UNIVERSITE D'ALGER

Ecole Nationale Polytechnique

DEPARTEMENT " ECONOMIE "

ÉTUDE DU SYSTEME PLAN

PROBLEM LANGUAGE ANALYZER

THESE DE FIN D'ETUDES

proposée par **C. GIDE**

étudiée par **A. AOUDJEHANE**

— PROMOTION 1973 —

Remerciements

Je tiens à remercier MM. Boumahrat, Adïba et Gide pour avoir bien voulu présider mon jury ainsi que tous mes Professeurs qui ont contribué à ma formation.

Que tous ce qui ont contribué à l'élaboration de cette thèse et en particulier à MM. R. Ouferhat et A. Mahreche recoivent ma sympathie.

Profonde gratitude à Mr. Gide pour tous les conseils qu'il m'a prodigué.

A mes parents

A mon ami Ahmed-Chaouch Abdellatif.

TABLE DES MATIERES

.../...

-=-=-=-==-

I N T R O D U C T I O N

L'ordinateur est à la mode. Les applications les plus spectaculaires l'on rendu célèbre.

Le premier réflexe quand on évoque l'ordinateur, est d'y voir la panacée, avec lui, tout va devenir simple.

Qu'il s'agisse d'imprimer un horoscope ou le cours de la bourse, d'établir un état de stock ou un programme optimum de découpage du verre, de donner le résultat du match Algérie-Brésil ou de préscrire à un malade un traitement à suivre, l'ordinateur est présent et a réponse à tout, l'intervention humaine semblant complétement disparue.

Cette image de l'ordinateur merveille du monde moderne c'est largement répondue. S'il est un mythe qu'il faut dénoncer, c'est bien celui-là.

On ne dira jamais assez la somme d'effort et d'ingéniosité q'uil a fallu à des dizaines d'ingénieurs pour établir les applications les plus spectaculaires tant vantées par la presse.

Le temps d'écriture et de mise au point de certains programmes se chiffre en dizaine d'années, que dire des travaux d'avant garde comme deux de la NASA par exemple !

Si , à première vue l'ordinateur apparaît comme un"outil miracle" capable de résoudre tous les problèmes, des réalités moins souriantes s'imposent au second abord, que connaissent bien ceux qui ont un contact direct avec la machine.

Aux réalités d'ordre financières s'ajoutent celles d'ordre technique, car le souci de l'utilisateur sera de tirer de la machine le meilleur parti possible, or qu'a-t-il devant les yeux ? Un ensemble complexe de circuits electroniques dont l'utilisation optimale ne peut se faire manifestement qu'à partir d'une connaissance approfondie de l'ordinateur, connaissance qu'il n'est pas imaginable d'exiger des utilisateurs.

.../...

La machine idéale au contraire, est bien celle dont l'emploi ne demande qu'un minimum de technicité et de compétence en informatique afin de reporter tous les efforts sur l'analyse proprement dite des applications.

C'est pour venir à bout de cette double contradiction entre un niveau technique de plus en plus élevé et un emploi plus facile qu'ont été mis au point les systèmes d'exploitation (Software de la machine). Sur la série IBM existe un ensemble de systèmes d'exploitation et parmi eux nous pouvons considérer le sous-moniteur PLAN (adaptable sur 1130 et 360).

PLAN, sans prétendre résoudre tous vos problèmes peut vous être d'un réel secours.

En effet son aspect modulaire vous permettra de l'adapter à une grande variété de problèmes, son interprète de commande vous facilitera le dialogue homme-machine, son chargeur vous fournira la possibilité d'exécuter des programmes dépassant la taille de la mémoire allouée par le constructeur, enfin ses modules utilitaires vous permettront d'utiliser d'une façon rationnelle tous les périphériques mis à votre disposition par votre fournisseur.

Dans les pages qui suivent vont vous être exposé les différents aspects de PLAN, au dernier chapitre de nombreuses applications toutes testées sur IBM 1130 à l'Ecole Nationale Polytechnique d'Alger, mettrons en valeurs les points cités ci-dessus.

Nous espérons pouvoir vous être utile.

-=-=-=0=-=-=-

Première partie : LE LANGAGE

CHAPITRE I

GENERALITES

I-1 - Introduction :

Au début PLAN a été développé comme un outil pour aider l'ingénieur dans ses applications sur le système 1130 et ce pour une utilisation facile et variée. Il permet grâce à une seule commande de faire exécuter un ensemble de tâches logiques.

I-1.1- Inconvénients de PLAN :

Le principal inconvénient de PLAN réside dans la durée d'exploitation des commandes.

I-1.2- Avantages de PLAN :

L'aspect modulaire de PLAN facilite beaucoup l'implantation de nouveau problèmes et applications.

Un ensemble de routines permet à l'utilisateur l'exploitation de toutes les ressources mises à sa disposition.

Le principal trait interne de PLAN est son chargeur résident qui permet d'exploiter plusieurs programmes modulaires et ce sans aucune étude particulière des jonctions inter-programme.

Du point de vue "langage utilisateur", chaque personne peut définir ses commandes dans son jargon professionnel.

Le chargeur PLAN et son interprète permettent de remplacer le programme principal, en effet à toute commande de PLAN ces deux composantes gèrent les différentes opérations à executer.

L'utilisateur sous PLAN peut à tout moment connaître l'état de ses variables.

I-2- VUE GENERALE DU SYSTEME

I-2.1- QU'EST CE QUE PLAN ?

A cette question nous pouvons dire que PLAN est un système d'exploitation conversationnel.

PLAN est le seul programme principal utilisé, il a le contrôle du chargeur et de l'interprète, il sert à initialiser les commutateurs (SWITCHES) et à collecter le données utilisées par les différents modules indépendants de l'utilisateur.

PLAN permet à un utilisateur d'ajouter, de supprimer ou de modifier une phrase.

Les modules logiques d'application peuvent être écrits en FORTRAN IV mais en utilisant des appels aux routines de PLAN pour : la jonction, l'exploitation des fichiers, les entrées sorties séquentielles et pour certaines fonctions utilitaires ainsi que pour les sous-programmes du système IBM.

I-2.2- Procédures mises en place :

Les deux grandes étapes d'utilisation du PLAN sont les suivantes :

1 - Définition du langage

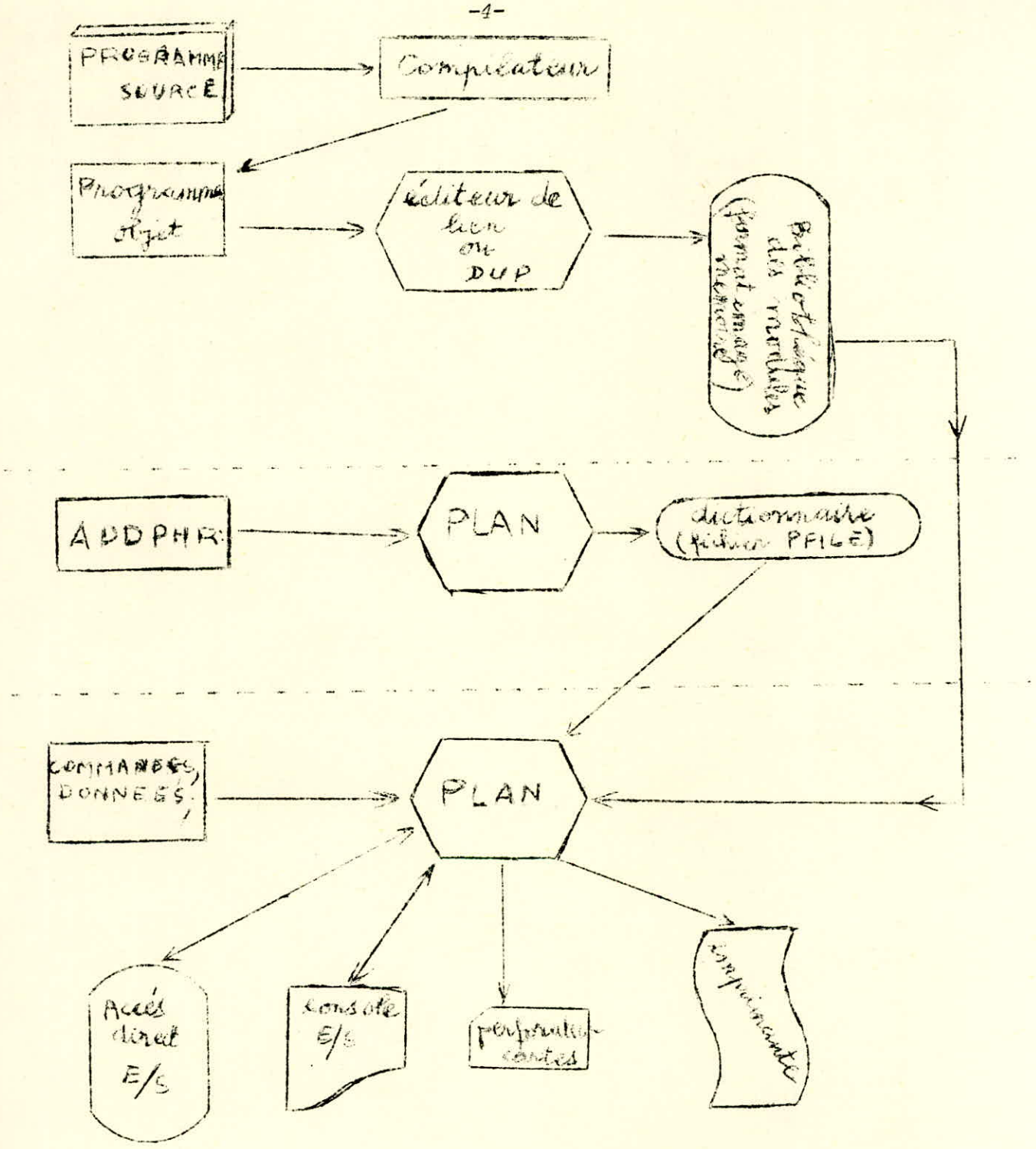
a - Chargement des modules indépendants dans le bibliothèque des programmes.

b) Addition de phrases dans le dictionnaire des phrases (fichier PFILE)

2 - Utilisation du langage

- Réception d'une commande et des données associées dans le langage utilisateur.

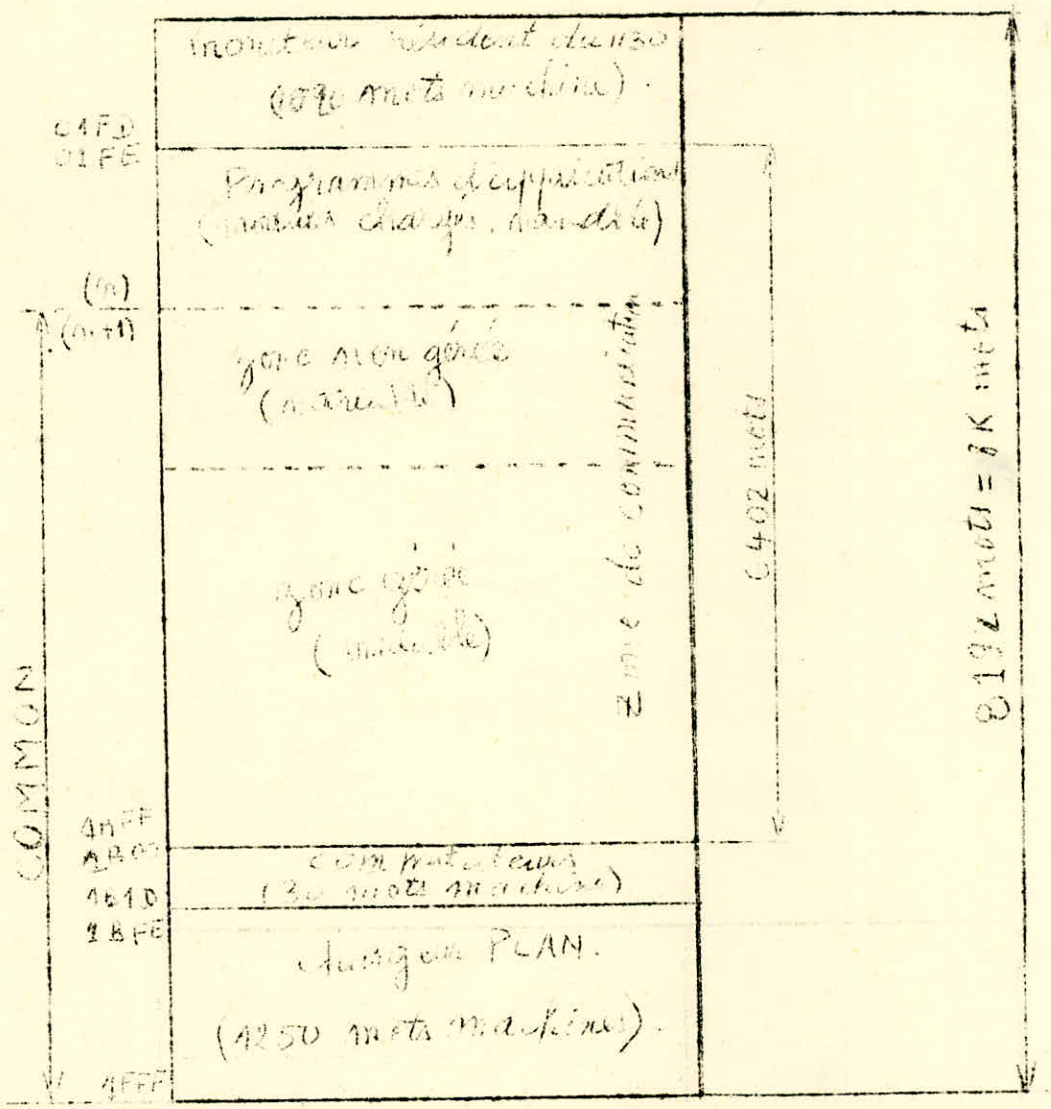
- Ces différentes opérations peuvent être schématisées de la façon suivante :



I-2.3- Carte de la mémoire :

Sous contrôle de PLAN la mémoire est divisée en 5 zones distinctes.

Cette répartition peut être schématisée comme ci-après :



Organisation de la mémoire de IBM 1130 sous PLAN.

I-2.4- Description des différentes zones :

1 - Zone système : Cette partie de la mémoire contient le moniteur résident; elle est recquise par le hardware et le système d'exploitation.

2 - Zone du chargeur PLAN : C'est la première partie du COMMON elle est en permanence occupée par le chargeur PLAN qui contrôle le chargement des programmes et l'exploitation des commandes.

3 - Zone des commutateurs (SWITCH WORD) : sur le 1130 nous avons 15 commutateurs d'un double mot chacun; ils servent de pointeurs et permettent la communication entre PLAN et les modules logiques.

4 - Zone de données gérées : zone protégée du recouvrement par les modules de PLAN, elle appartient au COMMAN. La zone gérée est variable.

5 - Zone de données non gérées : elle appartient au COMMON mais n'est pas protégée du recouvrement par les modules de PLAN. Elle peut servir au stockage des données; mais ne peut contenir ni le chargeur PLAN ni les commutateurs; sa taille est variable.

* On appelle zone de communication l'ensemble formé par les zones gérées et non gérée. (CA : Communication array).

I-2.5- Organisation fonctionnelle de PLAN :

L'organisation fonctionnelle de PLAN peut être résumée en 5 points :

- Supervision dynamique des travaux
- Exploitation du langage utilisateur
- Supervision des diagnostics
- Contrôle des entrées/sorties
- Utilitaires et fonctions de contrôle.

I-2.5.1- Supervision dynamique des travaux :

A moins d'utiliser les techniques d'overlay , il nous est impossible de résoudre au moyen d'un langage évolué (FOR, PL1, SL1...) un programme dépassant la capacité de la mémoire disponible des programmes.

Avec PLAN plus de problèmes d'allocation de mémoire en effet le chargeur PLAN prend le contrôle en fin de chaque programme modulaire ainsi nous n'avons qu'un module utilisateur en mémoire à exécuter; un programme peut être formé de plusieurs modules et le retour au chargeur PLAN se fait autant de fois qu'il y a de modules.

Le chargeur PLAN comprend 2 parties principales :

- Le programme chargeur
- La pop up list

Le chargeur PLAN est résident (2500 premiers bites du COMMON), il est protégé du recouvrement.

La pop up list est un mécanisme programmé pour l'exploitation des noms de programmes, pour le raisonnement nous la considérerons comme une liste de programme à exécuter.

Quand un nom de programme est déplacé de la pop up list un autre prend sa place au sommet jusqu'à ce qu'elle soit vide. Quand un nom de programme est ajouté à la pop up list, les noms existant sont décalés vers le bas jusqu'à son remplissage (50 au maximum).

Des noms peuvent être insérés au bas de la liste.

Les programmes à exécuter séquentiellement sont recensés du haut vers le bas de la liste ainsi le programme dont le nom se trouve au sommet de la pop up list est le prochain à charger en mémoire.

A tout moment l'utilisateur peut modifier la pop up list. Le chargeur PLAN note le nom au sommet de la pop up list, le charge en mémoire et lui donne le contrôle.

Le programme s'exécute (modifie si besoin la pop up list) puis rend le contrôle au chargeur.

La boucle se continue jusqu'à ce que la pop up list soit vide.

Les différentes options de l'utilisateur sont :

* Faire exécuter un module et donner le contrôle au chargeur PLAN
* Modifier la pop up list et retour au chargeur PLAN
* Modifier la pop up list, terminer le module en cours et donner le contrôle au chargeur PLAN.
* Modifier la pop up list, sauvegarder l'état du module en cours pour une étape ultérieure et retour au chargeur PLAN.

En plus du chargeur nous avons l'interprête (module P SCAN).

Le chargeur PLAN interpréte une pop up list vide comme un signal spécial lui ordonnant de mettre en mémoire l'interpréte qui prend en compte une nouvelle instruction utilisateur du flot d'entrée. Au décodage d'une instruction l'interpréte retrouve sa liste de programmes et la place dans la pop up list. Ainsi le langage utilisateur peut-il définir les séquences à exécuter.

* *Aucune nouvelle instruction n'est lue tant que la pop up list n'est pas vide.*

* *PLAN est conversationnel par le fait que l'utilisateur peut lire les résultats d'une commande avant l'exécution de la suivante. Les différéentes étapes de l'exécution d'un programme peut être résumée ci-dessous :*

  1°/ PLAN est chargé de la bibliothéque du système par le moniteur (carte de contrôle // XEQ PLAN)

  2°/ Pop up list initialisée à zéro (vide).

  3°/ Une fonction de PLAN détermine si le dictionnaire (PFILE) est initialisé.

  4°/ Pop up list vide : P SCAN est placé dans la pop up list, alors le chargeur met en mémoire (zone programme) l'interpréte de la bibliothèque du système.

  5°/ SCAN lit une commande, l'interpréte, si c'est : ADD, DELETE ou ALTER PHRASE nous avons modification du dictionnaire des phrases (fichier PFILE) et retour à (5), sinon aller à (6).

  6°/ Chercher dans PFILE la phrase correspondante, mettre les valeurs des données dans la zone de communication, empiler les noms des programmes associés à cette phrase dans la pop up list et donner le contrôle au chargeur PLAN.

  7°/ PLAN prend compte du premier nom de la pop up list et charge son programme associé dans la zone de programme de la bibliothèque. Le programme chargé, son nom et remplacé par le nom suivant de la pop up list.
  Le contrôle est donné au programme ainsi chargé.

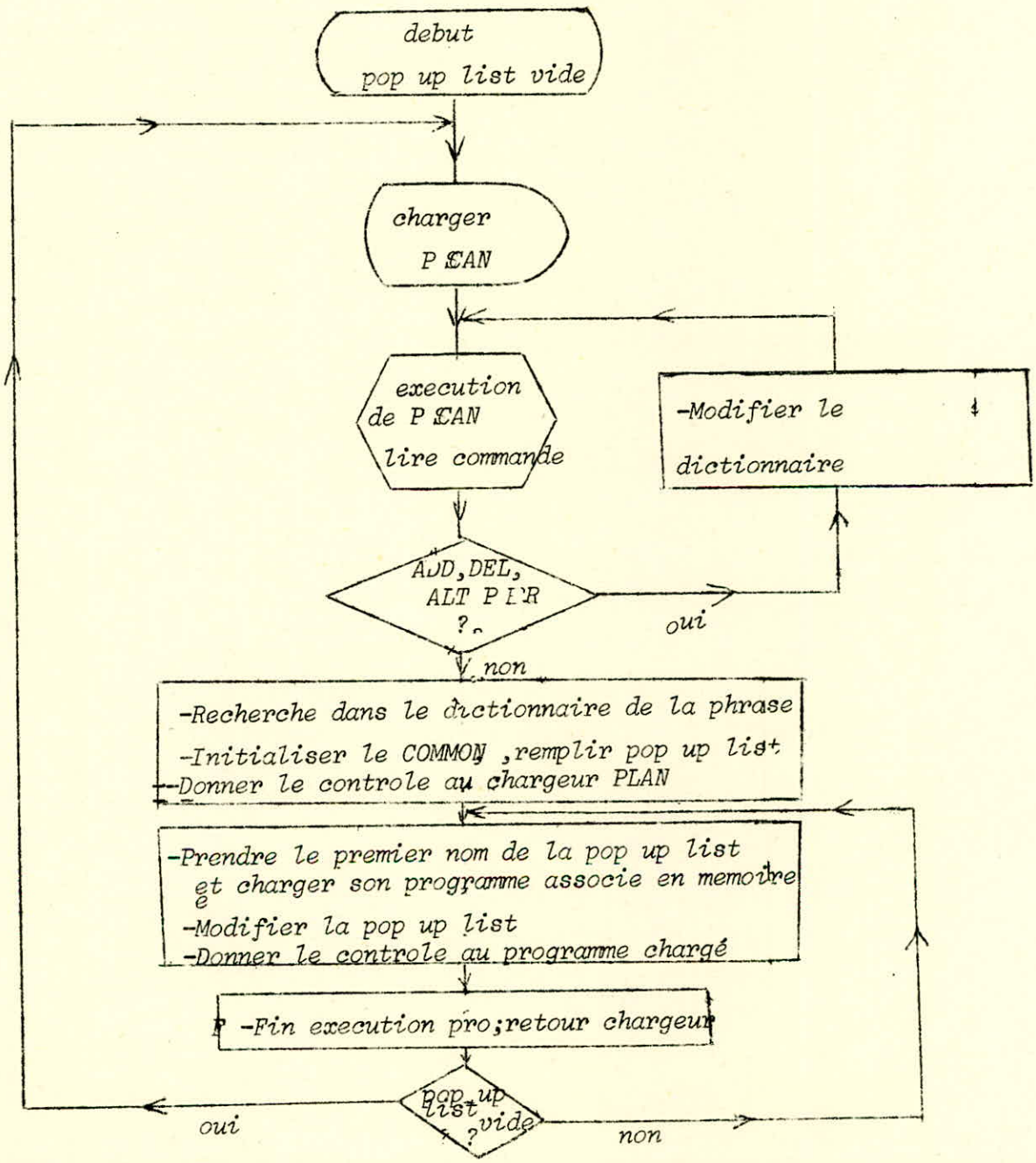
8°/ Le programme en mémoire s'exécute, il a accès aux informations stockées dans la zone de communication, et peut lui même modifier cette aire pour un usage altérieur.
Il peut utiliser les sous-programmes du chargeur pour une éventuelle modification de la pop up list.

9°/ Le programme après son exécution doit rendre le contrôle au chargeur PLAN, il y a test de la pop up liste alors 2 cas se présentent :

pop up list non vide aller à (7)

pop up list vide aller à (4)

L'organigramme suivant nous résume ces différentes phrases.



I-2.5.2- Exploitation du langage utilisateur :

Pour définir son langage de programmation l'utilisateur utilise l'instruction : ADD PHRASE.

Le format de cette instruction est :

ADD PHRASE : nom (, définitions) ;

Nom : nom de l'instruction à définir (5 mots au plus)

Définitions : niveau, liste des programmes, définitions des données valeurs de défaut, mode, facteur d'échelle, expressions arithmétiques, formules logiques ou arithmétique etc...

Une commande utilisateur serait :

nom de l'instruction, zone de données;

Exemple : RESOUDRE BAIRSTOW DEGRE 3, COFF5, 2,3,1;

A la lecture d'une telle commande l'interpréte cherche la phrase RES BAI dans le dictionnaire des phrases, il met les valeurs 5,2,3, et 1 dans des mémoires appropriées, il empile les noms des programmes associés à cette phrase dans la pop up list puis donne le contrôle au chargeur qui s'occupe de la mise en mémoire des différents modules associés à cette instruction.

Une fois la commande exécutée on aura à l'Unité de sortie les solutions de l'équation : $5x^3 + 2x^2 + 3x + 1 = 0$

Evidemment ceci suppose que cette phrase existe dans le dictionnaire des phraseset que ces modules associés on été chargé dans la bibliothèque (zone UA du disque).

I-2.5.3- Supervision des diagnostics :

Le succès d'un système d'exploitation est énormément conditionné par la richesse de ses diagnostics d'erreur.

PLAN alloue une grande part à cette tâche, il met à la disposition de l'utilisateur un ensemble de routines qui permettent la sortie des messages d'erreur prédéfinis.

I-2.5.4- Contrôle des Entrées/Sorties

Les E/S ont toujours posés des problèmes délicats à résoudre. Les difficultés sont surtout dues aux configurations différentes des ordinateurs.

PLAN alloue à l'utilisateur un ensemble de buffers auxquels sont associés des routines de manipulations; ainsi grâce à ces zones tampon l'usager peut facilement changer d'unité et ce aussi bien à la lecture qu'à l'écriture.

I-2.5.5- Utilitaires et fonctions de contrôles

PLAN met à la disposition de l'utilisateur une grande variété de routines permettant de faciliter la programmation.

Les principales fonctions permises par ces sous-programmes sont :

- les test, la lecture, l'écriture, l'exécution de commandes, recherche de fichier, modification des unités d'E/S , contrôle des diagnostics d'erreur, manipulation des buffers et de la pop up list etc...

C H A P I T R E II

LE LANGAGE PLAN

II-1- Introduction :

Définir le langage de l'utilisateur est une des premières opérations à s'affranchir.

Le langage utilisateur est stocké dans le dictionnaire de définition du langage (fichier PFILE) par l'interpréte (module P S CAN).

Une fois dans le fichier PFILE ces commandes sont immédiatement utilisables dans les programmes d'application.

Les données associées à cette commande se trouvent dans la zone de communication.

II-2- Terminologiedu langage :

II-2.1- Le mot :

Un mot est une suite de un ou plusieurs caractères alphabétiques non séparés par des blancs; mais en fait seules les 3 premières lettres sont prises en compte par P S CAN.

Exemple :	A	ABLE	ARROW	ARRA Y	ARRIVE
PLAN Lit :	Abb	ABL	ARR	ARR	ARR

Nous voyons donc que les 3 derniers mots sont identiques.

II-2.2_ La phrase :

Une phrase est une séquence de 1 à 5 mots séparés par des blancs. On distingue deux types de phrase :

- Phrase objet

C'est une phrase qui se suffit à elle-même.

- Phrase verbe :

C'est une phrase qui sert à modifier le sens d' une phrase objet, elle ne peut être utilisée toute seule.

Exemple :

phrase objet : RE S BAIR : permet de résoudre une équation
phrase verbe TEXT : permet d'imprimer un message
commande : TEXT RE S BAIR, TEX 'LA S OLUTION E ST', DEG 2,
CDE1,3,5;

Cette commande permettra de résoudre l'équation $x^2 + 3x + 5 = 0$
et imprimera en tête le message LA S OLUTION E ST

II-2.3 - La commande :

Une commande est une séquence d'une ou plusieurs phrases, elle ordonne une tâche.

Une commande peut contenir au maximum 8 phrases verbes mais ne doit contenir qu'une seule phrase objet.

II-2.4- L'instruction :

Une instruction est une commande optionnellement suivie de données, elle ne peut contenir plus de 450 caractères.

Pourun enrégistrement d'entrée sur carte l'instruction occupe les colonnes 1 à 75.

Une commande est séparée de ses données par une virgule, les instructions sont délimitées entre elles par des points virgule. Les blancs ne sont pas pris en compte.

Exemple :

COMPAREZ, TAX 'ABDERRA MANE', TEX'ABDELLATIF'; S WITC H;
La première commande permet d'ordonner par ordre alphabétique les deux noms entre quote et la dernière d'imprimer les commutateurs.

II-2.5: Les données :

Les données

Une donnée peut être identifiér par un nom. On a 3 types de données.

- *numériques ex. DEF7, S N3*
- *littérales ex. WORD'X', LTX "ABC"*
- *logiques ex. VRAI+, FAUX-*

Si des blancs séparent le nom et la valeur de la donnée ils ne sont pas pris en compte.

Normalement les valeurs de données sont considérées en virgules flottantes mais les types virgules fixes et EBCDIC peuvent être spécifiés.

Les valeurs réservées aux données logiques sont :

FAUX 7FFFFFFF représenté par -

VRAI 80000000 représenté par +

II-2.6- Nom d'une donnée (DAN : Data Name)

Un nom de donnée est un mot qui symbolise la valeur contenue dans une position mémoire de la zone de communication.

Lors de l'utilisation d'une commande il peut-être indicé.

Exemple :

COM, $N2, $N(2), $N(3) 'ABC' ;

est équivalent à COM, $N2, 3, 'ABC';

* Ne jamais utiliser E seul pour nom d'une donnée (confusion avec exponentielle).

* Ne jamais utiliser U seul pour nom d'une donnée (confusion avec U des programmes de conversion).

II-2.7- Constante (NUV : Numerical Value)

Une constante est une valeur entière ou réelle signée ou nom. La valeur numérique d'une constante peut être modifiée par une exponentiation mais ne peut contenir de caractères blancs. Une constante peut être logique.

Exemple : 1, 1., -2.3, +3.14, 3E-3, -, +

II-2.8- Les littéraux ($LV. : Set Litteral Value)

Un littéral est une donnée alphanumérique comprise entre 2 quotes (ou signe)) ou 2 double quotes.

Dans le cas de quotes simples le nombre de caractères du littéral occupe le premier mot référencé, dans l'autre ce nombre est omis. Par définition nous appelerons littéraux PLAN ceux définis par la première manière.

Chaque caractère du littéral occupe 1 byte (8 bits). Ainsi le nombre de mots occupés par M caractères entre quote est :

quote simple : IFIX (1+(M+3)/4)

quote double : IFIX((M+3)/4)

IFIX : permet de prendre la partie entière (FORTRAN 1130)

Exemple :

TITRE 'IBM 1130 PRO PLAN'

TITLE "IBM 1130 PRO PLAN"

En mémoire nous aurons :

TITLE	IBM	1130	PRO	PLA	N	(17+3)/4=5

TITRE	17	IBM	1130	PRO	PLA	N	(17+3)4+1=6

II-2.9- Opérandes arithmétiques (AOP : Arithmetique operand)

Un opérande arithmétique est constitué de termes séparés par des opérateurs. Les termes peuvent être des noms de données ou des constantes.

Les opérateurs dans l'ordre d'évaluation hierarchique sont :

* / + - avec leur sens usuel.

Des termes de différents modes peuvent coexister mais l'évaluation se fait en flottant et est arrondie avant stockage en mémoire par une troncature.

Exemple :

B + 3 - (C + 2 * A/B)

II-2.10- Expressions arithmétiques (AEX : arithmétique expression)

Une expression arithmétique est un opérande arithmétique précédé du signe = (égal).

Une expression arithmétique implique un stockage de données.

Une expression arithmétique est évaluée en mode flottant; il y a conversion éventuelle en mode fixe après calcul. Le résultat est stocké dans le mode indiqué par le nom ou le pointeur associé à cette expression.

exemple : A = 20, B = A *.017453296,

Cette expression permet de convertir 20 degrés en radians.

II-2.11- Opérandes logiques (LOP : Logical Operand)

Un opérande logique peut être un nom de donnée ou une opération de relation entre parenthèses.

Une opération de relation est constituée par un opérateur logique (> , < , =) suivit et précédé par des opérandes arithmétiques.

Suivant que l'assertion entre parenthèse est vraie ou fausse on a pour valeur logique VRAI ou FAUX.

Grâce à des masques on peut faire des tests logiques.

Exemple 1 : (B + 5 > A - D)

Exemple 2 : (DATA = "S bbLb")

permet de tester la première et le quatrième caractère de la donnée DATA.

II-2.12- Expression logique (LEX : Logical Expression)

Une expression logique est un opérande logique précédé du signe :

Une expression logique implique un stockage de donnée.

Une expression logique n'admet que 2 valeurs, VRAI ou FAU

Une expression logique peut contenir des opérateurs logiques qui dans l'ordre d'évaluation sont : ¬ , | , &

NON OU ET

Exemple : OPEN +, CLO &E-, A : ¬¬OPEN &¬ OPEN | ¬ CLO &E &OPEN,

Dans l'ordre d'évaluation nous avons :

- A : OPEN &CLO &E | OPEN &OPEN,
- A : CLO &E | OPEN
- A : OPEN

finalement A admet la valeur logique VRAIE.

II-3-Définition du langage PLAN :

Pour mettre à profit PLAN l'utilisateur se sert d'instructions.

Les phrases associées à ces instructions doivent être préalablement définies dans le dictionnaire.

Nous appelerons langage PLAN l'ensemble des instructions qui permettent d'exécuter des tâches.

Le langage PLAN se trouve dans le fichier PFILE (dictionnaire).

Les commandes ALTER, DELETE et ADD PHRASE permettent respectivement de modifier, d'effacer ou d'ajouter une phrase utilisateur du dictionnaire. Tout phrase ajoutée peut immédiatement être exploitée.

II-3.1- ADD PHRASE :

Format général :

ADD PHRASE : nom de la phrase, données;

Cette commande standard permet d'ajouter une nouvelle phrase dans le dictionnaire des phrases.

II-3.2- DELETE PHRASE

Format général :

DELETE PHRASE : nom de la phrase;

Cette commande standard permet d'effacer une instruction du dictionnaire des phrases.

Remarque : Une phrase verbe doit être signalée par le mot clé VERB

DELETE PHRASE : NOM, VERB ;

II-3.3- ALTER PHRASE :

Format général :

ALTER PHRASE : nom de la phrase, données;

Cette commande standard est une combinaison de DELETE et ADD PHRASE.

En cas d'erreur nous avons une erreur sur DELETE PHRASE.

II-3.4- Les données :

Lors de la définition d'une phrase nous avons un ensemble d'informations à fournir pour caractériser son instruction associée.

Les principales données à fournir sont :

II-3.4.1- Référence à une position dans la zone de communication (CAP):

Nous disposons de cinq moyens d'adressage pour pointer un mot de la zone de communication :

a- Constante entière

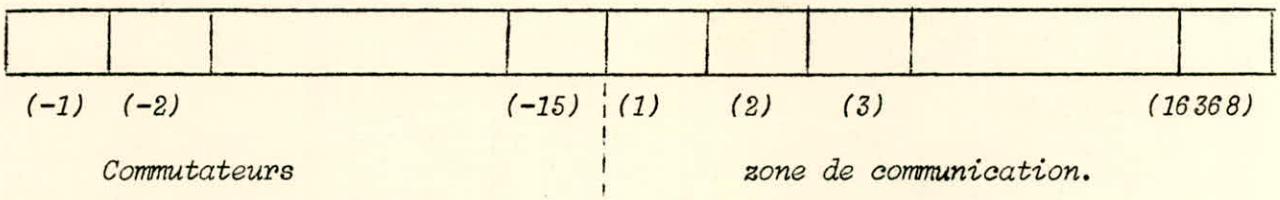
Un CAP (Communication array position) peut être défini par une constante entière entre parenthèses.

Format général :

(n)

n : entier compris entre 1 et 16368, qui indique une adresse relative dans la zone de communication.

Les commutateurs (SWITCH) sont répérés par des constantes négatives comprises entre -1 et -15.



Exemple : (3) permet de référencer le troisième mot de la CA
(CA : communication array)

(-7) : permet de pointer sur le 7ième commutateur.

* Il est possible d'évaluer une adresse symbolique grâce aux 2 caractères S'

Exemple 1 : (2) A implique : 2 = S'A
Ainsi (S'A) permet de référencer- le CAP2

Exemple 2 : si (1) A alors (S'A+10) pointe sur le CAP11

b- DO implicite :

Format général :

(I1, I2, I3)

I1 : première référence dans la CA

I2 : dernière référence dans la CA, ne peut être négatif

I3 : incrément, s'il est omis il est pris égal à 1.

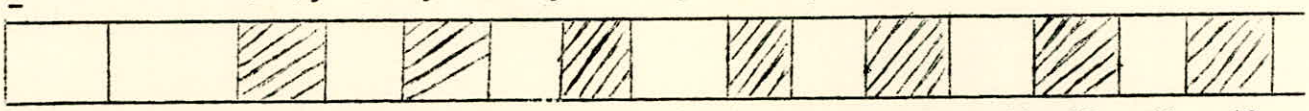
* (I2-I1) doit être divisible par I3.

Exemple :

(-7, 4, 2)

Les positions pointées sont :

(-7), (-9), (-11) , (-13), (-15), (2) et (4)



(-1) (-2) (-7) (-8) (-9) (-10) (-11) (-12) -13 -14 -15 -16 -17 -18 -19

c - Opérande arithmétique :

Format général :

(opérande arithmétique)

*Exemple : Si M et N sont connus; par exemple M = 2, N=4 alors : (M+2-3*N) permet de pointer le CAP-8*

d- Opérande arithmétique et DO implicite :

Format général :

(AOP , I2 , I3)

I1 : est remplacé par un opérande arithmétique

I2 doit être une constante entière divisible par I3.

** I1 est évalué à l'exécution.*

Exemple :

(M+2,10,2)

Si le contenu du mot adressé par M est 30 alors les positions référencées sont :

32, 34, 36, 38, 40, et 42

** I2 n'est pas l'adresse relative du dernier mot référencé de la zone de communication.*

e- Nom d'une donnée :

Pour référencer un mot de la zone de communication on peut utiliser un nom, mot formé de caractères alphabétiques.

Format général :

(n) NAME,

n : entier au plus égal à 8176

NAME : nom associé au CAPn

Ce mot doit vérifier les régles suivantes :

- A un mot de 32 bits ne peut être associé qu'un nom

- Il peut être une suite de plusieurs caractères non séparés par des blancs, mais en fait seuls lestrois premiers sont pris en compte.

- Il ne peut être la lettre E ou U seul, car on a confusion avec l'exponentiation et les programmes de conversions de données respectivement.

Exemple :

(10) ADR, ADD, (20)A

ADR : nom associé au CAP10

ADD : nom associé au CAP11

A : nom associé au CAP20

II-3.4.2- Valeur d'une donnée et mode :

Une donnée peut avoir 3 types :

- numérique
- littérale
- logique.

a- Valeur numérique :

La valeur de la donnée est placée à droite de la référence à sa position dans la zone de communication.

La valeur numérique peut être signée ou non, elle peut être une constante entière (virgule fixe), réelle (virgulle flottante) ou avec facteur d'echelle.

Contrairement au FORTRAN le nom de la variable ne donne aucune information sur son mode.

Dans le cas d'aucune spécification particulière elle est prise en réelle et occupe un mot de 32 bits.

Dans le cas où on référence une position dans la C.A. sans spécification de sa valeur on aura dans le mot pointé la valeur de défaut VRAI (+).

Exemples :

Définition de la phrase :

ADD PHRASE : DATA, I(20) ENT120, 1.3, P-1(24)EC H 5;

utilisation de la commande

DATA;

après exécution de cette commande on aura :

CAP20 contient 120

CAP21 contient 1,3

CAP24 contient $5 \times 10^{-1} = 0,5$

* *Le facteur d'échelle :*

format général;

P± n (CAP) NAME $\quad -7 \leq n \leq 7$

Le contenu du mot d'adresse relative CAP de la zone de communication est multiplié par une puissance de 10.

Exemple :

P-2(30) VAL 5

Le contenu du CAP30 est : $5 \times 10^{-2} = 0,05$

* *Le mode :*

Dans le cas où aucune indication de format n'est donnée, la valeur numérique prise sera en virgule flottante. Pour avoir le format virgule fixe on utilise le caractère alphabétique I.

Format général :

IP± n(CAP)NAME d

Exemple :

IP+2(20)ENT 1.352

interprêté de la façon suivante :

- multiplication par le facteur d'echelle 1,352 ⟶ 135,2

- conversion en entier : 135,2 ⟶ 135

- Stockage en mémoire 135 ⟶ CAP20

b- valeur littérale :

Une donnée peut avoir une valeur littérale, suite de caractères alphabétiques. La valeur littérale doit être placée entre quotes (ou signe $).

L'adressage de la position de la donnée dans la zone de communication se fait d'après les procédés définis précédemment (II-3-4-1).

exemples :

Définition de la commande :

ADD PHRASE : LITERAL, (1) ABC "ABC", (3,7,2) X"XYZT",
(10)LIT'PLAN';

utilisation :

LITERAL;

après exécution de cette commande nous avons :

	ABCb		XYZT		XYZT		XYZT			4	PLAN		
(1)	(2)		(3)	(4)	(5)		(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)

c- Valeur logique :

On peut définir deux valeurs logiques :

VRAI : + (80000000 en EBCDIC)

FAUX - (7FFFFFFF en EBCDIC)

Les régles d'adressages d'un mot sont celles définies en II-3-4-1.

Exemple :

ADD PHRASE : LOGIQUE, (2) TRUE +, (3) FALSE-;

utilisation :

LOGIQUE ;

après exécution de cette commande nous avons

CAP2 contient la valeur logique VRAI

CAP3 contient la valeur logique FAUX

II-3.5- Liste de programmes :

Une liste de programmes est une suite de tâches à effectuer pour satisfaire à l'exécution d'une commande.

Nous savons qu'à la lecture d'une commande, l'interpréte empile dans la "pop up list" un ensemble de programmes à exécuter séquentiellement.

Une commande doit contenir une phrase objet et au plus huit phrases verbes.

Une phrase sans liste de programmes ne modifie pas la pop up list.

II-3.5.1- Phrase objet :

Une phrase objet est une commande qui se suffit à elle même. La liste de programmes associée à une phrase objet est définie par le mot clé PRO-

Exemples :

ADD PHRASE : OBJET, PRO' MO7, MO8, INV';

Utilisation :

Objet;

A la rencontre d'une telle commande PSCAN empile les noms de programmes MO7, MO8 et INV dans la pop up list.

II-3.5.2- Phrase verbe :

Une phrase verbe est une commande incomplète, elle est utilisée pour modifiér le sens d'une phrase objet.

Une phrase verbe peut avoir deux types de listes de programmes définies respectivement par les mots clés =

VERB et PRO

Une phrase verbe doit être spécifiée par VERB pour être effacée par DELETE PHRASE

Les noms de programmes recensées dans une phrase verbe sous le mot clé PRO sont placé au sommet de la pop up list ceux nommés sous VERB occupent le bas de la pop up list.

Exemple :

phrase verbe :

ADD : SORTIE, VERB, I(203)TIP2, I(204)FOR158, I(210)-1,I(205)1, I(206)3, I(207)POSI, I(208)115100, I(209)100, VERB 'SORT';

Phrase objet :

ADD PHR : INT SIM,(1)AO,(2)BO, (3)RES, I(203)TIP2, I(204)FOR, 158, I(207)POSI, I(208)LIS 100, PRO'SOM';

utilisation :

INT SIM SOR , A1 ,B2 ;

A l'exécution de cette commande les modules dans la pop up list sont : , SOM et SORT.

SORT : module qui sert à sortir les informations de la

SOM : programme de calcul d'une intégrale

les 2 modules sont écrits en FORTRAN IV et stockées sur disque en format imagé mémoire. Les fonctions à intégrér doit avoir été stockées sur disque.

Les différentes valeurs contenues dans les CAP198 à 210 permettent d'initialiser certains paramètres des 2 premiers modules.
(pour une étude plus complète de ces modules se référer aux applications).

- Une phrase verbe peut modifier les données de défaut d'une phrase objet.

Exemple :

Phrase objet :

ADD PHRASE : VEHICULE A MOTEUR, I(1)AGE16, PRO'CAL';

Phrase verbe :

ADD PHRASE : BLIDA, VERB, I(1)AGE 18;

Utilisation :

BLIDA VEHICULE A MOTEUR;

Cette commande permettra d'exécuter le programme CAL avec pour valeur 16 à AGE au lieu de la valeur de défaut 18.

≠ La "pop up list" peut être modifiée par un programme de la liste.

II-3.6- Tests des données :

Pendant le traitement d'une commande (exécution de PS CAN) il est possible de tester le contenu d'un mot avant transfert à PLAN pour le chargement et l'exécution des programmes de la"pop up list"

Le format général du test est :

(N) TEST ACTION

N : adresse relative dans la zone de communication ($-15 \leq N \leq 8176$)

TEST : test à faire

ACTION : décision, résultat du test; si le test est négatif on exécute cette action dans le cas contraire on passe.

Exemple :

(23) ≠ TA'ERREUR'

- *CAP23 à la valeur logique + on passe*
- *sinon, la phrase est avortée et le diagnostic : ERREUR est imprimé.*

Les différentes tests et décisions prises sont :

	D E C I S I O N S								
TESTS	Aucun	'LIST'	C'DIAG'	A'DIAG'	P'PHR'	(N)	C(N)	A(N)	P(N)
#(NONFAUX)	1,4	3,6	3,5	1,5	2,3,7	3,9	3,8	1,8	2,3,10
#T(VRAI)	1,4	3,6	3,5	1,5	2,3,7	3,9	3,8	1,8	2,3,10
#F(FAUX)	1,4	3,6	3,5	1,5	2,3,7	3,9	3,8	1,8	2,3,10
#R(REEL)	1,4	3,6	3,5	1,5	2,3,7	3,9	3,8	1,8	2,3,10

C O D E

C O D E	Décision prise
1	- commande avortée - Le rétablissement de l'erreur de niveau PLAN est initialisée
2	- La dernière commande lue est exploitée
3	- La commande en cours est exploitée
4	- Le diagnostic PLAN 223-226 est généré
5	- Le diagnostic utilisateur "DIAG" est généré
6	- La liste de programmes définie par LIST est ajoutée à la pop up list
7	- La commande PHRASE est exécutée; la pop up list non modifiée
8	- Le diagnostic utilisateur localisé à la position (N) dans la zone de communication est généré
9	- La liste de programmes localisée à la position (N) est ajoutée à la pop up list
10	- La commande située à la position (N) dans la zone de communication est exécutée; pop up list non modifiée.

Nous voyons que les différentes décisions prises sont :

* *Modification de la pop up list :*

Deux options permettent de modifiér la pop list :

'LIST' et (N)

Exemples :

1) Définition de la phrase :

ADD PHRASE : AJOUTER, (7)A #(20), 12, 15, I(20)12, "PROGA", "PROGB", "PROGC";

Commande :

AJOUTE, A.

Cette commande permet d'ajouter PROGA, PROGB et PROGC à la pop up list. En effet le test est FAUX donc ajouter à la pop up list les programmes dont la longueur des noms occupe le CAP20 de la CA.

2) Définition de la phrase :

ADD PHRASE : DATA,(5) A# 'PROGA, PROGB';

Commande :

DATA;

La donné A n'ayant pas été fournie elle sera prise à la valeur logique VRAI donc le test est VRAI par suite on avorte la phrase; nous voyons donc que cette commande peut permettre de tester si une donnée a été donnée ou pas.

 # Génération de diagnostic et avortement de phrase :

Deux options permettent de générer un diagnostic et d'avorter une phrase :

A'DIAG' et A(N)

Exemple :

1) Définition de la phrase :

ADD PHRASE : TEST,(12) A#TA'LA DONNEE A N'A PAS LA VALEUR LOGIQUE VRAI', (50)FAUX#FA(72), (72)' LA DONNE FAUX N'A PAS LA VALEUR LOGIQUE FAUX;

Commande :

TEST, FAUX 12;

On aura avortement de la phrase et impression du message .

LA DONNEE FAUX N'A PAS LA VALEUR LOGIQUE FAUX

 # Génération de diagnostic et continuation d'une phrase :

Deux options permettent de générer un diagnostic et de continuer une phrase :

C'DIAG' dt C(N)

Exemple :

Définition de la phrase :

ADD PHRASE : TEST, (3) REEM ≠ RC' LA DONNEE REEL N'EST PAS REELLE', (29)VRAI ≠ TC(40)' LA DONNEE VRAI N'A PAS LA VALEUR LOGIQUE VRAI';

Commande :

TEST, REEL +, VRAI + 21

Deux test étant FAUX on aura impression des messages :

LA DONNEE REEL N'EST PAS REELLE

et continuation de la phrase

LA DONNEE VRAI N'A PAS LA VALEUR LOGIQUE VRAI

≠ Exécution d'une nouvelle phrase :

Deux options permettent à la fin de la phrase en cours d'exécuter une nouvelle commande :

P'PHRASE' et P(N)

Exemple :

Définition de la phrase :

ADD PHRASE : TEST, (-2) CAP ≠ RP'DUM SWI';

Commande :

TEST, CAP + ;

Cette commande permet de tester le contenu du CAP-2.

Le contenu de ce commutateur n'étant pas réel on aura un DUMP des commutateurs.

II-3-7- Les niveaux :

- *L'interprète PASCAL peut déceler 4 niveaux de phrase,*
- *Une donnée valable pour un certain niveau est utilisable par toutes les phrases de niveau inférieur.*
- *La notion de niveau permet de contracter l'espace des données dans une définition de phrase car toute information sera conservée pour les niveaux inférieurs.*

Régles :

1) *Les instructions de niveau 1 sont indépendantes des autres instructions*
2) *Les instructions de niveau 1,2,3 et 4 dépendent chacune du niveau qui leur est supérieur.*

3) L'exploitation de la structure de niveau permet de sauvegarder ou de faire suivre des données.

4) Uneerreur de niveau dans une commande crée le saut de cette dernière jusqu'à rencontre d'une phrase de même ou de niveau supérieur.

5) Le niveau 0 est réservé aux commandes du système.

ADD PHRASE, ALTER PHRASE, DELETE PHRASE, PLAN JOB...

6) Pour établir le niveau 0 il est conseillé d'utiliser la commande standard PLAN JOB.

7) Le nivdau "blanc" est le plus bas.

8) La zone de communication est recopiée sur disque si des phrases de niveau 1, 2 et 3 viennent d'être exécutées ou si une commande qui leur est inférieur est lue.

(Ceci n'est pas valable pour le niveau 4, car il n'y a aucune commande de niveau inférieur.)

* Une erreur de niveau 4 nous ramène au niveau 3.

Exemple :

Définition des phrases :

```
ADD PHRASE : A, LEV1, I(1), D, T0, V0, W0, R0, PRO'WRI';
ADD PHRASE : B, LEV2, I(1) S, T1, V1, W1, R1, PRO'WRI';
ADD PHRASE : C, LEV3, PRO'WRI';
ADD PHRASE : D, LEV4, PRO'WRI';
ADD PHRASE : E, PRO'WRI';
```

Utilisation des commandes :

	S	I	V	W	R
A;	0	0	0	0	0
B, S;	2	1	1	1	1
C,T2;	2	2	1	1	1
D,V3;	2	2	3	1	1
E,W4;	2	2	3	4	1
C,R5;	2	2	3	4	5
D,B6;	UNDEFINED SYMBOL IN INPUT STREAM				
D,V6;	PHRASE SKIPPED				
A,R7;	0	0	0	0	7

```
B,V8;      1   1   8   1   1
B,S10;    10   1   1   1   1
A,         0   0   0   0   0
D,S5;      5   0   0   0   0
B,V2;      1   1   2   1   1
```

* WRI est un programme qui permet d'imprimer les CAP 1 à 5.

II-3.8- Evaluation d'expression :

Les expressions arithmétiques et logiques nous permettent à partir des valeurs de données de générer des valeurs de données. Les expressions arithmétiques et logiques sont respectivement introduites par les signes = et :

Attention !! Si une donnée logique apparait dans une expression arithmétique alors le résultat est la valeur logique FAUX.

Exemple :

Définition de la phrase :

```
ADD PHRASE : EXP EVA, (1)X, Y, Z, C : ¬(¬X∨(¬Y ∈ Z))∧
(¬X ∈ ¬(¬(¬X ∈ ¬Z) ∈ ¬(X ∈ Y)), PRO'WRI';
```

utilisation de la phrase :

```
EXP EVA, X+, Y-, Z+;
EXP EVA, X+, Y+, Z+;
EXP EVA, X+, Y+, Z-;
EXP EVA, X+, Y-, Z-;
EXP EVA, X-, Y-, Z+;
EXP EVA, X-, Y+, Z+;
EXP EVA, X-, Y-, Z-;
EXP EVA, X-, Y+, Z-;
```

* On peut définir des valeurs arithmétiques ou logiques par défaut.

Exemple :

Définition de la phrase :

```
ADD PHRASE : CALCUL IMPEDANCE : (1)L, (2)C, (3) R, (4)OME; (5)
Z0= (R**2 +(L* OME-1/C*OME)**2)** 0.5;
```

Utilisation :

ØR CALCUL IMPEDANCE, 1,1,1,100,.....;

On aura alors le calcul de l'impédance Z pour L = 1, C = 1, R = 1 et OMEGA = 100.

ØR CALCUL IMPEDANCE,.....;

Z prend la valeur de défaut0.ù

II-3.9- Evaluation conditionelle :

Une valeur de donnée peut être générée à partir du résultat d'un test logique. Pour cela on utilise deux opérateurs et ce sous deux formes :

Première forme :

(n) NAME : (LEX 1) ? = AEX

(n) NAME : (LEX1) ? : LEX

Exemple :

Définition de la phrase :

*ADD PHRASE : TEST, (2)VALEUR : ((A<3) ET (A>1))? = 1+2*A;(5)T+,V-;*

(4) NAME : ((A > 3) OU (A 1)) ? : T ET V;

Utilisation :

TEST, A8;

NAME Prendra dans ce cas la valeur logique FAUX.

deuxième forme :

(n) NAME : (LEX1) ? = AEX ! = AEX

(n) NAME : (LEX1) ? = AEX ! = EX

(n) NAME : (LEX1) ? : LEX ! : LEX

par exemple pour le deuxième format nous avons :

si (LEX1) est VRAI alors NAME = AEX si non NAME : LEX

Exemple :

*ADD PHRASE : TEST,(52)L+, (3) M-, (4) N12, (6) NAME : (3*N < 5) ? :(L ET M) ! :(L OU M), (10) NOM : (L ET M)? = N * * 2, ! = N)*

I(-4)6, PRO'ECRI', I(-4)10, PRO'ECRI';

TEST;

d'où NAME : + et NOM = 12.

II-3.10- ZONE formule :

Une zone formule est un ensemble d'expressions référencées par un numéro de formule compris entre 0 et 1024.

Chaque numéro doit être précédé du signe dollard, ( ).

Une zone formule doit commencer par un numéro et se terminer par un point virgule (;)

Une zone formule doit se trouver après toutes les autres données d'une phrase (niveau, valeurs des données, liste des programmes...).

Les numéro 0 et ceux supérieurs à 1024 sont non référençable/

Les différentes composantes d'une zone formules sont :

- Evaluation d'une expression arithmétique :

  A = B * 100 + C

- Evaluation d'une expression logique :

  TRUE : (B/A ET C)

- Evaluation conditionnelle

  A : (B<100) ET (B>0) ? = 20 ! = 0

- Branchement conditionnel :

  NAME : (LEX) ? $ n ! $ m,

si LEX est VRAI aller à n sinon se brancher à m

  NOM : (A>5)? $ 3 ! $ 5;

- Branchement inconditionnel :

  : $n, aller à $ n.

- Melange des conditions :

  NAME : (LEX)? expression ! $ n;

  NAME : (LEX)? $ n !, expression;

Exemple : A : (B = "ABCD")? = 1000, $ 5;

Exemple : Soit à calculer le 50ième terme d'une progression arithmétique on a :

A1, S1, $ 1 S= S+3, A : (A=49)? $ 2 ! = A+1, : $ 1, $ 2;
Le résultat sera contenu dans S

II-3-11- Programmes de conversion qui permettent de prétraiter des données au cours de l'exécution de PASCAN.

Il est ainsi possible de définir jusqu'à trois programmes de conversion différents par phrase.

Exemple : convertir des kilomètres en mètres,
simple précision en double.

Lorsque P ŒAN rencontre un nom de donnée associé à un programme de conversion, ce dernier est appelé puis exécuté. Il existe un ensemble de sous-programmes mis à la disposition de l'utilisateur qui permettent de parcourir le flot d'entrée et de placer les données converties dans les positions de la zone de communication appropriées; Ce sont :

IU ŒR : début du programme de conversion

EU ŒR : fin du programme de conversion, retour à P ŒAN.

NU ŒR

GU ŒR : va chercher la caractère suivant à traiter.

* Un programme de conversion ne peut être utilisé pour tester une donnée dans un commutateur.

Un e donnée à traiter par un programme de conversion doit être spécifiée par : U m, ou m est le numéro du programme de conversion.
(m = 1,2 ou 3).

Format général :

U m I P ± n (CAP) NAMEd

Exemple : U2IP + 3(10) A7

La donnée A sera traitée par le programme de conversion numéro 2

* Les symboles I, P et n ne sont pas utilisés par le programme de conversion; ils ne sont utilisés par P ŒAN quesi une expression est utilisée par la suite.

Le programme de conversion est exécuté lorsqu'un nom de donnée associé à ce programme est rencontré par P ŒAN et que le caractère suivant n'indique pas le début d'une expression ou d'une constante littérale.

P ŒAN ne donnera donc pas le contrôle au programme de conversion si le caractère suivant le nom de la donnée est l'un des symboles : égal (=) quote (' ou "), deux points (:) point virgule (;) parenthèse (ou signe ↄ .

Etude des sous-programmes de conversion :

CALL IU ŒR : début d'un programme de conversion, il permet la liaison entre P ŒAN et ce module de conversion.

CALL GU ER (IC AR) : recherche le caractère suivant de la commande et le place dans le mot entier IC AR (en EBCDIC). On a zéro dans IC AR si le caractère rencontre est, ou ;
** Tout appel à GU ER ne sera pas pris en compte tant que l'on ne sera pas revenu à P CAN par EU ER.*

CALL NU ER (I UB, I W) : place le numéro du CAP courant dans I UB la première où il est rencontré dans le programme de conversion.

I W est mis à Zéro il est permis de stocker des données et est positif dans le cas contraire.

I W sera positif toutes les fois que le tableau référencé est trop grand ou si un programme de conversion est exploité pendant l'exécution d'un GO TO de la zone formule d'une commande.

Chaque appel à NU ER incrémente de 1 le numéro du CAP. Il faudra donc appeler (n+1) fois NU ER pour stocker n valeurs de 32 bits.

CALL EU ER (N1, N2, LIT) : rend le contrôle à P CAN :
si N1 = 0 : pas d'erreur
si N1 0 ; alors les paramètres N1, N2 et LIT sont utilisés pour un appel au module d'erreur ERRAT.

** Mot clé EXIT :*

Le mot clé EXIT permet de définir une liste de trois programmes de conversion.

Nous avons trois programmes de défaut; EXIT1, EXIT2 et EXIT3; ils sont exploités quand aucun programmes de conversion n'a été spécifié par l'utilisateur.

EXIT1 : conversion simple - double précision

L'utilisateur peut écrire deux programmes de conversion fréquemment utilisés en leur donnant les noms EXIT2 et EXIT3.

Exemple : EXIT' EXIT1, PROGA, EXIT3'
Si une donnée est U2(5)A3
Le préfixe U2 provoquera une conversion de la donnée A par le module PROGA.

Exemple :

Définition de la commande :

ADD PHRASE : NAME, U1(5) ABC

Utilisation :

NAME, ABC 7-4, 2-1;

Nous avons dans ce cas 2 appels au programme de conversion numéro 1, le premier se termine à la rencontre de la virgule, alors GUSER renvoie ICHAR = 0, un appel à NUSER permet de pointer sur le CAP suivant, on passe alors à ABC(2) pour une nouvelle utilisation du programme de conversion.

II-4- Utilisation du langage :

Dès leur définition les phrases du dictionnaire (fichier PFILE) sont exploitables par l'utilisateur qui pour définir ses tâches doit suivre certaines règles syntaxiques.

Le langage utilisateur est un ensemble d'instructions dont le format général est :

COMMANDE / DONNEES ;

II-4.1- La commande :

Une commande est une suite de phrase.

Elle doit contenir une phrase objet et au plus huit phrases verbes, chaque phrase est un ensemble d'au plus cinq mots dont seuls les trois premiers caractères de chaque mot sont pris en compte par PSCAN (interprete).

Si l'utilisateur omet une commande alors c'est la dernière demandée qui va s'exécutée avec de nouvelles données.

Exemples : COM, DON1, DAN2; , DON3;

La commande COM va s'exécuter avec pour données DON = 1 et DAN = 2, puis elle sera de nouveau exploitée avec DON = 3.

II-4.2- Les données :

Les différentes données associées à une instruction séparées de la commande et entre elles par une virgule, dans certains cas on peut l'omettre.

Les différentes composantes d'une donnée sont :

- NOM de la donnée
- Valorisation de la donnée
- Calcul d'une expression arithmétique ou logique.
- Utilisation des numéro de formules
- Indicage des données.

Remarque : Table des symboles :

Les noms de données se trouvent dans des tables de symbole

A chaque niveau est associé une table de 126 symboles au maximum. Les symboles associés aux noms de données sont construits au moment de l'exécution de la commande et ce en respect de la règle des niveaux.

Toute table de niveau inférieur peut être initialisée par celle d'un niveau supérieur;

Ainsi la table du niveau 2 est initialisée par celle du niveau 1 celle du niveau 3 par celle du niveau 2 et ainsi de suite.

Exemples :

1) ADD PHRASE : DON, LEV1,(5) A;
ADD PHRASE; DAT, LEV2;
DON;
DAT, A10;
On aura dans ce cas CAP 5 à la valeuf 10.

a) Calcul d'une expression arithmétique ou logique :

Lors du calcul d'une expression arithmétique toutes les variables utilisées doivent avoir été valorisées au moment de l'utilisation.

Exemple 1 :

Définition de la commande :

ADD PHR : CALCUL, LEV1,(100) RES, I(204)FOR126, I(207) POS 3, I(209) RTU100, I(206)100, I(210) SAU, I(203) TYP4, I(205)1, I(208)UNI100, I(1)TITRE'RES', PRO'SORT';

Utilisation :

EXP CALCUL, TITRE'SURFACE CERCLE R = 12.51':, RES= 3.1416 (12.51 * 12.51), POS24, SAUO ;*

L'execution de cette commande nous imprimera sur l'unité courante (en générale imprimante console).

SURFACE CERCLE R = 12.51 :

Remarques : EXP est une phrase verbe qui permet l'impression d'un titre.

- SORT est un module défini ultérieurement qui permet l'impression de certains mots de la zone de communication.

Exemple 2 :

ADD PHRASE : ONE,(20)A, (30)B;
ONE, 10 ;

La valeur 10 sera mise en virgule flottante dans la position 20 de la CA.

ADD PHR : TWO, (-8)ECO, (40) B;

TWO, 40.3;

La valeur 40.3 sera mise dans le CAP40 (car (-8) pointe sur un commutateur).

ADD PHRASE : THREE, (-2)DEUX, (5)AB, (20)PQR10, (8) MN2, (10)10;

THREE, 3, 5, 6, PQR12;

Après exécution de cette commande nous avons :

NOM	contenu du mot	CAP
AB	3	5
	5	6
	6	7
PQR	10	20
MN	2	8
	10	10

ADD PHRASE : STE, 13, (6)AB7, (23) PQR;

STE;

A l'execution de cette commande CAP1 contient 13.

ADD PHRASE : STE;

STE,13;

A l'exécution de cette commande CAP1 conteint 13.

ADD PHRASE : STE, 13, (-6) ABC, (23) PQR6;

STE, 5;

CAP1 et CAP23 contiennent respectivement les valeurs 13 et 5 ç k'exécytuib de cette commande.

ADD PHR : STE, 13, (-6) ABC;

STE, 5;

Dans ce cas CAP1 contient la valeur 13 car le commutateur 6 ne peut la recevoir.

ADD PHRASE; DONNEE, (6)XGS, YSG;

Les commandes ci-dessous sont équivalentes, elles permettent toutes de mettre les valeurs 50 et 60 dans les CAP6 et 7 respectivement.

DON, XGS 50, YSG 60;
DON, XGS 50 YSG 60;
DON, XGS 50, 60;
DON, XGS 50, XGS(2)60;
DON, XGS 50 60;

b) Les numéro de formules :

Il est possible d'utiliser les numéro de formules lors de l'appel à une commande.

Exemple :

Soit à calculer le cinquantième terme d'une progression arithmétique de raison 3 et de premier terme 2.

Définition de la commande :

ADD PHR : SOM, I(1)A, S;

Utilisation :

SOM, A = 1, S=2 $1 S=3, A : (A= 49)? $2 ! = A+1, : $ 1, $ 2;

A : permet de répérer l'ordre du terme de la profession
B : contiendra le terme demandé.

c) L'indicage :

L'utilisateur peut faire usage d'une variable indicée lors de l'exploitation d'une commande.

On distingue deux type d'indicage.

1- Premier type :

Format général ;

NOM(m) m : entier naturel positif

Exemple :

Définition de la phrase :

ADD PHR : DATA, (3) ADR, BETA, NBR;

Les deux commandes ci-dessous sont équivalentes :

DATA, ADR3.14, NBR 15;
DATA, ADR3.14, ADR(3)15;

2- Deuxième type :

Format général :

NOM (I,J,K)

I : indice initial

J : Indice final

K : pas d'incrémentation

Exemple :

Définition de la commande :

ADD PHRASE : DATA,(1)A;

Utilisation :

DATA, A(4,10,2)17;

Cette commande permet de mettre la valeur 17 dans les CAP 4,6,8 et 10 respectivement.

II-4.3- Instructions SAVE, EXECUTE et SEND :

On peut sauvegarder dans un fichier temporaire une suite d'instructions à exécuter ultérieurement.

L'identification d'une instruction sauvegardée se fait par un numéro compris entre 1 et 32767 et par usage des commutateurs 1, 2 et 3.

Le format général des instructions sauvegardées est :

n COMMANDE, DONNEE $ où n désigne le numéro de l'instruction

La sauvegarde des instructions peut se faire implicitement ou explicitement. La commande standard SAVE amorce la sauvegarde, L'instruction SEND (save end) la termine.

La sauvegarde peut aussi se finir sur la rencontre d'un autre SAVE ou d'une instruction non numérotée.

Durant l'opération de sauvegarde les commandes ne sont pas exécutées; toute instruction à sauvegarder doit être numérotée.

Exemple :

```
SAVE, FILE 45, DRIVE 0;
  5 COMMANDE, DONNEE $
  6 COMMANDE, DONNEE $
  7 COMMANDE, DONNEE $
SEND;
```

FILE : définit le fichier dynamique utilisé pour sauvegarder les instructions.

- valeur de défaut contenu du commutateur 1

- Il n'est pas necessaire d'ouvrir le fichier

DRIVE : définit l'unité sur laquelle le fichier temporaire sauvegardé se trouve.

- Valeur de défaut ; contenu du commutateur 3 divisé par 2048.

Les instructions PLAN numérotés et non sauvegardées explicitement le sont implicitement, mais sont aussi exécutées.

ADD PHR et DEL PHR ne peuvent pas être implicitement sauvegardés si un numéro d'instruction est le même que celui d'une autre prête dans le fichier, c'est la nouvelle qui remplace l'ancienne.

Toute erreur décelée par PLAN lors de la sauvegarde supprime cette derniere.

DAns le cas où aucun numéro de fichier n'est désigné on aura utilisation du fichier 254 sur l'unité 0 (unité courante).

Le test "P" (≠ P'PHRASE') demande une sauvegarde implicite de l'instruction PHRASE.

- Pour faire exécuter les instructions sauvegarées on utilise la commande EXECUTE.

Exemple :

EXECUTE, FROM5 TO 10, FILE 45, DRIVE 1;

FROM : indique le numérole plus bas de l'instruction sauvegardeée à exécuter.

TO : donne le numéro le plus haut de l'instruction sauvegardée à exécuter.

FILE et DRIVE : indiquent respectivement les numéro du fichier temporaire et de l'unité contenant les instructions sauvegardées à exécuter.

(valeur de défaut : FILE : contenu du CAP-1, DRIVE : contenu du CAP-3 divisé par 2048).

Le commutateur 2 pointe sur le numéro de la prochaine instruction à exécuter.

≠ L'ordre d'exécution des instructions sauvegardées ne suit pas forcément l'ordre de leur numéro, en effet le commutateur 2 peut être modifié lors de l'exécution de ces instructions.

Exemple :

Définition des phrases :

```
ADD PHRASE : GO, I(-2)TO
ADD PHRASE : IF, I(1)TEST, (-2) : TEST ? = TEST(2);
ADD PHRASE : DO, (5) LIT, I(3)A5;
```

Sauvegarde :

```
SAVE;
1     DO, 'THIS';
2     DO, 'THAT';
3     GO, TO 7;
4     DO, 'SOMETHNG', AO;
5 ----------------------
6 ----------------------
7     IF, : (A >4),4;
 END;
```

Execution :

```
EXECUTE, FROM 1 TO 7;
```

A la rencontre de cette instruction nous avons exécution des phrases sauvegardée; l'ordre d'exploitation de ces phrases est donné par le CAP-2.

GO : permet de faire un branchement inconditionnel vers une phrase sauvegardée.

L'ordre d'exécution de ces instructions est :

N° de l'instruction	CAP	nom	Contenu
1	3	A	5
	5	LIT	4,'THIS
2	5	LIT	4,'THAT'
	3	A	5
3	-2	TO	7
7	1	TEST	+
	2	TEST(2)	4
	-2	TO	4
4	3	A	5
	5	LIT	9,'SOME','THNG'
	3	A	0
5----------			
6----------			
7	1	TEST	—

II-4.4- Execution de PΣAN :

PΣAN est un module du système qui permet d'interpréter les instructions à exécuter.

Les différentes phrases de travail de PΣAN sont :

1- Recherche de la définition d'une phrase dans le dictionnaire des phrases PFILE.

2- Initialisation de la zone géné et ce en accord avec la règle des niveaux.

3- Les noms de programmes associés au mot clé VERB sont ajoutés à la pop up list puis viennent ceux définis sous le mot clé PRO.

4- Les tables de symboles sont initialisées.

5- Les valeurs définies dans les phrases verbes précedent celles des phrases objets.

6 -Conversion des données et stockage dans les positions spécifiées.

7- Evaluation des expressions définies sous ADD PHRAΣ

8 -Remplissage de la pop up list.

9- Analyse des tests et exécution des décisions résultantes.

Exemple :

Définition des phrases :

ADD PHRAΣ : POP UP LIΣT,(5)≠ R'PROGA',+,PRO'PROGB';

ADD PHRAΣ : MODIFICATION, VERB,(5)A,B,(255)T≠T'PROGC',
VERB'PROD', PRO'PROGE';

Utilisation :

· MODIFICATION POP UP LIΣT, A,B5,T3;

1) Exploitation de la phrase verbe

PROGC
PROGE
PROGD

car CAP255 non VRAI

définit sous VERB

Pop up list

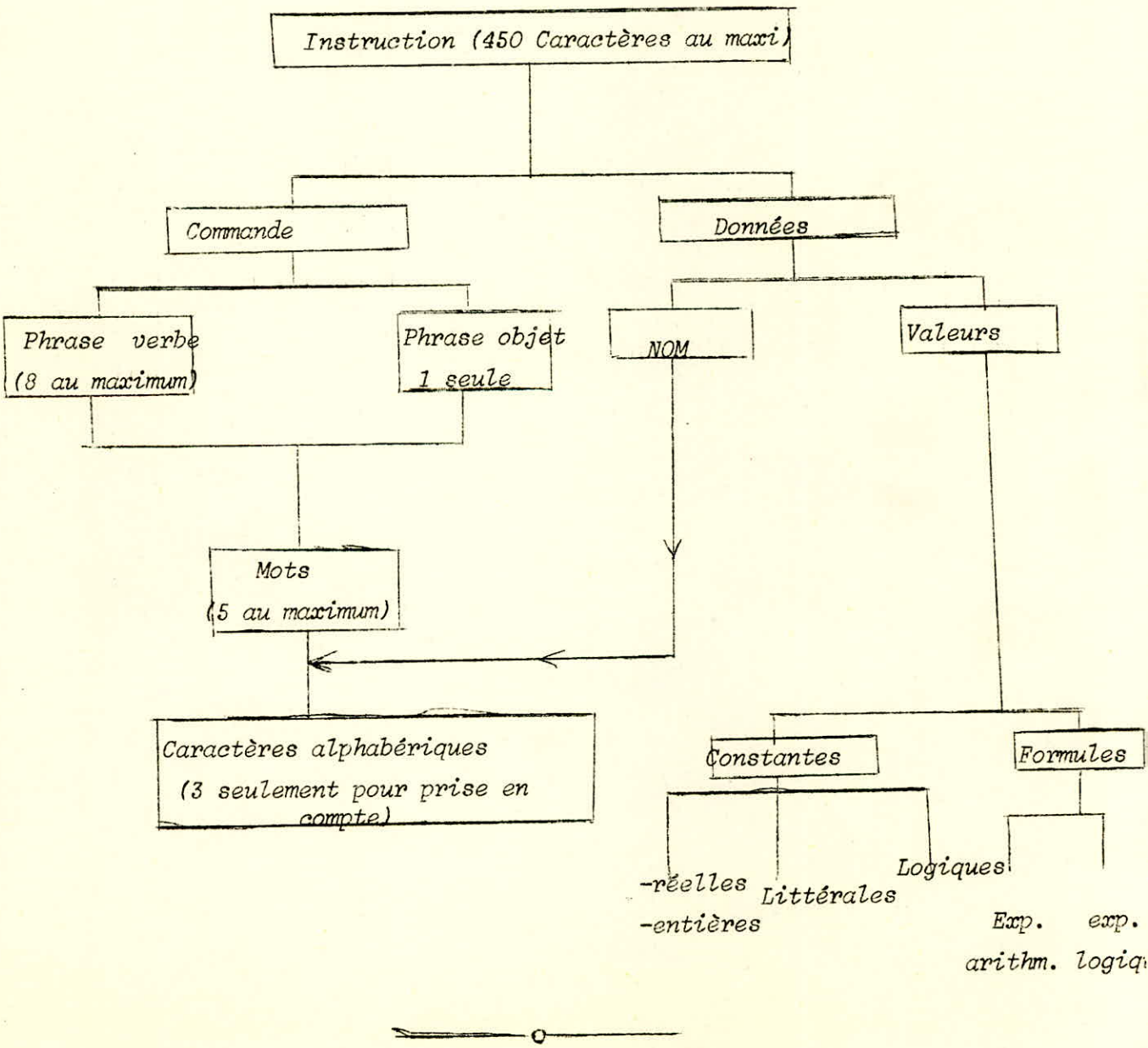
2) Exploitation de la phrase objet.

PROGC
PROGE
PROGA
PROGB
PROG D

car CAP5 non Réel

Pop up list

II-4-5- Syntaxe du langage :
La syntaxe du langage utilisateur peut être résumée par le graphe ci-dessous :



DEUXIEME PARTIE

La bibliothèque

Chapitre III. SOUS-PROGRAMME IBM :

III-1-Introduction :

Comme tout systeme d'exploitation PLAN admet une bibliothèque de sous-programmes, qui dans le cas du 1130 est stockée sur disque. (numéro de cartouche 1972 pour ENPA).

Ces routines peuvent être divisées en deux classes :

- Sous-programmes d'exploitation du système
- Sous-programmes utilisateur.

Les sous-programmes sont stockés en format image mémoire. L'appel aux routines du système se fait par l'ordre CALL.

III-2- Sous-programmes du chargeur :

Les routines du chargeur permettent à l'utilisateur d'avoir accès à la pop up list donc indirectement controler le chargeur; leur nom commence toutes par le lettre L.

Format général :

CALL XXXXX (N,L)

nom du S/p (5 caractères au maximum)

L : Liste de nom de programmes à ajouter à la pop up list ou emplacement où mettre ces noms retirés de la pop up list.

N : Nombre de mots occupéspar les noms de programmes à manipuler.

Chaque nom de programme occupe 2 mots PLAN.

$N > 0$: ajouter à la pop up list N/2 noms de programmes situés à partir de la position L.

$N < 0$: Retirer de la pop up list N/2 noms de programmes et les placer à partir de la position L.

* Si la valeur numérique zéro est rencontrée dans le premier mot du nom de programme, la pop up list est vidée.

* Si N est impair on a incrémentation automatique de 1.

III-2.1- Manipulation de la pop up list :

CALL LIST (N,L) : manipule la pop up list et retour à l'instruction suivant l'appel.

CALL LISTB(2,L) : Ajouter 1 programme au bas de la pop up list et retour au programme appelant après call.

CALL LEX (N,L) : Manipuler la pop up list, retour au chargeur PLAN pour le chargement des programmes suivant (programme en cours avorté).

CALL LCHEX(N,L) : Manipule la pop up list, sauvegarde du programme en cours d'exécution en vue d'une reprise ultérieure et retour au chargeur PLAN. (Ne pas oublier de réserver le fichier PCHPT pour le programme principal; voir exemple).

COMMON non affecté. Le programme ainsi sauvegardé sera rechargé à la rencontre d'un astérisque dans la pop up list et sera utilisé pour les instructions suivant CALL L C HEX (N,L).

CALL LOCAL(N;L) : Manipule la pop up list, charge le programme suivant de la pop up list et lui donne le contrôle, le retour au programme précedemment en cours se fait par CALL LRET dans le programme appelé par CALL LOCAL.
(Le retour se fait après CALL LOCAL).

III-2.2- Retour au chargeur PLAN

CALL LRET : permet le retour au chargeur PLAN, pas de modification de la pop up list, si cette dernière est vide on a lecture d'une nouvelle commande dans le cas contraire, c'est le programme dont le nom se trouve au sommet de la pop up list qui sera chargé et qui aura le contrôle.

CALL LNRET : permet le retour au chargeur PLAN dans le cas d'un programme appelé par CALL LOCAL.

CALL LREPT : permet la répétition de la commande en cours d'exécution, pas de modification de la pop up list.

Exemple : Par cet exemple nous allons illustrer les différentes phases d'exécution d'un programme sous-PLAN.

Etape 1 : Compilation d'un programme source et stockage sur disque en format image mémoire.

```
CALL START CARD (permet de charger le moniteur en mémoire)
// JOB (indique le début d'un travail)
// FOR (langage utilisé FORTRAN)
* LIST SOURCE PROGRAM
C  Réservation des zones en mémoires et données
        DIMENSION PLIST (4)
        COMMON L(625), LS(15), M(510)
        EQUIVALENCE (N,M(20)), (ABCD, M(21))
        DATA PLIST/'MO78', '3',  *       '/
                  :
                  :
                  :
        CALL LOCEX('PLIST')
                  :
                  :
                  :
        CALL LEX(N, ABCD)
        END
// DUP
*DELETE                    MO725
* STORCI   W S   VA        MO725
```

Etape 2 : Définition de la commande et rangement dans le dictionnaire (Fichier PERLE)

```
// JOB
// XEQ  PLAN chargement en mémoire du module PLAN
PLAN JOB, COMMON 510, MAN100, LONG; initialise le COMMON
ADD PHRASE : LOAD PROGRAM, I(20)NO, PRO'PRO'MO725';
IOCS INPUT1131, LIST 1131;
/*  fin de fichier carte (retour au moniteur)
```

A la rencontre de la carte IOCS INPUT1131, LIST1131 les unités 1131 (clavier console et imprimante) se mettent en attente pour recevoir des informations (clavier sélecté).

Etape 3 : Utilisation de la commande définiesous ADD PHRASE.
(on travail sur le pupitre 1131).

LON; initialiser le niveau à 1

LOAD PRO, N8"PROGA" "PROGB" "PROGC" "PROGD";

A la rencontre de cette commande nous avons exécution des phases suivantes :

1- Chargement du nom de programme MO725 dans la pop up list, puis exécution de ce module.

2- A la rencontre de L C EX on a =

- sauvegarde du programme en cours (MO725)
- Empiler dans la pop up list les noms de programmes :

MO788 et ≡

- Exécution des programmes de la pop up list donc exécution de MO788 puis à la rencontre de ≡ on a rechargement de MO785 et l'exécution se poursuit après l'appel à LC EX.

3- Rencontre de CALL LEX :

- Avortement de MO725
- Addition à la pop up list des noms de programmes :

PROGA, PROGB, PROGC et PROGD

4- Exécution de ces modules

5- Retour au chargeur PLAN, le pupitre se met en attente pour recevoir une nouvelle commande.

III-3- Sous-programmes d'entrées/sorties séquentielles :

Nous appellerons opération d'entrée/sortie tout processus visant à créer des échanges d'information entre la machine et le milieu extérieur.

III-3.1- Contrôle des E/S :

CALL IOCS (INPUT,LIST)

INPUT : numéro de l'unité d'entrée

LIST : numéro de l'unité de sortie

Cette routine permet de modifier les unités d'E/S courantes.

Les numéro des unités sur le 1130 sont :

ENTREE	SORTIE	LIBELLE
0		Unité courante d'entrée
	10C	Unité courante de sortie
1		Lecteur de cartes 1442
	101	Imprimante 1132
2		Lecteur de carte 2501
	102	Imprimante 1403
3		Clavier console 1131
	103	Imprimante console 1131
	104	Perforateur de carte 1442

Exemple :

1. Modification de l'unité d'entrée : CALL IOCS (3,100)
2. Modification de l'unité de sortie : CALL IOCS (0,101)
3. Modification des unités des 2 types : CALL IOCS (1,104)

Pour les exemples 1 et 3 les entrées se feront respectivement sur le clavier 1131 et le lecteur de carte 1442.
Pour les exemples 2 et 3 les sorties se feront respectivement sur l'imprimante 1132 et le perforateur de carte 1442.

III.3.2- Assignement de buffer (zone tampon) :

A toute opération d'entrée/sortie doit être assigné un buffer d'entrée/sortie, zone où doit se trouver l'information transférée ou à transférée.

Il existe deux types de buffers; les simples et les doubles.
Les routines permettant d'assigner un buffer du premier type sont :

CALL PSBFA (NOD)
CALL PSBFR (NOD)
CALL PSBFC (NOD)
CALL PSBFD (NOD)
CALL PSBFE (NOD)

NOD est le numéro associé à l'unité où se trouve le buffer (voir pour les numéro III-3.1 : contrôle des E/S.

Les routines permettant d'assigner un double buffer sont :

CALL PDBFA (NOD)

CALL PDBFB (NOD)

CALL PDBFC (NOD)

CALL PDBFD (NOD)

CALL PDBFE (NOD)

* La taille de buffer varie entre 80 et 120 caractères.

III-3.3- Lecture et écriture des fichiers séquentiels :

CALL { PLINP / PLOUT } (NOD)

CALL PLINP : permet de transférer un enrégistrement de l'unité NOD vers son buffer associé.

CALL PLOUT : permet de transférer un enrégistrement du buffer associé à l'unité NOD vers cette dernière.

* Les informations en dehors du buffer sont perdues.

* Un buffer est automatiquement "nettoyé" après une opération de sortie.

III-3.4- Test de l'état d'une unité :

L'instruction FORTRAN :

IF(PIOC(NOD))1,2,3

permet de tester l'état de l'unité NOD; on a branchement à :

1 si l'unité spécifiée est occupée

2 si l'unité NOD est invalide

3 si l'unité NOD est libre.

Un autre procédé consiste à faire attendre le programme appelant jusqu'à ce que l'unité spécifiée soit prête, ceci est obtenu grâce au module PBU SY.

Format d'appel : CALL PBU SY(NOD).

III-3.5- Sous-programmes de transfert inter buffer :

CALL PBFTR(NODF, NODT)

Ce module permet de transférer le contenu du buffer de l'unité NODF vers celui de l'unité NODT.

III-3.6- Contrôle d'unité :

CALL PCCTL(NOD,KTL)

Cette routine permet le contrôle d'une fonction d'E/S pour l'unité NOD.
KTL est le code de la fonction de contrôle.
Ce sous-programme doit être appelé avant utilisation de PLOUT Ou PLINP.
C'est toujours le dernier ordre de contrôle qui est pris en compte.

Codes de la fonction de contrôle

1.12 : saut du papier au canal KTL
0 : suppression d'espace avant impression
-1 : saut d'un espace avant impression.
-2 : saut d'un double espace avant impression
-3 : saut d'un triple espace avant impression
13 : utiliser le premier magasin
14 : utiliser le deuxième magasin
} lecteur/perforateur de cartes

III-3.7: Test de l'état de fin de fichier :

L'instruction FORTRAN :

IF(PEOF(NOD))1,2,3

permet de tester la fin de fichier.

NOD : code de l'unité
1 : indicateur de fin de fichier est 'ON'
2 : indicateur de fin de fichier physique est 'ON'
ou unité invalide (utilisé au lecteur/perforateur)
3 : sortie normale aucun indicateur n'est 'ON'.

- L'indicateur de fin de fichier logique se met 'ON' par lecteur d'un enrégistrement contenant UREND (positions 1 à 5) ou par analyse du canal 12 de l'imprimante; cet enrégistrement peut être accédé par les S/P de conversion.

- L'indicateur de fin de fichier physique se met 'ON' par lecture ou perforation de la dernière carte.

POur chaque test d'indicateur nous avons les actions suivantes :

URETND : initialise une phrase avortée et diagnostic
/* : fin de fichier physique on a retour au moniteur
(ou sort du contrôle de PLAN)
// : retour au moniteur et diagnostic.

Exemple :

Le module ci-dessous permet de transférer le contenu d'une ligne tapé à la console 1131 vers l'imprimante 1132 en sautant une ligne avant impression.

a- Ecriture du module et stockage sur disque :

```
// JOB
// FOR
* LIST SOURCE PROGRAM
C  RESERVATION DU COMMON
                COMMON L(625), LS(15), CA(500)
C  AFFECTATION DE S UNITE S
                CALL IOCS (3,101)
C  TEST CLAVIER CONSOLE ET IMPRIMANTE

           4    IF (PIOC(3)) 1,2,3

           1    IF (PIOC(101))4,2,4
C  AFFECTATION DE BUFFER
             3  CALL PSBFA (3)
                CALL PSBFB (101)
C  REMPLIR BUFFER ENTREE
                CALL PLINP (3)
C  TRANSFERT DE L'INFORMATION
                CALL PBFTR(3,101)
C  SAUT 1 LIGNE AVANT IMPRESSION
                CALL PCCTL(101,-1)
C  IMPRESSION A L'IMPRIMANTE
                CALL PLOUT (101)
C  RETOUR AU CHARGEUR PLAN
```

```
2 CALL LRET
  GOTO 2
  END
```

```
* DELETE                    TRANS
* STORECI   WS   UA         TRANS
```

b- Définition de la commande de transfert.

```
// JOB
// *COMMANDE DE TRANSFERT
// XEQ  PLAN
PLANJOB, ERA200, MAN510, LONG;
ADD PHRASE : TRA, LEU1, PRO 'TRANS';
IOCS, INP1131, LIS131;
/* **RETOUR AU MONITEUR
```

c- Utilisation :

Une fois le programme stocké sur disuqe les cartes définies en b) permettent de mettre en attente la console.

L'utilisateur tapera alors au clavier de la console

```
TRA;
VENDREDI LE 25 MAI 1973 (FDZ)
IOCS, INP1442;
```

après prise en compte de ces instructions nous avons impression de :

VENDREDI LE 25 MAI 1973

à l'imprimante 1132 et retour au moniteur.

III-4- Sous-programmes de conversion de données :

Un ensemble de sous-programmes de conversion permet à l'utilisateur de définir ses spécifications de traitement de variables.

Ces routines peuvent être réparties en deux classes;

Celles qui remplissent un buffer à partir de la mémoire et celles qui font subir à l'information le chemin inverse.

Les noms des sous-programmes de la première classe sont suffixées par les caractères "IN" et ceux de la deuxième par "OUT".

Une deuxième spécification donnée par ces routines est le format des données.

Le format général d'appel est :

$$CALL \left\{ \begin{matrix} PXIN \\ PXOUT \end{matrix} \right\} (NOD, J, NW, ARRAY)$$

X peut être un des caractères suivants:

I : type entier (utilisé en E/S)

F : type réel (utilisé en E/S)

A : type alphanumérique (utilisé en E/S)

E : type réel en simple longueur exprimés avec une mantisse et un exposant (utilisé en sortie seulement).

III-4.1 : CALL PXIN(NOD, J, NW, ARRAY)

Cet ensemble de sous-programmes permet de convertir une donnée prise d'un buffer et de la mettre dans la mémoire.

NOD : code de l'unité associée au buffer

J : position relative dans le buffer du premier caractère à convertir.

NW : nombre de caractères à convertir à partir de la position J.

Pour PXIN les dizaines et les centaines indiquent le nombre de positions à convertir et les unités le nombre de caractères numériques après le point décimal.

Nous voyons donc que NW nous renseigne sur le format de la donnée.

Exemple :

715 a pour format 3 et pour type I

785.321 a pour format 073 et pour type F

MESSAGE a pour format 7 et pour type A.

ARRAY : adresse dans la mémoire d'un mot PLAN où sera stockée la donnée convertie ce pour les type I et F pour le type A la donnée peut occuper plus d'un mot.

Régles de conversion :

PXIN : 1. Les blancs en tête du nombre sont ignorés

2. Le nombre peut être précédé des signes + ou -

3. Les caractères numériques sont collectés jusqu'à exploitation du format ou rencontre d'un signe particulier.

Exemples :

b-b156 = - 156

1-56 = 1

PFIN : 1. Les blancs en tête du nombre sont ignorés.

2. Les autres blancs sont pris égaux à zéro.

3. Un nombre peut être exprimé dans le format E.

Exemples :

bb+5b.b6 = 50.06

7bbb+b5 = 700000000

-.7E3 = - 700

1.E1bb = 10

PAIN : 1. Les caractères sont collectés et placés en mémoire en format A4(4 caractères EBCDIC par mot de 32 bits)

2. Les dernières positions du mot sont non affectées.

III-4.2 - CALL PXOUT(NOT, J, NW, ARRAY)

Cet ensemble de sous-programmes permet de convertir une donnée prise de la mémoire et de la mettre dans un buffer.

NOD : code de l'unité associée au buffer

J : position relative dans le buffer où le premier caractère convertie sera stocké.

NW : Nombre de positions occupée par la donnée après conversion. Pour PFOUT et PEOUT les dizaines et les centaines indiquant le nombre de positions occupées par la donnée convertie et les unités le nombre de caractères numériques après le point décimal.

Exemples :

17.014E-37 a pour format 103 et pour type E.

17.013 a pour format 063 et pour type E.

12452 a pour format 5 et pour type I.

ARRAY : adresse dans la mémoire d'un mot PLAN d'où la donnée à convertir sera prise, pour le type A cette donnée peut occuper plus d'un mot.

Règles de conversion :

PIOUT : 1. Les zéros en tête sont supprimés.

2- Les signe s'il est négatif est placé à gauche du premier caractère numérique.

PFOUT : 1- Les zéro en tête sont supprimés.

2- Aucune position n'est recquise si le signe n'est pas négatif.

3- La troncature à droite est automatique.

4- La troncature à gauche résulte d'un CALL PEOUT

5- Si (W-D- S-M-1) < 0 alors l'appel est traité comme un CALL PEOUT

W : nombre de positions occupées par le nombre

D : nombre de positions décimales.

S : nombre de positions significatives à gauche du point décimal

M = 1 si le nombre est négqtif, 0 s'il est positif.

Exemple :

-00.02339 sera considéré en type E car : W-D- S-M-1 = 8-5-2-1-1 <0

PEOUT : 1. Le signe + n'est pas obligatoire.

2. Si D = 0 et (W-D- S-M) 0 alors W = M+6

Si D 0 nous avons des astérisques.

PAOUT : 1. Nous avons conversion en format A4(code d'impression).

Exemple :

Le module ci-dessous permet de convertir une donnée prise dans le buffer d'entrée et de la placée dans la zone de communication dans le format spécifié.

Les CAP 200, 201 et 202 nous renseignent respectivement sur le type, le format et la position où doit être stockée la donnée convertie.

a) Définition du module et stockage sur disque :

```
// JOB
// FOR
* LIST SOURCE PROGRAM
C AOUDJE IANE ABDERRA MANE
C ----------PROJET PLAN-----
C RESERVATION DU COMMON ET DU BUFFER
C D ENTREE
```

```
            .COMMON L(625), LS(15), M(510)
             EQUIVALENCE (I,M(200)); (NBC, M(201))
             J = M(202)
             CALL PSBFA (0)
C LECTURE UNITE COURANTE
             CALL PLINP(0)
C TEST- TYPE DE DONNEE
             IF(I-1) 13,1,13
          13 CONTINUE
             IF(I-2)3,2,3
C CONVERSION DE DONNEES
          1  CALL PIIN(0,1,NBC,M(J))
             GO TO 4
          2  CALL PFIN (0,1,NBC,M(J)
             GO TO 4
          3  CALL PAIN(0,1,NBC,M(J))
C            RETOUR AU CHARGEUR PLAN
          4  CALL LRET
             GO TO 4
             END
#DELETE                              INTO
#STORECI   WS  UA                    INTO
```

b) Définition de la commande

```
ADD PHR : CONVERSION, LEV1, I(200) TYP2, I(201) FOR115, I(202)CAP1,
          PRO'INTO';
```

c) Utilisation de la commande :

```
CON, TYP1, FOR4, CAP20;
-173                  (FDZ)
```

A la rencontre de cette commande le clavier se met en attente alors l'utilisateur tape son nombre; puis appuit sur la touche (FDZ).

A la fin de l'exploitation de cette instruction le CAP20 réservera la valeur entière -173.

Les différentes valeurs de TP sont :

1 : valeur entière format I

2 : Valeur réelle format F

3 : valeur alphanumérique format A.

III-5- Programmes utilitaires :

Nous appelerons programmes utilitaires un ensemble de modules permettant de faciliter certaines opérations en mémoire centrales.

III-5.1- Tests logiques :

La fonction arithmétique :

NDEF(ARG) où ARG est une adresse en mémoire permet de tester la valeur logique d'un mot PLAN.

Ainsi le format du test sera :

IF(NDEF(ARG))1,2,3

1 : contenu de ARG admet la valeur logique FAUX

2 : contenu de ARG admet la valeur logique VRAI

3 : contenu de ARG admet aucun de ces deux valeurs.

Il existe d'autre part deux sous programmes permettant de valorisér le contenu de ARG.

CALL TRUE (ARG) : met le contenu de ARG à VRAI

CALL FALSE (ARG) : met le contenu de ARG à FAUX

III-5.2 - Recherche d'une commande :

La routine INPUT permet de mettre en mémoire et ce en EBCDIC le nom de la commande en cours d'exécution, ceci nous donne la possibilité de la tester ou de la faire imprimer.

Le format général de cette routine est :

CALL INPUT (N,ARRAY)

N : longueur de la zone ARRAY où se trouve en EBCDIC la commande en cours d'exécution.

ARRAY(1) contient la longueur (en caractères) de la commande.

Exemple :

Ce module permet d'imprimer la commande en cours d'exécution :

```
// JOB
// FOR
C RESERVATION DE ZONE ET BUFFER
          DIMENSION A(114)
          DIMENSION B(60)
          COMMON L(625), LS(15), M(510)
          CALL PSBFA(100)
C SAUVEGARDE DU BUFFER
          CALL PAIN(100,1,120,B(1))
C CHERCHE COMMANDE EN COURS
          CALL INPUT (114,A)
C REMPLIR BUFFER DE SORTIE ET IMPRIMER
          NWDS= (A(1)+7)/4
          DO 10 I = 2, NWDS,30
          CALL PAOUT (100, 1, 120, A(I))
          CALL PLOUT (100)
       10 CONTINUE
C RESTAURATION DU BUFFER
          CALL PAOUT(100,1,120,B(1))
C RETOUR AU CHARGEUR'PLAN.
        2 CALL LRET
          GO TO 2
          END
#DELETE                    WCOM
#STORECI     WS    UA      WCOM
```

Ce module peut être utilisé dans la définition d'une commande sous les mots clés PRO ou VERB.

III-5.3- Exécution d'une commande de la mémoire :

La routine PUSH permet de demander l'exécution d'une commande à partir de la mémoire.

Le format de l'appel est :

```
CALL PUSH(ADR)
```

ADR : adresse du mot contenant le nombre de caractères de la commande à exploiter.

Exemple :

Définition d'une phrase :

```
ADD PHRASE : DUMP,(15) 'DUM SWI';
```

Définition d'un module =

```
// JOB
// FOR
        COMMON L(640), CA(510)
        CALL PUSH (CA(15))
      2 CALL  LRET
        GO TO 2
        END
```

L'exécution du module ci-dessus nous permet d'avoir l'exécution de la commande DUM SWI.

III-6- Manipulation de zone

Un ensemble de routines permettent de déplacer des mots PLAN.

III-6.1- CALL PARGO et PARGI

Les S/P PARGO et PARGI permettent une application des commutateurs 4 à 7.

Si l'adresse "vers" leCOMMON est trop grande on a avortement de l'appel et aucun diagnostic d'erreur.

III-6.1.1- CALL PARGO (LS, ARRAY)

Ce sous-programme permet de transferer des informations de la position ARRAY(2) vers la zone de communication pointée par le commutateur LS.

ARRAY(1) contient le nombre de mots à déplacer.

Exemple :

```
// JOB
// FOR
// COMMON L(625), LS(15), M(255)
        F = 5
        LS(4) = 20
        CALL PARGO (4,F)
    2   CALL LRET
        GO TO 2
        END
```

Ce module permet de déplacer 5 mots à partir de F(1) vers CAP20, 21, 22, 23 et 24 respectivement.

III-6.1.2- CALL PARGO (LS,ARRAY)

Ce sous-programme permet de transferer des données de la zone de communication à partir de la position pointée par le commutateur LS vers la mémoire.

ARRAY(1) : contient le nombre de mots à transférer

ARRAY(2) : et le premier mot à recevoir les données.

III-6.2- CALL STVAL (A,N,B,I)

Cette routine permet de déplacer N mots PLAN de A(1) à A(N) vers B(I) à B(I+N-1)

III-6.3- CALL STVAL(A,N,B,I)

Cette routine permet de déplacer N mots PLAN de B(I) à B(I+N-1) vers A(1) à A(N).

III-6.4- Exploitation de bytes et caractères :

Un byte est un ensemble de 8 bits.

III-6.4.1- Comparaison logique de zone :

La fonction PCOMP permet de comparer deux zones logiques.

Le format d'utilisation est :

IF(PCOMP(A,B,N))1,2,3

1 si A est inférieur à B

2 si A est égal à B

3 si A est supérieur à B

A et B sont les zones à comparer

N est le nombre de mots PLAN à considérer.

Exemple :

```
//  JOB
//  FOR
        COMMON L(625), LS(15), M(510)
        EQUIVALENCE (N1,M(21)); (N2,M(26))
        CALL PSBFA(100)
        IF(PCOM(N1,N2,3))2,1,1
    1   CALL PAOUT(100,3,M(20),M(21))
        CALL PLOUT(100)
        GO TO 3
    2   CALL PAOUT(100,3,M(25), M(26))
        CALL PLOUT(100)
    3   CALL LRET
        GO TO 3
        END
```

Ce module permet d'ordonner 2 noms alphabétiquement et d'imprimer le deuxième des deux après comparaison.

III-6-4.2- Conversion de l'hexadécimal en EBCDIC :

CALL PHTOE (A,B,N)

PHTOE est un sous-programme qui permet de convertir une donnée de l'hexadécimal en EBCDIC :

A : zone contenant la donnée en hexadécimal

B : zone contenant la donnée en EBCDIC

N : nombre de mots de la zone A.

N mots de la zone A sont convertis en 2N mots dans la zone B.

Exemple :

Le module ci-dessous permet d'imprimer à l'unité courante le contenu des commutateurs.

```
      // JOB
      // FOR
C RESERVATION DE ZONE ET AFFECTATION DE BUFFER
                DIMENSION W(30)
                COMMON L(625), LS(15), M(510)
                CALL PSBFA(100)
C CONVERSION HEX - EBCDIC et IMPRESSION
                CALL PHTOE (LS, W,15)
                DO 10 I = 1,10
           10   CALL PAOUT (100,9*I-8,8,W(2*I-1))
                CALL PLOUT(100)
                DO 11 I = 11,15
           11   CALL PAOUT (100, 9*I-98,8,W(2*I-1))
                CALL PLOUT (100)
            3   CALL LRET
                GO TO 3
                END
```

III-6.4.3- Manipulation de bytes :

CALL PPACK (I,A,N)

Ce sous-programme permet de masquer l'octet spécifié par N du mot adressé par A de la valeur entière I; les autres bytes sont inchangés.

Exemple:

CALL PPACK(194,A,2)

Permet de mettre la lettre B (valeur décimale 194) dans les bits 8-15 du mot A.

CALL PUNPK (I,A,N)

Ce sous-programme permet de masquer l'octet spécifié par N du mot adressé par A par la valeur entière I.

Les bytes à gauche de la valeur insérée sont effacés.

Exemple :

```
CALL PUNPK(194,A,3)
```

met la lettre B dans les bits 16-23 les bits 0-16 sont effacés.

```
CALL BREAK (A,J)
```

Ce sous programme permet de cadrer à gauche dans les mots J,(J+1), (J+2) et (J+3) les 4 bytes de A; les autres bytes sont mis à zéro.

Exemple :

```
C INVERSION DE L'ORDRE DE S CARACTERES D'UN MOT
      DATA A  'ABCD'
      DIMENSION N(4)
      CALL BREAK (A,N)
C INVERSER L'ORDRE
      DO 5 I = 1,4
    5 CALL PPACK (N(I), A,5-I)
```

Après exécution de ces instructions A contiendra DCBA au lieu de ABCD.

III-7- Manipulation des données sur disque :

Il est possible de définir deux typesde fichiers sur le disque 1130, les fichiers temporaires et les fichiers parmanents.

Le nombre maximum de fichiers des 2 types est 5 pour le 1130 de taille 8 K.

III-7.1- Les fichiers temporaires :

Un fichier temporaire admet une taille variable.

III-7.1.1- Ouverture d'un fichier :

La routine qui permet d'ouvrir un fichier est : FIND

Format d'appel :

```
CALL FIND (ID, NPRI, NALLO, NDR)
```

ID(1) : contient le numéro du fichier (compris entre 1 et 255).

- fichier 255 unité 0 réservé aux méssages d'erreurs
- fichiers 201 à 255 unité 0 réservés aux utilitaires
- fichiers 1 à 200 peuvent être exploités par l'utilisateur

ID(2) : contient la table du fichier en cours.

NPRI : priorité du fichier

NPRI = 0 : priorité de la commande en cours
NPRI = 1 : Le fichier doit être fermé par RELE S
NPRI = 2,3 ou 4 : fichiers détruits par une commande de niveau supérieur.

NALLO : estimation de la place prise par le fichier
NALLO = 0 : on réserve de la place au fur et à mesure que l'on retourne dans le fichier
NALLO ≠ 0 : on réserve un nombre de mots correspondant à la taille estimée du fichier par la formule.

NWA = ((NALLO-1)/N S A + 1) * N S A = (NALLO-1)+N S A
N S A = 328 pour IBM 1130
NWA : nombre de mots actuellement alloués

NDR : numérode l'unité disque où se trouve le fichier, sur 1130 ce nombre peut être compris entre 0 et 3 (ENPA 1 seul disque d'où NDR = 0)

III-7.1.2- Fermeture d'un fichier.

Le format d'appel à la routine que permet de fermet un fichier est :

CALL RELE S (ID, 0, N S Z, NDR)

ID et NDR ont même signification que pour CALL FI ND
N S Z = 0 : fermeture du fichier et ID(2) non modifié
N S Z ≠ 0 : le fichier n'est pas fermé mais sa taille est ramenée à N S Z

Toute commande peut fermer un fichier d'une autre mais de niveau inférieu

CALL RELEAS est la dernière fonction à exécuter pour un fichier dont les données ne sont pas à réutiliser.

III-7.1.3 - Lecture et écriture d'un fichier :

Il existe deux s/p permettant l'accès au disque, leur appel général est :

CALL { READ / WRITE } (ID, KDIAS, KOUNT, ARRAY)

CALL READ : permet de transférer de l'information du fichier sur disque vers la mémoire centrale.

CALL WRITE : permet de transférer de l'information de la mémoire centrale vers le fichier sur disque.

ID : défini au moment de l'ouverture par CALL FIND, ne doit pas être modifié par l'utilisateur.

KDIAS : dont le rang du premier mot à transférer

POur transférer le premier mot on fait KDIAS = 0; pour le Nième on fait KDIAS = (N-1)

(N numéro dans le fichier).

KOUNT : nombre de mots à transférer

Pour l'écriture on doit avoir : ID(2) $\geq$ KDIAS + KOUNT

Si cela n'est pas le cas on a mise automatique de ID(2) à la valeur : KDIAS+KOUNT.

ARRAY : adresse en mémoire où/d'où doit être transférée l'information.

Remarques :

- Cet ensemble de sous-programme permet d'utiliser le disque comme une extension de la mémoire centrale pour y stocker des données.

- Les fichiers sont définis dynamiquement, ils sont automatiquement étendus à tout apport d'information.

- Si on tente de lire ou d'écrire plus de mots qu'il y a dans le fichier on a génération d'un EDF (END OF FILE), fermeture du fichier et passage à l'instruction suivante.

- Pour tester la validité d'un numéro de fichier on peut utiliser :

```
IF(ID(1)-256)1,1,2
```

1 : fichier ouvert

2 : fichier fermé.

III-7.1.4- Test de la place disponible sur disque :

Pour un niveau donné il est possible de faire un test pour connaitre la place disponible sur disque grâce à l'appel :

CALL PFSPC (0, NPRI, NALLO, NDR)

NPRI : priorité (0,1,2,3 ou 4)

NALLO : nombre de mots disponibles pour la priorité NPRI

NDR : numéro de l'unité disque (0,1,2,3 ou 4)

L'utilisation du paramètre DFI dans la commande PLAN JOB permet de fermer le fichier et de mettre une valeur négative dans ID(2) qui nous permet de connaître le type d'erreur.

Code des erreurs :

- 1 : taille du fichier mal définie
- 2 : NDR ou ID invalide dans FIND ou RELES
- 3 : ID invalide dans read ou WRITE
- 4 : KDIS ou KOUNT invalide dans READ ou WRITE
- 5 : fichier invalide
- 6 : taille insuffisante dans FIND ou WRITE
- 7 : réservé
- 8 : ID mal défini
- 9 : PFIND n'est pas dans la bibliothèque.

Exemple :

Remplir 700 mots d'un fichier temporaire par les 700 premiers nombres entiers.

a) Définitions du module

```
// JOB
// FOR
*LIST SOURCE PROGRAMME
C   RESERVATION DE ZONE
    DIMENSION NA(100), ID(2)
    COMMON L(625), LS(15), M(510)
    EQUIVALENCE (NA(1), M(20))
C TEST DE LA PLACE
    NPRI = 4
  1 CALL PFSPC(0,MPRI,KT,0)
    IF(KT-700)5,5,100
```

```
      5 NPRI = NPRI-1
        IF(NPRI)10,10,1
C AVORTER LA PHRASE
     10 CALL LEX (1,0)
C OUVERTURE DU FICHIER
    100 ID(1) = 27
        CALL FIND (ID,NPRI,700,0)
C ECRITURE SUR FICHIER
        DO 51 I = 1,100
     51 NA(I) = I
        DO 101 I=1,7
        CALL WRITE (ID, ID(2), 100, N1)
        DO 15  J = 1,100
     15 NA(J) = NA(J)+100
    101 CONTINUE
C LECTURE ET VERIFICATION
        DO 25 I = 1,7
        CALL READ (ID, (I-1)*100,100,NA)
        DO 25 J = 1,100
        IF (NA(J)-(J+(I-1)*100)) 20,25,20
     20 PAUSE 7
     25 CONTINUE
C FERMETURE DU FICHIER
        CALL REGLES (ID, 0, 0, 0)
C RETOUR AU CHARGEUR PLAN
      3 CALL LRET
        GO TO 3
        END

// DUP
*DELETE                    ESDT
*STORECI  W  S    UA       ESDT
```

b) Définition de la commande

```
ADD PHRASE : ESDT, PRO'ESDT';
```

c) Utilisation :

LON.
E S DT ;

Après exécution de cette commande nous aurons à partir de CAP20 les nombres 601,602,...700 et dans le fichier de taille 700 les nombres 1,2,3,...,700.

III-7.2- Les fichiers permanents :

Un fichier permanent admet une taille fixe et peut être utilisé, une fois défini, à n'importe quel moment.

III-7.2.1- Ouverture d'un fichier :

Un fichier permanent est initialisé en dehors de PLAN grâce aux routines de DUP.

Ainsi pour un fichier de données on a :

```
// JOB
// DUP
* ODREDATA  W S   UA   FILE nnnn
```

FILE : spécifie le nom du fichier
nnnn : taille du fichier en secteurs.

La routine GDATA permet d'ouvrir un fichier permanent prédéfini sousless routines du disque.

Le format général d'appel est :

CALL GDATA (ID, NAME, LR, NDR)

ID : même fonction que pour FIND (III.7.1.1)

NAME : nom du fichier, 8 caractères du minimum; mais seuls les 5 premiers caractères sont significatifs.

LR : taille du fichier en bytes (multiple de 640)

NDR : numéro de l'unité disque (0,1,2 ou 3)

III-7.2.2.- Lecture et écriture des fichiers sur disques :

Il existe deux sous-programmes qui permettent l'accès au disque pour les fichiers permanents; leur format général est :

CALL { RDATA / WDATA } (ID,KDI S, KOUNT, ARRA Y)

Les paramètres définis pour ces deux routines sont les mêmes que celle de READ et WRITE (III.7.1.3).

III-7.2.3- Test des routines des fichiers permanents :

L'utilisation du paramètre PFI dans les commandes standard PLAN JOB permet de tester les routines précédentes.

Les erreurs possibles sont :

1- NDR ou ID invalides dans GDATA

2- ID invalide dans RDATA ou WDATA

3- KDIS ou KOUNT négatifs dans RDATA ou WDATA

4- Fichier non ouvert lors de l'utilisation de RDATA ou WDATA

5- KDIS + KOUN > ID(2)

exemple :

1) création du fichier sous DUP.

```
// JOB
// *FICHIER PERMANENT NOM = FILEP (CREATION)
// DUP
*DELETE                          FILEP
* STOREDATA   CD    UA           FILEP  0008
[ 8 cartes de données constituant le contenu du fichier carte FILEP ]
```

2) Module de lecture du fichier permanent défini ci-dessus et impression à l'unité courante de ce dernier.

```
// JOB
// *LECTURE DU FICHIER FILEP
// FOR
* LIST SOURCE PROGRAM
C RESERVATION DE ZONE ET BUFFER
      DIMENSION ID(2), W(20), DAME(2)
      COMMON L(625), LS(15), M(510)
      EQUIVALENCE (NCA, M(20)), (IP, M(21)),(W(1), M(23)
      DATA DAME /'FILE','P  ')
      CALL PSBFA(100)
C OUVERTURE DU FICHIER PERMANENT ET LECTURE
      ID(1) = 1
      DO 2 J = 1,NCA
```

```
      CALL GDATA (ID,DAME, 80*NCA. ,0)
      CALL RDATA (ID,20*(J-1),20,W)
C CONVERSION ET IMPRESSION
      DO 1 LP =1,20
    1 CALL PAUT (100, IP+4*(LP-1),4,W(LP)
      CALL PLOUT (100)
    2 CONTINUE
C RETOUR CHARGEUR PLAN
    3 CALL LRET
      GO TO 3
      END
//  *STOCKAGE EN IMAGE MEMOIRE
//  DUP
*DELETE                    FPER
*STORECI   WS    UA        FPER
```

3) *Définition de la commande :*

ADD PHR : FIC CAR, LEV1, I(20)NCA1, I(21)POS, PRO'FPER';

4) *Utilisation :*

Après lecture des cartes de controle de PLAN au lecteur la console se met en attente alors taper :

FIC CAR, NCA8, POS20;

Après interprétation de cette commande nous aurons impression du fichier FILEP sur l'unité courante de sortie.

III-7.2.3- Option ONE WORD INTEGER

L'utilisation de l'option ONE WORD INTEGER permet de substituer respectivement aux modules :

FIND, READ, WRITE, RELES, GDATA, RDATA et WDATA les sous programmes :

PFND1, PRED1, PWRT1, PREL1, GDAT1, RDAT1, et WDAT1.

Le format d'appel est le même que celui des premiers énoncés.

exemple :

```
CALL PREL1(ID,0,NSQZ, NDR)
```

Dans le cas d'utilisation de cette deuxième suite de modules on aura des mots de 16 bites.

TROISIEME PARTIE

Le Dictionnaire
(fichier PFILE)

CHAPITRE IV : Les commandes standards :

Dans le dictionnaire des phrases nous avons un ensemble de commandes standards en plus de celles définies par l'utilisateur.

Le nombre maximum de phrases est cinquante.

Les commandes sont de niveau 0 aussi ont-elles priorité sur toutes les autres.

IV-1- ADD PHRASE :

Cette commande permet d'ajouter au dictionnaire une nouvelle phrase utilisateur. C'est la seule phrase programmée les autres s'obtiennent par son utilisation.

Exemple :

ADD PHRASE : LIST, I(2)100, PRO'ENTP';

Nous avons ainsi définie la phrase objet qui utilise le module ENTP et le CAP2.

IV-2- ALTER PHRASE :

Cette commande permet d'affacer une phrase et de la remplacer par une nouvelle version.

Exemple :

ALTER PHRASE : LIST, VERB, I(2)101, PRO'ENTP';

IV-3- DELETE PRASE :

Cette commande permet d'effacer une phrase du dictionnaire.

exemple :

DELETE PHRASE : VERB, LIST;

IV-4- PLAN JOB

PLAN JOB est une commande qui permet d'initialiser les fonctions de PLAN, elle est la première exploitée par PLAN.

Les paramètres de cette commande sont :

NOM	CAP	MODE	Valeur de défaut	FONCTION
FILE	-1	I		-définit le numéro du fichier d'où l'instruction suivante sauvegardée sera exécutée. -Ne pas l'utiliser si la prochaine instruction n'est pas sauvegardée.
SAVED	-2	I		-amorce la sauvegarde
TO	-3	I		-numéro de la dernière instruction sauvegardée à exécuter.
LIST	-4	I		-définissent des pointeurs dans la zone des données. -utilisés surtout par : PARGO, PARGI, et PSCAN
LB	-5	I		
LC	-6	I		
LD	-7	I		
ERASE	-8	I		-Donne la position du début de la zone de communication. -Ce commutateur se met à la valeur 490 toutes les fois qu'une commande de niveau 0 est rencontrée. -La zone effaçable s'étend de cette position à la fin de la zone de communication.
COMMON	-9	I		-Taille du COMMON (taille minimum 640)
MANAGED	-10	I		-Taille de la zone gérée par le niveau de la commande en cours.
NERM	-11	I		-Si les diagnostics d'erreur sont écrits sur le fichier dynamique 255 unité 0, alors ce paramètre définit le nombre maximum de messages alloués au fichier avant leur impression.
UMOD	-11	LIT		-Donne le nom du module écrit par l'utilisateur pour l'exploitation des erreurs.

DEVICE	-12	I		- spécifie le code de l'unité pour l'impression des diagnostics. Ce commutateur prend la valeur 100 à la rencontre d'une commande de niveau 0.
FORM	-13	I	0	- non référencé par l'utilisateur - format du diagnostic.
SHORT	1	LOG	FALSE	- Format court pour les diagnostics d'erreur (donne le numéro de l'erreur)
LONG	2	LOG	FALSE	- Format long pour les diagnostics d'erreur
STACK	3	LOG	FALSE	- Empil'er les messages d'erreurs et les donner par ordre au chargeur ou quand la pile déborde.
IMM	4	LOG	FALSE	- Donner immédiatement les erreurs
DRIVE	5	I	0	-numéro de l'unité utilisée pour la sauvegarde des instructions.
DFI	6	LOG	FALSE	- Diagnostics d'erreur provenant des S/P des fichiers temporaires.
PFI	7	LOG	FALSE	- Diagnostics d'erreur provenant des S/P de fichiers permanents.

Exemple :

PLAN JOB, MAN 200, ERA 240, COM 900, LIST 30, 60, 200, 209, SAVED 20 TO 30 FILE 3, DRI2 SHORT, STACKED, DEVICE 101;

- taille de la zone gérée : 200 mots PLAN
- Début de la zone de communication effaçable CAP 240
- taille du COMMON 900
- Les commutateurs 4 à 7 pointent respectivement les positions 30, 60, 200, et 209 dans la zone de communication.

- Sauvegarde des instructions 20 à 30 du fichier 3, unité 2.
- Format de diagnostics d'erreur : court
- Empiler les diagnostics
- Unité de sortie des diagnostics, imprimante 1132.

IV-5- DUMPS DE LA ZONE DE COMMUNICATION :

DUMP COMMON;

permet d'avoir un dump du COMMON en hexadécimal .

DUMP MANAGED;

permet d'avoir un dump de la zone gérée en hexadécimal.

DUMP NONMANAGED;

permet d'avoir un dump de la zone non gérée en hexadécimal.

DUMP SWITCHES

permet d'avoir un dump des commutateurs en hexadécimal.

Les paramètres de ces commandes sont :

NOM	CAP	MODE	Valeur de défaut	FONCTION
M	-8	I		- Pointeur défini dans PLAN JOB
N N N	M	I	0, 1, -1, -2	- Définit le type de dump
NOD	M+5		100	- Unité de sortie du dump

IV-6-DUMPS DE FICHIERS

DUMP PERMANENT : permet d'avoir un dump d'un fichier permanent

DUMP DYNAMIC : permet d'avoir un dump d'un fichier temporaire

Les paramètres de ces commandes sont :

NOM	CAP	MODE	Valeur de défaut	FONCTION
M	-8	I		- Définit par PLAN JOB
FILE	M	I	255	- Numérodu fichier à dumper
START	M+2	I	0	- Début du dump
END	M+3	I	0	- Fin du dump
DRIVE	M+4	I	0	- Numéro de l'unité où se trouve le fichier
NAME	M+12	LI	BLANC	- Nom du fichier à dumper
NOD	M+15	100		- Unité de sortie
	M+16	I	0,1	- Type de fichier

IV-7- Dump de la table des phrases :

La commande DUMP PHRASE permet d'avoir un listing du fichier PFILE (dictionnaire des phrases) :

Les paramètres de cette commande sont :

NOM	CAP	MODE	valeurde défaut	FONCTION
SYSTEM	500	I	1130	- Précise le système d'où le fichier PFILE Doit être dumpé.
DEVICE	501	I	100	- Unité de sortie
LEVEL	503	I	1	- Définit le niveau du dump

Il est possible d'avoir un listing des erreurs grâce à la commande :

DUMP ERROR $

IV-8- Commandes de sauvegarde d'instruction :

voir page (II-4-3)

IV-9- Commande IOC $:

La commande IOC $ permet d'affecter des unités d'E/S.

Les paramètres de cette commande sont :

NOM	CAP	MODE	Valeur de défaut	FONCTION
INPUT	1	I	1131	
LIST	2	I	1131	

IV-10- Commande LON

Cette commande permet d'établir le niveau 1.

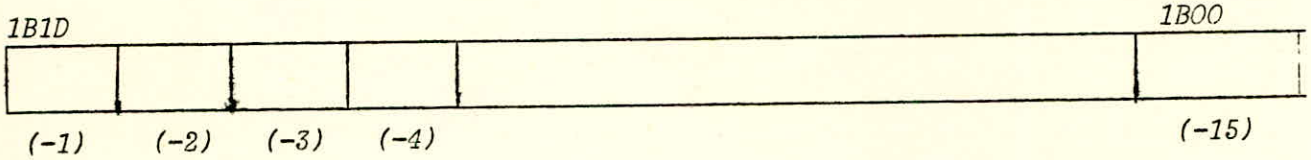
V-1- Les commutateurs :

Le 1130 admet 15 commutateurs de 32 bits chacun

Ils occupent les positions -1 à -15 du COMMON

Descriptions des commutateurs.

Les 15 mots des commutateurs occupent les adresses 1B 1D à 1A 00.



Commutateurs	FONCTION
1	'- Contient le numéro du fichier temporaire à sauvegarder '- Il peut aussi indiquer une ouverture ou une fermeture d'un fichier permanent.
2	'- Contient le numéro de l'instruction suivante à exécuter lors d'une sauvegarde; s'il est nul on exploitation d'une façon normale.
3	'- Contient le numéro de la dernière instruction sauvegardée à exécuter. '- Si le commutateur 2 contient 0 alors les switches 1 et 2 peuvent être utilisés pour une fonction du système.
4 à 7	'- Servent de pointeurs dans la zone de communication.
8	'- Pointe sur le début de la zone de communication effaçable, l'intéret de cette zone est de créer une aire non affectée par les modules (valeurs de défaut 490).
9	'- Permet de définir la taille maximum (mots de 32 bits) du COMMON en cours d'exécution; la taille minimum demandée est : 640 Chargeur PLAN 625 Commutateurs 30 Interpréte 510 Valeur de défaut 1150
10	'- Taille de la zone gérée par la commande en cours
11-12	'- Nom du module d'exploitation d'erreur de l'utilisateur
11	'- S'il est positif ou nul il indique le numéro des diagnostics à imprimer du fichier temporaire 255 unité 0 et ce à l'unité définie par le commutateur 12 ou par CALL ERLT. (cette option n'est pas utilisable par l'utilisateur) valeur de défaut 0
12	'- Donne le numéro de l'unité de sortie des messages d'erreur valeur de défaut 100.

13	- Donne le mode du diagnostic d'erreur - Valeur de défaut 0
14	- Réservé - Valeur de défaut 0
15	- Fonctions utilisateurs (non initialisé)

* Remarque :

Lescommutateurs sont initialisés à la rencontre d'une commande de niveau 0. (PLAN JOB en général)

QUATRIEME PARTIE

Programmation et exemple

Chapitre IV : Programmation :

IV-1- Introduction

Pour travailler sous PLAN nous devons suivre certaines règles de programmation, ce chapitre s'efforcera de donner les différentes étapes à suivre.

IV-2- Instructions interdites :

Les instructions ci-dessous ne doivent pas être utilisée dans un module utilisable par PLAN.

CALL EXIT : retour au superviseur.
Il ne faut jamais revenir au superviseur mais donner le controle au chargeur PLAN par CALL LRET ou CALL LEX(1,0) qui permet d'effacer la pop up list et de donne le contrôle au chargeur PLAN.

STOP : même effet que CALL EXIT

CALL LINK : doit être remplacé par l'une des routines PLAN: LRET, LEX ou LIST.

DEFINIE FILE, READ(a,b), WRITE (a'b) : seront remplacés par les routines FIND, READ-WRITE ou GDATA, RDATA, WDATA, WRITE (a,b) et READ(a,b) seront remplacés par PLOUT et PLINT.

Ne pas utiliser DISK1, DISKN ou DISK2 pour la lecture sur disque en Assembleur.

IV-3- Programmation en FORTRAN IV :

Les modules utilisateur peuvent être écrits en FORTRAN IV en tenant compte des restrictions ci-dessus et en n'oubliant pas de réserver une zone COMMON au moins de 640 mots PLAN (1280 mots machine) pour recevoir le chargeur et les commutateurs.

L'instruction FORTRAN EQUIVALENCE permet de nommer les mots du COMMON d'une façon plus explicite.

IV-4- Programmation en ASSEMBLEUR 1130.

Les modules utilisateurs peuvent être écrits en Assembleur 1130 dans ce cas la transmission des données pose des problèmes assez délicats à résoudre. Ne pas oublier de réserver une zone COMMON pour le chargeur et les commutateurs grâce à la carte de contrôle.

```
* COMMON  01280
```

IV-5- Exemple d'application.

Soit à écrire un programme qui permet d'imprimer le PGCD de 2 nombres par la méthode d'Euclide.

1) Stocker sur bibliothèque les modules utilisés par la commande PGCD qui calcul le PGCD de 2 nombres

```
CARTE C  S  (COLD  START)
//  JOB
//  *REMPLIR COMMON A PARTIR DE CAP(J), NBC : FORMAT
//  FOR
* LIST  SOURCE PROGRAM
C RESERVATION DE ZONE ET BUFFER
      DATA ETOI /'****'/
      COMMON L(625), LS(15), M(510)
      EQUIVALENCE (I,M(200)), (NBC, M(201)), (J,M(202)),
     1  (RET, M(198)), (INP,M(199))
      CALL PLINA(0)
C AFFECTATION D'UNITE D'ENTREE
      CALL IOCS (INP,100)
C LECTURE DE DONNEE, TEST DE FIN DE FICHER
   5  CALL PLINP (0)
      CALL PAIN (0,1,4,B)
      IF (ETOI-B) 15,4,15
C TEST DU TYPE DE DONNEE
  15  IF (I-1) 13, 1, 13
  13  IF(I-2)3,2,3
```

```
      C CONVERSION DE S DONNEES
      C FORMAT I
           1  CALL PIIN (0,1, NBC, M(J))
              J = J+1
              GO TO 5
      C FORMAT F
           2  CALL PFIN(0,1, NBC, M(J))
              J = J+1
              GO TO 5
      C FORMAT A
           3  CALL PAIN (0,1,NBC, M(J))
              J = J+1
              GO TO 5
           4  CALL IOCS (RET, 100)
      C RETOUR AU CHARGEUR PLAN
           6  CALL LRET
              GO TO 6
              END
// DUP
// *STOCKAGE SUR DISQUE EN FORMAT DCI
*DELETE                         ENTP
*STORECI      WS    UA           ENTP
// JOB
// *IMPRIMER COMMON A PARTIR DE ICA, FOR : FORMAT
// FOR
*LIST SOURCE PROGRAM
C RESERVATION DE ZONE ET BUFFER; UNITE DE SORTIE COMMON L(625),
     LS(15), M(510)
              EQUIVALENCE (I,M(203)), (IFOR, M(204)), (NCA, M)
           1  (205), (ICA, M(206)), (IP, M(207)), (LIST, M(208)),
           2  (SAUT M(210), (RET, M(209))
              CALL PSBFA(100)
              CALL IOCS (0, LIST)
C CONVERSION DE DONNEES
              J = ICA - 1
```

```
          4  J = J+1
             IF (I-1)13,1,13
          3  IF (I-2)3,2,3
C FORMAT I
          1  CALL PIOUT(100, IP,IFOR, M(J))
             GO TO 10
C FORMAT E
          2  CALL PEOUT (100, IP, IFOR, M(J))
             GO TO 10
          3  IF (I-3)7,8,7
C FORMAT A
          8  CALL PAOUT (100, IP, IFOR, M(J))
             GO TO 10
C FORMAT F
          7  CALL PFOUT (100, IP IFOR, M(J))
C IMPRESSION
          10 CALL PCCTL(100, SAUT)
             CALL PLOUT (100)
             NCA = NCA-1
             IF(NCA-1)5,4,4
C RETOUR CHARGEUR
             CALL IOCS (0,RET)
          5  CALL LRET
             GO TO 5
             END
// DUP
// * STOCKAGE SUR DISQUE
*DELETE                    SORT
* STORECI  WS  UA           SORT

// JOB
// ASM
*LIST
*COMMON  02300
          *CALCUL DE PGCD
          *ORI POINTE DANS LE COMMON
```

```
DEBUT   LD      I       ORI
        SRT             16
        MDM     L       ORI,-2
        D       I       ORI
        RTE             16
        BZ              FIN
        STO             R
        LD      I       ORI
        MDM     L       ORI,+2
        STO     I       ORI
        LD              R
        MDM     L       ORI,-2
        STO     I       ORI
        MDM     L       ORI,+2
        B               DEBUT
FIN     CALL            LRET    RETOUR CHARGEUR
R       DC              0
ORI     DC              /1AFE   ADRESSE DU DEUXIEME
                                MOT DE LOCCOM.
        END             DEBUT   16 bits du COMMON
```

```
// DUP
// *STOCKAGE SUR DISQUE
*DELETE          PGCD
*STORECI  WS  UA PGCD
```

2) *Définition de la commande PGCD et stockage dans le dictionnaire des phrases.*

```
// JOB
// XEQ PLAN
PLAN JOB, ERA200, MAN510, LONG;
ADD PHRASE : PGCD, LEV1, I(198) RETO, I(199) INPO,
I(200)1, I(201)EFOR1, I(202)1, I(203)1, I(204) SFOR1,
I(205)1, I(206)2, I(207)POS, I(208)LIS 100, I(209)RTU100,
I(210) SAU-1, PRO'ENTP, PRO'ENTP, PGCD, SDT;
```

```
PGCD, EFOR3, FOR4, P0513, AU-3;
021                    Première donnée
012                    deuxième donnée
####                   Sortir du module ENTP
IOC$ INP1131, LIS1131;
/#
```

Définition des paramètres de cette commande.

NOM	CAP	MODE	Valeur de défaut	
RET	198	I	0	- Utilisé par le module ENTP Donne l'unité d'entrée après exécution de ENTP
INP	199	I	0	- Utilisé par ENTP Donne l'unité d'entrée des informations
	200	I	1	- Utilisé par ENTP; permet de tester le type de donnée (1,2,3 entier, réel, alphabétique respectivement).
EFOR	201	I	1	- Permet de donner le format d'entrée des données. - Utilisé par ENTP
	202	I	1	Pointe sur le mot du COMMON qui recevra la première donnée, utilisé par ENTP
	203	I	1	Donne le mode de la donnée en sortie utilisé par SORT (1,2,3,4, mode I,E,A et F respectivement).
FOR	204	I	1	- Donne le format de sortie - utilisé par SOR
	205	I	1	Donne le nombre de mot du COMMON à imprimer - utilisé par SORT

	206	I	2	- Pointe sur le premier mot du COMMON à imprimer - utilisé par SORT.
POS	207	I	1	- Donne la fonction sur le listing à l'impression - utilisé par SORT
LIS	208	I	100	- Unité de sortie de l'information utilisé par SORT
RTU	209	I	100	- Unité à utiliser pour les sortie après exploitation de SORT.
SAU	210	I	-1	- Controle de saut à la sortie.

* Remarques :

Le programme ci-dessus doit être tapé sur carte (chaque ligne 1 carte) et mis au lecteur dans l'ordre qui a été donné.

Après lecture des cartes de contrôle de PLAN on aura impression à la console du message : -

DFJ700 C PFILE FOUND ON PACK1972

Après lecture des données 021 et 012 nous aurons à l'unité courante (imprimante 1132) la valeur 3.

(Les #### permettent de sortir de ENTP).

Quand la carte IOCS, INP1131, LIS1131; est lue le clavier de la console sera sélecté et on pourra taper ses commandes à partir de l'unité 1131.

- Nous voyons que les programmes d'entrée/ Sortie sont écrits en FORTRAN et celui de calcul du PGCD en ASSEMBLEUR, la transmission des données se fait par utilisation de la zone COMMON.

Le module PGCD calcul le PGCD des nombres contenus dans les CAP1 et 2 respectivement et le résultat se trouve dans CAP2.

ANNEXE A

Dictionnaire des phrases utilisateur

Cette annexe va recenser les principales commandes qui ont été définies et utilisées pour les applications sous PLAN. Pour les applications sur les fichiers se référer au chapitre III paragraphe 7 (manipulation de données sur disque).

A-1 : Transfert d'information d'une unité sur une autre

AKT, INPm, LIS n;

Cette commande permet d'imprimer les informations données à l'unité m vers l'unité n.

Définition des paramètres.

NOM	CAP	MODE	valeur de défaut	Légende
INP	20	I	0	- Défini l'unité d'entrée, peut prendre les valeurs : 1 ou 1442, 3 ou 1131.
LIS	21	I	100	- Défini l'unité de sortie, peut prendre les valeurs : 101 ou 1132, 103 ou 1131, 104 ou 1442

Définition de la commande :

ADD PHRASE : AKT, LEV1, I(20)INP0, I(21)LIS100, PROCES,
&0 INP : (INP = 1442)? = 1; INP : (INP=1131)? = 3, LIS : (LIS = 1442)? = 104, LIS : (LIS = 1131)? = 103, LIS : (LIS = 1132)? = 101, INP : (INP = 1)) (INP = 3) ? = INP! = 0, LIS : (LIS=101) (LIS = 103)? = LIS ! = 100;

Le module E S permet de transférer des données fournies à l'unité définie par INP vers celle indiquée par LI S.

*Pour sortir du contrôle de AKT, il faut lui faire lire, ****

Divers applications peuvent être exécutées grâce à cette commande.

- *list d'un paquet de cartes sur la console ou à l'imprimante.*
- *Impression de messages à l'imprimante 1132 de messages donnés au lecteur de cartes 1442 ou clavier de la console 1131.*

A-2 : *Impression d'un titre à l'unité de sortie des résultats* :

EXP phrase objet, données;

Cette phrase verbe permet d'imprimer à l'unité courante un message contenu à partir du CAP2; le nombre de caractères de ce titre sera contenu dans le CAP1- Cette phrase ne peut être utilisée seule.

Définition des paramètres

NOM	CAP	MODE	valeur de défaut	Légende
	1	I		Donne le nombre de caractères contenu à partir du CAP2 à imprimer.
	2			Contient 4 caractères.

Définition de la commande :

ALTER P ER : EXP, VERB, PRO'EXP';

Le module EXP permet d'imprimer à l'unité courante les caractères contenu à partir du CAP2 et dont le nombre se trouve au CAP1.

Exemple : -

On suppose le clavier sélecté.

Pour résoudre l'équation :

$$5x + 3 = 0$$

Taper au clavier de la console :

```
EXP RE$ BAI, DEG1, COE 5, 3, TIT 'SOLUTION DE L'EQUATION
5 x + 3 = 0';
```

Après prise en compte de cette commande on aura :

```
SOLUTION DE L'EQUATION 5 x + 3 = 0
- 59.9999904E-02
```

A-3 : Transfert d'informations d'une unité d'entrée vers la zone COMMON utilisateur :

INT phrase objet, TYPn, NBCm, CAPp, INPa, RETb;

Cette phrase verbe permet de remplir le COMMON utilisateur à partir du CAPp et ce de l'unité de code a.

n : indique le mode de la donnée à entrer, il peut prendre 3 valeurs :

1 : mode entier (type I)

2 : mode réel (type F)

3 : mode alphabétique (type A)

m : donne le format d'entrée de la donnée/

p : pointe sur le premier mot de la zone de communication qui recevra la première donnée.

b : indique l'unité d'entrée sur laquelle se fera le retour après utilisation de cette commande.

Définition des paramètres :

NOM	CAP	MODE	valeur de défaut	Légende
TYP	200	I	2	- indique le mode de la donnée à entrer
NBC	201	I	115	- indique le format d'entrée de la donnée.
CAP	202	I	1	- pointe sur le premier mot du COMMON qui recevra la première donnée.
INP	199	I	0	- Unité d'entrée pour exploiter ENTP Valeurs possibles : 0, 1, 3
RET	198	I	100	- Unité d'entrée après utilisation de ENTP 0, 1, 3

Définition de la commande :

```
ALTER PHRASE : INT, VERB, I(200)TYP2, I(201)NBC115, I(202)CAP1,
I(199)INPO, I(198)RETO, PRO'ENTP';
```

Le module ENTP permet d'entrer des informations d'une unité donnée vers la zone COMMON et ce à partir de l'adresse CAP.

Pour sortir du contrôle de ENTP donner pour message #### à l'entrée.

Exemple : Unité d'E/S console 1131.

```
INT LON, TYP3, NBC4, CAP27;

PROJ                (FDZ)
ET P                (FDZ)
LAN                 (FDZ)
ECOL                (FDZ)
E PO                (FDZ)
LYTE                (FDZ)
CHNI                (FDZ)
QUE                 (FDZ)
xxxx
```

Après lecture et interprétation de la commande on a dans le COMMON :

PROJ	ET P	LAN	ECOL	E PO	LYTE	CHNI	QUE			
(27)	(28)	(29)	(30)	(31)	(32)	(33)	(34)			

```
INT LON;
12343.78000
00734.24500
xxxx
```

après interprétation et exécution de cette commande on aura :

12343.78	734.245		
(1)	(2)	(3)	

Dans ce cas se sont les valeurs de défaut qui sont prises. Nous voyons que la commande standard LON de niveau 1 nous sert de phrase objet.

A-4 : Transfert d'informations du COMMON vers une unité de sorti

phrase objet, TIPm, FORn, NMOk, ICAj, POSp, LISa, RTUb, SAUx, NBMz;

Cette phrase verbe permet d'imprimer sur l'unité courante ou celle définie par le code a, k mots à partir du CAPj et ce à partir de la position p sur le listing.
m et n indiquent réspectivement le mode et le format des données à imprimér.
m peut prendre 4 valeurs :

1 : mode entier (type I)
2 : mode exponentiel (type E)
3 : mode alphanumérique (type A)
4 : mode mode réel (type F)

z indique le nombre de mot à imprimer par ligne.

Définition des paramètres :

NOM	CAP	MODE	valeur de défaut	Légende
TIP	203	I	2	- indique le type de la donnée à sortir - valeurs possibles : 1, 2, 3, 4
FOR	204	I	158	- Format de sortie
NMO	205	I	1	- Nombre de mots à imprimer
ICA	206	I	1	- Position dans le common du premier mot à imprimer.
POS	207	I	1	- Position sur le listing des données à imprim
LIS	208	I	100	- Code de l'unité de sortie des informations. valeurs possibles 100, 101, 103.
RTU	209	I	100	- Code de l'unité de sortie après impression des cap. Mots demandés.

SAU	210	I	-1	- Code définissant le contrôle de saut du papier valeurs possibles : -1, 2, -3,...,13.
NBM	211	I	1	- indique le nombre de mots à imprimer par ligne

Définition de la commande :

ALT PHR : ECR, VERB, I(203)TIP2, I(204)FOR158, I(205)NMO1, I(206)ICA1, I(207)POS1, I(208)LIS100, I(209)RTU100, I(210)SAU-1, I(211)NBM1, VERB'COPI';

Exemple :

Taper au clavier :

INT ECR LON, TYP3, NBC4, CAP15, TIP3, FOR24, ICA15, POS8, SAU-3;

PROJ
ET P
LAN
ANNE
E 19
73##
####

Après prise en compte de cette commande on aura à l'imprimante console :

PROJET PLAN ANNEE 1973##

Dans ce cas nous utilisons 2 phrases verbes et 1 phrase objet.

Exemple : On suppose le clavier sélecté

Taper à la console :

INT ECR LON, TYP2, TIP4, NBC62, FOR72, NMO10, NBM5, POS9, SAU-3;
123.11
566.
223.12
-12.07
152.32
007.81
000.24
555.55
238.21
####

On aura sur l'imprimante console :

123.11 566. 223.12 -12.07 152.32

7.81 .24 732.24 555.55 238.21

Remarque :

Il existe une autre phrase verbe utilisant le module SORT.

L'appel de cette phrase est :

SOR phrase objet, TIPm, FORn, NMOk, ICAj, PO Sp, LI Sa, RTUb, SAUx;

Cette commande est un cas particulier de ECR, elle ne permet d'imprimer qu'un mot par ligne, ses paramètres sont les mêmes que ceux définis dans ECR.

A-5 : Addition de deux entiers relatifs :

ADD, NBCn, FORm, INPa, LI Sb, PO Sp, RETr, RTUk, SAUx;

Cette commande permet de faire l'addition de deux entiers relatifs et d'imprimer le résultat à l'unité courante de sortie.

n : format d'entrée des 2 nombres.
m : format de sortie du résultat.
a : code de l'unité d'entrée.
b : code de l'unité de sortie.
p : position d'impression du résultat.
r et k : donnent réspectivement les unités de retour après lecture et écriture.
x : contrôle de saut du papier.

Définition des paramètres :

NOM	CAP	MODE	valeur de défaut	Légende
RET	198	I	0	Code de l'unité de retour pour lecture de nouvelles données.
INP	199	I	0	- Code de l'unté d'entrée pour lecture des deux nombres à additionner. Valeurs possibles : 0, 1, 3
	200	I	1	- Mode des nombres à ajouter (entier)
NBC	201	I		- Format des données d'entrée.
	202	I	1	- Pointe sur le mot du COMMON qui recevra la premiere donnée.
	203	I	1	- Mode de la donnée à sortir (entier)
FOR	204	I	2	- Format de la donnée de sortie
	205	I	1	- Nombre de mot à imprimer
	206	I	3	- Pointe sur le mot à sortir
POS	207	I	5	- Position sur le listingu du résultat. (nombre de blanc avant impression du résultat
LIS	208	I	100	- Code de l'unité de sortie du résultat Valeurs possibles 100, 101, 103.
RTU	209	I	100	- Utilisé par SORT pour définir le code de l'unité de sortie après impression.
SAU	210	I	-1	- Contrôle du saut de papier. Valeurs possibles, -1, -2, -3, ..., 13, (voir III-3-6)

* Les CAP 198 à 202 sont utilisés par le module ENTP et les autres par SORT.

Définition de la commande :

ALT PHR : ADD, LEV1, I(198)RETO, I(199)INPO, I(200)1, I(201)NBC, I(202)1, I(203)1, I(204)FOR2, I(205)1, I(206)3, I(207)POS5, I(208)LIS100, I(209)RTU100, I(210)SAU-1, PRO'ENTP, ADD, SORT;

Exemple : Nous supposons qu' l'unité d'E/S est la console.

Taper au clavier :

ADD, NBC4, FOR5, POS1;

1241

0173

permet de sortir de ENTP

résultat imprimer à l'imprimante console :

1414.

A-6 : Calcul d'une expression algébrique :

CAL, RES = expression algébrique, SAUx, TYPn, POSp, RTUA;

Cette phrase objet permet de faire imprimer le calcul d'une expression algébrique.

RES : adresse du résultat du calcul.

Définition des paramètres :

nom	CAP	MODE	Valeur de défaut	LEGENDE
TIT	1	I		permet grâce à la phrase verbe EXP d'imprimer un titre.
RES	100	I		Contient le résultat du calcul
TYP	203	I	2	Mode de la donnée à sortir (utilisé par le module SORT).
FOR	204	I	158	Format de sortie du résultat.
	205	I	1	nombre de mot à imprimer
	206	I	1	pointe sur le CAP à imprimer.

POS	207	I	1	Position sur le listing du résultat à imprimer
LIS	208	I	100	Code de l'unité de sortie du résultat. valeurs possibles : 100, 101, 103
RTU	209	I	100	Code de l'unité de sortie après impression du résultat.
SAU	210	I	-1	- Contrôle du saut de papier

* Les CAP 203 à 210 sont utilisés par le module SORT pour l'impression des résultats.

Définition de la commande :

ALTER PAR : CALL, LEV1, I(1)TIT' ', I(100)RES, I(203)TYP2, I(204)FOR158, I(205)1, I(206)100, I(207)POS, I(208)LIS100, I(209)RTU100, I(210) SAU-1, PRO'SORT';

Exemple : nous supposons que le clavier est sélecté.

taper sur le clavier :

EXP CAL, TITRE 'SURFACE DU CERCLE DE RAYON 3.141592 EST =', POS20, SAUO, TYP4, FOR159, RES=3.141592*3.141592*3.141592;

On aura sur l'imprimante console 1131 :

SURFACE DU CERCLE DE RAYON 3.141592 EST = 31.006278992

A-7 : Résolution d'un système d'équation linéaire par la méthode de GAUSS :

RES GAU, RANn, COE

a_{11}, a_{12}, a_{13}, b_1,

a_{21}, a_{22}, a_{23}, b_2

a_{32}, a_{32}, a_{33}, b_3 ;

n : donne le rang du système linéaire à résoudre.

Le système linéaire résolu est : AX=B

où $A = (a_{ij})$. $B = (b_i)$

La méthode utilisée est celle de GAUSS

Définition des paramètres :

NOM	CAP	MODE	Valeur de défaut	LEGENDE
RAN	1	I	0	Défini le rang du système d'équation linéaire à résoudre.
COE	2		1	Permet de transmettre les coefficients a_{ij} et b_j

Définition de la commande :

ALT PAR : RESGAU, LEV1, RANO, COE1, PRO'GAUS';

Le module GAU permet de déterminer et d'imprimer la valeur :

$$X = A^{-1}B$$

Exemple:1 : On suppose le clavier sélecté.

Taper à la console :

```
RESGAU, RAN3, COE
1.73, 8.24, 17, 8.31,
2.15, 8.21,  0, 0 ,
1.118, 824.22, 733,21;
```

On aura alors pour vecteur X à l'imprimante console

1.7185271
-.4500408
.5320754

L'exemple ci-dessus nous a permi de résoudre le système :

$$\begin{cases} 1.73\,X_1 + 8.24X_2 + 17X_3 = 8.31 \\ 2.15X_1 + 8.21X_2 = 0 \\ 1.118X_1 + 824.22X_2 + 733X_3 = 21 \end{cases}$$

Exemple 2 : pour résoudre le système d'équation :

$$\begin{cases} X + 7Y = 3 \\ X + 7Y = 3 \end{cases}$$

taper à la console :

1, 7, 3

1, 7, 3;

On aura alors à l'imprimante console le message

D V G

Effectivement ce système admet une infinité de solution (indétermination)

A-8 : Résolution d'une équation algébrique de degré n par la méthode de Bairstow :

RES BAI, DEGn COE a_n, a_{n-1}, ..., a_0, EPSk;

n : degré du polynome à résoudre

a_i : coefficient du polynôme

k : facteur de précision

avec un tel appel l'équation résolue est :

Définition des paramètres :

NOM	CAP	MODE	valeur de défaut	LEGENDE
DEG	10	I	0	- Donne le degré du polynôme à résoudre
EPS	21		10^{-3}	Facteur de précision
COE	22			pointe sur le CAP qui recevra le coefficient a_n
TIT	1			permet d'imprimer un titre.

Définition de la commande :

ALT PHR : RES BAI, LEV1, I(20)DEGO, (21)EPSI E-3, (22)COE, I(1)TIT' BAIR'; PRO BAIR

Le module BAIR permet de résoudre une équation algébrique de degré n par la méthode de Bairstow.

Exemple :; On suppose le clavier sélecté.

Pour résoudre l'équation :

$$5x^3 - 2x^2 + 1 = 0$$

taper au clavier de la console :

RES BAI, DEG3, COE5, -2, 0,1;

On aura alors à l'imprimante console :

438.7145638E-03 *475.8568406E-03*
438.7145638E-03 *-47.5856840E-02*
-47.7429091E-02

Nous voyons donc que cette équation admet 2 racines complexes et 1 réelle.

———— o ————

ANNEXE B

Etudes des gros programmes

Sur IBM 1130 de taille 8 K ne peuvent passer des programmes occupant 7820 mots mémoires en effet :

Taille de la mémoire :	8192 mots
Taille du superviseur :	1020 mots
Taille disponible :	7172 mots

Aussi pour faire exécuter des programmes dont le volume est supérieur à 7172 mots nous allons créer un fichier de nom PC I PT pour y stocker le programme principal lors de l'appel à un sous-programme.

Application de la méthode au calcul des lignes d'influences d'une poutre à 3 travées :

a) Rappel théorique :

Considérons une poutre à 3 travées de longueurs respectives l_1, l_2 et l_3, on se propose de calculer les lignes d'influences (1) des moments de flexion à l'abcisse x du premier appui, pour cela nous devons effectuer les calculs suivants :

1- Calcul des caractéristiques mécaniques de la poutre :

$$a_i = \int_0^{l_i} \left(1 - \frac{x}{l_i}\right)^2 \frac{dx}{EI(x)}$$

$$b_i = \int_0^{l_i} \frac{x}{l_i}\left(1 - \frac{x}{l_i}\right) \frac{dx}{EI(x)}$$

$$c_i = + \int_0^{l_i} \left(\frac{x}{l_i}\right)^2 \frac{dx}{EI(x)}$$

a_i, b_i et c_i : coefficients de souplesse.

(1) La ligne d'influence des moments à l'abcisse x est la fonction $M(x,\alpha)$, donnant le moment de flexion en x produit par une charge unité placée à l'abcisse α variable.

$$\omega'_i = \int_0^{l_i} \frac{\mu x}{l_i} \frac{dx}{EI(x)}$$

$$\omega''_i = -\int_0^{l_i} \frac{\mu}{l_i} (l_i - x) \frac{dx}{EI(x)}$$

E : Coefficient de YOUNG

I : moment d'inertie

μ *: moment de flexion dans la travée isostatique*

ω'_i *et* ω''_i *: angles de flexion.*

2- Résoudre le système d'équation linéaire donnant les moments sur appuis $M_i(\alpha)$

$$\begin{cases} M_0 = M_3 = 0 \\ l_i M_{i-1}(\alpha) + (C_i + a_{i+1}) M_i(\alpha) + l_{i+1} M_{i+1}(\alpha) = \\ \omega'_{i+1}(\alpha) - \omega''_i(\alpha) \quad \text{pour } \alpha = 1, 2 \end{cases}$$

3- Calcul des lignes d'influence :

$$M(x,\alpha) = \frac{x}{l} M_1(\alpha) + \mu(x,\alpha) \quad \text{pour } 0 < x < l_1$$

$$M(x,\alpha) = \mu(x,\alpha) + \left(1 - \frac{x}{l}\right) M_1(\alpha) + \frac{x}{l} M_2(\alpha) \quad \text{pour } l_1 < x < l_2$$

$$M(x,\alpha) = \left(1 - \frac{x}{l}\right) M_2(x) + \mu(x,\alpha) \quad \text{pour } l_2 < x < l_3$$

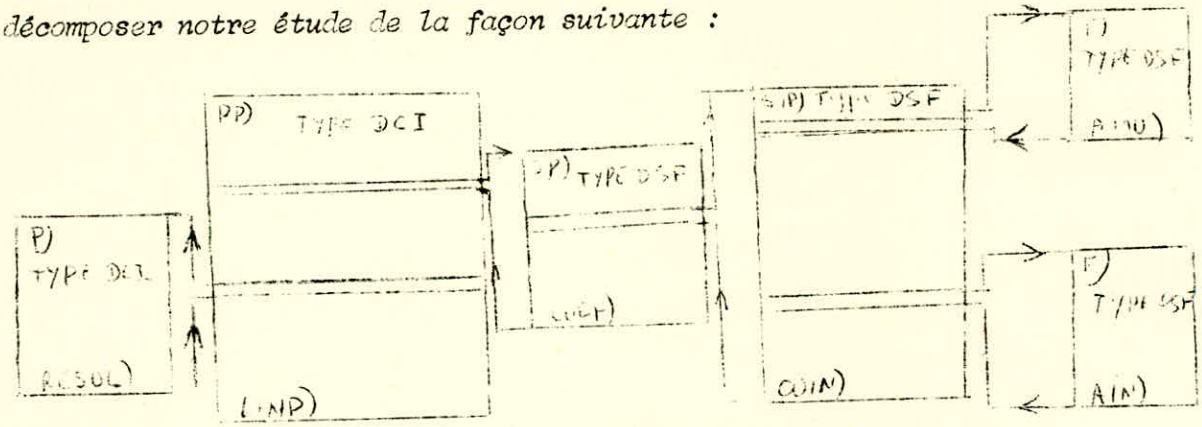
avec $\mu(x,\alpha) = \alpha\left(1 - \frac{x}{l}\right)$ si $x > \alpha$

$= x\left(1 - \frac{x}{l}\right)$ si $x < \alpha$

Cette dernière relation n'est valable que pour la travée où se trouve le point d'abcisse x, à l'extérieur : $\mu(x, \alpha) = 0$

b) Programmation :

L'écriture en un seul programme d'une telle tâche necessite une place mémoire plus grande que celle allouée par la machine aussi allons nous décomposer notre étude de la façon suivante :



* Le programme principal LINF taille 172DB
* COEF : taille 20D.B, calcul les coefficients du système (S)
  Il utilise pour déterminer les caractéristiques mécaniques le S/P d'intégration OGINT (méthode d'intégration de Simpson).
* AMU et AIN occupent réspectivement : 5 et 6 D.B, elles représentent les fonctions $\mu(x, \alpha)$ et $I(x)$.
* RESOL : programme de taille 64 D.B., il résoud par la méthode de GAUSS la système d'équation (S); il est appelé par LCHEX à la rencontre de cet appel le programme principal va être sauvegardé dans le fichier PCHPT ouvert sous DUP.

c) Définition de la commande :

ALTPAR : CAL LIG INF, LEV1, I(1)NC20, (2)LON48, 60, 48, I(300)N11, (301) SEC 6, 5, 15, 28, 33, 42, 45, 52, 56, 65, 75, (350)INE
0.05221847517, 0.07306129573, 0.07782213840,
0.06785296240, 0.04180180506, 0.07306129573,
0.8823880506, 0.07306129573, 0.0470101384,
0.0678 529624, 0.0778221384, PRO'LINF';

Définition des parmaètres :

NOM	CAP	MODE	Valeur de défaut	LEGENDE
NC	1	I	20	- Nombre d'intervalles par travée
LON	2		48	Longueur de la première travée
	3		60	deuxième travée
	4		48	Troisième travée
N	300	I	11	Nombre de discontinuité de la poutre
EC	301		6	Abcisse des discontinuités
			5	
			.	
			.	
			.	
			75	
INE	350		0.0522...	Inertie de la poutre pour chacun des tronçons délimité par 2 points de discontinuité.

Exemple : clavier sélecté.

Taper au clavier de la console :

CAL LIG INF, NC5, LON10, 30, 20, N3, EC10, 20, 30, INE1, 2, 3;

A la rencontre de cette commande nous avons exécution de LINF jusqu'à ce que les étapes 1) et 2) définies au rappel théorique soeint terminés (ona ainsi les moments de flexion aux appuits) ; puis le clavier se mettra en attente :

TRAVEE = 01, ABS = 10. (FDZ)

On aura alors impression à l'imprimante console de :

10.000	2.000	-.4277565
10.000	4.000	-.7485741
10.000	6.000	-.8555136
10.000	8.000	-.6416357
10.000	10.000	-.0000000

10.000	16.000	-2.1246896
10.000	22.000	-2.4702897
10.000	28.000	-1.8977404
10.000	34.000	-.9389902
10.000	40.000	.0000000
10.000	44.000	.3967475
10.0001	48.000	.5664445
10.000	52.000	.5522538
10.000	56.000	.3328898
10.000	60.000	.0000000

Taper au clavier :

TRAVEE = 00 (FDZ)

On sort du programme LINF

Interprétation des résultats :

$x = 10$, $\alpha = 2$ $M(x, \alpha) = -.4277565$

$x = 10$, $\alpha = 40$ $M(x, \alpha) = .0000000$

Nous avons ainsi les lignes d'influence à l'appui 1-

Recherche d'un module dans la zone UA :

T Poursavoire si un module existe dans la zone UA(utilitie AREA) il suffira de d lire au lecteur 442 les cartes de contrôles suivantes :

```
I  //  JOB
   //  XEWULETAA              1
   NAME 1
   :
   :
   :
   Carte vierge : fin de fichier carte :
```

NAME1, NAME2.... sont les noms de programmes à rechercher dans la zone utilisateur du disque.

LETAA : module stocké en format DCI qui permet de parcourir la LET

(LET : carte géographique de la U.A)

Les informations de sortie seront imprimées à l'unité 1132.

C O N C L U SI O N

Malgré les avantages qu'il présente PLAN ne doit pas être utilisé qu'après étude et choix, en effet l'analyse des possibilités et des contraintes imposées par ce système doivent être l'élément primordial de la décision, au même titre que l'étude des performances de l'ordinateur adopté.

Avant de rejetter systématiquement PLAN pour un système plus performant ne vous fiez pas au dictant :

"Qui peut le plus peut le moins"

Car n'oublions pas qu'un tel système occupera plus de place en mémoire donc déminuera la place allouée à l'utilisateur, de même le temps d'exécution d'un programme en sera grandement affecté.

-=-=-=0=-=-=-

-=-=-=-= BIBLIOGRAPHIE -=-=-=-

- *USER'S INTRODUCTION (IBM APPLICATION PROGRAM)*
  Problem Language Analysezr (PLAN)
- *APPLICATION Description Manual (IBM Application Program)*
  Problem Language Analysezr (PLAN)
- *Program Description Manual (IBM Application Program)*
  Problem Language Analysezr (PLAN)
- *1130 Problem Language Analysezr (PLAN)*
  Opérations Manual
- *Elements of IBM 1130 programming (W.T.PRICE)*
- *Cours d'Analyse Numérique de l'Ecole Centrale des arts et manufactures (M. Henri . VEYSEYRE)*
- *Cours d'Informatique de l'Ecole Nationale Polytechnique d'Alger (M.ADIBA)*