SMC

Manuel d'utilisation

LACTAMME

Octobre 1979

Sommaire du manuel

INTRODUCTION

Ce manuel a pour but de faciliter la formation des utilisateurs du système SMC (Système Multimédia Conversationnel). C'est donc un document pratique et non pas théorique. Son contenu est, à grands traits, le suivant :

- *présentation de la configuration informatique et audiovisuelle mise à la disposition de l'utilisateur ;*

- *documentation externe du système d'exploitation CMS, support du système SMC. Cette documentation externe du système d'exploitation ainsi que des divers programmes utilisables (assembleur, éditeur de fichiers, programmes utilitaires) s'adresse à ceux qui, ayant de bonnes connaissances en informatique, voudraient écrire eux-mêmes des programmes spécifiques. L'utilisateur non-informaticien n'est concerné que par le chapitre "Control cards interpreter" de cette partie ; il peut en ignorer le reste, car l'utilisation de SMC ne requiert pas de compétence en informatique de la part de l'utilisateur ;*

- *mode d'emploi du système SMC proprement dit, utilisable par tous, notamment par le non-informaticien. Ce mode d'emploi, très détaillé, contient de nombreux exemples facilitant la compréhension.*

Il convient de faire ici les deux remarques suivantes :

1 - Ce manuel d'utilisation n'est qu'un élément de la formation de l'utilisateur. En plus du manuel, celle-ci exige en effet :

- *une assistance de l'utilisateur débutant par un utilisateur expérimenté, celui-ci faisant profiter celui-là de son expérience;*

- *l'acquisition par l'utilisateur d'une expérience personnelle découlant de sa pratique du système.*

2 - Les exemples fournis tout au long de ce manuel ne peuvent donner qu'un aperçu des possibilités offertes par SMC, et ne sont donc pas exhaustifs. C'est l'utilisateur lui-même, qui devra explorer les possibilités du système, et qui cherchera à en tirer le meilleur parti.

Nota : Un certain nombre de conventions sont à connaître pour comprendre la notation utilisée dans ce manuel. Il convient donc de se reporter à l'annexe intitulée "conventions générales" afin d'en prendre connaissance.

Un certain nombre de nouveautés sont apparues dans certains processeurs de CMS et de SMC après la rédaction de cette documentation. Ces éléments supplémentaires ont été ajoutés sous forme d'additifs que l'on trouvera à la fin des processeurs concernés. C'est pourquoi il est conseillé de parcourir l'ensemble du mode d'emploi d'un processeur avant de l'utiliser, afin de prendre connaissance de ces additifs.

1ere PARTIE

CONFIGURATION INFORMATIQUE

ET AUDIOVISUELLE

Le système SMC est composé d'un ensemble de ressources en matériel informatique et audiovisuel, mis à la disposition des utilisateurs par l'intermédiaire de postes de travail, la coopération entre l'utilisateur et le système se faisant au travers d'un logiciel conversationnel, permettant le dialogue utilisateur-système.

1 - LES RESSOURCES INFORMATIQUES ET AUDIOVISUELLES.

Se reporter au schéma de la page suivante dont nous donnons ici un commentaire.

Les ressources informatiques et audiovisuelles sont présentées de façon symétrique :

- partie gauche : le système informatique est organisé autour d'un miniordinateur T1600, 32K mots de 16 bits de mémoire centrale, équipé de mémoires numériques (disques magnétiques à têtes fixes), d'une périphérie conventionnelle (lecteur de cartes, perforateur de cartes, imprimante, consoles de visualisation alphanumériques - graphiques), et il est éventuellement relié à d'autres systèmes informatiques par ligne téléphonique.

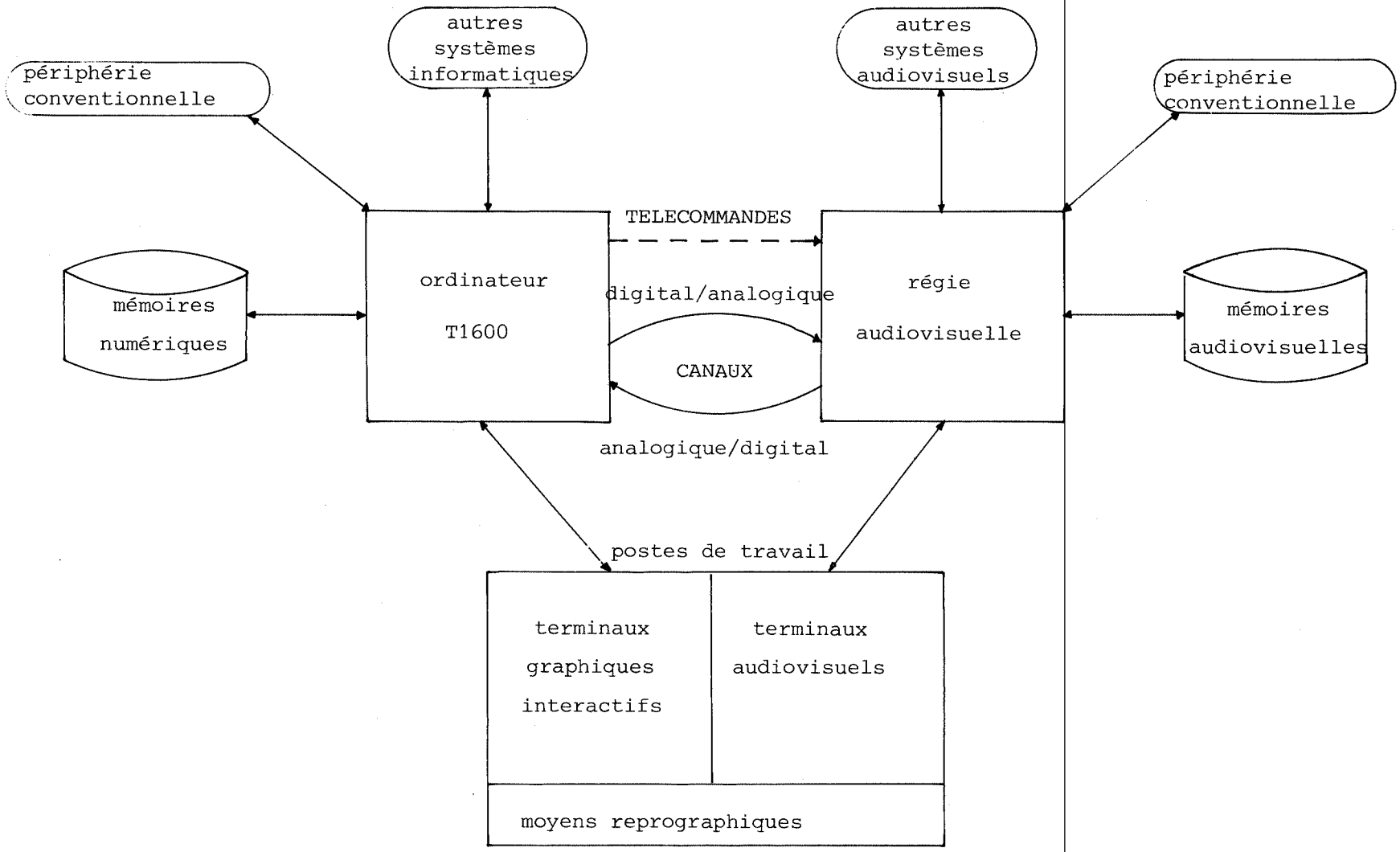
- partie droite : le système audiovisuel, est représenté d'une manière semblable. Il est organisé autour d'une régie audiovisuelle , à laquelle sont connectées des mémoires dites audiovisuelles (par analogie avec les mémoires numériques) qui sont en fait des magnétoscopes (dont on verra qu'ils sont adressables et télécommandables par l'ordinateur et que par conséquent ils peuvent de ce fait, être assimilés à des mémoires auxiliaires) ; une périphérie conventionnelle (caméras vidéo, moniteurs de TV) et éventuellement, cette régie peut être reliée à d'autres systèmes audiovisuels.

Les deux systèmes, système informatique et système audiovisuel, sont interconnectés de deux façons :

- par une télécommande : elle agit dans le sens ordinateur ⟶ régie ⟶ magnétoscopes ; l'ordinateur peut, d'une part assurer automatiquement les commutations sur la régie entre les canaux d'entrée et les canaux de sortie et, d'autre part, télécommander les magnétoscopes. En effet, il peut effectuer automatiquement sur n'importe quel magnétoscope toutes les opérations possibles ; démarrage, arrêt, diffusion, avance rapide, rembobinage rapide, le tout étant accompagné de la notion d'adresse physique sur la bande magnétoscope. Ainsi, on peut désigner à l'ordinateur une séquence audiovisuelle en lui spécifiant son adresse de début et son adresse de fin et l'ordinateur sera capable de rechercher et de diffuser cette séquence automatiquement. Nous verrons dans la présentation du système SMC (3ème partie du manuel), comment définir par exemple un enchaînement automatique de séquences.

- par deux canaux, analogique ⟶ digital et digital ⟶ analogique. Ces deux canaux permettent l'analyse et la synthèse d'images et de sons.

- analyse ; une image vidéo, en provenance d'une caméra vidéo par exemple, et transitant par la régie, peut être numérisée et rentrée en mémoire de l'ordinateur. A ce stade, on peut lui faire subir n'importe quel traitement informatique et notamment la stocker sur des mémoires numériques (disques magnétiques, cartes perforées, etc ...).

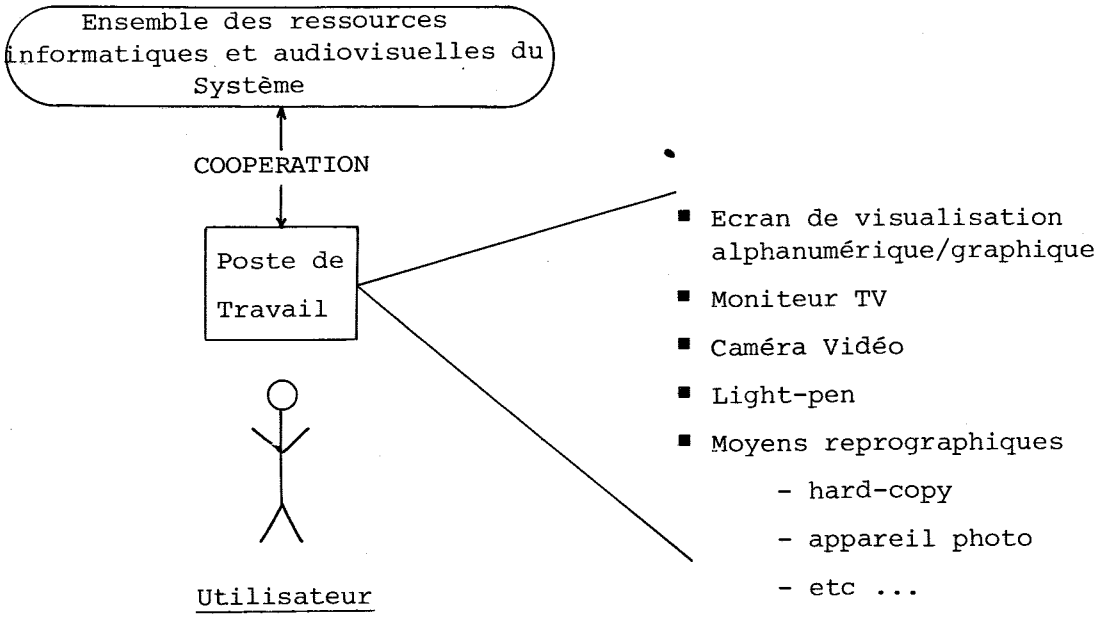

périphérie conventionnelle
autres systèmes informatiques
autres systèmes audiovisuels
périphérie conventionnelle
TELECOMMANDES
mémoires numériques
ordinateur T1600
digital/analogique
CANAUX
régie audiovisuelle
mémoires audiovisuelles
analogique/digital
postes de travail
terminaux graphiques interactifs
terminaux audiovisuels
moyens reprographiques

- synthèse ; une image numérique, placée en mémoire de l'ordinateur, peut être renvoyée sur la régie audiovisuelle où elle devient (après conversion numérique/analogique) une image vidéo dont on peut faire ce qu'on veut, par exemple la visualiser sur un moniteur de télévision.

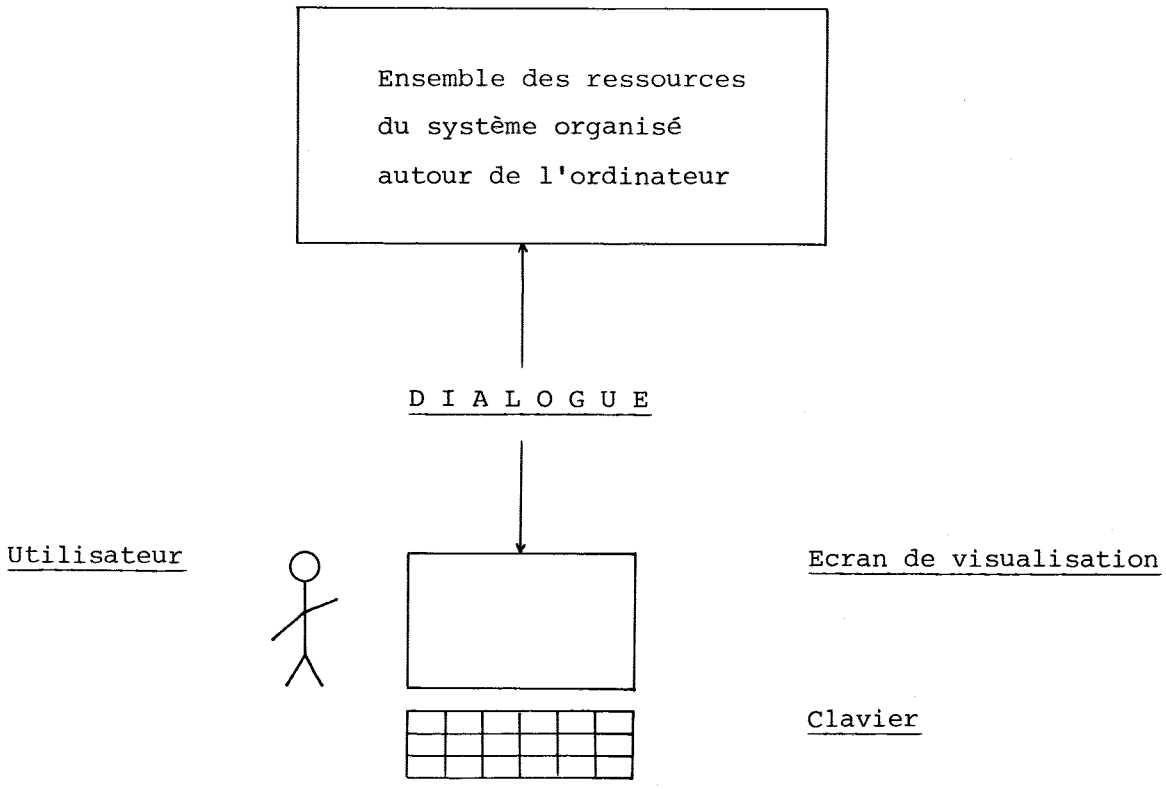
Par le moyen de la télécommande et des canaux de conversion analogique/digitale et digitale/analogique, les deux systèmes informatique et audiovisuel, sont parfaitement coopérants. On peut donc les utiliser conjointement pour élaborer des documents audiovisuels à partir de sources multiples, vidéo ou digitale. On voit qu'il est par exemple possible d'introduire dans le système, par l'intermédiaire d'une caméra, un certain nombre d'images appartenant à un mouvement, de stocker ces dernières sur les disques numériques, de les traiter à l'aide de programmes appropriés appartenant à SMC et décrits dans ce manuel, et enfin de les restituer en vue d'un stockage analogique sur magnétoscope ou bien d'une diffusion sur moniteur de télévision. Ainsi, l'ensemble des ressources intégrées dans le système SMC autorise aussi bien des activités de type réalisation (création et trucage d'images, montage électronique de bandes vidéo, enregistrement d'émissions etc ...), que des activités de type diffusion (dont l'enseignement assisté par ordinateur est une application).

2 - POSTE DE TRAVAIL ET DIALOGUE AVEC LE SYSTEME.

Le poste de travail est l'interface entre l'utilisateur et le système. En d'autres termes, il est le moyen permettant à l'utilisateur de disposer de l'ensemble des ressources informatiques et audiovisuelles du système.



La coopération entre l'utilisateur et le système se développe de manière conversationnelle au travers d'un dialogue homme-machine. Ce dialogue se fait en général en utilisant un écran de visualisation ; pour qualifier ce type de dialogue, on emploie le terme d'"utilisation interactive" du système, ou d'utilisation "en mode conversationnel" ; par opposition au mode "batch" ou "traitement par lots" également utilisable.



Utilisation du système en mode conversationnel.

En mode "batch" ou "traitement par lots", les choses se passent de manière différente :

- les commandes ne sont pas rentrées par un utilisateur travaillant sur une console de visualisation, mais elles sont sur cartes perforées et lues par l'ordinateur sur le lecteur de cartes.

- les commandes et les réponses du système sont listées sur imprimante.

Des précisions sur la façon de travailler en mode batch sont données dans le chapitre "Control Cards Interpreter" dans le deuxième partie du manuel (CMS).

Comme tout dialogue, celui qui se développe entre l'utilisateur et le système comporte les trois éléments suivants :

1 - début du dialogue (ou "Logon" ou "bonjour")

2 - corps du dialogue

3 - fin du dialogue (ou "Logoff" ou "Au revoir")

Les choses se déroulent, concrètement, de la façon suivante :

L'utilisateur s'installe à son poste de travail, comportant comme nous l'avons vu : écran de visualisation alphanumérique et graphique (pour simplifier on parlera par la suite de "visu"), moniteurs de TV, caméra vidéo, light-pen, moyens reprographiques). La visu est le support du dialogue utilisateur-système lors d'une utilisation du système en mode conversationnel.

Début du dialogue (ou "Logon").

Cette phase du dialogue est éventuellement précédée d'une mise sous tension de la visu utilisée :

- mettre la visu sous tension (bouton ON/OFF) sur la position "ON". Ici, l'écran devient progressivement lumineux ; lorsqu'il est totalement lumineux, faire "PAGE" ; cette fonction est locale à la visu.

- s'assurer que le sélecteur "LOCAL/LINE" est sur la position "LINE" (visu "on-line")

- faire "Alt-Mode". Cette touche a pour effet de signaler à l'ordinateur la présence de l'utilisateur. Le système répond alors sur l'écran.

UCPRS - CMS4 IS UP
07/04/79 - 11/53/50
? !L Ø : SYS D^C
! WELCØME
07/04/79 - 11/53/52
? mot de passe D^C
?

Le système est prêt ; on peut donc faire "Logon" : Commande "!L" : Logon, suivie du numéro de compte (ou ACN : account number). :SYS est un numéro de compte parmi d'autres, celui qui offre les plus larges possibilités. Son utilisation requiert un mot de passe (voir CMS).

A ce stade, le dialogue avec le système est établi. On a à sa disposition les ressources du système.

Corps du dialogue :

Le dialogue étant ouvert par le Logon, on peut alors le poursuivre à plusieurs niveaux :

- le premier niveau consiste à dialoguer avec le système d'exploitation CMS, et il se caractérise par le fait que l'ordinateur affiche un point d'interrogation sur l'écran.

- les niveaux suivants, hiérarchiquement inférieurs, sont tous les dialogues possibles avec des programmes spécialisés auxquels on a pu accéder à travers le système d'exploitation CMS.

On peut comparer cette procédure à un appel téléphonique :

- on décroche, on compose un numéro et l'on obtient le standard d'une entreprise : c'est le premier niveau de dialogue ; dans notre cas, c'est le dialogue avec le système d'exploitation CMS.

- on demande la personne à laquelle on veut parler d'un problème particulier : c'est le second niveau de dialogue : celui que l'on développe avec un programme spécialisé.

- à ce stade, ayant terminé la conversation avec cette première personne, on demande à celle-ci de nous redonner le standard. Ce qu'elle fait. C'est ce qu'on appelle le retour au système d'exploitation, ou plus simplement, le retour au niveau "?" puisque c'est sous ce "?" que l'on rentre des commandes à CMS.

- ayant terminé on raccorche ; c'est le Logoff (ou fin de dialogue).

Fin de dialogue (ou "Logoff").

C'est l'opposé de la commande !L, logon. Elle s'utilise de la manière suivante :

? !F D^{C}

!HI

07/04/79-15/28/34

Elle clôt le dialogue utilisateur-système. Un nouveau dialogue ne peut être établi qu'en reprenant la procédure de Logon.

On trouvera page suivante un exemple de dialogue.

DIALOGUE AVEC CMS

```
@!!UCPRS-CMS5 IS UP
 07/05/1979-11/53/52

?!L :SYS
!WELCOME
 07/05/1979-11/53/55

?
?!DATE 07/05/1979-11/54/02

?!GE                    SOUS-DIALOGUE AVEC SMC
 SMC VOUS SOUHAITE LA BIENVENUE!!
 07/05/1979-11/54/05

*G
!I+
ITEM>ITEM ESSAI
!I-
ITEM>ITEM ESSAI
!FF
?!F
   !HI
 07/05/1979-11/54/43
```

DIALOGUE AVEC CMS

Alt-Mode

? ! Logon

? ! demande de la date

? ! appel du système SMC — SOUS-DIALOGUE AVEC SMC

* appel du sous-niveau ! de SMC
! ! dialogue niveau !
! ! demande de retour à CMS

? ! Logoff

Exemple de dialogue utilisateur - système.

2eme PARTIE

C M S

COLONNA MONITOR SYSTEM

C M S

1 - GÉNÉRALITÉS.

2 - CARTES DE CONTRÔLE.

3 - PROCESSEURS DE BASE.

4 - AIDES À LA MISE AU POINT.

5 - INTERFACE UTILISATEUR-SYSTÈME.

6 - LES PROCESSEURS.

EDITS	EDITEUR DE FICHIERS
ASSYS	ASSEMBLEUR
LINK	EDITEUR DE LIENS
DUMP	DUMP
REST	RESTAURATION
CALCUL	CALCUL

1 - GÉNÉRALITÉS.

CMS est le système d'exploitation implanté sur les ordinateurs T1600 et SOLAR. On a coutume de le désigner par :

- CMS4 lorsqu'on parle du système d'exploitation T1600.

- CMS5 lorsqu'on parle du système d'exploitation SOLAR.

CMS5 est en réalité une version améliorée de CMS4 et adaptée à l'ordinateur SOLAR. Vu du côté utilisateur, ces deux systèmes sont semblables à quelques exceptions près, c'est pourquoi cette documentation est utilisable que l'on travaille sur T1600 ou SOLAR. Par la suite, de nouvelles extensions du système SOLAR donneront lieu à une documentation spécifique.

Nota : Dans la suite de la documentation, on emploie indifféremment les termes CMS4 et CMS.

CMS réalise les fonctions "classiques" d'un système d'exploitation en temps partagé (gestion espace mémoire, gestion espace disque, gestion des périphériques, gestion de bibliothèques, etc...). De plus, il supporte le système SMC qui fait l'objet d'une description spécifique dans ce manuel.

L'utilisation du système se fait en temps partagé ou en batch, au travers de commandes, de processus de base et de programmes utilisateurs, ces termes étant ainsi définis :

Les commandes (ou "cartes de contrôle").

Elles permettent à l'utilisateur d'effectuer des opérations simples (logon, assignation, etc...). Elles sont assimilables à ce que l'on appelle en général des "cartes de contrôle", et sont présentées dans le chapitre "Cartes de contrôle".

Les processeurs de base.

Ce sont des programmes - système permettant de réaliser des opérations plus complexes, et pouvant assurer un dialogue avec l'utilisateur. (Delete de fichier, appel d'un programme - utilisateur, lancement d'un programme utilisateur, appel du système SMC, etc...). Ces processeurs de base sont présentés dans le chapitre "Processeurs de base".

Les programmes - utilisateur (ou processeurs).

Seuls les plus importants sont présentés dans ce manuel : l'éditeur de fichier EDITS, l'assembleur ASSYS, l'éditeur de liens : LINK, et quelques programmes utilitaires. Il sont présentés dans le chapitre "processeurs".

Le système SMC.

Il est supporté par CMS et appelable en utilisant le processeur de base :!GE. Il donne lieu à une documentation spécifique dans la partie "Système SMC".

2 - CARTES DE CONTRÔLE.

Les cartes de contrôle sont des commandes interprétées par CMS4 , plus précisément par le CCI (Control Cards Interpreter) appartenant à CMS4 . Elles peuvent se présenter :

- sous forme de cartes perforées si l'on travaille en batch

- sous forme de caractères frappés par l'utilisateur sur une visu, si l'on travaille en mode conversationnel.

!L Logon

!L ∅ n° de compte D^C

L'utilisation de !L suppose que l'on se soit préalablement "connecté" au système, comme il a été indiqué précédemment (cf : dialogue avec le système). C'est-à-dire :

- en mode conversationnel : visu sous tension et "on line", on fait Alt-Mode, on obtient :

UCPRS - CMS4 IS UP	le système est prêt ;
07/05/79 - 11/53/52	il donne la date et l'heure
?	

- en mode batch (ou traitement par lots), le lecteur de cartes est prêt, avec un train de cartes dont la première est la carte !L (Logon). On simule l'envoi d'un Alt-Mode en utilisant la commande !X décrite ci-après.

Exemple de Logon :

? !L ∅ : SYS D^C

! WELCOME

07/05/79-11/52/03 (L'unité des minutes sera utilisée ensuite pour le mot de passe).

? mot de passe

? (Le mot de passe n'est pas visualisé sur l'écran).

Le numéro de compte (ou ACN : Account Number) est obligatoire (:SYS dans l'exemple ci-dessus). Si l'on utilise :SYS, celui qui offre les plus larges possibilités, un mot de passe est requis, à entrer immédiatement après la réponse du système à la commande !L. Si le numéro de compte n'est pas :SYS, aucun mot de passe n'est requis.

Actuellement, les ACN connus du système sont : :SYS, :EAØ et :USE.

Mot de passe :

- en mode conversationnel : C^{C} M^{CS} unité des minutes P^{C} D^{C}

- en batch : C^{C} M^{CS} D^{C}

Le mot de passe en batch se perfore sur carte comme suit :

$C^{C} \equiv$ 12-8-1 (perfo multi)
$M^{CS} \equiv$ 12-8-2 (perfo multi)
$D^{C} \equiv Z^{S}$

!F Logoff

! F D^{C}

C'est l'opposé de la commande Logon ; elle a pour effet de terminer la session. Elle assure la sauvegarde des fichiers de l'utilisateur et la désassignation des unités logiques.

? ! F D^{C}

! HI
07/05/79-12/56/32 Le système affiche la date et l'heure de la fin de la session.

! BYE

!BYE D^{C}

Cette commande est équivalente à !L (logoff) mais n'est acceptée que si les unités logiques 3 à 9 sont désassignées. Les deux séquences suivantes sont équivalentes :

! F D^{C} — ! CLØSE D^{C}
! BYE D^{C}

! DATE Date et heure

! DATE D^C

Permet d'obtenir la date et l'heure :

? ! DATE D^C 10/04/79 12/28/34

?

! ASSIGN : Assignation.

L'utilisateur dispose de 12 unités logiques numérotées de 1 à C (1,2,...,9,A,B,C). Les unités 1,2 et C sont automatiquement assignées par le système :

1 à l'organe d'entrée.

2 à l'organe de sortie.

C à une zone disque mise à la disposition de l'utilisateur (on parlera de zone "Scratch" ou "de travail").

On entend par organe d'entrée et organe de sortie :

- en mode conversationnel : la visu utilisée pour le dialogue avec le système : les commandes sont entrées par l'utilisateur sur cette visu, et les réponses du système sortent sur cette même visu.

- en mode batch : l'organe d'entrée est le lecteur de cartes, et l'organe de sortie est l'imprimante ; les ordres et les réponses apparaissent sur le listing.

Il reste donc à l'utilisateur les unités logiques 3 à B dont il peut disposer librement.

■ *Assignation d'organe d'Entrée/Sortie.*

! ASSIGN ∅ n = { I / ∅ } D^C	avec $3 \leqslant n \leqslant B$

I signifie : organe d'entrée (visu en conversationnel, lecteur de cartes en batch).(Input).

∅ signifie : organe de sortie (visu en conversationnel, imprimante en batch).(Out put).

■ *Assignation de périphérique physique.*

Utilisable seulement avec ACN = :SYS (sauf CR2 utilisable quel que soit l'ACN).

! ASSIGN ⌀ n = { CR1 | CR2 | LP1 | VI1 | VI2 | VI8 | CU1 | CU2 | CU3 | TY1 | DKU } D^C avec $3 \leqslant n \leqslant B$

Les périphériques physiques sont les suivants :

- CR1 : Lecteur de carte n°1.
- CR2 : Lecteur de carte n°2 (sur SOLAR seulement).
- LP1 : Imprimante.
- VIα : Visu n°α.
- CUi : Coupleur universel n°i.
  (CU1 : perforateur de cartes).
- TY1 : Télétype.
- DKU : Disque Utilisateur 50 mégaoctets (SOLAR seulement).

■ *Assignation à un périphérique vide.*

! ASSIGN ⌀ n = { D^C | Return } $3 \leqslant n \leqslant B$

L'assignation de l'unité logique étant vide, les demandes de service sur cette unité seront donc sans effet.

■ *Assignation de fichiers.*

! ASSIGN ⌀ n = { ⌀ | N } , nom du fichier D^C

avec :

- $3 \leqslant n \leqslant B$
- Ø : fichier Old (déjà existant).
  N : fichier New (création).
- nom du fichier = chaine de caractères quelconques de longueur 1 à 20. Attention : les blancs sont pris en compte, même en fin de chaîne.

■ *Remarque.*

Un périphérique aussi bien qu'un fichier, ne peut faire l'objet que d'une seule assignation. On en peut pas avoir par exemple :

```
! ASSIGN ⊬ 4 = VI2 D^C
! ASSIGN ⊬ 5 = VI2 D^C
```

La seconde assignation serait refusée.

■ *Désassignation.*

```
! ASSIGN ⊬ n = S D^C
```

$3 \leqslant n \leqslant B$

La désassignation telle qu'elle est décrite ici ne concerne qu'une unité logique (de 3 à B) précédemment assignée. Elle a pour effet de rompre l'assignation et donc de libérer : d'une part l'unité logique et d'autre part le périphérique ou le fichier ("close" du fichier). On a la possibilité de désassigner globalement les unités logiques 3 à B en utilisant la commande ! CLØSE.

```
! CLØSE D^C
```

désassignation globale des unités logiques 3 à B.

■ *Release de l'espace disque occupé par un fichier.*

```
! ASSIGN ⊬ n = R D^C
```

avec :

- $3 \leqslant n \leqslant B$
- n déjà assignée préalablement à un fichier :

  exemple :
  ```
  ! ASSIGN ⊬ 3 = Ø, SI ⊬ FICH D^C
  ! ASSIGN ⊬ 3 = R D^C
  ```

tout l'espace occupé par le fichier "SI ∅ FICH" est ainsi libéré. Nota : le nom du fichier existe toujours ; pour "deleter" totalement un fichier (nom du fichier + espace disque occupé par le fichier), utiliser le processeur de base !DF (cf : processeurs de base du système).

■ *Exemples d'assignations/désassignations.*

! ASSIGN ∅ A = I D^{C}	Assignation de l'unité logique A à l'organe d'entrée.
! ASSIGN ∅ A = S D^{C}	Désassignation de l'unité logique A.
! ASSIGN ∅ A = N, TØTØ ∅ 20/12/77 D^{C}	Assignation de l'unité logique A au fichier new dont le nom est "TØTØ ∅ 20/12/77".
! ASSIGN ∅ 5 = Ø, FIFI D^{C}	Assignation de l'unité logique 5 au fichier déjà existant de nom "FIFI".
! ASSIGN ∅ 6 = CR1 D^{C}	Assignation de l'unité logique 6 au lecteur de cartes.
! ASSIGN ∅ 4 = D^{C}	Assignation vide de l'unité logique 4.
! CLØSE D^{C}	Désassignation globale des unités logiques 3 à B.

! WAIT. Mise en attente.

! WAIT D^{C} ou ! W D^{C}

L'unité est mise en attente. Le dialogue utilisateur-système est suspendu. Pour débloquer, faire un Alt-Mode.

! X simulation du Alt-Mode sur un périphérique.

! X ∅ n	$3 \leqslant n \leqslant B$ Utilisable sous ACN = :SYS

!X crée un utilisateur dont le périphérique d'entrée est assigné à n. !X est utilisé notamment pour lancer un job batch ; il suffit de faire par exemple :

```
! ASSIGN ∅ A = CR1 D^C
! X ∅ A D^C
```

Ceci a pour effet de simuler l'envoi d'un Alt-Mode par CR1 : le système va donc chercher des commandes sur le lecteur et liste les commandes et ses réponses sur l'imprimante. Le flot de cartes sur le lecteur devra commencer par une carte !L (Logon).

Si l'on fait !X Ø n D^{C} alors qu'il y a déjà un job en cours ayant le périphérique associé à n pour organe d'entrée, alors !X est équivalente à 2 Alt-Mode sur ce job, sauf si ce job est en WAIT, auquel cas !X est sans effet.

Nota : !X n'est pas utilisable avec CR2. On ne peut donc rentrer un job batch que sur CR1.

!GØ lancement ou relance de programme.

!GØ D^{C}

Deux cas possibles :

- on vient de charger un programme par !CALL (voir processeurs de base de CMS 4, !CALL) ; on lance l'exécution de ce programme par !GØ.

- le programme en cours d'exécution a fait un retour au CCi (retour au Control Card Interpreter ou retour au ?) ; dans ce cas, !GØ relance ce programme à "l'instruction suivante". Utile notamment en test pour le debugging. Voir Interface Utilisateur-Système.

! Return

! Return provoque la relance du programme actif avant le retour au CCi (?), à une adresse précise (voir Interface Utilisateur-Système), et qui a effectué ce passage à l'état ?

- par 2 Alt-Mode
- par défaut programme
- par SVC programmé.

!TAB : tabulation explicite.

!TAB D^{C}

Il y a deux façons de définir la tabulation :

- tabulation "implicite" (ou tabulation standard) : celle qui est définie automatiquement en utilisant le processeur de base !CALL commande T. (voir processeurs de base).

- tabulation "explicite" : celle qui est définie par la commande !TAB. Faire !TAB D^{C} puis entrer les caractères de tabulation qui sont les suivants :

- 0 : zone.

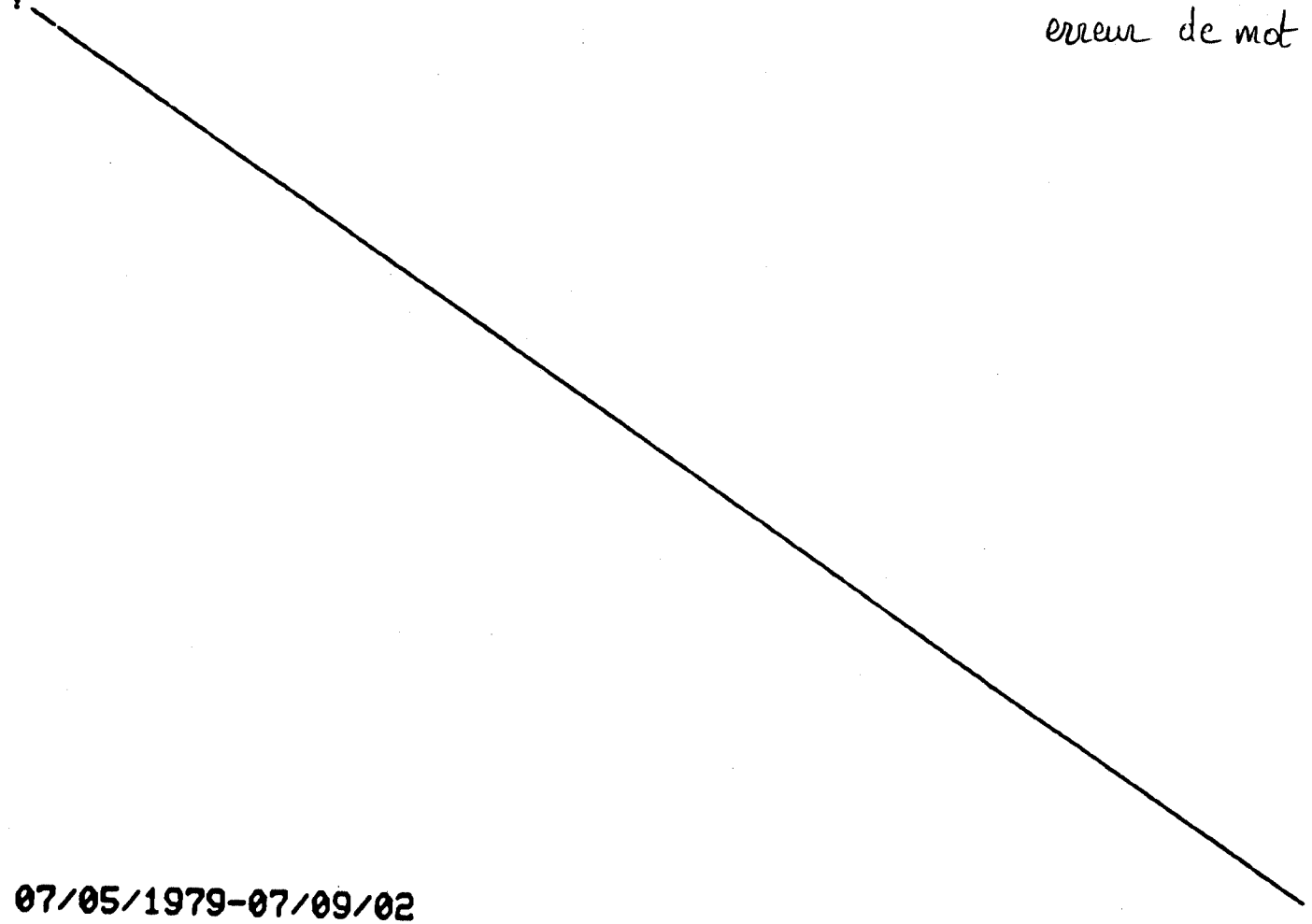
- 1 : début de zone.

Dans les deux cas, V^{C} est la touche de tabulation.

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/08/56

?!L :SYS
!WELCOME
 07/05/1979-07/08/58

?
```

logon

erreur de mot de passe.



```
07/05/1979-07/09/02
```

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/07/03

?!L :SYS
!WELCOME
 07/05/1979-07/07/05

?
?!DATE 07/05/1979-07/07/10

?!F
   !HI
 07/05/1979-07/07/11
```

logon

!DATE

logoff

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/04/03

?!L :SYS
!WELCOME
 07/05/1979-07/04/08

?
?!DATE 07/05/1979-07/04/16

?!WAIT
?!ASSIGN A=CR1
?!ASSIGN B=CU1
?!ASSIGN 3=VI2
?!ASSIGN 4=N.FICHIER EXEMPLE
?
??
?
??
?
??
?!ASSIGN A=S
?!ASSIGN A=O
?!CLOSE
?!ASSIGN 5=O.FICHIER EXEMPLE
?!F
   !HI
 07/05/1979-07/06/23
```

logon

!DATE

!WAIT déblocage par Alt-Mode

Assignations

touch

3 - LES PROCESSEURS DE BASE.

! I D^{C}

Les fonctions réalisables par ce processeur de base sont variées. Elles sont utilisables au moyen des sous-commandes présentées ci-après et illustrées par des exemples en fin de chapitre.

Sous-commandes de !I.

Elles ne sont utilisables que si le compte (ACN) est :SYS.

S — le système donne des renseignements (nombre d'actifs, affectation des disques, etc...).

V — inverse l'état de fonctionnement de la TV numérique. Celle-ci visualise 4K mots à partir de l'adresse a, celle-ci pouvant être spécifiée par la sous-commande @.

@ XXXX — visualisation de 4K mots à partir de l'adresse spécifiée par 4 chiffres hexadécimaux.

X — Exchange (disques de swapping).

N — supprime le mot de passe du système (personne ne peut plus se connecter sous :SYS).

P — rétablit le mot de passe du système.

C — demande de checkpoint. Le système demande ØK ?? répondre:

- . N non, pas de checkpoint.
- . Ø oui.
- . A faire un checkpoint périodique ; A n'est utilisable que si SYSID = 0.
  (le checkpoint périodique sera fait toutes les n secondes avec n fonction du nombre d'utilisateurs raccordés au système). La sauvegarde est faite en zone disque scratch. Elle est interruptible par Alt-Mode.

I — demande de restart au dernier checkpoint.
le système demande ØK ?? répondre :

- . Ø oui
- . N non
- . A

(le restart est interruptible par Alt-Mode).

M — Mail. Donner un texte de longueur $\leqslant$ 16 suivi de D^{C}.
Ce texte précédera le ? à chaque fois que celui-ci sera envoyé par le système.

F — Sortie de la commande !I (retour CCI).

PB 2

!CALL D^C : Appel processeur.

Le système demande de spécifier un mode d'appel : P, Q, R, S ou T

P	Privé (sous l'ACN de l'utilisateur)
Q	Système (sous l'ACN :SYS)
R	ACN (sous l'ACN spécifié ensuite avant le nom)
S	CCI
T	Tabulation implicite

Lui fournir le mode d'appel. Ensuite, le système demande des informations spécifiques fonction du mode d'appel : nom du processeur, etc... Les lui fournir. Le système se met ensuite en ? pour laisser la possibilité de faire diverses opérations - par exemple des assignations - avant de lancer le processeur. Pour lancer le processeur, faire !GØ D^C (voir les exemples).

!DF D^C : Delete file.

Permet de supprimer un fichier :

- tout l'espace occupé par le fichier est rendu.
- toutes les informations du système concernant ce fichier sont perdues (en particulier son nom).

```
? !DF D^C
FILE = FICHIER 1 D^C
? !DF D^C
FILE = FICHIER 2 D^C
??
?
```

le fichier "FICHIER 1" est détruit.

le fichier "FICHIER 2" ne peut être détruit (par exemple parce qu'il est utilisé en ce moment par un autre utilisateur...)

!GE D^C Appel de SMC.

Cette commande charge le "noyau" du système SMC avec lequel l'utilisateur se trouve alors en dialogue.

```
? !GE D^C
   SMC VOUS SOUHAITE LA BIENVENUE
   07/04/79-12/51/30
   *
```

→ SMC est disponible

→ rentrer une commande à SMC.

PB 2bis

ADDITIF PRØCESSEUR !I 1/8/79

Sur SØLAR, la télévision numérique fonctionne avec 3 composantes R, V, B. Voir le processeur TV de SMC §8. Le processeur !I sur Solar permet de commander la visualisation au moyen des 3 commandes R, V, B et H.

- commande R : diffusion sur composante RØUGE
- commande V : diffusion sur composante VERT
- commande B : diffusion sur composante BLEU
- commande H : inhibition diffusion (Halt).

La diffusion ne se fait que sur une composante à la fois. L'adresse de la zone mémoire à visulaiser est définie au moyen de la commande @ qui subsiste sur Solar.

Exemple :

```
> H          ——→  inhibition diffusion
> @ 4000     ——→  adresse zone à diffuser = '4000
> R          ——→  diffusion sur RØUGE
> H          ——→  inhibition diffusion
> @ 5000     ——→  adresse '5000
> V          ——→  diffusion sur VERT
> etc...
```

PAGE BLANCHE SUITE A ERREUR DE PAGINATION.

```
@!!UCPRS-CMS5 IS UP
 07/05/1979-14/11/41

?!L :SYS
!WELCOME
 07/05/1979-14/11/44

?
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=T
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=S
?
```

logon

!CALL tabulation standard

!CALL retour CCI

```
??
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=EDITS
?!GO
+:F
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=R
ACN=:SYS
NOM=EDITS.
?!GO
+:F
?
```

! CALL mode Q

appel du processeur EDITS

! CALL mode R

appel du processeur EDITS
sous l'ACN :SYS

```
??
?
??
?
??
?
??
?
??
?!DF
FILE-FICHIER EXEMPLE
?!GE
 SMC VOUS SOUHAITE LA BIENVENUE!!
 07/05/1979-14/15/50

*G
!FF
?
```

!DF (delete file)

!GE (appel de SMC)

retour au CCI par FF sous le niveau "!" de SMC

```
??
?!CLOSE
?!DF
FILE=BO EXEMPLE
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=ASSYS
?!ASSIGN 3=N.BO EXEMPLE
?!ASSIGN 4=LP1
?!GO
>          PROG
>          EOT          $SI CMS5 1$
```

!DF delete file

!CALL mode Q, appel du processeur ASSYS (assembleur)

assignations avant le lancement de ASSYS

lancement de ASSYS par !GØ

```
@!!UCPRS-CMS5 IS UP
 07/05/1979-09/41/53

?!L :SYS
!WELCOME
 07/05/1979-09/41/56

?
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=T
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=S
?!I 07/05/1979-09/42/07

ACN= :SYS
SYSID= 2 ON VI7
>S
$ACTIFS=0004
FREE DK=D4E3
ETAT=78B3
MEMOIRE VIRTUELLE=0383
SWAP ON 0025
>F
```

?

logon sous CMS5 (version Solar de CMS)

tabulation standard

!I sous CMS5

sous-commande S de ! I
- → nombre d'utilisateurs actifs.
- → espace disque disponible
- → informations Système

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/27/21

?!L :SYS
!WELCOME
 07/05/1979-07/27/23

?
?
??
?
??
?
??
?
??
?
??
?07/05/1979-07/27/32

ACN= :SYS
SYSID= 2 ON VI7
>S
$ACTIFS=0004
DKS1=0024
DKS2=0024
FREE DK=0178
>MAIL=BONNE ANNEE
>F
BONNE ANNEE
?
??
BONNE ANNEE
?
??
BONNE ANNEE
?
??
BONNE ANNEE
?
??
BONNE ANNEE
?
??
BONNE ANNEE
?
??
BONNE ANNEE
?!I 07/05/1979-07/27/44

ACN= :SYS
SYSID= 2 ON VI7
>MAIL=
>F
?
??
?
??
?
??
?
??
?
??
?
```

Exemple d'utilisation de la sous-commande M du processeur de base !I

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/23/11

?!L :SYS
!WELCOME
 07/05/1979-07/23/14

?
?!I 07/05/1979-07/23/21

ACN= :SYS
SYSID= 2 ON VI?
>S
$ACTIFS=0004
DKS1=0024
DKS2=0024
FREE DK=017A
>V OFF
>V ON 6FFF
>V OFF
>V ON 6FFF
>@3000
>V OFF
>V ON 3000
>@6FFF
>MAIL=BONNE ANNEE
>F
BONNE ANNEE
?!DATE 07/05/1979-07/24/24

BONNE ANNEE
?!CALL
 P=PRIVE,
 Q=SYSTEME,
 R=ACN,
 S=CCI,
 T=TABULATION.
 LOAD AND RUN
MODE=S
BONNE ANNEE
?
```

Exemples d'utilisation du processeur de base !I sous CMS4 :

- sous-commande V : vidéo on/off
- l'adresse à partir de laquelle doit être visualisée la mémoire (sur 4K mots) est précisée en utilisant la sous-commande @ de !I

```
??
?!I 05/09/1979-12/00/19

ACN= 'SYS
SYSID= 0 ON VI7
>S
$ACTIFS=0001
FREE DK=D9CA
ETAT=FEB3
MEMOIRE VIRTUELLE=0383
SWAP ON 0025
>@4000
>H4000
>R4000
>V4000
>B4000
>@2000
>R2000
>@4000
>V4000
>@6000
>B6000
>H6000
>F
?
```

Exemple d'utilisation du processeur de base !I sur CMS5-SOLAR :

- sous-commande @ : adresse de la zone mémoire à visualiser.
- sous-commande H : suspension des transferts mémoire → composante RVB.
- sous-commandes R, V, B : activation transferts. Une composante au plus est active à un instant donné.

```
@!!UCPRS-CMS4 IS UP
 07/05/1979-07/09/16

?!L :SYS
!WELCOME
 07/05/1979-07/09/20

?
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=T
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=S
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=EDITS
?!GO
+ED'FICHIER EXEMPLE'
```

```
*IN0
>(
>(        EXEMPLE DE DIALOGUE :
>(
>(
>( ICI, ON DIALOGUE AVEC L'EDITEUR
>( DE FICHIER 'EDITS' QUE L'ON A
>( APPELE EN UTILISANT LE PROCESSEUR
>( DE BASE !CALL.
>(
>:F
*:F
+DF'FICHIER EXEMPLE'
+:F
?!DATE 07/05/1979-07/12/35

?
```

Exemple de dialogue :

- logon.
- tabulation standard.
- retour au CCI.
- appel du processeur EDITS (éditeur de fichiers).
- lancement de EDITS par !GØ.
- dialogue avec EDITS lui-même.
- retour au CCI.
- demande de la date.

4 - AIDES À LA MISE AU POINT.

!DEBUG D^C

La commande !DEBUG permet l'utilisation de sous-commandes permettant diverses opérations de débugging de programmes (display registres, chargement de registres, affichage d'un mot mémoire, affectation de valeur à un mot mémoire, etc...).

On dispose sous !DEBUG de deux registres virtuels : @ et M. @ Registre adresse ; M "Registre" mémoire. On a toujours (@) = M. Exemple : si @ = 18C4, alors M = 53A0 si le contenu du mot mémoire d'adresse 18C4 est 53A0.

- *Les sous-commandes de !DEBUG.*

; D^C Sortie de Debug

r D^C Display du registre r

Cette sous-commande permet de displayer les registres tels qu'ils étaient avant un retour programmé au CCI (sauf X et P qui ont été détruits).

Exemple : A D^C display registre A
@ D^C " " @
M D^C " mot d'adresse (@)

r T D^C Display registre r

Même chose que la sous-commande précédente, mais utilisée dans le cas où le retour au CCI n'était pas programmé mais a eu lieu sur - un défaut programme
ou - deux Alt-Mode successifs.

r $\{^{'}_{<}\}$ [valeur] D^C Changement registre.

Si < valeur > n'est pas servie, changement du registre à 0.

Exemple : @ < 2A D^C
M, 1234 D^C

On met en mémoire, à l'adresse 2A, la valeur 1234.

RETURN. Incrémentation de 1 du registre @.
(@) : = (@) + 1

± n Incrémentation du registre @ de ± n. (@) = (@) ± n

D^C @ reçoit la dernière valeur soit visualisée soit introduite.

Exemple :

@ < 2A D^C	@ = 2A
Return	@ = 2B
+3 D^C	@ = 2E
M D^C	Affichage de M = 1234
D^C	@ = 1234

!DEBUG ABSOLU (ACN = :SYS).

appel : !DEBUG $C^C D^C$

utilisation : de la même façon que !DEBUG, mais les adresses sont absolues... d'où le danger de perturber le fonctionnement du système. Utiliser avec précautions.

```
??
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=PGMESS
?!DEBUG
?@.400
?M0101
?
?@0101
?M97EE
?M.1
?
?M97ED
?M.2
?@0102
?
?@0103
?M848D
?M.3
?;
?!GO
>
```

Exemple d'utilisation de !DEBUG pour "patcher" un programme.

Rappels :

- D^C -> @ := dernière valeur affichée ou rentrée.
- Return -> @ := (@) + 1

5 - INTERFACE UTILISATEUR-SYSTEME.

1 SVC 0

11 Arguments et Résultats.

111 Généralités

112 Mot 0 de la table des arguments

113 Codes-retour généraux

12 Demande de service immédiat.

121 f = 1 Retour au CCI interactif

122 f = 2 Appel du CCI non-interactif

123 f = 3 Assignation généralisée

124 f = 4 Changement taille espace mémoire

125 f = 5 Temporisation

126 f = 6 Fonction vide

127 f = 8 Fausse lecture

128 f = A Fausse écriture

13 UL = 1 à 'B en assignation "explicite" et UL = 'C.

131 UL ⊸→ CR1 (lecteur de cartes)

132 UL ⊸→ LP1 (imprimante)

133 UL ⊸→ VIi (visuel)

134 UL ⊸→ TY1 (teletype)

135 UL ⊸→ fichier

136 UL ⊸→ DKU (disque utilisateur)

137 UL 'C (zone scratch disque)

14 UL = 3 à 9 en assignation "implicite" (espace des Noms-valeurs).

141 Espace des Noms et valeurs - Présentation

142 Opérations sur l'espace des noms et valeurs

143 Compactage des noms

2 INSTRUCTION '1E16

3 INSTRUCTIONS '1Eα5 et '1E1D

4 PILE UTILISATEUR

5 SVC MAITRES

5-1 SVC 1

5-2 SVC 2

6 TEST D'ASSIGNATION D'UNE UL

7 ESPACE MEMOIRE ESCLAVE

8 CODES-RETOUR : TABLEAU RECAPITULATIF.

1 SVC 0

11 Arguments et Résultats.

111 Généralités.

Arguments : Registre A : adresse mot table d'arguments

Table d'arguments : 1 à 4 mots de 16 bits (mots 0 à 3).

Résultats : Registre A inchangé.

Registre X { = 0 si tout s'est bien passé
= code retour sinon.

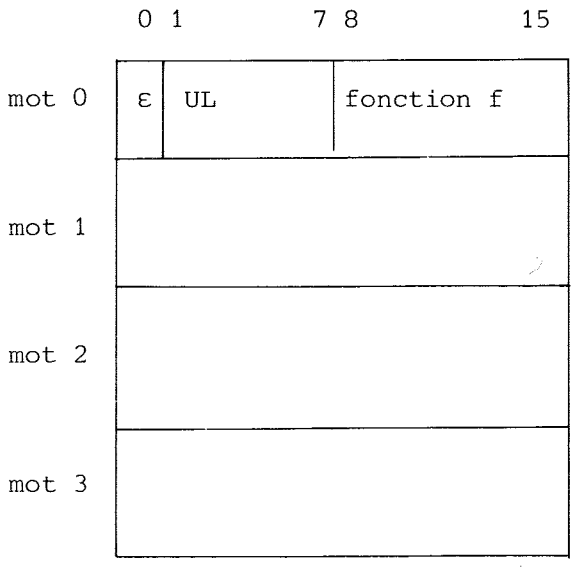
Registre S positionné par le système (CPZR X) avant le retour au programme.

Table d'arguments inchangée.

Séquence d'appel type :

```
    LAD   Table
    SVC   0
[   JNE   ERREUR ]
```

Table des arguments.



112 Mot 0 de la table des arguments.

UL	ε = 0 Assignations "explicites"	ε = 1 Assignations implicites
0	Demande de SERVICE IMMEDIAT	
1	1 Organe d'entrée	
2	2 Organe de sortie	
3	⟶ ! ASSIGN (éventuellement !SGN)	3 ⟶ DLN Delete Name
4		4 ⟶ STN Store Name
5		5 ⟶ LØN Load Name
6		6 ⟶ LNS Load :SYS
7		7 ⟶ LNU Load :ACN
8		8 ⟶ NXP Next Parallèle
9		9 ⟶ NXS Next Série
'A		
'B		
'C	'C ⟶ Zone scratch disque	

113 Codes-retours généraux Registre X

X = '81 Unité Logique non-assignée
'82 N° d'U.L. invalide
'83 Fonction invalide
'84 Compte d'octets erroné
'85 Violation mémoire
'86 Adresse table d'arguments erronée.

Les codes spécifiques à une opération sont donnés en même temps que les explications sur cette opération.

12 Demande de service immédiat.

Fonction f.

121 - f = 1 : Retour au CCI interactif.

Le CCI (Control Cards Interpreter) reprend la main ; il la rendra au programme à la rencontre d'une commande !GØ

Exemple : Permettre d'entrer des !ASSIGN à la visu, suivis de !GØ.

122 - f = 2 : Appel du CCI non-interactif.

Même chose que pour f = 1, mais cette fois, une commande au CCI est en mémoire. On a dans la table d'arguments :

mot 1 : adresse octet de la commande

mot 2 : longueur octets ($\leqslant$ 80)

---

123 - f = 3 : Assignation généralisée (Seulement sous ACN =:SYS)

Deux possibilités :

1/ assignation de l'UL n ($3 \leqslant n \leqslant$ 'B) à un périphérique désigné par son nom interne :

mot 1 { bits 0 - 7 = unité logique ($3 \leqslant UL \leqslant B$).
{ bits 8 - 15 = nom interne du périphérique.

Code retour : X = '83 si impossible

2/ on veut connaître le nom interne du périphérique assigné une unité logique (SOLAR seulement) :

{ bits 0 - 7 = unité logique
{ bits 8 - 15 = nom interne inexistant (exemple : '80)

On reçoit : X = '80 et BØX reçoit le nom interne du périphérique réellement assigné à l'unité logique.

Exemple :

(A) ⟶ | 3 |
| 01 80 |

Le SVC donnera :

X = '83 assignation impossible.

BØX = nom interne de l'organe d'entrée.
(par exemple NSPVi4 si on est sur la visu 4).

Le contenu de la 'BØX' est accessible par l'instruction '1E35 (voir ci-après).

124 - f = 4 : Changement de taille espace mémoire.

On donne dans le mot 2 de la demande la taille mémoire globale en K-octets que l'on veut se voir attribuer - (le mot 1 est inutilisé).

mot 2 = 2,4,8,12,16,20,24 K octets.

Exemple : mot 2 = '4000 ≡ 8 K octets.

Nota : si l'on demande une taille mémoire intermédiaire, on se verra attribuer la taille immédiatement supérieure. Si la taille demandée est incorrecte (exemple : > 24 K octets), erreur '84 : compte d'octets invalide.

125 - f = 5 : Temporisation.

On peut demander la mise en sommeil de son programme pour une durée d $1 \leqslant d \leqslant 60$ secondes. La temporisation est non-interruptible, c'est-à-dire qu'un Alt-Mode sera sans effet pendant la durée de la temporisation, et qu'il sera effectif dès la réactivation du programme.

126 - f = 6 : Fonction vide.

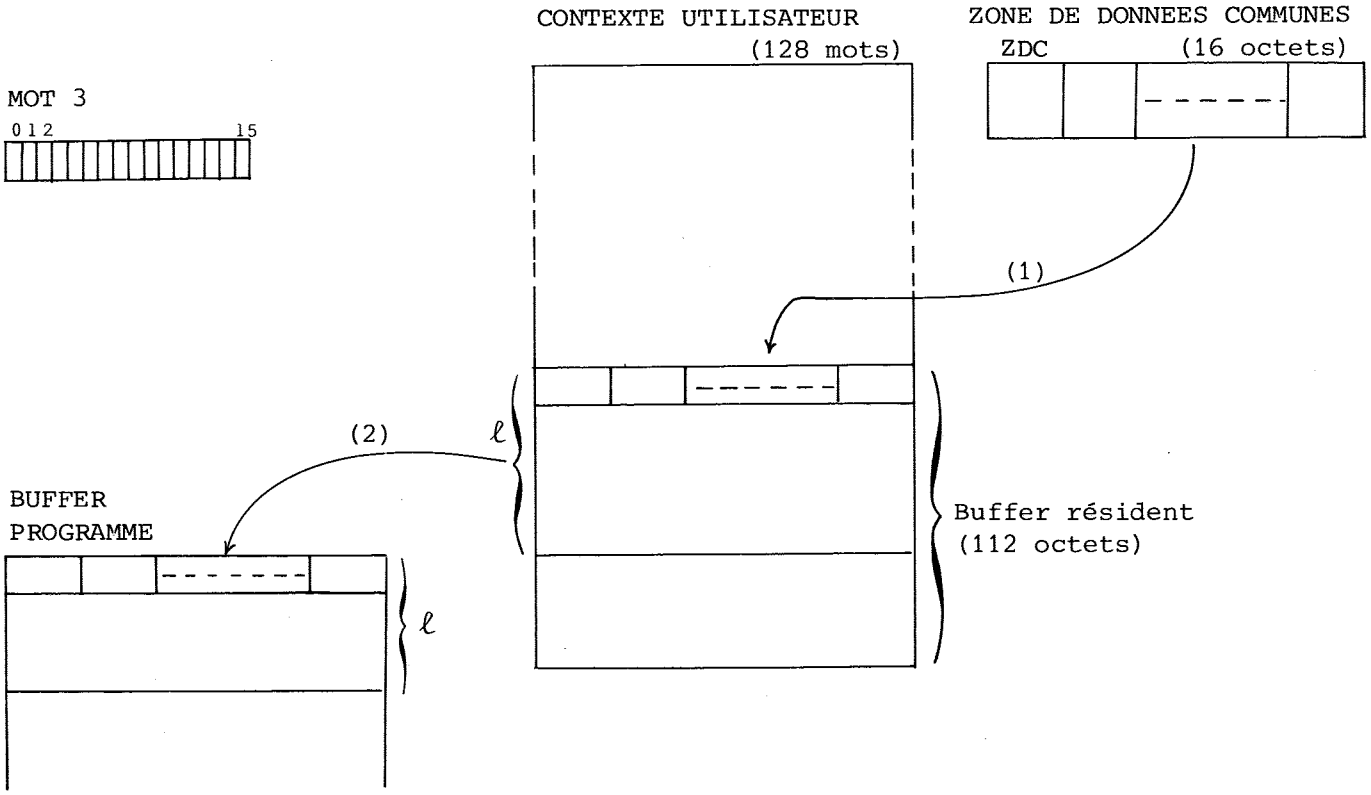
Le système ne réalise aucune fonction et rend la main au programme.

127 - f = 7 : Fausse lecture.

mot 1 : adresse octet buffer programme.

mot 2 : longueur octets.

mot 3 : table de 16 bits validant les 16 premiers octets de la zone de données communes.



La fausse lecture permet d'acquérir des données situées en mémoire dans un buffer résident appartenant au contexte utilisateur. Avant le mouvement des données du buffer vers le programme, chacun des 16 premiers octets du buffer résident est remplacé par l'octet de même rang (n) de la Zone des données Communes si le bit de rang n du mot 3 de la table des arguments vaut 1.

Exemple :

mot 3 = FOO1

ZDC = ABCD EFGH IJKL MNØP

buffer résident = 1234 5678 9000 0000 ....

La fausse lecture va :

1/ opérer un mouvement sélectif de la ZDC sur le buffer résident.

2/ transférer les données du buffer résident vers la zone programme spécifiée.

Le buffer résident devient :

ABCD 5678 9000 000P ....

128 - f = A : Fausse écriture.

mots 1,2 et 3 : idem fonction 8.

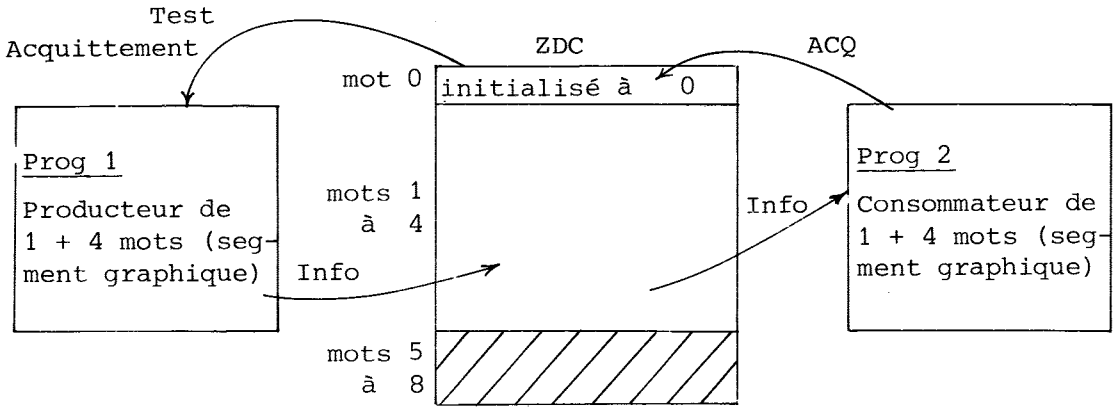
La fausse écriture :

1/ transfère les données de la zone programme d'adresse (mot 1) vers le buffer résident.

2/ opère un mouvement sélectif du buffer résident (16 premiers octets) vers la ZDC en fonction du mot 3.

L'utilisation des fonctions 8 et A permet de faire communiquer entre eux différents programmes par l'intermédiaire de la ZDC, (échange de données, synchronisation).

EXEMPLE D'UTILISATION DE LA ZDC



Prog 1

```
            LØCAL
             ⌇
ACQ :       WØRD  1
SEG :       DZS   4
             ⌇
             ⌇
STØRE :     WØRD  'ØØØA      < envoi segment
                             < en ZDC
            WØRD  ACQ-ZERØ+2< adresse octets
            WØRD  4+1 * 2    < longueur octets
            WØRD  'FFCØ <mot de validation ZDC


GETACQ :    WØRD  'ØØØ8  < test acquittement
            WØRD  ACQ-ZERØ*2<adresse octets
            WØRD  2          <longueur octets
            WØRD  'CØØØ<mot de validation ZDC
             ⌇
             ⌇
            PRØG
             ⌇
             ⌇
            LAD   GETACQ
E1 :        EQU   $
            SVC   Ø <- - - - - - - - - - (1)
            CPZ   ACQ  < acquittement ?
            JNE   E1         < pas encore
             ⌇
             ⌇
            LAD   STØRE
            SVC 0 <envoi segment suivant. - - -
             ⌇
             ⌇
```

Prog 2

```
            LØCAL
             ⌇
ACQ :       WØRD  Ø
SEG :       DZS   4
             ⌇
             ⌇
LØAD :      WØRD  'ØØØ8 <envoi segment en ZDC

            WØRD  ACQ-ZERØ 2<adresse octet
            WØRD  4+1*2 < longueur octets
            WØRD  'FFCØ <mot de validation ZDC


ACQUIT :    WØRD  'ØØØA <envoi acquittement
            WØRD  ACQ-ZERØ*2 <adresse octet
            WØRD  2 < longueur
            WØRD  'CØØØ < mot de validation
             ⌇                          ZDC
             ⌇
            PRØG
             ⌇
             ⌇
            LAD   LOAD
E1 :        EQU   $
            SVC   Ø <- - - - - - - - - - (2)
            CPZ   ACQ < nouveau segment ?
            JE    E1  < non
             ⌇
             ⌇
            STZ   ACQ
            LAD   ACQ
            SVC   Ø    < oui, acquitté
```

(1) Synchronisation P1 P2
(2) Echange d'information P1 P2

13 UL = 1 à B en Assignation "explicite" et UL = 'C.

$\varepsilon = \begin{cases} 0 \\ 1 \end{cases}$ et UL $\in [1,2]$
$\varepsilon = 0$ et UL $\in [3,8]$
$\varepsilon = \begin{cases} 0 \\ 1 \end{cases}$ et UL = 'C

■ *Lecteur de Cartes.*

131 | UL ⊖→ CR1 |

---

m1 : @ octet buffer

m2 : longueur octets $\begin{cases} \leqslant 80 & \text{si lecture ASCI} \\ \leqslant 160 & \text{si lecture binaire} \end{cases}$

X = '11 Ecriture demandée

X = '5 Longueur erronée

f = 0 Lecture ASCII
f = 1 Lecture ASCII et écho sur Imprimante $\Big\}$ $(m_2) = 80$
f = 8 Lecture binaire $\Big\}$ $m_2 = 160$

Remarques :

- en lecture ASCII, si le code n'existe pas, le système suppose le caractère "?".

- en lecture binaire, on obtient dans le buffer une suite de mots de la forme suivante :

0 ... 11	12 ... 15
Colonne 1	0
Colonne 2	0
----------	----------
Colonne 80	0

■ *Imprimante.*

132 | UL ⊖→ LP1 |

m1 = @ octet buffer

m2 = longueur

X' = 11 Lecture demandée

X' = '5 Longueur erronée (> 132)

f = 2 Ecriture Les caractères de saut sont à gérer par l'utilisateur :

buffer = Caract de Saut < texte > $\begin{cases} \text{RC} \\ \text{RC - LF} \end{cases}$

caract. de saut = { LF = 'OA = line feed
@ = '40 = Saut de page

. RC = 'OD = Carriage Return : provoque le vidage du buffer de l'imprimante.

. RC + LF = '6D = Carriage Return + Line feed

■ *Visuel.*

133 | UL ⊸ VIi |

Codes retour spécifiques.

X = '93	S^C reçu avant une sortie
'13	S^C reçu pendant une sortie
'7D	Alt mode reçu (voir exemples ci- après)
'11	La fonction n'existe pas
'5	Erreur en mode graphique
'3	Fonction inaccessible à l'instant t (voir séquences des fonctions)

Fonctions.

Les fonctions sont énumérées ci-dessous. Chacune d'elle (sauf la fonction 7) met un automate dans un certain état. Cet automate gère le passage d'un état à un autre (c'est-à-dire qu'il l'autorise ou l'interdit).

f = 0	état	L	Lecture
1		LE	Lecture avec écho sur le visuel
2		E	Ecriture
'A		EG	Ecriture mode graphique
9		LG	Lecture mode graphique
3		ØG	Open graphique (ou passage en mode graphique)
4		CG	Close graphique ("fermeture du " ")
5		ER	Erase (effacement écran)
6, 8		CU	Mise en fonction curseur graphique (réticule)
7		-	Ecriture directe, quel que soit l'état.

Séquences d'états autorisées :

(0) L ⟶ L, LE, E, EG, ØG, CG, ER, CU

(1) LE ⟶ L, LE, E, ØG, ER

(2) E ⟶ L, LE, E, ØG, ER

('A) EG ⟶ L, EG, ØG, CG, ER, CU

(9) LG ⟶ L, LE, E, ØG, ER

(3) ØG ⟶ L, EG, ØG, CG, ER, CU

(4) CG ⟶ L, LE, E, ØG, ER

(5) ER ⟶ L, LE, E, ØG, ER

(6,8) CU ⟶ EG, LG

Arguments

. f = 0, 1, 2 (L, LE, E) { m_1 = @ octet buffer ; m_2 = longueur octets

Remarque :

En écriture (f = 2) : Expansion et Extension alpha-numériques

- si le bit 0 du caractère K_i vaut 1, alors bits 1-7 (K_i) = compte de duplication de K_{i+1} ; par exemple, la séquence '8A '42 est interprétée comme une séquence de 'A(=10) caractères '42(=B) successifs.

- si '60 ≤ K_i ≤ '6F , il y a extension alphanumérique.

. '60 = "Curseur Home"

. '6D = Carriage return, Line feed. Ce caractère '6D est utilisable aussi sur l'imprimante.

(On entend par extension le fait que le caractère K ∈ ['60, '6F] est transformé en une séquence de caractères qui vont commander le visuel pour réaliser la fonction associée au caractère K).

f = 'A Ecriture graphique { m1 = @ octet buffer ; m2 = longueur octets

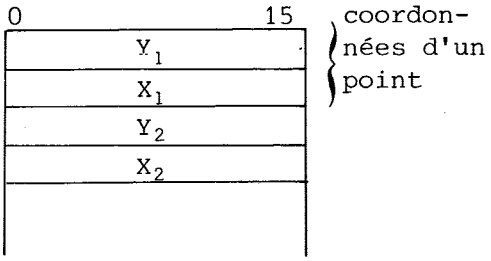
Le buffer contient un ensemble de couples (Y,X) qui sont les coordonnées de points sur l'écran graphique (1024 x 1024), exprimées en binaires. Ces coordonnées seront transformées par le système en une séquence de caractères qui commanderont le visuel pour l'affichage graphique.

Remarques :

- Si (@ buffer impaire) ou bien si (longueur octet non-multiple de 4), alors aucune conversion n'est effectuée sur le contenu du buffer qui est donc transmis tel quel.

- Si, pour l'un des couples (Y,X) Y ou X $\notin [0,1023]$, alors, aucune conversion n'est effectuée sur Y ou X, les autres données étant normalement converties.

Buffer



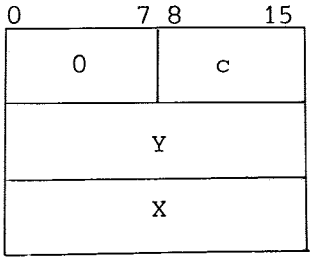
. f = 9 Lecture Graphique

- m1 = @ octet buffer
- m2 = 6 (longueur octet)

On obtient dans le buffer :

- c étant le caractère frappé par l'opérateur pour transmettre les coordonnées du réticule

- (Y,X) étant les coordonnées du réticule.



. f = 3 (ØG), 4(CG), 5(ER) Aucun argument

Remarque :

Lorsqu'on fait ER (Erase), mettre le programme en attente pendant environ 2 secondes (fonction longue).

. f = 6,8 Mise en fonction curseur (CU)

- . m1 = 0 si f = 6
- . m1 ≠ 0 si f = 8

. f = 7 Ecriture directe

- . m1 à m3 = séquences de caractères
- . pas de buffer.

On envoie n caractères K_1 à K_n

avec
- $n \leqslant 6$
- K_1 à $K_{n-1} \neq D^C$
- $Kn = D^C$

m_0	ε UL	7
m_1	K_1	K_2
m_2	K_3	K_4
m_3	K_5	K_6

Voir ci-après exemple d'utilisation de f = 7 pour Hard Copy.

Exemples.

- Lecture graphique : il faut utiliser n fois la séquence suivante :

1 - ØG Open graphique

2 - CU mise en fonction curseur graphique

3 - LG Lecture graphique

- Hard copy f = 7 : la table d'arguments contient :

	0	1 7	8 15
m_0	0	UL	07
m_1	1B		17
m_2	04		

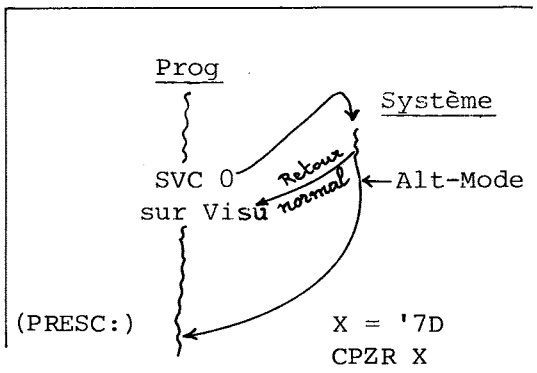
~~Programmer ensuite une temporisation~~ du programme de durée ⩾ 15 secondes, pour attendre la fin du balayage de l'écran.

- Exemple de code retour X = '7D.

Le code retour X = '7D signifie qu'un Alt-Mode a été reçu. Dans ce cas, on n'aura pas en général à exploiter ce code, car le déroulement séquentiel du programme est interrompu ; le système charge X à '7D, positionne S par CPZR X, et rend la main à l'adresse PRESC, celle-ci étant

. soit '12 (implicite)

. soit 'a, a étant spécifié par une instruction 1EB5 (cf cette instruction).



■ *Télétype.*

134 | UL ⟶ TY1 |

m_1 = 0 ou @ octet buffer

m_2 = longueur octet si $m_1 \neq 0$

f = 0 Lecture

f = 1 Ecriture

- Si $m_1 = 0$, le message envoyé sur le télétype sera le suivant : !! nom du demandeur, date, heure.

- Si $m_1 \neq 0$, le message envoyé sera le texte précédent, suivi du texte d'adresse (m_1) et de longueur (m_2).

■ *Fichier.*

135 | UL ⟶ fichier |

Fichier :

Ensemble de clés et d'enregistrements. A chaque clé est associé un seul enregistrement.

Clé :

C'est un couple (n1, n2) avec

$0 < n1 < 2^{13}$ et $n2 \geqslant 0$ et pair.

Enregistrement :

Ensemble de blocs chaînés entre eux. Le premier bloc de la chaîne est à accès direct par clé. Les blocs suivants sont à accès séquentiel par chaînage ; ils sont en nombre quelconque.

Bloc :

C'est Q x 128 mots d'information (ou moins) ; le mot 0 étant réservé au système qui l'utilise pour chaîner le bloc suivant. Il y a donc (Q x128)-1 mots d'informations utilisable par bloc.

Q (Quanta) est le nombre de secteurs physiques disque par bloc. C'est-à-dire :

- en T1600 : Q = 1 ; 1 secteur physique (128 mots) = 1 bloc.
- en SOLAR : Q = 3 ; 3 secteurs physiques (3 x 128 mots) = 1 bloc.

Lorsque l'utilisateur lit un bloc, le système le place en mémoire à l'adresse demandée (voir les fonctions de lecture ci-après), et place dans le premier mot du bloc :

- n $\geqslant$ 0 si le bloc n'est pas le dernier de la chaîne.
- n = '8000 si le bloc est le dernier de la chaîne.

Lorsqu'on parcourt la chaîne de blocs associés à une clé, on peut donc savoir si le dernier bloc lu est le dernier de la chaîne ou pas en testant le mot 0 du bloc lu :

- mot 0 $\geqslant$ 0 : il existe un bloc suivant.
- mot 0 < 0 : pas de bloc suivant (dernier bloc)

Ouverture/fermeture de fichier :

ouverture : !ASSIGN ⌀ n = $\left\{ \begin{matrix} \emptyset \\ N \end{matrix} \right\}$, nom fichier D^C

fermeture : !ASSIGN ⌀ n = S D^C ou !CLØSE D^C

Ouverture/fermeture enregistrement :

Ces opérations sont réalisées par l'utilisateur des fonctions "open new key", "open old key", et "close save key" et "close release key" :

- open new key : création d'un nouvel enregistrement.

- open old key : accès à un enregistrement déjà existant.

- close save key : fermeture et sauvegarde enregistrement.

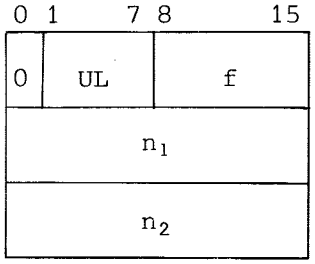
- close release key : destruction enregistrement.

Codes-retour spécifiques :

X = '8000	La clé existe et la fonction demandée est "open new key".
= '6000	L'ensemble des clés est vide, et la fonction demandée est "open old key"
= '2000 = '4000	La clé n'existe pas et fonction demandée = "open old key".
= 1	Deux open successifs demandés sans close entre les deux.
= 2	Close demandé sans open préalable.
= 3	Write demandé sans open préalable.
= 4	Read demandé sans open préalable.
= 5	Delete demandé sans open préalable.
= 6	Read demandé et l'enregistrement est vide ou terminé.
= 8	n_1 invalide.
= 9	n_2 invalide.

Fonctions.

- *Niveau clé (ou niveau enregistrement).*



f = 3 open next key — pas d'argument

f = 4 open new key — $m_1 = n_1$; $m_2 = n_2$

f = 5 open old key — $m_1 = n_1$; $m_2 = n_2$

f = 6 close release — pas d'argument. La clé courante et l'enregistrement associé sont détruits.

f = 7 close save — pas d'argument. La clé courante et l'enregistrement associé sont sauvegardés.

■ *Niveau bloc.*

f = 2 Ecriture

m_1 = @ octet buffer obligatoirement PAIRE
m_2 = longueur octets (⩽ (Q x 256))

Le premier mot est écrasé par le système.

f = 8 Lecture

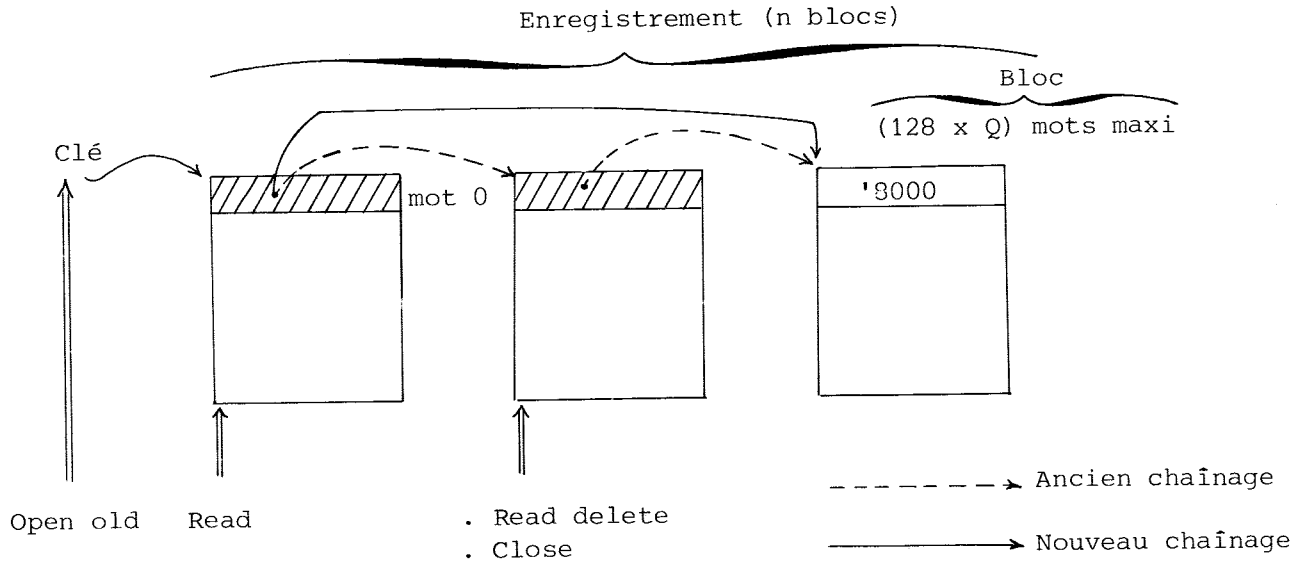
m_1 = @ octet buffer obligatoirement PAIRE
m_2 = longueur octets (⩽ (Q x 256))

Le premier mot du bloc contient un chaînage. Celui-ci peut être détruit par l'utilisateur ; le système sait le restaurer.

f = 9 Lecture-delete

m_1 = @ octet buffer obligatoirement PAIRE
m_2 = longueur octets (⩽ (Q x 256))

Le bloc est lu. Il est disponible en mémoire. Le système le détruit sur le fichier immédiatement après la lecture.



■ *Disque utilisateur.*

136 UL ⟶ DKU

SOLAR seulement !

Format de la demande :

mot 0	fonction
mot 1	adresse octet
mot 2	longueur octet
mot 3	adresse secteur

Codes retour spécifiques :

X = '900D : violation écriture (écriture demandée alors que le disque est en écriture protégée).

X = '41 : adresse secteur incorrecte.

Capacité physique du disque utilisateur :

24 secteurs de 128 mots par piste

20 pistes par cylindre

400 cylindres.

L'adresse secteur fournie dans la demande est une notion logique : sa signification dépend de la fonction demandée, voir ci-après.

Fonction (mot 0 de la demande).

Seuls les bits 12 et 14 du mot 0 sont significatifs.

bit 14 { = 0 lecture
= 1 écriture

bit 12 { = 0 quanta = 3 : 1 secteur logique = 3 secteurs physiques.
= 1 quanta = 1 : 1 secteur logique = 1 secteur physique.

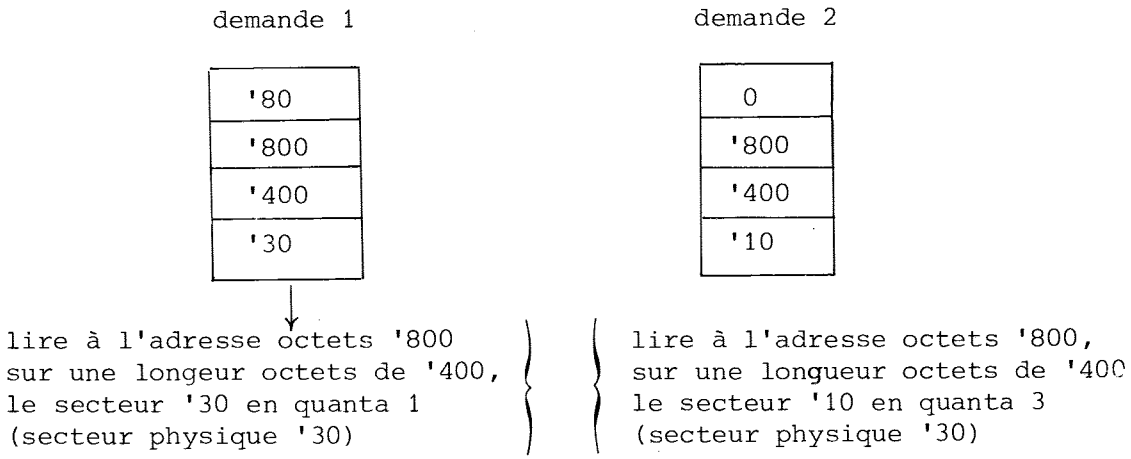
C'est l'utilisateur qui choisit son quanta pour utiliser DKU :

. en quanta 3 (bit 12 du mot 0 = 0) : un secteur logique = 3 secteurs physiques ; donc un secteur contient 3 x 128 mots = 384 mots. Dans ce mode, tout le disque DKU est accessible, et l'adresse secteur est limitée par la capacité du disque donnée ci-avant.

. en quanta 1 (bit 12 du mot 0 = 1) : un secteur logique = 1 secteur physique : donc un secteur contient 128 mots. Dans ce mode, l'utilisateur ne peut accéder qu'aux '10000 premiers secteurs du disque DKU, et l'adresse secteur n'est pas limitée (0 à 'FFFF).

Exemple :

Les deux demandes ci-dessous sont équivalentes :



■ *Zone scratch disque.*

137 | UL = 'C

Une zone disque à accès direct est allouée par le système à l'utilisateur pour la durée de son job. Elle contient :

. en T1600 : 32 secteurs de 128 mots.

. en SOLAR : 40 secteurs de 128 mots.

Format de la demande.

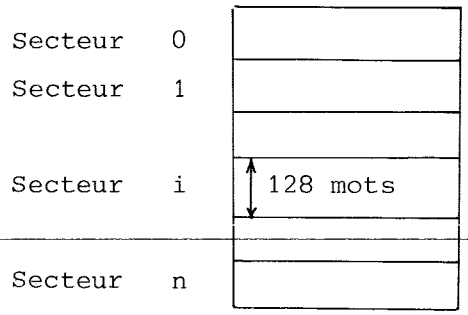
mot 0	fonction
mot 1	adresse octet
mot 2	longueur octet
mot 3	adresse secteur

f = { 0 lecture ; 2 écriture

{ de 0 à 31 en T1600 ; de 0 à 39 en SOLAR

Code retour spécifique :

X = '85 adresse secteur incorrecte.



Zone Scratch disque.

n = { 31 en T1600
    { 39 en SOLAR

■ *Disques DK2 DK3 (ACN = :SYS).*

L'utilisateur peut, moyennant certaines précautions, utiliser les disques du système :

- en T1600 DK2 et DK3 : chacun de ces disques contient '2000 secteurs de 128 mots chacun.

- en SOLAR : DK2 qui est un disque 50 mégaoctets comme DKU, et qui est utilisable en quanta = 1 ou quanta = 3 (voir DKU) ; et DK3 qui est un disque à têtes fixes de '1000 secteurs de 128 mots.

Pour s'assigner l'un de ces disques, l'utilisateur doit utiliser l'assignation généralisée (demandée par SVC, fonction = 3 et en précisant le nom interne du disque à utiliser).

Pour demander un échange, utiliser une demande semblable à la demande d'échange avec DKU, décrite précédemment à cette différence près : en T1600, moto de la demande = 0 : lecture;= 2 : écriture (bit 12 du mot 0 est inutilisé).

14 UL = 3 à 9 en assignation "implicite".

$\varepsilon = 1$	UL $\in [3,9]$

141 Espace des Noms et valeurs - Présentation.

On travaille sur un espace de noms, à chaque nom est associé une valeur. L'espace des noms possède une structure arborescente.

Tout nom est de la forme :

nom = ACN D^C Nom utilisateur D^C

ACN (Account Number ; n° de compte) et nom utilisateur sont chacun une chaîne de caractères quelconques sauf les caractères NULL ('00) et D^C ('04) et de longueur $\geqslant 1$.

La longueur totale est limitée à 40 caractères.

Exemples : :SYS D^C F J ∅ 25/12/77 ∅ # 20.D^C

:USE D^C PRØGRAMME ∅ P27 D^C

Représentation arborescente des noms.

Chaque noeud de l'arbre est un caractère (sauf 'NULL et avec les restrictions découlant de la structure des noms pour D^C).

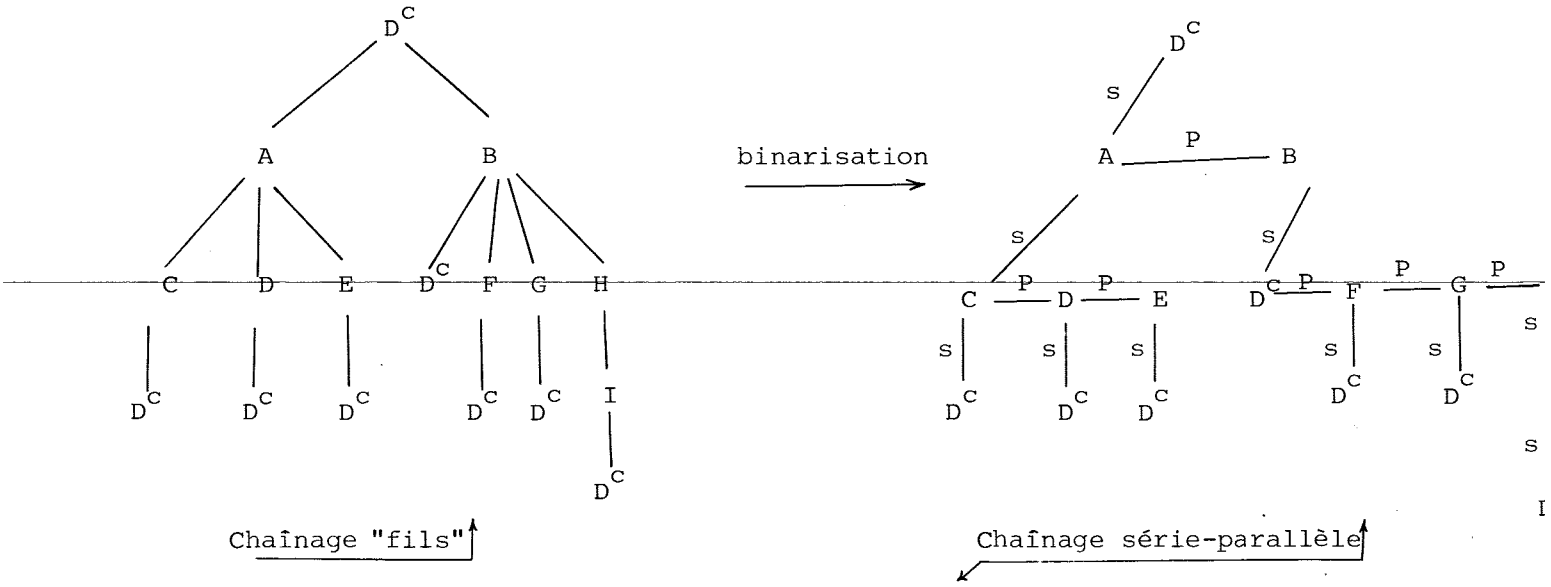
Chaque feuille de l'arbre est le caractère D^C et pointe sur la VALEUR associée au nom.

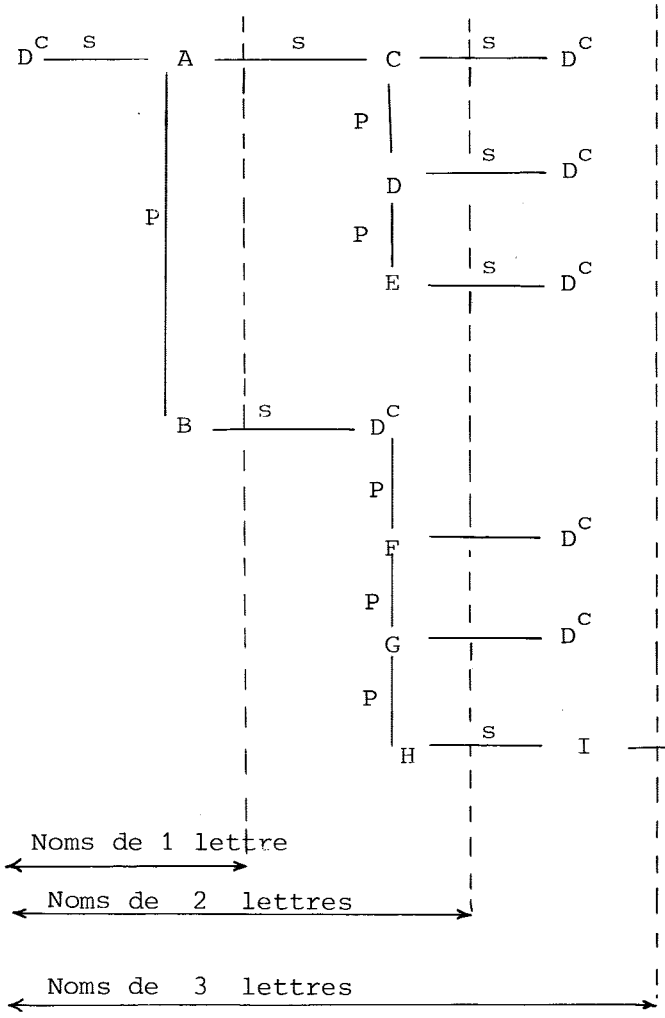
Les chaînages utilisés sont de type SERIE/PARALLELE, ce qui résulte de la binarisation de la structure arborescente classique. De cette façon, un noeud possède 2 liens au maximum.

Voir représentation et exemples page suivante.

IUS 19

Noms utilisateurs (la racine de l'arbre est le dernier caractère de l'ACN).





. Les nom contenus dans cet arbre sont :

A C
A D
A E
B
B F
B G
B H I

. Nota : n mots peuvent avoir une racine commune :

exemples :

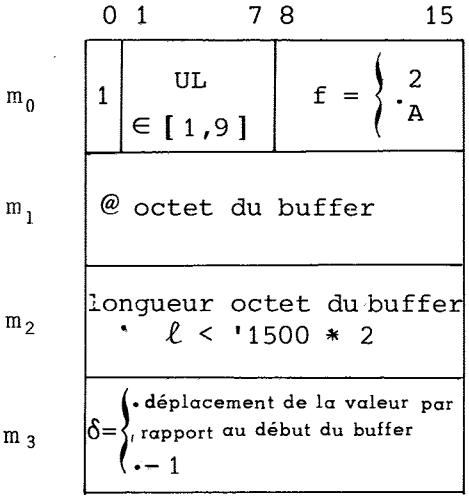
- MUSE D^C
- MUSEAU D^C
- MUSELIERE D^C

- Y = AX D^C
- Y = AX + B D^C
- Y = AX** 2 D^C

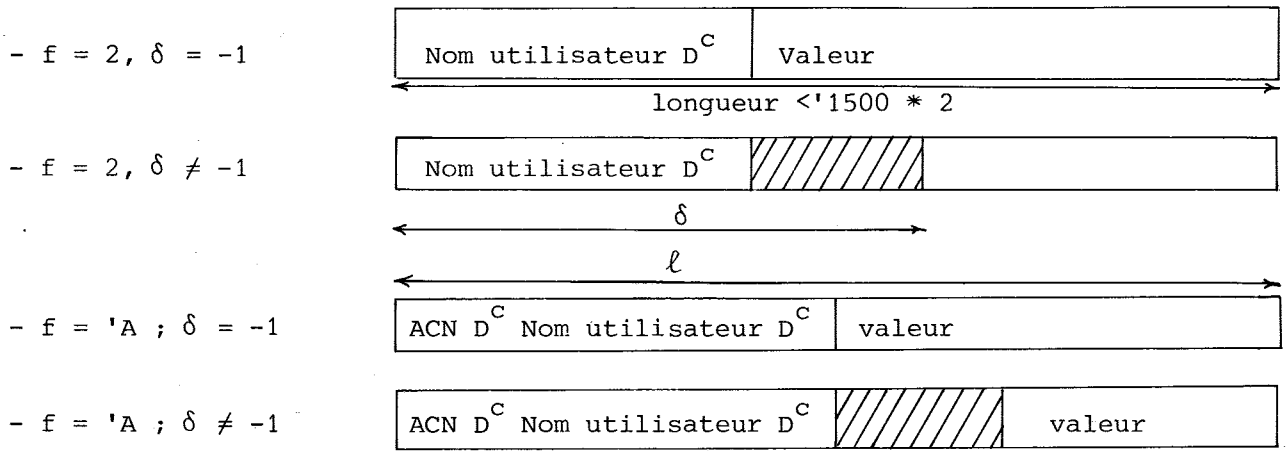
. Tout caractère terminal pointe sur sur une VALEUR.

142 Opérations sur l'espace des noms et valeurs.

■ *Table des arguments.*



■ *Buffer.*



■ *Fonction f* : Pour effectuer l'opération spécifiée par UL, si

- f = 2 : Le système concatène automatiquement l'ACN de l'utilisateur avec le nom servi dans le buffer (sauf si l'opération demandée est LNU)

- f = 'A: Le système ne concatène pas l'ACN au nom servi dans le buffer. f = 'A ne peut être utilisé que sous l'ACN :SYS.

. UL Opération effectuée.

UL = 3 : DLN — Delete Name
Le système détruit le nom et la valeur associée.

UL = 4 : STN — Store Name
Le système crée le nom et la valeur associée.

UL = 5 : LØN — Load Name
Le système va chercher la valeur associée au nom et la range dans le buffer en tenant compte du δ spécifié.

UL = 6 : LNS — Load Name sous ACN = :SYS
Le système concatène :SYS au nom servi dans le buffer au lieu de concaténer l'ACN de l'utilisateur.

UL = 7 : LNU — Load Name sous l'ACN d'un autre utilisateur.
Le système ne concatène pas l'ACN de l'utilisateur au nom servi dans le buffer. Analogue au f = 'A, mais ici, on a f = 2, ACN quelconque et Lecture seulement.

UL = 8 : NXP — Next Parallèle
Le système donne le nom suivant parallèle.
exemple : en se reportant ci-dessus (schéma de l'arbre) si nom = AC alors, NXP(AC) = ACD (et non pas AD). Pour chercher le suivant, il faut transformer par programme ACD en AD, et faire NXP(AD) qui donne ADE, etc...

UL = 9 : NXS — Next série
Le système donne le nom suivant série.
exemple : en se reportant ci-dessus (schéma de l'arbre) si nom = BH, alors NXS(BH)=BHI.

Remarques :

- NXP et NXS permettent le parcours complet de l'arbre des noms (jusqu'au code retour '5 : suivant nom trouvé (voir ci-dessous les codes-retour)).

- Pour mettre à jour une valeur, il faut faire Load Delete et Store.

■ *Codes retour spécifiques.*

X = 1 Nom erroné (contient NULL ou manque délimiteur D^C)

2 Argument erroné (δ > longueur par exemple)

3 Le nom existe et STN demandé.

4 Le nom n'existe pas et DLN, LØN, LNS, LNU demandé.

5 Le suivant série (ou parallèle) n'existe pas et NXS (NXP) demandé. (fin de parcours).

143 Compactage des noms.

Certains noms sont compactés sur 6 caractères de la manière suivante :

$$\text{Nom} = K_1\ K_2\ K_i\ K_n\ D^C \xrightarrow{\text{Compactage}} K_1\ K_2\ K_3\ f_1\ f_2\ f_3\ D^C = \text{Nom compacté}$$

avec $f_1 = f(n)$

$f_2 = f_2(K_i)$

$f_3 = f_3(K_i)$

Attention : La fonction de Compactage est NON-UNIVOQUE, il en résulte que deux noms différents peuvent se transformer en noms compactés identiques. Pour l'éviter, limiter la longueur des racines communes, sur les noms donnant lieu à compactage (noms d'items).

Remarque : Si δ est erroné, cela provoque toujours un retour en erreur (Code-retour = 2) quelle que soit la fonction demandée, même si δ est inutile à l'éxécution de cette fonction.

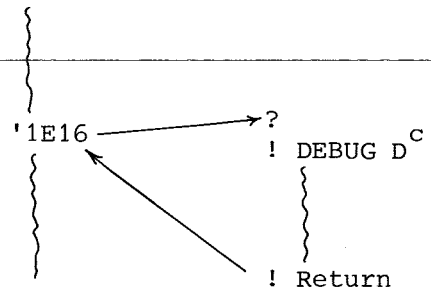
Exemple : On fait un DLN avec δ > longueur nom + valeur → code retour=2

2 INSTRUCTION '1E16

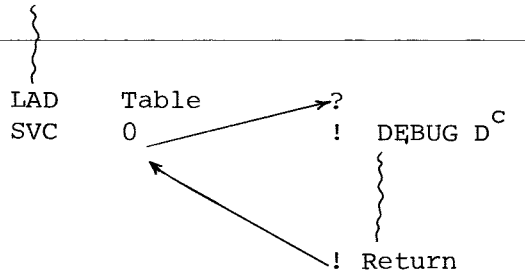
WØRD '1E16

Elle doit être implantée dans le programme par WØRD '1E16.

Elle assure une fonction semblable au SVC 0 (avec UL = 0 et f = 1), c'est-à-dire appel CCI interactif. La différence entre les deux étant qu'au retour dans le programme, AUCUN registre n'est modifié.



Sous !DEBUG, A, X, S, sont bons



Sous ! DEBUG : A, X, S, sont faux

RESTRICTION.

'1E16 ne doit être utilisé que si LECTEUR et IMPRIMANTE sont HORS-FONCTION sinon, il résulte de son exécution un "plantage" UC dû à une erreur de microprogramme T1600. Cette erreur n'existe pas sur le SOLAR. Par conséquent, sur T1600, '1E16 n'est utilisable que temporairement. (L'introduire par patch sur une instruction facile à patcher par exemple ADRI, LR,...)

③ INSTRUCTIONS '1Eα5 et '1E1D

WØRD '1Eα5

'1Eα5 exécutée en mode esclave, permet de réaliser les fonctions suivantes :

α = 0 Trappe programme (⟶retour CCI ?)
Mnémonique utilisable : ACTD

α = 1 Argument : A

résultat : - si A ⩾ 0 B = ((A)) A étant une adresse absolue.

- si A < 0 B = @ DCT dont NSP = (A).AND.'7FFF
(A se comportant comme un index).

α = 2 A et B reçoivent l'ACN de l'utilisateur

Exemple : A | : | S | B | Y | S |

α = 3 A := taille octet de l'espace mémoire utilisé

B := contenu de "BØX".

BØX : Les assignations Visu, LØN, LNS, LNU, entraînent le stockage en "box".

. du nombre d'octets réellement utilisés pour VISU

. de la longueur de la VALEUR pour LØN LNS LNU, même si on n'a pas lu toute la valeur.

Applications :

. Visu : on demande 10 caractères ; derrière le SVC on fait '1E35 ⟶ A contiendra le nombre d'octets réellement transmis (si l'utilisateur a envoyé ABD^{C} on aura A = 3 etc...)

. LØN : on demande un load name avec longueur du nom + ε et on fait '1E35 ⟶B contiendra la longueur de la valeur. Ceci permet par exemple d'AJUSTER son espace mémoire avant de faire un LØN de nom + valeur.

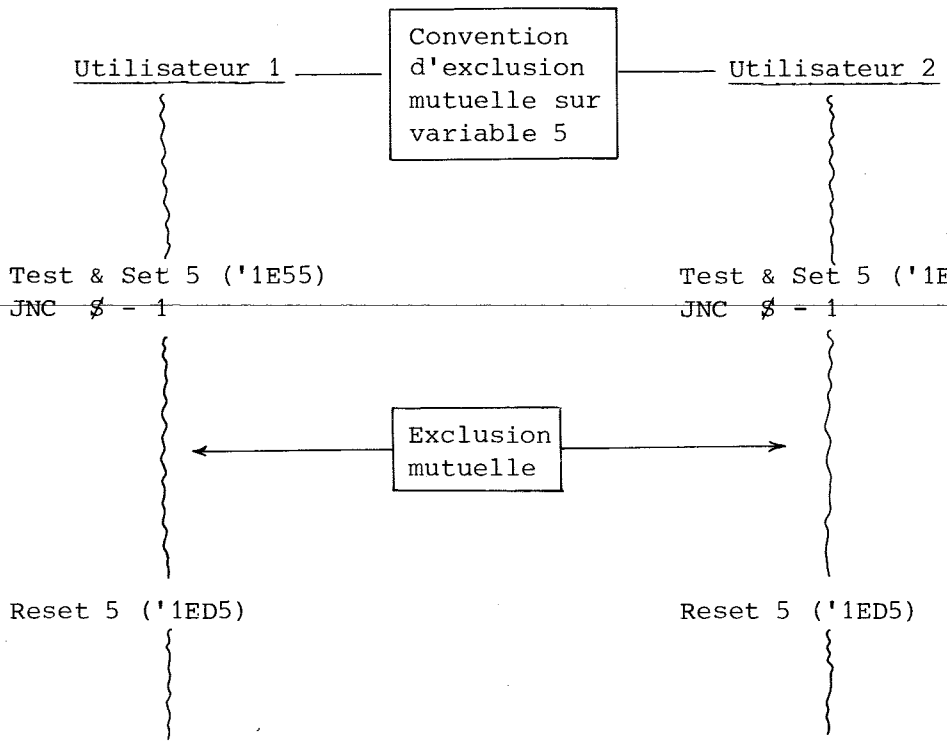
α = 4 A :=SYSID de l'utilisateur. (Cette fonction est utilisée par exemple par EDIT lors de la création de fichiers temporaires, pour éviter les homonymes).

α = 5,6,7 Reset 5,6,7

α = D,E,F Test and Set 5,6,7

} Synchronisation inter-programmes.

Exemple :



$\alpha = 8$ A := heure du jour exprimée en unité = 2 secondes, sur 16 bits, non-signé.

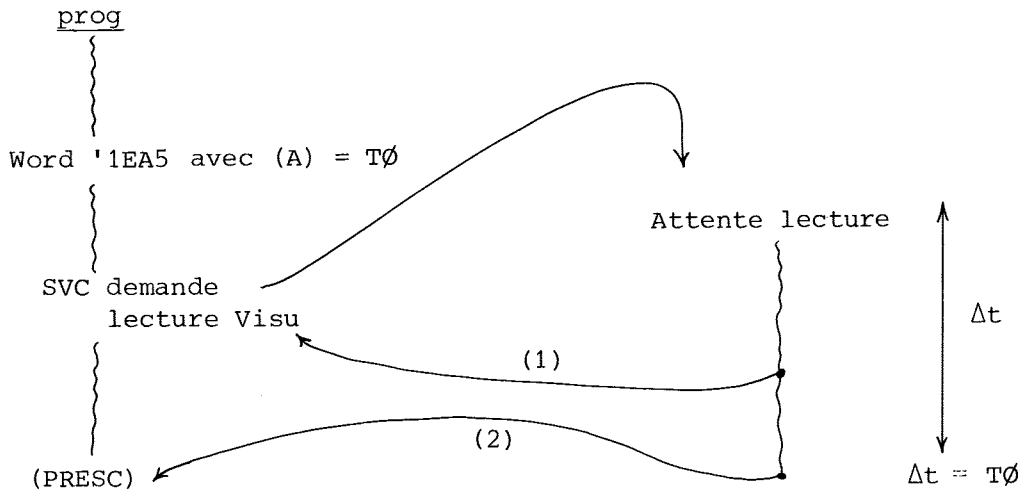
$\alpha = 9$ A := bits 0 - 7 Nom interne (NSP) organe d'entrée (assigné à 1)
bits 8 - 15 Nom interne (NSP) de VI1.

$\alpha = $ 'A Argument A

. si (A) = { bits 0 - 7 = 0 ; bits 8 - 15 = β caractère ASCII } β remplace le Alt-Mode pour le programme

. si (A) = { bit 0 = 1 ; bit 1 = 0 } alors, bits 2-15 sont pris comme décompteur de time out (TØ) sur la visu, en multiple de 5 secondes.

Exemple :



. retour 1 : réponse envoyée

. retour 2 : déroutement en PRESC si pas de réponse au bout de Δt = TØ

. Si (A) = { bit 0 = bit 1 = 1 ; bits 2-15 = 0 ⟹ Expansions et Extensions ('60 à '6F) visu sont autorisées.

bit 0 = bit 1 = 1 ; bits 2-15 ≠ 0 ⟹ Expansions et Extensions sont inhibées.

■ *Remarque importante.*

A chaque passage dans le CCI (état ?)

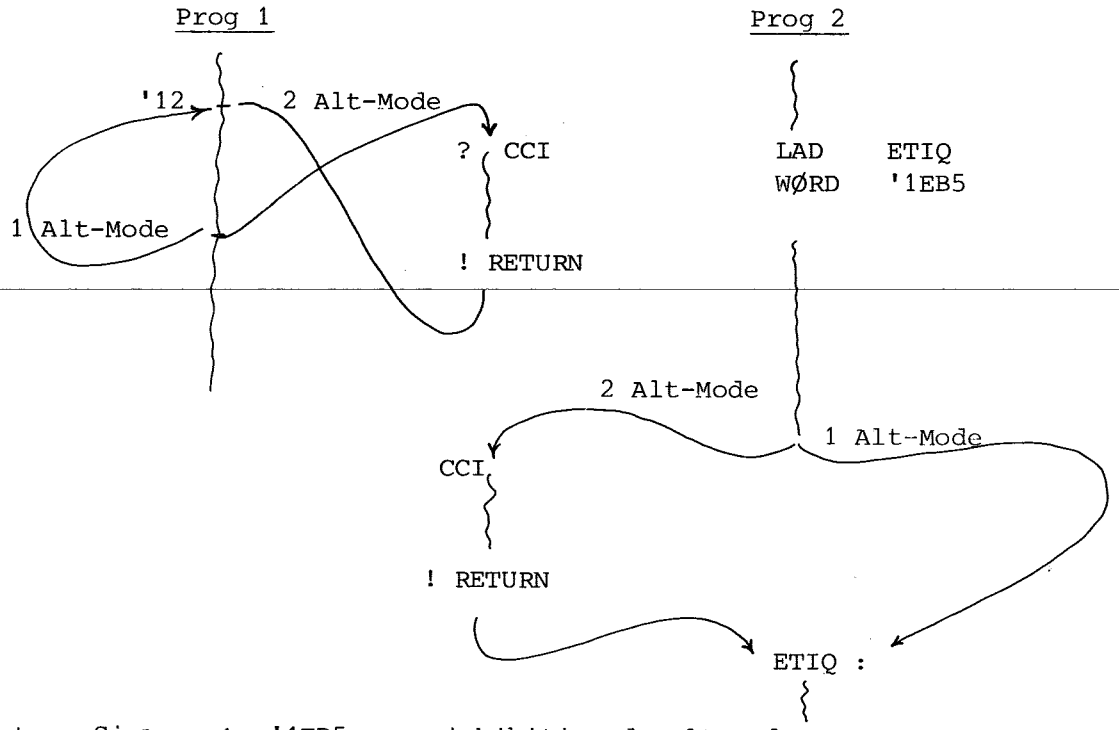
- Alt-Mode est RESTAURE
- Time Out est INHIBE
- Expansions et Extensions visus sont AUTORISEES

α = 'B

PRESC reçoit (A), Presc étant l'adresse programme à laquelle est dérouté le programme suite à 1 Alt-Mode ou ! Return.

Par défaut : Presc = '12

Exemple :



Nota : Si A = -1, '1EB5 ⟹ inhibition du Alt-Mode
(le programme reste cependant interruptible par 2 Alt-Mode).

α = 'C

Argument A ; sous ACN :SYS seulement.

Provoque l'éxécution en mode maître du sous-programme d'adresse (A), dans lequel l'utilisateur peut utiliser des SVC maîtres (cf : ci-après).

Ce genre de sous-programme doit être utilisé avec précautions.

WORD '1E1D

'1E1D exécutée en mode esclave (sur SOLAR seulement), et à condition d'avoir ACN =:SYS, permet d'exécuter un "store" (inverse de l'instruction '1E15),de la façon suivante :

Arguments : registres A, B et Y.

si Y = 0 : B est considéré comme une valeur absolue et sera donc pris tel quel.

si Y ≠ 0 : B est considéré comme une valeur translatable et le système prendra (B) + (SLØ * 16).

. si bit 0(A) = 0 : A est considéré comme l'adresse mémoire où (B) - ou ((B) + (SLØ *16)) - doit être stocké.

  si bit 0(A) = 1 : A (bits 8-15) est considéré comme un NSP (de 0 à '7F) et (B) - ou ((B) + (SLØ *16)) - sera stocké dans la table DCTSY des adresses de DCT, dans le mot correspondant au NSP désigné par A(bits 8-15).

Le mnémonique de '1E1D pour l'assembleur est STAR.

Exemple :

```
LAI     NSPVI5
SBT     0
WØRD    '1E15        < B reçoit l'adresse de la DCT VI5
LYI     0
LAI     NSPVI8
SBT     0
STAR                 < B, adresse de la DCT VI5 est stocké
                     < dans DCTSY à l'emplacement du
                     < NSPVI8. On aura donc dans
                     < DCTSY adresse DCTVI5 = adresse DCTVI8
```

④ PILE UTILISATEUR.

L'utilisateur doit prévoir dans sa pile 16 mots supplémentaires qui seront utilisés par le système en cas de défaut secteur (en SOLAR seulement).

De plus, il est nécessaire avant d'utiliser des SVC de s'assurer que le registre K pointe bien à l'intérieur de l'espace mémoire esclave. Sinon, on obtient des résultats imprévisibles avant que le système ne détecte l'erreur et s'arrête. Ceci, à cause d'une erreur de microprogrammation T1600.

⑤ SVC MAITRES (SOLAR seulement).

Plusieurs SVC maitres sont utilisables dans des sous-programmes maîtres appelés par '1EC5.

5.1 SVC 1

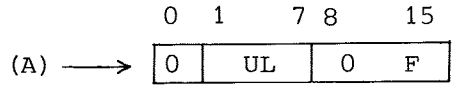
Provoque l'exécution d'un "release" du sémaphore d'adresse (A). Ce release est effectué par le système de façon programmée ; il ne faut en aucun cas utiliser l'instruction SOLAR RLSE !

5.2 SVC 2

Provoque l'insertion dans la trace circulaire du système du contenu du registre A. A utiliser pour la mise au point de sous-programmes maîtres.

⑥ TEST D'ASSIGNATION D'UNE UL.

Pour savoir si une unité logique est assignée ou non, par programme, il suffit d'écrire un SVC 0 avec :



et ensuite de tester le code retour :

- X = '81 : UL non-assignée.
- X = '83 : Fonction inexistante..... donc l'UL est assignée.

⑦ ESPACE MEMOIRE ESCLAVE.

C'est l'espace compris entre (SLØ) et (SLE).

Ses 3 premiers mots sont utilisés par le système et suite à :

- initialisation
- double Alt-Mode
- trappe programme

Ils ont la valeur suivante :

mot 0	'1C00 (SVC 0)		P ⟵ 0
mot 1	1 (retour CCi)	et	A ⟵ 1
mot 2	0 (pile)		K ⟵ 1

Dans ces 3 cas, faire !GO provoque donc un retour au CCi.

Nota : prévoir 3 mots pour le système en début de programme.

8 CODES-RETOUR : TABLEAU RECAPITULATIF.

CODE-RETOUR (X)	S I G N I F I C A T I O N
0	ØK
1	. SGF Deux OPEN successifs sans CLOSE . SGN Nom erroné (contient NULL ou mangue délimiteur D^C)
2	. SGF CLOSE demandé sans OPEN préalable . SGN Argument erroné (δ > longueur, par exemple)
3	. SGF WRITE demandé sans OPEN préalable . SGN Le nom existe et STN demandé . VIi Fonction inaccessible à l'instant t (exemple : EG = écriture en graphique sans ØG (open graphique) préalable.
4	. SGF READ demandé sans OPEN préalable . SGN Le nom n'existe pas et DLN, LØN, LNS, LNU demandé
5	. SGF DELETE demandé sans OPEN préalable . SGN Le suivant (série ou parallèle) n'existe pas et NXP(NXS) demandée (fin de parcours) . VIi Erreur en mode graphique . CR1 Longueur erronée (> 160) . LP1 Longueur erronée (> 132)
6	. SGF READ demandé et l'enregistrement est vide ou terminé

CODE-RETOUR (X)	S I G N I F I C A T I O N
8	. SGF n1 incorrect (clé = (n1,n2))
9	. SGF n2 incorrect (clé = (n1,n2))
'11	. CR1 écriture demandée . LP1 lecture demandée . VIi la fonction n'existe pas
'13 '93	. VIi S^C reçu pendant (avant) une sortie
'41	. DKU adresse secteur inexistante
'7D	. VIi Alt-Mode reçu (en général, non exploité car le système s'est branché en "PRESC")
'81	Unité logique non-assignée
'82	N° d'unité Logique invalide
'83	Fonction invalide
'84	Compte d'octets erroné
'85	. violation mémoire . UL ⟷ 'C (disque scratch) : n° de secteur ∉ [0,31] T1600 ∉ [0,39] SØLAR
'86	Adresse table d'arguments erronée
'2000 '4000 '6000 '8000	SGF } la clé n'existe pas et OPEN OLD KEY demandée SGF } SGF l'ensemble des clés est vide et OPEN OLD KEY demandée SGF la clé existe et OPEN NEW KEY demandée
'900D	DKU violation écriture.

ADDITIFS IUS 1/8/79

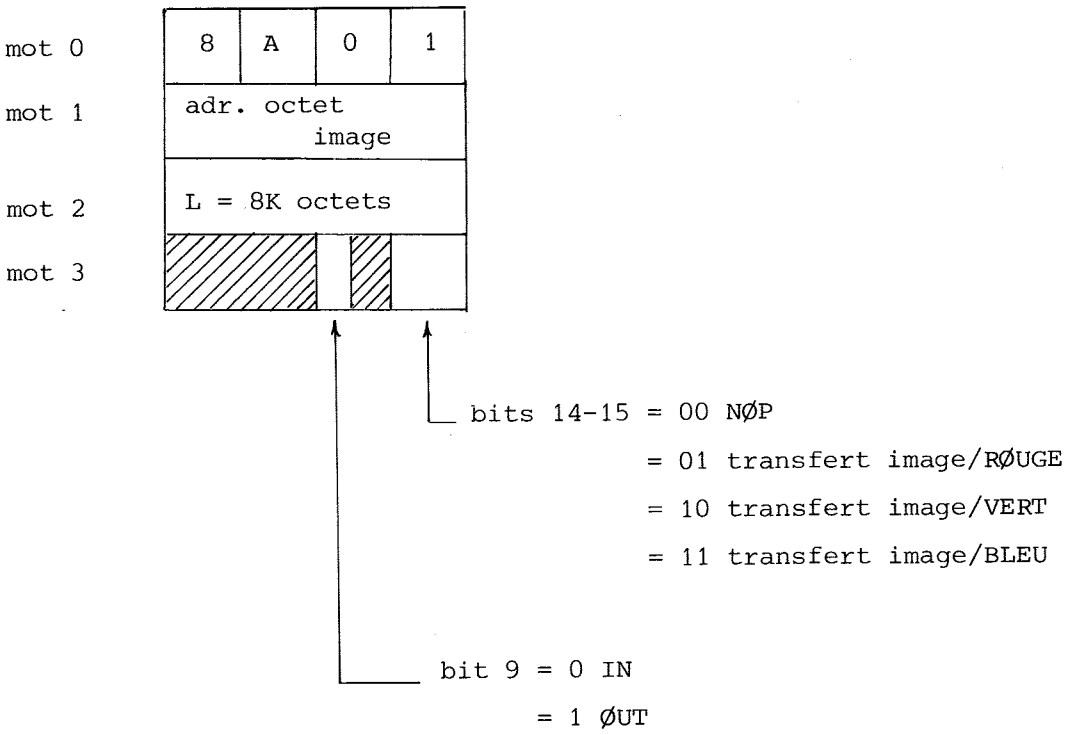
Nouvelles "assignations implicites"

Sur Solar on dispose de 4 nouvelles "assignations implicites" : l'octet 0 du mot 0 d'une demande peut contenir les 4 nouvelles valeurs suivantes :

- octet 0 du mot 0 = '8A : demande sur DKU (voir ci-dessous)
- " " " " " = '8B : demande sur CU3
- " " " " " = '8C : demande sur organe d'entrée
- " " " " " = '8D : demande sur organe de sortie.

Les fonctions demandées sur CU3, organe d'entrée et organe de sortie sont les mêmes que lorsqu'on travaille en assignation explicite.

Les fonctions pouvant êtres demandées sur DKU sont les mêmes que lorsqu'on travaille en assigantion explicite (0, 2, 8, A : respectivement lecture/écriture, en quanté = 3, lecture/écriture en quanta = 1, comme on l'a déjà vu). Une nouvelle fonction existe (fonction 1) et est utilisée pour commander la diffusion télévision numérique Solar ; la demande avec cette fonction ayant le contenu suivant :



- IN : la composante sélectionnée est copiée dans la zone image spécifiée par la demande.

- ØUT : transfert inverse, c'est-à-dire que l'image spécifiée est copiée sur la composante sélectionnée.

SVCM 3

On dispose sur Solar du SVCM 3 qui provoque un échange mémoire/composante R, V, B. 4 k mots sont échangés, les arguments (sens de l'échange, composante sélectionnée, adresse mémoire) étant acquis en mémoire débanalisée où ils ont été préalablement positionnés.

Gestion de fichiers : récupération de la clé dans BØX

Les demandes :

- open new key
- open old key
- open next key

entraînent le renvoi dans la BØX (récupérable comme on l'a vu par '1E35) de la partie entière de la clé courante (N1) en bits 3 à 15 de BØX, et le bit 0 de BØX est positionné à 0 si la partie décimale (N2) de la clé est nulle, à 1 si N2 est non nul.

Les demandes :

- close save key
- close release key

entraînent le renvoi dans la BØX de la partie décimale de la clé courante (N2)

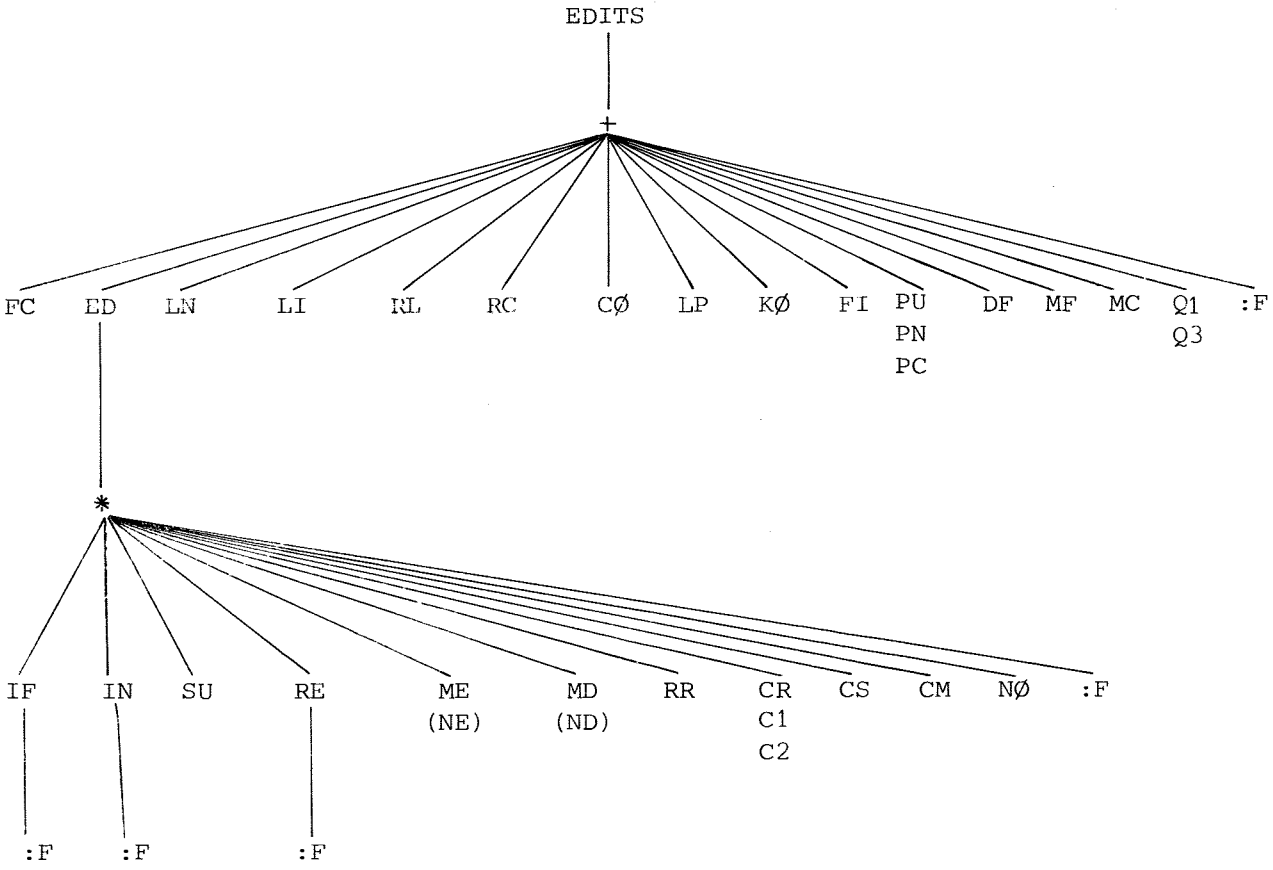
Ainsi, on peut par exemple, lors de l'exploration séquentielle de toutes les clés d'un fichier (par "open next"), récupérer au fur et à mesure les valeurs de ces clés. En effet, il suffit de récupérer la BØX après l'open next, ce qui donne N1, et permet de savoir si N2 est nul ou pas. Si N2 n'est pas nul, il suffit de faire une "close save" pour l'obtenir.

6 - LES PROCESSEURS

EDITEUR DE FICHIER : EDITS

Cet éditeur de fichier est appelé par ! CALL (avec mode = Q et nom = EDITS). Pour lancer le programme, faire !GØ D^{C}, on est alors sous EDITS.

L'arbre ci-dessous schématise les sous-commandes de EDITS.



:F sous + : sortie d'EDITS ⟶ ?

:F sous * : sortie de ED ⟶ +

:F sous IN : sortie de IN ⟶ *

:F sous RE : sortie de RE ⟶ *

:F sous IF : sortie de IF ⟶ *

ED

ED 'nom de fichier' D^{C}

ED agit sous l'état + c'est-àdire au niveau fichier. Si le fichier spécifié existe, il est reconnu; sinon il est créé. La prise en compte de ED

fait passer à l'état * qui est le niveau données (ou "enregistrement" ou "lignes") où l'on dispose des sous commandes IN, IF, SU, RE, ME, MD, NE, ND, RR, CR, CS, CM, NØ.

Pour revenir à l'état +, faire : F.

IN Insertion

```
IN n° de ligne D^C
```

Demande d'insertion derrière la ligne spécifiée par n° de ligne. Pour insérer en début de fichier, faire IN0. Les lignes sont numérotées par le système de 1 à n par pas de 1. Pour sortir de IN, faire : F

(différence entre D^C et RETURN : cf RR)

SU Suppression

```
SU n [-m] D^C
```

Suppression des lignes n à m (ou seulement de la ligne n si m n'est pas spécifié).

RE Remplacement

```
RE n [-m][p] D^C
```

Si p est spécifié, remplacement à partir de la colonne p des lignes n à m (ou de la ligne n seulement si m n'est pas spécifié).

Si p n'est pas spécifié, remplacement des lignes n à m (ou de la ligne n seulement) par les lignes rentrées ensuite. Dans ce cas pour sortir de RE, faire : F.

IF Insertion en fin de fichier

```
IF D^C
```

Insertion après la dernière ligne du fichier. Il est possible de faire plusieurs IF à la suite.

ME Modification avec écrasement

```
ME   n [-m] D^C
CH1  = ααα D^C
CH2  = βββ D^C
```

Recherche dans les lignes n à m la chaîne de caractères CH1 et, chaque fois qu'elle est trouvée, remplacement par la chaîne de caractères CH2. Si CH2 est plus longue que CH1, écrasement des caractères qui suivaient CH1.

Exemple

```
ME   1 D^C
CH1 = A D^C
CH2 = XX D^C
```

1 ABC ABC ⟶ 1 XXC XXC

Si CH2 est plus courte que CH1, elle est complétée par des blancs :

ME 1 D^{C}
CH1 = XX D^{C}
CH2 = AD^{C}

1 XXC XXC ⟶ 1 A ∅ C A ∅ C

MD Modification avec déplacement

MD n [-m] D^{C}
CH1 = ααα D^{C}
CH2 = βββ D^{C}

Recherche dans les lignes n à m la chaîne de caractères CH1 et, chaque fois qu'elle est trouvée, remplacement par la chaîne de caractères CH2. Si CH2 est plus longue que CH1, déplacement vers la droite des caractères qui suivaient CH1.

Exemple :

MD 10 D^{C}
CH1 = A D^{C}
CH2 = XX D^{C}

10 ABCABC ⟶ 1 XXBC XXBC

(pertes des caractères qui seraient déplacés après la colonne 80)

Si CH2 est plus courte que CH1, déplacement vers la gauche des caractères qui suivaient CH1 :

Exemple

MD 1 D^{C}
CH1 = XX D^{C}
CH2 = AD^{C}

1 XXBCXXBC ⟶ 1 ABC ABC

■ *Remarque.*

Les commandes NE et ND sont équivalentes à ME et MD, mais dans ce cas on ne demande pas CH1 et CH2 ; les commandes sont effectuées avec CH1 et CH2 fournies auparavant.

RR Remplacement

RR n [-m] D^{C}

Liste la ligne i pout tout i $\in \{n,m\}$. Possibilité de se déplacer sur cette ligne au moyen de I^{C} et H^{C} et de réécrire les caractères choisis (ne pas confondre SPACE qui met un blanc et I^{C}). D^{C} reliste la ligne modifiée alors que RETURN liste la ligne suivante.

CR Lecteur de cartes

CR D^C

Le clavier est remplacé par le lecteur de cartes pour l'entrée des données.

CS Version rapide de CR

CS D^C

CM Mode compressé

CM D^C

Indique que la séquence de cartes qui va suivre est en compressé

NØ Mode normal

NØ D^C

Indique que la séquence de cartes qui va suivre est normale (non compressée).

LN Liste fichier

LN 'nom fichier' [n [-m]] DC

Cette commande permet de lister le fichier de la ligne n à la ligne m. Si m n'est pas servi, liste de la ligne n jusqu'à la fin du fichier. Si n n'est pas servi,tout le fichier sera listé. Si n=m liste de n jusqu'à la fin du fichier.

Sur la liste, les lignes sont numérotées de 1 en 1. Le numéro de ligne indique le rang de la ligne dans le fichier mais il n'appartient pas à la ligne.

LI Liste fichier

LI 'nom fichier' [n [-m]] D^C

Identique à LN mais sans numérotation.

CØ Copie

CØ 'fichier1' [n [-m]] 'fichier2' [p] D^C

Copier le fichier 1, lignes n à m (ou tout le fichier) sur le fichier 2 derrière la ligne p (ou à la fin du fichier 2 si p n'est pas précisé)

Exemple : CØ 'F1' 2-3 'F2' 1 D^C

```
F1 : 1 AAA          F2 : 1 XXXX   ⟶   F2   1 XXXX
     2 BBB               2 YYYY            2 BBB
     3 CCC                                 3 CCC
     4 DDD                                 4 YYYY
```

LP Basculement visu - imprimante

LP D^C

(pour liste de fichier sur imprimante).

DF Suppression d'un fichier

DF 'nom fichier' D^C

Supprime le fichier spécifié.

RL Recherche et liste

RL 'nom fichier' [n [-m]] 'CH$_1$';'CH$_2$'..., 'CH$_n$' D^C
avec 'CH1' = 'CH1$_1$' 'CH1$_2$' ... 'CH1$_m$'

La virgule est l'opérateur 'ØR' et la concaténation l'opérateur 'AND'. Liste les lignes du fichier spécifié, comprises entre n et m et contenant la chaîne CH_1 ou la chaîne CH_2... ou la chaîne CH_n, CH_1 étant la concaténation des chaînes $CH1_1$ $CH1_2$... $CH1_m$. Si m n'est pas servi, la recherche aura lieu sur le fichier à partir de la ligne n jusqu'à la fin du fichier. Si n et m ne sont pas servis la recherche aura lieu sur tout le fichier.

■ *Remarque.*

1) si dans une des chaînes un des caractères est indifférent on peut le remplacer par $Ø^S$; dans ce cas, quelque soit le caractère rencontré à cette position, si le reste de le chaîne est bon, la ligne sera listée.
Ex : on veut rechercher tous les 1Ex5 dans un fichier.

RL 'nom fichier' '1E$Ø^S$5'

2) la formulation des chaînes de caractères recherchées n'est pas une grosse contrainte car 'OR' et 'AND' sont distributifs l'un par rapport à l'autre et toute expression logique peut se ramener à une somme ('OR') de produit ('AND').

RC Recherche et compte

RC 'nom fichier' [n [-m]] 'CH$_1$', 'CH$_2$',...,'CH$_n$' D^C
avec 'CH$_1$' = 'CH1$_1$' 'CH1$_2$'... 'CH1$_m$'

Identique à RL mais RC compte les lignes au lieu de les lister.

FI Exécution d'un fichier

FI 'nom fichier' [n [-m]] D^C

Exécute les commandes de EDITS contenues dans le fichier spécifié entre les lignes n et m (Si pas de m, ligne n à la fin. Si ni m ni n tout le fichier).

■ *Remarque*

Le fichier ne peut pas contenir lui-même de directives FI.

KØ Comparaison d'un fichier — KØ 'nom fichier' D^C

Permet de comparer un fichier sur carte au fichier spécifié.

EXEMPLE :

Si le fichier carte est en compressé, la séquence sera la suivante,

```
KØ 'TØTØ'
CR
CM

INO
  ← lecture du paquet de cartes
: F
⟹  réponse     IDENTIQUES
                    ou
          * * * DIFFERENTS * * *
```

PU Punch en symbolique — PU 'nom fichier' D^C

Punch du fichier spécifié en symbolique. Toujours global.

PN Punch en symbolique avec numérotation colonnes 73 à 80 — PN 'nom fichier' [n [-m]] D^C

Punch du fichier spécifié de la ligne n à la ligne m. Si pas de m, de la ligne n à la fin. Si ni n ni m, punch de tout le fichier.

PC Punch en compressé — PC 'nom fichier' D^C

Punch global du fichier spécifié.

■ *Remarque.*

Le punch s'arrête toutes les 128 cartes. Si on veut puncher les 128 cartes suivantes, taper un P. Si au contraire on veut sauter les 128 cartes suivantes, taper un S.

Attention : il faut mettre l'impression ØFF sur la perforatrice.

MF Modification fin — MF ['CH1'] D^C

Permet de modifier la commande : F en la remplaçant par les deux pre-

miers caractères de la chaîne CH1. La commande MF sans chaîne de caractères restaure la valeur :F

■ *Remarque.*

Il est impossible de remplacer le :F par une commande déjà existante :

Exemple :

MF 'ED' → commande rejetée

MC Modification du nombre de cartes par paquet — MC [n] D^C

Lors du punch en compacté, permet de modifier le nombre de cartes perforées entre chaque demande d'un caractère P ou S. Ce nombre vaut 128 par défaut. Cette commande permet en particulier de puncher directement la carte n.

FC Création d'un fichier de modifs — FC 'fichier 1' 'fichier 2' D^C

Enregistre sur le fichier 2 les modifications à effectuer sur le fichier 1. Génère : - au début de fichier 2 une ligne.

ED 'fichier 1'

- à la fin, un :F pour être compatible avec la commande FI.

Les commandes RR qui auraient été effectuées sous FC seront interprétées par FI comme des commandes RE.

Si le fichier 2 existe déjà la commande FC est rejetée.

Q1 Quanta = 1 — Q1 D^C

Le nombre de secteurs est égal à 1 à l'écriture des fichiers (compatibilité avec le TI600).

Q3 Quanta = 3 — Q3 D^C

Le nombre de secteurs est égal à 3 à l'écriture des fichiers. Cette valeur est la valeur prise par défaut.

<Ø Commentaires — < Ø commentaire D^C

Permet de placer des commentaires dans un fichier de modifications (utilisable par FI). Ces commentaires seront listés lors de l'interprétation de ce fichier par FI.

ADDITIFS EDITS 1/8/79

on peut rechercher une chaîne contenant le caractère "quote" ('). Il suffit de remplacer ce caractère par l'anti-slash (L^{S}).

Exemple : 'ABC\D'

- on peut tester l'absence de chaîne de caractères : il suffit de placer un tiret derrière la chaîne :

Exemple : RL 'nom fichier' 'ABCD' -

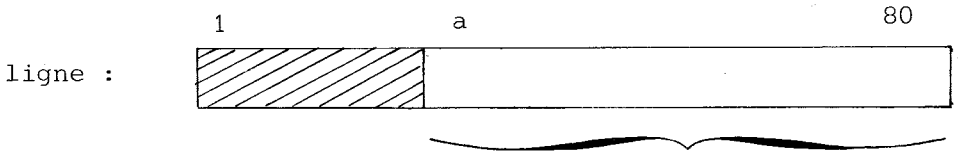
- on dispose des nouvelles commandes suivantes (sous le niveau +) :

- MA (marge): Utilisable sur visu 4014 seulement, elle provoque le passage de la visu à l'état "ESC;" (petite taille des caractères), et l'affichage d'une marge à droite, matérialisant la limite droite d'une ligne de n caractères ($0 \leqslant n \leqslant 80$).

Exemple : MA 75

- JP (nombre de lignes par page) : si l'on fait : JPO, il n'y aura pas de saut de page lors d'une liste de fichier sur imprimante (c'est l'état implicite). Si l'on fait par exemple JP55, il y aura saut de page toutes les 55 lignes.

- JE, JM, JI ("justification" ou cadrage) : permet d'obtenir de EDITS un cadrage de chaque ligne, lors de l'édition des listes de fichiers : ceci permet d'obtenir des textes plus présentables (notamment en listant avec LI qui ne sort pas les numéros de lignes). Pour chaque ligne, EDITS effectue le cadrage de la façon suivante (avec JIa, JEb, JMc)



x = nombre total d'espaces dans la ligne à partir de la colonne a.

EDITS calcule x, puis fait y:=x-b puis teste le résultat obtenu :

- si $y \leqslant c$: il effectue le cadrage
- si $y > c$: il ne fait pas de cadrage.

Ainsi, une ligne ayant beaucoup d'espaces ne sera pas cadrée (c'est par exemple la dernière ligne d'un paragraphe),et une ligne ayant peu d'espaces sera cadrée (EDITS rajoutera uniformément des espaces entre les mots pour que la marge de droite soit bonne).

Valeurs habituelles ; JI12, JE0,JM20

- NJ : cette commande supprime l'effet de toutes les commandes Jx précédentes (JP, JI, JE, JM). C'est le retour à l'état initial.

```
? CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=T
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=EDITS
?!GO
+LN'SI EXEMPLE'
FICHIER INEXISTANT
+ED'SI EXEMPLE'
*IN0
><
><          S I   E X E M P L E
><          ------------------
><
><            ILLUSTRATION DE QUELQUES COMMANDES
><          DE L'EDITEUR DE FICHIER 'EDITS'.
><
>!F
*!F
+
```

→ tabulation standard.

```
+LN'SI EXEMPLE'
     1  <
     2  <        S I   E X E M P L E
     3  <        ------------------
     4  <
     5  <          ILLUSTRATION DE QUELQUES COMMANDES
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'.
     7  <

+LI'SI EXEMPLE'
 <
 <        S I   E X E M P L E
 <        ------------------
 <
 <          ILLUSTRATION DE QUELQUES COMMANDES
 <        DE L'EDITEUR DE FICHIER 'EDITS'.
 <

+RC'SI EXEMPLE''DE'

NOMBRE DE LIGNES =0002
+RL'SI EXEMPLE''DE'
     5  <          ILLUSTRATION DE QUELQUES COMMANDES
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'.

+RL'SI EXEMPLE''DE''FICHIER'
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'.

+RL'SI EXEMPLE''DE''FICHIER','S I '
     2  <        S I   E X E M P L E
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'.

+
```

```
+LN SI EXEMPLE'
      1  <
      2  <          S I   E X E M P L E
      3  <          ------------------
      4  <
      5  <            ILLUSTRATION DE QUELQUES COMMANDES
      6  <          DE L'EDITEUR DE FICHIER 'EDITS'
      7  <

+DF'FC EXEMPLE'
+FC'SI EXEMPLE''FC EXEMPLE'
*RR3
 <          ********************
*IN7
><           POUR ILLUSTRER LA COMMANDE 'FC', PERMETTANT
><          D'OBTENIR AUTOMATIQUEMENT UN FICHIER DE
><          MODIFICATIONS EXPLOITABLE ENSUITE PAR LA COMMANDE 'FI'.
><
*:F
*:F
+LN'FC EXEMPLE'
      1   ED'SI EXEMPLE'
      2   RR3
      3   <          ********************
      4   :F
      5   IN7
      6   <           POUR ILLUSTRER LA COMMANDE 'FC', PERMETTANT
      7   <          D'OBTENIR AUTOMATIQUEMENT UN FICHIER DE
      8   <          MODIFICATIONS EXPLOITABLE ENSUITE PAR LA COMMANDE 'FI'.
      9   <
     10   :F
     11   :F

+
```

```
LN'SI EXEMPLE'
     1  <
     2  <          S I   E X E M P L E
     3  <          ------------------
     4  <
     5  <            ILLUSTRATION DE QUELQUES COMMANDES
     6  <          DE L'EDITEUR DE FICHIER 'EDITS'.
     7  <

+LN'FC EXEMPLE'
     1   ED'SI EXEMPLE'
     2   RR3
     3  <          **********************
     4  'F
     5   IN7
     6  <           POUR ILLUSTRER LA COMMANDE 'FC', PERMETTANT
     7  <          D'OBTENIR AUTOMATIQUEMENT UN FICHIER DE
     8  <          MODIFICATIONS EXPLOITABLE ENSUITE PAR LA COMMANDE 'FI'.
     9  <
    10  'F
    11  'F

+FI'FC EXEMPLE'
+LN'SI EXEMPLE'
     1  <
     2  <          S I   E X E M P L E
     3  <          **********************
     4  <
     5  <            ILLUSTRATION DE QUELQUES COMMANDES
     6  <          DE L'EDITEUR DE FICHIER 'EDITS'.
     7  <
     8  <           POUR ILLUSTRER LA COMMANDE 'FC', PERMETTANT
     9  <          D'OBTENIR AUTOMATIQUEMENT UN FICHIER DE
    10  <          MODIFICATIONS EXPLOITABLE ENSUITE PAR LA COMMANDE 'FI'.
    11  <

+
```

```
??
+CO'SI EXEMPLE'1-3'SI EXEMPLE'10
+LN'SI EXEMPLE'
     1  <
     2  <        S I   E X E M P L E
     3  <        ********************
     4  <
     5  <          ILLUSTRATION DE QUELQUES COMMANDES
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'
     7  <
     8  <         POUR ILLUSTRER LA COMMANDE 'FC', PERMETTANT
     9  <        D'OBTENIR AUTOMATIQUEMENT UN FICHIER DE
    10  <        MODIFICATIONS EXPLOITABLE ENSUITE PAR LA COMMANDE 'FI'.
    11  <
    12  <        S I   E X E M P L E
    13  <        ********************
    14  <

+ED'SI EXEMPLE'
*MD2-5
CH1=<        ********************
CH2=<        --------------------
*SU7-13
*:F
+LN'SI EXEMPLE'
     1  <
     2  <        S I   E X E M P L E
     3  <        --------------------
     4  <
     5  <          ILLUSTRATION DE QUELQUES COMMANDES
     6  <        DE L'EDITEUR DE FICHIER 'EDITS'.
     7  <

+
```

A S S Y S

1 - GENERALITES.

2 - SECTIONNEMENT DES PROGRAMMES ET ADRESSAGE

3 - LES CARTES DYNAMIQUES CD1 CD2 CD3

4 - CARACTERES SPECIAUX : @, %, # et '

5 - DIRECTIVES A L'ASSEMBLEUR ET A L'EDITEUR DE FICHIERS.

6 - EXPRESSIONS, OPERATEURS ET FONCTIONS.

7 - ANNEXES.

1) GENERALITES

1 - 1 Fonctions de l'assembleur ASSYS

1 - 2 Conventions à respecter pour l'utilisation de ASSYS

1 - 3 Utilisation de l'assembleur ASSYS

1 - 4 ASSYS : schéma organique

1 - GENERALITES

1 - 1 Fonctions de l'assembleur ASSYS

L'assembleur met à la disposition des utilisateurs les quatre fonctions suivantes.

ASSEMBLEUR — des instructions SOLAR et T1600 (1)

MACRO-ASSEMBLEUR — utilisation conjointe des directives de macro-assemblage développées une à une dans cette documentation.

EDITEUR DE FICHIERS — par la mise à disposition de l'éditeur de fichier EDITS dont toutes les fonctions sont inclues dans ASSYS.

INTERPRETEUR — car ASSYS autorise l'utilisateur à génèrer en cours d'assemblage des instructions en langage machine et à les exécuter.

1 - 2 Conventions à respecter pour l'utilisation de ASSYS.

- Respecter les normes de présentation des programmes données en annexe.
- S'efforcer de donner aux noms de fichiers :
  SOURCES des noms de racine : "S I Ø"
  BINAIRES " " " " : "B Ø Ø"
- Utiliser la tabulation standard c'est-à-dire :

0 étiquette	9 code Opération	21 arguments	37 Commentaires
E2306:	EQU LA	Ø & AXNUM	< INITIALISATION

(cette tabulation est automatiquement générée sur les visus par le processeur de base ! CALL mode T.)

---

(1) Le langage assembleur SOLAR-T1600 n'est pas développé dans cette documentation; il suffit de se reporter aux brochures "T1600" et "SOLAR", contenant toutes les instructions et leurs mnémoniques.

1 - 3 Utilisation de l'assembleur ASSYS.

Que ce soit en batch ou en interactif, le dialogue se déroule de la façon suivante :

```
? !CALL D^C
  }
MØDE = Q

NOM = ASSYS  D^C
```

? !ASSIGN 3 = N, BØ EXEMPLE D^C ⟶	assignation (facultative) du fichier binaire objet.
? !ASSIGNE 4 = ØD^C ⟶	liste sur l'organe de sortie
LP1 D^C ⟶	liste sur l'imprimante (implicitement : liste sur Ø)

```
? !GØ D^C

>                       PRØG

>     <
      <               PRØGRAMME EXEMPLE
      <

                      CALL      # SI SOURCE 1# ⟶ appel de fichier source
                       {
>                     ED ⟶ appel de l'éditeur de fichiers

+  ED 'SI SOURCE 2' D^C
{
+  :F ⟶ sortie de l'Editeur de fichiers, on revient à ASSYS

>                     EØT       # SI SOURCE 2#
                       {
>                     END ⟶ fin d'assemblage.
```

A noter, comme le montre le schéma organique de la page suivante, que l'on peut avoir plusieurs entrées possibles de source, dans un même assemblage.

- . entrée directe sur visu.
- . référence d'un fichier source sur disque.
- . référence d'un fichier cartes compressé, à lire sur le lecteur de carte (appel par EØT ## ; cf la directive EØT).

L'assembleur utilise les unités logiques de la façon suivante :

UL 1 et 2 : entrée/sortie de dialogue avec l'utilisateur.

UL 3 : fichier binaire objet en sortie.

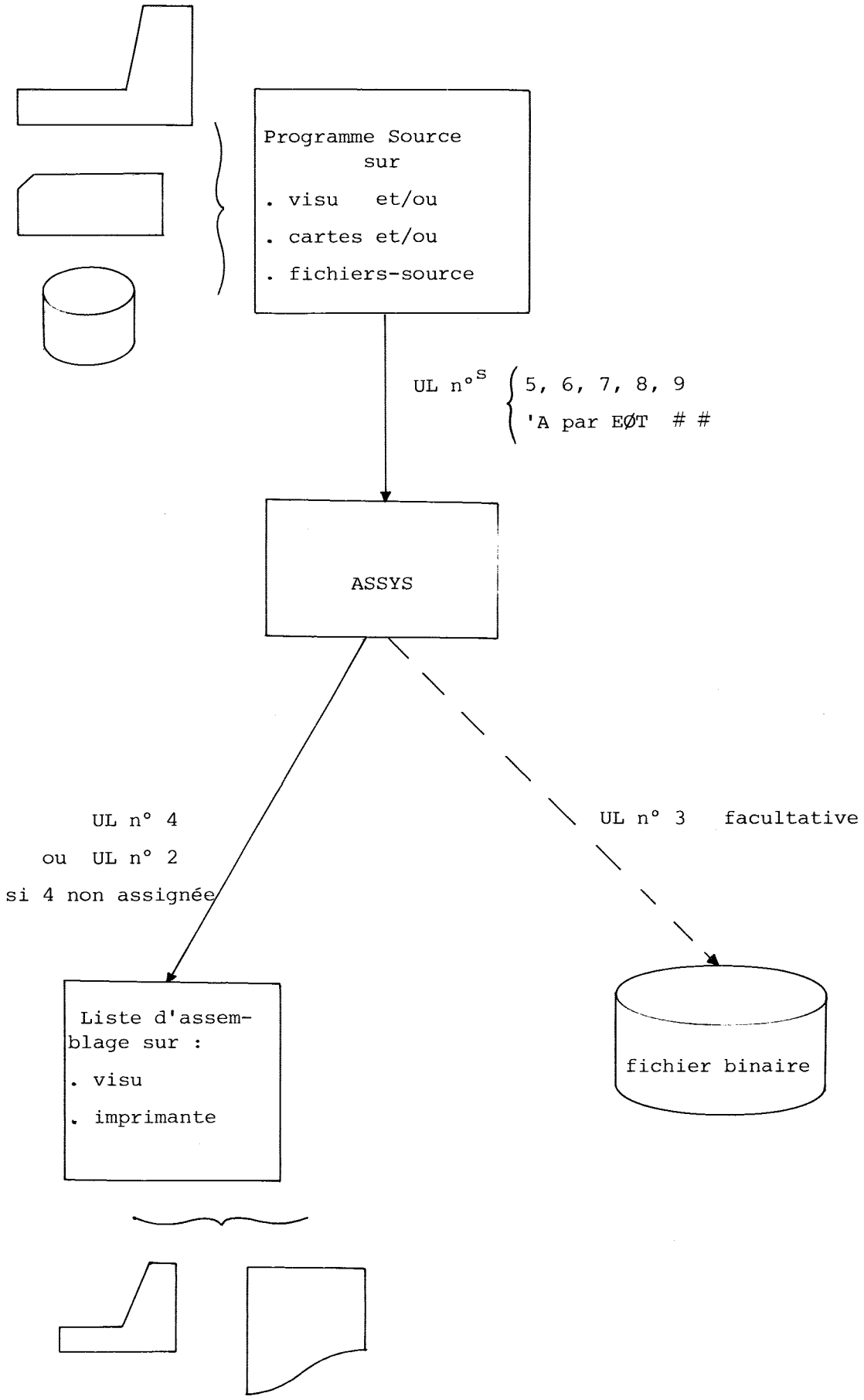
UL 4 : liste d'assemblage (elle sortira sur l'UL 02 si l'UL 4 n'a pas été assignée).

UL 5 et 9 : fichier référencé par EØT, 5 niveaux d'imbrication (cf : directive EØT).

UL 'A : fichiers-cartes référencé par EØT ## (cf : EØT)

UL 'B : utilisée par l'éditeur de fichier.

UL 'C : zone scratch disque.


Programme Source
sur
. visu et/ou
. cartes et/ou
. fichiers-source
UL n°S
5, 6, 7, 8, 9
'A par EØT # #
ASSYS
UL n° 4
ou UL n° 2
si 4 non assignée
UL n° 3 facultative
Liste d'assem-
blage sur :
. visu
. imprimante
fichier binaire

2) SECTIONNEMENT DES PROGRAMMES ET ADRESSAGE

2 - 1 Directive PRØG : Section Programme

2 - 2 Directive CØMMØN : Données Commune

2 - 3 Directive LØCAL : Données locales

2 - 4 Directive DSEC : Dummy Section

2 - 5 Directive USE : Affectation d'un registre de base

2 - SECTIONNEMENT DE PROGRAMMES ET ADRESSAGE

On dispose des directives suivantes :

2 - 1 Directive PRØG : Section "Programme"

PRØG

Ouvre une section pouvant contenir tout ce que l'on veut à l'exception de symboles à adressage basé.

2 - 2 Directive CØMMØN : Section "Commune"

CØMMØN

Ouverture d'une section "commune" contenant des symboles dont l'adressage sera basé.

- par le registre C en l'absence d'indication contraire.
- par l'un des registres C, L, W $\pm \Delta$ à définir par la directive USE.

2 - 3 DIRECTIVE LØCAL : Section "Locale"

LØCAL

Ouverture d'une section "locale" contenant des symboles dont l'adressage sera basé

- par le registre L en l'absence d'indication contraire.
- par l'un des registres C, L, W $\pm \Delta$ à définir par la directive USE.

2 - 4 Directive DSEC : Section Fictive

DSEC

Dummy section, dont les définitions de zones ne donnent pas lieu à génération de code ; les symboles qui y sont définis seront basés par l'un des registres C,L,W, $\pm \Delta$ à définir obligatoirement par la directive USE.

Nota : pas plus de 32 sections dans un même programme.

2 - 5 Directive USE ; Affectation d'un registre de base

USE

Affectation d'une base à une section CØMMØN ou LØCAL Ou DSEC

```
USE { C } , <expression>
    { L }
    { W }
```

Exemples

```
            }
          CØMMØN
CØ:       EQU       $
CØZ1:     DZS       -
CØZ2:     WØRD      -
            }
            }
CØTØTØ:   WØRD      '1234
            }
          PRØG
            }
          USE       C, CØ + '80
          LA        CØTØTØ
```

Remarques

1) Il ne peut y avoir qu'un CØMMØN ; en revanche ou peut avoir autant de LØCAL et et DSEC qu'on veut.
2) En l'absence de directive USE C et L sont utilisés par l'assembleur pour baser respectivement le CØMMØN et le dernier LØCAL défini.
3) <expression> indique à l'assembleur la valeur que contiendra à l'exécution le registre de base. Il en déduira les déplacements des zones référencées.
4) En l'absence de directive USE, l'assembleur fera comme si l'on avait écrit : "USE C, <adresse début du CØMMØN> + '80" pour un CØMMØN, et : "USE L, <adresse début du LØCAL> + '80" pour un LØCAL.

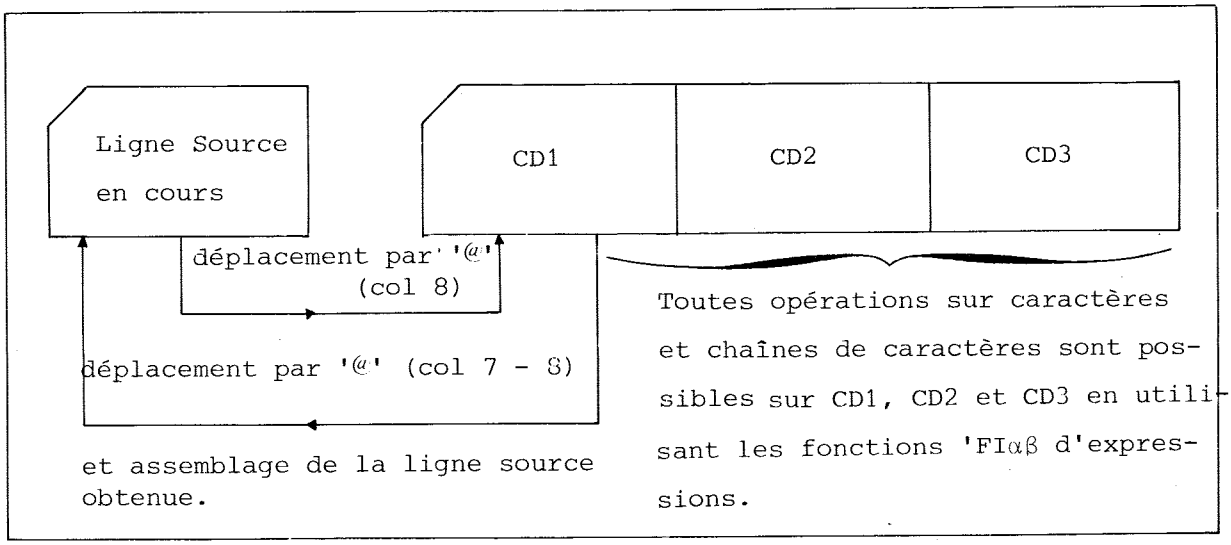
Nota : Pour changer de registre de base pour adresser une même section, il faut agir de la manière suivante :

```
                ≀
                ≀
                CØMMØN
CØ :            EQU         $
                ≀
                ≀
CØZ1 :          WORD        o
CØZ2 :          WORD        o
                ≀
                ≀
                PRØG
                ≀
                ≀
                USE         C,CØ+'80            <affection de 'C'
                ≀
                ≀
                LA          CØZ1                <CØZ1 est basée par 'C'
                LB          CØZ2                <CØZ2 est basée par 'C'
                ≀
                ≀
                USE         C,LØ+'80            <ce USE sur C doit être fait
                                                <même s'il est inutile  de ma-
                                                <nière à détruire l'effet du
                                                <USE C,CØ+'80 précédent (équi-
                                                <valent du DRØP que l'on ren-
                                                <contre dans d'autres assem-
                                                <bleurs).
                USE         W,CØ+'80            <c'est 'W' qui basera main-
                ≀                                tenant le common.
                ≀
                LA          CØZ1                <CØZ1 est basée par 'W'
                LB          CØZ2                <CØZ2 est basée par 'W'.
                ≀
                ≀
                ≀
```

3) LES CARTES DYNAMIQUES CD1 CD2 CD3

3 - LES CARTES DYNAMIQUES CD1 CD2 CD3

L'assembleur ASSYS offre à l'utilisateur des facilités de type "macro -assemblage", ceci par l'utilisation des procédures (appelables par CALL/EØT) ainsi que des cartes dynamiques . On a, schématiquement, ceci :



C'est toujours la ligne source en cours qui est assemblée par ASSYS, les cartes dynamiques servant à l'utilisateur de zones de travail et de passage de paramètres.

Les déplacements de 'ligne-source en cours' vers CD1 se font par le '@' colonne 8 (cf : @).

Le déplacements CD1 → 'ligne source en cours' + assemblage de la ligne source ainsi obtenue, se fait par @@ en colonnes 7 et 8 (cf : @). Noter que tous les autres caractères de la carte sont ignorés et perdus.

Tous déplacements, tests de caractères, recherches de caractères etc... dans CD1 CD2 CD3 sont possibles en utilisant l'opérateur '=' et les fonctions 'Fiαβ d'expression.

De plus les 'CDi' servent d'interface entre ASSYS et EDITS au moyen de la directive EE : celle -ci provoque l'appel de EDITS, et celui-ci va chercher ses commandes en 'CDi' à partir de la colonne o (CD1) et liste ses réponses au fur et à mesure de son travail dans cette même zone. Ainsi peut se réaliser un véritable dialogue ASSYS-EDITS permettant la génération de fichiers sous le contrôle de l'assembleur en fonction des paramètres gérés par l'utilisateur. Se reporter à la directive EE.

4) CARACTERES SPECIAUX @ % # '

4 - 1 @ en colonnes 2,3,4,5,23,24,...,33,34.

4 - 2 # @ en colonnes 7-8 et appel implicite de procédure

4 - 3 @ en colonne 8 : transfert en Carte Dynamique 1

4 - 4 @@ en colonnes 7-8 : déplacement et interprétation de la CD1

4 - 5 @ en colonne 9 : Code-opération.

4 - 6 @ en colonne 33 : récupération du niveau d'imbrication des NLS

4 - 7 @ en colonne 34 : récupération du type d'ordinateur.

4 - 8 @ autres colonnes : caractère assimilé alphabétique.

4 - 9 % : niveau d'imbrication des EØT / CALL

4 - 10 ' : génération de code hexadécimal par la directive de génération ASCI.

4 - CARACTERES SPECIAUX @ %

@

4 - 1

@ en colonnes 2,3,4,5,
23,24,25,26,27,...,33,34

@ en colonne 2 ou 23 est remplacé par le 1er
3 ou 24 est remplacé par le 2ème
4 ou 25 " " " " 3ème
5 ou 26 " " " " 4ème
} caractère de la zone "Adresse" (1)

27 est remplacé par le caractère
- ∅ si modification (même nulle) du compteur ordinal (voir note 2)
- ∅ sinon

28 " " " le caractère
- * référence en avant
- % constante adresse

29 " " " " 1er
30 " " " " 2ème
31 " " " " 3ème
32 " " " " 4ème
} Caractère de la zone "instruction"

33 " " " le niveau d'imbrication des NLS (cf : LST/NLS et le §4-6)

34 " " " le type d'ordinateur

De même que pour le %, si ASSYS a fait au moins une substitution, il l'indiquera en concaténant le caractère @ au numéro de ligne du source sur le listing d'assemblage.

---

(1) Pour la signification des termes "Zone adresse" "Zone instruction" etc..., voir en annexe la présentation de la liste d'assemblage.

(2) Modification (même nulle) du compteur ordinal : ∅ EQU, DZS x (même si X= o), code généré.

ASSYS 15

Exemple

@ col 7-8

SOURCE				LISTE D'ASSEMBLAGE			
183		PSR	A,B,X,	183	OABO ØØ 1AEO	PSR	A,B,X
184	XPSR:	VAL	'0C0000000 @@	184@	OOEO ØØ ØØ ØØ	XPSR: VAL	'000000000EO
185		JMP	ETIQ 1	185	OAB1 Ø O2OO	JMP	ETIQ1
186	XX @@@@@:	ASCI	"O@@@@@@@@@@"	186@	0AB2	XX0AB1: ASCI	"OOAB1Ø*O2OO"

4 - 2 # @ en colonne 7-8 et appel implicite de procédure

Lorsque l'assembleur rencontre #@ en colonnes 7-8 :

- il place la carte en carte-dynamique 1 (le @ remplacé par un blanc)
- il liste la carte sans le @ (remplacé par un blanc)
- il appelle la procédure placée entre # dans la zone commentaires. C'est ce que l'on nomme l'appel implicite de procédure, par opposition à l'appel explicite fait par EØT ou CALL.

Exemple

Dans CMS5 on veut générer automatiquement une pile en mémoire haute, alors que l'on est en mémoire basse, et se repositionner en mémoire basse :

On écrira :

```
LPILE :        VAL    10          < longueur de la pile
PILE1 :     #@ EQU    Ø-1         < # SI CMS5 - GENPILE #
```

La procédure SI CMS5 - GENPILE sera la suivante :

```
       EØT          # SI CMS5 DØL2 #      < positionnement en mémoire haute
@@                                         < génération de 'PILE' : EQU Ø-1'
       DZS          LPILE                 < réservation de l'emplacement
                                            de la pile
       EØT          # SI CMS5 DØL1 #      < repositionnement en mémoire
                                            basse
```

Nota : Un avantage important de l'appel implicite sur l'appel explicite est de pouvoir récupérer en début de procédure les zones adresse, indicateurs (Ø, * ,%), et instruction, de la dernièer ligne éditée, ce qui dispense d'un passage de paramètre, comme le montre l'exemple suivant :

Exemple :

@ col 7-8

On veut passer comme paramètre un code instruction et son adresse à la procédure SI PRØC qui l'exploitera :

1er moyen : Appel explicite

SOURCE	LISTE D'ASSEMBLAGE
41 SLLS 1	41 0 DBA ØØ 2981 SLLS 1
42 XWØRK1 : VAL ' OOOOOOO @@@@	42 2981 XWØRK1: VAL '00000002981
43 XWØRK2 : VAL Ø - ZERØ	43 0 DBA XWØRK2: VAL Ø - ZERØ
44 EØT # SI PRØC #	44 ODBA EØT # SI PRØC #

2ème moyen : Appel implicite

SOURCE	LISTE D'ASSEMBALGE
41 SLLS 1	41 0 DBA ØØ 2981 SLLS 1
42 # @ < # SI PRØC #	42 0 DBA ØØ 2981 #Ø < # SI PRØC #

Il suffira, en tête de la procédure SI PRØC de récupérer les zones adresse et instruction, comme ceci par exemple :

SOURCE	LISTE D'ASSEMBLAGE (suite)
1 XWØRK : ASCI "0 @@@@00@@@@"	42 0 DBA ØØ 2981 #Ø < # SI PRØC #
	SI PRØC
	1 @ 0 DBA ØØ 3030 XWØRK : ASCI "00 DBA00 2981"
	1 0 DBB ØØ 4442
	1 0 DBC ØØ 4130
	1 0 DBD ØØ 3032
	1 0 DBE ØØ 3938
	1 0 DBF ØØ 3100

4 - 3 @ en colonne 8 : transfert en carte dynamique 1

@ col 8 @@ col 7-8

Le @ placé en colonne 8 provoque le transfert de la carte en carte dynamique 1, le @ étant remplacé par un blanc ; l'image de la carte dynamique 1 est listée.

@ col 8
@@ col 7-8

Exemple :

SOURCE			LISTE D'ASSEMBLAGE		
288 CMUN : @ WØRD		1	288 CMUN	WØRD	1

4 - 4 @@ en colonnes 7-8 : déplacement et interprétation de la CD1

@@ placés en colonnes 7 et 8 provoquent le déplacement de la Carte-dynamique 1 en "ligne - source - en cours" et son interprétation.

Exemple :

SOURCE			LISTE D'ASSEMBLAGE		
288 CMUN :	@ WØRD	1	288	CMUN:WORD	1
289	@@		289 2AB4ØØ0001	CMUN:WORD	1

4 - 5 @ en colonne 9 : code opération

@
col 9

Lorsque le @ est placé en colonne 9, l'assembleur lui substitue le code opération associé à la valeur située dans les deux premiers octets de la zone adresse de la ligne précédente.

Exemple :

SOURCE				LISTE D'ASSEMBLAGE			
32 X :	VAL	'12>8		32 1200 ØØØØØØ X:	VAL	'12>8	
33	@	ETIQ		33 0DBA ØØ0C26	LBY	ETIQ	

Une application : générer conditionnellement l'une des deux séquences suivantes pour tester l'égalité entre une variable CMVAR et la valeur XVAL avec $-128 \leqslant XVAL \leqslant +127$:

si XVAL = 0,

alors générer :

```
LA      CMVAR
JAE     ETIQ0
}
```

si XVAL ≠ 0,

alors générer :

```
LA CMVAR
CPI XVAL
JE ETIQO
```

@
col 9

On écrira :

```
        LA    CMVAR
XWØRK1: VAL   XVAL
        EØT   # SI PRØC #
        @     ETIQ.0
```

avec SI PRØC ainsi définie :

```
              IF      XWØRK1, X%, ,X%
XWØRK1:       VAL     CØDJAE
X% :          VAL     0
              IF      XWØRK1, , XWØRK1,
              CPI     XWØRK1
XWØRK1:       VAL     CØDJE
X% :          VAL     0
X% :          VAL     XWØRK1 > 8
```

4 - 6 @ en colonne 33 : récupération du niveau d'imbrication des NLS

Si l'on écrit :

```
XWØRK:  VAL '0 0000 0000 00 @
```

alors on récupére :
- 0 si liste ØN
- 1 si 1 NLS
- 2 si 2 NLS imbriqués
- 3 si 3 NLS imbriqués

etc...

4 - 7 @ en colonne 34 : récupération du type d'ordinateur

Le @ placé en colonne 34 est remplacé

. par le caractère "S'' si ordinateur Solar.

. par le caractère "T" si ordinateur T1600.

Exemple :

```
XASCI:  ASCI    "ØRDINATEUR = ⊬ @" donnera
        "ORDINATEUR = ⊬ S" si SOLAR
        "ORDINATEUR = ⊬ T" si T1600
```

ASSYS 19

4 - 8 @, placé dans une colonne où il ne fait pas l'objet d'une interprétation particulière, est assimilé à un caractère alphabétique

@
autres col.

On peut donc écrire par exemple ceci :

```
@ @ 1:   EQU   ⌀
```

Les colonnes où @ ne fait pas l'objet d'une interprétation particulière sont les suivantes :

```
col  0
     1
     6
    10
    11
    12
     .
     .
     .
    21
    22
    35
    36
     .
     .
     .
    79
```

On peut donc écrire par exemple ceci :

SOURCE				LISTE D'ASSEMBLAGE			
328	XWØRK:	VAL	'2A	328 002A	XWØRK:	VAL	'2A
329	@@@@@@:	EQU	⌀	329@07B8	@@OO2A:	EQU	⌀
330		LXI	XWØRK2	330 07B8⌀⌀1109		LXI	XWØRK2
≀		≀		≀ ≀ ≀		≀	
421	XWØRK:	VAL	XWØRK	421 OO2A	XWØRK:	VAL	XWØRK
422		JMP	@@@@@@	422 @07C1⌀⌀02F7		JMP	@@ 00 2A

4 - 9 Caractère spécial % : niveau d'imbrication des EØT/CALL [%]

Ce caractère, placé n'importe où sur une ligne source (position 0 à 79) est remplacé - avant analyse de la ligne source - par un chiffre de 4 à 9 en fonction du niveau d'imbrication des EØT (CALL).

Exemple :

```
        début d'assemblage
          }
XWØRK%1:  VAL      %%   ——— sera remplacé par ——→ XWØR41:          VAL     44
          }
          EØT      # P1 #
                     }
                   XWØR%1:  VAL  "%A"  ————————→  XWØR51:          VAL     "5A"
                     }
                   EØT      # P2-% #  ————————→                  EØT # P2-5 #
                              }
                            XWØR%1:        VAL 0 ——→ XWØR61:       VAL 0
                              }
                            etc...
```

Permet notamment de définir des symboles "locaux" à une procédure.

NOTA : Lorsque l'assembleur fait au moins une substitution, il le signale sur la liste d'assemblage en concaténant le caractère @ au numéro de ligne du source (cf : exemples de listings d'assemblage).

ATTENTION :

Pour comparer (A) à "%" il vaut mieux écrire CPI '25
plutôt que CPI "%" !

Même recommandation pour d'autres cas de figure...

4 - 10 Caractère spécial ' : génération de code hexadécimal par la directive de génération ASCI [']

En assembleur T1600 et SOLAR, la directive ASCI est utilisée pour générer des caractères alphanumériques, par exemple :

```
ASCI            "ABCDEF"
```

ASSYS permet de l'utiliser pour générer du code hexadécimal de la façon suivante :

```
ASCI            "'______________"
                  \______________/
                  chaîne de caractères hexadécimaux 0-9A-F
```

auquel cas ASCI équivaut à un WØRD "long".

Exemples :

```
ASCI        "DEAD"   génère :    44 45 41 44
ASCI       "'DEAD"  }
WØRD        'DEAD   } génère :   'DEAD
ASCI       "'000128C42821F642"          }  ces deux façons d'écrire
WØRD        '0001;'28C4;'2821;'F642     }  sont équivalentes.
```

Attention :

On ne peut générer le caractère alphanumérique' en tête de chaîne alphanumérique par la directive ASCI.

Il faut donc écrire pour générer par exemple la chaîne alphanumérique < 'ABCDEF > :

```
BYTE '27;'41
ASCI "BCDEF"
```

5) DIRECTIVES A L'ASSEMBLEUR ET A L'EDITEUR DE FICHIERS

5 - 1 CALL : appel de fichier/procédure

5 - 2 CCI : retour au CCI

5 - 3 DATE : date et heure

5 - 4 DØ : itération

5 - 5 DEF et REF : définition et référence de symboles externes

5 - 6 ED : appel de l'éditeur de fichier

5 - 7 EE : demande d'interprétation de commandes à l'éditeur de fichiers.

5 - 8 END : fin d'assemblage

5 - 9 EØT : appel de fichier/procédure

5 - 10 EST : édition de la table des symboles

5 - 11 IDP : identification du programme.

5 - 12 IF : assemblage conditionnel

5 - 13 LST/NLS : autorisation/inhibition de la liste d'assemblage

5 - 14 MØT : directive équivalente à VAL

5 - 15 NDS : édition de la liste des symboles non définis

5 - 16 PAGE : saut de page

CALL

5 - 1 Directive CALL : appel de fichier/procédure

Cette directive est strictement équivalente à EØT.

5 - 2 Directive CCI : retour au CCI

CCI

Est utile notamment pour introduire des points d'arrêt dans un assemblage, ainsi que pour modifier des assignations.

Exemple :

```
? ! CALL D^C
}
MØDE = Q
NØM = ASSYS D^C ——————————————→ appel de l'assembleur
? ! ASSIGN 3=N, BØØPRØGRAMME D^C ——————→ fichier binaire objet
? ! ASSIGN 4=Ø D^C ——————————————→ liste sur l'organe de sortie
? ! GØ D^C
                    EØT # SI ∅ PRØGRAMME #
                                  }
                              CCI ——————→ provoque le retour au CCI
? ! ASSIGN ∅ 4=S D^C   )
? ! ASSIGN ∅ 4=LP1 D^C ) ——————→ assignation de l'UL 4 à LP1
                                  la liste sortira donc sur l'imprimante
? ! GØ D^C ——————————————→ reprise de l'assemblage à la ligne suivante dans le fichier 'SI ∅ PRØGRAMME'
                                  }
                                  }
                                  }
                                  }
                                  }
                                etc...
```

5 - 3 Directive DATE

DATE

A la rencontre de cette directive, ASSYS liste la date et l'heure

5 - 4 Directive DØ : itération

DØ

DØ < expression >

A la rencontre de cette directive, ASSYS calcule n = valeur de l'expression ; ensuite il traite n fois la ligne suivante du source.

L'intérêt de cette directive est principalement :

. que l'on peut itérer un VAL, un CALL, un EØT (entre autres).

. que l'on peut récupérer par une des fonctions 'Fi de l'opérateur '=' l'indice courant de la boucle DØ

Il en résulte de très larges possibilités d'utilisation.

Exemples :

On veut générer un masque XMASK dont les XNBITS premiers bits soient à 0 et les autres à 1 ; il suffit d'écrire :

```
XMASK :   VAL    0
          DØ     16 - XNBITS   ≡   XMASK : VAL 1 > XNBITS - 1
XMASK     VAL    XMASK > 1 ?1
```

$\left(2^n - 1 = 2^{n-1} + 2^{n-2} + \ldots 2^0\right)$

Voir aussi les exemples d'utilisation de DØ particulièrement dans l'exemple d'utilisation de la fonction 'FE02, et dans la procédure SI PAGE donnée en annexe.

DST / EST

5 - 5 Directives DEF et REF : définition et référence de symboles externes

DEF < symbole >
REF < symbole >

DEF permet de définir un symbole que l'on référencera dans un autre programme, les liens seront résolus lors du LINK.

REF permet, à l'inverse de référencer un tel symbole.

Exemple :

On veut assembler séparément deux sources : SI ∅ S1 et SI ∅ S2 pour obtenir deux programmes objet : BØ ∅ S1 et BØ ∅ S2 qui seront linkés ensemble pour donner le programme PRØG. Ces deux modules utilisent DEF et REF pour certains symboles.

SI ∅ S1

```
        CØMMØN
          ⌇
        REF     SP2
        DEF     SP1
ASP2:   WØRD    SP2
ASP1:   WØRD    SP1
          ⌇
        PRØG
          ⌇
        BSR     ASP1
          ⌇
        BSR     ASP2
          ⌇
SP1:    EQU     ∅
          ⌇
        RSR
          ⌇
        END
```

SI ∅ S2

```
        CØMMØN
          ⌇
        REF     SP1
        DEF     SP2
ASP1:   WØRD    SP1
ASP2:   WØRD    SP2
          ⌇
        PRØG
          ⌇
        BSR     ASP1
          ⌇
        BSR     ASP2
          ⌇
SP2:    EQU     ∅
          ⌇
        RSR
          ⌇
        END
```

LINK

```
? ! CALL D^C Q  LINK D^C
? ! GØ
FICH = BØ ∅ S1 D^C
FICH = BØ ∅ S2 D^C
FICH = D^C
NOM = PRØG D^C
RUN 'xxxx
    SP1 < adresse >
    SP2 < adresse >
?
```

ASSYS 28

ED

5 - 6 Directive ED : édition de fichier

A sa rencontre, l'assembleur charge en overlay le fabuleux éditeur de fichiers 'EDITS' avec lequel peut alors s'établir un dialogue, l'assemblage étant suspendu jusqu'à ce que l'on sorte de EDITS pour revenir à 'assembleur.

Restriction

La directive ED n'est pas acceptée dans un séquence de source appelée par EØT. La directive EE est moins restrictive (cf:EE)

Exemple :

```
? ! CALL D^C
}
MØDE = Q
NØM = ASSYS D^C
? ! ASSIGN ⌀ 3 = N, BØ ⌀ PRØG D^C
? ! ASSIGN ⌀ 4 = LP1 D^C
? ! GØ D^C
≥   EØT # SI PRØG #
≥XX : VAL 0                          } dialogue avec ASSYS
      .
      .
      .
≥   ED
± ED 'SI PRØG1'
✱ IN528
≥ ∿∿∿∿
≥ : F
✱ : F
± FI 'MØDIFS'                        } dialogue avec EDITS. Noter qu'on dispose de
± ED 'SI PRØG 2'                      toutes les commandes y compris FI.
      .
      .
      .
      etc...
✱ : F
± : F
>   LR  A,B                          } dialogue avec ASSYS
≥   etc
```

5 - 7 Directive EE : interprétation de commandes à l'éditeur de fichiers EDITS, générées en cartes dynamiques

EE

De la même façon que ED, on peut ainsi disposer de l'éditeur de fichier EDITS.

Mais EE va plus loin :

1°) On peut utiliser EE dans une séquence référencée par EØT/CALL au permier niveau et au second niveau, mais pas au-delà (on rappelle qu'avec EØT ## ou reste au même niveau d'imbrication).

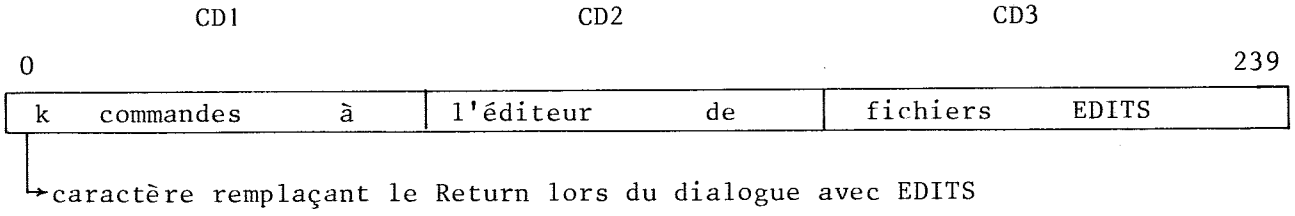
```
≥ on est sous ASSYS
  ≀
≥ EØT  # SI 1 #
         ≀
        EE                    ——————→   autorisé
         ≀
        EØT      # SI 2 #
                   ≀
                  EE          ——————→   autorisé
                   ≀
                  EØT   #  SI 3 #
                             ≀
                            EE  ——————→   interdit
```

2°) Les commandes à EE, placées dans les cartes dynamiques CD1, CD2 et CD3 peuvent elles-mêmes avoir été générées sous ASSYS.

3°) Les réponses de l'éditeur parviennent en cartes dynamiques et sont donc utilisables sous ASSYS pour générer soit des commandes à ASSYS soit des commandes à EDITS.

Autrement dit, les cartes dynamiques CD1, CD2 et CD3 constituent une zone de dialogue entre ASSYS et EDITS, que l'utilisateur peut manipuler à sa guise.

Le format des commandes à l'éditeur en cartes dynamiques est le suivant :



ASSYS 30

Exemple 1 :

```
EE
```

```
; ED 'SI ∅ PRØG' ; MD 1-25 ; ZØNE B ; ZØNE A ; :F; LN 'SI ∅ PRØG' 9-10; :F ;
```

en fin d'interprétation, on trouvera en cartes dynamiques :

```
Rlf + Rlf + RlfCH1 = RlfCH2 = Rlf  + Rlf  ∅∅∅∅ 9 ∅ ligne 9 Rlf ∅∅∅ 10 ∅ ligne 10 Rlf +
```

les lignes 9 et 10 pouvant être exploitées. (ici : Rlf = '6D (Return Line Feed))

Exemple 2 : on n'utilise que CD1

```
    colonnes
    0     8
≥   |     |  ⎧
    |     |  ⎪
    |     |  ⎪ Source exploité sous ASSYS
    |     |  ⎪
≥   |     |  ⎩
≥   | ; MF @  'ZZ' ; ED 'MØDIFS' ; IN0 ; DF 'SI P1' ; CØ 'SI ∅ PGM''SI ∅ P1';ZZ;ZZ ;
≥   |     |   EE
≥   | ; FI @  'SI ∅ P1' ;
    |     |  ⎩
```

Nota : ASSYS remplace le @ de colonne 8 par un blanc; Edits recevra donc :
MF ∅ ∅ ∅ ∅ ∅ ∅ 'ZZ', ce qui lui convient parfaitement.

5 - 8 Directive END : fin d'assemblage

END

A la rencontre de cette directive, l'assembleur édite la liste des symboles non-définis, ferme le fichier binaire et libère l'ensemble des ressources qu'il utilisait.

Sur la liste d'assemblage on voit apparaître le nombre des messages d'erreurs, ainsi que le nombre de pages virtuelles (de la table des symboles) disponibles, et le nombre de pages virtuelles réellement utilisées.

Exemple :

SOURCE

```
2805        RSR
2806        END
```

LISTE D'ASSEMBLAGE

```
2805   05B8 Ø   1 E02    RSR
       05B8             END
2805   0003     40 0A    END
```

nombre de pages virtuelles utilisées

nombre de pages virtuelles disponibles

nombre d'erreurs d'assemblage

5 - 9 Directive EØT : appel de fichier/procédure.

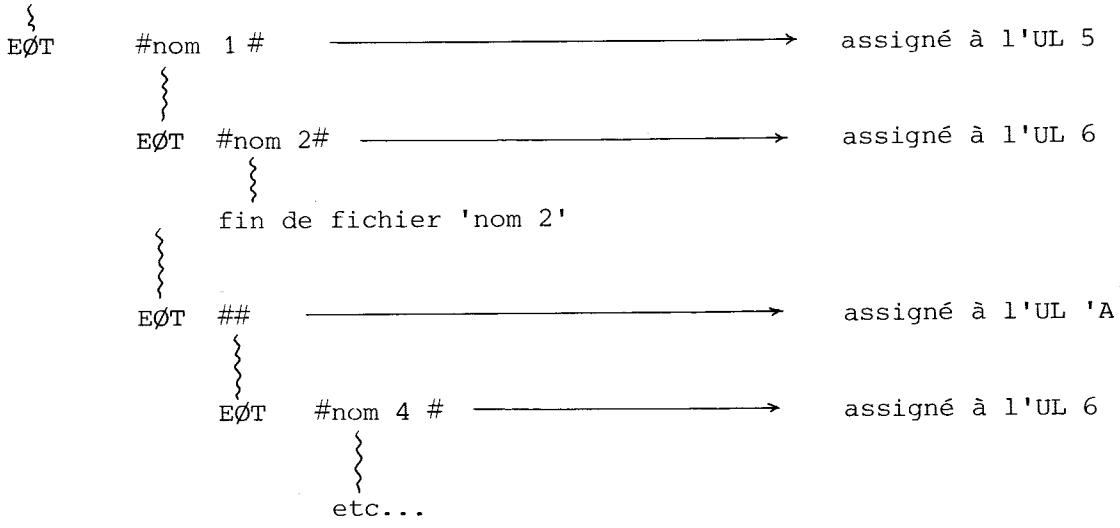
EØT

appel explicite	EØT	# nom fichier #	
appel implicite	[étiquette] # @ Code op	arguments	< ⌀ #nom fichier#
utilisation particulière (nom vide)	EØT	##	

A la rencontre de cette diective (ou de la directive CALL), l'assembleur assigne le fichier spécifié à l'UL 5 à 9, en fonction du niveau d'appel, les EØT pouvant s'imbriquer.

Appel avec nom de fichier vide : alors, l'assembleur lit la suite du source sur le lecteur de cartes (assigné à l'UL 'A) : dans ce cas on reste au même niveau d'imbrication

Exemple :



Ceci vaut également pour les appels implicites (voir l'utilisation des caractères spéciaux : '@', '#' et '%'. On remarquera que le caractère de substitution % reçoit la valeur 4 ou la valeur de l'UL associée au fichier dans lequel il est utilisé (utilisation de symboles "locaux" dans une procédure).

5 - 10 Directive EST : Edition table des symboles

EST

A la rencontre de cette directive, l'assembleur édite la table des symboles par ordre alphabétique (sur la première lettre seulement) selon le format suivant (dont des exemples figurent en annexes) :

- 8 symboles par ligne.
- pour chaque symbole figurent les informations suivantes :

AAAA ∅ X ∅ SSSSSS

AAAA

- valeur (en hexa) du symbole (s'il est défini)
- EXT ∅ si c'est un symbole externe.
- LIST si c'est une "liste" (voir ci-après)
- AVAN si le symbole est non défini et qu'il n'est pas une "liste" (AVAN signifie référence en avant le symbole n'ayant pas été défini).

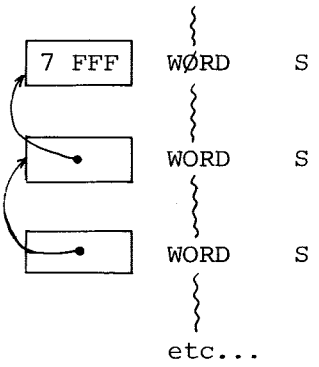
X (chiffre hexa)

- bit 0 à 1 signifie : défini en DSEC
- bit 1 à 1 signifie : symbole absolu
- bit 2 à 1 signifie : symbole "extern"
- bit 2 à 1 signifie : symbole "entry"

SSSSSS — nom du symbole

Qu'entend-on par symbole "liste" ?

Dans un programme, l'utilisation d'un symbole non encore défini provoque l'action suivante de l'assembleur :



(les chaînages sont destinés à l'éditeur de liens)

Dans cet exemple, trois possibilités existent :

1) S n'est jamais défini : alors, la directive END provoquera une erreur 17 : symbole S non défini.

2) S est défini plus "bas" dans le programme : l'assembleur génère les informations appropriées qui permettront à LINK de résoudre les adresses et donc de remplacer les chaînages ci-dessus.

3) S est défini par l'utilisateur comme étant une "liste" (on verra comment ci-après) : les chaînages créés à l'assemblage resteront inchangés par LINK : ainsi l'utilisateur aura-t-il pu générer facilement des chaînages, en vue de décrire par exemple une arborescence.

Comment définir le symbole comme "liste" ?

Il suffit, avant de faire END, de récupérer et de modifier l'"état du symbole" S, en utilisant les fonctions 'F700 et 'F701 (définies en partie 6 de ASSYS : "expression, opérateurs et fonctions").

Reprenons l'exemple précédent, et les 3 possibilités évoquées en nous intéressant aux bits 8 et 12 de l'état du symbole S :

1) on aura : bit 8 = 1 : S non défini
bit 12 = 1 : S déjà chaîné

ce qui provoquera une erreur 17 en fin d'assemblage;

2) on aura : bit 8 = 0 : S est défini
bit 12 = 1 : S déjà chaîné

donc pas d'erreur d'assemblage

3) on aura : bit 8 = 1 : S est non défini
bit 12 = 0 : S est non chaîné.

Ceci à la suite de la modification de l'état du symbole par l'utilisateur (en utilisant 'F700 et 'F701), ce qui entraînera que le symbole ne provoquera pas d'erreur d'assemblage qu'il figurera dans la table des symboles comme LIST (voir EST) et que LINK laissera inchangés les chaînages générés par ASSYS.

Pour mettre à 0 le bit 12 de l'état du symbole, il suffit d'écrire par exemple ceci :

```
SYMB        @
XWØRK :     VAL     '0080 + '0003 = 'F500 + '0080 = 'F600 (0 + '0004 = 'F701
```

Commentaire :

'0080 + '0003 = 'F500 entraîne SØM : = '0083 puis '0000
MASK : = '0083,

ceci pour isoler les bits 8 (symbole défini/non défini) et 14-15 (longueur) dans l'"Etat Symbole".

'0080 = 'F600 entraîne SØM : = '0080
puis VAL : = '0080,

on a ainsi isolé le bit 8 (défini/non défini) qui devra passer à 1 dans Etat Symbole.

(0 + '0004 = 'F700 entraîne SØM : = 0
puis fonction 'F7 sur le symbole placé en CD1 en colonne 0 sur une longueur de 4 ; l'état du symbole ES sera ainsi modifié : ES : = (ES.AND.MASK).ØR.VAL or MASK = '0083 et VAL = '0080 donc :

- bits 0 - 7 et 9 - 13 de ES sont mis à 0
- bit 8 de ES est mis à 1 : symbole non-défini
- bits 14 - 15 de ES inchangés (longueur symbole).

On a bien ainsi un symbole non-défini et non-chaîné qui devient un symbole "liste".

5 - 11 Directive IDP : Identification de Programme

IDP

```
IDP  "texte"
```

Placée n'importe où dans un source, elle entraîne la génération du texte spécifié dans le fichier binaire. Lors du LINK, ce texte sera édité, ce qui permettra d'identifier sans erreur le binaire, comme le montre l'exemple ci-dessous.

Exemple :

```
        {
        IDP     "EDITEUR DE FICHIERS"
        IDP     "AUTEUR : J. DURAND"
        IF      ØRDI-"T", XWØRK, ,XWØRK
        IDP     "VERSION EXECUTABLE SØUS CMS4-T1600"
XWØRK:  VAL     0
        IF      ØRDI-"S", XWØRK, ,XWØRK
        IDP     "VERSION EXECUTABLE SØUS CMS5 - SØLAR"
XWØRK:  VAL     0
        IDP     "RELEASE NUMERØ 9 ; 25/11/78"
        }
```

Nota : Le texte est exploité lors du link, mais n'affecte pas la taille du programme exécutable généré.

5 - 12 Directive IF : assemblage conditionnel

IF

IF < expression >,[étiquette 1] ,[étiquette 2] ,[étiquette 3]

A la rencontre du IF, l'assembleur calcule l'expression et reprend l'assemblage à

. étiquette 1 si résultat < 0
. " 2 " " = 0
. " 3 " " > 0

Si étiquette i est omise, ASSYS continue en séquence.

Exemple typique d'utilisation :

```
        }
        }
        }
ØRDI:   VAL     "S"             < type d'ordinateur = solar
        }
        IF      ØRDI - "S", XWØRK,,XWØRK
        IDP     "VERSION SØLAR"
XWØRK : VAL     0
        }
        }
        }
```

Nota : Les IF peuvent être imbriqués. On prendra les précautions nécessaires, c'est-à-dire qu'on fera :

```
        IF < expression >, X1,,X1           IF < expression > , X1,,X1
        }                                   }
        IF < expression >, X2,,X2           IF < expression > , X2,, X2
        }                       et non pas  }
X2:     VAL   0                             X1: VAL 0
        }                                   }
X1:     VAL   0                             X2: VAL 0
```

LST/NLS

5 - 13 Directives LST/NLS : autorisation/inhibition de la liste

On rappelle que la liste d'assemblage est envoyée :

- sur l'unité logique 2 (organe de sortie) si l'unité logique 4 n'est pas assignée.
- sur l'unité logique 4 sinon ; celle-ci pouvant être assignée soit à l'imprimante, soit à l'organe de sortie.

Les erreurs d'assemblage sont listées, non pas sur l'UL 4 mais toujours sur l'UL 2.

De plus, on doit savoir

1°) qu'en début d'assemblage la liste est active,

2°) qu'on peut l'inhiber en utilisant la directive NLS, celle-ci étant empilable (on peut faire n-fois NLS).

3°) qu'il vaut mieux inhiber la liste par NLS plutôt que par une assignation vide de l'unité 4 (meilleures performances).

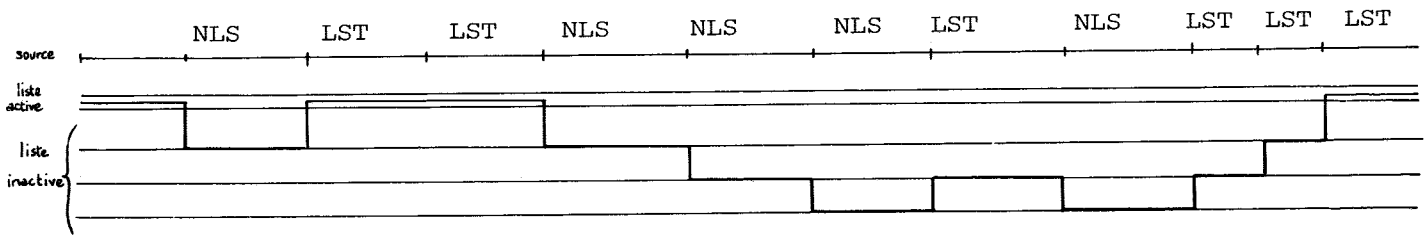
4°) qu'on peut réactiver la liste par la directive LST, celle-ci pouvant aussi se répéter n-fois, à ceci près qu'une directive LST rencontrée alors que la liste est déjà active sera ignorée (voir ci-dessous exemple 1).

5°) que le niveau d'imbrication des NLS est accessible par @ en colonne 33. (Voir exemple 2).

6°) que la directive CCI permet, en cours d'assemblage, de modifier la destination de la liste d'assemblage.

7°) que la directive NLS n'inhibe pas l'édition des messages d'erreur sur l'UL 2.

Exemple :



Exemple 2 :

Le @ placé en colonne 33 permet de récupérer :

. 0 si Liste active

. 1,..i..,n si liste inactive, i étant le niveau d'imbrication des NLS.

```
        LST
XLIST:  VAL   '0 0000 00 0000 @   ⟶ XLIST recevra 0   } liste active
        NLS
XLIST:  VAL   '0 0000 00 0000 @   ⟶ XLIST recevra 1   } liste
        NLS                                           } inactive
XLIST:  VAL   '0 0000 00 0000 @   ⟶ XLIST recevra 2   }
                                    etc...
```

5 - 14 Directive MØT : équivalente à VAL

Pour définir des constantes d'assemblage, on pourra écrire indifféremment :

Symbole: VAL expression

Symbole: MØT expression

Ceci permet d'améliorer la lisibilité des programmes en distinguant les différentes constantes. Voir à ce sujet en annexe l'exemple "Sous-programme SPTST".

De plus, dans le cas où l'on ne peut plus ouvrir de DSEC parce que l'on a déjà 32 sections dans le programme, on peut s'en tirer en utilisant MØT pour décrire une zone de données.

5 - 15 Directive NDS : symboles non-définis.

NDS

A la rencontre de cette directive, ASSYS édite la liste des symboles non-définis et n'étant pas non plus des symboles "liste" (voir dans les développements de la directive EST la signification de ce terme).

Il faut bien entendu, que la liste soit active pour que ces symboles apparaissent (cf : LST/NLS).

Noter que la directive END fait un NDS implicite enfin d'assemblage puisqu'elle provoque l'édition des symboles non définis (erreur 17).

5 - 16 Directive PAGE : saut de page

PAGE

Provoque un saut de page sur l'imprimante si la liste est demandée (cf : LST/NLS).

De plus, l'assembleur liste le numéro de page en héxadécimal dans la zone adresse, qu'on peut donc récupérer par des @.

Voir en annexe l'exemple de la procédure SI PAGE qui récupère, convertit en décimal, et édite les numéros de page.

6) EXPRESSIONS, OPERATEURS ET FONCTIONS

6 - 1 Opérateurs : généralités

OPERATEURS
Généralités

Pour écrire des expressions (non parenthèsées), on dispose des opérateurs suivants :

+ addition

- soustraction

* multiplication

/ division

? OU Logique

(ET Logique

) OU Exclusif

> DECALAGE logique gauche et droit (cf : ci-après)

= { Nombre de décalage / Fonction 'Fi } cf : ci-après

Nota : on ne dispose pas du parenthèsage ; en revanche, on peut empiler les résultats de calcul : voir les fonctions Fi.

Les expressions sont évaluées de gauche à droite.

6 - 2 Expressions : généralités

Dans une expression, l'opérateur '=' associé aux fonctions de types 'Fiαβ explicités ci-après ouvre de larges possibilités illustrées d'une part par des exemples et d'autre part en annexe dans des exemples de procédures. Avant d'expliciter cet opérateur '=' et ces fonction 'Fiαβ il convient de dire

1°) comment est analysée une expression

2°) quelles sont les variables accompagnant le calcul d'expression.

1°) Analyse d'une expression.

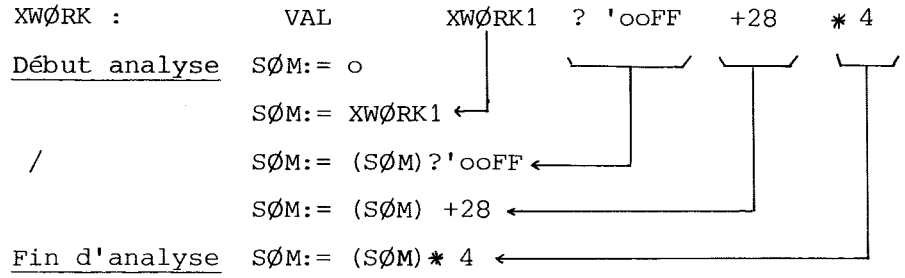
Une expression est analysée de gauche à droite. Il n'y a pas de parenthèsage (mais on dispose d'une pile utilisateur de 88 mots).

On conviendra de nommer 'SØM' ("Somme"!?) la valeur courante d'une expression en cours d'analyse. Par extension 'SØM' désignera la partie déjà analysée d'une expression.

Exemple :

```
XWØRK :          VAL       XWØRK1  ? 'ooFF   +28  *  4
```

L'analyse se déroulera de la manière suivante et en utilisant une notation conventionnelle :



2°) Variables accompagnant l'analyse

Ce sont :

SØM : valeur courante de l'expression ; initialement : 0

MASK : masque courant ; initialement inactif ; il est créé, modifié et exploité par les fonction 'F5, 'F6 et ' FB.

VAL : valeur cumulée sous MASK ; initialement 0; modifiée et utilisée par des fonctions 'F6 et 'F701

REST : dernier reste de division. Accessible en utilisant la fonction 'FD. Initialement 0.

PILE : pile de 88 mots à la disposition de l'utilisateur.
A noter qu'elle est entièrement gérée par l'utilisateur (fonctions 'FE00 'FE01 'FE02) et qu'elle n'est initialisée par l'assembleur qu'en début d'assemblage.
Sa durée de vie n'est donc pas limitée à l'analyse d'une expression mais à l'ensemble de l'assemblage, comme les exemples fournis le montrent.

6 - 3 Opérateur > : décalage gauche/droit

>

SØM > v

SØM: = (SØM). Shift Logigal. | v |

↓

Shift Logical Left si v ⩾ 0

Shift Logical Right si v < 0

Exemple :

```
XWØRK1 :   VAL   4                        < pour décalage gauche
XWØRK2 :   VAL   '00F0
XWØRK2 :   VAL   XWØRK2 > XWØRK1          < décalage gauche ;
                                          < XWØRK2 recevra '0F00

XWØRK1 :   VAL   -5                       < pour décalage à droite
XWØRK2 :   VAL   '00F0
XWØRK3:    VAL   XWØRK2 > XWØRK1          < décalage à droite ;
                                          < XWØRK3 recevra '0007
```

Nota : Pour faire des décalages autres que ceux autoriés par '>' (c'est-à-dire autres que SLLS, SLRS), utiliser la fonction 'FB permettant tout shifts (SARS SCRS etc...)

6 - 4 Opérateur = Nombre de décalage

=

SØM = valeur

Deux cas se présentent :

1 - si le 1er chiffre hexa de < valeur > est ≠ 'F alors = donne dans SØM le nombre de décalages à droite à faire sur SØM pour qu'il n'y ait plus de bits à zéro à droite ; puis il fait SØM: = (SØM) + < valeur >

Exemple :

```
XWØRK:      VAL    '00E0
XDEC:       VAL    XWØRK = 0
```

} donnera: XDEC = 5

```
XWØRK:      VAL    'OOEO
XDEC:       VAL    XWØRK = 2
```

} donnera: XDEC = 5 + 2 = 7

2 - si le 1er chiffre hexa de < valeur > est 'F alors, le premier octet de < valeur > est pris comme une fonction Fi (i ∈ [0,F]). Ces fonctions sont traitées une à une ci-après.

6 - 5 Fonction F1 : Get octet en carte dynamique

= 'F1xx

SØM = 'F1αβ

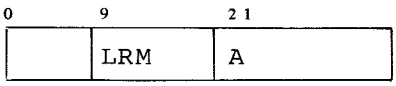
SØM: = CARACTERE (SØM) + αβ

c'est-à-dire que SØM reçoit le caractère de rang (SØM) en carte dynamique auquel est ajouté le second octet de < valeur >.

Exemple :

@ LRM A

CD1



XCAR: VAL 21='F100 → XCAR reçoit "A"

XCAR: VAL 10='F101 → XCAR reçoit "S" ("R"+'01)

Rappel :

Les caractères sont numérotés 0 à 239 en cartes dynamiques 1,2, et 3.

6 - 6 Fonction 'F2 : store octet en carte dynamique

= 'F2xx

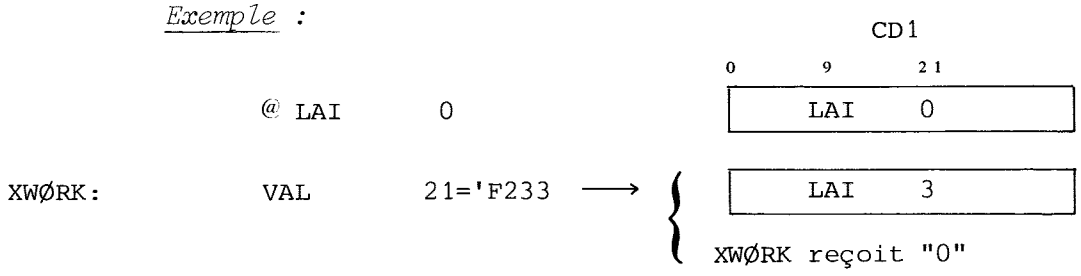
SØM = 'F2αβ

CARACTERE (SØM) : = αβ et

SØM : = CARACTERE (SØM)

c'est-à-dire que l'octet de rang (SØM) en carte dynamique reçoit αβ et que SØM reçoit l'ancien caractère de rang (SØM).

Exemple :



6 - 7 Fonction 'F3 : move chaîne d'octets

= 'F3xx

SØM = 'F3αβ

- 'αβ = longueur octets
- 1er octet de SØM = index zone émettrice
- 2e octet de SØM = index zone réceptrice
- le move est effectué caractère par caractère, de gauche à droite, puis SØM reçoit 0.

Exemple :

exemple 1: si CD1 contient

0	1	2	3	4	5
A	B	C	D	E	F

XWØRK: VAL '0003='F302 donnera :

0	1	2	3	4	5
A	B	C	A	B	F

SØM = 0

exemple 2: pour mettre CD2 en CD3, il suffit d'écrire :

XWØRK: VAL '50 > 8 + 'A0 = 'F350

zone émettrice : colonnes '50 à '9F (80 à 159)

zone réceptrice : colonnes 'A0 à 'EF (160 à 239)

longueur = '50 = 80

exemple 3: pour étendre le caractère "A" dans CD2, il suffit d'écrire :

XWØRK: VAL '50 = 'F241 (0 + '5051 = 'F34F

store octet '41 (="A") en colonne '50

SØM:= 0

z. émettrice: CD2(0)

z. réceptrice: CD2(1)

longueur: '50 - 1

(CD2(0)) est étendu.

='F4xx

6 - 8 Fonction 'F4 : recherche d'un octet

SØM = 'F4αβ

'αβ = octet cherché

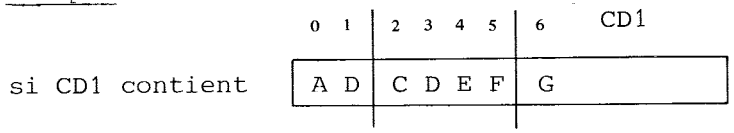
1er octet de SØM = index du 1er octet à tester en cartes dynamiques

2e octet de SØM = " " " " " ne pas tester " "

la recherche s'effectue puis SØM reçoit :

- soit l'index obtenu, si le caractère a été trouvé
- soit l'index de fin de recherche spécifié (2e octet de SØM) si la recheche a été infructueuse ou si les 2 index étaient incompatibles.

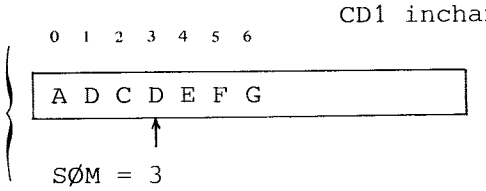
Exemple :



XWØRK: VAL '0205 = 'F444 donnera :

caractère "D"

CD1 inchangée



= 'F5xx

6 - 9 Fonction 'F5 : définition de MASK : masque courant

SØM = 'F5αβ

MASK reçoit (SØM)

puis SØM reçoit 'αβ

Exemple :

XWØRK: VAL 'C000 = 'F502 donnera :

MASK: = 'COOO

SØM: = 2

= 'F6xx

6 - 10 Fonction 'F6 : cadrage suivant le masque courant

SØM = 'F6αβ

La valeur contenue dans SØM est cadrée suivant le masque courant pour donner v puis

SØM: = v ? VAL + 'αβ puis

VAL: = SØM

VAL, initialisée à 0, étant le cumul de valeurs cadrées suivant le masque courant : MASK.

Exemples :

X: VAL 'OFFO = 'F501 + 3 = 'F600 donnera :

X = '0040 c'est-à-dire : 1 + 3 cadré suivant 'OFFO'

X: VAL 'OFOO = 'F5OO + 3 = 'F6OO + 'OOFO = 'F5OO + 7 = 'F6OO donnera :

X = '037O c'est-à-dire : 3 cadré suivant 'OFOO

+ 7 cadré suivant 'OOFO

X: VAL 'OFOO = 'F5OO + ' 1234 = 'F6OO donnera une erreur d'assemblage (02), car ' 1234 ne peut pas être cadré sous 'FOO.

Nota : Le cadrage se fait en recherchant dans MASK le premier groupe de bits à 1 en partant de la droite (c'est-à-dire bit 15 de MASK). Si le bit 15 de MASK est à 1, il n'y a donc pas de cadrage mais pas non plus de validation.

6 - 11 Fonction 'F7 : accès à la table des symboles

= 'F7xx

SØM = 'F700	get état d'un symbole
SØM = 'F701	modify état d'un symbole

avec F7 on a toujours :

1er octet de SØM = index du 1er caractère du symbole placé en carte dynamique.

2e octet de SØM : nombre de caractères du symbole.

1- 'F700: get état d'un symbole :

exemple:

0		9	21	0	CD1
TØTØ	@		⟶	TØTØ	
XWØRK :		VAL	'OOO4 = 'F7OO donnera l'état de TØTØ		

On reçoit dans XWØRK la valeur : 'ES' : état symbole :

ES = o: le symbole 'TØTO' n'existe pas

ES ≠ o: le symbole existe ; ES a la signification suivante :

bit 0 (ES)=1 une indexation a été demandée sur ce symbole alors qu'il n'est pas encore défini.

bits 1-7 (ES) : numéro de section si le symbole existe et est défini. numéro nul dans les autres cas.

bit 8 (ES)=1 : le symbole a été référencé mais n'est pas encore défini ; il sera donc translatable.

bit 9 (ES)=1 : le symbole a été défini de manière absolue (VAL).

bit 10 (ES)=1 : le symbole a été référencé par une directive REF.

bit 11 (ES)=1 : le symbole est translatable et déjà défini dans une DSEC.

bit 12 (ES)=1 : le symbole est déjà chaîné.

bit 13 (ES)=1 : le symbole est référencé dans une directive DEF.

bits 14-15(ES)= partie entière de (longueur du symbole + 1)/2, donc : 1 à 3 puisqu'un symbole fait 1 à 6 caractères.

2 - F701 : modification de l'état d'un symbole.

'ES' étant l'état courant du symbole, 'F7' fera :

'ES' : = ('ES'. AND. MASK). ØR. VAL

avec MASK = masque courant (défini par 'F5)

VAL = valeur définie par 'F6 (cumul de valeurs sous MASK)

puis | = 'F7xx |

SØM : = 'ES$^{\perp}$: nouvel état du symbole.

Noter que pour cette modification de l'état d'un symbole ait un sens, il faut que sa longueur soit non nulle, donc que 'ES' (bits 14-15) $\neq 0$, et donc que les bits 14-15 de MASK et VAL soient à '01' ou '11'.

='F8xx
='F9xx

6 - 12 Fonction 'F8 : accès au compte d'itération 'INDØ'

SØM = 'F8αβ

SØM: = (SØM) + 'αβ + 'INDØ'

'INDØ' étant le compte d'itération du DØ, ce compte variant de N-1 à O, pour un DØ N sur une ligne de source autre que EØT ou CALL.

6 - 13 Fonction F9 : accès au compte d'itération 'SINDØ'

SØM = 'F9αβ

SØM: = (SØM) + 'αβ + 'SINDØ'

'SINDØ' étant le compte d'itération du DØ sur EØT (ou CALL), ce compte variant de N-1 à O et référençable dans la procédure appelée.

Exemple 1 : accès à 'INDØ' :

```
        DØ    4
XWØRK:  VAL   0 = 'F800      XWØRK recevra successivement 3, 2, 1, O.
XWØRK:  VAL   0
        DØ    10                          } on obtiendra
XWØRK:  VAL   0 = 'F800 + XWØRK           } XWØRK = Σ (i=0 à 9) i = 9x10/2 = 45
```

Exemple 2 : accès à 'SINDØ':

```
DØ      NBPØST
CALL # SI GENTABLE #
```

Dans la procédure #SI GENTABLE# l'accès à l'indice 'SINDØ' se fera en utilisant 'F9; par exemple:

```
XWØRK: VAL  1='F900)'FFFF-1-NBPØST
```

= 'FAxx

6 - 14 Fonction 'FA : conversions Asci-binaire

SØM = 'FA00 conversion binaire → Asci hexadécimal
SØM = 'FA01 conversion asci hexadécimal → binaire

. 'FAOO: SØM est converti sous forme de 2 caractères Asci hedadécimaux si cela est possible.

Exemple :

```
XWØRK:   VAL   'OO4A = 'FAOO  donnera   '3441
                                          ↓  ↓
                                         "4""A"
```

. 'FA01: SØM est supposé contenir un caractère Asci hexadécimal qui est alors converti en binaire.

Exemple :

```
XWØRK:   VAL   "A" = 'FA01   donnera   '000A
```

Exemple d'application :

```
NVPF:          VAL    '9
NVPG:          VAL    'A
ASSF:          ASCI   "!ASSIGN ⌀"
               BYTE   NVPF='FA00;"="
               BYTE   "S" ; '04
ASSG:          ASCI   "!ASSIGN ⌀"
               BYTE   NVPG='FA00;"="
               ASCI   "Ø,"
               etc...
```

} génération automatique des NVP en asci, sans risques d'erreur.

= 'FBxx

6 - 15 Fonction 'FB : Exécution d'une instruction machine

SØM = 'FBαβ

. registre A reçoit SØM

. registre Y reçoit 'αβ

. MASK courant est supposé contenir une instruction donnant son résultat dans le registre 'A'.

. puis SØM reçoit 'A'.

Applications possibles :

LAI	α	
ADRI	α,A	
ANDI	α	
EØRI	α	
NGR	A,A	
ØRI	α	
RBT	α	(α ⩽ 15)
SARS	α	
SBT	α	(α ⩽ 15)
SCLS	α	
SCRS	α	
SLLS	α	
SLRS	α	
SWBR	A,A	

1 E α 5

etc...

ASSYS 60

='FCxx

6 - 16 Fonction 'FC : exécution d'une séquence en langage machine

SØM = 'FCαβ

Registre A reçoit SØM (argument)

L'assembleur exécute un BSR à 'adresse carte dynamique 1' dans lequel l'utilisateur est supposé avoir stocké un programme en binaire, et dont le point d'entrée supposé est à l'adresse 'adresse carte dynamique 1' + αβ, 'αβ étant considéré comme un déplacement mots.

Voir exemple en annexe.

'FDxx

6 - 17 Fonction 'FD : get REST (dernier reste de division)

SØM = 'FDαβ

SØM reçoit (SØM) + αβ + REST (dernier reste de division), REST étant réinitialisé à 0 au début de l'analyse de chaque expression.

Exemple :

Voir les exemples donnés pour les fonctions 'FEO1 et 'FEO2.

6 - 18

Fonctions 'FE00 : push
'FE01 : pull et addition
'FE02 : pull et '='

'FEOO
'FEO1
'FEO2

L'assembleur met à la disposition de l'utilisateur une pile de 88 mots !

. 'FE00 provoque l'empilement de la valeur courante de SØM

. 'FEO1 provoque le dépilement de la dernière valeur empilée
et fait SØM:= (SØM) + < valeur dépilée >.

. 'FEO2 provoque le dépilement de la dernière valeur empilée
et fait SØM:= (SØM) = < valeur dépilée >.

Cette valeur pouvant être de la forme 'FIαβ ! auquel cas elle est interprétée comme telle.

Exemple 1

On veut récupérer une constante NØLIGN et, la convertir en décimal de manière à l'utiliser comme numéro de ligne pour faire une insertion sous EDITS, l'éditeur de fichier.

Ayant défini les paramètres suivants :

```
BASE 10:    VAL     10
BASE 16:    VAL     16
FREST:      VAL     'FD00       < get dernier reste
FPUSH:      VAL     'FE00       < push
FPULLA:     VAL     'FE01       < pull and add
```

il suffira d'écrire :

```
NØLIGN : VAL    NØLIGN + 1          < ligne suivante
XWØRK  : VAL    NØLIGN
         DØ     4                   < car on a 4 chiffres décimaux à obtenir
XWØRK  : VAL    XWØRK/BASE 10 (0 = FREST = FPUSH (0 + XWØRK/BASE 10
                                      < on a empilé les restes successifs !
```

FExx

```
          IF    XWØRK,,XWØRK1,
          ERREUR! (numéro de ligne trop grand pour tenir sur 4 chiffres) !
XWØRK1:   VAL     O                          < il faut maintenant dépiler
                                             < les valeurs empilées en les multi-
                                               pliant
                                             < par 16
          DØ      4
XWØRK:    VAL     XWØRK * BASE 16 = FPULLA    < facile, non ?
                                             < générons en Carte Dynamique
                                             < des arguments pour EDITS
;ED       @ 'FIC';  IN @@@@;.....; etc...
                                             < et maintenant, appelons EDITS.
          EE
```

Exemple 2 : utilisation de 'FEO2

On veut générer ABCD...XYZ en CD1. Il suffit d'écrire :

```
XWØRK:    VAL          "Z"
FS:       VAL          'F200              store octet
FM:       VAL          'F500              définition de MASK
FV:       VAL          'F600              définition de la fonction de cadrage
FP:       VAL          'FEOO              PUSH
FPX:      VAL          'FEO2              PULL et exécute
FINDØ:    VAL          'F500              accès au compte d'itération 'INDØ'
          DØ            26                nombre de lettres de l'alphabet
XWØRK:    VAL          'OOFF=FM+XWØRK=FV?FS=FP (O=FINDØ=FPX(O+XWØRK-1
```

Ce qui donnera en CD1: ABCD....Z

6 - 19 Fonction 'FFαβ : test du signe de SØM

SØM = 'FFαβ

SØM reçoit : αβ + { 0 si SØM = 0 ; -1 si SØM < 0 ; +1 si SØM > 0

Exemple :

```
XWØRK:   VAL      5='FFOO          XWØRK recevra 1
XWØRK:   VAL    5-5='FFOO          XWØRK recevra 0
XWØRK:   VAL   5-10='FFOO          XWØRK recevra -1
XWØRK:   VAL   5-10='FFO3          XWØRK recevra 2 (-1+αβ=
                                                    -1+3 =2)
```

ADDITIFS ASSYS 1/8/79

- Code opération QUIT : on peut écrire QUIT n, ceci équivaut à WØRD '1En6. Avec 0 ⩽ n ⩽ 'F

Exemple: QUIT 1 (équivalent à WØRD '1E16).

- Code opération ACTD : on peut fournir un opérande n avec 0 ⩽ n ⩽ 'F

Exemple : ACTD 1 (équivalent à WØRD '1E15)

- Symboles définis en VAL : pour un symbole défini en VAL, on peut doubler les "deux-points" délimitant son nom ; ceci aura pour effet de provoquer une erreur d'assemblage (ERA 08) si ce symbole existe déjà ; pas d'erreur sinon. Ainsi, on peut s'assurer, quand on ajoute un nouveau symbole dans un programme, qu'il n'est pas déjà utilisé.

Exemple : ABC:: VAL x

- Symboles définis en EQU : là aussi, on peut doubler les "deux-Points" déliminant le nom du symbole, et ceci aura l'effet suivant : si le symbole n'existe pas déjà, il est créé comme d'habitude ; si le symbole existe déjà, alors il est détruit, puis recréé avec sa nouvelle valeur. Ceci permet d'éviter les erreurs de double-définition, mais doit être utilisé avec précaution.

Exemple : ETIQ:: EQU ⊄

- Fonction 'FB00 : cette fonction est étendue de telle sorte qu'elle permette d'exécuter une instruction donnant son résultat non plus dans le registre A seulement, mais dans l'un des registres A, B ou X.

SØM = 'FBαβ donne maintenant ceci :

. registre A reçoit SØM
. registre Y reçoit 'αβ
. registre B reçoit 0
. registre X reçoit 0
. MASK courant est supposé contenir une instruction donnant son résultat dans l'un des registres A, B ou X. Cette instruction est exécutée, puis :
. A reçoit (A). EØR.(B)
. A reçoit (A). EØR.(X)
.SØM reçoit (A).

Exemples :

- récupération par '1E15 du contenu de l'adresse '3000 :

```
X:  VAL '1E15 = 'F500 (0 + '3000 = 'FB00) '3000
```

- récupération du numéro du premier bit à 1 dans une valeur par exemple Y (utilisation de l'instruction DBT de code '1E03) :

```
X:  VAL '1E03 = 'F500 (0 + Y = 'FB00) Y
```

7) ANNEXES.

On trouvera successivement dans les annexes de ASSYS :

- notes sur la façon de présenter un listing.
- la liste d'assemblage.
- extrait d'une table des symboles.
- les messages d'erreurs d'assemblage.
- les commandes spécifiques à ASSYS-SØLAR.
- un exemple de procédure : 'SI PAGE' et son utilisation.
- un exemple d'utilisation de la fonction 'FCXY.
- un exemple d'utilisation de la directive MØT.

NOTES SUR LA FACON DE PRESENTER UN LISTING :

PRELIMINAIRES :

* LES NOTES SUIVANTES A CONSIDERER
* COMME DES NORMES DE PROGRAMMATION
* SONT REDIGEES EN FAISANT L'HYPOTHESE
* REALISTE QUE LES PROGRAMMEURS UTILI-
* SENT LES METHODES DE LA "PROGRAMMA-
* TION STRUCTUREE" ; AINSI, ON SUPPO-
* SERA QU'AUCUN PROGRAMME NE CONTIENT
* DE BLOCS DE CODE "MONOLITHIQUES" DEPAS-
* SANT DEUX A TROIS CENTS INSTRUCTIONS.
* ENFIN, PAR ABUS DE LANGAGE, ON ASSI-
* MILERA, ET APPELERA L'UNITE DE PROGRAM-
* MATION : "SOUS-PROGRAMME".

DE L'INTERET DU LISTING :

* LE LISTING DOIT ETRE CONSIDERE COMME
* UN INSTRUMENT DE TRAVAIL ET DE DOCU-
* MENTATION UTILISE PAR SOI-MEME, MAIS
* AUSSI PAR D'AUTRES. LES UTILISATIONS
* POTENTIELLES D'UN LISTING SONT LES
* SUIVANTES (LISTE NON EXHAUSTIVE) :
* - MISE AU POINT DU PROGRAMME,
* - DOCUMENTATION DU PROGRAMME,
* - MODIFICATION DU PROGRAMME (EXTENSION
* OU SUPPRESSION DE FONCTIONS, CHANGEMENT
* DE FORMATS DE DONNEES INTERNES OU EXTERNES),
* - AIDE PEDAGOGIQUE (COURS DE PROGRAMMA-
* TION OU DE SYSTEMES).

```
<                *    IL APPARAIT ALORS CLAIREMENT QUE SI
<                *  CERTAINES NORMES DE REDACTION SONT EDICTEES
<                *  ET RESPECTEES, ALORS LES LISTING POURRONT
<                *  DEVENIR DES OBJETS TRANSMISSIBLES, ET UTILI-
<                *  SABLES PAR TOUS (QUELQUE SOIT LE DEGRE D'ASTUCE
<                *  DES UTILISATEURS...).
<
<
<        REGLE 1 :
<        -------
<                *    LES PROGRAMMES QUELQUE SOIT LEUR
<                *  MODE D'ENTREE (CARTES OU FICHIERS)
<                *  DEVRONT UTILISER LA TABULATION STANDARD
<                *  VOIR LA COMMANDE "I" DU PROCESSEUR
<                *  "CALL" DANS LE CAS DES FICHIERS).
<
<
<        REGLE 2 :
<        -------
<                *    LES COMMENTAIRES NE COUTENT PAS CHERS,
<                *  IL FAUT DONC EN METTRE EN QUANTITE SUFFI-
<                *  SANTE, ET LES METTRE AU MOMENT DE L'ENTREE
<                *  DU CODE EN MACHINE, ET NON PAS ULTERIEUREMENT...
<                *    AINSI, ON DECRIRA EN ZONE COMMENTAIRE
<                *  LE RESULTAT ATTENDU D'UNE INSTRUCTION, OU
<                *  BIEN D'UN GROUPE D'INSTRUCTIONS, EN
<                *  INSISTANT PARTICULIEREMENT SUR LES SEQUEN-
<                *  CES "ESOTERIQUES", OU BIEN SUR LES ASTUCES,
<                *  OU BIEN ENFIN SUR LES OPTIMISATIONS ; AINSI
<                *  PAR EXEMPLE :
<        MOVE
<                *  EST PARFOIS UTILISEE EN EQUIVALENCE AVEC
<                *  LA SEQUENCE SUIVANTE :
<        MOVE
<        LXI         0
<                *  LA DEUXIEME INSTRUCTION N'ETANT PAS EXPLICITEE.
<                *    LES SEQUENCES D'INSTRUCTIONS FORMANT UN
```

```
<                  *  TOUT LOGIQUE SERONT SEPAREES PAR DES LIGNES
<                  *  DE COMMENTAIRES INTRODUISANT GLOBALEMENT
<                  *  LE TRAITEMENT A SUIVRE ; PAR EXEMPLE :
<<
<< DECODAGE ET ANALYSE DE LA COMMANDE :
<<
<         (...)
<<
<< EXECUTION DE LA COMMANDE 'XX' :
<< (LA COMMANDE 'XX' PERMET...)
<<
<         (...)
<
<
<         REGLE 3 :
<         -------
<                  *    DE MEME QUE L'ON FAIT DES MISES A
<                  *  A JOUR DU CODE DU PROGRAMME, ON FERA
<                  *  DES MISES A JOUR DES COMMENTAIRES,
<                  *  DE FACON A CE QU'IL N'Y AIT JAMAIS DE
<                  *  DEPHASAGE ENTRE LA REALITE (LES INSTRUC-
<                  *  TIONS ET LES DONNEES) ET SA DESCRIPTION...
<                  *    ON VERRA A CE PROPOS LES REGLES CONCER-
<                  *  NANT LE PARAMETRAGE...
<
<
<         REGLE 4 :
<         -------
<                  *    CHAQUE SOUS-PROGRAMME (CF. LES PRELI-
<                  *  MINAIRES) COMMENCERA EN HAUT D'UNE NOU-
<                  *  VELLE PAGE EN UTILISANT EXPLICITEMENT
<                  *  LA DIRECTIVE "PAGE". SUIVRA LE NOM ET/OU
<                  *  LA FONCTION (DECRITE EN DEUX OU TROIS
<                  *  MOTS) DE CE SOUS-PROGRAMME ECRIT DE FACON
<                  *  QUE CE TEXTE SE DISTINGUE DU RESTE D'UN
<                  *  SIMPLE COUP D'OEIL ; AINSI PAR EXEMPLE :
<         PAGE
```

```
<<
<<
<<      H E X I N E X  :  C O N V E R S I O N
<<      B I N A I R E   -->   H E X A - D E C I M A L E   A S C I  :
<<
<<
<                *    ENSUITE ON INDIQUERA CLAIREMENT LA
<                *  FONCTION REALISEE PAR CE SOUS-PROGRAM-
<                *  ME TANT AU POINT DE VUE EXTERNE QU'AU
<                *  POINT DE VUE INTERNE ; AINSI PAR EXEMPLE
<                *  SI UN SOUS-PROGRAMME REALISE UNE RECON-
<                *  NAISSANCE DE COMMANDES (FONCTION EXTERNE),
<                *  TOUT EN OPTIMISANT SES MOYENS DE RECON-
<                *  NAISSANCE PAR UN CERTAIN APPRENTISSAGE
<                *  (FONCTION INTERNE), IL FAUDRA LE PRE-
<                *  CISER NETTEMENT.
<<
<<
<<      FONCTION :
<<                 LE SOUS-PROGRAMME 'XXX', PROCEDE
<<               A UNE IDENTIFICATION DE LA COMMANDE
<<               CONTENUE DANS LE BUFFER 'BUF' DONT
<<               L'INDEX COURANT EST 'IND'. IL UTILISE
<<               LA TABLE 'TAB' (DONT NOUS RAPPELONS
<<               ICI LE FORMAT...) DONT IL MODIFIE A
<<               CHAQUE APPEL LE CONTENU, AFIN D'OPTIMISER
<<               LES RECONNAISSANCES SUIVANTES ; L'ALGORI-
<<               THME UTILISE EST LE SUIVANT : (...).
<<
>>
<                *    SI LE SOUS-PROGRAMME A POUR BUT
<                *  L'EVALUATION D'UNE OU PLUSIEURS FORMULES
<                *  CELLES-CI SERONT UTILEMENT FOURNIES EN
<                *  UTILISANT UNE NOTATION "NATURELLE" ; ENFIN
<                *  IL SERA INTERESSANT DE LA (OU LES) RAPPELER
<                *  AU MOMENT DE LEUR EVALUATION.
<                *    A LA SUITE DE LA FONCTION, ON DONNERA
```

```
<                *  AVEC LA PLUS GRANDE PRECISION L'INTERFACE
<                *  QUE POSSEDE CE SOUS-PROGRAMME AVEC SES
<                *  UTILISATEURS EVENTUELS ; ON SEPARERA NET-
<                *  IEMENT LES ARGUMENTS DES RESULTATS :
<<
<<
<<       ARGUMENTS :
<<                   REGISTRE A :
<<                                BIT 0=....
<<                                BITS 1 A 7=...
<<                                BITS 8 A 15=...
<<                   BUFFER 'BUF'=...
<<                   INDEX 'IND'=INDEX COURANT DU BUFFER 'BUF'.
<<
<<
<<       RESULTATS :
<<                   REGISTRE A : INCHANGE.
<<                   REGISTRE X=0 SI TOUT S'EST BIEN PASSE,
<<                             =CODE ERREUR SINON ; LES CODES
<<                              EXISTANT SONT LES SUIVANTS (...).
<<
<<
<                *    ENFIN, SI CELA EST NECESSAIRE ON RAJOUTERA
<                *  SOUS FORME DE NOTES, OU DE REMARQUES TOUTE
<                *  INFORMATION SUSCEPTIBLE D'AMELIORER LA
<                *  COMPREHENSION, LA MAINTENANCE OU LA MODI-
<                *  FICATION DU SOUS-PROGRAMME ; EN PARTICULIER
<                *  LORSQUE L'AMPLEUR DU SOUS-PROGRAMME LE
<                *  JUSTIFIE, ON POURRA, SANS ALLER JUSQU'AU NIVEAU
<                *  D'UN ORGANIGRAMME, DONNER EN QUELQUES PHRASES
<                *  BIEN "TROUSSEES" LA STRUCTURE DE CELUI-CI.
<
<
<        REGLE 5 :
<        -------
<                *    UN PROGRAMME EST UN ENSEMBLE STRUCTURE
<                *  DE MODULES ; POUR FACILITER LA COMPREHEN-
```

```
<                *  SION DU TOUT, ON FERA SOIT AU DEBUT, SOIT
<                *  A LA FIN DU LISTING DES COMMENTAIRES D'ORDRES
<                *  GENERAUX CONCERNANT LA OU LES FONCTIONS
<                *  GLOBALES DU PROGRAMME, AINSI QUE LA STRUCTURE
<                *  DE CELUI-CI (VOIR LA PRESENTATION DES SOUS-
<                *  PROGRAMMES, LE PROGRAMME POUVANT ETRE CONSI-
<                *  DERE COMME LE SOUS-PROGRAMME LE PLUS
<                *  EXTERNE).
<                *    ON REDIGERA AVEC PROFIT UN INDEX DU PROGRAMME
<                *  CONTENANT LES NOMS DES SOUS-PROGRAMMES, DE
<                *  CERTAINES DONNEES, CHACUN ETANT ACCOMPAGNE
<                *  D'UNE BREVE DEFINITION, AINSI QUE DU NUMERO
<                *  DE LA PAGE OU IL AURA ETE DEFINI (VOIR
<                *  A CE SUJET L'UTILISATION DE LA DIRECTIVE
<                *  "PAGE", SOUS L'ASSEMBLEUR 'ASSYS-SOLAR').
<                *    ENFIN, LORSQUE CELA EST NECESSAIRE,
<                *  ON RAJOUTERA UN MODE D'EMPLOI "EXTERNE"
<                *  PERMETTANT L'UTILISATION DE CE PROGRAMME
<                *  PAR L'UTILISATEUR "CIBLE"...
<
<
<        REGLE 6 :
<        -------
<                *    DE L'INTERET DU PARAMETRAGE : COMME
<                *  CELA FUT DEJA PRECISE, UN LISTING SERA
<                *  CONSULTE SOIT EN TANT QUE DOCUMENT
<                *  DE TRAVAIL (PEDAGOGIQUE,...), SOIT EN
<                *  TANT QU'OUTIL (LORS DE MODIFICATIONS,
<                *  OU DE MISES AU POINT). IL IMPORTE ALORS
<                *  DE CREER UN MAXIMUM DE LIENS LOGIQUES
<                *  A L'INTERIEUR D'UN PROGRAMME; CELUI-CI
<                *  EST EN FAIT CONSTITUE A L'AIDE DE VALEURS
<                *  NUMERIQUES (CONSTANTES, ETIQUETTES, CODES
<                *  OPERATOIRES); SI CES VALEURS NUMERIQUES
<                *  POSSEDENT DES RELATIONS ENTRE-ELLES, IL
<                *  CONVIENDRA DE LES EXPLICITER. AINSI,
<                *  DONNONS UN EXEMPLE MONTRANT CE QU'IL NE
```

```
<                   *  FAUT PAS FAIRE, PUIS LA "VOTE CELESTE"
<                   *  CONDUISANT A L'OLYMPE DES PROGRAMMEURS :
<TAB:     DZS          13                < DEFINITION D'UNE TABLE DE 13 MOTS.
<         (...)
<         LXI          67                < QUI VOIT LE RAPPORT ENTRE LA
<                                        < DEFINITION DE 'TAB' ET LA CONS-
<                                        < TANTE '67' ???
<                   *  CE RAPPORT ENTRE 'TAB' ET '67' N'EST
<                   *  PAS DIFFICILE A EXPLICITER, ENCORE FAUT-IL
<                   *  LE FAIRE...
<LTAB:    VAL          13                < LONGUEUR DE LA TABLE 'TAB'.
<TAB:     DZS          LTAB              < DEFINITION DE LA TABLE 'TAB', QUI
<                                        < CONTIENDRA (...) SUIVANT LE FORMAT (...).
<K1:      VAL          5                 < CONSTANTE DEFINISSANT (...).
<K2:      VAL          2                 < CONSTANTE DEFINISSANT (...).
<         (...)
<         LXI          LTAB*K1+K2        < ET LA TOUT LE MONDE A COMPRIS...
<                   *    L'AVANTAGE EST DOUBLE :
<                   *  - LA COMPREHENSION DU PROGRAMME EST AMELIORE,
<                   *  - LA MISE A JOUR DU PROGRAMME SIMPLIFIE ;
<                   *  REPRENANT NOTRE EXEMPLE PRECEDENT, UN
<                   *  CHANGEMENT DANS LA LONGUEUR DE LA TABLE
<                   *  'TAB' SE FERA NATURELLEMENT EN MODIFIANT
<                   *  LA CONSTANTE 'LTAB', ET NON PAS EN PARCOU-
<                   *  RANT LE LISTING A LA RECHERCHE DE REFEREN-
<                   *  CES PLUS OU MOINS EXPLICITES A LA LONGUEUR
<                   *  DE CETTE TABLE...
<                   *    EN CONSEQUENCE, TOUTE CONSTANTE UTILISEE
<                   *  SERA DEFINIE SOIT PAR UNE VALEUR NUMERIQUE
<                   *  LORSQU'IL S'AGIRA D'UNE CONSTANTE "PREMIERE"
<                   *  ('LTAB' DANS L'EXEMPLE PRECEDENT), SOIT PAR
<                   *  UNE EXPRESSION N'UTILISANT QUE DES CONSTANTES
<                   *  DEJA DEFINIES DANS LES AUTRES CAS (LTAB*K1+K2
<                   *  DANS L'EXEMPLE PRECEDENT).
<                   *    ON CONSULTERA AVEC PROFIT LA NOTICE D'UTILI-
<                   *  SATION DE L'ASSEMBLEUR 'ASSYS' QUI CONTIENT
<                   *  UN CERTAIN NOMBRE D'OPERATEURS PERMETTANT :
```

```
<                *  - LA GENERATION DYNAMIQUE DE SYMBOLES,
<                *  - LA GENERATION DYNAMIQUE DE CARTES,
<                *  - L'ECRITURE D'EXPRESSION COMPLEXE CONTE-
<                *  NANT LES OPERATEURS ARITHMETIQUES, LOGIQUES
<                *  ET LES AUTRES (???).
<
<
<       REGLE 7 :
<       -------
<                *    QUELQUES CONSEILS DIVERS :
<                *  - LES ETIQUETTES ET SYMBOLES NE
<                *  SERONT DEFINIS QUE PAR L'USAGE
<                *  DES DIRECTIVES 'EQU' ET 'VAL'.
<                *  - LES ZONES DE DONNEES SERONT
<                *  STRUCTUREES : AINSI, ON REGROUPERA
<                *  LES INFORMATIONS FONCTIONNELLEMENT
<                *  IDENTIQUES (RELAIS DE SOUS-PROGRAMMES,
<                *  BUFFERS, MESSAGES, CONSTANTES,...).
<                *  - ON PARTICULARISERA LA PREMIERE
<                *  LETTRES DES SYMBOLES LORSQUE CELA
<                *  EST POSSIBLE (AINSI PAR EXEMPLE,
<                *  LES RELAIS DEBUTERONT PAR LA LETTRE 'A',
<                *  LES MASQUES PAR 'M', LES ETIQUETTES
<                *  PAR 'E',...).
<                *  - ON EXCLUERA TOUTE REFERENCE
<                *  ABSOLUE DANS UN PROGRAMME, AINSI
<                *  PAR EXEMPLE :
<       JMP         $+41
<                *  SERAIT DU PLUS MAUVAIS GOUT !!!
<
<
<       DE BEAUX PROGRAMMES TU REDIGERAS,
<       DES COMMENTAIRES TU ECRIRAS,
<       DES PARAMETRES TU CREERAS,
<       POINT D'OMBRE TU NE LAISSERAS...
<
<
```

```
381   03C1$ 0000 (DF)    AXVAL:    DZS        1                 < NOM+VALEUR
382   03C2$%8364 (F0)    AXASS2:   WORD       ASS2,X            < POUR MOUVMT NOM
383   03C3$%8397 (F1)    AXMTN:    WORD       MTN,X             < IDEM
384   03C4$ 0000 (F2)    ALT:      DZS        1                 < POUR INFOS DEVANT VALEUR
385   03C5$%8397 (E3)    AXMTFI:   WORD       MTFI+4,X          < POUR TITRE SUR LP1
386   03C6$%A000 (F4)    AXTRAV:   WORD       ZERO,X
387   03C7$%A1BC (E5)    AXBFI:    WORD       BFI,X             < BUFFER IMPRIMANTE
388   03C8$ 0378 (F6)    AOBFI1:   WORD       BFI1-7ERO*2       < POUR EDITION IMPRIMANTE
389   03C9$ 03C8 (F7)    AOFCH:    WORD       BFIFCH-7ERO*2     < FIN DES CARACTERES HEXA CODES ASCI
390   03C9                                                     < DANS LE BUFFER IMPRIMANTE.
391   03CA$ 0000 (E8)    AOBUFF:   DZS        1                 < ADR OCTET BUFFER FICHIER
392   03CB$ 0000 (F9)    ABUFF:    DZS        1                 < ADR MOT BUFFER FICH
393   03CC$ 0000 (EA)    PBFI:     DZS        1                 < POINTEUR OCTET SUR BFI :
394   03CC                                                     < POUR LES CARACTERES HEXA CODES ASCI.
395   03CD$ 0000 (EB)    PBFI2:    DZS        1                 < POINTEUR OCTET SUR BFI :
396   03CD                                                     < POUR LES CARACTERES ASCI "IMAGE"
397   03CD                                                     < DES CARACTERES HEXA CODES ASCI.
398   03CE$ 0000 (EC)    ADRC:     DZS        1                 < ADRESSE COURANTE (POUR
399   03CE                                                     < EDITION IMPRIMANTE)
400   03CF$ 0000 (ED)    AOPAG:    DZS        1                 < ADRESSE OCTETS PAGE VIRTUELLE
401   03D0$ 0000 (EE)    AOFPAG:   DZS        1                 < ADR OCT FIN PAGE VIRTUELLE
402   03D1$ 0000 (EF)    AOCRIT:   DZS        1                 < ADRESSE CRITIQE POUR COMPACTAGE (FIN
403   03D1                                                     < DE PAGE-2)
404   03D2$*7FFF (F0)    AOPAG2:   WORD       PAG2              < ADR OCT DEB PAGE SI FICHIER
405   03D3$*7FFF (F1)    AOPAG0:   WORD       PAG0              < ADR OCT DEB PAGE SI AUTRE
406@  03D3                         IF         ORDI-"S",XWOR51,,XWOR51
407   03D4$*7FFF (F2)    ADPAG0:   WORD       FIN       .       < ADR MOT DEB PAGE VIRTUELLE
408@  0000               XWOR51:   VAL        0
```

déplacement par rapport à la base (dans LOCAL, COMMON, DSEC)

zone "instruction" (code généré)

% signifiant constante adresse, ou * signifiant référence en avant.

$ signifiant progression (même nulle) du compteur ordinal.

zone "adresse"

@ en cas de substitution effectuée sur la ligne du source (cf: caractères spéciaux # @ %)

```
03D8  0  AXBP      0048  4  ACK       0364  0  ASS2      035E  0  ASS       0378  0  ASS4      0379  0  ASS5      037E  0  ASSM
037A  0  ASSN      0363  0  ASS1      0377  0  ASS3      037B  0  ASS6      037D  0  ASS7      0383  0  ASSA      03C4  0  ALT
03D9  0  ABP       03DB  0  ABPF      043F  0  ADBV      0456  0  AULR      045A  0  AFDT      046E  0  ASP1      0465  0  APC1
045C  0  ATIK      0463  0  ASTC      045D  0  AFDC      0462  0  AWPG      0470  0  ASP2      03DA  0  AXBPM1    03D1  0  ADCRIT
03A0  0  ASSA2     03A2  0  ASS91     03C7  0  AXBFI     03C6  0  AXTRAV    03B7  0  ADRBL     03B5  0  ADKUF     0362  0  ASSUL
03A2  0  ADKU1     03A3  0  ADKU2     03A4  0  ADKUD     03AE  0  APIIM1    03C0  0  AXNOM     03C1  0  AXVAL     03C2  0  AXASS2
03C5  0  AXMTFI    03C8  0  ADBFI1    03C9  0  ADFCH     03CA  0  ADBUFF    03CB  0  ABUFF     03CF  0  ADPAG     03D0  0  ADFPAG
03D3  0  ADPAG0    03D4  0  ADPAG0    03F2  0  AXMSK1    03BF  0  AM        046D  0  ADMPDK    0452  0  ATRNC     044F  0  ADESAS
0464  0  AFDPG     03F3  0  AXSHF1    03E4  0  AXSHF2    0451  0  AGOSGN    0455  0  ASUPP     0469  0  AGESTM    0459  0  ADFICH
0453  0  ADNOM     044D  0  ADER2     044E  0  ARTCCT    0450  0  ADUDI     0454  0  ADUMP     0457  0  ATSTFT    0458  0  ADITEM
045F  0  ARCUPK    0460  0  AW1PG     0461  0  AWNPG     0467  0  AFNVOT    046A  0  APAR50    046B  0  ACONVH    046C  0  ACONVA
01FC  0  BFIF      01C0  0  BFIH      0435  0  BOX       01BC  0  BFI       01BC  0  BFI1      024D  0  BPF       022F  0  BVF
01F8  0  BFIASC    01F4  0  BFIFCH    003C  0  BFDKU     01FD  0  BP        01FD  0  BV        05AD  0  BLOCV     05AA  0  BLOCI
03D6  0  CCMP      0010  4  CLEFS     07E2  0  CONVH     0803  0  CONVA     07F1  0  CONVH2    07F9  0  CONVH3    07F5  0  CONVH1
0807  0  CONVA1    03E6  0  DIX       0012  0  DFB1      0474  0  DFB2      0349  0  DRBI      040C  0  DMTI      0547  0  DUMP
0559  0  DMF       0552  0  DMI       054B  0  DM1       0A12  0  DVAS1     0A1F  0  DVAS2     0A22  0  DVAS3     0A28  0  DVAS4
0001  4  DIALOG    0386  0  DMSUB0    043D  0  DMCISK    03F7  0  DMRFP     0444  0  DMWDKU    043E  0  DMLVI     0493  0  DKPAR1
042C  0  DMOPN     03F4  0  DMASNS    0412  0  DMTIK     040F  0  DMNINT    0426  0  DMLIG1    041D  0  DMIDK     0418  0  DMSPI
0448  0  DMRDKU    03FB  0  DMASS     03FD  0  DMOUT     03F7  0  DIXMIT    03F4  0  DMNBOC    055E  0  DITFM     043A  0  DMWBLC
041B  0  DMIDK1    03EE  0  DMASDK    0441  0  DMRACK    0415  0  DMSKIP    040B  0  DMCCI     03F0  0  DMRDK     0407  0  DMLON
0420  0  DMIAS     0423  0  DMDBG     0429  0  DMLIG2    042D  0  DMCIS     042E  0  DMREAD    0431  0  DMPCH     0434  0  DMTMPO
07FD  0  DFSAS     0571  0  DFICH     0736  0  DMPDK     0488  0  DKPAR     048E  0  DKPAR2    056C  0  DMPIV     057B  0  DMPF1
0583  0  DMPF3     058E  0  DMPF4     0595  0  DMPF6     0772  0  DMPDK1    05A4  0  DMPF9     0792  0  DMPDK7    07B1  0  DMPDK8
0579  0  DMPFI     0761  0  DMPDKF    074A  0  DMPDKH    0564  0  DMPTI     055D  0  DMFIN     059D  0  DMPF5     0593  0  DMPF41
0768  0  DMPDK3    0758  0  DMPDKK    0765  0  DMPDKJ    0758  0  DMPDK0    0759  0  DMPDKF    076A  0  DMPDK4    077D  0  DMPDK5
07B2  0  DMPDK9    07C5  0  DMPDKI    05BD  0  EDI1      05FD  0  EDC       05F6  0  EDC1      05F1  0  EDC2      05B2  0  EDI
082B  0  ENV1      0830  0  ENV2      0922  0  EDPGP     0927  0  EDPG1     093C  0  EDPG2     0940  0  EDPG3     0969  0  EDPG8
0973  0  EDPGD     0971  0  EDPGF     092F  0  EDPGH     0934  0  EDPGV     0966  0  EDPG9     0979  0  EDPGD2    098A  0  EDPGD1
03FA  0  ESPACF    0825  0  ENVOI     05C5  0  EDI10     05DB  0  EDI13     05B8  0  EDI14     05D4  0  EDI11     05C9  0  EDI100
0A2E  0  FIN       0834  0  GOSGN     083E  0  GFSTM     084B  0  GFSTMF    0391  0  IPCH      03BF  0  IPRW1P    03BC  0  IDEBUG
03BA  0  IONOM     03BB  0  INDFI     03BD  0  IPRDM     03BF  0  IPRWPG    03AD  0  KN1       03AE  0  KN2       0190  4  LPAV
03B9  0  LNC       044D  0  LOC       017F  4  LPAD      017F  4  LPAF      0032  4  LRV       0080  4  LPAP      03B8  0  LPP
03BA  0  LPF       03BB  0  LPD       0032  4  LRUFV     02B2  0  MAS       02C3  0  MQV       031A  0  MSPI      02A2  0  MULB
02B6  0  MPAS      03AB  0  MTF       0324  0  MNI1      031B  0  MACK      0267  0  MDSB      02C6  0  MASD      02CB  0  MASF
0327  0  MTIK      02B4  0  MNS       02BD  0  MRAC      0278  0  MDPF      02B0  0  MQDK      02E0  0  MTMP      0313  0  MQFS
```

EXTRAIT D'UNE TABLE DES SYMBOLES

```
<
<          N U M E R O S   D E S   M E S S A G E S   D ' E R R E U R S
<
<          D ' A S S E M B L A G E
<
< 1        ERREUR D'ECRITURE .
< 2        ERREUR DANS UNE EXPRESSION .
< 3        ETIQUETTE INTERDITE OU MANQUANTE .
< 4        SYMBOLE INCORRECT .
< 5        EXPRESSION INTERDITE .
< 6        EXPRESSION NON DEFINIE .
< 7        DEPLACEMENT HORS-LIMITES .
< 8        DOUBLE DEFINITION .
< 9        SATURATION DES TABLES .
< 10       CHAINAGE INCORRECT
< 11       ADRESSE INACCESSIBLE .
< 12       PASSAGE D'UNE ADRESSE AU DELA DE 64K .
< 13       CONTEXTE IMPROPRE .
< 14       ADRESSE SUPERIEURE A 32K SUR LAQUELLE
<          UNE INDEXATION A ETE DEMANDEE .
< 15       LIMITE DE SECTION 'PROG' NON ATTEINTE .
< 16       ENCHAINEMENT DES SECTIONS INCORRECT .
< 17       SYMBOLE(S) NON DEFINI(S) LORS DE LA RENCONTRE DU 'END' .
< 18       DEBORDEMENT SUR UN 'WORD' PARAIT-IL...
< 19       LE SYMBOLE TRANSLATABLE DEFINI VAUT '7FFF, OR
<          CETTE VALEUR INDIQUE UNE FIN DE CHAINAGE.
```

.../...

```
< 20      CHAINAGE SUPERIEUR A 32K, LORS DE LA RENCONTRE
<         D'UNE REFERENCE EN AVANT SUR LAQUELLE UNE
<         INDEXATION EST DEMANDEE; LORSQU'UNE REFERENCE
<         EN AVANT INDEXEE SERA FAITE AU DELA DE 32K,
<         (PORTANT EVIDEMMENT SUR UNE VALEUR INFERIEURE
<         A 32K DEFINIE PAR EXEMPLE PAR UN RETOUR ARRIERE
<         DU $), ON CREERA AUTANT DE SYMBOLES DIFFERENTS
<         QU'IL Y AURA DE REFERENCES INDEXEES; TOUS CES
<         SYMBOLES AYANT RIEN ENTENDU LA MEME VALEUR...
< 21      SATURATION DE LA TABLE DES SYMBOLES (FAUT LE FAIRE !!!)
< 22      LORS D'UNE REFERENCE EN AVANT, UN CHAINAGE>32K
<         EST SUIVI D'UN CHAINAGE<32K : ON INTERDIT AINSI
<         DES SYMBOLES<32K DEFINIS APRES AVOIR ETE REFE-
<         RENCES AU-DELA DES 32K (IL Y AURAIT ALORS POUR
<         POUR L'EDITEUR DE LIENS UNE AMBIGUITE ENTRE
<         LES CHAINAGES>32K, ET LES CHAINAGES<32K INDEXES!!!);
<         LA SOLUTION POUR PASSER OUTRE CETTE POSSIVBILITE
<         EST DE CREER AUTANT DE SYMBOLES EQUIVALENTS
<         QUE NECESSAIRES...
< 23      ETIQUETTE SUR UNE INSTRUCTION DE L'OPTION CDA.
< 24      ERREUR D'ACCES A UN FICHIER SUR 'EOT'.
< 25      'ED' ALORS QU'IL Y A UN 'EOT' EN COURS.
```

COMMANDES SPECIFIQUES A ASSYS-SOLAR

(ET DONC INCONNUES DE ASSYS-T1600)

- '@' COLONNE 33
- '@' COLONNE 34
- ASCI "'ABCD..." (WORD "LONG")
- DATE
- MOT
- CALL (UTILISER EST SUR T1600)
- CD3 (SEULEMENT CD1 ET CD2 SUR T1600)
- LES "FONCTIONS D'EXPRESSIONS" (SOM='FXYZ) SUIVANTES:
  - 'FDYZ (DERNIER RESTE)
  - 'FEYZ (PUSH, PULL, PULL ET EXECUTE)
  - 'FFYZ (TEST DE SIGNE)
- LES DOUBLEMENTS DE ":"
  - X:: VAL Y
  - ETTQ:: EQU 7

```
1           NLS
2   <
3   <       P R O C E D U R E   ' S I   P A G E '
4   <
5   <       FONCTION:
6   <       --------
7   <                   A APPELER POUR CHAQUE SOUS-PROGRAMME D'UN SOURCE.
8   <                   A PARTIR DES PARAMETRES QU'ON LUI FOURNIT EN
9   <                   CARTE DYNAMIQUE 1, CETTE PROCEDURE:
10  <                   - FAIT UN 'PAGE',
11  <                   - GENERE L'ETIQUETTE DU POINT D'ENTREE DU S/P,
12  <                   - REPERTORIE CE S/P ET SON NUMERO DE PAGE SUR LE
13  <                     LISTING, DANS UN FICHIER 'SI DOC' DE DOCUMENTA-
14  <                     TION. IL SUFFIRA DE LISTER CE FICHIER (PAR UN
15  <                     "  EOT   #SI DOC#  " EN FIN DE PROGRAMME) POUR
16  <                     OBTENIR LA TABLE DES MATIERES DES SOUS-PROGRAMMES.
17  <
18  <       APPEL:
19  <       -----
20  <                   METTRE EN CARTE DYNAMIQUE 1 LES PARAMETRES, PAR EXEMPLE:
21  <
22  <       S/P READFI: SOUS-PROGRAMME DE LECTURE FICHIER.
23  <       (AVEC LE @ EN COLONNE 8).
24  <
25  <                   PUIS APPELER LA PROCEDURE, COMME CECI:
26  <
27  <       CALL        #SI PAGE#
28  <
29  <
30  <       NOTA:
31  <       ----
32  <                   AVANT LE PREMIER APPEL DE LA PROCEDURE, FAIRE UN
33  <                   'DELETE FILE' DU FICHIER 'SI DOC'.
34  <                   EN FIN DE PROGRAMME, FAIRE UN EOT  #SI DOC#  POUR
35  <                   EDITER LA TABLE DES MATIERES, PUIS UN 'DELETE FILE'
36  <                   DU FICHIER 'SI DOC'.
37  <
38  XY7003:  VAL        4                   < NOMBRE DE CHIFFRES TRAITES LORS
39                                          < DE LA CONVERSION HEXADECIMAL-->
40                                          < DECIMAL DU NUMERO DE PAGE.
```

```
41  XYZ001:  VAL          '0959='F347      < DEPLACER LE COMMENTAIRE EN CD2.
42  XYZW:    VAL          '50='F220(0+'5051='F308(0+'50='F23C
43                                         < MISE A BLANC DE CD2('50-'58) ET
44                                         < MISE DE "<" EN '50.
45           @:F;:F;:F;
46  XYZ001:  VAL          '09A0='F309      < PLACER LES :F EN CD3.
47           LST                             ↝
48           PAGE
49           NLS
50  XYZ001:  VAL          '0@@@@           < RECUPERATION DU NUMERO DE PAGE
51                                         < EN HEXADECIMAL.
52           DO           XYZ003           < POUR CONVERSION EN DECIMAL.
53  XYZ001:  VAL          XYZ001/10(0='FD00='FE00(0+XYZ001/10
54  XYZ001:  VAL          0
55           DO           XYZ003           < RECUPERATION NOMBRE DECIMAL.
56  XYZ001:  VAL          XYZ001*'10='FF01
57                                         < AFFICHAGE SUR LE LISTING DE
58                                         < "PAGE NUMERO : NNNN"
59  <        @              @@@@                                PAGE NUMERO :
60  XYZW:    VAL          '1745='F304      < PLACER LE NUMERO DE PAGE.
61  XYZW:    VAL          '0117='F304      < EFFACER LE NUMERO MAL PLACE.
62           LST
63         @@
64           NLS
65           @EQU         $
66  XYZW:    VAL          '5D00='F307      < PLACER L'ETIQUETTE DE L'ENTRY
67                                         < DU SOUS-PROGRAMME.
68           LST
69         @@
70           NLS
71  XYZ001:  VAL          XYZ001           < RECUPERATION DU NUMERO DE PAGE
72                                         < EXPRIME EN DECIMAL QUE L'ON
73                                         < VA GENERER EN CARTE DYNAMIQUE.
74  <        @PAGE NUMERO : @@@@;
75  XYZ001:  VAL          '00A9='F31C      < SAUVEGARDE EN CD3.
76  ;ED      @'SI DOC';IF;<;
77  XYZ001:  VAL          'A917='F31C(0+'5033='F359
78           EF
79           LST
```

```
1337            ADRT        MNI1-MNT*2,Y    < ADRESSE OCTET EDITION NOM INTERNE
1338            BSR         ACONVA          < PLACE LE NOM INTERNE EN ASCI.
1339            LAD         DMNTNT          < EDITION NOM
1340            SVC         0
1341  TIF:      EQU         $
1342            RSR
1343           @S/P TIK:    EDITION TITRE-KEY SUR IMPRIMANTE/OUTPUT
1344            CALL         #ST PAGE#
1345  <
1346  <         E D I T I O N   T I T R E - K E Y   S U R   I M P R I M A N T E /
1347  <
1348  <         O U T P U T
1349  <
1350            PSR         A,X,Y           < SAUVEGARDES.
1351  <
1352            LA          KN1             < N1 DE LA CLE.
1353            LY          DMTTK+1
1354            ADRT        MTIK1-MTIK*2,Y  < ADRESSE OCTET EDITION DE N1.
1355            BSR         ACONVA          < QUI PLACE N1.
1356  <
1357            LA          KN2             < N2 DE LA CLE.
1358            ADRT        MTIK2-MTIK1*2,Y < ADRESSE OCTET EDITION DE N2.
1359            BSR         ACONVA          < QUI EDITE N2.
1360  <
1361            LAD         DMTTK           < EDITION DU TITRE - KEY .
1362            SVC         0
1363            PLR         A,X,Y           < RESTAURATIONS.
1364            RSR
1365           @S/P TSIFI:  TEST ITEM OU FICHIER
1366            CALL         #ST PAGE#
```

} paramètres et appel de 'SI PAGE' (fichier source)

```
0034  SI PAGE

    632 0000                <
    692 0618                TIK:      EQU         $                                  PAGE NUMERO : 0052
SI EXEMPLE PAGE
  1344  0000                <
  1345  0000                <         E D I T I O N   T I T R E - K E Y   S U R   I M P R I M A N T E /
  1346  0000                <
  1347  0000                <         O U T P U T
  1348  0000                <
  1349  0618$ 1AB0                    PSR         A,X,Y               < SAUVEGARDES.
  1350  0618                <
  1351  0619$ 50CB                    LA          KN1                 < N1 DE LA CLF.
  1352  061A$ 5631                    LY          DMTTK+1
  1353  061B$ 0808                    ADRI        MTIK1-MTIK*2,Y      < ADRESSE OCTET EDITION DE N1.
  1354  061C$ 869F                    BSR         ACONVA              < QUI PLACE N1.
  1355  061C                <
  1356  061D$ 50CC                    LA          KN2                 < N2 DE LA CLF.
  1357  061E$ 0808                    ADRI        MTIK2-MTIK1*2,Y     < ADRESSE OCTET EDITION DE N2.
  1358  061F$ 869F                    BSR         ACONVA              < QUI EDITE N2.
  1359  061F                <
  1360  0620$ 5D30                    LAD         DMTTK               < EDITION DU TITRE - KEY .
  1361  0621$ 1C00                    SVC         0
  1362  0622$ 1B0D                    PLR         A,X,Y               < RESTAURATIONS.
  1363  0623$ 1E02                    RSR
          S/P ISTFI:  TEST ITEM OU FICHIER
  1365  0623                          CALL         #ST PAGE#
SI PAGE
```

généré par 'SI PAGE'

```
3070             LR          B,A              < RESULTAT
3071             PLR         W
3072             RSR
3073  XWORX1:    VAL         0
3074  FIN:       EQU         $                < FIN DU PROGRAMME
3075  TOTO:      VAL         FIN-ZERO*2
3076  PAGU:      EQU         ZERO+TOTO
3077  PAG2:      EQU         ZERO+TOTO+2
3078             LST
3079             EST
3080             PAGE
3081  <
3082  <          T A B L E   D E S   M A T I E R E S
3083  <
3084             EOI         #SI DOC#
3085  :DF        @'SI DOC';:F;            < SUPPRESSION DU FICHIER 'SI DOC'
3086             EE
3087             END
```

liste et suppression du fichier 'SI DOC'

liste de 'SI DOC'

```
<
<          PAGE NUMERO : 0002
<          S/P QUOI:    QUOI FAIRE? DUMP,SUPPRESSION,ROF?
<
<          PAGE NUMERO : 0003
<          S/P QNOM:    QUE FAIRE SUR LE NOM EN COURS ?
<
<          PAGE NUMERO : 0004
<          S/P TRNC:    TRAITEMENT DU NOM EN COURS
<
<          PAGE NUMERO : 0005
<          S/P SUPP:    SUPPRESSION FICHIER OU ITEM EN COURS
<
<          PAGE NUMERO : 0006
<          S/P DUMP:    DUMP FICHIER OU ITEM EN COURS
<
<          PAGE NUMERO : 0007
<          S/P DITEM:   DUMP ITEM EN COURS
<
<          PAGE NUMERO : 0008
<          S/P DFICH:   DUMP FICHIER EN COURS
<
<          PAGE NUMERO : 0009
<          S/P BLOC:    DUMP D'UN BLOC DE FICHIER
<
<          PAGE NUMERO : 0010
<          S/P EDI:     EDITION SUR IMPRIMANTE/OUTPUT
<
<          PAGE NUMERO : 0011
<          S/P EDC:     EDITION DE 2 CARACTERES SUR IMPRIMANTE/OUTPUT
<
<          PAGE NUMERO : 0012
<          S/P TT:      EDITION TITRE SUR IMPRIMANTE/OUTPUT
<
<          PAGE NUMERO : 0013
<          S/P TTK:     EDITION TITRE-KEY SUR IMPRIMANTE/OUTPUT
```

```
37  <
38  <          PAGE NUMERO : 0014
39  <          S/P TSTFI:  TEST ITEM OU FICHIER
40  <
41  <          PAGE NUMERO : 0015
42  <          S/P ULB:    CHOIX D'UN SUPPORT DE DUMP A ASSIGNER A L'UL 'B
43  <
44  <          PAGE NUMERO : 0016
45  <          S/P SP1:     OPERATIONS DE FIN DE PARCOURS DE L'ARBRE
46  <
47  <          PAGE NUMERO : 0017
48  <          S/P SP2:     INITIALISATIONS SUR 'DKU'
49  <
50  <          PAGE NUMERO : 0018
51  <          S/P RCUPK:  RECUPERATION DE LA CLE DE L'ENREGISTREMENT EN COURS
52  <
53  <          PAGE NUMERO : 0019
54  <          S/P DMPDK:  DUMP DISQUE
55  <
56  <          PAGE NUMERO : 0020
57  <          S/P CONVH:  CONVERSION ASCI --> HEXA
58  <
59  <          PAGE NUMERO : 0021
60  <          S/P DESAS:  DESASSIGNATION D'UNE UNITE LOGIQUE
61  <
62  <          PAGE NUMERO : 0022
63  <          S/P CONVA:  CONVERSION HEXA --> ASCI
64  <
65  <          PAGE NUMERO : 0023
66  <          S/P RTCCI:  RETOUR AU CCI APRES DESASSIGNATION DES UL 3 ET 'B
67  <
68  <          PAGE NUMERO : 0024
69  <          S/P QREP:   ENVOI QUESTION ET DEMANDE REPONSE
70  <
71  <          PAGE NUMERO : 0025
72  <          S/P ENVOI:  ENVOI D'UN MESSAGE STANDARE SUR L'ORGANE DE SOUTIE
```

```
73  <
74  <          PAGE NUMERO : 0026
75  <          S/P GOSGN:  ACCES AU SGN
76  <
77  <          PAGE NUMERO : 0027
78  <          S/P GESTM:  GESTION MEMOIRE (OU AJUSTEMENT)
79  <
80  <          PAGE NUMERO : 002A
81  <          S/P W1PG:   WRITE 1 CARACTERE SUR PAGE VIRTUELLE
82  <
83  <          PAGE NUMERO : 0029
84  <          S/P STC:    STORE OCTET EN PAGE VIRTUELLE (COUNT OU CARACTERE)
85  <
86  <          PAGE NUMERO : 0030
87  <          S/P WNPG:   WRITE N CARACTERES EN PAGE VIRTUELLE
88  <
89  <          PAGE NUMERO : 0031
90  <          S/P WPG:    WRITE UNE PAGE VIRTUELLE
91  <
92  <          PAGE NUMERO : 0032
93  <          S/P EDPG:   WRITE PROPREMENT DIT D'UNE PAGE SUR SUPPORT EXTERNE
94  <
95  <          PAGE NUMERO : 0033
96  <          S/P PC1:    PREPARATION PUNCH DE 1 OCTET
97  <
98  <          PAGE NUMERO : 0034
99  <          S/P PCARD:  PUNCH CARTE
100 <
101 <          PAGE NUMERO : 0035
102 <          S/P DVAS:   DEMANDE ET VERIFICATION D'UNE ADRESSE DKU
```

```
 1              PROG
 2  ZERO·       EQU         $
 3              DZS         100
 4              PAGE
 5  <
 6  <           E X E M P L E   D ' U T I L I S A T I O N   D E
 7  <
 8  <           L A   F O N C T I O N   ' F C X Y .
 9  <
10  <
11  XCD2·       VAL         '50             < INDEX OCTET DE DEBUT DE GENERATION
12                                          < EN CARTES DYNAMIQUES DU PROGRAMME
13                                          < A EXECUTER PAR LA FONCTION 'FCXY.
14  XEQU·       VAL         $-ZERO          < SAUVEGARDE DU '$' COURANT.
15  <
16  <           GENERATION ET EXECUTION DU PROGRAMME.
17  <
18  XBUF·       ASCI        "!D"            < BUFFER MESSAGE.
19            $@                            < $SI STOCD2$
20              ASCI        "AT"
21            $@                            < $SI STOCD2$
22              BYTE        "E",'04
23            $@                            < $SI STOCD2$
24  XBUFF·      EQU         $               < FIN DU BUFFER.
25  XDEM·       WORD        '0002           < DEMANDE AU CCI.
26            $@                            < $SI STOCD2$
27              WORD        0               < ADRESSE OCTET BUFFER.
28            $@                            < $SI STOCD2$
29              WORD        XBUFF-XBUF*2    < LONGUEUR OCTET.
30            $@                            < $SI STOCD2$
```

cf. 'SI STOCD2' ci-après.

.../...

```
31  XDEB·     VAL         XCD2              < VALEUR COURANTE DE L'INDEX
32                                          < OCTET EN CARTE DYNAMIQUE (VOIR
33                                          < LA PROCEDURE 'SI STOCD2').
34            PSR         A.B.X.L           < SAUVEGARDES.
35          $@                              < $SI STOCD2$
36  XL·       LRP         L
37          $@                              < $SI STOCD2$
38            LAD         XBUF-XL.L         < ADRESSE MOT DU MESSAGE.
39          $@                              < $SI STOCD2$
40            SLLS        1                 < ADRESSE OCTET DU MESSAGE.
41          $@                              < $SI STOCD2$
42            STA         XDEM+1-XL.L       < MOT 1 DE LA DEMANDE AU CCI.
43          $@                              < $SI STOCD2$
44            LAD         XDEM-XL.L         < ADRESSE DE LA DEMANDE.
45          $@                              < $SI STOCD2$
46            SVC         0                 < DEMANDE...
47          $@                              < $SI STOCD2$
48            PLR         A.B.X.L           < RESTAURATIONS.
49          $@                              < $SI STOCD2$
50            RSR                           < RETOUR A 'ASSYS'.
51          $@                              < $SI STOCD2$
52            DO          6
53  X·        VAL         XDEB/2+'FC00-'FE00(0-'FE02
54                                          < EXECUTION DU PROGRAMME PLACE EN
55                                          < 'XDEB'/2 (INDEX MOTS) EN CARTE
56                                          < DYNAMIQUE.
57            $EQU        ZERO+XEQU         < ON REND L'ESPACE UTILISE...
58  <
59  <         SUITE DU PROGRAMME.....
60  <
```

```
+LN'SI STOCD2'
     1  X·          VAL          '0000000000
     2  XM8·        VAL          -8
     3  X1·         VAL          X('FF00)XM8+'F200='FE00(0+XCD2='FE02
     4  XCD2·       VAL          XCD2+1
     5  X1·         VAL          X('00FF+'F200='FE00(0+XCD2='FE02
     6  XCD2·       VAL          XCD2+1

+
```

'SI STOCD2' récupère une instruction et la génère en carte dynamique 2.

```
0002  SI EXEMPLE

     5  0002                <
     6  0002                <        E X E M P L E   D ' U T I L I S A T I O N   D E
     7  0002                <
     8  0002                <        L A   F O N C T I O N   ' F C X Y .
     9  0002                <
    10  0002                <
    11  0050                XCD2:    VAL         '50              < INDEX OCTET DE DEBUT DE GENERATION
    12  0050                                                      < EN CARTES DYNAMIQUES DU PROGRAMME
    13  0050                                                      < A EXECUTER PAR LA FONCTION 'FCXY.
    14  0064                XEQU:    VAL         $-ZERO           < SAUVEGARDE DU '$' COURANT.
    15  0064                <
    16  0064                <        GENERATION ET EXECUTION DU PROGRAMME.
    17  0064                <
    18  0064$ 2144          XBUF:    ASCT        "!D"             < BUFFER MESSAGE.
    19  0064$ 2144                 #                              < #SI STOCD2#
SI STOCD2
     1# 2144                X:       VAL         '000000002144
     2  FFF8                XM8:     VAL         -8
     3  0021                X1:      VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0051                XCD2:    VAL         XCD2+1
     5  0044                X1:      VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  0052                XCD2:    VAL         XCD2+1
SI EXEMPLE
    20  0065$ 4154                   ASCT        "AT"
    21  0065$ 4154                 #                              < #SI STOCD2#
SI STOCD2
     1# 4154                X:       VAL         '000000004154
     2  FFF8                XM8:     VAL         -8
```

PAGE BLANCHE SUITE A ERREUR DE PAGINATION.

```
0003  SI STOCD2

     3  0041                X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0053                XCD2:     VAL         XCD2+1
     5  0054                X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FE02
     6  0054                XCD2:     VAL         XCD2+1
SI EXEMPLE
    22  0066$ 4504                    BYTE        "F";'04
    23  0066$ 4504                #                                   < #SI STOCD2#
SI STOCD2
     1@ 4504                X:        VAL         '00000004504
     2  FFF8                XM8:      VAL         -8
     3  0045                X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0055                XCD2:     VAL         XCD2+1
     5  0004                X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  0056                XCD2:     VAL         XCD2+1
SI EXEMPLE
    24  0067                XBUFF:    EQU         $                   < FIN DU BUFFER.
    25  0067$ 0002          XDEM:     WORD        '0002               < DEMANDE AU CCI.
    26  0067$ 0002                #                                   < #SI STOCD2#
SI STOCD2
     1@ 0002                X:        VAL         '00000000002
     2  FFF8                XM8:      VAL         -8
     3  0000                X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0057                XCD2:     VAL         XCD2+1
     5  0002                X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FE02
     6  0058                XCD2:     VAL         XCD2+1
SI EXEMPLE
    27  0068$ 0000                    WORD        0                   < ADRESSE OCTET BUFFER.
    28  0068$ 0000                #                                   < #SI STOCD2#
```

```
0004  SI EXEMPLE

SI STOCD2
     1@ 0000              X:        VAL         '00000000000
     2  FFF8              XM8:      VAL         -8
     3  0000              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0059              XCD2:     VAL         XCD2+1
     5  0000              X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  005A              XCD2:     VAL         XCD2+1
SI EXEMPLE
    29  0069$ 0006                  WORD        XBUFF-XBUF*2      < LONGUEUR OCTET.
    30  0069$ 0006              #                                 < #SI STOCD2#
SI STOCD2
     1@ 0006              X:        VAL         '00000000006
     2  FFF8              XM8:      VAL         -8
     3  0000              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  005B              XCD2:     VAL         XCD2+1
     5  0006              X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  005C              XCD2:     VAL         XCD2+1
SI EXEMPLE
    31  005C              XDF8:     VAL         XCD2              < VALEUR COURANTE DE L'INDEX
    32  005C                                                      < OCTET EN CARTE DYNAMIQUE (VOIR
    33  005C                                                      < LA PROCEDURE 'ST STOCD2').
    34  006A$ 1AF4                  PSR         A,B,X,L           < SAUVEGARDES.
    35  006A$ 1AE4              #                                 < #SI STOCD2#
SI STOCD2
     1@ 1AE4              X:        VAL         '00000001AE4
     2  FFF8              XM8:      VAL         -8
     3  001A              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FF02
     4  005D              XCD2:     VAL         XCD2+1
```

```
0005  SI STOCD2

      5  00E4              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FE02
      6  005E              XCD2:     VAL       XCD2+1
SI EXEMPLE
     36  0068$ 2FED        XL:       LRP       L
     37  0068$ 2FED                  #                              < #SI STOCD2#
SI STOCD2
     1@ 2FFD               X:        VAL       '00000002FED
      2  FFF8              XM8:      VAL       -8
      3  002F              X1:       VAL       X('FF00>XM8+'F200='FE00(0+XCD2='FE02
      4  005F              XCD2:     VAL       XCD2+1
      5  00ED              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FE02
      6  0060              XCD2:     VAL       XCD2+1
SI EXEMPLE
     38  006C$ 9DF9                  LAD       XBUF-XL,L          < ADRESSE MOT DU MESSAGE.
     39  006C$ 9DF9                  #                            < #SI STOCD2#
SI STOCD2
     1@ 9DF9               X:        VAL       '00000009DF9
      2  FFF8              XM8:      VAL       -8
      3  009D              X1:       VAL       X('FF00>XM8+'F200='FE00(0+XCD2='FE02
      4  0061              XCD2:     VAL       XCD2+1
      5  00F9              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FE02
      6  0062              XCD2:     VAL       XCD2+1
SI EXEMPLE
     40  006D$ 2981                  SLLS      1                  < ADRESSE OCTET DU MESSAGE.
     41  006D$ 2981                  #                            < #SI STOCD2#
SI STOCD2
     1@ 2981               X:        VAL       '00000002981
      2  FFF8              XM8:      VAL       -8
```

```
0006  SI STOCD2

     3  0029              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0063              XCD2:     VAL         XCD2+1
     5  0081              X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  0064              XCD2:     VAL         XCD2+1
SI EXEMPLE
    42  006E$ 8DFD                   STA         XDEM+1-XL,L      < MOT 1 DE LA DEMANDE AU CCI.
    43  006E% 8DFD              #                                  < #SI STOCD2#
SI STOCD2
     1@ 8DFD              X:        VAL         '00000008DFD
     2  FFF8              XM8:      VAL         -8
     3  008D              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0065              XCD2:     VAL         XCD2+1
     5  00FD              X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  0066              XCD2:     VAL         XCD2+1
SI EXEMPLE
    44  006F$ 9DFC                   LAD         XDEM-XL,L        < ADRESSE DE LA DEMANDE.
    45  006F% 9DFC              #                                  < #SI STOCD2#
SI STOCD2
     1@ 9DFC              X:        VAL         '00000009DFC
     2  FFF8              XM8:      VAL         -8
     3  009D              X1:       VAL         X('FF00>XM8+'F200='FE00(0+XCD2='FE02
     4  0067              XCD2:     VAL         XCD2+1
     5  00FC              X1:       VAL         X('00FF+'F200='FE00(0+XCD2='FF02
     6  0068              XCD2:     VAL         XCD2+1
SI EXEMPLE
    46  0070$ 1C00                   SVC         0                < DEMANDE...
    47  0070% 1C00              #                                  < #SI STOCD2#
SI STOCD2
```

```
0007  SI STOCD2

     1@ 1C00              X:        VAL       '00000001C00
     2  FFF8              XM8:      VAL       -8
     3  001C              X1:       VAL       X('FF00>XM8+'F200='FE00(0+XCD2='FF02
     4  0069              XCD2:     VAL       XCD2+1
     5  0000              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FE02
     6  006A              XCD2:     VAL       XCD2+1
SI EXEMPLE
    48  0071$ 1B27                  PLR       A,B,X,L          < RESTAURATIONS.
    49  0071$ 1B27            #                              < #SI STOCD2#
SI STOCD2
     1@ 1B27              X:        VAL       '00000001B27
     2  FFF8              XM8:      VAL       -8
     3  001B              X1:       VAL       X('FF00>XM8+'F200='FE00(0+XCD2='FF02
     4  006B              XCD2:     VAL       XCD2+1
     5  0027              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FE02
     6  006C              XCD2:     VAL       XCD2+1
SI EXEMPLE
    50  0072$ 1E02                  RSR                        < RETOUR A 'ASSYS'.
    51  0072$ 1E02            #                              < #SI STOCD2#
SI STOCD2
     1@ 1E02              X:        VAL       '00000001F02
     2  FFF8              XM8:      VAL       -8
     3  001E              X1:       VAL       X('FF00>XM8+'F200='FE00(0+XCD2='FF02
     4  006D              XCD2:     VAL       XCD2+1
     5  0002              X1:       VAL       X('00FF+'F200='FE00(0+XCD2='FF02
     6  006E              XCD2:     VAL       XCD2+1
SI EXEMPLE
    52  006E                        DO        6
```

```
0008  SI EXEMPLE

 28/08/1979-14/22/28
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
 28/08/1979-14/22/28
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
 28/08/1979-14/22/28
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
 28/08/1979-14/22/28
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
 28/08/1979-14/22/28
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
 28/08/1979-14/22/29
    53  0000                X:        VAL         XDEB/2+'FC00='FE00(0='FF02
    54  0000                                                < EXECUTION DU PROGRAMME PLACE EN
    55  0000                                                < 'XDEB'/2 (INDEX MOTS) EN CARTE
    56  0000                                                < DYNAMIQUE.
    57  0064$                         $EQU        ZERO+XEQU < ON REND L'ESPACE UTILISE...
    58  0064                <
    59  0064                <         SUITE DU PROGRAMME.....
    60  0064                <
>
```

```
1   <
2   <       DOCUMENTATION DE ASSYS  EXEMPLE D'UTILISATION DE 'MOT'
3   <
4   SPTST   EQU     $
5   <
6   <       CE SOUS-PROGRAMME TESTE UNE FIN DE CHAINAGE
7   <       AVANT/ARRIERE DANS UNE TABLE
8   <
9   <       ARGUMENTS
10  <               - 'A' = 0 : TEST FIN DE CHAINAGE AVANT DEMANDE
11  <                     $ 0 : TEST FIN DE CHAINAGE ARRIERE DEMANDE
12  <               - 'U' = ADRESSE DU POSTE COURANT DE LA TABLE
13  <
14  <       RESULTAT
15  <               - 'B' = 0 : FIN DE CHAINAGE
16  <                     $ 0 : CE N'EST PAS LA FIN DE CHAINAGE
17  <
18  <       AU RETOUR, FAIRE JE/JNE
19  <
20  <
21  <       DESCRIPTION D'UN POSTE DE LA TABLE
22  <
23  VALEUR  MOT     0               < VALEUR ELEMENT
24  CHAV    MOT     1               < CHAINAGE AVANT
25  CHAR    MOT     2               < CHAINAGE ARRIERE
26  BITFC   VAL     14              < BIT INDIQUANT LA FIN DE CHAINAGE
27  <
28          PSR     A               < SAUVEGARDE
29          LBI     0               < FIN DE CHAINAGE A PRIORI
30          JANE    SPTST1          < CHAINAGE ARRIERE A TESTER
31          LA      CHAV,U          < CHAINAGE ARRIERE A TESTER
32          JMP     SPTST2
33  SPTST1  EQU     $
34          LA      CHAR,U          < CHAINAGE ARRIERE A TESTER
35  SPTST2  EQU     $
36          TBT     BITFC           < FIN DE CHAINAGE ?
37          JNC     $+2             < OUI
38          ADRI    1,B             < NON
39          CPZR    B               < POUR TEST EN RETOUR
40          PLR     A               < RESTAURATION
41          RSR
```

LINK : EDITEUR DE LIENS.

Ce programme est utilisable pour "linker" un programme objet, donc déjà assemblé : à partir du source, on doit donc procéder de la façon suivante :

- assembler le source (par ASSYS) en précisant que l'on veut un fichier binaire à qui l'on donne un nom (voir utilisation de ASSYS)

- assembler éventuellement d'autres sources de la même façon, si l'on veut linker ensemble plusieurs binaires.

- appeler LINK :

```
?!CALL D^C
}
MØDE = Q
NØM = LINK D^C
?!GØ D^C
FICH = BØ  ØBJET1 D^C
FICH = BØ  ØBJET2 D^C
FICH = D^C
NØM = [ nom du module exécutable ] D^C
```

Ici, LINK donne des informations concernant le programme :

- erreurs éventuelles de link

- références externes etc....

- adresse de début d'éxécution du programme, (fonction de la commande < END Etiq > de l'assemblage).

Il suffit de faire !GØ pour lancer le programme éxécutable ; ou de le rappeler par !CALL dans le cas où on lui a donné un nom au link en réponse à la question NØM = .

Exemple : On a créé par assemblage (ASSYS) à partir du fichier source SI ESSAI, un module objet contenu dans le fichier BØ ESSAI. Pour obtenir le module éxécutable, on procède ainsi :

```
?!CALL D^C
   ≀
   ≀
MØDE = Q
NØM = LINK D^C          → appel de LINK
?!GØ D^C                → lancement de LINK
FICH = BØ  ESSAI D^C    → un seul module objet
FICH = D^C
NØM = ESSAI D^C         → le module exécutable est stocké en bibliothèque
RUN '0012
?!GØ D^C                → lancement du programme ESSAI. Celui-ci peut aussi
                          être appelé par !CALL.
```

D U M P

1 FONCTION

DUMP permet de sauvegarder des informations et/ou de supprimer des informations :

- *Sauvegarde.*

On peut sauvegarder :

- des items
- des fichiers

} par parcours de l'arbre des noms, en pas à pas ou automatiquement.

- de l'espace disque

sur

- imprimante ou organe de sortie (imprimante si l'on est en batch, visu sinon) ; dans ce cas, il ne s'agit pas d'une opération de sauvegarde mais d'une simple visualisation de l'information.

- cartes perforées.

- fichier de sauvegarde.

- visu (ou ligne visu) ; cette dernière option permettant notamment d'échanger des informations d'un ordinateur à l'autre au moyen d'une ligne visu.

- disque utilisateur (DKU) : sur SOLAR seulement.

Les informations (items, fichiers, espace disque) sauvegardées par DUMP sur cartes ou fichier ou DKU ou ligne visu, sont récupérables ensuite en utilisant le programme REST (Restauration) décrit dans la suite de ce manuel.

- *Suppression.*

On peut supprimer des items et/ou des fichiers, par parcours de l'arbre des noms, en pas à pas ou automatiquement. En présence d'un nom, DUMP est capable de discerner s'il s'agit d'un item ou d'un fichier et donc de faire les opérations adéquates.

2 UTILISATION.

Les exemples donnés au §3 illustrent l'utilisation de DUMP. Nous donnons ici des précisions sur les questions posées et les réponses à donner.

■ *STANDARD ?*

Répondre Ø ou N (oui/non). En général, on répond oui. Si l'on répond non, DUMP posera par la suite des questions supplémentaires permettant :

- si DUMP sur imprimante ou organe de sortie, de ne lister qu'une partie des items, ou des blocs de fichiers (voir ci-après : options non-standard).

- si DUMP sur cartes, de choisir le nombre de cartes à puncher entre chaque pause pour rechargement (voir ci-après : options non-standard).

- si l'on sauvegarde de l'espace disque, de patcher au passage les blocs lus, et donc de les sauvegarder modifiés (voir ci-après : options non-standard).

■ *DUMP/SUPPRESSION/BØF (D/S/B) ?*

Répondre D, S ou B.

- D : Dump. On ne pourra faire que du dump par la suite, et donc pas de suppressions d'items et/ou de fichiers.

- S : Suppressions. On ne pourra faire que des suppressions d'items et/ou de fichiers ; et donc pas de sauvegardes.

- B : Bof. On pourra faire les deux. A noter que dans ce mode, on n'a pas la possibilité de parcours automatique de l'arbre des noms : on ne pourra travailler qu'en pas à pas.

■ *DK/PARC/FIN ?*

Répondre D, P ou F.

- D : Disque. Signifie que l'on veut faire de la sauvegarde d'espace disque (cette réponse est refusée si l'on est en mode suppression). DUMP demande ensuite : quel disque (DK2 ou DK3), quel quanta utiliser (en DK2 SOLAR seulement, puisque sur DK3 on a toujours quanta = 1, et sur T1600, quanta vaut toujours 1), l'adresse secteur à partir de laquelle on veut sauvegarder et le nombre de secteurs à sauvegarder. Se reporter aux exemples. Les adresses secteur et nombre de secteurs doivent être servies obligatoirement sur 4 chiffres hexadécimaux.

- P : Parcours. Signifie que l'on veut parcourir l'arbre des noms. DUMP propose la pas à pas (sauf si l'on est en mode BØF auquel cas le pas à pas est systématique puisque DUMP ne peut pas savoir lorsqu'il trouve un nom s'il doit faire sur ce nom un DUMP ou une suppression). Puis, DUMP demande la racine des noms à parcourir dans l'arbre. Les noms sont affichés au fur et à mesure qu'ils sont rencontrés ; si l'on est en pas à pas DUMP demande pour chacun ce qu'il doit faire : les réponses reconnues sont les suivantes :

- en mode D (dump)

nom ? { Ø Oui : dump ; N Non : ne rien faire ; D^C ou Return ou 1 Alt-Mode = équivalent à Non. ; F Fin de parcours sur cette racine.

- en mode S (suppression)

nom ? { - supprimer (le fichier ou l'item) ; + ne pas supprimer ; D^C ou Return ou 1 Alt-Mode = équivalent au +. ; F Fin de parcours sur cette racine.

- en mode B (Bof), toutes les réponses ci-dessus sont admises.

- F : Fin. Fin d'utilisation de DUMP. Dump termine les opérations en cours s'il en reste (par exemple perforation de cartes, ou des opérations d'écriture sur fichier de sauvegarde ou DKU ou ligne visu, en fonction du support de dump).

■ *SUR IMP, ØUTPUT, CARTES, FICH, VISU, DKU (I/Ø/C/F/V/D) ?*

Cette question est posée par DUMP dès que l'on demande effectivement un dump soit pour un nom de l'arbre des noms, soit pour de l'espace disque. Elle n'est posée qu'une seule fois.

Réponses possibles :

- I : Imprimante. Le dump sera fait sur LP1.

- Ø : Output. Le dump sera fait sur l'organe de sortie, c'est-à-dire sur la visu utilisée ou sur l'imprimante si l'on travaille en batch.

- C : Cartes. Le dump sera fait sur cartes perforées (sur CU1 ou sur CU2). DUMP, au début de chaque série de n cartes, demande 1 caractère à la visu :

≥ { P ⟶ perforation active ; S ⟶ perforation inhibée

Le nombre n vaut 128 en mode standard. En mode non-standard, DUMP demande à l'utilisateur de préciser ce nombre :

SUR IMP.....(I/Ø/C/F/V/D) ? C

PAQUET = 0100 ⟶ mode non standard, DUMP demande de servir n : le servir sur 4 chiffres hexa. Ici, n = 256.

- F : Fichier. DUMP demande le nom du fichier (ce fichier sera créé par DUMP, donc il ne doit pas exister au départ), et, sur SOLAR seulement, le quanta de ce fichier de sauvegarde :

SUR IMP.....(I/Ø/C/F/V/D) ? F FICHIER = SVG Ø 1 D^{C}

QFS = { 1 ⟶ quanta du fichier de sauvegarde = 1 ; 3 ⟶ quanta du fichier de sauvegarde = 3

Cette question n'est posée que sur SOLAR, puisqu'en T1600, on a toujours Quanta = 1.

- V : Visu. DUMP demande sur quelle visu :

SUR IMP.....(I/Ø/C/F/C/D) ? V VIα

Il suffit de donner le numéro de la ligne visu à utiliser. Ce type de dump suppose que se déroule en même temps le programme REST qui va exploiter au fur et à mesure le dump envoyé sur cette ligne visu. Actuellement, ce type de dump est utilisable de T1600 vers SOLAR et de SOLAR vers T1600.

Nota : En dump cartes et visu, sur SOLAR, DUMP pose cette question :

VERS T ØU S ? { T ⟶ vers T1600 ; S ⟶ vers SOLAR

Ce qui signifie :

- vers T : DUMP exploitera les fichiers à sauvegarder en quanta = 1 (128 mots par bloc).

- vers S : DUMP exploitera les fichiers à sauvegarder en quanta = 3.

Par exemple, si l'on veut puncher un fichier sur SOLAR en utilisant DUMP, dans le but d'utiliser les cartes sur T1600 par le programme REST, il faudra le préciser ; les fichiers seront considérés comme ayant un quanta = 1, et donc seront rechargeables sur T1600.

- D : DKU. En SOLAR seulement. Le support de dump sera une zone du disque utilisateur. DUMP demande de préciser les adresses secteur de début et de fin sur DKU comme ceci :

SUR IMP.....(I/Ø/C/F/V/D) ? D

AS DEB = 9 A 0 0
AS FIN = C 0 0 0 } ⟶ réponse sur 4 chiffres hexadécimaux.

Ces adresses sont validées par DUMP qui s'assure qu'elles sont comprises entre deux bornes définissant l'espace utilisable sur DKU. Actuellement, l'espace disponible sur DKU est compris entre les adresses secteur '8000 et 'F9FF.

Le quanta utilisé sur DKU est toujours 3.

Si tout le dump tient sur l'espace défini par l'utilisateur, le message suivant est envoyé en fin de dump :

{
DK, PARC, FIN ? F

DERNIER BLØC UTILISE = D856 (par exemple)

?

Si tout le dump ne tient pas sur l'espace défini par l'utilisateur, DUMP émet le message suivant :

FIN DE ZØNE DKU

?

et revient au CCi. Dans ce dernier cas, ce qui a été sauvegardé est utilisable par REST, mais il est possible que le dernier fichier à restaurer soit incomplet. La restauration se terminera par un message d'erreur.

■ *Options non standard.*

En mode non standard :

- DUMP sur Imprimante/Output

{
SUR IMP.....(I/Ø/C/F/V/D) ? I (ou Ø)

NBØCT = { D^C ⟶ nombre d'octets normal
0010 ⟶ nombre d'octets = 0010 (par exemple) (réponse sur 4 chiffres hexa).

Ce nombre d'octets est le nombre d'octets maxi à éditer sur imprimante ou organe d'out put ; il en resulte que :

- seuls les n premiers octets des blocs de fichier seront édités

- seuls les n premiers octets des blocs disque seront édités

- seuls les n premiers octets des items (nom + valeur) seront édités.

Voir exemples en fin de ce chapitre.

- Dump sur cartes :

}
SUR IMP.....(I/Ø/C/F/V/D) ? C

PAQUET = 0100 ⟶ réponse sur 4 chiffres hexa. Ici n = 256.

C'est le nombre de cartes à perforer entre chaque pause. Chaque pause permettant d'inhiber ou pas la perforation effective. Ceci est utile pour "rattraper" un dump cartes qui s'est mal passé par suite d'un défaut sur le perforateur.

- Dump d'espace disque :

{
DK, PARC, FIN ? D

DK 2

AS 2A04

NS 0002

AS '2A04 ADR BUFFER '0A93

? ! DEBUG D^{C}

? @, 0A93 D^{C}

? + x D^{C}

? M, xxxx D^{C}

? ! GØ D^{C}

AS '2A05 ADR BUFFER '0A93

? etc...

Comme on le voit sur cet exemple, en mode non standard, DUMP propose de modifier par !DEBUG chaque secteur disque à sauvegarder. Ce sera le secteur modifié qui sera sauvegardé. Si l'on ne veut pas modifier un secteur, il suffit de faire !GØ immédiatement.

3 EXEMPLES D'UTILISATION.

En pages suivantes.

ADDITIF DUMP 1/8/79 (SØLAR SEULEMENT)

1) DKU peut être sauvegardé

Lorsqu'on fait de la sauvegarde à partir d'espace disque la réponse à la question "DK" a la signification suivante :

DK
- 1 ⟶ c'est DKU (disque amovible utilisateur)
- 2 ⟶ c'est DKM (disque amovible système)
- 3 ⟶ c'est DKF (disque à têtes fixes)

2) DUMP sur DKU

Lorsqu'on demande un dump sur DKU, DUMP pose la questtion suivante :

MØDE CØNTINU ?
- Ø ⟶ c'est le mode déjà connu
- N ⟶ c'est le mode dit "multiple".

Dans ce mode multiple, DUMP "ouvre" un nouveau dump pour chaque entité à sauvegarder, une entité étant un fichier, un item ou un espace dique. Supposons qu'on fasse un dump sur DKU en mode multiple de la racine "SI" (fichiers sources) : tout se passe comme si on avait fait autant de dumps qu'il y a de noms ayant cette racine. Cela est fait automatiquement par DUMP en mode multiple. Ainsi, on pourra récupérer par REST tous ces dumps soit séparément (voir REST à partir de DKU mode continu), soit globalement (voir REST à partir de DKU mode multiple).

Exemple d'utilisation

Sauvegarde quotidienne sur DKU en mode multiple de tout l'arbre des noms. Le résultat sera un ensemble de dumps exploitables séparément ou globalement, pour récupérer des informations (items ou fichiers) en cas d'incident.

DUMP sur organe de sortie

```
•Q
NOM•DUMP
?'GO
STANDARD?O
-DUMP.SUPPRESSION.BOF (D/S/B) ?D
DK.PARC.FIN ?P
PAS A PAS ?O
RACINE>FI EXEMPLE
FI EXEMPLE 1  ?N
FI EXEMPLE 2  ?O
SUR IMP.OUTPUT.CARTES.FICH.VISU.DKU (I/O/C/F/V/D)?O0
FIC.FI EXEMPLE 2

NOM INTERNE• '00F2

k· N1'0002 N2'0000

0000     00003CFE 3C894649 43484945 52204558 454D504C 452032FE 3CFE3C20 43452046     __<_ <_FI CHIE R EX EMPL E 2_ <_<  CE F
0020     49434849 45522045 53542044 45535449 4E452041 2049D14C 55535452 45522044     ICHI ER E ST D ESTI NE A  I_L USTR ER D
0040     45532045 58454D50 4C455320 44275554 494C4953 4154494F 4E204445 2044554D     ES E XEMP LES  D'UT ILIS ATIO N DE  DUM
0060     50204554 20524553 542EFE3C FEFF0000 00000000 00000000 00000000 00000000     P ET  RES T _< ---- ---- ---- ---- ----
0080     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
00A0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
00C0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
00E0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0100     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0120     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0140     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0160     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0180     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
01A0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
01C0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
01E0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0200     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0220     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0240     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0260     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
0280     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
02A0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
02C0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
02E0     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     ---- ---- ---- ---- ---- ---- ---- ----
FI EXEMPLE 3  ?F
DK.PARC.FIN ?P
PAS A PAS ?O
RACINE>FI
FI EXEMPLE 1  ?
FI EXEMPLE 2  ?
FI EXEMPLE 3  ?
FI FI  ?
DK.PARC.FIN ?F
?
```

```
•Q
NOM•DUMP
?!GO
STANDARD?N
DUMP.SUPPRESSION.BOF (D/S/B) ?D
DK.PARC.FIN ?P
PAS A PAS ?N
RACINE>FI
FI EXEMPLE 1
SUR IMP.OUTPUT.CARTES.FICH.UISU.DKU (I/O/C/F/U/D)?O
NBOCT•00310
FIC FI EXEMPLE 1

NOM INTERNE• '00F4

K• N1'0002 N2'0000

0000    00003CFE 3C894649 43484845 52204558 454D504C 452031FE 3CFE3C20 43452046    __<_ <_FI CHIE R EX EMPL E 1_ <_<  CE F
0020    49434849 45522045 53542044 45535449 4E                                    ICHI ER E ST D ESTI N

FI EXEMPLE 2
FIC FI EXEMPLE 2

NOM INTERNE• '00F2

K: N1'0002 N2'0000

0000    00003CFE 3C894649 43484845 52204558 454D504C 452032FE 3CFE3C20 43452046    __<_ <_FI CHIE R EX EMPL E 2_ <_<  CE F
0020    49434849 45522045 53542044 45535449 4E                                    ICHI ER E ST D ESTI N

FI EXEMPLE 3
FIC FI EXEMPLE 3

NOM INTERNE• '00F1

K  N1'0002 N2'0000

0000    00003CFE 3C894649 43484845 52204558 454D504C 452033FE 3CFE3C20 43452046    __<_ <_FI CHIE R EX EMPL E 3_ <_<  CE F
0020    49434849 45522045 53542044 45535449 4E                                    ICHI ER E ST D ESTI N

FI FI
FIC FI FI

NOM INTERNE• '00F5

K• N1'0002 N2'0000

0000    00003C20 46492031 FE4C4E27 46492045 58454D50 4C452031 27FE3C20 46492032    __<  FI 1° _LN' FI E XEMP LE 1 '_<  FI 2
0020    FE4C4E27 46492045 58454D50 4C452032 27                                    _LN' FI E XEMP LE 2 '

DK.PARC.FIN ?F
?
```

DUMP sur organe de sortie en option non-standard. (nombre d'octets = 49 (31_{16}))

```
*0
NOM*DUMP
?*GO
STANDARD?N
DUMP SUPPRESSION.BOF (O/S/B) ?O
DK.PARC.FIN ?O
DEBUG BUFFERS DK?O
DK?
AS'0010
NS'0001
SUR IMP.OUTPUT.CARTES.FICH.VISU.DKU (I/O/C/F/U/D)?O
NBOCT*002A0
DK?
AS'0010 ADR BUFFER'0A03
?*GO

0010

0000    FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0020    FFFFFFFF FFFFFFFF FFFF

DK.PARC.FIN ?P
PAS A PAS ?O
RACINE>FI
FI EXEMPLE 1  ?O
FIC FI EXEMPLE 1

NOM INTERNE* '00F4

K  N1'0002 N2'0000

0000    00003CFE 3CB94649 43484945 52204558 454D504C 452031FE 3CFE3C20 43452046
0020    49434849 45522045 5354

FI EXEMPLE 2  ?O
FIC FI EXEMPLE 2

NOM INTERNE* '00F2

K* N1'0002 N2'0000

0000    00003CFE 3CB94649 43484945 52204558 454D504C 452032FE 3CFE3C20 43452046
0020    49434849 45522045 5354

FI EXEMPLE 3  ?F
DK.PARC.FIN ?F
?
```

```
----  ----  ----  ----  ----  ----  ----  ----
----  ----  --

__<_  <_FI CHIE R EX EMPL E 1_ <_<  CE F
ICHI ER E ST

__<_  <_FI CHIE R EX EMPL E 2_ <_<  CE F
ICHI ER E ST
```

DUMP sur organe de sortie, option non-standard (nombre d'octets = 42 ($2A_{16}$))

DUMP d'espace disque
en quanta 3 et quanta 1

```
•Q
NOM•DUMP
?!GO
STANDARD?O
DUMP SUPPRESSION.BOF (D/S/B) ?D
DK.PARC.FIN ?D
DK2
QDK•3
AS0003
NS0001
SUR IMP.OUTPUT.CARTES.FICH.VISU.DKU (I/O/C/F/U/D)?O@
DK2

0003

0000    1E020A01 F000ED01 D8C31E02 2BC22904 1230153A 05020807 29882B82 130F12F0
0020    153A0502 08072C90 1E02990D B00D980D D5020502 1E028513 70E22503 55230702
0040    50232984 1E02AB04 17E447EF A4DA1E02 AD0317E3 47EFA3F7 154E0604 155902FA
0060    1A001E02 471F28C5 0102581C 28C60107 28CE010S 1A80471D 582B1B01 1E0246E5
0080    441C07FF 5E1C1000 471E1BF3 31FF1E01 08011507 05FC1B04 20F328C8 01F15FFF
00A0    D2002840 CD001040 47E01100 47F7E000 471ECFF7 0A01CB02 110060E2 2203D102
00C0    20F60A01 D5010ZFA 46E61E02 1AC030FF 2F401AB2 508313FF 47DF6103 50024DEC
00E0    5DEC2BE6 47F759EC 030247FF FF312FF0 1B011FF3 20EC1A80 30FF2F40 1AB21E0D
0100    2BD046DF 4D491E17 1E032BD0 46DF4144 5D3D4D6C 5D6C2BC6 46E7592C 030246EB
0120    1E0D07EF 1B4D2FC0 1B01FFFF 20E50000 00000000 00000000 00000000 00000000
0140    00000000 00000000 03FF0000 00000000 00000000 00000000 00000000 00000000
0160    00000000 00000000 00000000 00000000 00000000 00000061 008C0169 016B0507
0180    40000000 00000030 000E0001 0000004E 00000001 00000001 00000032 00018000
01A0    00000000 000021CB 1E910000 01050001 00000000 00000000 0000A000 20800000
01C0    0000FFFF 03E80000 000E05C0 215513A9 00100SC0 216413FF 00008000 17F71747
01E0    17F01C2E 179118ED 184A14FC 18C71B39 1495147F 18531B5B 1FF31C6E 18661873
0200    189215AC 18D21914 1B4C1598 19741A0E 1C581984 1CCD1A43 1A4C1C73 14C914E0
0220    1B3F1A2D 171E1B58 1D10FFFF 1DCB1C44 18E41BDB 1BFB1C05 1C1215F3 146217C9
0240    15FC1C7E 1C831CAC 15881FFB 1D1A17F9 1CE11CE6 1C491CD4 15D016EB 1C7419C4
0260    14751DE2 1B701BC9 1FFF1DDE 1BE01DD6 149C1D4E 1CC220A7 20AF20B6 20BF20C7
0280    00B320D9 20E220ED 20862093 1E271E39 1E4D1E5D 20292038 1E721E78 1E831E95
02A0    1EB31ECF 9FD59FE4 1EF19FF7 9FC61F12 9F179F29 1F361F3C 1F241FB7 9FC51FEB
02C0    1FD11FD8 1FE11FE5 1FF01FF4 200A201F 1E0E209E 20F8A111 210E201B 211DA16A
02E0    9FE19FC4 A14BA14E A13CA02A A1249FFF A3F3A150 A141A12D A132A137 A1469FF5

DK.PARC.FIN ?D
DK2
QDK•1
AS0003
NS0001
DK2

0003

0000    8D0A4FFF 44C5A044 C5425547 A03F8D0A C3FF4FD4 CBFF57C5 D3F13FE0 30008D0A
0020    41CC41D2 4DC5A027 31873000 000044CF 4ECECFD3 04F304D7 04C00000 00000000
0040    00000081 00380000 FFF30038 06FD0725 1F552002 90000000 0FFF2C16 D0002BC6
0060    67F31BF0 1E02100A 4D2C47F7 5FEB3009 28C00507 B0062841 53F0AD06 900DAD01
0080    50241E15 90134FCB 47FB1000 37FF1E1E 100047E1 57065FC4 17F31104 1E090904
00A0    5DCB0BFF 2F5B02FA 5706DFF3 CC01F000 15100202 D8022982 C901CD01 46EC46EC
00C0    46EC0A01 C 0206F3 11047EE4 19FF46F6 120F2BC3 1E120BFF 2E5B01FD 50251E16
00E0    B00928C0 2800AD09 055CB006 1303225C 500447EF 900DCDC2 DE04D804 D00447DF

DK.PARC.FIN ?F
?
```

```
*DUMP
?!GO
STANDARD?0
DUMP.SUPPRESSION.BOF (D/S/B) ?D
DK.PARC.FIN ?P
PAS A PAS ?N
RACINE>FI
FI EXEMPLE 1
SUR IMP.OUTPUT.CARTES.FICH.VISU.DKU (I/O/C/F/U/D)?C
VERS T OU $ ?S
FI EXEMPLE 2
>S
FI EXEMPLE 3
FI FI
DK.PARC.FIN ?P
PAS A PAS ?0
RACINE>DUMP
DUMP  ?0
DK.PARC.FIN ?P
PAS A PAS ?R
PAS A PAS ?E
PAS A PAS ?S
PAS A PAS ?N
RACINE>REST
REST
DK.PARC.FIN ?P
PAS A PAS ?0
RACINE>EDITS
EDITS  ?0
>S
DK.PARC.FIN ?D
DK3
AS0010
MS0002
DK.PARC.FIN ?F
?
```

DUMP sur cartes

R E S T

1 FONCTION.

REST permet de restaurer des informations (items et/ou fichiers et/ou espace disque) à partir de données sauvegardées par DUMP sur l'un des supports suivants :

- cartes perforées
- fichier de sauvegarde.
- ligne visu (dump et restauration simultanés)
- zone disque utilisateur DKU (SOLAR seulement).

La restauration peut s'accompagner de modifications :

- des noms des items et fichiers à restaurer.
- des emplacements disque et des quantas disque pour la restauration d'espaces disques.

2 UTILISATION.

Les exemples au §3 illustrent l'utilisation de REST. Nous donnons ici des précisions sur les questions posées et les réponses à donner.

■ *RESTAURATION CARTES/FICHIER/VISU/DKU (C/F/V/D) ?*

Répondre C, F, V ou D.

- C : Cartes. Il s'agit d'une restauration à partir de cartes. En SOLAR, REST demande :

DE T ØU S ? { T ⟶ de T1600 ; S ⟶ de SOLAR }

ce qui signifie (voir aussi DUMP : "VERS T ØU S ?") :

- de T : les cartes ont été perforées sur T1600 donc tous les fichiers à restaurer sont en quanta 1. Les restaurer en quanta 1.

- de S : les cartes ont été perforées sur SOLAR, les fichiers à restaurer sont donc en quanta 3. Les restaurer en quanta 3.

- F : Fichier. Restauration à partir d'un fichier de sauvegarde.

REST 2

RESTAURATION CARTES ..... (C/F/V/D) ? F NØM DU FICHIER = SVG ∅ EXEMPLE D^C

AVEC SAUVEGARDE ? { Ø ⟶ conserver le fichier de sauvegarde
N ⟶ détruire ce fichier.

QFR = { 1 ⟶ quanta du fichier de restauration = 1
3 ⟶ quanta du fichier de restauration = 3 (SOLAR seulement)

Avec sauvegarde signifie que le fichier de sauvegarde sera conservé. Sans sauvegarde signifie que le fichier de sauvegarde sera détruit, bloc par bloc, au cours de la restauration. Cela est utile notamment quand on veut récupérer de l'espace disque au fur et à mesure de la restauration quand on a des problèmes de volume.

Le quanta du fichier de restauration ne peut être que 1 en T1600 c'est pourquoi la question "QFR =" n'est posée que sur SOLAR.

- V : Visu. Restauration à partir d'une ligne visu le dump se déroulant simultanément sur un autre ordinateur.

RESTAURATIØN CARTES .....(C/F/V/D) ? V VIα

DE T ØU S ? { T de T1600
S de SOLAR

On précise le numéro de la ligne visu utilisée, puis, en SOLAR seulement, l'ordinateur émetteur :

- T : T1600. Donc tous les fichiers à restaurer sont en quanta 1.

- S : SOLAR. Donc ces fichiers sont en quanta 3.

- D : DKU. En SOLAR seulement. La restauration se fait à partir d'espace disque DKU.

RESTAURATIØN CARTES.....(C/F/V/D) ? D

AS DEB = xxxx ⟶ adresse secteur de début sur DKU
(réponse sur 4 chiffres hexa).

Dans ce type de restauration, il peut se produire une erreur signalée par le message suivant :

ERREUR DANS LA ZONE DKU < TRAPPE >

Ce qui a déjà été restauré est bon, sauf éventuellement le dernier fichier qui peut n'avoir été que partiellement restauré. Cette erreur peut avoir plusieurs causes :

- le dump lui-même s'était mal terminé.

- erreur de lecture sur DKU (secteur en défaut).

- secteur erroné : il a pu être par exemple écrasé par suite d'une erreur de manipulation entre le moment du dump et celui de la restauration.

■ *AVEC CHANGEMENTS ?*

Répondre Ø ou N (oui/non).

Si l'on répond non, la restauration se fera sans modifications, c'est-à-dire que les fichiers et items seront restaurés sous leur nom au dump et que l'espace disque sera restauré au même emplacement qu'au dump.

Si l'on répond oui, REST proposera :

- pour les items et fichiers, de les restaurer sous un nouveau nom :

SI Ø SØURCE Ø 1 FICHIER NØM > { D^C ou Return / nouveau nom D^C

IMAGE Ø 5 ITEM NOM > { D^C ou Return / nouveau nom D^C

etc....

Un nom vide signifie la conservation du nom initial :

- pour l'espace disque, REST proposera de changer : le numéro de disque, l'adresse secteur sur T1600 et SOLAR, et le quanta sur SOLAR seulement. Pour chacune de ces questions, une réponse vide signifie "pas de changement".

Exemple :

DK 3 ⟶ DK au dump

DK 2 ⟶ DK à la restauration

QD = 1 ⟶ Quanta dump

QR = 3 ⟶ Quanta restauration

ASD 0020 ⟶ Adresse secteur dump

ASR 0AC4 ⟶ Adresse secteur restauration

NSD 0004 ⟶ Nombre de secteurs au dump

NSR 0002 ⟶ Nombre de secteurs restauration (calculé par REST)

Les changements demandés ne sont acceptés que s'ils sont possibles (exemple : on ne peut pas demander un quanta 3 sur DK3).

De plus, avant de lancer la restauration, REST pose la question suivante qui permet de contrôler ce qu'on vient de demander :

ØK ? { Ø ⟶ Oui
N ⟶ Non

puis il affiche, en fonction de la réponse :

RESTAURATIØN DK ACTIVE ou

RESTAURATIØN DK INHIBEE.

Nota : Il peut se produire qu'une restauration d'item ou de fichier soit impossible ; REST le signale quand cela se produit.

Cela provient du fait qu'on veut restaurer un item (ou un fichier), alors que le nom de cet item (ou de ce fichier) existe déjà en bibliothèque.

3 EXEMPLES D'UTILISATION.

En pages suivantes.

ADDITIF REST 1/8/79 (SØLAR SEULEMENT)

1) Positionnement automatique sur DKU

Lorsqu'on lance une restauration à partir de DKU, on fournit une adresse secteur de restauration. Trois cas sont alors possibles :

- cette adresse ne correspond à rien et REST envoie un message d'erreur.

- cette adresse est bien celle d'un début de dump, et REST effectue la restauration.

- cette adresse est celle d'un secteur appartenant à un dump mais n'étant pas le premier secteur du dump, et alors REST recherche automatiquement le premier secteur de ce dump et effectue alors la restauration.

Ceci permet de rechercher par exemple un fichier dans un dump "multiple ".

2) Restauration en mode multiple/continu.

Lors d'une restauration à partir de DKU, REST pose la question suivante :

MØDE MULTIPLE ? { Ø ⟶ oui, mode "multiple"
N ⟶ non, mode "continu"

En mode continu, REST agit comme on l'a déjà vu, c'est-à-dire qu'il exploite un seul dump. En mode multiple, il exploite successivement n dumps (ayant été créés par DUMP sur DKU en mode multiple).

```
•DUMP
?!GO
STANDARD?O
DUMP.SUPPRESSION.BOF (D/S'B') ?D
DK.PARC.FIN ?P
PAS A PAS ?N
RACINE)FI
FI EXEMPLE 1
SUR IMP.(UTPUT.CARTES.FICH.VISU.DKU (I/O/C/F/U/D)?F  FICHIER•SUG EXEMPLE
QFS•3
FI EXEMPLE 2
FI EXEMPLE 3
FI FI
DK.PARC.FIN ?D
DK3
ASKU20
MS0004
DK.PARC.FIN ?D
DK3
AS0700
MS0001
DK.PARC.FIN ?P
PAS A PAS ?N
RACINE)DUMP
DUMP
DK.PARC.FIN ?F
?!CALL
 P•PRIVE.
 Q•SYSTEME.
 R•ACN.
 S•CCI
 T•TABULATION
 LOAD AND RUN
MODE•Q
NOM•REST
?!GO
RESTAURATION CARTES.FICHIER.VISU.DKU (C/F/U/D)?F NOM DU FICHIER?SUG EXEMPLE
AVEC SAUVEGARDE?O
QFR•3
AVEC CHANGEMENTS ?O
FI EXEMPLE 1  FICHIER    NOM)  ASSIGNATION EN NEW IMPOSSIBLE
FI EXEMPLE 2  FICHIER    NOM)FI EXEMPLE 2 BIS
FI EXEMPLE 3  FICHIER    NOM)  ASSIGNATION EN NEW IMPOSSIBLE
FI FI  FICHIER    NOM)FI FI BIS
DK3
DK3
QD•1
QR•1
ASD0020
ASR0040
MSD0004
MSR0004  OK?N
RESTAURATION DK INHIBEE
DK3
DK
QD•1
QR•1
ASD0700
ASR
MSD0001
MSR0001  OK?N
RESTAURATION DK INHIBEE
DUMP  ITEM    NOM)DUMPBIS
```

DUMP sur fichier

RESTAURATION à partir de fichier.

```
??
?!CALL
 F=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=REST
?!GO
RESTAURATION CARTES.FICHIER.VISU.DKU (C/F/V/D)?C
DE T OU S ?T
AVEC CHANGEMENTS ?O
T-30   ITEM     NOM>T-30 BIS
T-31   ITEM     NOM>T-31 BIS
T-32   ITEM     NOM>T-32 BIS
T-33   ITEM     NOM>T-33 BIS
T-34   ITEM     NOM>T-34 BIS
T-35   ITEM     NOM>T-35 BIS
T-36   ITEM     NOM>T-36 BIS
T-37   ITEM     NOM>T-37 BIS
T-38   ITEM     NOM>T-38 BIS
T-39   ITEM     NOM>T-39 BIS
T-3A   ITEM     NOM>T-3A BIS
T-3B   ITEM     NOM>T-3B BIS
T-3C   ITEM     NOM>T-3C BIS
T-3D   ITEM     NOM>T-3D BIS
T-3E   ITEM     NOM>T-3E BIS
T-3F   ITEM     NOM>T-3F BIS
T-3G   ITEM     NOM>T-3G BIS
?
```

RESTAURATION à partir de cartes

C A L C U L

1 FONCTION.

Ce programme simule une calculatrice sur une console, il permet de calculer des expressions :

- arithmétiques et logiques
- parenthèsées
- entre constantes ou variables entières entrées en décimal, en héxadécimal, en binaire ou en asci.

2 APPEL/RETOUR.

On appelle CALCUL par !CALL, mode Q ; voir les exemples ci-après.

Les expressions à calculer sont rentrées sous ">".

Pour revenir au CCI, rentrer une expression vide, c'est-à-dire réduite au D^C.

3 UTILISATEUR.

Fournir à CALCUL ceci :

> expression = [variable] D^C

avec :

- *expression* : { opérandes, opérateurs, parenthèses }
- *opérande* : variable ou constante.
- *variable* : nom sur 1 à 32 caractères ; la variable ayant été définie à droite du signe " = " lors d'un calcul précédent. Dans le nom :

  - les caractères déjà utilisés dans la syntaxe (délimiteurs, premiers caractères d'opérateur, signes,...) ne sont pas autorisés.
  - le 1er caractère $\notin [0,9]$.
  - les blancs sont ignorés.
- *constante* : il en existe plusieurs types :

  - constante décimale : 1 à 5 chiffres décimaux, le nombre étant $\geqslant -2^{15}$ et $\leqslant 2^{15}-1$

- constante hexadécimale : délimiteur ' suivi de 1 à 8 chiffres hexadécimaux.

- constante binaire : délimiteur % suivi de 1 à 32 chiffres binaires.

- constante asci : chaîne de 1 à 4 caractères entre guillemets". (dans une constante asci, on peut rentrer un " en le doublant).

Exemples de constantes :

12345	constante décimale
'23ABF700	constante hexadécimale
%10010	constante binaire
"A@ = C"	constantes asci.
"AB" "C"	

Nota : dans une expression, tous les ∅ sont ignorés sauf dans une constante asci.

■ *opérateurs.*

+	addition
-	soustraction
*	multiplication
/	division (quotient entier)
\	reste de division
@	ou logique (ØR)
#	ou exclusif (EØR)
&	et logique (AND)
> A	décalage à droite arithmétique.
> C	" " " circulaire.
> L	" " " logique.
< A	décalage à gauche arithmétique (assimilé à un décalage à gauche logique!)
< C	décalage à gauche circulaire.
< L	" " " logique.

4 UTILISATION DES VARIABLES.

On ne peut référencer une variable que si elle a été précédemment définie. Pour définir une variable, il suffit de la rentrer à droite du signe égal. La variable placée à droite du signe égal reçoit toujours la valeur de l'ex-

pression ; si elle n'existe pas déjà, elle est créée et reçoit cette valeur ; si elle existe déjà, sa valeur est mise à jour.

On peut faire par exemple ceci :

1 + 2 = VAR1 D^C	⟶	VAR1 est créée et reçoit 3.
VAR1 + 4 = VAR2 D^C	⟶	VAR1 inchangée
		VAR2 est créée et reçoit 7.
VAR1 - VAR2 = VAR1 D^C	⟶	VAR1 reçoit - 4
	⟶	VAR2 inchangée

5 EVALUATION D'UNE EXPRESSION.

Une expression est toujours calculée de la gauche vers la droite sachant qu'il n'existe pas de hiérarchie des opérateurs ; pour "hiérarchiser" le calcul d'expression, il faut donc recourir aux parenthèses (jusqu'à 10 niveaux actuellement ; ce nombre étant un paramètre d'assemblage du programme).

6 FORMAT INTERNE D'UNE VALEUR.

Une valeur tient sur 2 mots (32 bits). Mais pour les opérateurs { * / \ }, les deux opérandes doivent avoir une valeur $-2^{15} \leqslant v \leqslant 2^{15} - 1$; de plus, cette limite s'applique aussi aux constantes décimales servies dans une expression. (cf : messages d'erreur).

7 RESULTAT DE CALCUL.

Il est visualisé en asci, hexadécimal, binaire et décimal. Le résultat décimal ayant le format suivant :

[*] [-] nnnnn

Exemples :

230 -3842	résultat décimal ØK
* 12 * -3473	résultat $\notin [-2^{15}, +2^{15} - 1]$; seuls les bits 16-31 du résultat sont convertis et visualisés précédés du caractère * .

8 MESSAGES D'ERREUR.

Une erreur donne lieu à l'émission d'un message d'erreur de la forme : ER nn, nn étant le numéro d'erreur. La signification des numéros d'erreurs est donnée dans le tableau de la page suivante.

MESSAGES D'ERREUR

Valeur de nn	Signification	Observations
$\leqslant 0$	syntaxe.	
20	trop de niveaux de parenthèses	agir sur le paramètre d'assemblage NMNIV. (actuellement, 10 niveaux maxi).
10	expression trop longue.	agir sur le paramètre d'assemblage LMMSG.
21	parenthèse fermante sans ouvrante.	
22	"Pairage" parenthèses	
30	nom de variable trop long. ou constante trop longue (décimale > 5 hexa > 8 binaire > 32 asci > 4)	agir sur le paramètre d'assemblage LMVC (actuellement = 32)
31	syntaxe (constante hexa ou binaire ou asci de longueur nulle).	
33	constante décimale en entrée de valeur $v \notin [-2^{15}, 2^{15}-1]$	
34	nombre de décalage < 0	exemple : A < L B avec B < 0
35	overflow sur addition	
36	overflow sur soustraction	
37	opérande $\notin [-2^{15}, +2^{15}-1]$ pour {* / \ }	
40	variable inconnue	
41	dépassement de capacité de la table des noms de variables.	agir sur le paramètre d'assemblage LTBNV.
42	dépassement de capacité de la table des valeurs de variables.	agir sur le paramètre d'assemblage NMAXV.

3 EXEMPLES D'UTILISATION.

En pages suivantes.

Exemples d'utilisation de CALCUL

```
?!CALL
 P=PRIVE.
 Q=SYSTEME.
 R=ACN.
 S=CCI.
 T=TABULATION.
 LOAD AND RUN
MODE=Q
NOM=CALCUL
?!GO
>1= 0000 0001                    .    .
0000 0000 0000 0000 0000 0000 0000 0001     1
>10= 0000 000A                   .    .
0000 0000 0000 0000 0000 0000 0000 1010     10
>'10= 0000 0010                  .    .
0000 0000 0000 0000 0000 0000 0001 0000     16
>%10= 0000 0002                  .    .
0000 0000 0000 0000 0000 0000 0000 0010     2
>"10"= 0000 3130                 .  10"
0000 0000 0000 0000 0011 0001 0011 0000     12592
>1=UN 0000 0001                  .    .
0000 0000 0000 0000 0000 0000 0000 0001     1
>2=DEUX 0000 0002                .    .
0000 0000 0000 0000 0000 0000 0000 0010     2
>3=TROIS 0000 0003               .    .
0000 0000 0000 0000 0000 0000 0000 0011     3
>UN+DEUX+TROIS= 0000 0006        .    .
0000 0000 0000 0000 0000 0000 0000 0110     6
>UN+(DEUX*TOIS)= ER 40
>UN+(DEUX*TROIS)= 0000 0007      .    .
0000 0000 0000 0000 0000 0000 0000 0111     7
>((((((((((1))))))))))= 0000 0001    .    .
0000 0000 0000 0000 0000 0000 0000 0001     1
>1-UN= 0000 0000                 .    .
0000 0000 0000 0000 0000 0000 0000 0000     0
>
```

```
2+4= 0000 0006      '     '
0000 0000 0000 0000 0000 0000 0000 0110      6
>2-5= FFFF FFFD     '     '
1111 1111 1111 1111 1111 1111 1111 1101      -3
>2/3= 0000 0000     '     '
0000 0000 0000 0000 0000 0000 0000 0000      0
>2\3= 0000 0002     '     '
0000 0000 0000 0000 0000 0000 0000 0010      2
>3/2= 0000 0001     '     '
0000 0000 0000 0000 0000 0000 0000 0001      1
>3\2= 0000 0001     '     '
0000 0000 0000 0000 0000 0000 0000 0001      1
>'FF00 & 'CCCC= 0000 CC00     '     '
0000 0000 0000 0000 1100 1100 0000 0000     *-13312
>'FF00 @ 'CCCC = 0000 FFCC     '     '
0000 0000 0000 0000 1111 1111 1100 1100     *-52
>'FF00 $ 'CCCC= 0000 33CC     '  3  '
0000 0000 0000 0000 0011 0011 1100 1100      13260
>'FF00 >L 4= 0000 0FF0     '     '
0000 0000 0000 0000 0000 1111 1111 0000      4080
>'00FF <L 4= 0000 0FF0     '     '
0000 0000 0000 0000 0000 1111 1111 0000      4080
>16 >A 8= 0000 0000      '     '
0000 0000 0000 0000 0000 0000 0000 0000      0
>16 >C 8= 1000 0000      '     '
0001 0000 0000 0000 0000 0000 0000 0000     *0
>16 <C 8 = 0000 1000     '     '
0000 0000 0000 0000 0001 0000 0000 0000      4096
>(16 <C 8) @ (16 >C 8)= 1000 1000     '     '
0001 0000 0000 0000 0001 0000 0000 0000     *4096
>I>J>K>L>M>N>O>P>Q>R>S>T>U>V>W>X>Y>Z>[>\>]>^>_>`>a>b>c>d>e>f>g>h>i>j>k>l>m>n>o>p>q>r>s>t>u>v>w>x>y>z>{>|>}>~>
```

3eme PARTIE

S M C

SYSTEME MULTIMEDIA

CONVERSATIONNEL

S M C

1 - APPEL DE SMC ET NIVEAUX DE DIALOGUE.

2 - FONCTIONS REALISABLES PAR SMC.

3 - OBJETS MANIPULES ET BIBLIOTHEQUE.

4 - SCHEMA DE SMC.

5 - LES OUTILS DE SMC : COMMANDES ET PROCESSEURS.

6 - SYSTEME VIDEO-GRAPHIQUE : GENERALITES.

7 - LES PROCESSEURS.

1 - APPEL DE SMC ET NIVEAUX DE DIALOGUE.

Comme nous l'avons vu en première partie, à propos du dialogue utilisateur-système, on ne peut accéder à SMC qu'à travers le système d'exploitation CMS. L'appel de SMC se fait en utilisant le processeur de base !GE (cf : CMS, processeurs de base), de la façon suivante :

? !GE D^{C}

SMC VOUS SOUHAITE LA BIENVENUE !!
07/05/79 - 13/46/24

* (ou ! selon l'ACN)

A ce stade, le dialogue avec SMC est commencé.

On dispose sous SMC de deux niveaux de dialogue ou de commandes :

1. Niveau "*" : ce niveau de dialogue n'est utilisable que si l'ACN (account number ou numéro de compte) est ":SYS". Voir la procédure de logon. A ce niveau peuvent être réalisées des opérations qui seront décrites ci-après, mais qui pour la plupart sont réservés aux spécialistes système et n'ont pas d'intérêt pour l'utilisation habituelle de SMC.

2. Niveau "!" : ce niveau de dialogue est utilisable quelque soit l'ACN de l'utilisateur et sous ce niveau sont disponibles toutes les commandes et tous les processeurs de SMC. On aura d'ailleurs coutume dans la suite du manuel à entendre par SMC ce seul niveau de commandes.

La séquence-type d'appel de SMC par un utilisateur a été donnée ci-dessus. Il faut préciser cependant que :

- si l'ACN de l'utilisateur est ":SYS", alors l'appel de SMC entraînera l'affichage de l'astérisque indiquant que l'on se trouve au premier niveau de dialogue. Pour passer au niveau suivant (!), il suffit de faire "G".

```
?   !GE  D^C        → appel de SMC (avec  ACN =:SYS)
*    }
*   G  }            → premier niveau de commandes
!                   → par la commande G on est passé au niveau "!"
```

- si l'ACN de l'utilisateur n'est pas ":SYS", alors l'appel de SMC entraînera directement l'affichage du point d'exclamation indiquant que l'on se trouve au second niveau de commandes.

```
?  !GE  D^C         →  appel de SMC (avec  ACN ≠ :SYS)
!                   →  on se trouve directement au niveau "!"
```

Pour sortir de SMC et revenir au CCI (control cards interpreter ; on emploie indifféremment les expressions : retour au CCI, retour à CMS, retour au niveau "?"), il suffit d'utiliser :

- si l'on est au niveau "*" : la commande F (fin).

- si l'on est au niveau "!" : la commande FF .

```
?  ! GE  D^C        ⟶   appel de SMC
   ⌇
   ⌇
*  F                ⟶   retour au CCI
?  ! GE  D^C        ⟶   appel de SMC
   ⌇
*  G                ⟶   passage au niveau "!"
!  ⌇
   ⌇
   ⌇
   ⌇
!  FF               ⟶   retour au CCI
?
```

2 - FONCTIONS REALISABLES PAS SMC.

Les fonctions réalisables par SMC sont essentiellement :

- la production assistée par ordinateur de messages audiovisuels divers : graphismes, images, animations, séquences audiovisuelles etc (1),

- la diffusion contrôlée de ces messages grâce à la possibilité de décrire des scénarios et, plus généralement, des graphes d'enchaînement qui seront parcourus automatiquement par l'ordinateur, et qui comporteront de la diffusion de séquences magnétoscopes, ou de films digitaux, ainsi que des dialogues utilisateur-machine. L'enseignement assisté par ordinateur est inclus dans ce type d'utilisation.

- la manipulation et la gestion de ces messages audiovisuels par des moyens informatiques (création, archivage, modification, consultation, suppression, etc ...).

---

(1) par l'intermédiaire de procédés multiples (entrée caméra, light pen, programmes graphiques, curseur graphique etc ...).

3 - OBJETS MANIPULES ET BIBLIOTHEQUE.

3.1 - Généralités.

On dispose sous SMC d'une bibliothèque qui contient un ensemble de noms, à chaque nom étant associée une valeur. On emploiera parfois le terme d'"espace des noms et valeurs" ou "catalogue" pour désigner cette bibliothèque.

Par exemple, une image numérique sera stockée en bibliothèque sous un certain nom, et sa valeur sera l'image numérique elle-même.

Il existe plusieurs types de noms et valeurs dans cette bibliothèque que nous énumérons ci-dessous :

1. Images numériques : elles sont créées et gérées par l'utilisateur lui-même, sous SMC en général .

2. Items utilisateur : ce sont des items créés par l'utilisateur au moyen des commandes de SMC. Ces items sont de types variés, le type étant fonction de la valeur (ou contenu) de l'item :

- type T : texte.
- type D : dessin (programme graphique)
- type P : programme en langage machine
- type C : retour au CCI
- type E : remise à blanc écran
- type G : retour à GE
- type I : film digital (ou scénario)
- type M : question à choix multiple
- type R : réponse en français libre
- type S : séquence audiovisuelle
- type V : item vide
- type W : pause.

Ces items utilisateur correspondent à des besoins et à des utilisations particulières qui apparaîtront lorsqu'on étudiera les commandes et les processeurs de SMC. D'ores et déjà signalons que lorsqu'un item est intégré dans un graphe d'enchaînement, son type permet à SMC de savoir par quel processeur il doit être exploité. En dehors de l'utilisation des items dans des graphes, les plus fréquemment utilisés sont les items de type T (texte), D (programme graphique), et I (film digital).

3. Noeuds : ils sont créés et gérés par l'utilisateur au moyen des commandes de SMC. Ils sont utilisés uniquement pour décrire des graphes d'enchaînement dont nous ferons la description ci-après.

4. Fichiers : ils sont créés et gérés par l'utilisateur, soit sous CMS, par programme, soit sous SMC au moyen de certains processeurs de SMC (processeur LP notamment). C'est ce dernier type de fichier qui nous intéresse ici et on verra qu'il s'agit en général de fichiers graphiques, c'est-à-dire contenant un ensemble de coordonnées de segments décrivant un dessin.

5. Autres items : ce sont principalement des informations utilisées par le système lui-même, notamment les processeurs de SMC, ou des items créés par des utilisateurs autrement que par SMC (par exemple des programmes utilisateurs).

3.2 - Structure de la bibliothèque.

La bibliothèque ou espace des noms et valeurs est organisée de manière arborescente. Il s'agit d'un arbre de noms, et chaque feuille de l'arbre est la valeur associée au nom. On trouvera une description détaillée de cette structure en seconde partie de ce manuel : CMS interface utilisateur-système, espace des noms et valeurs.

D'une manière générale, on peut faire la description suivante de l'espace des noms et valeurs.

1 : les items de service et images numériques sont décrits dans l'espace des noms et valeurs par

. leur nom externe

. leur valeur.

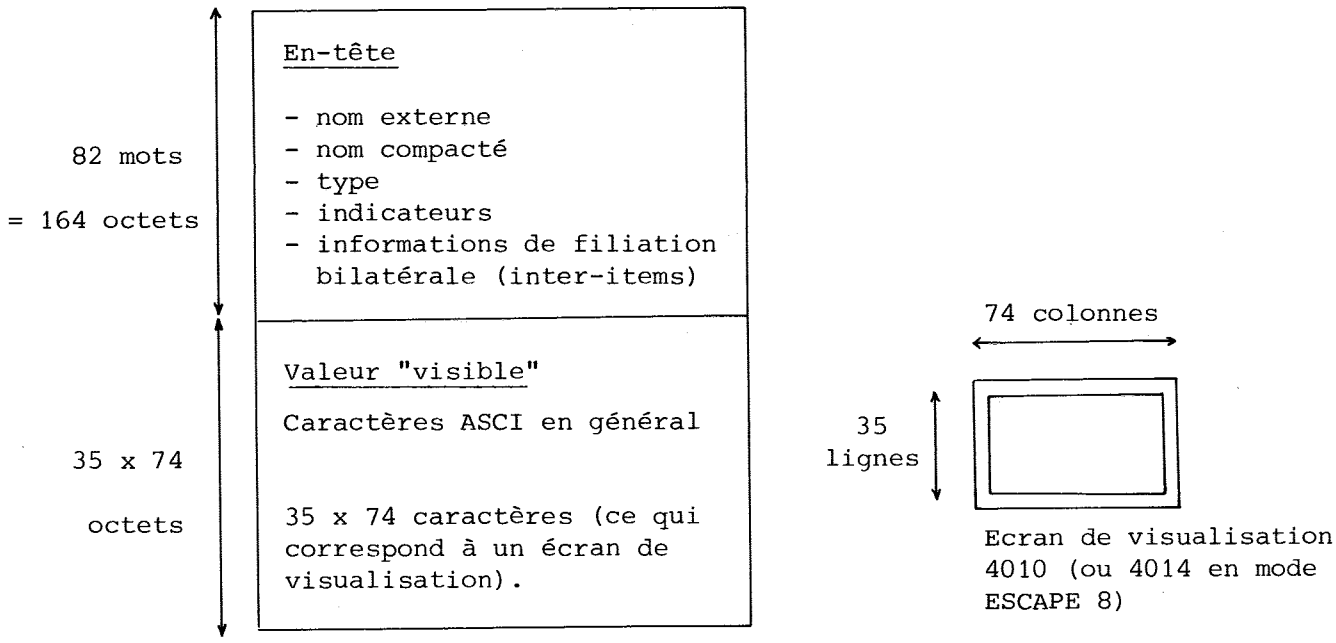
2 : les items utilisateur sont décrits dans l'espace des noms et valeurs par :

. leur nom compacté

. leur valeur contenant :

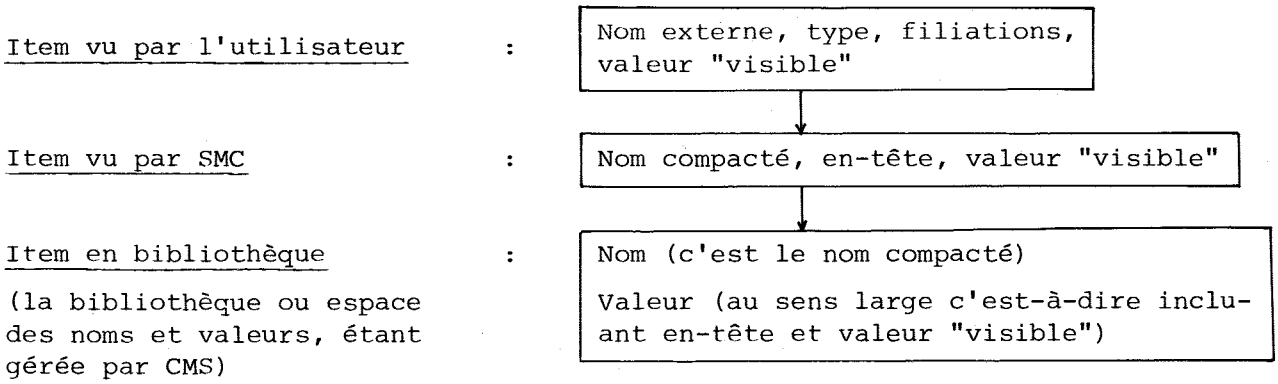
- un en-tête
- la valeur "visible" (celle qu'à donnée l'utilisateur).

Exemple :



L'utilisateur ne connaît que le nom externe de l'item, c'est-à-dire le nom qu'il a donné à cet item en utilisant les commandes de SMC ; son type ; les informations de filiation concernant cet item et enfin la valeur "visible" de l'item.

Les notions d'indicateurs et de nom compacté sont donc internes au système SMC.



On verra que le type de l'item est affecté à l'item soit implicitement, soit explicitement :

- implicitement : l'utilisateur crée à l'aide des commandes de SMC un item dans lequel il écrit un programme graphique en utilisant le processeur éditeur de textes ED : l'item reçoit automatiquement le type T (Texte). S'il exécute ce programme graphique à l'aide de l'un des processeurs graphiques de SMC (par exemple GR), l'item recevra automatiquement le type D (dessin).

- explicitement : l'utilisateur a créé un item et veut lui affecter explicitement un type ; il peut le faire au moyen du processeur SE de SMC. Cela n'est pas possible avec tous les types mais seulement avec les types acceptés par SE (cf ce processeur) ; pour les autres, le type ne pourra être affecté qu'implicitement.

Le nom compacté, notion interne au système, est construit par celui-ci à partir du nom externe. Un nom compacté fait toujours 6 caractères :

- les trois premiers sont ceux du nom externe.

- les trois suivants sont calculés en fonction des caractères suivants du nom externe (donc à partir du quatrième) et de la longueur de ce nom externe.

Il peut se produire que deux noms externes différents donnent le même nom compacté. C'est pourquoi il est conseillé de ne pas donner de racines communes trop longues aux noms d'items pour éviter ce phénomène. Noter que des noms externes n'ayant pas leurs trois premiers caractères identiques garantissent l'absence de synonymie des noms compactés.

Nota : Il existe dans SMC une commande permettant de lister des noms d'items en spécifiant une racine sur 1 à 3 caractères : cette commande (il s'agit de la commande LI comme liste), liste les items de racine spécifiée en affichant leurs noms externes. On verra qu'il est possible aussi de lister des items sur leur nom compacté, ce qui est possible avec le processeur EX de SMC. En d'autres termes, la commande LI permet de lister des items tels que l'utilisateur les a décrits, donc sur leur nom externe, alors que le processeur EX permet de parcourir l'arbre des noms de la bibliothèque telle qu'elle a été construite par le système, et donc à ce niveau c'est le nom compacté qui est à considérer. En général, l'utilisateur n'a d'ailleurs pas à se préoccuper des noms compactés. Toutefois signalons qu'il est possible de demander à SMC de donner le type, le nom compacté, et les informations de filiation concernant un item, en utilisant soit la commande I= de SMC soit le processeur SE.

3 : les noeuds sont décrits dans l'espace des noms et valeurs par

. leur nom compacté

. un ensemble d'informations rappelant l'en-tête d'item, c'est-à-dire notamment : nom compacté, informations de filiation inter-noeuds et noeuds vers items.

Ce que l'on peut dire des noeuds est assez semblable à ce qu'on a dit des items utilisateurs :

- l'utilisateur connaît le nom externe qu'il a donné au noeud, et les informations de filiation relatives à ce noeud puisque c'est lui qui crée les chaînages inter-noeuds et noeuds vers items lorsqu'il construit un graphe (voir ci-après la notion de graphe) ; le tout au moyen des commandes de SMC.

- SMC connaît en plus le nom compacté du noeud qu'il a calculé à partir du nom interne, de la même façon que pour les items.

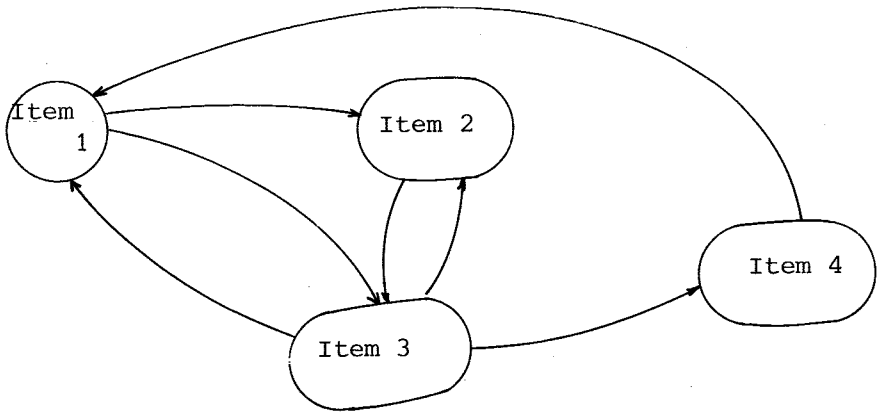
- quant à CMS qui gère l'espace des noms et valeurs, il ne connaît que Nom et Valeur , le nom étant le nom compacté, la valeur étant toutes les informations associées (nom externe, filiations, etc ...), et n'ayant de sens que pour SMC.

De la même façon que pour les items on peut lister un ensemble de noeuds de racine commune en utilisant une commande de SMC : la commande LN qui liste les noms externes ; on peut utiliser la commande N= pour demander des informations sur un noeud spécifié par son nom externe (nom compacté et filiations), et enfin, parcourir l'arbre des noms (noms compactés) au moyen du processeur EX.

3.3 - Les graphes d'enchaînement.

SMC offre la possibilité de décrire des graphes d'enchaînement. Un graphe d'enchaînement est comme tout graphe un ensemble d'arcs et de sommets (ou noeuds). Il faut cependant distinguer la structure logique et la structure physique d'un graphe décrit sous SMC :

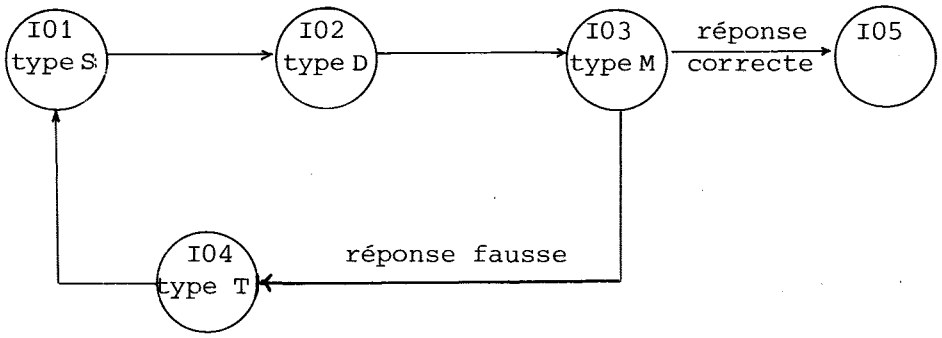
- *Structure logique :*



Chaque item de ce graphe sera un item utilisateur, et par conséquent il aura un type appartenant à l'ensemble de types énumérés précédemment (T,D,P,C,E,G,I,M,R,S,V,W) ; le graphe décrit par l'utilisateur est appelé à être parcouru automatiquement par SMC, le lancement du parcours se faisant en utilisant le processeur GØ de SMC.

Au cours de l'exploitation du graphe, SMC va exploiter l'item courant : en fonction de son type, il va utiliser un processeur spécialisé qui va exploiter l'item, et en fonction du résultat de cette exploitation et des filiations existantes, SMC va choisir l'item suivant, etc ...

Exemple : enseignement assisté par ordinateur.



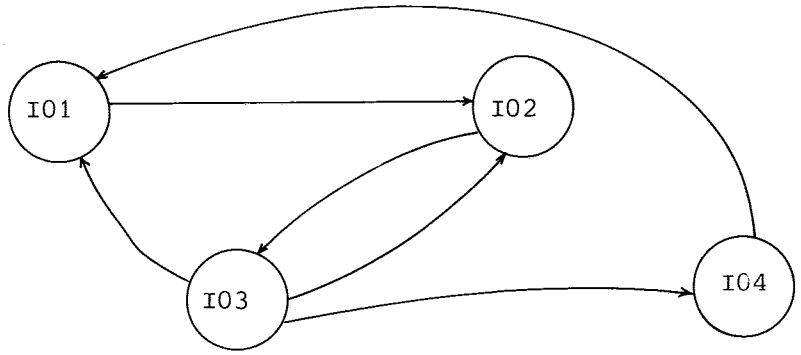
SMC rencontre l'item IO1 de type S : séquence audio-visuelle : il appelle alors le processeur VI qui va rechercher et diffuser cette séquence sur magnétoscope. Ensuite, SMC rencontre IO2, de type D : il appelle le processeur adéquat (dans ce cas il y en a plusieurs possibles, l'utilisateur en a spécifié un lors de la création de l'item), par exemple l'interpréteur graphique GR qui va exécuter le programme graphique contenu dans l'item IO2. Par exemple, il s'agira d'un dessin, suivi du texte d'une question. Puis, GR ayant terminé, SMC exploite IO3 de type M (question à choix multiples). Une réponse est donc demandée à l'utilisateur (l'élève en l'occurrence). Celui-ci répond et, en fonction de sa réponse, SMC va aller soit vers IO4 soit vers IO5 suivant les chaînages définis par l'utilisateur. IO4 sera par exemple un item de type T, donc exploité par le processeur ED (éditeur de textes), qui affichera sur l'écran de visualisation un texte avertissant l'élève que la séquence audiovisuelle va être rediffusée ... etc ...

Etant donnée la variété des items constituant un graphe, les applications possibles sont nombreuses, puisque la quasi totalité des processeurs de SMC peuvent être utilisés dans un graphe.

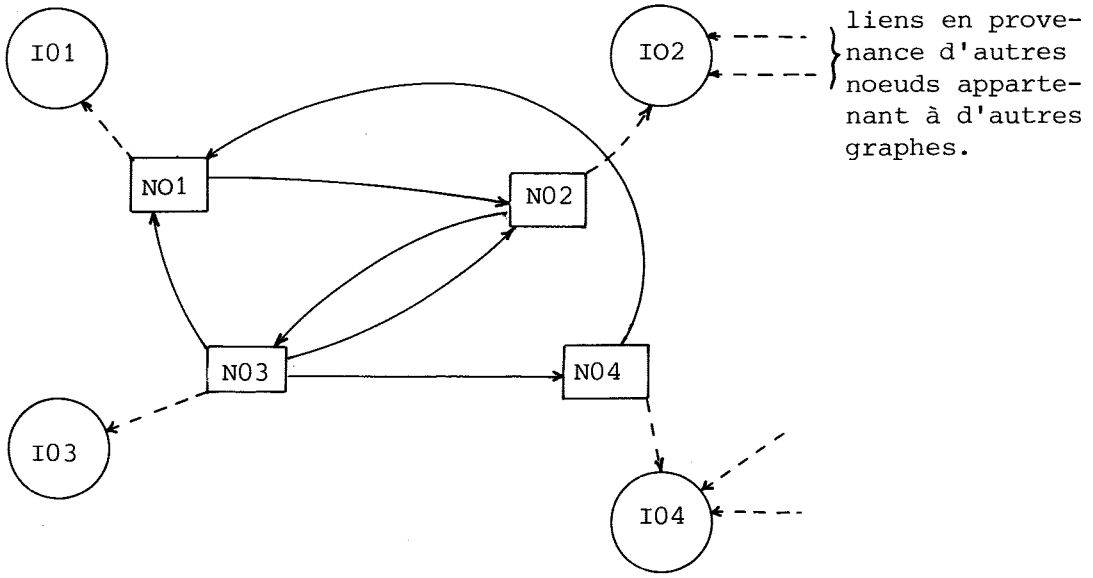
L'exemple ci-dessus fait apparaître la structure logique d'un graphe, qui est une structure d'items. En fait, cette structure logique d'items est supportée physiquement par une structure physique de noeuds.

■ *Structure physique :*

Exemple : Soit la structure logique suivante à réaliser :

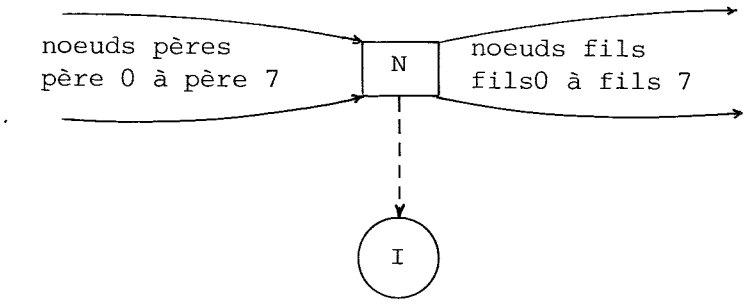


Elle sera représentée au moyen de la structure physique suivante :



Les contraintes à respecter sont les suivantes :

- un noeud ne peut posséder qu'au maximum 1 lien vers un item.
- un item peut recevoir au maximum 16 liens (venant forcément de noeuds).
- un noeud peut avoir :
  - de 0 à 8 noeuds pères.
  - de 0 à 8 noeuds fils.



4 - SCHEMA DE SMC.

4.1 - Espace mémoire utilisateur.

Lorsqu'un utilisateur appelle SMC, un espace mémoire virtuel lui est alloué. Cet espace mémoire peut varier dans le temps en fonction des opérations effectuées :

- si l'utilisateur travaille sur des items (édition de texte par exemple), l'espace mémoire alloué est de 4K mots et contient notamment :

. le noyau du système SMC

. le processeur courant, c'est-à-dire GE quand l'utilisateur utilise les commandes de SMC sous le niveau "!" (cf : les outils de SMC), ou un processeur quelconque de SMC, processeur XX si l'utilisateur l'a appelé.

. deux zones : ITEM1 et ITEM2 qui reçoivent les items courants sur lesquels l'utilisateur travaille.

- si l'utilisateur travaille sur des items et sur des images numériques (processeurs de télévision, light pen, diffusion de scénarios etc...), l'espace mémoire alloué passe à 8K ou 12K mots et contient :

. le noyau du système SMC

. le processeur courant

. les deux zones ITEM1 et ITEM2 qui contiennent :

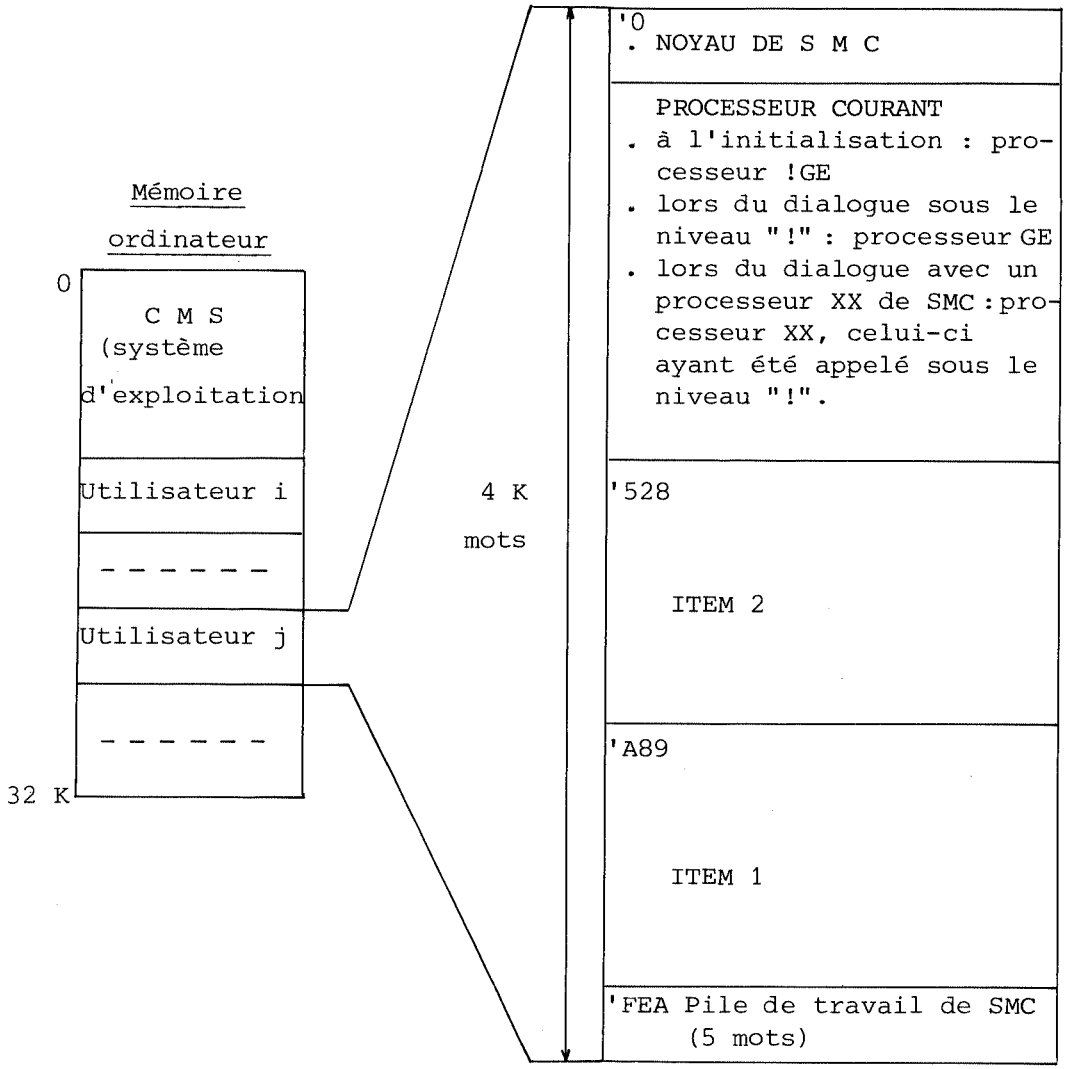
- soit les items courants si le processeur courant en a besoin (processeurs GT, SG, TI par exemple)

- soit autre chose, elles peuvent servir de zone de travail au processeur courant (processeur TV par exemple).

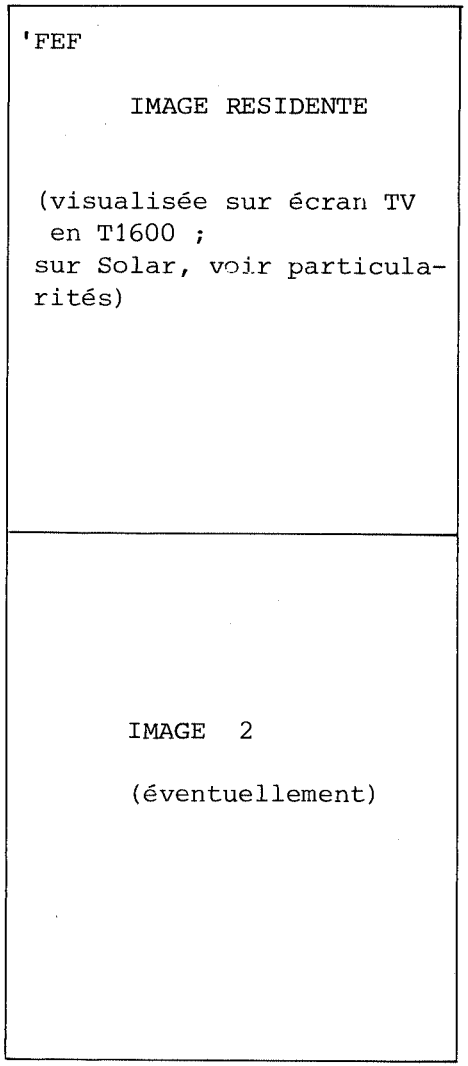
. l'image résidente ; cette image étant visualisée en permanence quand l'utilisateur travaille (cas de tous les processeurs de télévision).

éventuellement une seconde image (processeur TN notamment), auquel cas l'espace mémoire utilisateur, qui est de 8K mots sans cette seconde image, passe à 12K mots.

On trouvera en page suivante le schéma de l'espace mémoire utilisateur.



Extensions pour traitement d'images.



Selon le processeur de télévision courant, Item 1 et Item 2 peuvent être utilisés comme zone de travail

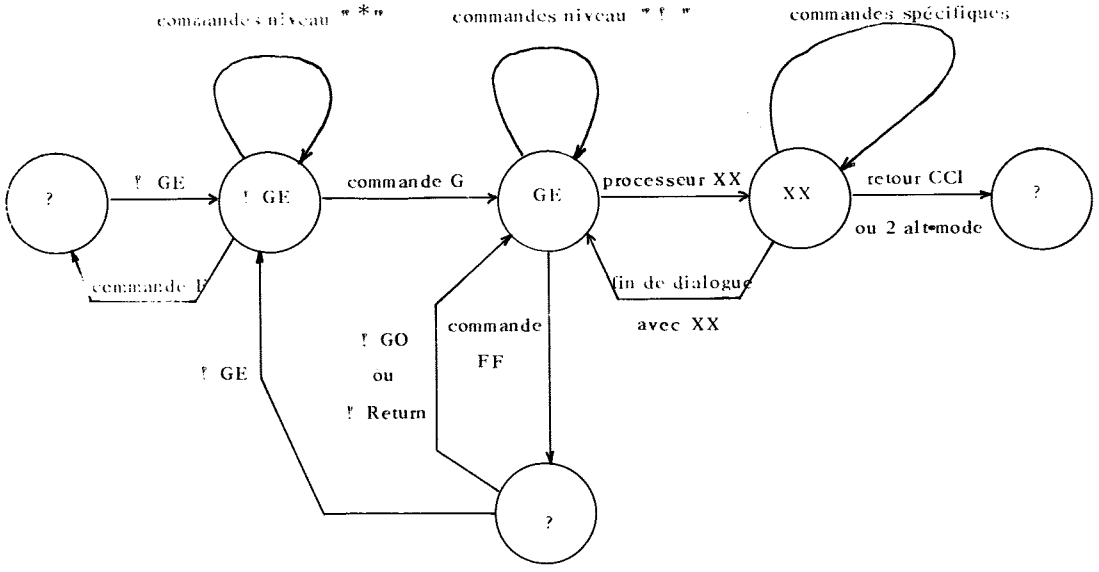
4.2 - Graphe d'occupation de la zone "processeur courant".

Dans la zone "processeur courant" se succèdent en overlay (recouvrement) les différents programmes nécessaires à l'utilisateur.

Lors de l'initialisation (appel de SMC), c'est le processeur de base !GE de CMS qui occupe cette zone, ce processeur ayant été appelé par l'utilisateur sous le niveau "?" (commande !GE D^C). Ce processeur !GE satisfait au commandes rentrées sous le niveau "*" de SMC, en particulier à la commande G qui permet de passer au niveau de commandes "!" de SMC. Sous ce niveau, c'est le processeur GE de SMC qui occupe la zone "processeur courant" et qui satisfait à toutes les commandes rentrées par l'utilisateur. Toute commande fait 2 caractères :

- si les deux caractères correspondent à une commande existante, cette commande est exécutée par GE.

- sinon, GE regarde si ces deux caractères correspondent à un nom de processeur de SMC ; si c'est le cas, GE appelle le processeur ainsi spécifié qui vient lui succéder dans la zone "processeur courant" ; si ce n'est pas le cas (commande non reconnue), GE envoie un message d'erreur et attend une autre commande.



Ce graphe montre qu'en zone "processeur courant" se succèdent :

- le processeur !GE capable de satisfaire aux commandes rentrées sous le niveau "*", dont la commande G qui permet de passer à GE.

- le processeur GE capable de satisfaire aux commandes et aux appels de processeurs, sous le niveau "!" ; dont la commande FF qui permet de revenir à CMS (niveau "?").

- les processeurs de SMC appelés sous GE. Avec chaque processeur se développe un dialogue spécifique. En général, la fin de travail d'un processeur est provoquée par une commande de fin ou 1 alt-mode.

Remarques.

- le niveau de dialogue "*" n'est accessible que si l'ACN de l'utilisateur est :SYS.

- l'ensemble des commandes et des processeurs de SMC est exposé dans le paragraphe suivant : "les outils de SMC".

5 - LES OUTILS DE SMC : COMMANDES ET PROCESSEURS.

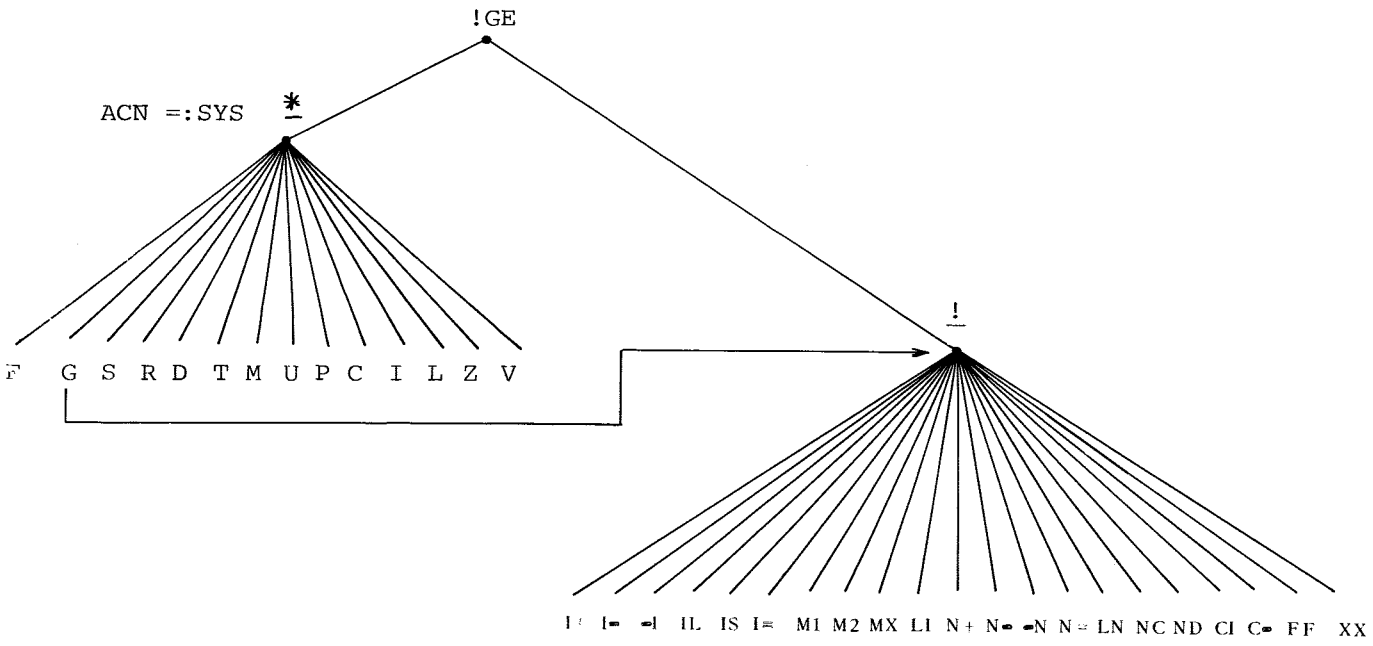
SMC offre deux niveaux de commandes :

- niveau "*", accessible seulement si l'ACN de l'utilisateur est :SYS. Il permet des opérations système qui n'entrent pas dans le cadre habituel d'utilisation de SMC.

- niveau "!", accessible quelque soit l'ACN, et qui donne accès à toutes les commandes et processeurs de SMC :

. commandes : elles apparaissent toutes sur le schéma ci-après et sont détaillées ensuite.

. processeurs : ils sont listés ci-après et chacun d'eux fait l'objet d'une documentation spécifique dans la suite de ce manuel.



Remarques.

1 - Si l'on frappe 1 alt-mode :

- sous le niveau "*" : on est ramené à l'état initial (on réentre dans SMC)

- sous le niveau "!" : on reste au niveau "!" (on réentre dans GE).

2 - Lorsqu'on entre dans SMC, les unités logiques 3 à 9 sont désassignées d'office par !GE et assignées à l'espace des noms et valeurs. On ne dispose donc plus que des unités logiques A et B. On verra qu'en général, on utilisera l'unité logique A pour rentrer des informations (par exemple des items stockés sur cartes perforées), et l'unité logique B pour sortir des informations (par exemple édition sur imprimante ou perforation de cartes).

3 - Lors d'un parcours de graphe, lorsque l'on interrompt l'exécution d'un processeur graphique ou du processeur ED par 1 alt-mode, le parcours continue par le fils 2 du noeud courant. Pour ces mêmes processeurs, si leur exécution n'est pas interrompue et se termine normalement, le parcours continuera par le fils 0 du noeud courant.

5.1 - Niveau de commandes "*" (ACN = :SYS)

- *Commande F : retour au CCI ("?")*

- *Commande G : passer au niveau de commandes "!"*

- *Commande S : sauvegarde sur cartes de l'un quelconque des processeurs de SMC*. Cette commande n'est utilisable que si l'on a préalablement assigné le perforateur de cartes à l'unité logique B (!ASSIGN Ø B = CU1 D^C). Le système demande le nom du processeur à sauvegarder. Le lui donner (un nom de processeur fait toujours 2 caractères). Le paquet de cartes obtenu comportera :

- une carte R en début
- les cartes de sauvegarde du processeur
- une carte Return en fin

Ce qui permet par la suite de restaurer le processeur en batch. Les cartes de sauvegarde du processeur sont numérotées modulo 64 en ligne 12 comme ceci :

- carte 1 : punch en colonne 1 ligne 12
- carte 2 : " " " 2 " 12
- carte 64 : " " " 64 " 12
- carte 65 : " " " 1 " 12 etc ...

Exemple :

```
* F              → retour au CCI pour assigner B
? !ASSIGN ⌀ B = CU1 D^C
? !GØ D^C        → retour à !GE
* S              → commande S : sauvegarde processeur
NØM > XX         → spécification du nom du processeur ; la perforation est
                   lancée
*                → fin de l'opération de sauvegarde.
```

■ *Commande R : restauration à partir de cartes de l'un quelconque des processeurs de SMC*. C'est l'opération inverse de S. Cette commande n'est utilisable que si l'on a au préalable assigné l'unité logique A au lecteur de cartes (!ASSIGN ⌀ A = CR1 D^C). Le système exploite les cartes de sauvegarde du processeur qui contiennent son nom et sa valeur. Il propose de changer son nom, on peut alors lui répondre par :

1 = un nouveau nom : le processeur sera donc restauré avec ce nouveau nom.

2 = Return ; le nom du processeur sera inchangé.

3 = D^C : provoque un retour au CCI ("?") : on peut alors patcher le processeur sous !DEBUG et revenir par !GØ. Le processeur est alors restauré modifié (sous !DEBUG on a pu modifier son nom et/ou sa valeur).

Exemple :

```
* F              → retour au CCI pour assigner A.
? !ASSIGN ⌀ A = CR1 D^C
? !GØ D^C
* R              → commande R : restauration.
NØM > XX         → le système affiche le nom du processeur
NØM > D^C        → et demande une réponse
? !DEBUG D^C     → on veut patcher le processeur
? séquence de patch
      ⌇
? ; D^C          → fin de patch
? !GØ D^C        → retour à !GE
NØM > YY         → le système demande à nouveau une réponse. Cette fois, on dé-
                   cide que le processeur soit restauré sous un nouveau nom : YY
*                → fin de restauration
```

■ *Commande D : Disque.* Permet d'atteindre les disques du système en absolu. Ce sont les disques DK1, DK2 et DK3. Le système demande :

- sur quel disque on veut travailler
- à quelle adresse secteur (adresse physique)
- quelle opération on veut faire : il y en a 4 possibles : P (punch), R (Read), W (Write), S (restauration d'espace disque à partir de cartes).

1 - P : Punch n K mots (!ASSIGN Ø B = CU1 D^C requis)

Le système demande combien de K mots on veut puncher. Répondre par un nombre hexadécimal de 2 chiffres. P permet de puncher n'importe quelle zone disque, et notamment les processeurs de base de CMS qui sont implantés sur DK2 et DK3 en T1600 et sur DK3 seulement en Solar, aux adresses suivantes :

! DF	sur	1 K mots en	'OF90	sur T1600	et 'OF20	en SOLAR
! I	sur	1 K mots en	'OF68	sur T1600	et 'OEF8	en SOLAR
! GE	sur	4 K mots en	'OF70	sur T1600	et 'OF00	en SOLAR
! CALL	sur	1 K mots en	'OF20	sur T1600	et 'OEB0	en SOLAR

Exemple : punch du processeur de base !CALL

* F	→ retour au CCI pour assignation
? !ASSIGN Ø B = CU1 D^C	
? !GØ D^C	→ retour à !GE
* D	→ commande D : accès disques en absolu
DK 3	→ disque DK 3
AS 0F20	→ adresse secteur 0F20
> P	→ punch
KM 01	→ 1 K mots
*	→ fin de punch

2/3 - R/W (Read/Write).

La taille de l'échange est implicitement de 16 secteurs. (16 secteurs de 128 mots = 2K mots). Elle peut être fixée à 1 secteur un utilisant la commande T décrite ci-après. Read et Write utilisées conjointement permettent notamment de patcher des secteurs disque comme le montre l'exemple suivant.

Exemple :

```
* D
DK 3                    ⟶  disque DK3
AS OOA5                 ⟶  adresse secteur
> R                     ⟶  read (implicitement : 16 secteurs = 2 k mots)
* F                     ⟶  retour au CCI pour debug.
? ! DEBUG D^C
? @, 528 D^C            ⟶  la zone disque est toujours chargée en '528. Donc on
                           est ici au mot O du 1er secteur échangé.
? + n D^C               ⟶  positionnement sur le mot à patcher.
? M, valeur D^C         ⟶  le mot est modifié en mémoire
etc...
? ; D^C                 ⟶  fin de séquence de patch
? ! GØ D^C              ⟶  retour à !GE pour faire le write.
* D
DK 3
AS D^C                  ⟶  réponse vide : même  adresse secteur
*  W                    ⟶  write espace  disque  modifié.
*
```

4 - S : restauration à partir de cartes

Les cartes ont été punchées par P. Pour cette restauration, l'assignation de l'unité logique A au lecteur de cartes CR1 n'est pas obligatoire. Pendant toute la durée de la restauration dique, le système est bloqué.

Nota : la première carte du paquet est inutile ; il faut donc l'enlever ; on constatera que la seconde carte porte le numéro 1.

Exemple :

```
* D
DK 3                    ⟶  dique DK3
AS 0020                 ⟶  adresse secteur
> S                     ⟶  restauration à partir de cartes de n K mots
*
```

■ *Commande T : Taille des échanges disques.*

La taille est implicitement 16 secteurs. T permet de les fixer à 1 secteur ou à 16 secteurs.

Exemple :

```
* T
1 SECTEUR ? N          ⟶ réponse = Ø ou N (oui/non)
ALØRS 16
*
```

■ *Commande M : Modification (ou suppression) d'un processeur de SMC.*

Les processeurs de SMC appartiennent à l'espace des noms et valeurs. Par la commande M, on peut charger en mémoire un processeur XX, et ensuite utiliser l'une des 3 possibilités suivantes :

- le stocker sous le même nom,
- le stocker sous un nom différent,
- le patcher,

exactement comme on pouvait le faire avec la commande R décrite précedemment.

Exemple 1

```
* M                          ⟶ Modification
NOM > XX                    ⟶ du processeur XX, celui-ci est chargé en mémoire
                               et supprimé en bibliothéque.

NOM > XX
NOM > D^C                   ⟶ on veut le patcher
? ! DEBUG D^C
? @ , 528 D^C          )
? - B D^C              )    pour patcher le mot n du processeur, il
? + n D^C              )    faut se positionner en '528 - 'B + 'n
? M, valeur D^C        )    sous !DEBUG
? ; D^C                     ⟶ fin de la séquence de  patch
? ! GØ D^C                  ⟶ retour à !GE. Si l'on avait fait, au lieu de
                               !GØ, !GE D^C ou ! Return, alors le processeur
                               XX aurait été perdu (cf : exemple suivant)
NØM >                       ⟶ à nouveau trois réponses possibles.
etc...
```

Exemple 2 :

suppression du processeu XX : trois séquence équivalentes.

```
* M                    * M                    * M
NØM > XX               NØM > XX               NØM > XX
NØM > XX               NØM > XX               NØM > XX
NØM > D^C              NØM > D^C              NØM > Alt. Mode
? ! GE D^C             ? ! Return             *
*                      *
```

■ *Commande U : chargement d'un processeur de SMC.*

Permet de charger en mémoire un processeur XX de SMC désigné par son nom. Le processeur est chargé en '528 - 'B et on peut donc le patcher par !DEBUG exactement comme on l'a vu dans les exemples précédents. Pour le replacer, éventuellement modifié, en bibliothèque, on utilisera la commande P, décrite au paragraphe suivant.

■ *Commande P : inverse de U.*

Permet de stocher en bibliothèque un processeur de SMC préalablement chargé en mémoire, en modifiant éventuellement son nom, et en l'ayant éventuellement patché par ! DEBUG.

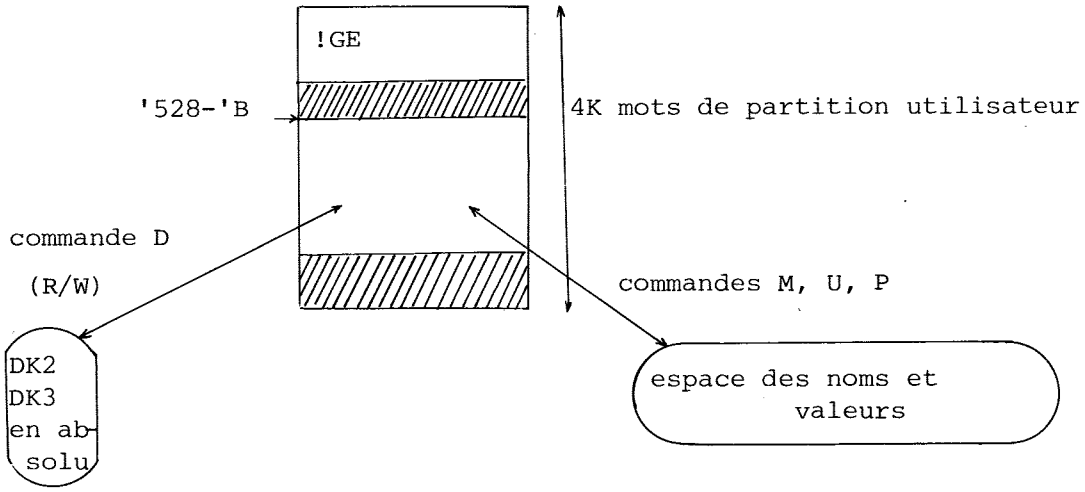
Exemple :

```
* U
NØM > XX                  →  chargement du processeur XX
* F                        →  retour au CCI pour patch
? !DEBUG D^C
? @, 528 D^C
? - B D^C                  }  patch du mot n du processeur, placé en '528 - 'B
? + n D^C
? M, valeur D^C
? ; D^C                    →  fin de la séquence de patch
? !GØD^C                   →  retour à !GE
* P                        →  demande de copie en bibliothèque
NØM > XX
NØM > YY                   →  ici, on a 3 réponses possibles (YY/Return/D^C)
*
```

■ Remarques sur les commandes D (Read/Write) M, U et P

Avec la commande D,on travaille sur de l'espace disque absolu, désigné comme on l'a vu, en spécifiant le numéro de disque. C'est de cette façon qu'entre autres, on peut accéder, pour les sauvegarder, les restaurer ou les modifier, aux processeur de base du système CMS (!I, !CALL, !GE, !DF notamment). Ces proceseurs de base sont : sur T1600 dupliqués sur DK2 et DK3 où ils sont implantés aux mêmes adresses, et sur Solar implantés uniquement sur DK3.

En revanche, avec les commandes M, U et P, on travaille sur la bibliothèque (ou espace des noms et valeurs), qui contient, entre autres, les processeurs de SMC. Ceux-si sont donc accessibles par leur nom.



■ *Commande C* : Copie de l'état du système CMS (T1600 seulement)

Cette copie de l'état du système se fait en mémoire dans la zone '528 -'B de l'espace utilisateur. Lorsqu'on demande la copie, le système demande : "ØK ?" Trois réponses sont alors possibles :

1. N : Non. pas de copie
2. Ø : Oui. Copie en '528 - 'B
3. A : Automatique. Il y aura copie périodique avec écriture sur disque en zone scratch utilisateur. La périodicité est fonction de la charge du sytème. La copie est interruptible par alt-mode.

Nota : on ne peut demander de copie automatique que si le SYSID de l'utilisateur est 0 (voir CMS).

■ *Remarque* : copie avec Oui suivie de Disque Write permet de sauvegarder sur disque l'état du système à un instant donné.

■ *Commande I : restauration de l'état du système CMS (T1600 seulement).*

Cette opération est l'inverse de la copie. L'état du système est réinitialisé à partir de la zone '528 - 'B de l'espace utilisateur. Comme pour la copie, le système demande "ØK ?" et trois réponses sont alors possibles :

1. N : Non. Pas de restauration
2. Ø : Oui. Restauration à parir de la zone '528 - 'B
3. A : Automatique. Restauration périodique à partir de la zone scratch utilisateur. Comme pour la copie, il faut que le SYSID de l'utilisateur soit O.

■ *Remarque* : restauration (I avec Oui) précédée de Disque Read permet de restaurer l'état du système tel qu'il était à un instant donné, cet état ayant été sauvegardé précédemment par Copie Oui suivie de Disque Write.

■ *Commande L : liste état et occupation des magnétoscopes*

Les neuf magnétoscopes raccordés au système sont MG00 à MG08. L permet de lister pour chacun :

- son état : en service ou hors service
- son occupation : occupé (bande en place) ou non occupé (pas de bande).

■ *Commande Z : réinitialisation de la liste des magnétoscopes*

Tous les magnétoscopes (MG00 à MG08) sont mis en service et non occupés

■ *Commande V : modification de la liste des magnétoscopes*

Permet de faire sur un magnétoscope l'une des opérations suivantes : mise en service; mise hors service; démonter la bande.

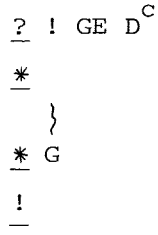
Exemple :

```
* V
> 01 +          → mise en service de MGO1 (s'il n'y était pas)
* V
> 04 -          → mise hors service de MGO4
* V
> 04 D          → démonter la bande sur MG04
*
```

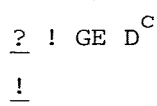
5 - 2 Niveau de commande "!" (ACN quelconque)

On passe à ce niveau de commande de deux façons :

- on est sous l'ACN : SYS et on a fait !GE... etc... puis G :



- on est sous un ACN ≠ : SYS et on a fait seulement !GE :



On dispose sous ce niveau : des commandes de SMC (permettant de travailler sur des items utilisateurs et sur des noeuds), et des processeurs de SMC. Les commandes sont celles citées sur le schéma figurant en tête du paragraphe 5 "outils et processeurs de SMC". Elles sont détaillées ci-après. Toutes ces commandes sont interruptibles par alt-mode ainsi que la quasi-totalité des processeurs. Toute commande fait 2 caractères. Toute commande XY non-reconnue est considérée comme l'appel du processeur XY qui est chargé (s'il existe) dans la zone "processeur courant" et va assurer un dialogue spécifique avec l'utilisateur. Si XY n'est ni une commande ni un processeur, alors SMC émet un message d'erreur et attend une nouvelle commande.

Nota : un nom de processeur de SMC doit commencer par un caractère $\notin \{C, F, I, L, M, N, -\}$ qui sont les premières lettres des commandes de SMC.

5 - 2 - 1 Les commandes de SMC

On peut les regrouper de la façon suivante :

- *Commandes de manipulation d'items.*

I + création item

I - suppression item

- I suppression items de racine commune

I L chargement item (Item Load)

I S stockage item (Item Store)

I U chargement item sous ACN : SYS

I = affichage d'informations relatives à un item

L I liste items de racine commune

■ *Commandes de manipulation de noeuds* :

N + création noeud

N - suppression noeud

- N suppression noeuds de racine commune

N = affichage d'informations relatives à un noeud

L N liste noeuds de racine commune

■ *Commandes de structuration de graphe (filiations)*

N C connexion noeud vers item

N D déconnexion noeud vers item

C i C0 à C7 ; connexion fils i noeud père vers noeud fils

C - déconnexion premier fils noeud père vers noeud fils

■ *Autres commandes* :

M 1 duplication en mémoire Item 1 dans Item 2

M 2 duplication en mémoire Item 2 dans Item 1

M X échange en mémoire Item 1 et Item 2

F F sortie de SMC (retour au CCI)

■ *I + Création item*

L'item qu'on crée reçoit le nom donné en argument, une valeur vide et le type V (vide).

Exemples

! I+

NØM > EXEMPLE 1 D^C → EXEMPLE 1 est créé. Il existe donc désormais en bibliothèque , sa valeur initiale est vide et son type initial est V.

! I+

NØM > EXEMPLE 2 D^C ?S → la création de EXEMPLE 2 est refusée. Cela signifie qu'un item portant ce nom existe déjà (ayant le même nom externe).

■ *I - Suppression item*

L'item argument est supprimé en bibliothèque (nom et valeur), s'il existe. Cette commande est refusée s'il existe des filiations sur cet item.

Exemples

```
! I-
NØM > EXEMPLE 1 D^C          → EXEMPLE 1 n'existe plus.
! I-
NØM > EXEMPLE 3 D^C ?S       → la suppression de EXEMPLE 3 est refusée car
                                cet item n'existe pas
! I-
NØM > EXEMPLE 2 D^C ?F       → la suppression de EXEMPLE 2 est refusée car il
                                existe des filiations sur cet item.
```

■ *- I suppression d'items de racine commune*

On donne en argument une racine de 0 à 3 caractères. Les items dont le nom commence par cette racine (tous les items de la bibliothèque si la racine spécifiée est vide) seront supprimés quelque soit les filiations relatives à ces items (noter la différence avec I -). Les items supprimés sont listés au fur et à mesure par le système. Comme toutes les autres commandes de SMC, -I est interruptible par un alt-mode.

Exemples

```
! -I
RACINE > PRØ D^C             → suppression des items de racine : PRØ
- PRØGRAMME 1
- PRØGRAMME 2
! -I
RACINE > SC D^C              → suppression des items de racine SC : il n'y en
                                a aucun
!
```

■ *IL chargement item en mémoire (Item Load)*

L'item argument est lu en bibliothèque et chargé en mémoire dans la zone Item 1 de l'espace utilisateur.

Exemple

```
! IL                         → EXEMPLE 1 est chargé en mémoire dans la zone
                                Item 1
NØM > EXEMPLE 1 D^C
! IL
NØM > EXEMPLE 2 D^C          → EXEMPLE 2 succède à EXEMPLE 1 dans la zone
                                Item 1
! IL
NØM > EXEMPLE 3 D^C ?S       → EXEMPLE 3 n'existe pas, il n'est donc pas chargé.
                                Mais le nom de l'Item courant de Item 1 est deve-
                                nu EXEMPLE 3.
!
```

■ *IS stockage item en bibliothèque (Item Store).*

C'est l'opération inverse de IL : l'item placé en zone Item 1 de la partition utilisateur est stocké en bibliothèque sous le nom courant de l'item de Item 1. Autrement dit, l'item courant est mis à jour en bibliothèque.

Exemple

! IL		
NØM > EXEMPLE D^C	→	chargement item EXEMPLE en zone Item 1
! ED		
} opérations d'édition sous le processeur ED	→	opérations diverses sur Item 1 courant, par exemple, édition de texte (processeur ED), pour modifier la valeur de cet item. Cette valeur est par exemple un programme graphique que l'on veut mettre au point.
! GR	→	exécution par le processeur graphique GR du programme contenu en Item 1.
! ED }	→	nouvelles modifications du programme sous le processeur ED éditeur de texte.
! GR { etc...	→	nouvelle exécution du programme, etc..., et ainsi de suite jusqu'à ce que le programme soit au point.
! IS	→	ici,on décide de stocker en bibliothèque l'item courant EXEMPLE ; en bibliothèque EXEMPLE est donc mis à jour avec sa nouvelle valeur. Tant qu'on n'a pas fait IS, la valeur de l'item courant reste inchangée en bibliothèque.

Nota : on peut, en utilisant IL et IS, dupliquer un item en bibliothèque avec changement de nom ; comme ceci :

! IL NØM > CHARLES D^C	→	chargement de l'item CHARLES en Item 1
! IL NØM > GERARD D^C ?S	→	chargement de l'item GERARD dont on sait qu'il n'existe pas, d'où l'erreur signalée par le système. Mais, le nom courant de Item 1 est devenu GERARD,
! IS	→	et donc IS va stocker l'item courant GERARD avec la valeur courante de Item 1 qui est celle de CHARLES.
! IL NØM > CHARLES D^C	→	CHARLES existe en bibliothèque
! IL NØM > GERARD D^C	→	GERARD aussi ; et leurs valeurs sont identiques

■ *IU chargement item sous ACN : SYS*

En bibliothèque, tout nom est composé de deux éléments :

- ACN de l'utilisateur (:USE par exemple)
- nom (IMAGE par exemple)

} —> :USE D^C IMAGE D^C

L'ACN (Account number ou n° de compte) de l'utilisateur est celui qu'il a spécifié au logon (voir CMS). Lorsque l'utilisateur travaille sur la bibliothèque (par IL, IS par exemple), il a accès à tous ses items sans avoir à spécifier à chaque fois son ACN puisque celui-ci est connu du système depuis le logon. Mais en revanche, s'il veut charger un item créé sous l'ACN : SYS et que son propre ACN n'est pas : SYS, il peut le faire mais en utilisant non pas IL mais IU. Cela lui permet d'avoir accès à des informations (programmes graphiques, scénarios, par exemple) qui ne lui appartiennent pas mais qu'il veut utiliser.

Exemple :

! IU
NØM > SCENARIØ D^C —> chargement de l'item SCENARIØ sous ACN = :SYS

Nota : si l'ACN de l'utilisateur est :SYS, IL et IU sont équivalents.

■ *I = affichage d'informations relatives à un item*

Cette commande permet de connaître le type, le nom compacté et les filiations de l'item argument.

Exemple :

! I =
NØM > EXEMPLE 1 D^C
TYPT —> Type T (Texte)
NØMC = EXE8LP —> nom compacté
PERE CN01
PERE CN02 } —> noeuds pères (16 au maximum)
PERE N04

■ *LI liste items de racine commune*

La racine spécifiée par l'utilisateur doit faire de 0 à 3 caractères. Si elle est vide, tous les items de la bibliothèque sont listés (pour l'ACN de l'utilisateur). Comme toutes les autres commandes, LI est interruptible par un alt-mode.

Exemple

! LI

RACINE > PRØD^{C}

- PRØGRAMME

- PRØCEDURE

!

■ *N + création noeud*

Exemple

! N+ NØEUD > PFN1 D^{C} ?S → création refusée, ce nom existe déjà

! N+ NØEUD > PFN2 D^{C} → création du noeud PFN2

■ *N- suppression noeud*

Cette commande n'est acceptée que si le noeud argument existe et qu'il n'existe aucune filiation relative à ce noeud. Dans ce dernier cas, le système émettra l'un des messages d'erreur suivants :

?I s'il existe un lien de ce noeud vers un item

?P s'il existe au moins un lien père de ce noeud vers un autre noeud.

?F s'il existe au moins un lien fils d'un autre noeud vers ce noeud.

Exemple

! N- NØEUD > PFN2 D^{C} → PFN2 est détruit

! N- NØEUD > PFN3 D^{C} ?I → la suppression de PFN3 est refusée car il existe un lien de ce noeud vers un item.

■ *N = affichage d'informations relatives à un noeud*

Cette commande permet de connaître le nom compacté et les filiations relatives au noeud argument.

Exemple :

! N = NØEUD > CØ1 D^{C}

NØMC = CØ1 3 @ 0 → nom compacté

ITEM = CHAP1 → item fils de ce noeud

PERE 0 V01 → noeud père o de ce noeud (8 noeuds pères au maximum).

FILS 0 Q01 → noeud fils o de ce noeud (8 noeuds fils au maximum).

■ *- N = suppression de noeuds de racine commune*

Cette commande est semblable à - I mais s'applique à des noeuds.

■ *LN liste des noeuds de racine commune*

Cette commande permet de lister les noeuds de racine commune, celle-ci étant spécifiée sur 0 à 3 caractères. Si la racine spécifiée est vide, tous les noeuds sont listés (sous l'ACN utilisateur). Comme les autres commandes, LN est interruptible par un alt-mode.

Exemple :

```
! LN
RACINE > NTA D^C
- NTA01
- NTA02
!
```

■ *NC connexion noeud vers item*

On rappelle qu'un noeud ne peut possèder plus d'un lien vers un item (cf : structure physique d'un graphe d'enchaînement). Donc, si un noeud possède déjà un lien vers un item, la création d'une connexion semblable à celle décrite dans l'exemple suivant serait refusée.

Exemple

```
! NC   NØEUD > N01 D^C
ITEM > I01 D^C
!
```

■ *ND déconnexion noeud vers item*

Opération inverse de NC. La connexion noeud vers item est détruite si elle existe. Sinon, la commande est refusée.

Exemple

```
! ND   NØEUD > N01 D^C
!
```

■ *Ci C0 à C7 : connexion fils i noeud père vers noeud fils*

Si la connexion qu'on tente d'établir existe déjà (fils i déjà existant), la commande est refusée.

Exemple

```
! C2
PERE > N04 D^C
FILS > N08 D^C
!
```

■ *C - déconnexion premier fils noeud père vers noeud fils*

Cette commande détruit un lien d'un noeud père vers un noeud fils si ce lien existe ; sinon la commande est refusée. Le lien détruit est le premier lien fils (de fils 0 à fils 7) du noeud père argument vers le noeud fils argument.

Exemple

```
! N = NØEUD > NEX1 D^C
NØMC = NEX488
PERE 0    NEX0
PERE 1    NEXA
FILS 0    NEX2
FILS 1    NEX3
FILS 2    NEX5
FILS 3    NEX3
! C-
PERE > NEX1 D^C          → détruit le premier lien fils de NEX1
FILS > NEX3 D^C            vers NEX3, c'est-à-dire le lien fils 1.
! C -
PERE > NEX1 D^C          → le premier lien fils de NEX1 vers NEX3
FILS > NEX3 D^C            étant maintenant le lien fils 3, c'est
! N = NØEUD > NEX1 D^C      ce lien qui est détruit.
NØMC = NEX488
PERE 0    NEX0
PERE 1    NEXA
FILS 0    NEX2           → seuls subsistent les liens fils 0 et
FILS 2    NEX5             fils 2
!
```

■ *M1 duplication en mémoire Item 1 dans Item 2*

L'item courant de la zone Item 1 de la partition utilisateur est dupliqué en zone Item 2. S'il n'y a pas d'item courant en Item 1, la commande est refusée.

■ *M2 duplication en mémoire Item 2 dans Item 1*

L'item courant de la zone Item 2 de la partition utilisateur est dupliqué en zone Item 1. S'il n'y a pas d'item courant en Item 2, la commande est refusée.

■ *MX échange en mémoire Item 1 et Item 2*

Exemple

```
! IL                          ⟶ EXEMPLE 1 est chargé en Item 1
NØM > EXEMPLE 1 D^C
! M2                          ⟶ EXEMPLE 1 est dupliqué en Item 2
! IL                          ⟶ EXEMPLE 2 est chargé en Item 1
NØM > EXEMPLE 2 D^C
! MX                          ⟶ échange Item 1 et Item 2 : on a donc
!                               EXEMPLE 1 en Item 1 et EXEMPLE 2 en Item 2.
```

■ *FF sortie de SMC.*

On peut sortir de SMC soit définitivement, soit pour effectuer par exemple des assignations, auquel cas on peut revenir par !GØ

Exemple

```
! FF
? ! ASSIGN Ø A = CR1 D^C
? ! GØ D^C
!
```

5 - 2 - 2 Les processeurs de SMC

On trouvera ci-après la liste par ordre alaphabétique des processeurs de SMC, suivie d'un commentaire sur l'ordre de présentation qui a été adopté pour leur présentation détaillée dans la suite du manuel.

3D Exécution d'un programme en langage machine placé en Item 2 et contenant éventuellement des primitives graphiques 3 dimensions.

DG Génération dans Item 1 d'un programme graphique compatible GR (c'est-à-dire éxécutable par le processeur graphique GR) à partir de coordonnées de segments acquises en ZDC (zone de données communes ; voir ci-après : "système vidéo-graphique, généralités").

DI Gestion des mots du dictionnaire de la langue française.

ED Editeur de textes dans Item 1 et Item 2

EI Parcours d'un graphe d'enchaînement (pour enseignement assisté par ordinateur par exemple).

ES Extraction d'une chaîne de segments dans une image TV.

EX Editeur sélectif (exhaustif) de l'espace des noms et valeurs.

EZ Extraction exhaustive des contours d'une image TV.

G2 Interpréteur graphique 2 dimensions.

G3 Interpréteur graphique 3 dimensions.

GC Assemblage d'un programme placé en Item 1 avec rangement du module objet en Item 2 (ce programme en langage machine sera exécutable par le processeur 3 D).

GE Gestion du dialogue avec SMC (niveau "!").

GF Acquisition en ZDC et transformation d'un dessin avec création d'un fichier graphique.

GG Visualisation graphique et/ou vidéo d'un dessin, dont les coordonnées sont acquises en ZDC.

GI Processeur identique à GE, mais les commandes à SMC sont acquises en Item 2 et le caractère "D^C" est remplacé par ";".

GL Extraction des grandes lignes d'un dessin dont les coordonnées sont acquises en ZDC

GØ Lancement du parcours d'un graphe (le parcours sera poursuivi par le processeur EI).

GR Interpréteur graphique : sur-ensemble de G3.

GT Version vidéo de GR.

GV Version vidéo de G3.

GW Version rapide de GV.

KA Sauvegarde et restauration d'item (item argument en Item 1 ; KA détruit Item 2).

KI Calcul des axes d'inertie d'une partition d'une image TV.

KØ Extraction d'un contour contenu dans une image TV avec remplissage éventuel.

LP Editeur de fichiers graphiques, construits à partir de ZDC ou du curseur visu, ou du light-pen.

PA Recherche et remplacement d'une séquence carcatéristique de points dans une image TV.

PH Analyse lexicographique, syntaxique et sémantique d'une phrase en français libre.

RE Commande directe de la baie de commutation vidéo et des magnétoscopes.

SA Sauvegarde, restauration et modification d'un nom quelconque et de sa valeur de l'espace des noms et valeurs.

SC Diffusion d'un film digital et trucages.

SE Création et/ou mise à jour d'items de divers types et réinitialisation de la ZDC.

SØ Synthèse de son à partir du clavier d'une console de visualisation.

ST Extraction des grandes lignes d'un dessin dont les segments sont acquis en ZDC.

TA Interpolateur (pour animations) graphique-graphique (segment à segment) ou vidéo-vidéo (point à point).

TB Interpolateur graphique (comme TA) avec émission en ZDC des dessins intermédiaires.

TI Processeur identique à TV (voir TV) mais les commandes sont acquises en Item 1 et la caractère "D^C" est remplacé par le caracatère ";".

TN Création d'une image TV à partir de moyens divers et souples (light-pen notamment).

TØ Interpolateur vidéo-vidéo octet à octet ou mot à mot (son numérique ou image numérique).

TV Création, modification, suppression d'images TV, opérations sur image et entre images, trucages.

VA Reproduction d'une image TV sous forme de caractères alphanumériques.

VG Reproduction d'une image TV sous forme de segments.

VI Télécommande, recherche et diffusion de séquences sur magnétoscope.

VØ Génération au curseur d'une image TV (génération de volets notamment).

VT Emission en ZDC de segments calculés à partir d'une image TV (voir VG).

ZØ Transformation d'image (dont le zoom), extracteur de contours, etc.

Les modes d'emploi des processeurs de SMC énumérés ci-dessus figurent dans la suite de ce manuel et appellent les remarques suivantes :

- les processeurs graphiques (avec éventuellement sortie vidéo), sont documentés ensemble au chapitre consacré à GR. Ce sont les processeurs : GR, G2, G3, GT, GV, GW.

- de même pour EI et GØ ainsi que GC et 3D.

- certains processeurs peuvent utiliser la ZDC, soit comme producteurs, soit comme consommateurs (voir ci-après le § 6-4 consacré à la ZDC).

- le processeur GE qui gère le dialogue (niveau !) avec l'utilisateur (commandes et appels de processeurs) ne fait pas l'objet d'un chapitre particulier. Cela n'est pas nécessaire car le dialogue niveau ! a été explicité au § 5.2.1 ci-avant.

L'ordre de présentation des processeurs adopté dans ce manuel n'est pas contraignant, l'utilisateur ne s'intéressant qu'aux processeurs répondant à ses besoins. Toutefois il faut souligner l'importance des processeurs ED (éditeur de textes) permettant notamment de rentrer des programmes graphiques et des scénarios d'enchaînement d'images ; et des processeurs GR et TV, GR permettant de s'initier au graphique et TV à l'image numérique.

L'ordre de présentation des processeurs peut être commenté comme suit :

- éditeur de textes ED.

- processeur graphiques : GR, G2, G3, GT, GV, GW regroupés dans le chapitre consacré à GR.

- processeurs de télévision numérique : TV, TI, SC, VØ, TN

- interpolateurs : TA, TB, TØ

- extracteurs de contours : KØ, ES, EZ

- autres processeurs, de manipulation de graphismes, images numériques et sons : DG, LP, VG, VA, VT, GG, GF, ST, GL, ZØ, PA, KI, GC, 3D, SØ.

- processeurs de "service" : EX, SA, KA, GI, SE.

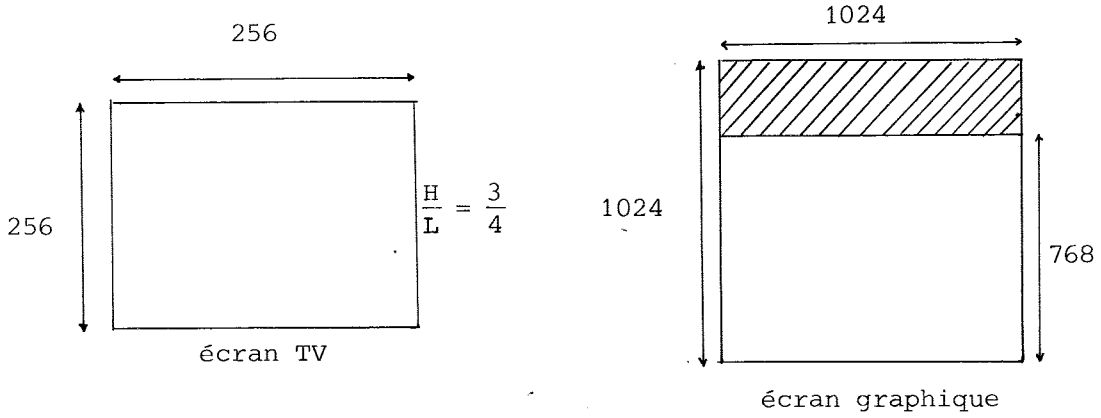
- processeurs orientés "graphes d'enchaînement" et enseignement assisté par ordinateur : DI, PH, RE, EI, GØ, VI.

6 - SYSTEME VIDEO-GRAPHIQUE : GENERALITES

Ce paragraphe a pour but de présenter des notions souvent référencés dans les chapîtres consacrés aux processeurs de SMC, du moins ceux qui font intervenir les graphismes et les images numériques.

6 - 1 Ecran TV et écran graphique

1°) Ecran TV : la définition sur écran TV (liée non pas au moniteur de télévision utilisé, mais au système de numérisation) est la suivante : 256 points sur 256. De plus, le rapport entre la hauteur et la largeur de l'écran TV est de 3/4, d'où il suit que les points sont plus "tassés" sur l'axe des Y que sur l'axe des X.



2°) Ecran graphique : la définition sur écran graphique (4010 et 4014) est de 1024 points sur 1024 points. Tous ces points sont accessibles par programme mais, en ordonnées, seuls 768 points sont visualisés.

Conséquences de tout ceci :

1) supposons par exemple qu'on dessine un carré par programme graphique sur l'écran visu : si l'on exécute ce même programme en activant la sortie vidéo (processeur GT par exemple), on obtiendra sur l'écran TV un rectangle ; il faut donc modifier les échelles si l'on veut voir un carré (voir GR).

2) étant donnée la faible définition sur l'écran TV, les lignes obliques, parfaites sur visu graphique, donneront sur écran TV des effets de marches d'escalier.

3) sur visu, seuls 768 points sont visualisés et accessibles par curseur manuel. Donc lorsqu'on utilisera le curseur pour se positionner sur une image TV (processeurs TV, VØ, ES, EZ, etc...), on n'aura pas accès au quart supérieur de l'image. Il y a une solution qui consiste, quand le cas se présente, à inverser l'image (processeur TV), et à la remettre à l'endroit ensuite, après avoir fait les opérations envisagées.

6 - 2 Repérage des points

En général, les points sont repérés de la manière suivante :

- sur écran TV : origine en haut à gauche.
- sur écran visu : origine en bas à gauche.

Ceci est toujours vrai pour l'écran visu, mais pour l'image numérique, cela dépend des processeurs :

- s'il s'agit d'un processeur de la famille des processeurs graphiques : GR, GT, etc..., l'origine est en bas à gauche, comme sur un dessin, ce qui est normal puisque l'image numérique est, dans ce cas, une sortie vidéo d'un dessin tracé en même temps sur écran TV et/ou sur écran graphique.

- pour d'autres processeurs, notamment les interpolateurs TA, TB, l'origine est en haut à gauche pour les interpolations faisant intervenir des images TV, et en bas à gauche pour les interpolations faisant intervenir des dessins.

6 - 3 Différentes formes d'un dessin

Un dessin peut se présenter dans le système SMC sous trois formes équivalentes :

1 - sous forme de programme graphique

2 - sous forme d'image numérique.

3 - sous forme de fichier graphique.

1°) sous forme de programme graphique

Un programme graphique est un texte contenu dans un item de type D (dessin) ou T (Texte). Ce programme peut avoir été créé :

. soit directement par un utilisateur. Dans ce cas l'utilisateur a écrit son programme graphique en utilisant le processeur ED (éditeur de textes) ; ce programme est ensuite exécutable par un processeur graphique comme GR (et les processeurs de la même famille : GT, GV, GW, G2, G3) ;

. soit automatiquement par le processeur DG qui, à partir de coordonnées de points qu'il reçoit en ZDC est capable de générer un programme graphique dans un item. Tout comme un programme rentré manuellement par un utilisateur, ce programme graphique est interprétable par l'un des processeurs graphiques (GR, GT, etc...).

. soit d'une manière mixte, c'est-à-dire que l'utilisateur a créé un programme graphique dont il a écrit une partie lui-même, et dont une autre partie a été générée automatiquement, par exemple à partir du curseur d'une visu graphique (en utilisant le processeur G2, primitive I par exemple).

2°) sous forme d'image numérique

Mis à part le problème de la définition, une image numérique est équivalente à un programme graphique ; elle peut d'ailleurs avoir été générée par un programme graphique interprété par le processeur GT.

Une image numérique peut avoir plusieurs sources, qui peuvent d'ailleurs fort bien coexister :

. saisie par caméra

. traçage au light-pen sur écran TV.

. génération par programme (sortie vidéo d'un dessin, utilisation d'un processeur permettant de générer et/ou transformer et/ou combiner des images numériques : TV, VØ, ZØ, ES, etc...)

. traçage au curseur de la visu avec sortie vidéo (processeur TV (commande M), VØ, LP, etc...).

3°) sous forme de fichier graphique

Un fichier graphique strictement équivalent à un dessin peut être généré automatiquement par l'un des processeurs LP ou GF. Le dessin origine peut, indifféremment :

- être rentré au curseur sur une visu ou au lightpen sur écran TV.

- être le résultat de l'interprétation par un processeur graphique (par exemple GR), d'un programme graphique.

- être le résultat d'un calcul effectué par l'un des processeurs de SMC, par exemple un processeur extracteur de contour sur une image numérique (KØ, ES, EZ).

6 - 4 La zone de données communes ou ZDC

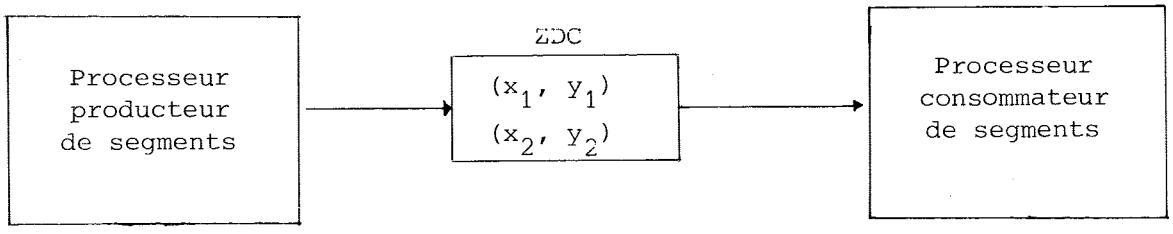
La ZDC est une zone mémoire accessible en lecture et écriture à l'ensemble des utilisateurs du système. Elle permet donc à deux programmes se déroulant concurremment de dialoguer et de se synchroniser entre eux.

Dans le cas de processeurs de SMC, cette zone est utilisée en général pour échanger des coordonnées de segments : $((x_1, y_1), (x_2, y_2))$

(x_1, y_1) : coordonnées de l'origine du segment

(x_2, y_2) : coordonnées de l'extrémité du segment.

On appelle "processeur producteur de segments" un processeur qui envoie des coordonnées de segments en ZDC ; et "processeur consommateur de segments" un processeur qui lit des coordonnées de segments en ZDC.



Le schéma de la page suivante énumère les processeurs de SMC pouvant travailler sur la ZDC comme producteurs et consommateurs de segments. On remarquera que certains processeurs peuvent être utilisés aussi bien comme producteurs de segments que comme consommateurs de segments.

L'utilisation de la ZDC fournit de très larges possibilités à l'utilisateur des processeurs de SMC, comme on pourra le constater en parcourant la documentation de ces processeurs et en accumulant de l'expérience sur leur utilisation.

Utilisation de la ZDC

La fonction de la ZDC est à la fois l'échange de segments et la synchronisation inter-processeurs (entre le producteur et le consommateur). Le processus d'échange étant le suivant :

- le processeur producteur s'assure, avant d'émettre un nouveau segment en ZDC, que le segment précédent a été consommé ; si ce n'est pas le cas, il attend.

- le processeur consommateur s'assure, avant de consommer un nouveau segment, que celui-ci a été produit ; si ce n'est pas le cas, il attend.

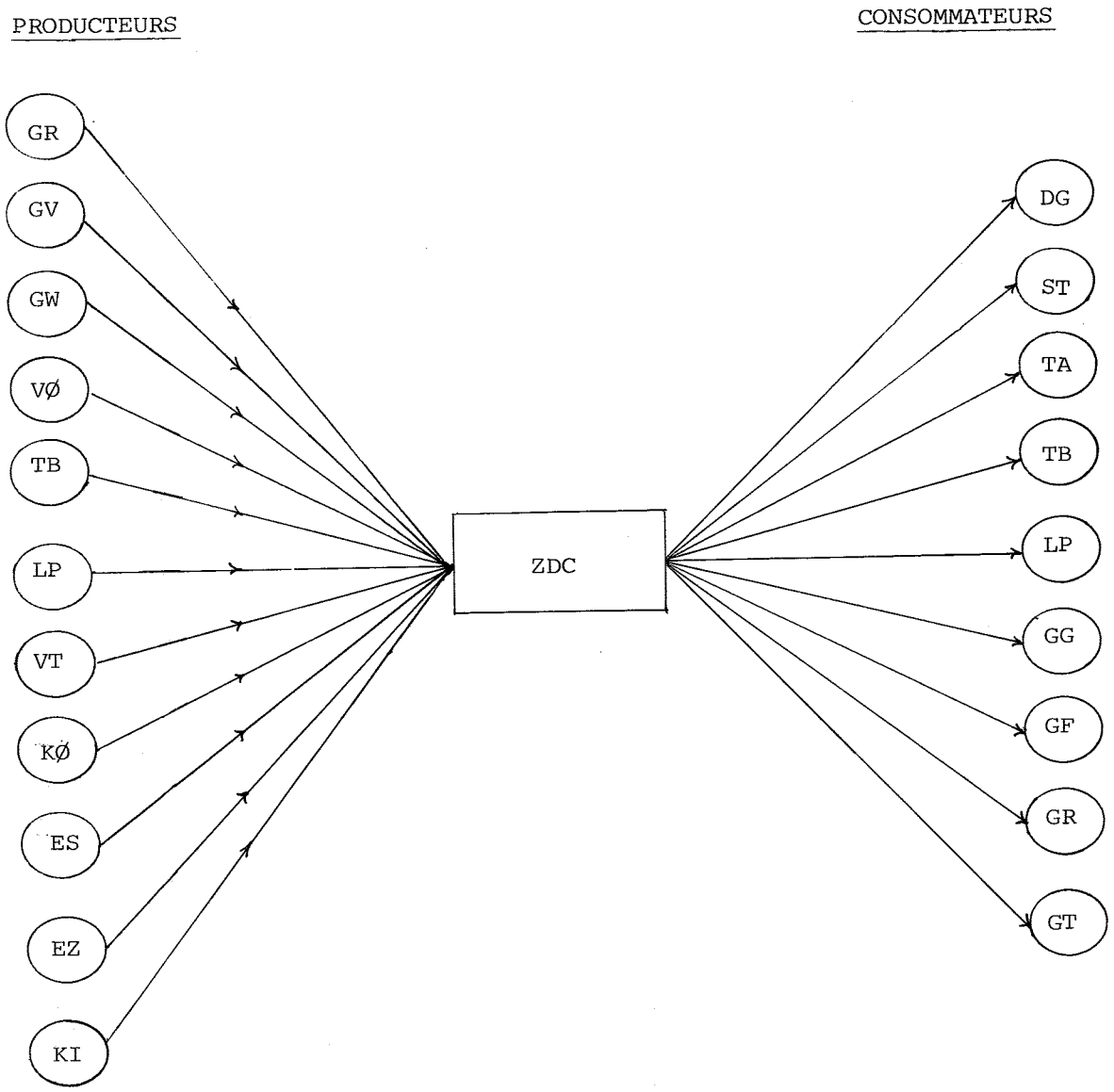
La fin d'échange est provoquée par l'envoi en ZDC par le producteur, d'un segment de coordonnées ((0,0), (0, 0)), qu'on a coutune de désigner dans la documentation par "segment nul à l'origine" ou "ØAB" (voir processeur GR).

Nota : avant de lancer un échange de segments, il faut éventuellement réinitialiser à 0 la ZDC. Ceci se fait en utilisant le processeur SE de la façon suivante :

```
! SE
TYPE > 0                    remise à zéro de la ZDC
!
```

Cela n'est pas nécessaire si le dernier échange que l'on a effectué s'est bien terminé.

SMC 40


PRODUCTEURS
CONSOMMATEURS
GR
GV
GW
VØ
TB
LP
VT
KØ
ES
EZ
KI
ZDC
DG
ST
TA
TB
LP
GG
GF
GR
GT

6 - 6 Transformations

Un certain nombre de processeurs de SMC offrent la possibilité à l'utilisateur :

1°) de définir une transformation matricielle sur image numérique : dans ce cas, l'utilisateur rentre au cours du dialogue les paramètres de cette transformation, comme on le verra en étudiant les processeurs concernés. (TV, VG, ZØ, etc...).

2°) de faire intervenir, en plus de la transformation définie au cours du dialogue avec le processeur, un programme de transformation que l'utilisateur a lui-même écrit en fonction de ses besoins propres. Ce programme, assemblé et linké par le processeur GC, est contenu dans un item de type P créé par l'utilisateur, et doit être placé en Item 2 pour être exécuté. On verra en étudiant chaque processeur, comment il est possible de faire intervenir un tel programme. En général, cela se fait en répondant oui (Ø) au cours du dialogue à la question : "PRØGRAMME DE TRANS ?". L'écriture d'un tel programme, répondant à des besoins particuliers de l'utilisateur, suppose de sa part la connaissance du langage assembleur (vois CMS, assembleur ASSYS), et l'interface entre un tel programme et le système SMC, est le suivant :

- *en graphique* :

Il s'agit de transformer des coordonnées de segments. Le programme placé en Item 2 sera appelé par le processeur pour chaque segment, avec, dans le registre A, l'adresse mot des coordonnées $((x_1, y_1), (x_2, y_2))$ du segment courant, les coordonnées étant rangées dans cet ordre et occupant 4 mots au total.

Le programme de l'utilisateur doit sauvegarder tous les registres qu'il utilise, effectuer sa transformation sur les coordonnées du segment courant, puis restaurer et rendre la main au processeur par RSR.

- *en vidéo (+ graphique éventuellement)*

Il s'agit de modifier les coordonnées du point blanc courant, et éventuellement de générer un segment graphique (processeur VG par exemple). Les paramètres passés par le processeur au programme de l'utilisateur sont les suivants :

- registres X et Y : coordonnées du point blanc courant.
- registre B :
  - si B < 0 : alors A contient l'adresse mot d'une demande d'écriture graphique que l'utilisateur peut utiliser (voir CMS).
  - si B = 1 : c'est l'image d'arrivée
    si B = 2 : c'est l'image de départ ou une image intermédiaire

    } interpolateur (TA, TB)

Là aussi, l'utilisateur doit sauvegarder les registres (sauf X et Y dont il veut modifier le contenu), et les restaurer avant de rendre la main au processeur par RSR.

7 - LES PROCESSEURS

Ils sont présentés ci-après, dans l'ordre indiqué précédemment au § 5.2.2.

PROCESSEUR ED : EDITEUR DE TEXTES

1 - GENERALITES.

■ *Appel* : On est sous GE (!), on frappe ED, le processeur ED est chargé et positionne le curseur HOME (en haut à gauche de l'écran).

■ *Items traités* : ED ne permet de travailler que sur un ou deux items, ces items ayant été préalablement chargés dans Item 1 (et Item 2).

■ *Choix de l'Item* : implicitement, on travaille sur Item 1.

■ *Test de l'Item* : frapper T^C, 2 réponses possibles :

1 ⇒ néant : on est dans l'item 1

2 ⇒ cloche : on est dans l'item 2

■ *Changement d'Item* :

1 ⇒ A^C : on travaille sur Item 1
2 ⇒ B^C : on travaille sur Item 2

} voir paragraphe "Entrée dans GE et type d'item" ci-dessous. Les mêmes règles s'appliquent.

■ *Sortie du processeur ED* :

1 - F^C : on sort de ED on reste dans GE (!)(voir aussi M^{CS} ci-dessous)

2 - W^C : on sort de ED et on sort de GE(?)(faire !GØ, on revient en ED Home).

■ *Classes de caractères* : ED distingue deux classes de caractères :

1 - les caractères affichables : de "Ø" à "-" (codes '20 à '5F ; pas de minuscules)

2 - les caractères k^C (CTRL k) et k^{CS} (CTRL SHIFT k) qui sont des caractères de commande du processeur, k désignant n'importe quelle touche.

■ *Repérage des informations sur l'écran* : les informations sont toujours repérées par coordonnées sur l'écran (position du curseur alphanumérique) et jamais en donnant un numéro de ligne ou de caractère.

■ *Répétitions et compactages* : on compte en base 36 (0,...,9,A,...Z)

■ *Appel automatique de ED lors d'un parcours de graphe* :

Quand ED est appelé automatiquement lors du parcours d'un graphe de noeuds-items (cf processeurs GØ et EI), le 1er caractère de l'item est interprété par ED :

- si 1er car ≠ ".", l'écran est effacé avant l'édition de l'item (E^C).
- si 1er car = ".", l'écran n'est pas effacé.

Après l'édition, ED rend la main au système d'enchaînement des items (cf : GØ et EI).

■ *Utilisation de ED en batch* : il est pratiquement impossible d'utiliser ED en batch du fait que toutes ses commandes sont des CTRL.

■ *Recommandation* : l'effacement de l'écran par PAGE est une fonction interne à la visu qui ramène le curseur Home, alors que pour ED la position en cours du curseur peut être différente. Il est donc déconseillé de faire PAGE sous ED, utiliser la commande d'édition C^C présentée ci-dessous.

■ *Entrée dans ED et type d'item* : A l'appel de ED, 3 cas sont possibles pour le type de l'item 1.

1°) Type ≠ T, ≠ D et ≠ I : ⟶ Cloche ; il faut sortir de ED (par F^C ou W^C)

2°) Type = V (item vide) : ⟶ ED initialise Item 1 à blanc, et, s'il possède un nom, il prend le type T ; on est ainsi ramené au cas 3.

3°) Type = T ou D ou I : ⟶ On peut travailler sur le texte contenu dans l'item.

- rappel : type T : texte.
  type D : texte à valeur de programme graphique.
  type I : film digital.

2 - COMMANDES DU CURSEUR.

. I^C ⟶ une position à droite (en bout de ligne ⟹ début de ligne suivante
" " " page ⟹ curseur home)

. K^C ↑ une position au-dessus (sur 1ère ligne ⟹ ineffectif)

. H^C ⟵ une position à gauche (en début de ligne ⟹ ineffectif)

. J^C ou LF ↓ une position au-dessous (LF sur dernière ligne ⟹ curseur home)

. M^C ou Return ⟸ retour début de ligne.

. Z^C ↖ Curseur home

Nota : V^C, commande habituelle de tabulation, est sans effet sous ED.

3 - COMMANDES D'EDITION.

. C^C effacement écran et curseur home ; l'item courant reste intact.

. E^C réaffichage du contenu de l'item et curseur home. E^C est utile notamment pour faire apparaître le résultat d'une opération de mise à jour, déplacement chaîne de caractères, etc... qu'on vient de faire sur l'item.

Remarque : E^C est interruptible par S^C (qui ramène le curseur home).

. $R^C R^C$ Remise à blanc item en cours, effacement écran, et curseur home.

Remarque : R^C doit être doublé pour être effectif. Un R^C non-doublé est sans effet (pour éviter les remises à blanc involontaires).

. G^C Remet sur lignes tous les caractères placés en interlignes. Cela est utile parce que le système graphique met en interligne tous les caractères qu'il ne reconnaît pas ou qui sont en erreur de syntaxe.

. S^C Si l'item est de type D ou I il reprend le type T.

. U^C Up. Tous les caractères affichables qui suivent vont aller en interligne.

. L^C Contraire de Up ; les caractères vont aller en lignes. C'est l'état implicite de ED.

4 - DEPLACEMENT DE CHAINE DE CARACTERES.

On peut déplacer une chaîne de longueur quelconque (pas forcément limitée à une ligne), avec si l'on veut écrasement d'une autre chaîne, déplacement de la chaîne sur elle-même, etc... Il suffit d'indiquer l'origine de la chaîne (1er caractère), sa "queue" (dernier caractère) et sa nouvelle origine :

. $\emptyset^C$ Origine de la chaîne.

. Q^C Extrémité de la chaîne (queue).

. $\emptyset^{CS}$ Nouvelle origine (ou destination).

Nota : si la chaîne est trop longue pour être déplacée, elle ne le sera pas. Visualiser le résultat de l'opération par E^C.

5 - REPETITIONS.

. X^C répétition d'un caractère. Pour répéter n fois le caractère k on tapera : k X^C n on obtiendra : kkk.....k.
n fois

Cette commande vaut aussi pour les caractères de déplacement du curseur : exemple I^C X^C 2 déplacement curseur 2 positions à droite.

Nota : - si n = 0 répétition jusqu'au bout de la page.
- X^C est interruptible par S^C.
- on est en 1ère ligne ; pour passer en dernière ligne on peut faire : J^C X^C Y.

. Y^C répétition du dernier caractère ou caractère de contrôle jusqu'en bout de ligne, en laissant intact le dernier caractère de la ligne ; le curseur restant positionné sur ce dernier caractère de la ligne.

Exemple : effacement d'une ligne : ⌀ Y^C ⌀.
positionnement en fin de ligne : I^C Y^C.

6 - COMPACTAGES.

On ne compacte que des chaînes continues d'1 même caractère, en prenant l'étendue maximum de cette chaîne.

. L^{CS} toute chaîne maxi de blancs est remplacée par 1 blanc.

. K^{CS}% Itérations graphiques : ne travaille que sur des chaînes de caractères 1, 2, 3 ou 4 (cf : GR).

K^{CS} appliqué à une chaîne de n caractères 1111111 1
n

1°) Calcule $\frac{n}{35} \Longrightarrow \begin{cases} Q \\ R \end{cases}$

2°) $\begin{cases} Q \geqslant 6 & \text{génère } \%Z(\%q(k)) \\ 6 > Q > 0 & \text{génère } \%Z(\underbrace{kkk.....k}_{q \text{ fois}}) \\ Q = 0 & \text{rien.} \end{cases}$

3°) $\begin{cases} R \geqslant 6 & \text{génère } \%\, r(k) \\ 6 > R > 0 & \text{génère } \underbrace{kk...k}_{r \text{ fois}} \\ R = 0 & \text{rien} \end{cases}$

} la concaténation de ces deux chaînes se substitue à la chaîne initiale

Exemples : $\underbrace{3333.....3}_{300 \text{ fois}} \xrightarrow{K^{cs}\%} \%Z(\%\ 8(3))\ \%K(3)$ (300/35 Q = 8 R = 20)

$222222 \xrightarrow{K^{cs}\ \%} \%6(2)$

$22222 \xrightarrow{K^{cs}\%} 22222$ (inchangé)

. $K^{cs}K$ <u>changement échelle graphique</u> : mêmes chaînes (1, 2, 3, 4) que $K^{cs}\%$. Chaînes limitées à $n \leqslant 35 * 5 + 34$ (35 = Z ; 34 = Y)

$K^{cs}K$ appliqué à la chaîne $\underbrace{1111.....1}_{n \text{ fois}}$

1°) Calcule $\frac{n}{35}$ $\begin{cases} Q \\ R \end{cases}$ $(\leqslant 5)$

2°) $\begin{cases} Q \neq 0 & \text{génère } KZ\ \underbrace{11....1/}_{q \text{ fois}} \\ Q = 0 & \text{rien} \end{cases}$

3°) $\begin{cases} R \geqslant 6 & \text{génère } Kr * 1/ \\ 6 > R > 0 & \text{génère } \underbrace{11....1}_{r \text{ fois}} \\ R = 0 & \text{rien} \end{cases}$

} La concaténation de ces deux chaînes se substitue à la chaîne initiale.

Exemple : 444 —— 4 $\xrightarrow{K^{CS}\ K}$ KZ*44/K3*4/ (voir GR).

73 fois

7 - D^{C} N^{CS} SERONT PRECISES DANS LA NOTICE SUR LES PROCESSEURS 3D ET GC.

8 - M^{CS} FIN D'ITEM.

M^{CS} sert à désigner le dernier caractère significatif de l'item ; les caractères suivants jusqu'à la fin de page ne seront pas sauvegardés sur disque. Lors d'un rechargement, ils seront mis à blanc. M^{CS} provoque, comme F^{C} un retour à GE.

LANGAGE ET PROCESSEURS GRAPHIQUES

1 - GENERALITES.

1.1 - Programme graphique.

1.2 - Interprétation.

1.3 - Visualisation graphique.

1.4 - Le tracé Vidéo.

2 - LE LANGAGE GRAPHIQUE 2 DIMENSIONS.

2.1 - Référentiel absolu, référentiels relatifs.

2.2 - Unités.

2.3 - Déplacements élémentaires.

2.4 - Visualisation graphique/vidéo et dispositifs externes.

2.4.1 - Taille de l'écran graphique.

2.4.2 - Taille de l'écran vidéo.

2.4.3 - Primitives agissant sur la visualisation.

2.4.4 - Dispositifs externes Hard-Copy, Camera.

2.5 - Marquage.

2.6 - Changements d'unités.

2.7 - Itérations.

2.8 - Transformations.

2.9 - Pile des constantes.

2.10- Contextes graphiques.

2.11- Conditions.

2.11.1 - Autorisation/inhibition de primitives.

2.11.2 - Autorisation/inhibition de séquences.

2.12 - Sous-programmes et bibliothèques.

2.12.1 - Définitions sous-programmes et bibliothèques.

2.12.2 - Appel de bibliothèque.

2.12.3 - Appel de sous-programme.

LANGAGE ET PROCESSEURS GRAPHIQUES.

1 - GENERALITES.

1.1 - Programme graphique.

Un programme graphique s'exprime dans un langage imterprétatif décrit dans les paragraphes suivants. Il est obligatoirement placé dans un item de type D ou T (Dessin ou Texte), pour être exploité par un processeur graphique (G2,G3,GV,GW,GR,GT). Le programme est écrit de l'une des façons suivantes :

- sous ED (voir processeur ED) ;
- par programme utilisateur générant un programme graphique dans un item (voir le processeur DG par exemple) ;
- par génération interactive (voir primitive I du langage graphique).

1.2 - Interprétation.

Un programme graphique est interprété de gauche à droite et de haut en bas . Les blancs sont neutres pour l'interpréteur.

Le programme graphique placé dans Item 1 est interprété par un processeur graphique ; on verra que certaines commandes font intervenir aussi Item 2 et "Item 3". Lorsqu'une erreur de programme est détectée, les caractères concernés du programme sont mis en interligne ; on les fait apparaître en affichant l'item (commande d'affichage E^C sous ED). Après correction, on fera redescendre les caractères résiduels par G^C (sous ED).

1.3 - Visualisation graphique.

Le tracé obtenu par interprétation du programme graphique est visualisé en graphique :

. par G2 G3 sur UL 2

. par GV GW sur UL B (qu'il faut donc assigner !)

. par GR GT sur { UL 2 / UL B } en bascule, la primitive V rencontrée.

dans le programme inverse la visualisation entre les UL 2 et B. Initialement c'est sur l'UL 2 que le traçage est effectué.

1.4 - Le Tracé Vidéo.

Les processeurs graphiques GV,GW (et GT, voir nota) en plus du tracé graphique (éventuellement inhibé par assignation vide de l'UL B par !ASSIGN B = D^C), construisent une image TV équivalente compatible avec les processeurs de TV.

GV est équivalent à G3 avec en plus le tracé vidéo.

GW est équivalent à GV, mais en plus rapide.

GT est équivalent à GR avec en plus le tracé vidéo.

Sur les particularités respectives de l'écran graphique et de l'écran vidéo, voir au § 2.4 écran graphique, écran vidéo.

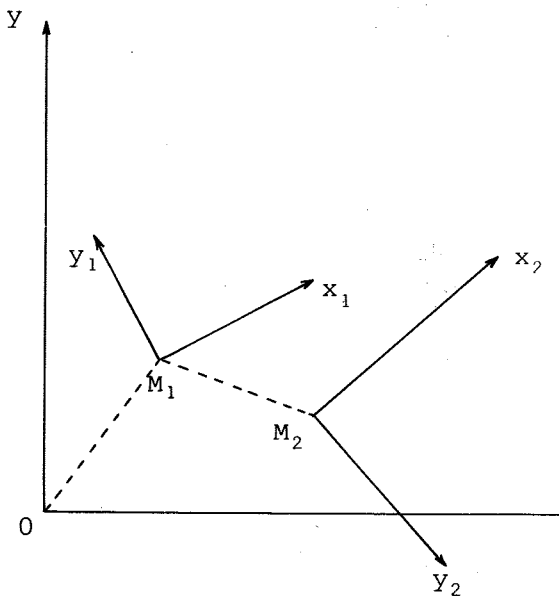
Nota : on trouvera dans la présentation de SMC "système vidéo-graphique, généralités" des précisions en ce qui concerne notamment les moyens de génération et les différentes formes d'un dessin.

2 - LE LANGAGE GRAPHIQUE 2 DIMENSIONS.

2.1 - Référentiel absolu ; référentiels relatifs.

Référentiel absolu : ØXY, Ø étant le point inférieur gauche de l'écran.

Référentiel relatif : tout référentiel M xy obtenu en appliquant une transformation $\mathscr{C}$ et un déplacement D au référentiel courant (qui peut être lui-même soit le référentiel absolu soit un référentiel relatif). Toute transformation et tout déplacement appliqués à un référentiel sont exprimés par rapport à ce référentiel (et non pas par rapport au référentiel absolu).



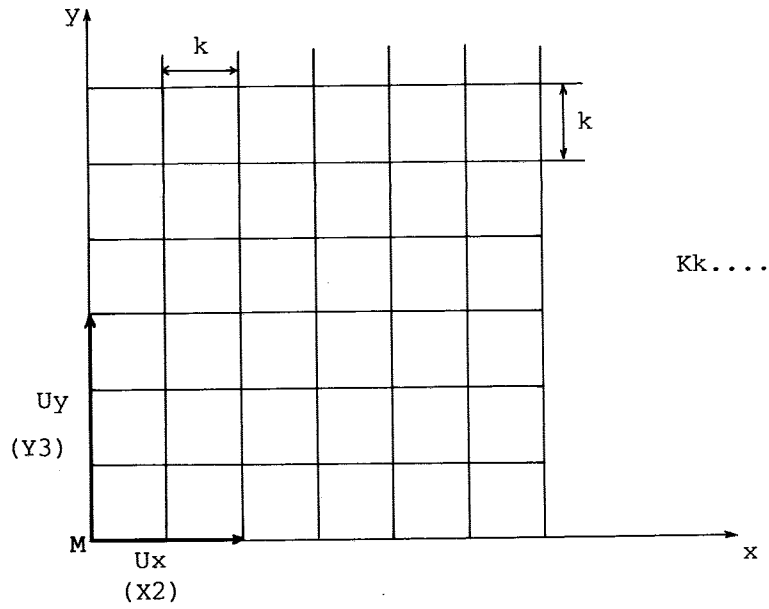
$$\text{ØXY} \xrightarrow[\overrightarrow{OM_1}]{\mathscr{C}_1} M_1x_1y_1$$

$$M_1x_1y_1 \xrightarrow[\overrightarrow{M_1M_2}]{\mathscr{C}_2} M_2x_2y_2$$

$\mathscr{C}_2$ et $\overrightarrow{M_1M_2}$ s'expriment par rapport au référentiel $M_1x_1y_1$

2.2 - Unités (dans le référentiel relatif).

K	Quantification du plan relatif. S'exprime en nombre de points physiques : Kk avec $1 \leqslant k \leqslant Z$ (base 36, $Z = 35_{10}$). Un écran graphique fait 1024 x 1024 points (voir § 2.4).
X	Echelle sur l'axe des X. Xx signifie $Ux = k \times x$ avec $1 \leqslant x \leqslant Z$
Y	Echelle sur l'axe des Y. Yy signifie $Uy = k \times y$ avec $1 \leqslant y \leqslant Z$



Kk..... X2 ..... Y3

Syntaxe Kk ..... Xx ..... Yy séquences rencontrables partout dans un programme ou sous-programme graphique.

Valeur implicites k = x = y = 1

Attention : L'affectation de valeur à K ne change pas x et y courants :

Exemple : K4 X4 Y4 (⟶ Ux = Uy = 16) puis K2 rencontré ne change pas Ux et Uy.

E Echelle générale. (GR GT uniquement)

S'applique à toute coordonnée z après que celle-ci ait été calculée dans le référentiel courant. Soit z une coordonnée quelconque,

z tracé = z calculé x $\frac{\alpha}{8}$ si on fait Eα $1 \leqslant \alpha \leqslant Z$

Valeur implicite : $\alpha = 8$

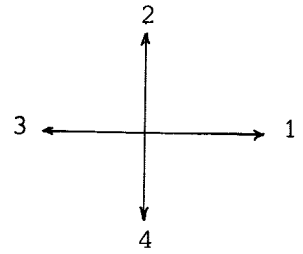
Pour réduire globalement un dessin, faire Ep avec $p < 8$.

Pour agrandir globalement un dessin, faire Eq avec $p > 8$.

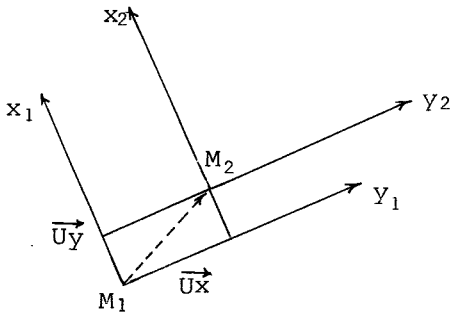
2.3 - Déplacements élémentaires du référentiel relatif courant.

Ces déplacements se font par rapport au référentiel lui-même.

1	+ $\overrightarrow{Ux}$
2	+ $\overrightarrow{Uy}$
3	- $\overrightarrow{Ux}$
4	- $\overrightarrow{Uy}$



Exemple : déplacement 12 (≡ 21)



Nota : Toute translation est une combinaison linéaire de déplacements 1,2,3,4.

2.4 - Visualisation graphique/vidéo, dispositifs externes.

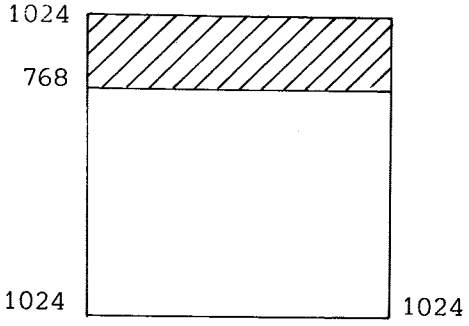
2.4.1 - Taille de l'écran graphique.

L'écran graphique fait 1024 × 1024 points physiques. En y, seuls 768 points sont visualisés (les 1024 restant programmables). Lorsque le point courant M sort de l'écran, deux possibilités.

1 - sous G2 G3 GV GW : M réentre dans l'écran, les coordonnées x/y étant calculées (kx $\frac{\alpha}{8}$ / ky $\frac{\alpha}{8}$) modulo 1024.

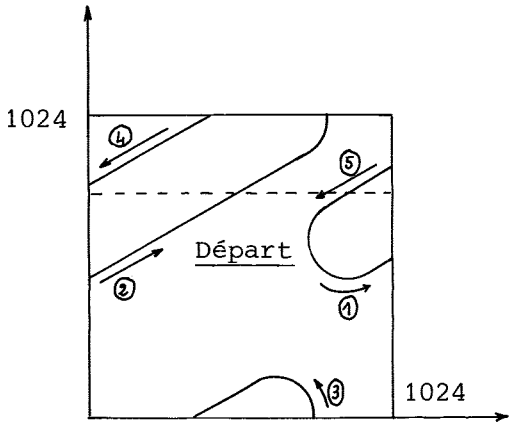
2 - sous GR GT : M se déplace "indéfiniment", ses coordonnées en x/y peuvent aller de -2^{16} à + (2^{16}-1).

Ecran graphique

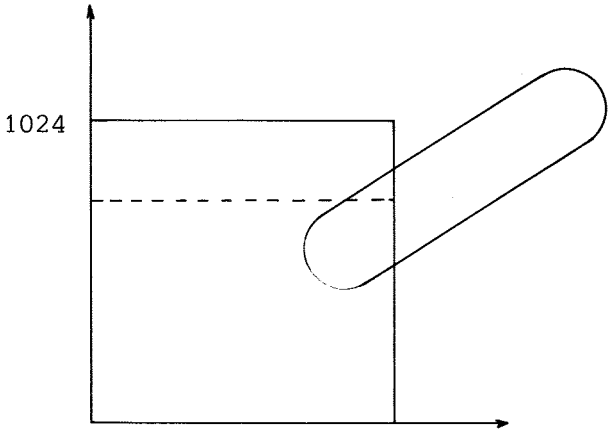


GR 6

G2 G3 GV GW



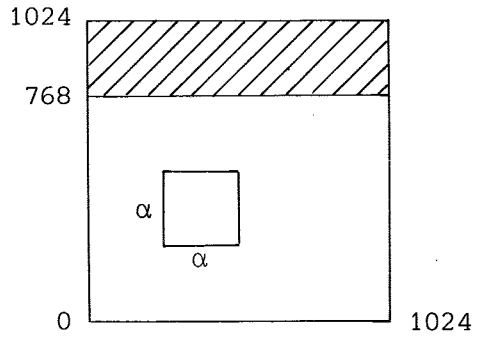
GR GT



2.4.2 - Taille de l'écran Vidéo.

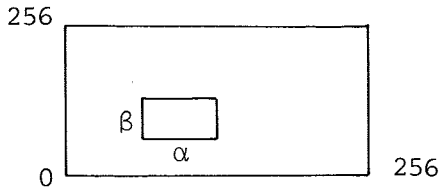
L'écran vidéo fait 256 × 256 points contre 1024 × 1024 pour l'écran graphique. Il en résulte que certains points, distincts sur l'écran graphique, se trouveront confondus sur l'écran vidéo (définition 4 fois moins bonne).

De plus, étant donnée la forme rectangulaire de l'écran Vidéo, ce qui apparaît comme un carré sur l'écran graphique apparaît comme un rectangle sur l'écran vidéo et inversement.



Ecran graphique

carré de $\alpha \times \alpha$



Ecran Vidéo

rectangle $\alpha \times \beta$ $\beta = \frac{3\alpha}{4}$

2.4.3 - Primitives agissant sur la visualisation.

[Z] Effacement écran (visu ; TV) Nota : pour GT, Z initialise l'image TV à noir (ou à blanc) suivant l'état de la primitive "," (voir cette primitive).

[V] Pour GR GT seuls. Basculement de la visualisation entre UL 2 et UL B (initialement, c'est sur l'UL 2 qu'est effectué le traçage).

[U] Pour GR GT seuls. Permet de commander l'écran de visualisation graphique 4014 : Syntaxe : Uα : $\alpha \in \{@,A,B,C,D,G,H,I,J,K,L,Ø,P,Q,R,S,T,W,8,9,:,;\}$

Valeurs de α :

TRACE	Intensité Normale	Intensité forte	Tracé non-mémoire
———	@ ou G	H ou Ø	P ou W
.....	A	I	Q
.—.—.—	B	J	R
- - - - -	C	K	S
— — —	D	L	T

α = 8 Grande taille des caractères alpha-numériques
α = 9 Moyenne " " " "
α = : Petite " " " "
α = ; Très petite " " "

[.] Pour GV, GW, GT sur TV. Cette primitive "point" signifie : tracé blanc (mode implicite).

[,] Pour GV GW GT sur TV. Cette primitive "virgule" inverse la "couleur" du tracé, on obtiendra un tracé { noir si blanc. / blanc si noir.

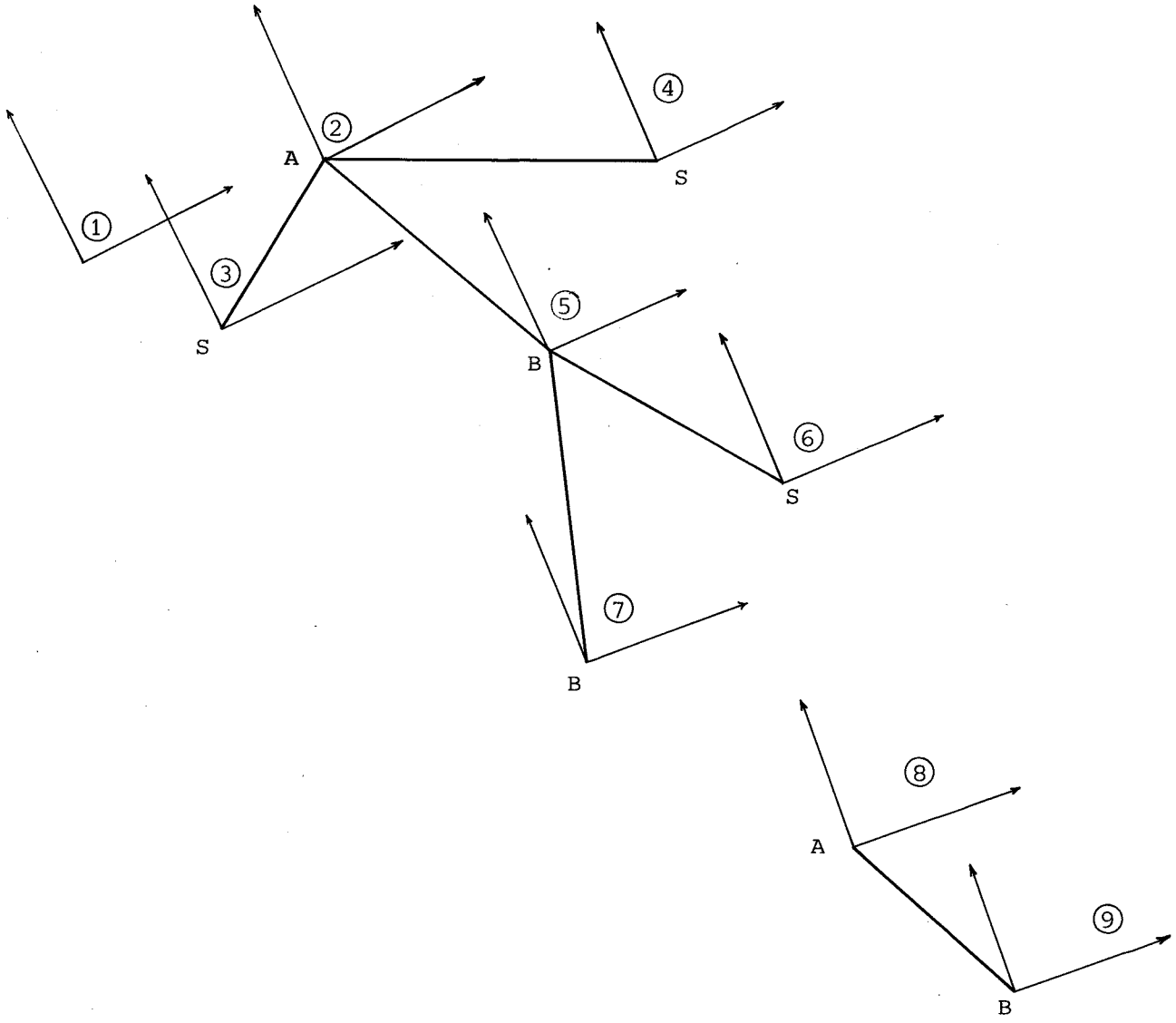
2.4.4 - Dispositifs externes (Hard-Copy, caméra).

H	La primitive H rencontrée dans un programme graphique : . sous GV, GW est ignorée . sous G2, G3, GR, GT provoque le déclenchement du dispositif externe (hard-copy ou caméra), suivi d'une temporisation de 15 secondes.

2.5 - Marquage.

On peut matérialiser les déplacements du point courant M en utilisant les primitives A, B, S.

A	M courant devient l'origine d'un nouveau segment courant (implicitement c'est l'origine absolue).
B	M courant devient l'extrémité du segment courant, le segment courant est tracé, puis M devient l'origine d'un nouveau segment courant.
S	M courant devient l'extrémité du segment courant, le segment courant est tracé, et l'origine reste inchangée.



2.6 - Changements d'unités (à l'intérieur de l'échelle générale définie par Eα).

. changements d'unités explicites : en écrivant [Kk] [Xx] [Yy], K, X et Y resteront inchangés jusqu'à ce qu'on leur affecte une nouvelle valeur. (Rappelons que l'affectation d'une nouvelle valeur à K ne modifie pas X et Y courants).

. Changements d'unités par incrémentation ou décrémentation : on utilise les primitives suivantes, qui peuvent apparaître partout dans un programme.

[X+]	Ux = Ux + k	[Y+]	Uy = Uy + k	[+]	≡ X + Y +
[X-]	Ux = Ux - k	[Y-]	Uy = Uy - k	[-]	≡ X - Y -
[X]	Ux = Ux * k	[Y]	Uy = Uy * k	[*]	≡ X * Y *
[X/]	Ux = Ux / k	[Y/]	Uy = Uy / k	[/]	≡ X / Y /

Exemple : K1 ... X4 ... YB ... K4 ... X* ... Y- ... K2 ... + ...

- X* → Ux = G (16)
- Y- → Uy = 7
- K2 → k = 2 ; Ux et Uy inchangés
- + → Ux = I(18) ; Uy = 9

2.7 - Itérations (répétitions de séquence).

$$\boxed{< \text{séquence} > :: = \% \left\{ \begin{matrix} n \\ @ \end{matrix} \right\} (< \text{séquence} >)}$$

avec $n \in \{ 1, \ldots, 9, A, \ldots Z \}$

@ ≡ (n = valeur courante de la constante k)

Exemples :

..... % 4 (A1B) ≡ A1B A1B A1B A1B

..... K3 ....% @(2B) ≡ 2B 2B 2B

.....% V(% @ (% H (A1 B2))) ≡ .... A1 B2 ..... A1 B2

(V × k × H fois)

Nota : pour GR et GT, @ peut être ⩽ 0, auquel cas la séquence à itérer est sautée :

~~~~~ % @ (~~~~~~~~) ~~~~~
~~~~~

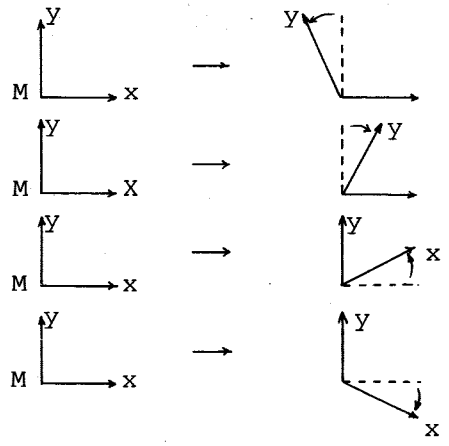
pour avoir @ ⩽ 0 on peut faire par exemple : K8 X1 X - MIX ; MØK ; ... %@(-) (pour MIX ; MØK ; voir gestion pile des constantes § 2.9).

2.8 - Transformation T.

Rappel : De même que toute translation, toute transformation s'applique au référentiel courant.

On dispose d'une transformation T (initialement neutre) que l'on peut à tout moment soit réinitialiser, soit modifier, à l'aide d'une (ou plusieurs) transformation(s) élémentaire(s) prise(s) parmi les suivantes :

RP	Rotation +16° par rapport à M	pourquoi 16° ? parce que $\alpha = 16°$ $\sin(\alpha)=\frac{p}{q}$ $\cos(\alpha)=\frac{p'}{q'}$ avec $p,q,p',q' \in N$
RM	Rotation -16° / M	
R1	Rotation + $\frac{\pi}{2}$ / M	
R3	Rotation - $\frac{\pi}{2}$ / M	
SØ	Symétrie/origine courante (M)	
SY	Symétrie / MY	
SX	Symétrie / MX	
Ø	Rotation +16° MY/M	
A	Rotation -16° MY/M	
H	Rotation +16° MX/M	
B	Rotation -16° MX/M	



Syntaxe :

< initialisation de T > :: = T = [{ RP / RM / = / = / = } [< initialisation de T >]] ;
< modification de T > :: = T [{ RP / RM / = / = / = } [< modification de T >]] ;

Nota : $(RP)^M = (RM)^M = \emptyset$ $(M_{35} = 22_{10})$

$(R1)^4 = (R3)^4 = \emptyset$

Exemple 1 : Tracer un cercle :

```
% M (A1B TRP ;)
```

Exemple 2 : Tracer un carré : les deux programmes suivants sont équivalents

```
1) A 1 B 2 B 3 B 4 B

2) A % 4 (1 B T R 1 ;)
```

On note que la seconde manière dessine un carré quelle que soit la position de l'axe des Y par rapport à l'axe des X, et quel que soit Uy (unité sur l'axe des Y) contrairement à la première manière.

2.9 - Pile des constantes.

On notera K la constante k en cours

X l'unité Ux en cours

Y l'unité Uy en cours

T la transformation T en cours

C (Curseur graphique) la position ABSOLUE du point courant M

1 la coordonnée ABSOLUE sur ØX du point courant M

2 la coordonnée ABSOLUE sur ØY du point courant M

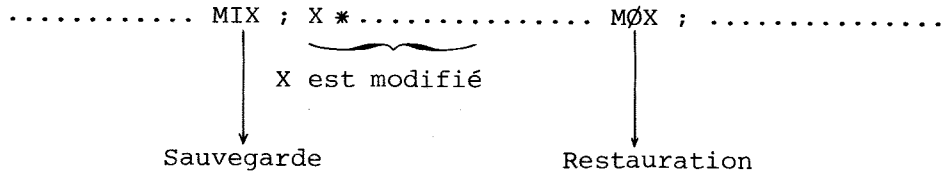
} ∀ la transformation T courante.

A l'"Argument formel" (voir paragraphe 2.13 sa signification).

Syntaxe :

Empilement :: = MI { < constante > } ; de 0 à n constantes I comme IN	Dépilement :: = MØ { < constante > } ; de 0 à n constantes Ø comme ØUT

Exemple : Soit la séquence suivante de sauvegarde et restauration :



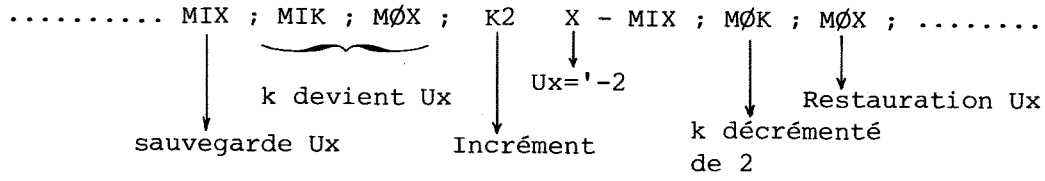
Applications : entre autres applications :

1 - Sauvegarde/restauration d'une constante (cf exemple ci-dessus)

2 - Echange de constantes : soit à inverser par exemple Ux et Uy :

..... MIX ; MIY ; MØX ; MØY ; .....

3 - Calcul sur constante : soit à décrémenter k de 2 unités :



2.10 - Contextes graphiques.

Un contexte graphique est un ensemble = k, C, x, y, T, A, Q_1, Q_2, Q_3 avec k valeur courante de K

C (curseur graphique) coordonnées ABSOLUES du point courant M

x unité Ux courante

y unité Uy courante

T transformation T courante

A Argument formel courant (voir § 2.13 argument formel)

Q_1 Q_2 Q_3 : les trois plans de référence (voir § 3 : langage graphique trois dimensions)

On dispose à tout moment de 31 contextes actifs (1,...,9,A,...,V).

Syntaxe :

Sauvegarde contexte n :: = Wn	Restauration contexte n :: = Rn
W comme Write contexte	R comme Read contexte

Remarques importantes :

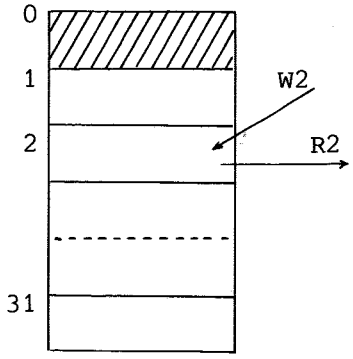
1) R/W n'est pas géré en pile ! ⟹ la séquence W1 W2 R1 R2 n'a rien d'anormal.

2) Faire Rn alors que Wn n'a pas été fait peut provoquer les pires catastrophes y compris une TRAPPE programme.

En effet les ordres R/W utilisent la zone scratch disque (unité logique C), et un contexte lu n'est jamais validé.

Zone scratch

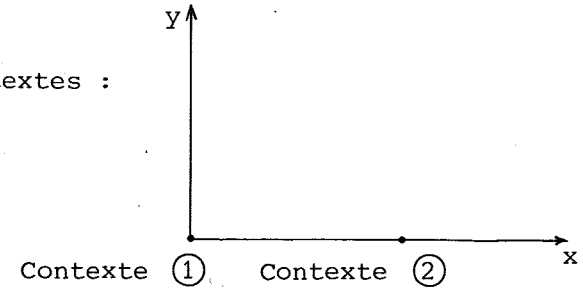
UL C



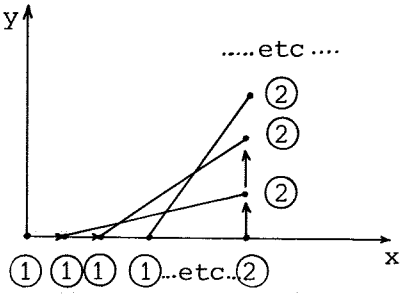
PROGRAMMES GRAPHIQUES

Exemples :

K8 X8 Y8 W1 % 8(1) W2 crée deux contextes :



% 8 (R1 1 A W1 R2 2 B W2) trace 8 traits ainsi



2.11 - Conditions.

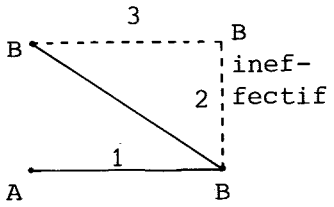
2.11.1 - Autorisation/inhibition de primitives.

"< primitive > ⟶ autorise l'utilisation de < primitive >
:< primitive > ⟶ inverse l'état d'autorisation de < primitive >
Initialement : Toutes les primitives sont autorisées.

Exemples : 1) " < p > .... : < p >

p autorisée p inhibée

2) Séquence A1B : B 2 B : B 3 B ⟶



2.11.2 - Autorisation/inhibition de séquences.

Les primitives 5 6 7 8 9 sont réservées à rendre conditionnelles des séquences. En leur appliquant " ou : (cf ci-dessus) on autorise ou on inhibe les séquences concernées. Une telle séquence doit être placée entre parenthèses, la parenthèse ouvrante étant placée juste derrière le numéro (5 à 9).

Exemple : ..... " 5 5(A1B2B3B4B)..... ⟶ trace un carré

..... : 5 5(A1B2B3B4B)..... ⟶ ne trace rien

la séquence est ignorée

Nota : Si l'on écrit : .....| "A ... :A.....| A / B (séquence S ~) ~~~ la séquence S est sautée

"B

:B

2.12 - Sous-programmes et bibliothèques.

2.12.1 - Définitions sous-programmes et bibliothèques.

< nom > :: = < chaîne de caractères quelconques sauf le ; >

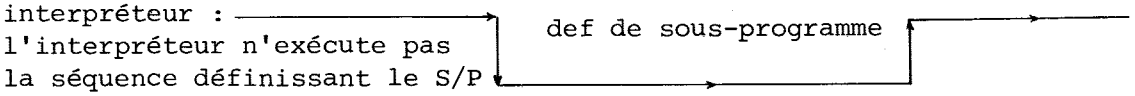
■ *Sous-programme.*

< def. d'un sous-programme > :: = > < nom > (séquence)

Nota : une définition de S/P peut se rencontrer n'importe où dans un programme graphique, mais elle ne sera exécutée que si le S/P est appelé (voir ci-dessous appel de S/P).

Exemple :

.....K8 X8 Y8 A 2 B 3 B > CARRE (A1 B2 B3 B4 B) 1 4 B .....

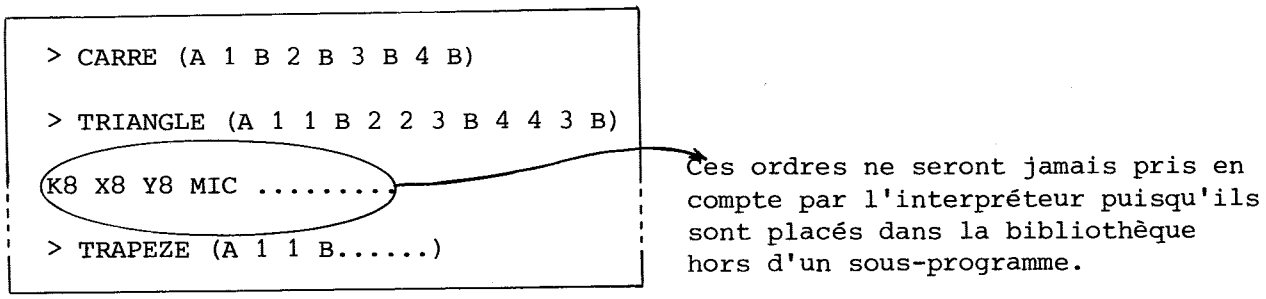


■ *Bibliothèque.*

Item contenant des définitions de S/P. Elle peut avoir deux formats : le format normal et le format spécial.

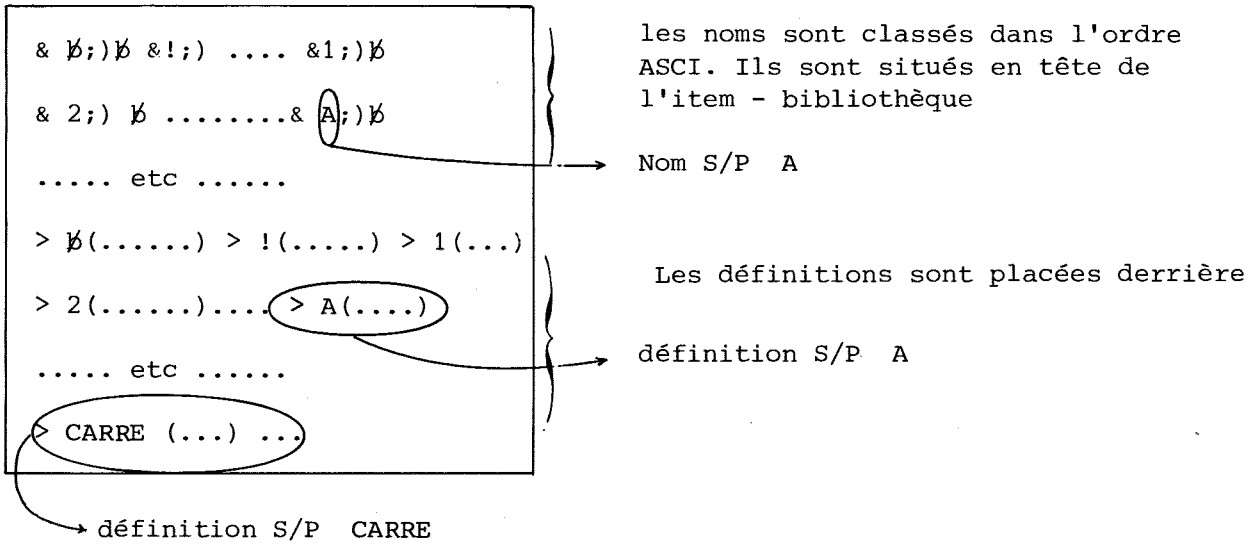
1) Format normal : séquence quelconque de définitions de S/P.

Exemple : bibliothèque GEOMETRIE.



2) Format spécial : elle contient des S/P dont les noms ne contiennent qu'un seul caractère et pouvant être appelés par appel implicite (cf : ci-dessous appel implicite de S/P). Elle peut contenir aussi d'autres S/P comme en format normal.

Exemple : bibliothèque ALPHABET



Nota :-les S/P dont les noms figurent en tête de l'item (& A;)∅ ... etc ..., pourront être appelés par appel implicite ou par & ou par ? } cf=§2.12.3 appel de S/P

- les autres S/P pourront être appelés par & ou par ?

2.12.2 - Appel de bibliothèque.

1) ∅ < nom >;

(dollar)

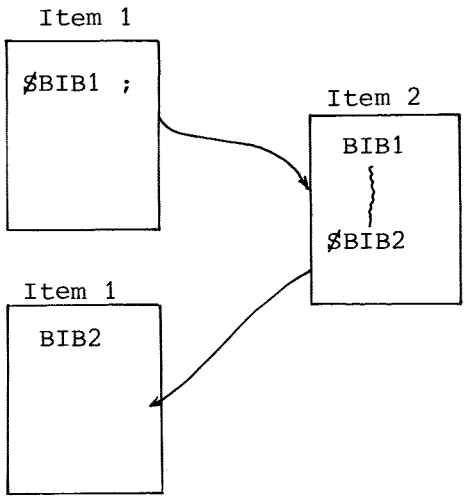
TOUS PROCESSEURS.

Exécuté dans Item 1 ⟶ chargement de la bibli < nom > dans Item 2

Exécuté dans Item 2 ⟶ " " " " " dans Item 1

(Item 2 et Item 1 fonctionnent donc en bascule).

Nota :



① Appel de BIB1 ⟶ chargement de BIB1 dans Item 2

② l'un des S/P de BIB1 référence BIB2 ⟶ chargement de BIB2 dans Item 1.
BIB2 écrase Item 1.
Cela doit avoir été prévu.

2) Processeurs GV GW - Référence d'une bibliothèque avec nom vide.

Ø ;
GV GW

Ceci a pour effet d'inverser l'état du "numérotage des segments"; Initialement, ceux-ci sont numérotés.

3) Processeurs GR GT.

ØTØTØ'A ;
GR GT

L'un des caractères du nom de la bibli est une quote.
⟹ la quote devient le SYSID de l'utilisateur converti en ASCI (voir l'instruction '1E45 de CMS4). Ceci permet d'appeler dynamiquement des bibliothèques différentes.

Dans l'exemple ØTØTØ'A ; devient ØTØTØ2A; si SYSID = 2

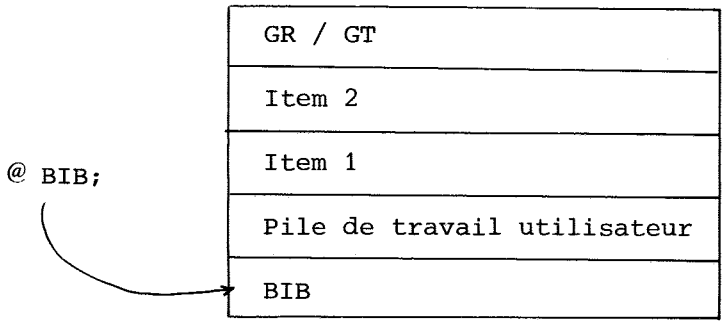
Nota : valable aussi pour # et @.

< nom > ;
GR GT

Rencontré dans Item i, provoque le chargement de la bibli < nom > dans Item i (écrasement), et le "pointeur instruction graphique" est remis à 0 (début d'item). Ceci permet d'avoir des programmes graphiques de taille supérieure à un item (overlay).

@ < nom > ;
GR GT

Provoque le chargement de la bibliothèque < nom > au-delà de la pile de travail utilisateur (cf : schéma de SMC). Cette bibliothèque est obligatoirement de format spécial, ses S/P ne pouvant être référencés que par appel implicite.



Remarque importante :

A l'instant t on ne peut avoir qu'une seule bibliothèque ∅ < nom >; active sauf sous GR / GT où l'on peut avoir simultanément 2 bibliothèques actives : . l'une référencée par ∅ < nom > ; ⟶ Item 1 ou 2

. l'autre " " " @ < nom > ; ⟶ "Item 3"

2.12.3 - Appel de sous-programme.

Trois modes d'appel : explicite, implicite, édition alphanumérique.

- *Appel explicite.*

& < nom > ;	Provoque la recherche dans l'item courant (1 respectivement 2) du S/P de nom < nom >, puis dans l'autre item (2 respectivement 1) si recherche infructueuse.
? < nom > ;	Recherche du S/P de nom < nom > dans l'autre item (2 si 1, 1 si 2) à priori et uniquement.

Exemple : Soit dans l'item courant, le programme suivant :

K8 XA YA ..... > CARRE (A1B2B3B4B) .....

∅ ALPHABET ; ⟶ charge la bibliothèque dans l'autre item.

& CARRE ; ? A ;

↓ ↓

dessine un carré

dessine la lettre A (est équivalent à & A ; si l'item courant ne contient pas de définition de S/P A.)

■ *Appel implicite.*

La bibliothèque contenant le S/P est en format spécial et doit avoir été chargée par ∉ < nom > ; (ou # < nom > ; ou @ < nom > ; si GR GT).

Le S/P peut être référencé

. soit de façon explicite par &α ; ou ?α ;

. soit de façon implicite : la recherche du S/P se fait comme pour ? (2 si 1, 1 si 2).

((((..... α β γ .....))))

au moins 4 niveaux de parenthèses comptés à partir
. de la dernière déclaration d'itération ou
. du dernier appel de S/P ou
. de la dernière conditionnalité.

appel implicite du S/P γ défini en bibliothèque par > γ (.....)

Exemple : ∉B ; B contient des S/P de dessin de lettres.

% 3 ((((A B C))))

4 niveaux

⟹ traçage des lettres A B C trois fois.

Rappel : Pour GR et GT les biblis référencées par @ < nom > ; ne peuvent être utilisées que par appels implicites de leurs S/P.

■ *Edition alphanumérique.* (GR GT uniquement)

S = (((((..... α β γ .....)))))

au moins 5 (et non plus 4) niveaux de parenthèses comptés à partir :

. de la dernière déclaration d'itération, ou
. du dernier appel de sous-programme, ou
. de la dernière conditionnalité.

Ce mode d'appel entraîne l'édition alphanumérique de α, β, γ, ou l'éxécution du S/P α, β, γ, en fonction de l'état de l'édition alphanumérique donné par la primitive J ci-après :

J
GR,GT

. Rencontrée un nombre pair de fois par l'interpréteur, elle entraîne l'interprétation de S comme des appels de S/P.

. Rencontrée un nombre impair de fois, elle provoque l'interprétation de S comme l'ordre d'édition alphanumérique des caractères α β γ (texte édité en utilisant le générateur alphanumérique de la visu).

(fonctionnement en bascule appel S/P / Edition texte).

2.13 - Argument formel.

L'argument formel désigne la dernière séquence définie ainsi :

= (< séquence >)

rencontrée par l'interpréteur (initialement, séquence vide).

Cette séquence ne sera exécutée que lorsqu'elle sera référencée

. dans le même item par ' (Quote)

. de l'autre item par 0 (zéro).

Cette séquence ne sera changée qu'à la prochaine rencontre d'une définition =( ).

Exemple 1.

..... = (& P;) ⟶ argument formel = &P; (exécuter le S/P P)

> P(~~~~ = (B) ~~~~& C;)

> C(A1 ' 2 ' 3 ' 4 ')

.....' ⟶ exécution de l'argument formel &P; ⟶ le S/P &P redéfinit l'argument formel, puis exécute le S/P C. Celui-ci trace un carré.

Exemple 2 : Soit à épaissir les lettres de l'alphabet.
On dispose des S/P A, B, ... , Z dessinant des lettres simple épaisseur, dans la bibliothèque B par exemple.

On fait : ⊄B; ⟶ chargement bibliothèque des S/P A....Z

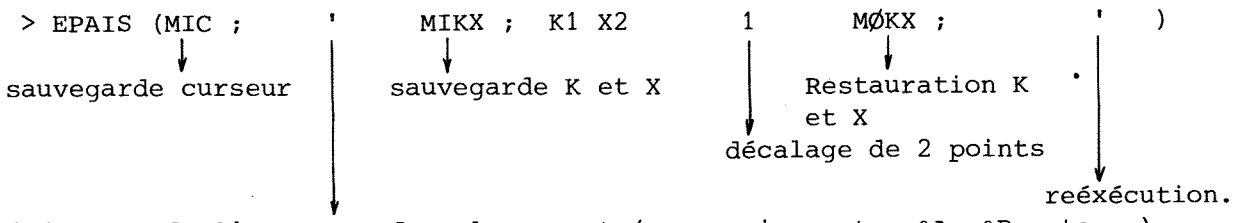
> AA(=(&A;) & EPAIS;) ⟶ def du S/P AA

> BB(=(&B;) & EPAIS;) ⟶ " " " BB

& AA ; & BB ; ⟶ exécution S/P AA, BB, etc.....

(suite page suivante)

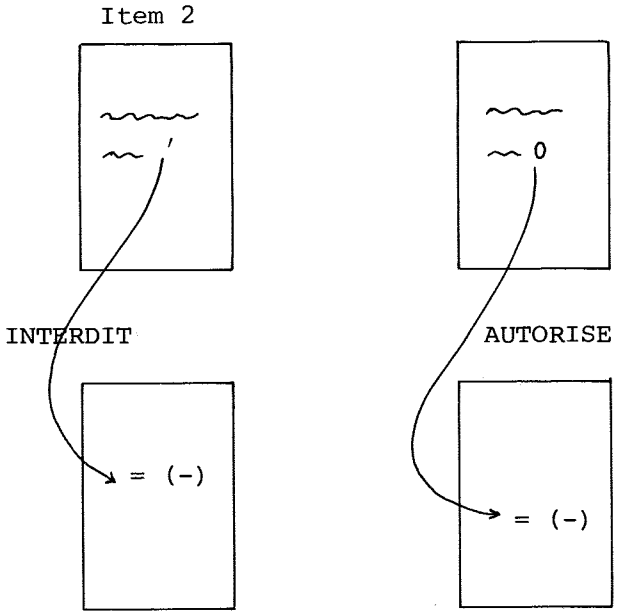
GR 22



⟹ traçage suivant : A B etc.....

Rappel : l'Argument formel peut être empilé-dépilé (MIA ; MØA;)

Appel inter-item. Utiliser 0(zéro) et non pas '(quote).



2.14 - Textes alphanumériques.

< suite de caractères terminée par !
G3 GR GT GW. Pris comme commentaire par G2 et GV

Ce texte sera édité en alphanumérique à partir de l'emplacement courant du curseur graphique, celui-ci sera inchangé.

Nota : Voir aussi appel implicite de sous-programme (§ 2.12.3)

2.15 - Commentaires.

< suite de caractère terminée par ;	Ce texte est pris comme commentaire par l'interpréteur graphique.
TOUS PROCESSEURS	

2.16 - Point d'arrêt pour retour au CCI (?).

C Commande de retour au CCI ⟶ ? revenir par !GØ

Attention : S'il y a un segment en cours (exemple : A 1 C B) celui-ci ne sera pas tracé pour les processeurs G2, G3, GV, GW.

2.17 - Validation/invalidation de l'envoi en ZDC.

D
- pour GR GV GW : fonctionne en bascule (validation/invalidation) initialement : invalidation.
- pour G2 G3 : erreur de syntaxe.
- pour GT : neutre.

Lorsque l'envoi en ZDC est validé, les primitives B et S provoquent l'envoi en ZDC du segment courant (coordonnées origine, coordonnées extrémité).

Exemple :
A .... B ⟶ $((x_1\ y_1), (x_2\ y_2))$ ⟶ ZDC
B .... S ⟶ $((x_2\ y_2), (x_3\ y_3))$ ⟶ ZDC
..... S ⟶ $((x_2\ y_2), (x_4\ y_4))$ ⟶ ZDC
etc ...

2.18 - Interactivité.

I et P pour G2 G3 GV GW

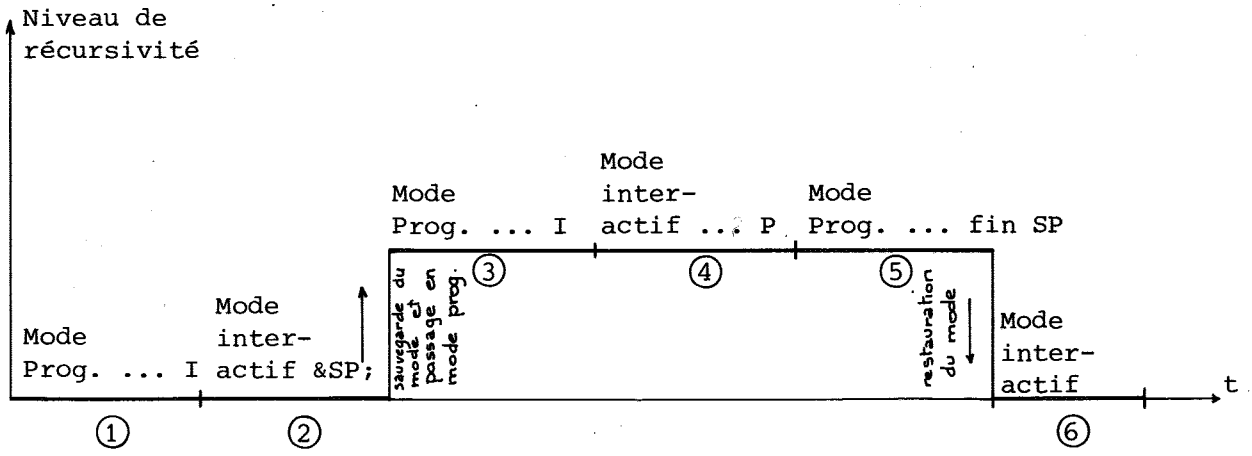
I entraîne le passage en interactif ⟶ le réticule est affiché sur la visu ce qui permet :

1 - de désigner un point de l'écran (X,Y) dans le référentiel absolu

2 - d'entrer au clavier une séquence de programme graphique tout en désignant des points d'écran (X,Y) dans le référentiel absolu. La séquence frappée est insérée dans l'item (à partir de la primitive I ayant déclenché

le passage en interactif ; donc celle-ci est écrasée), et les déplacements du réticule sont convertis en suites de (1,2,3,4) en fonction des valeurs courantes de K,X,Y.

3 - de revenir en mode programme en frappant P.



(1) On est en mode programme ; l'interpréteur rencontre la primitive I → passage en interactif.

(2) On travaille en interactif ; on frappe & SP; le sous-programme SP est éxécuté en mode programme.

(3) Dans ce sous-programme, la primitive I est rencontrée → passage en interactif.

(4) En interactif..... on frappe P → retour mode programme.

(5) Le sous-programme SP s'achève, retour au niveau appelant qui est interactif

(6) On travaille en interactif .....etc.....

D et I pour GV GW.

Si, sous GV ou GW, la primitive D autorisant l'envoir en ZDC, a précédé I, le réticule est simulé par l'entrée ZDC, c'est-à-dire que le mode interactif sera assuré non pas entre programme et visu, mais entre programme et un autre programme, celui-ci envoyant en ZDC des coordonnées de points.

Restrictions : Les segments lus en ZDC entraîneront la génération dans le programme de primitives A,B,1,2,3,4 seulement (contrairement au mode interactif où l'on n'est pas limité à ces primitives).

Retour au mode programme : Lorsqu'un segment de coordonnées origine (0,0) et extrémité (0,0) est récupéré en ZDC, ceci est considéré comme un retour au mode programme, et génère un P dans le programme.

[I] cas de l'utilisation sous GR GT.

Sous GR GT I entraîne la récupération en ZDC des coordonnées d'un seul segment, dont l'extrémité devient brutalement le curseur graphique, sans tenir compte de K,X,Y,T, etc.....

Nota :

1 - Le retour au mode programme est immédiat après chaque segment.

2 - L'item-programme n'est pas modifié, c'est-à-dire que la primitive I n'est pas écrasée, et que rien n'est généré dans le programme.

Exemple : sous GR on exécute le programme suivant : %Z(%Z(I&SP;))... GR va donc lire les segments produits en ZDC par un autre utilisateur (qui peut par exemple faire un dessin sous GR avec primitive D active). A chaque extrémité de segment, GR va exécuter le S/P SP puis lire le segment suivant etc...

2.19 - Retour à l'origine absolue.

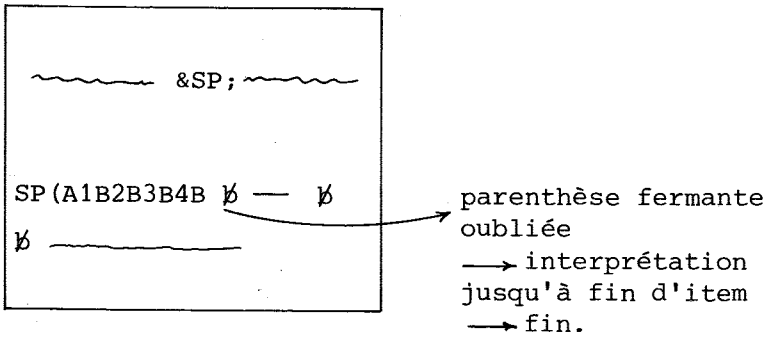
[Ø] Cette primitive (lettre Ø) entraîne le retour à l'origine absolue sans modification de K, X, Y, T.

Application : envoi du segment ((0,0),(0,0)) en ZDC pour provoquer le retour en mode programme (voir primitives D et I sous GV GW). Il suffit d'écrire ØAB (D étant active).

2.20 - Fin de programme graphique.

[F] La primitive F lorsqu'elle est rencontrée provoque la fin d'interprétation par le processeur graphique. Si elle est omise, l'interprétation cessera lorsque l'interpréteur atteindra la fin de l'item, celle-ci pouvant intervenir lors de l'interprétation du programme principal, ou d'un sous-programme.

Exemple :



(rappel : les ⊬ sont neutres pour l'interpréteur).

3 - LE LANGAGE GRAPHIQUE 3 DIMENSIONS.

3.1 - Les plans Q1 Q2 Q3.

En deux dimensions on travaille sur un plan ØXY.

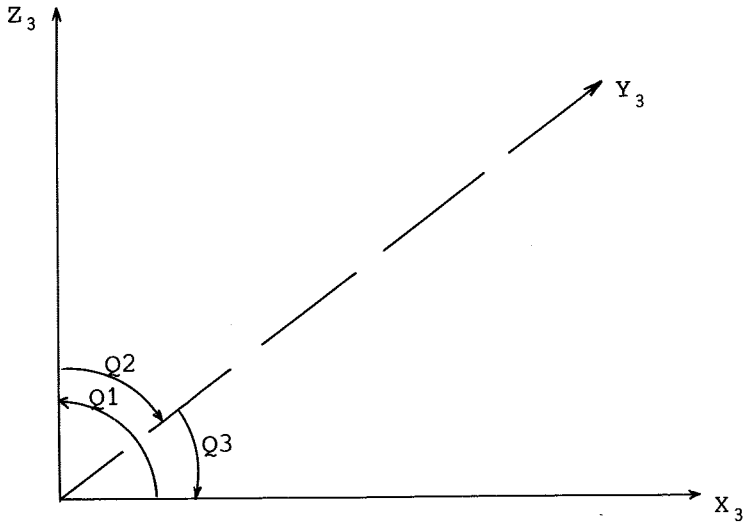
En trois dimensions, on travaille sur trois plans :

Q1 = $ØX_3Z_3$

Q2 = $ØZ_3Y_3$

Q3 = $ØY_3X_3$

tous 3 équivalents à OXY.



3.2 - Particularités du langage graphique 3 dimensions.

Toutes les primitives graphiques utilisées en 2 dimensions sont utilisables, la seule chose à retenir est que toute primitive est relative au plan courant, celui-ci étant spécifié par Q1, Q2, Q3 rencontrées dans le programme (initialement : Q1).

Attention : les piles, contextes, K, E sont COMMUNS.

Exemples :

```
1)   Q1   TRP;   Q2 .....
          ↑      ↑
```

rotation du plan Q1

T est vide initialement sur Q2.

2) Initialisation de K X Y : ce qu'il ne faut pas faire !

. K3 X2 Y3 Q2 X2 Y3 Q3 X2 Y3

sur Q1 :

Ux = 3×2 = 6 ($\emptyset X_3$)

Uy = 3×3 = 9 ($\emptyset Z_3$)

sur Q3 : mêmes remarques

Ux = 3×2 = 6 ($\emptyset Y_3$)

Uy = 3×3 = 9 ($\emptyset X_3$)

sur Q2 :

l'axe des "X" de Q2 : $\emptyset Z_3$ est commun avec l'axe des "Y" de Q1 donc

Ux = 3×2 = 6 ($\emptyset Z_3$)

Uy = 3×3 = 9 ($\emptyset Y_3$)

On obtient donc :

. sur OX_3 : unité = 9

. sur OY_3 : unité = 6

. sur OZ_3 : unité = 6

Il aurait mieux valu écrire :

K3 X3 Y2 Q2 Y3 qui aurait eu le même effet.

3) Initialisation de K X Y ce qu'il faut faire !

K2 X4 Y4 Q2 Y4 ⟶

(ici on veut un espace avec mêmes unités sur les différents axes)

. sur ØX3 : unité = 8

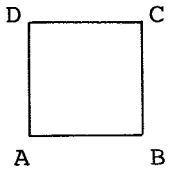
. sur ØY3 : unité = 8

. sur ØZ3 : unité = 8

4) Tracer un cube.

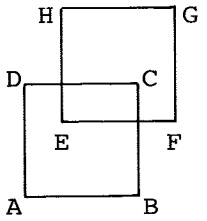
KG XG YG Q2 YG — définition des unités

Q1 A1B2B3B4B



Q2 2 — on se déplace sur ØY

Q1 A1B2B3B4B



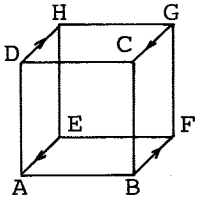
Q2 4B — on trace $\vec{EA}$

Q1 1A Q2 2B — on trace $\vec{BF}$

Q1 2A Q2 4B — on trace $\vec{GC}$

Q1 3A Q2 2B — on trace $\vec{DH}$

4 Q1 4 — on revient au point A (en l'occurrence c'est l'origine).



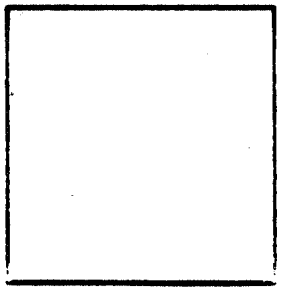
4 - CROSS-REFERENCE - PRIMITIVES GRAPHIQUES/PROCESSEURS GRAPHIQUES.

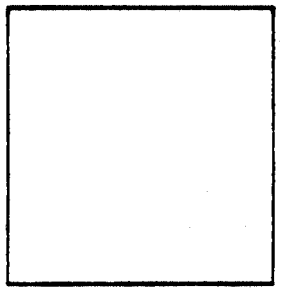
PRIMITIVES		PROCESSEURS							REMARQUES	§
		G2	G3	GV	GW	GR	GT			
Définition d'unité	K	X	X	X	X	X	X			2.2
	X	X	X	X	X	X	X			
	Y	X	X	X	X	X	X			
Echelle générale E						X	X			
Déplacements élémentaires	1	X	X	X	X	X	X			2.3
	2	X	X	X	X	X	X			
	3	X	X	X	X	X	X			
	4	X	X	X	X	X	X			
Primitives agissant sur la visualisation	Z	x	x	x	x	x	x		effacement écran	2.4.3
	V					x	x		basculement visualisation	
	U					x	x		spécifique visu 4014	
	.			X	X		X		tracé blanc sur TV	
	,			X	X		X		inversion couleur tracé TV	
Hard-Copy, caméra H		X	X			X	X		ignoré par GV GW	2.4.4
Marquage	A	X	X	X	X	X	X			2.5
	B	X	X	X	X	X	X			
	S	X	X	X	X	X	X			
Charge-ments d'unités	X+ Y+ +	X	X	X	X	X	X			2.6
	X- Y- -	X	X	X	X	X	X			
	X* Y* *	X	X	X	X	X	X			
	X/ Y/ /	X	X	X	X	X	X			

PRIMITIVES	PROCESSEURS							REMARQUES	§
	G2	G3	GV	GW	GR	GT			
Itérations %	X	X	X	X	X	X			2.7
Transformations T	X	X	X	X	X	X			2.8
Gestion de piles : MI/MØ	X	X	X	X	X	X			2.9
Curseur graphique empilable	X	X	X	X	X	X			2.9
Read/Write contexte R/W	X	X	X	X	X	X		31 contextes (1...9,A...W)	2.10
Conditions "	X	X	X	X	X	X		Autorisation	2.11.1
Conditions :	X	X	X	X	X	X		autorisation/inhibition	2.11.1
Def. de séquence 5,6,7,8,9	X	X	X	X	X	X			2.11.2
Def de sous-prog > (..)	X	X	X	X	X	X			2.12.1
Quote de substitution '					X	X		dans un nom de bibliothèque	2.12.2
Argument formel '	X	X	X	X	X	X		référence à arg. formel	2.13
Argument formel =(-)	X	X	X	X	X	X		def.de l' arg. formel	2.13
Appel bibliothèque $<nom>;	X	X	X	X	X	X		voir particularité GV GW (comptage segments)	2.12.2
Appel bibliothèque #<nom>;					X	X		voir particularité GV GW (comptage segments)	2.12.2
Appel bibliothèque @<nom>;					X	X		voir particularité GV GW (comptage segments)	2.12.2
Appel de sous-programme &<nom>;	X	X	X	X	X	X			2.12.3
Appel de sous-programme ?<nom>;	X	X	X	X	X	X		voir aussi édition αnum GR/GT	2.12.3
Appel de sous-programme ((((...))))	X	X	X	X	X	X			2.12.3
Edition alphanumérique J									
Edition alphanumérique ((((...))))					X	X		interprétation((((..)))) appel S/P ou édition αnum	2.12.3
Edition alphanumérique <texte!	Δ	X	Δ	X	X	X		Δ : pris comme commentaire	2.14

PRIMITIVES	PROCESSEURS							REMARQUES	§
	G2	G3	GV	GW	GR	GT			
Commentaire < texte ;	X	X	X	X	X	X			2.15
Point d'arrêt retour C CCI	X	X	X	X	X	X			2.16
Interactivité I/P/D	X	X	X	X	X	X		réticule et ZDC	2.17
Retour origine absolue Ø	X	X	X	X	X	X			2.19
Fin de programme F	X	X	X	X	X	X			2.20
Q1 Q2 Q3	X	X	X	X	X	X		Trois dimensions	3.1

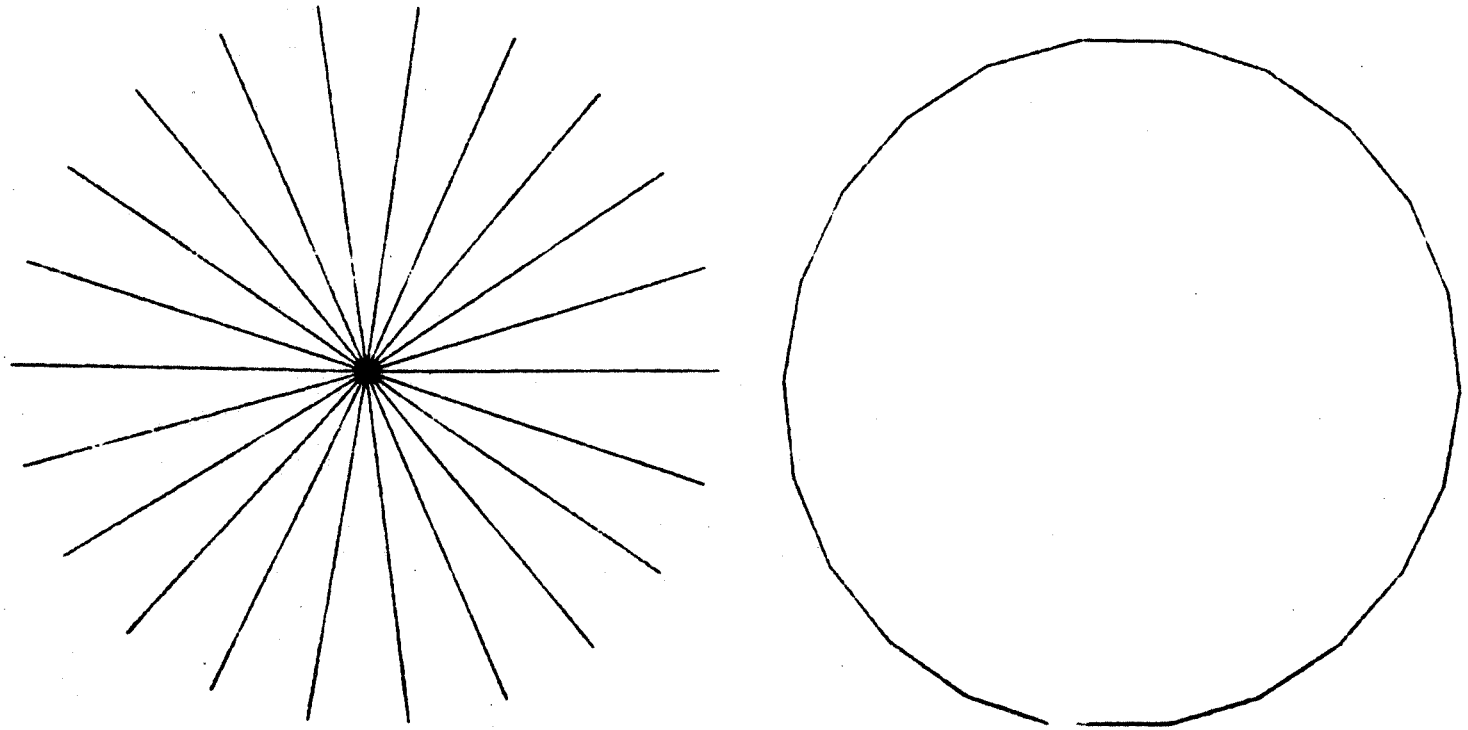
```
<DEUX MANIERES DE TRACER UN CARRE; UH KDXDYD 12 &C1; 111 &C2;

        <LA PREMIERE;
        >C1(A1B2B3B4B)

        <LA DEUXIEME;
        >C2(A %4(1B TR1;))

!GR
```





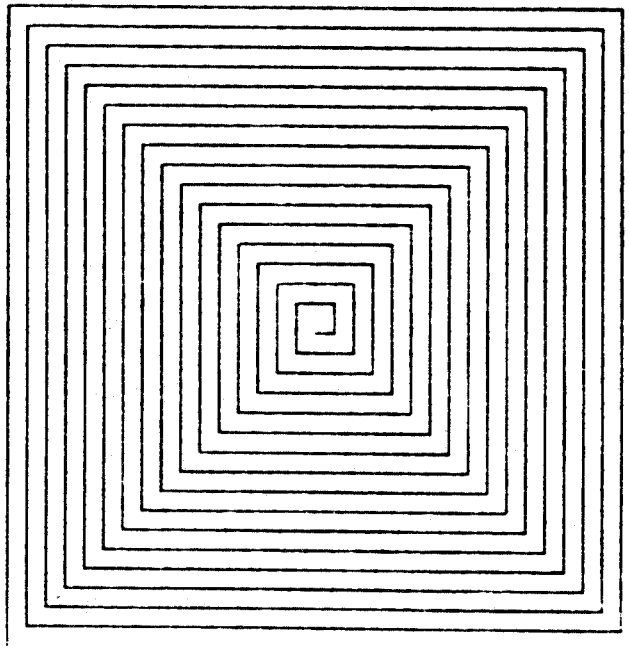
```
!
```

```
<TRACER UNE ETOILE, UN CERCLE . . ;  KFXFYF12 &ETOILE;114X4 &CERCLE

      >ETOILE(A  %M( MIC;1S  TRP; MOC;))
      >CERCLE(A  %M(     1B  TRP;     ))

!GR
```

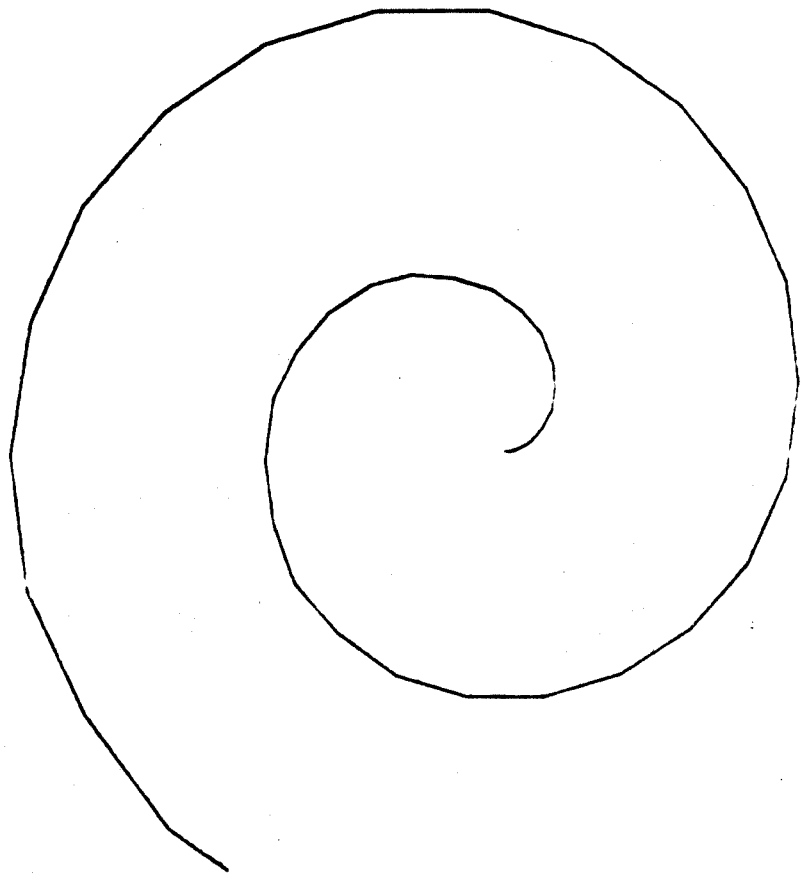


```
< TRACE D'UNE SPIRALE CARREE;
< ---------------------------;

KGXGYG12              < POSITIONNEMENT;
K6X2                  < INCREMENT=6, VALEUR INITIALE=6*2;
A%G(%4(1BTR1;X+))     < BOUCLE DE TRACE ET D'INCREMENTATION;
```

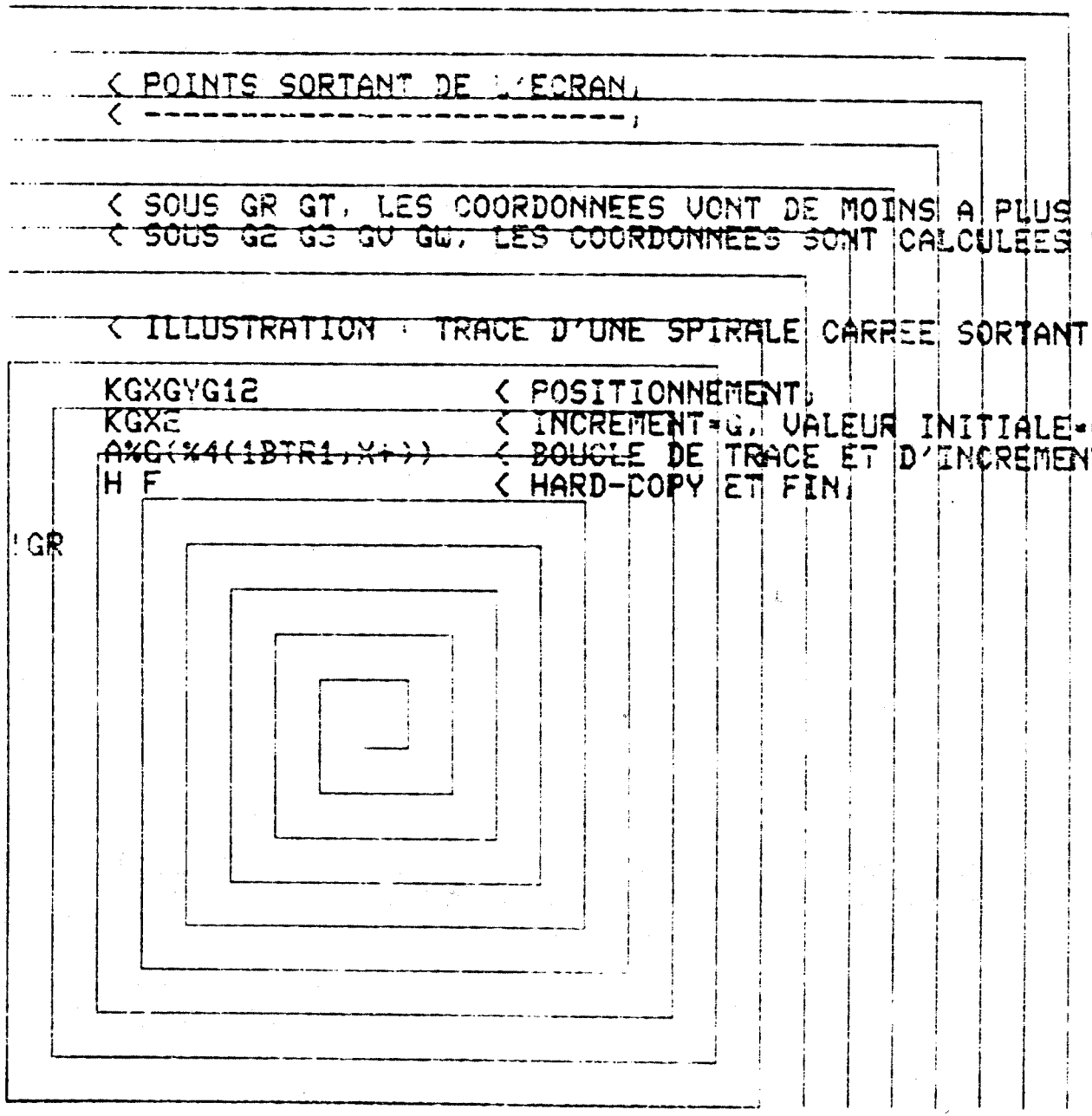
```
!GR
```



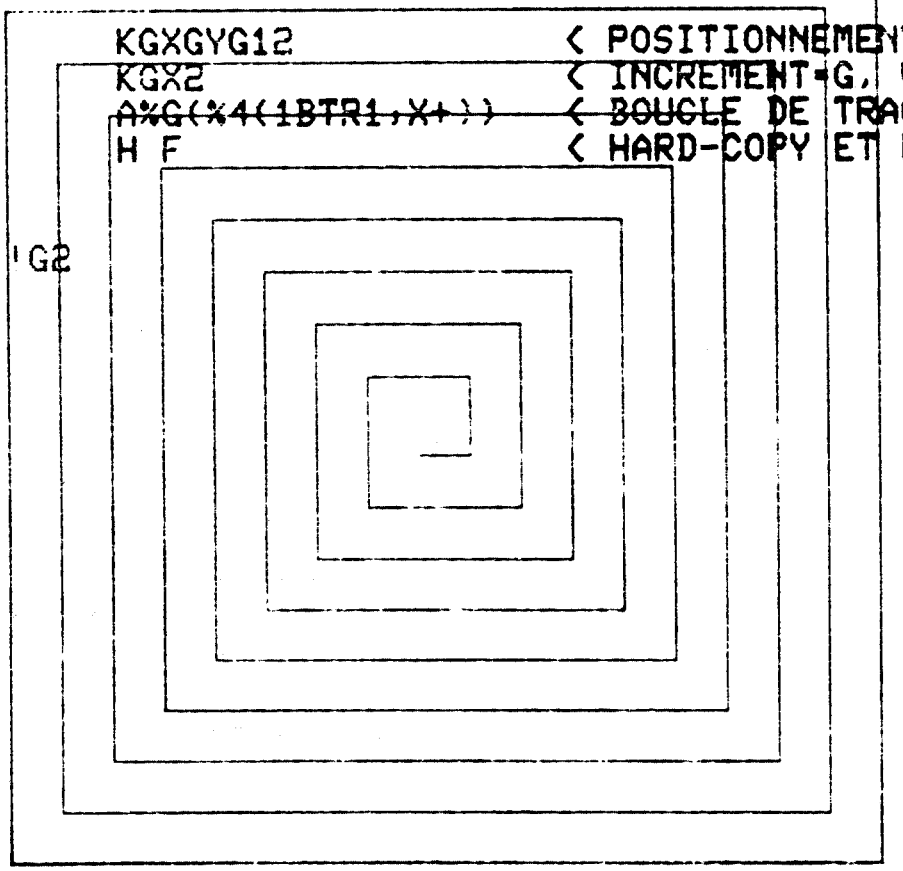
```
< TRACE D'UNE SPIRALE;
< ------------------;

KCXGYG112              < POSITIONNEMENT;
K2X2                   < INCREMENT=2, VALEUR INITIALE=2*2;
A%B(%4(1BTRP;X+))      < BOUCLE DE TRACE ET D'INCREMENTATION;

!GR
```



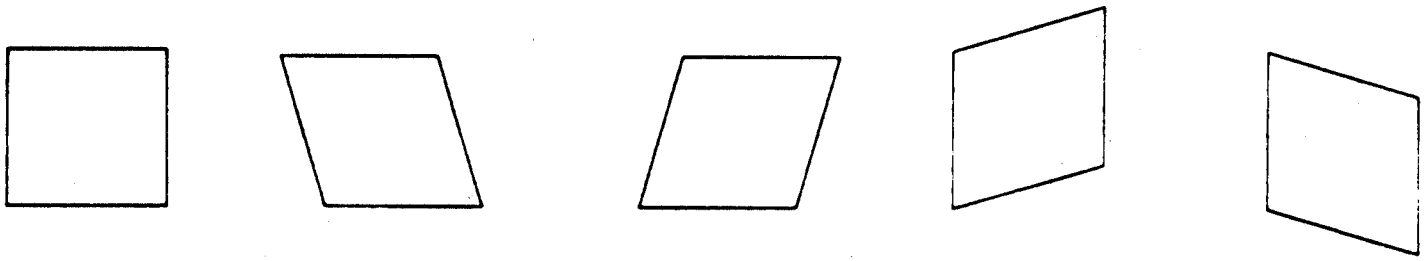
```
< POINTS SORTANT DE L'ECRAN;
< ------------------------;

< SOUS GR GT, LES COORDONNEES VONT DE MOINS A PLUS L'*INFINI*;
< SOUS G2 G3 GV GW, LES COORDONNEES SONT CALCULEES MODULO 1024;

< ILLUSTRATION : TRACE D'UNE SPIRALE CARREE SORTANT DE L'ECRAN;

KGXGYG12                < POSITIONNEMENT;
KGX2                    < INCREMENT=G, VALEUR INITIALE=G*2;
A%G(%4(1BTR1,X+))       < BOUCLE DE TRACE ET D'INCREMENTATION;
H F                     < HARD-COPY ET FIN;

!GR
```



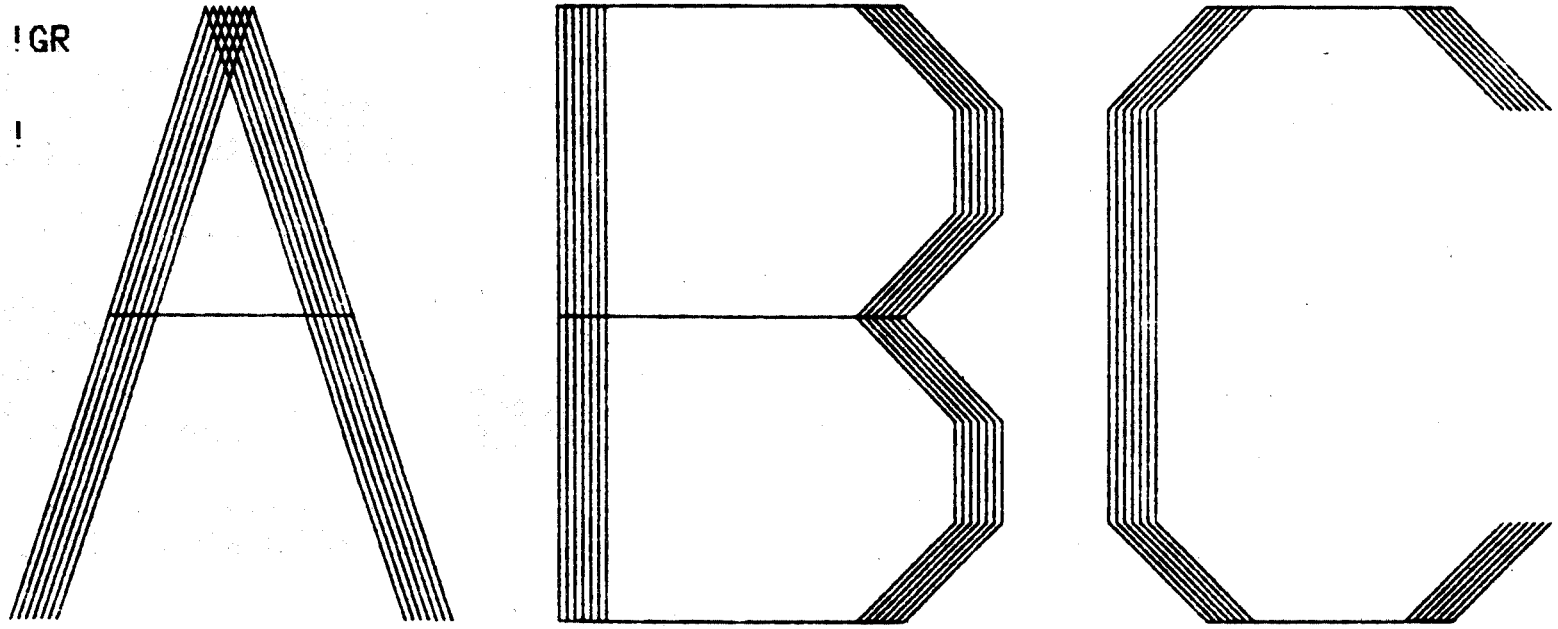
```
< POINTS SORTANT DE L'ECRAN;
< ------------------------;

< SOUS GR GT, LES COORDONNEES VONT DE MOINS A PLUS L'"INFINI";
< SOUS G2 G3 GV GW, LES COORDONNEES SONT CALCULEES MODULO 1024;

< ILLUSTRATION : TRACE D'UNE SPIRALE CARREE SORTANT DE L'ECRAN;

KGXGYG12                < POSITIONNEMENT;
KGX2                    < INCREMENT=G, VALEUR INITIALE=G*2;
A%G(%4(1BTR1,X+))       < BOUCLE DE TRACE ET D'INCREMENTATION;
H F                     < HARD-COPY ET FIN;
```



```
          <TRANSFORMATIONS O,A,H,B;

KAXAYA2 &C;T=O; &C;T=A;  &C; T=H;  &C; T=B; &C;
          >C(A1B2B3B4B T=; 11)

!GR
```

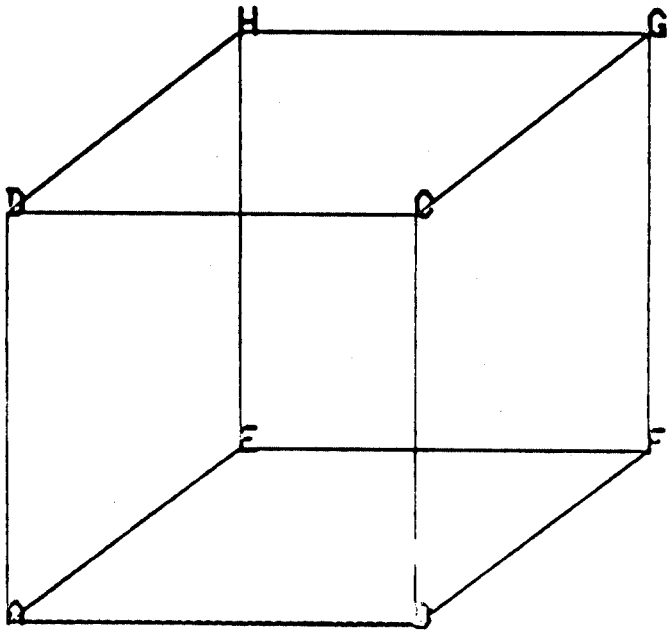


```
     <UTILISATION  ARGUMENT FORMEL  PILE DES CONSTANTES  CONTEXTES;;

K9X7Y72  $B;    <UNITES, CADRAGE, CHARGEMENT BIBLIOTHEQUE,
MIK;K7 W1 MOK;  <SAUVEGARDE DE K ET SPECIFICATION DE N POUR S/P EPAIS;
                <N EST SAUVEGARDE DANS K DU CONTEXTE 1;

&AA;&BB;&CC;   <...ETC...;

>AA( =(&A;)&EPAIS;)
>BB( =(&B;)&EPAIS;)
>CC( =(&C;)&EPAIS;)

<ON EPAISSIT N FOIS;
>EPAIS(W2 R1 %@(R2    MIC; ' MOC; MIKX; K5X1  1 MOKX;W2)11111)

!GR

!
```



```
< TRACE D'UN CUBE;
< ---------------;

KGXGYG 1 Q2YG          < SPECIFICATION DES UNITES ET POSITIONNEMENT;
U8                     < SPECIFICATION TAILLE DES CARACTERES ALPHA;
&CABCD;                < FACE A B C D;
&CEFGH;                < FACE E F G H;
&CARET;                < ARETES EA BF GC DH;

< SOUS-PROGRAMMES;

>CABCD(Q1 A1B2B3B4B)
>CEFGH(Q2 2 &CABCD;)
>CARET(Q2 <E!4B<A! Q1 1A<B! Q2 2B<F! Q1 2A<G! Q24B<C! Q1 3A<D! Q2 2B
<H!)
```

!GR



PROCESSEUR TV

1 Fonction

Le processeur TV permet :

- de faire des opérations sur une image numérique, celle-ci étant visualisée en permanence sur écran TV, c'est l'image résidente. Ceci est vrai sur T 1600, pour le Solar, voir § 8

- de faire des opérations entre cette image et une autre image placée en "mémoire scratch" et qu'on désignera indifféremment par les termes "image scratch" ou "image de travail".

- de créer, supprimer, mettre à jour toute image numérique dans l'espace des noms et valeurs.

Par exemple :

- opérations ne concernant que l'image résidente : inversion, transposition, transformation matricielle, suppression des points isolés, etc...

- opérations entre image résidente et image scratch : opérations logiques (ØU logique, ET logique, ØU exclusif), opérations arithmétiques (addition et soustraction mot à mot).

- opérations sur l'espace des noms et valeurs (bibliothèque d'images) : chargement à partir de la bibliothèque d'une image en image résidente, suppression d'une image en bibliothèque, stockage d'une image en bibliothèque.

Nota : en bibliothèque, une image est l'ensemble { nom, valeur }, le nom étant celui donné à l'image par l'utilisateur lors de la création, la valeur étant l'image numérique elle-même.

Cette énumération des fonctions du processeur TV n'est pas exhaustive, mais on trouvera ci-après une présentation détaillée de toutes les commandes disponibles sous TV.

2 Schéma d'utilisation

Le schéma d'utilisation de TV figure à la page suivante, et on en donne ici un commentaire.

Lorsqu'on appelle TV, l'espace mémoire de l'utilisateur est conforme à ce qui a été dit dans la présentation de SMC (§ schéma de SMC, espace mémoire utilisateur), c'est-à-dire qu'on y trouve :

- le processeur TV occupant la zone "processeur courant"
- une zone de travail utilisée par le processeur TV, occupant les zones Item 1 et Item 2 : il faut donc noter que lors de l'appel de TV, les items courants éventuels occupant Item 1 et Item 2 sont perdus, et doivent donc être sauvegardés au préalable si nécessaire (voir la commande IS de SMC).
- l'image résidente. Cette zone mémoire, de 4 K mots de 16 bits est visualisée en permanence sur écran TV lorsque l'utilisateur travaille (si plusieurs utilisateurs utilisent TV, ils se partagent la mémoire dans le temps et par conséquent chacun ne verra apparaître son image résidente que par intermittence. Ceci est vrai sur T 1600 ; pour le Solar, voir § 8).

Le dialogue de l'utilisateur avec le processeur TV se fait au moyen des commandes de TV présentées ci-après. TV s'utilise de préférence en mode conversationnel ; une utilisation en mode batch est cependant possible (mais pas habituelle). Ces commandes permettent de réaliser les fonctions énumérées plus haut, et permettent donc notamment de travailler sur l'image résidente, l'image scratch et la bibliothèque d'images. En outre, elles permettent l'entrée caméra (ce qui est entré par caméra est analysé , chargé en image résidente et visualisé dynamiquement sur écran TV), et l'interaction graphique (voir commande M).

Tout ceci concerne la télévision numérique sur T 1600. C'est également le cas pour tout ce qui suit jusqu'au paragraphe 7. Le paragraphe 8 est consacré aux particularités de la télévision numérique Solar par rapport au T 1600.

Schéma d'utilisation du processeur TV

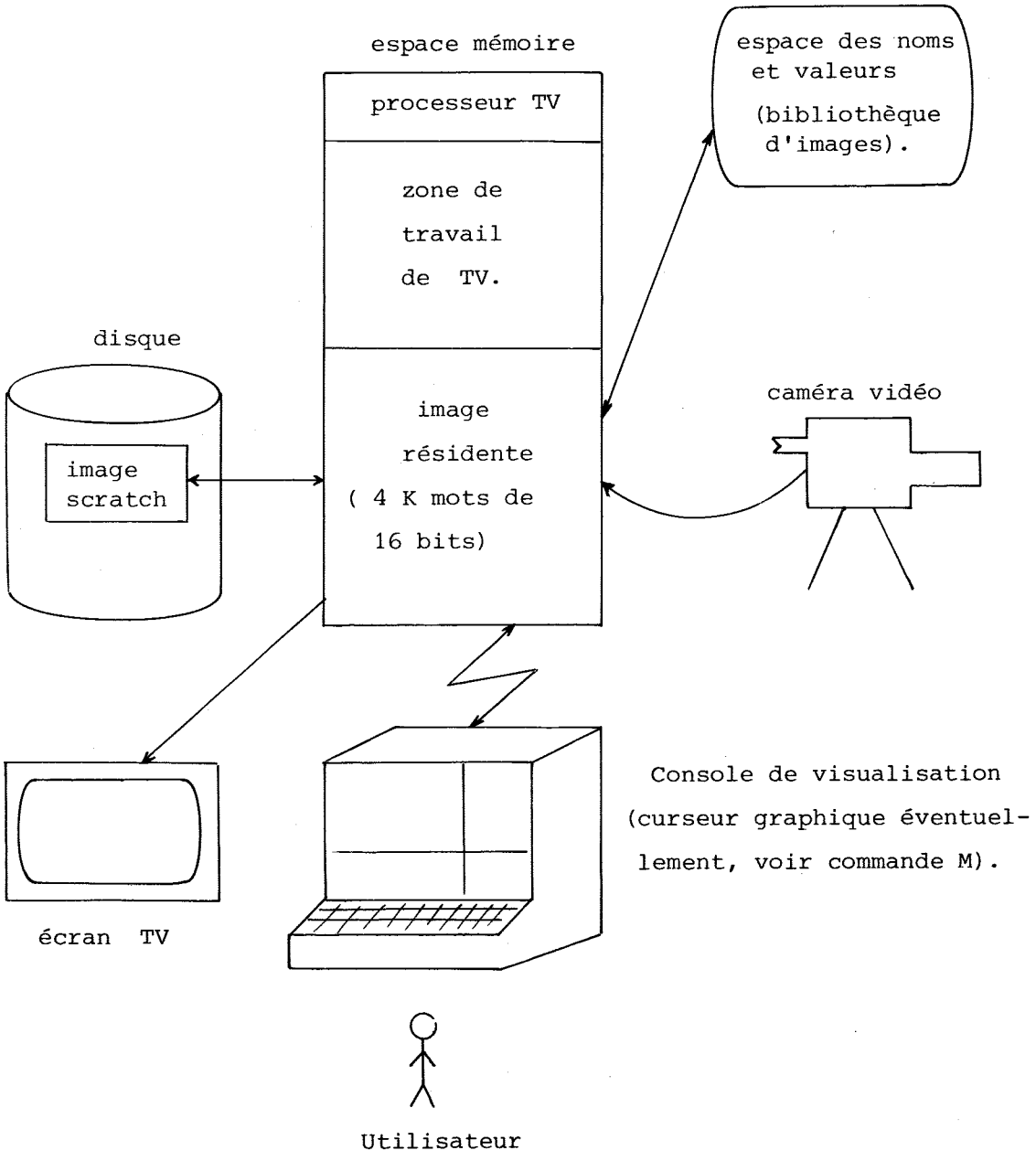


Image numérique = 256 x 256 points ; 1 point est blanc (bit à 1) ou noir (bit à 0).

3 Les commandes de TV

On trouvera à la page suivante un schéma représentant l'enchaînement des commandes de TV.

Comme on le voit sur le schéma, il y a sous TV trois niveaux de commandes, et donc trois niveaux de dialogue :

- niveau 1 : appel image.
- niveau 2 : traitement image (ou opérations sur image).
- niveau 3 : interaction graphique (utilisation du curseur graphique).

Ces trois niveaux doivent être bien distingués, car une même commande a une signification différente suivant qu'elle est utilisée à l'un ou l'autre de ces niveaux.

On peut résumer ainsi les fonctions réalisables sous chacun des trois niveaux de commandes :

1°) appel image :

A ce niveau, on va placer une image en image résidente, de l'une des façons suivantes :

- à partir de l'entrée caméra (commade V) ou
- à partir de la bibliothèque d'images (commande D) ou
- en spécifiant explicitement une valeur numérique (commande R) ou
- en conservant inchangée l'image résidente courante (commande E).

L'image résidente ayant été ainsi chargée, on va pouvoir la traiter.

2°) traitement image :

A ce niveau, on peut faire toute une série d'opérations sur l'image résidente, entre l'image résidente et l'image scratch, plus des opérations diverses.

On sort de ce niveau

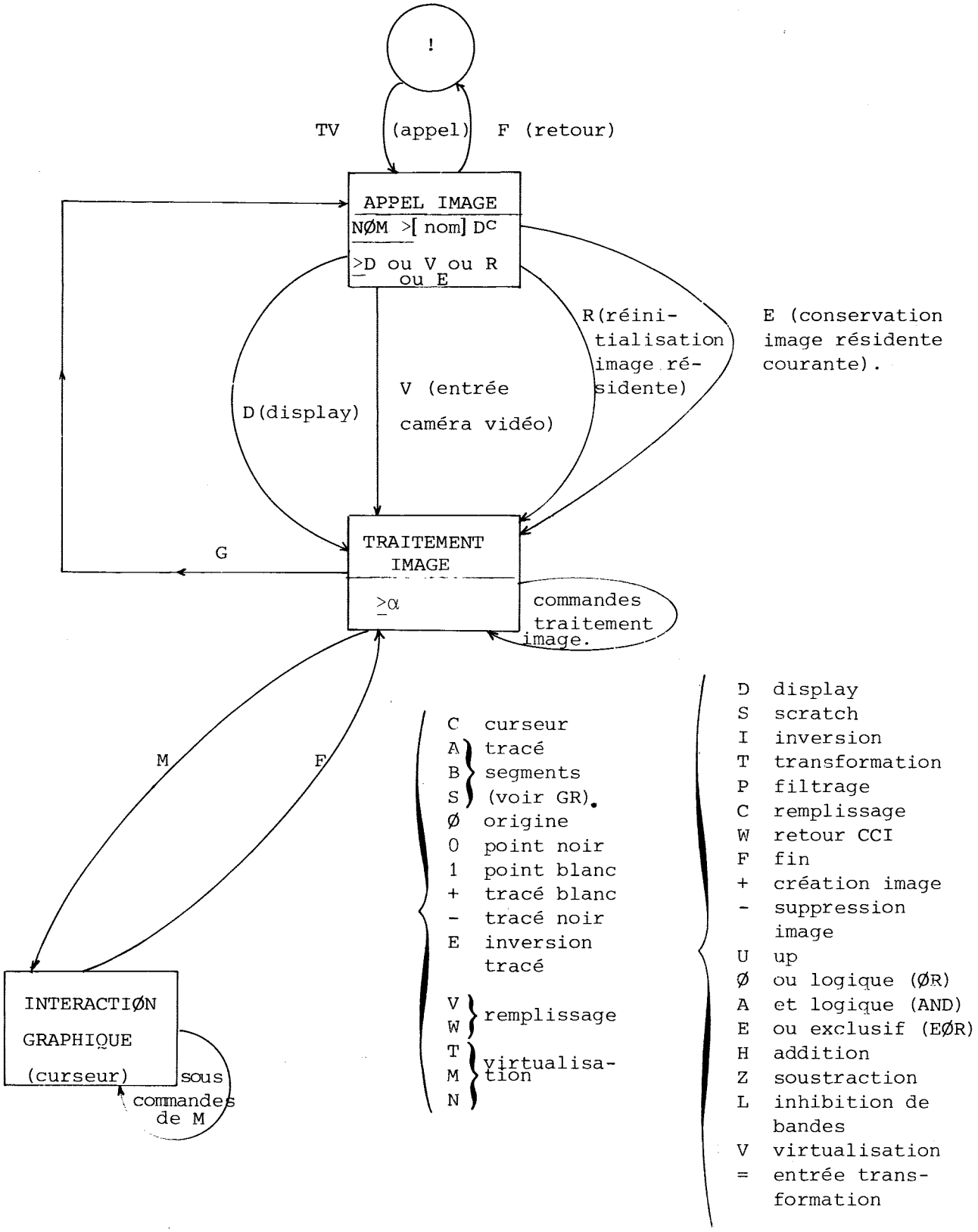
- soit pour remonter au niveau appel image par la commande G.
- soit pour descendre au niveau interaction graphique par la commande M.

3°) interaction graphique :

A ce niveau, on n'opére que sur l'image résidente, par différentes commandes : traçage de points ou de segments, remplissage de contours etc..

On ne peut sortir de ce niveau que par la sous-commande F, qui fait remonter au niveau traitement image.

Schéma des commandes disposnibles sous TV



Nota : toute commande est interruptible par un alt-mode.

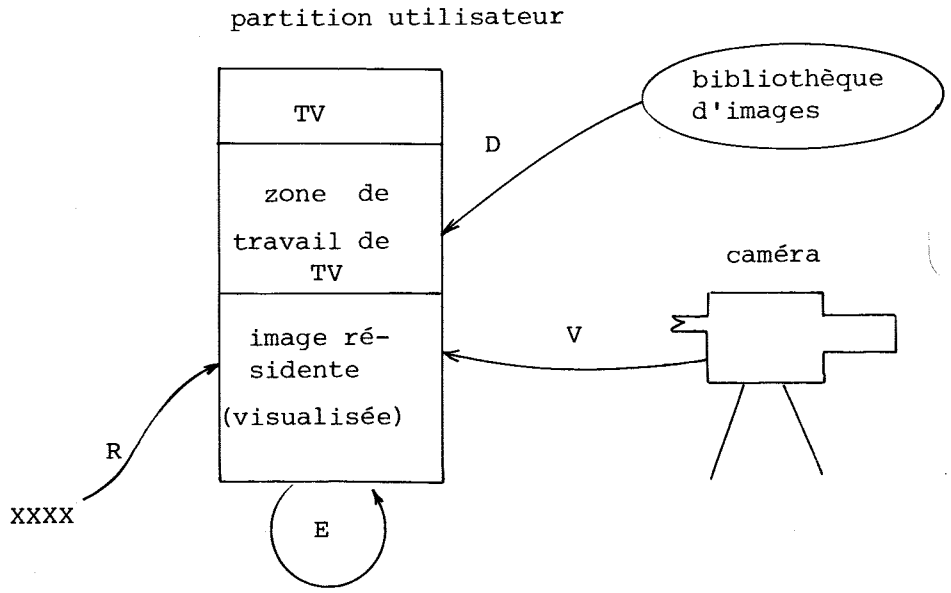
4 Les commandes "appel image"

On arrive au niveau de commande "appel image" :

- soit de GE ("!") sous lequel on a appelé le processeur TV.
- soit de TV lui-même, niveau"traitement image" sous lequel on a frappé la commande G.

Le dialogue se déroule comme suit :

! TV	→	appel de TV (si on vient de GE)
NØM > [nom image] D^C	→	spécification éventuelle d'un nom.
>	→	ici, on peut frapper : D, V, R ou E.



4.1 Commande D: display image

! TV

NØM > nom image D^C	→	nom de l'image à charger à partir de la bibliothèque.
> D	→	display ; l'image est visualisée
>	→	on est maintenant au niveau traitement image.

Ne pas spécifier de nom d'image n'a pas de sens avec la commande D puisque dans ce cas, le chargement à partir de la bibliothèque est impossible, on obtient donc un message d'erreur, et ensuite, TV redemande un nom ; de même s'il n'existe pas d'image en bibliotèque portant le nom spécifié :

! TV

NØM > SPIRYLE D^{C}

> D

? ? → erreur, pas d'image portant ce nom.

NØM > SPIRALE D^{C} → TV redemande un nom.

> D → Ø K : cette image existe, elle est chargée.

> D = 03AC → si on redemande display, TV donne le nombre de points de l'image (sur 4 chiffres hexadécimaux).

>

4.2 Commande V : entrée image par caméra

! TV

NØM > [nom image] D^{C} → spécification éventuelle d'un nom.

> V → V : entrée caméra.

Ici, l'image résidente est l'image entrée par caméra. Elle est "dynamique" et le dialogue est suspendu. Quand l'image est au point (après réglage de distance, exposition, etc), on la "fige" en frappant un alt-mode et le dialogue reprend.

> → on est maintenant au niveau traitement image.

Pour l'entrée caméra, on n'est pas obligé de spécifier de nom d'image. Le nom qui est servi (vide ou non-vide) n'a pas de tout le même sens que pour la commande D vue précédemment. Ici, il a l'effet suivant : l'image résidente a pour nom courant le nom servi (éventuellement vide). Ensuite, au niveau traitement image, on pourra par exemple stocker l'image résidente en bibliothèque, où elle sera rangée sous son nom courant.

Exemple :

On veut rentrer et stocker en bibliotèque plusieurs images rentrées par caméra. On procédera ainsi :

! TV

NØM > IMAGE 1 D^C → nom courant de l'image résidente

> V → entrée caméra.

ici, entrée caméra ; on fige l'image par un alt-mode et le dialogue reprend

> + → ici, on est au second niveau, on catalogue l'image résidente sous le nom courant qui est IMAGE 1.

> G → retour au niveau appel image

NØM > IMAGE 2 D^C → nom courant de l'image résidente

> V → entrée caméra

alt-mode

> + → même chose avec IMAGE 2

> G → retour au niveau appel image

NØM > → etc...

4.3 Commande R : réinitialisation image résidente

! TV

NØM > [nom image] D^C → comme pour V, c'est seulement le nom courant à donner à l'image résidente.

> R → réinitialisation demandée .

FØND = XXXX → servir un nombre hexadécimal de 4 chiffres (= 16 bits = 1 mot). Ce mot sera dupliqué dans l'image résidente

exemples :

FØND = 0000 image noire

FØND = FFFF image blanche.

FØND = F0F0 image barrée verticalement.

On a 2^{16} possibilités.

REC ? N → toujours répondre N. Sans signification.

> → on est maintenant au niveau traitement image.

4.4. Commande E: conservation de l'image résidente

! TV

NØM > [nom image] D^C → comme pour V et R, c'est seulement le nom courant à donner à l'image résidente

> E → conservation demandée

FØND = XXXX → les 4 chiffres hexadécimaux servis sont ignorés par TV mais on doit quand même les donner.

REC ? N → toujours répondre N. Sans signification.

> → on est maintenant au niveau traitement image.

La commande E est utilisable dans beaucoup de cas, notamment dans le cas suivant donné en exemple : on veut tracer un dessin avec sortie vidéo par le processeur GT et ensuite, cataloguer l'image obtenue en bibliothèque. Il suffit de faire ceci :

! IL

NØM > PRØGRAMME D^C → chargement du programme graphique

! GT → interprétation par GT

tracé du dessin par GT avec sortie vidéo.

! TV

NØM > IMAGE ⌀ EXEMPLE D^C → nom courant à donner à l'image résidente

> E → conservation de l'image résidente

FØND = 0000

REC ? N

> + → niveau traitement image : l'image résidente est stockée en bibliothèqye sous le nom courant : IMAGE ⌀ EXEMPLE. Puis on sort de TV par F.

> F

!

5 Les commandes "traitement image"

5.1 Commande D : display

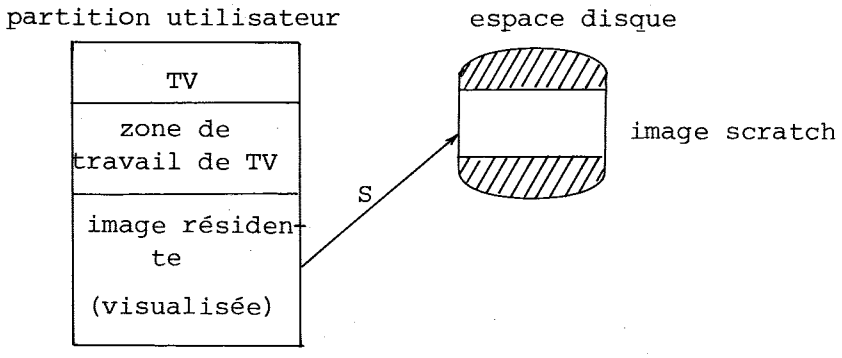
L'image résidente est inchangée. Si l'on fait D plusieurs fois de suite, TV affiche à partir de la deuxième fois le nombre de points de l'image sur 4 chiffres hexadécimaux, comme on l'a vu dans un exemple pour la commande D du niveau appel image.

5.2 Commade S : envoi en scratch

La commande S (scratch ou save), provoque la sauvegarde en image scratch de l'image résidente (qui reste inchangée). Elle est utile notamment pour préparer une opération faisant intervenir deux images (image scratch et image résidente) : il suffira d'en placer une en image scratch et l'autre en image résidente comme dans l'exemple suivant :

```
!TV
NØM > IMAGE 1 DC          →  chargement, à partir de la bibliothèque de IMAGE 1.
> D                       →  display (et passage au niveau traitement image).
> S                       →  image résidente est sauvée en image scratch.
> G                       →  retour au niveau appel image
NØM > IMAGE 2 DC          →  chargement de IMAGE 2
> D                       →  display (et passage au niveau traitement image).
>                         →  ici, on est prêt à faire une opération entre IMAGE 1 et IMAGE 2.
```

Le schéma ci-après illustre la commande Scratch.



5.3 Commande I : inversion

L'image résidente est inversée, c'est-à-dire que tous les points à 1 (blanc) sont remplacés par des 0 (noir) et inversement. Deux commandes I successives redonnent l'image résidente initiale.

5.4 Commande T : transformation

L'image résidente est transformée. TV opère point par point, de bas en haut et de droite à gauche. La transformation effectuée est la suivante :

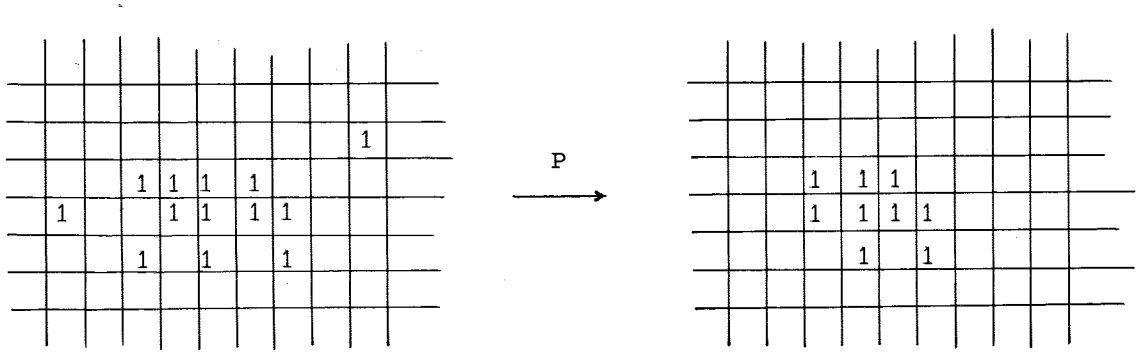
- si l'on a défini une matrice de transformation et une translation (ce qui est possible par la commande = décrite ci-après), alors l'image résidente subit cette transformation.

- si l'on n'a pas défini de transformation, alors TV effectue une transposition de l'image (échange lignes et colonnes).

5.5 Commade P : "filtrage"

Cette opération se déroule par balayage de l'image résidente de bas en haut et de droite à gauche. Chaque ligne (ligne 'FF, ligne 'FE, ..., ligne 1, ligne 0) est analysée point par point. Tout point à 1 rencontré est supprimé (autrement dit mis à 0) si ses voisins verticaux sont à 0. Ce qui permet, entre autres, d'éliminer sur une image les points isolés

Exemple :



5.6 Commande C : remplissage

Cette opération se déroule par balayage de l'image résidente de bas en haut et de droite à gauche. Chaque colonne est analysée point par point.

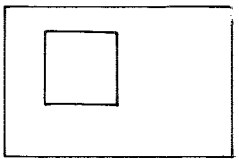
Tout point P (i,j) = 0 est mis à 1 (c'est-à-dire à blanc) si P (i, j+1) = P (i+1, j) = 1.

Exemples :

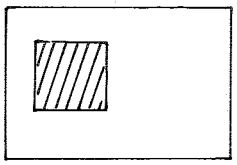
0	0	1
0	1	0

C →

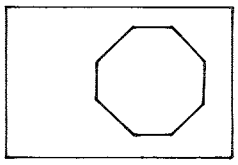
0	1	1
0	1	0



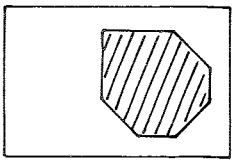
C →



(hachures et traits noirs symbolisent des surfaces et traits à 1).



C →



Nota :

- la commande C permet de remplir l'intérieur d'un contour "simple" On peut aussi utiliser les sous-commandes V et W de la commande M. Voir ci-après.
- pour le remplissage d'un contour quelconque, voir le processeur KØ.

5.7 Commande W : retour au CCI

```
! TV
  ⌇
  ⌇
> W                          ⟶   retour au CCI.

? ! G Ø D^C                  ⟶   retour par ! G Ø.

>
```

5.8 Commande F : retour à GE

La commande F (fin) permet de sortir de TV et de revenir au dialogue avec GE ("!").

```
! TV
  ⌇
> F                          ⟶   fin du dialogue avec TV

!
```

L'image résidente n'est pas détruite, elle peut être récupérée par TV lui-même ou un autre processeur de traitement d'images numériques.

5.9 Commande + : création image en bibliothèque

L'image résidente est cataloguée en bibliothèque sous le nom courant qu'on lui a donné en répondant à la dernière question "NØM >" posée par TV. Si ce nom est vide, la création est refusée. Si ce nom est non-vide, la création est effectuée. Si le nom existait déjà en bibliothèque, son ancienne valeur est perdue ("écrasée") et remplacée par la nouvelle valeur (image résidente).

On trouvera des exemples d'utilisation de cette commande aux paragraphes 4.2 et 4.4 (commandes V et E du niveau appel image).

Remarques : quand on fait + avec un nom d'image, on détruit l'ancienne image portant ce nom si elle existe. Si l'on veut savoir avant de faire + si le nom qu'on veut utiliser pour stocker l'image résidente n'existe pas déjà, il suffit d'essayer la commade D (display) sur ce nom au niveau appel image. Si la commande est refusée, c'est que le nom n'existe pas.

5.10 Commande -: suppression image en bibliothèque

Cette commande supprime en bibliothèque l'image dont le nom = nom courant de l'image résidente, c'est-à-dire le nom spécifié en réponse à la dernière question "NØM >" posée par TV. Si ce nom est vide ou si aucune image ne porte ce nom, la suppression est refusée.

Nota :

si l'on veut mettre à jour une image en conservant son nom, il n'est pas nécessaire de la détruire avant de la recréer, puisque, comme on l'a vu dans la commande +, la destruction de l'ancienne image est automatique avant sa re-création sous le même nom.

5.11 Commande U : "Up"

En T 1600, cette commande sert à deux choses :

1°) rappel image scratch : l'image scratch est rechargée en mémoire en zone image résidente. Cette opération est faite en fonction des "topographie" et "inhibition" courantes (voir ci-après § 5.14) et de la réponse donnée à la question "ØRG =" posée par TV (cf : ci-après).

2°) coloration image résidente : dans ce cas, la commande U ne concerne que l'image résidente dont elle force une valeur dans le 1er mot (masque de couleur de l'image). Cette valeur étant donnée par l'utilisateur :

≀
≀
> U

NØM >[nom] D^C ⟶ nom courant à donner à l'image résidente

ØRG = X L L M ⟶ réponse sur 4 chiffres hexadécimaux

>

1°) si X ≠ E et ≠ F : c'est un rappel de l'image scratch ; traité dans ce paragraphe.

2°) si X = E ou = F : c'est une coloration de l'image résidente ; voir ci-après § 7.

En Solar, la commande U a quelques particularités qui seront traitées lorsqu'on abordera la télévision numérique Solar. Dans le présent paragraphe, on ne parle que de la commande U, T 1600, rappel image scratch.

La réponse à la question "ØRG = " posée par TV est un nombre hexadécimal de 4 chiffres : X L L M avec, dans le cas qui nous intéresse :

X ≠ E et ≠ F (seuls les 2 premiers bits de X sont utiles ici)

LL = n° de ligne (de 00 à FF c'est-à-dire de 0 à 255)

M = n° de mot dans la ligne (de 0 à F c'est-à-dire de 0 à 15).

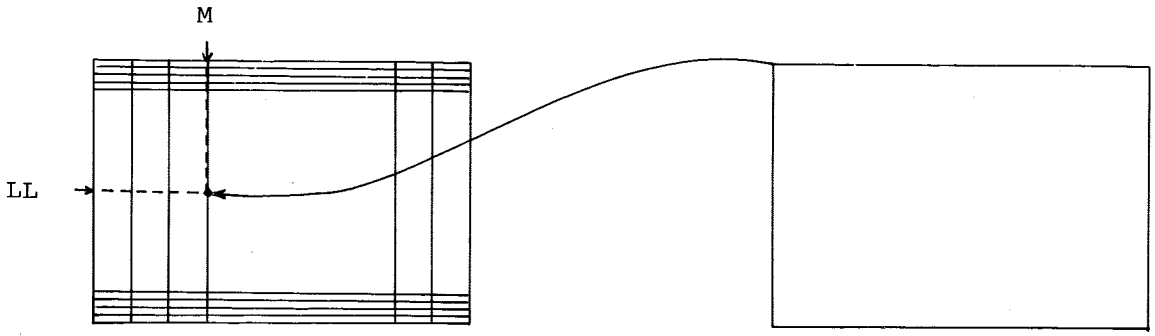


image résidente image scratch

Ici, seuls les 2 premiers bits de X, x_0 et x_1 sont utiles, et on supposera $x_2 = x_3 = 0$.

X = 0 ($x_0 = x_1 = 0$) signifie : prendre l'image scratch telle quelle, c'est-à-dire du premier point au dernier point.

. X = 8 ($x_0 = 1$, $x_1 = 0$) signifie : prendre l'image scratch du dernier point au premier point (c'est-à-dire à l'envers).

. X = C ($x_0 = 1$, $x_1 = 1$) signifie : prendre l'image scratch à l'envers au niveau du mot (c'est-à-dire du dernier mot au premier mot), et à l'endroit au niveau du bit dans le mot (premier bit du mot au dernier bit du mot).

L'image scratch, traitée selon le mode spécifié par X est chargée en image résidente à partir du point spécifié par LL et M avec

. LL = n° de ligne (voir schéma ci-avant)

. M = n° de mot dans la ligne (premier bit de ce mot).

Exemple :

```
! TV
NØM > BØTTICELLI ⌀ 5 DC          ⟶   chargement image
> D
> S                               ⟶   envoi en scratch
> U                               ⟶   rappel image scratch
NØM > DC
ØRG =  0000                       ⟶   l'image apparaît inchangée
> U                               ⟶   rappel image scratch
NØM > DC
ØRG = 8000                        ⟶   l'image apparaît à l'envers sur l'écran
> U
NØM > DC
ØRG = C000                        ⟶   l'image apparaît à l'envers globalement
                                      mais les points sont à l'endroit au ni-
> F                                   veau du mot.
!
```

Ce qui a été dit dans ce paragraphe sur la réponse à donner à la question "ØRG =" vaut également pour les commandes que nous allons voir maintenant qui sont les opérations faisant intervenir image résidente et image scratch.

5.12 Commande Ø, A, E : opérations logiques or, and, exclusive or.

L'opération demandée est effectuée entre image résidente et image scratch, avec résultat en image résidente. On voit donc immédiatement sur l'écran le résultat de l'opération et l'image scratch reste inchangée. Comme pour la commande U :

- TV demande un nom : si on en donne un, il devient le nom courant de l'image résidente, ce qui permet ensuite, si on le désire, de cataloguer l'image obtenue en bibliothèque (voir la commande +).

- TV demande "ØRG =". Lui donner une réponse pour préciser le mode d'utilisation de l'image scratch (voir le paragraphe précédent).

Exemple 1

> G	→	niveau appel image
NØM > BØTTICELLI ∅ 5 D^C	→	chargement à partir de la bibliothèque
> D		
> S	→	niveau traitement ; envoi image en scratch.
> E	→	exclusive or (ou exclusif)
NØM > D^C	→	pas de nom courant.
ØRG = 0020	→	ou exclusif de l'image avec elle-même, avec un décalage de 2 lignes. On obtiendra un "contour" (ce n'est pas un contour au sens des processeurs extracteurs de contours : KØ, ES, EZ).
>		

Exemple 2

Une application du ou logique : épaississement d'un dessin, ou d'un titre, tracé par exemple par GT. On a obtenu une image, contenant un dessin dont le trait ne fait qu'un point d'épaisseur. On veut l'épaissir pour obtenir une image sur laquelle le tracé fera 3 points d'épaisseur. Il faut donc épaissir de 2 points en lignes et de 2 points en colonnes. En lignes, c'est facile, puisque ØRG permet de décaler en lignes. En colonnes, ce n'est pas possible puisqu'on ne peut décaler que d'un multiple de 16 bits (1 mot). Pour épaissir de 1 point en colonnes, il faudra donc transposer l'image, épaissir de 1 point en lignes et transposer à nouveau. Ce qui donne le dialogue suivant avec TV pour épaissir de 2 points en lignes et en colonnes.

! TV

NØM > TITRE D^C → chargement image

> D

> S → envoi en scratch

> Ø → ou logique avec elle-même

NØM > D^C

ØRG = 0010 → décalée d'un point en ligne (vers le bas)

> Ø → ou logique

NØM > D^C

ØRG = 0FF0 → avec décalage de -1 point en ligne (vers le haut)

> T → transposition

> S → et même chose, mais on part d'une image transposée.

> Ø

NØM > D^C

ØRG = 0010 → 1 point en ligne (vers le bas)

> Ø

NØM > D^C

ØRG = 0FF0 → -1 point en ligne (vers le haut)

> T → nouvelle transposition

> S → envoi en scratch de l'image finale

> U → qu'on rappelle par U en donnant un nom courant.

NØM > TITRE D^C

ØRG = 0000

> + → ce qui permet de stocker en bibliothèque l'image résidente sous le nom spécifié.

> F

!

5.13 Commandes H et Z : opérations arithmétiques

Ces opérations s'utilisent comme les opérations logiques vues précédemment. L'effet de ces commandes est le suivant :

- commande H : addition mot à mot (16 bits par 16 bits) de image résidente et image scratch, image résidente recevant le résultat. L'éventuel report est perdu pour chaque addition.

- commande Z : soustraction mot à mot (16 bits par 16 bits) de image résidente et image scratch, image résidente recevant le résultat. L'éventuel report est perdu pour chaque soustraction.

Exemple :

```
≷
> H                          →    addition (mot à mot)

NØM > D^C                  →    pas de nom courant

ØRG = 0000                 →    image scratch prise telle quelle.

>
```

5.14 Commande V et L : virtualisation et inhibition

Lorsqu'on fait des opérations comportant une utilisation par TV de l'image scratch (rappel image scratch, opération logique, opération arithmétique), celle-ci est utilisée

- selon ce qu'on a spécifié en réponse à "ØRG =" comme on l'a déjà vu.

- selon ce qu'on a éventuellement spécifié en définissant une virtualisation (par commande V) et des inhibitions éventuelles (par commande L).

Ce dernier point nécessite les commentaires suivants : une image est découpée en 32 bandes horizontales numérotées 0,1,2,...,9,A,B,...,U,V. On peut :

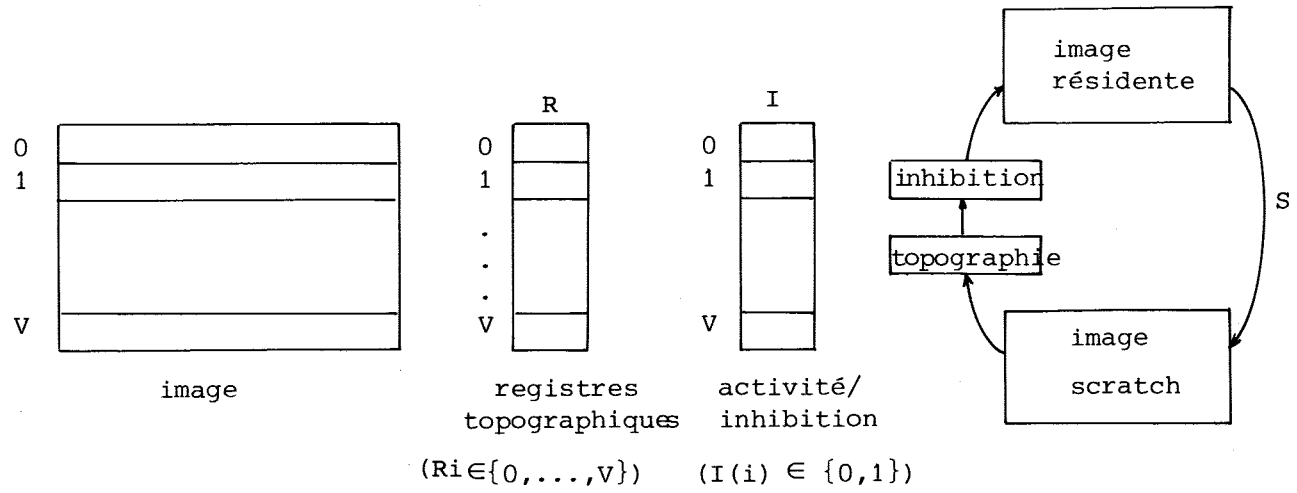
1°) par la commande V, modifier la topographie de l'image, en agissant sur les registres topographiques figurant sur le schéma ci-après. Initialement, c'est-à-dire quand on appelle TV et tant qu'on n'a pas utilisé la commande V, le registre topographique i est associé à la bande i, c'est-à-dire que les bandes de l'image sont dans l'ordre habituel. Si on affecte au registre topographique i la valeur j, alors la bande j remplacera la bande i. On verra par la suite le dialogue avec TV, commande V.

Exemples :

- on met tous les registres topographiques à 0 : l'image virtuelle sera formée de 32 fois la bande 0.

- on met dans le registre topographique i la valeur V-i (c'est-à-dire dans les registres 0 à V les valeurs V, U, T, ..., 1,0) : l'image virtuelle deviendra l'image initiale avec inversion de l'ordre des bandes.

2°) par la commande L, inhiber ou non chaque bande de l'image. Initialement, c'est-à-dire quand on appelle TV et tant qu'on n'a pas utilisé la commande L, toutes les bandes sont actives. L'inhibition d'une bande est provoquée par la mise à 1 de son bit d'activité. On verra par la suite le dialogue avec TV, commande L.



Le schéma qu'on vient de présenter illustre le fait que lorsque l'image scratch est lue pour satisfaire une commande, elle est exploitée en fonction de la topographie et des éventuelles inhibitions.

Pour utiliser V et L, on procède comme suit:

```
≀
≀
> V                              →  commande V (niveau traitement image).

0 1 2 3 4 5 ... 9 A ... V      →  TV liste les 32 numéros des registres
0 0 1 1 V A ...                   topographiques, et l'utilisateur leur
>                                 affecte les valeurs qu'il veut comme le
≀                                 montre cet exemple.
≀
≀
> L                                commande L (niveau traitement image).
```

0 - F = F080

10 - 1F = FC00

TV demande deux nombres hexadécimaux de 4 chiffres chacun. Soit au total 8 chiffres hexa donc 32 bits.
Si le bit i est à 0, la bande i est inhibée. S'il est à 1, elle sera active. Dans l'exemple ci-dessus, les bandes actives seront les bandes 0,1, 2,3,8,16,17,18,19,20,21.

Exemple :

On veut éliminer sur une image les deux premières et les deux dernières bandes pour ne garder que les bandes 3 à 29. Il suffit de faire ceci :

! TV

NØM > IMAGE D^{C} → chargement de l'image

> D

> S → envoi en scratch

> G → retour au niveau appel image, pour réinitialiser l'image résidente à 0 (image noire)

NØM > D^{C}

> R

FØND = 0000

REC ? N

> L → pour inhibition

0 - 15 = 3 FFF → des bandes 0, 1

10 - 1F = FFFC → et 30, 31

> U → rappel image scratch

NØM > IMAGE D^{C} → on donne un nom courant ; seules seront rappelées les bandes 2 à 29.

ØRG = 0000

> + → on catalogue l'image courante.

> L → et on rétablit l'activité de toutes les bandes, pour la suite...

0 - F = FFFF

10 - 1F = FFFF

>

Si l'on avait voulu par exemple éliminer les bords droit et gauche de l'image, il aurait suffit de la transposer, de faire les mêmes opérations que celles ci-dessus sur l'image transposée, puis de retransposer l'image, mais seulement après avoir rétabli l'activité de toutes les bandes, car la commande T tient compte des inhibitions.

5.15 Commande = : entrée transformation

On a vu que la commande T permet d'effectuer une transformation de l'image, qui sera :

- une transposition si on n'a pas défini de transformation.

- une transformation si on a défini une transformation au moyen de la commande =.

L'utilisation de cette commande donne lieu au dialogue suivant, dont chaque point est expliqué par la suite.

> = ⟶ commande = (niveau traitement image).

M - CLASSE X, Y = abcd

T ✱ T = I ? { Ø / N

M ij = v

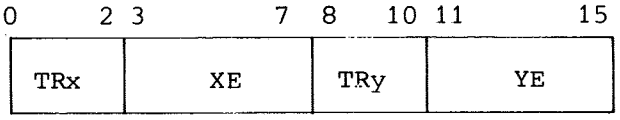
1/2 = { Ø / N

> T ⟶ application de la transformation courante à l'image résidente.

>

1°) M - CLASSE X,Y = abcd

abcd sont 4 chiffres hexadécimaux (donc 16 bits) ayant la signification suivante :



- TRx (bits 0 à 2) et TRY (bits 8 à 10) sont respectivement translation en X et translation en Y, exprimés en nombre de points (TRx : nombre de points en ligne ; TRy : nombre de points en colonnes).

- XE (bits 3 à 7) et YE (bits 11 à 15) définissent respectivement un pas sur l'axe des X et un pas sur l'axe des Y (voir le schéma).

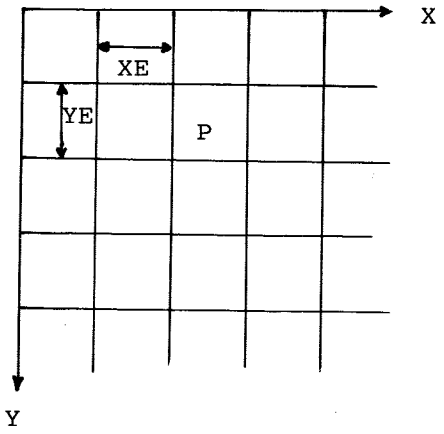


image TV

Pour tout point P de coordonnées (x,y), on aura :

$$x = Ex \times 2^{XE} + Rx$$

$$y = Ey \times 2^{YE} + Ry$$

avec

$$Ex = E\left(\frac{x}{2^{XE}}\right)$$ partie entière.

$$Rx = R\left(\frac{x}{2^{XE}}\right)$$ reste.

(de même pour y).

2°) T * T = I ?

Cette question à laquelle il faut répondre par oui ou par non (Ø ou N) signifie :

- si l'on répond oui : T * T est une identité, auquel cas TV ne demandera ensuite que les coefficients M_{21} et M_{22} de la matrice de transformation et calculera lui-même les coefficients M_{11} et M_{12} .

- si l'on répond non : dans ce cas, TV demandera tous les coefficients de la matrice.

3°) M i j =

Servir les coefficients sur 1 à 4 chiffres hexadécimaux (chiffres négatifs : donner le complément à 2).

- si l'on avait répondu oui à "T* T = I ?", TV ne demande que M_{21} et M_{22} et on obtient :

$$M = \begin{pmatrix} -M_{22} & \dfrac{1 - M_{22} * M_{22}}{M_{21}} \\ M_{21} & M_{22} \end{pmatrix}$$

tous les coefficients doivent $\in Z$; si ce n'est pas le cas pour un coefficient calculé, TV le remplace par la valeur la plus voisine dans Z, et modifie la matrice pour qu'on ait bien $M * M = I$.

- si l'on avait répondu non à "T * T = I ?", TV demande les quatre coefficients.

A ce stade, la transformation est définie. Lorsqu'on utilisera la commande T, TV appliquera la transformation de la façon suivante :

$$P_{(x,y)} \xrightarrow{T} P_{(x',y')}$$

Les coordonnées (x,y) de tout point P de l'image deviennent (x',y') avec :

$$\begin{cases} x = Ex \times 2^{XE} + Rx \\ y = Ey \times 2^{YE} + Ry \end{cases} \qquad \begin{cases} x' = X \times 2^{XE} + Rx \\ y' = Y \times 2^{YE} + Ry \end{cases}$$

avec :

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} Ex \\ Ey \end{pmatrix} + \begin{pmatrix} TRx \\ TRy \end{pmatrix}$$

4°) 1/2 ?

On doit répondre oui ou non (Ø ou N).

- oui : la transformation ne sera appliquée, quand on utilisera T, qu'aux points P (x,y) appartenant à la moitié 2 de l'image (cf : schéma ci-après).

- non : la transformation sera appliquée à tous les points de l'image.

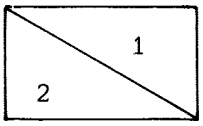


image TV

Exemple :

Avec une matrice de transformation : $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

- si 1/2 = oui, alors la transformation sera une symétrie par rapport à la diagonale principale (2 vers 1 ; transposition).

- si 1/2 = non, alors cette transformation sera une identité.

5.15 Commande G : retour au niveau appel image

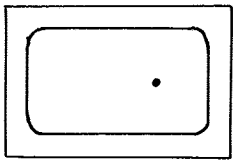
Se reporter qu schéma des commandes de TV et aux exemples où est utilisée la commande G.

5.16 Commande M : passage au niveau interaction graphique

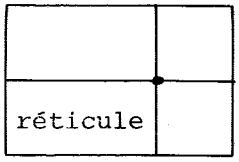
Cette commande permet de passer du niveau traitement image au niveau interaction graphique, conformément au schéma des commandes de TV qui a été exposé.

6 Les commandes "interaction graphique"

Ce sont les commandes du niveau 3 de TV, dont on a parlé précédemment. On accède à ce niveau de commandes en utilisant la commande M du niveau traitement image. Lorsqu'on frappe cette dernière, on voit apparaître le réticule (ou curseur graphique) sur la visu qu'on utilise (4010 ou 4014). Pour déplacer le réticule sur l'écran, on utilise les molettes situées sur la droite du clavier. Pour rentrer une commande, il suffit de frapper la lettre correspondante sur le clavier, lorsque le réticule est sur l'écran.



TV



visu

6.1 Commande C : curseur

Cette commande permet de connaître la position du curseur sur l'image TV. Elle provoque en effet le scintillement sur l'image TV du point correspondant à la position courante du curseur. Ce point est habituellement désigné "point courant".

6.2 Commandes A, B, S : tracé de segments

Ces commandes ont le même effet que les primitives A, B et S du langage graphique (voir processeur GR), c'est-à-dire :

- A : le point courant devient l'origine du segment courant.

- B : le point courant devient l'extrémité du segment courant ; celui-ci est tracé sur l'image, et le point courant devient la nouvelle origine du segment courant.

- S : le point courant devient l'extrémité du segment courant ; celui-ci est tracé sur l'image, mais, contrairement à ce qui se passe avec la commande B, l'origine du segment courant reste inchangée.

6.3 Commande Ø : retour à l'origine

L'origine est le point situé en bas à gauche de l'image.

6.4 Commande 0 : point noir

Le point courant est mis à 0, quelle que soit sa valeur initiale.

6.5 Commande 1 : point blanc

Le point courant est mis à 1, quelle que soit sa valeur initiale

6.6 Commande + : mode tracé blanc

Le tracé sera fait en blanc (donc formé d'un ensemble de points à 1) ; c'est le mode de tracé implicite.

6.7 Commande - : mode tracé noir

Le tracé sera fait en noir (donc formé d'un ensemble de points à 0).

6.8 Commande E : inversion mode tracé

Le mode de tracé est inversé : il passe à blanc s'il était noir, et à noir s'il était blanc.

6.9 Commandes V et W : remplissage

L'image va être balayée ligne par ligne du bas vers le haut. Sur chaque ligne :

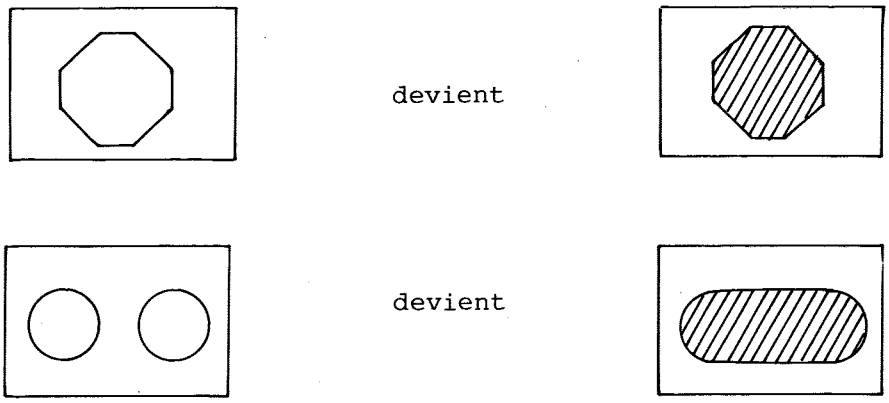
- commande V : tous les points compris entre les deux points extrêmes à 1 (blanc) seront mis à 1 (blanc).

- commande W : tous les points compris entre les deux points extrêmes à 0 (noir) seront mis à 0 (noir).

On peut ainsi remplir des contours simples (et non pas complexes ; voir pour cela le processeur K Ø).

On notera la différence entre ces commandes, V et W, et la commande C du niveau traitement image.

Exemples



6.10 Commandes T, M, N : virtualisation

Ces commandes permettent de réaliser la même fonction que la commande V du niveau traitement image, mais cette fois, on réalisè une modification de la topographie de l'image en "montrant" avec le curseur la modification qu'on veut réaliser :

- la commande T indique que l'on veut modifier la topographie de l'image. Elle doit être suivie des commandes M et N accompagnées des mouvements adéquats du curseur.

- la commande M sert à "montrer" la bande que l'on virtualise. C'est la bande de l'image incluant le point courant.

- la commande N sert à montrer la bande réelle que l'on veut associer à la bande virtuelle désignée par M.

7 La couleur sur T 1600

7.1 Diffusion couleur

Sur le T 1600, l'image résidente est diffusée en noir et blanc sur écran TV noir et blanc, et est diffusée en couleurs sur écran TV couleur. Le principe de diffusion couleur est le suivant : à toute image est associé un "masque-couleur" composé de 3 fois 3 bits :

- 3 bits pour le rouge : R1 R2 R3 = masque rouge
- 3 bits pour le vert : V1 V2 V3 = masque vert
- 3 bits pour le bleu : B1 B2 B3 = masque bleu

Ce masque de 9 bits est défini par l'utilisateur, on verra comment par la suite.

La diffusion de l'image sur téléviseur couleur est réalisée à travers un dispositif hardware qui agit de la manière suivante :

- les bits de l'image à diffuser (image résidente) sont pris 3 par 3 (les trois premiers bits, puis les trois suivants, etc).

- pour chaque triplet de trois bits (P1, P2, P3), on fait simultanément :

P1P2P3 . AND . R1R2R3 = PR1 PR2 PR3

P1P2P3 . AND . VIV2V3 = PV1 PV2 PV3

P1P2P3 . AND . B1B2B3 = PB1 PB2 PB3

- les trois bits : PR1, PR2, PR3 sont envoyés sur le canal rouge.
- les trois bits : PV1, PV2, PV3 sont envoyés sur le canal vert.
- les trois bits : PB1, PB2, PB3 sont envoyés sur le canal bleu.

Exemples :

- P1 P2 P3 = 111 (trois points blancs)
- R1 R2 R3 = 011 = masque rouge.
- V1 V2 V3 = 010 = masque vert.
- B1 B2 B3 = 100 = masque bleu.

on obtiendra :

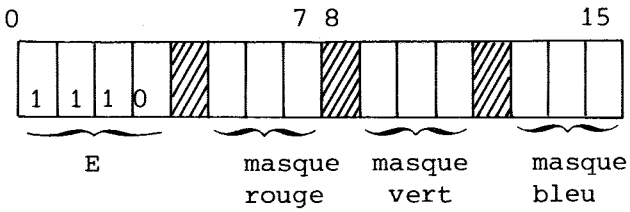
	point 1	point 2	point 3	
- PR1 PR2 PR3 = 111 . AND . 011 =	0	1	1	→ canal rouge
- PV1 PV2 PV3 = 111 . AND . 010 =	0	1	0	→ canal vert
- PB1 PB2 PB3 = 111 . AND . 100 =	1	0	0	→ canal bleu
	↓	↓	↓	
les 3 points résultants seront :	bleu	jaune	rouge	

Nota : Les couleurs qu'on peut obtenir sont : le rouge, le vert, le bleu, le jaune (rouge+vert), le magenta (rouge+bleu) et le cyan (vert+bleu).

Pour forcer une valeur dans le masque-couleur de l'image résidente, il faut utiliser la commande U de la façon décrite ci-après.

```
> U
NØM > [ nom ] D^C          ⟶   nom courant éventuel
ØRG =  E XXX               ⟶   définition masque-couleur.
>
```

E XXX sont 4 chiffres hexadécimaux (donc 16 bits)



Exemple :

```
> U
NØM > D^C
ØRG = E421                 ⟶   masque rouge = 4 = 100 = MR
                               masque vert  = 2 = 010 = MV
>                              masque bleu  = 1 = 001 = MB
```

Le masque-couleur, formé des 9 bits MR MV MB reçoit la valeur servie par l'utilisateur. Ce masque est implanté en tête de l'image, dont il occupe les 9 premiers bits. Il fait donc partie intégrante de l'image. Par conséquent :

- lorsqu'on stocke une image couleur, et qu'ensuite on la rappelle, on n'a pas besoin de redéfinir son masque-couleur (sauf si l'on veut le modifier).

- lorsqu'on fait des opérations entre images (opérations logiques etc), tous les bits des images interviennent, et par conséquent on a toutes les chances de modifier le masque-couleur courant, mais cela peut être voulu : par exemple on peut forcer le masque-couleur à zéro, puis faire une série d'additions (commande H) entre l'image résidente et l'image scratch, celle-ci étant à 0 sauf son premier mot qui sert donc d'incrément du masque-couleur.

On rappelle que la commande U avec ØRG = EXXX ne fait pas intervenir l'image scratch ; elle provoque seulement une modification du masque-couleur de l'image résidente.

7.2 Le multiplexage

Supposons que l'on veuille construire une image couleur (T 1600), à partir de 3 images dont on veut que chacune soit l'une des composantes R, V, B de l'image finale. Etant donné le principe de diffusion couleur sur T 1600, on va être obligé de multiplexer ces trois images : on va donc construire une image multiplexée dont chaque groupe de 3 bits sera formé de 1 bit pris sur la composante R, 1 sur la composante V et 1 sur la composante B. Ce qui revient à dire que l'image multiplexée sera formée de la fusion (ou logique) des trois images R, V et B, mais dont on n'aura retenu qu'un bit sur trois.

Il existe de nombreux items, interprétables par les processeurs TI et SC, permettant à l'utilisateur de construire facilement des images couleurs, et d'explorer, sur une image donnée, les différentes combinaisons de couleurs. Ils sont une aide précieuse à la création; et affranchissent l'utilisateur des opérations fastidieuses.

8 La télévision numérique sur Solar

On dispose sur le Solar de toutes les commandes de processeur TV qui ont déjà été présentées, auxquelles viennent s'ajouter :

- des commandes nouvelles concernant la diffusion des images.
- des commandes des niveaux de gris et de couleur (registes de couleurs).
- des commandes d'accès direct disque permettant le stockage et le chargement d'images désignées non plus par un nom mais par leur adresse physique sur disque, ceci ayant pour principal intérêt d'accélérer la cadence de diffusion des animations.

8.1 La visualisation sur Solar

Contrairement à ce qui se passe sur le T 1600, il n'y a pas de visualisation directe et permanente de l'image résidente. En Solar, ce qui est visualisé est la"somme" des 3 composantes R V B qui résident dans un espace mémoire distinct de celui de l'utilisateur, et cette visualisation est faite à travers un dispositif apparaissant sur le schéma de la page suivante sous le nom de "registres de couleur". Ce dispositif, présenté ci-après au paragraphe 8.2., fonctionne sous le contrôle de l'utilisateur et permet à celui-ci d'obtenir des niveaux de gris et de couleur.

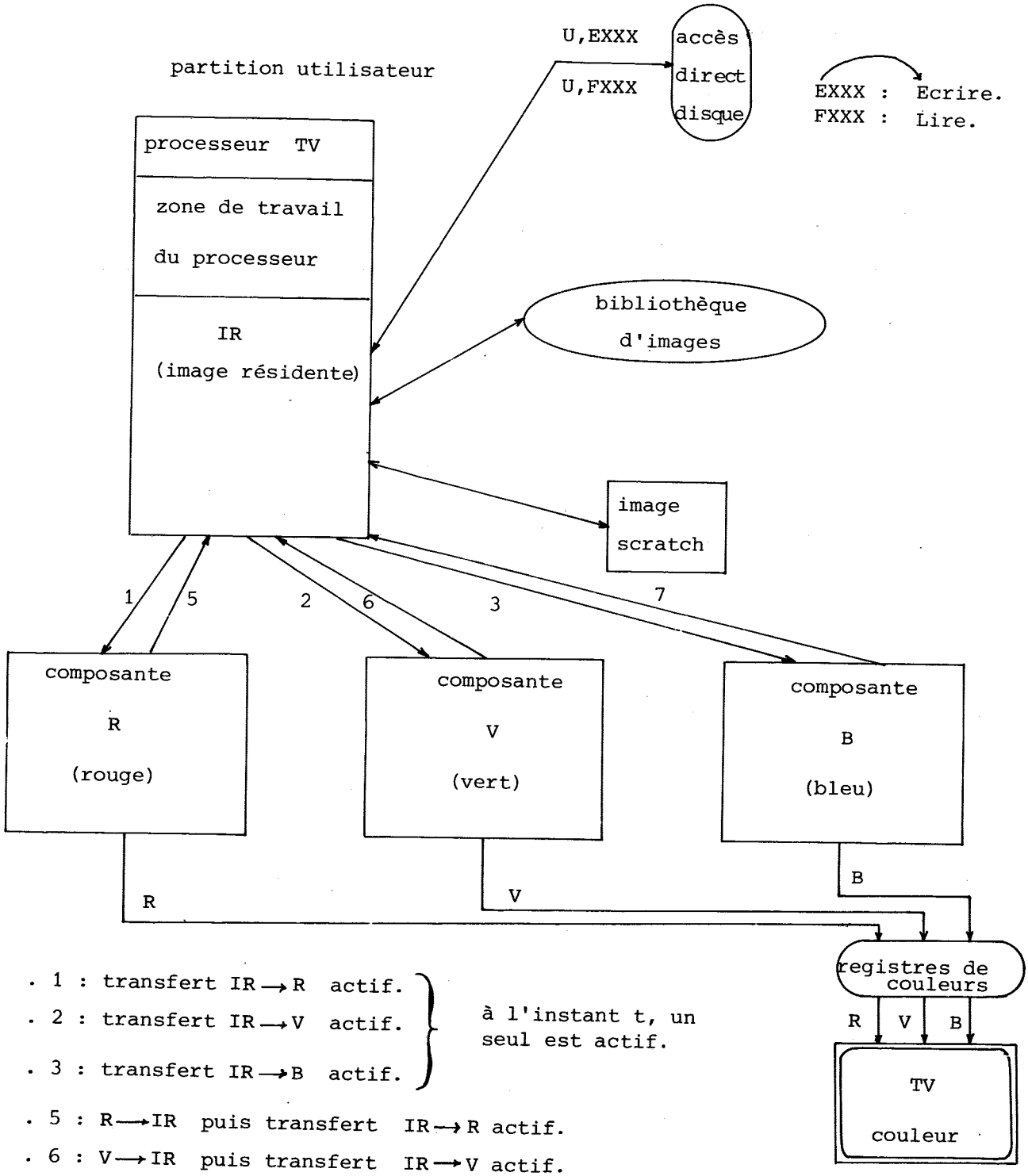
Le contenu des trois composantes R V B dépend de l'utilisateur qui décide d'activer ou d'inhiber le transfert de son image résidente vers l'une des trois composantes R, V ou B, l'activation et l'inhibition étant réalisables par les commandes 0, 1, 2, 3, 4, 5, 6, 7 qui agissent conformément au schéma présenté. On entend pas transfert actif (voir le schéma) le fait que l'image résidente est envoyée dans la composante sélectionnée à chaque commande, ou plus exactement dès qu'une commande a été exécutée par le processeur TV. Par exemple, si l'on frappe la commande T, en supposant le transfert actif, on ne verra pas se dérouler la transposition de l'image résidente, mais on n'en verra, en fin de transposition, que le réultat.

Remarques :

- les commandes 5, 6, 7 qui permettent de recharger en image résidente l'une des composantes R V B (transfert inverse), réalisent au passage une fonction comparable au rappel de l'image scratch. On dispose donc virtuellement en TV Solar de 4 images de travail : l'image scratch proprement dite et les trois composantes.

- les notions, habituelles en T 1600, de masque-couleur et de multiplexage, n'ont plus aucune signification en Solar puisqu'on dispose réellement sur Solar de 3 composantes R, V, B.

Schéma TV Solar



- 1 : transfert IR ⟶ R actif.
- 2 : transfert IR ⟶ V actif.
- 3 : transfert IR ⟶ B actif.

} à l'instant t, un seul est actif.

- 5 : R ⟶ IR puis transfert IR ⟶ R actif.
- 6 : V ⟶ IR puis transfert IR ⟶ V actif.
- 7 : B ⟶ IR puis transfert IR ⟶ B actif.
- 0 et 4 : inhibition transferts.

Exemple :

! TV → (on suppose être en mode transparent)

NØM > D^C

> R → réinitialisation image résidente à 0 (noir)

FØND = 0000

REC ? N

> 1 → activation transfert IR → R

> 2 → activation transfert IR → V

> 3 → activation transfert IR → B

> 0 → l'image TV est noire; inhibition des transferts.

> 1 → activation rouge.

> G

NØM > BØTTICELLI D^C → chargement image.

> D → cette image apparaît en rouge.

> 0 → inhibition des transferts.

> 2 → activation Vert : Botticelli devient jaune (R+V)

> I → inversion de l'image résidente : on a maintenant sur l'écran l'image initiale en rouge sous fond vert.

>

8.2 Niveaux de gris et de couleur

Les registres de couleur apparassant sur le schéma TV Solar, permettent de colorier avec des niveaux de gris les différentes combinaisons de 3 points de mêmes coordonnées appartenant aux trois composantes B, V et R ; soit PB, PV, PR un triplet de tels points.

Chaque point étant binaire (0 ou 1), chaque combinaison (PB, PV, PR) donne un nombre binaire de 0 à 7 :

PB = 0	PV = 0	PR = 0	→	0
PB = 0	PV = 0	PR = 1	→	1
PB = 0	PV = 1	PR = 0	→	2, etc...
PB = 1	PV = 1	PR = 1	→	7

Soit I cette valeur : elle sert d'index dans un jeu de 3 mémoires parallèles MB, MV et MR contenant chacune 8 mots de 4 bits permettant donc chacun de coder 16 niveaux par couleur (0 à 15)

Soit donc (voir schéma page suivante) le point P (PB, PV, PR) ; à ce point P on associe le nombre I avec I = PB x 4 + PV x 2 + PR x 1 ; le point P apparaîtra donc sur l'écran couleur comme la superposition :

- du bleu avec le niveau - MB (I),
- du vert avec le niveau - MV (I),
- du rouge avec le niveau - MR (I),

le signe moins indique que les niveaux sont complémentés à 1 dans les mémoires MB, MV et MR ; c'est-à-dire que dans ces mémoires :

- niveau = 0 est représenté par le nombre binaire 1111 (F)
- niveau = 1 est représenté par le nombre binaire 1110 (E)
- etc
- niveau 15 est représenté par le nombre binaire 0000 (0).

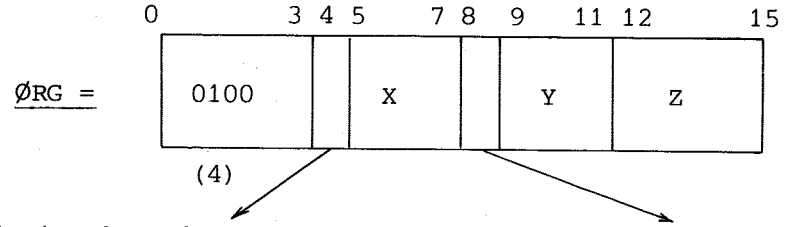
Remarques :

- chaque point P (PB, PV, PR) a 8 valeurs possibles permettant d'accéder dans chaque table à un niveau compris entre 0 et 15 : par conséquent on ne pourra obtenir à un instant donné et sur chaque couleur que 8 niveaux différents, choisis parmi les 16 possibles.

- si l'on place des valeurs identiques dans les 3 mémoires MB, MV, MR, on obtiendra sur l'écran une image noir et blanc, avec des niveaux de gris.

- il existe deux jeux de registres de couleur, et l'utilisateur peut sélectionner l'un ou l'autre (jeu 0 ou jeu 1, voir ci-après). Celui qui est représenté sur le schéma suivant est celui que l'utilisateur a choisi pour la diffusion.

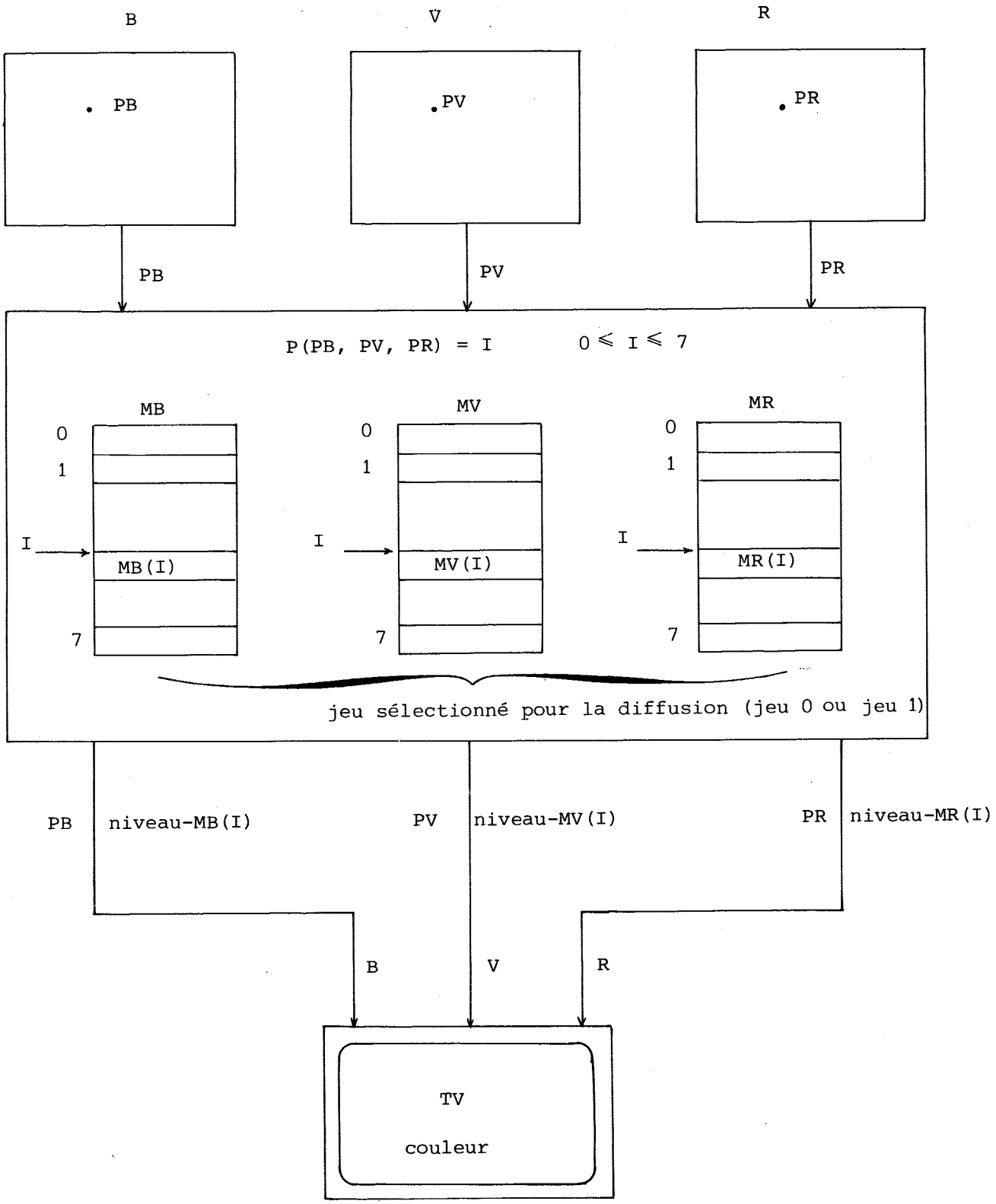
Le contenu des mémoires est défini par l'utilisateur qui agit sur l'un des jeux de 3 mémoires par ses réponses à la question "ØRG =" des différentes opérations inter-images ; les réponses devront être du type suivant :



Sélection du jeu à utiliser pour la diffusion :
0 = jeu 0
1 = jeu 1

Sélection du jeu dans lequel on veut écrire le niveau Z :
0 = jeu 0
1 = jeu 1

Schéma d'un jeu des registres de couleur



- X sélectionne de 0 à 3 mémoires dans lesquelles on veut écrire

  X = 4 sélectionne MB
  X = 2 sélectionne MV
  X = 1 sélectionne MR
  X = 0 n'en sélectionne aucune, et l'on peut combiner ces différentes valeurs (par exemple, X = 6 sélectionne MB et MV).

- Y donne l'index I dans les mémoires sélectionnées ($0 \leqslant I \leqslant 7$)

- Z donne le niveau, complémenté à 1 comme on l'a vu, c'est-à-dire que Z=0 donne le niveau maximum (blanc) et Z = F donne le niveau minimum (noir).

Exemples

ØRG = 470F ⟶ sélectionne jeu 0 en diffusion et écrit dans jeu 0 où MB (0), MV (0) et MR (0) reçoivent F (niveau 0)

ØRG = 4A17 ⟶ sélectionne jeu 1 en diffusion et écrit dans jeu 0 où MV (1) reçoit 7 (niveau 8).

On remarquera qu'il est possible de sélectionner l'un des jeux de mémoires pour la diffusion, et en même temps d'écrire dans l'autre jeu, ce qui permet notamment de basculer brutalement d'un jeu à l'autre en diffusion.

Nota : Si l'on veut faire uniquement des basculements des jeux 0 et 1 en diffusion, sans écriture de niveaux dans les mémoires, il suffit de servir X = 0 dans les réponses à la question "ØRG =".

Remarque 1 : mode transparent

Le mode transparent est un cas particulier d'utilisation des mémoires de couleurs. Dans ce cas, les contenus des mémoires MB, MV, MR, sont les suivants (pour simplifier, les valeurs des niveaux sont écrites en clair et non pas en mode binaire inversé).

	PB	PV	PR		MB	MV	MR	
0	0	0	0	noir	0	0	0	
1	0	0	1	rouge	0	0	15	(jeu utilisé
2	0	1	0	vert	0	15	0	en diffusion)
3	0	1	1	jaune	0	15	15	
4	1	0	0	bleu	15	0	0	
5	1	0	1	magenta	15	0	15	
6	1	1	0	cyan	15	15	0	
7	1	1	1	blanc	15	15	15	

Remarque 2 : les facilités d'utilisation de la TV Solar :

Tout comme sur le T 1600, il existe sur le Solar de nombreux items, interprétables par les processeurs TI et SC, permettant à l'utilisateur de réaliser facilement des opérations complexes telles que, par exemple, coloration, combinaisons et permutations de couleurs, opérations logiques faisant intervenir l'image résidente et les 3 composantes, etc...

8.3 L'accès direct disque

Cette fonction est propre au processeur TV Solar.

On peut :

- écrire l'image résidente sur dique, à une certaine adresse physique (commande U, EXXX XXX étant cette adresse).

- charger une image à partir du disque en image résidente en spécifiant son adresse physique sur le disque (commande U, FXXX).

Exemple :

> U → écriture de l'image résidente sur disque, à l'adresse E40.

NØM > D^{C}

```
ØRG = EE40
> U                          ⟶   chargement en image résidente
                                 de l'image préalablement stockée sur
NØM >D^C                         disque à l'adresse E50.

ØRG = FE50

>
```

Ce mode de stockage et d'accès à des images a pour avantage d'être plus rapide que le stockage et l'accès à une image en bibliothèque. Il est donc utilisé pour les animations. Il suffit de décrire dans un item un scénario de stockage sur disque de l'ensemble des images entrant dans une animation, de l'exécuter par le processeur SC, et les images sont stockées.

Ensuite, on pourra écrire autant de scénarios qu'on voudra, exécutables par SC, et dans lesquels on référencera les images par leur adresse physique.

Nota 1 :
A l'heure actuelle, les adresses disque utilisables sont les suivantes: E00 ⩽ adresse ⩽ EFF. Dans cette limite, le disque est accessible par tous les utilisateurs en lecture et en écriture, sans protection. Il faut donc être prudent.

Nota 2 :
Si l'on fait U avec ØRG = ED^C ou ØRG = FD^C, c'est-à-dire sans spécifier d'adresse physique, TV incrémentera automatiquement de 1 l'adresse physique courante. Ceci permet, dans un scénario par exemple, de ne spécifier que l'adresse de la première image d'une série d'images que l'on veur exploiter.

Exemple :

```
> U
NØM > D^C
ØRG = FE30                  ⟶ lecture image d'adreese E30
> U
NØM > D^C
ØRG = FD^C                  ⟶ lecture image d'adresse E31
> etc.
```

PROCESSEUR TI

TI remplit les mêmes fonctions que TV, mais il va chercher ses commandes en Item 1. L'utilisateur peut donc élaborer des scénarios destinés à être interprétés par TI. Il faut cependant noter entre TV et TI les différences suivantes :

- les commandes à TI étant placées en Item 1, le caractère de fin de message D^C est remplacé par un point-virgule :

Exemple :

On a dans Item 1 la séquence suivante :

IMAGE 1; DSE ; 8000 D etc...

Ce qui équivaut sous TV au dialogue suivant :

! TV
NØM > IMAGE 1D^C
> D
> S
> E
NØM > D^C
ØRG = 8 000
> D
>

- sous TI, la question "REC?" (niveau "appel image", voir TV) n'est pas posée. On peut donc avoir ceci en Item 1 : IMAGE2 ; E000D etc...

- la commande V du niveau "appel image" (entrée vidéo) enregistre une seule image au lieu d'enregistrer en continu comme sous TV.

- pour passer au niveau "interaction graphique", la commande M doit être suivie de G, et deux cas se présentent :

1 - si TI rencontre la commande MG et si l'unité logique B est assignée à la visu α (! ASSIGN B = $\left\{ \begin{matrix} VI\alpha \\ \emptyset \end{matrix} \right\}$), alors le réticule apparaît sur l'écran de la visu assignée à B et on travaille comme sous TV.

2 - si TI rencontre MG et que l'unité logique B n'est pas assignée, les commandes d'interaction graphique sont dans l'item à la suite de la commande MG (difficile à utiliser).

Nota : Il est possible d'appeler TI dans un scénario interprétable par SC ce qui enrichit ses possibilités d'utilisation (itérations sur des appels de TI notamment).

PROCESSEUR SC

SC permet
- l'enchaînement avec "trucage" d'images de TV(cf §1),
- le traitement des sons (cf : §2)
- l'éxécution de sous-programmes assembleur en overlay dans SC, ceux-ci pouvant référencer des S/P de SC lui-même (cf §3)
- le Jeu de la Vie (cf : §4)

1 - IMAGES DE TV.

SC permet :

- *l'enchaînement entre images de TV avec*
- *trucages éventuels sur ces images.*

Les directives à SC sont placées dans Item 1 (et Item 2 par référence à des bibliothèques (cf : GR)).

1.1 - Directives.

- *appel d'une image :*

implicite `< nom de l'image > ;`

explicite `- < nom de l'image > ;`

Exemples :
- `ABCD ;` → SC charge l'image ABCD
- `-%IM Ø Z ;` → SC charge l'image %IM Ø Z

- *itérations (imbricables)* `% α (< séquence >)` $\alpha \in \{1,2,..,9,A,..,Z,@\}$

constante courante
cf : ci-après.

- *Sous-Programmes.*

définition : > < nom du sous progr. > (< séquence >)

appel : & < nom du S/P > ;

? < nom du S/P > ;

- *Bibliothèques.* ¢ < nom bibliothèque > ;

- *Argument formel.*

définition : = (< séquence >)

appel : '

} cf : GR

- *Appel du processeur TI dans SC :* *

* < suite de commandes à TI terminées par F >

appel de TI — suite des commandes à SC

Exemple : à partir d'une image d'origine SPIRALE, on veut faire se succéder sur la TV une suite de transformations de cette image.

Placer dans Item 1 le scénario suivant :

SPIRALE ; %Z (%Z (* ; G S E ; 0001 F)) .....

- SPIRALE ; : chargement d'image
- * : appel TI
- G : display image
- S : scratch
- E : EØR
- 0001 F : ØRG
- F : retour

SC — TI — SC

Nota : on peut par exemple faire ceci :

TI

< nom image > ; * ; G I + G

- G : display image
- I : opération sur cette image : ici inversion
- + : Store image transformée
- G

Application : on peut faire ceci :

1 - définir un S/P de transformation d'image (exemple : épaississement de Jolly Jumper).

2 - appliquer cette transformation successivement à toute une série d'images :

```
> TRANS ( * ; ..... + F)                    définition du S/P

IMAGE 1 ; & TRANS ; IMAGE 2 ; & TRANS ; ...  transformation des images
```

@ α	Réinitialisation constante courante α ∈ { 1,2,..,9,A,...Z } (@ = 1 initialement) @ est référençable par exemple dans une itération % @ (...)
+	Arrêt/marche vidéo ; fonctionne en bascule (cf : commande V sous !I) nota : pour être sûr de mettre ØFF faire ++
.	Retour au CCI (état ?) Revenir par !GØ
/	Fin du programme ⟶ provoque le retour de SC à l'état !
⌀ ;	Elements neutres (sauf dans des noms)
9	Commande dispositif externe (Hard-Copy, Caméra). Sur Hard-Copy faire une temporisation de 15 secondes par ! (cf : ci-dessous). (cf : GR primitive H)
!α	Pause de α secondes (α ∈ { 1,2,..,9,A,..,Z,@ })
< texte ;	Commentaire
< texte !	Texte à éditer en alpha-numérique

} cf : GR

2 - LE SON.

# < suite de caractères différents de ! >	Diffusion d'une suite de sons: les caractères sont interprétés selon une correspondance caractère → fréquence ∈ octave en cours. (Voir processeur SØ)
, α	Changement d'octave $\alpha \in \{1,2,..,9,A,...,Z,@\}$

3 - Appel de Sous-Programmes en overlay.

Ces S/P sont écrits en assembleur sous le processeur GC ; il y a deux façons de les utiliser :

1°)

∅;	Référence bibliothèque vide provoque : . appel du S/P assemblé contenu dant Item 2 . ce S/P ayant été préalablement chargé par ∅ < nom > ; ↓ nom d'un item de type P contenant le S/P

Exemple : ..... ∅SP ; %3(∅;)

↓

exécution 3 fois de suite du SP

2°)

4	Provoque le balayage de l'image résidente, ligne par ligne, et en chaque point (X,Y) le S/P contenu dans Item 2 est exécuté.

balayage: ligne 0 → ligne 1 → ligne 255 →

Le S/P - reçoit les coordonnées du point courant dans les registres X et Y

- peut lui-même appeler les S/P suivants contenus dans SC :

```
RESET (mise à 0)     par    BSR   'FF80,L
SET   (mise à 1)     par    BSR   'FF81,L
INVER (inversion)    par    BSR   'FF82,L
TEST                 par    BSR   'FF83,L
  └──→ résultat du test dans le registre A
       A = 0  :  le point existe et vaut 1
       A = 1  :  le point n'existe pas ou
                 le point existe et vaut 0
```

Exemple : on veut réduire SØN1, SØN2 et diffuser la somme SØN1 réduit + SØN2 réduit :

```
SØN1; ∅ P - REDUCTIØN SØN ; ∅ ;   * ; G   S   F
      └──── réduction SØN1 ────┘            ↓
                                  SØN1 réduit ⟶ Scratch

                                      TI
SØN2; ∅ P - REDUCTIØN SØN ; ∅ ;   *  ;  GH   ; 0000 F
                                      ↓
                                  "Haddition"

∅ P - DIFFUSIØN SØN ; % 8(∅;)
```

pour écouter 8 fois SØN1 + SØN2

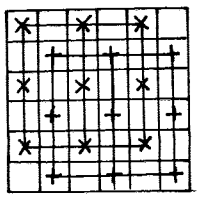
4 - LE JEU DE LA VIE.

S'il n'y a pas de bibliothèque courante (item 2 inutilisé), la primitive [4] fait 1 itération du Jeu de la Vie.

L'image (256 × 256 bits) est quadrillée par 2 quadrillages.

. un quadrillage pair (128 × 128)

. un quadrillage impair (128 × 128)



Pour tout point M(x,y) le voisinage est examiné dans 8 directions :

1 - si M = 1 . s'il y a plus de 4 voisins (5,6,...,8)
alors M est mis à 0 (M meurt étouffé)

. s'il a moins de 3 voisins (0,1,2)
alors M est mis à 0 (M meurt de solitude)

2 - si M = 0 et s'il a 3 voisins, alors M est mis à 1 (M naît)

On peut itérer ce jeu %n(4), on constate que la population est stable au plus.

Nota : on entend par voisins de M ses voisins sur son quadrillage (pair/impair)

itérations image 1 ⟶ image 2 ⟶ image 1 ⟶ image 2

SC 7

ADDITIFS SC 1/8/79

Sur Solar, on a le choix de la diffusion du scénario sur l'une des 3 composantes R, V, B. La primitive + a l'effet suivant :

- rencontrée seule : inverse l'état courant de la diffusion. Si celce-ci est active sur RØUGE par exemple, elle devient inactive.

- rencontrée suivie de 0, 1, 2, 3 : active la diffusion sur l'une des 3 composantes (1 = RØUGE; 2 = VERT ; 3 = BLEU ; 0 = activation sur composante vide).

Exemple :

+1 (diffusion rouge) + (désactivation) + (activation rouge)

+2 (activation vert) etc...

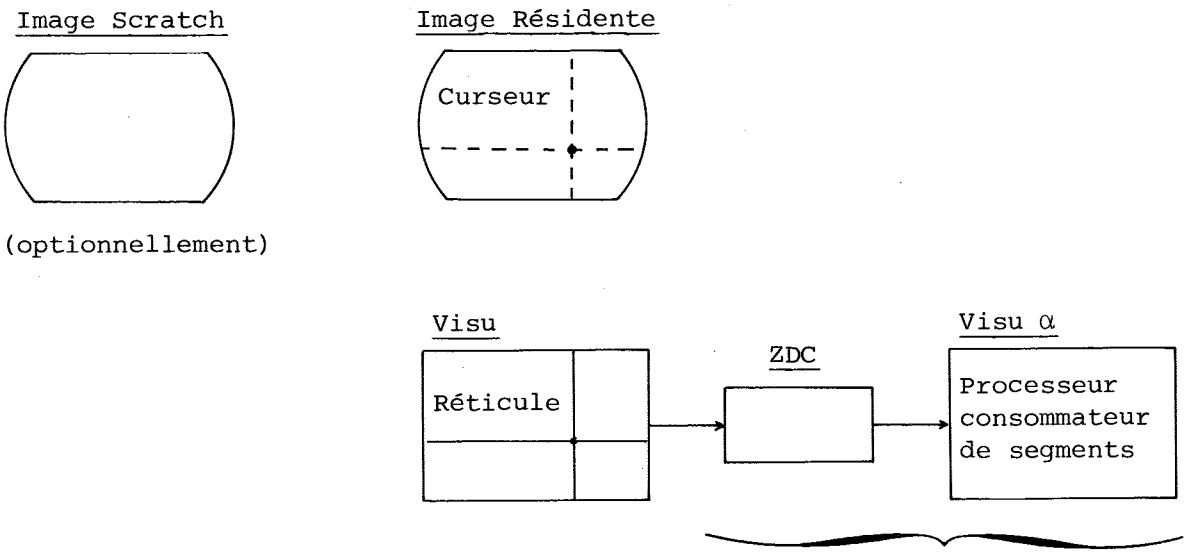
Nota 1 :

+1 suivi de ++ entraînera diffusion rouge puis désactivation de la diffusion puis diffusion rouge car le second + réactive la diffusion sur la dernière composante sélectionnée.

Nota 2 :

Implicitement, la diffusion a lieu sur le vert (donc un + rencontré seul en début de scénario entraînera la diffusion de celui-ci sur le vert).

PROCESSEUR VØ (VOLETS)



Le processeur VØ fait intervenir :

- l'image résidente.
- et, facultativement, l'image scratch.

Il autorise l'émission en ZDC des segments tracés.

Il permet :

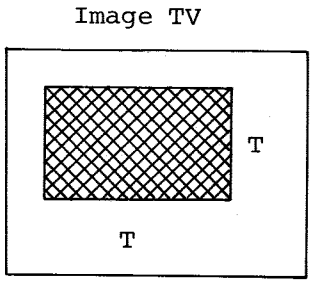
- de tracer des segments d'épaisseur variable (de 1 à'FF points d'épaisseur).
- de tracer des surfaces (ou volets) de taille 1x1 à 'FFx'FF points (voir ci-après la définition précise du mot volet).
- de remplir des contours connexes simples (voir processeur TV, commande M sous commandes V et W).

VØ est particulièrement utile pour :

- générer rapidement des images synthétiques en couleur.
- faire des incrustations.
- retoucher"manuellement" (c'est-à-dire au curseur) des images.

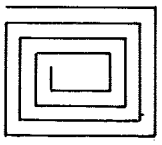
Qu'entend-on par volet ?

Lorsqu'on demande à VØ de générer un volet de taille T, il génére une surface de TxT points, en commençant par le centre (position courante du curseur). Etant donné le rapport entre la hauteur et la largeur de l'écran TV : $\frac{h}{\ell} = \frac{3}{4}$, la surface obtenue, de TxT points est en fait un rectangle de rapport $\frac{3}{4}$.



Volet de T × T

En fait, le volet est engendré par une spirale que VØ développe autour du point courant (c'est-à-dire correspondant à la position courante du curseur).



Spirale engendrant un volet

Les commandes disponibles sous VØ sont développées ci-après.

Utilisation :

Le dialogue avec VØ se déroule de la façon suivante :

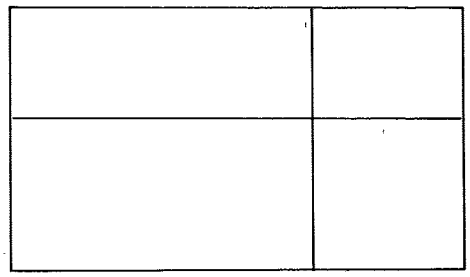
! VØ

MØDE (1 = AD/ 2 = SB/ 3 = ØR/ 4 = AND/ 5 = EØR) =
- 1 (addition)
- 2 (soustraction)
- 3 (ou logique)
- 4 (et logique)
- 5 (ou exclusif)

RAZ ?
- Ø (mise à zéro image scratch)
- N (pas de mise à zéro image scratch)

GRAPHIQUE ZDC ?
- Ø (émission en ZDC des segments tracés)
- N (pas d'émission en ZDC).

MEMØIRE SCRATCH ØN (l'image scratch est active à priori, elle peut être désactivée/réactivée par la primitive M).



Le Réticule apparaît, on peut alors se positionner où l'on veut sur l'image, et utiliser les primitives :
A, B, S, 0, 1, 2, +, -, I, V, W, Z, T, C, R, M, F.

Au réticule de la visu est associé le curseur sur l'image résidente (voir ci-après commande C).

MØDE : spécifie l'opération à effectuer entre :

- ce qui est rentré par l'opérateur au réticule (surfaces, segments) d'une part,
- l'image scratch d'autre part, le résultat étant reçu par l'image scratch.

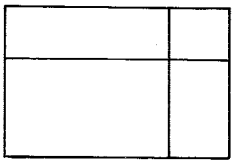
En effet, si l'on a "mémoire scratch on ", VØ effectue à tout moment

Image scratch : = Image scratch. { + (AD) ; - (SB) ; ØR (ØR) ; AND (AND) ; EØR (EØR) } . Points rentrés par l'opérateur.

RAZ : remise à zéro Image scratch ; répondre oui (Ø) ou Non (N). Il est inutile de demander la remise à zéro image scratch, si l'on n'a pas l'intention de l'utiliser (autrement dit si l'on a l'intention de travailler en "mémoire scratch off").

GRAPHIQUE ZDC : transmission en ZDC des segments tracés en vue de leur utilisation par un processeur "consommateur" de segments.

MEMØIRE SCRATCH ØN : à priori, la mémoire scratch est ØN, et VØ effectuera donc pour chaque point rentré par l'opérateur, l'opération spécifiée par MØDE. Pour inverser l'état de mémoire scratch (ØN/ØFF), utiliser la primitive M décrite ci-dessous. Dans le cas où l'on n'a pas besoin de la mémoire scratch, il faut la mettre ØFF, ce qui permet de gagner du temps car les opérations sur image scratch sont longues.



RETICULE : on dispose alors pour travailler : du curseur graphique que l'on peut positionner où l'on veut, et des primitives A, B, S, 0, 1, 2, +, -, I, V, W, Z, T, C, R, M, F dont l'utilisation est assez semblable à ce que l'on trouve sur le processeur TV en mode interaction graphique auquel on peut se reporter.

A, B, S, +, -, I, V, W : même fonctions que sous TV en mode interaction graphique à ceci près qu'ici intervient la notion de largeur du segment.

A origine segment

B fin segment ; origine segment suivant

S fin segment sans changement d'origine

\+ mode tracé 1 (blanc)

\- mode tracé 0 (noir)

I mode tracé inversion (blanc sur fond noir et noir sur fond blanc)

V remplissage blanc

W remplissage noir

Pour A, B, S, la largeur du segment tracé sur l'image est fonction de la taille spécifiée par la primitive T décrite ci-après

Z : remise à zéro image résidente.

R : retour temporaire au CCI ; revenir par ! GØ.

M : activation/ désactivation de l'option "mémoire scratch on (off)". Si l'on fait M alors que mémoire scratch est ØN, celle-ci sera désactivée,

et on verra apparaître sur l'écran le message : "mémoire scratch off". Et inversement. En "mémoire scratch off", le fonctionnement est plus rapide.

0, 1, 2, T

0, 1, et 2 tracent une spirale qui engendre un volet de TxT points, T étant spécifié en utilisant la commande T : taille du volet.

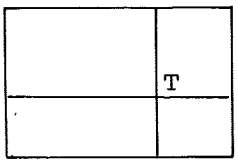
Implicitement, T = 1, et donc 0,1 et 2 n'engendrent qu'un seul point :

- 0 engendre un point noir
- 1 engendre un point blanc
- 2 engendre un point noir si fond blanc, blanc si fond noir (mode inversion).

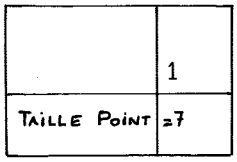
Pour modifier T, il suffit de rentrer la primitive T ; on voit alors apparaître sur l'écran la question suivante :

TAILLE PØINT = 10 D^C répondre un nombre héxadécimal de 1 à 'FF suivi de D^C ou Return. Ainsi, lorqu'on à T > 1, les primitives 0, 1 et 2 engendreront un volet autour du point courant ; en fait 0, 1, 2 provoquent le tracé d'une spirale autour du point courant.

Exemple :



on frappe T
on voit apparaître :
TAILLE PØINT = 7 D^C on demande taille = 7.

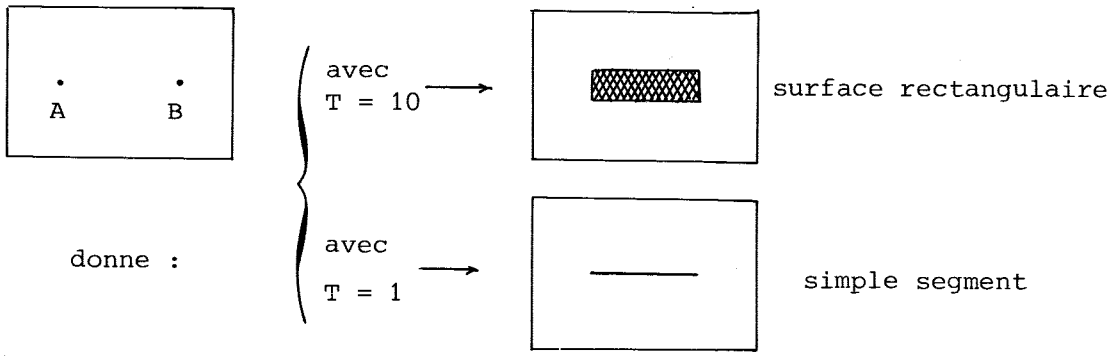


on frappe 1
il va se passer ceci sur l'image résidente autour du point courant :

on obtient donc un volet blanc de 7x7 points.

En supposant qu'on frappe immédiatement après 0, le carré disparaît.

T s'applique aussi aux primitives A, B et S ce qui permet par exemple de tracer des segments d'épaisseur > 1 ; il suffit de faire : T = 10 A B pour obtenir un rectangle



C : Curseur. Scintiilement sur l'image TV du point correspondant à la position courante du curseur.

F : Fin Provoque la sortie de VØ et donc le retour au ! . L'image résidente et l'image scratch sont récupérables sous TV, comme ceci :

```
! VØ
  ≀
ici, on travaille sous VØ, avec par exemple l'image scratch active ; on sort
de VØ par F.
  ≀
! TV                          ⟶   appel de TV
NØM >D^C
> E                           ⟶   conservation image résidente
FØND = 0000
REC ? N
> D                           ⟶   display image résidente
> U                           ⟶   récupération image scratch
NØM > D^C
ØRG = 0000
> D                           ⟶   display image scratch
>
```

VØ 7

ADDITIFS VØ 1/8/79

Sur Solar, ayant obtenu le curseur sur l'écran de la visu, on dispose des commandes 5, 6 et 7 pour sélectionner la composante sur laquelle on veut travailler. Ces commandes ont l'effet suivant :

- 5 la composante RØUGE est chargée en image résidente et la diffusion Image Résidente ⟶ Composante RØUGE est activée.

- 6 même chose avec composante VERT

- 7 même chose avec composante BLEU

Ainsi, on peut travailler successivement sur chacune des composantes.

Nota :
Si l'on ne sélectionne aucune composante, alors l'Image Résidente est diffusée sur la composante VERT mais sans le chargement préalable de cette composante en Image Résidente.

PROCESSEUR TN

Fonction :

TN permet de créer ou de modifier une image en utilisant une entrée caméra ou light-pen. Le light-pen est particulièrement utile pour rentrer manuellement un tracé en couleur sur un écran TV.

La partition mémoire de l'utilisateur a le format suivant :

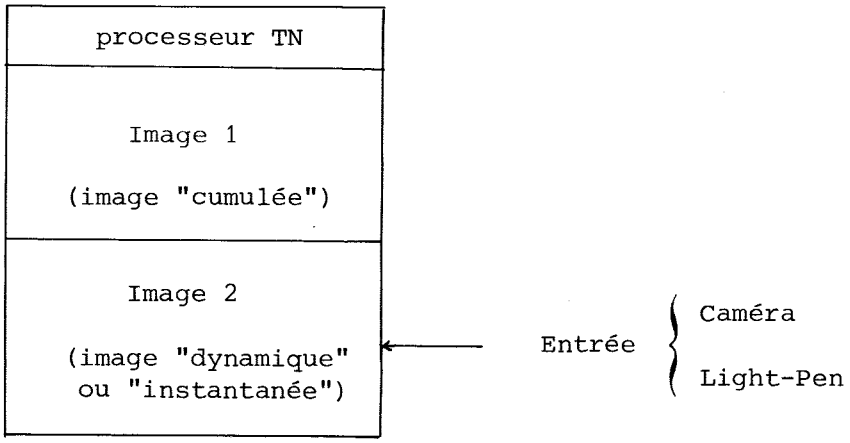


Image 1 est recalculée en permanence à partir de Image 2 comme ceci :

Image1 ⟵ Image1 . Opérateur . Image 2

L'opérateur étant spécifié par l'utilisateur (ou logique, et logique, ou exclusif etc...) et s'applique selon différents modes (voir ci-après).

Utilisation de TN comme light-pen.

Sous TN, on peut utiliser le light-pen comme un pinceau et comme une gomme :

- comme un pinceau car on peut tracer ce que l'on veut avec la couleur qu'on veut sur l'écran TV.

- comme une gomme parce que l'on peut effacer un tracé : soit globalement, soit sélectivement en n'effaçant par exemple qu'une couleur ; on a donc avec TN une gomme sélective (exemple : effacer le rouge sur un tracé jaune ⟶ le tracé deviendra vert ; car en effet, jaune = rouge + vert).

Le dialogue avec TN se déroule comme suit :

!TN ⟶ appel de TN

ALT-MØDE IMMEDIAT ! ⟶ ici, TN prévient qu'il travaille en mode "Alt-Mode immédiat", ce qui veut dire que si par la suite on interrompt la composition de Image 1 et Image 2 par l'opérateur courant (voir commandes ci-après), cette interruption sera immédiate, c'est-à-dire qu'elle sera effective même si l'opération courante :

Image 1 ⟵ Image 1 . Opérateur . Image 2 n'est pas terminée.

Ceci par opposition au mode "Alt-Mode différé" qui fait que l'interruption par Alt-Mode ne sera effective qu'en fin d'opération entre Image 1 et Image 2.

>	A	Et logique (And)	Opérateurs utilisables
	E	Ou exclusif (Exclusive Or)	
	Ø	Ou logique (Or)	
	C	Curseur	
	N	Non-opération	
	R	Rouge	Inversion état couleur (couleur active/inactive). Noter que les couleurs sont cumulables.
	V	Vert	
	B	Bleu	
	X	Inversion état Alt-Mode (immédiat/différé)	
	0	pas d'inversion vidéo	
	1	inversion vidéo	
	=	adresse image 2 = adresse image 1 (recouvrement des deux images).	
	#	adresse image 2 ≠ adresse image 1 (distinction)	
	W	retour au CCI (?) ; revenir par !GO D^C	
	F	Fin de travail.	

Sont explicitées ci-après les commandes : A,E,Ø,C,N,R,V,B,X,0,1.

Commandes A E Ø C N : Opérateurs.

Ces commandes définissent l'opérateur à utiliser entre Image 1 et Image 2. On rappelle qu'à tout moment TN calcule :

Image 1 ⟵ Image 1 . Opérateur . Image 2,

Image 1 étant visualisée.

A : Et logique (And)

E : Ou Exclusif (Exclusive Or)

Ø : Ou Logique (Or)

C : Curseur : cette fonction Curseur est réalisée en faisant deux "Ou Exclusif" à la suite ; ceci a pour effet de faire clignoter le point courant, et donc de simuler un curseur.

N : Non opération.

Le dialogue se déroule comme suit :

```
! TN
ALT-MODE IMMEDIAT !
> A                    → And par exemple ;
```

ici, on utilise le light-pen ; si l'on veut revenir au dialogue avec TN, faire Alt-Mode.

> → rentrer une nouvelle commande.

Commandes R V B : Rouge Vert Bleu.

Ces commandes : R,V,B ont pour effet d'inverser l'activité de la couleur correspondante (Rouge, Vert Bleu) :

```
! TN
ALT-MODE IMMEDIAT !
> R ØN          →  commande R ; TN répond "ØN", c'est-à-dire que la couleur
                   Rouge est active.
> R             →  à nouveau commande R : Rouge devient inactive...
> B ØN          →  Bleu actif
> Ø             →  Ou logique
```

Ici, le tracé entré au light pen est bleu. On fait Alt-Mode pour revenir au dialogue.

```
> B             →  Bleu inactif
> V ØN          →  Vert actif
> Ø             →  Ou logique à nouveau
```

Ici, tracé vert, puis Alt-Mode

```
> R ØN          →  Rouge actif, (Vert est toujours actif)
> Ø
```

Ici, tracé Jaune (Vert + Rouge).

etc....

Commande X : inversion état Alt-Mode (immédiat/différé).

```
! TN
ALT-MODE IMMEDIAT !
>
}
}
}
> X
ALT-MODE DIFFERE !
>
|
etc....
```

Rappel :
Si "Alt-Mode immédiat", l'interruption de l'opération.
Image 1 ← Image 1 . Opérateur . Image 2 sera immédiate.
Avec Alt-Mode différé, l'interruption ne sera effective qu'en fin d'opération.

Commandes 0 et 1 : non-inversion vidéo/inversion vidéo.

A priori, TN travaille en non-inversion vidéo ; c'est-à-dire que Image 2 (image instantanée) est prise telle quelle pour l'opération entre Image 1 et Image 2. Avec 1 et 0 on peut inverser et rétablir ce mode. C'est ainsi qu'on a l'effet de gomme sélective des couleurs :

! TN
ALT-MØDE IMMEDIAT !
> R ØN
> V ØN
> Ø → Ou logique

Ici, on trace en jaune (rouge + vert).

On fait Alt-Mode pour revenir au dialogue :

> 1 → mode inversion ; les points rentrés au light-pen seront donc pris à 0, et, en utilisant l'opérateur And, on pourra effacer.

> A → And

Ici, on efface avec le light-pen le tracé jaune rentré précédemment. On fait "Alt-Mode" pour revenir au dialogue.

> 0 → mode "non-inversion"

> Ø → Ou

Ici, on refait un tracé jaune.

> R → Rouge devient inactive.

> 1 → mode inversion

> A → And

Ici, en repassant sur le tracé jaune, on n'efface que le vert ; et donc le tracé devient rouge là où l'on passe. On fait "Alt-Mode" pour revenir au dialogue > etc...

TN 6

ADDITIFS TN 1/8/79

Sur Solar, on a 3 composantes au choix (non exclusif) pour la diffusion de l'image "cumulée", et 3 composantes au choix (non-exclusif) pour l'entrée image "dynamique".

Les commandes R, V, et B utilisables sur T1600 sont remplacées par une commande unique V suivie d'un chiffre de 0 à 7 (voir processeur TV) :

- V1 } activation/inhibition composantes
- V2 } RØUGE (1), VERT (2), BLEU (3) en diffusion
- V3 } de l'image résidente.

- V5 } activation/inhibition composantes
- V6 } RØUGE (5), VERT (6), BLEU (7) en entrée
- V7 } (image dynamique)

- V0 équivalent à V2
- V4 équivalent à V6. V2 et V6 sont toujours actives initialement.

Exemple :

> V0 } → désactivation des composantes actives initialement
> V4 }

> V1 } à chaque entrée : l'opération demandée est faite entre l'image
> V2 } résidente et composante VERT puis avec composante BLEU, et l'image
> V6 } résidente obtenue sera diffusée sur les composantes RØUGE ET VERT.
> V7 }

PROCESSEUR TA (INTERPOLATEUR)

TA est utilisable suivant deux modes :

- 1er mode : interpolation entre deux dessins.
- 2ème mode : interpolation entre deux images TV.

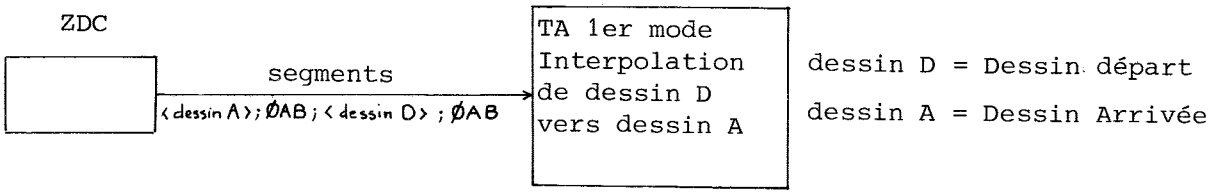
TA 1er mode : interpolation entre deux dessins

Dans ce mode, TA est consommateur de segments ZDC. Il doit recevoir en ZDC la séquence de segments suivante :

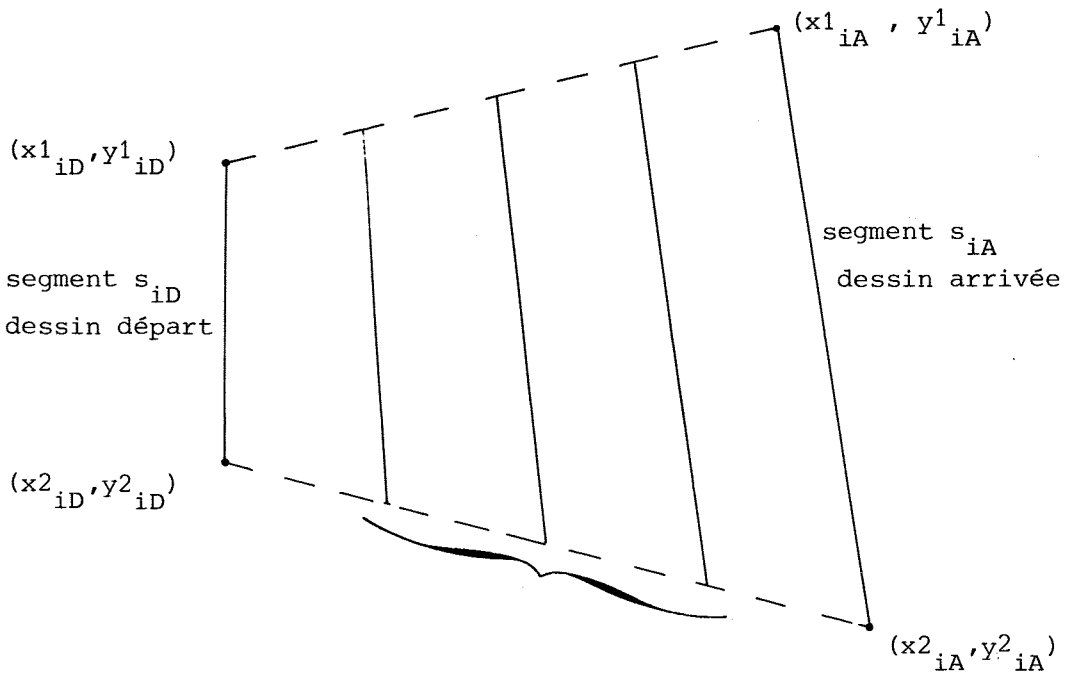
< ensemble de segments du dessin d'arrivée > <segment nul à l'origine >

< ensemble de segments du dessin de départ > <segment nul à l'origine >

On entend par segment nul à l'origine, un segment pour lequel $((x_1, y_1), (x_2, y_2)) = ((0,0), (0,0))$, ce segment étant facile à générer dans un programme graphique par ØAB. (se reporter au langage graphique, processeur GR).



Pour interpoler entre 2 dessins, TA agit de la manière suivante : il prend chacun des dessins segment par segment ; pour chaque couple de segments : s_{iD} de dessin D et s_{iA} de dessin A , il calcule les coordonnées des (p-1) segments intermédiaires, p étant le nombre de pas spécifié par l'utilisateur en réponse à la question "nombre de pas" (cf : utilisation de TA). Les coordonnées des segments intermédiaires sont calculées en découpant en p parties égales l'écart entre coordonnées de départ et coordonnées d'arrivée comme le montre le schéma ci-après.



(p-1) segments intermédiaires
calculés par TA.
Ici, nombre de pas p=4.

TA travaillant segment par segment, il doit y avoir un même nombre de segments dans le dessin de départ et dans le dessin d'arrivée. Si ce n'est pas le cas, TA complète le dessin ayant le moins de segments par des segments à l'origine ((0,0), (0,0)).

Les dessins intermédiaires et le dessin d'arrivée peuvent être sortis en vidéo (voir utilisation).

Si l'on veut que les dessins intermédiaires et le dessin d'arrivée soient émis en ZDC au fur et à mesure de leur production, il faut utiliser le processeur TB.

TA second mode : interpolation entre deux images TV

Il faut fournir à TA une image d'arrivée et une image de départ. Elles peuvent être désignées explicitement par leur nom, ou placées en mémoure et mémoire scratch (cf. dialogue avec TA).

Les images sont parcourues de gauche à droite et de haut en bas et TA les associe point à point.

Parcours des images

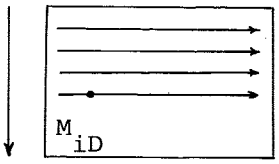


Image Départ

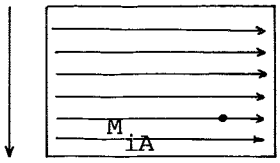


Image Arrivée

Pour interpoler , il prend chaque image point par point (point blanc bien entendu) et pour chaque couple de points :

M_{iD} de coordonnées (x_{iD} , y_{iD}) de image D (départ)

M_{iA} de coordonnées (x_{iA} , y_{iA}) de image A (arrivée),

il calcule les $p-1$ points intermédiaires, p étant le nombre de pas spécifiés par l'utilisateur.

Dans le cas où le nombre de points de l'image départ N_D est différent du nombre de points de l'image d'arrivée N_A :

1/ si $N_D < N_A$: alors TA réexamine image D autant de fois que nécessaire jusqu'à exploitation complète de image A.

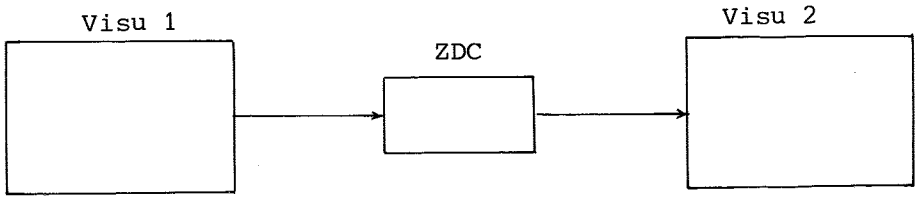
2/ si $N_D > N_A$: alors TA complète image A par ($N_D - N_A$) points à l'origine (c'est-à-dire en haut à gauche de l'image).

Nota : si l'on veut obtenir les images intermédiaires entre deux images image1 et image 2 avec $N_1 \neq N_2$ on devra choisir le sens de l'interpolation.

1/ si l'on ne veut pas que soient créés des points à l'origine, mais qu'on préfère le réexamen de l'image de départ, on placera l'image ayant le moins grand nombre de points comme image de départ image 1 = image Départ si N1 < N2).

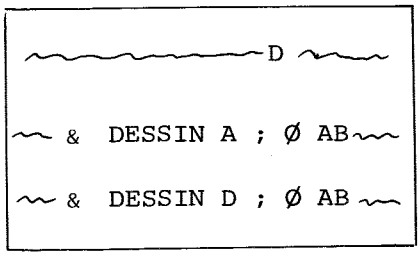
2/ si l'on préfère que soient créés des points à l'origine, alors on placera l'image ayant le moins grand nombre de points comme image d'arrivée (image 1 = image Arrivée si N1 < N2).

Utilisation de TA 1er mode (dessin ⟶ dessin)



Transmission par ZDC des 2 dessins : dessin A et dessin D.

Sur visu 1, on a par exemple un item graphique contenant :



sur lequel on fait GR.

Cette visu sera libre dès que GR aura fini son travail, alors que sur la visu 2 commencera l'interpolation.

! SE

TYPE > 0 ⟶ Remise à zéro ZDC

! TA

VIDEØ ? N ⟶ non ; on veut donc interpoler du graphique (TA 1er mode)

DELETE ?
- Ø ⟶ remise à zéro de la ZDC
- N ⟶ pas de remise à zéro de la ZDC.

TRAJET ?
- Ø ⟶ on veut tracer non seulement les dessins intermédiaires, mais aussi la trajectoire de chaque extrémité de segment. Sur une visu 4014 cette trajectoire est tracée en pointillés (cf. exemples d'utilisation)
- N ⟶ pas de tracé de la trajectoire.

EFFACEMENT ?
- Ø ⟶ effacement écran demandé entre chaque tracé de dessin intermédiaire ; de plus, après tracé du dessin de départ, le système affichera le nombre de points du dessin d'arrivée et le nombre de points du dessin de départ (c'est-à-dire nombre de segments x 2).
- N ⟶ pas d'effacement d'écran entre chaque tracé de dessin intermédiaire ; en fin de travail on aura donc sur l'écran l'ensemble { dessin départ ; dessins intermédiaires ; dessin arrivée }

CØPY ? { Ø / N →

Ø → on demande une ou plusieurs copies de l'écran sur hard-copy (ou télécommande de caméra) ; le système demande :

NBRE DE CØPIES/TEMPØRISATION = nt

répondre 2 chiffres hexa : n nombre de copies
t temporisation.

.si PAS A PAS est demandé, le système déclenchera la copie après chaque dessin intermédiaire.

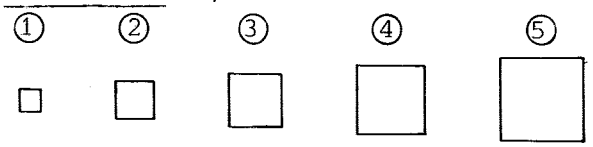
.si PAS A PAS n'est pas demandé, le système ne déclenchera la copie qu'en fin de tracé du dessin final.

En combinant cette option avec EFFACEMENT, on peut obtenir, sur film par exemple :

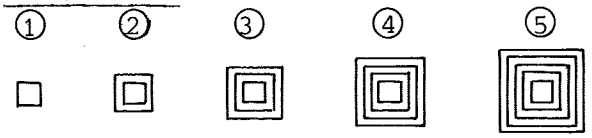
.soit la séquence de dessins intermédiaires.
.soit la progression du dessin origine jusqu'au dessin final avec toutes les étapes intermédiaires superposées.

Exemple :

.EFFACEMENT = Ø



.EFFACEMENT = N



Nota : lorsqu'on n'est pas en pas à pas, on ne peut obtenir que la copie du dessin final (ici, dessin ⑤).

CØNVERSIØN VIDEØ ? { Ø / N

Ø → tous les dessins intermédiaires et le dessin d'arrivée seront également sortis sous forme d'image TV. On les verra donc s'afficher successivement.

N → pas de conversion vidéo.

Il faut noter ceci :

. si l'on veut avoir la possibilité de conserver les images vidéo obtenues, il faut travailler en pas à pas (voir ce paramètre).

. sur l'image vidéo, on n'obtient que l'image intermédiaire en cours, même si l'on travaille avec EFFACEMENT = Non. On pourra ensuite composer les images intermédiaires entre elles en utilisant le processeur TV.

. sur l'image vidéo n'apparaît pas la trajectoire, même si l'on travaille avec TRAJET = Oui.

PAS A PAS ?

- Ø → on travaille en pas à pas. Après chaque tracé intermédiaire, le système avertira qu'on est en fin de tracé en envoyant la cloche.

  Il proposera, si et seulement si VIDEØ=Øui:
  SAVE ?
  - Ø → on veut cataloguer l'image vidéo.
    NØM > nom de l'image D^C
  - N → on ne veut pas cataloguer cette image.

  Si CØPY = Øui, le système déclenchera le dispositif de hard-copy ou caméra, le nombre de fois demandé avec la temporisation demandée, entre chaque copie.

- N → on ne travaille pas en pas à pas ; donc le système ne proposera pas de cataloguer les images éventuelles ni ne fera de copie entre chaque dessin intermédiaire. En revanche, on pourra avoir une copie du dessin final si l'option CØPY est active.

NBRE DE PAS = { X D^C ; X X } → répondre un nombre hexa de 1 à'FF. C'est le nombre d'étapes du dessin de départ au dessin d'arrivée. Si, par exemple, on répond 4, on obtiendra 3 dessins intermédiaires + le dessin d'arrivée soit 4 dessins au total.

TRANSFORMATION ?

- Ø → on veut faire subir aux coordonnées arrivant en ZDC une transformation au moyen d'un programme placé en Item 2 (programme assembleur assemblé par GC ; voir ce processeur).
- N → pas de transformation.

A ce stade, TA efface l'écran est exploite les données ZDC :

< Dessin arrivée > ØAB traçage du dessin d'arrivée.

Fin du traçage du dessin d'arrivée.

N — TA annonce par un signal sonore (cloche), la fin de traçage du dessin d'arrivée ; répondre N .

A ce stade, TA efface l'écran et exploite les données ZDC :

< Dessin départ > ØAB traçage du dessin de départ.

Puis TA trace successivement tous les dessins intermédiaires jusqu'au dessin final avec, entre deux tracés :

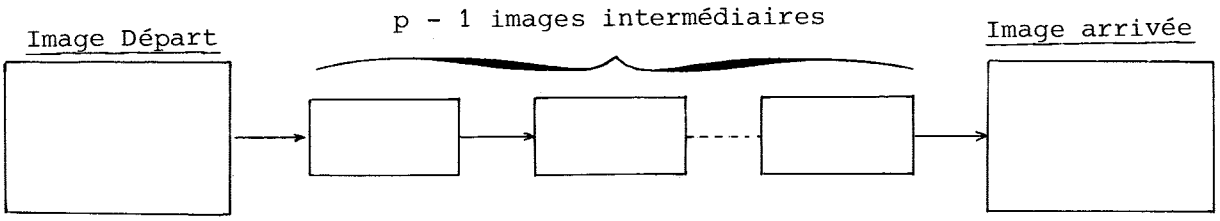
- . effacement ou non
- . Hard-COPY/ Caméra ou non
- . proposition de SAVE (cataloguage) des image vidéo ou non

} en fonction des réponses données auparavant aux options :

ƎFFACEMENT / CONVERSIØN VIDEØ / CØPY / PAS A PAS.

! → fin du travail.

Utilisation de TA second mode (image TV → image TV)



Le dialogue avec TA se déroule de la façon suivante.

! TA

VIDEØ ? Ø → oui, on veut donc travailler sur des image vidéo.

PAS A PAS ?

- Ø → oui, on veut travailler en pas à pas ; le système proposera donc de cataloguer chaque image intermédiaire ; après tracé d'une image, on aura :
  - SAVE ?
    - Ø : oui, alors le système demande un nom : NØM > nom de l'image D^C
    - N : on ne veut pas cataloguer cette image.
- N → non : on ne veut pas travailler en pas à pas, par conséquent on verra s'afficher successivement les images intermédiaires sans pouvoir les cataloguer.

NBRE DE PAS =

- X D^C
- XX

→ répondre un nombre hexadécimal de 1 ou 2 chiffres, compris entre 0 et'FF. Ce sera le nombre d'étapes d'interpolation (on aura p-1 images intermédiaires plus l'image d'arrivée = p images).

TRANSFORMATION ?

- Ø → oui: on veut faire subir aux images d'arrivée et de départ une transformation au moyen d'un programme placé en Item 2 (programme assembleur assemblé par GC).
- N → pas de transformation.

ARRIVEE RESIDENTE ?

- Ø → l'image d'arrivée est résidente, ce qui suppose qu'on l'y ait préalablement placée.
- N → l'image d'arrivée n'est pas résidente ; il faut donc la désigner explicitement : le processeur TA demande son nom :

  ARRIVEE > nom image d'arrivée D^C

TA analyse l'image d'arrivée. Quand il a terminé, il prévient par un signal sonore (cloche) ;

 N → répondre N pour continuer le dialogue.

DEPART SCRACH ?

- Ø → l'image de départ est en mémoire scratch, ce qui suppose qu'on l'y ait préalablement placée.
- N → l'image de départ n'est pas en mémoire scratch. Il faut donc la désigner explicitement : le processeur TA demande son nom :

  DEPART >nom image de départ D^C

TA analyse l'image de départ. Quand il a terminé, TA affiche :

PØINTS DEPART = 0588
PØINTS ARRIVEE = 0BD7

} nombre de points image départ et image d'arrivée (par exemple).

puis il prévient en envoyant un signal sonore (cloche) ;

 N → répondre N pour relancer TA qui commence alors la génération des images intermédiaires.

Si l'option PAS A PAS est active, TA demande après la génération de chaque image intermédiaire :

SAVE ?

- Ø → on veut sauvegarder l'image ; TA demande alors :
  NØM > nom de l'image D^C (Si le nom est vide, l'image n'est pas sauvegardée).
- N → on ne veut pas cataloguer cette image.

Et ainsi de suite jusqu'en fin de travail.

! fin de travail.

Remarques

1/ En l'absence de l'option PAS A PAS, on ne peut cataloguer les images intermédiaires, mais on les voit se succéder sur l'écran de TV ; ce qui permet d'avoir une idée de ce que peut donner uné animation avant de décider si on la conserve ou non.

2/ Les images constituant une animation pourront être par la suite reprises par l'utilisateur pour subir des modifications (par exemple être coloriées sous TV ou TN).

3/ A partir des images obtenues et à partir d'autres images, on pourra décrire un quelconque scénario exploitable par le processeur SC.

```
< EXEMPLE D'UTILISATION DE TA;
<--------------------------;

$B;              < BIBLIOTHEQUE UTILISEE;

D                < PRIMITIVE 'D' ACTIVATION DE L'ENVOI EN ZDC;

&ARRIVEE; OAB    < DESSIN D'ARRIVEE SUIVI DE 'OAB';

&DEPART;  OAB    < DESSIN DE DEPART SUIVI DE 'OAB';

F                < FIN;

< SOUS-PROGRAMMES;

>DEPART( KEXGYG 222 X1Y1    (((((CMS)))))  )

>ARRIVEE(KEXGYG 111 X1Y1    (((((SMC)))))  )
```

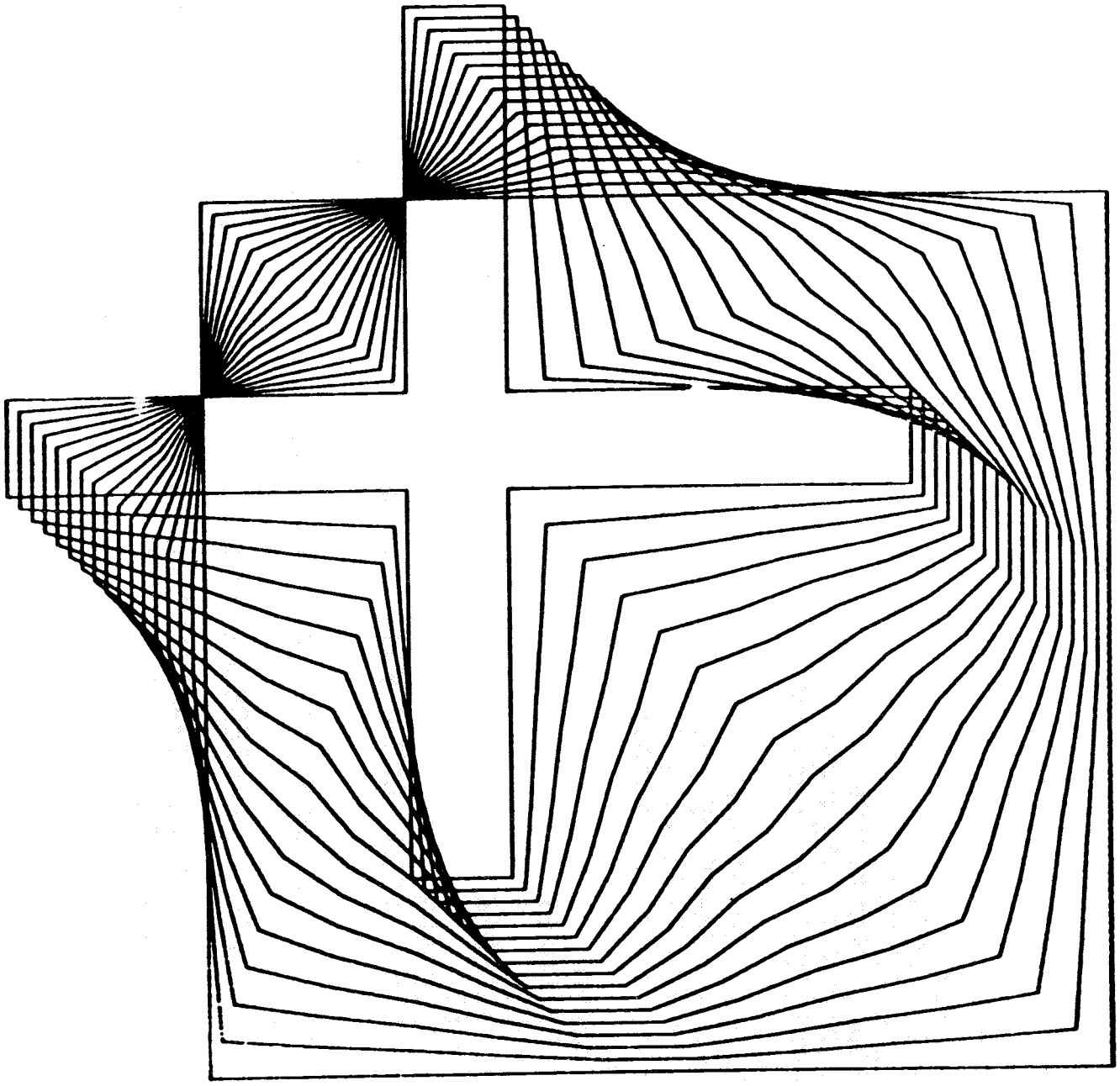
```
!TA
VIDEO?N
DELETE?N
TRAJET?N
EFFACEMENT?N
COPY?O
NBRE COPIES/TEMPORISATION=1F
CONVERSION VIDEO?N
PAS A PAS?N
NBRE DE PAS=7
TRANSFORMATION?N
```

TA 13

```
< EXEMPLE D'UTILISATION DE TA;
<--------------------------;

K8X9Y9 12        < DEFINITION DES ECHELLES ET POSITIONNEMENT;

D                < PRIMITIVE 'D' ACTIVATION DE L'ENVOI EN 2DC;

&ARRIVEE; 0AB    < DESSIN D'ARRIVEE SUIVI DE '0AB';

&DEPART;  0AB    < DESSIN DE DEPART SUIVI DE '0AB';

F                < FIN;

< SOUS-PROGRAMMES;

>DEPART( A%4(%9(1B)TR1;))

>ARRIVEE(12%4(12) A%4(%4(4B)1B%4(2B)TR1;))
```

```
!TA
VIDEO?N
DELETE?N
TRAJET?N
EFFACEMENT?N
COPY?0
NBRE COPIES/TEMPORISATION=1F
CONVERSION VIDEO?N
PAS A PAS?N
NBRE DE PAS=10
TRANSFORMATION?N
```

TA 16



Exemple d'utilisation de TA en vidéo.

```
!TA
VIDEO?O
PAS A PAS?O
NBRE DE PAS=4
TRANSFORMATION?N
ARRIVEE RESIDENTE?N
ARRIVEE>IMAGE ARRIVEE
N
DEPART SCRATCH?N
DEPART>IMAGE DEPART
#POINTS DEPART=00AC
#POINTS ARRIVEE=00AC
N
SAVE?O
NOM>IMAGE1
SAVE?O
NOM>IMAGE2
SAVE?
NOM>IMAGE3
SAVE?N   (NON, CAR C'EST L'IMAGE D'ARRIVEE...)
!

NOTA   ON POURRA UTILISER CES IMAGES DANS UN SCENARIO EXPLOITABLE
----
       PAR LE PROCESSEUR SC ET CONTENANT PAR EXEMPLE CECI:

>T(!1) < S/P DE TEMPORISATION; %Z(%Z(IMAGE DEPART;&T;IMAGE1;&T;
IMAGE2;&T;IMAGE3;&T;IMAGE ARRIVEE;&T;IMAGE3;&T;IMAGE2;&T;IMAGE1;&T;))F
```

PROCESSEUR TB

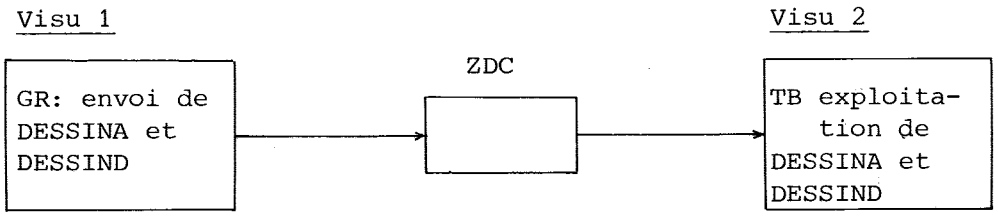
(INTERPOLATEUR GRAPHIQUE AVEC ENVOI EN ZDC)

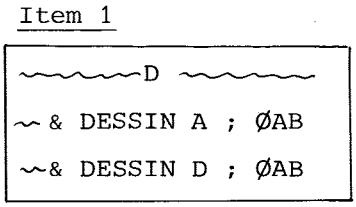
Fonction :

TB contient toutes les fonctions du processeur TA mode graphique ; de plus il autorise l'envoi en ZDC de tous les segments des dessins intermédiaires ; à la demande, il intercale un segment nul à l'origine (ØAB) à la fin de chaque dessin intermédiaire.

On peut donc récupérer les dessins intermédiaires par un processeur consommateur de segments ZDC.

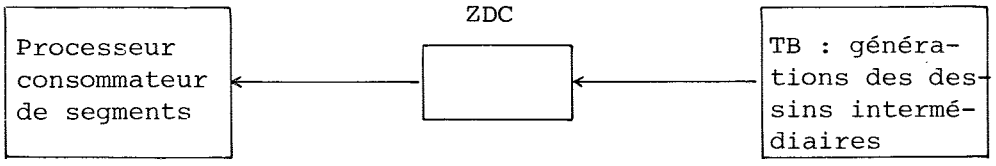
Exemple :





On envoie les dessins d'arrivée et de départ à TB ; celui-ci les exploite de la même façon que le ferait TA.

Puis TB génère les dessins intermédiaires en ZDC, ce qui permet de les récupérer.



Utilisation :

Le dialogue avec TB est très voisin de celui avec TA mode graphique. On ne précisera ci-dessous que ce qui est spécifique à TB.

! TB

DELETE ? { Ø / N — Remise à zéro ZDC.

TRAJET ? { Ø / N — Trajectoire des segments.

EFFACEMENT ? { Ø / N — Effacement entre chaque dessin intermédiaire.

CØPY ? { Ø / N — hard-copy/ caméra ; si oui :
NBRE DE CØPIES/ TEMPØRISATION = nt

CØNVERSION VIDEØ ? { Ø / N — conversion vidéo.

PAS A PAS ? { Ø / N — pas à pas

ØAB INTERMEDIAIRES? { Ø — oui : à la fin de chaque dessin intermédiaire, TB générera un segment nul à l'origine.
N — non : TB ne générera pas ce segment.

Le choix de la réponse à cette question dépend de l'utilisation qu'on veut faire des segments produits par TB.

Supposons par exemple qu'on veuille créér, par DG :

. autant de programmes graphiques qu'il y a de dessins intermédiaires dans l'interpolation : répondre Oui.

. un seul programme graphique regroupant l'ensemble des dessins intermédiaires: répondre Non.

NBRE DE PAS = { X D^C / X X — nombre de pas (en hexadécimal sur 1 ou 2 chiffres)

TRANSFORMATION ? { Ø / N — transformation éventuelle (programme dans Item 2).

A ce stade, TB efface l'écran, exploite les segments ZDC : il trace donc le dessin d'arrivée ; quand celui-ci est achevé ; il prévient : en envoyant un signal sonore (cloche) ;

N répondre N pour poursuivre le travail.

TB lit en ZDC le dessin de départ qu'il trace.

Les dessins d'arrivée et de départ ayant été tracés, TB commence l'interpolation avec envoi ZDC ; pour le reste il fonctionne comme TA, jusqu'à la fin de travail.

! fin de travail.

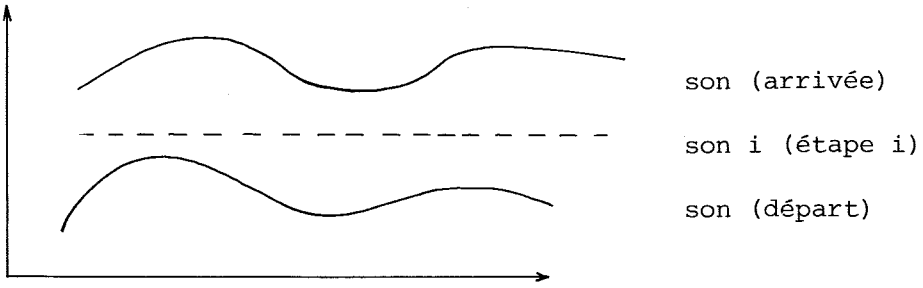
PROCESSEUR TØ

Fonction :

Tout comme le processeur TA (second mode), TØ interpole entre deux images ; la différence avec TA tient en ceci :

TA interpole de point à point alors que
TØ interpole { d'octet à octet (8 bits)
ou de mot à mot (16 bits)

Ce type d'interpolation, qui n'a pas de sens pour une image, a du sens pour du son : on obtient ainsi un effet de fondu enchaîné entre deux sons.



Utilisation de TØ.

Le dialogue avec TØ est presque identique au dialogue avec TA second mode. La seule différence concerne le début du dialogue :

! TØ

OCTETS ? { Ø ⟶ Oui : interpolation octet à octet (8 bits)
N ⟶ Non : alors interpolation mot à mot (16 bits)

PAS A PAS ? { Ø ⟶ pas à pas
N

etc... la suite du dialogue est la même que pour TA second mode (interpolation entre deux images).

PAGE BLANCHE SUITE A ERREUR DE PAGINATION.

PROCESSEUR KØ

Fonction

KØ permet l'extraction et le remplissage de contours connexes. Ces contours peuvent être complexes comme le montrent les exemples ci-après.

Qu'est-ce qu'un espace (de points) connexe ?

C'est un espace de points tel que, quels que soient Pi et Pj appartenant à l'espace, il existe un chemin entre Pi et Pj.

Image 1

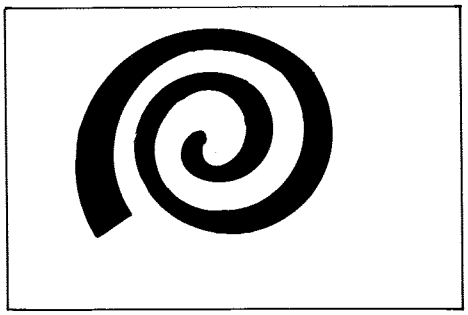


Image 2

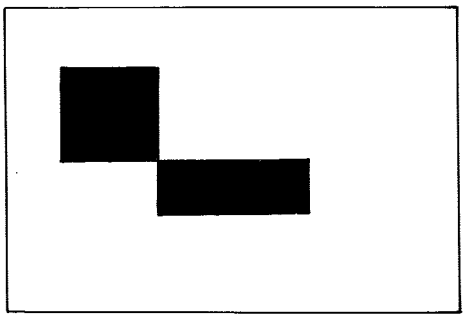


Image 3

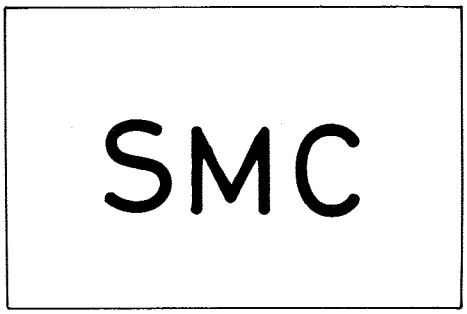
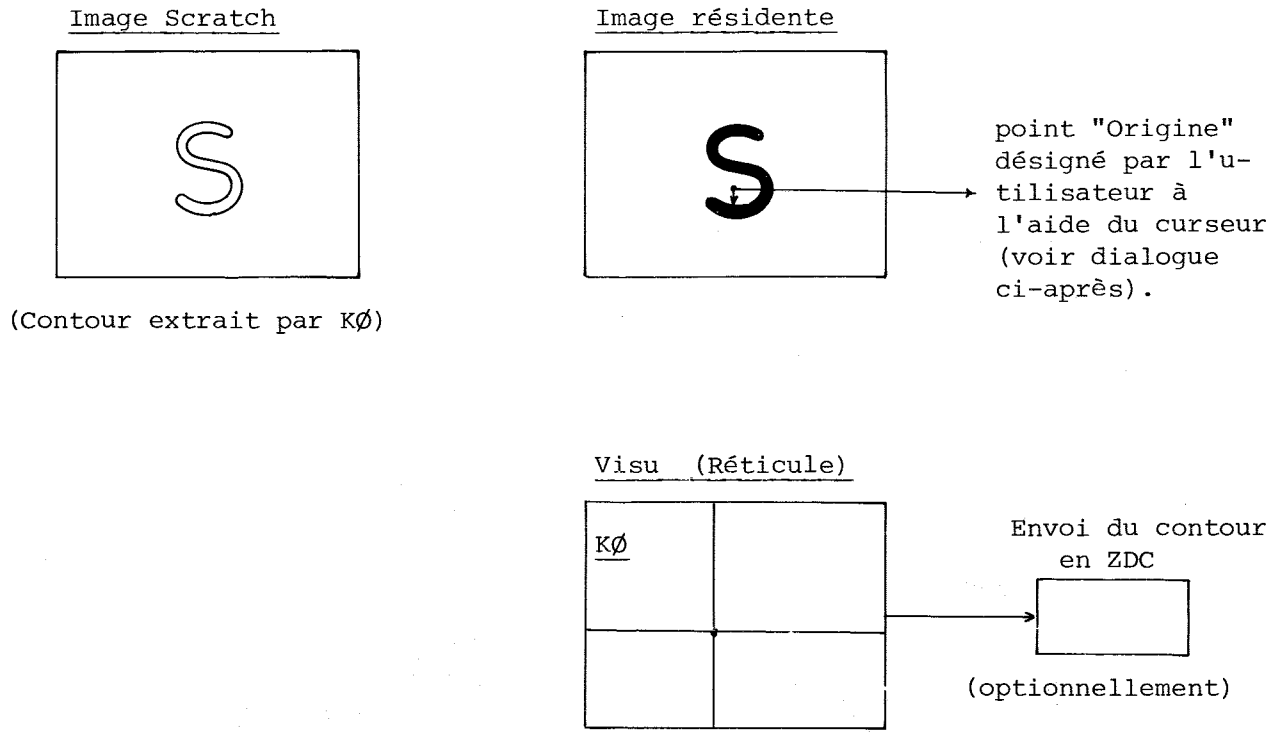


Image 1 et Image 2 contiennent chacune un espace connexe dont KØ saura extraire le contour en une seule fois. Image 3 contient un espace de points non-connexe . KØ saura extraire le contour de chaque sous espace connexe de cette image, ceci en 3 fois.



Le contour recherché par KØ est le contour externe.

Après extraction de ce contour, KØ propose :

- soit de le garder (il est disponible en image scratch)
- soit de le remplir, auquel cas on obtiendra :

le contour rempli en image résidente, et le "squelette" de l'image en image scratch. (Voir ci-après : utilisation de KØ).

Utilisation

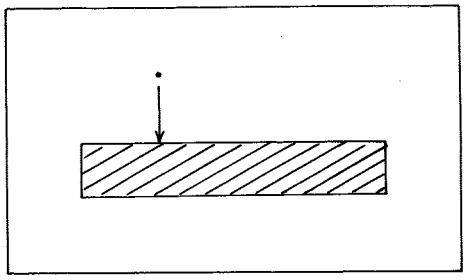
! SE

TYPE > 0 → mise à 0 éventuelle de la ZDC

! KØ

ZDC ? { Ø → oui : KØ enverra le contour en ZDC ; il sera donc récupérable par un processeur consommateur de segments.
N

Ici, KØ fait apparaître le réticule de la visu ; l'utiliser pour désigner le point de départ de la recherche du contour. Ce point étant choisi, taper " Ø " (origine). KØ va chercher à la verticale de ce point, vers le bas, le point de départ du contour.



Lorsque KØ a terminé, il affiche "INTERIEUR" ou "EXTERIEUR" suivant que le curseur était à l'intérieur où à l'extérieur du contour, puis il propose :

CØNTØUR ?

- Ø → oui, le contour est en image scratch, et il est donc récupérable sous TV par exemple.
- N → non : KØ effectue le remplissage du contour et l'on obtient en fin de travail :
  - . en image résidente : le contour rempli.
  - . en image scratch : le "squelette de l'image".

! fin de travail.

Nota :

Après la fin de travail de KØ, pour visualiser l'image scratch, il suffit de la rappeler sous TV comme ceci :

! TV → appel de TV

NØM > D^C
>E
FØND = 0000
REC ? N

→ conservation image résidente

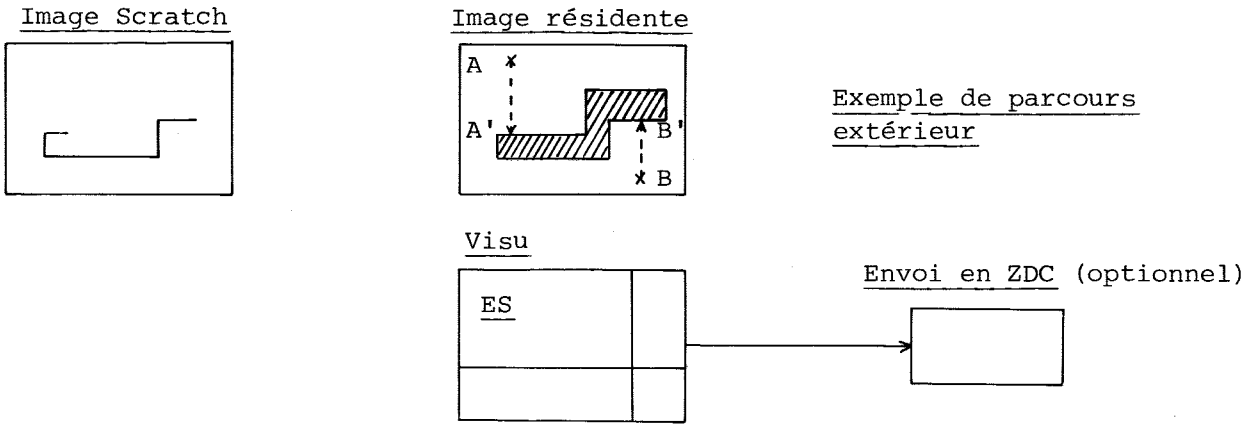
>U
NØM > D^C
ØRG = 0000
>

→ rappel image scratch

PROCESSEUR ES

Fonction

ES est un processeur de recherche de parcours sur une image TV.



Nota :

on peut demander
- un parcours extérieur, ou
- un parcours intérieur.

On lui fournit (à l'aide du réticule de la visu), deux points A et B (départ et arrivée). Pour chacun de ces points on indique une direction (haut/bas). A partir de ces deux points et des directions associées, ES détermine un point de départ : A' et un point d'arrivée : B' de recherche de parcours, le parcours trouvé est placé en image scratch et, optionnellement, il est émis en ZDC ce qui permet de la récupérer par un processeur consommateur de segments. On remarquera que si les points de départ et d'arrivée A' et B' sont confondus, le parcours obtenu est un contour.

Utilisation

! SE

TYPE > 0 → remise à 0 éventuelle de la ZDC.

! ES

PARCØURS EXTERIEUR ?
- Ø → parcours extérieur
- N → parcours intérieur.

Nota : les points de départ et d'arrivée devront respecter ce choix ; sinon ES émettra un message d'erreur.

ZDC ?

- Ø → émission du parcours en ZDC. ES demande alors :

  PAS = D^C ou Return → ES prendra 1

  n D^C → 1 à 4 chiffres hexa ; cccl indique la précision demandée à ES pour l'envoi en ZDC : Si PAS = p, ES envers les coordonnées du point courant si elles diffèrent de p points au moins en X ou en Y de celles du point précédent.

- N → pas d'émission ZDC.

DEPART

Ici ES fait apparaître le réticule ; choisir sur l'image un point de départ (curseur, commande C), et, sur le point choisi, frapper "Ø"
ES demande alors :

DEPART	Ø
VERS LE HAUT ?	Ø / N

VERS LE HAUT ?

- Ø → vers le haut
- N → vers le bas.

c'est la direction de recherche d'un point de départ de parcours.

Ici, deux cas se présentent :

. il y a erreur (ES n'a pas trouvé de point de départ : il prévient et propose à nouveau : "DEPART")

. ES a trouvé un point de départ ; il demande alors :

ARRIVEE

Ici, ES fait apparaître le réticule, choisir un point d'arrivée et un sens de la même façon que ci-dessus pour le point de départ. Quand on a répondu, ES recherche le parcours, et, lorsqu'il a terminé, il annonce :

ARRIVEE	Ø
VERS LE HAUT ?	Ø / N

[PASSAGE AU DEPART] éventuellement ; ceci dans le cas où point de départ et point d'arrivée coïncident, puis :

- CØNCAVE / CØNVEXE → en fonction de la forme du contour extrait.
- ! TV → fin de travail de ES. Appel de TV pour visualiser l'image scratch où ES à placé le contour.

NØM > D^C
> E
FØND = 0000
REC ? N
>U
NØM > D^C
ØRG = 0000
>

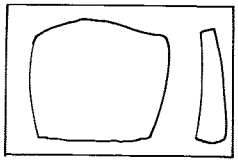
→ conservation image résidente.

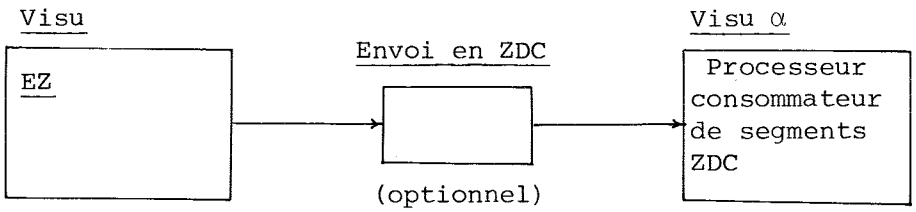
→ rappel image scratch.

PROCESSEUR EZ

Fonction :

Image résidente





EZ permet l'extraction exhaustive de tous les contours d'une image de la façon suivante :

EZ balaie l'image à la recherche d'un point blanc ; quand il l'a trouvé, il extrait un contour $\mathscr{C}$ fermé à partir de ce point et le transmet (optionnellement) en ZDC, puis Image Résidente ← (Image Résidente - $\mathscr{C}$)

et ainsi de suite jusqu'à ce que Image Résidente soit vide.

Le balayage par EZ est fait de haut en bas et de gauche à droite.

Les contours obtenus peuvent être approximatifs (paramètre : "PAS =" du dialogue).

Utilisation

! SE

TYPE > 0 → remise à zéro éventuelle de la ZDC

! EZ

ZDC ?
- Ø → envoi en ZDC ; EZ demande alors :
  - PAS=
    - DC ou Return → EZ prendra PAS=1.
    - p → 1 à 4 chiffres hexa.
- N

ceci indique la précision demandée à EZ pour l'envoi en ZDC : si PAS = p, EZ enverra en ZDC les coordonnées du point courant seulement si elles diffèrent de p points au moins en X ou en Y de celles du point précédent.
puis :

ØAB INTERMEDIAIRES ? { Ø → oui, ainsi chaque contour envoyé / N en ZDC sera

délimité par un segment nul à l'origine (ØAB) ce qui permet au processeur consommateur de distinguer entre eux les contours successifs.

PØINTS ISØLES ? { Ø → oui, traiter même les points isolés. / N → ignorer les points isolés.

Ici, EZ extrait successivement tous les contours jusqu'à épuisement de l'image résidente.

! → fin de travail

Exemple d'utilisation de EZ et LP.

On veut obtenir un fichier graphique utilisable sous LP qui contienne toute une image TV. Le but est de lire ce fichier sous LP, avec sortie vidéo active, de façon à voir se construire progressivement l'image TV par une succession de tracés de contours.

Il suffit de procéder comme suit, en supposant :

- IMAGE image à exploiter.
- FI ⌀ IMAGE : fichier équivalent à IMAGE.

1°) Sur Visu 1 : display IMAGE sous TV.

```
!TV
NØM > IMAGE D^C
> D
> F

!
```

2°) Sur Visu 1 : remise à zéro ZDC et lancement de EZ.

!SE		
TYPE > 0	→	remise à zéro ZDC
!EZ	→	appel de EZ
ZDC ? Ø	→	option ZDC active
ØAB INTERMEDIAIRES ? N	→	pas d'ØAB intermédiaires (on n'isole pas les contours les uns des autres pour pouvoir les récupérer globalement par LP dans un même fichier).
PØINTS ISØLES ? Ø	→	ainsi, on conservera notamment les gris de IMAGE.

Ici, EZ analyse IMAGE et envoie en ZDC les segments (ou points) extraits. Sur une autre visu (Visu 2), on va les consommer par LP.

3°) Sur Visu 2 : lancer LP et créer FI Ø IMAGE.

!LP	→	appel de LP
SGF ? Ø	→	option SGF active ; un fichier temporaire LP sera donc créé.
> G	→	commande G, avec consommation des segments en ZDC.
ZDC ? Ø		

Ici se déroule le processus d'échange de segments par ZDC entre EZ et LP. Quand EZ a extrait tous les contours successifs de l'image, il envoie un ØAB qui provoque la fin d'échange ZDC. LP revient alors au dialogue :

> N	→	on nomme le fichier temporaire - LP pour en faire un fi-
NØM = FI Ø IMAGE D[C]		chier permanent réutilisable.
> F	→	fin ; on sort de LP
!		

4°) Exploitation du fichier FI Ø IMAGE.

Le fichier est utilisable sous LP. Pour obtenir l'effet recherché, qui est de voir se reconstituer l'image par contours successifs, il suffit de faire ceci :

EZ 4

! TV	→	appel de TV
NØM > D^C		
> R		réinitialisation à 0 de l'image résidente
FØND = 0000		
REC ? N		
> F		
! LP	→	appel de LP
SGF ? Ø	→	option SGF active
> V		
VISU ? N	→	sortie visu inactive
VIDEØ ? Ø	→	sortie vidéo active
> R	→	read fichier
FICHIER = FI ∅ IMAGE D^C		le fichier FI ∅ IMAGE est exploité par LP avec sortie vidéo active. IMAGE se reconstruit progressivement sur l'écran TV.
SEGMENT 1 = 1 D^C		
SEGMENT 2 = 7 FFF		
ENVØI ZDC ? N		
PAS A PAS ? N		
>		

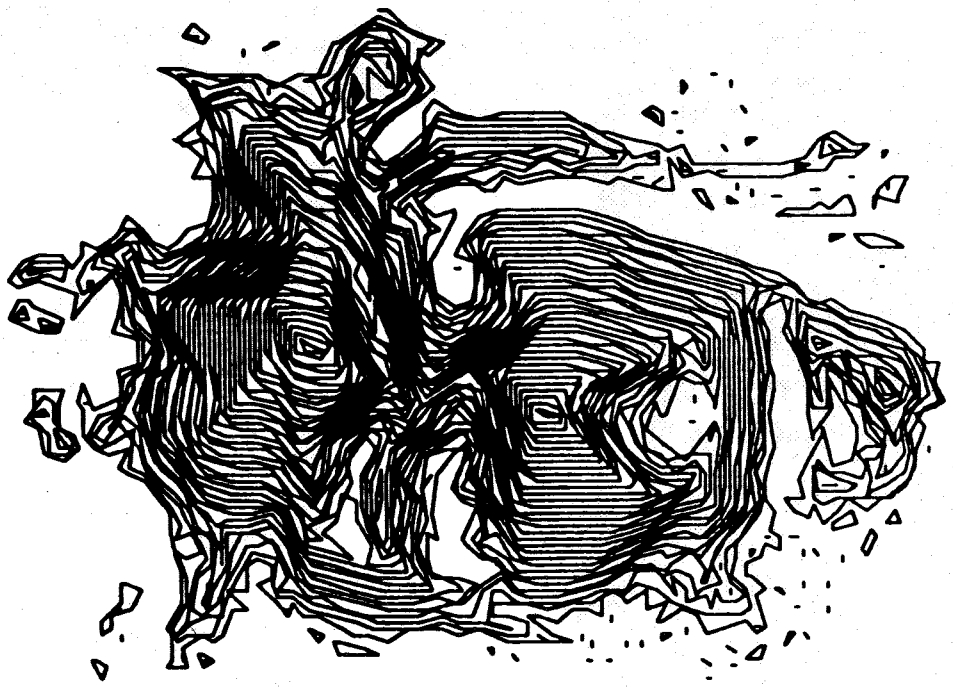
```
!TV
NOM>BOTTICELLI V
>D
>F
!EZ
ZDC?O
PAS=4
OAB INTERMEDIAIRES?N
POINTS ISOLES?N
!
```

Exemple d'utilisation de EZ :

On charge sous TV l'image BOTTICELLI V (cette image est reproduite page suivante et constitue au passage un exemple d'utilisation de VG). Ensuite, on appelle EZ en lui demandant l'envoi en ZDC des contours qu'il extraira. Sur une autre console, on récupère ces contours par un processeur consommateur de segments ZDC. En l'occurrence, on les récupère par GF, ce qui permet d'appliquer une transformation (rapport 3/4 sur les Y) et de créer un fichier contenant l'ensemble des contours obtenus. Ce fichier, lu sous LP, donne le tracé ci-après (page EZ 7)

EZ 6

EZ 7



PROCESSEUR DG

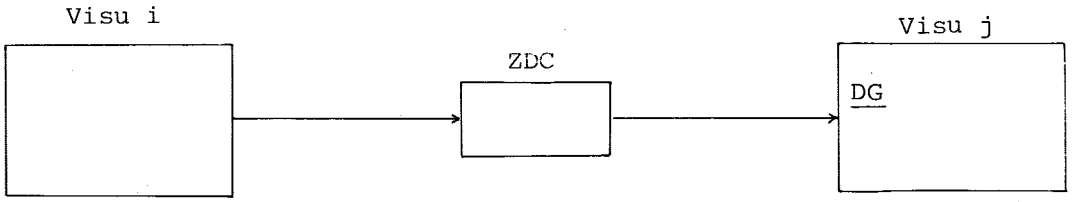
Fonction

DG est un processeur "consommateur" de segments en ZDC. A partir des coordonnées de segments qu'il va lire en ZDC, il génère dans Item 1 un programme graphique optimisé, c'est-à-dire contenant des interactions imbriquées. Le programme est généré en fonction des paramètres fournis à DG par l'utilisateur (voir ci-après : utilisation de DG).

Pour comprendre le fonctionnement de DG il convient de connaître :

- le langage graphique (voir processeur GR), notamment :
  - . les primitives de traçage de segments : A,B.
  - . les primitives K,X,Y de définition d'échelle.
  - . la primitive d'itération : % .
- le processus d'échange de coordonnées de segments en ZDC (voir généralités sur les processeurs de SMC).

Utilisation de DG



Processeur producteur de segments

DG, consommateur de segments, génère en Item 1 un programme graphique.

Avant de lancer (dans n'importe quel ordre) le processeur producteur de segments sur la visu i et le processeur DG sur la visu j, il faut réinitialiser la ZDC au moyen du processeur SE comme ceci :

! SE	⟶	appel de SE
TYPE > 0	⟶	remise à 0 de la ZDC

Ensuite on lance le processeur producteur de segments sur visu i et DG sur visu j. Le dialogue avec DG se déroule comme suit :

! DG → appel de DG.

EDITIØN ?
- Ø → Oui : le programme que DG va générer dans Item 1 sera visualisé au fur et à mesure de sa production sur l'écran de la visu.
- N → Non : le programme généré ne sera pas visualisé.

TRACE PØINTS ?
- Ø → Oui : si DG reçoit un segment de longueur nulle c'est-à-dire de coordonnées : $((x_1, y_1), (x_2, y_2))$ avec $x_1 = x_2$ et $y_1 = y_2$, DG générera la séquence < AB >.
- N → Non : si DG reçoit un tel segment réduit à un seul point, il ne générera rien.

A = < chaîne de caractères éventuellement vide > D^C → donner une chaîne de caractères éventuellement vide, de 9 caractères au maximum. C'est la chaîne que DG devra générer dans le programme graphique à chaque nouvelle origine de segment. Si l'on donne une chaîne vide, DG générera normalement la primitive A à chaque nouvelle origine de segment (il s'agit de la primitive A du langage graphique : voir processeur GR).

Si la chaîne est non vide par exemple : A = & SP1 ; D^C , alors DG au lieu de générer la primitive A générera la séquence < & SP1 ; >.

B = <chaîne de caractères éventuellement vide > donner, comme pour A une chaîne de caractères éventuellement vide, de 9 caractères au maximum. Si elle est vide, DG générera normalement la primitive B à chaque extrémité de segment (il s'agit de la primitibe B du langage graphique : voir processeur GR). Si la chaîne est non vide, par exemple : B = & SP2 ; D^C alors DG générera la séquence < & SP2 ; > au lieu de la primitive B.

REDUCTIØN # SEGMENTS ?

Ø ⟶ Oui : DG réduira à un seul segment les segments consécutifs, liés, de même direction et de même sens : c'est-à-dire qu'au lieu de générer par exemple :

<A % 9 (1) B % G (1) B 1 1 B >, il générera : < A % R (1) B >. Dans cet exemple on constate qu'on n'aura qu'un seul segment au lieu de trois.

N ⟶ Non : de tels segments seront générés tels quels sans réduction de leur nombre.

DEPLACEMENT = n ⟶ donner un chiffre hexadécimal (de o à F). Ce sera le déplacement dans Item 1 du premier caractère généré par DG.

K k
X x
Y y
⟶ DG demande les valeurs de K X et Y définissant un quadrillage au sens de GR. On rappelle que

k, x et y ∈ {1, 2,..., 9, A, B,..., Z}

Chaque coordonnée reçue par DG sera éventuellement arrondie pour s'ajuster au quadrillage ainsi défini ; un quadrillage large provoquera une "stylisation" du dessin.

DG commence alors la lecture des coordonnées de segments en ZDC et génère le programme en Item 1.

Si l'item 1 est complètement rempli alors que la transmission de coordonnées en ZDC n'est pas terminée, DG revient au ! ;

! DG
{
fournir les paramètres
}
Item 1 est plein, il y a encore des segments à consommer ; DG revient au !

! IS Sauvegarder Item 1

! I+ Créer un nouvel item

N Ø M >ITEM SUIVANT DC

! DG
{ et rappeler DG en lui fournissant les mêmes paramètres ou des paramètres différents.

{
DG continue de consommer les segments arrivant en ZDC.
}
etc...

Nota: DG termine son travail dès qu'il reçoit un segment nul à l'origine, c'est-à-dire de coordonnées : ((0,0),(0,0)). Autrement dit, un ØAB (au sens du langage graphique) est considéré comme une fin d'échange.

Exemple d'utilisation de DG:

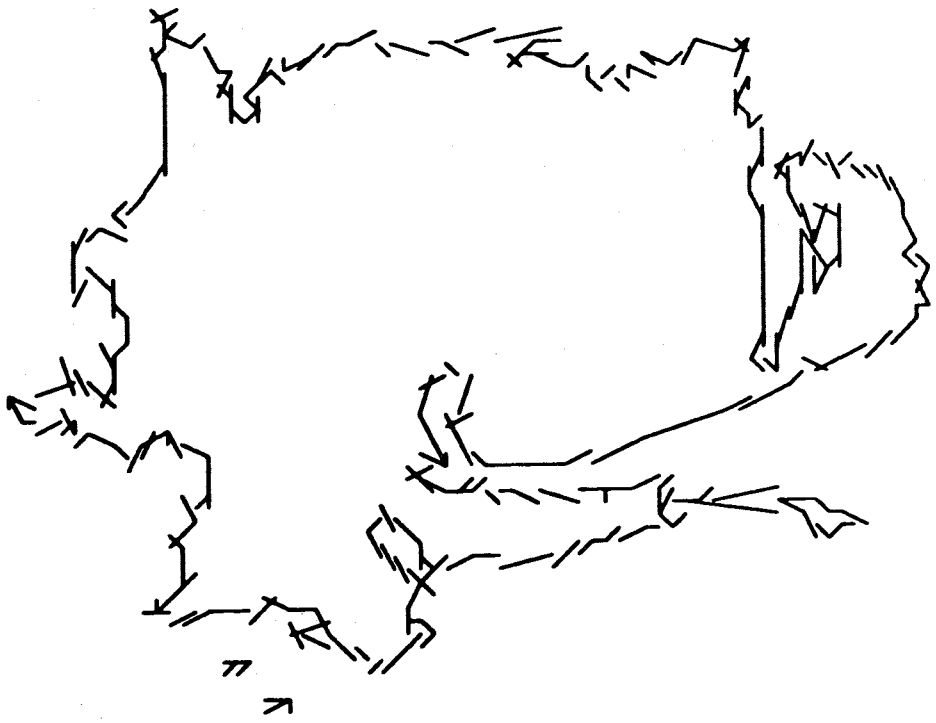
```
!I+
ITEM>ITEM EXEMPLE DG
!DG
EDITION?O
TRACE POINTS?N
A=&X;
B=&Y;
REDUCTION #SEGMENTS?O
DEPLACEMENT=F
K2
X4
Y4
```

Génération par DG dans un item d'un programme graphique, à partir de coordonnées de segments acquises en ZDC. Ces segments sont ici envoyés par le processeur LP et représentent des contours extraits par EZ de l'image BOTTICELLI V.

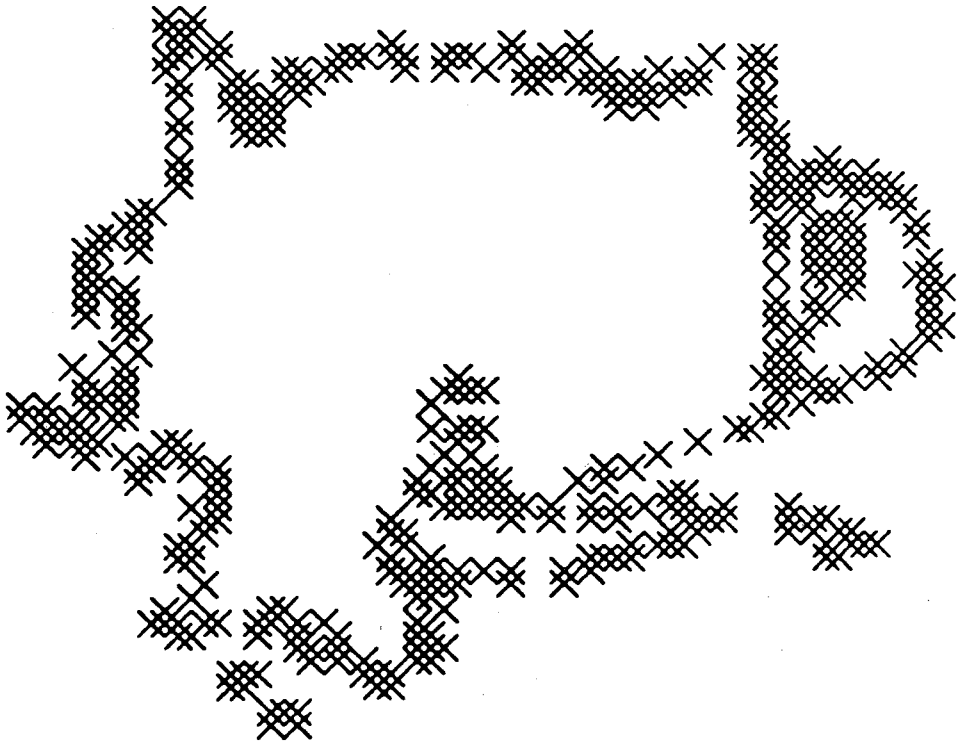
On trouvera en pages suivantes:

- le programme graphique obtenu (dans lequel on a inséré les définitions des sous-programmes X et Y).
- l'exécution de ce programme. On notera la différence entre ce qui est obtenu et l'"original" (cf: page LP 12), dûe uniquement à l'échelle choisie lors de l'appel de DG.
- nouvelle exécution de ce programme dans lequel les S/P X et Y ont été modifiés.

```
                K2X4Y4&X;%Z(1)111%Z(2)%O(2)&X;4&Y;4&Y;3&Y;1&X;322&Y;1111233
&X;4&Y;2&X;44&Y;32&Y;1&X;32&Y;1%B(4)&X;12&Y;3&X;1122&Y;3&X;122&Y;3&X;112&Y
;3&X;1444&Y;3&X;112&Y;22&Y;122&Y;4&X;32&Y;32&Y;1&X;222&Y;322&Y;14&X;322&Y
1&X;22&Y;4&X;1&Y;3&X;1144&Y;14&Y;32&X;111&Y;12&Y;3&X;14&Y;14&Y;112&Y;34&X;
14&Y;3&X;111&Y;3&X;11&Y;122&Y;3&X;112&Y;3&X;1&Y;322&Y;14&X;332&Y;1&X;332&Y
;1&X;12&Y;122&Y;32&Y;1&X;1&Y;3&X;112&Y;34&X;32&Y;1&X;322&Y;1&X;2&Y;112&Y;3
&X;1&Y;344&Y;1&X;1444&Y;3&X;1112&Y;34&X;334&Y;1&X;3344&Y;2&X;114&Y;3&X;111
&Y;3&X;112&Y;34&X;14&Y;11&Y;12&Y;3&X;111&Y;3&X;1122&Y;3&X;332&Y;1&X;1&Y;1&
Y;3&X;111&Y;3&X;114&Y;3&X;14&Y;344&Y;1&X;12&Y;14&Y;3&X;14&Y;11144&Y;3&X;11
&Y;11&Y;1&Y;111&Y;3&X;1112&Y;3&X;14&Y;11&Y;12&Y;3&X;144&Y;3&X;32&Y;344&Y;1
4&Y;3&X;334&Y;4&Y;332&Y;1&X;34&Y;1&X;333&Y;1&X;34&Y;14&Y;3&X;11&Y;3&X;12&Y
;1144&Y;32&X;114&Y;3&X;34&Y;1&X;1&Y;3&X;144&Y;32&X;1144&Y;3&X;14&Y;4&Y;144
&Y;3&X;34&Y;1&X;3444&Y;1&X;4&Y;344&Y;12&X;1444&Y;3&X;144444&Y;3&X;22&Y;332
2&Y;1&X;34&Y;1&X;44&Y;2&X;344&Y;4&Y;14&Y;3&X;3&Y;34&Y;1&X;14&Y;3&X;34&Y;1&
X;1&Y;344&Y;1&X;12&Y;344&Y;14&Y;3&X;114&Y;3444&Y;14&Y;2&X;34&Y;1&X;3332&Y;
4&X;3&Y;1&X;332&Y;1&X;3&Y;1&X;33&Y;1&X;34&Y;3&Y;14&Y;3&X;3&Y;1&X;3&Y;33&Y;
1&X;332&Y;1&X;33&Y;334&Y;1&X;33&Y;1&X;333&Y;33&Y;3&Y;3&Y;333&Y;1&X;332&Y;3
4&Y;1&X;34&Y;11&Y;3&X;11&Y;3&X;1114&Y;3&X;1114&Y;3&X;111&Y;11&Y;3&Y;1&X;33
344&Y;12&X;333&Y;114&Y;14&Y;3&X;11&Y;1&Y;11&Y;3&X;122&Y;4&X;3332&Y;1&Y;3&X
;1112&Y;3&X;112&Y;11&Y;3&X;12&Y;3&X;114&Y;32&X;144&Y;3&Y;114&Y;3&X;34&Y;1&
X;334&Y;1&X;14&Y;3&X;34&Y;1&X;34&Y;1&X;334&Y;1&X;334&Y;3&Y;334&Y;32&Y;34&Y
;1&X;34&Y;332&Y;1&X;334&Y;2&Y;3&Y;3322&Y;1&X;3322&Y;1&X;322&Y;322&Y;4&X;12
2&Y;3&X;32&Y;322&Y;322&Y;14&X;322&Y;3222&Y;3222&Y;322&Y;322&Y;1&X;322&Y;%6
(2)&Y;12&Y;3&X;112&Y;112&Y;3&X;144&Y;2&X;1114&Y;2&X;12&Y;3&X;322&Y;14&X;33
32&Y;334&Y;334&Y;1&X;3&Y;122&Y;34&X;322&Y;1&X;3344&Y;12&X;344&Y;44&Y;2&X;1
4&Y;3&X;14&Y;3&X;34&Y;1&X;4&Y;344&Y;1&X;3444&Y;1&X;44&Y;3&Y;1&X;44&Y;144&Y
;3&X;14&Y;32&X;33&Y;114&Y;3&X;44&Y;14&Y;3&X;144444&Y;3&X;44&Y;4&Y;34&Y;4&Y
;344&Y;2&X;2&Y;32&Y;12&Y;3&X;112&Y;122&Y;3&X;1%7(2)&Y;3&X;32&Y;12&Y;3&X;2&
Y;322&Y;1&X;32&Y;2&Y;32&Y;1&X;3322&Y;1&X;32222&Y;1&X;22&Y;322&Y;1&X;3322&Y
;14&X;12&Y;3&X;11&Y;1122&Y;3&X;112&Y;3&X;32&Y;334&Y;1&X;334&Y;1&X;334&Y;1&
X;3344&Y;12&X;332&Y;33&Y;1&X;4&Y;34&Y;32&Y;1&X;32&Y;3322&Y;1&X;32&Y;
< DANS L'ITEM OBTENU, ON AJOUTE LA DEFINITION DES S/P 'X' ET 'Y';
>X(A)  >Y(B)   PAR EXEMPLE...;
```



```
!IL
!ITEM>ITEM EXEMPLE DG
!GR
```



```
!IL
!ITEM>ITEM EXEMPLE DG
!GR
```

PROCESSEUR LP

Fonction

Le processeir LP permet de générer des dessins :

. sur visu graphique (avec éventuellement sortie vidéo).

. sur écran vidéo

. sur fichier "temporaire LP" (optionnellement)

. sur fichier permanent (à partir d'un fichier "temporaire LP" que l'on peut cataloguer en lui donnant un nom au moyen de la commande N).

. en ZDC (optionnellement) ce qui permet d'exploiter par un processeur "consommateur de segments" les coordonnées de segments émises par LP.

à partir :

- du réticule de la visu (curseur).
- du light-pen sur écran TV.
- de segments ZDC.
- d'un fichier graphique.

De plus, LP permet d'envoyer un fichier "temporaire LP" en ZDC en autorisant la modification du nombre de segments qu'il contient (ajout/Suppression : très utile pour "ajuster" des dessins entre lesquels on veut faire des interpolations par TA ou TB ; autrement dit pour faire en sorte que deux dessins aient le même nombre de segments). Cette émission des segments contenus dans un fichier "temporaire LP" se fait au moyen de la commande S.

LP est donc un processeur dont les utilisations possibles sont multiples. On trouvera ci-après des explications sur les différentes commandes utilisables sous LP, avec leurs différents modes d'utilisation.

Pour bien comprendre le fonctionnement de DG, il convient de bien connaître les généralités sur les processeurs de SMC : processus d'échange de segments en ZDC, différentes formes d'un dessin (programme graphique, fichier graphique, matrice de points).

Utilisations de LP

■ *Commande V : Autorisation/intribition des sorties graphique et vidéo.*

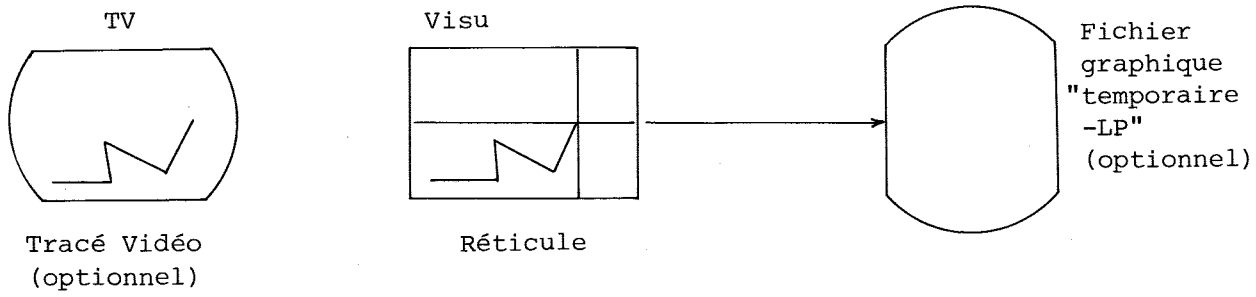
! LP		→ appel de LP
SGF ? N		→ Non ; on ne veut pas générer de fichier temporaire LP ; cette option est précisée ci-après à propos des autres commande de LP.
> V		→ Commande V ; choix des sorties
VISU ?	Ø	→ Oui ; les dessins que l'on traitera par la suite (voir les autres commandes) apparaîtront sur l'écran graphique.
	N	→ Non ; ces dessins n'apparaîtront pas sur l'écran graphique.
VIDEØ ?	Ø	→ Oui ; les dessins que l'on traitera par la suite apparaîtront sur l'écran de télévision.
	N	→ Non ; ces dessins n'apparaîtront pas sur l'écran de télévision.
>		→ suite de l'utilisation de LP ; voir ci-après les autres commandes de LP.

Nota :Initialement, c'est-à-dire à l'appel de LP, on a :

- sortie visu active
- sortie vidéo inactive.

■ *Commande G*

- Entrée dessin au curseur de la visu :



! LP

SGF ? { Ø → on veut que soit généré un fichier temporaire LP
N → on ne veut pas que soit généré un fichier temporaire LP

> G

ZDC ? N → ce n'est pas une entrée ZDC

VIDEØ ? N → ce n'est pas une entrée Light pen

CURSEUR ? Ø → c'est une entrée curseur.

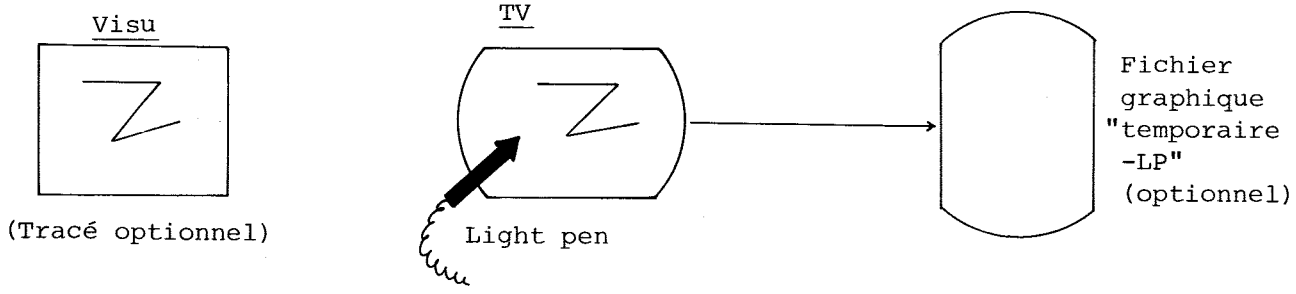
ici, le système affiche le réticule, renter le dessin en utilisant les primitives :

(voir VØ) {
A origine segment
B tracé segment ; nouvelle origine segment
S tracé segment ; origine courante inchangée
F fin de tracé.

> → un fichier temporaire LP est créé contenant ce dessin.

■ *Commande G (suite)*

- Entrée dessin au light-pen sur écran TV



! LP

SGF ? { Ø → option SGF active
N → option SGF inactive.

> G

ZDC ? N → ce n'est pas une entrée ZDC

VIDEØ ? Ø → c'est une entrée vidéo, light pen

SEUIL = nnnn → seuil S exprimé sur 4 chiffres hexadécimaux. Si le nuage de points généré par le light pen sur l'écran contient moins de S points, le point désigné par le light pen est ignoré ; sinon il est pris en compte (c'est le"centre de gravité" du nuage de points).

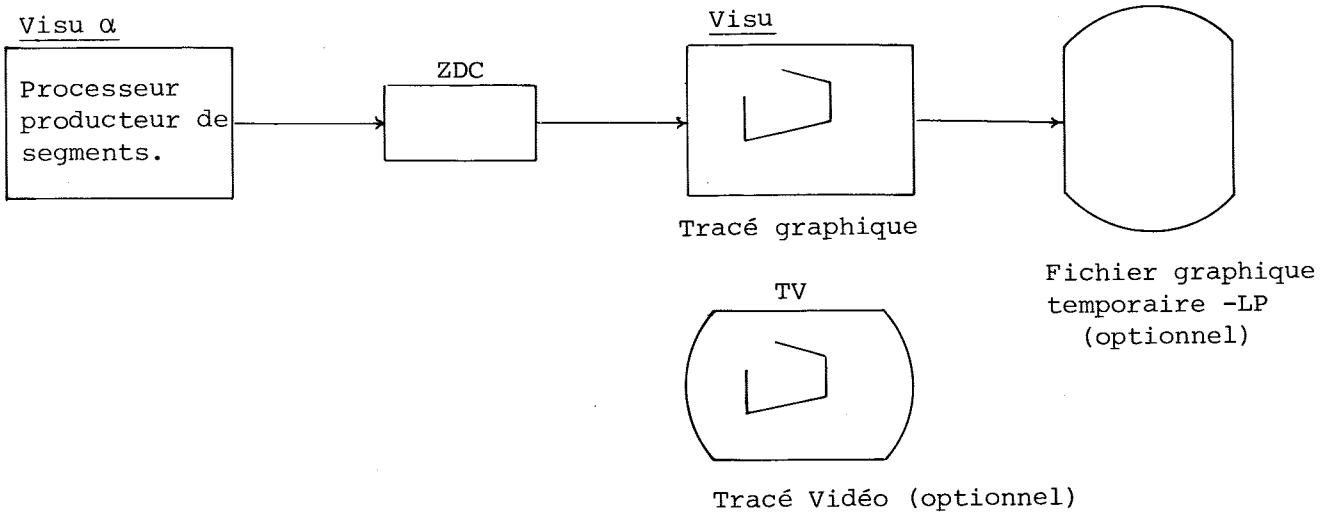
Ici, on entre un dessin au light pen ; interrompre par 1 alt-mode.

> → si l'option SGF est active, un fichier temporaire LP est créé et contient le dessin.

Nota : l'image résidente est récupérable sous TV.

■ *Commande G (suite)*

- Entrée dessin à partir de la ZDC



! LP

SGF ? { Ø → fichier temporaire LP sera créé
N → pas de fichier

> G

ZDC ? Ø → c'est une entrée ZDC

> → fin de réception ZDC ; fichier temporaire LP éventuellement créé. La fin de réception est provoquée par l'acquisition en ZDC d'un segment nul à l'origine (ØAB).

■ *Commande N : Nommer un fichier temporaire LP*

! LP

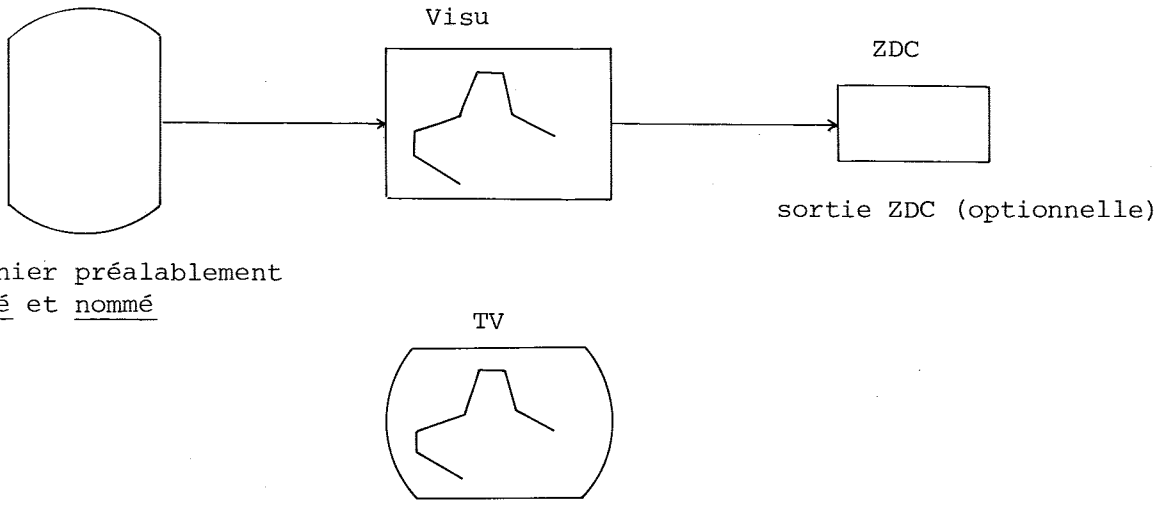
SGF ? Ø

>G ⟶ génération d'un fichier temporaire LP par l'un des modes décrits précédemment (réticule , light pen, ZDC)

≥ N

NØM = nom fichier D^C ⟶ nom qu'on veut donner au fichier. Ce fichier pourra ensuite être lu sous LP par la commande R (Read).

■ *Commande R : Read fichier*



Sous LP, commande R, on a la possibilité :

. de ne tracer qu'un sous-ensemble des segments.

. de compter les segments.

. de tracer les segments en pas à pas.

! LP

SGF ? Ø

> R

FICHIER = nom fichier D^C

SEGMENT 1 = n 1 → sur 1 à 4 chiffres hexa : n° du premier segment à tracer.

SEGMENT 2 = n 2 → sur 1 à 4 chiffres hexa : n° du dernier segment à tracer.

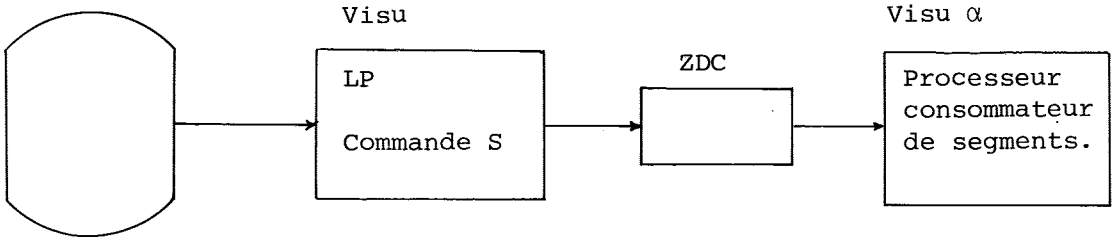
COMPTAGE ?
- Ø → comptage des segments (voir exemples)
- N → pas de comptage

ENVØI ZDC ?
- Ø → envoi ZDC
- N → pas d'envoi ZDC

PAS A PAS ?
- Ø → oui : LP tracera les segments un par un ; à la fin de chaque segment il s'arrêtera et demandera : > Ø répondre oui pour le tracer (voir exemples)
- N

> Ø envoi d'un segment nul à l'origine pour terminer la transmission (voir processus d'échange de segments ZDC).

■ *Commande S : Send fichier "temporaire LP "*



Fichier temporaire LP venant d'être créé.

Sous LP commande S on a la possibilité :

. d'ajout ou suppression de segments

. de tracé "point-pont"

. d'envoyer des " ØAB intermédiaires", c'est-à-dire un segment nul à l'origine entre chaque segment.

! LP	→	appel de LP
SGF ? Ø	→	option SGF active
>	→	génération d'un dessin sur fichier temporaire LP par l'un des modes décrits précédemment (réticule, light-pen, ZDC, Read fichier).
> S	→	Send fichier temporaire LP
# SEG GENERES = 024 A	→	nombre de segments contenus dans le fichier.
# SEGMENTS = 0200	→	nombre de segments à obtenir (par ajout ou suppression automatiques).
ENVØI ZDC ? Ø	→	envoi ZDC
POINT-POINT ? Ø	→	oui, tracé point-point
POINT-POINT ? N	→	non, tracé normal
ØAB INTERMEDIAIRES ? Ø	→	LP intercalera un segment nul à l'origine entre chaque segment du dessin
ØAB INTERMEDIAIRES ? N	→	LP enverra normalement les segments sans intercaler d'ØAB intermédiaires.

ici, l'échange de segments entre LP et le processeur consommateur s'effectue. Lorsque tous les segments ont été envoyés le dialogue se poursuit avec LP :

> Ø	→	Commande "Origine". LP envoie le segment nul à l'origine en ZDC. Cela permet d'achever le processus d'échange de segments avec le processeur consommateur. (voir généralités sur les processeurs de SMC, échange de segments en ZDC).

ADDITIFS LP 1/8/79

Sur Solar, on a le choix pour l'entrée vidéo et pour la sortie vidéo entre les 3 composantes RVB :

- Commande G, choix de l'entrée vidéo :

≥ G

ZDC ? N

VIDEØ ?
- N → non
- Ø → oui
- 5 → entrée à partir de composante RØUGE
- 6 → " " " " " VERT
- 7 → " " " " " BLEU

si l'on répond Ø (oui), l'entrée se fera à partir de la composante VERT (donc les réponses Ø et 6 sont équivalentes).

- Commande V, choix de la sortie vidéo :

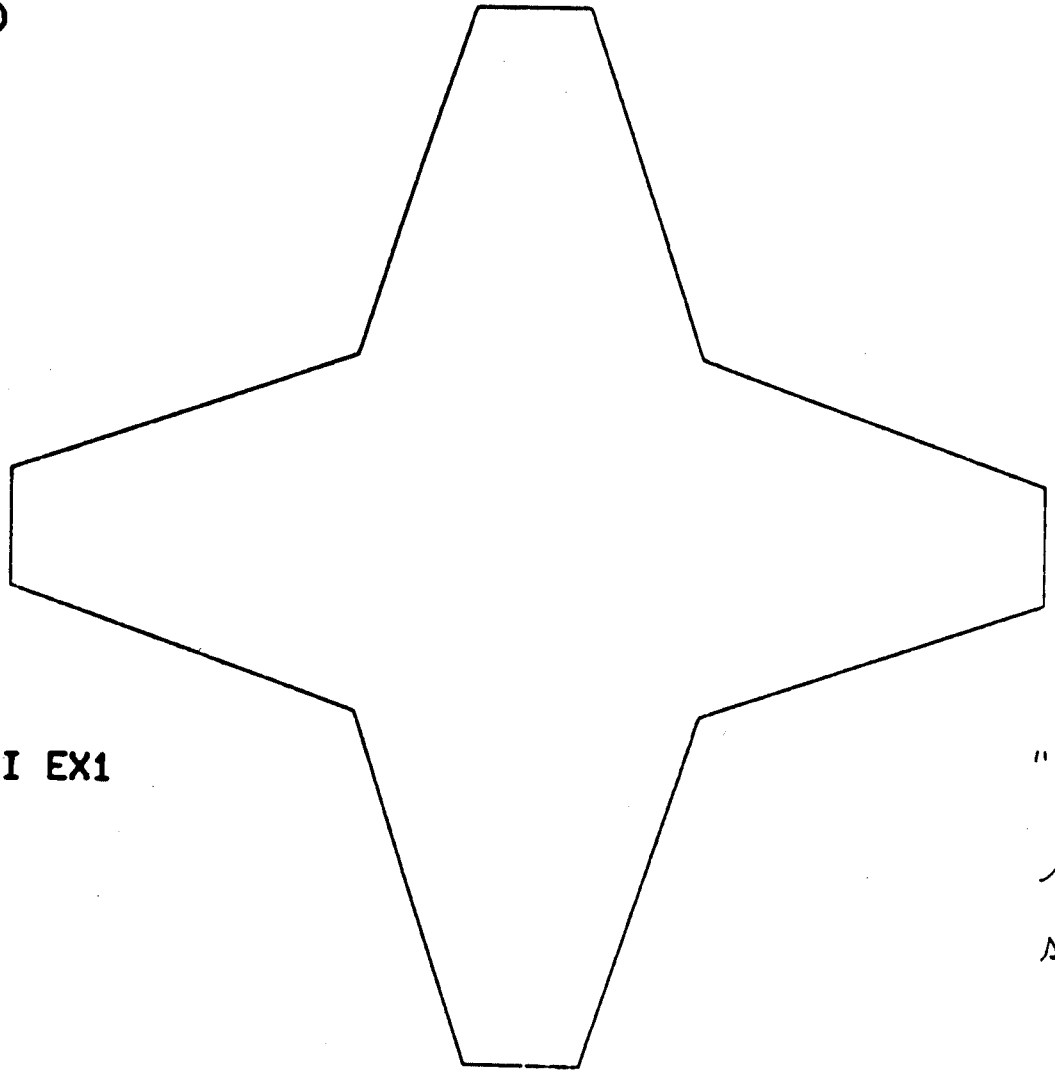
≥ V

VISU ?
- Ø
- N

VIDEØ ?
- N → non
- Ø → oui
- 1 → sortie sur la composante RØUGE
- 2 → " " " " VERT
- 3 → " " " " BLEU

si l'on répond Ø (oui), la sortie se fera sur la composante VERT, (donc les réponses Ø et 2 sont équivalentes).

```
!LP
SGF?O
>G
ZDC?O
```

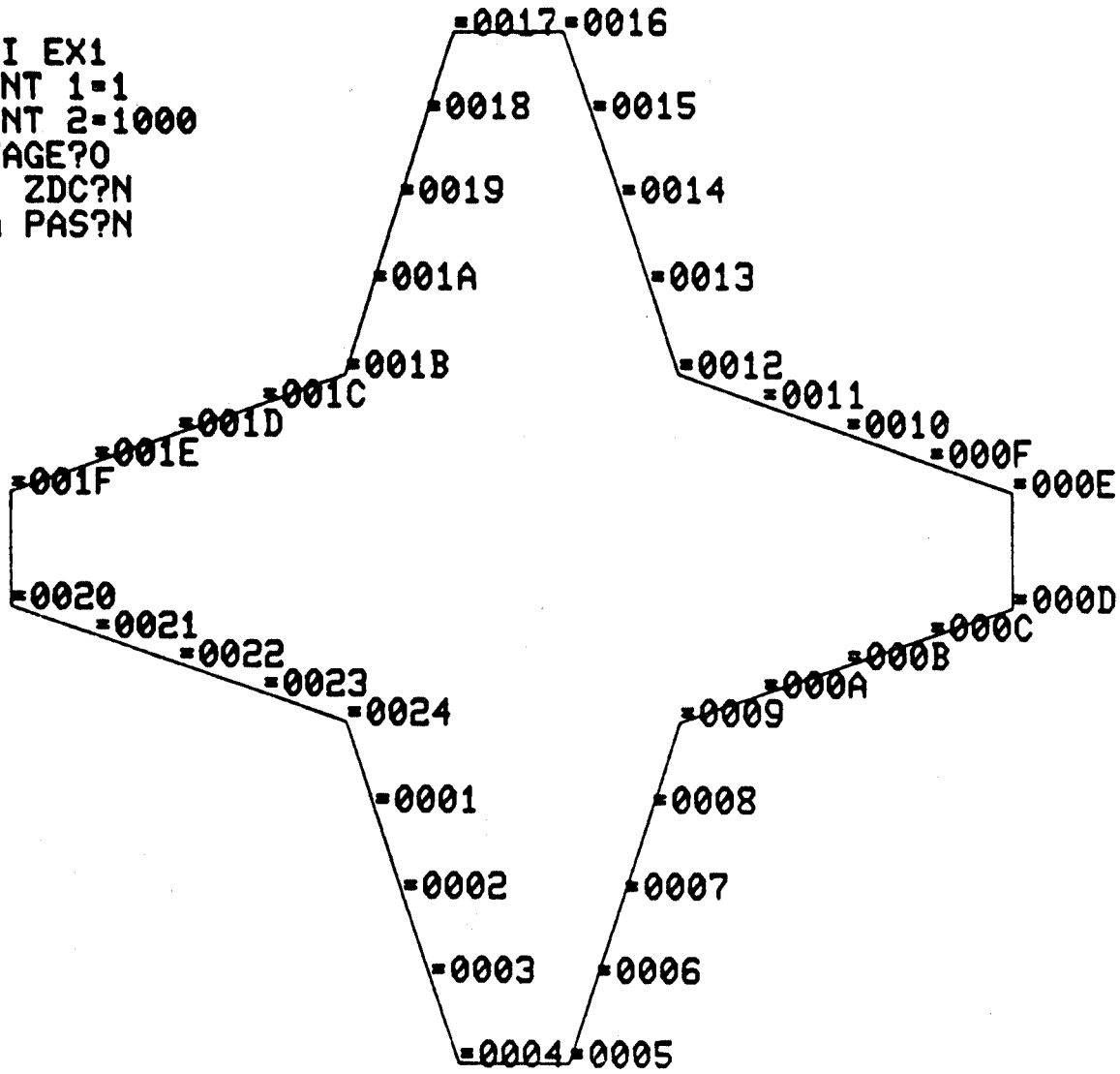


Acquisition en ZDC d'un dessin.

```
>N
NOM=FI EX1
>
```

"Baptême" du fichier temporaire - LP obtenu sous le nom "FI EX1".

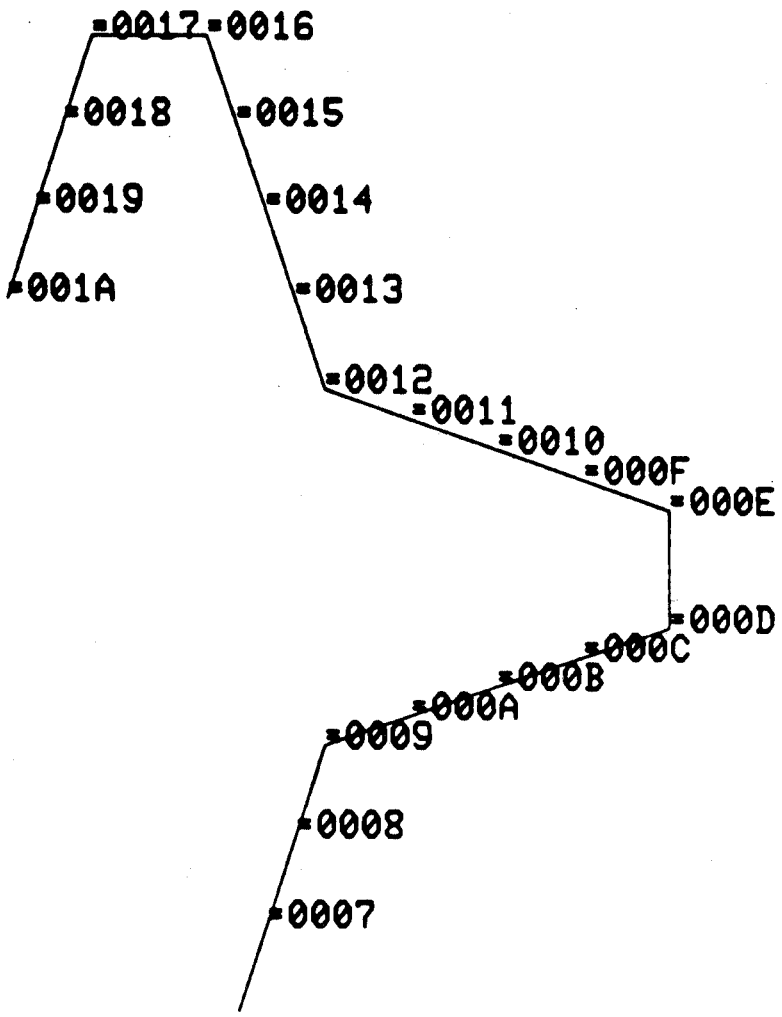
```
!LP
SGF?O
>R
NOM=FI EX1
SEGMENT 1=1
SEGMENT 2=1000
COMPTAGE?O
ENVOI ZDC?N
PAS A PAS?N
```




```
>F
!
```

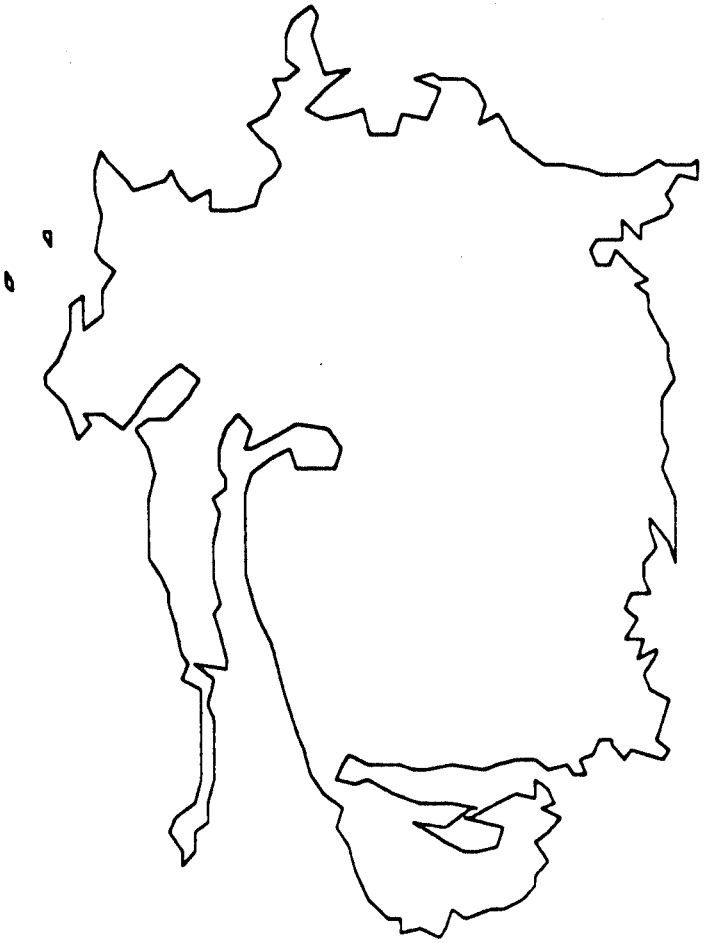
Lecture du fichier "FI EX1" avec comptage des segments.

```
!LP
SGF?O
>R
NOM=FI EX1
SEGMENT 1=7
SEGMENT 2=1A
COMPTAGE?O
ENVOI ZDC?N
PAS A PAS?N
>
```



Lecture partielle du fichier "FI EX1" avec comptage.

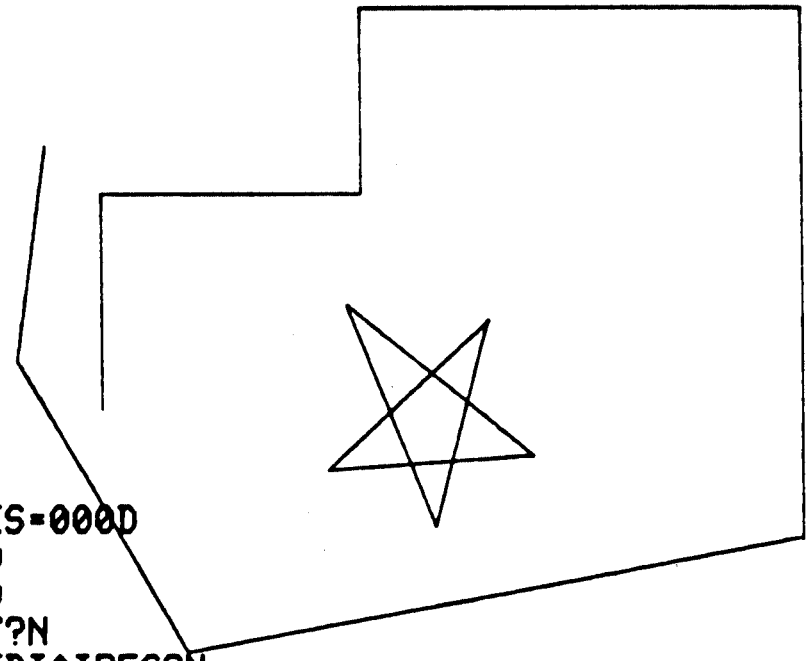
```
!LP
SGF?O
>R
NOM=FI CONTOUR
SEGMENT 1=1
SEGMENT 2=131
COMPTAGE?N
ENVOI ZDC?O
POINT-POINT?N
OAB INTERMEDIAIRES?N
PAS A PAS?N


>O
>
```



Lecture partielle du fichier "FI CONTOUR" créé par GF et EZ (voir pages GF 5 et EZ 7).

Ce tracé est envoyé en ZDC pour être récupéré par DG (voir page DG 4)

```
!LP
SGF?O
>V
VISU?O
VIDEO?N
>G
ZDC?N
VIDEO?N
CURSEUR?O
```



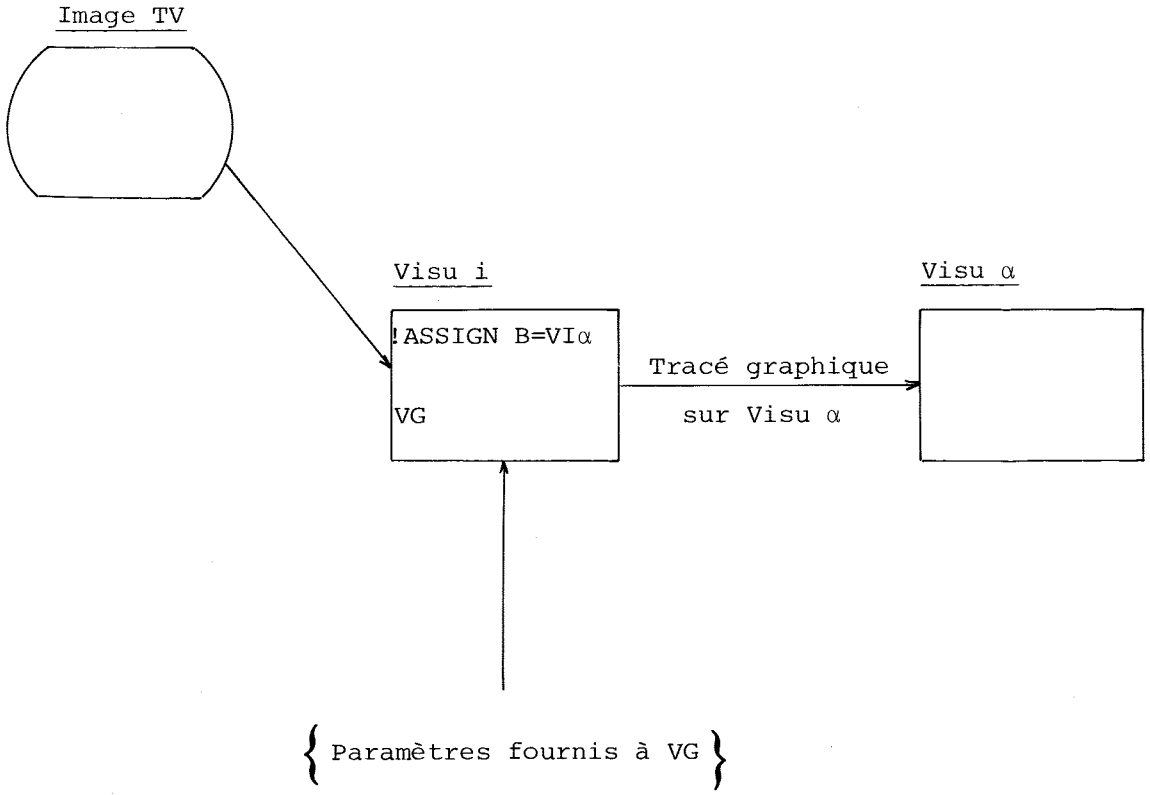
```
>S
#SEG GENERES=000D
#SEGMENTS=D
ENVOI ZDC?O
POINT-POINT?N
OAB INTERMEDIAIRES?N
>O
>F
-
```

Tracé effectué au curseur d'une console de visualisation sous LP. Le fichier temporaire obtenu est envoyé en ZDC sans changement du nombre de segments. Noter l'utilisation de la commande Ø permettant de clore le processus d'échange de segments en ZDC.

PROCESSEUR VG (VIDÉO-GRAPHIQUE)

Fonction :

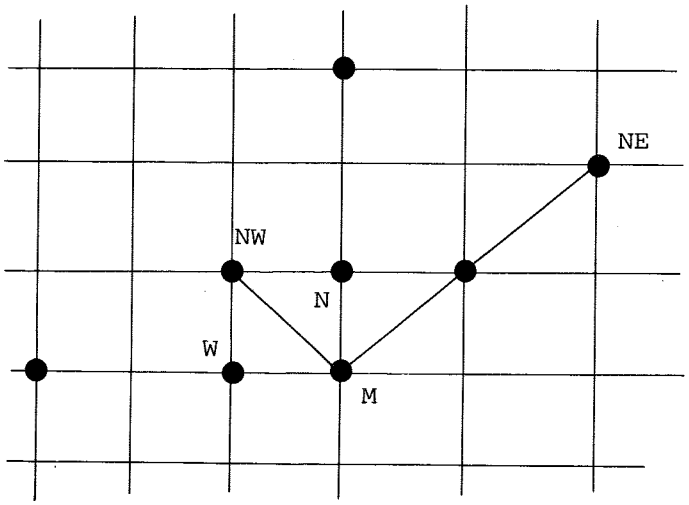
Ce processeur permet de tracer un dessin (en graphique) élaboré à partir d'une image vidéo, et fonction des paramètres fournis à VG.

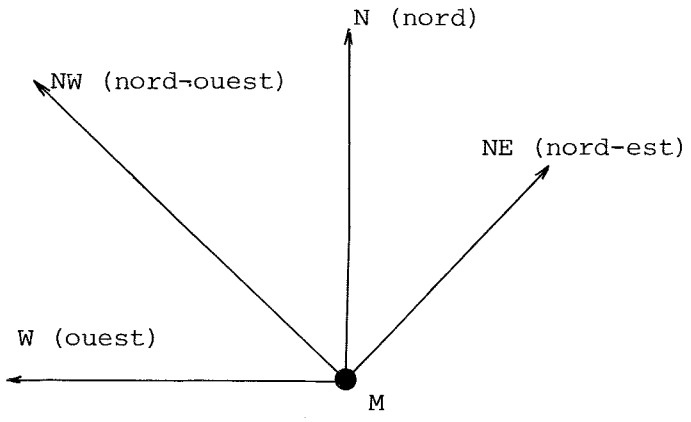


Nota :

En l'absence d'assignation par l'utilisateur de l'unité logique 'B, celle-ci sera assignée par VG à l'organe de sortie, c'est-à-dire que le tracé sortira sur la visu utilisée pour le dialogue.

Analyse d'une image par VG





Directions de recherche des voisins.

(rappel : une image numérique fait 256 x 256 points).

L'image est parcourue point par point de droite à gauche et de bas en haut. Pour chaque direction (NE, N, NW, W) successivement. Pour chaque direction de recherche :

. si M a déjà été exploité comme appartenant au voisinage d'un point précédemment traité, VG ne fait rien.

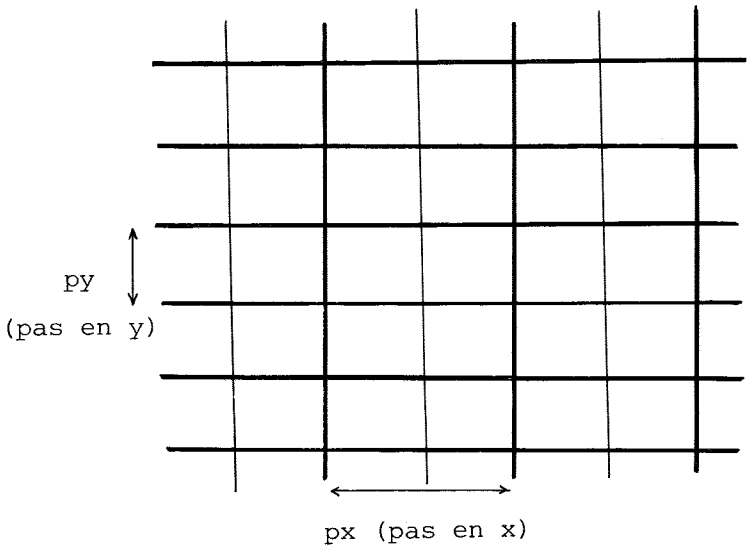
. si M n'a pas été exploité, VG examine son voisinage dans la direction courante et trace éventuellement un trait le plus long possible en fonction du voisinage dans cette direction. (cf figure ci-dessus).

Cet algorithme s'applique sachant que :

. on entend par parcours de l'image, le parcours au sens des "pas" en lignes et colonnes définis par la réponse donnée à la question "PAS L/C =" .

. on entend par voisinage de M dans une direction donnée, son voisinage au sens des intervalles (en X et Y) de recherche des voisins, donnés en réponse à la question : " DX / DY = "

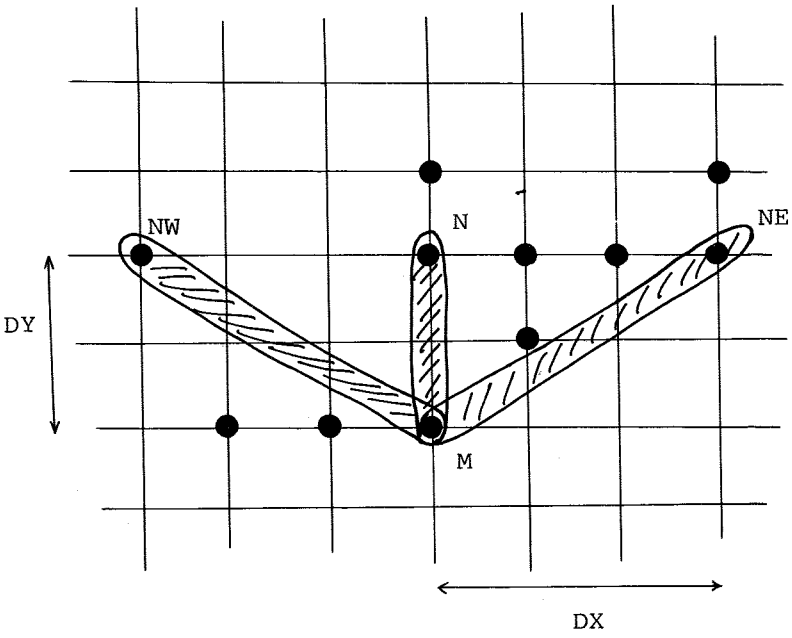
Ceci est illustré dans les figures suivantes .



La réponse :

PAS L/C = 0201

définira le quadrillage ci-contre de parcours de l'image.



La réponse :

DX / DY = 0302
définira le quadrillage ci-contre de recherche des voisins.

Utilisation de VG

Le dialogue se déroule de la manière suivante :

! VG

NØM > BØTTICELLI Ø 5 D^C nom de l'image

> G (ou D)

REDUC = n → n = chiffre de 0 à 9. Entraînera une réduction des coordonnées de coefficient 2^n.

- n=0 pas de réduction
- n=1 réduction dans un rapport de $2^1=2$
- n=2 réduction dans un rapport de $2^2=4$
- etc...

TRANS ?
- Ø → transformation demandée. Voir ci-après au paragraphe Transformation les paramètres de transformation.
- N → pas de transformation.

Cette question est posée une première fois :

- si on répond Non, VG passe à la suite du dialogue.
- si on répond Oui, VG demande les paramètres de la transformation (voir ce paragraphe), puis repose la question "TRANS ?", on peut à ce moment-là soit répondre Oui pour rentrer d'autres paramètres de transformation qui remplaceront les précédents. Ainsi de suite jusqu'à ce qu'on ait répondu N à la question "TRANS ?" (cf. exemple).

FAST?
- Ø → mode "fast" : dans ce mode, VG fait un parcours de l'image sans exploitation de voisinage : c'est-à-dire qu'il se contente de tracer un point sur la visu si le point courant de l'image est à 1.
- N → mode lent : il y aura exploration du voisinage.

En fonction de la réponse à la question "FAST ? " la suite du dialogue se déroule de deux façons :

■ *Mode FAST : suite du dialogue*

4014 ?
- Ø → effets spéciaux 4014
- N

PAS L/C =
- px py → définition du quadrillage de parcours de l'image. Réponse, facultative, sur 4 chiffres hexadécimaux (implitement : 0101).
- D^C

DX/DY = { dx dy | D^C } → définition du quadrillage de recherche des voisins. Réponse facultative, sur 4 chiffres hexa (implitement : 0101).

STEPS ? { Ø → procéder par étapes (arrêt tous les n segments) | N → pas d'étapes (mode habituel) }

ERASE ? { Ø | N } → effacement écran avant traçage du dessin.

Ici, VG effectue le tracé, puis propose d'effectuer un hard-copy.

CØPY ? { Ø → hard-copy automatique, VG fait la copie et propose une nouvelle copie. | N → Non : fin de travail. }

!

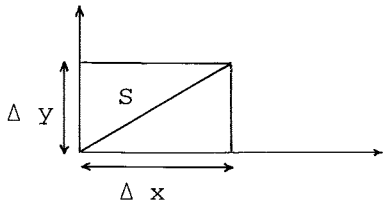
■ *Mode "lent" (FAST? N) : suite du dialogue*

ISØLES? { Ø → tenir compte des points isolés | N → ignorer les point isolés. }

SEUIL = { D^C | ΔX ΔY } → spécification éventuelle d'un seuil, seuil en X, seuil en Y ; même signification que dans le processeur ST, c'est-à-dire :

SX = { D^C | ΔX }

SY = { D^C | ΔY }



Pour chaque segment S VG calcule ses normes Δ, ΔX et ΔY avec

Δ= | ΔX | + | ΔY |

Ensuite si Δ < SEUIL ou

ΔX < SX ou

ΔY < SY ,alors S n'est pas tracé.

SEUIL, SX et SY sont facultatifs. S'ils sont servis, ils sont exprimés par 1 à 3 chiffres hexadécimaux (cf. exemples donnés dans processeur ST).

PARA ? { Ø / N → paraglyphes (effet photo connu)

4014? { Ø / N → effets spéciaux 4014

W? { Ø / N → Directions W (ouest)
NW? { Ø / N → NW (Nord-Ouest)
E? { Ø / N → E (Est)
NE? { Ø / N → NE (Nord est)

on choisit celles qu'on veut parmi les quatre directions de recherche de voisins et de tracé des segments.

PAS L/C = { px py / D^C → définition du quadrillage de parcours de l'image. Réponse sur 4 chiffres hexadécimaux (Implicitement : 0101)

DX/DY = { dx dy / D^C → définition du quadrillage de recherche des voisins. Réponse sur 4 chiffres héxadécimaux (implicitement : 0101).

STEPS ? { Ø → procéder par étapes
N → pas d'étapes

ERASE ? { Ø → effacement écran avant traçage du dessin.
N → pas d'effacement.

Ici, VG effectue le tracé ; puis il propose d'effectuer un hard-copy.

CØPY ? { Ø → hard-copy automatique ; VG fait la copie et propose une nouvelle copie
N → non, pas de copie ; fin de travail.

!

Remarque : Image négative/positive

On doit se souvenir que :

. ce qui apparaît en blanc sur l'écran TV
apparaîtra en vert sur l'écran de la visu
apparaitra en noir sur le hard-copy.

. ce qui apparaît en noir sur l'écran TV
apparaîtra en noir sur l'écran de la visu
apparaîtra en blanc sur le hard-copy

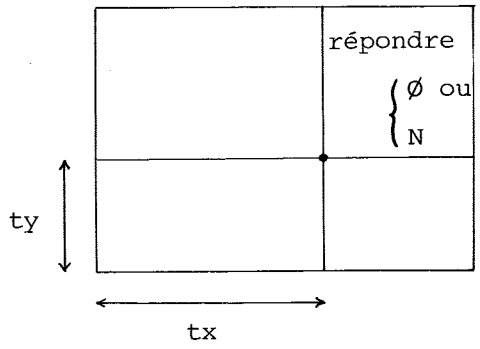
Autrement dit, l'image de TV noir et blanc sera inversée sur le hard-copy. S'en souvenir si l'on veut par exemple tirer un visage rentré par caméra : inverser l'image avant de la traiter par VG !

■ *Transformation demandée*

On a répondu Oui à la question "TRANS ?"

TRANS ? Ø

Ici, VG (mais ceci est valble pour les autres processeurs faisant référence à ce paragraphe), fait apparaître le réticule sur l'écran de la visu assinée à l'unité logique B, c'est-à-dire sur la visu où se déroule la conversation avec VG si l'utilisateur n'a pas explicitement assigné B, soit sur la visu à laquelle l'utilisateur a lui-même assigné B (il s'agit de la visu où se fera le tracé).



Si l'on répond Ø (Oui), le point désigné par le réticule définit la translation en X et la translation en Y sur la visu où sera fait le tracé.

Si l'on répond N (Non), alors VG demandera la translation en X et la translation en Y de la façon suivante :

TX = { tx ; D^C ⟶ Translation en X ; implicitement, tx = 0.

Cette translation, exprimée en hexadécimal (voir les exemples) est un nombre de points sur la visu dont l'écran fait, rappelons-le 1024 x 1024 points ('400x'400 en hexa), dont 768 visualisés en Y.

TY = { ty / D^C → Translation en Y ; même chose que pour TX.

On note qu'il y a donc 2 manières de définir TX et TY

- soit en positionnant le curseur et et répondant "Ø".
- soit en répondant "N" auquel cas on fournit explicitement les valeurs de TX et TY (voir les exemples ci-après).

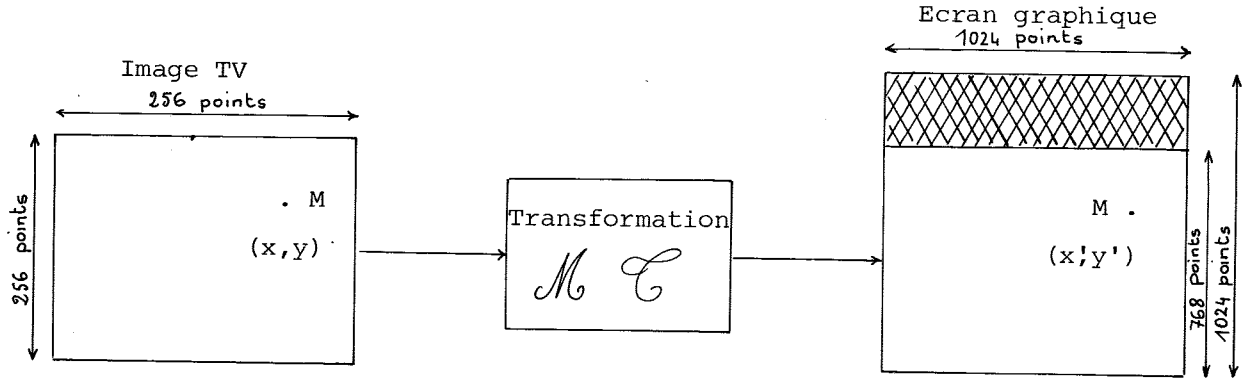
VG demande ensuite de rentrer une matrice de transformation :

M11 = $a_{11}D^C$ / $b_{11}D^C$
M12 = $a_{12}D^C$ / $b_{12}D^C$
M21 = $a_{21}D^C$ / $b_{21}D^C$
M22 = $a_{22}D^C$ / $b_{22}D^C$

Matrice de transformation ; chaque coefficient s'exprime ainsi : un numérateur servi en hexadécimal sur 1 à 4 chiffres ; puis VG affiche la barre de fraction ; on rentre alors le dénominateur (non nul de préférence). Un nombre non servi est supposé nul par VG.

PX ? N
PY ? N → répondre Non (Utilisation délicate).

La translation TX et TY et la matrice fournis à VG sont utilisés par celui-ci de la manière suivante :



$$\mathcal{M} = \begin{pmatrix} M_{11} & M12 \\ M_{21} & M_{22} \end{pmatrix} \qquad \mathcal{C} = \begin{pmatrix} tx \\ ty \end{pmatrix}$$

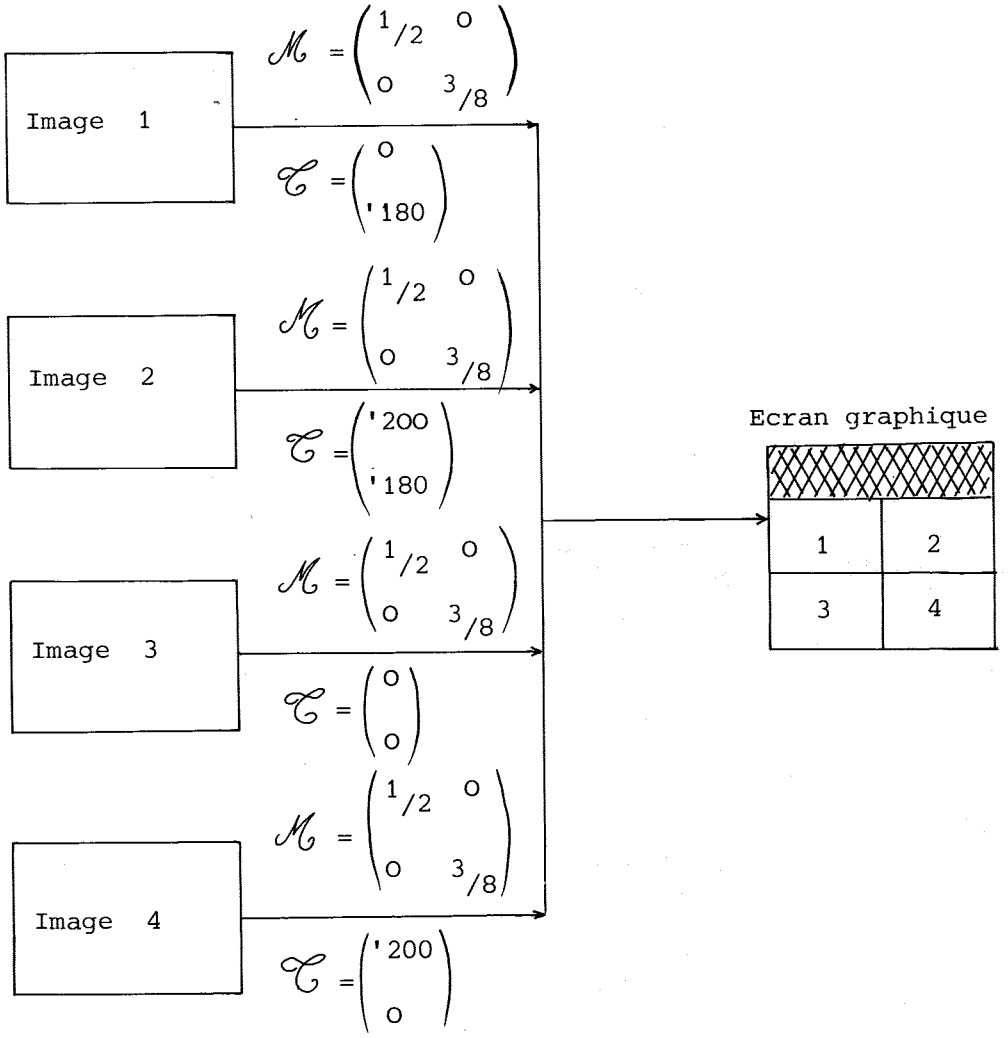
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{12} \end{pmatrix} \begin{pmatrix} 2x \\ 2y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

Le facteur multiplicatif 2 est dû au fait qu'on a 2 fois plus de points en X et 2 fois plus en Y sur l'écran graphique que sur l'image TV. On note que 768 points seulement sont visualisés en Y sur l'écran graphique. Pour obtenir une simple reproduction (en respectant les proportions) de l'image TV sur l'écran de visualisation, il suffit de définir $\mathcal{M}$ et $\mathcal{C}$ comme suit :

$$\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & 3/4 \end{pmatrix} \qquad \mathcal{C} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

C'est ce que l'on fait par exemple pour tirer un portrait rentré par caméra, à partir de l'image numérique.

Autre exemple : pour reproduire 4 images sur un même écran graphique, il suffit d'utiliser VG 4 fois de suite, avec EFFACEMENT = Øui la première fois, puis EFFACEMENT = N les fois suivantes, en donnant les matrices et translations suivantes :



①

②

Tracés obtenus par VG à partir de l'image TV : BOTTICELLI 5, avec les paramètres suivants :

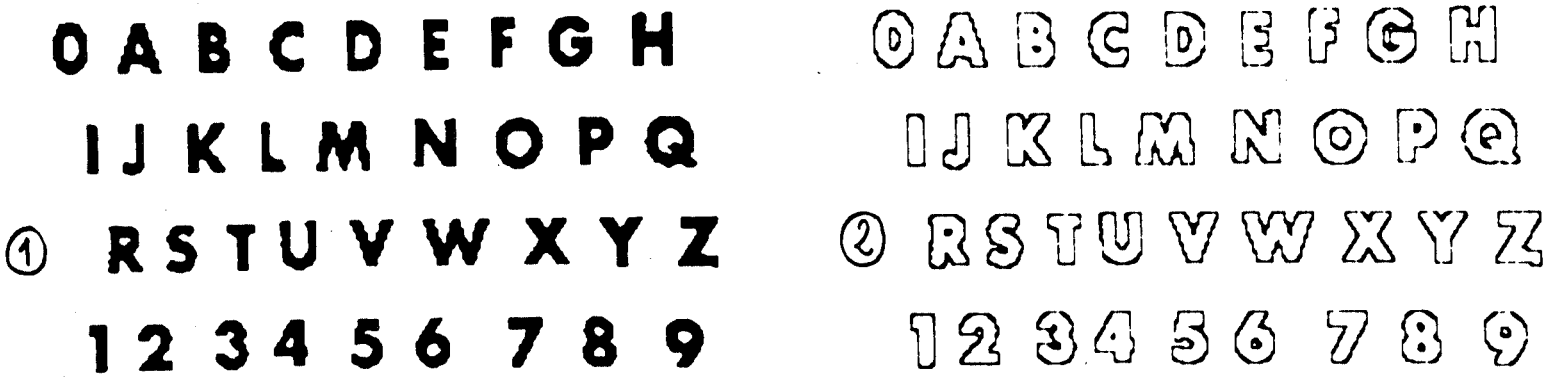
① REDUC = 1 ; TX = 0 ; TY = 180 ;
$M = \begin{pmatrix} 1 & 0 \\ 0 & 3/4 \end{pmatrix}$; Direction de tracé : Nord-Est ; PAS L/C = 0202 ; DX/DY = 0202.

② comme 1 avec direction de tracé Ouest.

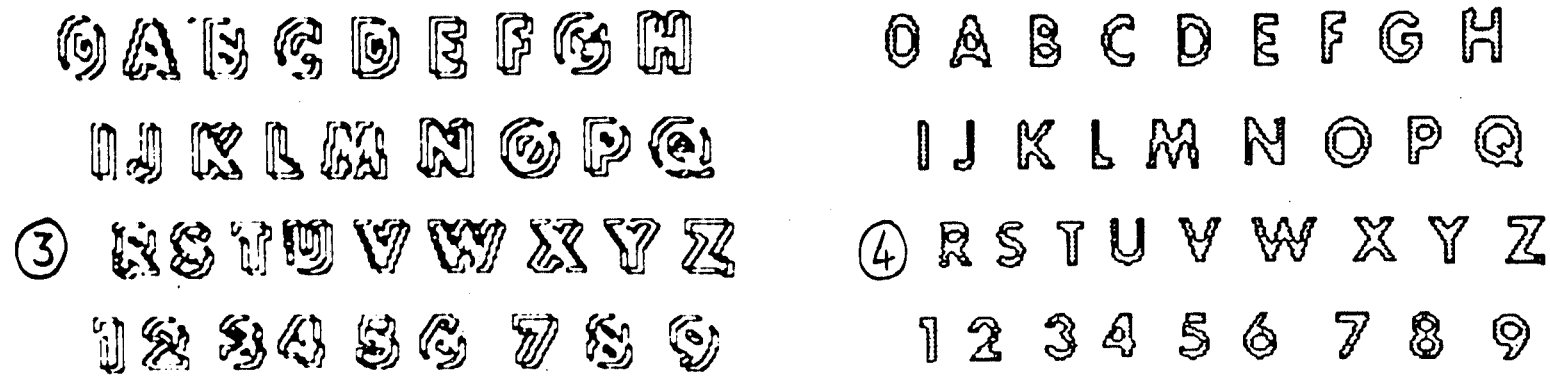
③

REDUC = 2 ; TX = 200 ; TY = 0 ;
$M = \begin{pmatrix} 1/2 & 1 \\ 1 & 3/8 \end{pmatrix}$; directions de tracé Nord et Ouest ; PAS L/C = 0202 ; DX/DY = 0202.

Polices de caractères tracées sous VG. La police 1 a été rentrée par caméra et l'image TV reproduite telle quelle. La police 2 est faite à partir de la



même image que 1 mais on a demandé à VG l'effet paraglyphes. Les polices 2 et 3 sont la reproduction de deux images obtenues sous le processeur TV à partir de l'image initiale entrée par caméra (police 1), en utilisant des

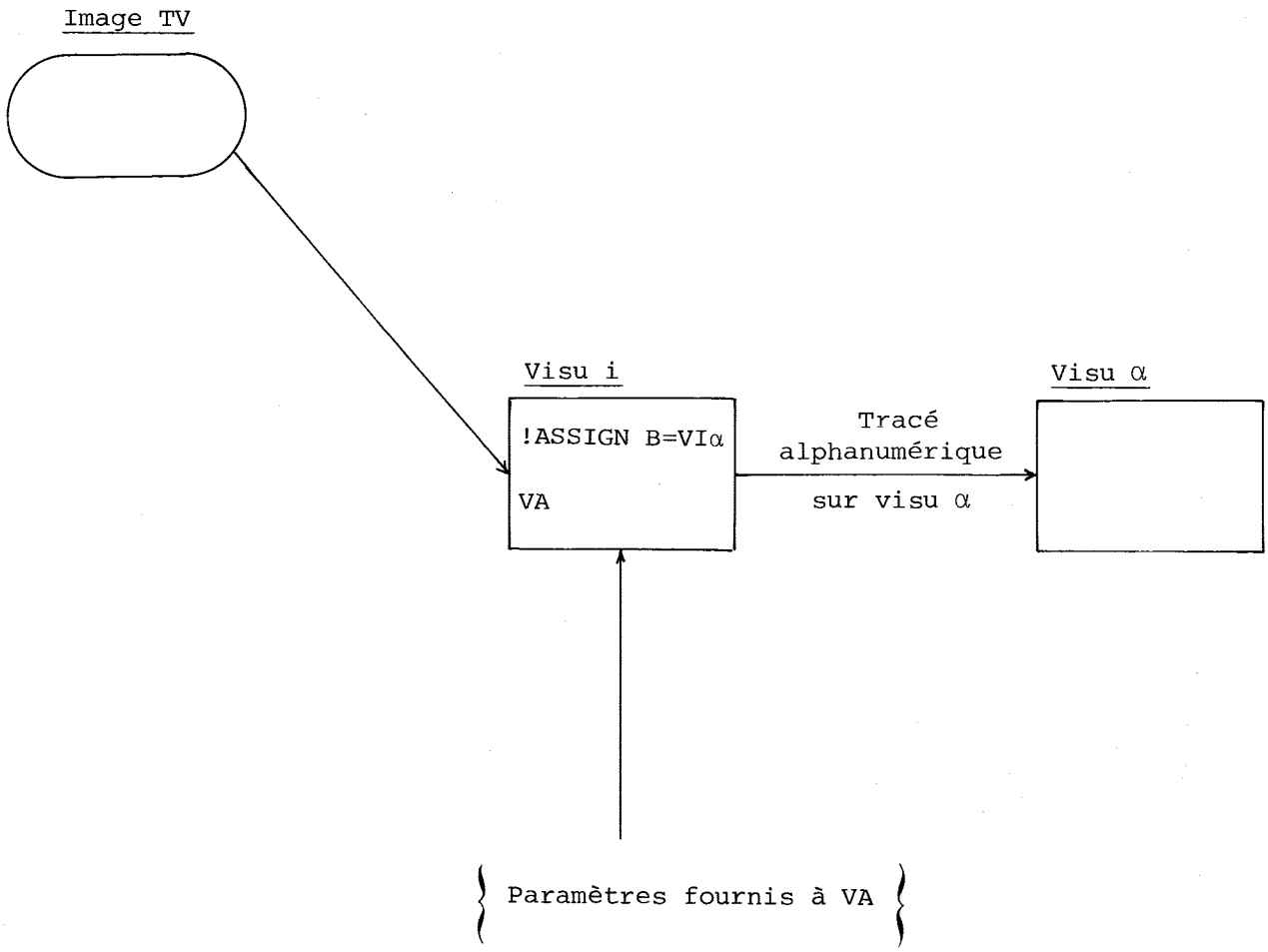


opérations logiques avec décalage. Remarquer que 3 = (2 . EOR. 2 avec décalage).

PROCESSEUR VA (VIDEO-ALPHANUMERIQUE)

Fonction :

Ce processeur permet de sortir un "dessin" fait de caractères alphanumériques, élaboré à partir d'une image vidéo, en fonction des paramètres fournis à VA. Dans son schéma de fonctionnement et dans son utilisation, VA est assez proche de VG.



Nota :

En l'absence d'assignation par l'utilisateur de l'unité logique 'B, celle-ci sera assignée par VA à l'organe de sortie, c'est-à-dire que le tracé sortira sur la visu utilisée pour le dialogue.

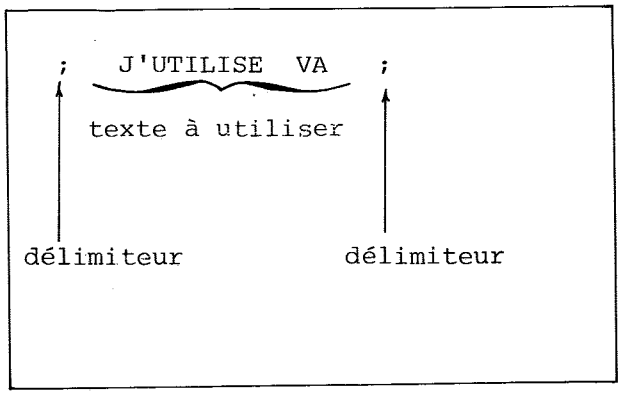
Analyse d'une image par VA

Elle se fait de la même manière que par VG et les réponses fournies aux questions "PAS L/C=" et "DX/DY=" ont la même signification.

VA diffère de VG dans la mesure où il ne s'agit pas d'un tracé graphique mais d'un tracé alphanumérique. Ce tracé alphanumérique peut se faire suivant 3 modes différents, au choix de l'utilisateur :

■ *Mode TEXTE :*

VA utilisera un texte placé par l'utilisateur en Item 1 ; celui-ci aura donc placé en Item 1 une chaîne de caractères dont le premier caractère servira de délimiteur ; par exemple ceci :



Le caractère servant de délimiteur est au choix de l'utilisateur ; ici on a pris le caractère : " ; "

■ *Mode CONFIGURATION*

VA étudiera la configuration des points de l'image à partir de critères internes et à partir des paramètres "PAS L/C" et "DX/DY" fournis par l'utilisateur ; il en déduira les caractères alphanumériques à utiliser, en fonction de leurs densités respectives (voir les exemples).

■ *Mode COMPTAGE*

VA calculera en chaque point la valeur V :

$$V = \left\{ \begin{array}{l} 1 \quad \text{si point blanc} \\ 0 \quad \text{si point noir} \end{array} \right\} + \left\{ \begin{array}{l} \text{nombre de voisins à 1} \\ (\text{de } 0 \text{ à } 8) \end{array} \right\}$$

On a donc $0 \leqslant V \leqslant 9$

A chaque valeur de V, VA associe un caractère C :

V =	0	1	2	3	4	5	6	7	8	9
C =	∅	.	:	_	+	=	H	*	$	@

Dans ce mode, VA donne la possibilité à l'utilisateur de changer les caractères associés aux valeurs V (voir les exemples).

Utilisation de VA

! VA

NØM > T-MACARØN D^C → nom de l'image

REDUC = n → n est un chiffre hexa de 0 à 9. Entraînera une réduction de l'image de coefficient 2^n :

. n=0 pas de réduction

. n=1 réduction dans un rapport de 2

. n=2 réduction dans un rapport de 4

etc...

TRANS ? { Ø → transformation demandée. Voir processeur VG le paragraphe développant cette option.

N → pas de transformation demandée.

VA propose une nouvelle transformation tant qu'on n'a pas répondu N à la question TRANS ? (voir utilisation de VG).

PAS L/C = { Px Py / D^C ou Return → définition du quadrillage de parcours de l'image. Réponse sur 4 chiffres hexadécimaux (implicitement 0101)

DX/DY = { dx dy / D^C ou Return → définition du quadrillage de recherche des voisins. Réponse sur 4 chiffres hexadécimaux (implitement 0101)

MØDE (0 = TEXTE, 1=CØNFIGURATION, 2=CØMPTAGE) = { 0 / 1 / 2

→ mode choisi ; on rappelle que si l'on choisit le mode TEXTE, on devra avoir placé en Item 1 le texte à utiliser. D'autre part, si l'on choisit le mode CØMPTAGE, VA propose de changer les caractères à utiliser, comme le montre l'un des exemples d'utilisation de VA.

Ici, VA sort le tracé alphanumérique associé à l'image analysée ; puis il propose d'effectuer un hard-copy :

CØPY ? { Ø → hard-copy automatique ; VA fait la copie et propose une nouvelle copie

N → non, pas de copie, fin de travail.

! → fin de travail

```
!VA
NOM>T-MACARON
REDUC=0
TRANS?O

TX=
TY=
M11=1/1
M12=0/1
M21=0/1
M22=3/4
PX?N
PY?N
TRANS?N
PAS L/C=0606
DX/DY=0404
MODE (0=TEXTE, 1=CONFIGURATION, 2=COMPTAGE)=0
ERASE?O
```

Utilisation de VA en mode "texte". On a préalablement placé en Item 1 ceci :

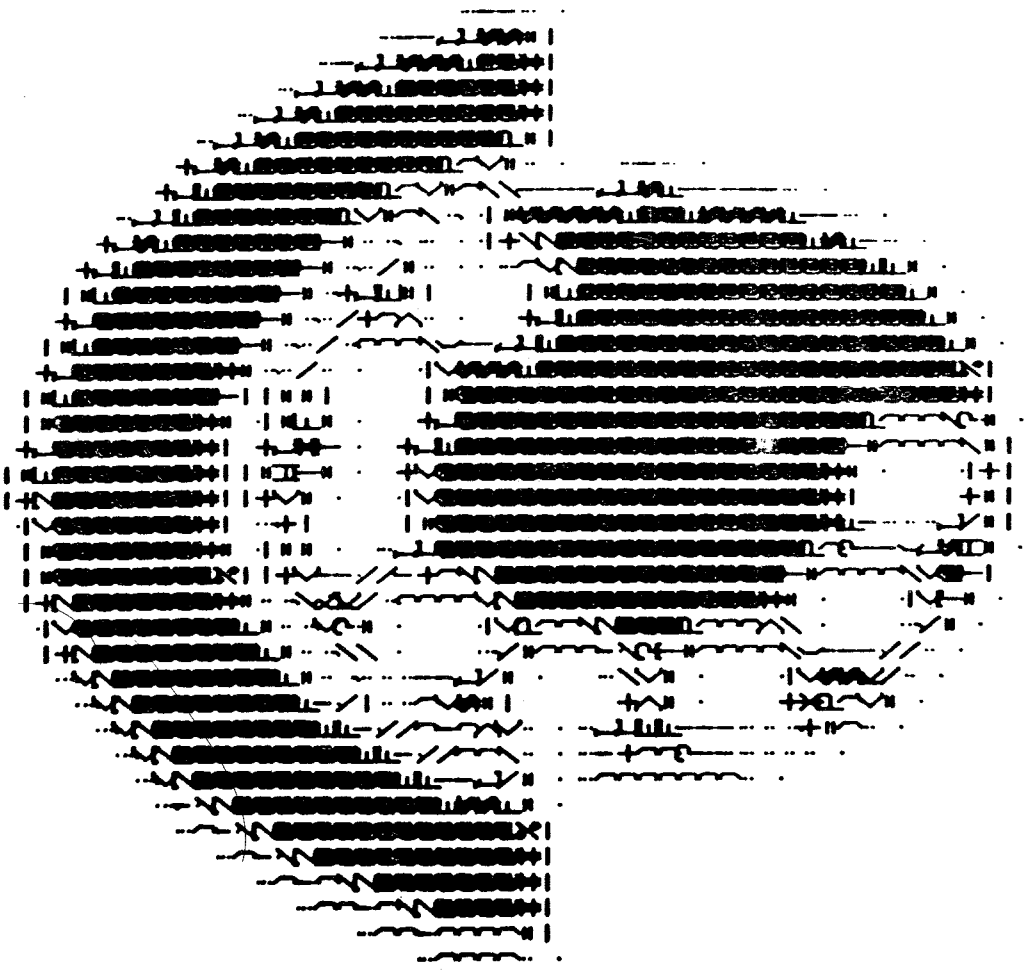
;MACARON EXEMPLE;

VA 5

COPY?O

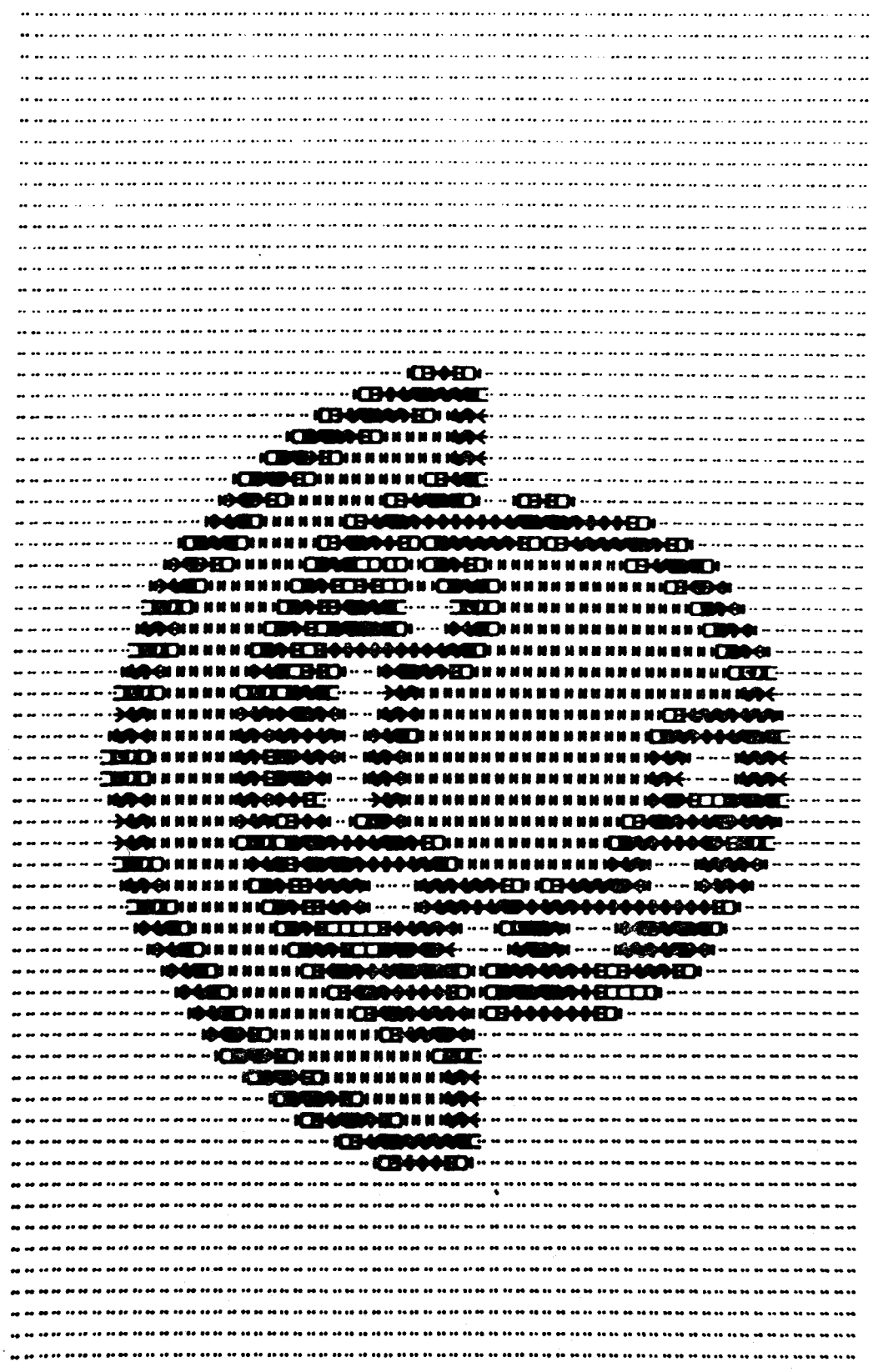
```
!VA
NOM>T-MACARON
REDUC=0
TRANS?O

TX=
TY=
M11=1/1
M12=0/1
M21=0/1
M22=3/4
PX?N
PY?N
TRANS?N
PAS L/C=0404
DX/DY=0404
MODE (0=TEXTE, 1=CONFIGURATION, 2=COMPTAGE)=1
ERASE?O
```

Utilisation de VA en mode "configuration".



```
!VA
NOM>T-MACARON
REDUC=0
TRANS?O

TX=
TY=
M11=1/1
M12=0/1
M21=0/1
M22=3/5
PX?N
PY?N
TRANS?N
PAS L/C=0404
DX/DY=0404
MODE (0=TEXTE, 1=CONFIGURATION, 2=COMPTAGE)=2
CHGT?O
$VOISINS=
0123456789
  .:-+=H*$@
 :=H*$@$*H=
ERASE?O
```

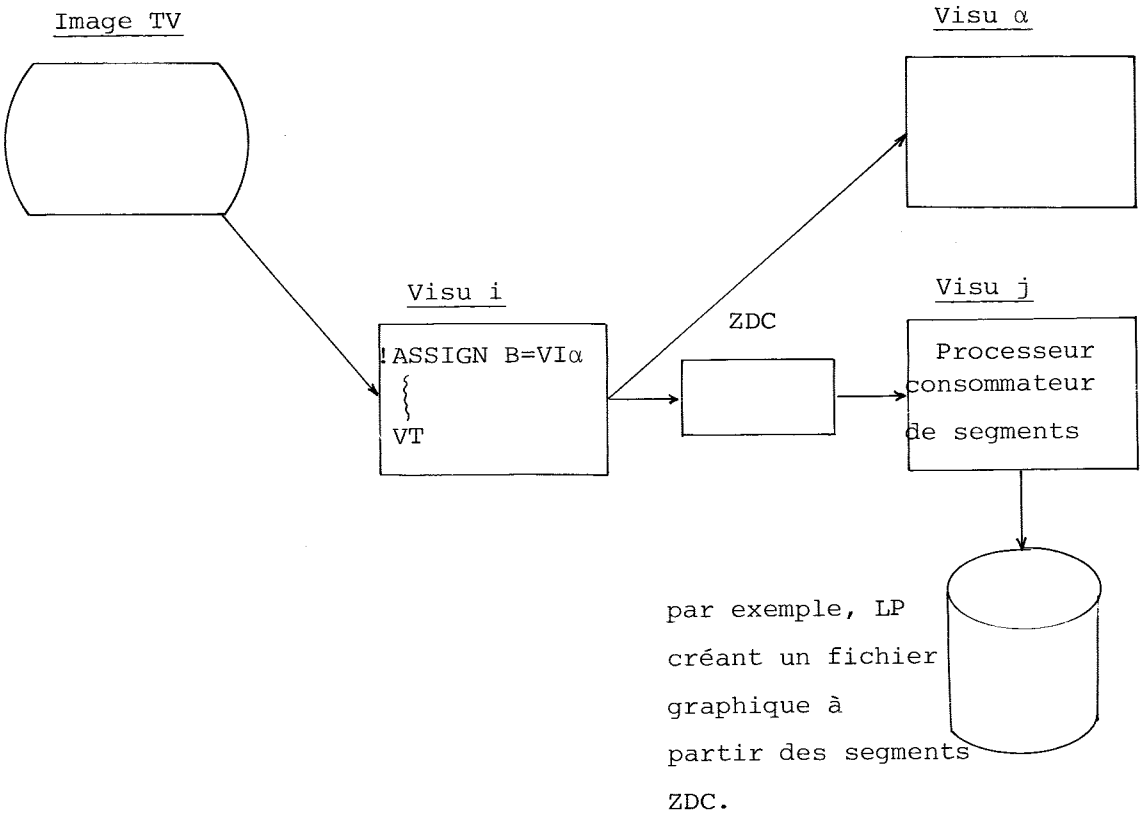
Utilisation de VA en mode comptage, avec changement du jeu de caractères à utiliser par VA.



PROCESSEUR VT

Fonction

Le processeur VT permet d'effectuer un tracé graphique à partir d'une image (comme VG) avec envoi en ZDC des coordonnées de segments.



Utilisation de VT

Le dialogue avec VT se déroule de la même manière qu'avec le processeur VG. VT demande en plus si l'on veut envoyer en ZDC des segments nuls à l'origine entre chaque segment (question "ØAB INTERMEDIAIRES ?" : répondre oui "Ø" ou non "N").

Remarques

1 - ne pas oublier de réinitialiser à zéro la ZDC avant d'utiliser VT.

2 - ne pas oublier d'assigner l'unité logique B à la visu sur laquelle on veut faire le tracé.

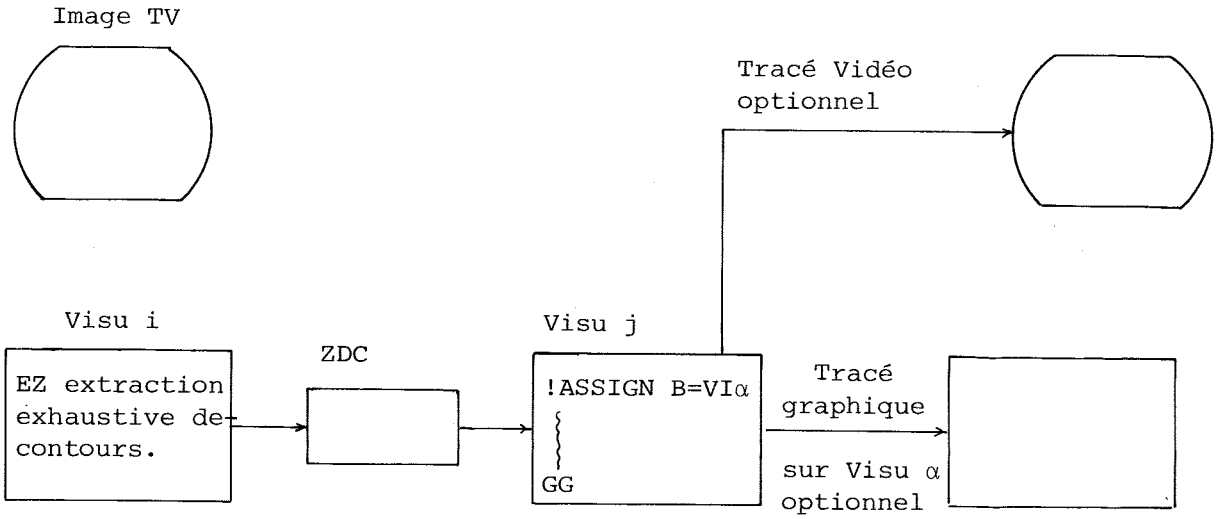
PRØCESSEUR GG (GRAPHIQUE → GRAPHIQUE)

Fonction

Le processeur GG est consommateur de segments ZDC.
Les segments reçus

- font l'objet d'une transformation si l'utilisateur le spécifie.
- puis d'un filtrage sur leur norme (seuils en X et y), si l'utilisateur le demande ;
- et enfin sont exploités pour donner un tracé graphique et / ou vidéo.

Exemple d'utilisation :



Nota :
En l'absence d'assignation par l'utilisateur de l'unité logique 'B, celle-ci sera assignée par GG à l'organe de sortie, et donc le tracé sortira sur la visu utilisée pour le dialogue.

Utilisation de GG

Le dialogue avec GG se déroule comme suit :

! GG

GRAPHIQUE ? { Ø → Oui, tracé graphique demandé.
{ N → Non

VIDEØ ?
- Ø → Oui, tracé vidéo demandé ; dans ce cas GG propose :
  - RAZ?
    - Ø → Oui, remise à zéro image résidente
    - N → non, pas de remise à zéro.
- N → Non.

REDUC = n → n est un chiffre hexadécimal de 0 à 9 . Entraînera une réduction des coordonnées arrivant en ZDC dans un rapport de 2^n :

. n=0 pas de réduction

. n=1 réduction dans un rapport 2

. n=2 réduction dans un rapport 4

TRANS ?
- Ø → transformation demandée. Voir processeur VG le paragraphe développant cette option.
- N → pas de transformation demandée.

GG propose une nouvelle transformation tant qu'on n'a pas répondu Non à la question "TRANS ?" (voir utilisation du processeur VG).

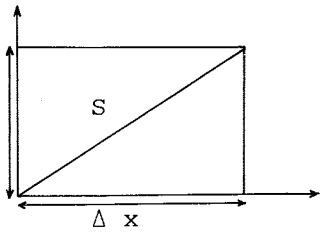
PRØGRAMME DE TRANS ?
- Ø → même signification que dans le processeur ST c'est-à-dire qu'il s'agit d'un programme en assembleur placé dans Item 2.
- N → pas de programme de transformation.

SEUIL =
- D^C ou Return
- Δ

SX =
- D^C ou Return
- ΔX

SY =
- D^C ou Return
- ΔY

spécification éventuelle d'un seuil, seuil en X, seuil en Y ; même signification que dans le processeur ST, c'est-à-dire :



Pour chaque segment S, GG calcule ses normes Δ, ΔX et ΔY avec

$\Delta = |\Delta X| + |\Delta Y|$

Ensuite si Δ < SEUIL ou

ΔX < SX ou

ΔY < SY , alors S n'est pas tracé.

SEUIL, SX et SY sont facultatifs. Sils sont servis, ils sont exprimés sous forme de 3 chiffres héxadécimaux (voir exemples dans le processeur ST).

ERASE ? { Ø ——→ oui, effacement écran avant tracé
N ——→ non, pas d'effacement écran.

Ici, GG effectue le tracé à partir des segments de ZDC puis il propose de faire un hard-copy.

CØPY ? { Ø ——→ hard-copy automatique ; GG fait la copie puis propose une nouvelle copie
N ——→ non, pas de copie ; fin de travail.

! ——→ fin de travail.

PROCESSEUR GF (Graphique Fichier)

Fonction

Le processeur GF est consommateur de segments ZDC.
Il est utilisable de deux façons.

1 - commande G:

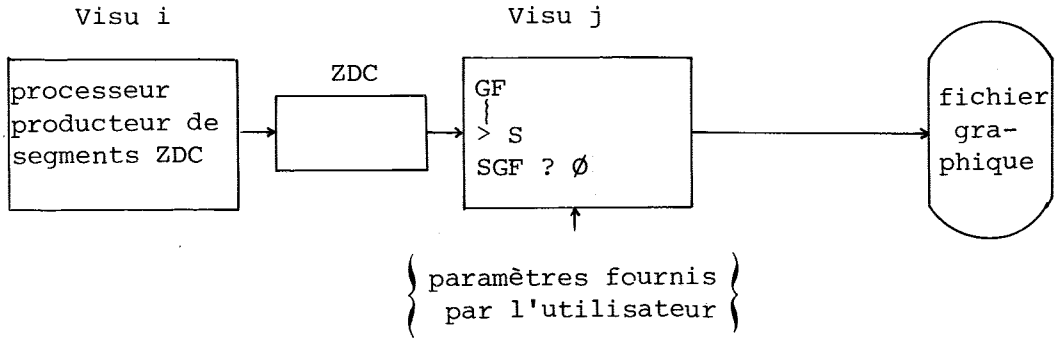
GF trace le dessin dont il reçoit les segments en ZDC, et en fin de tracé, il trace le centre de gravité G du dessin et ses axes d'inertie.

2 - commande S :

GF reçoit les segments en ZDC, et, en fonction des paramètres fournis par l'utilisateur :

- il applique éventuellement une transformation (identique à la transformation qu'on peut rentrer sous VG)
- il applique éventuellement un programme de transformation (programme écrit en assembleur, assemblé par GC, et placé par l'utilisateur en Item 2).
- il applique éventuellement un "filtre" sur la norme des segments, (identique au filtre qu'on peut utiliser sous ST, VG etc...)
- il arrondit les coordonnées de segments pour les adapter au quadrillage défini par l'utilisateur en réponse aux questions "K", "X" et "Y" (K, X et Y définissant un quadrillage au sens du langage graphique ; voir GR)
- enfin, et optionnellement, il accompagne le tracé du dessin sur l'écran de la visu, d'une sauvegarde de ce dessin dans un fichier graphique qui sera ensuite récupérable sous le processeur LP.

Exemple d'utilisation



Utilisation de GF

1 - Commande G : centre de gravité et axes d'inertie :

!GF

> G

Ici, GF consomme les segments ZDC ; il trace le dessin sur l'écran ;

à la fin du tracé, il indique l'emplacement du centre de gravité par la lettre G et trace les axes d'inertie.

> F

!

2 - Commande S : transformation et sauvegarde SGF :

! GF

> S

SGF ? { Ø → oui ; sauvegarde sur fichier ; GF demande le nom du fichier :
FICHIER = nom du fichier D^C
N → non, pas de sauvegarde sur fichier.

REDUC= n

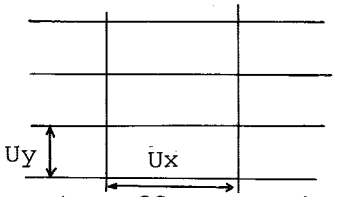
TRANS ? { Ø → oui, transformation } Voir VG
N

PROGRAMME DE TRANS ? { Ø → oui, l'utilisateur a placé un programme de transformation en Item 2 (programme assembleur).
N

SEUIL = Δ → spécification d'un "filtre" sur norme, norme en
SX = Δ x x, norme en y des segments.
SY = Δ y Voir ST, VG etc...

K= k → spécification d'échelles en X et Y. Les coordon
X= x nées de segments ZDC seront arrondies pour s'a-
Y= y juster au quadrillage ainsi défini.

$k \in \{1,2,..,9,A,..,Z\}$
$x \in \{1,2,..,9,A,..,Z\}$
$y \in \{1,2,..,9,A,..,Z\}$



Une case du quadrillage fait Ux sur Uy points avec:

Ux = k x α

Uy = k x y

ERASE ? { Ø → oui ; effacement écran avant traçage
N → non, pas d'effacement écran.

Ici, GF effectue son tracé à partir des segments ZDC ainsi que la sauvegarde éventuelle du dessin sous forme de fichier graphique.

CØPY ? { Ø → oui ; déclenchement automatique du système de hard-copy (ou caméra). Cette question est re-posée jusqu'à ce qu'on ait répondu Non.
N → non

> F → fin de travail

!

Remarques sur GF

. Comme pour la plupart des processeurs graphiques, le tracé est fait sur la visu utilisée sauf si l'on a explicitement assigné l'unité logique 'B à une autre visu, auquel cas le tracé est fait sur cette dernière.

. On dispose aussi sous GF de la commande W : retour temporaire au cci (?). Par exemple pour faire une assignation. Revenir par !GØ

Exemple :

```
!GF                          → appel de GF
> W                          → retour temporaire au CCI
? ! ASSIGN B=VI6D^C          → assignation de l'unité logique B à la visu 6 sur
? ! GØ D^C                     laquelle on veut faire le tracé
> S
SGF? Ø
etc...
```

```
! GF
>S
SGF?O
FICHIER=FI CONTOUR
REDUC=0
TRANS?O

TX=
TY=
M11=1/1
M12=0/1
M21=0/1
M22=3/4
PX?N
PY?N
TRANS?N
PROGRAMME DE TRANS?N
SEUIL=
SX=
SY=
K=2
X=2
Y=2
ERASE?O
```

Création sous GF du fichier "FI CONTOUR"
Les segments sont acquis en ZDC où ils sont émis dans cet exemple par le processeur EZ (voir page EZ 5).
On leur applique une transformation pour respecter sur l'écran graphique les proportions de l'image de télévision.
Le fichier obtenu pourra par la suite être exploité par LP (voir page LP 12).
En page suivante, le tracé effectué par GF.

GF 5

PROCESSEUR ST (" STYLISATION ")

Fonction

ST est un processeur consommateur de segments en ZDC.A chaque fois qu'il reçoit un segment

- ST lui applique éventuellement la transformation spécifiée par l'utilisateur si celui-ci a répondu Oui à la question "TRANS" (voir utilisation de ST), pour obtenir le vecteur $\vec{s}_n$.

- ST trace le segment s_n en traits pleins.

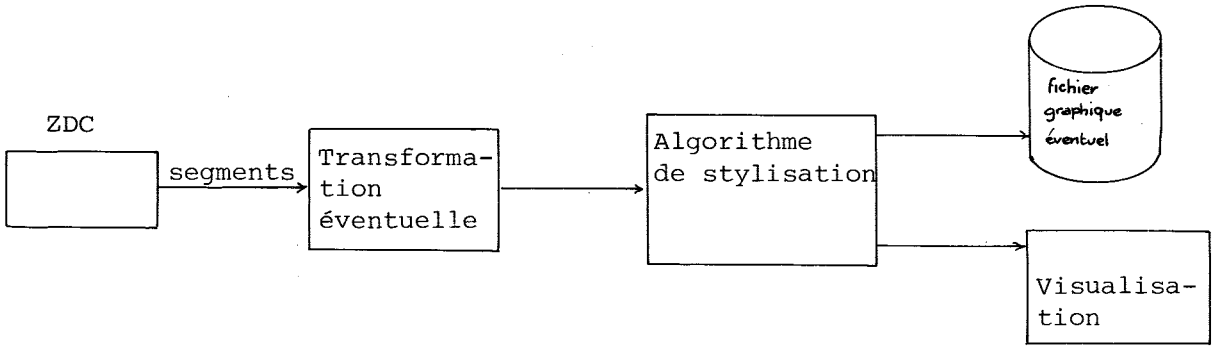
- ST calcule l'angle β du vecteur $(\vec{\Sigma} + \vec{s}_n)$ avec le vecteur $\vec{\Sigma}$; avec $\vec{\Sigma} = \vec{s}_0 + \vec{s}_1 + \ldots + \vec{s}_{n-1}$, somme des n-1 vecteurs précédents, et $\vec{s}_n$ étant le dernier vecteur reçu (voir schéma ci-après).

- ST compare la valeur absolue $|\beta|$ de l'angle β avec le paramètre α spécifié par l'utilisateur en réponse à la question "EPS ★ 256 = " (voir utilisation de ST).

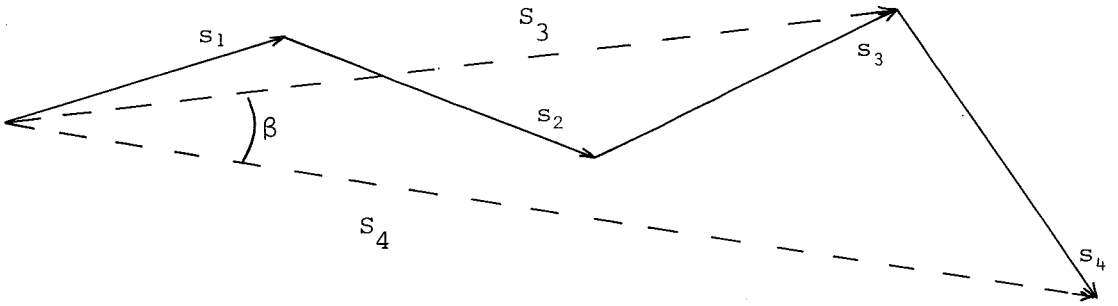
si $|\beta| \leqslant \alpha$, alors $\vec{\Sigma} := \vec{\Sigma} + \vec{s}_n$ et c'est tout.

si $|\beta| > \alpha$, alors $\vec{\Sigma}$ est tracé (en pointillés sur 4014) et éventuellement envoyé sur fichier graphique si l'utilisateur l'a demandé (réponse à la question "SGF ?"), puis $\vec{\Sigma} := \vec{0}$ et on repart avec i = 0.

On obtient ainsi, en fonction du paramètre α un dessin stylisé, qui ne contient que les grandes lignes du dessin original (voir les exemples).



ST remplacera par exemple une suite de segments s_1 s_2 s_3 par un segment S_3 comme ceci



remarquer l'angle β, formé par les vecteurs : S_3 et S_4 avec
$\vec{S_3} = \vec{s_1} + \vec{s_2} + \vec{s_3}$ et $\vec{S_4} = \vec{S_3} + \vec{s_4}$

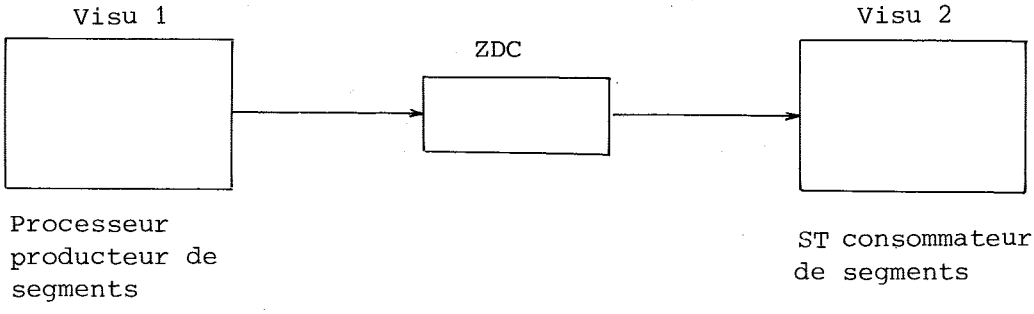
Utilisation

Avant d'utiliser ST, remettre à 0 la ZDC en utilisant SE comme ceci :

! SE
TYPE > 0 ZDC est remise à 0
!

Puis lancer, dans n'importe quel ordre

- le processeur producteur de segments sur une visu
- le processeur ST sur une autre visu.



Le dialogue avec ST se déroule de la façon suivante :
(voir aussi les exemples d'utilisation donnés ci-après)

! ST

4014 ?
- Ø → Oui, on utilise une visu 4014 : ST tracera en traits pleins le dessin non-stylisé et en pointillés le dessin stylisé.
- N → Non : ST tracera tout en traits pleins. nota : rien n'empêche si l'on utilise une 4014 de répondre Non à cette question ; en revanche il est dangereux de répondre oui alors qu'on n'est pas sur une 4014.

SGF ?
- Ø → Oui : on veut que le dessin stylisé soit aussi envoyé sur un fichier graphique que l'on pourra récupérer ensuite (par exemple en utilisant le processeur LP). Ceci permet d'isoler le dessin stylisé du dessin origine. ST demande alors un nom de fichier, le lui donner.
- N → Non : pas de fichier graphique en sortie.

REDUC = 0 → servir un chiffre de 0 à 9. Si l'on sert Reduc = 1, les coordonnées récupérées en ZDC seront divisées par $2^1 = 2$; si on sert 2, de $2^2 = 4$ ..etc...
Si l'on sert 0, pas de réduction (car $2^0 = 1$).

TRANS ?
- Ø → Oui : on demande d'appliquer une transformation aux coordonnées reçues et éventuellement réduites. Voir cette option dans le processeur VG où sont données toutes les explications.
- N → pas de transformation.

PROGRAMME DE TRANS ?
- Ø → On a placé en Item 2 un programme écrit en assembleur et assemblé au moyen du processeur GC. Ce programme sera appliqué aux coordonnées de segments.
- N → pas de programme de transformation.

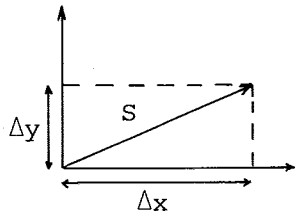
EPS * 256 = 100 → servir un nombre hexadécimal de 0 à 'FFF. C'est le paramètre α de stylisation (voir fonctionnement du processeur ST). Plus α est petit, plus le dessin stylisé se rapproche du dessin origine (identité si α=0).

SEUIL =

SX =

SY =

⟶ seuil, seuil en X, seuil en Y, pour chaque segment S reçu, ST calculera ses normes Δ, ΔX et ΔY avec $\Delta = |\Delta X| + |\Delta Y|$



Si $\Delta <$ seuil ou

$\Delta X <$ SX ou

$\Delta Y <$ SY ,

alors le segment n'est pas tracé.

SEUIL, SX et SY sont tous facultatifs (répondre Return ou D^C pour ne pas les servir). Ils sont exprimés par 3 chiffres hexadécimaux.

Exemples :

SEUIL = D^C
SX = 1 D^C
SY = D^C
⟶ élimination de tous les segments strictement verticaux

SEUIL = D^C
SX = D^C
SY = 1 D^C
⟶ élimination de tous les segments strictement horizontaux.

ERASE ?
- Ø ⟶ oui, effacement écran avant traçage.
- N ⟶ non, pas d'effacement écran.

Le traçage a lieu, ainsi que l'éventuelle création de fichier graphique si l'option SGF est active ; puis :

CØPY ?
- Ø ⟶ oui, faire un hard-copy de l'écran et proposer une nouvelle fois CØPY ?
- N ⟶ non, pas de hard-copy, et fin de travail.

! ⟶ fin de travail.

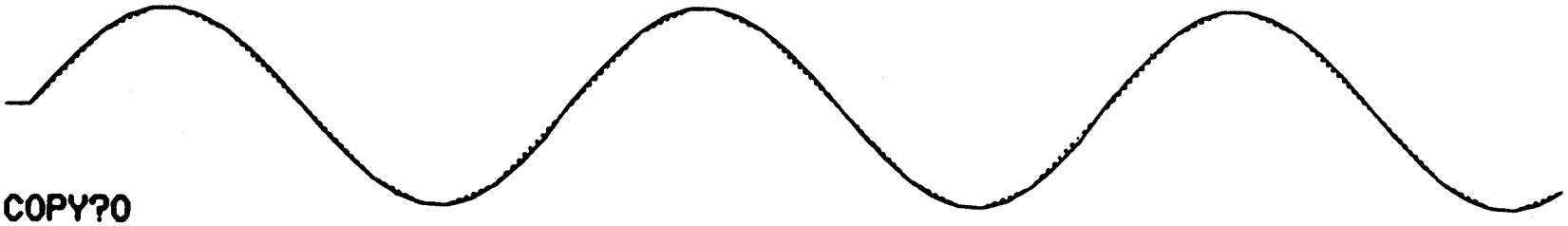
Nota :

On rappelle que, lorsque l'option SGF est active, seul le dessin stylisé est envoyé sur le fichier graphique, ce qui permet ensuite de le récupérer (en utilisant par exemple le processeur LP).

Exemples d'utilisation de ST en pages suivantes.

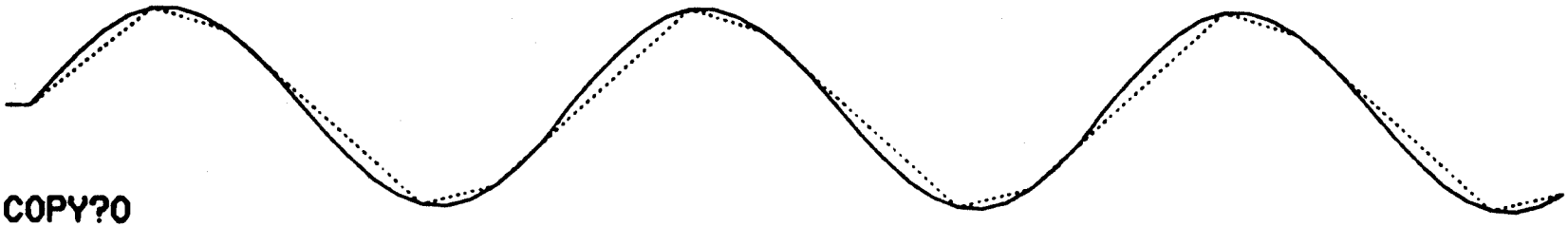
```
!ST
4014?O
SGF?N
REDUC=0
TRANS?N
PROGRAMME DE TRANS?N
EPS*256=25
SEUIL=
SX=
SY=
ERASE?N
```

Ici et en pages suivantes, variantes obtenues en appliquant le processeur ST sur un même dessin, en jouant sur le paramètre α (EPS*256).



```
COPY?O
```

```
!ST
4014?O
SGF?N
REDUC=0
TRANS?N
PROGRAMME DE TRANS?N
EPS*256=50
SEUIL=
SX=
SY=
ERASE?N
```

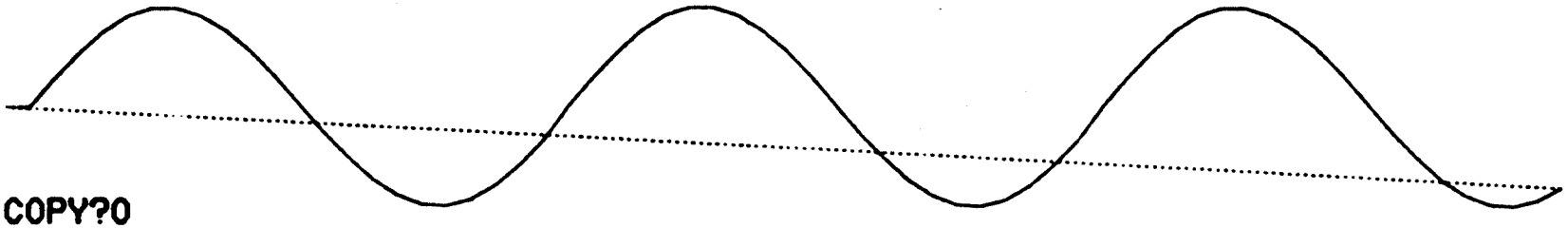


```
COPY?O
```

ST 7

!ST
4014?0
SGF?N
REDUC=0
TRANS?N
PROGRAMME DE TRANS?N
EPSx256=100
SEUIL=
SX=
SY=
ERASE?N
COPY?0

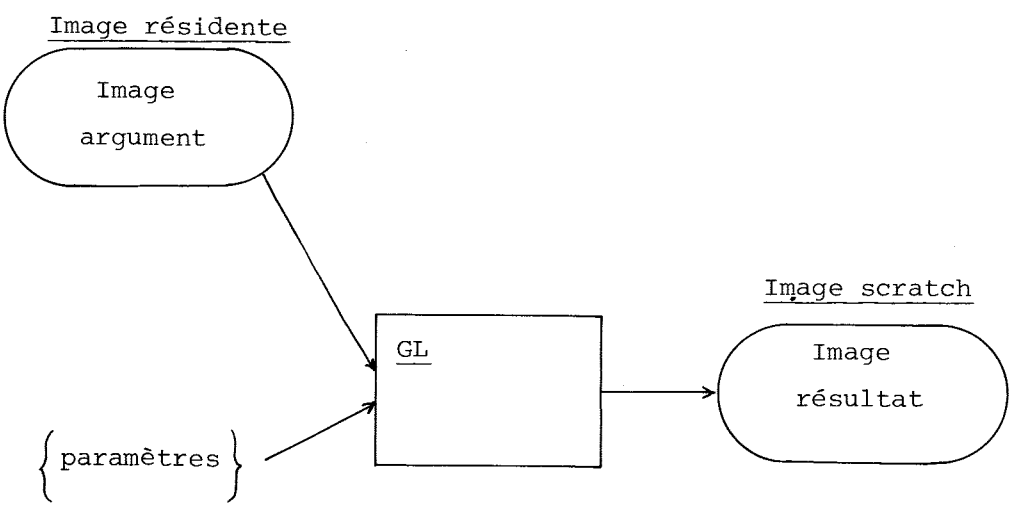
```
!ST
4014?O
SGF?N
REDUC=0
TRANS?N
PROGRAMME DE TRANS?N
EPS*256=150
SEUIL=
SX=
SY=
ERASE?N
```



```
COPY?O
```

PRØCESSEUR GL (GRANDES LIGNES)

Fonction.



GL travaille en deux phases :

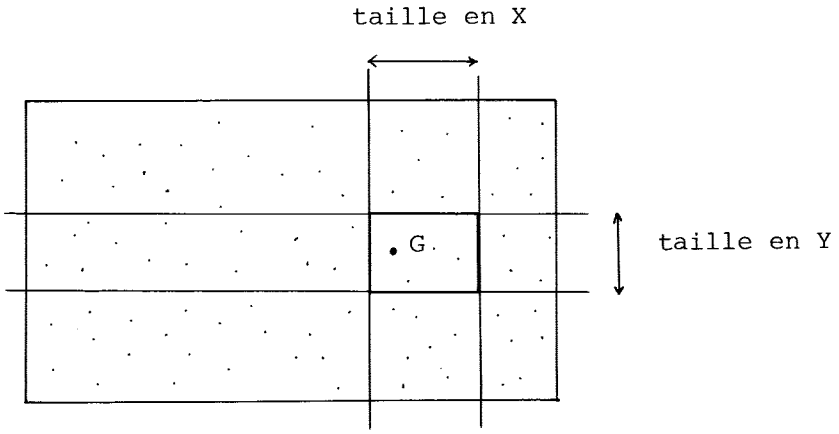
- 1ère phase : on lui fournit les paramètres :

. "mode" (comme dans VØ : c'est l'opérateur à utiliser pour construire l'image scratch)

. "raz" : remise à zéro ou non de l'image scratch (en général : mode = 3 et raz = oui).

. taille en X et taille en Y définissant un quadrillage de l'image.

A partir de ces paramètres, GL calcule, pour chaque rectangle R du quadrillage, son centre de gravité G, comme le montre le schéma ci-dessous.



Tous les points G obtenus sont stockés en image scratch suivant le mode spécifié.

On peut se limiter à cette première phase de GL, récupérer en scratch l'image obtenue et en faire ce que l'on veut (par exemple la travailler sous le processeur ZØ : voir les exemples). On peut aussi passer à la deuxième phase.

- 2ème phase : On répond "Oui" à la question "TREILLIS ?". GL va alors faire en vidéo sur l'image qu'il vient de calculer (image scratch) un traitement rappelant celui de VG (c'est-à-dire : tracé selon quadrillage, selon voisinage et selon directions spécifiées), à ceci près que GL travaille non pas sur des points (comme VG) mais sur les centres de gravités G (voir les exemples). On obtient ainsi des images très intéressantes.

Utilisation de GL.

Le dialogue se déroule de la façon suivante :

! TV
NØM > nom image D^C — Avant d'appeler GL, on charge sous TV l'image sur laquelle on veut travailler.
> D
> F
! GL → appel de GL

MØDE (AD=1/SB=2/ØR=3/AND=4/EØR=5)

- 1 → Addition
- 2 → Soustraction
- 3 → Ou logique
- 4 → Et logique
- 5 → Ou exclusif

(mode usuel =3; voir VØ)

RAZ ?

- Ø → remise à zéro image scratch
- N → pas de remise à zéro.

TAILLE X = tx D^C ⟶ répondre sur 1 à 4 chiffres hexadécimaux.

TAILLE Y = ty D^C ⟶ de même pour Y ;

tx et tx définissent un ensemble de rectangles R dans l'image.

Ici, GL calcule l'ensemble des centres de gravité G de chacun des rectangles R ; ces points G sont stockés en image scratch suivant le mode spécifié.

Cette première phase du travail de GL étant terminée, le dialogue se poursuit :

TREILLIS ?
- Ø ⟶ oui ; voir la suite du dialogue.
- N ⟶ non ; fin de travail pour GL ; l'image scratch est récupérable pour être exploitée comme on veut. (voir exemples).

NE ? { Ø / N } ⟶ spécification des directions de traçage par GL (Nord-Est, Nord, Nord-"West", "West")

N ? { Ø / N }

NW ? { Ø / N }

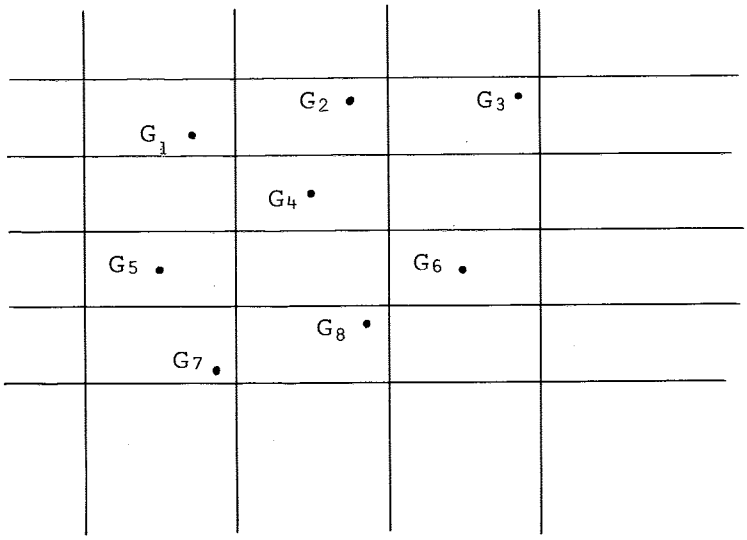
W ? { Ø / N }

PAS X = px D^C
PAS Y = py D^C ⟶ pas en X et Y ; définissent un quadrillage de parcours de l'image (analogue à "PAS L/C" de VG)

DELTA X = dx D^C
DELTA Y = dy D^C ⟶ pas en X et Y de recherche des voisins (analogue à "DX/DY" de VG).

Nota : On rappelle que GL travaille non pas sur des points mais sur les centres de gravités qu'il a calculés.

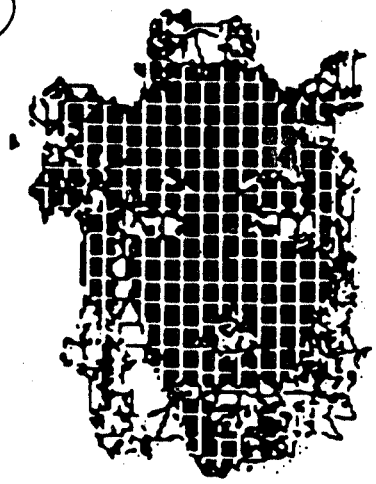
Ces centres de gravité des rectangles ne sont pas forcément alignés. En effet, la position du centre de gravité d'un rectangle dépend de la répartition des points dans ce rectangle, ainsi donc, en supposant par exemple qu'on



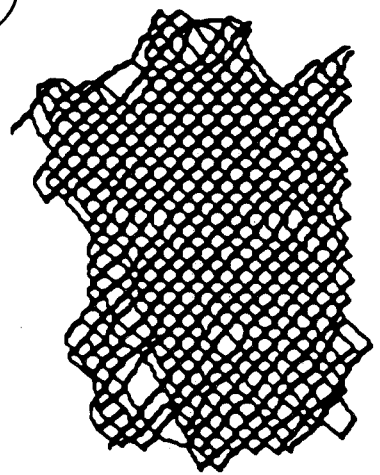
ne demande que la direction Nord, on n'obtiendra pas pour autant un tracé constitué de lignes verticales ; ces lignes pourront être brisées.

Exemples d'utilisation de GL

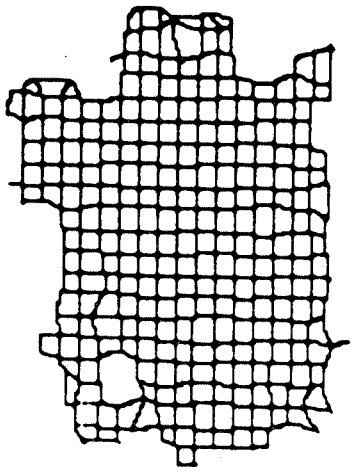
(1)



(2)



(3)



L'image origine est BOTTICELLI V (voir page EZ 6).
Les paramètres utilisés sont les suivants :

(2) mode = 3 avec remise à zéro ; tx = 4 ; ty = 6 ; treillis avec directions NE et NW (nord-est et nord ouest).

(3) même chose que 2 avec directions de tracé N et W (nord et ouest).

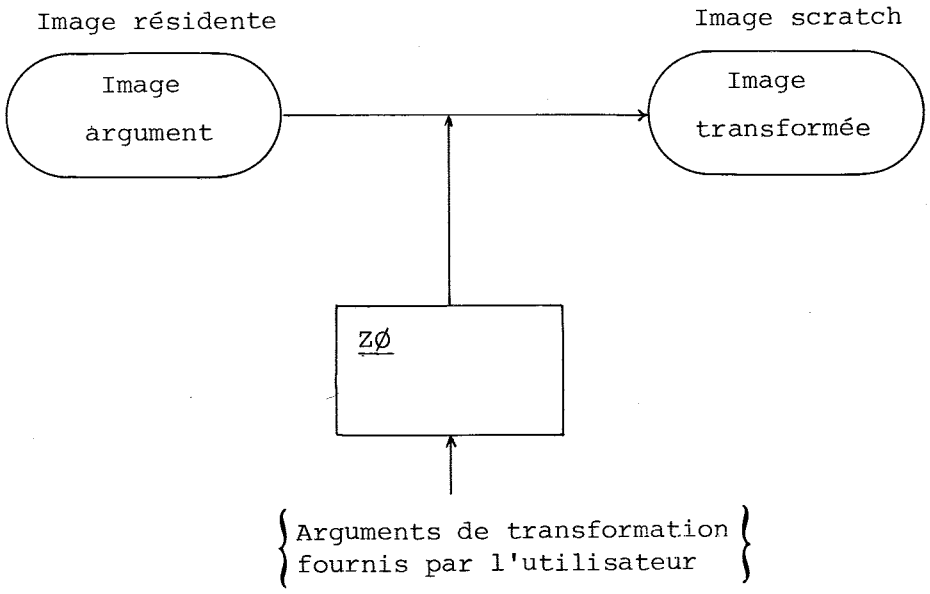
(1) résultat d'un "ou exclusif" effectué sous le processeur TV entre 3 et l'image origine.

PROCESSEUR ZØ ("ZOOM")

Fonction.

Ce processeur permet de définir et d'appliquer une transformation plus ou moins complexe sur une image numérique ; l'effet de zoom n'est qu'un cas particulier de transformation.

ZØ travaille à partir de l'image résidente, et stocke le résultat de la transformation en image scratch.



Les arguments sont principalement :

. matrice à coefficients fractionnaires.

. translation en X et Y.

. "filtre" : prise en compte du voisinage des points.

. taille des points : permet de remplacer des points par des surfaces unies ou striées.

Utilisation :

Avant d'appeler ZØ, il faut placer en mémoire l'image argument, par exemple sous TV comme ceci :

! TV
NØM > nom image D^C
> D
> F
!

Le dialogue avec ZØ se déroule comme suit :

! ZØ
PAS X = px D^C ⟶ définition du quadrillage de parcours de l'image. Donner un pas en X et un pas en Y sur 1 à 4 chiffres hexa.
PAS Y = py D^C

Nota : plus le pas est large, et plus le temps de traitement de l'image par ZØ est faible.

VØISINS ? { Ø ⟶ tenir compte du voisinage de chaque point lors du traitement
N ⟶ ne pas en tenir compte, c'est-à-dire traiter tout point blanc, quel que soit le nombre de ses voisins immédiats.

Si l'on a répondu Oui, ZØ va demander de préciser les conditions de prise en compte d'un point en fonction de son voisinage, c'est-à-dire :

. la distance de recherche des voisins.
. le nombre de voisins pour lequel le point courant doit être soit pris en compte soit ignoré.

Ce qui donne le dialogue suivant :

DELTA X = dx D^C } ⟶ distance de recherche des voisins ; réponse sur 1 à 4 chiffres hexa (voir exemples)
DELTA Y = dy D^C }

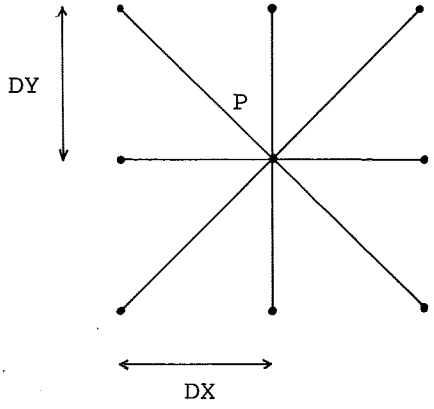
8 VØISINS ? { Ø ⟶ prendre en compte (ou non) le point s'il à 8 voisins.
N

7 VØISINS ? { Ø ⟶ de même pour 7 voisins.
N

etc...

1 VØISIN ? { Ø / N

0 VØISIN ? { Ø / N



un point P possède 0 à 8 voisins.

On notera que l'utilisation de cette option permet, par exemple, d'extraire les "contours" (il ne s'agit pas ici de contours au sens des processeurs extracteurs de contours : ZØ, ES et EZ) d'une image, en ne retenant, par exemple, que les points ayant de 0 à 7 voisins.

Ici, ZØ affiche le réticule pour que l'utilisateur désigne le centre de la transformation de l'image argument. Celui-ci peut alors frapper "Ø" ou "N" (oui/non).

- Oui : le centre de la transformation est désigné par la position du curseur.
- Non : on veut désigner le centre par ses coordonnées en X et Y. ZØ demande alors :

X (CENTRE) = { D^C ou Return / x

Y (CENTRE) = { D^C ou Return / y

Coordonnées du centre de l'image argument. Par défaut, X(CENTRE) = Y(CENTRE) = 0 ; l'image résultat sera centrée dans le coin supérieur gauche de l'écran TV.

Pour que l'image soit centrée au milieu de l'écran TV, servir X(CENTRE) = Y(CENTRE) = 80 D^C (=128 en décimal). ZØ demande ensuite les coefficients de la matrice de transformation ; les numérateurs puis les dénominateurs :

MUL 11 = { D^C / a_{11} D^C

MUL 12 =
MUL 21 =
MUL 22 =

Les coefficients s'expriment sur 1 à 4 chiffres hexadécimaux.

Les dénominateurs (b_{ij}) doivent être non nuls.

DIV 11 = { D^C ou Return / b_{11}

DIV 12 =
DIV 21 =
DIV 22 =

Exemples de matrices :

$\begin{pmatrix} 1/3 & 0/1 \\ 0/1 & 1/3 \end{pmatrix}$ Zoom de rapport 1/3

$\begin{pmatrix} 1/1 & 0/1 \\ 0/1 & 1/1 \end{pmatrix}$ Identité

Nota : Les coefficients MUL_{ij} et DIV_{ij} exprimés, comme on l'a déjà dit, sur 1 à 4 chiffres hexadécimaux, peuvent être négatifs. Un coefficient négatif s'exprime en utilisant le complément à deux, car on ne dispose pas du signe.

Exemple : FFFF = -1
FFFE = -2
FFF0 = -16 etc....

TRANSLATIØN X = { D^C ou Return / tx D^C

TRANSLATIØN Y = { D^C ou Return / ty D^C

Translation en X et Y.

A défaut, tx = ty = 0
Réponse sur 1 à 4 chiffres hexa.

MØDE (AD=1/SB=2/ØR=3/AND=4/EØR=5)= { 1 Addition, 2 Soustraction, 3 ØU logique, 4 ET logique, 5 ØU exclusif.] mot pour mot

C'est le mode de "stockage" en image scratch de l'image-résultat. Modes usuels = 3 et 5 (ou logique ; ou exclusif).

EPAISSEUR X = Ex D^C

EPAISSEUR Y = Ey D^C

MAILLE X = Mx D^C

MAILLE Y = My D^C

Ex, Ey, Mx, My s'expriment sur 1 à 4 chiffres hexa.

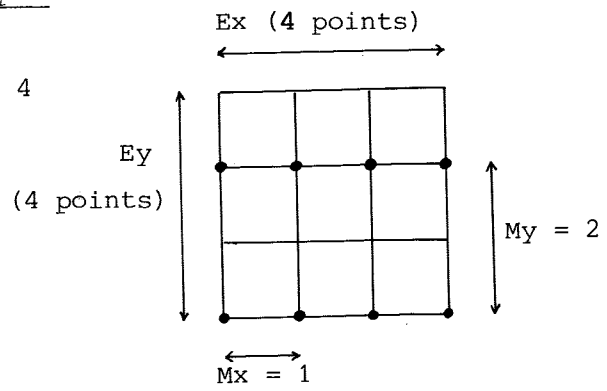
Ex et Ey définissent la surface (ou la taille) d'un point de l'image résultat.

Par exemple, si on sert Ex = Ey = 4,

un point de l'image résultat sera de taille 4 × 4. Mx et My indiquent comment ce point doit être tracé :

Exemple :

Ex = Ey = 4



Sur les points de la surface (Ex × Ey), seuls les points marqués sur le schéma seront générés.

(avec Mx = My = 1, on aurait eu les 16 points).

Remarques :

Lorsqu'on fait un grossissement (par exemple Zoom dans un rapport > 1), il se produit un "éclatement" des points de l'image qui se retrouvent disjoints. Pour éviter cela, leur donner une épaisseur suffisante.

La combinaison des différents paramètres ZØ permettent de très riches transformations (voir les exemples).

TRANSFØRMATION ?
- Ø ⟶ oui ; on a placé en item 2 un programme assembleur (assemblé par GC). Voir "Système vidéo-graphique, généralités", § transformations.
- N ⟶ non ; utilisation habituelle.

RAZ ?
- Ø ⟶ remise à zéro de l'image scratch.
- N

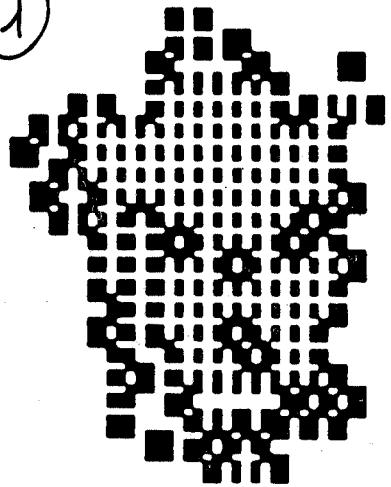
Ici, ZØ effectue la transformation demandée, dont le résultat est placé en image scratch.

```
! TV                    fin de travail de ZØ ; on va rappeler l'image scratch
NØM > D^C               sous TV.
> E
FØND = 0000
REC ? N
> U              ⟶      rappel et visualisation de l'image scratch.
NØM > D^C
ØRG = 0000
>
```

Exemples d'utilisation de ZØ

Image origine = BØTTICELLI V
(voir page EZ 6).

(1)

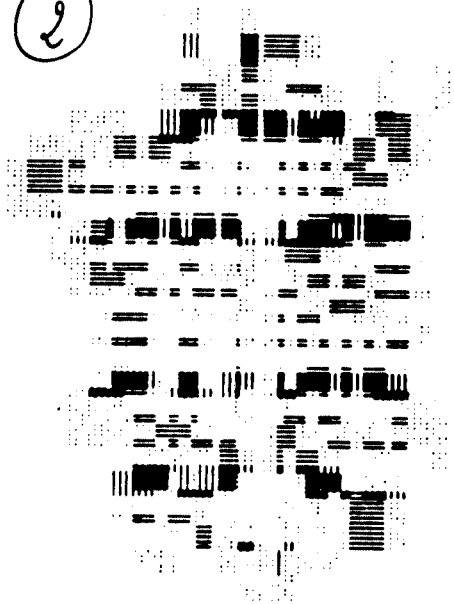


Paramètres utilisés :

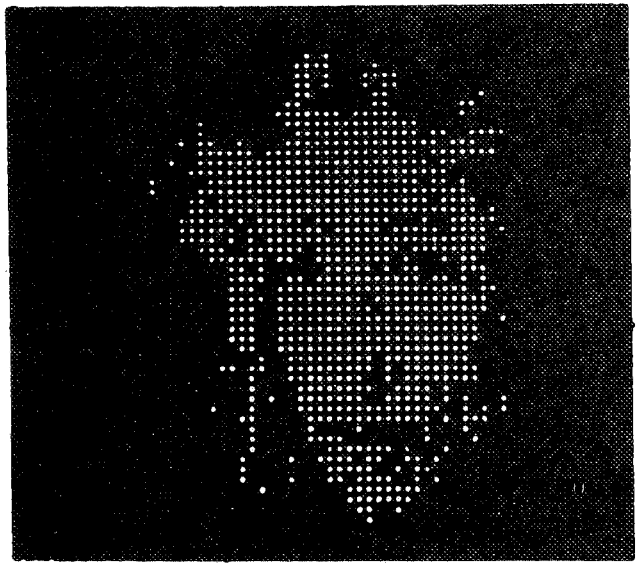
(1) px = 6 ; py = 9 ; voisins = non ;
X(centre) = Y(centre) = '80 ; $M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;
tx = ty = 0 ; mode = 5 avec remise à zéro ;
Mx = My = 1 ; Ex = 9 ; Ey = 'C .

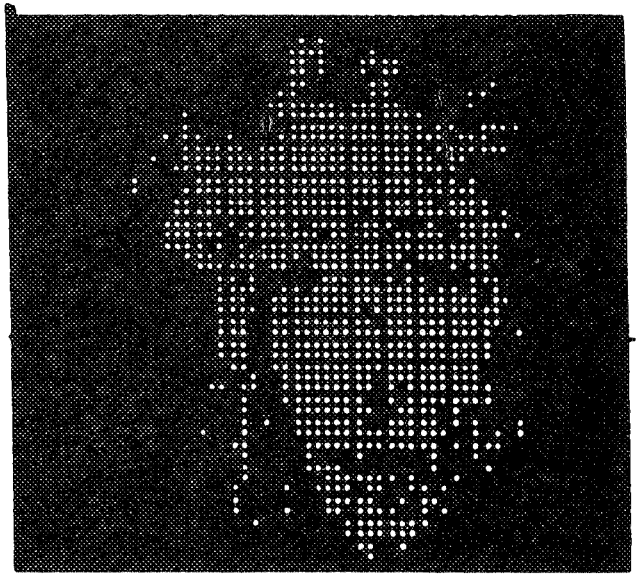
(2) idem avec $M = \begin{pmatrix} 9/8 & 0 \\ 0 & 9/8 \end{pmatrix}$ et Mx = My = 2,
ce qui donne des hachures et des points isolés.

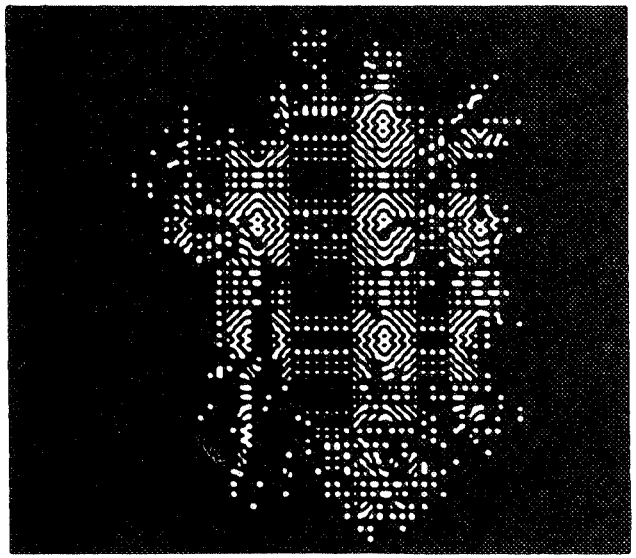
(2)



Exemples d'utilisation de ZØ (image origine = BØTTICELLI V ; voir page EZ 6)

① 

② 

③ 

Les paramètres utilisés sont les suivants :

① px = 3 ; py = 4 ; voisins = non ;
X(centre) = Y(centre) = '80 ; $M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;
tx = ty = 0 ; mode = 3 avec remise à zéro ;
Ex = 2 ; Ey = 3 ; Mx = 1 ; My = 1 ; pas de programme de transformation.

② idem avec $M = \begin{pmatrix} 'A/9 & 0 \\ 0 & 'A/9 \end{pmatrix}$

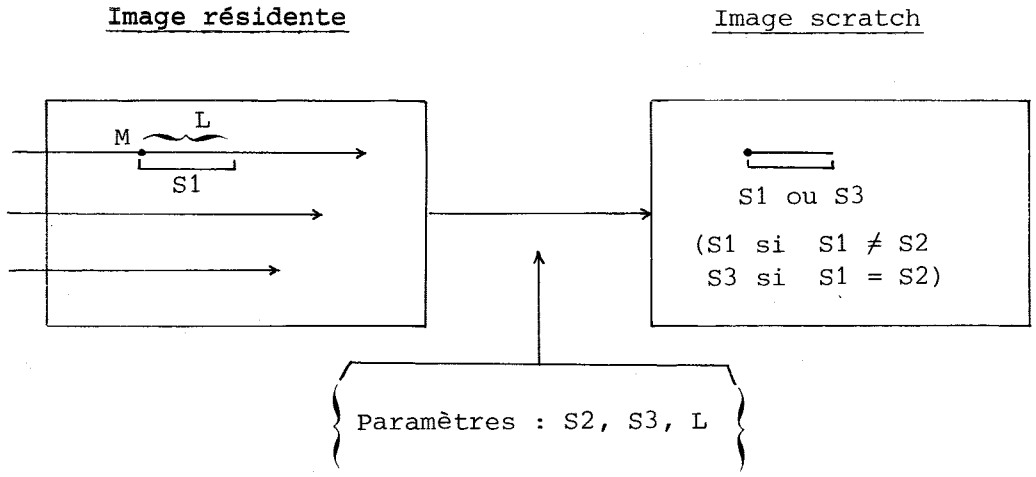
③ obtenue sous TV en faisant un "ou exclusif" entre 1 et 2.

PROCESSEUR PA (Pattern)

Fonction

Ce processeur permet de rechercher et de remplacer, dans une image numérique, une séquence de points caractéristique. Cette recherche et ce remplacement d'une séquence caractéristique de points par une autre séquence, se fait en fonction des paramètres suivants :

- mode de stockage de l'image obtenue en image scratch avec remise à zéro ou non de celle-ci en début de travail.
- quadrillage de parcours de l'image, c'est-à-dire "pas en x" et "pas en y".
- sens de parcours de l'image : horizontal ou vertical (la recherche est plus rapide dans le sens horizontal que dans le sens vertical).
- longueur de la séquence caractéristique de points.
- séquence caractéristique de points (ou pattern).
- séquence à substituer à la séquence caractéristique.



PA balaie l'image résidente

- horizontalement ou verticalement selon spécification ;
- selon le quadrillage spécifié ;

En chaque point M du parcours, PA examine la séquence de bits S_1 de longueur L (L est spécifiée par l'utilisateur) :

- si $S_1 = S_2$ (S_2 est la séquence de bits donnée par l'utilisateur en réponse à la question "PATTERN1=", voir utilisation de PA), PA remplace S_1 par S_3 en image scratch (S_3 étant la séquence de bits donnée par l'utilisateur

en réponse à la question "PATTERN2=" ; voir utilisation de PA).

- si S1 ≠ S2, S_1 n'est pas remplacée par S_3 et donc S_1 est stockée inchangée en image scratch.

Le stockage se fait selon le mode spécifié par l'utilisateur (en général, ou logique).

Utilisation de PA

Le dialogue avec PA se déroule comme suit :

!TV — chargement sous TV par exemple

NØM > nom image argument D^C — de l'image à traiter

> D

> F

!PA → appel de PA

MØDE (AD=1/SB=2/ØR=3/
AND=4/EØR=5)=

1	Addition	(c'est le mode de stockage en image scratch)
2	Soustraction	
3	Ou logique	
4	Et logique	
5	Ou exclusif	

RAZ ?

Ø	Oui, remise à zéro image scratch en début de travail
N	pas de remise à zéro.

HØRIZONTAL ?

Ø →	Oui, sens de parcours horizontal
N →	Non : le parcours sera vertical

PAS X = px D^C → Pas en x et Pas en Y définissent le quadrillage de parcours de l'image. Répondre un nombre de 1 à 4 chiffres hexa.

PAS Y = py D^C

LØNGUEUR = lD^C → longueur de la séquence caractéristique de points (ou bits). Répondre un nombre hexadécimal de 1 à '20 (on peut donc aller jusqu'à 32 bits).

PATTERN1 = abcd → première partie de la séquence caractéristique ; abcd sont 4 chiffres hexadécimaux, de '0000 à 'FFFF permettant donc d'exprimer de 1 à 16 bits (séquence S2).

--> a'b'c'd' → séquence de remplacement (séquence S_3).

Ici, si la longueur l est ⩽ 16 (10 en hexadécimal), PA commence son travail de recherche et de remplacement, si la longueur l est > 16 bits (> 10 hexadécimal), PA demande la suite de la séquence caractéristique :

PATTERN2 = efgh → seconde partie de la séquence caractéristique.

--> e'f'g'h' → séquence de remplacement.

PA effectue la recherche et le remplacement. Quand il a terminé, l'image scratch est à récupérer pour visualiser le résultat par exemple sous TV.

```
!TV                    → Récupération sous le processeur TV
NØM > D^C                de l'image scratch générée par PA.
> E
FØND = 0000
REC? N
> U
NØM > D^C
ØRG = 0000
> D                    → display image obtenue
>
```

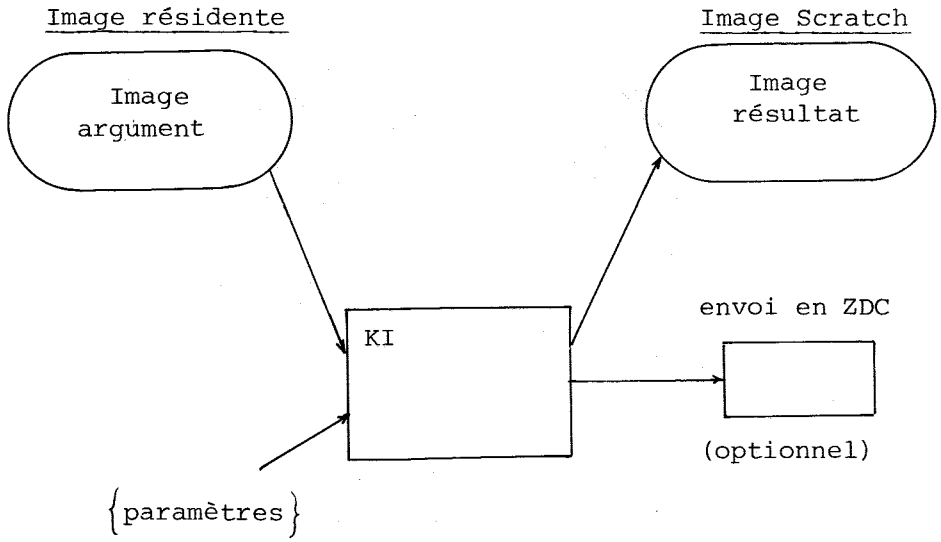
```
!TV
NOM>BOTTICELLI V
>D
>F
!PA
MODE (AD=1/SB=2/OR=3/AND=4/EOR=5)=3
RAZ?0
HORIZONTAL?0
PAS X=1
PAS Y=1
LONGUEUR=8
PATTERN1=FF00
-->CC00
!
```

Exemple d'utilisation de PA

Ci-contre la reproduction sous VG de l'image obtenue en appliquant le processeur PA sur l'image BOTTICELLI V (dont la reproduction du négatif sous VG est donnée page EZ 6).

PROCESSEUR KI

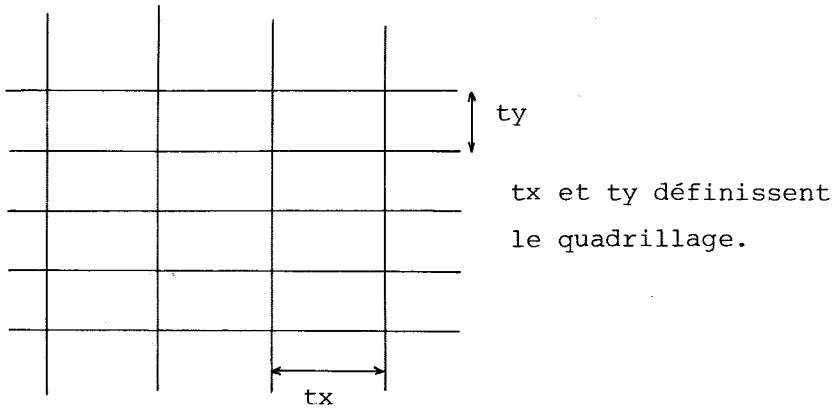
Fonction.



Les paramètres sont les suivants :

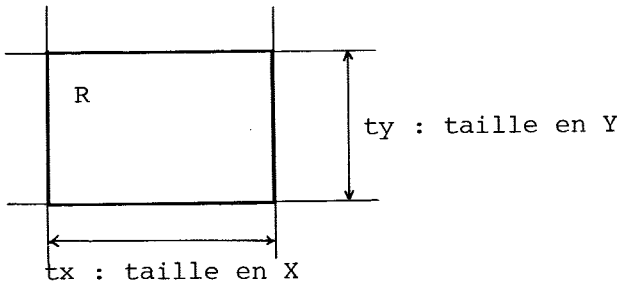
. un mode de stockage en image scratch (comme pour VØ, GL etc...)

. un quadrillage de l'image (taille en X, taille en Y, voir le dialogue) : l'image est alors découpée en un ensemble de rectangles.



un facteur multiplicatif α dont nous verrons ci-après l'utilisation.

A partir de ces données (image argument et paramètres), KI va calculer, pour chaque rectangle R de l'image, les vecteurs $\vec{V1}$ et $\vec{V2}$ situés sur les axes d'inertie du rectangle, et fonction de la répartition et de la densité des points à l'intérieur du rectangle.



KI calcule :

$$X_G = \frac{\sum_R m\ X}{\sum_R m} \qquad Y_G = \frac{\sum_R m\ Y}{\sum_R m}$$

coordonnées du centre de gravité du rectangle R

Nota : on note $\sum_R$ = somme sur tous les points du rectangle R

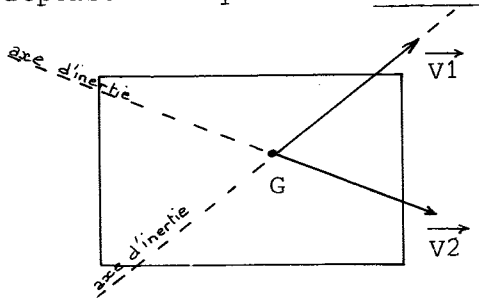
m = "masse" du point (1 ou 0)

$$\mathcal{M} = \begin{pmatrix} I_{XX} & I_{XY} \\ I_{YX} & I_{YY} \end{pmatrix} \quad \text{avec} \quad \begin{cases} I_{XX} = \sum_R m\ X^2 \\ I_{XY} = I_{YX} = \sum_R m\ XY \\ I_{YY} = \sum_R m\ Y^2 \end{cases}$$

KI calcule ensuite les vecteurs propres $\vec{V1}$ et $\vec{V2}$ de $\mathcal{M}$.

$\vec{V1}$ et $\vec{V2}$ sont les solutions de l'équation : $\mathcal{M}\ \vec{V} = s\vec{V}$ avec s = valeur propre de $\mathcal{M}$ (KI résout : déterminant ($\mathcal{M}$ - sI) = 0 pour obtenir s1 et s2 avec en général s1 ≠ s2, à partir desquels il calcule $\vec{V1}$ et $\vec{V2}$)

Enfin, KI applique à $\vec{V1}$ et $\vec{V2}$ le facteur multiplicatif α spécifié par l'utilisateur en réponse à la question "AMPLI IXY=" (voir utilisation de KI).



$|\vec{V1}| \longleftarrow \alpha\,|\vec{V1}|$

$|\vec{V2}| \longleftarrow \alpha\,|\vec{V2}|$

Utilisation de KI.

Le dialogue avec KI se déroule comme suit :

! TV
NØM > nom de l'image argument D^C
> D
> F
! KI
MØDE (AD=1/SB=2/ØR=3/AND=4/EØR=5) =

- 1 Addition
- 2 Soustraction
- 3 Ou logique
- 4 Et logique
- 5 Ou exclusif

mode de stockage en image scratch; mode usuel = 3.

RAZ ?
- Ø ⟶ remise à zéro image scratch.
- N ⟶ pas de remise à zéro.

TAILLE X = tx D^C ⟶ taille en X (1 à 4 chiffres hexa).
TAILLE Y = ty D^C ⟶ taille en Y (1 à 4 chiffres hexa).

AMPLI IXY = α D^C ⟶ facteur multiplicatif à appliquer aux vecteurs obtenus ; réponse sur 1 à 4 chiffres hexadécimaux.

GRAPHIQUE ZDC ?
- Ø ⟶ oui, envoi en ZDC des coordonnées de segments.
- N ⟶ non, pas d'envoi en ZDC.

Ici, KI effectue ses calculs, stocke les vecteurs obtenus en image scratch et les envoie éventuellement aussi en ZDC.

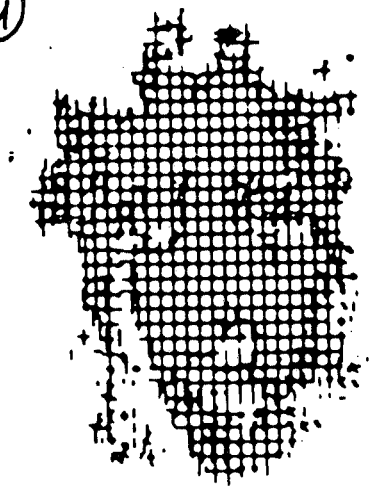
! ⟶ fin de travail ; le résultat est disponible en image scratch ; on peut l'afficher comme ceci :

! TV
NØM > D^C
> E
FØND = 0000
REC ? N
> U
NØM > D^C
ØRG = 0000
>

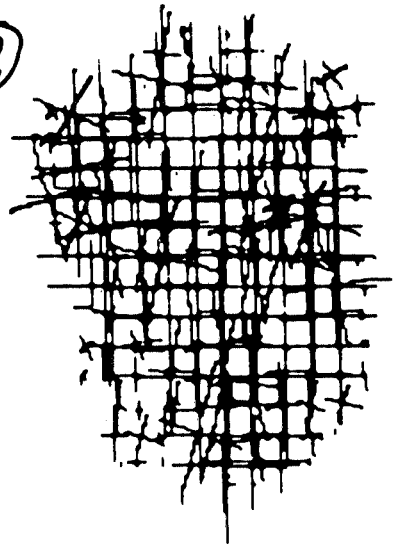
affichage sous TV de l'image scratch construite par KI.

Exemples d'utilisation de KI

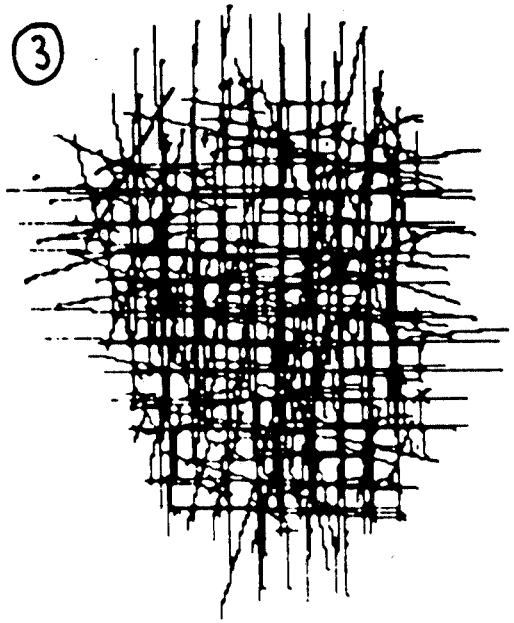
①



②



③



L'image origine est BOTTICELLI V (voir page EZ 6).

Les paramètres utilisés sont les suivants :

① mode = 3 avec remise à zéro ;
$tx = 4$; $ty = 6$; $\alpha = 6$.

② mode = 3 avec remise à zéro ;
$tx = 9$; $ty = C$; $\alpha = 6$.

③ comme 2 avec $\alpha = C$.

PROCESSEURS GC et 3D

Fonction

- GC permet d'assembler un programme source écrit en assembleur et placé dans Item 1 et place le binaire en Item 2. Il permet de plus d'éditer en clair un programme binaire (dump), et de le modifier en utilisant le processeur ED, ce qui facilite la mise au point.

- 3D permet d'exécuter un programme binaire placé en Item 2 (item de type P), ce programme pouvant contenir :
  - du code machine (programme binaire),
  - des séquences graphiques (type GR) à interpréter, pouvant contenir les primitives 5 et 6 pour la troisième dimension.

Utilisation

Dans un programme source à assembler par GC, le D^C ou le Return, habituels sous l'assembleur ASSYS, sont remplacés par le caractère # :

~~~ # LAIØ0 # ETI : EQUØ Ø # STAØ TRAV # ~~~ etc...

On dispose sous GC des commandes G, D, U, C et F qui réalisent les fonctions suivantes :

- Commande G : assemblage du programme source placé en Item 1, le binaire obtenu sera placé en Item 2, celui-ci recevra le type P (programme). Il est nécessaire avant d'utiliser cette commande d'avoir au préalable donné un nom à Item 2 si l'on veut pouvoir le stocker ensuite en bibliothèque.

Exemple :

```
! I+
NØM > BINAIRE D^C  ------>  nom du futur item programme.
! MX               ------>  BINAIRE qui pour le moment est de type V (vide) est
                             placé en Item 2
! IL
NØM > SØURCE D^C   ------>  chargement de l'item de type T contenant le programme source.
! GC               ------>  appel du processeur GC
> G                ------>  commande G : assemblage, link et load. Le binaire sera placé en Item 2 qui recevra le type P.
! MX               ------>  BINAIRE est placé en Item 1,
! IS                        ce qui permet de le stocker en bibliothèque.
!
```
~~~

- Commande D : édition dans Item 1 d'un programme binaire (D comme Dump). Par cette commande, GC convertit en asci le binaire, un mot de 16 bits étant remplacé par 4 caractères asci suivis de 4 blancs. L'utilisateur peut ainsi éditer et modifier son programme (voir exemple ci-après).

- Commande U : opération inverse de D. Item 1 contient le programme édité en asci et éventuellement modifié par l'utilisateur. La commande U provoquera la conversion de ce texte en binaire.

Exemple :

! IL

NØM > BINAIRE D^C ⟶ chargement du programme à modifier

! GC

> D ⟶ commande D : le programme est converti en asci pour que l'utilisateur puisse le visualiser et le modifier en utilisant ED.

! ED ⟶ appel de ED.

≀

ici, l'utilisateur fait ses modifications, puis sort de ED par F^C.

! GC

> U ⟶ commande U : le programme est remis en binaire.

! IS ⟶ stockage en bibliothèque du nouveau binaire.

!

- Commande C : retour au CCI (revenir par !GØ)

- Commande F : fin ; retour à GE (!).

Nota 1 : Pour éditer son dump sous ED, l'utilisateur dispose de toutes les commandes de ED, y compris D^C et N^{CS} que nous explicitons ici :

- D^C : ED mémorise l'adresse α dans Item 2 du mot correspondant à celui désigné par la position du curseur.

- N^{CS} : ED place en Item 1 à la position courante du curseur l'adresse α précédemment définie par le D^C.

Ceci facilite l'intégration de "patches" dans un programme.

Nota 2 : Sous le processeur SC (scénario), on a la possibilité de charger un item de type programme et de l'exécuter comme ceci :

~~~ ØPRØG1 ; % 2(Ø ;) ~~~

chargement en Item 2 de l'item PRØG1, de type P.

exécution de ce programme deux fois de suite.
~~~

Nota 3 : Le processeur 3D peut exécuter du code binaire et des séquences graphiques, celles-ci devant être placées dans le programme sous forme de constantes asci référencées derrière l'appel de l'interpréteur comme le montre l'exemple ci-après. Sur cet exemple on notera aussi que les séquences à interpréter sont délimitées par le caractère ⌀, et que, pour sortir de 3D, il suffit de fournir à l'interpréteur la primitive F.

Exemple de programme exécutable par le processeur 3D

```
                    ⋮
SEQ1:               ASCI        "A1B2B3B4B⌀"
SEQ2:               ASCI        "A1B2B⌀"
                    ⋮
SEQFIN:             ASCI        "F"
                    ⋮
                    BSR         'FF80,C     < appel interpréteur
                    WØRD        SEQ1        < pour séquence SQ1
                    LAI         2
                    STA         TRA
                    LXI         8           < nombre d'itérations
ETIQ:               EQU         ⌀
                    BSR         'FF80,C     < appel interpréteur
                    WØRD        SEQ2        < pour séquence SQ2
                    JDX         ETIQ        < itération
                    ⋮
                    BSR         'FF80,C     < appel interpréteur
                    WØRD        SEQFIN      < pour séquence finale,
                                            < qui provoquera le
                                            < retour à GE.
```

Nota 4 : C'est au moyen du processeur GC qu'on peut générer des items de type P pour les utiliser ensuite pour faire des transformations, lors de l'utilisation de certains processeurs de SMC comme par exemple TA, TB, ZØ, ST, etc... Pour l'interface entre les processeurs offrant cette possibilité et SMC, se reporter à la présentation de SMC § 6 - 6.

PROCESSEUR SØ (Son)

Fonction

SØ transforme le clavier de la visu en un "piano", sans possibilité d'accord car on ne peut émettre qu'un son à la fois. Il établit en effet une correspondance :

caractère ⟶ (fréquence x constante α)

- caractère : voir ordre des caractères ci-après
- constante α ∈ [0,9], elle définit l'"octave".

Implicitement, le son émis est sinusoïdal ; optionnellement, on peut :

- remplacer la sinusoïde par le contour supérieur d'une image TV.
- ou encore remplacer la sinusoïde par la moyenne du contour de l'image TV

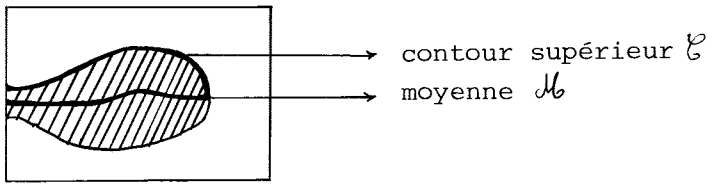


image TV

la sinusoïde ∿ peut être remplcée par $\mathcal{C}$ ou $\mathcal{M}$.

Le son numérique est envoyé sur le coupleur universel CU1 en T1600, CU3 en Solar.

SØ est utilisable sous ACN = : SYS uniquement.

Utilisation

Le dialogue avec SØ se déroule de la façon suivante :

! SØ

≥ M ⟶ "musique"

Ici, on peut rentrer une suite quelconque de caractères dont chacun provoquera l'émission d'un son.

Si l'on veut changer d'octave, il suffit de taper : !α avec α∈[0,9]

Pour revenir au dialogue avec SØ, taper Ø ou Return.

≥ I ⟶ "interprétation" d'une image.

NØM ≥ nom image D^C

ADDITION SIGNAUX ?
- Ø ⟶ oui : SØ prendra la moyenne $\mathcal{M}$.
- N ⟶ non : SØ prendra le contour supérieur $\mathcal{C}$ de l'image.

```
> S  ------>  retour au signal sinusoïdal.
> etc...
:
> F  ------>  fin
!
```

Caractères utilisables

Ce sont les caractères de code hexadécimal '22 à '5F. C'est à dire :

" (22 en hexadécimal)
(23)
:
/ (2F)
0 (30)
:
9 (39)
: (3A)
:
? (3F)
@ (40)
A (41)
:
Z (5A)
:
- (5F)

La fréquence obtenue est d'autant plus élévée (son aigu) que la valeur du caractère est grande.

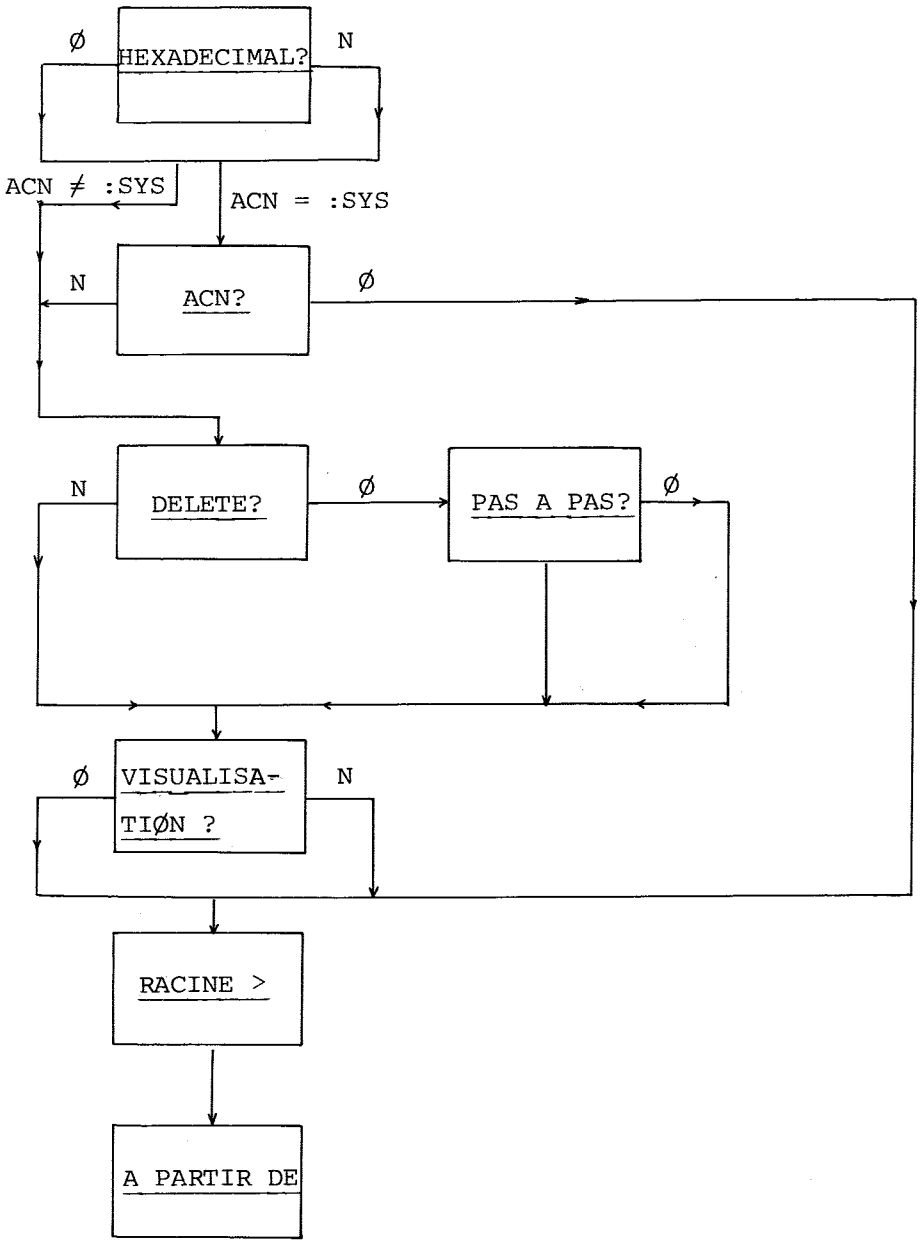
PROCESSEUR EX

1 - GENERALITES.

■ *Fonction.*

EX permet de lister tout ou partie du catalogue des noms avec affichage optionnel de la valeur sur TV numérique ; de plus il permet de deleter tout ou partie de ce catalogue (nom et valeur).

■ *Dialogue.*



■ *Exemple de dialogue.*

!EX
HEXADECIMAL ? N
ACN ? N on est sous ACN = :SYS
DELETE ? N
VISUALISATIØN ? N
RACINE > ABCDE D^{C}
A PARTIR DE D^{C}
{
}
liste des noms de racine ABCDE
{
}
!

Remarques :

1 - Alt-Mode a pour effet d'interrompre EX ; on revient à GE(!)

2 - On manipule sous EX les noms tels qu'ils figurent au catalogue. Il en résulte que l'on désigne les Items par leur nom compacté.

2 - UTILISATION.

■ *HEXADECIMAL ?* répondre { Ø / N }

Si l'on répond Ø (oui), les noms seront listés en clair.

Si l'on répond N (non), ils seront listés sous formes hexadécimale et aussi en clair. Cela a de l'intérêt notamment quand un nom contient des blancs et/ou des caractères spéciaux non visualisables.

■ *ACN ?* répondre { Ø / N }

Cette question n'est posée que si l'ACN de l'utilisateur est :SYS.

S'il répond Øui, cela signifie que l'on veut travailler sur le catalogue des n^{OS} de compte.

■ *DELETE ?* répondre { Ø / N }

Si l'on répond oui, cela signifie que l'on veut deleter un ensemble de noms et valeurs sur le catalogue. On le deletera en pas à pas ou globalement en fonction de la réponse donnée à la question "Pas à Pas ?".

■ PAS A PAS ? répondre { Ø / N }

Si l'on répond oui, le processeur EX affichera un par un les noms sélectionnés, laissant la possibilité de deleter ou non chacun d'entre eux et la valeur associée : pour chaque nom, on répondra "-" pour le deleter, et "+" ou D^C ou Return pour le conserver (cf : exemple 2 ci-après).

■ VISUALISATION ? répondre { Ø / N }

Si l'on répond oui, EX chargera en mémoire la valeur de chaque nom sélectionné, et cette valeur sera affichée en TV numérique ; cette option est intéressante notamment quand la valeur est une image TV, ce qui permet de visualiser rapidement un ensemble d'images TV de même racine (cf : exemple 3 ci-après).

■ RACINE > répondre par une chaîne de 0 à 19 caractères suivie de D^C. La racine spécifiée sélectionne un sous-ensemble de noms ; si elle est vide, tous les noms sont sélectionnés.

■ A PARTIR DE répondre { chiffre de 0 à 9 / D^C }

Ce chiffre indique de quelle façon doit se faire la sélection dans l'arbre des noms à partir du 1er nom obtenu à partir de la racine. Par exemple si on sert 2, les noms seront sélectionnés à partir du nom obtenu par la racine à condition qu'ils aient au moins leur deux premiers caractères communs avec celui-ci (cf : exemples ci-après).

Exemple 1 :

Soit le contenu suivant du catalogue des noms : A
ASMS
B0
B1
2
D^C

```
!EX
HEXADECIMAL ? N
ACN ? N
DELETE ? N
RACINE > AS D^C
A PARTIR DE 0
```

ASMS
ASMSL
ASMT
ASMTL
ASSYS
AT1
B0
B1
.
.
.
tous les noms suivants dans le catalogue sont listés
!

A PARTIR DE 1

ASMS
ASMSL
ASMT
ASMTL
ASSYS
AT1
!
tous les noms commençant par A sont listés à partir du premier nom de racine AS

A PARTIR DE 2 / D^C

ASMS
ASMSL
ASMT
ASMTL
ASSYS
!
tous les noms commençant par AS sont listés à partir du 1er nom de racine AS

A PARTIR DE 3

ASMS
ASMSL
ASMT
ASMTL
!
tous les noms commençant par ASx sont listés à partir du premier nom de racine AS avec x = 3e caractère de ce nom.

A PARTIR DE 4 aurait sélectionné ASMS et ASMSL

Nota : Si n = longueur de la racine, alors il est équivalent de répondre D^C ou n à la question A PARTIR DE.

Exemple 2 : Delete en pas à pas.

```
!EX
HEXADECIMAL ? N
ACN ? N
DELETE ? Ø
PAS A PAS ? Ø
VISUALISATIØN ? N
RACINE > SI Ø PF Ø D^C
A PARTIR DE D^C
SI Ø PF Ø CØM ?? +
SI Ø PF Ø TAB ?? -
SI Ø PF Ø CØM ?? +
SI Ø PF Ø VAL ??
SI Ø PF Ø CØM ??
!
```

En pas à pas, chaque nom est listé et on a le choix pour chacun :

. soit de le conserver : répondre +

. soit de le deleter : répondre -

Le catalogue est réordonné après chaque delete d'où la séquence obtenue.
Contenu initial du catalogue :
SI PF CØM
SI PF TAB
SI PF VAL

EX 5

Exemple 3 : Visualisation rapide d'un ensemble d'images TV de racine commune C-

```
!EX
HEXADECIMAL ? N
ACN ? N
DELETE ? N
PAS A PAS ? N
VISUALISATIØN ? Ø
RACINE > C - D^C
A PARTIR DE D^C

C - EXEMPLE 1                    Les images sont visualisées
C - EXEMPLE 2                    au rythme du parcours
C - TRAVAIL 1                    de l'arbre.
C - TRAVAIL 2
etc...
```

PROCESSEUR SA

1 - GENERALITES.

1.1 - Fonctions.

Le processeur SA permet la sauvegarde, la modification, la restauration de toute valeur accessible par un nom appartenant à l'espace des Noms-valeurs : programme appelable par !CALL, processeur XY de GE, item en format interne, image TV, etc... Il s'agit toujours de noms appartenant à l'arbre des noms (donc, pour les items donnant lieu à compactage, il s'agit du nom compacté).

1.2 - Assignation des Unités logiques.

La restauration se fait à partir de l'UL 'A

La sauvegarde se fait sur l'UL 'B

Assignation de 'A, sous ACN ≠ :SYS on est obligé de restaurer en batch.

- si temps partagé et ACN = :SYS !ASSIGN A = CR1 (CR1 étant le lecteur de cartes)
- en batch, quel que soit l'ACN !ASSIGN A = I

Assignation de 'B,

- perforation de cartes !ASSIGN B = CU1 (CU1 étant le perforateur)

2 - COMMANDE M MISE A JOUR NOM ET/OU VALEUR.

Exemple :

```
! SA
>M
NØM > EXEMPLE D^C          Chargement nom-valeur éventuellement modification nom et/ou valeur
? !DEBUG D^C

? !GØ D^C
NØM > EXAMEN               le nom est affiché pour vérification de sa nouvelle valeur éventuelle.
!
```

La commande M agit de la manière suivante :

1°) le nom est chargé en '200 (s'il existe bien entendu)
la valeur est chargée en '20A

2°) SA delete le nom et la valeur dans l'espace des noms-valeurs.

3°) appel CCI pour faire éventuellement des modifications du nom et/ou de la valeur sous Debug

4°) si l'on revient par !GØ SA recrée le nom et la valeur éventuellement modifiés.

si l'on ne revient pas par !GØ (par exemple si on fait !Return ou !GE, etc...), le nom et la valeur sont perdus.

3 - COMMANDE D Delete Nom et Valeur.

Exemple :

```
!SA
>D
NØM > EXEMPLE DC        EXEMPLE (nom et valeur) est détruit
!
```

4 - COMMANDE S Punch binaire sur 'B nom et valeur.

Avec PRINT OFF sur le perforateur !

Le deck de cartes obtenu contient le nom et la valeur en format binaire. Il n'y a pas de carte jouant le rôle de délimiteur de fin (pas nécessaire).

Les cartes sont numérotées modulo 64 en ligne 12

```
- carte 1   punch colonne  1  ligne 12
-   "  2     "       "     2    "    "
- ....
-   " 64     "       "    64    "    "
-   " 65     "       "     1    "    "
  etc ...
```

Exemple :

```
! SA                          'B ayant été assignée
> S
NØM > EXEMPLE DC
!
```

5 - COMMANDE R RESTAURATION A PARTIR DE UL'A.

Le nom et la valeur se trouvent dans le deck obtenu par S.

Exemple :

```
?!ASSIGN Ø A = CR1 D^C      → assignation de l'UL'A à CR1 (ACN = :SYS)
?!GE D^C                    → appel de SMC
* G
!SA                         → appel de SA
>R                          → restauration ; ici le deck de cartes est lu
NØM > EXEMPLE                 sur le lecteur de cartes, puis SA affiche le
!                             nom de l'item restauré.
```

6 - COMMANDE F RETOUR A GE (!).

Exemple :
```
!SA
>F
!
```

7 - NOTA : UTILISATION DE ALT-MODE.

L'interruption par Alt-Mode sous SA provoque un retour à GE(!). Attention : si on est en cours de punch, la perfo peut se trouver en attente de caractères auquel cas on aura un décalage de perforation à la reprise. Donc, si l'on interrompt et si on redemande un punch, faire un CLEAR de la perfo avant la reprise.

PROCESSEUR KA

1 - GENERALITES.

1.1 - Fonctions.

KA permet de travailler sur des items de type T (texte) I (film digital) et D (programme graphique).

Sur ces items il permet de faire des opérations de Sauvegarde et Restauration de leur VALEUR. Quant à leur nom il se trouvera dans des cartes en-tête dont nous parlerons ci-dessous.

1.2 - Assignations des Unités Logiques.

La restauration se fait à partir de l'UL'A.

La sauvegarde se fait sur l'UL 'B.

Assignation de 'A, sous ACN ≠ :SYS on est obligé de restaurer en batch :

- Si Temps Partagé et ACN = :SYS !ASSIGN A = CR1 (CR1 étant le lecteur de cartes)
- En batch, quel que soit l'ACN !ASSIGN A = I

Assignation de 'B, deux cas sont possibles :

- perforation de cartes !ASSIGN B = CU1 (CU1 étant le perforateur).
- liste sur imprimante { ACN = :SYS !ASSIGN B = LP1 ; ACN quelconque !ASSIGN B = Ø

1.3 - Format des cartes.

La valeur de l'item est stockée sur (ou restaurée à partir de) n cartes 74 colonnes avec :

- soit n = 35
- soit n < 35 auquel cas on a - soit une carte fin "*/" colonnes 1.2
  - soit 35 - n cartes à blanc.

On a aussi un en-tête de 4 cartes (voir ci-après les différentes commandes de KA). Il contient notamment le nom de l'Item. Cet en-tête peut être utilisé ou non, utilisé avec la valeur ou utilisé seul.

1.4 - Item 2.

Item 2 est écrasé par KA car celui-ci utilise GI.

2 - COMMANDE W - RETOUR CCI.

Exemple : !KA
> W
? !ASSIGN ...
{
}
? !GØ D^C
>

utile pour faire assignations ou pour patcher KA lui-même. On revient par !GØ D^C.

3 - COMMANDE F - RETOUR A GE.

Exemple : !KA
{
}
> F
!

4 - COMMANDE P - PUNCH SUR 'B (item courant).

C'est la VALEUR de l'item placé dans Item 1 qui est punchée sur 35 cartes de 74 colonnes (35 lignes d'écran), à moins que l'une des lignes ne commence par * / ce qui aura pour effet de terminer le punch. Le punch se fait en format alphanumérique on a donc intérêt à mettre la perforatrice en PRINT ØN.

Voir compléments sur Punch au §5.

Exemple : !IL
ITEM > EXEMPLE D^C Item 1 contient la valeur de l'item EXEMPLE
!KA
> W
?!ASSIGN B = CU1 D^C
?!GØ D^C

> P sauvegarde sur cartes de Item 1
>

5 - COMMANDE S - PUNCH SUR UL 'B (item à préciser).

Exemple 1 : !KA (l'assignation de 'B est faite)

```
> S
NØM > EXEMPLE D^C
```

tout ceci apparaît sur l'écran

```
KA
R
* + EXEMPLE ; KA
I
!IL
ITEM > EXEMPLE
!KA
>
```

Ceci est l'en-tête perforé sur cartes. Sera utilisé comme directives à GI lors d'une restauration sous KA.

A ce stade, l'en-tête étant perforé, on a le choix entre puncher ou non la valeur de l'item par la commande P.

Exemple 2 :

```
!KA
> S
NØM > EXEMPLE D^C
ERR - ASGN                    'B n'est pas assignée.
?!ASSIGN B = CU1 D^C
?!GØ D^C
> S
  :
  :
 etc...
```

Exemple 3 :

```
!KA
> S
NØM > EXEMPLE A D^C
```

cet item n'existe pas, les 4 cartes d'en-tête sont tout de même perforées.

```
KA
R
* + EXEMPLE A ; KA
I
!IL
ITEM > EXEMPLE A ?S
!KA
>
```

L'IL ne peut se faire car EXEMPLE A n'existe pas; on peut demander un punch par P et alors la valeur de l'item courant (Item 1) sera perforée.

On peut utiliser S et P comme dans l'exemple 3 pour faire une sauvegarde d'un item IA chargé préalablement en Item 1 avec 4 cartes d'en-tête contenant un nom différent IB que l'on pourra ensuite restaurer sous ce nom (il y a d'autres moyens plus simples de faire une duplication d'item avec changement de nom voir les commandes IL et IS de GE).

Attention ! Le punch (P ou S) est interruptible par Alt Mode ou par S^C. Mais la perfo peut se trouver en attente de caractères auquel cas on aura un décalage de perforation à la reprise. Donc, si l'on interrompt et si l'on redemande un punch, faire un CLEAR de la perfo avant la reprise.

6 - COMMANDE R - RESTAURATION A PARTIR DE L'UL 'A.

Deux cas sont possibles suivant qu'on restaure la valeur seule ou l'en-tête et la valeur.

6.1 - Restauration de la valeur seule (deck sans en-tête).

Exemple :

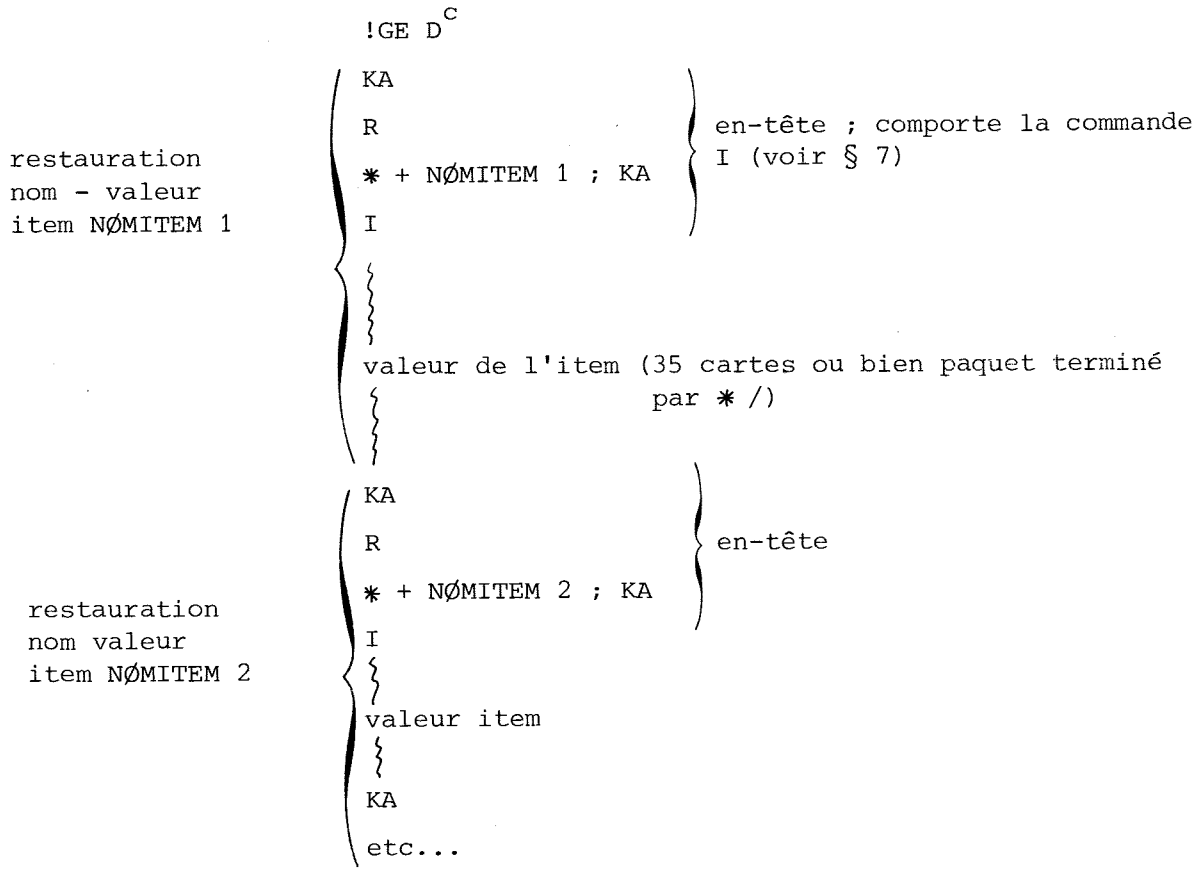
```
!I+
ITEM > EXEMPLE B D^C        Création nouveau nom, valeur
                            vide 'A ayant été assignée.
!KA
> R                         La valeur est chargée en Item 1
{                           édition de l'item (interruptible
{                           par S^C ou Alt-Mode)
{
> F                         sortir de KA
!IS                         Store item courant, nom et valeur
!
```

6.2 - Restauration de la valeur et du nom (deck avec en-tête).

Dans ce cas R fait à partir de 'A :

- le chargement en Item 1 de Nom item et Valeur item.
- le rangement de l'item courant sur disque (IS Item store).
- le retour à GE(!) ce qui permet d'enchaîner immédiatement la restauration suivante sans avoir besoin de sortir de KA par F, particulièrement utile en batch.

Exemple : flot de cartes pour restaurations successives d'items (nom et valeur) en batch.



7 - COMMANDE I - RESTAURATION A PARTIR DE 'A et STORE ITEM.

I assure la restauration du nom et/ou de la valeur, la sortie de KA et le Store de l'item restauré (fonction IS).

Elle est utilisée par R.

Elle peut être utilisée seule.

Elle n'assure pas le listage du deck de cartes sur le visuel contrairement à A.

Exemple 1 : Restauration valeur.

```
!I +
ITEM > EXEMPLE D^C
!KA
> I
!
```

Création nouveau nom, valeur vide 'A ayant été assignée la valeur est chargée en Item 1. Pas de liste. Pas de IS, ce n'est pas nécessaire.

Exemple 2 : Restauration nom et valeur.

!KA	'A ayant été assignée ; le deck comportant un en-tête. Ni I+ ni IS ne sont requis car ils sont assurés par I.
>I	
!	

8 - COMMANDE L - LISTE ITEM EN COURS SUR UL 'B.

C'est l'item en cours (donc contenu dans Item 1) qui est listé (valeur seule). Il faut donc l'avoir préalablement chargé.

- si 'B est assignée à l'imprimante, la liste sera précédée d'un saut de page.

- si 'B est assignée à un écran, la liste ne sera pas précédée d'un effacement écran.

PROCESSEUR GI

Le processeur GI est identique à GE aux différences suivantes près :

. les commandes sont placées dans Item 2 ;

. D^C, lorsqu'il est nécessaire, est remplacé par ";".

Exemple : sous ED on place dans Item 1 ceci : LIA ; LIB ; FF
on nomme cet item EXEMPLE.

On pourra l'utiliser par GI :

!IL	chargement de l'item EXEMPLE dans Item 1.
ITEM > EXEMPLE D^C	
!MX	l'item EXEMPLE est placé dans Item 2.
!GI	appel de GI qui va chercher ses commandes
!LI	dans Item 2, d'où la suite du dialogue qui se
RACINE > AD^C	déroule automatiquement sur l'écran.
!LI	
RACINE > B D^C	
!FF	
?	

De même que pour GE, on sort de GI :

- par FF pour revenir au CCI (?)
- par GE pour revenir à GE (!)

Remarques importantes :

1 - si on appelle un processeur XY sous GI, les commandes suivantes sont perdues, car XY écrase GI.

Exemple : item 2 ILDESSIN ; GR LIA ; etc....

DESSIN est chargé dans Item 1

GR est chargé, il écrase GI → Ces commandes sont perdues.

2 - Il est possible sous GI d'écraser Item 2 lui-même (par M2 ou Mx). Dans ce cas la poursuite de l'interprétation de Item 2 par GI donnera des résultats imprévisibles (à moins qu'on les ait soigneusement prévus !).

PROCESSEUR SE

1 - GENERALITES.

En utilisant le processeur SE on peut soit demander des informations sur l'item courant (placé en Item 1), soit affecter un type et une valeur à cet item :

Exemples :

```
!SE
TYPE > ?
ITEM > EXEMPLE
C : RETØUR CCI
PUIS EFFACEMENT ECRAN
!
```

Demande d'informations sur l'item courant
cf : §2 ci-dessous

```
!SE
TYPE > t
```

On veut affecter un type et une valeur
cf §3 ci-dessous

Nota : Sur les items de type C,G,M,S,W on peut demander l'effacement de l'écran après exploitation de l'item au cours d'un parcours de graphe (cf : les processeurs EI et GØ). Lors d'une opération d'affectation de type, le système demandera (si le type est C,G,M,S,W) :

EFFACEMENT >

et on répondra Ø (oui) ou N (non). Lors d'une opération de demande d'informations (?) le système indiquera s'il y a effacement ou non (comme dans l'exemple ci-dessus), et cette option sur l'effacement sera utilisée au cours des parcours du graphe dans lequel on intègrera ces items.

2 - TYPE > ? DEMANDE D'INFORMATIONS SUR L'ITEM COURANT.

S'il y a un item courant, le système donne toujours son nom et son type.

On obtient en outre d'autres informations en fonction du type de l'item :

- type S (séquence audiovisuelle) : on obtient l'adresse de début et l'adresse de fin sur la bande magnétoscope de la séquence.
- type W (pause) : il est indiqué si la durée est indéterminée (pause dite "asynchrone") ou déterminée (pause dite "synchrone") ; dans ce dernier cas, on obtient aussi la durée en minutes de la pause.
- types C,G,M,S,W : il est indiqué s'il y a effacement de l'écran ou non après exploitation de l'item.

3 - TYPE > t AFFECTATION DE TYPE ET DE VALEUR A L'ITEM COURANT.

Le dialogue avec SE dépend du type t.

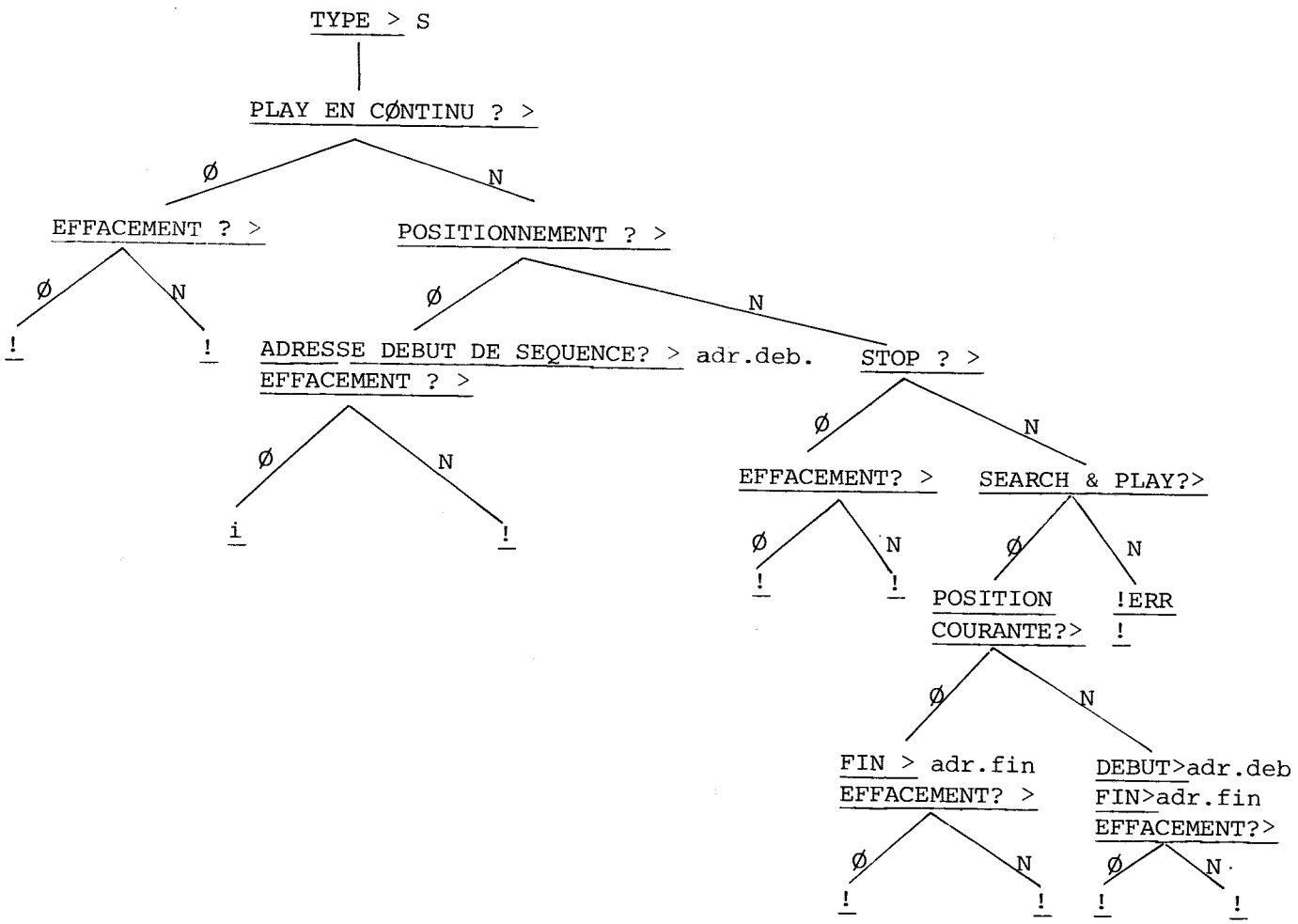
- *Type V* : affectation de valeur vide.
- *Type C* : retour CCI ; quand on passera sur l'item au cours d'un parcours de graphe (cf : les processeurs EI et GØ) celui-ci provoquera un retour au CCI (?), la poursuite du parcours pourra se faire par !GØ.
- *Type E* : type effacement écran. Lors d'un parcours, l'item provoquera l'effacement écran.
- *Type G* : retour à GE. Lors d'un parcours l'item provoquera le retour à GE, c'est-à-dire la fin de parcours du graphe. Il peut y avoir dans un graphe plusieurs points de sortie.
- *Type W* : pause. Deux types de pause : "Asynchrone" (A) - durée indéterminée ; "Synchrone" (S) : durée déterminée (de 1 à 99 minutes, servie en décimal).

Exemple :

```
!SE
TYPE > W
TYPE - PAUSE > S
DURE (MN) > 25
EFFACEMENT ? > Ø
!SE
TYPE > ?
ITEM > EXEMPLE
W : PAUSE
PAUSE 25 MINUTES
PUIS EFFACEMENT ECRAN
!
```

- *Type S* : séquence audiovisuelle. Le dialogue se déroule suivant l'arbre présenté page suivante.

SE 3


TYPE > S
PLAY EN CØNTINU ? >
Ø
N
EFFACEMENT ? >
POSITIONNEMENT ? >
Ø
N
Ø
N
!
!
ADRESSE DEBUT DE SEQUENCE? > adr.deb.
EFFACEMENT ? >
STOP ? >
Ø
N
Ø
N
i
!
EFFACEMENT? >
SEARCH & PLAY?>
Ø
N
Ø
N
!
!
POSITION COURANTE?>
!ERR !
Ø
N
FIN > adr.fin
EFFACEMENT? >
DEBUT>adr.deb
FIN>adr.fin
EFFACEMENT?>
Ø
N
Ø
N
!
!
!
!

Définitions :

- bandes magnétoscopes : elles comportent trois pistes : une piste image, une piste son n°1 utilisée pour le son ; une piste son n°2 utilisée comme piste d'adressage, des adresses pouvant y être lues ou écrites par l'ordinateur.

- adresse séquence magnétoscope : de la forme a b c d avec $a,b,c \in [0,9]$, $d \in [0,3]$.

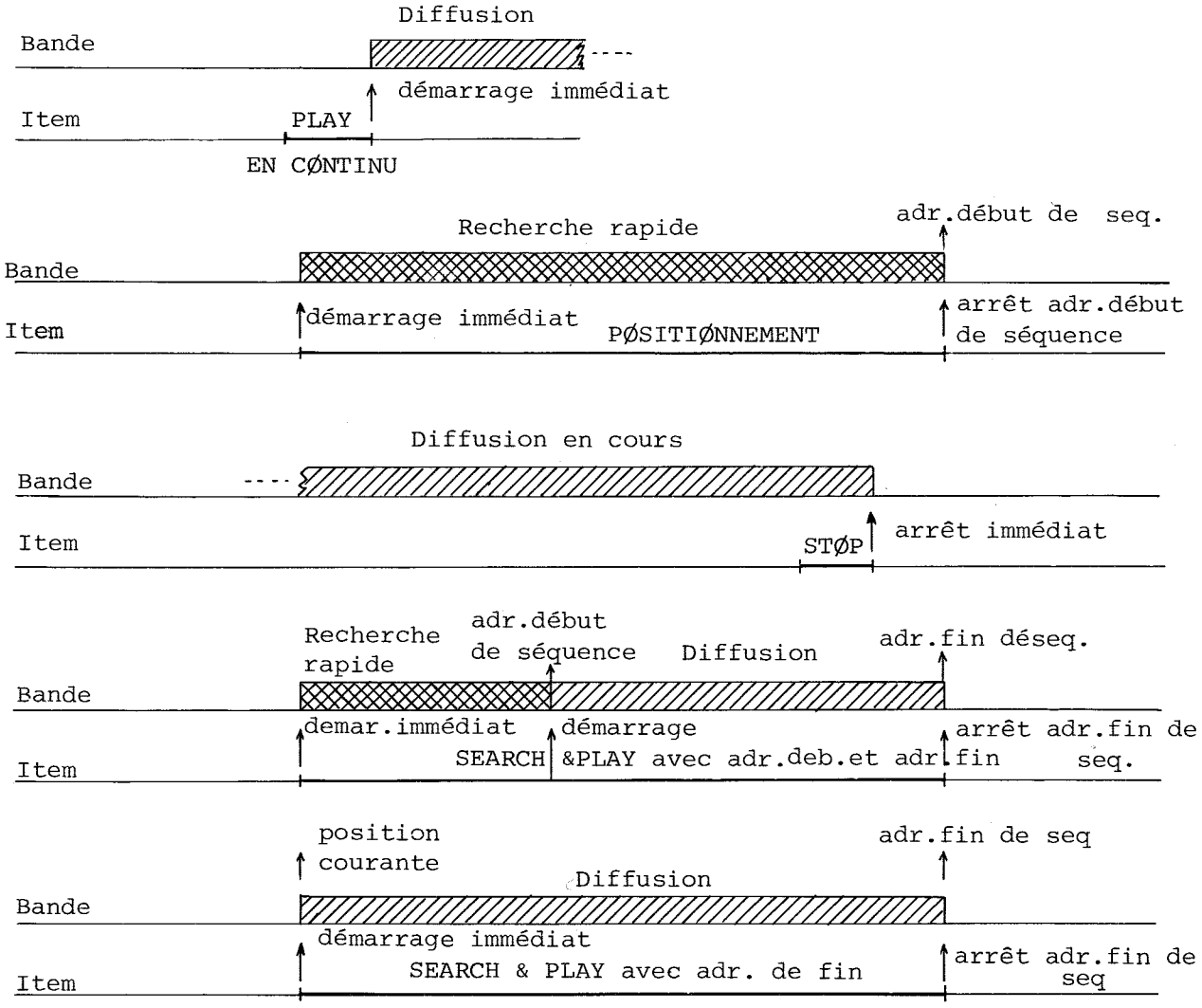
- play en continu : le magnétoscope est mis en diffusion à partir de sa position courante.

- positionnement : se place à une adresse donnée, sans diffusion puis arrêt. L'adresse en question devient la position courante.

- stop : le magnétoscope est arrêté, alors qu'il était en diffusion.

- scratch and play : on se positionne en début de séquence et on diffuse jusqu'à la fin de séquence ; l'adresse de début de séquence pouvant être la position courante (cf : l'arbre ci-dessus).

Représentation schématique :

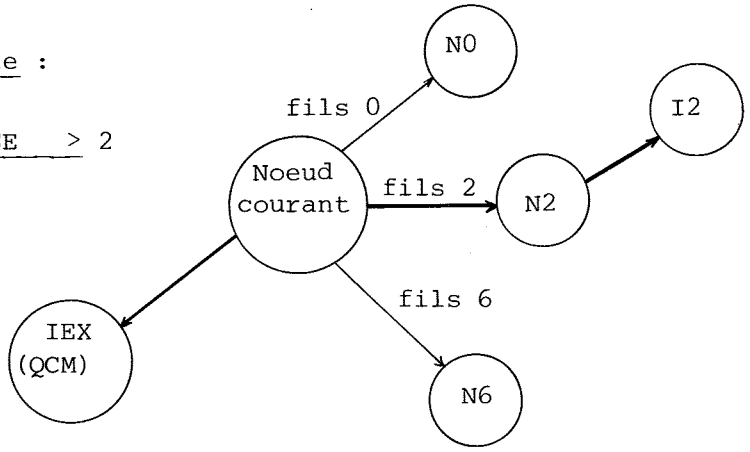


■ *Type M* : Question à choix multiples.

Lors d'un parcours de graphe, un tel item sera exploité de la manière suivante : le système demande REPØNSE > ; il faut lui répondre par un chiffre de 0 à 7, cette valeur désignant le n° de noeud fils du noeud courant. Si ce noeud fils existe, le système passe à ce noeud ; sinon la question est reposée.

exemple :

REPONSE > 2



Lors du parcours :
On passe de l'item courant IEX à I2 en répondant 2 à la question posée.
En mémoire, on trouvera donc en Item 1, IEX puis I2.

■ *Type R* : Question avec réponse en français libre.

On opère sur Item 1 et Item 2. On place dans Item 2 le texte d'une forme possible de la réponse i avec $0 \leqslant i \leqslant 7$; et dans Item 1 le système placera les valeurs sémantiques des réponses 0 à 7. S'il n'y a pas encore de réponse dans Item 1, il s'agira d'une initialisation.

Valeur de Item 1

réponse 0
réponse 1
réponse 2
réponse 3
réponse 4
réponse 5
réponse 6
réponse 7

(valeur sémantique des réponses).

exemple :

```
!SE
TYPE   > R
INITIALISATIØN ? > Ø
NUMERØ = 0
!
```

La valeur sémantique de la phrase contenue en Item 2 devient la réponse n°0.
Item 1 contient la réponse 0.
Item 2 est écrasé.

```
{
!SE
TYPE    > R
INITIALISATIØN ?   N
NUMERØ = 3
!
```

La valeur sémantique de la phrase contenue en Item 2 devient la réponse n°3.

Item 1 contient les réponses 0 et 3.

Nota : Les n°s de réponses ont les mêmes effets que pour le type M.

■ *Types spéciaux 0 et F.*

- type 0 : Remise à zéro des deux premiers octets de la ZDC, les octets de synchronisation inter-programmes.

- type F : Retour à GE (!), les items 1 et 2 restant intacts (notamment leur type).

■ *Type D* : programme graphique.

Ce type n'est accepté que si l'item placé dans Item 1 le possède déjà. SE demande quel processeur graphique il faudra utiliser pour exploiter cet item lors d'un parcours de graphe (cf : processeurs EI/GØ). Les processeurs graphiques qu'on peut spécifier sont G2,G3,GR,GT,GV,GW.

Exemple :

```
!SE
TYPE > D        n'est accepté que si l'item de Item 1
G T             est de type D.
!SE
TYPE > ?
ITEM > DESSIN 1
D : DESSIN
GT
!IS              stockage de l'item.
```

PROCESSEUR DI

1 - GENERALITES.

- *Fonction.*

Gestion d'un dictionnaire de mots.

- *Dictionnaire de mots.*

Il se compose de pages, chaque page contenant des mots. Chaque page a un nom et une valeur dans l'espace des noms-valeurs. Le nom d'une page est de la forme :XY avec :

- XY = ØØ pour la page contenant les noms usuels (:ØØ)

- XY ≠ ØØ " la page contenant les noms non-usuels de racine XY.

Exemple : On rangera le mot LA dans la page :ØØ.
On rangera le mot LACTAMME dans la page :LA.

Remarque : La commande LI de GE donnera, entre autres, toutes les pages du dictionnaire si l'on sert RACINE > :D^C

2 - UTILISATION.

! DI

MØT > chaîne de caractères de longueur ⩽ 39 terminé par D^C (voir ci-après §Mot)

USUEL ?
- F ⟶ retour à GE (!)
- C ⟶ retour au CCI (?)
- Ø ⟶ mot usuel
- N ⟶ mot non-usuel

INDICATEURS
- Ø ⟶ le mot a une fonction grammaticale
- N ⟶ le mot n'a pas de fonction grammaticale

PRØ.REL ?
ARTICLE ?
CØNJØNCTIØN ?
PREPØSITIØN ?
INTERJECTION ?
PRØ.DEM ?
PRØ.PERS ?
ADVERBE ?
ADJECTIF ?
VERBE ?
NØM ?

N'apparaît que si l'on a répondu par oui à la question précédente (Indicateurs).

A chacune de ces questions on répondra par oui ou non (Ø ou N).

VAL.SEMANTIQUE = voir ci-après, § Valeur sémantique.

Le processeur DI écrase Item 2 et y place à destination de GI les arguments suivants : I + { :ØØ / :XY } ; IL { :ØØ / :XY } ; ED ainsi que la description du mot, et GI prend la main.

! I+

ITEM > { :ØØ / :XY }

! IL

ITEM > { :ØØ / :XY }

la page :ØØ ou :XY est créée si elle n'existe pas et chargée dans Item 1.

! ED

On est sous ED et l'on doit placer dans Item 1 (donc dans la page à laquelle appartient le mot) la description du mot placée par DI en Item 2, puis storer l'item. Donc faire :

1°) B^C on travaille dans Item 2.

2°) E^C affichage Item en cours (Item 2).

3°) $Ø^C$ désigne l'origine de la chaîne à déplacer (cette chaîne étant la description du mot).

4°) Q^C désigne la fin (queue) de cette chaîne.

5°) A^C on travaille dans Item 1.

6°) E^C affichage Item en cours (Item 1).

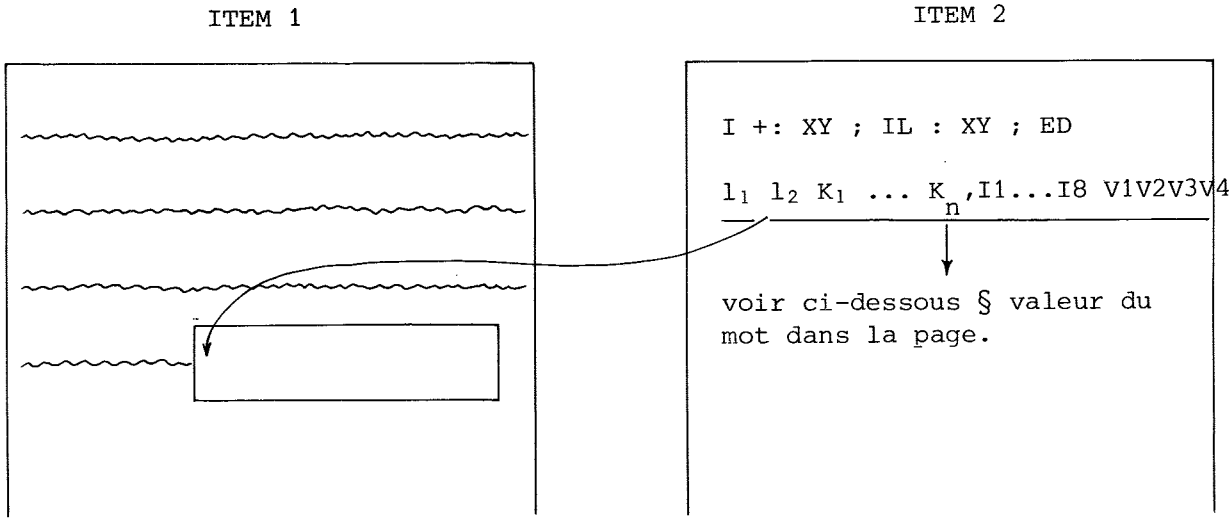
7°) $Ø^{CS}$ désignation nouvelle origine de la chaîne (destination) ; la chaîne est déplacée.

8°) E^C affichage de l'item courant (Item 1), vérifier que le mouvement s'est bien passé.

9°) F^C sortir de ED.

10°) IS Item Store : Item 1, la page { :ØØ / :XY } mise à jour est rangée en bibliothèque.

Nota : bien entendu, ces opérations supposent des positionnements du curseur adéquats (voir processeur ED).



- *Mot.*

MØT > chaîne de caractères de longueur ⩽ 39 caractères terminée par D^C

les caractères sont quelconques (alphabétiques, numériques, caractères spéciaux, expression arithmétique, etc...)

on peut rentrer des caractères "blanc" mais il faut savoir que le blanc est pris comme délimiteur de mot lors de l'analyse syntaxique (voir processeurs PH et ØD). Il vaut donc mieux utiliser par exemple des tirets plutôt que des blancs.

un mot vide provoque le retour à GE (!)

caractères particuliers :

1°) Caractère "." Les caractères qui suivent le point dans le mot seront ignorés lors d'une recherche dans le dictionnaire ; le caractère qui précède le point devra être le dernier caractère du mot recherché pour qu'il y ait identification :

Exemple : mot du dictionnaire : ABC.DEF
mot recherché : ABC
} identification

2°) Caractère "#" Dans le mot, signifie que n'importe quel caractère peut se trouver à cette place dans le mot recherché.

Exemple : mot du dictionnaire ABC # EF
mot recherché { ABCDEF, ABCHEF, ... } identification

mot du dictionnaire DESSINE #
mot recherché { DESSINER, DESSINES, ... } identification

3°) Caractère ":" Les caractères qui suivent seront ignorés et dans la recherche dans le dictionnaire, et dans le mot recherché (noter la différence avec le caractère ".").

Exemple : mot du dictionnaire : ABC : EF
mot recherché : ABC123
} identification

mot du dictionnaire : CHANTE : X
mot recherché : CHANTES, CHANTER
} idendification.

Noter l'utilité pour "prévoir" certaines fautes d'orthographe comme par exemple un impératif de verbe en ER écrit avec un S (CHANTES au lieu de CHANTE), ou encore le S de pluriel, etc...

4°) Caractère en exposant : si un caractère est mis en exposant (ou interligne), cela signifie qu'il est facultatif.

Exemple : mot du dictionnaire CHAT^{TE}
mot recherché { CHAT, CHATTE } identification

(cela permet d'appeler un chat un chat.....ou une chatte !)

■ *Valeur sémantique.*

Chaîne de caractères de longueur 0 (chaîne vide) à 4 suivie de D^C; à noter qu'un mot ne peut avoir qu'une seule valeur sémantique (monosémie). Actuellement, deux formes de valeurs sémantiques sont reconnues :

1°) ! { Commande de GE (I+ I- IL, etc...) ; XY (processeur appelé sous GE) } voir analyse processeurs PH et ØD

2°) & [K1] [K2] ;

■ *Valeur d'un mot dans la page* : elle est de la forme :

$l_1 l_2$ $K_1 K_2 \ldots K_n$, $I_1 I_2 \ldots I_8$ $V_1 V_2 V_3 V_4$

$l_1 l_2$: longueur du mot en décimal codé asci.

$K_1 K_2 \ldots K_n$: mot

$I_1 I_2 \ldots I_8$: catégories grammaticales

$V_1 V_2 V_3 V_4$: Valeur sémantique

PROCESSEUR PH

1 - GENERALITES.

1.1 - Fonction.

Le processeur PH demande une phrase en français "libre". Cette phrase ayant été introduite, elle est traitée en trois étapes :

1°) Analyse lexicographique

2°) Analyse syntaxique réalisée par l'un des processeurs ØX ou ØD

3°) Analyse sémantique réalisée par le processeur ØC.

1.2 - Appel : deux types d'appels.

1°) Explicite : par !PH

2°) Implicite : lors d'un parcours de graphe (voir EI/GØ).

1.3 - Choix du processeur d'analyse lexicographique et syntaxique.

On a le choix entre ØX ou ØD. ØD réalise les mêmes fonctions que ØX avec en plus l'édition du résultat de l'analyse syntaxique (cf : paragraphes suivants).

Pour choisir le processeur, utiliser M de GE (sous *) faire (avec ACN =:SYS) :

!GE	
*M	PH appellera ØX
NØM > ØX	donc il suffit de donner ce nom au processeur ØD en invalidant ØX ou inversement, de donner ce nom au processeur ØX en invalidant ØD.
NØM > ØX	
NØM > XX	
*M	
NØM > ØD	
NØM > ØD	
NØM > ØX	
*	

1.3 - Remarques.

Item 1 et Item 2 sont détruits.

1.4 - Retour.

Répondre à PH une phrase vide

```
!PH
PHRASE > D^C
!
```

2 - FONCTIONNEMENT.

2.1 - Exemple.

```
!PH
PHRASE > TU DESSINES LE CARRE D^C
```

```
[ ??? mot
  ??? mot
    .
    .
    .
    .  ]
```

PH fait une analyse lexicographique. Il sort éventuellement les mots non-trouvés au dictionnaire. Il considère ∅ et caractères usuels de ponctuation comme des séparateurs. Il associe à chaque mot sa fonction grammaticale et sa valeur sémantique (cf : processeur DI, création des mots). PH appelle ensuite ∅D ou ∅X pour l'analyse syntaxique. Si c'est ∅D, on obtient le résultat de l'analyse.

```
CORRECT
> TU /            = PRO.PERS
> DESSINES / !GR  = VERBE
> LE /            = ARTICLE
> CARRE / &C ;    = NOM
1 PROPOSITIONS
```

Analyse syntaxique.
Chaque mot apparaît avec sa valeur sémantique (éventuellement vide) et sa fonction grammaticale. Chaque mot est précédé de n fois " > " indiquant le niveau de relativité de la proposition à laquelle il appartient. Ensuite est affiché le nombre de propositions reconnues.

```
* /TU/DESSINES/LA CARRE
```

Analyse sémantique :
1°) Enumération de toutes les propositions reconnues. Chacune d'elle est précédée de n fois "* " indiquant son niveau de relativité dans la phrase.

```
< R > TU/DESSINES/LA CARRE/
```

2°) Réécriture de toutes les propositions: chacune est réécrite sous la forme < sujet > < verbe > [< attribut >]

A ce niveau de l'analyse les < verbe > sont considérés comme des opérateurs, et les < sujets > et < attributs > comme des opérandes. Dans cet exemple, la valeur sémantique du verbe < Dessine > étant "!GR" et la valeur sémantique de l'attribut < carré > étant "&C;", le processeur GR va être lancé avec comme programme graphique en entrée l'appel du sous-programme C (cf : le langage graphique). On va donc obtenir un carré

→ chargement dans item 1 de : ∅BIBS ; &C ;

!GR → appel de GR

→ GR exécute le programme graphique

! → retour à GE

Cela suppose que le sous-programme C existe. Il faut donc l'avoir défini précédemment et l'avoir placé dans la bibliothèque BIBS. (voir exemples §4).

Sur le traitement lexicographique, l'analyse syntaxique, l'analyse sémantique et la réécriture des propositions, voir ci-après.

3 - TRAITEMENT LEXICOGRAPHIQUE.

La phrase φ en entrée est une chaîne de caractères.

Elle est décomposée en mots successifs par utilisation de séparateurs (espace, caractères usuels de ponctuation).

L'existence de chaque mot est contrôlée à l'aide du dictionnaire de mots (cf : processeur DI) qui permet d'obtenir pour chaque mot existant :

. sa valeur sémantique (éventuellement vide)

. ses fonctions grammaticales possibles (nom, verbe, articles etc...)

Lorsqu'un mot n'existe pas au dictionnaire, on lui associe une valeur sémantique vide et l'ambiguité fonctionnelle maximale (nom + verbe + article ... etc ....).

$\varphi \rightarrow \varphi 1$

φ = < chaîne de caractères >

φ = < chaîne de (< mot > < fonction(s) grammaticales > < signification >) >

Nota : . Certains mots sont reconnus sans consultation du dictionnaire (pronoms relatifs : QUI, QUE, DONT ; etc...) car ils appartiennent à la grammaire.

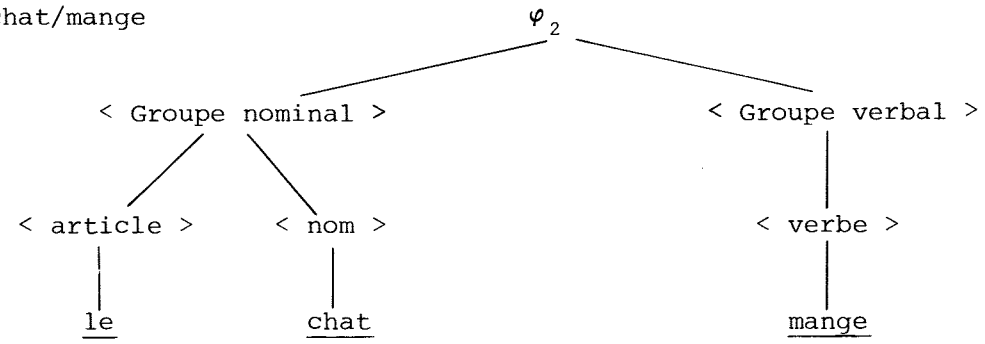
4 - ANALYSE SYNTAXIQUE.

La phrase $\varphi 1$ est analysée en fonction d'une grammaire asserto-négative, et elle aboutit à la construction d'un arbre $\varphi 2$ représentant syntaxiquement $\varphi 1$, à chacun des mots étant associée sa fonction réelle dans la phrase (le mot n'a qu'une fonction dans la phrase alors qu'au dictionnaire il pouvait avoir plusieurs fonctions possibles), ainsi que son niveau (qui est celui de la proposition à laquelle il appartient).

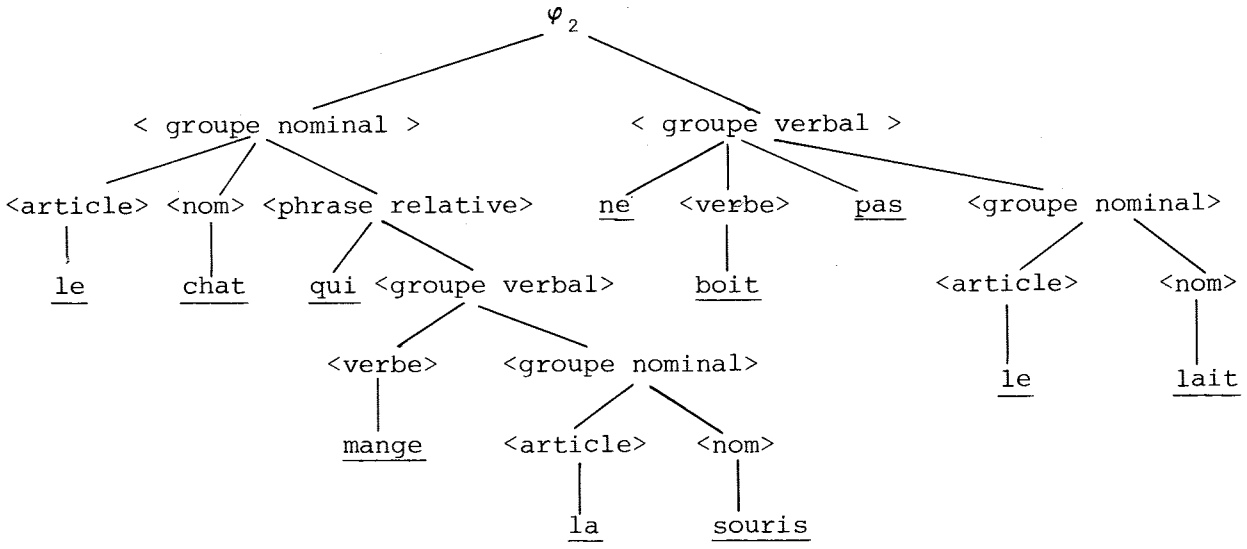
4.1 - Grammaire : voir page suivante.

4.2 - Exemples.

φ = le chat mange
φ_2= le /chat/mange



φ = le chat qui mange la souris ne boit pas le lait
φ_2 = le/chat/qui/mange/la/souris/ne/boit/pas / le/lait



La grammaire :

< phrase > :: = < phrase complète > / < phrase interjective >

< phrase interjective > :: = < interjection >

< phrase complète > :: = < groupe nominal > /
< groupe nominal > < groupe verbal >

< groupe nominal > :: = < article > [[< adverbe >$_n$] < adjectif >$_n$]
<nom> [[<adverbe>$_n$] <adjectif>$_n$] [<phrase relative>] /
< pronom personnel > [< phrase relative >]

< phrase relative > :: = { 'DONT' / 'QUE' } < groupe nominal > < groupe verbal > /
{ 'QUE' / 'QUI' } < groupe verbal >

< groupe verbal > :: = < verbe > [< adverbe >$_n$] [< groupe nominal >] /
< verbe > [< adverbe >$_n$] 'DE' < groupe nominal > /
'NE' < verbe > 'PAS' [< adverbe >$_n$] [< groupe nominal]
'NE' <verbe> 'PAS' [<adverbe>$_n$] 'DE' < groupe nominal >
'NE' <verbe> <groupe nominal> 'QUE' <groupe nominal>

4 - ANALYSE SEMANTIQUE ET REECRITURE.

De l'arbre φ_2 obtenu on extrait toutes les propositions $\varphi_{2\,i}$ (i=1,n) ainsi que leurs dépendances relatives. Chacune d'elles est ensuite réécrite sous forme de < sujet > < verbe > < attribut >, comme ci-dessous :

< N1 > < V > < N2 > ⟶ < N1 > < V > < N2 >

< N1 > < QUI > < V > < N2 > ⟶ < N1 > < V > < N2 >

< N1 > < QUE > < V > < N2 > ⟶ < N2 > < V > < N1 >

< N1 > < QUE > < N2 > < V > ⟶ < N2 > < V > < N1 >

pour obtenir un ensemble de propositions φ_{3j} (j=1,n) assimilé à un programme arborescent φ_4 écrit dans un langage dans lequel < sujet > et < attribut > sont des opérandes et < verbe > les opérateurs, la forme arborescente de φ_4 venant des relations de dépendance entre les propositions φ_{3j}. Ces opérandes et ces opérateurs sont les valeurs sémantiques associées aux mots.

Ce programme est ensuite exécuté.

Exemples : voir pages suivantes.

Exemple d'analyse de phrase par le processeur PH :

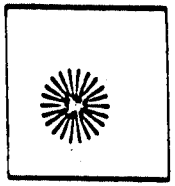
```
!PH
PHRASE)LE CHAT QUI MANGE LA SOURIS QUI MANGE LE FROMAGE NE BOIT PAS LE BON LAIT QUE NE BOIT PAS LE RAT QUI DORT
??? FROMAGE
??? BOIT
??? BON
??? LAIT
??? BOIT                    < 7 MOTS INCONNUS
??? RAT
??? DORT
CORRECT
)  LE /       •ARTICLE
)  CHAT /     •NOM
))  QUI /
))  MANGE /     •VERBE
))  LA /      •ARTICLE
))  SOURIS /    •NOM
)))  QUI /
)))  MANGE /    •VERBE
)))  LE /     •ARTICLE
)))  FROMAGE / •NOM
)  NE /
)  BOIT / •VERBE                                    < ANALYSE SYNTAXIQUE
)  PAS / -
)  LE /       •ARTICLE
)  BON / •ADJECTIF
)  LAIT / •NOM
))  QUE /
))  NE /
))  BOIT / •VERBE
))  PAS / -
))  LE /      •ARTICLE
))  RAT / •NOM
)))  QUI /
)))  DORT / •VERBE                          R
5 PROPOSITIONS                                E
I   / LE CHAT / NE BOIT PAS / LE BON LAIT       E
II   / LE BON LAIT / QUE / NE BOIT PAS / LE RAT     C
III   / LE RAT / QUI / DORT                           R
II   / LE CHAT / QUI / MANGE / LA SOURIS                I
III   / LA SOURIS / QUI / MANGE / LE FROMAGE              T
(R) LE CHAT  / NE BOIT PAS  / LE BON LAIT  /                U
(R) LE RAT  / NE BOIT PAS  / LE BON LAIT  /                   R
(R) LE RAT  / DORT  /                                           E
(R) LE CHAT  / MANGE  / LA SOURIS  /
(R) LA SOURIS  / MANGE  / LE FROMAGE  /
?
```

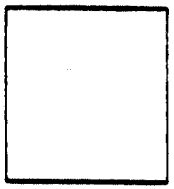
< ANALYSE SEMANTIQUE

Analyse de phrases et exécution des actions demandées.

①

```
!PH
PHRASE>TU DESSINES LE CARRE QUI DESSINE LA STAR
CORRECT
>  TU /       =PRO.PER
>  DESSINES / !GR =VERBE
>  LE /       =ARTICLE
}  CARRE / &CC;=NOM
>>  QUI /
>>  DESSINE / !GR =VERBE
>>  LA /       =ARTICLE
>>  STAR / &ST;=NOM
2 PROPOSITIONS
*   / TU / DESSINES / LE CARRE
**   / LE CARRE / QUI / DESSINE / LA STAR
<R> TU  / DESSINES  / LE CARRE  /
<R> LE CARRE  / DESSINE  / LA STAR  /
!GR
```





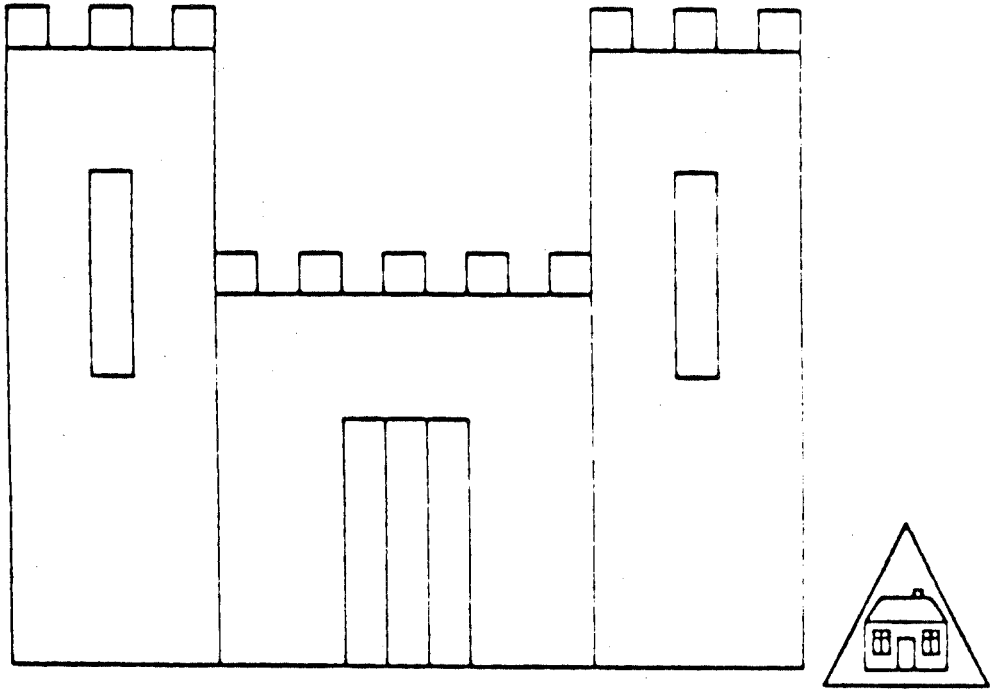
②

```
PHRASE>TU LISTES A
??? A
CORRECT
>  TU /       =PRO.PER
>  LISTES / !LI =VERBE
>  A / =ADVERBE
1 PROPOSITIONS
*   / TU / LISTES A
<R> TU  / LISTES A  /
!LI
RACINE>A
-ALPHA
-AX
!GE
!
```

① { Item 1 = $BIBS,&CC,,&CC,,&ST,,F
  { Item 2 contient la bibliothèque BIBS

Exécution des actions demandées dans une phrase

!PH
2 FAUTES D'ORTHOGRAPHE CORRIGEES
PHRASE>TU DESSINNE UN GRAND CHATEAU ET, TU DESSINES A-DROITE UN PETIT MAIS
2 FAUTES D'ORTHOGRAPHE CORRIGEES
ON ET UN GRAND TRIANGLE ET,TU PHOTOCOPIE L'ECRAN
!G3
1 FAUTE D'ORTHOGRAPHE CORRIGEE



PROCESSEUR RE

1 - GENERALITES.

■ *Fonction.*

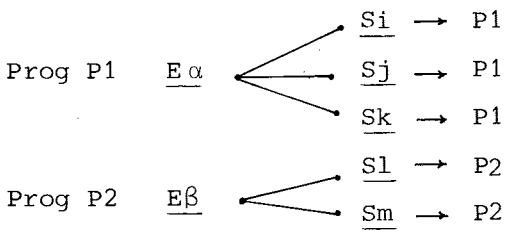
Commande de la baie de commutation Vidéo et télécommande explicite des magnétoscopes (explicite : par opposition à la télécommande avec recherche de séquences ; voir les processeurs VI et EI/GO).

■ *Habilitation.*

RE est réservé à l'ACN : SYS

■ *Baie de commutation.*

C'est une matrice de commutation à 20 entrées et 7 sorties.
Plusieurs connexions peuvent exister en même temps dans la baie.
Une même entrée peut être raccordée à plusieurs sorties.
Une sortie ne peut recevoir qu'une entrée.



■ *Etat des connexions.*

Cet état est mémorisé dans une table commune à l'ensemble des utilisateurs. Cette table est rangée dans l'espace des noms et valeurs sous le nom :CØMØ^{CS}. L'état des connexions est modifiable par RE (commandes C, D, Z) et on peut le lister sous RE (commande L).

■ *Assignations.*

Les assignations des UL 'A et 'B sont détruites par RE qui établit les assignations suivantes :

!ASSIGN A = CU2 ⟶ raccordé à un bus magnétoscope

!ASSIGN B = CU3 ⟶ raccordé à la baie de commutation

2 - APPEL DE PE - SORTIE DE RE.

. appel : RE — !RE

. F Retour à GE (!) — >F

. W Retour au CCI (?) — !RE / >W / ?!GØ D^C / >

3 - COMMANDES DE TELECOMMANDE DE LA BAIE.

. N° de voie — Un n° de voie s'exprime en décimal.

On a $\begin{cases} 0 \leqslant n_1 \leqslant 19 & \text{si } n_1 \text{ désigne une entrée} \\ 0 \leqslant n_2 \leqslant 7 & \text{si } n_2 \text{ désigne une sortie} \end{cases}$

. C Etablissement d'une connexion.

> C IN = n_1 ØUT = n_2 — établit la connexion de l'entrée n_1 à la sortie n_2 après avoir automatiquement déconnecté la sortie n_2 (Commande D implicite de n_2, voir commande D ci-après).

. D Déconnexion d'une sortie.

> D ØUT = n_2 — déconnecte la connexion vers la sortie n_2

. Z Déconnexion de toutes les sorties. (Z ≡ { D })

> Z

. L Liste de l'état des connexions.

```
> Z
> C  IN = 08   ØUT = 2
> C  IN = 03   ØUT = 3
> C  IN = 04   ØUT = 2   (08 ⟶ 2   devient 04 ⟶ 2)
> C  IN = 05   ØUT = 7
> C  IN = 03   ØUT = 0
> L

05 --- > 7
03 --- > 3
04 --- > 2
03 --- > 0
>
```

4 - COMMANDE DE TELECOMMANDE DES MAGNETOSCOPES.

. N° de magnétoscope : Un n° de magnétoscope, n, s'exprime en décimal, avec 00 ⩽ n ⩽ 08. Le servir sur 2 caractères.

. Commande globale des magnétoscopes : Pour transmettre un ordre s'appliquant globalement à tous les magnétoscopes, servir n = D^C. Ceci n'est appliquable que pour les commandes P,R,F,S,E, et n'est pas applicable pour la commande @.

. M Commande magnétoscope :

> MAGNETØ = { n / D^C } ØRDRE = { P / R / F / S / E / @ } ⟶ pas d'ordre global

avec :

P : Play (diffusion)
R : Rewind (rembobinage rapide)
F : Forward (marche avant rapide)
S : Stop
E : Enregistrement
@ : Lecture de la piste SON 2 du magnétoscope n et affichage des adresses obtenues sur la visu ; l'affichage est interruptible par Alt-Mode (on revient à >). @ peut s'utiliser lorsque le magnétoscope est en Play, Rewind, Forward ou Enregistrement, donc quelles que soient la vitesse et le sens de fonctionnement.

Exemple :

```
≀
> MAGNETØ = 03 ØRDRE = P
> MAGNETØ = 03 ORDRE = @
0201
0201
0202
0202
0204
 .
 .
 .
Alt-Mode
> MAGNETØ = 03  ØRDRE = S
```

affichage toutes les secondes de l'adresse courante

Le magnétoscope 03 est en play

⟶ stop magnétoscope 03.

PROCESSEURS EI/GØ

1 - GENERALITES.

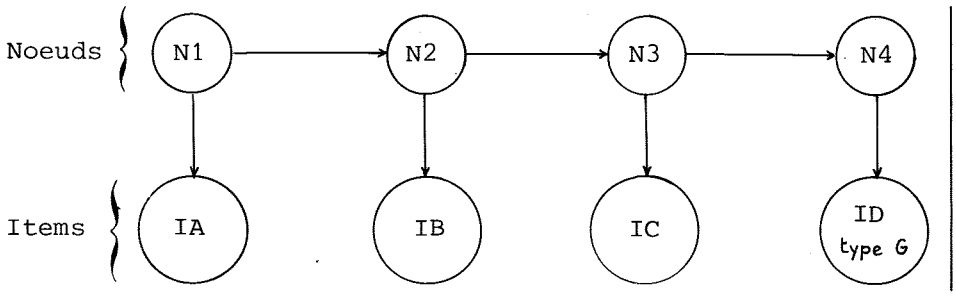
Le processeur GØ est appelé explicitement sous GE(!) pour initialiser le parcours d'un graphe. Ce parcours est ensuite poursuivi automatiquement par le processeur EI appelé implicitement.

On se reportera pour étudier l'exemple donné ci-dessous à la présentation générale de SMC (schéma de SMC, items, noeuds)

Exemple : cf page suivante.

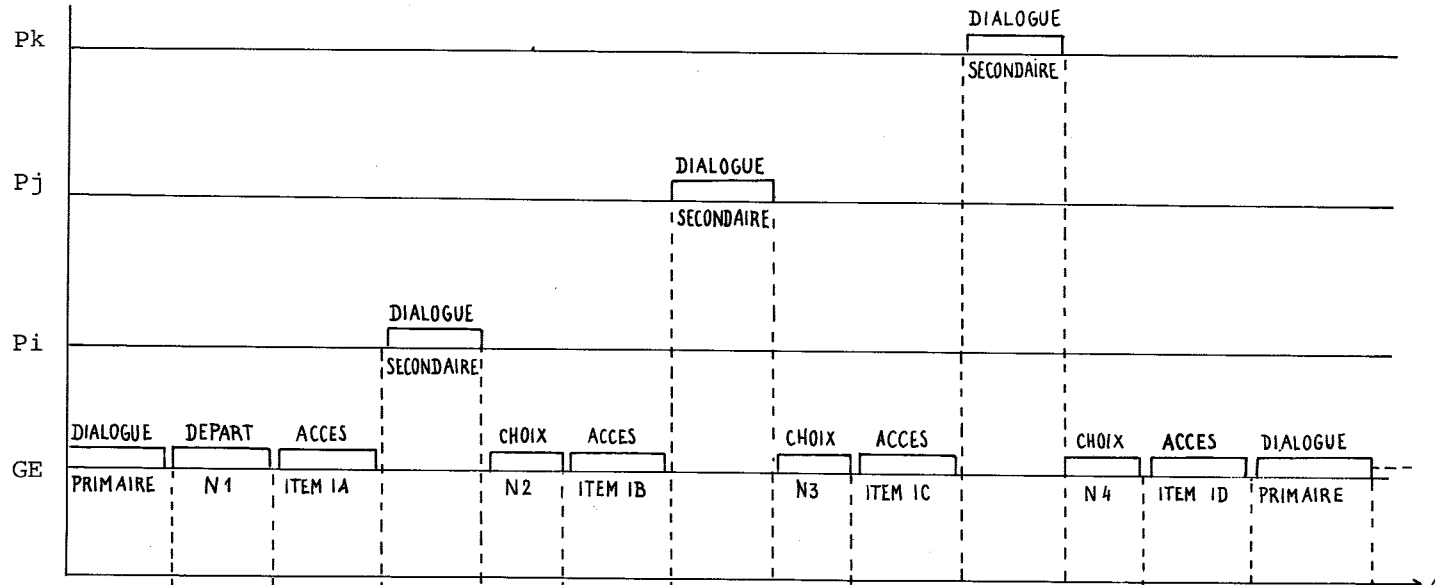
Exemple d'utilisation de EI/GØ.

Soit le graphe g :

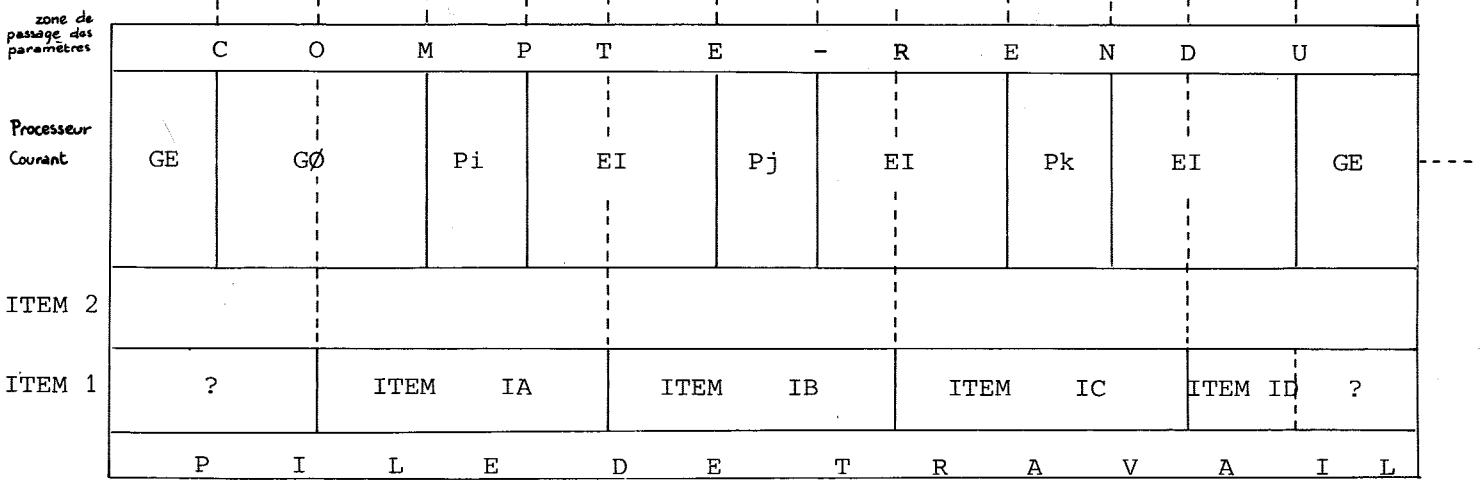


IA est exploité par le processeur Pi
IB par Pj
IC par Pk
Type G = fin de graphe

L'exploitation de g conduira au diagramme temporel suivant :



et aux états successifs de l'espace mémoire SMC du demandeur représentés ci-dessous.



Nota : le contenu de Item 2 est fonction de l'état initial et des besoins du processeur en cours.

2 - PROCESSEUR GØ.

GØ initialise le parcours du graphe. Il demande le noeud origine. Tout noeud d'un graphe peut être donné à GØ comme noeud origine.

Exemple 1 : !GØ

ØRIGINE > NEXEMPLE D^C

parcours automatique (par EI)

Exemple 2 : !GØ

ØRIGINE > NN1 D^C Le noeud n'existe pas (par exemple erreur de frappe)

!ERN

?!GØ D^C retour à GE par !GØ D^C sous ?

!GØ rappel du processeur GØ

ØRIGINE > NM1 D^C Le noeud NM1 existe

parcours automatique (par EI)

Nota : on peut aussi revenir de cette erreur par !Return.

!ERN

?!Return

ØRIGINE > NM1 D^C

3 - PROCESSEUR EI.

EI assure le parcours du graphe. Il exploite le noeud courant.

- Si le noeud n'a pas d'item associé, EI demande quel item on désire lui associer :

!ITEM DESIRE =

- D^C ou Return → EI fait comme si un item vide était rattaché au noeud (voir ci-dessous exploitation item type V)
- nom item D^C → EI fait comme si cet item était rattaché au noeud et l'exploite en conséquence. Si cet item est inconnu, EI boucle sur ITEM DESIRE.

- Si le noeud courant est associé à un item, EI exploite cet item en fonction de son type : il appelle le processeur adéquat.

■ *type = C retour au CCI(?)* : On reprend le parcours par !GØ D^C; EI passe au fils 0 du noeud courant après avoir fait l'effacement de l'écran si celui-ci est demandé par l'item courant. EI affiche le temps passé entre la suspension du parcours (rencontre de l'item de type C) et sa reprise (!GØ D^C).

■ *type = D programme graphique* : EI appelle le processeur Gx avec $x \in \{ 2, 3, R, T, V, W \}$ avec x = 3 implicite (cf processeur SE type D).

Quand l'interprétation de l'item par le processeur graphique Gx est terminée, EI reprend le parcours en passant :

- au fils 0 du noeud courant si Gx a terminé son travail normalement
- au fils 2 du noeud courant si Gx a été interrompu par Alt-Mode ou par S^C.

■ *type = E effacement écran* : EI assure l'effacement de l'écran et passe au fils 0 du noeud courant.

■ *type = G fin de graphe* : EI revient à GE(!) après effacement éventuel si celui-ci est demandé dans l'item (cf : processeur SE type G).

■ *type = I film digital* : EI appelle le processeur SC, et quand l'interprétation de l'item en cours par le processeur SC est terminée, EI reprend le parcours en passant :

- au fils 0 du noeud en cours si SC a terminé normalement son travail.
- au fils 2 du noeud en cours si SC a été interrompu par Alt-Mode ou par S^C.

■ *type = M QCM (question à choix multiples)* : EI demande une réponse ; répondre n avec $0 \leqslant n \leqslant 7$; EI passe alors au fils n du noeud en cours si celui-ci existe, sinon il redemande une réponse. Le passage au fils n du noeud en cours est précédé d'un effacement de l'écran si celui-ci est demandé dans l'item (cf : SE type M).

Nota : si l'on répond n avec $n \notin [0,7]$, EI redemande une réponse.

■ *type = P Programme en langage machine* : EI appelle le processeur 3D et quand l'interprétation de l'item en cours par 3D est terminée EI reprend le parcours en passant :

- au fils 0 du noeud en cours si 3D a terminé normalement son travail
- au fils 2 du noeud en cours si 3D a été interrompu par Alt-Mode ou par S^C
- au fils n, ($n \in [0,7]$) si le programme a fait un changement de PRESC et s'il a correctement positionné le Compte-Rendu (cf schéma de fonctionnement du processeur EI/GO).

■ *type = R réponse en français libre* : EI appelle le processeur PH, celui-ci demande une phrase ; la lui fournir. Ensuite PH exploite cette phrase et en tire une valeur sémantique v. Cette valeur v est comparée aux valeurs v_0 à v_7 placées dans l'item courant. Si une coïncidence $v = v_i$ est trouvée EI passe au fils i du noeud courant. Sinon PH redemande PHRASE >.

■ *type = S séquence audiovisuelle* : EI appelle le processeur VI qui va exploiter l'item (voir processeur SE type S). Ensuite EI reprend le parcours en passant :

- au fils 0 du noeud courant si VI a normalement terminé son travail
- au fils 1 du noeud courant si rien n'a été fait par VI, ceci pouvant se produire dans 2 cas :

1°) l'item en cours contient des arguments erronés pour VI (bien qu'ayant été acceptés sous SE type S).

2°) ACN ≠ :EAØ (cette contrainte ACN = :EAØ existe uniquement lors de l'exploitation d'un graphe).

- au fils 2 de l'item en cours si VI a été interrompu par S^C (ceci peut être utilisé pour rediffuser un item).

Nota : le passage au fils du noeud en cours peut être précédé d'un effacement de l'écran si celui-ci est demandé dans l'item (cf : processeur SE type S).

■ *l'item en cours de type S peut demander l'une des opérations suivantes sur magnétoscope : play en continu, positionnement, stop, search and play - Se reporter au processeur SE type S.*

■ *type = T Texte* : EI appelle le processeur ED. Celui-ci exploite l'item ; on note que

- si le 1er caractère du texte ≠ "." ED assure l'effacement préalable de l'écran
- si le 1er caractère du texte = "." ED n'efface pas l'écran.

Après exploitation de l'item par ED, EI passe :

- au fils 0 du noeud en cours si l'édition s'est terminée normalement
- au fils 2 du noeud en cours si l'édition a été interrompue par S^C ou par Alt-Mode.

■ *type = V item vide* : EI passe au fils 0 du noeud en cours.

■ *type = W Pause* : deux types de pauses : Asynchrone (A) ou Synchrone (S) ; voir processeur SE type W.

1°)-type A : EI envoie date et PAUSE > plusieurs réponses sont possibles :

- G ou Return : EI sort la date et l'heure et passe au fils 0 du noeud courant.

- C^C : provoque un retour au CCI. On revient par !GØ et alors EI sort la date et l'heure et réaffiche PAUSE > . Cela permet par exemple d'aller patcher le processeur EI en cours de parcours.

Exemple : date et heure
PAUSE > C^C
?
séquence de patch
? !GØ D^C
Date et heure
PAUSE > G
Date et heure
poursuite du parcours

- F^C : provoque un retour à GE(!). On revient en frappant EI sous ! et alors EI passe au fils 0 du noeud en cours. Ceci permet d'aller effectuer des opérations sous GE en cours de parcours.

Exemple :

date et heure
PAUSE > F^C
!

! EI
poursuite du parcours fils 0 du noeud courant.

- autre : EI boucle sur l'affichage de date-heure et PAUSE >

2°)- type S pause synchrone : EI affiche : n MINUTES DE PAUSE et reprend automatiquement le parcours lorsque le temps est écoulé.

Exemple :

≀

03 MINUTES DE PAUSE

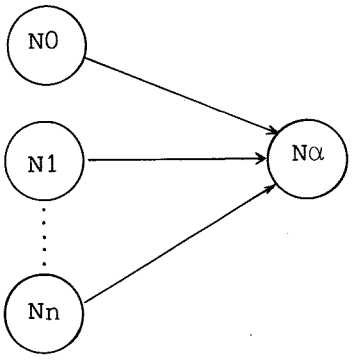
reprise du parcours au bout de 3 minutes

≀

Nota : Quel que soit le type de pause, la reprise du parcours se fait après effacement d'écran si celui-ci est demandé dans l'item (cf : processeur SE type W).

Remarque générale :

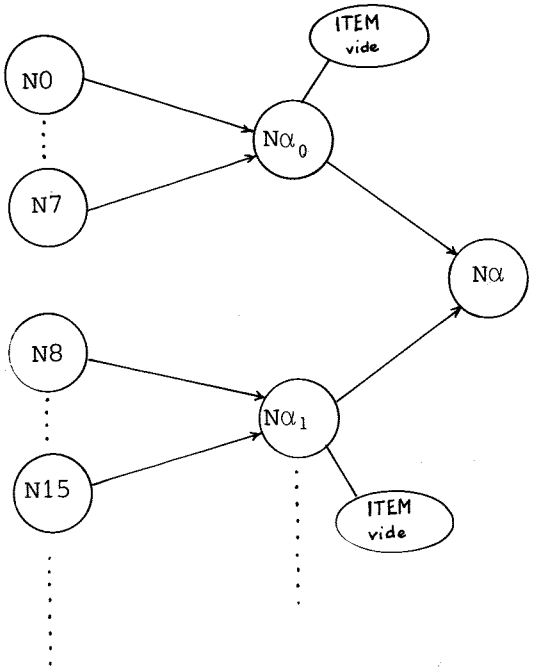
Soit la structure de noeuds suivante :



cette structure ne peut être réalisée car Nα ne peut avoir plus de 8 pères.

Ce problème peut être résolu en intercalant des noeuds intermédiaires $N\alpha_0$, $N\alpha_1$,....,$N\alpha_7$. A chacun de ces noeuds sera associé un item vide.

$7 < n < 63$



PROCESSEUR VI

Fonction

VI permet d'exploiter un item de type S (séquence audiovisuelle), celui-ci ayant été créé par l'utilisateur au moyen du processeur SE (voir ce processeur). On rappelle que la séquence audiovisuelle en question peut contenir l'un des ordres suivants :

- stop magnétoscope
- positionnement de la bande magnétoscope à une adresse
- play à partir de la position courante
- search and play à partir de la position courante ou à partir d'une adresse de début, et jusqu'à une adresse de fin.

Utilisation

VI est utilisable de deux façons :

- sous GE (!). Il suffit de charger l'item-séquence audiovisuelle par IL puis d'appeler VI.
- lors d'un parcours de graphe d'enchaînement ; dans ce cas, l'item-séquence audiovisuelle est intégré à une structure de noeuds et d'items (voir SMC généralités). VI sera appelé automatiquement lorsque l'item sera exploité, du fait que celui-ci possède la type S.

Il faut noter ceci :

- l'item séquence audiovisuelle doit avoir un nom de la forme : < nom de bande > [∅ < nom de séquence >]. Lors de son exploitation, VI demandera sur un télétype si l'on est en T1600, sur l'écran de la visu si l'on est en Solar, si la bande à utiliser est montée ou non. Il sortira le message suivant (par exemple) :

MG08BD1 { Ø ou Return (oui) ; N (non) }

qui signifie : "la bande BD1 est-elle montée sur le magnétoscope 08 ?", puis il attendra une réponse. L'utilisateur peut alors :

- monter la bande et répondre oui, auquel cas VI continuera normalement.
- répondra non, auquel cas VI interrompra immédiatement l'exploitation de l'item et sortira en erreur (c'est-à-dire, au cours d'un parcours de graphe, par le fils 2 du noeud courant).

VI ne réclame le montage d'une bande que pour le premier item rencontré référençant cette bande. Pour les séquences suivantes de cette bande, il ne sortira donc pas de message.

Nota : Pour utiliser VI il faut, en général, travailler sous l'ACN : EAØ.

ANNEXES

CONVENTIONS GENERALES

1 - Notations

■ *un nombre hexadécimal* est représenté précédé du caractère "quote", pour le distinguer d'un nombre décimal.

Exemples :

'10 = 16 en décimal

'2A = 42 en décimal

'FF = 255 en décimal

■ *lettre o et chiffre 0* : sont représentés respectivement dans les modes d'emploi par Ø et 0.

■ *touches CTRL et SHIFT* : pour exprimer le fait d'appuyer simultanément sur une touche (par exemple A) et la touche CTRL, on écrira A^{C}. De même pour SHIFT, on écrira A^{S}. Pour CTRL-SHIFT on écrira A^{CS}.

■ *dialogue utilisateur-système* : dans les exemples de dialogue utilisateur-système, on distingue ce qui est écrit par le système et ce qui est rentré par l'utilisateur de la façon suivante :

- ce qui est écrit par le système est souligné.
- ce qui est rentré par l'utilisateur n'est pas souligné.

Exemple :

!
NØM > ABCDEFD^{C}
>

2 - Caractères particuliers

■ *D^{C} (CTRL - D) caractère de fin de message*. Ce caractère n'est pas toujours nécessaire ; il ne l'est que pour des messages de longueur variable ("carte" de contrôle, nom de fichier, nom d'item par exemple). Dans les exemples, il est toujours mentionné quand il est requis. Noter qu'on peut souvent employer Return au lieu de D^{C} mais qu'un nom (nom de fichier, nom d'item) doit toujours être délimité par D^{C} et jamais par Return.

■ *Alt - Mode* : ($\}^{S}$ sur visu 4014 ; touche Alt-Mode sur 4010). Ce caractère est utilisé :

- pour établir la connexion avec le système en début de session (voir procédure de logon).
- en cours de session

- un seul Alt - Mode : interrompt le déroulement normal du programme en cours

Celui-ci reprenant son exécution à une certaine adresse.

- deux Alt - Mode successifs rapprochés : provoque l'abandon brutal de l'exécution du programme en cours, et le retour au CCI (?)

■ S^C : interrompt la sortie en cours. S'il y a d'autres sorties à la suite, elles apparaîtront. Pour interrompre frapper alors un Alt - Mode.

■ RUB ØUT : annulation du dernier caractère du message qu'on est en train de rentrer. Cette touche est sans effet si ce caractère était le denier caractère attendu par le système pour un message de longueur fixe (c'est-à-dire pour lequel D^C n'est pas requis). RUB ØUT peut se répéter n fois (effacement des n derniers caractères rentrés).

■ BREAK : annulation de tous les caractères du message qu'on est en train de rentrer. Voir ci-après particularités Solar.

■ V^C : caractère de tabulation. La tabulation étant établie soit implicitement (par !CALL mode T, voir CMS) ou explicitement (par !TAB, voir CMS).

3 - Particularités Solar

En Solar on a la possibilité de frappe anticipée de caractères, on peut donc rentrer plusieurs messages à la suite sans avoir à attendre pour rentrer un message que le système ait répondu au message précédent. Cela est rendu possible par l'existence d'un buffer circulaire de 22 caractères, où sont stockés les caractères frappés. Si l'on dépasse 22 caractères anticipés, les premiers rentrés seront perdus.

Dans ce contexte, le BREAK annule tous les caractères en attente (il peut y avoir plusieurs messages en attente) et n'est pas lui-même mémorisé. De même, le Alt-Mode entraîne aussi la remise à zéro du buffer circulaire, en plus de son effet habituel (déroutement du programme en cours) et n'est pas lui-même mémorisé.

4 - Utilisation des visus 4010 et 4014.

Pour utiliser ces visus, procéder de la façon suivante :

- mettre sous-tension, et attendre que l'écran devienne progressivement lumineux.

- faire PAGE

- s'assurer que la visu est "on line"

- effectuer le dialogue avec le système (logon, session, logoff, voir CMS)

- le dialogue étant terminé, faire PAGE et mettre hors-tension.

Lorsque l'on utilise une visu 4010 ou 4014, son écran passe en faible intensité au bout d'un certain temps de non utilisation. Pour rafraichir l'écran, appuyer sur une touche (SHIFT par exemple).

Sur visu 4014, la taille des caractères alphanumériques est modifiable.

Pour la fixer :

- passer en mode LØCAL,

- appuyer sur la touche ESC, puis sur l'une des touches 8 ou 9 ou : ou ; (8 donne la taille maximum et; la taille minimum)

- repasser en mode LINE.

5 - Table de vérité des opérations logiques

a	b	a ØR b	a AND b	a EOR b
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

ØR : ØU logique

AND : ET logique

EØR : ØU exclusif ("exclusive ØR")

LEXIQUE

ET

INDEX

Le présent lexique est utilisable à la fois comme lexique et comme index.

Les mots ne commençant pas par un caractère alphabétique (par exemple les commandes de CMS qui commencent toutes par un "!") ont été placés en fin de lexique.

La notation utilisée est naturelle, c'est-à-dire que l'on n'a pas distingué comme habituellement lette O et chiffre 0.

ACN : abréviation de "account number" ; voir NUMERO DE COMPTE.

ALT-MODE : voir page 534 dans les "conventions générales".

ALPHANUMERIQUE : qui concerne les caractères alphabétiques, les chiffres, les signes de ponctuation et autres signes spéciaux. On parlera,par exemple de VISUEL (voir ce mot) alphanumérique, c'est-à-dire sur lequel on ne pourra qu'afficher des textes.

ANALOGIQUE : qualifie la représentation d'un phénomène physique par un signal, électrique en général, dont les variations sont continues. On oppose analogique à NUMERIQUE ou DIGITAL (voir ces mots).

ANALYSE D'UN SIGNAL ANALOGIQUE : consiste à en extraire une suite d'échantillons numériques régulièrement espacés dans le temps (on dira aussi "digitalisation"). C'est ainsi qu'une image de télévision (signal vidéo) est "numérisée" ou "digitalisée" pour devenir une image numérique, exploitable sur ordinateur.

ANALYSE D'UNE PHRASE : voir page 512 le processeur PH de SMC.

AND : opération logique (en français : ET logique) utilisée notamment sous le processeur TV de SMC pour faire des opérations entre images. Table de vérité en page 536.

ANIMATION : sous SMC on peut construire des animations en utilisant les processeurs TA, TB, TØ, et les diffuser en utilisant généralement le processeur SC. Voir ces processeurs.

ARBORESCENCE : ou "structure d'arbre" ; voir page 61 et suivantes.

ARGUMENT FORMEL : voir page 292 (processeur GR).

ASCI : nom du code utilisé pour représenter en mémoire de l'ordinateur les différents caractères alphabétiques et numériques.

ASSEMBLEUR : programme traduisant en langage-machine (c'est-à-dire en une suite de codes binaires directement exécutable par l'ordinateur) un programme-source écrit en langage d'assemblage. On appelera "langage d'assemblage" un langage-machine symbolique où les codes binaires d'instructions et les adresses sont exprimés respectivement par des mnémoniques et des symboles (ou noms).

LEXIQUE 2

ASSIGNATION : opération consistant à associer une unité logique (ou périphérique logique), souvent désignée par un numéro (1, 2, 3, ..., 9, A, B, C, D, E dans CMS), à un organe physique d'entrée - sortie (imprimante, lecteur de cartes, disque, etc) ou à un fichier ou, plus généralement, à un programme spécialisé (appelé "tâche") du système d'exploitation. Voir pages 19 et suivantes et page 45).

ASSIGNATION GENERALISEE : voir pages 46 et 78.

ASSYS : nom de l'assembleur de CMS, page 94.

BASE 2, BASE 10, BASE 16 : En base 2, on a deux symboles (1 et 0) pour exprimer un nombre appelé alors nombre binaire. En base 10, 10 symboles : c'est la base décimale habituelle. En base 16, ou base hexadécimale, 16 symboles (0,1,2 ..., 9, A, B, C, D, E, F). Les bases binaire et hexadécimale sont bien adaptées au calcul sur ordinateur.

BATCH : on désigne ainsi le "batch-processing" ou TRAITEMENT PAR LOTS (voir ce mot).

BIBLIOTHEQUE : voir ESPACE DES NOMS ET VALEURS.

BINAIRE : qualité de toute information exprimée en BASE 2 (voir ce mot).

BINARISATION : action consistant à traduire une information en base 2 ; par extension, on peut parler de binarisation d'une arborescence (voir page 62).

BIT : abréviation de "binary digit". Désigne un chiffre binaire (0 ou 1), servant à la représentation des informations sur ordinateur.

BLOC : dans CMS désigne le plus petit sous-ensemble d'informations appartenant à un fichier (voir page 55).

BOX : où "boîte à lettres" utilisée par CMS pour transmettre une information à un utilisateur suite à l'exécution d'une demande en provenance de celui-ci. Voir page 67 comment récupérer dans un programme le contenu de BOX.

BUFFER : ou "mémoire tampon" utilisée notamment lors d'un échange entre un programme et une mémoire auxiliaire (entrée-sortie).

CALCUL : nom d'un processeur de CMS simulant une calculette sur une console de visualisation (voir page 214).

CAMERA VIDEO : c'est l'une des sources possibles d'images en entrée du système SMC. Elle est utilisable sous le processeur TV de SMC.

CARTES DE CONTROLE : voir page 17 et suivantes.

CCI : "control cards interpreter" ou interpréteur des cartes de contrôle.

CHAINAGE : information constituant un lien logique entre deux autres informations. Pour le chaînage entre blocs de fichiers voir pages 55 et 57. Pour le chaînage entre items et noeuds voir page 229 et suivantes et pages 251, 252.

CHARGEMENT : en général, opération consistant à transférer en mémoire centrale de l'ordinateur une information stockée sur un support externe (exemple : chargement d'une image numérique à partir d'un disque).

CHARGEUR : programme capable de charger en mémoire un autre programme en vue de l'éxécution de celui-ci. Dans CMS, c'est le processeur !CALL qui fait ce travail.

CHECKPOINT : littéralement : point de contrôle ; sauvegarde périodique d'une "photographie" de l'état d'un programme, et du système d'exploitation en particulier, à un instant donné, et permettant par la suite un redémarrage à la suite d'un défaut éventuel. Sous CMS5 le checkpoint du système est automatique.

Ci (CO, C1, ..., C7), C - : commandes de connexion - déconnexion entre noeuds dans un graphe d'enchaînement (voir pages 251 et 252).

CLE : identificateur d'un enregistrement de fichier. Une clé est composée d'une partie entière (notée N1) et d'une partie décimale (notée N2). Voir page 55.

CLOSE : mot anglais signifiant en français "fermer" ou "fermeture". Opération consistant à fermer un fichier ou un enregistrement de fichier, préalablement ouvert (voir OUVERTURE). On ferme un enregistrement par un "close key" (pages 55 - 56) et un fichier par désassignation (page 21).

CMS, CMS 4, CMS 5 : initiales de COLONNA MONITOR SYSTEM, nom du système d'exploitation, logiciel supportant SMC, implanté sur les ordinateurs T1600 et SOLAR du LACTAMME (CMS4 sur T1600 et CMS5 sur SOLAR). Voir page 15.

CODE RETOUR : information rendue par CMS à la suite de l'exécution d'une demande en provenance d'un programme. Voir pages 76 et 77.

COMPACTAGE : opération consistant à réduire le volume occupé par une information. Compactage des noms d'items-utilisateurs et de noeuds, voir pages 64 et 226 et suivantes.

CONSOLE DE VISUALISATION : voir VISUEL.

CONSOMMATEUR (processeur consommateur) : voir pages 259 à 261.

CONTEXTE GRAPHIQUE : ensemble d'informations relatives à un dessin ou à une partie d'un dessin, interrompu à un certain instant de son tracé, et permettant sa reprise ultérieure par l'interpéteur graphique de SMC (voir page 284, processeur GR).

CONTOUR : voir processeurs KO, ES et EZ de SMC page 391 et suivantes.

CONVENTIONS GENERALES : concernant notamment les notations utilisées dans ce manuel. Voir page 534.

CONVERSATIONNEL : qualifie un mode de traitement de l'information permettant un dialogue entre l'utilisateur et le système. On dira que CMS et SMC sont conversationnels.

COUPLEUR UNIVERSEL : dispositif matériel permettant à l'ordinateur d'envoyer des informations vers un organe externe non standard (piste son d'un magnétoscope, perforateur de cartes, etc...).

CR1, CR2 : noms utilisés, dans les assignations notamment, pour désigner les lecteurs de cartes. Voir page 20. CR = Cards Reader.

CU1, CU2, CU3 : noms utilisés, dans les assignations notamment, pour désigner les coupleurs universels. Voir page 20 (CU = Coupleur Universel).

CURSEUR : objet matériel ou logiciel désignant un point quelconque de l'écran d'un visuel. Ce point peut être matérialisé sur l'écran du visuel par l'intersection d'une ligne verticale et d'une ligne horizontale, auquel cas on parle de RETICULE, et il peut être déplacé par l'utilisateur en utilisant deux mollettes situées sur le clavier. Sous certains processeurs de SMC (TV, VO, KO, ES, LP notamment), une commande permet de faire clignoter sur l'image de télévision le point correspondant à la position courante du curseur du visuel utilisé. Voir ces processeurs. Pour la notion de curseur dans GR voir page 283.

DEBUG : anglicisme signifiant recherche et correction d'erreurs, notamment lors de la mise au point de programmes. Voir page 39.

DEFINITION : désigne dans ce document la qualité de représentation d'un dessin sur écran graphique, d'une image numérique sur écran de télévision, et s'exprime en nombre de points (voir page 257).

DELETE : en français, signifie annuler ou détruire (exemple : delete file = suppression fichier).

DEMANDE : requête adressée par un utilisateur au système d'exploitation, exprimée sous forme d'une table contenant des arguments et transmise par un SVC (voir page 44).

DEPILER : opération de gestion de PILE (voir ce mot) consistant à récupérer l'information placée au sommet de la pile (c'est le contraire de l'opération EMPILER).

DESASSIGNATION : c'est le contraire de l'assignation : la désassignation détruit l'effet d'une ou plusieurs assignations (voir page 21).

DG : processeur de SMC, page 403

DI : processeur de SMC, page 507.

DIGIT : groupe de 4 BITS. Deux digits forment un OCTET (voir ce mot).

DIGITAL : anglicisme, synonyme de NUMERIQUE ; voir ce mot.

DISPLAY : mot anglais signifiant affichage ou visualisation d'une information, ou d'un ensemble d'informations (exemple : display image sous le processeur TV de SMC).

DISQUE : mémoire auxiliaire d'un ordinateur. On distingue les disques inamovibles et à têtes fixes, et les disques amovibles, à têtes mobiles. Voir les mots suivants.

DK1, DK2, DK3 : nom des disques (à têtes fixes) du T1600.

DKF, DKM, DKU : nom des disques du SOLAR ; DKF est un disque à têtes fixes, DKM et DKU sont amovibles ; DKU étant le disque - utilisateur, et étant assignable (voir pages 20, 57 et 78).

DLN : fonction du SGN, voir page 63.

DUMP : nom d'un processeur de CMS ; voir page 195.

EAO : initiales de Enseignement Assisté par Ordinateur.

ECRAN GRAPHIQUE : voir page 257.

ED : processeur de SMC éditeur de textes ; voir page 264.

EDITEUR DE LIENS : programme produisant un programme exécutable à partir d'un ou plusieurs "modules binaires", ceux-ci ayant été obtenus par assemblage à partir de programmes-source. Sous CMS, c'est le processeur LINK.

EDITEUR DE TEXTES, EDITEUR DE FICHIERS : programmes permettant de créer et de mettre à jour des textes contenus soit dans des fichiers soit dans des items. Sous CMS, l'éditeur de fichier est le processeur EDITS (page 81). Sous SMC, l'éditeur de textes est le processeur ED (page 264). Pour créer un programme graphique, ou un scénario, par exemple, sous SMC, on utilisera le processeur ED.

EDITS : nom de l'éditeur de fichiers de CMS, page 81.

EI : processeur de SMC, voir page 525.

EMPILER : opération de gestion de PILE (voir ce mot) consistant à placer une information au sommet de la pile (c'est le contraire de l'opération DEPILER).

ENREGISTREMENT : dans le contexte SGF, c'est un sous-ensemble d'un fichier, identifié par une clé (voir pages 55 et 57).

EOR : opération logique (Exclusive OR, OU exclusif), utilisée notamment sous le processeur TV de SMC pour faire des opérations entre images. Table de vérité en page 536.

EOT : initiales de END OF TEXT, caractère désignant la fin d'une chaîne de caractères (d'un message) rentré par exemple au clavier d'un visuel. Dans les exemples de dialogue, ce caractère est symbolisé par D^C, parce qu'étant obtenu par action simultanée sur les touches CTRL et D.

ES : processeur de SMC, voir page 394.

ESCLAVE : mode de fonctionnement d'une unité centrale dans lequel seules les

opérations qui ne modifient pas l'état général du système sont autorisées. Ce mode est en général réservé aux utilisateurs. On l'oppose au mode MAITRE (voir ce mot).

ESPACE DES NOMS ET VALEURS : on désigne ainsi le catalogue et la bibliothèque du système. Il contient un ensemble d'élèments, chacun de ceux-ci étant un couple $\{$nom, valeur$\}$ appelé ITEM. Cet espace est géré par CMS pour ses besoins propres et pour ceux des utilisateurs. Il est partitionné par ACN. Voir sa description pages 60 et suivantes, 225 et suivantes.

ESPACE MEMOIRE UTILISATEUR : c'est une partie de la mémoire centrale de l'ordinateur, allouée par le système d'exploitation à un utilisateur. Voir MULTIPROGRAMMATION, TEMPS PARTAGE et PARTITION MEMOIRE UTILISATEUR.

EX : processeur de SMC ; voir page 485.

EZ : processeur de SMC ; voir page 396.

FERMETURE : voir CLOSE.

FF : commande de SMC, voir page 253.

FICHIER : collection organisée d'informations, regroupées parce qu'étant par exemple de même nature, ou traitées par un même programme. On parlera par exemple de fichier-source (contenant un programme-source), de fichier-objet (contenant un programme assemblé), de fichier graphique (contenant la description d'un dessin).

FILE : mot anglais signifiant FICHIER.

FILS : qualifie un chaînage (chaînage-père, chaînage-fils).

GC : processeur de SMC, page 480.

GE : processeur de SMC, assurant l'exécution des commandes de SMC et des appels de processeurs (pages 245 à 256).

GF : processeur de SMC, page 447.

GG : processeur de SMC, page 444.

GI : processeur de SMC, page 499.

GL : processeur de SMC, page 460.

GO : processeur de SMC, page 525.

GR : processeur de SMC, page 270.

GRAPHE : structure logique formée de sommets (ou noeuds), reliés par des arcs (ou liens, ou chaînages), permettant de décrire par exemple un enchaînement plus ou moins complexe de séquences audiovisuelles (voir page 229).

GRAPHIQUE : qui se rapporte au dessin. On parlera par exemple de visuel graphique (sur l'écran duquel peuvent être tracés des dessins), d'interpréteur graphique (programme capable d'exploiter un programme graphique, comme le processeur GR de SMC) de programme graphique, de fichier graphique, etc...

GT : processeur de SMC, voir GR.

GV : processeur de SMC, voir GR.

GW : processeur de SMC, voir GR.

G2 : processeur de SMC, voir GR.

G3 : processeur de SMC, voir GR.

HARD - COPY : machine permettant de reproduire sur papier sensible le contenu de l'écran d'un visuel, notamment pour conserver un dessin.

HARDWARE : anglicisme signifiant MATERIEL. Voir ce mot.

HEXADECIMAL : qui est exprimé en BASE 16. Voir ce mot.

HOME : curseur HOME signifie en haut à gauche de l'écran.

IL : commande de SMC, page 247.

IMAGE : désigne en général une image numérique produite par SMC et se composant de 256 lignes de 256 points chacune, chaque point pouvant être sombre (bit à 0) ou lumineux (bit à 1). Sur SOLAR, on peut visualiser simultanément sur écran de télévision couleur, un ensemble de trois images, par exemple les composantes rouge, verte, bleue (R, V, B), avec choix des couleurs et intensités lumineuses (voir notamment processeur TV de SMC page 311).

IMAGE RESIDENTE, IMAGE SCRATCH : on entend par image résidente une image numérique placée en mémoire de l'ordinateur dans la partition utilisateur et visualisée éventuellement sur écran de télévision. L'image scratch est une image de travail placée non pas en mémoire mais sur disque, l'utilisateur pouvant effectuer notamment des opérations entre image résidente et image scratch (opérations logiques ou arithmétiques). Voir le processeur TV de SMC page 311.

INTERACTIF : qualifie ce qui relève de l'INTERACTION (voir ce mot).

INTERACTION : dans SMC, on nomme ainsi un dialogue entre un utilisateur et le système, par l'intermédiaire d'un appareil (visuel par exemple) permettant à l'utilisateur d'émettre des informations (questions ou réponses) à destination de l'ordinateur et réciproquement.

INTERFACE : limite commune à deux ensembles (par exemple à deux appareils, à deux programmes).

INTERPOLATEUR : programme capable d'INTERPOLER. Voir ce mot.

INTERPOLER : c'est, dans SMC, construire, à partir d'un dessin (ou d'une image) "de départ" et d'un dessin (ou image) "d'arrivée", un certain nombre (spécifié

par l'utilisateur) de dessins (ou images) "intermédiaires", ce qui permet d'obtenir des animations. Dans SMC, les processeurs TA, TB et TO sont des interpolateurs.

INTERPRETER : interpréter un programme, c'est l'exécuter sans le traduire en langage machine. C'est le cas notamment du processeur graphique GR de SMC, capable d'exécuter des programmes graphiques écrits par les utilisateurs dans un langage graphique (décrit dans ce manuel, voir processeur GR). Pour chaque instruction ou PRIMITIVE (voir ce mot) du programme, l'interpréteur appelle un sous-programme spécifique écrit en langage machine. On peut dire qu'un interpréteur simule une unité centrale d'ordinateur dont le langage-machine serait composé des instructions qu'il interprète. L'interprétation hausse la machine au niveau du langage (alors que la compilation abaisse le langage au niveau de la machine).

IS : commande de SMC, page 248.

ITEM : ensemble d'informations formant un tout logique ou représentant une notion élémentaire dans un certain contexte, notamment pédagogique (par exemple une démonstration de théorème). Dans SMC un item est rangé dans l'ESPACE DES NOMS ET VALEURS (voir ce mot), cette opération étant effectuée par CMS pour qui un item est réduit à un couple $\{$nom, valeur$\}$. Pour les différents types d'items, les noms compactés et les filiations, se reporter aux pages 225 et suivantes. Une image numérique est un exemple d'item : son nom est celui que lui a donné l'utilisateur, sa valeur est l'image numérique elle-même.

ITEM 1, ITEM 2 : désignent deux zones de la PARTITION MEMOIRE (voir ce mot) d'un utilisateur travaillant sous SMC, et destinées à recevoir chacune le contenu d'un item-utilisateur (voir page 234). C'est notamment dans ces zones que la plupart des processeurs de SMC prennent leurs arguments.

ITERATION : action consistant à répéter n fois une séquence de programme. Lorsqu'il s'agit d'un programme graphique, voir page 281.

IU : commande de SMC, page 249.

I+ : commande de SMC, page 246.

I- : commande de SMC, page 246

I= : commande de SMC, page 249.

K : facteur multiplicatif valant $2^{10} = 1024$. Par exemple, une mémoire de 32K mots contient 32768 mots.

KA : processeur de SMC, page 493.

KEY : mot anglais signifiant CLE, voir ce mot.

KI : processeur de SMC, page 476.

KO : processeur de SMC, page 391.

LACTAMME : Laboratoire Commun des Techniques Audiovisuelles et des Moyens Modernes d'Enseignement.Commun à l'Ecole Polytechnique et à l'Ecole Nationale Supérieure des Télécommunications.

LI : commande de SMC, page 249.

LIGHTPEN (ou PHOTOSTYLE) : "crayon" permettant de dessiner directement sur un écran. Dans SMC cet appareil permet de rentrer des tracés en couleur sur écran de télévision, ainsi que d'effacer. Voir processeur TN de SMC page 364.

LINK : nom de l'EDITEUR DE LIENS (voir ce mot) de CMS (page 193).

LKSYS : nom d'un processeur de CMS, non documenté dans ce manuel, assurant les mêmes fonctions que LINK, mais plaçant le code exécutable sur disque, à l'adresse spécifiée par l'utilisateur. Utilisé pour "linker" le système lui-même.

LN : commande de SMC, page 251.

LNS : fonction du SGN, voir page 64.

LNU : fonction du SGN, voir page 64.

LOAD : mot anglais signifiant CHARGEMENT, voir ce mot.

LOCAL : mode LOCAL (par opposition au mode LINE) qualifie le mode de fonctionnement d'un visuel non relié à l'ordinateur.

LOGICIEL : ensemble de programmes , procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de l'information (ce terme équivaut au mot "software" dont l'usage est à proscrire).

LOGIN ou LOGON : désigne la procèdure consistant pour un utilisateur à établir le dialogue avec le système (voir pages 9 à 11 et pages 17 et 18).

LOGOUT ou LOGOFF : désigne la procédure consistant pour un utilisateur à rompre le dialogue avec le système (voir pages 9 à 11 et pages 17, 18).

LON : fonction du SGN, page 63.

LP : processeur de SMC, page 410.

LP1 : nom utilisé, dans les assignations notamment, pour désigner l'imprimante. Voir page 20. LP = Lines Printer.

M1 : commande de SMC, page 252.

M2 : commande de SMC, page 252.

MAGNETOSCOPE : appareil analogue fonctionnellement à un magnétophone et permettant l'enregistrement de signaux de fréquences très élevées (de l'ordre de quelques méga-hertz) ou "vidéo-fréquences". Il permet le stockage et la restitution d'images de télévision et du son associé. On classe habituellement les

magnétoscopes suivant la largeur des bandes utilisées :

- 1/2 pouce (VHS) et 3/4 de pouce (1 pouce = 25,4 mm) : destinés au grand public.

- 1 pouce : équipement semi-professionnel.

- 2 pouces : réservés aux professionnels. Ils permettent en effet de faire du montage de séquences d'images du type cinématographique (c'est-à-dire à l'image près).

MAITRE : mode de fonctionnement d'une unité centrale dans lequel toutes les opérations sont autorisées. Ce mode est en général réservé au système d'exploitation. On l'oppose au mode ESCLAVE (voir ce mot).

MATERIEL : ensemble des éléments physiques (machines) employés pour le traitement de l'information (ce terme équivaut au mot "hardware" dont l'usage est à proscrire).

MEDIUM ; au pluriel MEDIA : système complet de communication allant de l'émetteur aux récepteurs. Il se décompose en deux parties : le support, contenant le message à transmettre, et le canal, assurant la transmission de ce dernier.

MEMOIRE CENTRALE, MEMOIRES AUXILIAIRES : la mémoire centrale de l'ordinateur est accessible directement par le processeur, dans des temps très brefs, de l'ordre de la micro-seconde ; les mémoires auxiliaires, supports externes d'informations ont,contrairement à la mémoire centrale une capacité très élevée et un temps d'accès long (de l'ordre de plusieurs dizaines de millisecondes pour un disque par exemple).

MISE AU POINT (d'un programme) : opérations consistant à rechercher et à corriger les erreurs d'un programme.

MONITEUR : voir SYSTEME D'EXPLOITATION

MOT : sur T1600 et sur SOLAR, un mot est formé de 2 OCTETS (ou 16 BITS).

MULTIMEDIA : qualifie un système de communication comportant plusieurs MEDIA.

MULTIPLEXER : transmettre plusieurs informations sur une même voie simultanément, en allouant par exemple à chaque information une bande de fréquences distincte. On peut également multiplexer dans le temps.

MULTIPROCESSEUR : se dit d'un ordinateur possèdant plusieurs unités centrales, celles-ci pouvant être banalisées, c'est-à-dire jouer des rôles identiques, ou non. Le système CMS fonctionne en multiprocessing sur l'ordinateur SOLAR (avec des unités centrales débanalisées).

MULTIPROGRAMMATION : on dit qu'un ordinateur fonctionne en multiprogrammation lorsque plusieurs programmes-utilisateurs se trouvent activables à un moment donné. Ce mode permet en général une meilleure utilisation d'un système, en occupant la mémoire par un maximum de programmes, en récupérant les attentes d'entrées-sorties d'un programme au profit d'autres programmes, etc... Une exploitation en TEMPS PARTAGE (voir ce mot) en est un cas particulier.

MX : commande de SMC, page 252.

NC : commande de SMC, page 251.

ND : commande de SMC, page 251.

NEW : mot anglais signifiant nouveau. Utilisé dans les assignations de fichiers ("assign new" pages 20, 21), et dans les ordres SGF d'ouverture d'enregistrements (open new key ; page 56).

NIVEAUX DE DIALOGUE : voir pages 10 à 12.

NOEUD : élément d'un GRAPHE (voir ce mot) utilisé dans la description de celui-ci (voir page 228 la description d'un noeud, et page 231 la structure physique d'un graphe d'enchaînement).

NOM COMPACTE : c'est le NOM INTERNE calculé par SMC à partir du NOM EXTERNE donné par l'utilisateur aux items-utilisateurs (page 226) et aux noeuds (page 228). Ce nom compacté est toujours de 6 caractères dont les 3 premiers sont ceux du nom externe, et les 3 suivants calculés en fonction des autres caractères du nom externe et de la longueur de celui-ci. Rappelons que dans l'espace des noms et valeurs, les items-utilisateurs et les noeuds sont catalogués sous leur nom interne.

NOM EXTERNE : c'est le nom donné par l'utilisateur aux items, noeuds, fichiers, programmes, images qu'il crée. Dans le cas des items-utilisateurs et des noeuds (pages 226 et suivantes), SMC calculera un NOM COMPACTE ou NOM INTERNE à partir du nom externe, et c'est sous ce nom compacté que seront catalogués les items et noeuds concernés dans l'espace des noms et valeurs. Dans tous les autres cas (notamment les images numériques) il n'y a pas compactage des noms et donc il y a identité entre nom externe et nom interne.

NOM INTERNE : des entités cataloguées dans l'espace des noms et valeurs : voir NOM COMPACTE et NOM EXTERNE. Nom interne de périphérique : c'est un numéro associé à chacun des périphériques de l'ordinateur, et plus généralement à chacune des tâches du système, chaque tâche étant spécialisée dans l'accomplissement d'un certain service, la gestion d'un périphérique constituant un cas particulier de service. Le nom interne d'un périphérique est aussi appelé NSP (nom symbolique de périphérique), et c'est lui qu'on utilise lors d'une "assignation généralisée" (page 46).

NSP : voir NOM INTERNE (de périphérique).

NUMERIQUE : en anglais "digital", qualifie quelque chose qui opère sur des nombres ou dont la représentation est discrète une IMAGE NUMERIQUE (voir ce mot) en est un exemple.

NUMERISATION : synonyme de conversion ANALOGIQUE-NUMERIQUE (voir ces mots ainsi que ANALYSE D'UN SIGNAL ANALOGIQUE). On dit aussi "digitalisation".

NUMERO DE COMPTE : code en général confidentiel et qui permet d'accéder à un ensemble d'informations protégées comme des items ou des fichiers dans un système informatique. L'espace des noms et valeurs est partitionné par numéro de compte (ou ACN = "account number").

NXP : fonction du SGN, page 64.

NXS : fonction du SGN, page 64.

N+ : commande de SMC, page 250.

N- : commande de SMC, page 250.

N1, N2 : utilisés respectivement pour désigner la partie entière et la partie décimale de la CLE (voir ce mot) d'un fichier.

N= : commande de SMC, page 250.

OAB : couple de coordonnées décrivant un segment dit "segment nul à l'origine", et donc ayant pour valeur : ((0,0), (0,0)). Cette notation OAB provient du langage graphique (voir processeur GR). Lors d'un échange de segments par ZDC (voir ce mot) entre deux processeurs, l'émission en ZDC d'un OAB par le PROCESSEUR PRODUCTEUR (voir ce mot) provoque la fin du processus d'échange.

OCTET : groupe de 8 BITS. Il y a deux octets par MOT sur les ordinateurs T1600 et SOLAR.

OFF LINE : "hors-ligne", qualifie tout périphérique non relié à l'ordinateur (par opposition à ON LINE). Un visuel en mode LOCAL est off-line.

OLD : mot anglais signifiant ancien. Utilisé dans les assignations de fichiers ("assign old", pages 20, 21), et dans les ordres SGF d'ouverture d'enregistrement (open old key, page 56).

ON LINE : "en-ligne", qualifie tout périphérique connecté à l'ordinateur, c'est-à-dire pouvant échanger des informations avec celui-ci.

OPEN : mot anglais signifiant ouvrir ou ouverture. Opération consistant à ouvrir un fichier ou un enregistrement de fichier (voir OUVERTURE).

OR : opération logique (en français : OU logique), utilisée notamment sous le processeur TV de SMC pour faire des opérations entre images. Table de vérité en page 536.

ORDINATEUR : machine conçue pour recevoir de l'information, la mémoriser, la traiter, et en restituer tout ou partie de ses éléments transformés. Ces opérations sont réalisées à l'aide de programmes enregistrés et constitués de directives arithmétiques et logiques. Les PERIPHERIQUES (voir ce mot) sont inséparables de l'ordinateur dans le cadre normal de son exploitation.

OUVERTURE : se rapporte à un fichier ou à un enregistrement appartenant à un fichier. Cette opération est indispensable pour utiliser un fichier : un BLOC appartenant à un ENREGISTREMENT ne peut être écrit ou lu que si l'enregistrement est ouvert (open old/new/next key, page 56), et un enregistrement ne peut être ouvert que si le fichier a été ouvert (par un assign new/old, pages 20, 21). Après avoir été utilisé, un enregistrement doit être fermé, de même pour le fichier (voir CLOSE).

OVERLAY : mot anglais signifiant recouvrement. S'applique en général à un programme constitué d'un "noyau" et de "branches d'overlay" qui sont des sous-programmes chargés successivement dans le même espace mémoire où ils s'exécutent et s'enchaînent en fonction des nécessités du traitement. SMC est un exemple d'une telle structure, car il est constitué d'un noyau et ses branches d'overlay sont les processeurs de SMC qui se succèdent dans la zone "processeur courant" de la partition mémoire utilisateur (pages 234, 235).

PA : processeur de SMC, page 472.

PARTITION UTILISATEUR : c'est l'espace mémoire virtuel alloué par le système à un utilisateur, et dans lequel sont placés les programmes appelés par celui-ci; ainsi que les données sur lesquelles il travaille (enregistrements de fichiers, items) à un instant donné. Voir notamment le schéma de la partition mémoire d'un utilisateur travaillant sous SMC, page 234.

PATCH : de l'anglais signifiant morceau. C'est l'opération, et le résultat de l'opération, consistant à changer par exemple une instruction dans un programme lors de sa mise au point ou pour des besoins particuliers. Cette opération est possible sous CMS en utilisant le processeur !DEBUG (page 39). On dit (encore) : " patcher un programme".

PATTERN : mot anglais (littéralement : forme), c'est, dans SMC, une séquence caractéristique de points (voir page 472 le processeur PA de SMC).

PERE : qualifie un chaînage (chaînage - père, chaînage - fils).

PERIPHERIE (d'un ordinateur) : c'est l'ensemble des PERIPHERIQUES (voir ce mot) qui lui sont raccordés.

PERIPHERIQUE (d'ordinateur) : désigne en général un organe MATERIEL (voir ce mot) permettant des échanges d'informations (mono ou bilatéraux) entre l'ordinateur et le monde extérieur. Par exemple, une imprimante, un disque magnétique, un VISUEL (voir ce mot) sont des périphériques.

PH : processeur de SMC, page 512.

PILE : structure sur laquelle deux opérations sont définies : EMPILER (anglais "push") permet de ranger une inforamtion au "sommet" de la pile. DEPILER (anglais "pull") permet de récupérer puis de supprimer l'information contenue au "sommet". Ainsi, la dernière information rangée sera la première à être restituée (last in, first out).

POSTE DE TRAVAIL : voir page 8.

PRESC : nom symbolique de l'adresse de déroutement d'un programme s'il reçoit un alt-mode. Cette adresse est modifiable dans le programme lui-même. Voir pages 69, 70.

PRIMITIVE ; élément de base d'un langage, notamment du langage graphique de SMC (voir processeur GR page 270).

PROCESSEUR : désigne tantôt l'unité centrale d'un ordinateur, tantôt un programme faisant partie d'une bibliothèque du système. Par exemple, l'éditeur

de liens, l'assembleur, sont des processeurs de CMS ; l'interpréteur graphique GR est processeur de SMC.

PROCESSEUR CONSOMMATEUR : se dit d'un processeur de SMC allant acquérir ("consommer") des informations en ZDC (voir ce mot).

PROCESSEUR DE BASE : qualifie des processeurs de CMS ayant pour particularité de ne pas être stockés en bibliothèque comme les autres processeurs, mais d'être stockés sur disque à des adresses physiques précises (données page 239), sur disques à têtes fixes (donc d'accès rapide). Ce sont les processeurs !CALL, !DF, !GE et !I répertoriés en fin de ce lexique et documentés dans ce manuel.

PROCESSEUR PRODUCTEUR : se dit d'un processeur de SMC allant stocker des informations en ZDC (voir ce mot), à destination d'un processeur consommateur.

PROGRAMME : expression d'un algorithme dans un certain langage.

PUNCH : de l'anglais, signifie perforer (des cartes).

QCM : initiales de "question à choix multiple" ; dans SMC, ce sont des items de type M pouvant être intégrés dans un graphe d'enchaînement (voir notamment les processeurs SE et EI/GO de SMC).

QUANTUM, QUANTA : facteur multiplicatif (valant actuellement 1 ou 3), indiquant le nombre de secteurs physiques disque (de 128 mots) constituant un secteur logique. Sur DKM et DKU (du SOLAR), le quanta utilisé est 3 ou 1. Sur les autres disques (DKF du SOLAR, DK1, DK2 et DK3 du T1600), le quanta est 1. Un BLOC (voir ce mot) de fichier par exemple, fait donc (3 x 128) mots en général sur SOLAR, et toujours 128 mots sur T1600.

RACINE : désigne en général les premières lettres consécutives communes à un ensemble de mots. Par exemple, les mots PROCESSEUR et PROGRAMME appartiennent à l'ensemble des mots de racine PRO. Voir page 62 la structure d'arbre.

RE : processeur de SMC, page 521.

READ : mot anglais signifiant lire ; ordre du SGF (voir lecture d'un bloc, SGF page 57).

RECOUVREMENT : voir OVERLAY.

REFERENTIEL : voir page 274, processeur GR.

REGIE AUDIOVISUELLE : désigne un ensemble de matériels, vidéo notamment, ainsi que leur lieu d'implantation, et à l'aide desquels sont réalisées les émissions de télévision. C'est là que sont reçues et choisies les images de toutes les caméras, et que l'on effectue les trucages et la diffusion.

RELEASE : mot anglais signifiant annulation ou destruction. Ordre du SGF (close release key, page 56, permettant de détruire un enregistrement de fichier).

RESIDENT : qualifie tout objet (programme, donnée) stocké en mémoire centrale de l'ordinateur à un instant donné et donc accessible directement par l'unité centrale. Exemple : image résidente. Le système d'exploitation CMS est résident.

REST : nom d'un processeur de SMC, voir page 207.

RESTART : mot anglais signifiant redémmarage. S'applique au système d'exploitation que l'on redémarre à la suite d'un défaut, en utilisant le dernier CHECKPOINT (voir ce mot).

RESTAURATION : récupération d'informations préalablement sauvegardées (Voir SAUVEGARDE).

RETICULE : organe d'entrée graphique utilisé dans certains visuels et permettant par exemple de tracer un dessin sur leur écran. Le réticule est moins souple qu'un light pen (voir aussi CURSEUR).

RETURN : touche du clavier d'un visuel provoquant le retour du curseur en début de ligne, et interprétée par le système comme une fin de message (voir conventions générales, CTRL - D page 534).

SA : processeur de SMC, page 490.

SAUVEGARDE : rangement d'informations (items, fichiers...) que l'on désire conserver "à l'abri", sur cartes perforées ou sur disque magnétique par exemple. Sous CMS on peut faire des sauvegardes de fichiers sous EDITS, de fichiers et/ou d'items par DUMP à restaurer respectivement sous EDITS et REST. Sous SMC on peut faire des sauvegardes/restaurations sous les processeurs SA et KA.

SC : processeur de SMC, page 350.

SCRATCH : mot anglais utilisé en général pour qualifier un espace mémoire ou un espace disque "de travail". On parlera par exemple d'IMAGE SCRATCH (voir ce mot), car cette image est placée dans un espace "disque scratch".

SE : processeur de SMC, page 500.

SECTEUR LOGIQUE : c'est un ensemble de secteurs physiques contenant Q secteurs physiques consécutifs, Q étant le QUANTA utilisé (voir ce mot).

SECTEUR PHYSIQUE : plus petit espace disque adressable physiquement. Sur les disques T1600 et SOLAR, il fait 128 mots. On peut en lire plusieurs (consécutifs) à la fois en une seule opération physique de lecture.

SESSION : désigne le temps passé par un utilisateur à dialoguer avec le système (du premier au dernier message échangé, c'est-à-dire du LOGON au LOGOFF, voir ces mots).

SGF : Système de Gestion de Fichiers ; sous-ensemble du système d'exploitation CMS. Voir page 54 et suivantes.

SGN : Système de Gestion des Noms ; sous-ensemble du système d'exploitation CMS. Voir page 60 et suivantes. C'est lui qui gère l'espace des noms et valeurs.

SLO et SLE : "Slave Origin" et "Slave End" ; ce sont des registres contenant respectivement l'adresse de début et l'adresse de fin de la partition d'un utilisateur. Leur contenu n'a le sens indiqué ici que si l'utilisateur travaille

en mode ESCLAVE (voir ce mot). Ils permettent entre autres au système de détecter les erreurs dites de "violation mémoire".

SMC : Système Multimédia Conversationnel.

SO : processeur de SMC, page 483.

SOFTWARE : voir LOGICIEL.

SOLAR : l'un des mini-ordinateurs du LACTAMME, supportant CMS et SMC.

SOURCE : qualificatif se rapportant à un programme rédigé par un utilisateur en vue d'être assemblé, compilé ou interprété , et par extension aux fichiers contenant de tels programmes (on dit "programme source", "fichier source").

SOUS-PROGRAMME : sous-ensemble d'un programme, spécialisé dans une certaine fonction. On parlera de programme principal et de sous-programme.

ST : processeur de SMC, page 452.

STN : fonction du SGN, page 63.

STORE : opération inverse du LOAD (voir ce mot), utilisé notamment pour qualifier l'une des fonctions du SGN (voir STN page 63).

STRUCTURE LOGIQUE ET STRUCTURE PHYSIQUE d'un graphe d'enchaînement : voir pages 229 et suivantes.

STYLISATION : d'un dessin ou d'une image. Effet obtenu en utilisant le processeur ST de SMC, ou l'un des processeurs DG, GF, etc... Il s'agit dans ce dernier cas d'une stylisation obtenue par le biais des paramètres K, X et Y définissant les échelles graphiques (voir GR page 274).

SVC : initiales de "supervisor call". Voir pages 44, 73 et 79.

SYNCHRONISATION INTER-PROGRAMMES : ensemble des opérations consistant pour deux programmes s'échangeant des informations à respecter des conventions de dialogue de manière à accorder leurs rythmes respectifs.

SYNTHESE D'UN SIGNAL : c'est l'opération inverse de l'ANALYSE D'UN SIGNAL (voir ce mot) et qui consiste à reconstituer un signal analogique à partir d'une suite d'échantillons numériques régulièrement espacés dans le temps.

SYSID : "system identifier" ; numéro affecté par le système CMS à chaque utilisateur lors de la procédure de logon. Ce numéro est différent pour chaque utilisateur, et l'on peut le connaître en utilisant le processeur de base !I, ou par programme par l'instruction '1E25 (voir page 67).

SYSTEME D'EXPLOITATION : programme ou ensemble de programmes destinés à gérer et optimiser l'utilisation des ressources matérielles et logicielles d'un ordinateur (allocation mémoire, allocation espace disque, gestion des périphériques, gestion des utilisateurs, etc...)

SWAPPING : de l'anglais swap signifiant échanger. Opération consistant à enlever de la mémoire centrale la partition complète d'un utilisateur et à la remplacer par celle d'un autre utilisateur. La partition mémoire du premier est sauvegardée sur disque et celle du second est chargée à partir du disque. Il s'agit d'un "disque de swapping". La fonction de swapping est l'une des fonctions remplies par un système d'exploitation en TEMPS PARTAGE (voir ce mot).

T1600 : l'un des mini-ordinateurs du LACTAMME supportant CMS et SMC.

TA : processeur de SMC, page 370.

TABULATION : la tabulation dite "standard" est obtenue en utilisant le processeur de base !CALL de CMS (page 29). La tabulation non standard est à définir par l'utilisateur en utilisant la commande !TAB de CMS (page 23).

TB : processeur de SMC, page 387.

TEMPORISATION : fonction pouvant faire l'objet d'une demande et qui aura pour effet de suspendre l'exécution du programme demandeur pour la durée spécifiée. Voir page 47.

TEMPS PARTAGE : le fonctionnement d'un ordinateur est dit en temps partagé lorsque plusieurs personnes peuvent l'utiliser simultanément et indépendamment les unes des autres. Ce fonctionnement sous-entend en général une interaction entre chaque utilisateur et le système par l'intermédiaire d'un terminal conversationnel (par exemple un visuel). On doit alors garantir un temps de réponse maximal (contrairement à la multiprogrammation dont le temps partagé n'est qu'un cas particulier). A cette fin, un système d'horloge interrompt les opérations de trop longue durée. Le système d'exploitation sera alors appelé "moniteur de temps partagé". CMS en est un exemple.

TERMINAL : périphérique autorisant l'interaction d'un utilisateur et du système.

TI : processeur de SMC, page 349.

TN : processeur de SMC, page 364.

TO : processeur de SMC, page 389.

TRAITEMENT PAR LOTS : en anglais "batch processing". Mode de traitement suivant lequel les programmes à exécuter ou les données à traiter sont groupés en lots et qui est à opposer à un traitement interactif de type temps partagé. Les données et les résultats utilisent en général le lecteur de cartes et l'imprimante respectivement (voir pages 9 et 22).

TRANSPOSITION : opération consistant dans une matrice à permuter lignes et colonnes. Opération applicable sous le processeur TV, à une image numérique, qui est une matrice de points.

TRAPPE PROGRAMME : erreur de programme fonctionnant en mode esclave et détectée par le système d'exploitation qui par conséquent interrompt le déroulement du programme et en avertit l'utilisateur.

TV : processeur de SMC, page 311.

TY1 : nom utilisé, dans les assignations notamment pour désigner le télétype. Voir page 20.

TYPES D'ITEMS : voir page 225 et processeurs SE et EI/GO.

UNITE LOGIQUE : Le système d'exploitation CMS met à la disposition de chaque utilisateur un certain nombre d'unités logiques, numérotées de 1 à 9 et de A à E. L'utilisateur peut disposer librement de 9 d'entre elles (3, 4, 5..., 9, A, B), notamment pour les assigner à des périphériques ou à des fichiers. Voir ASSIGNATION et ASSIGNATION GENERALISEE.

VA : processeur de SMC, page 434.

VALEUR : élément du couple {nom, valeur} formant un item de l'espace des noms et valeurs.

VALEUR SEMANTIQUE : voir processeurs DI et PH (pages 507 et 512).

VG : processeur de SMC, page 423.

VI : processeur de SMC, page 532.

VI1, VI2, ..., VI8 : nom utilisé, dans les assignation notamment, pour désigner les visuels 1 à 8. Voir page 20.

VIDEO : abréviation désignant les techniques gravitant autour des vidéo-fréquences (télévision), et s'utilisant aussi comme qualificatif : par exemple on parlera de bande vidéo, de caméra vidéo.

VIOLATION ECRITURE : erreur détectée par le système à la suite d'une tentative d'écriture par un programme sur un disque en mode "protection écriture", c'est-à-dire sur lequel la lecture seule est autorisée.

VIOLATION MEMOIRE : erreur détectée par le système à la suite d'une tentative d'accès mémoire fait par un programme travaillant en mode esclave, en dehors de la partition mémoire de l'utilisateur délimitée par les contenus de SLO et SLE (voir ces mots).

VIRTUALISATION : commande du processeur TV de SMC, pages 328 et 337.

VISUEL ou CONSOLE DE VISUALISATION : dispositif muni d'un clavier et d'un écran sur lequel l'ordinateur peut afficher des textes (visuel alphanumérique) et parfois des dessins (visuel graphique). Un visuel permet le dialogue utilisateur-système. Dans le manuel, on parle indifféremment de "visuel", "visu", "console de visualisation".

VO : processeur de SMC, page 357.

VT : processeur de SMC, page 443.

WAIT : mot anglais signifiant attendre. Voir la commande !WAIT de CMS page 22.

WRITE : mot anglais signifiant écrire. Ordre du SGF (voir page 57). S'oppose à READ (lire).

ZDC : initiales de Zone de Données Communes. C'est un ensemble de mots de la mémoire centrale de l'ordinateur, mis à la disposition des utilisateurs par le système CMS et permettant le dialogue et la synchronisation entre programmes de plusieurs utilisateurs se déroulant concurremment (voir page 47). Elle est utilisée notamment par certains processeurs de SMC pour échanger des coordonnées de segments. On parle alors de processeurs producteurs et processeurs consommateurs de segments. Voir pages 259 et 260.

ZO : processeur de SMC, page 465.

!ASSIGN : commande de CMS, page 19 et suivantes.

!BYE : commande de CMS, page 18.

!CALL : processeur de base de CMS, page 29.

!CLOSE : commande de CMS, page 21.

!DATE : commande de CMS, page 19.

!DEBUG : processeur de CMS d'aide à la mise au point, page 39.

!DF : processeur de base de CMS, page 29.

$!D^{C}$: réinitialisation à 0 de PRESC.

!F : commande de CMS, page 18.

!GE : processeur de base de CMS, page 29.

!GO : commande de CMS, page 23.

!I : processeur de base CMS, pages 28 et 29 bis.

!L : commande de CMS, page 17.

!Return : commande de CMS permettant de se rebrancher à l'adresse PRESC du programme interrompu (voir page 23).

!TAB : commande de CMS, page 23.

!W ou !WAIT : commande de CMS, page 22.

!X : commande de CMS, page 22.

-I : commande de SMC, page 247.

-N : commande de SMC, page 250.

1Eα5 : instruction "privilégiée", pages 67 à 71.

LEXIQUE 20

3D : processeur de SMC, page 480.

4010/4014 : nom des visuels graphiques utilisés actuellement au LACTAMME. Voir conventions générales page 535.

:EAO : numéro de compte utilisable sous CMS (initiales d'"enseignement assisté par ordinateur".

:SYS : numéro de compte utilisable sous CMS.

:USE : numéro de compte utilisable sous CMS.

TABLE DES MATIERES