 300,200,10,1).

Pour détruire complètement le programme, il faut placer le pointeur au début ([CURSOR] [T]), puis faire [KILL] [CURSOR] [V]