ASELEC

INFORMATIQUE PEDAGOGIQUE

L S E G - E D L

Langage Symbolique d'Enseignement Graphique

THOMSON TO7 TO7-70 MO5

ASELEC Avenue de Paris 78820 Juziers Tél : (3) 475.21.02

TABLE DES MATIÈRES

INTRODUCTION

Conçu dès 1970 par l Ecole Supérieure d'Electricité, à la demande du Ministère de l'Education Nationale, le L.S.E. (Langage Symbolique d'Enseignement) a été, au niveau du logiciel de base, le principal support de l'introduction de l'informatique dans le système éducatif français.

Il a été d'abord implémenté sur les mini-ordinateurs T 1600 et Mitra 15, dans le cadre de l'expérience dite des « 58 lycées », puis sur les micro-ordinateurs, à base de Z 80, de l'opération « 10 000 micros ». On le trouve à nouveau, dans le marché dit « 4e tranche », sur des micros à base de 8088/8086 ou 6809.

Normalisé et enrichi par l'AFNOR, ce langage évolué à syntaxe française mérite qu on s'y arrête :
— Mise en œuvre commode, avec une analyse syntaxique par ligne.
— Possibilité de définir et d'utiliser de véritables procèdures, avec passage de paramètres, par valeur et par adresse, récursivité simple et croisée.
— **Gestion autonome des fichiers, rendant le système d'exploitation transparent à l'utilisateur.**
— Prise en compte d'objets, d'opérateurs et de fonctions graphiques.

Le L.S.E. que nous allons décrire est celui de la société E.D.L. implémenté sur T07, T07-70 et M08, sous le nom de LSEG-EDL. Il a été édité spécialement pour l'Éducation Nationale et l'UGAP par la société ACT Informatique.
— **Il gère complètement les 16 couleurs du T07-70 et du M05 de telle manière qu'un** programme écrit sur T07 puisse tourner sans aucune modification sur T07-70 et M05 Un programme écrit sur T07-70 ou M05 et utilisant les couleurs pastel fonctionnera sans modification sur T07 mais remplacera chacune des couleurs pastel par la couleur saturée de même nom.
— **L'utilisation en mode Calculateur de Bureau (CALBUR) est très riche : elle par**met en particulier l'appel des procédures (qu'elles soient écrites en L.S.E. ou binaires, internes ou externes).
— Un soin tout particulier a été apporté à la transmission des paramètres dans les procédures, y compris les cas particuliers (paramètres synonymes transmis par adresse, par exemple)
— **La gestion de la mémoire, lors de l'utilisation de procédures externes, optimise** le nombre de chargements des procédures qui sont conservées tant qu'il y a de la

place, même si elles sont inactives.
— L'arithmétique permet de faire des calculs sur des entiers sans aucun arrondi avec 8 chiffres. Les fonctions mathématiques donnent le plus souvent 9 chiffres significatifs exacts ce qui est un maximum, compte tenu de la représentation interne des nombres.
— Enfin, on trouve un certain nombre d'innovations intéressantes : les chaînes formatées, par exemple, ou encore la possibilité de lecture directe d'expressions numériques ou booléennes.

Les pages qui suivent n'ont pas pour ambition d'enseigner la programmation : elles s'adressent en fait (et qu'il nous soit permis de le regretter sans autre commentaire), à des utilisateurs effectifs ou potentiels qui connaissent déjà par ailleurs le L.S.E., et ou un autre langage sur les nano-machines Thomson :
— le Manuel Utilisateur est conçu pour faire acquérir rapidement les notions propres au L.S.E. et ne se veut nullement un document « autodidactique » ;
— le Manuel de Référence et les Annexes, plus classiques, s'attachent néanmoins à mettre en évidence la qualité et les spécificités de l'implémentation.

MANUEL UTILISATEUR

CHAPITRE I

I MISE EN ROUTE

MISE EN ROUTE

I.1 MISE EN ROUTE SUR TO7, TO7-70 ET MO5

Lunite centrale est munie de sa cartouche LSEG-EDL

Allumer dans l'ordre :
- le téléviseur
- le lecteur - enregistreur de cassettes
- l'unité (ou les unités) de disquettes
- l'unité centrale

Sur TO7 un MENU s'affiche à l'écran. Répondez par 1 ou 2 selon votre choix
Vous avez choisi 2 : (règlez votre crayon et revenez au MENU)
Vous avez choisi 1 : le MENU disparaît de l'écran.
Le message DATE (JJ, MM, AA)?
saffiche en haut de l écran (ce message apparaît directement sur MO5, sans passer par le MENU)

Exemple :

Nous sommes le 6 février 1984
Vous tapez 6 2 84 puis vous validez en appuyant sur la touche « ENTREE »
Vous devez encore taper BO puis appuyer sur la touche « ENTREE » : Le système complete et affiche BONJOUR.
Vous venez de taper votre première commande Son rôle est principalement d'initialiser la zone utilisateur dans laquelle vous allez travailler.

I.2 SYNTAXE DES COMMANDES L.S.E.

Il n'est pas possible d'utiliser le L S.E sans connaître son langage de commande
Les commandes L S E. s'activent en tapant les deux premiers caractéres suivis d'une validation par la touche ENTREE (symbolisée dans la suite par le signe Ø). Il pourra ensuite vous être demande des compléments d'information pour certaines des commandes

Exemple :

EXØECUTER A PARTIR DE 1Ø
(les caractères soulignés sont ceux tapes par l utilisateur)
On s aperçoit qu une commande est exécutée lorsque le curseur passe au debut de la ligne suivante.
La totalite des commandes est décrite dans la partie « Manuel de Référence » mais nous aurons l'occasion d'en voir un certain nombre au fur et à mesure des exemples.

I.3 DISQUETTES ET CASSETTES

Le L.S E permet d utiliser indifférement cassettes et disquettes Ces unités sont connues du L.S.E. sous le nom de disques, numérotés de la manière suivante .
disque 0 : lecteur-enregistreur de cassettes
disque 1 : unité de disquettes du bas
disques 2 a 4 : autres unités de disquettes numérotées de bas en haut.
A la mise en route L S.E. fait le choix du disque par defaut : si il y a une unité de disquettes sous tension c est le disque 1. sinon c'est le disque 0. (Voir dans le Manuel de Référence la commande Disque)

I.4 USAGE DE TOUCHES PARTICULIÈRES

- La touche « ENTREE » (Ø) valide tout message tapé
- La touche « STOP » permet d'interrompre le L S E. Ce caractère provoquera l'apparition du message « PRET »
- La touche « FLECHE GAUCHE » permet d'effacer le dernier caractére tapé
- La touche « RAZ » permet d effacer l ecran et de positionner le curseur en haut et a gauche de l écran
- La touche « m/M » : il s'agit sur TO7 de la touche marquée d'un point jaune, et sur MO5 de la touche jaune. En appuyant sur cette touche « m/M » et en tapant simultanément sur la touche « BARRE D ESPACEMENT » vous passez en mode

« minuscules » (le voyant min s'allume sur TO7) :
Toutes les lettres que vous taperez alors seront en minuscules. Dans ce mode vous pourrez obtenir une lettre majuscule en appuyant simultanément sur « m/M » et la lettre désirée
Vous repasserez en mode majuscule en appuyant à nouveau simultanement sur la touche « m/M » et sur la « barre d'espacement »
• La touche « ACC » permet d'obtenir 12 caracteres qui ne figurent pas tous sur le clavier
Il suffit pour cela d'appuyer sur la touche « ACC » puis sur l'une des touches situées en haut du clavier.
On obtient dans l'ordre

\		←	[	]	é	e	ù	ç	à	)	}
1	2	3	4	5	6	7	6	9	0	–	+

Pour vous en souvenir, dessinez ces caractères sur de petites pastilles et collez celles-ci sur le clavier.

Cette même touche doit être utilisée pour obtenir des minuscules avec accent circonflexe ou trema*. La méthode à suivre pour obtenir ces minuscules est la suivante :
- se mettre en mode minuscules
- appuyer sur la touche « ACC »
- appuyer simultanément sur :
  - la touche « m/M »
  - la touche « CNT » (pour le tréma seulement)
  - la touche sur laquelle est dessiné l accent circonflexe.
- appuyer sur la minuscule désirée

Il est ainsi possible d'obtenir : â é î ô û et e ï u*
Les erreurs de manipulation se traduisent par des taches uniformes sur l'écran (caractere de code 127)

(*) Sur TO7 seulement.

CHAPITRE II
DÉCOUVERTE DU L.S.E.

- **UN PROGRAMME SÉQUENTIEL**
- **ÉDITEUR DE LIGNE**
- **EXEMPLE DE MENU**
- **APPRENONS A LIRE**

DÉCOUVERTE DU LSE

II.1 UN PROGRAMME SÉQUENTIEL

Pour commencer, voici un petit probleme tout simple. Il s'agit de faire un programme reproduisant le dessin ci-dessous.

```
* * * * *
*
*
* * * * *
*
*
* * * * *
```

Pour obtenir ce résultat, convenons de traiter les lignes du dessin l'une après l'autre en partant du haut. On peut traduire cela de la manière suivante

- début de traitement
- on dessine d'abord * * * * *
- ensuite * sur la ligne suivante
- encore *
- a nouveau * * * * *
- on dessine *
- on dessine *
- on dessine * * * * *
- fin de traitement

Ce que realise le programme suivant :

```
1 * Dessin d'un E
20
30 AFFICHER '* * * * *
40 AFFICHER '*'
50 AFFICHER '*'
60 AFFICHER '* * * * *'
70 AFFICHER '*'
80 AFFICHER '*'
90 AFFICHER '* * * * *'
100
110 TERMINER
```

Comme tout programme celui la comporte.

- Un début : la ligne 1 qui comporte un commentaire (le symbole « * » indique que la suite est un commentaire et ne doit pas être interpretée par le système)
- La partie principale qui développe l'algorithme ici les lignes 30 a 90
- Une fin. ici en ligne 110

Les lignes 20 et 100 ne servent qu à « aérer » le listing Ce programme L.S.E. est donc forme de lignes contenant soit des commentaires (ligne 10), soit des instructions (les lignes 30 à 90, ainsi que la ligne 110), soit rien (lignes 20 et 100)
Attention, quand on tape une ligne on doit taper un espace après le numéro de ligne. sinon on voit apparaitre sur l'ecran le message :
ERREUR C 7
Un tel message (ERREUR C xx - voir en Annexe la signification des numéros d'erreurs) nous indique que la ligne n'est pas syntaxiquement correcte Il nous faudra soit la corriger (voir plus loin « Editeur de ligne » soit la retaper.
Ce programme utilise deux instructions : l'instruction AFFICHER qui sert ici à faire apparaître des messages sur l'écran et l'instruction TERMINER. Cette instruction est la derniere qui est exécutee par le programme.
Regardons le titre du paragraphe. il annonce un programme séquentiel cela signifie que le déroulement du programme se fera en commençant par la ligne 10. puis la ligne 20, puis la 30 et ainsi de suite jusqu'à la ligne 110 sans en sauter aucune et sans revenir en arriere
Le fonctionnement de l instruction AFFICHER est le suivant ; regardons ou se trouve le curseur (« _ ») avant l'exécution de cette instruction, à partir de cette position le curseur ira au debut de la ligne suivante. puis dessinera la suite de caractéres représentée entre ' (on dit entre apostrophes) et s arrêtera à droite du dernier caractère dessiné Il est temps de lancer l'exécution du programme;

pour cela nous allons taper une commande (ce qui signifie que nous allons donner un ordre au L.S.E.) Il s'agit de la commande EXecuter. Cela se passe en trois temps

- on tape d'abode EX puis on valide par ENTREE (Ø)
- le L.S.E complète en EXECUTER A PARTIR DE et attend un complement
- on tape 1 et on valide

Ce qui donne à l'écran :

```
EXØECUTER A PARTIR DE 1Ø

* * * * *
*
*
* * * * *
*
*
* * * * *
TERMINE EN LIGNE 110
```

II.2 ÉDITEUR DE LIGNE

L'éditeur de ligne permet de corriger une ligne de programme sans avoir à la retaper. Sa syntaxe est la suivante

/caractères à remplacer/caractères remplaçants

ou encore

%caracteres à remplacer%caractères remplaçants

Les deux formes sont nécessaires car l'utilisation de / ... / .. ne permet pas de remplacer une chaîne contenant le caractère « / » de même que % .. % ne permet pas de remplacer « % ».

Exemples :

```
320 AFICHER 'Bonjour'
ERREUR C 3
/F/FF
320 AFFICHER 'Bonjour'
 qu/ou
320 AFFICHER  Bonjour'

   LIØSTER LIGNES 400Ø
400 C←X*5/Y
%/Y%*Z
400 C←X*5*Z
LISTER LIGNES 400
400 C←X*5*Z

LISTER LIGNES 320
320 AFFICHER 'Bonjour'

/32/83
830 AFFICHER 'Bonjour'
//5
5830 AFFICHER 'Bonjour'
LIØSTER LIGNES *Ø


.....
320 AFFICHER 'Bonjour'

.....

830 AFFICHER 'Bonjour'

.....

5830 AFFICHER 'Bonjour'

.....
```

Un seul « / » (ou un seul « % ») permet des ajouts en fin de ligne :

```
LISTER LIGNES 320
320 AFFICHER 'Bonjour'

/;RLLER EN 999
320 AFFICHER 'Bonjour';RLLER EN 999
```

II.3 UN EXEMPLE DE MENU

Soit à réaliser un programme qui dessinerait sur l'écran le menu suivant :

```
M E N U

*****

SALADE NICOISE

*****

ENTRECOTE-FRITES

*****

FROMAGE OU DESSERT

*****
```


On peut réaliser le programme demande sous la forme suivante :

```
1 * Menu
20
30 AFFICHER [2/,'       M E N U'I
40 AFFICHER [2/,'         *****']
50 AFFICHER [2/,'    SALADE NICOISE']
60 AFFICHER [2/,'         *****']
70 AFFICHER [2/,'  ENTRECOTE-FRITES']
80 AFFICHER [2/,'         *****'I
85 AFFICHER [2/,' FROMAGE OU DESSERT'I
90 AFFICHER [2/,'         *****'I
100
110 TERMINER
```

Une nouveauté dans ce programme les « 2/ » contenus dans chacune des instructions AFFICHER. D'une part « / » provoque le passage au début de la ligne suivante, d'autre part, le « 2 » indique que l'opération est réalisée 2 fois, ce qui revient a laisser une ligne vide.

Une autre maniere d analyser notre problème est décrite dans l algorithme suivant

- debut
- réaliser 4 fois :
  - affichage d'un message
  - affichage des astérisques
- fin

Bien sûr le message est différent à chaque fois Nous allons mettre les 4 messages dans une table, nous désignerons chaque message par son numero. Le troisieme message est par exemple « ENTRECOTE FRITES » Ecrivons le programme :

```
1 * Menu (2eme version)
20
30 TABLEAU CHAINE MES[4]
40 MES[1]←'      M E N U';MES[2]←'   SAL
ADE NICOISE'
50 MES[3]←'  ENTRECOTE-FRITES';MES[4]←'
FROMAGE OU DESSERT'
60
70 FAIRE 100 POUR [←1 JUSQUA 4
80 AFFICHER [2/]MES[I]
100 AFFICHER [2/,7X,5'*']
110
120 TERMINER
```

En ligne 30 on désigne par « MES » une table contenant 4 élements Ces éléments sont des chaînes c'est à dire des suites de caracteres. La declaration « tableau chaîne » est imposee par L.S.E Nous choisissons le nom et le nombre d'eléments. En ligne 40 et 50 on met les messages dans la table. La ligne 70 indique que les lignes 80 et 100 seront exécutées 4 fois chacune En ligne 80 MES[I] designe le ième message. Dans la ligne 100 il y a une liste de 3 éléments entre crochets :

- le premier indique 2 sauts de ligne
- le second le trace de 7 espaces
- le troisième le tracé de 5 astérisques

Noter que le X désigne un espace lorsqu il est entre crochets dans l'instruction AFFICHER. Si l'on execute ce programme on obtient le dessin desire.
Pour remplir les éléments de la table on utilise l instruction d'affectation symbolisée par le signe « ← » (remplacé par « _ » sur certains terminaux - cas de la majorité des imprimantes) C est cette même instruction qui sert à initialiser la variable I de la ligne 70.

II.4 APPRENONS À LIRE

Les exemples que nous avons rencontrés jusqu à présent ne comportaient que des affichages figes et n avaient pas besoin de donnees supplementaires pour s exécuter
L instruction LIRE permet de transmettre des donnees a un programme a partir du clavier
Prenons l exemple d'un programme calculant un quotient. On peut l ecrire :

```
1 * Calcul d'un quotient
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 TERMINER
```

Lors de la rencontre de l'instruction LIRE A en ligne 10, l'ordinateur s'interrompt et attend la frappe d une valeur numérique au clavier. Cette valeur sera rangée dans un emplacement mémoire repéré par l'étiquette A ; on dit que A est un identificateur En ligne 20, la valeur tapée sera repérée par B et en ligne 30 la machine affichera « Le quotient est » suivi de la valeur de A divisé par B (A/B)

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Dividende ? 23
Diviseur ? 7
Le quotient est 3.2857143
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Dividende ? 3.55
Diviseur ? 0.71
Le quotient est 5
TERMINE EN LIGNE 40
```

Il est fastidieux d'être obligé de tapez EX@ et 1@ a chaque fois pour relancer le programme. Aussi, allons-nous le modifier : au lieu de lui dire de terminer en ligne 40, nous allons lui dire de recommencer c'est-à-dire de revenir en ligne 10
Tapons :

```
40 ALLER EN 10
LISTER LIGNES *
1 * Calcul d'un quotient
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 ALLER EN 10
```

La nouvelle ligne 40 a remplacé l'ancienne.

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Dividende ? 32747
Diviseur ? 2672
Le quotient est 12.255614

PRET EN LIGNE 10
```

Notre programme ne s'arrête plus et « boucle » indéfiniment. Le seul moyen de l'interrompre est d'appuyer sur la touche « STOP ».
Ce moyen n'est guère élégant. Il est préférable de poser, a la fin du calcul, une question à l'utilisateur.

Tapons

```
5 CHAINE REP
40 AFFICHER 'Voulez-vous continuer (O ou
 N) ? ';LIRE REP
50 SI REP='N' ALORS TERMINER
60 ALLER EN SI REP='O' ALORS 10 SINON 40
LISTEN LIGNES *
1 * Calcul d'un quotient
5 CHAINE REP
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Voulez-vous continuer (O ou
 N) ? ';LIRE REP
50 SI REP='N' ALORS TERMINER
60 ALLER EN SI REP='O' ALORS 10 SINON 40
```

Remarquons que les lignes que l'on vient d'ajouter se sont correctement insérées aux bons endroits.
La ligne 5 est une ligne de déclaration Elle est obligatoire pour indiquer au système que l'identificateur REP représentera non pas un nombre comme A et B mais une chaîne de caracteres (normalement réduite à un seul caractère O ou N), formée par la réponse de l'utilisateur à la question qui lui est posée en ligne 40
En ligne 50 nous avons une instruction conditionnelle TERMINER ne sera executé que dans le cas ou le contenu de REP sera effectivement un « N »
En ligne 60, ALLER EN est suivi d une expression numérique conditionnelle Cette expression vaut 10 si le contenu de REP est un « O », elle vaut 40 dans tout autre cas

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Voulez-vous continuer (O ou N) ? O
Dividende ? 63
Diviseur ? 7
Le quotient est 9
Voulez-vous continuer (O ou N) ? NON
Voulez-vous continuer (O ou N) ? n
Voulez-vous continuer (O ou N) ? N
TERMINE EN LIGNE 50
```

Une autre maniere de résoudre ce problème est l utilisation de la structure de « boucle »
Tapons :

```
LISTER LIGNES 5
5 CHAINE REP

 ;REP←'O'
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
EP@FACER LIGNES 600

LISTER LIGNES *
1 ' Calcul d'un quotient
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ?  ';LIRE B
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
```



```
EXECUTER A PARTIR DE 1

Dividende ? 39
Diviseur ? 3
Le quotient est 13
Tapez "O" pour continuer O
Dividende ? 72
Diviseur ? 9
Le quotient est 8
Tapez "O" pour continuer o
TERMINE EN LIGNE 50
EXECUTER A PARTIR DE 1

Dividende ? 45
Diviseur ? 9
Le quotient est 5
Tapez "O" pour continuer ZUT
TERMINE EN LIGNE 50
```

Cette boucle est équivalente a un saut conditionnel (SI REP = O' ALORS ALLER EN 10) en fin de ligne 40.

Essayons quelque chose

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 0
ERREUR E 31 EN LIGNE 30
```

Dans la liste des erreurs d execution (voir Annexes) nous trouvons
ERREUR E 31 Division par zéro
Effectivement nous avons donne une valeur nulle au diviseur, B valait donc zéro et la machine a tente, en ligne 30, d executer A/B Or comme chacun sait, une division par zéro est impossible. Modifions notre programme pour lui apprendre cette regle et lui éviter de partir en erreur

Tapons

```
25 SI B=0 ALORS DEBUT AFFICHER 'Une divi
sion par zero est impossible ';ALLER EN
 40 FIN
LISTER LIGNES *
1 * Calcul d'un quotient
5 CHAINE REP;REP←'O'
8 FAIRE 40 TANT QUE REP='O'
10 AFFICHER 'Dividende ? ';LIRE A
20 AFFICHER 'Diviseur ? ';LIRE B
25 SI B=0 ALORS DEBUT AFFICHER 'Une divi
sion par zero est impossible ';ALLER EN
 40 FIN
30 AFFICHER 'Le quotient est ',A/B
40 AFFICHER 'Tapez "O" pour continuer ';
LIRE REP
50 TERMINER
```

Les mots DEBUT et FIN en ligne 25 ont un rôle de parenthésage afin de regrouper les deux instructions AFFICHER et ALLER EN en une seule instruction conditionnelle. Ainsi si B est nul nous enverrons le message « Une division par zero est impossible ! » à l'écran et nous sauterons la ligne 30 afin de ne pas évaluer A/B et donc ne pas produire d'erreur

```
EXECUTER A PARTIR DE 1

Dividende ? 12
Diviseur ? 3
Le quotient est 4
Tapez "O" pour continuer O
Dividende ? 12
Diviseur ? 0
Une division par zero est impossible '
Tapez "O" pour continuer O
Dividende ? 5
Diviseur ?              ... etc
```


Notre programme est maintenant muni d une alternative et est capable d exécuter le traitement adapté aux données qui lui sont communiquées.
Le problème que nous nous sommes posé est un problème très simple et il n est point besoin d'utiliser un ordinateur pour calculer un quotient (une calculatrice le fera tout aussi bien) Il nous a cependant permis de voir outre les instructions LIRE et AFFICHER les notions de répétition et d'alternative.

CHAPITRE III
NOTIONS FONDAMENTALES DU L.S.E.

- MODE CALBUR
- MODE PROGRAMME
- LIGNES
- CONSTANTES
- VARIABLES SIMPLES
- VARIABLES INDICÉES
- OPÉRATEURS
- EXPRESSIONS

NOTIONS FONDAMENTALES DU LSE

III-1 MODE CALBUR ou calculateur de bureau

Ce mode permet d'évaluer toute expression L.S.E. et d'en afficher la valeur
– L'expression doit être precedée du symbole ? signifiant AFFICHER et suivie du caractère ENTREE signalant la fin de l'instruction
– Toutes les opérations et les fonctions L S E peuvent être utilisées

Exemple :

```
? LGR( 'Bonjour ' ' Madame ')
14
```

– On peut aussi écrire une ligne de programme (sans numéro de ligne). Apres ENTRÉE la ligne est analysée syntaxiquement et, si elle est correcte elle est immédiatement exécutée.

Exemple :

```
A←2 ; B←5 ; ?A>B
.FAUX.
```

Les instructions interdites sont PAUSE, TERMINER, EXECUTER, PROCEDURE, RESULTAT, RETOUR et les instructions faisant référence à un numéro de ligne (RETOUR, RETOUR EN, ALLER EN).
– En mode CALBUR on peut également effectuer les opérations sur fichiers (GARER, CHARGER, SUPPRIMER)

Le mode CALBUR est aussi souvent utilise pour la mise au point des programmes. En utilisant la commande PAs à pas ou l'instruction PAUSE on peut connaître les valeurs prises par certaines variables.

III-2 MODE PROGRAMME

– Un programme est une suite d'instructions qui permettent à la machine d'effectuer l'ensemble des opérations nécessaires à la réalisation d'un travail donné
– Un programme est constitué de lignes commençant par un numéro. Lexecution des traitements contenus dans ces lignes est différe jusqu à ce que l on envoie à la machine la commande EXécuter

III-3 LIGNES

Pour ecrire une ligne de programme, on doit taper dans l'ordre :
– Un numéro de ligne (compris entre 1 et 32000) suivis d au moins un espace.
– Une ou plusieurs instructions séparées par des points virgules. La dernière instruction peut être suivie d'un commentaire : celui-ci doit être alors précédé d une étoile (*).

Exemple :

```
60 A←2;B←5;* Initialisation des variable
s A et B
```

Le caractere ENTREE, signalant la fin de la ligne.
A ce moment la ligne est analysée :
Si elle est syntaxiquement correcte elle est intégrée au programme (s'il existe déja une ligne de même numéro, elle la remplace).
Sinon un message d'erreur est envoye :
ERREUR C n (erreur dite de compilation, voir liste en Annexe)
On peut corriger cette ligne (voir Editeur de ligne) ou la retaper.
Une ligne ne peut contenir plus de 250 caractères.

III-4 LES CONSTANTES

Une donnée introduite une fois pour toute dans le programme s'appelle une constante,

IV-4.1 Les constantes numeriques

Ce sont des nombres écrits sous leurs formes usuelles a deux details pres
La virgule decimale est remplacee par un point

Exemple :

4 5 secrit 4 5

Les puissances de 10 sont symbolisees par la lettre E

Exemple :

1 234567 10^{6} s écrit 1 234567E6
0 25 10^{-1} s écrit 2 5E-5

IV-4.2 Les constantes chaînes

Une constante chaîne est une suite de caracteres Ces caracteres peuvent être des lettres des chiffres, des caracteres particuliers (+ − ∗ /.) et d une maniere générale n importe lequel des 256 caractères pouvant être representés sur un octet
Une constante chaîne peut être représentee de plusieurs manières :
– En encadrant les caractères qui constituent la chaîne par des apostrophes ( )
Dans ce cas si l on veut mettre des apostrophes dans la chaîne on mettra 2 apostrophes (ne pas confondre avec le guillemet) A l impression le texte apparaîtra avec une seule apostrophe

Exemple :

```
? 'L''enseignement de l''informatique est
 utile
L'enseignement de l'informatique est uti
le
```

– En désignant les caractères par leurs équivalents decimaux. donnes par le code ASCII separés par un espace la chaîne est alors encadree par deux points

Exemple :

```
? .76 39 73 78 70 79 82 77 65 84 73 81 6
5 69.
L'INFORMATIQUE
```

Ce mode est surtout utilise pour représenter les caracteres de contrôle ou de gestion d'écran ainsi que les caracteres non disponibles au clavier.
– On peut auss mélanger les deux types de representations

Exemple : (le code 10 representant un passage a la ligne)

```
? 'A'.10. E'
A
 E
```

Remarques :
– est une chaîne vide; elle ne contient aucun caractere.
– Les constantes numériques et les constantes chaînes sont de types différents

Exemples :

– ne pas confondre le nombre 10 avec la chaîne '10'
– l expression 10 + '10' n a aucun sens en L S E.

III-4.3 Les constantes booléennes

Ces constantes sont au nombre de 2
la constante notée VRAI
la constante notee FAUX

III-5 LES VARIABLES SIMPLES

Une variable simple est un objet auquel on peut affecter une valeur. cette valeur pouvant être modifiee au cours du programme.
Il existe 4 types de variables en L S.E
Les variables numériques
Les variables chaînes
Les variables booléennes
Les variables graphiques

Les noms de variables appelés IDENTIFICATEURS sont soumis a la règle suivante :
Le nom d'un identificateur est une suite d au plus 5 caracteres alphanumériques (lettres ou chiffres) le premier caractére est obligatoirement une lettre Les minuscules du nom sont automatiquement transformées en majuscules

Exemple :

A, TEXTE, X1, B12 sont des identificateurs
PAS, CIBLE, DEBUT ne peuvent pas être des identificateurs car ce sont des mots réservés par L.S E (voir liste en Annexe).
1A, T 4, PAULINE ne peuvent pas être des identificateurs.
En mode de bureau on ne peut créer que des identificateurs d'une lettre
Les variables chaines booleennes et graphiques doivent être soumises à une déclaration préalable au moyen de l'une des instructions suivantes :
CHAINE . BOOLEEN , FORME

Exemples :

```
20 CHAINE CH,C,LISTE
30 BOOLEEN B,ENCOR
40 FORME CARTE,F,G;* Voir chapitre sur l
e graphique
```

Les variables non déclarees sont considerées comme numériques Les déclarations se font generalement en début de programme et bien sûr avant leur première utilisation.
Une variable peut être definie par l'instruction d'affectation : ' ← .

Exemples :

```
130 A←RAC(2)
250 CHAINE TEXT;TEXT←'rien ne sert de co
urir'
```

Toute tentative d'utilisation d une variable non definie provoquera une erreur d'execution.

III-6 LES VARIABLES INDICÉES

1er exemple :

```
10 LIRE [/,'Nombre d''éléments de la lis
te N=',U,']N
11 * Au signal sonore, on entrera au cla
vier le nombre N puis l'on tapera sur la
 touche ENTREE
16
17 TABLEAU NOMB[N]
20 LIRE NOMB
21 *On entre au clavier la liste de nomb
res en tapant ENTREE après chaque nombre
25 MAX←NOMB[1]
30 FAIRE 36 POUR I←2 JUSQUA N
36 SI NOMB[I]>MAX ALORS MAX←NOMB[I]
40
45 AFFICHER 'Le plus grand élément est:
',MAX
50 TERMINER

EXECUTER A PARTIR DE 1
.
Nombre d'éléments de la liste N=12

45 2 -3 5 6 7 8 112 -1 13 17 59

Le plus grand élément est: 112
TERMINE EN LIGNE 50
```

Nous avons creé un tableau NOMB de N nombres. Ce tableau est de dimension 1
NOMB[1] correspond au 1er élément de la liste
. . .

NOMB[3] correspond au 3e elément de la liste.
. .

Ce qui se trouve entre crochets est l'indice
Ce tableau a été déclaré en utilisant l'instruction TABLEAU.

TABLEAU NOMB[N]

L'instruction de déclaration définit donc
- la dimension du tableau
- le nombre d'éléments du tableau.

2e exemple :

Proposer 5 mots en français et faire écrire à l'utilisateur du programme les traductions en anglais de ces 5 mots (si la réponse est fausse on donne la « bonne » traduction)

```
1 * Traduction de mots français en angla
is
5
10 TABLEAU CHAINE MOT[2,5];CHAINE REP
20 MOT[1,1]←'CHIEN';MOT[1,2]←'CHAT';MOT[
1,3]←'POISSON';MOT[1,4]←'OISEAU';MOT[1,5
]←'VACHE'
25 MOT[2,1]←'DOG';MOT[2,2]←'CAT';MOT[2,3
]←'FISH';MOT[2,4]←'BIRD';MOT[2,5]←'COW'
30
35 AFFICHER [/,'On te propose 5 mots en
français, tu les traduiras en anglais',/
]
40
45 FAIRE 70 POUR I←1 JUSQUA 5
50 AFFICHER [/,U,' ----------> ']MOT[1,I
]
55 LIRE REP;REP←GRL(TMA(REP),1);* On "ne
ttoie" la réponse
60 SI REP=MOT[2,I] ALORS AFFICHER [/,'C'
'est bien',/] SINON AFFICHER [/,'Tu t''e
s trompé,',','la bonne réponse est: ',U,
/]MOT[2,I]
70
80 TERMINER
```



```
EXECUTER A PARTIR DE 1

On te propose 5 mots en français, tu les
 traduiras en anglais

CHIEN ----------> DOG
C'est bien

CHAT ----------> CAT
C'est bien

POISSON ----------> FICH
Tu t'es trompé,
la bonne réponse est: FISH

OISEAU ----------> BEARD
Tu t'es trompé,
la bonne réponse est: BIRD

VACHE ----------> COW
C'est bien

TERMINE EN LIGNE 80
```

Dans ce programme on a crée un tableau à 2 lignes et à 5 colonnes. Ce tableau a été déclaré par l'instruction . TABLEAU CHAINE

TABLEAU CHAINE MOT[2 5] définit un tableau à 2 dimensions et à 10 éléments
Pour repérer un élément de ce tableau il suffit de connaître son numéro de ligne et son numéro de colonne.
MOT[2 3] est l'élément situé à la 2e ligne et à la 3e colonne
MOT[2 3] est une variable indicée.
Un tableau est un ensemble de variables de même type (numériques chaînes booléennes ou graphiques)
Les éléments d'un tableau (variables indicées) sont désignés par
- le nom du tableau
- une liste d'indices (le nombre d'indices est égal à la dimension du tableau).

Il existe 4 types de tableau en L.S.E :
les tableaux numériques les tableaux chaînes: les tableaux booléens les tableaux formes
Ces tableaux doivent être déclarés par l'instruction TABLEAU

Exemple :

```
20 TABLEAU N[5];* Déclaration d'un table
au numérique
30 TABLEAU CHAINE TEXTE[24];* Déclaratio
n d'un tableau de chaînes
40 TABLEAU BOOLEEN EXIST[3,25,25,40];* D
éclaration d'un tableau de booléens
50 TABLEAU FORME SCENE[8];* Declaration
d'un tableau de formes - voir chapitre
sur le graphique
```

III-7 LES OPÉRATEURS

III-7.1 Les opérateurs numériques

- 1 operateur unaire
- 5 opérateurs binaires

- pour le changement de signe
+ pour l'addition
- pour la soustraction
* pour la multiplication
/ pour la division
↑ pour l'exponentiation

Le calcul s'effectue en principe de la gauche vers la droite en tenant compte d'un ordre de priorité decroissante
l'exponentiation
le changement de signe
la multiplication ou la division
l'addition ou la soustraction
Cet ordre peut 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

Exemples :

```
A←10;B←-3;C←4

?A-3*B+C/4
20

?(A+B)/C
1.75

?(-B+(B-3*A*C)↑2)/5
3026.4

?(A+3*B)/(4*C)
0.0625
```

Une expression numérique ne peut évidemment pas contenir 2 operateurs l'un a côté de l'autre
Il est interdit d'élever un nombre negatif à une puissance non entière
Il est également interdit d'élever 0 à une puissance négative ou nulle

III-7.2 La concaténation

Le seul operateur sur chaîne est l'opérateur de concatenation noté ! (point d'exclamation)
Il permet de réunir 2 chaînes en une seule les 2 chaînes étant mises bout a bout.

Exemple :

```
CHAINE A,B,C;A←'François apprend sa leço
n ';B←'d''anglais'
C←A!B
?C
François apprend sa leçon d'anglais
```

III-7.3 Les operateurs booléens

1 opérateur unaire	NON	
3 opérateurs binaires	ET	
	OU	(ou inclusif)
	DIJ	(ou exclusif)

Dans l'évaluation d une expression booléenne l ordre des priorites est le suivant
NDN
ET
OU ou DIJ
On utilisera des parentheses si l on veut modifier l ordre des priorites

III-7.4 Les operateurs de relation

Ces operateurs permettent de comparer 2 expressions de même nature (numériques, chaînes ou booleennes).

=	pour egal
≠	pour different
<	pour inférieur
>	pour supérieur
< =	pour inferieur ou égal
> =	pour supérieur ou egal

Seuls les operateurs = et ≠ peuvent être utilises entre deux expressions booleennes. La comparaison de deux expressions graphiques est pour l instant denuée de fondement

Exemples :

```
? 2<=2
.VRAI.

? 2<RAC(3)
.FAUX.

? 'FRANCE'>'FRANCOIS'
.FAUX.

? 'BEAU''BEAUCOUP'
.VRAI.
```


III-7.5 Les opérateurs graphiques

Ces opérateurs sont au nombre de 2
- la juxtaposition (symbole :)
- la superposition (symbole %)

Voir le chapitre traitant du graphique.

III-8 LES EXPRESSIONS

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

Exemples :

A ← 3; SI A < 2 ALORS 5 SINON 0 est une expression numérique de valeur 0
A ← 2 B ← 1 B * &F (A) est une expression numerique si le résultat de la procedure &F(A) est numérique

C etant une chaîne.
C'SI Z < 10 ALORS ECHEC' SINON 'SUCCES est une expression chaîne.

P ← 15; Q ← 7
P > 7 ET Q < 11 est une expression booleenne de valeur .VRAI.
P > 17 OU Q < = 7 est une expression booléenne de valeur VRAI
P > 7 DIJ Q < 11 est une expression booléenne de valeur FAUX
NON (P > 16) est une expression booléenne de valeur .VRAI.
NON (P < 7) ET Q > 11 est une expression booléenne de valeur .FAUX.
NON (P > 7 ET Q > 11) est une expression booleenne de valeur .VRAI.
CHAINE A: SI A > JEUDI ALORS VRAI SINON .FAUX. est une expression booleenne.

EXPRESSIONS CONDITIONNELLES

Il est possible en L S E de creer des expressions dont la valeur dépend du résul tat d une expression booléenne.
Syntaxe : SI eb ALORS exp1 SINON exp2
eb est une expression booléenne
exp1 et exp2 sont des expressions de même nature
Si eb est vraie l'expression vaut exp1 dans le cas contraire elle vaut exp2

Exemple :

```
A←3

?SI A=1 ALORS 2+2 SINON 3+3
6

?'Plus '!SI A<5 ALORS 'petit' SINON 'gra
nd'
Plus petit
```


CHAPITRE IV
RUPTURES DE SÉQUENCE ET ITÉRATIONS

- INSTRUCTIONS CONDITIONNELLES
- INSTRUCTION ALLER EN
- BOUCLES

RUPTURES DE SÉQUENCE ET ITÉRATIONS

IV-1 LES INSTRUCTIONS CONDITIONNELLES.

Dans un programme il est rare qu'un traitement puisse s'exécuter sequentiellement (c'est-à-dire que toutes les instructions puissent s'executer dans l'ordre croissant des numéros de ligne)
Le plus souvent, selon le résultat de certains tests le traitement sera different

On utilise alors l'instruction SI .. ALORS .
ou l'instruction SI .. ALORS .. SINON .

1er exemple :

instruction SI . ALORS .

Sachant qu'une disquette coûte 20 FRANCS, et que pour un achat d'au moins 25 unités, le commerçant vous fait une remise de 5%, ecrire un programme qui permet, selon le nombre de disquettes achetees, d'afficher le prix a payer.


```
1 * UNE FACTURE
2
10 LIRE [/,'Combien de disquettes désire
z-vous ? N= ']N
15
20 PRIX←N*20
25 SI N>=25 ALORS PRIX←PRIX-PRIX*5/100
30
35 AFFICHER [2/,'TOTAL A PAYER :',U,' FR
ANCS',2/]PRIX
40
45 TERMINER

EXECUTER A PARTIR DE 1

Combien de disquettes désirez-vous ? N=
30

TOTAL A PAYER :570  FRANCS


TERMINE EN LIGNE 45
```

La variable PRIX n'est modifiée que si l'expression booléenne N > = 25 a la valeur VRAI.

2e exemple :

instruction SI . ALORS .. SINON

On se propose de lire 2 nombres et de les comparer

```
1 * Comparaison de 2 nombres
5
10 LIRE [/,'A=',U,'    B=',U,/]A,B
15
20 SI A<B ALORS AFFICHER A,'< ',B SINON
SI A>B ALORS AFFICHER A,'>',B SINON AFFI
CHER A,'= ',B
25
30 TERMINER

EXECUTER A PARTIR DE 1

A=2     B=5

2 < 5
TERMINE EN LIGNE 30
```

Lorsque plusieurs instructions conditionnelles sont imbriquees, chaque SINON est associé au ALORS qui le précède.

3e exemple :

Instruction SI .. ALORS DEBUT ... FIN
Même énonce que dans l'exemple 2

```
10 LIRE [/,'A=',U,'     B=',U,/]A,B
15
20 SI A<B ALORS DEBUT AFFICHER [U,'< ',U
]A,B;TERMINER FIN
25 SI A>B ALORS DEBUT AFFICHER [U,'> ',U
]A,B;TERMINER FIN
30 AFFICHER [U,'= ',U]A,B
35
40 TERMINER
```

Le traitement a effectuer dans le cas ou l'expression booleenne A > B a la valeur .VRAI comporte 2 instructions. Il est obligatoire d'encadrer ces 2 instructions par DEBUT FIN

IV-2 L'INSTRUCTION ALLER EN

Exemple :

```
1 * exemple d' ALLER EN
10
20 AFFICHER 'Ce programme propose d''aff
icher'
25 AFFICHER 'soit: * * * * (choix 1)'
30 AFFICHER 'soit: & & & & (choix 2)'
40 AFFICHER 'soit: ! ! ! ! (choix 3)'
50 AFFICHER 'soit: - - - - (choix 4)'
60
70 TABLEAU CHOIX[4];CHOIX[1]←@110;CHOIX[
2]←@120;CHOIX[3]←@130;CHOIX[4]←@140
80 LIRE [/,'Quel est votre choix ? (donn
er le numéro)    ']I
90 ALLER EN CHOIX[I]
100
110 AFFICHER '* * * *';TERMINER
120 AFFICHER '& & & &';TERMINER
130 AFFICHER '! ! ! !';TERMINER
140 AFFICHER '- - - -';TERMINER

EXECUTER A PARTIR DE 1

Ce programme propose d'afficher
soit: * * * * (choix 1)
soit: & & & & (choix 2)
soit: ! ! ! ! (choix 3)
soit: - - - - (choix 4)
Quel est votre choix ? (donner le numéro
)    3
! . ! !
TERMINE EN LIGNE 130
```

Noter l'utilisation du marqueur @ qui permettra la prise en compte de tous les numéros de ligne par la commande de renumérotation ESpacer lignes.

IV-3. LES BOUCLES

IV-3.1. Présentation

Lors de l'execution d'une tâche, nous avons souvent besoin de répeter un certain nombre de fois une action avant de pouvoir passer a la suite. Par exemple lorsque nous prenons notre café au lait matinal nous devons le sucrer (3 sucres) puis tourner pour faire fondre le sucre.
Cette tâche peut se décrire de la facon suivante
- Repeter 3 fois mettre un sucre dans le cafe
- Si le sucre n'est pas fondu tourner

En utilisant le vocabulaire du L.S.E cette action peut s'ecrire
FAIRE POUR NBSUC ← 1 JUSQUA 3
    - Mettre un sucre dans le cafe
FAIRE TANT QUE sucre non fondu
    - Tourner

Nous avons deux boucles c'est-a-dire deux actions qui se repètent tant que certaines conditions ne sont pas réalisees. Ces deux boucles sont de natures differentes au niveau des conditions de sortie de boucle :
Dans la première, nous comptons le nombre de sucres et c'est la comparaison de ce nombre a 3 qui décide de la sortie de la boucle. Cela impose en particulier de connaître à l'avance le nombre de sucres necessaire.
Dans la deuxième, la condition de sortie est différente : nous ne pouvons pas savoir a l'avance combien de fois ou combien de temps il nous faudra tourner le seul critère d'arrêt etant que le sucre soit complètement fondu.

Il existe deux types de boucles les boucles « JUSQUA » et les boucles « TANT QUE »

Imaginons que nous ne sachions pas combien il faut de sucres pour que notre café soit correctement sucré. Nous pouvons proceder de la manière suivante :
FAIRE TANT QUE café pas assez sucré
    - Mettre un sucre dans le cafe
    - FAIRE TANT QUE sucre pas fondu
        - Tourner

Nous avons ici deux boucles « TANT QUE » imbriquées

IV-3.2 Boucles TANT QUE

Premier exemple :

Nous allons ecrire un programme qui lit une serie de mots au clavier et les place « bout a bout » dans une chaine. Nous decidons de taper FIN pour arrêter la saisie

```
1 * Exemple de boucle TANT QUE
10 CHAINE LISTE,MOT;MOT←'';LISTE←''
20 FAIRE 40 TANT QUE MOT≠'FIN'
30 LISTE←LISTE'MOT!'-'
40 LIRE MOT;AFFICHER [*];MOT←TMA(MOT)
50 AFFICHER 'La liste est: ',LISTE
60 TERMINER

EXECUTER A PARTIR DE 1
chien
chat
canard
oiseau
fin

La liste est: -CHIEN-CHAT-CANARD-OISEAU-
TERMINE EN LIGNE 60
```

Remarques :
- FAIRE est suivi du numero de la derniere ligne de la boucle.
- Une telle boucle peut être écrite avec des ALLER EN : il suffirait de remplacer les lignes 20 et 40 par

```
20 SI MOT = FIN ALORS ALLER EN 50
40 LIRE MOT MOT←TMA(MOT) ALLER EN 20
```

mais le programme serait moins lisible

Deuxième exemple :

Ecrivons un programme qui calcule la moyenne d une série de notes Nous decidons de taper une note non valide (inferieure a 0 ou superieure à 20) pour arrêter la saisie Nous allons avoir besoin d un compteur, qui nous ind que le nombre de notes ainsi rentrees

```
1 * Calcul de moyenne
10 TOTAL←0
20 AFFICHER 'Tapez les notes: '
30 LIRE NOTE
40 FAIRE 50 POUR NBRE←0 TANT QUE NOTE>=0
 ET NOTE<=20
50 TOTAL←TOTAL+NOTE:LIRE NOTE
60 AFFICHER NBRE, 'note(s) tapee(s)'
70 SI NBRE≠0 ALORS AFFICHER 'La moyenne
est: ',TOTAL/NBRE
80 TERMINER

exECUTER A PARTIR DE 1

Tapez les notes: 8 10 12 -1
3  note(s) tapee(s)
La moyenne est: 10
TERMINE EN LIGNE 80
```

Le compteur NBRE est incrémente a chaque passage dans la boucle

Troisième exemple :

Nous allons utiliser e mot PAS pour ecrire un programme qui effectue des decomcositions de nombres entiers en produits de facteurs premiers.
Pour decomposer un nombre en facteurs premiers il faut le diviser par la suite des nombres premiers (soit 2,3 5 7 11 13.17 19.23) Cela allonge rait par trop le temps d execution de faire generer cette suite par l'ordinateur aussi allons-nous nous contenter de faire des divisions par la suite des nombres entiers Cependant nous pouvons faire l economie d un grand nombre de divisions apres 2 les nombres premiers ne peuvent pas être pairs il nous suffit donc d examiner les nombres de 2 en 2 apres 3. les nombres premiers ne peuvent être des multiples de 3. (nous avons deja enleve les multiples pairs il reste donc les multiples impairs qui sont disposés de 6 en 6) Pour les « sauter », il suffit donc de parcourir la suite ces entiers avec un pas alternativement egal a 2 et a 4 (a partir de 7) Notre parcours sera donc 2 2 + 1 = 3 3 + 2 = 5 5 - 2 = 7 7 + 4 = 11 11 + 2 = 13 13 + 4 = 17 17 + 2 = 19 19 + 4 = 23 23 + 2 = 25 25 + 4 = 29
Nous eliminons bien ainsi tous les multiples de 2 et de 3

```
1 * Décomposition en produit de facteurs
 premiers
10 AFFICHER 'Décomposition de ? ';LIRE N
15 SI N<2 OU ENT(N)≠N ALORS DEBUT AFFICH
ER 'Valeur incorrecte';ALLER EN 10 FIN
20 P←1
30 FAIRE 50 POUR I←2 PAS P TANT QUE I<=R
AC(N)
40 SI I*ENT(N/I)=N ALORS DEBUT AFFICHER
[U,X]I;N←N/I;ALLER EN 40 FIN
50 P←SI I)6 ALORS 6-P SINON SI I>2 ALORS
 2 SINON 1
60 AFFICHER [U,X]SI N=1 ALORS '' SINON N
70 TERMINER

EXECUTER A PARTIR DE 1

Décomposition de ? 1001 7  11  13

TERMINE EN LIGNE 70
```

La ligne 50 permet de gerer la valeur du pas si I est egal a 2. P sera egal a 1 ce qui nous donnera bien la valeur de 3 pour I dans le passage suivant Ensuite tant que I est compris entre 3 et 7, le pas P vaut 2, ce qui nous donne pour I les valeurs 5 et 7 Ensuite P reçoit la valeur 6-P et est donc egal alternativement à 4 et à 2 (6 – 2 = 4 6 – 4 = 2)

IV-3.3 Boucles JUSQUA

On utilise ce type de boucle lorsque le nombre de répetitions est connu a l'avance.

Exemple :

```
1 * Affichage d'une phrase N fois
10 CHAINE PH;PH←'Je ne dois pas bavarder
 en classe.'
20 AFFICHER 'Nombre de répétitions ? ';L
IRE N
30 FAIRE 40 POUR I←1 JUSQUA N
40 AFFICHER PH
50 TERMINER

EXECUTER A PARTIR DE 1

Nombre de répétitions ? 5
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
Je ne dois pas bavarder en classe.
TERMINE EN LIGNE 50
```

Il est également possible d utiliser cette instruction lorsque le nombre de repétitions n'est pas connu à l'avance mais peut être calculé dans la boucle (la valeur qui suit JUSQUA est recalculee a chaque passage)

Exemple :

On désire compter et supprimer toutes les voyelles d une chaîne :

```
1 * Compter et supprimer les voyelles
10 CHAINE CH,VOY;VOY←'AEIOUYaeiouyàâéèêë
îïôûù'
20 AFFICHER 'Tapez la chaîne: ';LIRE CH
25 NBRE←0
30 FAIRE 40 POUR I←1 JUSQUA LGR(CH)
40 SI POS(VOY,1,SCH(CH,1,1))≠0 ALORS DEB
UT NBRE←NBRE+1;CH←MCH(CH,I,1,'');I←I-1 F
IN
50 AFFICHER 'Il y avait ',NBRE,' voyelle
(s)'
60 AFFICHER 'La chaîne est maintenant: '
,CH
70 TERMINER

EXECUTER A PARTIR DE 1

Tapez la chaîne: bonjour madame
Il y avait 6  voyelle(s)
La chaîne est maintenant: bnjr mdm
TERMINE EN LIGNE 70
```

Il est possible dans une boucle « JUSQUA » de preciser la valeur de l increment

Exemple :

Multiples de 7 inférieurs a 100 :

```
1 * Multiples de 7
10 FAIRE 20 POUR I←0 PAS 7 JUSQUA 100
20 AFFICHER [U,X]I
30 TERMINER

EXECUTER A PARTIR DE 1
0  7  14  21  28  35  42  49  56  63  70
  77  84  91  98
TERMINE EN LIGNE 30
```

Tout comme pour les boucles « TANT QUE » le PAS peut être variable :

Exemples :

```
 1 * Une façon originale d'obtenir les p
uissances de 2
10 FAIRE 20 POUR I←1 PAS I JUSQUA 1000
20 AFFICHER [U,X]I
30 TERMINER

EXECUTER A PARTIR DE 1
1  2  4  8  16  32  64  128  256  512
TERMINE EN LIGNE 30
```


Ou bien encore.

```
1 * Etude de la série des "1 sur 2 puiss
ance n"
10 AFFICHER 'Précision ? ';LIRE STOP;SOM
ME←0
15 AFFICHER ''
20 FAIRE 30 POUR I←1 PAS -I/2 JUSQUA STO
P
30 SOMME←SOMME+I;AFFICHER [U,X]SOMME
40 TERMINER

EXECUTER A PARTIR DE 1

Précision ? 0.02
1  1.5  1.75  1.875  1.9375  1.96875
TERMINE EN LIGNE 40
EXECUTER A PARTIR DE 1

Précision ? 0
1  1.5  1.75  1.875  1.9375  1.96875  1.
984375  1.9921875  1.996093B  1.99B0469
 1.9990234  1.9995117  1.9997559  1.999B
779  1.999939  1.9999695  1.9999B47  1.9
999924  1.9999962  1.9999981  1.999999
1.9999995  1.999999B  1.9999999  1.99999
99  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2  2  2  2
ERREUR E 44 EN LIGNE 20
```

IV-4.4 Remarques concernant les boucles (N1 FAIRE N2...)

– Si dans la boucle on rencontre l'instruction ALLER EN N3 (N3 non compris entre N1 et N2) on sort de la boucle même si la condition de sortie n'est pas réalisée.

– Si l'on arrive dans la boucle par l'instruction ALLER EN N3 (N3 compris entre N1 et N2) alors la ligne N3 et les suivantes se comportent comme des lignes ordinaires (fortement déconseillé)
– Si dans le corps de la boucle, une instruction renvoie a l instruction FAIRE celle-ci est réexécutée et tous les paramètres sont réinitialisés

CHAPITRE V
PROCÉDURES

- INTRODUCTION
- INSTRUCTION PROCEDURE
- VARIABLES LOCALES, VARIABLES GLOBALES
- PASSAGE DE PARAMÈTRES PAR ADRESSE
- PASSAGE DE PARAMÈTRES PAR VALEUR
- INSTRUCTION RETOUR
- INSTRUCTION RETOUR EN
- INSTRUCTION RESULTAT
- PROCÉDURES BINAIRES
- PROCÉDURES EXTERNES
- PROCÉDURES ET CALBUR
- LE L.S.E. ET LA RÉCURSIVITÉ
- PARAMÈTRES NOMS DE PROCÉDURES

PROCÉDURES

V.1 INTRODUCTION

La notion de la sous-programme (terme qui englobe les procédures de type RETOUR et celles de type RESULTAT) est un des concepts les plus intéressants des langages modernes. Lorsqu'un problème complexe est posé, une des méthodes les plus efficaces pour le résoudre est de le décomposer en sous-problèmes de complexité moins grande. Ceci permet d'imaginer, de résoudre et de vérifier la solution d'un seul sous-problème à la fois. Ceci permet aussi le partage d'un travail par une équipe.

Cette méthode permet au niveau de l'écriture des algorithmes de les décomposer en sous-algorithmes, chacun plus facile à écrire et à tester que le programme entier.
On découpe le programme en composants accomplissant une action bien définie, particulièrement en ce qui concerne :
- les données d'entrée et de sortie
- les contraintes sur les données d'entrée
- le traitement des cas exceptionnels ou anormaux

Bien que cela ne soit pas obligatoire, il est de bonne pratique de ne pas utiliser dans un sous-programme de variables définies extérieurement, sans qu'elles soient passées en paramètres.

Les deux types de procédures L.S.E.

- celles qui font un certain travail mais qui ne retournent pas de valeur au programme appelant. Elles se terminent par RETOUR ou RETOUR EN. Elles s'utilisent dans un programme comme des instructions

Exemple :

```
12000 PROCEDURE &EFECR()
12005 AFFICHER[.12.];RETOUR
```


Cette procédure pourra s'utiliser, par exemple, de la façon suivante :

```
30 C←X-1;SI C=0 ALORS DEBUT &EFECR();AFF
ICHER 'C''est fini!';TERMINER FIN
```

- celles qui retournent une valeur au programme appelant. Elles se terminent par l'instruction RESULTAT. Elles interviennent dans le programme comme des expressions. On peut les considérer comme des fonctions écrites en L.S.E. Elles peuvent néanmoins en plus de la valeur qu'elles rendent, effectuer un certain travail

```
16300 PROCEDURE &DERCA(C);* Résultat: de
rnier caractère de la chaîne C
16310 RESULTAT SCH(C,LGR(C),1)
```

Ce qui pourra donner, à l'utilisation :

```
260 CAR←&DERCA(FRASE);SI CAR='.' ALORS .
....
```

V.2 INSTRUCTION PROCEDURE

Cette instruction constitue la déclaration d'une procédure interne L.S.E. Elle obéit à l'une des syntaxes suivantes :
PROCEDURE &NOM (liste1)
PROCEDURE &NOM (liste2) LOCAL liste2

Cette instruction doit se trouver en début de ligne.

NOM désigne le nom de la procédure. Tout nom respectant la syntaxe des noms L.S.E. peut être utilisé y compris les noms de fonctions et les mots réservés L.S.E. Ainsi, on peut à la fois utiliser la fonction LGR et la procédure &LGR.

liste1 désigne la liste des paramètres formels. Les éléments de la liste sont séparés par des virgules. Le nombre de paramètres est limité à 16 par le compilateur actuel. Ces paramètres formels sont des identificateurs (variables simples ou noms de tableaux) ou des noms de procédure. Chacun d'eux correspond au paramètre de même rang de l'instruction d'appel. Lors de l'appel, il prend la valeur et le type de celui-ci.

Tout élément de liste2 qui n'est pas dans liste1 est une variable locale (voir paragraphe suivant). Tout élement commun aux deux listes est un paramètre formel par valeur (voir plus loin). liste2 ne peut contenir plus de 16 éléments

V.3 VARIABLES LOCALES, VARIABLES GLOBALES

Dans les communications entre le programme appelant et la procédure appelée, il est possible, mais non souhaitable, d utiliser des variables communes non transmises en paramètres; on les désigne sous le nom de variables globales.

Exemple :

Considérons une procédure &AFF destinée à afficher une chaine en insérant un espace entre chaque caractère :

```
1 * Inconvénient des variables globales
5 CHRINE CH
10 RFFICHER 'Chaîne ? ';LIRE CH
20 &RFF(CH);TERMINER
30
1000 PROCEDURE &RFF(C);RFFICHER [/]
1010 FRIRE 1020 POUR I←1 JUSQUR LGR(C)-1
1020 RFFICHER [U,X]SCH(C,I,1)
1030 RFFICHER [U,X]SCH(C,I,1);RETOUR

EXECUTER R PRRTIR DE 1

Chaîne ? Bonjour Madame
B o n j o u r   M a d a m e
TERMINE EN LIGNE 20
```


Nous pouvons avoir besoin d'utiliser la même procédure cette fois avec les éléments d'un tableau de chaînes

```
1 * Inconvénient des variables globales
5 TRBLERU CHRINE T[3]
10 RFFICHER 'Tapez 3 chaînes:'
20 LIRE T
30 FRIRE 30 POUR I←1 JUSQUR 3;&RFF(T[I])
40 TERMINER
1000 PROCEDURE &RFF(C);RFFICHER [/]
1010 FRIRE 1020 POUR I←1 JUSQUR LGR(C)-1
1020 RFFICHER [U,X]SCH(C,I,1)
1030 RFFICHER [U,X]SCH(C,I,1);RETOUR

EXECUTER R PRRTIR DE 1

Tapez 3 chaînes:
Bonjour Mesdames
Bonjour Mesdemoiselles
Bonjour Messieurs


B o n j o u r   M e s d a m e s
TERMINE EN LIGNE 40
```

Que s'est-il passé ? Pourquoi n'avons nous pas eu l'affichage des 3 chaînes du tableau T? Demandons

```
?I
17
```

La variable I est utilisée en deux endroits du programme : une fois en ligne 30 pour la gestion de la boucle parcourant les trois éléments du tableau T, une autre fois dans la procédure &AFF, en ligne 1010 pour extraire les uns après les autres les caractères de la chaîne C. Cette seconde utilisation a changé la valeur de I et donc perturbé le bon fonctionnement du programme.

Le premier remède qui vient à l'esprit consiste à remplacer l'un des deux I par un autre nom de variable, par exemple J pour la boucle qui est dans la procédure. Correction faite ça marche très bien. Pourtant ce n'est pas une bonne méthode car elle impose lors de l'écriture de la procédure de connaître le nom des variables des programmes appelant ce qui ne facilite guère la réalisation de procédures indépendamment du contexte de leur utilisation.

Une autre solution consiste à utiliser dans la procédure une variable locale et donc à modifier la ligne 1000 comme suit :

```
LISTER LIGNES 1000
1000 PROCEDURE &AFF(C);AFFICHER [/]

/;/ LOCAL I;
1000 PROCEDURE &AFF(C) LOCAL I;AFFICHER
[/]
```

Ainsi dans le programme appelant, I représente la variable qui contrôle la boucle d'extraction des éléments du tableau T, et dans la procédure I désigne la variable qui contrôle l'affichage des caractères successifs de la chaîne Les deux variables tout en ayant le même nom ne représentent pas la même chose. A un instant donné, on n'a accès qu'à l une d'entre elles. Noter en conséquence, que le fait de déclarer locale une variable, interdit l'usage d'une variable globale de même nom.

Essayons notre programme ainsi modifié :

```
EXECUTER A PARTIR DE 1

Tapez 3 chaînes:
Bonjour Mesdames
Bonjour Mesdemoiselles
Bonjour Messieurs


B o n j o u r   M e s d a m e s
B o n j o u r   M e s d e m o i s e l l
e s
B o n j o u r   M e s s i e u r s
TERMINE EN LIGNE 40
```

Ça marche !!!
Une variable locale à une procédure est donc connue :
- dans la procédure où elle est définie
- dans les procédures qu'elle appelle à condition dans ce cas qu elle ne soit pas elle même declaree locale dans la procédure appelee.

Une bonne manière de traiter la question consiste, dans une procédure, a déclarer locale toute variable qui n'est pas un paramètre.
Si les variables locales représentent des chaînes, des formes, des booléens ou des tableaux de type quelconque, elles devront être déclarées comme telles à l'intérieur de la procédure.
Lors du traitement de la fin de la procédure par les instructions RETOUR. RETOUR EN ou RESULTAT, les variables locales de la procédure sont libérees et leurs valeurs sont perdues.

Notons que les motifs, qui ne sont pas des variables, ont un statut global. Dès qu'un motif est défini, il est connu dans tout le programme mais aussi dans les procédures externes appelées par le programme (voir chapitre sur le graphique)

V.4 PASSAGE DE PARAMETRES PAR NOM (ou adresse)

Les paramètres qui se trouvent dans l'instruction d appel sont les paramètres effectifs de l'appel. Ceux qui se trouvent dans la déclaration sont les parametres formels de la déclaration.

Dans le cadre de notre exemple précèdent nous transmettons a la procédure le nom du paramètre effectif. Tout ce passe comme si le paramètre effectif (CH ou T [I]) s'appelait C à l intérieur de la procédure. Au retour de cette procédure les noms sont changés en sens inverse ce qui permet de récupérer dans les paramètres effectifs les résultats des traitements éventuels effectués dans la procédure. Cela signifie en particulier que si la procédure modifié le paramètre formel le paramètre effectif sera également modifié

Exemple :

```
1
10 R←3
20 AFFICHER 'Avant l''appel, R vaut ',R
30 &TRUC(R)
40 AFFICHER 'Au retour de la procédure,
R vaut ',R
50 TERMINER
300 PROCEDURE &TRUC(X)
310 AFFICHER 'Le paramètre transmis vaut
 ',X
320 X←2*X
330 AFFICHER 'Son double est donc ',X
340 RETOUR

exECUTER A PARTIR DE 1

Avant l'appel, R vaut 3
Le paramètre transmis vaut 3
Son double est donc 6
Au retour de la procédure, R vaut 6
TERMINE EN LIGNE 50
```

Le parametre est transmis par nom (on dit aussi par adresse) Le paramètre formel X est utilise dans la procédure à la place de la variable A. Il ne sera pas possible dans la procédure d'acceder à A en la traitant comme une variable globale. on ne pourra le faire que par l'intermédiaire de X Par contre si c'est nécessaire. on pourra déclarer une variable locale de nom A qui n'aura de commun avec la variable de même nom du programme appelant De même, si le programme appelant utilisait une variable X, cette dernière ne serait pas accessible dans la procédure

Tapons :

```
&truc(7)
ERREUR E 15 EN LIGNE 300 APPEL LIGNE 0UR
```

En effet, 7 est une constante n'a pas de nom et ne peut donc pas être transmis à la procédure

Lors d'un appel de procédure par nom, les paramètres effectifs peuvent être :
- des variables simples
- des variables indicées (éléments de tableau numérique, booléen, chaine ou forme)
- des noms de tableaux
- des noms de procédures
- Il ne peuvent être ni des constantes, ni des expressions, ni des noms de motifs.

V.5 PASSAGE DE PARAMÈTRES PAR VALEUR

Modifions notre procédure &TRUC :

```
LISTER LIGNES 300
300 PROCEDURE &TRUC(X)

/ LOCAL X
300 PROCEDURE &TRUC(X) LOCAL X



&TRUC(7)
Le paramètre transmis vaut 7
Son double est donc 14




EXECUTER A PARTIR DE 1

Avant l'appel, R vaut 3
Le paramètre transmis vaut 3
Son double est donc 6
Au retour de la procédure, R vaut 3
TERMINE EN LIGNE 50
```

La modification de la ligne 300 provoque le passage du paramètre par valeur. Ce passage se fait en recopiant dans les paramètres formels les valeurs correspondant aux paramètres effectifs

Au retour les valeurs contenues dans les variables correspondant aux paramètres formels sont perdues et les variables retrouvent le rôle qu'elles avaient avant l'appel
Lors d'un appel de procédure certains parametres peuvent être transmis par nom et d'autres par valeur
Lors d'un passage par valeur les parametres effectifs peuvent être :
- des expressions (et en particulier des variables)
- des noms de tableau

Il ne peuvent être des noms de motifs.

Puisqu'il y a recopie de la valeur du parametre effectif dans le paramètre formel cette transmission est coûteuse en place mémoire.
Noter que dans un appel du type &PROC (X) on ne saura de quelle manière se fait la transmission qu'en regardant la ligne de déclaration de procédure. Si la transmission se fait par nom, au retour la variable X aura pu être modifiée. si la transmission se fait par valeur elle sera restituée sans changement

V.6 INSTRUCTION RETOUR

Cette instruction termine les procédures qui sont utilisées comme instructions. C'est elle qui rend le contrôle au programme appelant (à l'instruction qui suit l'instruction d'appel).

Exemple :

t0 A←5; &AFFIC (A); A←10, AFFICHER A; TERMINER
100 PROCEDURE &AFFIC (A); AFFICHER A; RETOUR
Le déroulement du programme principal se fait de la manière suivante
- A prend la valeur 5
- la procédure AFFIC affiche 5 sur l'écran
- A prend la valeur 10
- la valeur 10 est affichée
- le programme est terminé.

V.7 INSTRUCTION RETOUR EN

Cette instruction permet de forcer le retour à un endroit donné du programme

Exemple :

```
1
10 AFFICHER 'Ce programme montre une uti
lisation de  RETOUR EN'
20 &BRAN(@10,@900)
30 AFFICHER 'Cette instruction permet de
 préciser un numéro de ligne de RETOUR'
40 &BRAN(@10,@900)
900 AFFICHER 'C''est fini !';TERMINER
10000 PROCEDURE &BRAN(DEB,FINI) LOCAL FI
NI,DEB,C
10005 CHAINE C
10010 AFFICHER 'Tapez "S" pour passer à
la suite,'
10020 AFFICHER 'ou "D" pour reprendre au
 début,'
10030 AFFICHER 'ou "T" pour terminer.',1
0 13.
10040 LIRE C;C←TMR(C)
10050 SI C='S' ALORS RETOUR
10060 SI C='T' ALORS RETOUR EN FINI SINO
N SI C='D' ALORS RETOUR EN DEB SINON ALL
ER EN 10040
```

```
exECUTER A PARTIR DE 1

Ce programme montre une utilisation de
RETOUR EN
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
S
Cette instruction permet de préciser un
numéro de ligne de RETOUR
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
d
Ce programme montre une utilisation de
RETOUR EN
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
s
Cette instruction permet de préciser un
numéro de ligne de RETOUR
Tapez "S" pour passer à la suite,
ou "D" pour reprendre au début,
ou "T" pour terminer.
s
C'est fini !
TERMINE EN LIGNE 900
```

V.8 INSTRUCTIONS RÉSULTAT

RESULTAT ex
ex est une expression qui sera rendue au programme appelant et qu'il pourra afficher, affecter à une variable ou utiliser dans un calcul.
En dehors de cette valeur rendue, l'instruction RESULTAT réalise les mêmes opérations que l'instruction RETOUR.

Exemple :

En L.S.E. il n'existe pas de fonction permettant d'explorer une chaîne de la droite vers la gauche : on peut enrichir ses possibilités à l'aide de la procédure suivante qui extrait les derniers caractères d'une chaîne :

```
1000 PROCEDURE &FCH(C,N) LOCAL N,C
1010 RESULTAT SCH(C,LGR(C)-N+1,'')



?&FCH('Bonjour Madame',5)
adame
```

V.9 PROCÉDURES BINAIRES

On désigne par ce terme des procédures en langage machine que l'on utilise comme les procédures L.S.E.
La manière d'écrire de telles procédures est décrite dans l'Annexe V.

Une procédure binaire est désignée par son nom qui respecte la syntaxe des noms L.S.E. S'il s'agit d'une procédure fonction, seul le nom la distingue des fonctions du L.S.E.

V.10 PROCÉDURES EXTERNES

Une procédure est dite interne si elle est déclarée dans le programme. Dans le cas contraire, elle est considérée par le L.S.E. comme étant externe : elle est recherchée comme telle sur le disque de travail puis lancée. Ceci concerne aussi bien les procédures L.S.E que les procédures binaires. Une erreur est détectée si L.S.E. ne la trouve pas.

Le temps de chargement d'une procédure externe est loin d'être négligeable. Afin d'optimiser ce temps, il a été décidé de conserver en mémoire une procédure externe inactive dans le cas où l'on ne manque pas de place.
Lors d'un appel de procédure L.S.E., l'interpréteur :
– recherche la procédure dans les procédures internes L.S.E ,

– s'il ne la trouve pas, il poursuit la recherche dans les procédures externes L.S.E déjà chargées,
– s'il ne la trouve pas, il la recherche dans les procédures L.S.E. du disque de travail Trois cas peuvent se présenter :
  – la procédure n'existe pas sur le disque, il y a dans ce cas ERREUR E t8
  – la procédure existe et il y a en mémoire assez de place pour la charger et la lancer, ce qui est fait.
  – la procédure existe mais il n'y a pas assez de place, alors il est décidé de supprimer les procédures externes chargées et inactives jusqu'à ce qu'il y ait assez de place. Deux cas se présentent :
    – l'opération est possible, ce qui est fait.
    – l'opération s'avére impossible aprés suppression de toutes les procédures externes inactives, alors il y a ERREUR E 3.

Cette suppression éventuelle commence par les procédures inactives chargées depuis le plus de temps. Cela permet par exemple de se débarasser d'une procédure d'initialisation qui ne sert plus.

Le fichier contenant une procédure externe doit :
– avoir pour nom de la procédure externe L.S.E.
– contenir une procédure L.S.E. ayant ce méme nom.

En dehors de cela, le fichier peut contenir un véritable programme L.S E. et pouvoir être utilisé indépendamment de l'usage en procédure externe. Cependant, il est préférable, pour des raisons d'encombrement mémoire et de portabilité par rapport à d'autres implémentations du L.S.E., de ne pas mettre dans ce fichier autre chose que la procédure et de déclarer celle-ci en ligne 1.

Notons de plus que toutes les variables autres que les paramètres d'une procédure externe sont systématiquement des variables locales et qu'une telle procédure ne peut donc avoir accès aux variables du programme appelant

V.11 LES PROCEDURES ET LE CALBUR

Il n'est pas possible de déclarer des procédures en CALBUR.

L'utilisation de procédures en CALBUR est possible, qu'il s'agisse de procédures L.S.E. ou de procédures binaires. qu il s'agisse de procédures internes ou externes.

Néanmoins la rencontre dans une procédure L.S.E. de certaines instructions interdites en CALBUR provoquera une erreur d'exécution. Cela concerne par

exemple l'instruction PAUSE, qui, si elle était autorisée, conduirait à un dilemme lors de la commande COntinuer. D'autres instructions, telles ALLER EN ne sont pas concernées par cette restriction.

Lors d'une erreur d'exécution dans une procédure utilisée en CALBUR la pile d'exécution est remise dans l'état où elle était avant l'appel. L'exécution d'une procédure en mode CALBUR ne permet donc pas sa mise au point.

V.12 LE L.S.E. ET LA RÉCURSIVITÉ

La récursivité simple est la possibilité de réaliser des procédures qui s'appelent elles-mêmes. La récursivité croisée concerne des procédures s'appelant les unes les autres de façon qu'indirectement la procédure appelante soit appelée par une autre.

Le L.S.E. permet la récursivité simple et croisée sans aucune limitation (si ce n'est celle de la place mémoire).

Voici 2 exemples d'utilisation du L.S.E. en récursivité simple :

```
1 * Calcul de n! (Factorielle du nombre
n: produit des entiers inférieurs ou éga
ux à n).
10 AFFICHER 'Factorielle de ? ';LIRE N;S
I N<1 OU ENT(N)≠N ALORS ALLER EN 10
20 AFFICHER &FACT(N);TERMINER
30
1000 PROCEDURE &FACT(X) LOCAL X
1010 RESULTAT SI X=1 ALORS 1 SINON X*&FA
CT(X-1)

exECUTER A PARTIR DE 1

Factorielle de ? 5
120
TERMINE EN LIGNE 20
```

```
1 * Tours de Hanoï
10 AFFICHER ['Tour de départ numéro 1',/
,'Tour d''arrivée numéro 3',/]
20 LIRE ['Combien de disques à déplacer
? ',U,/]N;SI N<1 OU ENT(N)≠N ALORS ALLER
 EN 20
30 &HANOI(1,3,N);TERMINER
40
1000 PROCEDURE &HANOI(DEP,ARR,NBRE) LOCA
L NBRE,ARR,DEP
1010 SI NBRE=1 ALORS DEBUT &DEPL(DEP,ARR
);RETOUR FIN
1020 &HANOI(DEP,6-DEP-ARR,NBRE-1);&DEPL(
DEP,ARR);&HANOI(6-DEP-ARR,ARR,NBRE-1)
1030 RETOUR
1040
2000 PROCEDURE &DEPL(D,A) LOCAL A,D
2010 AFFICHER 'Déplacez un disque de ',D
,' en ',A;RETOUR

exECUTER A PARTIR DE 1
Tour de départ numéro 1
Tour d'arrivée numéro 3
Combien de disques à déplacer ? 3

Déplacez un disque de 1  en 3
Déplacez un disque de 1  en 2
Déplacez un disque de 3  en 2
Déplacez un disque de 1  en 3
Déplacez un disque de 2  en 1
Déplacez un disque de 2  en 3
Déplacez un disque de 1  en 3
TERMINE EN LIGNE 30
```


V.13 PARAMÈTRES NOMS DE PROCÉDURE

Il peut être intéressant de réaliser une procédure pouvant prendre en compte, indifféremment, le traitement effectué par d'autres procédures : On passe à cette procédure, le nom d'une procédure à prendre en compte dans le traitement. Voici un exemple d'un tel programme :

```
1
10 &REPET(&BONJ,3)
20 &REPET(&AURE,5)
30 TERMINER
40
100 PROCEDURE &REPET(&P,X) LOCAL X,I
110 FAIRE 110 POUR I←1 JUSQUA X;&P()
120 RETOUR
130
140 PROCEDURE &BONJ();AFFICHER 'Bonjour'
;RETOUR
150
160 PROCEDURE &AURE();AFFICHER 'Au revoi
r';RETOUR

exECUTER A PARTIR DE 1

Bonjour
Bonjour
Bonjour
Au revoir
Au revoir
Au revoir
Au revoir
Au revoir
TERMINE EN LIGNE 30
```

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).
La 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 execution.

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 B00;B00←.VRAI.
40 E←2.7182818;PI←3.1415927
50 GARER E,3,'FICH';GARER T,8,'FICH';GAR
ER SALUT,9,'FICH'
60 GARER B00,21,'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 :

aprés avoir tapé un programme
SOrtie NOMPRO
LIster lignes ✳
STandard (ou SOrtie .t0)

Place le source du programme dans un fichier NOMPRO LST.

VI-9 COMMANDE PErsévérer

Cette commande n'a pas de paramétre.

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 géneres 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. La 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)
```

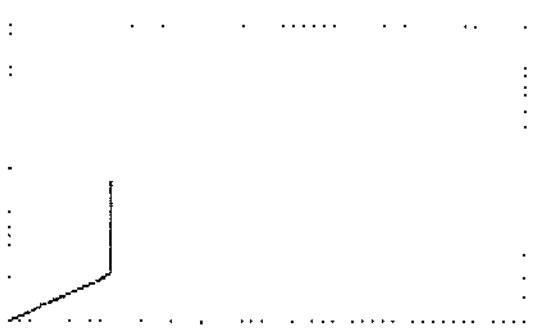
Tout comme les autres types de variable L.S.E. une forme peut être garée en fichier, chargée depuis un fichier et passée en paramètre à une procédure

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
```

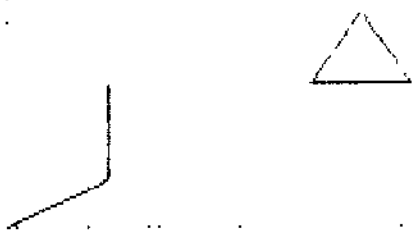


Puis :

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 extremité. 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
```

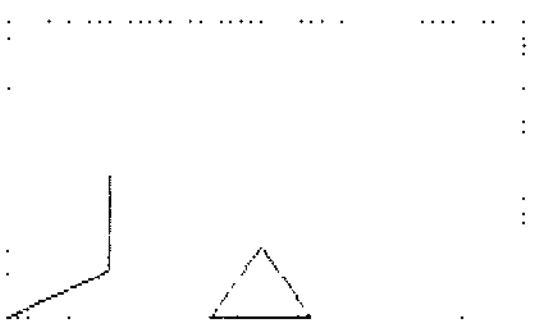
Lorsque nous juxtaposons des formes, le vecteur équivalent du résultat de la juxtaposition est la somme des vecteurs équivalents des formes. On peut remarquer que VEQ(F;G) = VEQ(G;F) alors, qu'en général, les formes F;G et G;F sont distinctes (et produisent des « dessins » différents).

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

Exemples :


```
?VEC('A',200,100)%VEC('A',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).

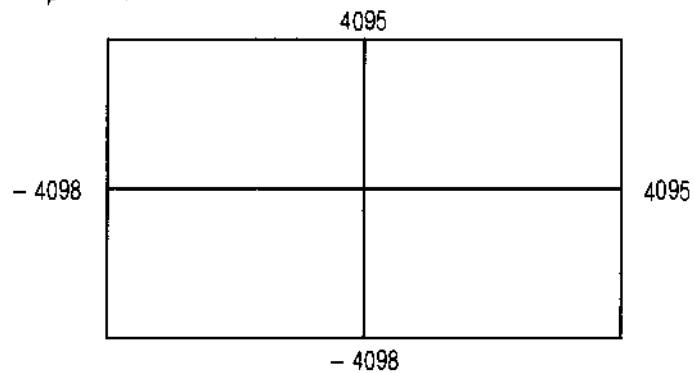
Exemples :

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

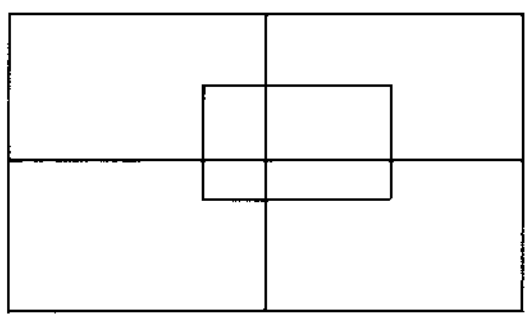
Les formes sont définies, nous l'avons vu, à partir des éléments d'un plan vectoriel en théorie illimité. En réalité, pour des raisons d'encombrement mémoire et de rapidité, le L.S.E.G.-E.D.L. limite les coordonnées à des valeurs absolues inférieures à 4096. De même la plus petite coordonnée non nulle 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 respectees
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 depart 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
```

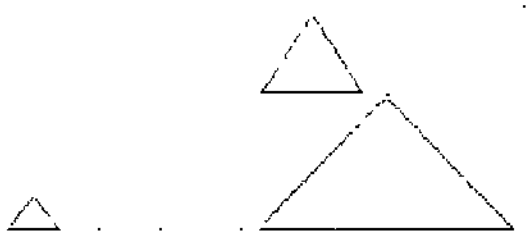
VII-2.3 Marge

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 numero de page graphique (0 sur T07 et M05) et des coordonnées du coin inferieur gauche et du coin supérieur droit de la « fenêtre », ces coordonnées étant définies par rapport aux dimensions du cadre standard.

Exemple :

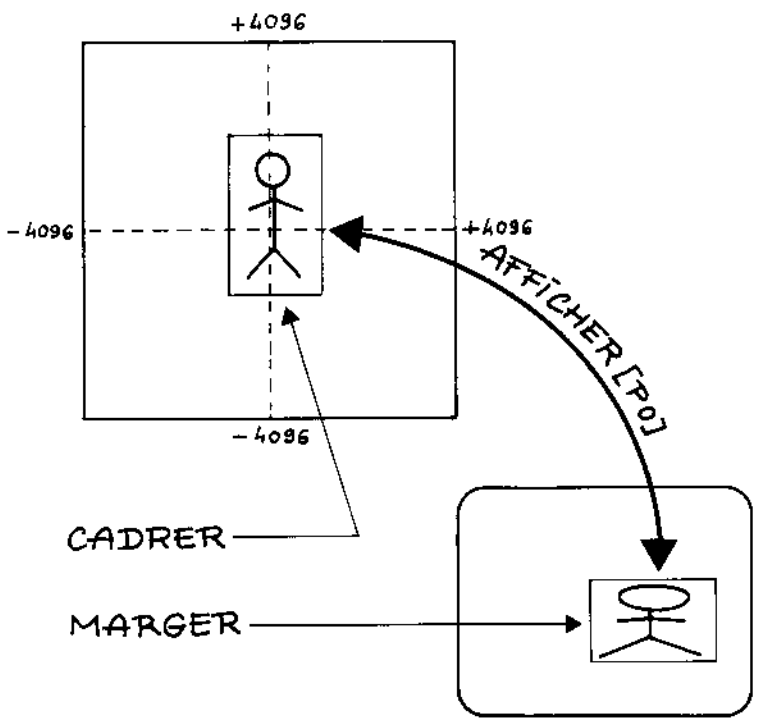
```
I
10 FORME TRGLE;NETTOYER 0
20 TRGLE←VEC('A',200,0):VEC('A',-100,173
):VEC('A',-100,-173)
30 MARGER 0,0,0,500,310;AFFICHER [P0]TRG
LE
40 CADRER 0,0,500,300
50 MARGER 0,500,310,1000,620;AFFICHER [P
0]TRGLE
60 CADRER 0,0,200,175
70 MARGER 0,500,0,1000,310;AFFICHER [P0]
TRGLE
80 ALLER EN 80

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 t (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 :

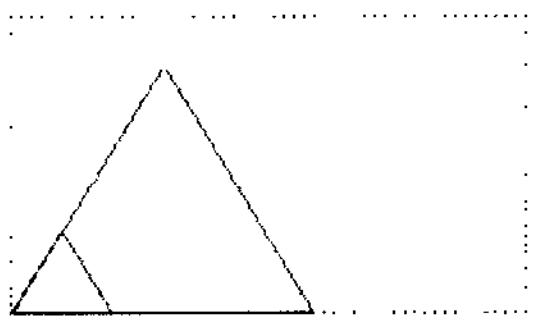
```
FORME F;F ← TRA(VEC('A',300,300),'R1'):VEC('A',200,200)
TRACE 'B1';?F
affiche le premier vecteur de F en rouge, le second en bleu;
TRACE 'RVt';?F
affiche le premier vecteur de F en rouge, le second en jaune.
```

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).

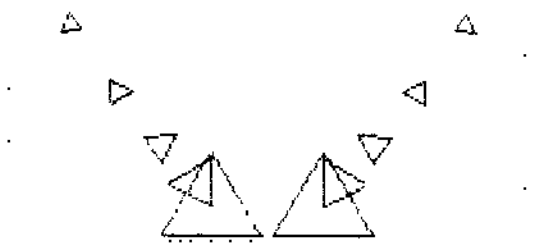
Exemples :

```
FORME T ; T ← VEC('A',200,0):VEC('A',–100,t73):VEC('A',–100,–173)
?T,HOM(T,3)
```



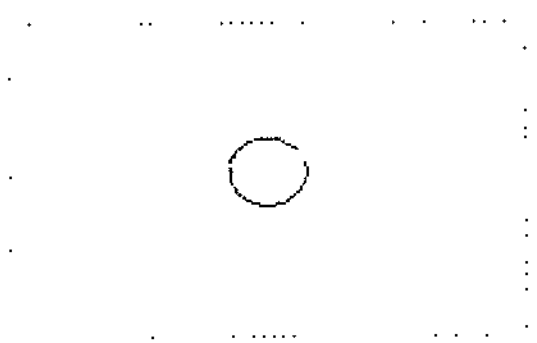

```
1 * TRIANGLES
10 FORME F,T;F←VEC('E',-90,-20)
20 T←VEC('A',200,0):VEC('A',-100,173):VE
C('A',-100,-173)
30 FAIRE 40 POUR I←1 JUSQUA 5
40 F←F:TRN(ROT(HOM(T,1/I),(I-1)/2),100,3
0*I)
50 F←F%SM1(F)
60 CADRER -512,0,512,700;AFFICHER [U]F
70 ALLER EN 70;* appuyer sur STOP pour a
rr@ter
```

```
EXECUTER R 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:RDT(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 operateurs 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 probleme en utilisant la notion de MOTIF. Nous allons, a 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 a 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 interêt à définir également des motifs ROUE, FENET, .. etc.

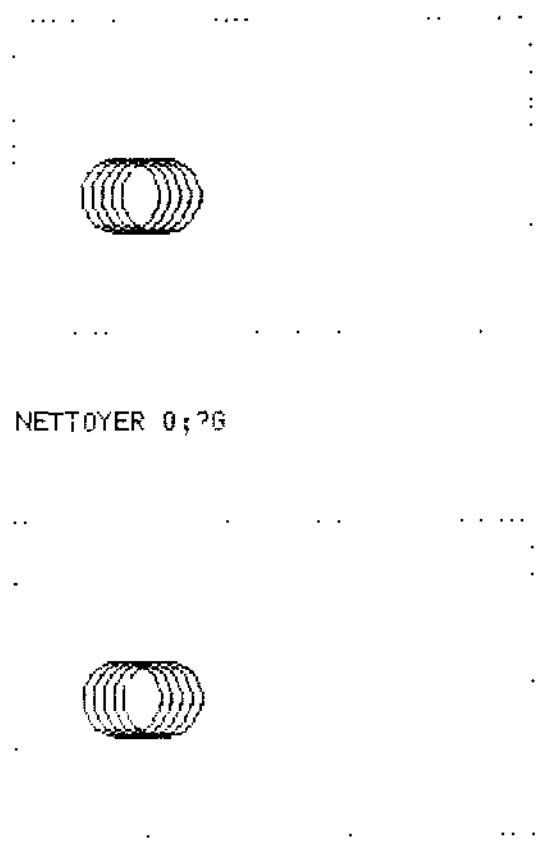
Mettons en évidence le gain de place grâce à l'exemple suivant

```
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
```

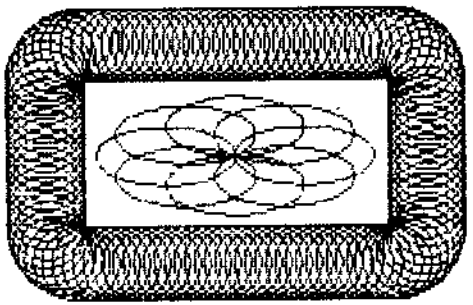


```
NETTOYER 0;?G
```


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.]EOC(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 chaînes de valeurs respectives (ec), (ec1) ...

eg, eg1, ... designeront 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 t25 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 recupère 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 » apparaît

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 modifier 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 numerotees à partir de 0 et qui peuvent être utilisees par exemple dans des instructions LIRE ou AFFICHER

Les affectations standard sont en entree
0 au clavier
1 au clavier sans écho
2 a la voie V24

– Les voies physiques qui sont de deux sortes .
– celles qui correspondent a des périphériques et qui sont symbolisées par
10 pour clavier ou ecran
.20 pour imprimante ou clavier sans écho
30 pour voie V24
40 pour inhiber l entree ou la sortie
– celles qui correspondent à des noms de fichiers qui sont symbolisees 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 reférence à un numéro de ligne. Ainsi sont modifiees 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 renumérote

Pour pouvoir atteindre les nombres qui représentent des numeros 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 numero 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 nt

• EXécuter à partir de nt A n2
Dans les deux cas, le programme est lancé à partir de la premiere 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 paramêtre, 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 utilisee 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 paramêtre, 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 t20 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 è 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 a 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 numerotés a partir de 0 et qui peuvent être utilisées par exemple dans des instructions LIRE ou AFFICHER

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

– Les voies physiques sont de deux sortes :
– celles qui correspondent à des périphériques et qui sont symbolisees 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 symbolisees 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 chaines est l'opérateur de concaténation noté ! (point d'exclamation).
Il permet de réunir 2 chaines en une seule : les 2 chaines 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 booleenne l ordre des priorites est le suivant
NON
ET
OU ou DIJ

On utilisera des parenthèses si l'on veut modifier l ordre des priorites

OPÉRATEURS DE COMPARAISON

Ces operateurs 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 booleennes évaluées de la manière suivante :
On compare les 2 chaînes caractère par caractere (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 operateurs 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 repere 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 isocéle. Le vecteur équivalent est nul

La superposition . (symbole %)

Evaluation 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 remplacee par le point.
les puissances de 10 sont représentées a l'aide de la lettre E.

2. Les constantes chaines.

Une constante chaine est une suite de caractères. Ces caracteres peuvent être des lettres, des chiffres, des caracteres 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 a une déclaration préalable au moyen des instructions respectives . CHAINE, BOOLEEN, FORME

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

Les variables non déclarees sont considerées 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 evaluée, puis sa valeur est affectée à la variable. En cas de mélange de types une erreur d'exécution est detectée.

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 resultat 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 booleenne.

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 utilise dans un programme, il est possible de récupérer la place occupée en mémoire par ces objets. Le 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 ou 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 * Périmétre 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 lon-
gueur 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 aleatoire
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 specifications de mise en page

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

Chacune de ces spécifications peut être précedee d'un facteur de repetition 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 correspon dre, 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 decimal
- le point décimal éventuel
- les décimales necessaires
- un espace

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

Exemples :

```
? — 1155
— 1155
?0 095
0.095
?0 0005
5E—4
?123456789
1 23456769E8
```

b) La spécification F (syntaxe . Fn1 n2) :
Cette specification se compose :
– de la lettre F
– du nombre n1 de caractères (chiffres, signe ou espaces) avant le point decimal,
– 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 decimal sont remplaces par des espaces ; si après le point décimal il y a des chiffres superflus, un arrondi est effectue ; 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 respecte).

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'afliche :
236 0000

TERMINE EN LIGNE 25
```


c) La specification E (syntaxe En1.n2) .

n1 et n2 ont le même rôle que dans la specification F
Ce format est utilise pour représenter des nombres très grands ou tres 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'atfiche
 2.1998E4
8 8212636E6 s'atfiche
 8.8213E6
TERMINE EN LIGNE 25
```

– Pour les expressions chaînes, booléennes ou graphiques, seule est autorisee la specification universelle U
La variable est alors affichee 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 a 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)

Si 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

III. INSTRUCTIONS DE STRUCTURATION

- FAIRE .... JUSQUA ...
- FAIRE ... TANT QUE...
- PROCEDURE
- RETOUR
- RETOUR EN
- RESULTAT
- PROCÉDURES EXTERNES
- PROCÉDURES BINAIRES
- RÉCURSIVITE
- PROCÈDURES EN CALBUR

INSTRUCTION FAIRE .... JUSQUA ...

Syntaxe : N1 FAIRE N2 POUR I ← ea1 JUSQUA ea2
ou N1 FAIRE N2 POUR I ← ea1 PAS ea3 JUSQUA ea2

N1 et N2 sont des numeros de ligne (N2 ⩾ N1)
I est une variable numerique simple appelée variable de contrôle de la boucle.
(ea1) valeur initiale
(ea2) : valeur finale
(ea3) : pas (nombre positif, négatif ou nul, entier ou non) ; par défaut (ea3) est egal a 1

Au premier passage à la ligne N1 la variable I prend la valeur (ea1) Le pas et la valeur finale sont alors évalues :
SI
– la valeur de I est supérieure à (ea2) et le pas positif
ou
– la valeur de I est égale à (ea2) et le pas nul
ou
– la valeur de I est inférieure à (ea2) et le pas negatif
ALORS
on sort de la boucle et l'exécution se poursuit a la première ligne dont le numero est superieur a N2
SINON
le traitement situé entre les lignes N1 et N2 (incluses) est executé. On revient a la ligne N1 pour réévaluer le pas qui est alors ajouté à I, et l'on recommence ..

Remarque : La ligne N2 peut ne pas exister la boucle prend alors en compte la ligne existante la plus proche dont le numéro est inferieur a N2

INSTRUCTION FAIRE ... TANT QUE ...

Syntaxe : N1 FAIRE N2 TANT QUE eb
ou N1 FAIRE N2 POUR I ← ea1 TANT QUE eb
ou N1 FAIRE N2 POUR I ← ea1 PAS ea2 TANT QUE eb
N1 et N2 sont des numéros de ligne (N2 ⩾ N1)

Au premier passage à la ligne N1 l'expression booléenne eb est évaluee :
SI
eb a la valeur . FAUX . on sort de la boucle et l'execution se poursuit à partir de la premiere ligne dont le numéro est superieur à N2.
SINON
le traitement situe entre les lignes N1 et N2 est exécute, puis on revient a la ligne N1 pour réévaluer l'expression eb, et l'on recommence.

INSTRUCTION PROCEDURE

Il existe en L.S.E. deux types de procédures :
– Les procédures de type « sous programme » qui se terminent par RETOUR ou RETOUR EN. Elles s'utilisent dans un programme comme des instructions.
– Les procédures de type « fonction » qui retournent une valeur au programme appelant. Elles se terminent par l'instruction RESULTAT. Elles interviennent dans le programme comme des expressions. On peut les considérer comme des fonctions écrites en L.S.E.
L'utilisation des procédures comporte deux phases :
a) La déclaration de la procédure
b) L'appel de la procédure
Nous décrivons ici le point a), et nous renvoyons au manuel utilisateur pour le b).

L'instruction PROCEDURE est l'instruction de déclaration d'une procédure interne L.S.E. Elle obéit à l'une des syntaxes suivantes

PROCEDURE &NOM(liste1)
PROCEDURE &NOM(liste 1) LOCAL liste2

Cette instruction doit se trouver en debut de ligne, et ne peut être executée.
NOM désigne le nom de la procédure Tout nom respectant la syntaxe des noms L.S.E. peut être utilisé y compris les noms de fonctions et les mots réservés L.S.E

liste1 désigne la liste des paramètres formels. Les éléments de la liste sont séparés par des virgules. Le nombre de paramètres est limité à 16 par le compilateur actuel. Ces paramètres formels sont des identificateurs, ou des noms de procédure. Chacun d'eux correspond au paramètre de même rang de l'instruction d'appel. Lors de l'appel, il prend la valeur et le type de celui-ci.

OPTION LOCAL

Syntaxe : LOCAL liste2

liste2 est une suite de noms de variables de type quelconque separés par des virgules.
Cette instruction ne peut apparaitre que derrière la déclaration d'une procédure.

Tout elément de liste2 qui n'est pas dans liste1 est une variable locale. Tout élément commun aux deux listes est un paramètre formel transmis par valeur.
liste2 ne peut contenir plus de 16 éléments
Toute variable non déclarée LOCAL dans une procedure interne est considérée comme globale.

INSTRUCTION RETOUR

Cette instruction termine les procédures sous programme et rend le contrôle au programme appelant, au niveau de l'instruction qui suit l'instruction d'appel.

INSTRUCTION RETOUR EN

1re forme : RETOUR EN ea

(ea) représente le numéro de la ligne de retour Cette instruction rend la main au programme appelant au début de la ligne de numéro (ea). Elle réalise à la fin de son execution un véritable ALLER EN. Son usage, comme celui du ALLER EN doit être strictement limité à des traitements particuliers.

Autres formes : RETOUR EN ea, ✳
RETOUR EN ea, eat

(ea) est toujours le numéro de la ligne de retour. ✳ signifie que le programme appelant reprend la main au niveau d'appel 0.
eat indique que le niveau d'appel est diminué de (ea1) unités (RETOUR EN ea,t est équivalent à RETOUR EN ea)

INSTRUCTION RESULTAT

Syntaxe : RESULTAT exp

exp est une expression dont la valeur sera rendue au programme appelant.
En dehors de cette valeur rendue, l'instruction RESULTAT réalise les mêmes opérations que l'instruction RETOUR.

PROCÉDURES EXTERNES

Une procédure est dite interne si elle est déclarée dans le programme Dans le cas contraire elle est considérée par le L S E. comme étant externe et donc recherchée comme telle sur le disque de travail puis lancée. Ceci concerne aussi bien les procédures L.S E que les procédures binaires Une erreur est détectée si le L.S.E. ne la trouve pas.

Le temps de chargement d'une procedure externe L.S E. est loin d'être négligeable. Afin d optimiser ce temps, il a été décidé de conserver en mémoire une procédure externe inactive dans le cas ou l'on ne manque pas de place.
Lors d un appel de procédure L.S.E., l interpréteur :
– recherche la procédure parmi les procédures internes L S.E.,
– s'il ne la trouve pas, poursuit la recherche parmi les procédures externes L.S.E déjà chargées.
– s'il ne la trouve pas, la recherche parmi les procédures L S.E. du disque de travail. Trois cas peuvent se présenter :
– la procédure n'existe pas sur le disque : il y a dans ce cas ERREUR E 18
– la procedure existe et il y a en mémoire assez de place pour la charger et la lancer : ce qui est fait
– la procédure existe mais il n'y a pas assez de place, alors il est décidé de supprimer les procedures externes chargées et inactives jusqu à ce qu'il y ait assez de place. Deux cas se présentent.
a) l opération est possible, ce qui est fait,
b) l'operation s'avère impossible après suppression de toutes les procédures externes inactives, alors il y a ERREUR E 3.

Cette suppression éventuelle commence par les procédures inactives chargées depuis le plus longtemps

Le fichier contenant une procédure externe doit :
– avoir pour nom le nom de la procédure externe L.S E
– contenir une procédure L.S.E ayant ce même nom.

Le fichier peut contenir un véritable programme L S.E. et être utilisé indépendamment de l usage comme procédure externe

PROCÉDURES BINAIRES

On désigne par ce terme des sous-programmes écrits en langage machine de type RETOUR ou RESULTAT et que l'on utilise comme les procédures L S.E.
La manière d écrire ces procédures fait l'objet de l'Annexe V.
Une procédure binaire est appelée par son nom, qui doit respecter la syntaxe des noms L.S.E. S'il s'agit d'une procédure fonction, seul le nom la distingue des fonctions du L.S.E.

Exemples :

10 A ← MIN (B,C) est un appel de procédure binaire fonction
20 CADRE (10,20,10,0) est un appel de procédure binaire de type RETOUR.

Remarque : La différence entre l'appel d'une procédure écrite en L S.E. et l'appel d'une procédure binaire est l'absence de & avant le nom.

INSTRUCTION PROCEDURE BINAIRE

Cette instruction est l'instruction de déclaration de procédure binaire. Elle obéit à la syntaxe suivante :

10 PROCEDURE BINAIRE NOM

Cette instruction doit être la seule instruction de la ligne, les commentaires sont interdits en fin de ligne.
NOM représente le nom d'un fichier de type bin, situé sur le disque de travail. Le compilateur recevant une telle ligne, va chercher sur le disque de travail le fichier correspondant à ce nom et l'incorpore au programme en cours de compilation. Il ne fait aucune vérification sur la structure du fichier.
L'interpréteur saute la ligne de déclaration de procédure binaire qui peut donc être mise n'importe où dans le programme. Il est toutefois préférable de regrouper les déclarations de procédure binaire pour permettre de les retrouver plus aisément. L'interpréteur vérifie que le nombre de paramètres est le même è la déclaration et à l'appel, il vérifie également la compatibilité de type (appel et déclaration de type RETOUR ou de type RESULTAT). Ces vérifications ne garantissent pas un bon fonctionnement de la procédure binaire : avant tout essai la prudence recommande de faire une copie du programme sur disquette ou cassette.

LE L.S.E. ET LA RÉCURSIVITÉ

La récursivité simple est la possibilité de réaliser des procédures qui s'appelent elles-mêmes directement. La récursivité croisée concerne des procédures s'appelant les unes les autres de telle sorte que la procédure appelante soit appelée par une autre.

Le L.S.E. permet la récursivité simple et croisée sans aucune limitation si ce n'est celle de la place mémoire.

Se reporter au manuel utilisateur.

LES PROCÉDURES ET LE CALBUR

Il n'est pas possible de **déclarer** des procédures en CALBUR, mais leur appel est possible, qu'il s'agisse de procédures L.S.E. ou de procédures binaires, qu'il s'agisse de procédures internes ou externes.

Néanmoins la rencontre dans une procédure L.S.E. de certaines instructions interdites en CALBUR provoquera une erreur d'exécution. Cela concerne par exemple l'instruction PAUSE, qui, si elle était autorisée, conduirait à une ambiguité au niveau de la commande COntinuer.

En cas d'erreur d'exécution dans une procédure utilisée en CALBUR la pile d'exécution est remise dans l'état où elle était avant l'appel. L'exécution d'une procédure LSEG-EDL en mode CALBUR ne permet donc pas sa mise au point.

IV. INSTRUCTIONS SUR FICHIERS

- **GARER**
- **CHARGER**
- **SUPPRIMER**
- **EXECUTER**

INSTRUCTION GARER

Syntaxe :
GARER id, ea, ec ou GARER id, ea, ec, id1
id désigne une variable simple ou un tableau
(ea) est le numéro d'enregistrement
(ec) est le nom du fichier
id1 désigne une variable numérique

La donnée contenue dans id est transférée dans l'enregistrement numéro (ea) du fichier de nom (ec). Si id1 est présent, il reçoit un compte rendu de l'échange. S'il y a un défaut pendant l'exécution de cette instruction :
– id1 est absent, le programme est interrompu et un message d'erreur affiché sur l'écran.
– id1 est présent, cette variable reçoit un numéro de compte rendu et l'exécution du programme n'est pas interrompue (il est conseillé dans ce cas de tester (id1)).
Ce compte rendu sera :
– nul si l'opération s'est déroulée correctement,
– négatif s'il y a eu un défaut (cf. Annexe IX).

INSTRUCTION CHARGER

Syntaxe :
CHARGER id, ea, ec ou CHARGER id, ea, ec id1
les paramètres ont la même signification que dans l'instruction GARER

Le L.S.E. va chercher sur disque le contenu de l'enregistrement (ea) du fichier (ec) et l'affecte à la variable id. Si id1 est présent il reçoit le compte rendu de l'échange. La variable id n'a pas besoin d'être déclarée au préalable : si elle existait, elle est libérée et dans tous les cas l'instruction CHARGER la crée du type de la donnée chargée.

Compte rendu du chargement :
– si l'échange s'est bien déroulé id1 prend la valeur correspondant au type de la donnée chargée (le même que celui donné par la fonction TYP)
– dans le cas contraire id1 prend une valeur négative :
  – 1 si l'enregistrement n'existe pas
  – 2 si le fichier n'a pas été trouvé
Se reporter à l'Annexe IX.
Si id1 est absent et que se produit un défaut de chargement, une erreur d'exécution est détectée.
Se reporter également à l'Annexe VI.

INSTRUCTION SUPPRIMER

Syntaxe :
SUPPRIMER ec. ✳ ou SUPPRIMER ec, ✳, id
SUPPRIMER ec. ea ou SUPPRIMER ec, ea, id
– (ec) est le nom du fichier données à supprimer
– (ea) est le numéro de l'enregistrement

Cette instruction permet dans un programme de supprimer un fichier données (1re et 2e forme) ou 1 enregistrement d'un fichier données (3e et 4e forme).
Si id est présent, il reçoit le compte rendu de l'opération (se reporter à l'Annexe IX).

Exemples :

```
SUPPRIMER 'TOTO', ✳ ; SUPPRIMER 'TOTO' 2
CHAINE A; A ←' FIC' : SUPPRIMER A, 5 ; SUPPRIMER A !' .1' ✳
```

INSTRUCTION EXECUTER

Syntaxe :
EXECUTER ec
EXECUTER ec, ea
EXECUTER ec, ea, id
(ec) désigne un nom de fichier programme
(ea) désigne un numéro de ligne

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.
Si id est présent et si un défaut se produit lors du chargement, cette variable reçoit un compte rendu qui peut être exploité par le programme appelant. Un défaut qui se produit apres que le chargement ait commencé provoquera une erreur d'exécution (se reporter à l'Annexe IX).
Cette instruction permet de chainer des programmes, mais elle ne permet pas le passage de paramètres entre les programmes. Ces parametres ne pourront être passés qu'à l'aide d'un fichier de données. Cette instruction fait disparaître de la memoire vive le programme qui comporte l'instruction de chaînage.

V. INSTRUCTIONS GRAPHIQUES

- TRACE
- CADRER
- MARGER
- NETTOYER
- CIBLE
- MOTIF

INSTRUCTION TRACE

Syntaxe :
TRACE ec
(ec) est une chaîne dans laquelle on analyse la présence de caractères, qui peuvent se trouver dans un ordre quelconque.

Cette instruction permet de définir le type de tracé courant. C'est ce type de tracé qui est pris en compte pour l'affichage de toute forme n'ayant pas de type de tracé propre, c'est-à-dire n'ayant pas utilisé la fonction TRA.

Cette instruction agit sur :
- le type de trait
- la luminosité du trait
- la stabilité du trait : sans effet sur TO7 et MO5
- la couleur dut trait

Pour le type de trait on utilisera le caractère t pour un trait plein (le seul qui peut être obtenu pour l'instant).

Pour la luminosité la présence du caractère S permet d'accéder aux 8 couleurs supplémentaires du TO7-70 et du MO5.

Pour la couleur on pourra rencontrer 0 à 3 des caractères RVB pour obtenir :
- rien pour le noir
- R pour le rouge
- V pour le vert
- RV pour le jaune
- B pour le bleu
- BR pour le magenta
- BV pour le cyan
- RVB pour le blanc

A l'initialisation le type de tracé pris par défaut correspond à B1 soit à un trait bleu continu.

INSTRUCTION CADRER

A la mise en route du L.S.E. le cadre standard est le cadre rectangulaire dont les extrémités de la diagonale ont pour coordonnées (0,0) et (1024,631).
Ce cadre peut être modifié par l'instruction CADRER.

Syntaxe :
CADRER ea1, ea2, ea3, ea4
(ea1) et (ea2) sont les coordonnées du coin inférieur gauche du cadre à définir.
(ea3) et (ea4) sont les coordonnnées du coin supérieur droit.

INSTRUCTION MARGER

La marge est la partie de l'écran qui peut, à un instant donné, recevoir le cadre.
Au lancement du programme la marge est égale à la totalité de l'écran. L'instruction MARGER permet de la modifier.

Syntaxe :
MARGER ea, eat, ea2, ea3, ea4
(ea) est le numéro de la page physique sur TO7 et MO5 la page 0
(ea1) et (ea2) sont les coordonnées du coin inférieur gauche de la marge.
(ea3) et (ea4) sont les coordonnées du coin supérieur droit.

INSTRUCTION NETTOYER

Syntaxe :
Première forme : NETTOYER ea
Cette instruction remet à zéro la marge de la page physique, sans modifier la couleur du fond.

Deuxième forme : NETTOYER ea, ec
Comme précédemment, cette instruction remet à zéro la marge de la page physique, mais cette fois la couleur du fond est celle définie par la chaîne (ec), sous la même forme que dans l'instruction TRACE.

INSTRUCTION CIBLE

Cette instruction permet de saisir les coordonnées d'un point visé par le crayon optique.

Syntaxe :
CIBLE id1, id2
id1 et id2 sont des variables numériques qui prennent pour valeurs respectives les coordonnées d'un point, exprimées en coordonnées écran.
(id1) est compris entre 0 et 1024, (id2) entre 0 et 631.

INSTRUCTION MOTIF

Syntaxe :
MOTIF ec = eg
(ec) représente le nom du motif. C'est lui qui représentera le motif lorsque l'on s'en servira. Ce nom satisfait à la syntaxe des noms L.S.E., c'est-à-dire qu'il contient de 1 à 5 caractéres alphanumériques le premier étant une lettre.
(eg) représente le contenu du motif. Cette expression graphique peut parfaitement se référer à d'autres motifs déjà définis.

Exemple :

MOTIF 'SEGA' = VEC ('A', 100,200) indique que le motif de nom 'SEGA' est un segment allumé de composantes 100,200

Un motif ne peut être redéfini. Dès qu'il a été créé, il subsiste jusqu'à la fin de l'exécution du programme.
Si un motif ou une partie de motif ont subi la fonction TRA avant l'exécution de l'instruction MOTIF le type de tracé ainsi attribué ne pourra plus être modifié par la suite.

Exemple :

MOTIF 'TRIAN' = VEC('A', 200,200) : TRA (VEC('A', 0, – 200), 'BV1') : VEC ('A', – 200, 0) représentera un triangle ayant un côté vertical de couleur cyan, les autres côtes étant de la couleur courante au moment de l'affichage

FONCTIONS

FONCTIONS ARITHMÉTIQUES

- ABS
- ALE
- ATG
- CH
- COS
- ENT
- EXP
- LGN
- RAC
- SGN
- SH
- SIN
- TAN
- TH

ABS

Syntaxe :
ABS (ea)

Résultat :
donne la valeur absolue de (ea)

Exemples :

```
? ABS (– 5)
5
? ABS (10 03)
10 03
```

ALE

Syntaxe :
ALE (ea) $0 \leqslant (ea) < 1$.

Résultat :
Si (ea) = 0 alors ALE (0) est un nombre aléatoire compris entre 0 et 1 (1 exclu). Sinon le résultat est un nombre pseudo-aléatoire, fonction de la valeur de ea, ce qui permet de générer des séries reproductibles.

Exemples :

```
1 * nombres pseudo-aléatoires
4X← 0.2
5 FAIRE 10 POUR J← 1 JUSQUA 10
10X← ALE (X); AFFICHER [F5.5,/] X
15 TERMINER
EXécuter à partir de 1
0.60000
0.80000
0.90000
0.95000
0.97500
0.98750
0.99375
0.49688
0.24844
0.62422
TERMINE EN LIGNE 15
```

Si l'on exécute plusieurs fois ce programme, on obtiendra toujours la même série de nombres.

ATG

Syntaxe :
ATG (ea)

Résultat :
Nombre en radians compris entre – pi/2 et + pi/2 dont la tangente est (ea).

Exemple :

```
? ATG (100)
1.5607967
```

CH

Syntaxe :
CH (ea)

Résultat :
Cosinus hyperbolique de (ea).

Exemples :

```
? CH (0)
1
? CH (3)
10.067662
```

COS

Syntaxe :
COS (ea)

Résultat :
Cosinus de (ea), (ea) étant la mesure d'un angle en radians.

Exemples :

```
? COS (0)
1
P ← 3 1415926536
? COS (P/2)
0
? COS (P/2 + 0 0001)
- 0.001
```

ENT

Syntaxe :
ENT (ea)

Résultat :
Partie entière de (ea), c'est-à-dire le plus grand entier inférieur à (ea).

Exemples :

```
? ENT (5.325)
5
? ENT (- 21 3)
- 22
```


EXP

Syntaxe :
EXP (ea)

Résultat :
Exponentielle de (ea).

Exemples :

```
? EXP (0), EXP (1)
1   2.7182818
```

LGN

Syntaxe :
LGN (ea)
La valeur de ea doit être strictement positive.

Resultat :
Logarithme népérien de (ea)

Exemples :

```
? LGN (1)
0
? LGN (3)
1.0988123
? LGN (- 3)
ERREUR E 42
```

RAC

Syntaxe :
RAC (ea)
(ea) doit être positif ou nul.

Résultat :
Racine carrée de (ea).

Exemples :

```
? RAC (2)
1.4142136
1←3; B←5; C← – 4; ? RAC (B  2 – 4∗A∗C)
8.5440037
? RAC (– 5)
ERREUR E 41
```

SGN

Syntaxe :
SGN (ea)

Résultat :
1 si (ea) >0
0 si (ea) = 0
1 si (ea)<0

Exemples :

```
? SGN (3)
1
? SGN (– 25)
– 1
? SGN (0)
0
```


SH

Syntaxe :
SH (ea)

Résultat :
Sinus hyperbolique de (ea).

Exemples :

```
? SH (0)
0
? SH (1)
1.1752012
```

SIN

Syntaxe :
SIN (ea)
(ea) est la mesure d'un angle en radians.

Résultat :
Sinus de (ea)

Exemples :

```
? SIN (0)
0
P – 3.1415926536
? SIN (P/2); ? SIN (7∗P/6);? SIN (1000∗P)
1
– 0.5
– 1.9895686E – 7
```

TAN

Syntaxe :
TAN (ea)
(ea) est la mesure d'un angle en radians.

Résultat :
Tangente de (ea)

Exemples :

```
? TAN (0)
0
P ← 3.1415926536; ? TAN (P/4); ? TAN ( - 0.45)
1
- 0.48305507
```

TH

Syntaxe :
TH (ea)

Résultat :
Tangente hyperbolique de (ea)

Exemples :

```
? TH (2)
0 96402758
? TH (- 5)
- 0.9999092
```


FONCTIONS CHAÎNES

- CCA
- CNB
- DAT
- EQC
- EQN
- GRL
- ICH
- LGR
- MCH
- POS
- PTR
- REP
- SCH
- SKP
- TMA
- TMI

CCA

Syntaxe :
CCA (ea)

Résultat :
Chaîne de caractères représentant le nombre (ea), au format U.

Exemples :

```
? CCA (56)
56 (chaîne composée des caractères 5 et 6)
? CCA (10000000)
1E7 (chaîne composée des caractères 1,E et 7)
? CCA (5/6)
0.83333333
```


CNB

Syntaxe :
CNB (ec,ea) ou CNB (ec, ea, id)
(ea) doit être positif et au plus égal à LGR ((ec)) + 1

Résultat :
Cette fonction convertit une chaîne de caractères représentant un nombre en L.S.E. en la valeur numérique correspondante.
(ea) est le rang à partir duquel débute la conversion.
La conversion s'arrête dès la rencontre d'un caractère qui ne respecte pas la syntaxe d'écriture des nombres en L.S.E.

Si id est présent, il prend la valeur de la position dans la chaîne (ec) du premier caractère qui a arrêté la conversion.

Remarques :
Si la conversion est arrêtée tout de suite le résultat est 0, et il n'y a pas d'erreur d'exécution.

Pour la conversion les blancs en tête sont ignorés; il en est, par contre, tenu compte dans l'évaluation de la valeur de P.

Exemples :

```
CHAINE T; T ← 'VENDREDI 19 MAI'
? CNB (T,1,P) ,P
0 1
? CNB (T,9,P) ,P
t9 12
? CNB ('t2 AVRIL' ,9,P) ,P
0 9
? CNB ('25 - 13',t)
25
? CNB (' - 25E3',1)
- 25000
? CNB (' - 25E - 3',1)
- 0.025
? CNB ('',1)
0
```

DAT

Syntaxe :
DAT ( )

Résultat :
Chaîne contenant la date, donnée à l'initialisation du système

Exemple :

```
? dat ( )
12/08/83
```

EQC

Syntaxe :
EQC (ea)
(ea) doit être compris entre 0 et 255.

Résultat :
Chaîne composée d un seul caractère.
Le code de ce caractère est (ea).

Exemple :

```
? EQC (65)
A
```


EQN

Syntaxe :
EQN (ec) ou EQN (ec, ea)
(ea) doit être positif et au plus égal à LGR ((ec)) + 1.

Résultat :
Code du caractère de la chaîne qui se trouve en position (ea).
Par défaut la valeur de ea est 1.
Si la valeur de ea est égale à la longueur de la chaîne + 1, le résultat est – 1

Exemples :

```
? EQN ('ABC'), EQN (.65 66 67.) ,EQN ('abcdef',3) ,EQN ('AB',3), EQN ('')
65 65 99  – 1  – 1
```

GRL

Syntaxe :
GRL (ec, ea) ou GRL (ec, ea, id)
(ea) doit être positif et au plus égal à la longueur de (ec) + 1.

Résultat :
Chaîne, composée uniquement de lettres, obtenue de la façon suivante :
On parcourt la chaîne à partir de la position (ea), en allant vers la droite, jusqu'à ce que l'on rencontre une lettre ; celle-ci et les suivantes éventuelles constituent le résultat.
Tout caractère autre qu'une lettre est un caractère d'arrêt.
Si aucune lettre n'est trouvée, le résultat est la chaîne vide ('').
Lorsque id est présent, il prend pour valeur :
– soit la longueur de la chaîne + 1 si on l'a parcourue jusqu'au bout.
– soit la position dans la chaîne du caractère qui a provoqué l'arrêt.

Exemple :

```
CHAINE X; X ← 'IL FAIT BEAU !... C''EST LE PRINTEMPS'
? GRL (X,9,P); ? P; ? GRL (X,P,P); ? P
BEAU
13
C
18
? GRL ('20 MAI 1983', 8,P); ? P
(chaîne vide)
12
? GRL ('20000 personnes',1)
personnes
```


ICH

Syntaxe :
ICH (ec)

Résultat :
Chaîne obtenue en écrivant (ec) de la droite vers la gauche.

Exemple :

```
? ICH ('tintin et milou')
uolim te nitnit
```

Remarque :
ICH (ICH (ec)) = (ec)

LGR

Syntaxe :
LGR (ec)

Résultat :
Nombre de caractéres contenus dans la chaîne (ec).

Exemples :

```
? LGR ('FRANÇOIS')
8
? LGR ('HAUTE' !–' l'garonne')
t3
? LGR (    c''est dimanche')
15 (ATTENTION ne compter qu'une fois le caractère ' mais compter l'espace
avant la lettre c.
? LGR (.65 66 67.)
3 (rappel . .65 66 67  est la chaîne 'ABC')
? LGR ('')
0
```

MCH

Syntaxe :
MCH (ec1. ea1, ea2, ec2) ou MCH (ec1. ea1, ec3, ec2)
(ea1) doit être positif ou nul et au plus egal à la longueur de (ec1) + 1
(ea2) doit être positif ou nul.

Résultat :
Chaine obtenue en remplaçant une partie de (ec1) par (ec2).
(ea1) est la position du 1er caractère à remplacer.
– Si le 3e paramètre est numérique : (ea2) est le nombre de caractéres à remplacer.
– Si le 3e paramètre est de type chaine : (ec3) est constituée des caractères d'arrêt.

Exemples :

```
CHAÎNE X; X ← ' Je n"aime pas les pois'
? MCH (X,11,3  'guère')
Je n'aime guére les pois
? MCH (X,19,0,'petits')
Je n'aime pas les petits pois
X ← 'Ces fleurs sont très fraiches'
? MCH (X,16,5,'')
Ces fleurs sont fraiches
```


POS

Syntaxe :
POS (ec1, ea ec2)
(ea) doit être positif et au plus égal à la longueur de (ec1) + 1

Résultat :
C'est un nombre égal à :
– 0 si l'on ne trouve aucune occurence de (ec2) dans (ec1) à partir de la position (ea)
– Sinon le rang du premier caractère de la première occurence de (ec2) dans (ec1), à partir du rang (ea).

Exemples :

```
? POS ('IL FAUT CORRIGER LA LIGNE'  .3  ,'LA')
18
I ←5  ? POS ('liste des élèves de la classe'  . 2 ∗ I  . 'de')
18
? POS (JEUDI 18 MAI 1983'  .1  ,':)
0
```

Cas particuliers :

```
? POS ( LA'  .3  ,'P )
0
? POS ('LA'  3  .'')
3
? POS ('  ,1  .')
1
```

PTR

Syntaxe :
PTR (ec, ea) ou PTR (ec1, ea, ec2)
(ea) doit être positif et au plus égal LGR ((ec)) + 1.

Résultat :
Pour PTR (ec, ea) :
On repère dans la chaine (ec) à partir de la position (ea), la position de la première lettre, le résultat est alors la position du premier caractère suivant qui n'est pas une lettre.
Si on ne rencontre pas de lettre, le résultat est égal à la longueur de la chaîne + 1.
Pour PTR (ec1, ea, ec2) :
Le résultat est la position dans la chaine (ec1), à partir de la position (ea), du premier caractère qui appartient à (ec2).
Si, à partir de la position (ea), aucun caractére de (ec1) n'est dans (ec2), le résultat est égal à la longueur de (ec1) + 1.

Exemples :

```
? PTR (J''AIME LA MER BLEUE' .t , 'BLEUE')
6
? PTR (Jeudi 16 mai 1963' ,6)
t3
? PTR ('ABCD ; 12' ,3 ,'XY')
6
? PTR ('ABCD t23' ,4)
7
? PTR ('1234' ,5 ,'12XY')
5
```


REP

Syntaxe :
REP (ec. ea)

Résultat ;
Chaîne qui résulte de la concaténation de (ea) chaînes identiques à (ec).
Si (ea) = 0 le résultat est la chaine vide ('').

Exemple :

```
? REP ('ZUT !',5)
ZUT ! ZUT ! ZUT ! ZUT ! ZUT !
```

SCH

Syntaxe :
SCH (ec1, ea1, ea2)
SCH (ec1, ea1, ea2, id)
SCH (ec1, ea1, ec2)
SCH (ec1, ea1, ec2, id)
(ea1) doit être positif et au plus égal à LGR ((ec1)) + 1
(ea2) doit être positif ou nul

Résultat :
Chaîne extraite de (ec1) de la façon suivante :
Le premier caractère du résultat est le caractère de (ec1) situé en position (ea1).
Si (ea1) est égal à LGR (ec1) + 1 alors le résultat est la chaîne vide (").
– Si le 3e paramètre est numérique : (ea2) indique la longueur de la sous-chaîne à extraire, si (ec1) n'a pas assez de caractères, l'extraction s'arrête au dernier caractère de (ec1).
– Si le 3e paramètre est de type chaîne : les caractères de (ec2) constituent un ensemble de caractères d'arrêt de l'extraction (le caractère d'arrêt ne fera pas partie du résultat). Si l'on ne rencontre pas de caractère d'arrêt, l'extraction se poursuit jusqu'au dernier caractère de (ec1).
Si id est présent, il prend pour valeur :
soit la position dans (ec1) du caractère d'arrêt
soit LGR ((ec1)) + 1, lorsque l'on a été jusqu'au bout de (ec1)

Exemples :

```
CHAINE X: X ← 'CE BLOUSON EST GRISATRE
? SCH (X,9,11.P); ?P
ON EST GRIS
20
? SCH (X.20,50 P); ?P
ATRE
24
? SCH (X.LGR (X) – 1")
RE
```


SKP

Syntaxe :
SKP (ec1, ea) ou SKP (ec1, ea, ec2)
(ea) doit être positif ou au plus égal LGR ((ec1)) + 1.

Résultat :
1re **forme** : SKP (ec1, ea)
On explore la chaîne à partir de la position (ea).
Le résultat est la position de la première lettre rencontrée, ou à défaut, la longueur de la chaîne + 1.
2e **forme :** SKP (ec1, ea, ec2)
On explore (ec1) à partir de la position (ea).
Le résultat est la position du premier caractère rencontré dans (ec1) qui ne soit pas dans (ec2).
Si tous les caractère de (ec1) sont des caractères de (ec2) le résultat est la longueur de (ec1) + 1.

Exemples :

```
? SKP ('RESULTAT NUMERIQUE' ,1. 'REPONSE').
4
? SKP ('10 et 10 font 20' ,4 , 'exact')
6
? SKP ('TRA–LA–LA' ,10 ,'RAT')
10
? SKP ( L'AN 2000 APPROCHE' ,5)
11
? SKP (" ,1 .")
? SKP ('A' ,1 ,'A')
2
```

TMA

Syntaxe :
TMA (ec)

Résultat :
Chaîne obtenue en remplaçant chaque lettre minuscule (accentuée ou non) par la lettre majuscule correspondante.

Exemple :

```
? TMA ('la lumière est tamisée')
LA LUMIERE EST TAMISEE
```

TMI

Syntaxe :
TMI (ec)

Résultat :
Chaîne obtenue en remplaçant chaque lettre majuscule (non accentuee) correspondante.

Exemple :

```
? TMI ('LA table EST MISE')
la table est mise
```


FONCTIONS GRAPHIQUES

- AFX
- AFY
- HOM
- MAT
- MTF
- REF
- ROT
- SMO
- SMX
- SMY
- TRA
- TRN
- VEC
- VEQ
- VEX
- VEY

AFX

Syntaxe :
AFX (eg, ea)

Résultat :
Objet graphique, dont le repère est celui de (eg), chaque élément de (eg) ayant subi l'affinité orthogonale d'axe OX et de rapport (ea)
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent est le transformé du vecteur équivalent de (eg).

AFY

Syntaxe :
AFY (eg, ea)

Résultat :
Voir AFX (eg, ea), chaque élément de (eg) ayant cette fois subi l'affinité d'axe OY et de rapport (ea).

HOM

Syntaxe :
HOM (eg, ea)

Résultat :
Objet graphique dont le repère est celui de (eg), chaque élément de (eg) ayant subi l'homothétie de centre l'origine de (eg) et de rapport (ea).
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent du résultat est le transformé du vecteur équivalent de (eg).

MAT

Syntaxe :
MAT (eg, ea1, ea2, ea3, ea4)

Résultat :
Objet graphique dont le repère est celui de (eg), chaque élément de (eg) ayant subi la transformation géométrique définie par la matrice :
(ea1) (ea2)
(ea3) (ea4)
Le repère du résultat est celui de (eg).
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent du résultat est le transformé du vecteur équivalent de (eg).

MTF

Syntaxe :
MTF (ec)

Résultat :
Expression graphique formée par le motif de nom (ec).

REF

Syntaxe :
REF (eg)

Résultat :
Chaîne formée de la liste des noms de motifs contenus dans l'objet graphique (eg). Chaque élément de la liste est précédé de tous les motifs qu'il référence. Les noms de motifs sont séparés par un octet nul. En l'absence de motif, la chaîne est vide.

ROT

Syntaxe :
ROT (eg, ea)
(ea) est un nombre exprimé en radians.

Résultat :
Objet graphique de même repère que (eg). Chaque élément de (eg) a subi la rotation de centre l'origine de (eg), et d'angle (ea).
Le repère du résultat est le repère de (eg).
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent est le transformé du vecteur équivalent de (eg).

SMO

Syntaxe :
SMO (eg)

Résultat :
Objet graphique de même repère que (eg). Cet objet est obtenu en remplaçant chaque vecteur composant (eg) par son opposé.
Le repere du résultat est celui de (eg).
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent est le transformé du vecteur équivalent de (eg).

SMX

Syntaxe :
SMX (eg)

Résultat :
Objet graphique de même repère que (eg) :
chaque élément composant (eg) a subi la symétrie d'axe OX, c'est-à-dire que chaque vecteur composant (eg) est transformé en un vecteur de même composante horizontale, et de composante verticale opposée.

SMY

Syntaxe :
SMY (eg)

Résultat :
Objet graphique de même repère que (eg) :
chaque élément composant (eg) a subi la symétrie d'axe OY, c'est-à-dire que chaque vecteur composant (eg) est transformé en un vecteur de même composante verticale, et de composante horizontale opposée.

TRA

Syntaxe :
TRA (eg, ec)
(ec) chaîne définie de la même manière que dans l'instruction TRACE. (cf. page 00)

Résuttat :
Donne à l'objet graphique les caractéristiques définies par (ec) en supprimant les types de tracé antérieurs éventuels portant sur tout ou une partie de l'objet. Cette fonction est sans effet sur les motifs référencés dans (eg).

Exemples :

```
? TRA (VEC ('A' ,200 ,200) , 'RB1') :VEC('A' ,200 ,0)
```

Vous obtenez la juxtaposition de 2 segments, le premier de couleur MAGENTA, le second de la couleur courante au moment de l'affichage.

```
10 FORME F; F← VEC ('A' ,100 ,0) : VEC ('A' ,0 ,100)
20 MOTIF 'F' = TRA (F ,'R1')
40 AFFICHER TRA (F : MTF ('F') ,'1')
50 TERMINER
```

A l'exécution, on obtient la juxtaposition de 2 dessins, le premier de couleur NOIRE, le second ROUGE. La couleur du MOTIF n'a donc pas été modifiée par la fonction TRA à la ligne 40.

TRN

Syntaxe :
TRN (eg, ea1, ea2)

Résultat :
Objet graphique de même repère que (eg), chaque élément de (eg) ayant subi une translation de vecteur de composantes (ea1), (ea2)
L'extrémité du résultat est le transformé de l'extrémité de (eg).
Le vecteur équivalent du résultat est le transformé du vecteur équivalent de (eg).

VEC

Syntaxe :
VEC (ec, ea1, ea2)

Résultat :

L'origine de cet objet est l'origine du repère, son extrémité le point de coordonnées (ea1), (ea2).
Le vecteur équivalent est le vecteur de coordonnées (ea1), (ea2). Le contenu de la chaîne (ec) précise le mode de tracé du segment :
A : à l'affichage, tous les points de l'écran correspondant aux points du segment prendront l'état ALLUMÉ.
E cela revient au déplacement de l'origine au point de coordonnées (ea1), (ea2).
G : le vecteur est du type GOMME (à l'affichage, tous les points de l'écran correspondant aux points de ce segment prendront l'état ÉTEINT).

VEQ

Syntaxe :
VEQ (eg)

Résultat :
Vecteur équivalent de (eg).

VEX

Syntaxe :
VEX (eg)

Résultat :
Composante horizontale du vecteur équivalent de (eg).

VEY

Syntaxe :
VEY (eg)

Résultat :
Composante verticale du vecteur équivalent de (eg).

FONCTIONS SYSTÈME

- IND
- NIV
- TYP
- TZL

IND

Syntaxe :
IND (id) ou IND (id, ea)
id est un identificateur de tableau.

Résuttat :
IND (id) donne la dimension du tableau de nom id.
IND (id, ea) donne la valeur de la dimension (ea).

Exemples :

```
TABLEAU T[2,4,5]
? IND (T)
3
? IND (T,3)
5
```


NIV

Syntaxe :
NIV ( )
Résultat :
Nombre entier donnant le niveau d'appel de procédures :
– Dans le programme principal NIV ( ) = 0
– Quand on rentre dans une procédure NIV ( ) augmente de 1 ; quand on en sort NIV ( ) diminue de 1
Exemptes :

```
1 * Exemple 1
4 &AP ( )
5 AFFICHER NIV ( )
10 TERMINER
15 ****************
100 PROCEDURE &AP ( )
105 AFFICHER NIV ( )
110 RETOUR
EXécuter à partir de 1
1
0
TERMINE EN LIGNE 10
1 * Exemple 2
4 AFFICHER NIV ( )
5 &AP ( )
10 AFFICHER NIV ( ); TERMINER
15 ****************************
100 PROCEDURE &AP ( )
110 AFFICHER NIV ( ); SI NIV ( ) = 4 ALORS RETOUR
115 &AP ( ); AFFICHER NIV ( )
120 RETOUR
EXecuter à partir de 1
0
1
2
3
4
3
2
1
0
TERMINE EN LIGNE 10
```

TYP

Syntaxe :
TYP (id)
id variable numérique, chaîne, booléenne, graphique, tableau ou élément de tableau.

Résultat :
Donne le type de id d'après le code ci-dessous :
0 : variable numérique affectée.
1 : tableau numérique de dimension 1.
2 : tableau numérique de dimension supérieure à 1.
3 : variable chaîne affectée.
4 : tableau chaîne de dimension 1.
5 : tableau chaîne de dimension supérieure à 1.
6 : variable booléenne affectée.
7 : tableau booléen de dimension 1.
8 : tableau booléen de dimension supérieure à 1.
9 :
10 : variable numérique non affectée.
11 : variable chaîne non affectée.
12 variable booléenne non affectée
13 variable forme affectée.
14 tableau forme de dimension 1
15 . tableau forme de dimension supérieure a 1.
16 variable forme non affectée.

Exemple :

```
CHAINE A; ? TYP (A)
11
```


TZL

Syntaxe :
TZL ( )

Résultat :
nombre d'octets libres en mémoire.
L'utilisation au moment opportun de cette fonction peut permettre d'éviter une erreur d'exécution 03, par exemple en libérant des variables.

ANNEXES

ANNEXE I
ERREURS DE COMPILATION DU LSEG-EDL

C 01 utilisation d'un caractère interdit
C 02
C 03 identificateur ou nom de procédure trop long ou mot clé de plus de 5 caractères mal orthographié
C 04 plus de 230 identificateurs de plus d un caractère
C 05
C 06 nom de procédure ne commençant pas par une lettre
C 07 numero de ligne incorrect ou non suivi d'un espace
C 08 absence de quote en fin de chaîne
C 09 nombre mal formé ou de taille incorrecte
C 10 expression non booléenne
C 11 melange de types dans une expression
C 12 nom de variable incorrect ou absent
C 13
C 14 nom de tableau non suivi d'un [
C 15 expression indicée non arithmétique
C 16 séparateur d'indice incorrect ou] l'absent
C 17
C 18
C 19 expression incorrecte
C 20 parenthèse fermante mal placée ou absente
C 21 absence de SINON dans expression conditionnelle
C 22 absence de ALORS dans expression ou instruction conditionnelle
C 23
C 24 expression trop imbriquée (plus de 10 niveaux)
C 25 plus de 16 paramètres dans lire, afficher, appel ou déclaration de procédure
C 26
C 27 expression non arithmétique
C 28 expression non chaîne
C 29 nombre de paramètres incorrect
C 30
C 31
C 32 numero de ligne de fin de boucle incorrect ou absent
C 33 variable de boucle incorrecte ou affectation absente
C 34 TANT QUE ou JUSQUA incorrect ou absent
C 35 identificateur incorrect ou absent dans LIBERER, CHAINE, LOCAL ..

C 36 il manque « : » ou FIN dans une instruction composée
C 37 ALLER non suivi de EN
C 38 nom de variable isolée
C 39 nom de tableau absent
C 40 dans un format séparateur incorrect ou] absent
C 41
C 42 entier court incorrect ou absent
C 43 dans un format specification incorrecte ou interdite ou facteur de répétition incorrect
C 44 nom de variable incorrect ou absent dans CHARGER ou GARER
C 45
C 46
C 47 variables indentiques dans LIBERER CHAINE, . , ou paramétres identiques dans une déclaration de procédure
C 48 double déclaration de procédure ou noms de procédures paramètres identiques ou nom de procédure paramètre egal au nom de la procédure déclarée
C 49 déclaration de procédure non suivie de & nom de proc. (
C 50 expression différente d'une expression arithmétique ou chaîne
C 51 declaration de procédure avec parametre formel incorrect ou) absente
C 52
C 53 instruction incorrecte
C 54 plus de spécifications que de variables
C 55
C 56 ligne codee trop longue
C 57 zone utilisateur pleine : provoquer un tassage de la CPF
C 56 instruction interdite en mode de bureau
C 59
C 60
C 61
C 62
C 63
C 64
C 65 création en mode de burau d'une rèference de plus de 1 caractere
C 66
C 67 erreur au chargement d'une procédure binaire
C 68 ligne de dèclaration de procédure binaire contenant d'autres instructions ou un commentaire
C 69 numéro de voie d'entree/sortie incorrect
C 70
C 71 usage incorrect du point
C 72

C 73 expression non graphique
C 74 BINAIRE non précédé de PROCEDURE ou nom de procédure binaire égal à un mot réservé
C 75 signe = absent dans l'instruction MOTIF.

ANNEXE II
ERREURS D'EXÉCUTION DU LSEG-EDL

E 01 chaine valeur de REP trop longue
E 02 programme trop long, exécution impossible
E 03 manque de place pour les données même après tassages de procédures externes inactives
E 04 déclaration en CHAINE, BOOLEEN ou FORME d'un identificateur d'un autre type
E 05 operande non numérique dans une expression numérique
E 06 mélange de types à l'affectation ou variable non numérique dans CIBLE.
E 07 branchement à une ligne inexistante
E 08 opérande non chaîne dans un expression chaîne
E 09 paramètre de ALE non dans l'intervalle [0,1[
E 10 deuxième opérande incorrect dans une comparaison
E 11 procédure externe ne contenant pas la procédure de même nom
E 12 tentative d'exécution d'une ligne de déclaration de procédure
E 13 nombre de paramètres à l'appel d'une procédure différent de celui de la déclaration
E 14 variable non affectée comme paramètre effectif par valeur
E t5 valeur comme paramètre effectif par référence
E 16 deuxième paramètre de SCH, GRL, PTR. SKP, MCH, EQN, ou CNB négatif ou nul
E 17 paramètre effectif différent d'un nom de procédure, le paramètre formel étant un nom de procédure
E 18 procédure inexistante, ni interne ni externe
E 19 procédure paramètre effectif inexistante
E 20 RETOUR ou RESULTAT hors d'une procédure
E 21 RETOUR dans une procédure utilisée comme expression
E 22 RESULTAT dans une procédure utilisée comme instruction
E 23 troisième paramètre de SCH, PTR, MCH ni nombre ni chaîne
E 24 deuxième paramètre de SCH. GRL, PTR, MCH, SKP. EQN, ou CNB non numérique
E 25 puissance non entière d'un nombre négatif
E 26 deuxième paramètre de SCH, GRL. PTR, MCH, SKP, EQN ou CNB superieur a la longueur de la chaîne plus un
E 27 opérande différent d'une variable simple dans POPVR
E 28 identificateur inexistant dans AFFEC

E 29 déclaration de tableau numérique, chaîne, booléen ou forme portant sur un identificateur d'un autre type
E 30 paramètre de EQC non dans l'intervalle [0,255]
E 31 division par zéro
E 32
E 33
E 34 calcul de zéro à une puissance négative
E 35 utilisation de $ hors d une procédure
E 36 types ou nombres de parametres d une procédure binaire différents a l'appel et à la declaration
E 37 numéro de ligne inferieur à un
E 38 nombre non convertible en entier
E 39 numéro de ligne de fin de boucle inferieur au numéro de ligne de debut de boucle.
E 40 pas de ligne après la ligne de fin de boucle
E 41 racine carrée d un nombre négatif
E 42 logarithme neperien d'un nombre négatif ou nul
E 43 variable de contrôle de boucle non numérique
E 44 debordement d'un nombre flottant
E 45 déclaration d'un tableau trop grand
E 46 acces avec indices à une variable simple
E 47 nombre d'indices d'une variable indicée incorrect
E 48 indice inférieur à un ou trop grand
E 49 paramètres de MARGER incorrects
E 50 paramètre compte rendu de GARER, CHARGER ou SUPPRIMER différent d'une variable numérique
E 51
E 52 paramètre nom de fichier de GARER, CHARGER, SUPPRIMER ou EXECUTER incorrect
E 53 numero d'enregistrement non numérique
E 54 numéro d'enregistrement négatif ou nul
E 55 premier paramètre de GARER non défini
E 56 nombre d'enregistrements trop grand (plus de 84)
E 57 boucles mal imbriquées
E 58 pas de ligne après la ligne courante
E 59 RESULTAT d'une procédure différent d'une expression
E 60 variable non définie
E 61 numéro de voie ou de page physique incorrect ou inexistant ou format variable incorrect
E 62 format U seul autorisé pour chaîne, booléen ou forme
E 83 trop d'imbrications de superposition (plus de 20)

E 64 manque de place sur disque pour GARER
E 65 erreur disque
E 66 premier paramètre de CHARGER de type différent de la donnee à charger
E 67 manque de place en mémoire pour CHARGER ou EXECUTER
E 68 fichier inexistant
E 69 enregistrement inexistant
E 70 instruction interdite en mode de bureau
E 71 paramètre procédure interdit en procedure externe
E 72 utilisation d'un paramètre effectif au lieu d un paramètre formel
E 73
E 74 RETOUR EN dans une procedure appelée en mode de bureau
E 75 EXECUTER porte sur un numéro de ligne inexistant
E 76 fichier protégé en lecture
E 77 fichier protégé en ecriture
E 78 élément à affecter ni nombre ni chaîne ni booleen ni forme
E 79 motif en mémoire différent de celui qui est chargé
E 80 trop d'imbrications de motifs referencés
E 81 appel d'une procédure binaire inexistante
E 82 lecture incorrecte d'une expression numérique ou booléenne
E 83 catalogue incorrect
E 84 catalogue plein
E 85 unité inexistante
E 86 opérande non booléen dans une expression booléenne
E 87 deuxième paramètre de IND inférieur à un ou supérieur à la dimension du tableau
E 88 deuxième paramètre de REP négatif
E 89 type de tracé incorrect
E 90 opérande non FORME dans une expression FORME
E 91 deuxième paramètre de NETTOYER incorrect
E 92 le motif existe déjà
E 93 paramètre de VEQ, VEX, VEY ou REF non FORME
E 94 référence à un motif inexistant
E 95 premier paramètre de VEC de valeur incorrecte
E 96 numéro de page physique incorrect
E 97 nom de motif incorrect
E 98 erreur provoquée dans une procédure binaire
E 99 programme en mémoire incorrect
E 100 instruction non interprétée
E 200 à 255 réservées pour personnaliser les erreurs des procédures binaires

ANNEXE III
ERREURS SUR FICHIERS DU LSEG-EDL

F 01 ✳ code de fonction FMS illégal
F 02 ✳ le fichier est déjà utilisé
F 03 le fichier existe déjà
F 04 le fichier n'a pas été trouvé
F 05 + erreur de catalogue
F 06 catalogue plein
F 07 disque plein
F 08 ✳ fin de fichier en lecture
F 09 erreur de lecture disque
F 10 erreur d'écriture disque
F 11 fichier ou disque protégé en écriture
F 12 essai de suppression d'un fichier protége
F 13 ✳ bloc de contrôle de fichier illégal
F 14 ✳ adresse disque illégale
F 15 numéro de disque illegal
F 16 disque non prêt
F 17 accès refusé à un fichier protégé en lecture
F 16 ✳ attributs d'accès à un fichier incorrects
F 19 ✳ pointeur d'accès disque erroné
F 20 recharger le système
F 21 ✳ spécification de fichier incorrecte
F 22 erreur de fermeture de fichier
F 23 ✳ débordement de la table d'allocation
F 24 numéro d'enregistrement inexistant
F 25 + fichier endommagé
F 26
F 27
F 28 ✳ configuration hardware insuffisante

Les numéros suivis de ✳ concernent des erreurs qui ne peuvent se produire que dans des procédures binaires par suite d'une mauvaise utilisation des fichiers. Les numéros suivis de + concernant des défauts de la disquette ou du système et ne doivent pas en principe se produire.

ANNEXE IV
MOTS RÉSERVÉS DU LSEG-EDL

Les mots de cette liste qui ont moins de 6 caractères ne peuvent être des noms de variables. Les noms de fonctions, qui ne sont pas mis dans cette liste, ne peuvent être utilisés pour désigner des procédures binaires.

afficher
aller
allumer
alors
binaire
booleen
cadrer
chaine
charger
cible
couleur
debut
dij
et
eteindre
executer
faire
fin
forme
garer
jusqua
liberer
lire
local
marger
motif
nettoyer
non
ou
pas
pause

pour
procedure
resultat
retour
si
sinon
supprimer
tableau
tant
terminer
trace

ANNEXE V : RÉALISATION DE PROCÉDURES BINAIRES EN LSEG-EDL

I. GÉNÉRALITÉS

La réalisation de procédures binaires en LSEG-EDL suppose connues les instructions du 6809. Elle supppose aussi que l'on ait une certaine idée du fonctionnement de l'interpréteur et en particulier de la gestion de la pile d'exécution. Pour générer une procédure binaire, il faut disposer d'un assembleur.

II. LIAISON AVEC L'INTERPRÉTEUR DU LSEG-EDL

Un fichier de nom PROBIN définit les points de communication entre la procédure binaire et le LSEG-EDL. Le source de la procédure binaire doit contenir la directive LIB PROBIN pour pouvoir utiliser ces points de communication. Ces points permettent de :
- récupérer les paramètres d'appel de la procédure
- retourner un résultat au programme appelant
- effectuer des calculs en flottant
- faire des entrées-sorties simples
- traiter les erreurs d'exécution de la procédure
- faire d'autres traitements impossibles à réaliser en langage évolue.

Ces points d'entrée ont en général une structure de sous-programme et on fait appel à eux par l intermédiaire de l'instruction JSR. Les registres du 6809 ne sont pas modifiés par cet appel à l'exception de ceux qui reçoivent les résultats.

1. Extraction d'un élément de la pile

Les points d'entree suivants permettent de récupérer l'élément qui est au sommet de la pile d'exécution. Une erreur d'exécution se produit si le type de l'élément n'est pas correct
POP16 extrait un entier signé sur t6 bits et le met dans le registre D. Une erreur se produit si l'élément n'est pas convertible en entier, c'est-à-dire s'il déborde.
POPFL extrait un flottant court de la pile d exécution et met son adresse dans le registre X.

POPCH extrait une chaîne, met dans X l'adresse du premier octet de cette chaîne et dans D le nombre d'octets.

POPBL extrait un booléen de la pile et met le résultat dans le registre B (0 si FAUX, 1 si VRAI)

POPFO extrait une forme de la pile d'exécution, met dans X l'adresse du premier octet et dans D la longueur.

POPVR extrait un identificateur de variable simple et met son numéro dans le registre B. Une erreur E 27 est détectée si l'identificateur est un nom de tableau.

EXTOP extrait l'opérande qui est au sommet de la pile, met dans X l'adresse de cet opérande et dans B son type L.S.E. La seule erreur possible est une pile d'exécution vide ou un élément haut de pile qui n'est pas un opérande. Il est nécessaire de connaître la structure des données en mémoire pour utiliser la valeur qui est dans X.

2. Mise d'un élément dans la pile

Ces points d'entrée mettent l'élément désigné au sommet de la pile d'exécution. La seule erreur possible est le manque de place.

PSHt6 met l'entier signé de D au sommet de la pile

PSHFL met le flottant court dont l'adresse est dans X au sommet de la pile.

PSHCH met la chaîne dont l'adresse est dans X et la longueur dans D au sommet de la pile.

PSHBL met le booleen de B au sommet de la pile.

PSHFO met la forme dont l'adresse est dans X et la longueur dans D au sommet de la pile.

3. Effectuer des calculs en flottant

Les nombres flottants en zone de données ou dans la pile d'exécution sont des flottants courts sur 5 octets (t pour l'exposant et 4 pour la mantisse). Les opérations en flottant se font à partir de deux pseudo-accumulateurs flottants désignés par FA et FB. Pour les opérations binaires le premier opérande est FA, le deuxième opérande FB, le résultat est mis dans FA, le pseudo-accumulteur FB peut être détruit dans le calcul. Dans FA et FB on met des flottants longs, c'est-à-dire sur 7 octets (2 pour l'exposant et 5 pour la mantisse). La mise des flottants dans FA et FB ainsi que la récupération du résultat à partir de FA sont à la charge du programmeur qui doit aussi traiter le débordement dans les calculs et les arrondis consécutifs au passage de flottant long à flottant court.

ADXFA met dans X l'adresse de FA.

ADXFB met dans X l'adresse de FB

FLADD effectue l'addition de FA avec FB. résultat dans FA

FLSOU met dans FA la difference entre FA et FB

FLMUL effectue le produit de FA par FB avec resultat dans FA

FLDIV met dans FA le quotient de FA par FB

FLNOR normalise le flottant de FA

4. Entrées-sorties simples

ENTCA attend un caractère au clavier et le retourne dans le registre A avec écho à l'écran.

SORCA affiche sur l'écran le caractère contenu dans le registre A.

VOLEE teste si un caractére a été tape au clavier. Si oui retourne ce caractère dans le registre A et met la condition NON ZERO. sinon met la condition ZERO.

5. Erreurs d'exécution

Dans une procédure binaire on peut distinguer d'une part les erreurs détectées par l'interpréteur qui ont le même sens que dans les séquences écrites en LSEG-EDL et d'autre part les erreurs détectées à l'intérieur de la procédure. Nous ne parlons ici que de ces dernières.
Pour s'y retrouver plus facilement, convenons d'utiliser l'erreur 98 pour des paramètres incorrects et les numéros 200 à 249 pour des defauts rencontrés lors de l'exécution de la procédure.

ERROR mettre dans le registre A le numéro de l'erreur puis appeler ce point d'entree par un JMP

5. Autres points d'entrée

DMDPL demande de place en zone de données. Avant l'appel, mettre dans D le nombre d octets demandés, au retour X contient l'adresse du premier octet

INHIT inhibe le traitement d'interruption en fin de ligne et permet ainsi d'avoir un programme impossible à interrompre.

ACTIT rétablit le traitement normal en fin de ligne. Annule l'effet du point d'entree precédent.

AFFEC réalise l'affectation d'une valeur à une variable. Avant l'appel, mettre la valeur dans la pile d'exécution et le numéro de l'identificateur dans le registre B. En plus des erreurs habituelles de l'affectation, une erreur E 28 est possible si l'identificateur dont le numero est donné dans B n'existe pas. Au retour, la valeur a été extraite de la pile d'exécution. L'utilisateur de ce point d'entrée suppose que les registres U et Y n'ont pas été modifiés dans la procédure

AFFE2 fonctionne comme le précédent. Avant l'appel, il doit y avoir dans la pile d'exécution l'identificateur puis la valeur. La plus souvent, l'identificateur sera un paramètre non encore extrait. A la sortie les deux éléments sont extraits de la pile.

III. LISTING DE PROBIN

```
                ORG  $ 0100
0100 POP16      RMB  3 extrait un entier 16 bits signé de la pile et le met
                        dans D
0103 POPFL      RMB  3 extrait un flottant de la pile, X pointe sur adresse
                        flottant
0106 POPCH      RMB  3 extrait une chaîne de la pile, X pointe sur adresse.
                        D = longueur
0109 POPBL      RMB  3 extrait un booléen de la pile, B = 0 ou 1
010C POPFO      RMB  3 extrait une forme de la pile, X pointe sur adresse,
                        D = longueur
010F PSH16      RMB  3 D entier 16 bits signé est mis dans la pile
0112 PSHFL      RMB  3 le flottant court pointé par X est mis dans la pile
0115 PSHCH      RMB  3 la chaîne pointée par X de longueur D est mise dans
                        la pile
0116 PSHBL      RMB  3 le booléen B = 0 ou 1 est mis dans la pile
011B PSHFO      RMB  3 la forme pointee par X de longueur D est mise dans
                        la pile
011E EXTOP      RMB  3 extrait un opérande de la pile, X pointe sur adresse,
                        B = typa L.S E.
0121 DMDPL      RMB  3 Demande de D octets, X pointe sur le 1er octet libre,
                        TZL = U – X
0124 ERROR      RMB  3 A, erreurs parsonnalisées entre 200 et 249
0127 FLADD      RMB  3 FA ← FA + FB
012A FLSOU      RMB  3 FA ← FA – FB
```



```
012D FLMUL      RMB  3 FA ← FA * FB
0130 FLDIV      RMB  3 FA ← FA/FB
0133 FLNOR      RMB  3 normalisation flottant
0136 ADXFA      RMB  3 met dans X l'adresse de FA
0t39 ADXFB      RMB  3 met dans X l'adresse de FB
013C VOLEE      RMB  3 A = caractère tapé si Z = 0
013F ENTCA      RMB  3 A = caractère tapé
0142 SORCA      RMB  3 affiche le caractère mis dans A
0145 ENTFA      RMB  3 FA est mis dans D en entier; RTS !!
0146 RESPL      RMB  3 RTS !!
014B LIBPL      RMB  3 RTS !!
014E ADRPL      RMB  3 RTS !!
0151 POPVR      RMB  3 extrait un identificateur de la pile et met son numero
                        dans B
0154 AFFEC      RMB  3 affecte la valeur mise dans la pile à l'identificateur
                        dont le numéro est dans B
0157 AFFE2      RMB  3 affecte la valeur mise dans la pile à l'identificateur
                        qui l'y précède
015A CBIN       RMB  3 RTS !!
015D INHIT      RMB  3 inhibe STOP
0160 ACTIT      RMB  3 restaure STOP
0163 FMS        RMB  3 relai FMS
0166 COMPI      RMB  3 relai compilateur RTS !!
0169            RMB  3
016C            RMB  3
016F            RMB  3
0172            RMB  3
0175            RMB  3
```

IV. CONTENU D'UNE PROCÉDURE BINAIRE

Afin de permettre l'appel de la procedure binaire par l'interpréteur, le programmeur doit respecter un certain nombre de règles :

– Les deux premiers octets contiennent une instruction BRA de branchement au début de la procédure binaire. Cette instruction etant un branchement relatif court, cela impose une longueur maximum de 126 octets à la séquence qui suit.

– L'octet 2 contient le numero de version de la procédure binaire. L'interpréteur ne se sert pas de cet octet qui n'est là que pour assurer la comptabilite avec les utilitaires du GOUPIL III équipé d'un 6809.

– Les octets 3 et 4 contiennent la longueur totale en octets de la procédure binaire Ces octets sont utilisés au chargement de la procedure binaire. C'est le seul moyen dont dispose le LSEG-EDL pour connaître la taille exacte de la procedure qui ne figure pas dans le catalogue. Pour l'instant cette longueur ne peut dépasser 1,5K octets mais aucun contrôle n'est fait sur ce dépassement.

– Loctet 5 sert à définir la première possibilité d'appel de la procédure. Le bit 7 (bit de fort poids) est à zéro pour une procédure de type résultat (fonction) et à un pour une procedure de type retour (sous-programme). Les bits 0 et 6 contiennent le nombre de paramètres correspondant à cet appel L'interpréteur compare les éléments de l'appel de la procedure binaire avec le contenu de cet octet, en cas d'égalité le lancement de la procédure se fait avec le CARRY à zéro. Dans le cas contraire l interpreteur analyse l'octet 6...

– L'octet 6 sert à définir une deuxième possibilité d'appel. Si cet octet est à zéro, cette possibilité n'a pas eté prevue... S'il est non nul les bits ont la même signification que dans l'octet 5 Noter que cet octet 6 ne peut servir à définir un appel de type fonction sans paramètre car il conduirait à un octet nul. Si l'interpréteur n'a pu faire l'appel avec les élements de l'octet 5 alors :

– si l'octet 6 est nul, il retourne une erreur d'exécution
– si l'octet 6 n'est pas nul et si les paramètres sont compatibles le lancement de la procédure se fait avec le CARRY à un
– si les paramètres ne sont pas compatibles, la méme erreur d'exécution est renvoyée.

– Loctet 7 est destiné à un usage futur et est à zéro. Le LSEG-EDL ne prend pas en compte cet octet.

– Les octets suivants contiennent les instructions et les données de la procédure. Cette procedure binaire doit avoir une structure de sous-programme et donc se terminer par un RTS. Le code généré doit être tel que la procedure soit translatable, aucune vérification n'est faite dans le LSEG-EDL, alors prenez garde. . Le LSEG-EDL utilise le registre U pour la gestion de la pile d'exécution, la procédure binaire ne devra pas perturber cette utilisation, le mieux étant de ne pas toucher à ce registre; si toutefois on a besoin de l'utiliser entre le moment où on a extrait tous les opérandes et celui ou on mettra le resultat dans la pile, il suffit de l'empiler dans la pile S puis de le récupérer à la fin Le registre Y ne doit pas non plus être modifié tant que l'on n'a pas extrait tous les opérandes.

V. SQUELETTE D'UNE PROCEDURE BINAIRE

Voici quel doit être le squelette de toute procédure binaire en LSEG-EDL :

```
        nam  nom de la procédure binaire
        lib  probin
        org  $ C100
debut   bra  deb        vers première instruction
        fcb  t          version 1
        fdb  fin-début  longueur en octets
        fcb  ??         premier appel
        fcb  ??         deuxième appel
        fcb  0          inutilisé
* zone de données éventuelles
* attention à sa longueur à cause du bra du début.
* zone de programme
deb     equ  *
* zone de programme et de données
fin     equ  *
        end
```

Remarques :
– Ne pas utiliser Y et U avant de faire les « POP ».
– Restaurer Y et U avant de faire les « PSH ».

ANNEXE VI
COMMODITÉS PARTICULIÈRES A L'IMPLÉMENTATION DU LSEG-EDL

I. INSTRUCTION LIRE :

1) LIRE [@1] permet une lecture sans écho à l'écran.
2) A la lecture d'une valeur numérique on peut répondre par une expression à condition de la faire précéder de « : ». Cette expression doit bien évidemment respecter la syntaxe L.S.E et donner un résultat numérique

Exemple :

```
1 LIRE A
2 AFFICHER A: TERMINER
EXECUTER A PARTIR DE 1
: 1/3
0.3333333
TERMINE EN LIGNE 2
EXECUTER A PARTIR DE 1
: SIN (1.2) * SIN (1.7)/COS (2 1)
- 1 8307963
TERMINE EN LIGNE 2
```

II. CHAÎNES DE CARACTÈRES :

Tout ce qui peut être affiché peut être affecté à une chaine (y compris les formats)

Exemple :

```
CHAINE C
P ← 3.141593
C ← [U,2/,5X,F2.4,/,U] 'Le nombre PI vaut',P, 'à peu de chose près !'
? C
Le nombre PI vaut


        3.1416

à peu   de chose près !
```

III. NUMÉROS DE LIGNES :

Les numéros de lignes peuvent dépasser 32000 et aller jusqu'à 32767.
Lors d'un EXECUTER A PARTIR DE N, si la ligne de numéro N n'existe pas, l'exécution se produira à partir de la première ligne de numéro supérieur à N.

Exemple :

```
t0 AFFICHER 'Ligne 10'
30 AFFICHER 'Ligne 30'
40 TERMINER
EXECUTER A PARTIR DE 20
Ligne 30
TERMINE EN LIGNE 40
```

Si la ligne de fin de boucle n'existe pas (FAIRE N... sans ligne de numéro N), la dernière ligne de la boucle sera la dernière ligne de numéro inférieur à N.

Exemple :

```
10 FAIRE 30 POUR I ← 1 JUSQUA 3
20 AFFICHER 'Ligne 20'
40 AFFICHER 'Ligne 40'
50 TERMINER
EXECUTER A PARTIR DE 1
Ligne 20
Ligne 20
Ligne 20
Ligne 40
TERMINE EN LIGNE 50
```

IV. VARIABLES SYSTÉME :

1) Variable @ :

Il est possibe en LSEG-EDL d'utiliser une variable numérique appelée @ (symbole aroba) et dont le contenu est égal au numéro de ligne sur laquelle on l'utilise. Cette valeur est affectée par l'interpréteur au moment de l'utilisation de la variable

2) Variable $:

Lors d'un appel de sous-programme, cette variable contient le numéro de la ligne d'appel Elle a été créée pour faciliter l'usage de RETOUR EN.

Exemple :

```
.
10 CHAINE REP  CH
.
200 LIRE [/ ,'Voulez-vous continuer (O/N)?'] REP; & BRAN (REP , 'O' , 'N' .
@210 ,@999)
210 .
630 LIRE [ ,Quel est votre choix (A ou B)?'] CH; & BRAN (CH ,'A' ,'B' ,@700 .
@800) .
700 ... * Choix A
.
800 ... * Choix B
. .
999 TERMINER
.
10000 PROCEDURE &BRAN (C , C1 . C2 , L1 . L2) LOCAL L2 , L1 . C2 ,
C1 . C
10010 RETOUR EN SI C = C1 ALORS L1 SINON SI C = C2 ALORS L2 SINON $
. .
```

V. NUMÉROS D'ENREGISTREMENT NÉGATIFS :

Lorsque l on veut parcourir un fichier du début à la fin, il est souhaitable de ne pas chercher inutilement un enregistrement inexistant (cela prend du temps). C'est pourquoi on a choisi une formule qui permet au moment du chargement d un enregistrement de connaître le numéro du suivant, s'il existe.

On utilise l'instruction CHARGER :

CHARGER id, id2, ea ou CHARGER id, id2 ea id1

Le deuxième paramètre doit obligatoirement être un nom de variable numérique car il est modifié pendant l'execution de l'instruction.

Principe : considérons l'instruction CHARGER A N 'F' et parcourons tout le fichier avec cette instruction

– au depart on affecte 0 à N puis on execute l instruction qui charge dans A le premier (celui de plus petit numéro) enregistrement trouvé et met dans N l opposé du numéro de l'enregistrement.

– sans modifier N on relance le chargement. Deux cas se presentent

  – le fichier contient encore un enregistrement, alors l'enregistrement suivant est charge dans A l'oppose du numero de cet enregistrement est mis dans N.

– L'exploration du fichier est terminée, alors 0 est mis dans N

Exemple :

Soit a parcourir un fichier F contenant les enregistrements 12 et 100, on fera .

N ← 0, CHARGER A N, 'F'
La valeur récupérée dans N est – 12 (A contient l enregistrement 12)
CHARGER A, N, 'F'
La valeur récupérée dans N est – 100 (A contient l'enregistrement 100)
CHARGER A, N, 'F'
La valeur récupérée dans N est 0 (A n'a pas été modifié). Noter que dans ce cas malgré l'absence du 4e paramètre il n'y a pas eu d'erreur d'exécution

Cela permet de simuler facilement la fonction NES (Numéro d'Enregistrement Suivant) qui n'est pas implémentée en LSEG-EDL. Il suffit d'utiliser par exemple la procédure &NES :

```
10000 PROCEDURE &NES (FIC, NUM) LOCAL NUM. FIC. X
10010 NUM ← – NUM; CHARGER X, NUM, FIC
10020 RESULTAT – NUM
```

VI. ENTRÉE DE CODES AU CLAVIER :

La touche « INS » permet d'entrer sans aucun transcodage et sous forme de deux chiffres hexadécimaux tout code de 0 à 255

Exemple :

« INS » 41 affiche A sur l'écran
« INS » 37 affiche 7 sur l'écran

Cette touche sera utilisée chaque fois que l'on aura besoin d'un caractère qui n'est pas représenté sur le clavier.

Exemple :

ESCAPE s'obtient par « INS » 1B
US s'obtient par « INS » 1F

ANNEXE VII
POSSIBILITÉS D'AFFICHAGE SUR T07, T07-70 et M05

Cette annexe traite de notions propres a chacune des trois machines. Quelques differences existent entre ces matériels; certaines options n'existent pas sur tous. Chaque fois que cela est necessaire, il est precisé si l'option existe ou pas

I. PRÉSENTATION DE L'ÉCRAN

Votre écran se compose de
– 25 lignes numérotées de 0 à 24
– 40 colonnes numérotées de 1 à 40

Cet écran est divisé en deux parties

– Le tour pour lequel vous ne pourrez choisir que la couleur
– L'interieur, partie dans laquelle vous allez travailler

Le T07 met à votre disposition huit couleurs :
NOIR ROUGE - VERT - JAUNE - BLEU - MAGENTA - CYAN - BLANC

Le T07-70 et le M05 mettent en plus 8 autres couleurs à votre disposition. Ces couleurs dites pastel, s'obtiennent en ajoutant du BLANC aux couleurs de même rang de la liste precedente (sauf le BLANC qui donne l'ORANGE). On obtient ainsi les couleurs suivantes :

GRIS - ROUGE clair - VERT clair - JAUNE clair BLEU clair
ROSE (MAGENTA clair) - BLEU ciel (CYAN clair) - ORANGE

Chacune de ces couleurs peut définir :
– la couleur du caractère tapé
– la couleur du fond de ce caractère

Chacune de ces possibilités peut être obtenue par affichage d'une séquence de codes

Exemple :

Tapez au clavier la sequence suivante

ESC P	ESC A	T07
(fond noir)	(caractères rouges)	

Vous devez voir sur l'écran le mot T07 en rouge sur fond noir.

Remarque :
Pour obtenir le caractère ESC, vous devez taper successivement sur les touches « INS » t et B

II. DÉFINITION D'UNE FENÊTRE

Une fenêtre est définie par le numéro de la ligne du haut et par le numéro de la ligne du bas.

Definition de la ligne haute :

US ($20 + D) ($20 + U)

- Le code « US » s'obtient en tapant « INS » 1F
- D désigne la dizaine du numéro de ligne
- U désigne l'unité du numéro de ligne

Définition de la ligne basse :

US ($t0 + D) ($10 + U)

Il n'est pas nécessaire de définir les deux lignes, si une seule est modifiée par rapport à la situation précédente.

Exemple :

Fenêtre commençant à la 5e ligne et finissant à la 15e (ATTENTION les lignes sont numérotés de 0 à 24)

1F 20 24 1F 11 14
Pour taper cette séquence il est nécessaire d'appuyer sur la touche « INS » avant chaque code.

Vous pouvez aussi utiliser le code décimal, vous taperez alors :

?.31 32 36 3t 17 20.

Si l'on veut revenir au plein écran, il suffit de définir la fenêtre de ligne haute 0 et de ligne basse 24, ce qui peut s'obtenir de la façon suivante :

?.3t 32 32 31 t8 20.

III. LA GESTION DU CURSEUR

Il vous est possible :

- de faire apparaître ou disparaître le curseur.
- de déplacer le curseur d'un espace ou d'une ligne.
- de positionner le curseur en un endroit précis.

Exemple :

Afficher le mot INFORMATIQUE, la première lettre de ce mot devant se trouver à la 10e ligne et à la 20e colonne.

? [U]. 31 73 94. 'INFORMATIQUE'

Codage résumé de la gestion du curseur

Codage clavier	codage HEXADÉCIMAL	codage DÉCIMAL	FONCTION
CNT Q	11	17	Curseur allume
CNTR	12,X	18,X	Fonction de répétition $40< = X< = $7 F
CNT T	14	20	Curseur éteint
« FG »	08	08	Déplace le curseur d'un espace vers la gauche. Si l'on se trouve en début de ligne le curseur va à la fin de la ligne précédente.
« FD »	09	09	Déplace le curseur d'un espace vers la droite Le curseur va au début de la ligne suivante s'il est en fin de ligne.
« FB »	0A	10	Déplace le curseur vers le bas. Si l'on est en mode PAGE et à la dernière ligne, le curseur revient à la première ligne
« FH »	0B	11	Déplace le curseur vers le haut. Le curseur ne bouge pas si l'on se trouve à la première ligne, mais chaque ligne descend d'un cran.
« FR »	1E	30	Le curseur revient en haut à gauche.
RAZ	0C	t2	Nettoie l'écran et le curseur revient dans le coin en haut à gauche.
ENTREE	0D	13	Retour chariot.
US	1F. y. x	31. y, x	y et x doivent rester compris entre $40 et $7 F. Le curseur est positionné en (y,x) y : numéro de ligne à partir de 0 x numéro de colonne à partir de 1 tous deux en ajoutant 40 en HEXA ou 64 en décimal.

Remarques :

Les symboles « FH », « FB », « FG », « FD », et « FR » représentent les touches que vous avez à droite sur votre clavier.

« FH » symbolise la touche ↑
« FB » symbolise la touche ↓
« FD » symbolise la touche →
« FG » symbolise la touche ←

Exemples :

Répétition du dernier caractère N fois
Ecrivons :
LSEG EDL 12 44 (n'oubliez pas la touche « INS » avant de taper 12 et 44)

On obtient :
LSEG EDLLLLL

Tapons encore :
?'LSEG EDL' .18 74.
LSEG EDLLLLLLLLLL

Positionnement du curseur

Si l'on veut positionner le curseur à la 10e ligne et à la 20e colonne, on tapera la séquence suivante :
1F 49 54

IV. POSITIONNEMENT D'UN ATTRIBUT COURANT (COULEUR - DIMENSION - VIDÉO - PAGE)

La définition de l'attribut se fait à l'aide du code ESC (que l'on obtient en tapant « INS » 18)
Chaque attribut s'obtient en tapant une séquence de codes.

IV.1 ATTRIBUT COULEUR

Il peut y avoir un attribut pour la couleur des caractères, la couleur du fond de ces caractères, et un autre attribut pour la couleur du tour.

Exemple :

ESC B ESC P ESC a LSE-EDL

permet d'obtenir :

- un tour rouge
- les lettres du mot LSE-EDL en vert sur fond noir

IV.2 ATTRIBUT TAILLE DES CARACTÈRES

Les caractéres peuvent s'afficher :

- en taille normale
- en double hauteur (le caractère envoyé est alors affiche sur 2 lignes)
- en double largeur (le caractère envoyé est alors affiché sur 2 colonnes)
- en double taille (le caractére envoyé est alors affiché sur 2 lignes et sur 2 colonnes)

Remarque :
On ne peut pas afficher des caractères en double taille sur la premiere ligne.

Exemple :

Reprenons l'exemple précedent, mais avant de taper LSE-EDL ajoutons la séquence ESC O (ESC s sur MO5)
Nous avons alors les lettres en double taille.

IV.3 ATTRIBUT VIDEO

- Inversion vidéo :

Cet attribut permet d'inverser la couleur des caractères et la couleur du fond de ces caractêres

Exemple :

Écrivons le mot VIDEO en rouge sur fond bleu ce qui s'obtient en tapant :
? ,27 65 27 84 'VIDEO'

Tapons maintenant :
? ,27 92. 'VIDEO' (?, 27 123. 'VIDEO' sur MO5)
Le mot VIDEO s'est alors écrit en bleu sur fond rouge

Pour rétablir la situation initiale vous utilisez la même séquence.

- Masquage-démasquage : (sur T07 et T07-70 seulement).
  - L'attribut masquage permet d'écrire en caractéres noirs sur fond noir, ce qui rend invisibles les caractêres tapés.
  - L'attribut démasquage permet de faire apparaître les caractères precédemment invisibles.

- Incrustation vidéo (sur T07-70 et M05 seulement) :

Si votre T07-70 (ou votre M05) est équipé de la carte d'incrustation, il vous sera possible de faire apparaître une image T.V. dans toutes les zones noires de l'écran, comme si la couleur noire devenait transparente

IV 4 ATTRIBUT PAGE

SCROLL
Décale les lignes à l'intérieur d'une fenêtre.
La premiére ligne est éliminée et toutes les lignes sont remontées afin de dégager la dernière.
Ce SCROLL peut être réalisé à deux vitesses.

MODE PAGE
Permet de gérer le déplacement du curseur du coin droit en bas vers le coin haut à gauche.
Ce mode permet d'abandonner le mode SCROLL.

IV.5 ECRITURE TRANSPARENTE

Écrivons quelques lettres en caractères rouges sur fond noir
1B 41 1B 50 AZERTY
Puis le mot ORDINATEUR en caractères verts sur fond rouge.
1B 42 1B 51 ORDINATEUR

Plaçons-nous en début d'écriture transparente en tapant :
1B 68 (tB 74 sur MO5) et déplaçons le curseur sur l'un des caractères écrits, le Z de AZERTY par exemple ; on peut alors remplacer la lettre Z par un autre caractère, celui-ci sera toujours écrit en rouge sur fond noir ; déplaçons le curseur sur une lettre du mot ORDINATEUR et remplaçons un caractère ; ce nouveau caractère sera en vert sur fond rouge.

On termine la séquence d'écriture transparente en tapant tB 69 (1B 75 sur M05)

V. DÉFINITION D'UN ATTRIBUT PLEIN ÉCRAN

Il est également possible de changer la couleur de tous les caractéres affichés à l'écran, ainsi que la couleur du fond de l'écran.

Il suffit pour cela de faire précéder le code couleur de la séquence :
1B 23 20

Exemple :

Affichez sur l'écran quelques caractères en utilisant des couleurs différentes, puis tapez :
?. 27 35 32 66.
Tous les caractères précédemment tapés seront de couleur verte.

VI. CARACTÈRES PARTICULIERS sur T07 et T07-70

Pour obtenir un caractère spécial, il faut taper la séquence suivante :
ACC / code du caractère spécial

Les graphiques obtenus ne correspondent à aucun code L.S.E. ; ils peuvent être utilisés dans des chaînes sous forme de 2 caractères que l'on pourra afficher sur l'écran.

Codage résumé des caractères spéciaux (sur T07 et T07-70)

CODAGE CLAVIER	CODAGE HEXADECIMAL	CODAGE DECIMAL	RESULTAT
ACC ≠	16 23	22 35	Livre sterling
ACC $	16 24	22 36	Dollar
ACC &	16 26	22 36	Dièse
	16 30	22 46	* Degré
ACC /	t6 2C	22 44	Flèche vers la gauche
	16 2D	22 45	* Fléche vers le haut
ACC .	16 2E	22 46	Flèche vers la droite
ACC /	t6 2F	22 47	Flèche vers le bas
	16 31	22 49	* Plus ou moins
	16 38	22 56	* Division
ACC <	t6 3C	22 60	t/4
ACC =	16 3D	22 61	1/2
ACC >	t6 3E	22 62	3/4
ACC j	16 6A	22 t06	OE DANS L'O
ACC z	16 7A	22 t22	œ dans l'o

Les codes marqués de * ne peuvent être obtenus au clavier.

VII. CARACTERES SEMI-GRPAHIQUES (sur T07 et T07 - 70)

Le T07 possède aussi un jeu de caractères permettant la création de certains dessins. Ces caractères sont appelés SEMI-GRAPHIQUES. Chaque caractère est inscrit dans un carré découpé en 6 rectangles.

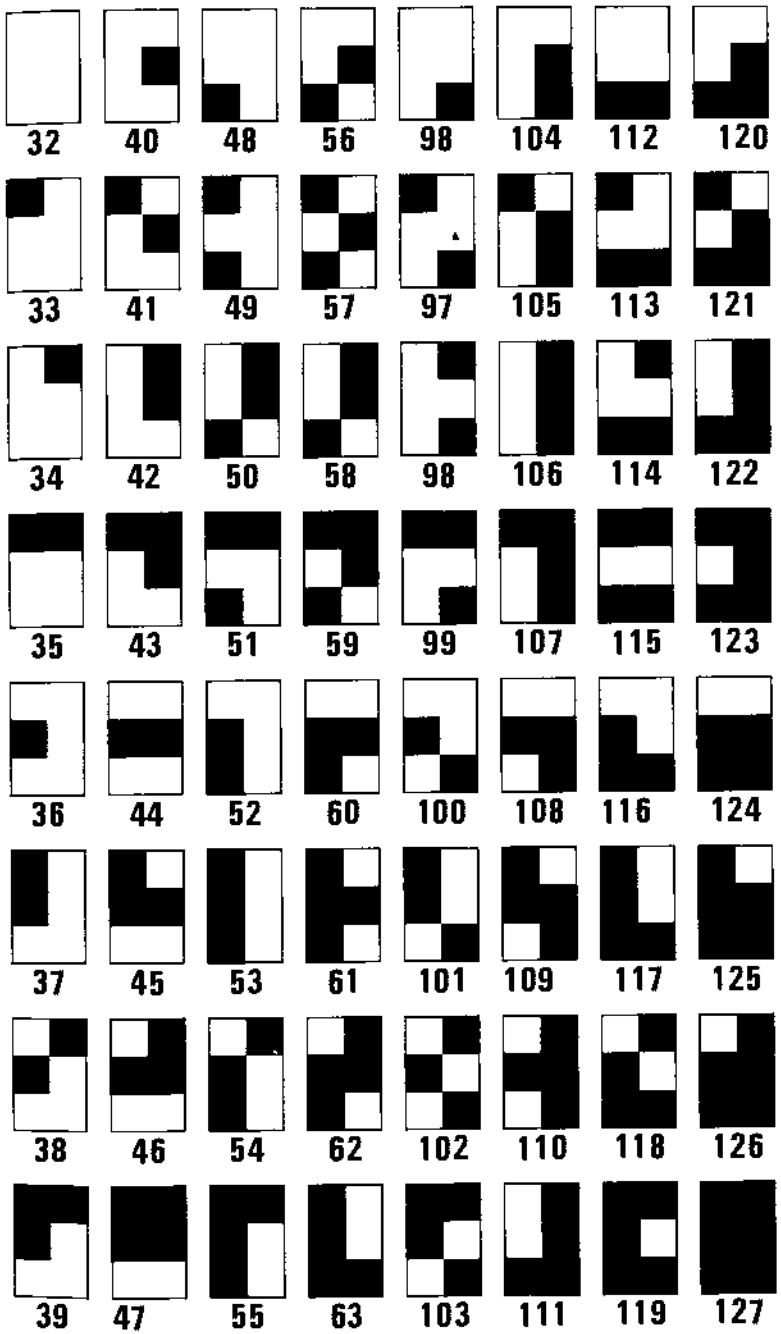
Chacun de ces rectangles peut être « allumé » ou « éteint ». Il y a donc 64 caractères semi-graphiques. Chaque caractère est désigné par son code (voir tableau page suivante).

Pour passer en mode semi-graphique le codage est le suivant :

CODAGE CLAVIER :	CNT N
CODAGE HEXADECIMAL :	0E
CODAGE DECIMAL	14

Pour revenir au jeu alphanumérique standard.

CODAGE CLAVIER :	CNT O
CODAGE HEXADECIMAL :	0F
CODAGE DECIMAL :	15



VIII. CARACTÈRES DÉFINIS PAR L'UTILISATEUR

Un caractère est codé sur un octet. On peut donc en principe en définir 256, codés de 0 à 255.
Les 127 premiers sont définis dans le code ASCII. Il est possible d'en définir 128 autres. Cependant certains de ces codes sont utilisés pour les minuscules accentuées (de 208 à 220). On ne pourra donc créer qu'un jeu de 117 nouveaux caractères. Une fois créés, ces caractères pourront être utilisés comme ceux du jeu standard.

La définition d'un tel caractère se fait de la manière suivante :

- On dessine le caractère dans une matrice carrée 8 * 8
- On marque chaque carré faisant partie du caractère
- Pour chaque ligne on calcule la valeur décimale correspondant au nombre binaire obtenu en comptant :

  1 pour les carrés marqués
  0 pour les carrés non marqués

- On introduit alors dans une chaîne les 8 nombres ainsi obtenus. Les nombres sont séparés par des blancs, le premier nombre écrit est celui de la ligne la plus basse de la matrice, le dernier celui de la plus haute. Les huit nombres sont encadrés par des points.
- L'utilisation d'une procédure binaire est alors nécessaire pour faire le lien avec le moniteur.

Le premier caractère ainsi créé sera désigné par le nombre 128, le deuxième par 129..... Ces caractères peuvent être alors utilisés dans l'instruction AFFICHER.

Exemple :

			*	*				24
		*			*			38
		*			*			38
	*	*			*	*		101
		*	*	*	*	*		82
*		*			*		*	184
*		*			*		*	185
	*					*		66

Supposons que l'on dispose de la procédure binaire (donnons lui le nom MOU)
Le petit programme suivant permettra alors d'afficher le caractère dessiné dans la grille.

```
1 * Dessin du A
10
20 CHAINE A : A ← . 66 165 164 62 101 36 36 24.
30 MOU (A)
40 AFFICHER [U]  128 .
50 TERMINER
60
500 PROCEDURE BINAIRE MOU
```


ANNEXE VIII
CODES L.S.E. DES MINUSCULES ACCENTUÉES

206 â	209 : à	210 : ç	211 é
212 · ê	213 : e	214 : e	215 : î
216 : ı	217 : ô	218 : û	219 . ù
220 : u			

Nota : Les codes de 128 à 207 et de 221 à 255 sont disponibles pour la définition de caractères par l'utilisateur (voir le programme CRMOU du C.N.D.P).

ANNEXE IX
INSTRUCTIONS SUR DISQUE AVEC COMPTE RENDU NEGATIF

Compte rendu	Charger	Exécuter	Supprimer	Garer
– 1	enregistrement inexistant			
– 2	fichier inexistant			
– 3	fichier protégé en lecture		fichier protégé en écriture	
– 4	nom de fichier incorrect			
– 5	manque de place			disque plein
– 6	erreur disque			
– 7	type incorrect			
– 8	catalogue incorrect			
– 9	unité inexistante			
– 10				catalogue plein
– 11				déjà 84 enregistrements

INDEX

Les mots cles sont en majuscules, ainsi que les deux premiers caractères des commandes. Le classement est fait par ordre alphabétique du mot clé isolé. Ainsi Lister est avant LIBERER

Les références concernent des numéros de pages.

N° d'Éditeur : 6565 – Dépôt légal Février 1985

Imprimerie REGAZZI 77 Thorigny

COMPLÉMENT LSEG - EDL

Afin d'augmenter encore la compatibilité entre T07 et M05, nous avons donné à chaque machine la possibilité d'utiliser les deux types de LEP (lecteur enregistreur de programme). Cela nous a conduit à ajouter une commande au LSEG-EDL : la commande CA.

Utilisation de la commande CA

Le complément de commande est ssette. On doit donner ensuite un numéro :

0 pour utiliser un LEP de T07.
1 pour utiliser un LEP de M05.

Le paramètre correspondant est initialisé par défaut à 0 pour T07 et à 1 pour M05. Ainsi sans rien faire vous utiliserez sur T07 un LEP de T07 et sur M05 un LEP de M05.

Exemples d'utilisation :

- vous avez déjà un équipement de T07 avec des LSEG-EDL et vous lui ajoutez des M05 avec des LSE. Il ne sera pas nécessaire d'avoir deux types de LEP et donc deux types de cassettes ce qui est intéressant pour les mises à jour.

- vous avez déjà un equipement de T07 avec un coupleur CL07 (voir bulletin EPI de Mars 84) et vous lui ajoutez des M05. Il suffira en LSEG-EDL, après avoir dit Bonjour, de taper Cassette 0 et le M05 se comportera comme un To7 vis à vis du coupleur.

- vous achetez des T07, des T07-70, des M05, des LSEG-EDL version 3.2 et un coupleur CLD75 qui permet de communiquer au format T07 ou au format M05. Avec cet équipement, il vous sera possible de communiquer entre toutes les machines au format M05 à condition de taper sur les T07 et T07-70 la commande Cassette 1. Les transferts se font ainsi à 1200 bauds au lieu de 900 au format T07.

Bien noter dans ces exemples que, contrairement à ce qui se passe pour d'autres langages, vous n'avez pas besoin de charger au préalable un utilitaire qui reduirait votre espace de travail.