III. INSTRUCTIONS DE STRUCTURATION

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

INSTRUCTION FAIRE .... JUSQUA ...

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

N1 et N2 sont des numéros de ligne (N2 ⩾ N1)
I est une variable numérique 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 égal à 1

Au premier passage à la ligne N1, la variable I prend la valeur (ea1). Le pas et la valeur finale sont alors évalués :
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 négatif
ALORS
on sort de la boucle et l'exécution se poursuit à la première ligne dont le numéro est supérieur à N2
SINON
le traitement situé entre les lignes N1 et N2 (incluses) est exécuté. On revient à 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 inférieur à 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 évaluée :
SI
eb a la valeur . FAUX . on sort de la boucle et l'exécution se poursuit à partir de la première ligne dont le numéro est supérieur à N2.
SINON
le traitement situé entre les lignes N1 et N2 est exécuté, puis on revient à 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 début de ligne, et ne peut être exécutée. 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, séparés par des virgules.
Cette instruction ne peut apparaître que derrière la déclaration d'une procédure.

Tout élément 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 procédure 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 exécution 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, ea1

(ea) est toujours le numéro de la ligne de retour. ✳ signifie que le programme appelant reprend la main au niveau d'appel 0.
ea1 indique que le niveau d'appel est diminué de (ea1) unités (RETOUR EN ea,1 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 procédure 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 où 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 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 :
a) l'opération est possible, ce qui est fait,
b) l'opération 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 à 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 ambiguïté 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 après que le chargement ait commencé provoquera une erreur d'exécution (se reporter à l'Annexe IX).
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. Cette instruction fait disparaître de la mémoire 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 1 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, ea1, 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 caractères 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 côtés étant de la couleur courante au moment de l'affichage.

FONCTIONS

- FONCTIONS ARITHMÉTIQUES
- FONCTIONS CHAÎNES
- FONCTIONS GRAPHIQUES
- FONCTIONS SYSTÈME

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.

Résultat :
Logarithme népérien de (ea).

Exemples :

```
? LGN (1)
0
? LGN (3)
1.0986123
? 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
19 12
? CNB ('12 AVRIL' ,9,P) ,P
0 9
? CNB ('25 – 13',1)
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 caractères contenus dans la chaîne (ec).

Exemples :

```
? LGR ('FRANÇOIS')
8
? LGR ('HAUTE' !'–' !'garonne')
13
? 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 égal à la longueur de (ec1) + 1
(ea2) doit être positif ou nul.

Résultat :
Chaîne 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 caractères 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 guère les pois
? MCH (X,19,0,'petits')
Je n'aime pas les petits pois
X ← 'Ces fleurs sont très fraîches'
? MCH (X,16,5,'')
Ces fleurs sont fraîches
```


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 chaîne (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 chaîne (ec1), à partir de la position (ea), du premier caractère qui appartient à (ec2).
Si, à partir de la position (ea), aucun caractère de (ec1) n'est dans (ec2), le résultat est égal à la longueur de (ec1) + 1.

Exemples :

```
? PTR ('J''AIME LA MER BLEUE' ,1 , 'BLEUE')
6
? PTR ('Jeudi 18 mai 1983' ,6)
13
? PTR ('ABCD ; 12' ,3 ,'XY')
8
? PTR ('ABCD123' ,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 chaîne 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 accentuée) 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 repère 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)

Résultat :
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.

Résultat :
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
Exemples :

```
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 boolénne 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 à 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 numéro 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 mélange 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 chaine
C 29 nombre de paramètres incorrect
C 30
C 31
C 32 numéro 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 spécification 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 paramètres identiques dans une déclaration de procédure
C 48 double déclaration de procédure ou noms de procédures paramètres identiques ou nom de procédure paramètre égal 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 déclaration de procédure avec paramètre formel incorrect ou) absente
C 52
C 53 instruction incorrecte
C 54 plus de spécifications que de variables
C 55
C 56 ligne codée trop longue
C 57 zone utilisateur pleine : provoquer un tassage de la CPF
C 58 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 référence de plus de 1 caractère
C 66
C 67 erreur au chargement d'une procédure binaire
C 68 ligne de déclaration de procédure binaire contenant d'autres instructions ou un commentaire
C 69 numéro de voie d'entrée/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 chaîne 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 opérande 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 15 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 supérieur à 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 paramètres d'une procédure binaire différents à l'appel et à la déclaration
E 37 numéro de ligne inférieur à un
E 38 nombre non convertible en entier
E 39 numéro de ligne de fin de boucle inférieur au numéro de ligne de début 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 népérien d'un nombre négatif ou nul
E 43 variable de contrôle de boucle non numérique
E 44 débordement d'un nombre flottant
E 45 déclaration d'un tableau trop grand
E 46 accès 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 numéro 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 63 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 donnée 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 procédure externe
E 72 utilisation d'un paramètre effectif au lieu d'un paramètre formel
E 73
E 74 RETOUR EN dans une procédure 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 écriture
E 78 élément à affecter ni nombre ni chaîne ni booléen ni forme
E 79 motif en mémoire différent de celui qui est chargé
E 80 trop d'imbrications de motifs référencés
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 protégé
F 13 ✳ bloc de contrôle de fichier illégal
F 14 ✳ adresse disque illégale
F 15 numéro de disque illégal
F 16 disque non prêt
F 17 accès refusé à un fichier protégé en lecture
F 18 ✳ 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 évolué.

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'entrée 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 16 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.

PSH16 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 booléen 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 (1 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 différence entre FA et FB

FLMUL effectue le produit de FA par FB avec résultat 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 caractère a été tapé 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 LSE-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 défauts rencontrés lors de l'exécution de la procédure.

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

6. 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'entrée précédent.

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 numéro 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. Le 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
0118 PSHBL      RMB 3 le booléen B = 0 ou 1 est mis dans la pile
011B PSHFO      RMB 3 la forme pointée 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 = type 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 personnalisées 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
0139 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!!
0148 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 numéro
                      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 procédure 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 étant un branchement relatif court, cela impose une longueur maximum de 126 octets à la séquence qui suit.

– L'octet 2 contient le numéro de version de la procédure binaire. L'interpréteur ne se sert pas de cet octet qui n'est là que pour assurer la comptabilité 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 procédure binaire. C'est le seul moyen dont dispose le LSEG-EDL pour connaître la taille exacte de la procédure 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.

– L'octet 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 procédure 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 procédure 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'interpréteur 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 été prévue... 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 éléments 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 même erreur d'exécution est renvoyée.

– L'octet 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 procédure binaire doit avoir une structure de sous-programme et donc se terminer par un RTS. Le code généré doit être tel que la procédure 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 résultat 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
début   bra  deb        vers première instruction
        fcb  1          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 chaîne (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 :

```
10 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 SYSTÈME :

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'exécution de l'instruction.

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

– au départ on affecte 0 à N puis on exécute 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 présentent :
  – le fichier contient encore un enregistrement, alors l'enregistrement suivant est chargé dans A, l'opposé du numéro de cet enregistrement est mis dans N.

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

Exemple :

Soit à 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 à chacune des trois machines. Quelques différences existent entre ces matériels; certaines options n'existent pas sur tous. Chaque fois que cela est nécessaire, il est précisé 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'intérieur, 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 précédente (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 séquence 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 », 1 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.

Définition 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 ($10 + 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 31 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 :

?.31 32 32 31 18 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 allumé
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	12	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 caractères peuvent s'afficher :

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

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

Exemple :

Reprenons l'exemple précédent, 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 VIDÉO

- Inversion vidéo :
Cet attribut permet d'inverser la couleur des caractères et la couleur du fond de ces caractères.

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 caractères noirs sur fond noir, ce qui rend invisibles les caractères tapés.
  - L'attribut démasquage permet de faire apparaître les caractères précédemment 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 première 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 (1B 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 1B 69 (1B 75 sur M05)

V. DÉFINITION D'UN ATTRIBUT PLEIN ÉCRAN

Il est également possible de changer la couleur de tous les caractères 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 38	Dièse
	16 30	22 48	* Degré
ACC /	16 2C	22 44	Flèche vers la gauche
	16 2D	22 45	* Flèche vers le haut
ACC .	16 2E	22 46	Flèche vers la droite
ACC /	16 2F	22 47	Flèche vers le bas
	16 31	22 49	* Plus ou moins
	16 38	22 56	* Division
ACC <	16 3C	22 60	1/4
ACC =	16 3D	22 61	1/2
ACC >	16 3E	22 62	3/4
ACC j	16 6A	22 106	OE DANS L'O
ACC z	16 7A	22 122	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.

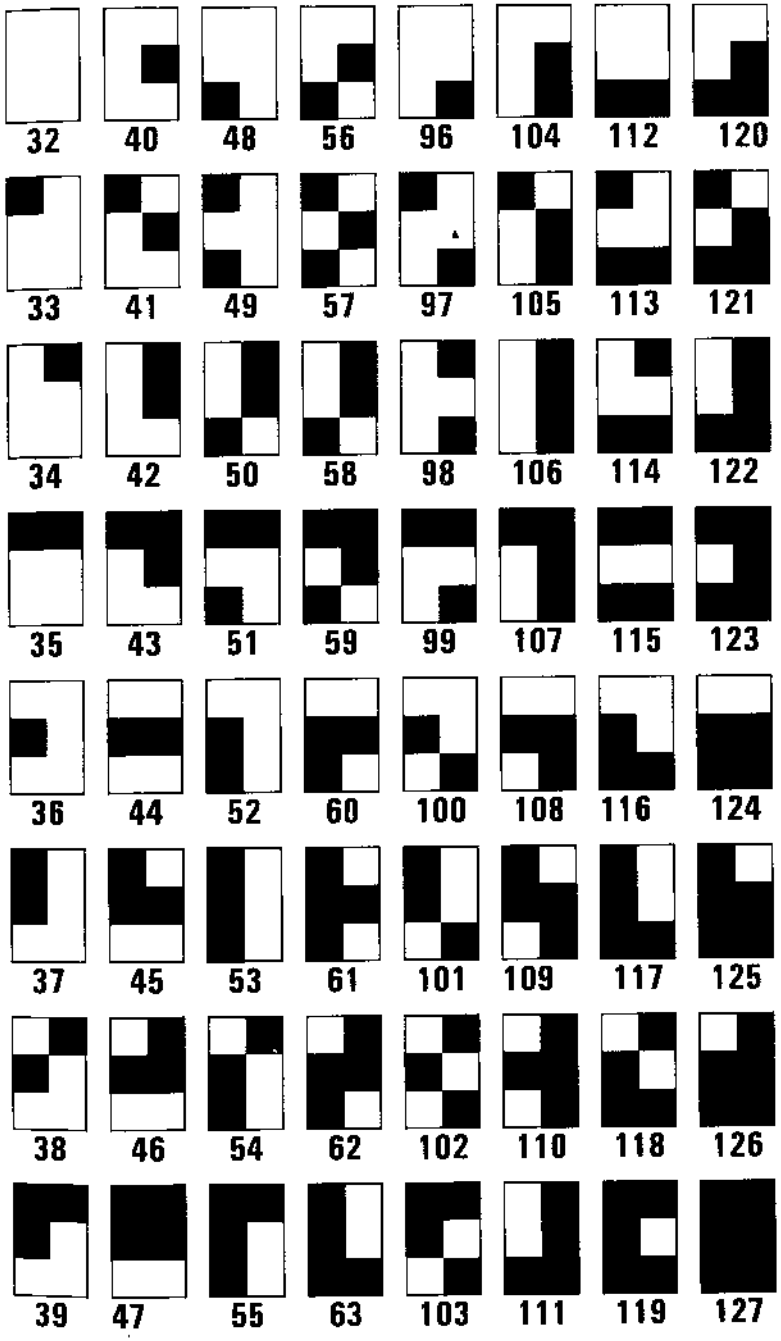
Chacun de ces rectangles peut être « allumé » ou « éteint ». Il y a donc 64 caractères semi-graphiques. Chaque caractère est désigné par son code (voir tableau page suivante).

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
		*			*			36
		*			*			36
	*	*			*	*		101
		*	*	*	*	*		62
*		*			*		*	164
*		*			*		*	165
	*					*		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

208 : â	209 : à	210 : ç	211 : é
212 : ê	213 : è	214 : ë	215 : î
216 : ï	217 : ô	218 : û	219 : ù
220 : ü			

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 clés 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 équipement 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 réduirait votre espace de travail.