CHAPITRE VI
FICHIERS ET ENTRÉES-SORTIES

- FICHIERS :
  - FICHIERS PROGRAMMES
  - INSTRUCTION EXECUTER
  - FICHIERS DONNÉES
  - FICHIERS PROGRAMMES DÉCODÉS
  - COMMANDE SUPPRIMER
  - AUTRES COMMANDES SUR FICHIERS

- ENTRÉES-SORTIES
  - VOIES D'ENTREE-SORTIE
  - COMMANDES ENTREE ET SORTIE
  - COMMANDE PERSEVERER
  - COMMANDE STANDARD

FICHIERS L.S.E.

Il existe trois sortes de fichiers en LSEG-EDL :
- les fichiers programmes (extension .LSP)
- les fichiers données (extension .LSD)
- Les fichiers programmes décodés (extension .LST)

Remarques :

Dans tout ce qui suit, le terme « DISQUE » désignera un support pouvant contenir des fichiers. Si le « DISQUE » est en réalité une cassette, l'utilisateur devra gérer le positionnement de la bande afin d'assurer un fonctionnement correct.

Rappelons que le LSEG-EDL connaît comme unités de disque les unités numérotées :
.0 pour le lecteur de cassette
.1 pour le premier lecteur de disquette
.2 à .4 pour les lecteurs de disquettes suivants (numérotés de bas en haut).
Le disque de travail est, par défaut, le disque .1 si un lecteur de disquette sous tension est connecté sinon le dique .0 (lecteur de cassettes). Cette assignation peut être changée à tout moment par la commande Disque (cf. « Manuel de Référence »).

VI.1 FICHIERS PROGRAMMES

On peut sauvegarder un programme que l'on vient de taper en utilisant la commande RAnger

Exemple :

RA⊠NGER PROG⊠ créera un fichier programme de nom PROG.LSP sur le disque de travail. Une erreur sera détectée s'il existe déjà un fichier de ce nom (qui alors ne sera pas altéré).
On peut spécifier le numéro de disque dans le nom du programme : PRO.2 par exemple.
Les noms de programme se construisent selon les règles édictées au sujet

des identificateurs L.S.E. (5 caractères alpha-numériques maximum, le premier étant alphabétique). Notons qu'il est possible en LSEG-EDL d'avoir des noms de programmes pouvant aller jusqu'à huit caractères mais l'utilisation de cette possibilité est déconseillée pour des raisons de portabilité.

Nous pouvons charger un programme qui se trouve sur le disque en utilisant la commande APpeler

Exemple :

AP⊠PELER PROG.1 ⊠
Une erreur sera détectée si le programme n'existe pas sur le disque spécifié (ou sur le disque de travail si le numéro de disque n'a pas été précisé).
La commande LAncer simule un APpeler suivi d'un EXécuter à partir de 1.

Exemple :

LAncer PROG chargera le programme de nom PROG et lancera son exécution.

Nous avons vu tout à l'heure que le système refuse de ranger un programme si celui-ci existe déjà sur le disque. Or il arrive souvent que l'on ait besoin de modifier ou de mettre à jour un programme.

Nous devons donc appeler ce programme, le modifier puis remplacer sur le disque l'ancienne version par la nouvelle. Cela se fait par la commande REmplacer.

Exemple :

RE⊠MPLACER PROG.1 ⊠
Une erreur sera détectée s'il n'existe pas de fichier PROG.LSP sur le disque 1.

VI.2 INSTRUCTION EXÉCUTER

EXECUTER ec
EXECUTER ec, ea
ec désigne un nom de fichier programme
ea désigne un numéro de ligne de début
Cette instruction provoque le chargement puis le lancement du programme à partir de la ligne de numéro (ea) s'il y a deux paramètres, à partir de 1 dans le cas contraire.

Cette instruction permet de chaîner des programmes, mais elle ne permet pas le passage de paramètres entre les programmes. Ces paramètres ne pourront être passés qu'à l'aide d'un fichier de données.

VI.3 FICHIERS DONNÉES

Un fichier données L.S.E. est constitué d'enregistrements repérés par des numéros allant de 1 à 32000. Ces numéros ne sont pas nécessairement consécutifs (il peut y avoir des « trous » et, par exemple, un fichier peut être constitué de 3 enregistrements portant les numéros 5, 47 et 321). Il faut noter que la version actuelle de LSEG-EDL limite les fichiers à 84 enregistrements maximum.
Un fichier données L.S.E. est repéré par son nom (même règle que pour les identificateurs) éventuellement suivi d'un numéro de disque.
Chaque enregistrement d'un fichier données L.S.E. est l'image d'un identificateur (contenu et type). On peut avoir dans un même fichier des enregistrements contenant des informations de natures différentes (numérique, chaîne, tableau,... etc).
Pour créer un fichier données L.S.E., il suffit de créer un enregistrement. A l'inverse, supprimer tous les enregistrements détruit le fichier.

Les instructions portant sur les fichiers données sont : GARER pour créer et/ou remplir des enregistrements, CHARGER pour lire des enregistrements et SUPPRIMER pour les détruire.

Exemple :

```
1
10 CHAINE SALUT;SALUT←'Bonjour'
20 TABLEAU T[3];T[1]←34;T[2]←56;T[3]←89
30 BOOLEEN BOO;BOO←.VRAI.
40 E←2.7182818;PI←3.1415927
50 GARER E,3,'FICH';GARER T,8,'FICH';GAR
ER SALUT,9,'FICH'
60 GARER BOO,21,'FICH';GARER PI,567,'FIC
H'
70 TERMINER

EXECUTER A PARTIR DE 1

TERMINE EN LIGNE 70
EFFACER LIGNES *
```



```
1
10 CHARGER A,3,'FICH';CHARGER B,8,'FICH'
20 CHARGER C,567,'FICH';CHARGER D,21,'FI
CH'
30 CHARGER E,9,'FICH'
40 AFFICHER [/,U,/,U,/,U,/,U,/,U,/]A,B,C
,D,E
50 CHAINE F;F←'C''est fini';GARER F,9,'F
ICH'
60 CHARGER G,9,'FICH';AFFICHER G;TERMINE
R

exECUTER A PARTIR DE 1

2.7182818

34 56 89

3.1415927
.VRAI.
Bonjour

C'est fini
TERMINE EN LIGNE 60
```

Noter la déclaration implicite des variables B, D, E et G respectivement comme tableau, booléen, chaîne et chaîne.
Les instructions GARER et CHARGER peuvent être suivies d'un paramètre compte rendu :
GARER X, N, NOMFI, CR1 et CHARGER Y, NUM, FIC, CR2
Ce paramètre compte rendu aura une valeur positive ou nulle si l'opération s'est correctement effectuée. (Voir dans la partie Manuel de Référence les valeurs prises en cas de problème).

L'instruction SUPPRIMER s'emploie de la façon suivante :
30 SUPPRIMER 'FICH', 9 ; * Cette ligne supprime l'enregistrement numéro 9 du fichier FICH.
340 SUPPRIMER 'TOTO', *; * Cette ligne détruit le fichier de nom TOTO.
L'instruction SUPPRIMER peut éventuellement utiliser un paramètre de compte-rendu (cf. Manuel de Référence).

VI.4 FICHIERS PROGRAMMES DÉCODÉS

Ces fichiers portent une extension .LST et sont créés par une commande LIster lignes ou un AFFICHER exécuté après une affectation de la voie de sortie sur un fichier (cf. pages suivantes). Il est indispensable d'utiliser ensuite la commande STandard pour refermer un tel fichier qui sera disponible en lecture par une commande ENtrée.

VI.5 COMMANDE SUpprimer

Cette commande permet de supprimer des fichiers programmes, programmes décodés, ou donnés. On l'utilise de la façon suivante :
SUpprimer PROG.LSP supprime le fichier programme PROG du disque de travail.
SUpprimer FICH.LSD,3 supprime l'enregistrement numéro 3 du fichier données FICH du disque de travail.
SUpprimer FICH. LSD supprime le fichier données FICH du disque de travail.
SUpprimer TOTO. LST supprime le fichier programme décodé TOTO du disque de travail.

VI.6 AUTRES COMMANDES

LSEG-EDL n'a pas d'autre commande au sujet des fichiers.
Les commandes :
UTilisation des fichiers
TAble des fichiers
DUpliquer
CAtaloguer
ainsi que :
FOrmater disquette
COpier disquette
ne sont pas implémentées et peuvent remplacées par des utilitaires (des procédures externes généralement).
Le C.N.D.P. diffuse une disquette qui contient de tels utilitaires.

ENTRÉES/SORTIES

VI-7 LES VOIES D'ENTRÉE et SORTIE

Les transferts de caractères entre la mémoire centrale et les différents périphériques (écran, clavier, imprimante, unités de disque, magnétophone...) se font au moyen d'opérations d'entrée/sortie.

En L.S.E. on distingue :
- d'une part les périphériques qui constituent des voies physiques;
- d'autre part les voies logiques numérotées à partir de 0 et qui peuvent être utilisées par exempe dans des instructions LIRE ou AFFICHER.

La voie logique 0 est celle qui est utilisée pour le traitement des commandes et des instructions LIRE ou AFFICHER sans format (ou ayant choisi la voie 0). Les autres voies logiques ne peuvent être utilisées que par les instructions LIRE et AFFICHER avec format.

Les voies physiques sont de deux sortes :
- celles qui correspondent à des périphériques et qui sont symbolisées par :
  . 10 pour clavier ou écran
  . 20 pour imprimante ou clavier sans écho
  . 30 pour voie V24
- celles qui correspondent à des noms de fichiers qui sont symbolisées par le nom du fichier.

Les assignations standard sont :
- en entrée 0 au clavier
  1 au clavier sans écho
  2 à la voie V24
- en sortie 0 à l'écran
  1 à l'imprimante
  2 à la voie V24

VI-8 COMMANDES ENTRÉE et SORTIE

Les assignations standard peuvent être modifiées par les deux commandes ENtrée et SOrtie. Elles utilisent un complément de commande identique et ont pour syntaxe :
ENtrée [voie logique =] voie physique
SOrtie [voie logique =] voie physique

En l'absence de voie logique, c'est la voie logique 0 qui est concernée. La voie physique obéit à la syntaxe présentée ci-dessus.

La version actuelle du LSEG-EDL ne permet pas la réaffectation des voies logiques autres que la voie logique 0.

Ces commandes servent surtout à :
- obtenir des exécutions ou des listings sur imprimante ;
- échanger des informations avec un autre micro-ordinateur ;
- mettre ou récupérer dans un fichier les instructions d'un programme.

Exemple :

après avoir tapé un programme
SOrtie NOMPRO
LIster lignes *
STandard (ou SOrtie .10)

Place le source du programme dans un fichier NOMPRO.LST.

VI-9 COMMANDE PErsévérer

Cette commande n'a pas de paramètre.

Cette commande rétablit l'affectation de la voie logique après, par exemple, une interruption sur erreur.

Si on était en mode standard, elle sera sans effet.

Si on était en entrée ou sortie dans un fichier, l'échange sera poursuivi.

Les commandes COntinuer et POursuivre simulent un PErsévérer au début de leur exécution.

VI-10 COMMANDE STandard

Cette commande rétablit l'affectation standard des voies logiques.

Remarques :

a) Imprimante et commandes :

Si l'on veut obtenir sur l'imprimante les messages générés par les commandes, il faudra attribuer à la voie logique 0, la voie physique .20.

Exemple :

SOrtie .20
LIster lignes *.
Le listage du programme se fait sur l'imprimante.
STandard (annule l'effet du SOrtie .20).

Bien noter qu'après la commande SOrtie .20 tous les AFFICHER sans format se feront sur l'imprimante.

b) Fusionner deux programmes :

Supposons que nous avons deux programmes et que nous désirons les regrouper en un seul. Cela n'est pas possible avec la seule commande APpeler car l'appel d'un programme supprime celui qui se trouvait en mémoire avant l'appel. La solution consiste à entrer le second programme à partir d'un fichier décodé. Nous supposons que les programmes n'ont pas des numéros de ligne qui se chevauchent sinon il faudrait tout d'abord les renuméroter.

Soit à créer un programme PROG à partir des programmes P1 et P2 rangés sur disque en fichiers programmes.

APpeler P1
SOrtie AUX
LIster lignes *
STandard
APpeler P2
ENtrée AUX
STandard (quand l'exécution de la commande ENtrée est terminée
RAnger PROG

Il ne restera qu'à supprimer le fichier AUX.LST.

CHAPITRE VII
L.S.E. GRAPHIQUE

- NOTIONS FONDAMENTALES
- REPRÉSENTATION A L'ÉCRAN
- FONCTIONS GRAPHIQUES
- INSTRUCTION CIBLE
- MOTIFS

L.S.E. GRAPHIQUE

VIII-1 NOTIONS FONDAMENTALES

Contrairement à de nombreux langages de programmation qui se contentent de manipuler des points ou des ensembles de points de l'écran (pixels), L.S.E. manipule des objets graphiques, peut leur appliquer des opérateurs, les ranger en fichiers et bien évidemment les afficher à l'écran. Ces objets sont définis à partir d'objets graphiques élémentaires : les vecteurs.

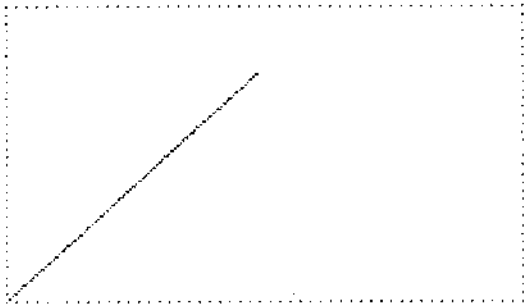
VII-1.1 Vecteurs

Un vecteur est un élément du plan vectoriel et est défini par ses coordonnées et son type. Le L.S.E.G.-E.D.L. connaît, sur T07 et M05, trois types de vecteurs :
- Le vecteur allumé (VEC('A', X, Y)) qui produira lors de l'affichage un trait à l'écran.
- Le vecteur éteint (VEC('E', X, Y)) qui ne produira pas de trace à l'écran.
- Le vecteur gomme (VEC('G', X, Y)) dont l'affichage produit un effacement (c'est un vecteur allumé de la couleur du fond).

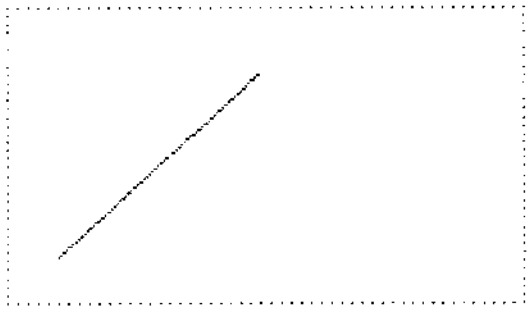
Exemples :

```
?VEC('E',500,500)
```


```
?VEC('A',500,500)
```



```
?VEC('G',100,100)
```



VII-1.2 Formes

A l'aide des vecteurs, nous pouvons construire des formes. Ces formes sont désignées par des identificateurs et devront être déclarées avant emploi.

Exemple :

```
30 FORME F,MAIN,CERCL
40 TABLEAU FORME TG[4],VUE[16,4]
```

La forme la plus simple est composée d'un seul vecteur.

Exemple :

```
70 F←VEC('A',200,500)
```

Tout comme les autres types de variable L.S.E. une forme peut être garée en fichier, chargée depuis un fichier et passée en paramètre à une procédure.

VII-1.3 Opérateurs

A partir de formes élémentaires composées d'un seul vecteur, nous pouvons en fabriquer de plus élaborées en utilisant les opérateurs graphiques. Ces opérateurs sont au nombre de deux :
– la juxtaposition :
Cet opérateur symbolisé par « : » permet de mettre deux formes « bout à bout ».

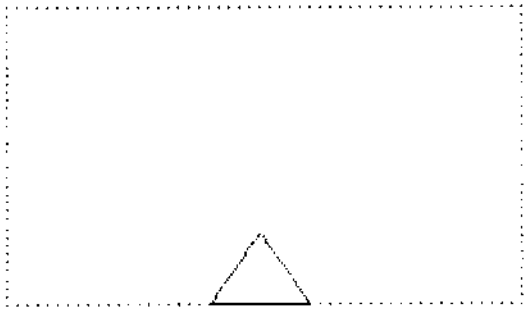
Exemple :

```
FORME F:F←VEC('A',200,100):VEC('A',0,200)
?F
```

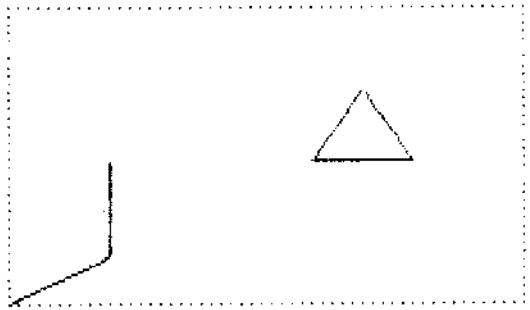
Puis :

FORME G ; G ← VEC ('E', 400, 0) : VEC ('A', 200, 0) : VEC ('A', – 100, 150) : VEC ('A', – 100, – 150)

```
?G
```



```
NETTOYER 0:?F:G
```



Pour bien comprendre l'action de cet opérateur de juxtaposition lorsqu'il s'applique à des formes complexes (non réduites à un seul vecteur), il est nécessaire de connaître la notion de vecteur équivalent. Une forme étant constituée de vecteurs, objets orientés, est elle-même orientée et ordonnée : elle a une origine et une extrémité. Cet ordre correspond, dans des cas simples, à l'ordre dans lequel on voit s'effectuer le tracé à l'écran.
Le vecteur équivalent d'une forme est le vecteur qui possède même origine (note pour les habitués des espaces vectoriels : il ne peut pas en être autrement puisque nous sommes dans un plan vectoriel !) et même extrémité que la forme.
Par exemple, pour les deux formes que l'on vient d'utiliser, le vecteur équivalent de F a pour coordonnées 200 et 300, le vecteur équivalent de G a pour coordonnées 400 et 0. (Lorsque, comme c'est le cas ici, les formes sont fabriquées par des juxtapositions de vecteurs, le vecteur équivalent est la somme des vecteurs composant la forme).
On obtient le vecteur équivalent d'une forme en utilisant la fonction VEQ, ses coordonnées pouvant être connues grâce aux fonctions VEX et VEY.

Exemple :

```
VEQ(F) est le vecteur VEC('E',200,300)
?VEX(F)
200
?VEY(F)
300
```

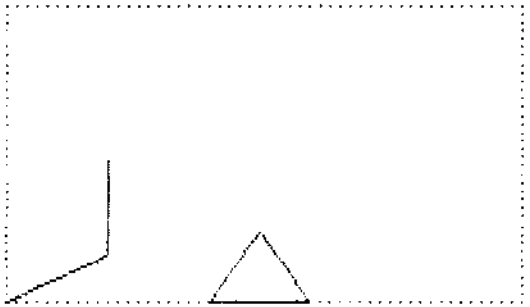
Lorsque nous juxtaposons des formes, le vecteur équivalent du résultat de la juxtaposition est la somme des vecteurs équivalents des formes. On peut remarquer que VEQ(F:G) = VEQ(G:F) alors, qu'en général, les formes F:G et G:F sont distinctes (et produisent des « dessins » différents).

– La superposition :
Cet opérateur symbolisé par « % » permet à partir de 2 formes d'en fabriquer une troisième en les « superposant » c'est-à-dire en faisant coïncider leurs origines.

Exemples :


```
?VEC('R',200,100)%VEC('R',200,200)
```

```
NETTOYER 0;?F%G
```



Remarques :

Le vecteur équivalent d'une superposition est nul (VEQ(F%G) = VEC ('E',0,0)).
Si les formes F%G et G%F produisent le même dessin, leurs codifications internes sont en général différentes.
L'opérateur : (juxtaposition) a priorité sur l'opérateur % (superposition). Par exemple F%G:H est égal à F%(G:H). On peut utiliser des parenthèses pour modifier cet ordre de priorité.

VII-2 REPRÉSENTATION A L'ÉCRAN

VII-2.1 Page graphique

Le T07 et le M05 possèdent une seule page graphique utilisée en permanence. En L.S.E. cette page porte le numéro 0.
On efface cette page graphique par l'instruction NETTOYER 0. A cette occasion, on peut changer la couleur du fond en faisant suivre cette instruction d'un code de couleur. Ce code est une chaîne composée des caractères R, V et B (pour Rouge, Vert et Bleu) ainsi que S (pour obtenir la couleur « claire » - sur T07-70 et M05).

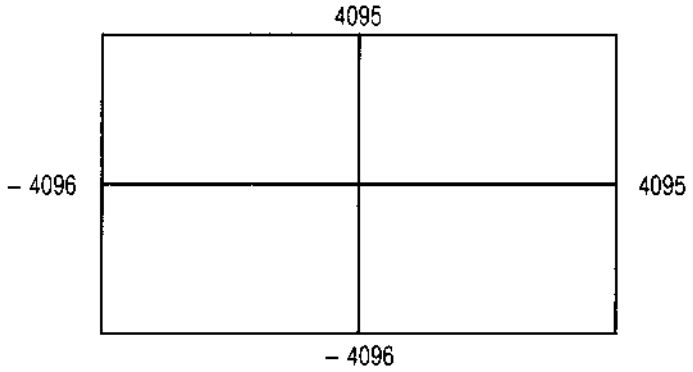
Exemples :

NETTOYER 0 efface l'écran sans changer la couleur du fond
NETTOYER 0, 'RV' efface l'écran et colore le fond en jaune
NETTOYER 0, 'VS' efface l'écran et colore le fond en vert clair.

Remarque : les instructions ALLUMER 0 et ETEINDRE 0 sont sans effet sur ces matériels puisque l'écran est en permanence utilisé en mode graphique.

VII-2.2 Cadre

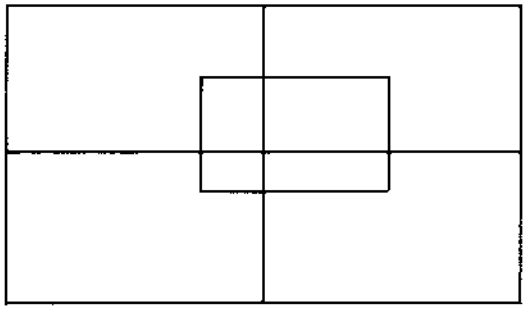
Les formes sont définies, nous l'avons vu, à partir des éléments d'un plan vectoriel en théorie illimité. En réalité, pour des raisons d'encombrement mémoire et de rapidité, le L.S.E.G.-E.D.L. limite les coordonnées à des valeurs absolues inférieures à 4096. De même la plus petite coordonnée non nulle vaut, en valeur absolue, 1/8 c'est-à-dire 0.125. Nous construisons donc nos formes dans un plan qui a l'aspect suivant :



L'écran est une fenêtre que l'on peut définir et déplacer dans ce plan. Cela se fait par l'instruction CADRER en précisant les coordonnées des angles inférieur gauche et supérieur droit de l'écran.

Exemple :

CADRER – 1000, – 500, 2000, 1000 nous permettra d'afficher à l'écran la partie du plan représentée ci-dessous :



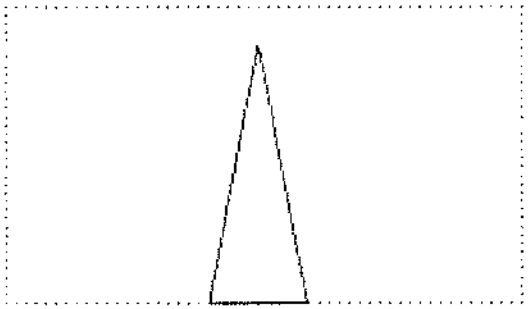
Notons que dans ce cas les proportions horizontales et verticales ne seront plus respectées.
En l'absence d'instruction CADRER, le cadre standard est tel que l'angle inférieur gauche de l'écran ait comme coordonnées 0 et 0, l'angle supérieur droit ayant comme coordonnées 1023 et ce qu'il faut pour que l'image ne soit pas déformée (repère orthonormé) soit 631 sur un T07 ou un M05.

Exemple :

(Nous sommes au départ en cadre standard).

```
FORME T ; T ← VEC( 'A',200,0):VEC('A',-100,173 ):VEC('A',-100,-173)
?T
```

```
NETTOYER 0;CADRER -400,0,600,200;?T
```

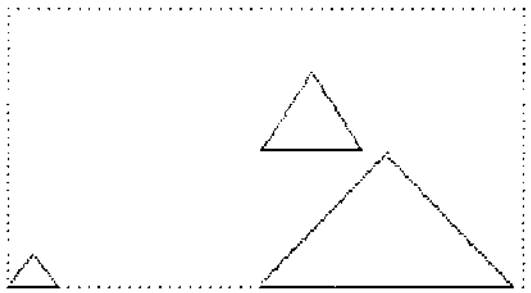


VII-2.3 Marge

On peut souhaiter ne pas utiliser l'écran dans sa totalité pour l'affichage de graphiques. Il est possible de définir une « fenêtre » sur l'écran en utilisant l'instruction MARGER. Cette instruction doit être suivie d'un numéro de page graphique (0 sur T07 et M05) et des coordonnées du coin inférieur gauche et du coin supérieur droit de la « fenêtre », ces coordonnées étant définies par rapport aux dimensions du cadre standard.

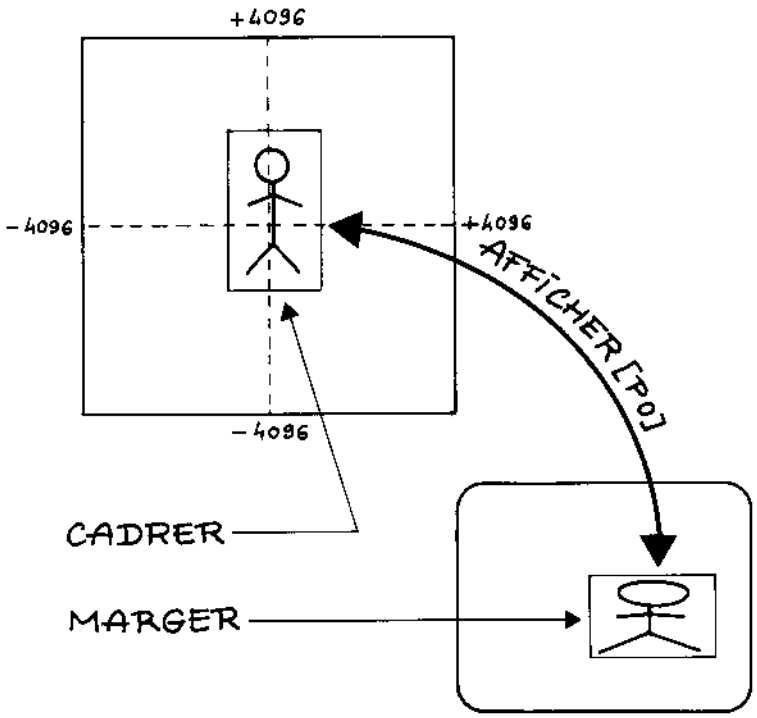
Exemple :

```
I
10 FORME TRGLE;NETTOYER 0
20 TRGLE←VEC('A',200,0):VEC('A',-100,173
):VEC('A',-100,-173)
30 MARGER 0,0,0,500,310;AFFICHER [P0]TRG
LE
40 CADRER 0,0,500,300
50 MARGER 0,500,310,1000,620;AFFICHER [P
0]TRGLE
60 CADRER 0,0,200,175
70 MARGER 0,500,0,1000,310;AFFICHER [P0]
TRGLE
80 ALLER EN 80

EXECUTER A PARTIR DE 1
```



Appuyer sur la touche STOP pour interrompre ce programme.

L'instruction AFFICHER [P0] permet d'établir la coïncidence entre le cadre logique (défini par CADRER) et le cadre physique (page n°0, définie par MARGER).



VII-2.4 Gestion de la couleur

Le L.S.E.G.-E.D.L. gère les huit couleurs plus les huit demi-teintes du T07-70 et du M05. La gestion de ces couleurs se fait par synthèse additive des primaires rouge, vert et bleu. On utilise pour cela une chaîne composée de caractères choisis parmi R, V, B et S :
'' représente la couleur noir
'R' représente la couleur rouge
'V' représente la couleur verte
'B' représente la couleur bleu
'RV' représente la couleur jaune
'RB' représente la couleur magenta
'VB' représente la couleur cyan
'RVB' représente la couleur blanc.

On obtient la demi-teinte correspondante en ajoutant le caractère 'S' :
Par exemple :
'S' représente la couleur gris
'RS' représente la couleur rouge clair
... etc.
A la chaîne définissant la couleur on doit ajouter un code définissant le type de tracé. Pour l'instant, seul existe le type de tracé de code 1 (trait continu). Une désignation complète (couleur et tracé) sera donc de la forme 'RVS1' ou 'B1' par exemple.
Pour définir la couleur du tracé courant, on utilise l'instruction TRACE suivie d'une désignation de couleur.

Exemple :

```
TRACE 'RV1';?VEC('A',500,500)
affiche le vecteur en jaune;
TRACE 'R1';?VEC('A',500,500)
réaffiche le vecteur en rouge.
```

La fonction TRA assigne une couleur à une forme.

Exemple :

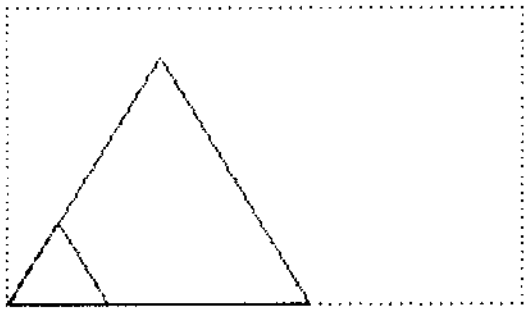
```
FORME F;F ← TRA(VEC('A',300,300),'R1'):VEC('A',200,200)
TRACE 'B1';?F
affiche le premier vecteur de F en rouge, le second en bleu;
TRACE 'RV1';?F
affiche le premier vecteur de F en rouge, le second en jaune.
```

VII·3 FONCTIONS GRAPHIQUES

Il est possible d'appliquer des fonctions à des objets graphiques. Toutes les transformations matricielles planes sont possibles (voir liste de ces fonctions dans la partie Manuel de Référence).

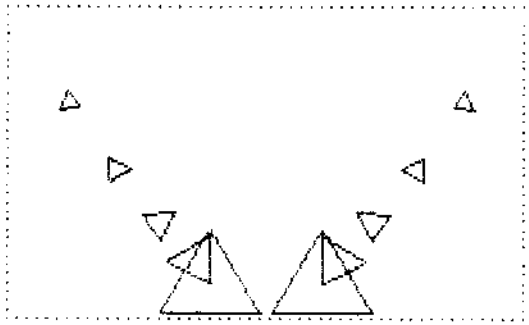
Exemples :

FORME T ; T ← VEC('A',200,0):VEC('A',–100,173):VEC('A',–100,–173)
?T,HOM(T,3)



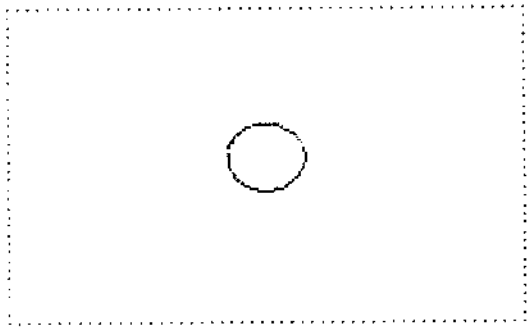

```
1 * TRIANGLES
10 FORME F,T;F←VEC('E',-90,-20)
20 T←VEC('A',200,0):VEC('A',-100,173):VE
C('A',-100,-173)
30 FAIRE 40 POUR I←1 JUSQUA 5
40 F←F:TRN(ROT(HOM(T,1/I),(I-1)/2),100,3
0*I)
50 F←F%SMY(F)
60 CADRER -512,0,512,700;AFFICHER [UIF
70 ALLER EN 70;* appuyer sur STOP pour a
rr@ter
```

EXECUTER A PARTIR DE 1



```
1 * CERCLE
10 FORME CERCL;CERCL←VEC('E',0,0)
20 PI←3.1415927
30 FAIRE 40 POUR I←0 JUSQUA 24
40 CERCL←CERCL:ROT(VEC('A',20,0),I*2*PI/
24)
50 AFFICHER [U]VEC('E',500,300):CERCL
60 ALLER EN 60;* appuyer sur STOP pour a
rrêter
```

EXECUTER A PARTIR DE 1



VII-4 L'INSTRUCTION CIBLE

Cette instruction permet de saisir les coordonnées d'un point visé par le crayon optique. Ces coordonnées sont exprimées par rapport au cadre standard (0,0,1023,631) même si une instruction CADRER a changé le cadre courant.

Exemple :

```
1 * Utilisation de CIBLE
10 AFFICHER ['Pointer le crayon optique
à l''endroit où vous voulez commencer le
 dessin',/]
15 CIBLE A,B
20 AFFICHER ['Pointer le crayon optique
à l''endroit où vous voulez aller']
25 FAIRE 50 TANT QUE .VRAI.
30 CIBLE X,Y
40 AFFICHER [U]VEC('E',A,B):VEC('A',X-A,
Y-B)
45 A←X;B←Y
50
```

Exécutez ce programme (pour l'interrompre, appuyez sur la touche STOP).
Modifiez-le de la manière suivante :
10 CADRER 0,0,2048,1262
et relancez son exécution...

VII-5 LES MOTIFS

Lorsque nous avons dans un programme à répéter plusieurs fois la même séquence, nous créons une procédure. Cela permet de n'écrire cette séquence qu'une seule fois et donc d'obtenir un programme moins volumineux.
En graphique, si nous voulons, par exemple, dessiner un train, c'est-à-dire construire une forme TRAIN, nous écrirons une ligne de programme qui aura à peu près l'allure suivante :
230 TRAIN ← LOCO:WAGON:WAGON:WAGON:WAGON:WAGON
WAGON étant une forme assez complexe faisant elle-même référence plusieurs fois à des formes ROUE, FENET, ...

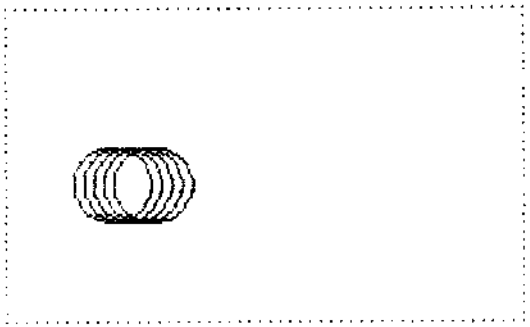
Cette façon de procéder est très coûteuse en place mémoire. En effet, si la forme WAGON occupe n octets, la forme TRAIN occupera plus de 5 fois n octets puisqu'elle contient 5 WAGONs plus la LOCOmotive plus les opérateurs de juxtaposition.
D'autre part, il est inutile que cette forme TRAIN contienne cinq fois la description du dessin d'un wagon puisque c'est le même que nous redessinons à chaque fois.
Nous pouvons régler ce problème en utilisant la notion de MOTIF. Nous allons, à partir de la forme WAGON créer un motif de nom WAG (nous pourrions l'appeler lui aussi WAGON car, vous le verrez, il n'y a pas d'ambiguïté possible). Cela se fait en utilisant l'instruction MOTIF :
190 MOTIF 'WAG' = WAGON
La forme TRAIN peut maintenant s'écrire :
230 TRAIN ← LOCO:MTF('WAG'):MTF('WAG'):MTF('WAG'):MTF('WAG'):MTF('WAG')

Quelle est la différence ? C'est exactement la même que lorsque l'on remplace la répétition de lignes de programme par des appels à une même procédure : le programme est plus court. Ici, c'est la forme TRAIN qui est plus petite (en occupation mémoire !). En effet elle ne contient plus en dehors de la forme LOCO et des opérateurs de juxtaposition que des descripteurs du motif WAG et non plus les dessins des wagons eux-mêmes. On dit que la forme TRAIN fait référence au motif WAG par son nom et non pas par sa valeur.
On aura donc intérêt à définir également des motifs ROUE, FENET, ... etc.

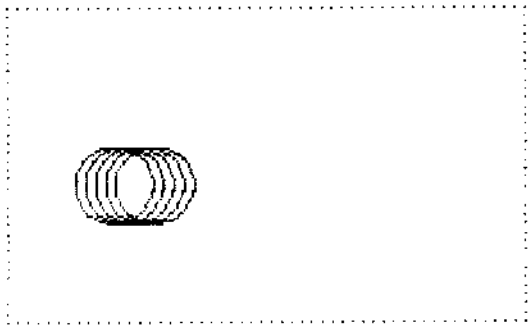
Mettons en évidence le gain de place grâce à l'exemple suivant :

```
1
10 FORME CERCL;CERCL←VEC('E',0,0)
20 PI←3.1415927
30 FAIRE 40 POUR I←0 JUSQUA 24
40 CERCL←CERCL:ROT(VEC('A',20,0),I*2*PI/
24)
50 FORME F,G;F←VEC('E',200,200);G←F
60 MOTIF 'C'=CERCL
100 A1←TZL()
110 F←F:CERCL:CERCL:CERCL:CERCL:CERCL
120 A2←TZL()
130 AFFICHER 'Taille de F: ',A1-A2
200 B1←TZL()
210 G←G:MTF('C'):MTF('C'):MTF('C'):MTF('
C'):MTF('C')
220 B2←TZL()
230 AFFICHER 'Taille de G: ',B1-B2
240 TERMINER

EXECUTER A PARTIR DE 1

Taille de F: 655
Taille de G: 75
TERMINE EN LIGNE 240
```

```
?F
```



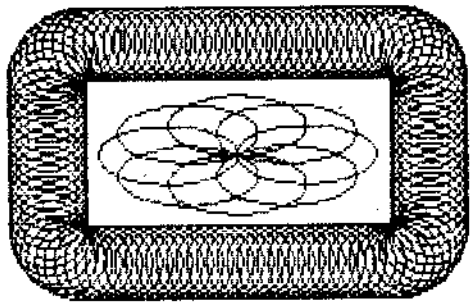
```
NETTOYER 0;?G
```



Notons que l'affichage de F est plus rapide que celui de G.
Les noms de motifs suivent la règle des identificateurs L.S.E. Cependant ils peuvent être écrits avec des minuscules mais L.S.E. ne fera pas la différence (ROUE, Roue et roue sont un même nom de motif).
Une fois un motif défini, il est figé et ne peut être ni modifié, ni redéfini, ni libéré. On ne peut lui faire référence que par l'emploi de la fonction MTF.
Un motif est un objet global. Il ne peut pas être passé en paramètre à une procédure mais il est connu dans tous les modules, y compris dans les procédures externes.

Pour terminer, un « joli(?) dessin » :

```
1 * un "JOLI" dessin
5 AFFICHER [.27.' P'.27.]EQC(96);* Ecran
 noir, tour noir
6 AFFICHER [.20.];* effacer curseur
10 FORME CERCL;CERCL←VEC('E',0,0)
15 FORME LARG,HAUT,ANGLE;LARG←VEC('E',0,
0);HAUT←LARG;ANGLE←LARG
20 PI←3.1415927
30 FAIRE 40 POUR I←0 JUSQUA 24
40 CERCL←CERCL:ROT(VEC('A',20,0),I*2*PI/
24)
60 MOTIF 'C'=CERCL
70 FAIRE 80 POUR I←1 JUSQUA 15
80 HAUT←HAUT:MTF('C')
90 LARG←HAUT
100 LARG←HAUT:HAUT
120 FAIRE 130 POUR I←0 JUSQUA 7
130 ANGLE←ANGLE%ROT(MTF('C'),-I*PI/2/7)
190 TRACE 'RV1'
200 AFFICHER [U]VEC('E',200,460):LARG:AN
GLE:ROT(HAUT,-PI/2):ROT(ANGLE,-PI/2):ROT
(LARG,PI):ROT(ANGLE,PI):ROT(HAUT,PI/2):R
OT(ANGLE,PI/2)
210 FORME FLEUR;FLEUR←VEC('E',0,0)
220 FAIRE 230 POUR I←0 JUSQUA 7
230 FLEUR←FLEUR%TRA(ROT(TRN(MTF('C'),-10
,0),I*2*PI/8),SI ENT(I/2)=I/2 ALORS 'R1'
 SINON 'RB1')
240 AFFICHER [U]VEC('E',497,305):AFX(AFY
(FLEUR,1.8),0.8)
900 ALLER EN 900;* appuyer sur STOP pour
 arrêter
```



MANUEL DE RÉFÉRENCE

ea, ea1, ... désigneront des expressions arithmétiques de valeurs respectives (ea), (ea1) ...

eb, eb1, ... désigneront des expressions booléennes de valeurs respectives (eb), (eb1) ...

ec, ec1, ... désigneront des expressions chaines de valeurs respectives (ec), (ec1) ...

eg, eg1, ... désigneront des expressions graphiques de valeurs respectives (eg), (eg1) ...

COMMANDES

- **ABréger**
- **ALler en**
- **APpeler**
- **AU revoir**
- **BOnjour**
- **COntinuer**
- **DEscendre**
- **DIsque**
- **EFfacer lignes**
- **ELiminer commentaires**
- **ENtrée**
- **ESpacer lignes**
- **EXécuter**
- **FIn**
- **IDentification**
- **IN extenso**
- **LAncer**
- **LIster lignes**
- **NOrmal**
- **NUméro lignes**
- **PAs à pas**
- **PErsévérer**
- **POursuivre**
- **PRendre état console**
- **RAnger**
- **REmplacer**
- **SOrtie**
- **STandard**
- **SUpprimer**

Pour utiliser une commande L.S.E. on doit taper les deux premiers caractères puis les valider (touche ENTREE).

Dans les pages qui suivent, on distinguera ces deux premiers caractères (en majuscules), des compléments de commande affichés par le système (en minuscules).

ABréger

Cette commande, sans paramètre, supprime l'affichage du complément de commande jusqu'à la première utilisation de IN extenso.

Exemple :

BOnjour
APpeler TOTO (le complément de commande est complet)
ABréger
AP TOTO (le complément se réduit à un espace)
IN (IN extenso)
APpeler TOTO (retour à la normale)

ALler en

Cette ***commande**** admet un ou deux paramètres, le premier représentant la ligne de branchement, le second la ligne d'arrêt éventuelle. L'intérêt de cette commande est de permettre de continuer l'exécution d'un programme quand la commande COntinuer ne peut être utilisée : (après une erreur d'exécution par exemple).
– ALler en 320 reprend l'exécution à partir du début de la ligne 320 et jusqu'à la fin.
– ALler en 250 A 125 démarre au début de la ligne 250 et s'arrête si le programme passe en ligne 125 en affichant sur l'écran LIGNE 125, cette ligne n'étant pas encore exécutée.

Si le programme est arrêté dans une boucle et si le ALler en le branche à l'extérieur de la boucle, il y a simulation d'une sortie extraordinaire de la boucle; si le ALler en le branche à l'intérieur de la boucle l'exécution se poursuit à l'intérieur de la boucle.

Si le programme est arrêté à l'intérieur d'une procédure, on ne pourra en sortir par la commande ALler en qu'en passant sur un RETOUR, RETOUR EN ou RESULTAT. Il faut donc se brancher à un endroit convenable pour obtenir une exécution normale du programme.

*(*) Ne pas confondre avec l'**instruction** ALLER EN*

APpeler

On récupère en mémoire vive un fichier programme en utilisant la commande APpeler :

APpeler nomf ou nomf est le nom d'un fichier programme.

Le système va chercher sur le disque un fichier programme de nom nomf. S'il le trouve, il charge son contenu en mémoire centrale à la place de celui qui s'y trouvait éventuellement, sinon un message du type « fichier inexistant » apparait.

AU revoir

Cette commande est sans paramètre. Elle remet le L.S.E. dans le même état qu'après chargement. La seule commande autorisée après AU revoir est BOnjour. Il est cependant possible de modifer la date du jour, il suffit pour cela d'appuyer sur la touche STOP : le L.S.E. vous demande alors la date comme après la mise sous tension.

BOnjour

Le rôle de cette commande est d'initialiser la zone utilisateur. Elle est obligatoire après le chargement du L.S.E.

COntinuer

Cette commande n'a pas de paramètre. Elle permet de poursuivre l'exécution d'un programme interrompu, soit par une instruction PAUSE, soit par la touche STOP.

Par contre il n'est pas possible d'utiliser la commande COntinuer après une erreur d'exécution.

DEscendre

Cette commande n'admet pas de paramètre. Elle permet en cas d'arrêt dans une procédure externe L.S.E. de revenir au niveau principal, ce qui donne accès en CALBUR aux variables du niveau principal.

Remarque :

Lors d'un arrêt, voulu ou pas, dans une procédure externe :
- la commande LIster porte sur le module principal.
- les opérations que l'on peut faire en CALBUR portent sur le module interrompu.

(Si l'on veut pouvoir faire quelque chose d'utile à ce niveau, il faut disposer d'une liste sur papier de la procédure externe.)

DIsque

Cette commande demande comme paramètre un numéro de disque. Ce disque sera utilisé comme disque de travail par la suite. Par défaut, après chargement du système, le disque de travail est le disque 1 si vous avez des disquettes, sinon le disque 0 (qui désigne le lecteur de cassettes).

EFfacer lignes

La syntaxe de cette commande admet plusieurs variantes. Son rôle est de permettre la suppression de une ou plusieurs lignes de programme.

Il est possible de supprimer la totalité des lignes du programme en mémoire.
La syntaxe à utiliser dans ce cas est :
EFfacer lignes *
Après exécution de la commande la zone utilisateur se trouve dans le même état qu'après l'exécution de la commande BOnjour.

Si l'on veut supprimer du programme la ligne de numéro 20 on utilise la forme :
EFfacer lignes 20

Pour supprimer toutes les lignes de la ligne de numéro 5 à la ligne de numéro 1215 on tapera :
EFfacer lignes 5 A 1215

La commande suivante est reconnue :
EFfacer lignes 12, 23, 54 A 120
Elle supprime du programme successivement la ligne 12, puis la ligne 23 et ensuite toutes les lignes qui restent et dont le numéro est compris entre 54 et 120, bornes comprises.

Remarques :
Si l'on efface des lignes pendant l'exécution d'un programme la commande COntinuer peut encore être utilisée (à condition de ne pas avoir à reprendre dans une ligne effacée).

ELiminer commentaires

Cette commande sans paramètre permet de récupérer la place occupée par les commentaires. Des programmes volumineux ou créant des données de grande taille peuvent ainsi s'exécuter sans provoquer l'erreur E 03. Attention, il ne faut pas modifier le programme sur disque sous peine de perdre les commentaires.

ENtrée

Les affectations ***standard**** peuvent être modifiées par la commande ENtrée. Elle exige un complément de commande :

ENtrée [voie logique =] voie physique

Par défaut, c'est la voie logique 0 qui est concernée. La voie physique obéit à la syntaxe rappelée ci-dessous.
La version actuelle du LSEG-EDL ne permet pas la réaffectation des voies logiques autres que la voie logique 0.

Cette commande sert essentiellement à :
- échanger des informations avec un autre micro-ordinateur
- récupérer dans un fichier les instructions d'un programme.

(*) En L.S.E. on distingue :

– Les voies logiques numérotées à partir de 0 et qui peuvent être utilisées par exemple dans des instructions LIRE ou AFFICHER.

Les affectations standard sont en entrée :
0 au clavier
1 au clavier sans écho
2 à la voie V24

– Les voies physiques qui sont de deux sortes :
– celles qui correspondent à des périphériques et qui sont symbolisées par :
.10 pour clavier ou écran
.20 pour imprimante ou clavier sans écho
.30 pour voie V24
.40 pour inhiber l'entrée ou la sortie
– celles qui correspondent à des noms de fichiers qui sont symbolisées par le nom du fichier.

ESpacer lignes

Le rôle de cette commande est de permettre une remunérotation partielle ou totale du programme en mémoire. Cette renumérotation doit atteindre tous les éléments qui font référence à un numéro de ligne. Ainsi sont modifiées les instructions suivantes, n étant un numéro de ligne :
– faire n...
– aller en n
– retour en n
– retour en n, ✳

Attention :
Retour en n,p n'est pas renuméroté.

Pour pouvoir atteindre les nombres qui représentent des numéros de ligne et qui n'appartiennent pas à l'une des instructions précédentes il est indispensable de les faire précéder de @ (symbole aroba).

Les différentes possibilités de la commande sont les suivantes (n, n1, n2,... représentant des entiers) :

ESpacer lignes n force la première ligne du programme à avoir un numéro supérieur ou égal à n.
ESpacer lignes n1, n2 espace de n2 toutes les lignes de numéro supérieur ou égal à n1.
ESpacer lignes n1, n2, n3 espace de n3 toutes les lignes de numéro compris entre n1 et n2.
Espacer lignes n1, n2, n3, n4 espace de n3 toutes les lignes de numéro compris entre n1 et n2 et de n4 les autres lignes.

Remarques :
– Si le programme à renuméroter comporte des instructions telles que : ALLER en ea (ea étant une expression arithmétique et non un nombre) la commande signalera la ligne en question et ne fera pas de renumérotation à moins que l'on n'ait fait suivre la commande du paramètre final, F pour forcer la modification. Dans ce dernier cas, la commande donne le numéro de ligne avant et après la renumération et ne modifie pas l'expression.
– Si le programme comporte des instructions telles que : FAIRE n..., alors que la ligne n n'existe pas la commande le signale en faisant précéder le numéro de ligne du caractère ≠. Dans ce cas on ne pourra obtenir la renumérotation qu'après avoir créé une ligne vide de numéro n.

EXécuter à partir de

Cette commande admet les formes suivantes :

• EXécuter à partir de n1

• EXécuter à partir de n1 A n2
Dans les deux cas, le programme est lancé à partir de la première ligne dont le numéro est supérieur ou égal à n1.
Dans le second cas, l'exécution s'arrête si l'on passe à la ligne n2 (arrêt qui se fait avant l'exécution de cette ligne), et le message : LIGNE n2 apparaît à l'écran.

FIn

Cette commande, sans paramètre, provoque le passage sous le moniteur. Le curseur se trouve alors en début de ligne juste après un caractère >. Le retour à L.S.E. reste possible par RESET. A condition de ne pas avoir détruit la zone mémoire utilisée par le L.S.E., vous ne perdez ni votre programme ni les données qui ont été créées.

IDentification

Cette commande est reconnue par les micro-ordinateurs en réseau.
Elle demande un complément de commande qui est analysé par le réseau, et qui sera donc fourni avec la documentation du réseau.

IN extenso

Cette commande, sans paramètre, rétablit l'affichage du complément de commande.

LAncer

La commande LAncer enchaîne, de manière transparente à l'utilisateur les commandes APpeler et EXécuter (à partir de la ligne du plus petit numéro).
LAncer nomf
où nomf est le nom du orogramme.

LIster lignes

Cette commande rétablit sous forme « lisible » le texte source des lignes du programme et permet de le visualiser sur l'écran, sur une imprimante ou sur d'autres organes de sortie qui auront été choisis à l'aide de la commande SOrtie.

LIster lignes * permet d'obtenir la totalité du programme. Il est possible d'arrêter le listage à la fin de la ligne en cours (que la sortie se fasse sur l'écran ou ailleurs); il suffit pour cela de taper sur la touche « – ». Pour obtenir la continuation du listage taper sur une touche autre que « – ». (Pour terminer prématurément, taper sur STOP).

Il est également possible de moduler la vitesse d'affichage des lignes de la manière suivante :

En tapant sur la touche 1, il s'écoule 1/10 sec. entre l'affichage de deux lignes.
En tapant sur la touche 2, l'intervalle de temps entre l'affichage de deux lignes est de 2/10 sec., etc. jusqu'à 9.

– Le listage d'une ligne isolée (par exemple 10), s'obtient par LIster lignes 10.
– Pour obtenur sur l'écran la liste du programme entre les lignes 120 et 200, on tapera : LIster lignes 120 A 200.

NOrmal

Cette commande annule l'effet de la commande PAs à pas, et permet de remettre le L.S.E. dans son mode habituel de fonctionnement.

NUméro lignes

Cette commande admet la même syntaxe que la commande LIster lignes (voir cette commande). Elle permet d'obtenir la liste des numéros de ligne du programme en mémoire. Elle peut par exemple être utilisée après un transfert de programme sur la voie V24 pour vérifier que toutes les lignes du programme source ont bien été reçues.

PAs à pas

Cette commande n'admet pas de paramètre.
Quand la commande PAs à pas est tapée, le L.S.E. est mis dans un mode de fonctionnement « ligne par ligne ».
On progressera dans l'exécution du programme en tapant chaque fois ENTREE.
On pourra ainsi suivre le déroulement du programme en ayant à chaque arrêt la possibilité de consulter des variables et de les modifier.

Remarque :
Les numéros de lignes affichés indiquent la ligne qui va être exécutée et non celle qui vient de l'être.

PErsévérer

Cette commande n'admet pas de paramètre.
Elle restitue le dernier état d'une voie logique malgré, par exemple, une interruption du programmme.
Si on était en mode standard, elle sera sans effet.
Si on était en entrée ou sortie dans un fichier, l'échange sera poursuivi à partir du point d'arrêt.

POursuivre jusqu'en

Cette commande nécessite un paramètre : le numéro de la ligne d'arrêt. Elle se comporte comme la commande COntinuer avec en plus la possibilité de s'arrêter avant l'exécution de la ligne indiquée.
POursuivre jusqu'en n continue l'exécution du programme à partir du point d'arrêt précédent et s'arrête si le programme passe en ligne n (avant d'exécuter cette ligne et en affichant LIGNE n).

PRendre état console

Cette commande demande un nombre n en complément. Elle ne peut fonctionner que dans un système de TO7/MO5 montés en réseau. Elle est refusée si le TO7/MO5 n'est pas relié à un réseau en état de marche. Son exécution provoque la recopie du contenu de la mémoire du TO7/MO5 de numéro n dans la zone mémoire. Cela concerne le programme de l'utilisateur numéro n, ses données et le contenu de sa mémoire d'écran.

RAnger

On sauvegarde le contenu de la zone programme en utilisant la commande RAnger :
Cette commande exige en paramètre le nom du fichier dans lequel aura lieu la sauvegarde :
RAnger nomf
où nomf est le nom du fichier (identificateur d'au plus 8 caractères alphanumériques).

REmplacer

Cette commande permet de sauvegarder la nouvelle version d'un programme. Bien entendu, cela implique la perte définitive de l'ancienne version.

Elle nécessite un paramètre :

REmplacer nomf où nomf est le nom du fichier programme.

SOrtie

Les affectations ***standard**** peuvent être modifiées par la commande SOrtie. Elle exige un complément de commande :
SOrtie [voie logique =] voie physique
Par défaut, c'est la voie logique 0 qui est concernée. La voie physique obéit à la syntaxe présentée ci-dessous.
La version actuelle du LSEG-EDL ne permet pas la réaffectation des voies logiques autres que la voie logique 0.
Cette commande sert surtout à :
- obtenir des exécutions ou des listes sur imprimante.
- échanger des informations avec un autre micro-ordinateur.
- mettre dans un fichier les instructions d'un programme.

Exemple :

Après avoir tapé un programme
SOrtie nompro
LIster lignes *
STandard (ou SOrtie.10)

L'usage de la commande STandard (ou SOrtie.10) est impératif si on ne veut pas perdre le fichier qui vient d'être créé.

(*) En L.S.E. on distingue :
- Les voies logiques numérotés à partir de 0 et qui peuvent être utlisées par exemple dans des instructions LIRE ou AFFICHER.

Les affectations standard sont en sortie :
0 à l'écran
1 à l'imprimante
2 à la voie V24

- Les voies physiques sont de deux sortes :
- celles qui correspondent à des périphériques et qui sont symbolisées par :
.10 pour clavier ou écran
.20 pour imprimante ou clavier sans écho
.30 pour voie V24
.40 pour inhiber l'entrée ou la sortie
- celles qui correspondent à des noms de fichiers qui sont symbolisées par le nom du fichier.

STandard

Cette commande n'a pas de paramètre. Elle rétablit l'affectation standard des voies logiques en entrée commme en sortie.

SUpprimer

Cette commande permet de supprimer un fichier programme ou un fichier données sur le disque.
- SUpprimer nomf. LSP où nomf est le nom du fichier programme à supprimer (LSP indique qu'il s'agit d'un programme)
La place occupée par le programme est récupérée, et son nom a disparu du catalogue.

- SUpprimer nomf. LSD [,n] où nomf est le nom du fichier à supprimer (LSD indique qu'il s'agit de fichier données)
n est un numéro d'enregistrement.

Exemple :

SUpprimer TOTO. LSD, 2 va supprimer l'enregistrement 2 du fichier données de nom TOTO.

OPÉRATEURS

- OPÉRATEURS NUMÉRIQUES
- OPÉRATEUR CHAÎNE
- OPÉRATEURS BOOLÉENS
- OPÉRATEURS DE COMPARAISON
- OPÉRATEURS GRAPHIQUES

OPÉRATEURS NUMÉRIQUES

Il existe :
– 1 opérateur unaire : – pour le changement de signe
et
– 5 opérateurs binaires : + pour l'addition
– pour la soustraction
* pour la multiplication
/ pour la division
↑ pour l'exponentiation

Le calcul s'effectue de la gauche vers la droite, en tenant compte de l'ordre de priorité décroissante suivant :
l'exponentiation
le changement de signe
la multiplication ou la division
l'addition ou la soustraction

Cet ordre peut toutefois être modifié par l'introduction de parenthèses. Dans ce cas l'expression calculée en premier est celle qui est placée entre les parenthèses les plus internes.

Remarques :
Une expression numérique ne peut pas contenir 2 opérateurs l'un à côté de l'autre.
Il est interdit d'élever un nombre négatif à une puissance non entière.
Il est également interdit d'élever 0 à une puissance négative ou nulle.

OPÉRATEUR CHAÎNE

Le seul opérateur sur chaînes est l'opérateur de concaténation noté ! (point d'exclamation).
Il permet de réunir 2 chaînes en une seule : les 2 chaînes sont mises bout à bout.

Exemple :

```
CHAINE A,B,C; A← 'FRANÇOIS APPREND SA LEÇON'; B← ' D'' ANGLAIS'
C←A ! B; ? C
FRANÇOIS APPREND SA LEÇON D'ANGLAIS
```


OPÉRATEURS BOOLÉENS

Il existe 1 opérateur unaire : NON
et 3 opérateurs binaires : ET
OU (ou inclusif)
DIJ (ou exclusif)
Les tables de vérité ci-dessous donnent le résultat des opérations.

P	Q	P ET Q	P OU Q	P DIJ Q
.VRAI.	.VRAI.	.VRAI.	.VRAI.	.FAUX.
.VRAI.	.FAUX.	.FAUX.	.VRAI.	.VRAI.
.FAUX.	.VRAI.	.FAUX.	.VRAI.	.VRAI.
.FAUX.	.FAUX.	.FAUX.	.FAUX.	.FAUX.

P	NON P
.VRAI.	.FAUX.
.FAUX.	.VRAI.

Dans l'évaluation d'une expression booléenne l'ordre des priorités est le suivant :
NON
ET
OU ou DIJ

On utilisera des parenthèses si l'on veut modifier l'ordre des priorités.

OPÉRATEURS DE COMPARAISON

Ces opérateurs permettent de comparer 2 expressions de même nature (numériques, chaînes ou booléennes)

Ces opérateurs sont au nombre de 6 :

= pour égal
≠ pour différent
< pour inférieur
> pour supérieur
< = pour inférieur ou égal
> = pour supérieur ou égal

Seuls les opérateurs = et ≠ peuvent être utilisés entre expressions booléennes. On ne peut pas comparer deux expressions graphiques.

Exemples :

• comparaison entre nombres :

```
? 2< =2
.VRAI.
? 2<RAC (3)
.FAUX.
```

• comparaison entre chaînes de caractères :

ec1 et ec2 sont des expressions chaînes.

Les expressions (ec1) < (ec2) ou (ec1) > (ec2) sont des expressions booléennes évaluées de la manière suivante :

On compare les 2 chaînes caractère par caractère (sur la longueur de la plus courte) de la gauche vers la droite en utilisant les codes ASCII de ces caractères :

– Si l'on trouve 2 caractères différents, la chaîne la plus « grande » est celle qui contient le caractère de plus grande valeur.

– Si l'on ne trouve pas de caractère différent, on compare les longueurs des 2 chaînes : si les longueurs sont les mêmes les 2 chaînes sont égales, sinon la plus « grande » est la plus longue.

Exemple :

```
? 'FRANCE' >'FRANÇOIS'
.FAUX.
? 'BEAU' <'BEAUCOUP'
.VRAI.
```

• comparaison entre booléens :

```
? .VRAI. = .FAUX.
.FAUX.
```


OPÉRATEURS GRAPHIQUES

Ces opérateurs sont au nombre de 2.

La juxtaposition : (symbole :)

Le résultat de l'évaluation de eg1 : eg2 (eg1 et eg2 étant des expressions graphiques) est la mise bout à bout des valeurs des expressions graphiques eg1 et eg2, l'origine du second opérande étant placée à l'extrémité du premier.
Le repère du résultat est le repère du premier objet. L'origine du résultat est l'origine du premier objet. Le vecteur équivalent est la somme des vecteurs équivalents.

Exemple :

VEC ('A', 50, 0) : VEC ('A', –50, 50) : VEC ('A', 0, –50)
correspond au tracé d'un triangle rectangle isocèle. Le vecteur équivalent est nul.

La superposition : (symbole %)

Évaluation de eg1 % eg2 (eg1 et eg2) étant des expressions graphiques) :
Les repères des 2 opérandes sont confondus. Par définition l'extrémité du résultat est l'origine : le vecteur équivalent est donc nul.

Exemple :

VEC ('A', 100,0) % VEC ('A', 50, 50)
correspond au tracé de 2 segments de même origine faisant entre eux un angle de 45°

La juxtaposition est prioritaire sur la superposition.

INSTRUCTIONS

I. LES ÉLÉMENTS DE BASE

- CONSTANTES
- VARIABLES SIMPLES
- AFFECTATION
- VARIABLES INDICÉES
- EXPRESSIONS
- LIBERER

LES CONSTANTES

En L.S.E. on distingue :

1. Les constantes numériques.

Ce sont des nombres écrits sous leur forme usuelle.
Cependant :
la virgule décimale est remplacée par le point.
les puissances de 10 sont représentées à l'aide de la lettre E.

2. Les constantes chaines.

Une constante chaîne est une suite de caractères. Ces caractères peuvent être des lettres, des chiffres, des caractères particuliers.

3. Les constantes booléennes.

Ces constantes sont au nombre de 2 :
la constante .VRAI.
la constante .FAUX.

LES VARIABLES SIMPLES

Il existe 4 types de variables :
Les variables numériques
Les variables chaînes
Les variables booléennes
Les variables graphiques.

Les variables chaînes, booléennes et graphiques sont soumises à une déclaration préalable au moyen des instructions respectives : CHAINE, BOOLEEN, FORME

Syntaxe :
CHAINE id2, id2,...
BOOLEEN id1, id2,...
FORME id1, id2,...
où id1, id2,... sont des identificateurs.

Les variables non déclarées sont considérées comme des variables simples numériques.

Les noms des variables, appelés IDENTIFICATEURS, sont soumis aux règles suivantes :
Un identificateur est une suite non vide d'au plus 5 caractères alphanumériques (lettres ou chiffres) dont le premier est obligatoirement une lettre. Les minuscules du nom sont automatiquement transformées en majuscules.

Exemples :

A, TEXTE, X1, B12 sont des identificateurs
1A, T 4, PAULINE ne peuvent pas être des identificateurs.
En mode CALBUR on ne peut créer que des identificateurs d'une lettre.

INSTRUCTION D'AFFECTATION : ←

Syntaxe :
id ← exp
id représente un identificateur de variable simple ou indicée,
exp est une expression.

exp est évaluée, puis sa valeur est affectée à la variable. En cas de mélange de types une erreur d'exécution est détectée.

Exemples :

A ← 3.14
CHAINE C; C ← 'BONJOUR'
BOOLEEN B; I ← 5; B ← I>3

LES VARIABLES INDICÉES ET LES TABLEAUX

Il existe 4 sortes de tableau :
Les tableaux numériques, les tableaux chaînes, les tableaux booléens, les tableaux formes.
Ces tableaux doivent être déclarés par l'instruction TABLEAU.

Syntaxe :
TABLEAU id [ea1, ea2, ..], id1 [ea'1, ea'2, ..],...
TABLEAU CHAINE id [ea1,..], id [ea'1,..],...
TABLEAU BOOLEEN id [ea1,..], id1 [ea'1,...],...
TABLEAU FORME id [ea1,...], id1 [ea'1,..],...

id, id1... sont des identificateurs
ea1, ea2, ea'1,... sont des expressions numériques dont les valeurs (après arrondi) sont au moins égales à 1.

Un tableau est constitué de variables de même type (numériques, chaînes, booléennes ou graphiques).
Le nom d'un tableau est un identificateur.
Les éléments d'un tableau sont désignés par :
– le nom du tableau
– une liste d'indices (le nombre d'indices est la dimension du tableau) entre crochets et séparés par des virgules.
Un tableau admet au maximum 255 dimensions.
La seule limitation aux valeurs des indices est la taille mémoire disponible.

LES EXPRESSIONS

Une expression est une combinaison de variables, de constantes, d'expressions entre parenthèses, de résultats de procédures, de résultats de fonctions ou d'expressions conditionnelles, liés par des opérateurs. Cette combinaison doit respecter les règles de grammaire du L.S.E.
Selon que le résultat est numérique, chaîne, booléen ou graphique l'expression est appelée expression numérique, chaîne, booléenne ou graphique.

EXPRESSIONS CONDITIONNELLES

Il est possible en L.S.E. de créer des expressions dont la valeur dépend d'une expression booléenne.

Syntaxe : SI eb ALORS exp1 SINON exp2
eb est une expression booléenne.
exp1 et exp2 sont des expressions de même type.
Si eb est vraie l'expression vaut (exp1), dans le cas contraire elle vaut (exp2).

Remarque :
Les opérateurs de relation sont prioritaires sur les opérateurs booléens, en l'absence de parenthèses.

INSTRUCTION LIBERER

Dès qu'une variable simple ou un tableau n'est plus utilisé dans un programme, il est possible de récupérer la place occupée en mémoire par ces objets. La nécessité de cette récupération peut se faire sentir lorsque l'on écrit des programmes utilisant des données importantes en volume ou bien encore si l'on veut réutiliser un identificateur pour un autre type de variable.
On utilisera l'instruction LIBERER

Syntaxe : LIBERER id1, id2,.....
id1, id2... sont des identificateurs de variables simples ou des noms de tableaux.
La place occupée par les variables id1, id2... est libérée et bien sûr les valeurs de ces variables sont perdues.

Remarque : Il n'est pas possible de libérer des motifs.

II. INSTRUCTIONS FONDAMENTALES

- TERMINER
- PAUSE
- AFFICHER
- LIRE
- ALLER EN
- INSTRUCTIONS CONDITIONNELLES

INSTRUCTION TERMINER

Cette instruction est la dernière instruction exécutée d'un programme.
Elle provoque l'affichage du message TERMINE EN LIGNE n.
Une erreur d'exécution est détectée, quand la dernière instruction exécutée dans une chaîne de programme(s) n'est pas l'instruction TERMINER.

INSTRUCTION PAUSE

Cette instruction provoque la suspension de l'exécution du programme et l'affichage du message PAUSE EN LIGNE n, n étant le numéro de la ligne où se trouve l'instruction PAUSE.
L'utilisateur peut alors reprendre l'exécution en utilisant soit la commande COntinuer, soit la commande POursuivre jusqu'en...

INSTRUCTION AFFICHER sans format

Syntaxe : AFFICHER exp1, exp2 ....
exp1, exp2... sont des expressions ou des noms de tableaux.

Exemple 1 :

```
1 * Périmètre et surface d'un rectangle.
10 LO ← 12 ; LA ← 5
15 P ← 2 * (LO + LA) ; S ← LO * LA
20 AFFICHER P
25 AFFICHER S
30 TERMINER

EXECUTER à partir de 1
34
60
```

Remarque :

L'instruction AFFICHER provoque un passage, à la ligne. Si celui-ci n'est pas désiré, on peut retaper la ligne 20 (et bien sûr effacer la ligne 25) :
20 AFFICHER P , S
A l'exécution on obtiendra alors :
34 60

Exemple 2 :

```
1 * afficher un tableau
11 TABLEAU T1[4], T2[2, 3], T3[2,3,4]
16 * Les tableaux ne sont pas initialisés pour ne pas rendre fastidieuse la
lecture.
21 AFFICHER 'Affichage du tableau T1[4] :', T1
26 AFFICHER 'Affichage du tableau T2[2,3] :', T2
31 AFFICHER 'Affichage du tableau T3[2,3,4] :', T3
36 TERMINER

EXECUTER A PARTIR DE 1

Affichage du tableau T1[4] :
≠ ≠ ≠ ≠

Affichage du tableau T2[2,3] :
≠ ≠ ≠
≠ ≠ ≠
Affichage du tableau T3[2,3,4] :
≠ ≠ ≠ ≠
≠ ≠ ≠ ≠
≠ ≠ ≠ ≠

≠ ≠ ≠ ≠
≠ ≠ ≠ ≠
≠ ≠ ≠ ≠
TERMINE EN LIGNE 36
```

Remarque :

Dans tous les cas, c'est l'indice le plus à droite qui varie le premier.
La première ligne du tableau T3 contient dans l'ordre les éléments : T3[1,1,1], T3[1,1,2], T3[1,1,3], T3[1,1,4]
La dernière ligne contient dans l'ordre :
T3[2,3,1], T3[2,3,2], T3[2,3,3], T3[2,3,4]
Noter que chaque fois qu'un indice a fini de varier il y a passage à la ligne.
Pour les tableaux de chaînes il y a retour à la ligne après chaque élément.
L'instruction AFFICHER peut être utilisée en mode immédiat pour visualiser des objets graphiques (en utilisant le mode de tracé courant et le cadre standard).

Exemple :

AFFICHER VEC ('A',255,0) dessine sur l'écran un segment horizontal de longueur 255, soit à peu près le quart de l'écran.

INSTRUCTION AFFICHER avec format

Toutes les indications concernant le format souhaité doivent être mises entre crochets.
Dans un format peuvent figurer :
- des spécifications de libellés.
- des spécifications de mise en page.
- des spécifications de description.

1. Spécification de libellés

Un libellé est une constante chaîne ; il peut être précédé d'un facteur de répétition fixe ou variable (compris entre 1 et 255).

Exemple 1 : facteur de répétition fixe.

```
10 AFFICHER [Le train sifflera 2 fois : ',2'TUT...']
15 TERMINER

EXECUTER A PARTIR DE 1
Le train sifflera 2 fois : TUT...TUT...
TERMINE EN LIGNE 15
```

Des libellés successifs doivent être séparés par des virgules.

Exemple 2 : facteur de répétition variable.

```
10 * ——& va s'afficher de façon aléatoire
15 FAIRE 20 POUR I * 1 JUSQUA 20
20 AFFICHER [*X,'——&']ALE(0)*20
25 TERMINER
```

Au lieu du facteur de répétition fixe, on a utilisé *, symbole de répétition variable. Sa valeur est celle (après arrondi) de l'expression numérique placée après] ; cette valeur doit être positive ou nulle.

2. Spécification de mise en page

4 codes permettent les spécifications de mise en page :

/ : provoque le passage au début de la ligne suivante.
L : provoque un saut de ligne (sans retour en début de ligne).
X : laisse un espace sur la ligne en cours.
C : provoque un retour en début de ligne (sans passer à la ligne suivante).

Chacune de ces spécifications peut être précédée d'un facteur de répétition, fixe ou variable.

Exemple :

```
10 AFFICHER [16X, 'TITRE', /, 16X, 5'-']
15 TERMINER
EXECUTER à partir de 1

                TITRE
                -----

TERMINE EN LIGNE 15
```


3. Spécification de description

On ne peut pas mettre dans un format des noms de variables, mais seulement des constantes ou des spécifications. A chaque spécification doit correspondre, après le crochet fermant, une expression.
Pour les expressions numériques 3 types de spécifications peuvent être utilisées :
- La spécification U
- La spécification F
- La spécification E

a) La spécification U : provoque l'affichage d'un nombre sous sa forme usuelle (avec une précision de 8 chiffres significatifs) soit :
- le signe éventuel et,
- les chiffres avant le point décimal
- le point décimal éventuel
- les décimales nécessaires
- un espace

ou
- la mantisse (comprise entre 1 et 10, 10 exclu)
- le point décimal éventuel
- les décimales nécessaires
- E
- l'exposant (1 ou 2 chiffres précédés éventuellement du signe —)

Exemples :

```
? — 1155
— 1155
?0.095
0.095
?0.0005
5E—4
?123456789
1.23456789E8
```

b) La spécification F (syntaxe : Fn1.n2) :
Cette spécification se compose :
- de la lettre F
- du nombre n1 de caractères (chiffres, signe ou espaces) avant le point décimal,
- du point décimal,
- du nombre n2 de chiffres après le point décimal.

n1 et n2 doivent être compris entre 0 et 255

Remarques :

- Les chiffres manquants avant le point décimal sont remplacés par des espaces ; si après le point décimal il y a des chiffres superflus, un arrondi est effectué ; s'il en manque ils sont remplacés par des 0.
- Si le nombre de chiffres avant le point décimal est supérieur à n1, tous les chiffres seront affichés (le format n'est pas respecté).

Exemple :

```
10 A ← 0.0236
12 AFFICHER [/,'AU FORMAT F2.4 :',/]
15 FAIRE 20 POUR I ← 1 JUSQUA 4
16 A ← A* 10
20 AFFICHER [U,'s"affiche : ',/,F2.4,/]A,A
25 TERMINER

EXECUTER à partir de 1

AU FORMAT F2.4 :
0.236 s'affiche :
 0.2360
2.36 s'affiche :
 2.3600
23.6 s'affiche :
23.6000
236 s'affiche :
236.0000

TERMINE EN LIGNE 25
```


c) La spécification E (syntaxe En1.n2) :

n1 et n2 ont le même rôle que dans la spécification F.
Ce format est utilisé pour représenter des nombres, très grands ou très petits en valeur absolue. Ces nombres sont alors mis sous la forme :

$$a * 10 \uparrow (n) \text{ avec } 1 \leqslant a < 10$$

Exemple :

```
10 A ← 0.0036
12 AFFICHER [/ ,' AU FORMAT E2.4 :',/]
15 FAIRE 20 POUR I ← 1 PAS 100 JUSQUA 500
16 A ← A * I
20 AFFICHER [/, U, 's" affiche :', / ,E2.4]A, A
25 TERMINER
EXECUTER A PARTIR DE 1
AU FORMAT E2.4 :
0.0036 s'affiche :
 3.6000E—3
0.3636 s'affiche :
 3.6360E—1
73.0836 s'affiche :
 7.3084E1
21998.164 s'affiche :
 2.1998E4
8.8212636E6 s'affiche :
 8.8213E6
TERMINE EN LIGNE 25
```

- Pour les expressions chaînes, booléennes ou graphiques, seule est autorisée la spécification universelle U.

La variable est alors affichée telle qu'elle se présente.

Exemple :

```
10 CHAINE A;BOOLEEN B1,B2
15 A ← ' LES 2 CONSTANTES BOOLEENNES SONT :
20 B1 ← . VRAI . ; B2 ← . FAUX.
25 AFFICHER [/,U,U,3X,U,/]A,B1,B2
30 TERMINER
EXECUTER A PARTIR DE 1
LES 2 CONSTANTES BOOLEENNES SONT : . VRAI .      . FAUX .
TERMINE EN LIGNE 30
```

INSTRUCTION LIRE sans format

Syntaxe :

LIRE var1, var2, var3,....

var1, var2, var3... sont, soit des variables simples, soit des variables indicées, soit des noms de tableaux (à l'exception des variables formes et des tableaux formes).

Cette instruction arrête le déroulement du programme, pour que l'utilisateur puisse taper au clavier la liste des données. Un signal sonore est émis à chaque demande de donnée. Lorsque var est un nom de tableau, il est nécessaire de rentrer autant de valeurs que le tableau comporte d'éléments (sans qu'il y ait sifflement entre chaque élément).

Exemple :

```
10 TABLEAU TAB[2,3]
15 LIRE TAB
20 TERMINER
```

Au sifflement, lors de l'exécution, il faut rentrer les données et les valider une par une, dans l'ordre croissant des indices : les éléments sont lus ligne par ligne.

```
EXECUTER à partir de 1
1 2 3
4 5 6 (le passage à la ligne s'est fait automatiquement)
TERMINE EN LIGNE 20

?TAB
1 2 3
4 5 6
```


INSTRUCTION LIRE avec format

On peut utiliser les formats avec l'instruction LIRE.
Ceci permet de faire imprimer des libellés avant la lecture des données.
- Les variables à lire sont au format U
- Les facteurs de répétition sont fixes.

Exemple :

```
LIRE[/', rentrer au clavier', / , 'Le nombre a = ' , U, / , 'Le nombre b =' ,U,
/]A,B
```

INSTRUCTION LIRE expression

Il a été prévu la possibilité de rentrer directement des expressions à l'aide de l'instruction LIRE, lorsque les données à lire sont de type numérique ou booléen. Il suffit pour cela de faire précéder la donnée tapée du caractère :

Exemple :

```
10 LIRE A
20 AFFICHER A
30 TERMINER
EXECUTER A PARTIR DE 1
:3/4
0.75
TERMINE EN LIGNE 30
```

Cette possibilité n'est pas offerte en CALBUR.

LIRE et AFFICHER avec numéros de voie

En plus des spécifications déjà rencontrées à l'intérieur des formats, il est possible d'utiliser des spécifications de voies d'entrée-sortie logiques selon la syntaxe suivante :

AFFICHER [@n.....].... ou AFFICHER [@*, ...]n.....
LIRE [@n....]...

n est le numéro de voie logique d'entrée ou de sortie.
Dans le cas de l'instruction AFFICHER, il est possible d'utiliser un numéro de voie variable.
Dans le cas de l'instruction LIRE le numéro de voie est un numéro de voie d'entrée : cela signifie que les libellés affichés par LIRE sont toujours sur l'écran.

Exemples :

AFFICHER [@1]' coucou'
sort coucou sur l'imprimante
AFFICHER [@*]V, 'coucou'
affichera coucou, sur l'écran si V = 0, et sur l'imprimante si V = 1

INSTRUCTION ALLER EN ...

Syntaxe : ALLER EN ea
ea étant une expression numérique dont la valeur arrondie est un numéro de ligne compris entre 1 et 32767

Au lieu de se poursuivre à l'instruction suivante, le programme va directement à la ligne de numéro (ea).

Remarque : La ligne de numéro (ea) doit effectivement exister, sinon une ERREUR E 7 est détectée.

INSTRUCTIONS CONDITIONNELLES

1re syntaxe :

SI eb ALORS inst1
eb expression booléenne,
inst1 est une instruction (éventuellement vide).

Si eb a la valeur . VRAI . , inst1 est exécutée; Si eb a la valeur . FAUX . inst1 est ignorée.

2e syntaxe :

SI eb ALORS inst1 SINON inst2

eb expression booléenne,
inst1 et inst2 sont des instructions (éventuellement vides).

SiI eb a la valeur . VRAI ., inst1 est exécutée et inst2 est ignorée; au contraire si eb a la valeur . FAUX ., inst1 est ignorée et inst2 est exécutée.

Remarque :
Après le ALORS ou le SINON on peut créer un bloc d'instructions; ce bloc doit être encadré par :

DEBUT (liste d'instructions séparées par des points-virgules) FIN
inst1 ou inst2 peuvent être elles-mêmes des instructions conditionnelles; dans ce cas chaque SINON est associé au ALORS le plus proche.