Conception et réalisation d'un système graphique interactif autour d'une architecture à microprocesseurs multiples : application dans le contexte du système réparti CORAIL

Alejandro Villavicencio Ramirez

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR INGENIEUR
«Génie Informatique»

par

A. VILLAVICENCIO RAMIREZ

CONCEPTION ET REALISATION D'UN SYSTEME GRAPHIQUE INTERACTIF AUTOUR D'UNE ARCHITECTURE A MICROPROCESSEURS MULTIPLES: APPLICATION DANS LE CONTEXTE DU SYSTEME REPARTI CORAIL.

Thèse soutenue le 12 septembre 1979 devant la commission d'examen.

L. BOLLIET	Président
F. ANCEAU R. A. GUEDJ M. LUCAS	Examinateurs

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR INGENIEUR
«Génie Informatique»

par

A. VILLAVICENCIO RAMIREZ

CONCEPTION ET REALISATION D'UN SYSTEME GRAPHIQUE INTERACTIF AUTOUR D'UNE ARCHITECTURE A MICROPROCESSEURS MULTIPLES: APPLICATION DANS LE CONTEXTE DU SYSTEME REPARTI CORAIL.

Thèse soutenue le 12 septembre 1979 devant la commission d'examen.

L. BOLLIET	Président
F. ANCEAU R. A. GUEDJ M. LUCAS	Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET

M. Georges LESPINARD

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Electronique - automatique
	BESSON Jean	Chimie minérale
	BLOCH Daniel	Physique du solide - cristallographie
	BONNETAIN Lucien	Génie chimique
	BONNIER Etienne	Métallurgie
*	BOUDOURIS Georges	Electronique - automatique
	BRISSONNEAU Pierre	Physique du solide - cristallographie
	BUYLE-BODIN Maurice	Electronique - automatique
	COUMES André	Electronique - automatique
	DURAND Francis	Métallurgie
	FELICI Noël	Electronique - automatique
	FOULARD Claude	Electronique - automatique
	LANCIA Roland	Electronique - automatique
	LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie - physique
	PAUTHENET René	Electronique - automatique
	PERRET René	Electronique - automatique
	POLOUJADOFF Michel	Electronique - automatique
	TRAYNARD Philippe	Chimie - physique
	VEILLON Gérard	Informatique fondamentale et appliquée

* en congé pour études

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuël	Electronique - automatique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electronique - automatique
	GUYOT Pierre	Métallurgie physique
	LACOUME Jean-Louis	Electronique - automatique
	JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM.	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	ZADWORNY François	Electronique - automatique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	CHARTIER Germain	Electronique - automatique
	CHIAVERINA Jean	Biologie, biochimie, agronomie
	IVANES Marcel	Electronique - automatique
	LESIEUR Marcel	Mécanique
	MORET Roger	Physique nucléaire - corpusculaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie Physique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

E.N.S.E.R.G.

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

E.N.S.E.G.P.

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	Centre national recherche scientifique

A mes parentes,
mes frères,
mes soeurs,
mon oncle Luis
et sa famille.

Je tiens à remercier,

Monsieur L. BOLLIET, Professeur à l'Université de Grenoble qui a bien voulu me faire l'honneur de presider le jury,

Monsieur F. ANCEAU, Maître de conférences à l'Institut National Polytechnique de Grenoble, qui a bien voulu m'accepter au sein de l'équipe de recherche en architecture d'ordinateurs qu'il dirige, ainsi que pour les conseils et observations qui m'a fait tout au long de ce travail,

Monsieur R.A. Guedj, Directeur du Laboratoire de Communication Homme-Machine de la Thomson-CSF, qui a bien voulu juger mon travail et faire partie du jury,

Monsieur M. Lucas, Maître de Conférences à l'Université de Nantes et ancien résponsable de l'Equipe de Techniques Graphiques à l'IMAG, pour les suggestions et intérêt qu'il m'a accordé tout au long de ce travail,

Les membres de l'équipe de recherche en architecture d'ordinateurs avec qui j'ai eu des échanges fructueux, spécialement M. Schoellkopf qui a eu le soin de corriger la rédaction de cet ouvrage.

M. Boughlam, auteur du système de traitement de textes REDAK, avec lequel a été réalisé cet ouvrage.

TABLE DE MATIERES

III : LE PROCESSEUR LOGIQUE DE MODELISATION (PLM)

IV : LE PROCESSEUR LOGIQUE DE PREPARATION (PLP)

V : LE PROCESSEUR LOGIQUE D'AFFICHAGE (PLA)

INTRODUCTION

Le travail présenté ici décrit les principes de conception et de réalisation d'un système graphique interactif implémenté dans le contexte d'un système informatique réparti de manière fonctionnel: le Système CORAIL (POU-76).

Les motivations de ce travail peuvent être résumées comme étant la recherche d'organisation et de mise en harmonie de deux mondes: celui des utilisateurs et celui des architectes de machines informatiques.

Dans le domaine des utilisateurs, les applications modernes demandent au système graphique d'avoir une grande puissance de traitement ainsi que d'être muni de consoles graphiques qui aient des fonctions spécialisées pour les besoins de l'application concernée.

Dans le domaine de l'architecture de machines, tout système informatique ainsi que le travail qu'il exécute peuvent être considérés comme étant un ensemble de niveaux qui s'interprètent mutuellement. La définition correcte de ces niveaux est l'oeuvre de l'architecte qui doit chercher à minimiser la complexité de chacun d'entre eux. Pour cela, les grands développements de la technologie de circuits intégrés à grande échelle et à très grande échelle, permettent à l'architecte d'aujourd'hui de disposer d'organes spécialisés dans l'exécution d'une fonction donnée. Par exemple, les microprocesseurs sont spécialisés dans le traitement, le PIA, PIO, ACIA,... sont spécialisés dans les entrées/sorties, ...etc.

Dans ce travail nous définissons un système graphique interactif comme étant un ensemble de mécanismes logiciels et matériels destinés à établir et contrôler la communication entre l'homme et l'ordinateur par le biais de dispositifs d'affichage.

Pendant la conception, le double but que nous nous sommes fixé a été de:

- Offrir aux utilisateurs des outils de programmation de haut niveau (voire langages évolués) bien adaptés à leur besoins, ainsi que de garantir l'indépendance vis-à-vis des caractéristiques de la console graphique à utiliser.

- Définir une architecture multimicroprocesseur qui soit performante et modulaire afin de pouvoir adapter et exploiter au maximum les caractéristiques de tout type de console graphique évoluée et implémenter des mécanismes destinés à palier les fonctions manquant dans le cas de consoles peu évoluées.

Ce travail se décompose en cinq chapitres. Dans le premier chapitre nous décrivons la méthode de conception suivie, ainsi que le découpage fonctionnel qui nous a guidé lors de la répartition des fonctions du système graphique interactif dans le contexte du système CORAIL.

Dans le deuxième chapitre nous définissons le formalisme de description de l'architecture du système graphique interactif. Une présentation générale de l'architecture à microprocesseurs multiples ainsi que de divers mécanismes d'implémentation sont aussi montrés.

Le troisième chapitre est consacré à la définition et à la déscription des mécanismes matériels et logiciels qui permettront de réaliser la fonction de modélisation.

Dans le quatrième chapitre, nous montrons une description de mécanismes logiciels et matériels destinés à exécuter la fonction de mise en page des objets créés par la modélisation. Un contrôle du flux de l'information est fait afin d'éviter la

saturation des ressources du système.

Le cinquième chapitre contient une étude des diverses parties des consoles graphiques ainsi que de leur influence possible lors de la conception du système graphique interactif. Ici, nous montrons aussi une description des mécanismes qui permettront d'exploiter et contrôler le terminal graphique 30-N de DEC afin d'obtenir le dessin sur l'écran.

CHAPITRE I

METHODOLOGIE DE CONCEPTION DU SYSTEME GRAPHIQUE INTERACTIF SGI

1.- Introduction.
2.- Méthodologie de conception suivie:
    La méthode intermédiaire.
3.- Découpage fonctionnel du SGI.
4.- Le logiciel GRIGRI: Niveau de départ de la conception du SGI.
5.- Place du SGI dans le Système CORAIL.
6.- Notion de Processeur Logique.
7.- Structure du SGI.
8.- Synthèse.

I.- METHODOLOGIE DE CONCEPTION DU SYSTEME GRAPHIQUE INTERACTIF (SGI)

1.- Introduction

Nous définissons un système graphique interactif comme étant un ensemble de mécanismes logiciels et matériels, destinés à établir et contrôler la communication entre l'homme et l'ordinateur par le biais de dispositifs d'affichage.

Dans tout Système Informatique il est possible de distinguer une structure hiérarchisée de composants fonctionnels.

La hiérarchie peut provenir de l'existence d'un graphe acyclique d'appels inter-modules, d'un arbre généalogique de processus, d'une imbrication de niveaux d'interprétation, d'une structure d'adressage arborescente, etc.

Dès le début de la conception du Système, il est très important de bien définir cette hiérarchie de composants fonctionnels afin de déterminer un découpage en niveaux d'abstraction. Il reste ensuite à choisir une méthode de construction ou de réalisation, par exemple:

- La méthode descendante qui progresse des spécifications externes (Projet) aux fonctions primitives (Moyens). C'est-à-dire qu'on spécifie toutes les fonctions nécessaires pour construire un niveau n avant de les définir dans le niveau inférieur n-1.

- La méthode ascendante qui utilise les fonctions déjà définies (Moyens) dans un niveau n-1 pour bâtir les fonctions du niveau supérieur n.

L'expérience montre que la méthode descendante est plus naturelle puisqu'avant de définir le fonctionnement interne d'un élément, on préfère connaître son rôle précis.

Cependant la pratique montre qu'on ne respecte jamais jusqu'au bout aucune des méthodes mentionées.

Dans la méthode descendante on a le danger inhérent d'avoir des caractéristiques attractives qu'il peut être difficile, voire impossible, de réaliser sur le matériel disponible.

Une méthode ascendante peut avancer à travers plusieurs niveaux de conception avant qu'on se rende compte que le prototype final n'est pas celui que l'utilisateur avait demandé.

Les bons architectes de machines ont en général une idée de ce que seront les niveaux extrêmes: celui d'en haut et celui d'en bas; et d'une manière consciente ou inconsciente ils utilisent cette information pendant le processus de conception. Par exemple, un architecte d'expérience qui utilise la méthode descendante ne va inclure aucune caractéristique dans le niveau haut à moins qu'il ne soit sûr qu'on pourra la bâtir d'une façon ou d'une autre.

Donc, la façon dont la nature du but à atteindre influence chaque pas de la méthode de conception est très variable et doit faire l'objet d'une grande attention. Cette influence peut être schématisée dans la figure suivante.

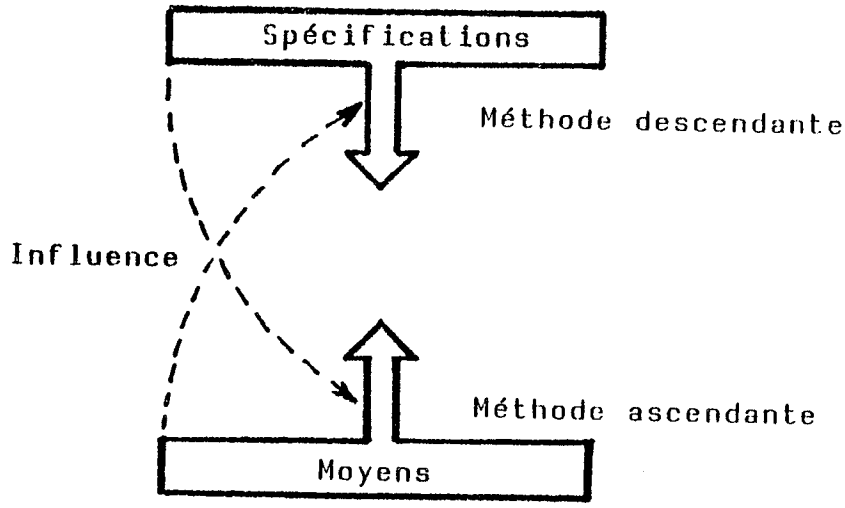


fig. 1.1 Influence dans deux méthodologies de conception.

2 - Méthode de conception suivie: La méthode intermédiaire.

Dans la conception du Système Graphique Interactif (SGI), nous avons suivi une méthode appelée Méthode Intermédiaire et qui consiste à définir d'abord un niveau intermédiaire dont la conception doit être indépendante aussi bien des fonctions que le Système doit réaliser que des moyens disponibles.

Ce niveau intermédiaire constitue donc l'interface entre les fonctions à réaliser et le matériel disponible pour les exécuter.

Après la définition de ce niveau, la méthode propose de travailler dans les deux phases suivantes:

- Conception de sous-systèmes (logiciels) dont chacun a comme fonction de transformer la représentation d'un algorithme écrit en langage de haut niveau, dans une autre représentation codée avec des objets déjà définis dans le niveau intermédiaire. Cette phase permet d'offrir des langages évolués bien adaptés aux besoins des utilisateurs.
  Cette phase ne considère que les caractéristiques externes du Système.

- Réalisation matérielle du Système. C'est-à-dire l'étude et la réalisation des mécanismes pour l'exécution des sous-systèmes ainsi que pour l'interprétation du niveau intermédiaire. Cette réalisation peut se faire en appliquant la méthode descendante.
  Dans cette phase on ne considère que les propriétés internes de la machine et on va essayer de minimiser, voire éliminer, toutes les contraintes technologiques et économiques.

Un exemple de machine conçue en utilisant la méthode intermédiaire est la Burroughs B1700 (WIL-72) où:

- Le niveau intermédiaire est celui défini par l'ensemble des S-Machines.

- Les sous-systèmes ne sont que des traducteurs qui vont traduire des programmes écrits en langage évolué en d'autres programmes écrits en langage de la S-Machine correspondante.

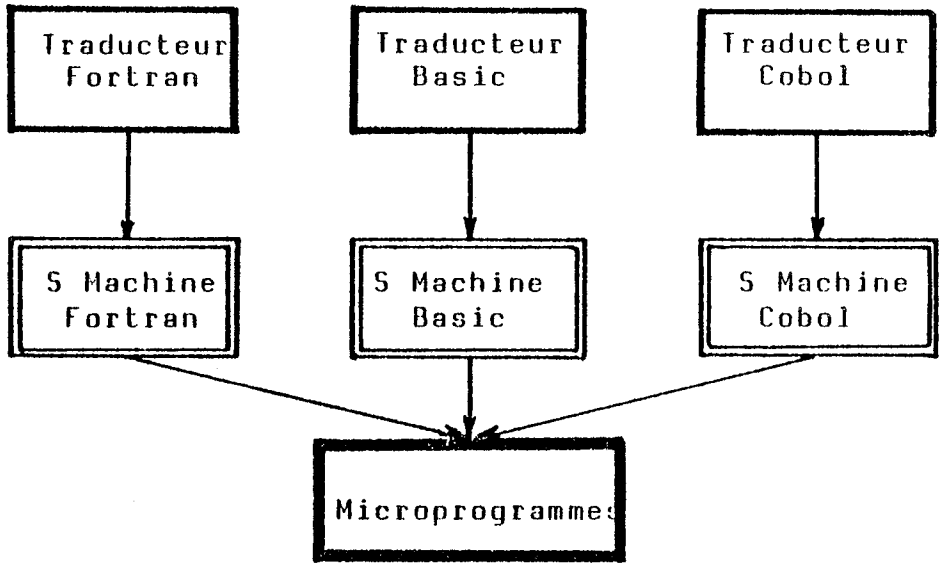


Fig. 1.2 Conception de la machine B1700.

L'objet engendré pour un traducteur est interprété par la S-Machine qui à son tour est interprétée par des microprogrammes qui eux sont interprétés par les niveaux inférieurs de la machine.

La méthode intermédiaire est aussi très utilisée dans la conception de compilateurs, par exemple dans le projet MU5 (LAV-77), dont le niveau intermédiaire est celui défini par un langage intermédiaire.

Dans notre travail, avant de définir le niveau intermédiaire, nous allons faire une étude des fonctions que doit réaliser tout système graphique interactif, afin de bien identifier ces fonctions et pouvoir les classer d'accord à leur contexte d'exploitation.

3.- Découpage fonctionnel du Système Graphique Interactif.

Plusieurs auteurs (MIF-78) ont déjà essayé de faire un découpage des fonctions que doit réaliser un système graphique interactif, notamment R. A. Guedj et al (GUE-76) qui considère que l'interface entre un programme d'application et la console graphique est constituée de deux fonctions:

- Une fonction de Mise en forme (Modeling System) qui permet de passer des informations de la structure de données de l'application aux informations graphiques et vice-versa.

- Une fonction graphique proprement dite (Core System) qui permet d'afficher sur l'écran une représentation des informations graphiques qui lui sont transmises et de recueillir les données graphiques destinées au programme d'application via la fonction de Mise en forme.

Les relations entre ces fonctions peuvent être schématisées dans la figure suivante:

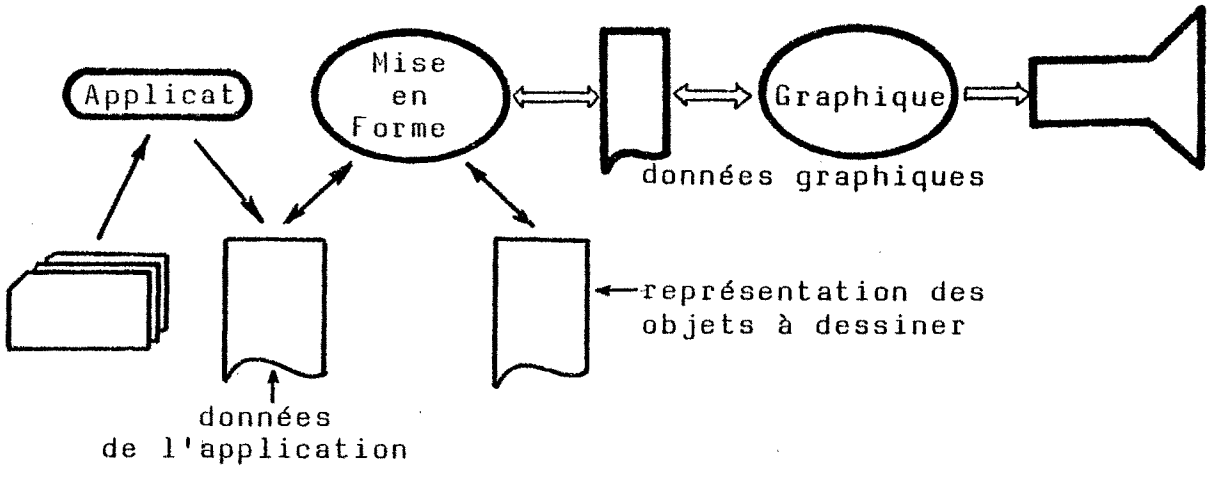


Fig. 1.3 - Découpage fonctionnel proposé par R.A. Guedj et al.

D'acord avec les études réalisées par M. Lucas (LUC-77) nous considérons que dans un système graphique interactif on peut trouver les fonctions suivantes:

- L'Application (utilisateurs).
- La Mise en forme ou Modélisation.
- La Préparation à la visualisation.
- L'Affichage élémentaire.

Au niveau de l'Application, nous trouvons les utilisateurs dont les origines sont très diverses, par exemple on utilise des organes d'affichage dans les domaines de la physique nucléaire, recherche médicale, contrôle de processus, arts graphiques, ... etc.

Le rôle d'une fonction d'application est de traiter le programme de l'application afin d'engendrer une structure de données appelée structure de données de l'application.

La Mise en forme ou Modélisation est la fonction qui, à partir des informations fournies par le programme d'application, va coder les objets (scènes) qui permettront plus tarde d'obtenir une représentation graphique. Le code à utiliser peut être entièrement indépendant du dessin proprement dit.

La fonction de Préparation à la visualisation se divise en deux phases: le décodage et la mise en page.

Le décodage consiste à composer, à partir de la scène, une représentation graphique à deux dimensions que nous appelons fichier graphique. En général, celui-ci est codé à base d'entités graphiques élémentaires telles que points, segments et caractères.

La tâche de la mise en page consiste à disposer le fichier graphique sur une surface de visualisation fictive afin d'obtenir un objet appelé dessin virtuel. En général, la surface

de visualisation fictive est structurée sous la forme d'une liste appelée liste d'affichage virtuelle.

La fonction d'Affichage va engendrer une liste d'affichage réelle à partir des indications contenues dans la liste d'affichage virtuelle. Cette fonction est très dépendante de l'organe d'affichage puisque la gestion de l'écran est faite par elle même.

Une structuration de ces fonctions conduit à la hiérarchie suivante:

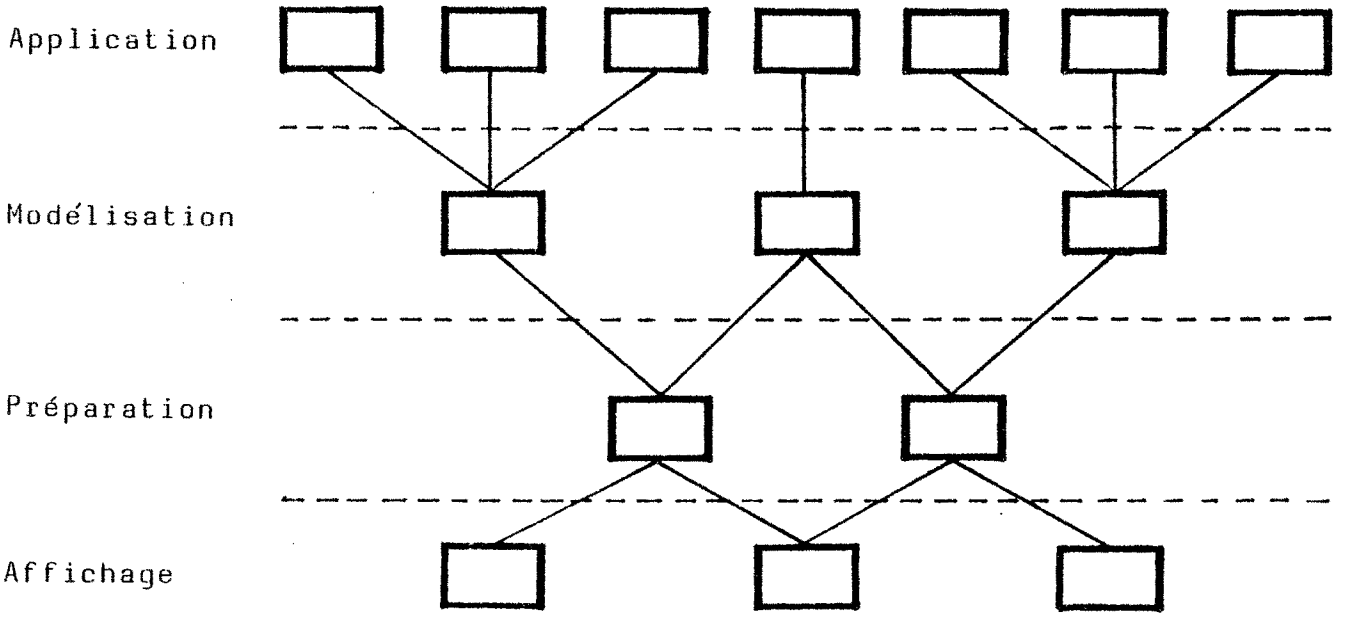


Fig. 1.4 Hiérarchie fonctionnelle dans un système graphique interactif.

Dans cette hiérarchie, les différents niveaux ont pour rôle de transcoder les informations transmises et de les restituer sous une forme utilisable, soit vers un niveau inférieur lors de

l'affichage, soit vers un niveau supérieur lors du dialogue.

3.1 Classement des fonctions du SGI.

Le principe qui nous a guidé pour classer les fonctions, est basé sur une étude approfondie de leur nature ainsi que de leurs relations avec l'environnement d'exploitation.

Les origines des domaines d'application des consoles graphiques étant aussi nombreuses que différentes, entraînent que l'on ne peut pas définir une fonction de Modélisation universelle adaptable à toutes les applications. Par contre, on peut chercher à regrouper dans une même classe, les applications ayant des bases communes de telle sorte que l'on pourra concevoir une fonction de modélisation bien adaptée à cette classe d'application.

Au niveau de la fonction de Préparation à la visualisation, l'expérience montre que la partie de décodage est très dépendante du code utilisé par la Modélisation. La diversité de ces codes (cf. III.2.1) rend difficile la normalisation de la partie de décodage. Par contre la mise en page peut être définie sans ambiguité, c'est-à-dire que l'on peut normaliser, puisque les objets (fichier graphique) produits par le décodage ne sont composés que de points, segments et caractères.

La diversité des technologies des consoles graphiques entraîne la difficulté de concevoir une fonction unique d'Affichage. Par contre, on peut concevoir une fonction d'affichage qui soit spécialisée dans un seul type de consoles graphiques.

Ces considérations nous permettent d'isoler les fonctions qui

peuvent être exécutées avec un certain degré de parallélisme et de les classer en deux catégories:

- Fonctions liées au contexte d'exploitation:

  - La Modélisation qui dépend fortement de l'application concernée.

  - La partie décodage de la Préparation à la visualisation qui dépend du code utilisé par la Modélisation.

  - L'Affichage élémentaire qui dépend des caractéristiques du terminal graphique.

- Fonctions standards:

  - La partie mise en page de la Préparation qui peut être indépendante tant de l'application que du type de terminal graphique.

L'indépendance de cette dernière fonction permet de définir un langage de mise en page, lequel peut être vu comme le niveau intermédiaire, point de départ de la conception du SGI.

Dans la suite de notre travail, le décodage de la Préparation à la visualisation, sera consideré comme une partie de la fonction de Modélisation.

4 - Le logiciel GRIGRI:
Niveau de départ de la conception du SGI.

Les dialogues que nous avons eu avec l'équipe de Techniques Graphiques de l'IMAG nous ont permis de connaître leur logiciel de mise en page: GRIGRI (LED-77).

Le principe de conception du logiciel GRIGRI a été de chercher l'indépendance vis-à-vis de son contexte d'utilisation (application, matériel, système d'exploitation, ... etc.). Ceci permet de le choisir comme le niveau intermédiaire, point de départ de la conception du SGI.

Le logiciel GRIGRI permet de:

-Offrir aux utilisateurs une programation indépendante tant de l'application que de la console graphique à utiliser.

-Fournir aux utilisateurs des mécanismes de description statique des objets (dessins) en utilisant des structures de données élémentaires.

- Fournir aux utilisateurs une grande facilité de programmation du dialogue.

Une fois défini le niveau intermédiaire, la méthode propose la conception de deux phases suivantes:

- Conception de sous-systèmes, logiciels de modélisation ou de description, dont chacun va permettre d'offrir à une classe d'utilisateurs un langage évolué bien adapté à ses besoins. Chaque sous-système va transformer la représentation d'un algorithme écrit dans ce langage évolué, dans une autre représentation, codée à base des

primitives du langage GRIGRI.

- Conception et réalisation matérielle du Système Graphique Interactif. C'est-à-dire l'étude et la réalisation à l'aide du matériel disponible ou à définir, des mécanismes pour l'exécution des sous-systèmes ainsi que pour l'interprétation des primitives GRIGRI.

Ainsi, la structure du Système Graphique Interactif est divisée en deux parties et la réalisation sera plus aisée si, dans chaque partie, les outils disponibles ou à définir sont bien adaptés aux fonctions à réaliser.

Le produit final sera plus facilement modifiable dans les deux sens: ajouter un nouveau sous-système pour traiter une classe d'application non envisagée au départ ne posera aucun problème. Réciproquement, un changement du matériel n'affectera pas les sous-systèmes puisque seuls les niveaux inférieurs du Système devront être remplacés.

Dans la suite, nous allons faire une étude des fonctions que doit réaliser le Système Graphique Interactif afin de répartir ces fonctions dans le cadre du Système CORAIL.

5.- Place du SGI dans le Système CORAIL.

Dans la philosopohie du Système CORAIL, la fonction-système Graphique sera exécutée par un Processeur Fonctionnel que nous appelons Processeur Fonctionnel Graphique (PFG). Ce Processeur sera spécialisé dans la gestion de la ressource-système Graphique; c'est-à-dire de l'ensemble d'organes d'affichage et de dessin.

Ceci pose la question de savoir quelles fonctions du SGI seront exécutées par le PFG et quelles seront exécutées par le Processeur de Traitement. Il s'agit donc de trouver la meilleure solution pour répartir les fonctions du SGI parmi le PFG et le(s) Processeur(s) de Traitement.

5.1 Choix de la répartition des fonctions.

Pour déterminer le meilleur choix de répartition, il faut tenir compte de plusieurs facteurs dont les plus importants sont:

- puissance de calcul nécessaire,
- débit de l'information à travers l'interface CORAIL,
- niveau du protocole de communication,
- découpage des caractéristiques et nature d'organes d'affichage,
- nature des fonctions à réaliser,
- nature des relations entre ces fonctions.

Il faut constater que ces facteurs peuvent être en conflit entre eux. Une étude approfondi sera faite afin de choisir une solution raisonable.

1) Si l'on décide que Modélisation, Préparation et Affichage sont exécutées dans le PFG, ceci entraîne:

- Le PFG doit avoir une grande puissance de calcul, puisqu'il doit exécuter la Modélisation et tout le traitement graphique.

- Le temps de réponse du PFG est fortement dépendant de la charge momentanée du SGI, c'est-à-dire du nombre d'utilisateurs qui travaillent en même temps, au sens de la multiprogrammation, avec le PFG.

- Le niveau de la structuration de l'information à échanger entre le PFG et les Processeurs de Traitement, est plus élaboré, alors les échanges sont minimaux et le débit d'information à travers l'interface CORAIL sera faible.

- Le protocole de communication entre le PFG et ceux de Traitement sera de haut niveau puisqu'il doit permettre la construction et la gestion des objets structurés. L'expérience montre qu'il est difficile de définir un protocole adapté à plusieurs classes d'applications.

Dans cette solution, le Processeur Fonctionnel Graphique (PFG) ne peut pas être standardisé puisqu'il dépendra de l'application concernée.

2) Dans le cas où la Modélisation est exécutée par le Processeur de Traitement et les fonctions de Préparation et Affichage sont executées par le PFG, ceci entraîne:

- Le PFG ne va exécuter que du traitement graphique. Cette approche correspond bien à la philosophie de CORAIL.

- Il y aura une amélioration du temps de réponse du PFG puisqu'il a moins de travail à exécuter.

- La structure de l'information à échanger entre le PFG et les Processeurs de Traitement correspond à un niveau intermédiaire. Le débit des échanges devient moyen.

- La communication entre le PFG et les Processeurs de Traitement va utiliser un protocole de niveau intermédiaire que l'on peut normaliser. Ce protocole sera unique pour toutes les fonctions de Modélisation.

3) Dans le cas où la Modélisation et la Préparation sont exécutées par le Processeur de Traitement et l'affichage est exécuté par le PFG alors:

- Le Processeur de Traitement sera très chargé.

- Le PFG ne va exécuter que la fonction d'affichage, pourtant il fera très peu de traitement graphique.

- La structuration de l'information à échanger entre le PFG et les Processeurs de Traitement, est d'un bas niveau et il y aura un fort débit de transfert à travers l'interface CORAIL.

- Le protocole de communication sera plus élémentaire, puisque l'information à échanger ne va consister qu'en des ensembles d'instructions élémentaires qui seront decodées afin d'engendrer les commandes propres au terminal graphique.

Toutes ces considérations nous ont amené à répartir les fonctions du SGI de la manière suivante :

- La Modélisation, étant fortement dépendante de l'application sera laissée à la charge du Processeur de Traitement. La partie décodage de la Préparation à la visualisation, étant aussi dépendante de la Modélisation, sera

aussi executée par ce Processeur.

- La mise en page de la Préparation à la visualisation sera standardisée c'est-à-dire commune à toutes les classes d'applications et à tous les types de terminaux graphiques, elle sera executée dans le PFG.

- La fonction d'Affichage qui est dépendante du type de terminal graphique sera executée aussi dans le PFG.

La répartition des fonctions du SGI est montrée dans la figure 1.3.

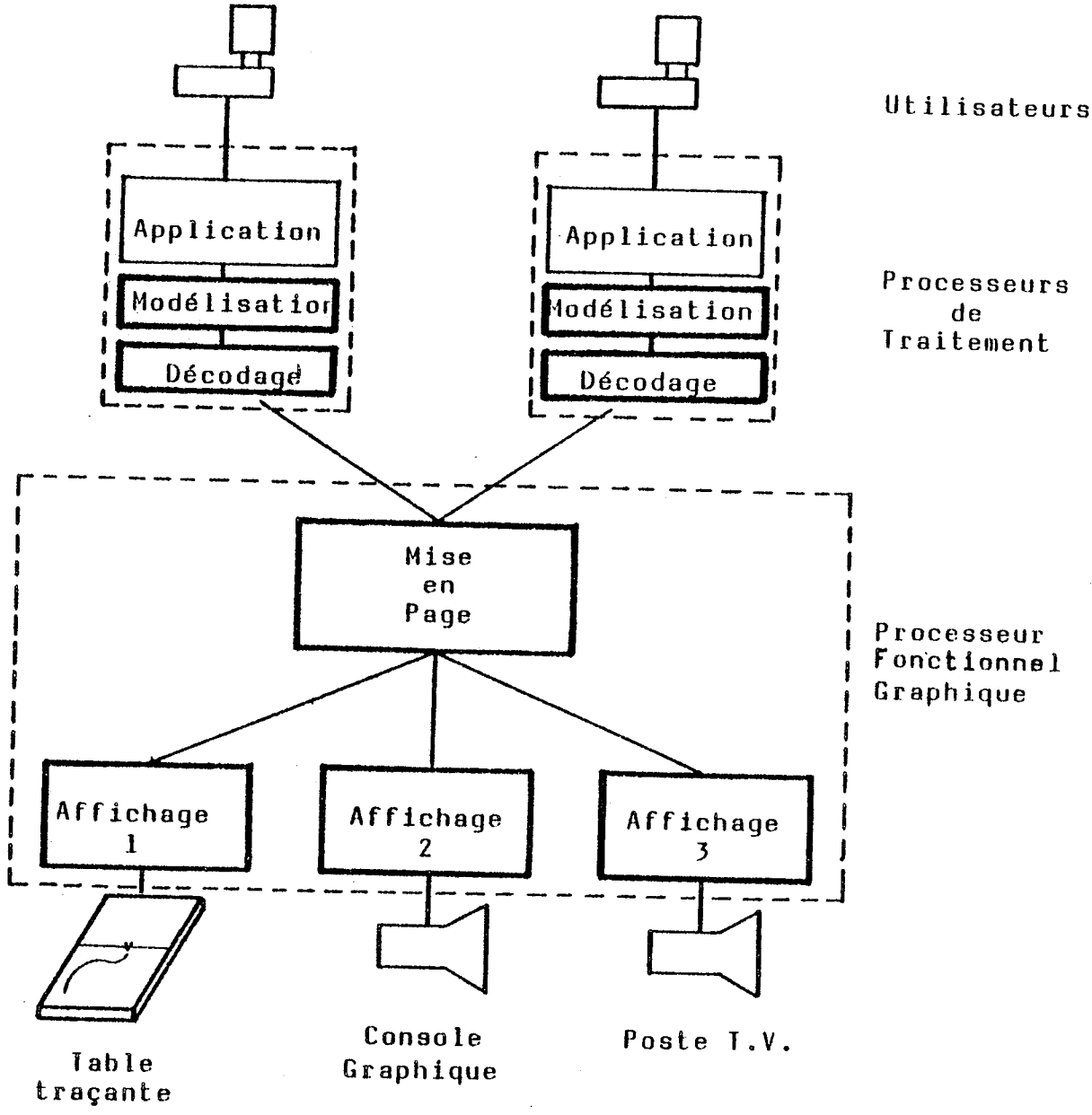


Fig. 1.3 Répartition des fonctions du SGI dans le contexte du Système CORAIL.

Ce découpage permet d'avoir dans le Système CORAIL un Processeur Fonctionnel Graphique unique et universel pour toutes les classes d'applications

5.2 - Analyse des interactions.

Les travaux de Foley et Wallace (FOL-74) mettent en évidence les types des analyses des interactions que l'on trouve dans tout système graphique interactif.

Nous proposon de répartir l'analyse des interactions du SGI de la façon suivante:

Dans le Processeur de Traitement il y aura :

- Une analyse lexicale pour reconnaître les symboles de base du langage de commande de ce processeur.

- Une analyse syntaxique afin de valider les séquences créées par l'utilisateur à base des symboles de base du langage de commande.

- Une analyse sémantique pour créer les fichiers graphiques en utilisant les primitives du logiciel de mise en page GRIGRI (langage de commande du PFG).

Dans le Processeur Fonctionnel Graphique on fera:

- Une analyse sémantique pour composer les scènes bidimensionnelles sur des surfaces de visualisation fictives.

- Une analyse lexicale pour reconnaître et valider les informations provenant des dispositifs de dialogue tels que photostyle, tablette, ... etc.

5.3 - Mode d'exploitation du PFG.

Pour mieux rentabiliser l'utilisation du PFG, il faut qu'il soit capable de répondre à tout instant aux demandes de service des utilisateurs. Il sera donc exploité en mode multiprogrammé.

La philosophie du Système CORAIL prévoit que les requêtes de demande de service peuvent arriver en mode asynchrone. Le PFG sera donc muni d'un Superviseur de communication (Annexe 1) qui lui permettra de gérer ces requêtes. En plus il sera muni des mécanismes de description et d'identification pour manipuler les objets créés par l'utilisateur via la Modélisation.

6. - Notion de Processeur Logique.

Une fois définies les fonctions à réaliser par le Système Graphique Interactif SGI: La Modélisation, la Préparation à la visualisation et l'Affichage, nous proposons d'associer à chaque fonction un processeur logique spécialisé dans l'exécution de celle-ci.

Nous définisons un processeur logique comme une entité formelle représentant le regroupement de certains travaux à réaliser. Il peut être vu comme une machine virtuelle capable de:

- Gérer une représentation de chacun des composants fonctionnels (objets).
- Fournir des opérateurs pour manipuler ces objets.
- Contrôler l'utilisation de ces opérateurs, d'après des règles de cohérence.

La conception et la réalisation de chaque processeur logique correspond à la deuxième phase de la méthode intermédiaire et

elles seront faites en utilisant la méthode descendante (SCH-77).

La conception de chaque processeur logique va commencer par une spécification externe de la fonction à réaliser. Ceci correspond à la définition du langage de commande de ce processeur.

Une fois défini le langage de commande, on continue dans la recherche des mécanismes d'interprétation de ce langage jusqu'à ce que l'on arrive à la concrètisation du processeur logique qui pourra être entre autres:

- Sur un processeur physique existant, par exemple un microprocesseur.

- Sur un processeur physique à définir, par exemple un système multi-microprocesseur.

- Comme étant une tâche particulière d'un processeur ou groupe de processeurs physiques.

Le rôle qu'aura chaque processeur logique sera décrit dans la suite:

6.1 Processeur Logique de Modélisation : PLM

La tâche d'un PLM est de manipuler la structure de données de l'application pour coder à partir des informations fournies par le programme d'application des objets en deux dimensions que nous appelons fichier graphique. Celui-ci constitue l'interface avec le Processeur Logique de Préparation PLP.

La fonction de Modélisation étant fortement dépendante de

l'application nous conduit à concevoir autant de Processeurs Logiques de Modélisation que de classes d'application. Ainsi dans le SGI, un PLM ne sera spécialisé que dans une classe d'application.

La notion de PLM permet d'offrir aux utilisateurs d'une classe d'application un langage évolué bien adapté à leurs besoins. Ce langage peut être vu comme le langage de commande du PLM et il va comporter des primitives qui permettront, d'une part la description formelle des objets élémentaires (vocabulaire de base) et d'autre part la définition des règles de composition (grammaire).

D'une façon plus formelle, un PLM va transformer la représentation d'un algorithme écrit dans un langage évolué, dans une autre représentation codée à base de primitives d'un langage de mise en page.

La philosophie de conception du logiciel GRIGRI (LED-77) nous permet de l'utiliser comme le langage de mise en page pour coder les fichiers graphiques.

6.2 - Processeur Logique de Préparation : PLP.

Dans I.3.1 nous avons considéré que la fonction de mise en page peut être une fonction standard, indépendante de l'application et de la nature des terminaux graphiques. Ceci aura l'avantage de ne concevoir qu'un seul PLP dans le SGI.

L'interface entre les PLM et le PLP, sera donc unique de telle façon que les PLM vont utiliser le même langage pour coder les fichiers graphiques. Ce langage est celui de GRIGRI et il peut être vu comme le langage de commande du PLP. On constate ainsi que le PLP devient une véritable machine GRIGRI.

Le PLP fera donc la mise en page des objets créés par les PLM. C'est-à-dire qu'il va les disposer sur une surface de visualisation fictive.

La mise en oeuvre des fonctions d'effacement sélectif et d'identification d'objets dans le cas de dialogue, implique la nécessité de disposer d'une représentation structurée de l'image présente sur l'écran. Alors le PLP sera muni d'un mécanisme d'identification pour pouvoir accéder à certaines parties de la représentation graphique.

La tâche du PLP consiste alors à transformer la représentation de l'algorithme codé en langage GRIGRI, en une représentation plus élémentaire (liste d'affichage virtuelle). Celle-ci va constituer l'interface avec le processeur logique d'affichage (PLA).

6.3 - Processeur Logique d'Affichage : PLA

L'expérience montre que la fonction d'affichage est dépendante du terminal graphique, puisque on souhaite toujours utiliser au maximum les qualités de celui-ci.

La diversité des technologies des consoles graphiques empêchent la définition d'une fonction unique d'affichage. Pour cela nous proposons d'avoir autant de PLA que de types d'organes d'affichage et de dessin dans le SGI, afin qu'un PLA ne soit spécialisé que pour un organe ou type d'organes d'affichage.

Dans le cas de terminaux comportant une mémoire d'entretien (cf. V.2.1), le PLA associé va créer sur cette mémoire une liste d'affichage réelle à partir des informations contenues dans la liste d'affichage virtuelle fournie par le PLP.

Une liste d'affichage réelle ne contiendra que des instructions graphiques interprétables directement pour le

terminal graphique.

L'obtention du dessin sur l'écran sera faite lorsque la "partie intelligente" (cf. V.2.3) du terminal, interprète le contenue de la mémoire d'entretien, c'est-à-dire la liste d'affichage réelle.

Si le terminal graphique n'a pas de mémoire d'entretien, alors le PLA va créer, à partir des indications contenues dans la liste d'affichage virtuelle, la suite de commandes propres à l'organe d'affichage afin d'obtenir le dessin final sur l'écran.

L'interface entre les PLA et le PLP est définie par l'ensemble des listes d'affichage virtuelle créées par le PLP. La façon dont le PLP va structurer une liste d'affichage virtuelle dépendra du type de terminal graphique, ceci dans le but d'exploiter au maximum les caractéristiques du terminal graphique.

Le langage à utiliser pour coder la liste d'affichage virtuelle constitue le langage de commande du PLA.

7.- Structure du SGI.

La structure du SGI est schématisée dans la figure 1.6

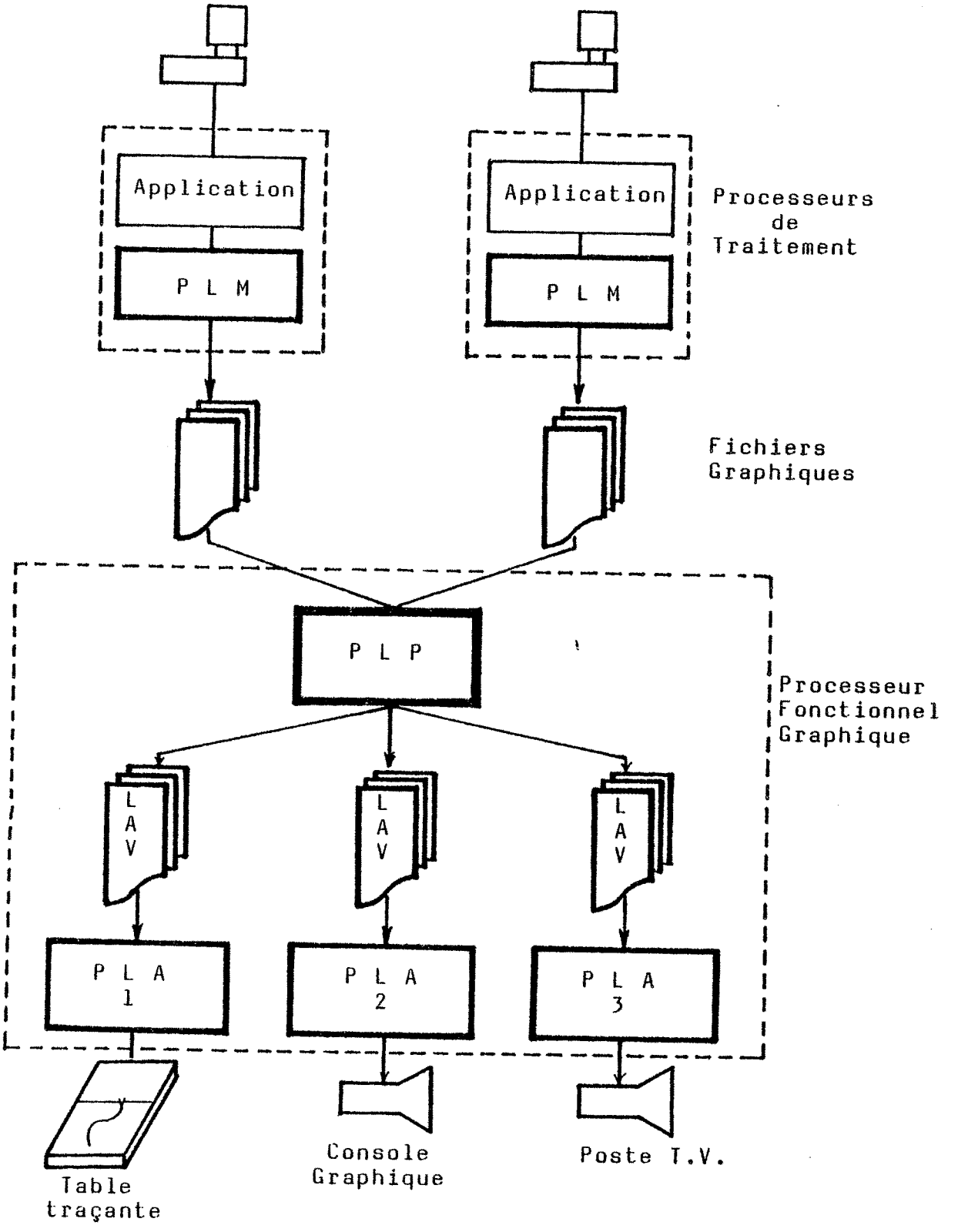


Fig. 1.6 - Le Système Graphique Interactif dans le Système CORAIL.

8.- Synthèse.

Le découpage fonctionnel met en evidence le caractère universel de la mise en page de la Préparation à la visualisation. Ceci nous permet de n'avoir qu'un seul Processeur Logique de Préparation PLP dans le SGI de CORAIL.

La modularité du SGI permet entre autres:

-d'offrir des langages évolués et des logiciels de modélisation bien adaptés à chaque classe d'application,

-des possibilités d'adaptation aux différents contextes d'utilisation, par exemple:

. Si dans une installation donnée, plusieurs Processeurs de Traitement ont besoin de la même fonction de modélisation, il suffit de l'implémenter dans chacun de ces Processeurs.

. Si dans un même Processeur de Traitement, l'utilisateur a besoin de plusieurs classes de modélisation, il suffit de les implémenter dans ce Processeur ou éventuellement de créer un nouveau Processeur de Traitement.

Dans le SGI de CORAIL, la représentation des objets en trois dimensions, ainsi que la définition de sous-dessins, sont laisées au niveau du Processeur Logique de Modélisation.

Le Processeur Logique de Préparation est sensé connaître toutes les caractéristiques des consoles graphiques du SGI afin de bien leur adapter la liste d'affichage virtuelle qui sera interpretée par le Processeur Logique d'Affichage associé à cet organe d'affichage.

La notion de Processeur Logique d'Affichage, permet d'une

part de pouvoir exploiter au maximum les caractéristiques de terminaux performants et d'autre part dans le cas des terminaux moins performants, le PLA associé sera muni d'une bibliothèque d'algorithmes programmés afin de pallier les fonctions manquantes.

Le fait que GRIGRI soit le langage de commande du PLP, permet de voir le Processeur Fonctionnel Graphique comme une véritable machine GRIGRI.

CHAPITRE II

FORMALISME DE DESCRIPTION DE L'ARCHITECTURE DU SGI.

1.- Introduction.
2.- Notion d'objet.
3.- Notion de processus.
4.- Relation entre processus.
5.- Notion de processeur.
6.- Architecture du SGI.
7.- Mécanisme d'allocation.
8.- Mécanisme de synchronisation.
9.- Réalisation des liens fonctionnels dans le SGI.

II.- FORMALISME DE DESCRIPTION DE L'ARCHITECTURE DU SGI.

1.- Introduction.

Ici, nous voulons définir le sens des termes que nous allons utiliser plus tard dans la description de l'architecture du SGI.

Ceci est très important surtout pour clarifier et préciser la signification de certains termes, tel par exemple processus, qui en général sont trop chargés de sous-entendus dépendant de l'origine du lecteur.

Plusieurs de ces termes sont empruntés à diverses domaines comme les systèmes d'exploitation, architecture de systèmes(GUI-78), ... etc.

2.- Notion d'objet.

D'un point de vue statique, nous considérons qu'un système informatique est formé d'un ensemble de constituants que nous appelons objets. Ces objets sont des êtres inactifs, par exemple:

- Le code d'une procédure.
- Une zone de données.
- ... etc

3.- Notion de processus.

Comme le fait Anaceau (ANC-74), nous confondons la notion de processus avec celle de machine séquentielle.

Des exemples de processus sont: un automate, une machine de Turing, un ordinateur, ... etc.

Si l'on considère l'existence d'un ensemble infini de ressources, un processus se comporte comme une machine n'ayant que deux états d'éxecution possibles: il peut être dans l'état actif quand il s'éxecute ou dans l'état bloqué quand il est en l'attente d'événements qui doivent avoir lieu dans d'autres processus.

3.1 - Liens fonctionnels.

Nous appelons liens fonctionnels les liaisons entre les entrées et sorties des processus qui appartiennent au même niveau.

Un niveau est défini comme l'ensemble des processus distinguables les uns des autres vis-à-vis d'un processus du même niveau.

3.2 - Langage de communication entre processus.

Pour pouvoir dialoguer avec un processus il faut s'adresser à lui en utilisant un langage qu'il puisse comprendre. Ce langage est défini par le processus lui même et constitue le langage de commande sur les entrées du processus. Par contre le langage sur les sorties sera celui défini par les processus destinataires.

On peut dire que la vision du monde extérieur d'un processus

est donnée par la syntaxe et la sémantique des requêtes émises sur les sorties.

Les processus peuvent échanger de l'information de deux manières différentes:

- par des connexions d'entrée/sortie.
- par des zones partagées de leur contexte.

3.3 - Espace d'exécution d'un processus.

L'ensemble d'objets mis en activité à un instant donné par un processus forme ce que nous appelons l'espace d'exécution de ce processus.

4. - Relations entre processus.

4.1 - Notion d'interprétation.

Si à un instant donné, un ensemble de processus du niveau n-1 s'occupent de la réalisation d'un processus du niveau n, nous parlons d'une relation d'interprétation du processus du niveau n par des processus du niveau n-1.

4.2 - Notion d'allocation.

Dans les relations d'interprétation, un processus du niveau n-1 peut être utilisé pour la réalisation de deux ou plusieurs processus du niveau n. Il se pose le probleme de décider quelle relation d'interprétation devient effective à un instant donné.

C'est le problème de l'allocation: il doit exister un allocateur pour résoudre ce problème. Cet allocateur va manipuler des états d'allocation propres à chaque relation.

La structure de la relation d'interprétation peut être représentée par une structure hiérarchisée comportant des allocateurs associés à tous les noeuds ayant un demi-degré entrant supérieur à 1.

4.3 - Notion de synchronisation.

La synchronisation est la distribution de temps d'interprétation entre les processus d'un même niveau.

La synchronisation peut se faire de deux façons: soit directe lorsqu'un processus agit sur un autre processus, soit indirecte par le biais d'objets dits de synchronisation.

Les outils de synchronistion peuvent être vus comme des objets spécialisés pour réaliser la communication entre les processus d'un même niveau

5.- Notion de processeur.

Considérons les niveaux n et n-1 d'une hiérarchie d'interprétation, nous appelons le processeur à n-1 et le processus à n.

Sous cet aspect, la notion de processus et de processeur est relative à la coupure que l'on fait dans une hiérarchie d'interprétation. L'intêret d'une telle décomposition est de permettre à tout instant de remplacer une réalisation d'un processeur par une autre dans une technologie différente.

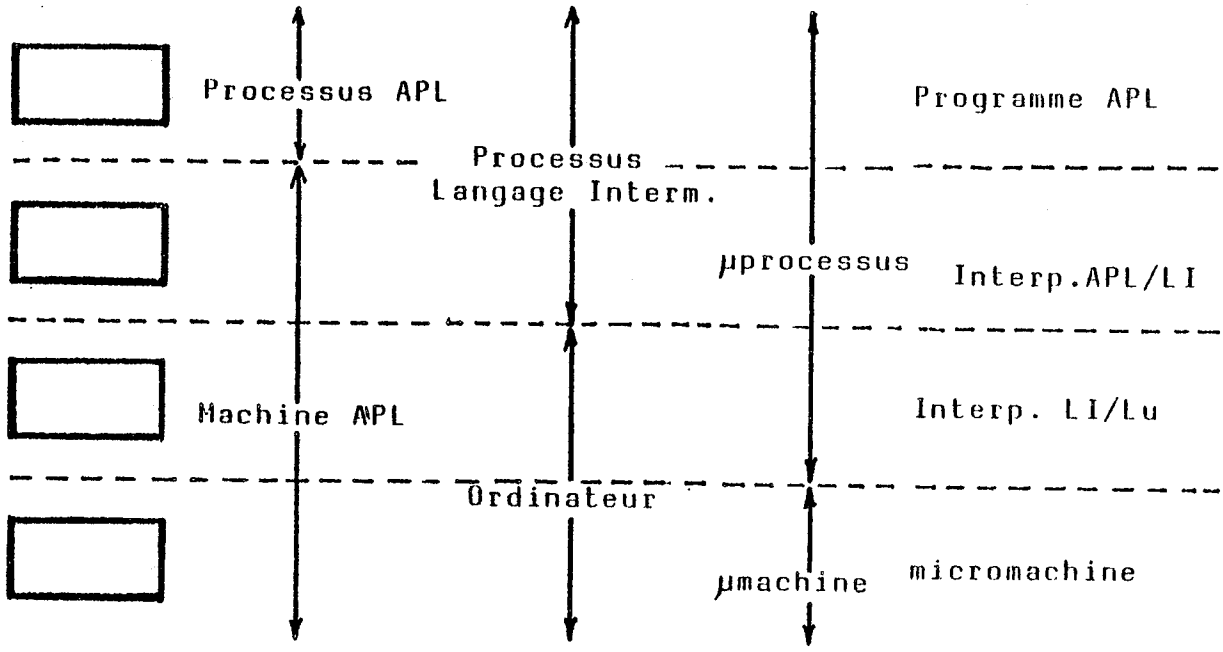


Fig. 2.1 - Notion de Processeur-Processus.

6.- Architecture du Système Graphique Interactif SGI .

Le découpage fonctionnel du SGI (cf. I.3) nous a permis de concevoir des processeurs logiques dont chacun est spécialisé dans l'exécution d'une fonction du SGI.

Dans la réalisation du SGI nous avons une architecture, à base de microprocesseurs, fonctionnellemente répartie où à chaque microprocesseur nous avons asigné, d'une façon statique, l'exécution d'une des fonctions du SGI.

Chaque microprocesseur a été équipé des ressources

matérielles nécessaires à l'exécution de la fonction assignée. Ces ressources matérielles sont:

- Des organes de mémorisation statique (ROM) et dynamique (RAM).
- Des organes spécialisés dans les entrées/sorties (PIA, ACIA).
- Des organes de calcul (Unité Arithmétique AM9511).

L'architecture générale du SGI est montrée dans les figures 2.2 et 2.3. Une description plus détaillée est faite dans IV.8 et V.6.

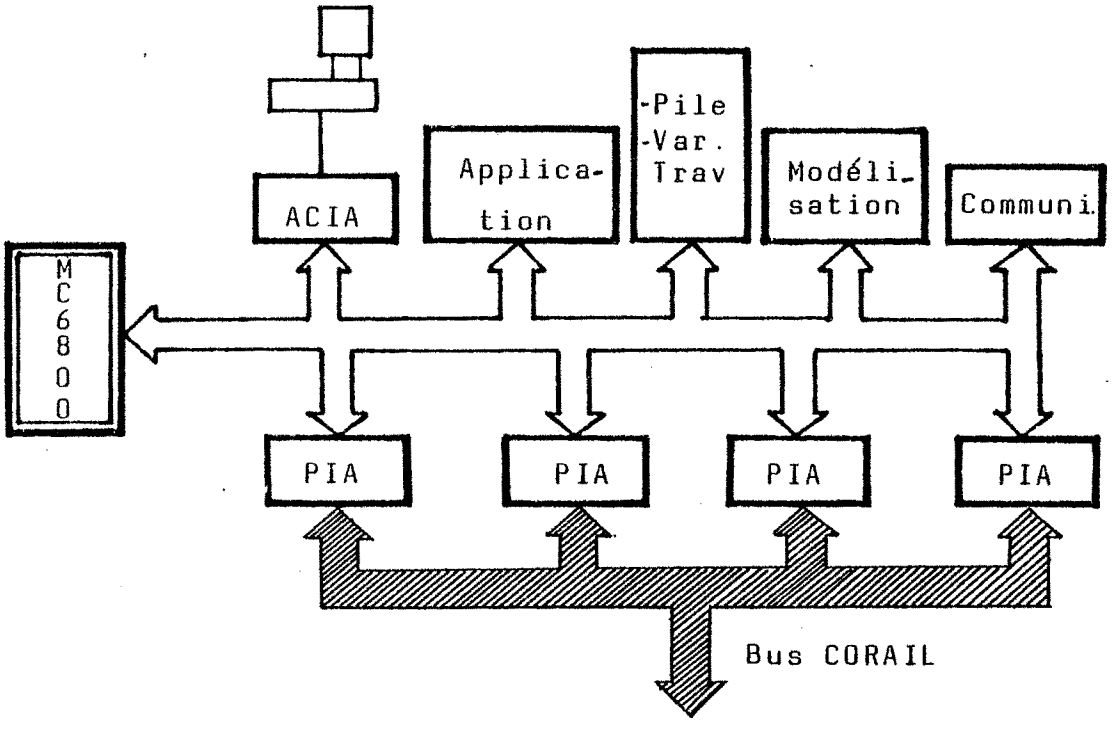


Fig. 2.2 - Architecture d'un Processeur de Traitement.

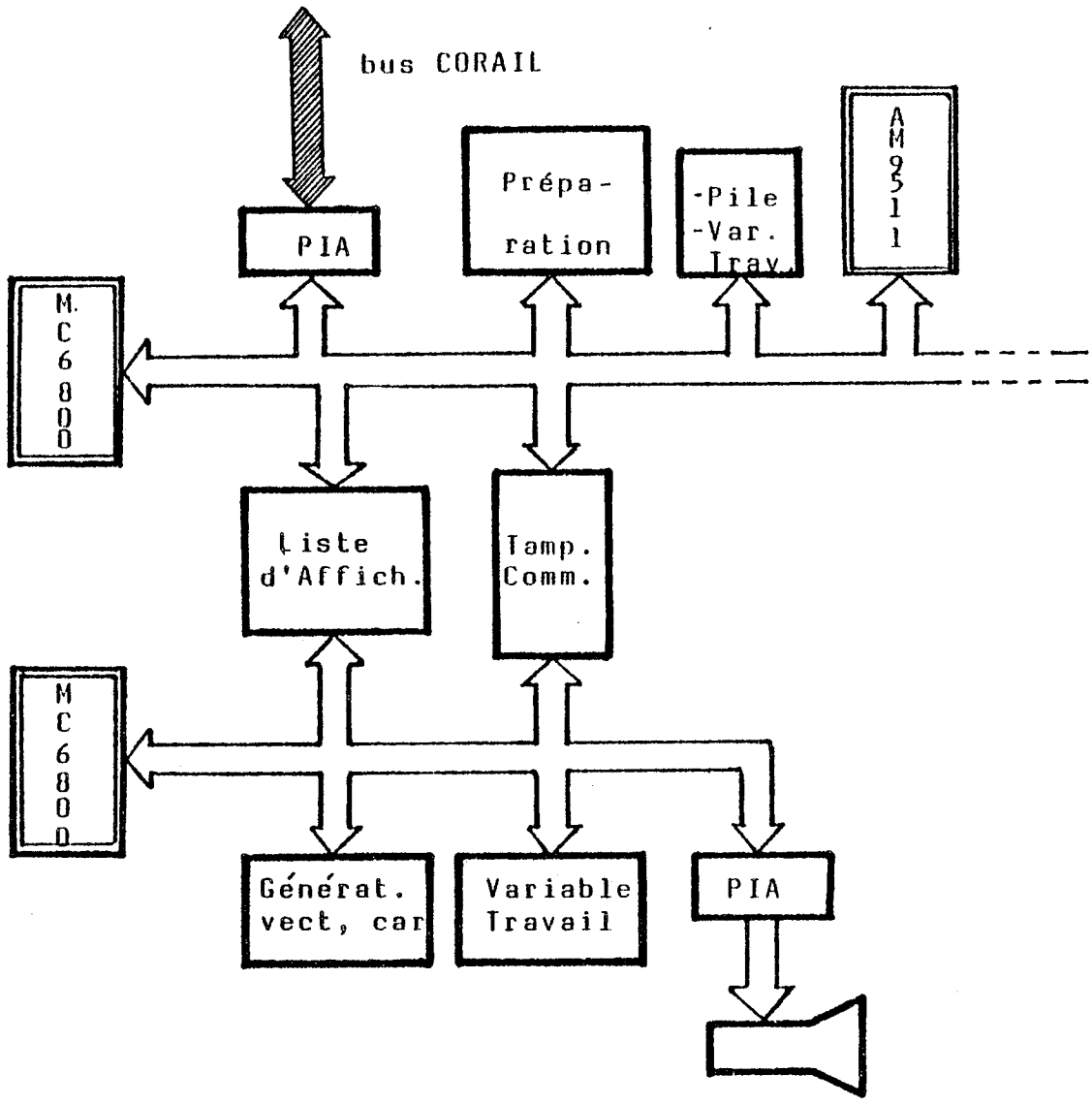


Fig. 2.3 - Architecture du Processeur Fonctionnel Graphique.

Dans cette architecture, la communication entre microprocesseurs est réalisée par des échanges de messages utilisant comme ressource de communication:

- Soit le bus CORAIL pour la communication entre le processeur qui exécute la fonction de Modélisation et celui qui exécute la Préparation à la visualisation.

- Soit par mémoire commune, pour la communication entre le processeur qui exécute la Préparation et ceux qui exécutent l'Affichage.

On constate que, dans le PFG, un bloc mémoire n'est partagé que par une paire de microprocesseurs, dont l'un est toujours celui qui exécute la Préparation.

Un bloc mémoire va contenir la liste d'affichage virtuelle ainsi que les tampons nécessaires à la communication entre le PLA et le PLP lors du dialogue.

L'ensemble de ces blocs mémoire font donc partie de l'éspace d'adressage du microprocesseur qui fait la Préparation.

7.- Mécanisme d'allocation.

Dans le SGI, chacun des processeurs logiques qui le composent a été concrétisé par un microprocessuer MC6800. L'interpréteur de cette machine permet la demande d'allocation du CPU de deux façons :

- Implicite ou matérielle lors du positionnement d'une des bascules RESET, NMI ou IRQ.

- Explicite ou logicielle en utilisant les instructions : RTI ou SWI.

Ces dernières instructions ont été utilisées pour implémenter les outils de synchronisation entre les processus créés dans un même microprocesseur (cf. Moniteur-allocateur CPU, IV.6.3.1).

8.- Mécanismes de synchronisation dans le SGI.

8.1 - Synchronisation logicielle ou inter-processus.

Dans chacun des processeurs du SGI, la synchronisation des processus créés dans le premier niveau d'interprétation, est faite par des primitives constituant des moniteurs. Ces primitives aggisent sur des files d'attente de pointeurs de contextes de processus, de telle façon qu'un moniteur permet de controler l'utilisation d'un ressource du SGI.

Pour garantir la cohérence du Système, il suffit d'exécuter les primitives en exclusion mutuelle dans le temps.

Le SGI comporte des ressources (microprocesseurs et unité arithmétique AM9511) dont le contrôle d'utilisation de chacune est fait par un moniteur (cf. IV.6.3.1). Chaque moniteur contient deux types de primitives :

- SIGNAL(file)_ Cette primitive permet l'allocation immédiate de la ressource au processus qui se trouve dans la tête de la file des processus prêts à utiliser la ressource.

- QUEUE(file)_ Cette primitive permet à un processus de demander l'allocation d'une ressource. Pour ce faire on va mettre le pointeur du contexte du processus dans la queue de la file d'attente des processus en attente de cette ressource.

L'interprétation de ces primitives correspond à la réalisation de la communication entre processus.

Dans le SGI lors de la création d'un processus, on va allouer une zone-pile, qui sera libérée à la fin du processus. Chaque

zone-pile a un descripteur pour indiquer l'état (libre/occupé) et l'adresse en mémoire.

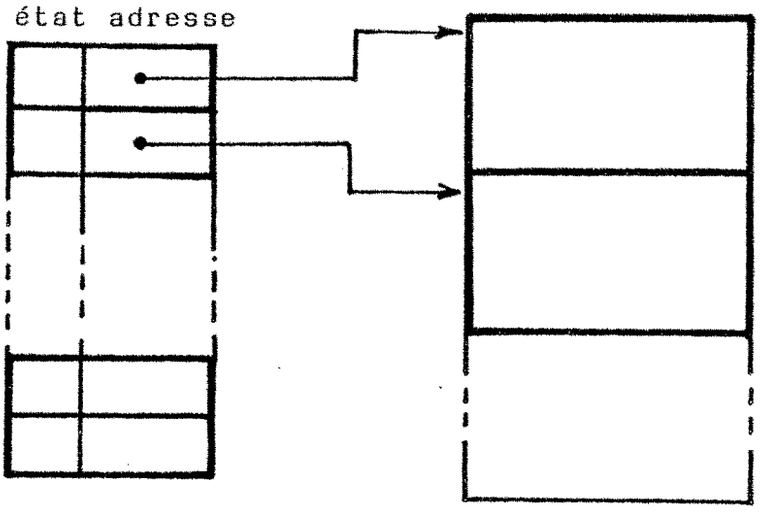


Fig. 2.4 - Les zone-piles.

Dans le PFG lorsqu'on a épuisé les zone-piles on va interdire la création de tout processus Pour une demande de service des Processeurs de Traitement. Pour ce faire il suffit de masquer les IT(IRQ).

Dans le SGI il faut aussi garantir la cohérence de l'information lorsque deux ou plus processus vont accèder à une région appelée critique, par exemple la lecture du pointeur (2 octets) de la primitive graphique en exécution par le PLP lorsque le PLA fait la mise à jour de ce pointeur. Ce problèmme a été résolu en utilisant de mécanismes logiciels de mutuelle exclusion décrits dans IV.6.1.

8.2 - Synchronisation matérielle ou inter-microprocesseurs.

Le microprocesseur MC6800 est rythmé par une horlogerie composée de deux phases non superposées Ø1 et Ø2.

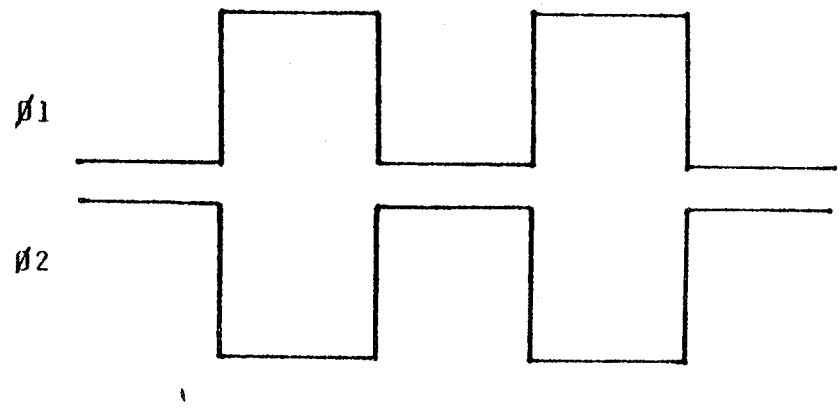


FIG. 2.5 - Les phases Ø1 et Ø2 du MC6800.

L'interpréteur de ce microprocesseur est conçu de telle façon que l'accès à la mémoire n'est fait que pendant Ø2. Cette caractéristique permet à un deuxième organe d'accéder à la même mémoire pendant la phase Ø1.

Le générateur des deux phases peut être vu comme l'allocateur de la mémoire qui assure l'exclusion mutuelle entre Ø1 et Ø2.

Dans l'architecture du SGI (cf. II.6) lorsqu'on va faire la communication par mémoire partagée, chacun des blocs mémoire n'est accédé, au plus, que par deux microprocesseurs.

Ces deux constatations nous permettent d'utiliser des horloges croisées pour synchroniser l'accès à un même bloc mémoire, partagé par deux microprocesseurs. La phase Ø1 de l'un devient la phase Ø2 de l'autre.

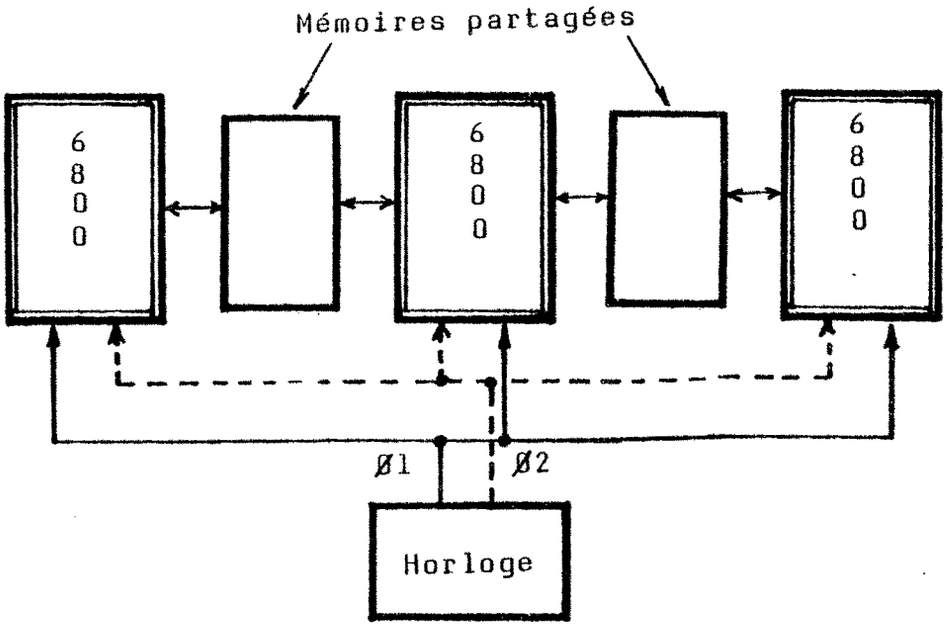


Fig. 2.6 Synchronisation des microprocesseurs du PFG.

9.- Réalisation des liens fonctionnels.

Le dialogue entre objets appartenant au même niveau est réalisé par des liens fonctionnels (cf. II.3.1).

Une condition nécessaire pour la réalisation du lien fonctionnel entre deux objets est l'existence d'une ressource de communication commune à ces deux objets.

Le caractère de la ressource d'être commune entraîne la nécessité d'avoir un allocateur de ressource. L'algorithme d'allocation doit être bien élaboré pour garantir la validité et l'efficacité du mécanisme de communication. Par exemple, l'algorithme doit refuser d'allouer la ressource au processus consommateur lorsque le processus producteur s'en sert.

Dans le SGI, les liens fonctionnels entre les processus du

premier niveau de la hiérarchie d'interprétation de chaque processeur, corespondent à un échange de messages. Ces messages étant de longueur variable vont rester dans un espace mémoire fixe et seuls les pointeurs des descripteurs de ces messages seront transmis et rangés dans des listes circulaires simulant des files d'attente. On constate que la ressource de communication commune à deux objets est concrétisée par une file d'attente.

Le descripteur d'un message contient l'information suivante:

- L'état du tampon (support physique du message) qui peut être: vide ou plein.

- L'adresse réelle en mémoire où se trouve le tampon.

Dans le SGI, une file d'attente n'est commune qu'à un couple de processus producteur-consommateur, l'algorithme d'allocation pour cette file est réparti entre les deux objets partenaires de la manière suivante:

- Le processus producteur de messages commence par demander un tampon vide pour y ranger le message, ensuite il va ranger le pointeur du descripteur de celui-ci à la fin de la file d'attente du processus conssommateur. Ensuite, s'il ne reste aucun tampon vide on interdit la création de tout processus producteur provoqué par une demande de service provenant de l'extérieur. Ceci est fait en masquant les IT(IRQ).

- Le processus consommateur commence par tester si sa file d'attente n'est pas vide, si c'est le cas alors on fait le traitement du message dont le descripteur est au début de la file. L'algorithme de gestion de la file est de type FIFO.

Pour tester l'état de la file il suffit de comparer les

pointeurs de tête et queue: si la tête rejoint la queue alors la file est vide, par contre si c'est la queue qui rejoint la tête alors la file est pleine.

CHAPITRE III

LE PROCESSEUR LOGIQUE DE MODELISATION (PLM)

1.- Introduction.
2.- Conception du PLM.
3.- Les processus du PLM.
4.- Niveaux de priorité et synchronisation des processus du PLM.

III.- LE PROCESSEUR LOGIQUE DE MODELISATION (PLM).

1.- Introduction.

Les domaines d'application des consoles graphiques sont aussi nombreux que différents, par exemple on trouve l'utilisation de consoles graphiques en:

- Physique nucléaire,
- Recherche médicale,
- Conduite et surveillance de processus industriels,
- Cartographie, création graphique, ... etc.

Ces origines différentes des utilisateurs d'un système graphique interactif, conduisent à ne pas pouvoir envisager une fonction de modélisation qui soit bien adaptée à toutes les applications. Par contre, on peut chercher à regrouper toutes les applications ayant des bases communes dans une même classe, afin de concevoir une fonction de modélisation qui soit bien adaptée à cette classe d'application.

C'est pour cette raison que dans le Sytème Graphique Interactif SGI de CORAIL, nous proposons d'avoir autant de PLM que de classes d'application.

Dans I.6.1, nous avons défini le PLM comme étant un processeur capable de transformer la représentation d'un algorithme écrit dans un langage évolué, dans une autre représentation codée à base de primitives GRIGRI.

Le rôle du PLM est donc d'extraire de la structure de données de l'application les éléments nécessaires pour coder un programme GRIGRI, appelé aussi fichier graphique.

Le programme GRIGRI sera découpé en messages CORAIL et envoyé au Processeur Logique de Préparation PLP (cf. IV) qui va le traiter pour définir un dessin virtuel. Celui-ci sera interprété par le Processeur Logique d'Affichage PLA (cf. V) afin d'obtenir le dessin sur l'écran.

Le PLM constitue donc l'interface entre l'application et la fonction de mise en page exécutée par le PLP.

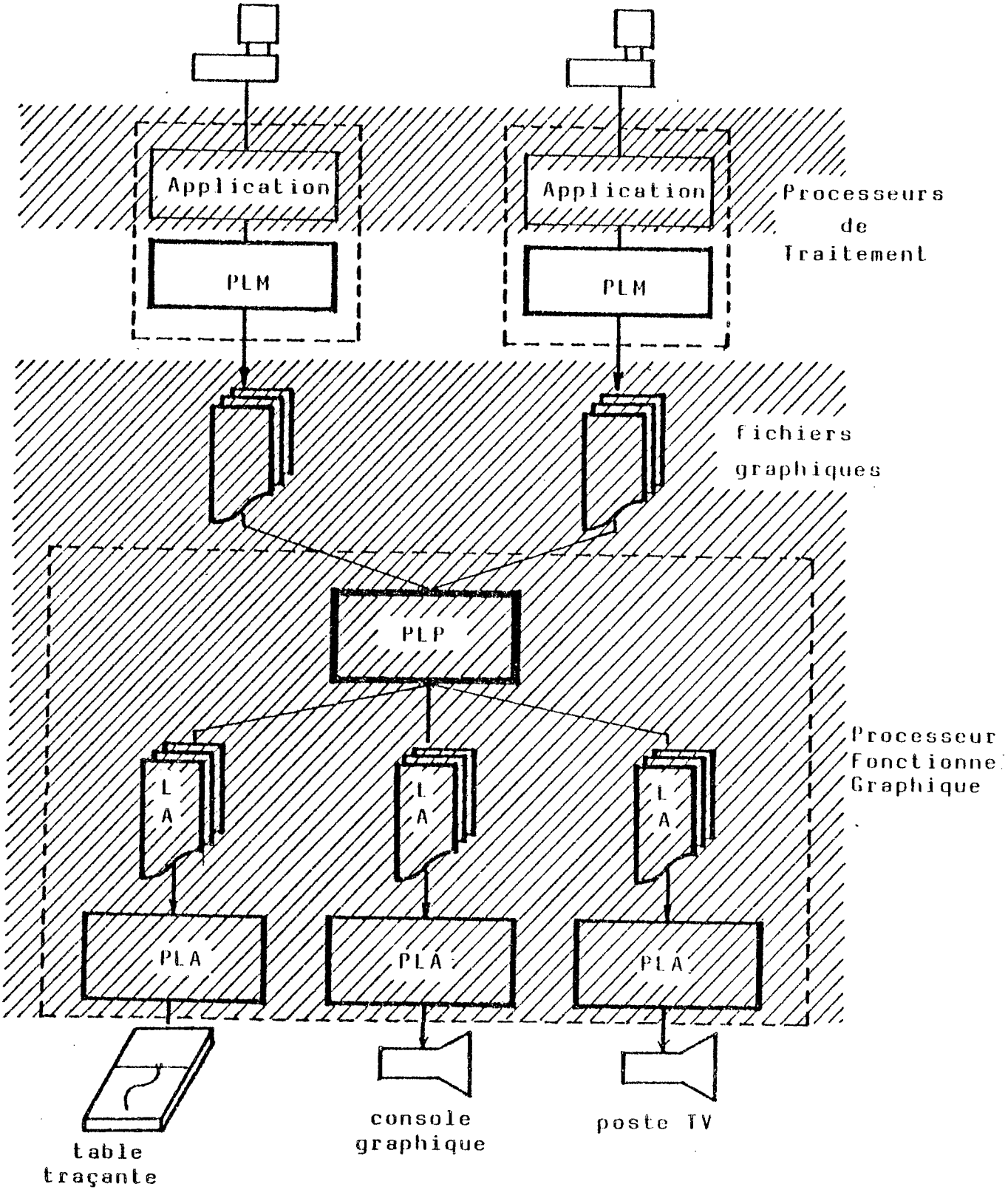


Fig. 3.1 Topologie du Processeur Logique de Modélisation.

2.- Conception du PLM.

La conception des PLM correspond à la phase de conception de sous-systèmes logiciels de la méthode de conception suivie, la méthode intermédiaire (cf. I.2)

Lorsque nous avons fait la répartition des fonctions du SGI (cf. I.5.1), nous avons décidé d'exécuter dans le Processeur de Traitement la fonction de Modélisation ainsi que la partie décodage de la préparation à la visualisation. Les mécanismes, qui permettront d'exécuter ces fonctions, seront donc implémentés dans le Processeur de Traitement et ils vont constituer le Processeur Logique de Modélisation (PLM).

2.1 - Le langage de commande du PLM.

La conception d'un PLM va commencer par la définition d'un langage de modélisation; ce langage sera considéré comme le langage de commande du PLM. Les critères qui vont guider le choix des primitives ainsi que leur nombre et niveau de paramétrisation, seront issus de la classe d'application.

Il reste ensuite à choisir le code que l'on va utiliser pour coder l'objet (scène) engendré pendant la phase de modélisation. Le choix de ce code, en général dépend lui aussi de l'application concernée. Ceci soit:

- parce qu'il est très simple d'extraire les éléments à coder des informations telles qu'elles sont traitées.
- parce que le code choisi se prête particulièrement bien à certains traitement ultérieurs.
- pour de simples raisons d'encombrement de la mémoire, par exemple, il est moins coûteux de conserver l'image d'un

cercle sous la forme de <coordonnées du centre,rayon>, plutôt que sous la forme de centaines de points.

2.1.1 - Les codes utilisés dans la modélisation.

Dans les systèmes de modélisation, plusieurs codes ont été utilisés pour engendrer les scènes. D'après M. Lucas (LUC-77), les codes les plus répandus à l'heure actuelle sont:

2.1.1.1. - Codes à base de directions élémentaires.

Ces codes réduisent la modélisation d'une scène à des chaines de directions élémentaires sur des distances plus ou moins longues. Un exemple de ce type de codes est le code de Freeman (FRE-74) qui permet de représenter un dessin au trait discrétisé, sur une surface à pointillage. Cette discrétisation conduit à une entité graphique par une liste de petits vecteurs élémentaires orientés suivant huit directions.

2.1.1.2. - Codes syntaxiques.

Ces codes font intervenir un vocabulaire de base (tracés élémentaires) ainsi qu'une grammaire pour définir des règles de composition. Lorsqu'on utilise ces codes, les scènes sont considérées comme étant des graphes de R^2. Les codes permettent non seulement de retrouver tous les composants du graphe (sommets et arêtes), mais aussi de conserver une trace de la structure de modélisation du graphe. Pour ce faire, on utilise une grammaire G, produisant un langage L(G). Les propriétés attendues de telles grammaires sont d'une part de permettre un codage économique du graphe, et d'autre part de donner des

moyens efficaces pour la manipulation, l'analyse et la génération de graphes. Une étude complète des modèles de grammaires décrivant une famille de graphes est faite dans (AZE-75).

2.1.1.3 - Codes géométriques.

Ici, le codage des scènes est fait à l'aide de primitives géométriques élémentaires et d'opérations de transformations géométriques telles que la rotation, la translation, le changement d'échelle, le changement de systèmes de coordonnées,...etc. L'idée de base est que toute scène à représenter sera décomposée en éléments géométriques dont on indiquera, pour chaque composant, le type, la position, la taille et divers autres attributs qui serviront plus tard à réaliser le tracé. On distingue en général les codes dans le plan et les codes dans l'espace.

En ce qui concerne le plan, les primitives généralement employées sont les points, les segments de droite, les coniques et arcs de coniques, les polynômes de degré quelconque. Les codes diffèrent essentiellement par le nombre, la paramétrisation et le niveau d'abstraction des primitives. Les opérations généralement autorisées concernent la définition de systèmes de coordonnées, la translation, la rotation et le changement d'échelle.

En ce qui concerne l'espace à trois dimensions, les codes diffèrent par le type des objets que l'on peut définir. Citons par exemple:

- Utilisation de segments de droites de l'espace, permettant de définir des graphes de R^3 par la donnée de sommets et d'arêtes.

- Utilisation de polyèdres, où le nombre de facettes est

d'autant plus élevé que l'objet à représenter est complexe. Un exemple de ces codes est EUCLID (THE-72), développé au LIMSI, qui permet de décrire des formes complexes avec un langage de commande.

- Définition de solides à l'aide de volumes élémentaires. Par exemple Braid (BRA-75) utilise six formes élémentaires qui sont combinées à l'aide d'opérations d'addition, soustraction et union pour définir de formes complexes.

- Utilisation de surfaces définies par des approximations polynomiales, telles que les b-splines. Ces surfaces offrent de plus grands degrés de liberté et se prêtent donc mieux à la conception de formes tourmentées. De plus, les paramètres définissant les formes sont choisis de telle manière que la conception puisse se faire de manière interactive.

Dans notre travail nous nous sommes limités à implémenter seulement trois primitives pour le langage de commande du PLM; ces primitives correspondent à la définition de cercles, polygones ouverts et textes.

2.2 - Le décodage.

Un fois définies les primitives du langage de commande du PLM, il reste à définir les mécanismes qui permettront de transformer la scène engendrée par la modélisation, dans une forme qui soit compréhensible par le Processeur Logique de

Préparation PLP. C'est-à-dire qu'il s'agit de transformer la scène sous la forme d'un programme écrit dans le langage de commande du PLP. Ceci sera réalisé par le processus de décodage décrit plus loin.

2.3 - Le dialogue.

Pendant la phase du dialogue, le PLM va être capable de prendre en charge les ordres d'interaction issus de l'utilisateur, afin de les retransmettre au Processeur Logique de Préparation PLP. Ces ordres permettront de valider les informations issues des dispositifs de dialogue. Le Processeur Logique d'Affichage PLA (cf. V) recueillera ces informations afin de les envoyer au PLP qui va les traiter (passage de coordonnées-écran à coordonnées-utilisateur) pour les transmettre ensuite au PLM qui va les identifier afin de les ranger dans la structure de données de l'application.

3.- Les processus du PLM.

Dans la réalisation du PLM, les mécanismes qui permettront d'interpréter le langage de commande, d'exécuter le décodage et de traiter les informations provenants du PLP pendant la phase de dialogue, ont été implémentés dans un ensemble de modules dont chacun à son tour est constitué par un ensemble de procédures .

Chacune des procédures sera interprétée par une suite d'instructions du MC6800 (qui à leur tour seront interprétées par les niveaux inférieurs du microprocesseur).

3.1 - Le processus de modélisation.

Le module de modélisation est constitué par un ensemble de procédures qui, lorqu'elles seront interprétées, vont définir le processus de modélisation. On définit ce processus comme une entité dynamique à durée de vie finie, qui va agir sur la structure de données de l'application afin de créer une autre structure de données appelée scène.

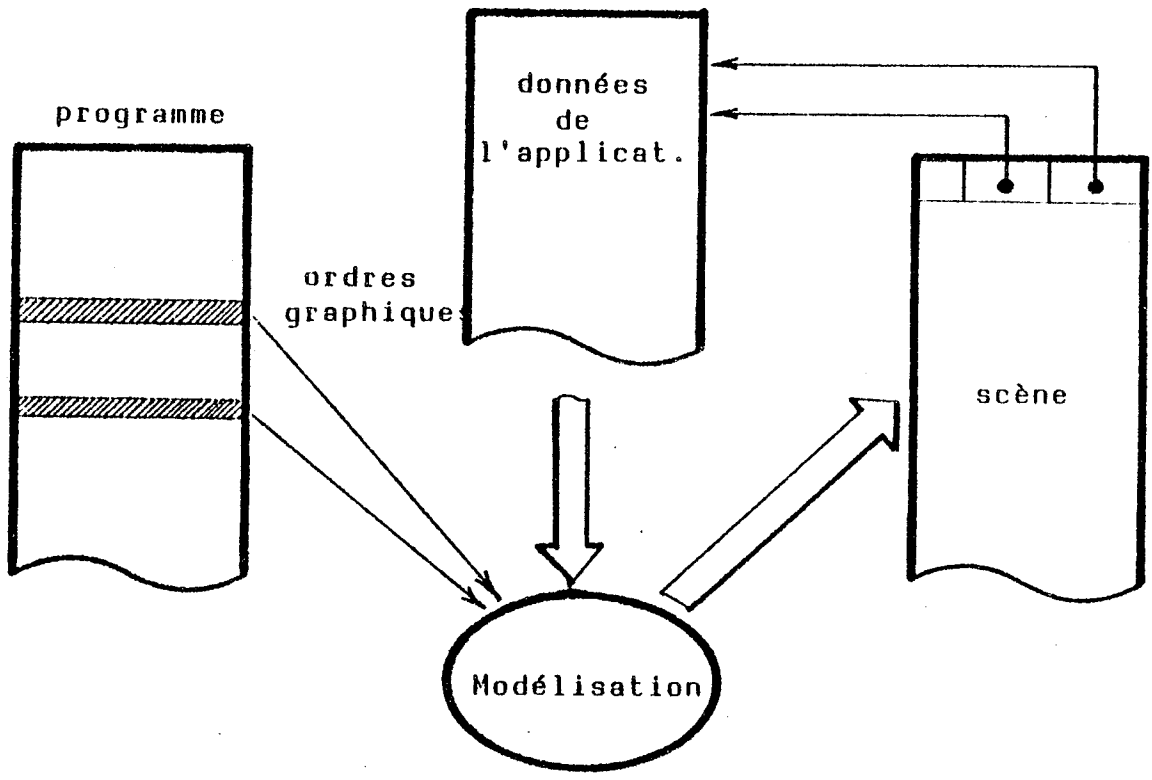


Fig. 3.2 Génération de la scène.

Le processus de modélisation est montré ensuite.

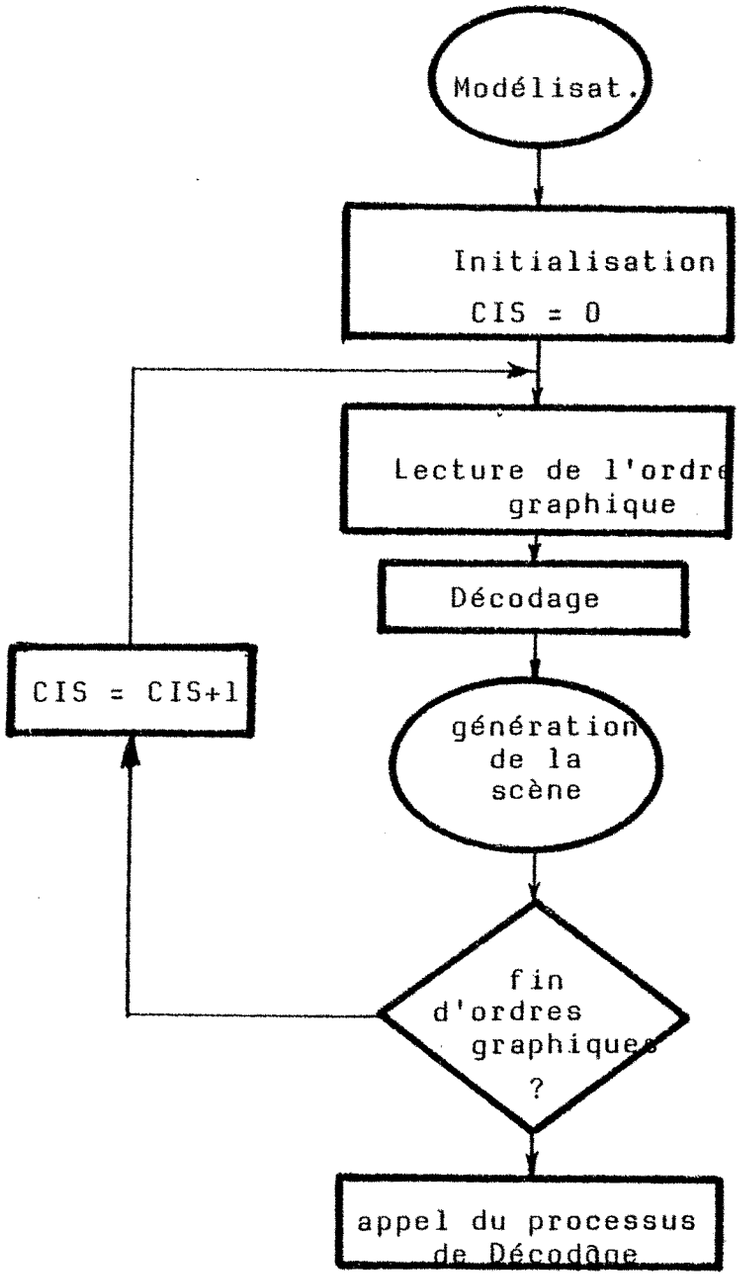


Fig. 3.3 Le processus de modélisation.

3.2. - Le processus de décodage.

Dans le module de décodage nous avons regroupé les procédures qui contiennent la programation de l'algorithme de décodage de la scène. La mise en activité de ces procédures définit le processus de décodage qui va agir sur la scène engendrée par la modèlisation afin d'engendrer une autre représentation de la scène, mais codée à base des primitives GRIGRI. Cette dernière représentation constitue le programme GRIGRI, appelé aussi fichier graphique.

Compte tenu de l'implémentation du SGI dans le cadre du Système CORAIL, le programme GRIGRI ne sera pas stocké dans le Processeur de Traitement, mais envoyé au Processeur Fonctionnel Graphique PFG, sous la forme de messages CORAIL.

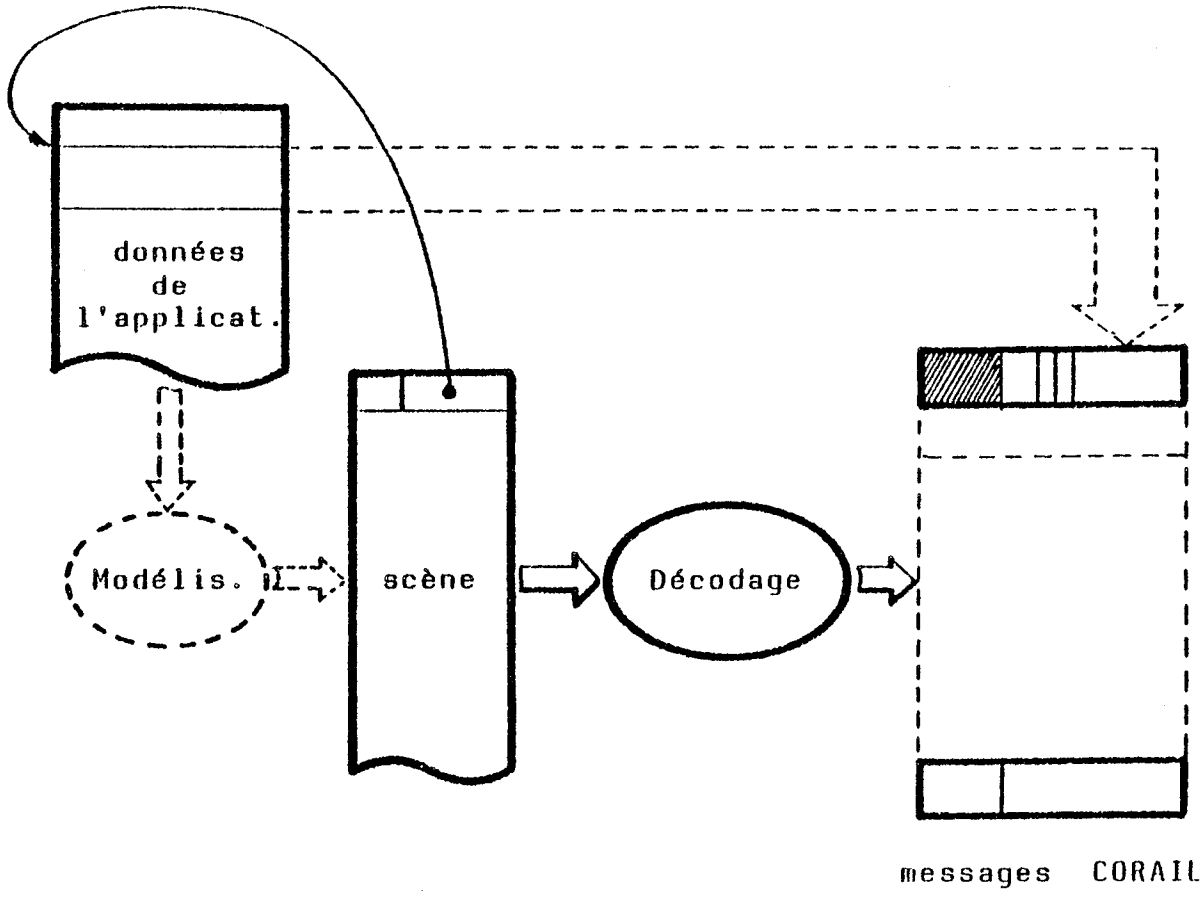


Fig. 3.4.- La génération du fichier graphique (programme GRIGRI).

Dans le corps de chacun des messages CORAIL, on va ranger le code de la primitive GRIGRI en question ainsi que ses paramètres et données. Ceci est montré dans la fig. 3.5. Une liste des codes des primitives GRIGRI est donnée dans IV.4.2.

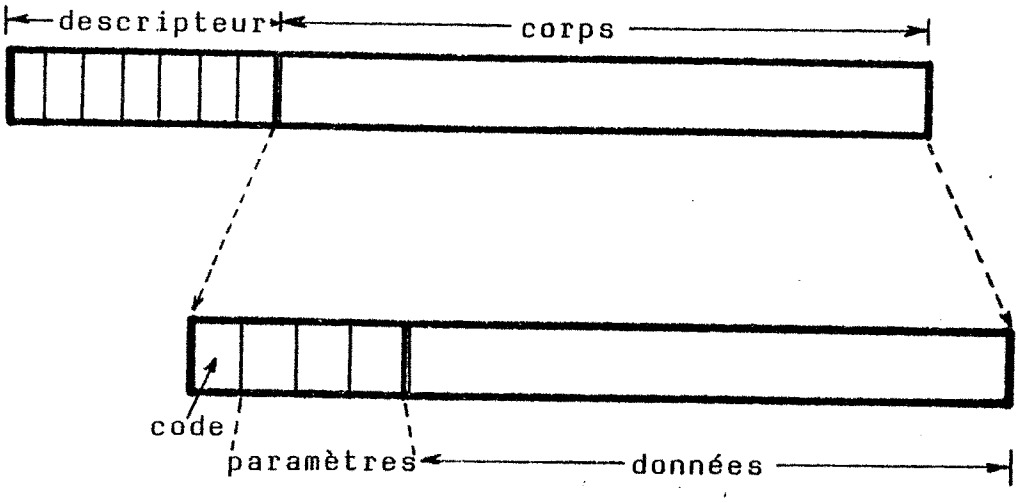


Fig. 3.5 Codage de la primitive GRIGRI dans le message CORAIL

Il faut signaler que, dans le Système CORAIL, chacun des Processeurs physiques, soit de Traitement, soit Fonctionnel, a un Superviseur de communication SUPCOM (Anexe 1) qui est composé de deux modules: l'un pour l'émission de messages de CORAIL et l'autre pour la réception de ces messages. Le SUPCOM est identique pour tous les Processeurs physiques de CORAIL et permet d d'établir et de contrôler la communication entre les Processeurs. La ressource commune de communication est le bus CORAIL qui est muni d'un allocateur.

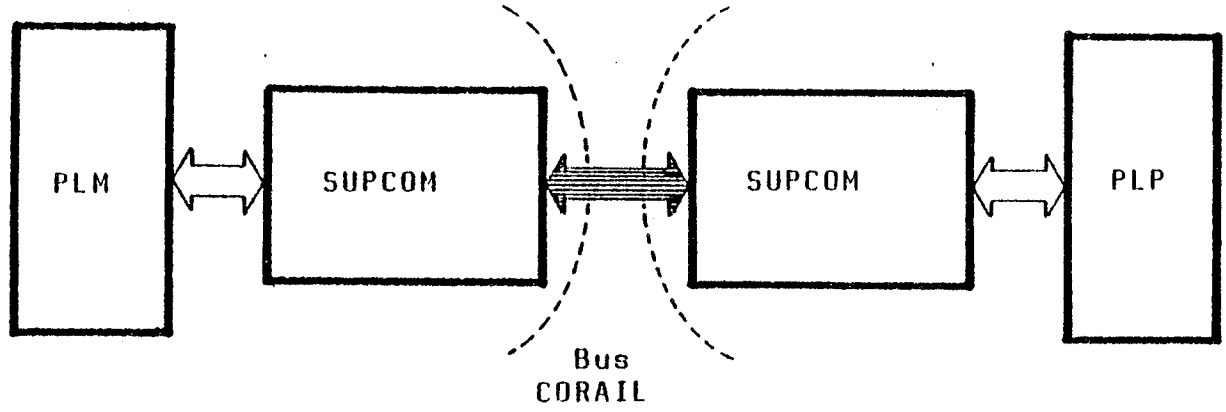


Fig. 3.6 Communication entre un PLM et le PLP du PFG.

Le dialogue entre les procesus de décodage et celui d'émission est réalisé par des liens fonctionnels réalisés au moyen d'une file d'attente de messages. Les messages CORAIL engendrés par le décodage, vont rester dans le même tampon et seuls les pointeurs des messages seront transmis et rangés dans la file d'attente FAMC de messages prêts à être consommer par le processus d'émission.

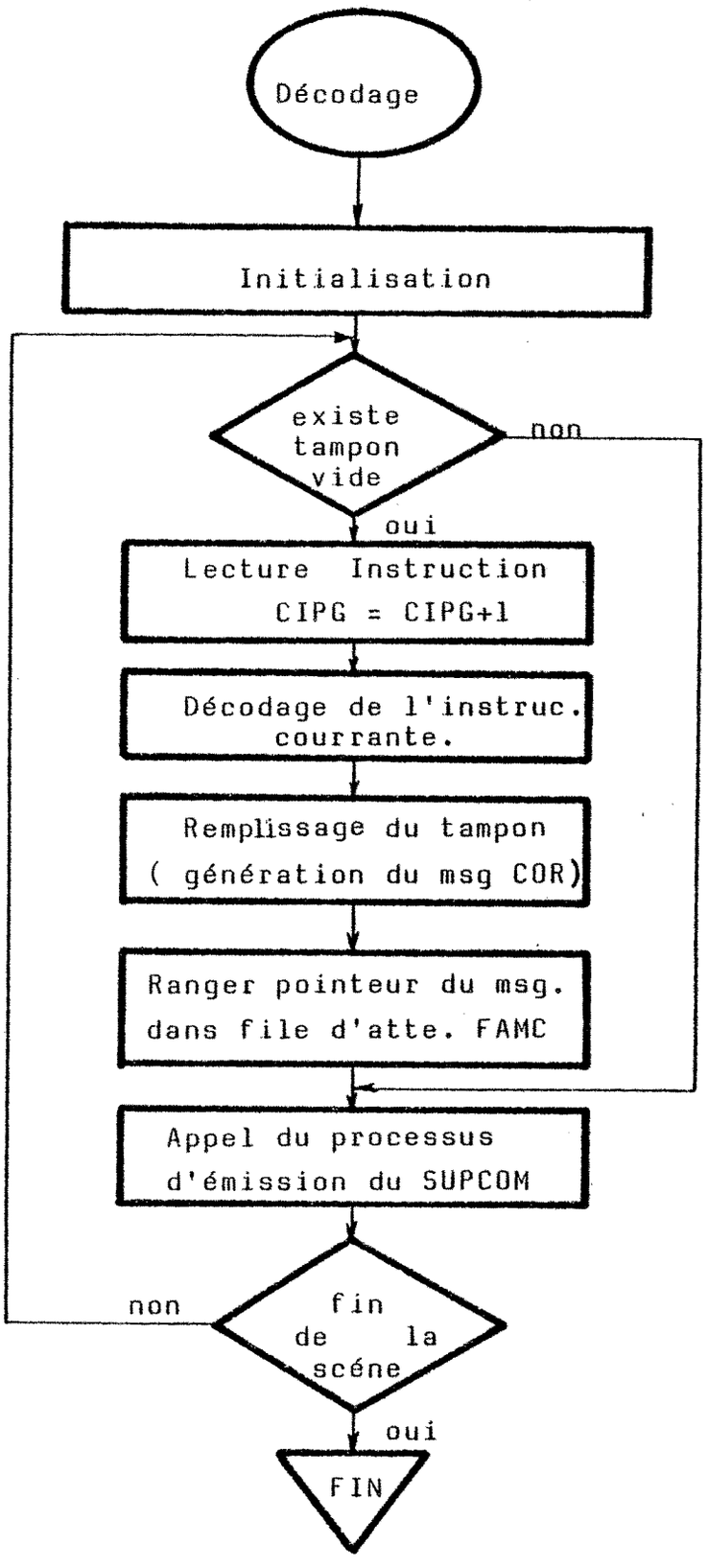


Fig. 3.7 Le processus de décodage du PLM.

3.3 - Le processus de ramassage.

Durant le dialogue, le PLP va traiter les informations issues des dispositifs de dialogue (cf. processus TRAIDIAL IV.4.6) et les envoyer au PLM concerné par le biais de leur Superviseur de communication, sous la forme de messages CORAIL.

La tâche du processus de ramassage est de traiter ces messages afin de ranger l'information envoyée par le PLP, dans la structure de données de l'application.

Les messages CORAIL constituent le lien fonctionnel entre les processus de réception et celui de ramassage.

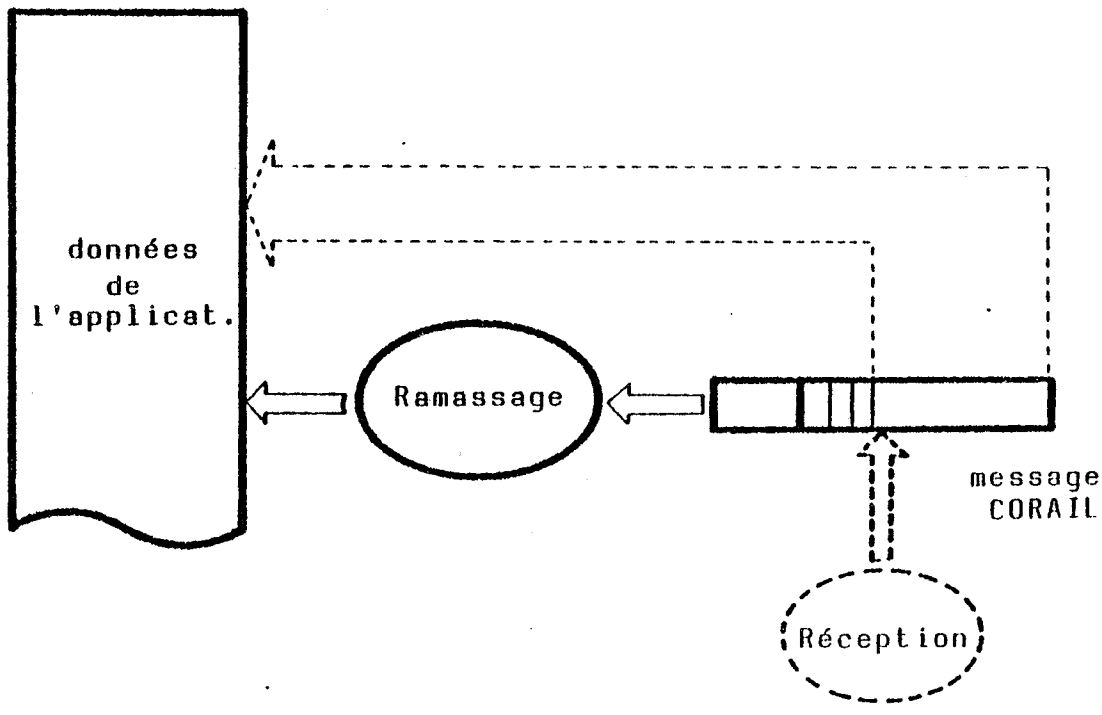


Fig. 3.8 - Ramassage de l'information pendant le dialogue.

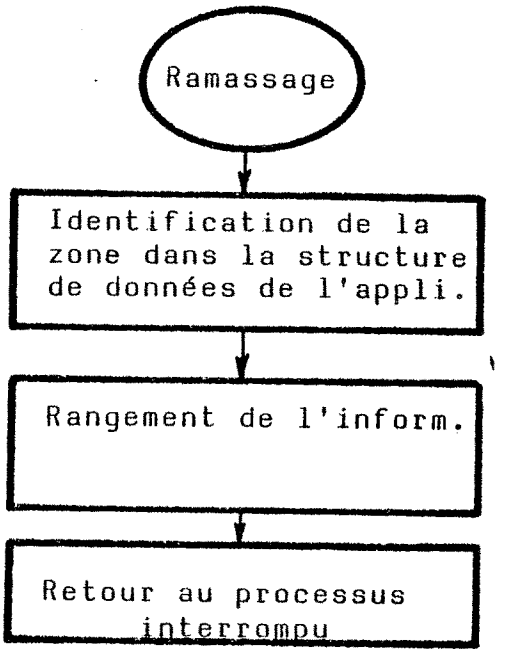


Fig. 3.9 - Le processus de ramassage.

4. - Niveaux de priorité dans les processus du PLM.

Dans le PLM, nous distinguons deux niveaux de priorité lors de la demande d'allocation du CPU.

Dans le niveau le plus prioritaire, nous avons les procesus définis par les parties d'émission et réception de messages CORAIL, ainsi que le processus de ramassage.

Cette priorité est donnée aux processus de façon statique, et correspond à un des objectifs du SGI, de traiter le plus tôt possible les demandes d'interaction et de répondre aussi vite que possible à ces demandes.

Dans le deuxième niveau nous avons les processus de modélisation et celui de décodage. Cette priorité a été aussi

attribuée de façon statique.

4.1 Synchronisation des processus du PLM.

Vis-à-vis de la demande d'allocation de CPU, les processus dont la priorité est un se comportent comme non-concurents. L'allocation du CPU pour les processus de réception et ramassage se fera dans un ordre déjà prédéfini: on va exécuter d'abord le processus de réception et ensuite celui de ramassage.

La création du processus d'émission, sera faite par le processus de décodage lorsqu'on aura des messages CORAIL prêts à transmettre.

Pendant l'exécution de l'un de ces processus, nous allons interdire la prise en compte de demandes de service provenant des autres Processeurs de CORAIL. Pour ce faire il suffit de masquer les IT(IRQ).

Les processus de priorité deux, sont aussi non-concurents et l'on va exécuter d'abord le processus de modélisation et ensuite celui de décodage. Nous considérons que le processus de décodage est un processus qui exécute un nombre fini de cycles, de telle sorte qu'à chaque cycle il va engendrer un message CORAIL; dans cette optique, à la fin de chaque message CORAIL (ou dans le cas d'épuisement de tampons), le processus de décodage va donner le contrôle au processus d'émission . A la fin de celui-ci, il va redonner le contrôle au processus de décodage.

Une représentation temporelle de l'exécution des processus du PLM est montrée dans la fig. 3.10.

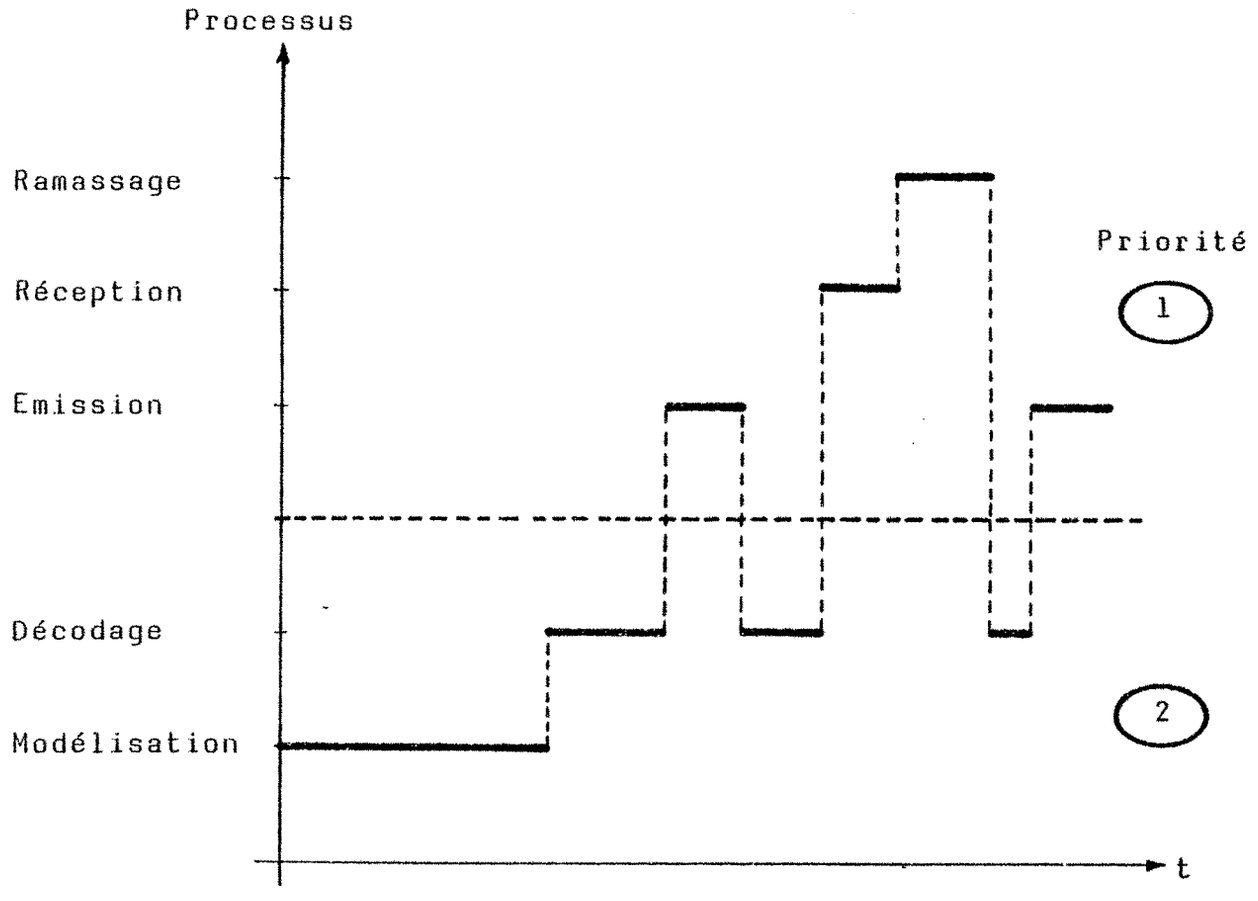


Fig. 3.10 - Représentation temporelle de l'exécution des processus du PLM.

4.2 Implémentation de la synchronisation.

Dans la partie II.7 nous avons dit que l'interpréteur câblé du microprocesseur MC6800 permet l'allocation du CPU, soit de manière implicite en positionnant les bascules NMI ou IRQ, soit d'une manière explicite à l'aide des instructions spéciales.

Dans le Système CORAIL, on utilise la ligne d'IRQ pour demander le service de l'un des Processeurs. Il suffit donc de

positionner la bascule IRQ du microprocesseur qui exécute la Modélisation pour lui demander un service.

La condition nécessaire à la création du processus de réception du SUPCOM est que la prise en compte d'IT (IRQ) soit autorisée (I=0). Ainsi, lorqu'il arrive une IT (IRQ), l'interpréteur câblé du microprocesseur exécutera la séquence qui permettra d'allouer le CPU au processus de réception. A la fin ce processus donnera le contrôle au processus interrompu.

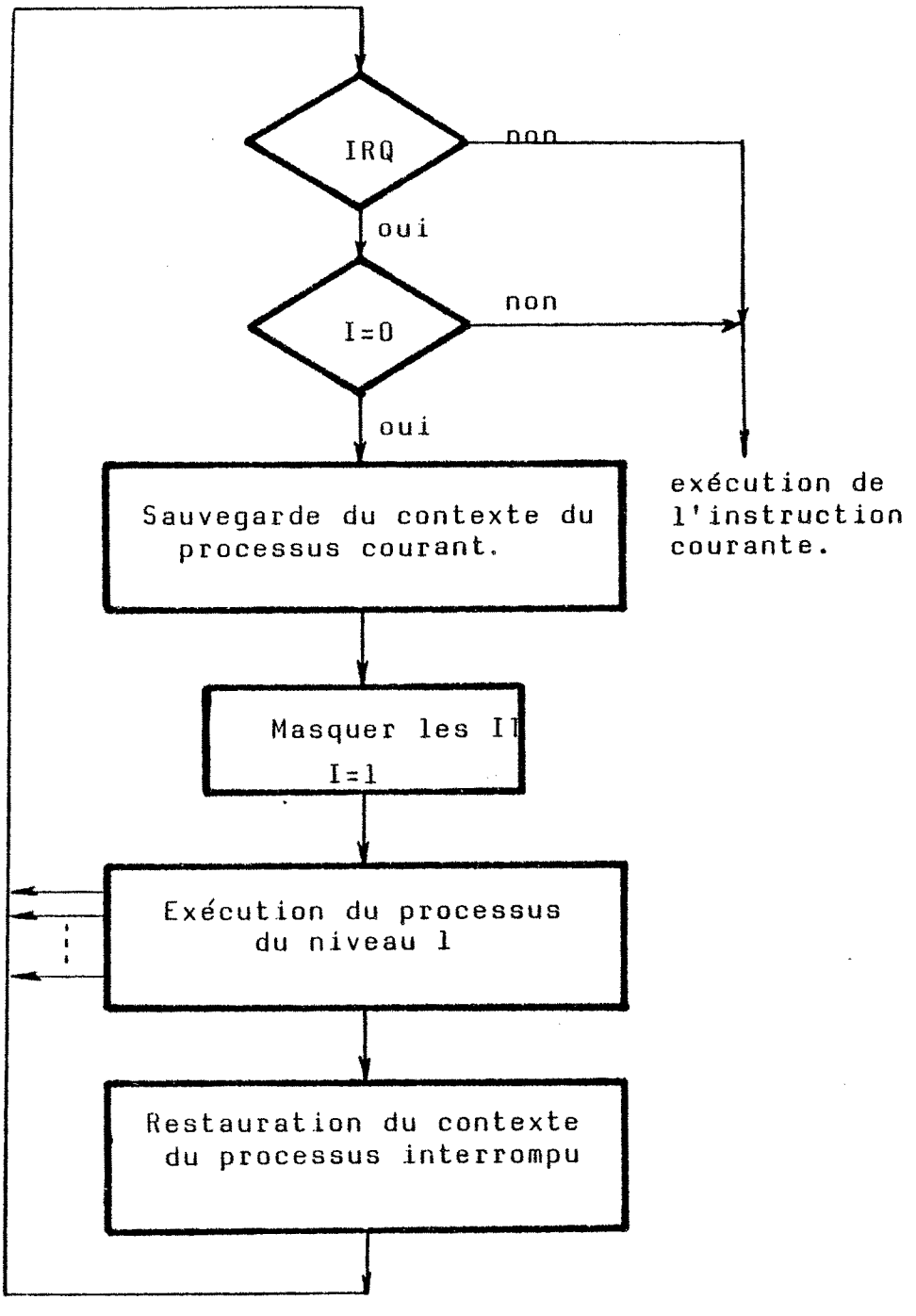


Fig. 3.11 Séquence lors de la prise en compte d'une IT (IRQ).

CHAPITRE IV

LE PROCESSEUR LOGIQUE DE PREPARATION (PLP)

1.- Introduction.
2.- Le langage de commande du PLP.
3.- Découpage fonctionnel du PLP.
4.- Les processus du PLP.
5.- Niveaux de priorité dans l'exécution des processus du PLP
6.- Synchronisation des processus du PLP.
7.- Le fichier GRIGRI.
8.- Architecture du PLP.
9.- Traitement d'erreurs.
10.- Evaluation de performances du PLP.

IV.- LE PROCESSEUR LOGIQUE DE PREPARATION PLP.

1.- Introduction.

Dans la conception du SGI, la répartition des fonctions entre processeurs logiques (cf. I.5.1) nous permet de n'avoir qu'un seul PLP et suggère d'imposer aux processeurs logiques de modélisation PLM, une façon unique de présenter les fichiers graphiques. Les caractéristiques du langage GRIGRI nous ont permis de l'utiliser pour coder ces fichiers graphiques.

Le PLP constitue l'interface entre l'ensemble des PLM et l'ensemble des processeurs logiques d'affichage (PLA). Topologiquement il représente la zone commune à toutes les applications et à tous les terminaux graphiques. Ceci est montré dans la figure 4.1

Pendant la phase d'affichage, les fichiers graphiques engendrés par les PLM, seront découpés en messages CORAIL et envoyés au PLP par le biais de leurs Superviseurs de communication (cf. Annexe 1). Dans le PLP on va interpréter ces messages (fichier graphique) afin de fournir des ensembles de dessins sur des surfaces de visualisation fictive (dessin virtuel).

Inversement, pendant la phase de dialogue, le PLP va fournir aux PLM les informations de nature graphique issues des dispositifs de dialogue (photostyle, tablette,...etc).

Le PLP sera muni de mécanismes d'identification pour qu'il puisse établir un lien entre les informations contenues dans la structure de données de l'application et les ordres graphiques envoyés par l'utilisateur. Comme fonction d'identification on va

utiliser des noms internes identiques à ceux donnés par l'utilisateur, ceci constituera le lien entre les éléments d'un dessin et les éléments de la structure de données de l'application.

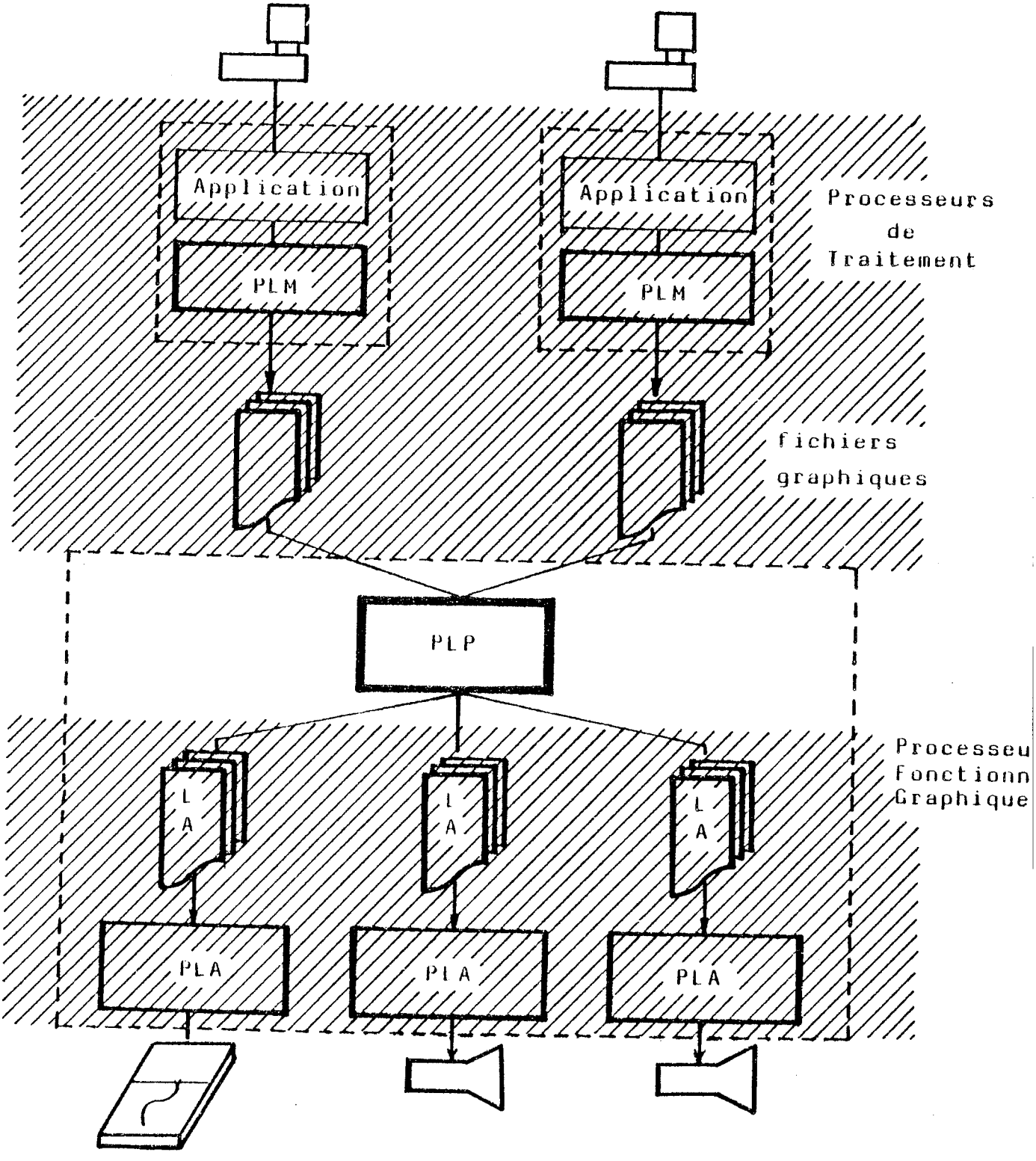


Fig. 4.1 Topologie du Processeur Logique de Préparation.

2.- Le langage de commande du PLP.

Les dialogues que nous avons eu avec l'équipe de Techniques Graphiques de l'IMAG nous ont permis de connaître leur logiciel de base GRIGRI. Ce logiciel a été largement expérimenté dans le Centre Inter-Universitaire de Calcul de Grenoble (CICG) et le résultat obtenu a été très intéressant. Ceci est dû à la qualité et à la simplicité qu'il offre, et surtout à son indépendance vis-à-vis du contexte d'utilisation.

Ceci nous a incité à étudier de près les primitives du logiciel GRIGRI et de les conserver comme base du langage de commande du PLP.

2.1 Principes de conception de GRIGRI (LED-77).

Les principes de conception de GRIGRI peuvent être résumés dans les termes suivants:

- Offrir des primitives indépendantes de l'application et du type de matériel à utiliser.

- Offrir des mécanismes de description statique du dessin en utilisant des structures de données élémentaires. Offrir aussi des règles pour l'interprétation graphique de ces données.

- Donner des règles simples pour manipuler les structures de données.

- Offrir une grande facilité pour la programmation du dialogue

- Donner des primitives de dialogue aussi puissantes que les primitives d'affichage.

Dans GRIGRI, la programmation du dialogue se fait en utilisant la notion de dispositif logique. On définit un dispositif logique comme étant un ensemble de mécanismes (logiciel + matériel) destiné à la simulation d'un dispositif réel de dialogue.

La notion de dispositif logique de dialogue permet de définir les outils de dialogue en termes fonctionnels, de telle sorte que l'on peut écarter délibérément la définition de primitives faisant appel explicitement à tel ou tel dispositif de dialogue.

2.2 Terminologie.

Dans ce paragraphe, nous voulons préciser le sens des termes que nous allons utiliser dans la suite.

2.2.1 Les espaces.

Fenêtre.

On définit une fenêtre comme étant une zone rectangulaire de l'espace utilisateur. Cette notion permet d'offrir à l'utilisateur les mêmes conditions de travail sur l'écran (espace limité) que dans son propre espace (espace illimité), la solution consiste à n'afficher qu'une partie du dessin initial sur l'écran. Tout se passe comme si l'utilisateur disposait d'une

fenêtre sur son espace de travail et que l'on affiche sur l'écran ce qui apparaît dans la fenêtre.

En approchant ou en reculant la fenêtre on provoque des effets de grossisement. En déplaçant la fenêtre on visualise différentes parties de l'espace de travail de l'utilisateur.

Clôture.

Une clôture est une portion rectangulaire de l'écran. Les bords d'une clôture sont parallèles aux bords de l'écran et définis par rapport au système de coordonnées de celui-ci.

La fenêtre définit donc un espace de départ, tandis que la clôture définit un espace d'arrivée.

2.2.2 Entités graphiques.

Les entités graphiques définissent les structures de données, et elles constituent une hiérarchie dont nous allons présenter les différents niveaux.

Entité élémentaire.

Une entité élémentaire est la plus petite entité graphique que l'on peut manipuler ou repérer. Celle-ci peut être:

- un point ou une suite de points.
- un segment ou une suite de segments.
- un caractère ou un texte.

Section.

On définit une section comme étant une entité logique régroupant un ensemble d'entités élémentaires.

Figure:

On définit une figure comme étant un ensemble de sections. Il faut préciser que les sections appartenant à une même figure sont définies par rapport au même système de coordonnées chez l'utilisateur.

Dessin:

Un dessin est composé par une ou plusieurs figures et c'est ce qui apparaît sur l'écran à un instant donné. D'une façon plus formelle on peut définir un dessin comme étant un objet structuré à plusieurs niveaux de telle sorte que les outils nécessaires à la définition d'un dessin forment une hiérarchie comme le montre la figure 4.2.

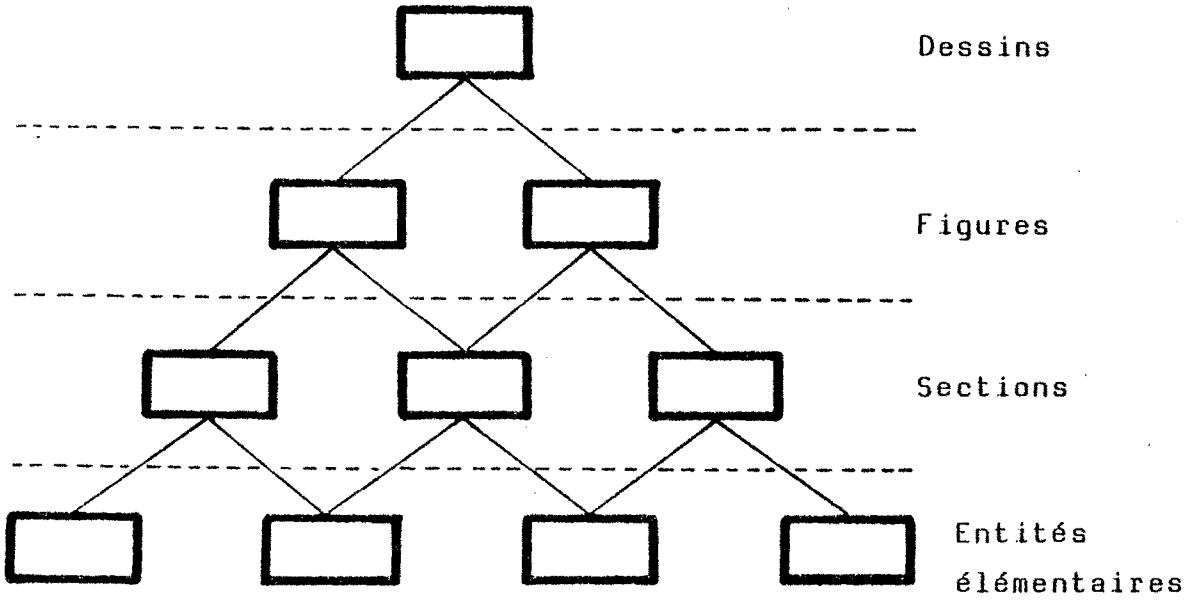


Fig. 4.2 Hiérarchie de composants dans un dessin.

Dans cette hiérarchie, le niveau figure permet, par l'intermédiaire d'un identificateur de regrouper plusieurs sections afin de créer une entité logique. L'évocation de cet identificateur permettra de retrouver toutes les sections attachées à la même figure. Ce niveau comporte les primitives du langage de commande du PLM (cf. III.2.1).

Le niveau section va comporter les primitives du langage de commande du PLP. Il ne dépend ni de l'application ni du terminal graphique. Il peut pallier certaines carences du matériel graphique, par exemple, l'absence de génération automatique de coniques.

Le dernier niveau, celui des entités élémentaires, dépend du type de terminal graphique, et il va comporter les primitives du langage de commande du Processeur Logique d'Affichage (cf. V.4.1).

2.2.3 Entités de contrôle.

Mode:

Le mode est l'ensemble de caractéristiques ou attributs graphiques associé aux éléments d'une section. Le mode permettra de définir d'une façon explicite les attributs tels que texture, couleur, taille, ... etc.

Marqueur:

Un marqueur sert à identifier une section. Le marqueur peut être:

- Un caractère alphanumérique,
- Une suite de caractères (4 maximun).

C'est à l'utilisateur de fournir les marqueurs. D'une manière interne, le PLP associe le marqueur au premier point de la structure de données définissant la section.

Message:

On définit un message comme une ligne de texte. Il contient des indications données à l'opérateur et des zones réservées à l'édition ou à l'acquisition de données alphanumériques. Les messages vont servir de support au dialogue interactif avec l'utilisateur.

Numéro de corrélation:

Le numéro de corrélation sert à regrouper plusieurs sections. Pour cela, il suffit d'associer le même numéro de corrélation (entier) à toutes les sections.

2.3 Les primitives GRIGRI.

Ici nous allons décrire brièvement les actions que permettent les primitives GRIGRI. Une présentation plus détaillée se trouve dans l'annexe 2.

Le logiciel GRIGRI, étant un logiciel de mise en page offre:

- La définition de dessins à l'aide de:
  . Primitives pour la définition des systèmes de coordonnées.

. Primitives de déclaration de données graphiques.

- La génération de dessins en utilisant:
  . Primitives d'interprétation graphiques des données declarées.
  . Primitives pour l'affichage et l'effacement.

- La programmation du dialogue interactif grâce à des:
  . Primitives pour envoyer des messages à l'utilisateur,
  . Primitives de dialogue qui permettent:
    + L'édition de données alphanumériques,
    + La collecte de coordonnées,
    + L'identification d'une suite d'éléments d'un dessin,
    + Le choix d'une suite de commandes à l'aide d'un menu.

3. Découpage fonctionnel du PLP.

Tenant compte de la description fonctionnelle faite jusqu'ici, le PLP aura comme tâche l'exécution des fonctions suivantes:

3.1 Génération de dessins.

Le PLP doit être capable d'interpréter les programmes GRIGRI (fichiers graphiques) fournis par les PLM afin de créer des dessins dans le plan (dessin virtuel) destinés aux PLA. Le code à utiliser pour coder un dessin virtuel sera fortement dépendant des caractéristiques du terminal graphique.

3.2 Gestion de dessins.

Le PLP va s'occuper de la gestion des objets qu'il va manipuler, donc il doit être capable de les identifier, afin de pouvoir, soit les restituer sur une demahde d'identification, soit retrouver un dessin ou une partie du dessin à partir d'un identificateur.

3.3 Traitement du dialogue.

Pendant la phase de dialogue, le PLP doit être capable de fournir aux PLM des informations de nature graphique issues des dispositifs de communication. Il fera une analyse syntaxique et sémantique de ces informations afin de les contrôler et de les valider.

Il faut noter que le PLP ne manipulera pas la structure de données de l'application, mais seulement des données graphiques sous la forme d'un programme GRIGRI ou fichier graphique, découpé en messages CORAIL.

Le PLP va donc supporter, par le biais des PLM toutes les applications envisageables.

4.- Les processus du PLP.

Le découpage fonctionnel fait dans III.3 nous permet d'isoler les fonctions que va réaliser le PLP et de prévoir un certain degré de parallélisme lors de leur exécution. Pour la réalisation de ces fonctions nous avons élaboré un ensemble de procédures dont la mise en activité constitue les procesus du PLP. Ces processus sont les suivants:

- COMPLP : communication avec les PLP,
- DEFDES : définiton de dessins,
- GENDES : génération de dessins,
- DEMDIAL : demande de dialogue,
- TRAIDIAL : traitement du dialogue.

Le fichier graphique créé par le PLM est découpé en messages CORAIL et envoyé au PLP par le biais de leurs Superviseurs de communication. Dans chaque message CORAIL, les octets contenant le numéro d'expéditeur, la longueur et le corps constitueront un nouveau message: le message MSGPLP

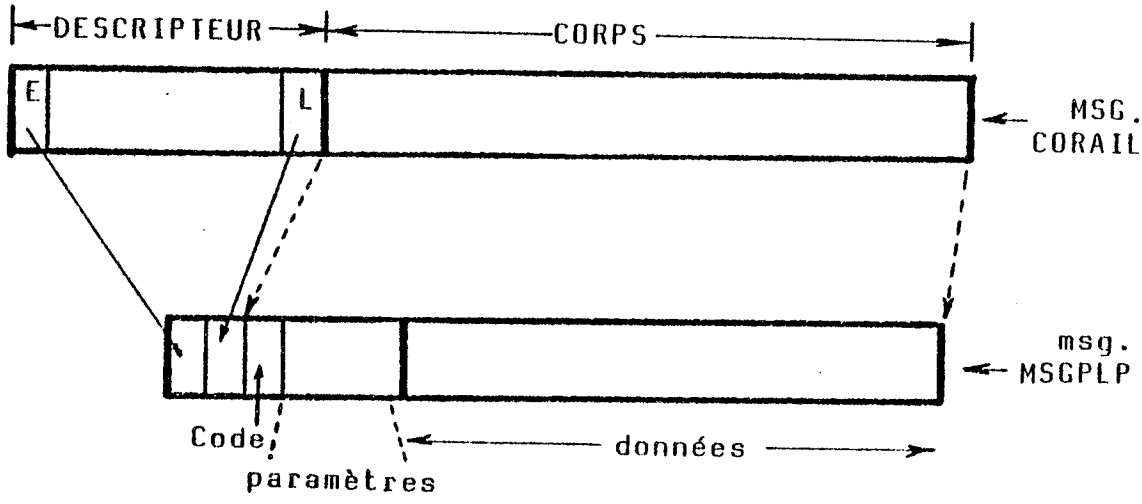


Fig. 4.3 - Le message PLP (MSGPLP).

Dans le descripteur du message PLP on trouve l'action qui sera appliquée au corps de ce message.

4.1 Communication entre les processus.

Le dialogue entre les processus est fait par des liens fonctionnels réalisés à base de files d'attente de messages. Chaque file se comporte comme une ressource commune au plus à deux processus. Elle a donc besoin d'un allocateur.

Les messages à échanger entre les processus sont de longueur variable, et ils vont rester dans un espace mémoire fixe, de telle sorte que seuls les pointeurs des descripteurs seront transmis et rangés dans des listes circulaires simulant des files d'attente.

Pendant la phase d'affichage le processus COMPLP se comporte comme un producteur de messages, tandis que DEFDES, GENDES et DEMDIAL sont des processus consommateurs. Pendant le dialogue, le producteur est TRAIDIAL tandis que le consommateur est le processus d'émission du Superviseur de communication CORAIL. L'échange de messages est schématisé dans la fig. 4.4

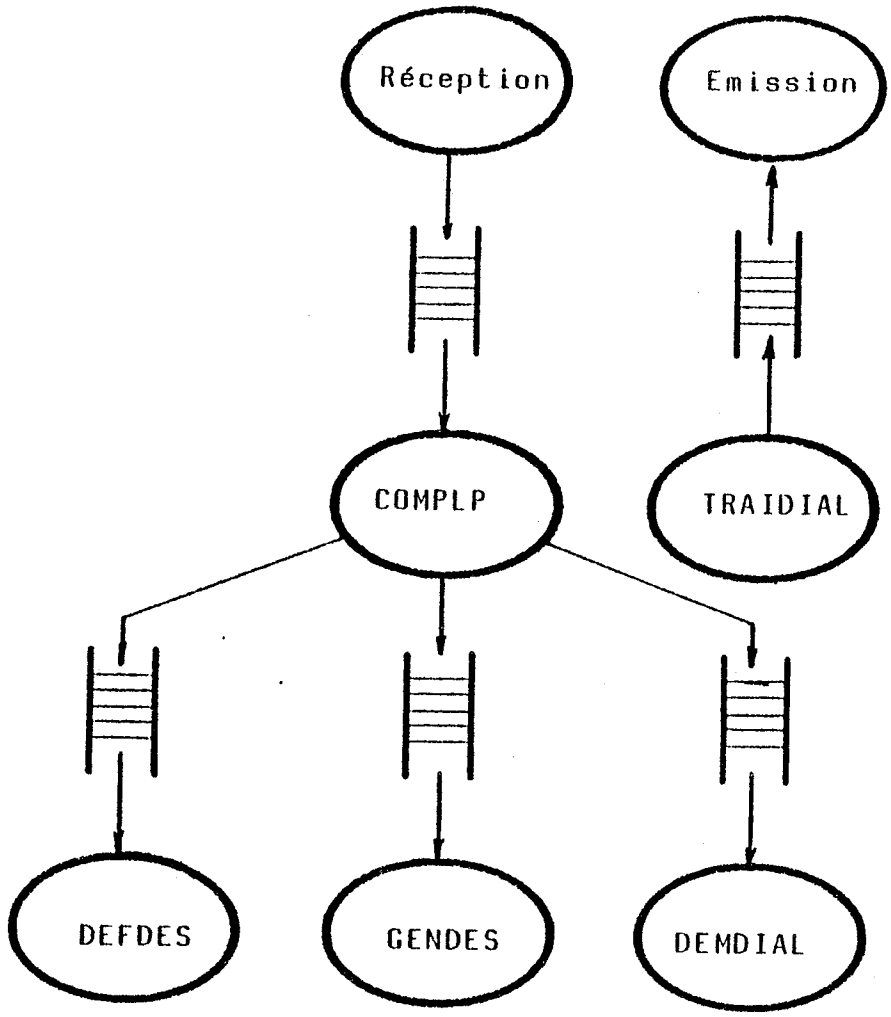


Fig. 4.4 - Echange de messages dans le PLP.

4.2 - Le processus de communication avec le PLP: COMPLP

La tâche de ce processus est de traiter le message MSGPLP afin de l'analyser et le transmettre au processus consommateur.

Le support de la communication entre ces processus, comporte trois files d'attente et un ensemble fini de tampons de telle sorte que vis-à-vis du processus COMPLP, ces files et ces tampons sont des objets qui appartiennent à son espace d'exécution. Par contre, pour un des processus consommateurs, à un instant donné, seuls une file d'attente et les tampons dont les pointeurs sur

leurs descripteurs sont rangés dans cette file, font partie de l'espace d'exécution de ce processus.

Le processus COMPLP va donc analyser le descripteur du message MSGPLP afin de ranger son pointeur :

- Soit dans FADD, file d'attente de messages prêts à consommer par le processus DEFDES, s'il s'agit d'une déclaration de données.

- Soit dans FAGD, file d'attente de messages prêts à consommer par le processus GENDES, s'il s'agit d'une génération de dessin.

- Soit dans FADL, file d'attente de messages prêts à consommer par le processus DEMDIAL, s'il s'agit d'une demande de dialogue.

Dans le descripteur de chaque message MSGPLP (cf. fig. 4.3),il existe un mot qui définit le code de la primitive GRIGRI. Ce code indique l'action à appliquer au corps du message MSGPLP. La liste de ces codes est donnée ci-après:

- Codes des messages à consommer par DEFDES:

  03 : Déclaration de données type points.
  04 : Déclaration de données type textes.
  05 : Déclaration du mode d'interprétation graphique des données.
  06 : Déclaration d'une fenêtre.
  07 : Déclaration d'une clôture.

- Codes des messages à consommer par GENDES:

  08 : Afficher (nfig, ncor, nfen, nclo).

09 : Effacer (nfig, ncor).

- Codes des messages à consommer par DEMDIAL:

0A : Déclaration de message à l'opérateur.
0B : Effacer message (nmsg).
0C : Edition de données type réel.
0D : Edition de données type entier.
0E : Edition d'une chaîne.
0f : Introduction de données type réel.
10 : Introduction de données type entier.
11 : Introduction d'une chaîne.
12 : Collecte de coordonnées.
13 : Identification d'une suite d'éléments d'un dessin.
14 : Demande d'affichage d'un menu.

A la fin de l'exécution du processus COMPLP, on va tester s'il y a encore des tampons vides, si ce n'est pas le cas alors on va interdire la prise en compte de toute demande de service du PFG: ceci est fait en masquant les IT (IRQ).

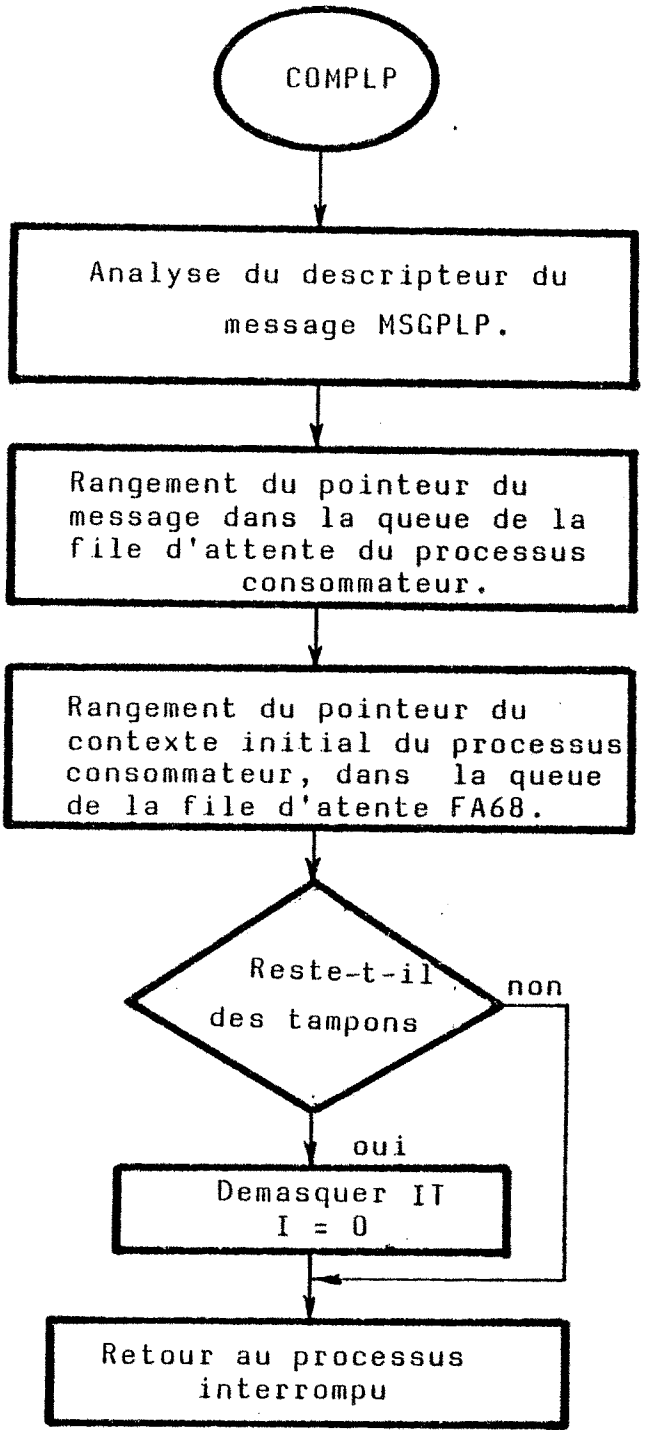


Fig. 4.5 - Le processus COMPLP.

Les différents états du processus COMPLP sont:

I: Inactif ou état initial.

A: Actif lorsqu'on alloue le CPU pour son exécution.
B: Bloqué lorsqu'on commute le CPU pour exécuter un processus plus prioritaire (cf. TRAIDIAL IV.4.6).

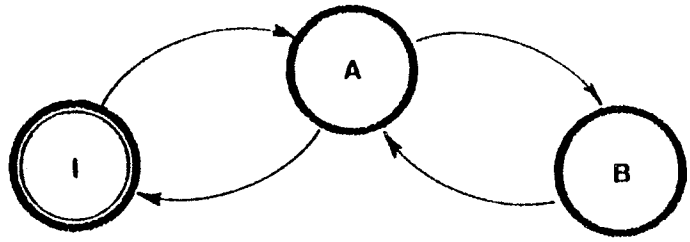


Fig. 4.6 - Graphe d'états du procesus COMPLP.

4.3 - Le processus de définition de dessin: DEFDES.

La création de ce processus est faite lorsque le processus COMPLP, ayant un message MSGPLP prêt à être traité par DEFDES, range le pointeur du contexte de départ de ce dernier dans la queue de la file FA68 (file d'attente de processus prêts en attente de CPU).

A chaque fois que ce processus passe à l'état actif, il traite le message MSGPLP qui est en tête de la file d'attente FADD afin d'engendrer une structure de données que nous appelons fichier GRIGRI (cf. IV.7). Ce fichier constitue la représentation structurée du dessin qui apparaît sur l'écran à un instant donné.

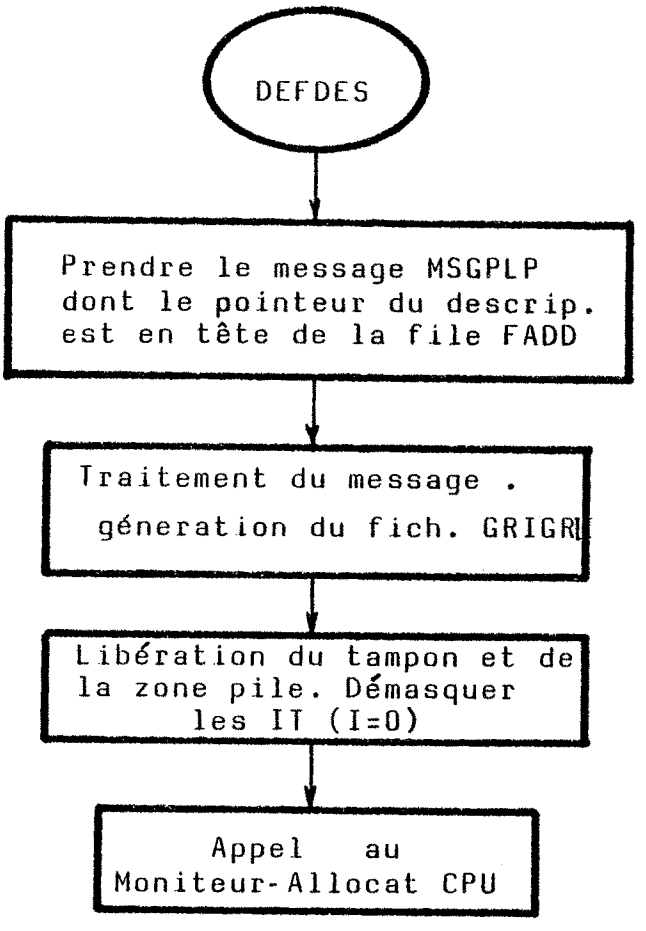


Fig. 4.7 - Le processus DEFDES

Les différents états du processus DEFDES sont:

I: Inactif
P: Prêt lorqu'il attend le CPU pour s'exécuter.
A: Actif lorqu'on lui alloue le CPU.
B: Bloqué lorsqu'on commute le CPU pour exécuter un processus plus prioritaire.

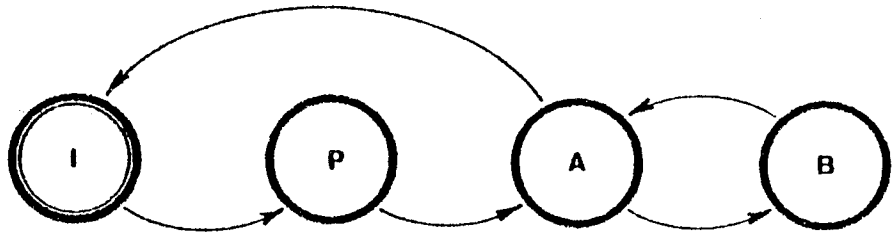


Fig. 4.8 - Graphe d'états du processus DEFDES.

4.4 - Le processus GENDES.

Lorsque ce processus est actif, il analyse le message MSGPLP dont le pointeur est dans la tête de la file d'attente FAGD, et selon le type d'action demandée, il va engendrer (cas d'affichage) ou modifier (cas d'effacement), à partir du fichier GRIGRI, une structure de données que nous appelons dessin virtuel ou liste d'affichage virtuelle.

Pendant l'exécution de ce processus, on peut demander le service de l'unité arithmétique AMD9511. Dans ce cas là, le processus GENDES, après avoir lancé l'exécution du AMD9511, va se mettre en attente active, c'est-à-dire boucler en testant l'état du AMD9511; dès que celui-ci a fini (Busy=0), GENDES sort de la boucle afin de lire les résultats qui sont dans la pile du AMD9511, après quoi il continue son exécution.

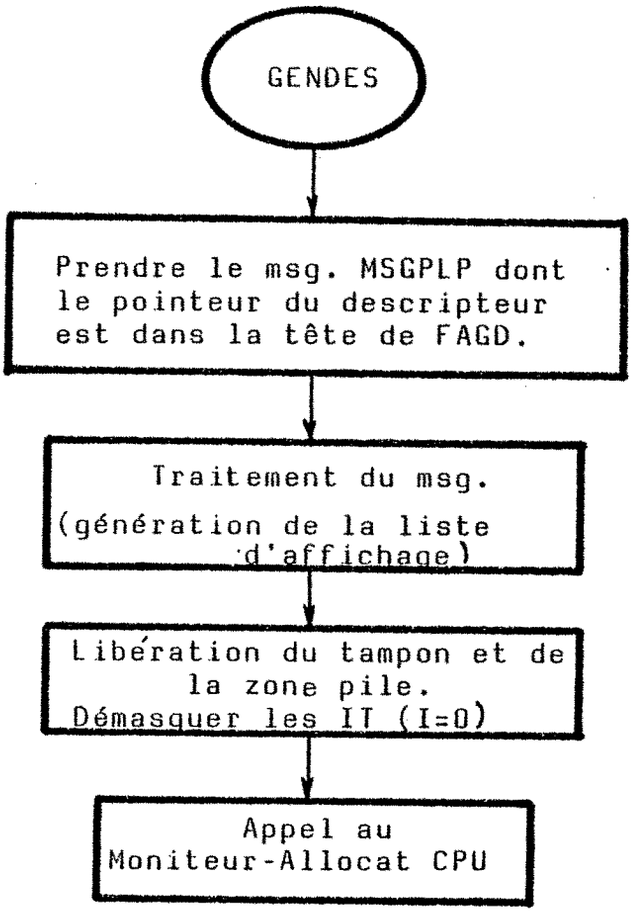


Fig. 4.9 - Le processus GENDES.

Les différents états du processus GENDES sont:

I: Inactif
P: Prêt s'il est en attente du CPU pour pouvoir s'exécuter.
A: Actif lorqu'on lui alloue le CPU.
B: Bloqué lorsqu'on commute le CPU pour exécuter un processus plus prioritaire (arrivée d'une IT).
Atte: En attente active lorqu'il attend la fin d'une opération du AM9511

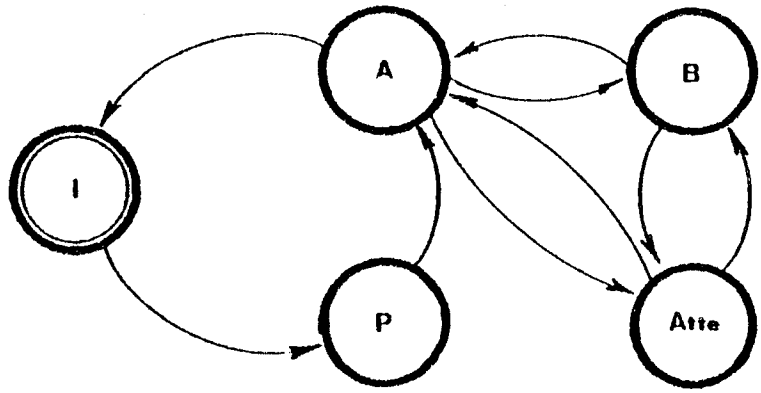


Fig. 4.10 - Graphe d'états du processus GENDES.

4.5 - Le processus de demande de dialogue: DEMDIAL.

Ce processus est créé par le processus COMPLP lorsque l'action demandée au PFG correspond à une demande de dialogue. La mise en activité de ce processus va permettre de:

- Contrôler la validité des informations reçues depuis le terminal de l'utilisateur lors des actions d'introduction de données alphanumériques. Ces données seront rangées dans le tableau de données (cf. IV.7) du fichier GRIGRI.

- Prévenir le processus TRAIDIAL du type de données à manipuler et du type d'action à réaliser lors de la mise en service des dispositifs de dialogue.

- Communiquer au processeur logique d'affichage PLA, le type et le nombre d'interactions demandées. Cette communication se

fera par le biais de tampons rangés dans la "zone de communication" de la mémoire partagée.

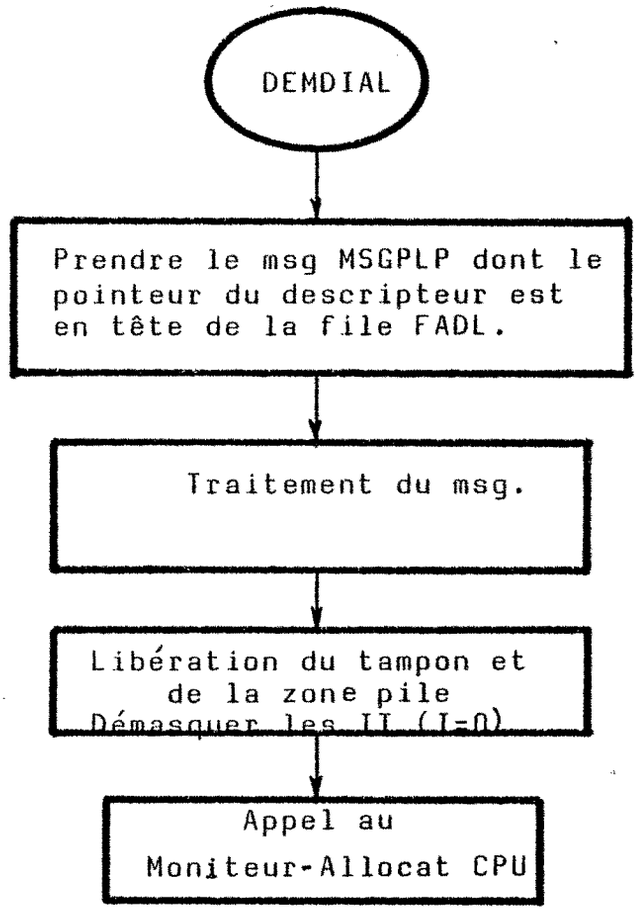


Fig. 4.11 - Le processus DEMDIAL.

Les différents états du processus DEMDIAL sont:

I: Inactif

P: Prêt lorsqu'il attend le CPU pour s'exécuter.

A: Actif lorsqu'il s'exécute.

B: Bloqué lorsqu'on commute le CPU pour exécuter un processus plus prioritaire.

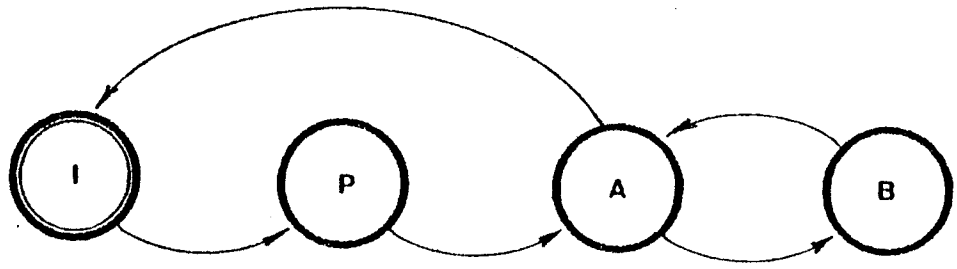


Fig. 4.12 Graphe d'états du processus DEMDIAL.

4.6 - Le processus du traitement du dialogue TRAIDIAL.

Pendant la phase de dialogue, le PLA va recueillir les informations issues des dispositifs de dialogue et les ranger dans les tampons rangés dans la zone de communication avec le PLP. Après ceci, il va créer (par le biais de la ligne NMI) le processus TRAIDIAL qui, grâce aux informations fournies par le processus DEMDIAL, va analyser d'un point de vue syntaxique et sémantique les informations fournies par le PLA afin de les ranger dans le fichier GRIGRI ainsi que les transformer (en utilisant l'unité arithmétique AMD9511) en les coordonnées de l'utilisateur pour les envoyer à l'application via le PLM concerné.

Vis-à-vis du processus d'émission du SUPCOM, le processus TRAIDIAL se comporte comme un producteur de messages. La ressource de communication commune à ces deux processus est la

file d'attente FAME qui contient les pointeurs des descripteurs des messages à envoyer aux PLM.

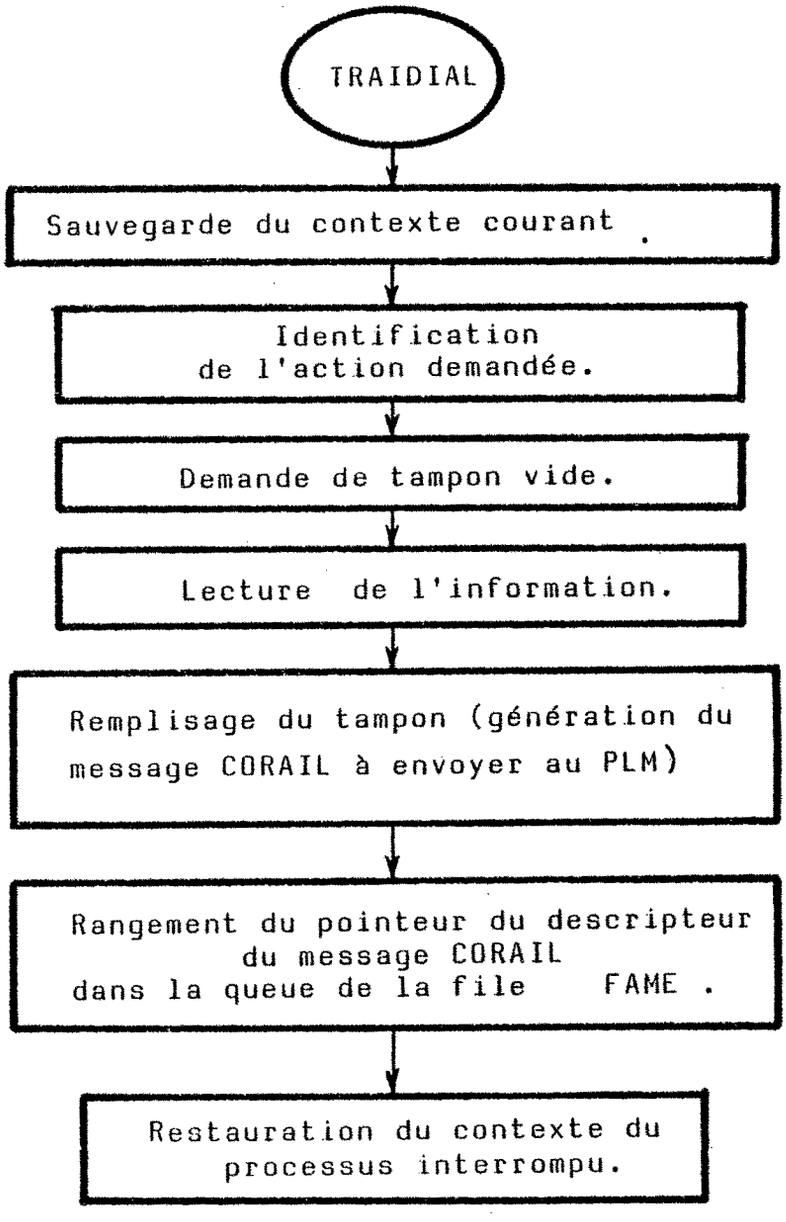


Fig. 4.12 Le processus TRAIDIAL.

Les différents états de ce processus sont:

I: Inactif

A: Actif dès que le PLA fait une demande de service par le biais de la ligne NMI.

Atte: En attente active lorqu'il attend la fin d'une opération du AMD9511.

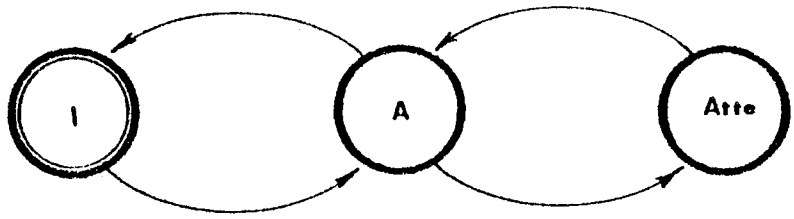


Fig. 4.13 Graphe d'états du processus TRAIDIAL.

5.- Niveaux de priorité d'exécution des processus du PLP.

Dans le PLP, nous distinguons trois niveaux de priorité pour l'allocation du CPU. La répartition des processus dans ces trois niveaux a été faite de la manière suivante:

Un des objectifs du PFG étant de repondre le plus tôt possible aux demandes d'interactions, nous a amené à assigner au processus TRAIDIAL la plus haute priorité, priorité "un".

La priorité "deux" a été assignée aux processus d'émission et de réception du SUPCOM ainsi qu'au processus COMPLP. Ces processus ne sont pas concurrents. Pendant la phase d'affichage, on va exécuter d'abord le processus de réception et après le procesus COMPLP. Pendant la phase de dialogue, après avoir exécuté le processus TRAIDIAL, on va exécuter celui d'émission du SUPCOM.

Dans le niveau le moins prioritaire, nous avons les processus de DEFDES, GENDES et DEMDIAL. Ces processus seront concurrents et pour les synchroniser on va utiliser un allocateur du CPU (cf. IV.6.3.1).

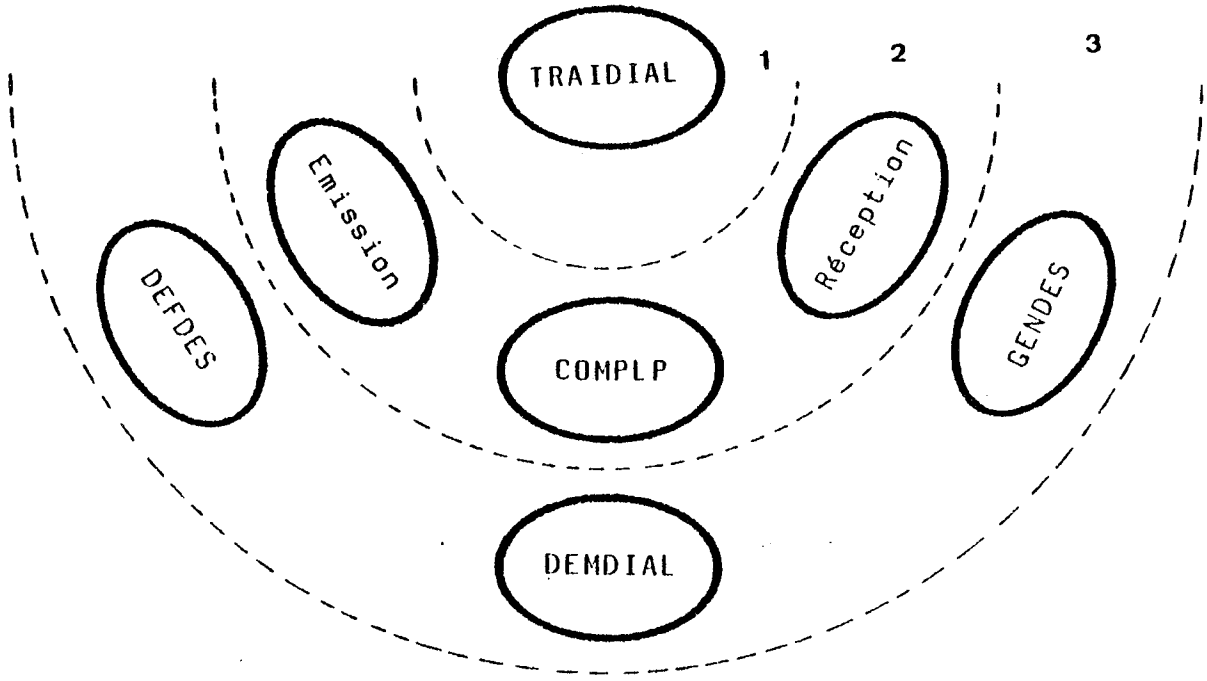


Fig. 4.15 Les niveaux de priorité des procesus du PLP.

5.1 Implémentation des priorités.

Dans II.7 nous avons indiqué que l'allocation du CPU dans le microprocesseur MC6800 peut se faire de deux façons: implicite en utilisant les bascules d'IT (NMI et IRQ), et explicite en utilisant des instructions spéciales (SWI,RTI).

Dans l'implémentation du PLP nous avons utilisé la ligne des interruptions non masquables (NMI) pour demander l'exécution du

processus TRAIDIAL le plus prioritaire. Cette ligne sera positionnée par le PLA à la suite d'une mise en service des dispositifs de dialogue.

La ligne des interruptions masquables (IRQ) sera utilisée pour la prise en compte d'une demande de service du PFG et donc pour l'exécution du processus COMPLP (niveau deux). Par exemple, le CPU sera alloué au processus de réception et après au processus COMPLP, seulement dans les cas où les conditions suivantes seront remplies:

- La bascule NMI n'est pas positionnée.
- Les IT masquables (IRQ) sont autorisées (I=0).
- La bascule IRQ est positionnée comme conséquence d'une demande de service au PFG.

L'exécution d'un des processus du niveau "trois", le moins prioritaire, sera réalisée dans le cas où il n'y a aucune demande d'allocation de la part d'un des processus plus prioritaires. Autrement dit lorque les bascules NMI et IRQ ne sont pas positionnées.

6.- Synchronisation des processus du PLP.

6.1 Processus du niveau "un".

La création du processus TRAIDIAL est faite par le PLA lorsqu'ayant des informations à transmettre au PLP, il positionne la bascule NMI du microprocesseur qui exécute la Préparation.

L'interpréteur câblé du microprocesseur, avant de lire la prochaine instruction à exécuter, exécute la séquence suivante:

- Sauvegarder le contexte du processus courant.
- Masquer les interruptions masquables IRQ.
- Charger le compteur ordinal avec le contenu de l'adresse FFFC-FFFD.

Alors, l'adresse de la première instruction du programme TRAIDIAL sera rangée dans FFFC-FFFD.

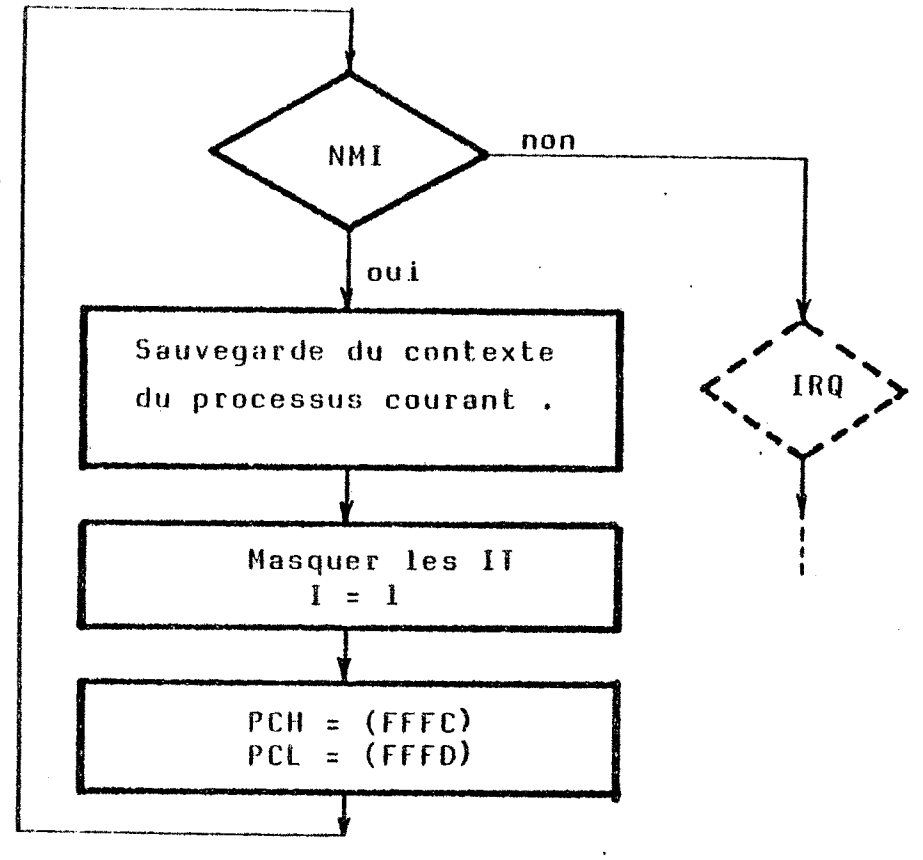


Fig. 4.16 - Séquence lors de la prise en compte d'une IT NMI.

Remarque: Chaque fois que l'interpréteur va lire l'instruction suivante, il teste l'état de la bascule NMI.

Pour garantir la cohérence de l'information à envoyer aux PLM, nous avons défini la procédure TRAIDIAL comme non-réentrante, c'est-a-dire que l'exécution de ce processus ne devra jamais être interrompue. Pour ce faire nous allons interdire au PLA de lancer une nouvelle demande de traitement

pendant l'execution de TRAIDIAL.

Ce mécanisme a été réalisé par le biais d'une variable commune localisée dans la mémoire partagée, et destinée a synchroniser la communication entre le PLP et le PLA. Cette variable joue le rôle de masque pour la ligne NMI. Sa manipulation correspond au type "poignée de main" (hand-shaking).

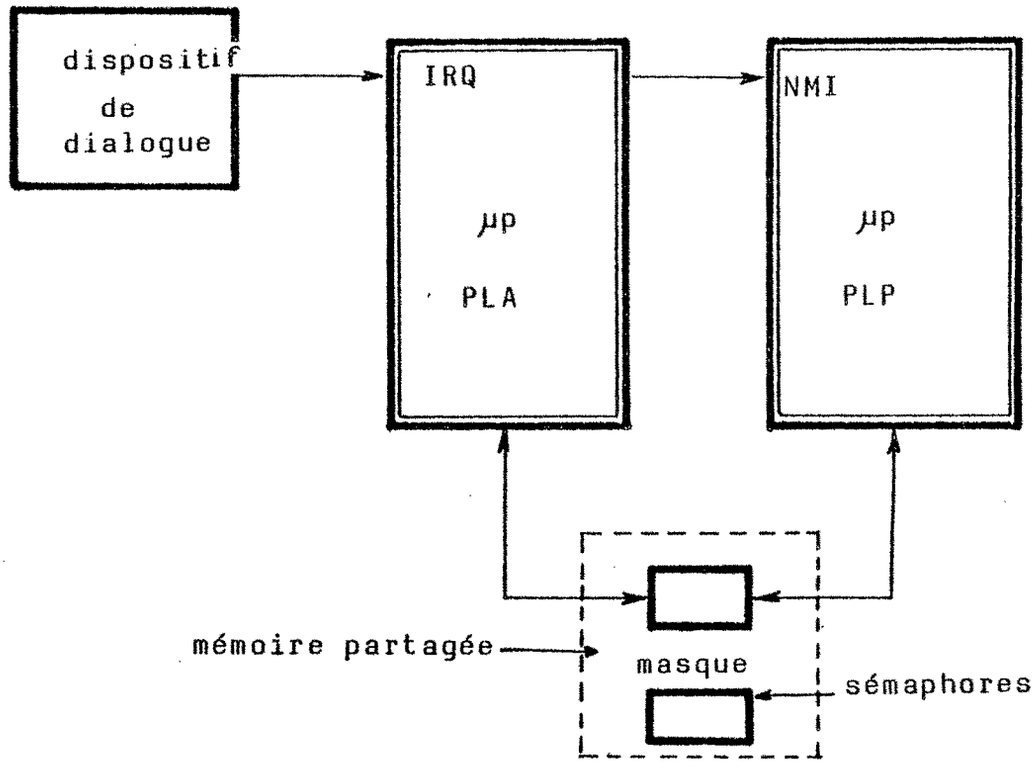


Fig. 4.17 Mécanisme de communication entre le PLP et le PLA

A l'initialisation du Système, le masque sera positionné à zéro, indiquant ainsi la validation des IT NMI. Le PLA (voir processus DIAL V.5.2) après avoir ramassé et rangé dans la zone de communication l'information provenant des dispositifs de dialogue, va tester l'état du masque:

- Si sa valeur est zéro alors il va tout d'abord provoquer

une IT (NMI) au microprocesseur qui fait la Préparation (création du processus TRAIDIAL) et après il mettra le masque à un. C'est au processus TRAIDIAL de valider à nouveau les IT (NMI) en mettant à zéro le masque.

- Si sa valeur est un alors il boucle en attendant la fin du procesus TRAIDIAL (validation à nouveau des IT NMI).

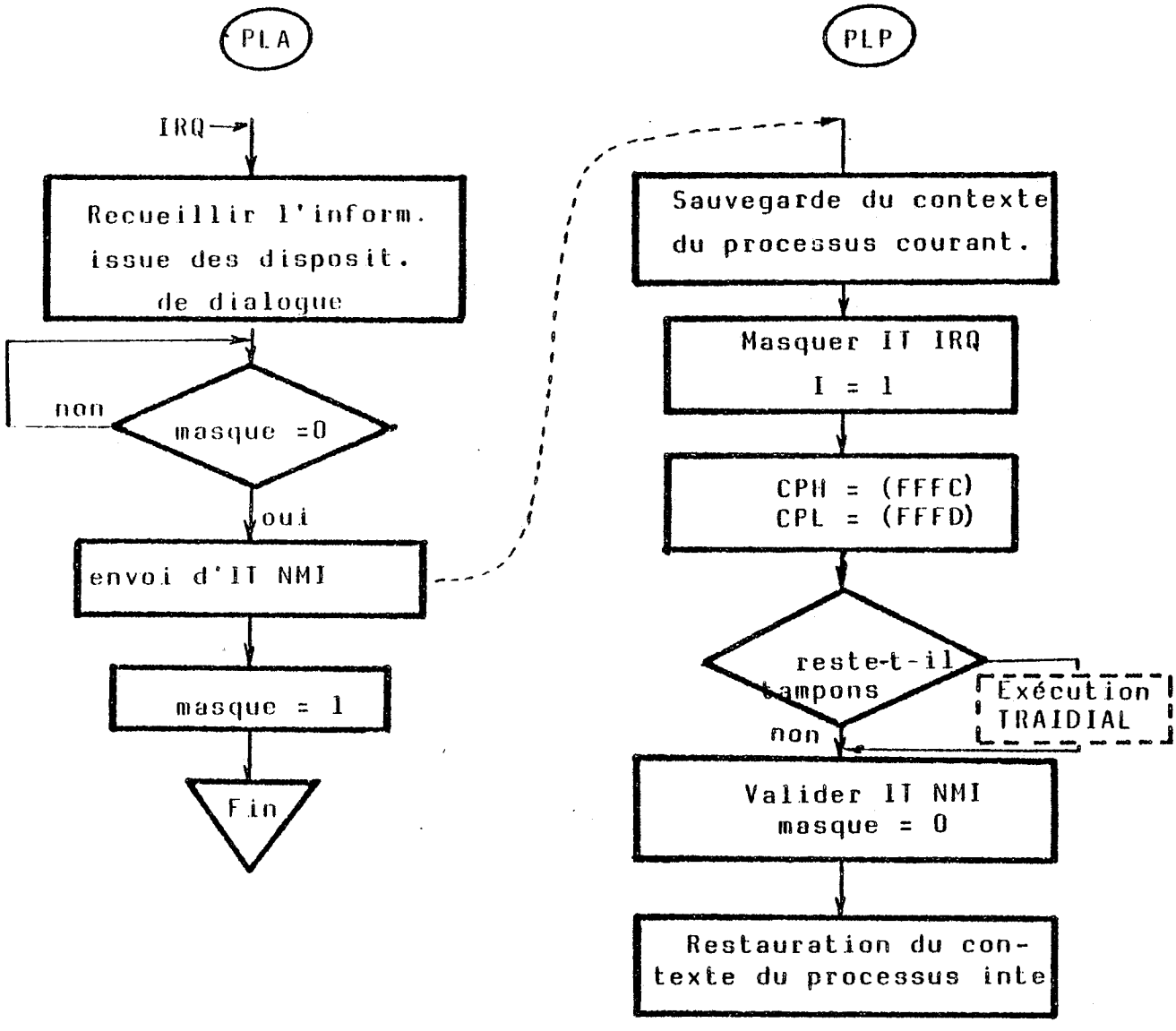


Fig. 4.18 Communication entre le PLP et le PLA lors du dialogue.

Le mécanisme que nous avons décrit permet de synchroniser

l'exécution des processus TRAIDIAL et DIAL, maintenant il faut garantir la cohérence de l'information à échanger entre le PLP et le PLA. Pour ce faire, dans une première étape, nous avons utilisé la technique d'horloges croisées (cf. II.8.2) qui resoud déjà le problème de l'accès élémentaire (1 octet) en exclusion mutuelle à la mémoire partagée.

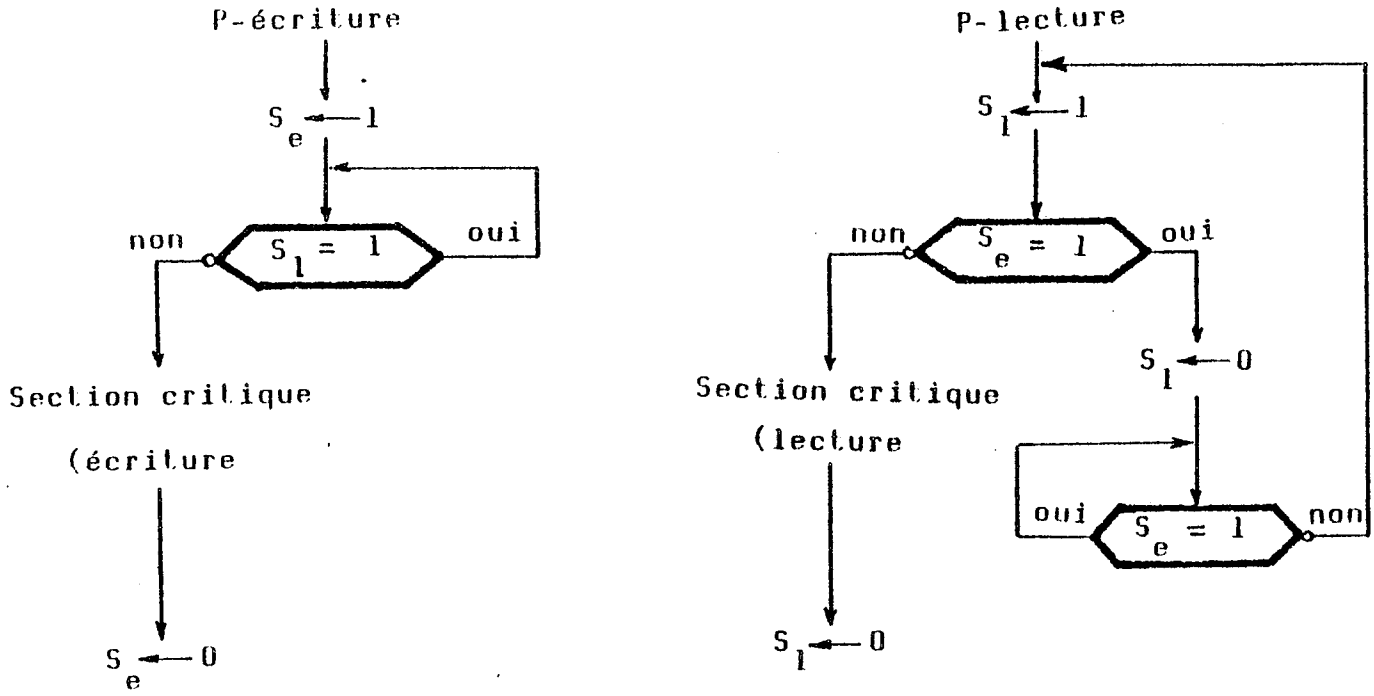
Maintenant il faut définir des mécanismes qui permettent l'accès, en exclusion mutuelle, à toute une zone critique (plusiers octets) de la mémoire partagée.

Dans le cas du PLA du termeinal 30-N (cf. V.4), un exemple zone critique est le pointeur CI de l'entité graphique en exécution lequel tient sur 2 octets. La mise à jour de ce pointeur est faite par le PLA à chaque fois que l'on affiche une nouvelle entité de la liste d'affichage virtuelle.

Pendant la phase de dialogue le processus TRAIDIAL va lire ce pointeur afin d'identifier l'information que l'n veut transmettre au PLM. Par exemple un caractère alphanumérique dont le code ASCII est dans l'opérande de l'entité que l'on vient d'afficher.

Il est tout à fait possible que pendant la lecture de l'octet de poids fort, le PLA ait modifié l'octet de poids faible; alors à la fin de la lecture, la valeur lue par le PLP est erronée.

Afin de garantir qu'à un instant donné il n'y a qu'un seul processus dans la zone critique, nous avons implémenté des mécanismes de mutuelle exclusion répartis dans les deux processus partenaires de la façon suivante:



Remarque: Pour chaque zone critique on utilise deux sémaphores s_e et s_l qui sont rangés dans la mémoire partagée. Une démonstration de la validité de ces algorithmes on peut la trouver dans MAR-79.

6.2 - Synchronisation entre les processus du niveau "deux".

Les processus d'émission, de réception du SUPCOM ainsi que celui de COMPLP ont priorité "deux" et ils ne sont pas concurrents.

La demande de service au PLP se traduit par le positionnement de la bascule IRQ du microprocesseur qui exécute la Préparation. La prise en compte de cette demande constitue la création du processus de réception du SUPCOM du

PLP. Cette prise en compte n'est faite que si les IT masquables sont autorisées. Si c'est le cas alors l'interpréteur câblé exécute la séquence suivante:

- Sauvegarde du contexte du processus courant.
- Masquage des IT IRQ (I=1).
- Chargement du compteur ordinal avec le contenu de l'adresse FFF8-FFF9.

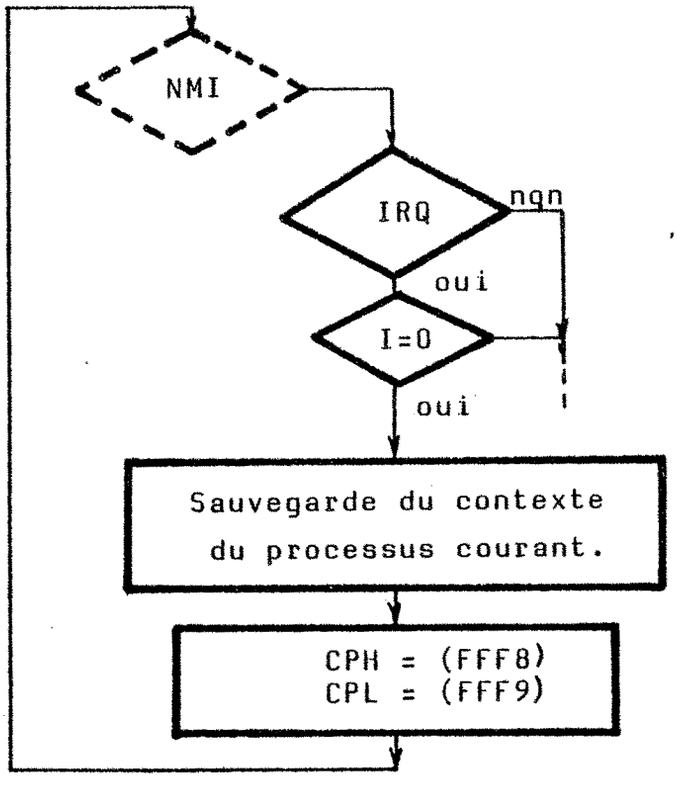


Fig. 4.19 Séquence lors de la prise en compte d'une IT masquable.

A la fin de l'exécution du processus de réception on va passer le contrôle au processus COMPLP, celui-ci, avant de rendre le contrôle au processus interrompu, va tester s'il reste des tampons vides pour pouvoir loger un autre message CORAIL, si c'est le cas alors il va démasquer les IT (I=0).

6.3 - Synchronisation entre les processus de priorité "trois".

Dans le niveau "trois" on trouve les processus de définition de dessins DEFDES, de génération de dessins GENDES et le processus de demande de dialogue DEMDIAL. Ces processus sont concurents vis-à-vis de la demande d'allocation du CPU.

Hansen et Hoare (Han-74) ont developpé le concept de moniteur comme étant un outil pour la structuration de systèmes d'exploitation. L'idée consiste à contrôler l'utilisation d'une ressource par construction d'un "scheduler". Celui-ci étant implémenté comme un moniteur.

La condition minimale pour garantir la cohérence de l'information dans le système est d'exécuter les procédures du moniteur en exclusion mutuelle dans le temps.

Un moniteur est constitué par une collection de procédures et de données qui définissent un type abstrait. La mise en activité de ces procédures constitue un nouveau processus qui joue le rôle d'arbitre.

Dans le PLP nous avons développé un moniteur pour contrôler la ressource CPU du MC6800.

6.3.1 Le moniteur-allocateur du CPU.

L'appel du moniteur allocateur du CPU va permettre de contrôler l'utilisation du CPU MC6800. Ce moniteur comporte deux primitives: SIGNAL et WAIT qui agissent sur la file d'attente FA68 afin de permettre la synchronisation entre les processus de priorité "trois".

La primitive SIGNAL(FA68) permet d'allouer le CPU au processus dont le pointeur de son contexte est dans la tête de la file d'attente FA68.

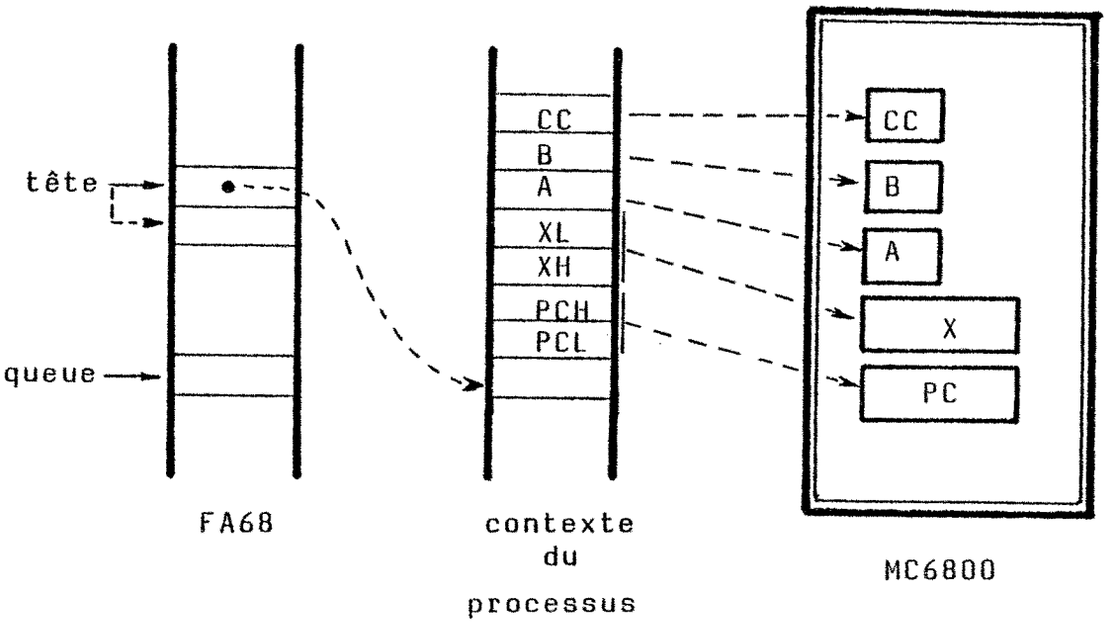


Fig. 4.20 - Séquencement de la primitive SIGNAL(FA68).

La primitive WAIT(FA68) va ranger le pointeur du contexte du processus dans la queue de la file d'attente FA68.

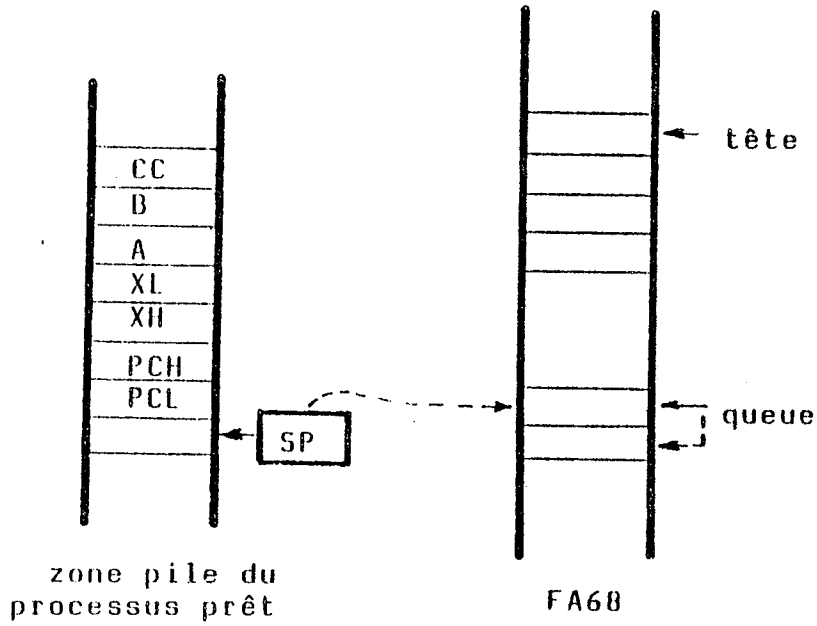


Fig. 4.21 Séquencement de la primitive WAIT(FA68).

Il faut signaler que la gestion de la file d'attente est du type "premier arrivé premier servi" (FIFO). Le schèma du moniteur allocateur du CPU est montré dans la figure 4.22.

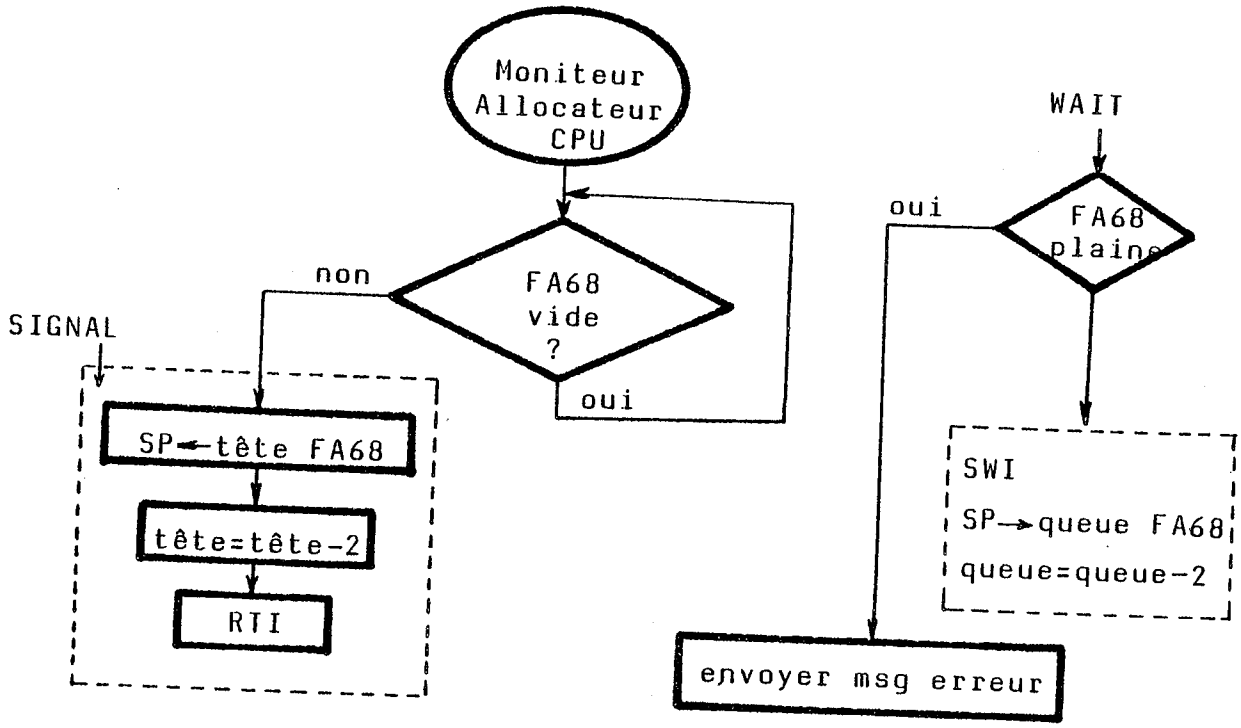


Fig. 4.21 Le moniteur allocateur du CPU MC6800.

Alors, pendant l'exécution du processus COMPLP il analyse le message MSGPLP afin de reconnaître le processus consommateur possible, après quoi il utilise la primitive WAIT pour ranger, dans la file d'attente FA68, le pointeur du contexte initial du processus consommateur. Ceci correspond donc à la création de ce processus consommateur qui passe de l'état inactif à l'état prêt à s'exécuter.

7.- Le fichier GRIGRI.

Dans la suite nous allons faire une description de la structure interne du fichier GRIGRI. Dans cette structure nous n'avons pris en compte que les paramètres nécessaires pour la génération de la liste d'affichage virtuelle destinée au terminal 30-N de DEC.

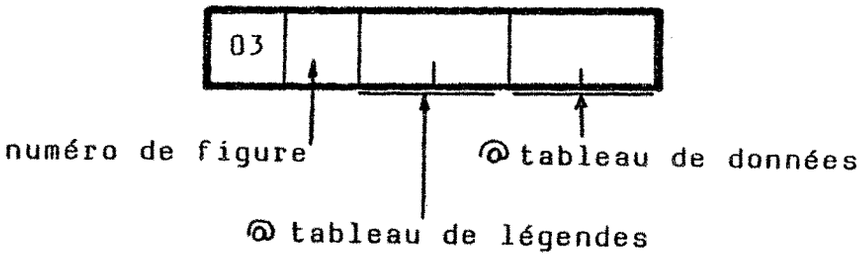
Le fichier GRIGRI est un objet créé au fur et à mesure que le processus DEFDES travaille sur les messages MSGPLP de génération de dessins. L'information du fichier GRIGRI est répartie dans l'ensemble de tableaux suivants:

- Tableau de descripteurs des entités de définition de données
- Tableau de données.
- Tableau de descripteurs de modes de représentation.
- Tableau de descripteurs des légendes.
- Tableau de descripteurs des fenêtres.
- Tableau de descripteurs des clôtures.
- Tableau des entités affichées.
- Tableau des entités pour le dialogue.

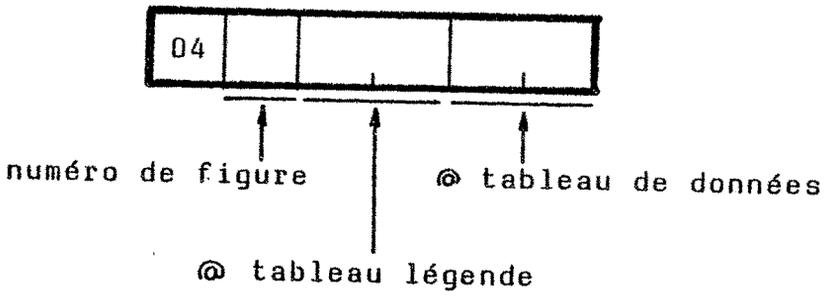
7.1 - Tableau des descripteurs des entités de définition de données.

Dans ce tableau on va ranger les descripteurs des données définies au cours d'une séance. Ces descripteurs correspondent aux quatre entités suivantes:

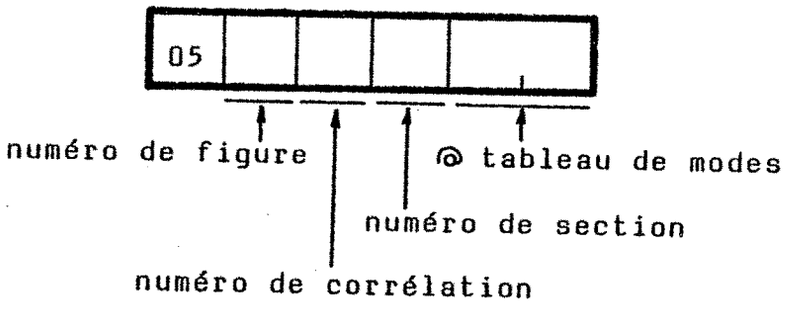
- Définition de points:



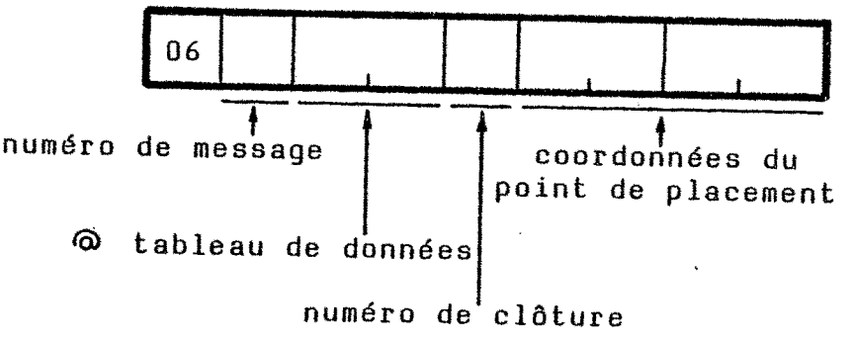
- Définition de textes:



- Définition de mode:



- Définition d'un message:



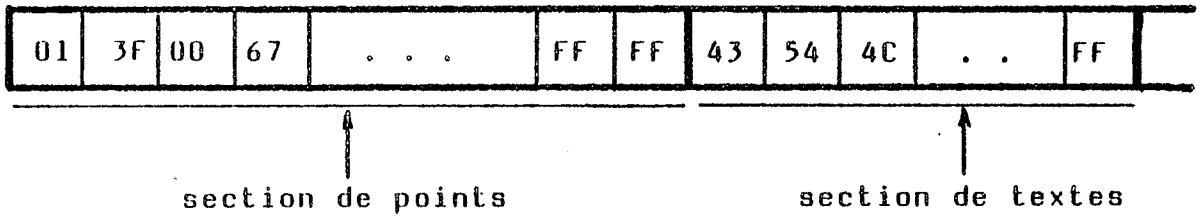
7.2 - Tableau de données.

Dans ce tableau on range les données proprement dites du fichier graphique. Le remplissage du tableau se fait au fur et a mesure que le processus DEFDES consomme les messages de la file d'attente FADD. Ce tableau est divisé en sections de longueur variable, de telle sorte que chaque section contient un type spécifique de données. Par exemple une section de points n'est composée que de coordonnées de points, par contre une section du type texte ou message est composée de caractères codés en ASCII.

Pour indiquer la fin d'une section, nous avons pris la convention suivante:

- Pour une section de points, on range à la fin de la section les coordonnées FFFF FFFF.

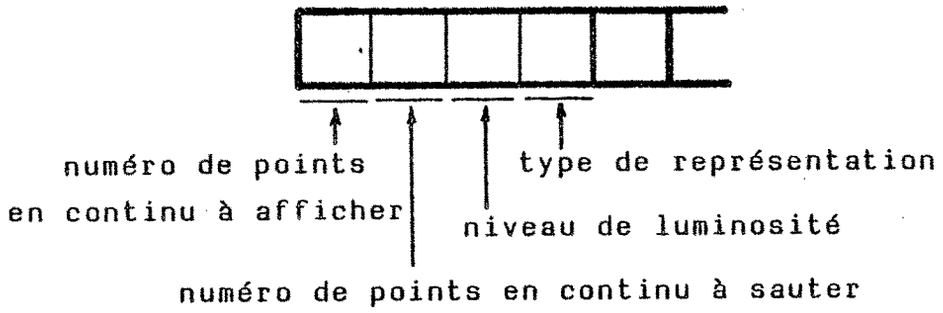
- Pour une section de texte ou message, on range à la fin le code FF.



7.3 - Tableau de descripteurs de mode de représentation.

Ce tableau contient les descripteurs qui définissent les attributs associés à chaque section de données. Ces attributs sont codés de la manière suivante:

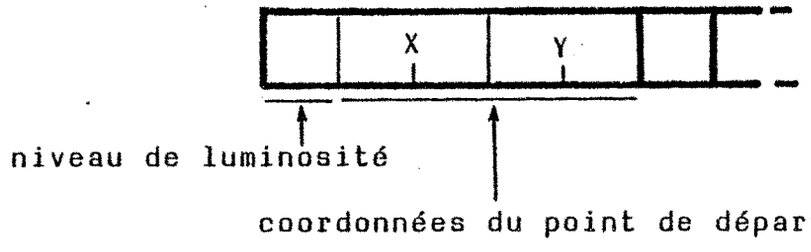
- Attributs pour une section de points:



Les deux premiers paramètres spécifient le type de tracé souhaité. Pour le niveau de luminosité nous n'avons défini que huit niveaux dont le niveau 00 correspond au moins lumineaux et le niveau 07 correspond au plus lumineux. Le type de représentation définit le type d'entité graphique qu'il faut créer dans la liste d'affichage virtuel. Ces entités sont codées de la façon suivante:

00 : points isolés.
01 : vecteurs isolés.
04 : vecteurs enchaînés.

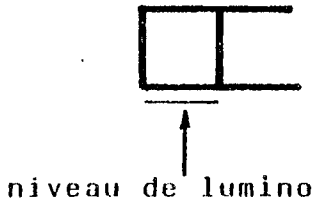
- Attributs pour une section type texte:



Les coordonnées du point de départ définissent l'emplacement dans l'écran où va se trouver le premier point du premier

caractère du texte.

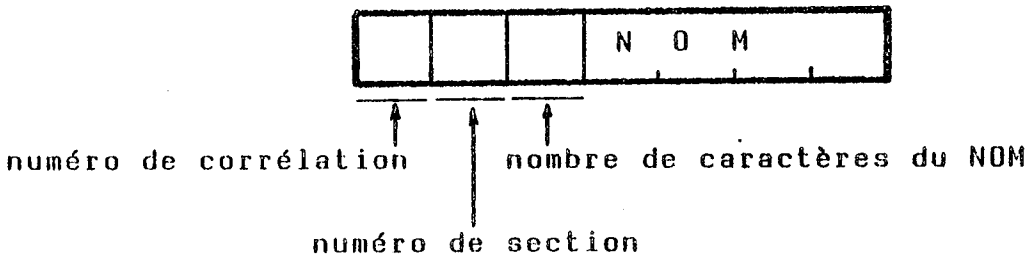
- Attributs pour une section type message:



Les coordonnées du point du premier caractère du message sont données dans le descripteur type message du tableau de définition de données (IV.7.1).

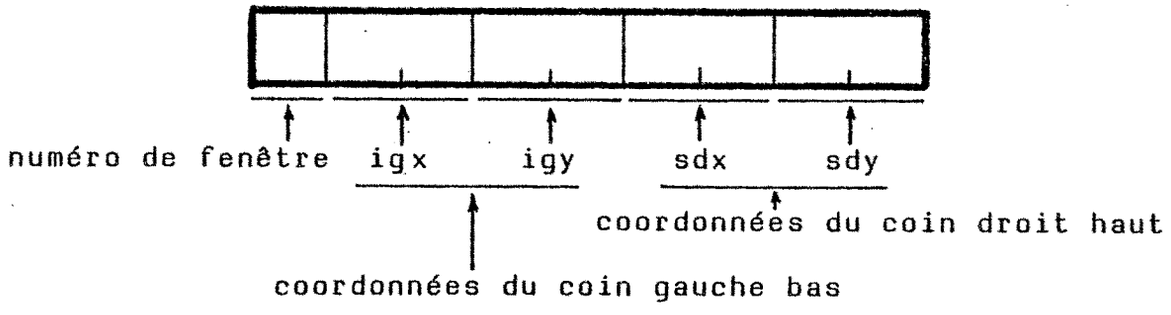
7.4 - Tableau de descripteurs de légendes:

Ce tableau est constitué par des descripteurs qui contiennent des informations associées à chacune des sections du tableau de données (IV.7.2) Le descripteur contient le numéro de corrélation associé à la section, le numéro de la section ainsi que le nom (marqueur) que l'utilisateur a associé à cette section.



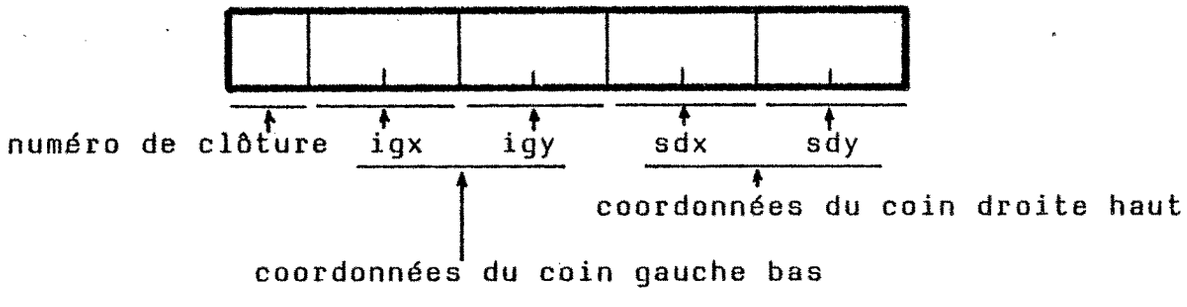
7.5 Tableau de descripteurs des fenêtres.

Ce tableau contient les descripteurs qui correspondent à la représentation interne de la définition des fenêtres. Le format du descripteur est le suivant:



7.6 Tableau de descripteurs des clôtures.

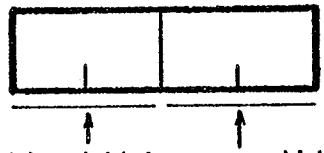
Ce tableau est similaire au dernier , le format du descripteur est le suivant:



7.7 Tableau de descripteurs des entités affichées.

Dans ce tableau, les descripteurs qui le composent, contiennent l'information nécessaire qui permet de retrouver dans la liste d'affichage, l'entité (ou entités) graphique(s) correspondant à une section du fichier GRIGRI. Le descripteur est formé par un couple d'adresses dont la première correspond à

l'adresse d'implantation de l'entité dans le tableau de définition d'entités (IV.7.1), et la deuxième correspond à l'adresse d'implantation de l'entité graphique dans la liste d'affichage.



@ d'implantation de l'entité dans la tableau d'entités définies

@ d'implantation dans la liste d'affichage

Ce tableau va permettre de retrouver une entité graphique lorsqu'on veut l'identifier pour la modifier.

7.8 Tableau de demande de dialogue.

Ce tableau sera rempli par le processus DEMDIAL (cf.IV.4.5), et il va contenir l'information nécessaire au processus DIAL (cf. V.5.2) pour qu'il puisse contrôler et traiter l'information issue des dispositifs de dialogue et à envoyer aux PLM. Les différents types d'actions ainsi que leurs paramètres sont codés de la manière suivante:

- Pour l'édition de données alphanumériques:

code numéro de zone

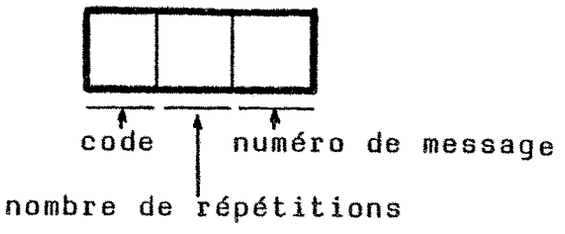
numéro de message

Où les codes sont:

00 : Edition de données type entier.

01 : Edition de données type réel.
02 : Edition d'une chaîne.

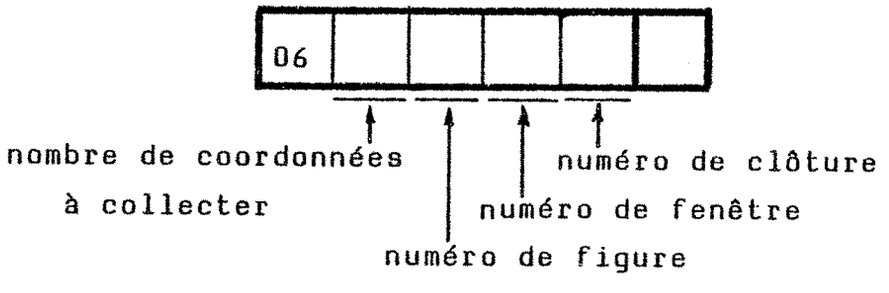
- Pour l'introduction de données alphanumériques:



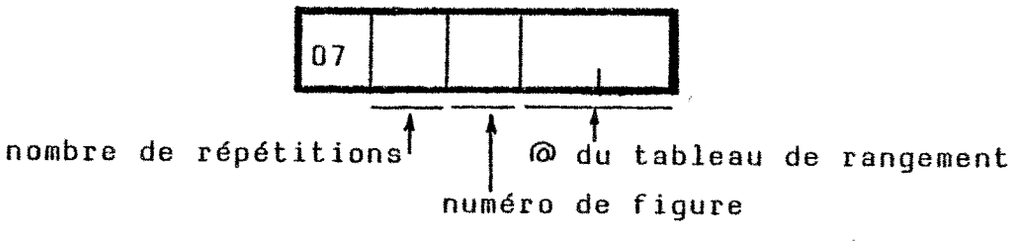
Où les codes sont:

03 : Introduction de données réelles.
04 : Introductions d'entiers.
05 : Introduction d'une chaîne.

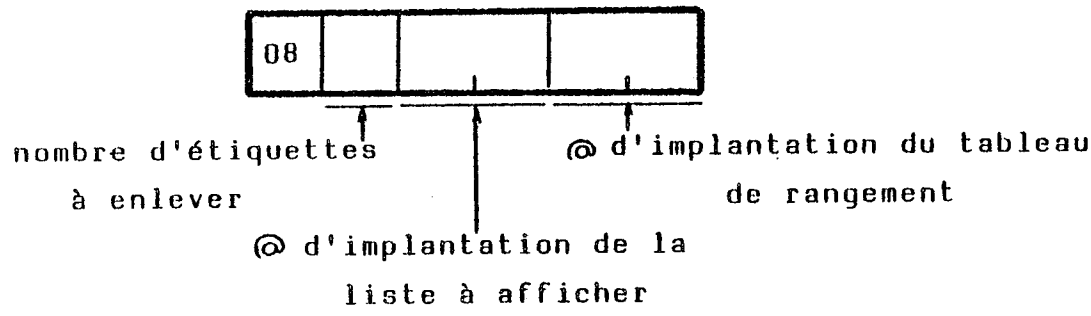
- Pour la collecte de coordonnées:



- Pour l'identification:



- Pour un menu:



Le Processeur Fonctionnel Graphique, étant exploité en mode multiprogrammé, doit garantir une manipulation correcte de l'information de la part des utilisateurs. Par exemple, pour un objet particulier à un utilisateur, lui seul aura le droit d'y accéder.

Ceci conduit à classer les objets du fichier GRIGRI en deux groupes:

- Les objets qui peuvent être partagés, c'est-à-dire auxquels n'importe quel utilisateur peut accéder.

- Les objets particuliers à chaque utilisateur.

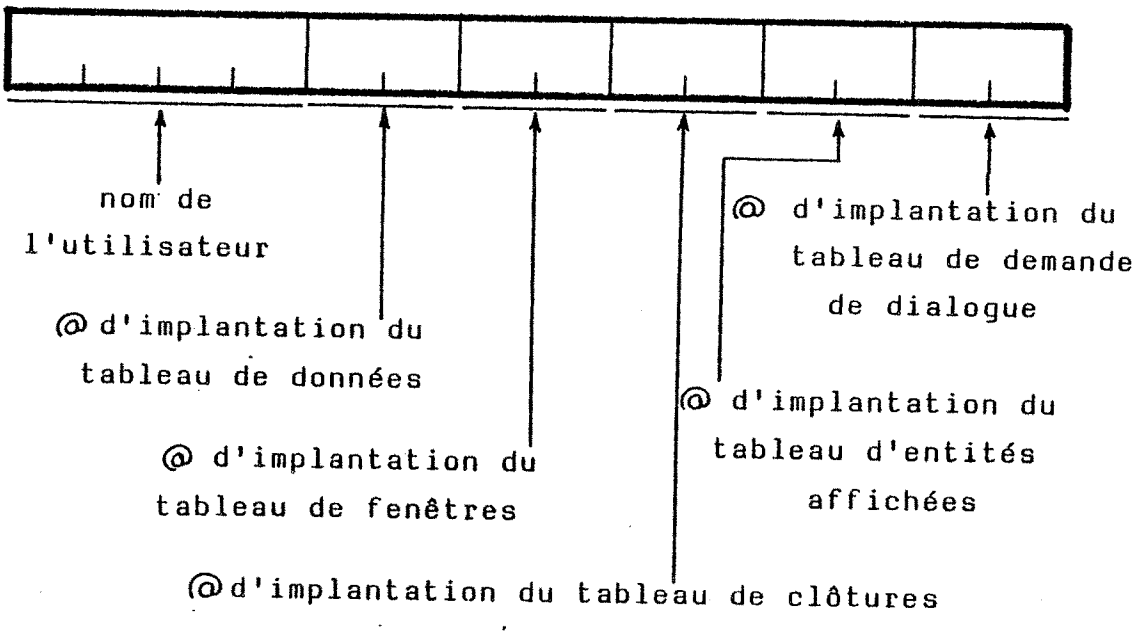
Dans le premier groupe nous trouvons:

. Le tableau de données.
. Le tableau de descripteurs de légende.
. Le tableau de descripteurs de mode de représentation.

Par contre, le reste des tableaux sont des objets particuliers pour chaque utilisateur.

Dans le PLP, le processus COMPLP va associer à chaque utilisateur un jeu d'objets de telle sorte que pendant une

séance, ils ne seront accédés que par son propriétaire. La gestion de ces objets se fera à l'aide d'un catalogue qui permettra d'identifier les objets de chaque utilisateur. Le format d'un descripteur du catalogue est le suivant:



8.- Architecture du PLP.

L'architecture interne du PLP est constituée par un ensemble d'organes nécessaires aux fonctions à réaliser. La ressource commune de communication entre ces organes est un bus parallèle constitué d'un ensemble de lignes de données, adresses et contrôle. Les organes qui composent l'architecture sont:

- Un microprocesseur MC6800.

- Un organe pour les calculs numériques, AM9511.

- Un organe specialisé en entrées/sorties PIA, pour la communication avec les Processeurs de Traitement de CORAIL (voir les PLM).

- Des organes de mémorisation statique (ROM) pour y ranger les procédures des processus qui seront exécutées dans le PLP.

- Des organes de mémorisation dynamique (RAM) pour y ranger:
  + Les zones-piles ainsi que les variables de travail des processus,
  + Le fichier GRIGRI,
  + La liste d'affichage virtuelle destinée au PLA du 30-N de DEC.
  + Les variables pour la synchronisation avec le PLA.
  + Les informations nécessaires au contrôle des interactions.

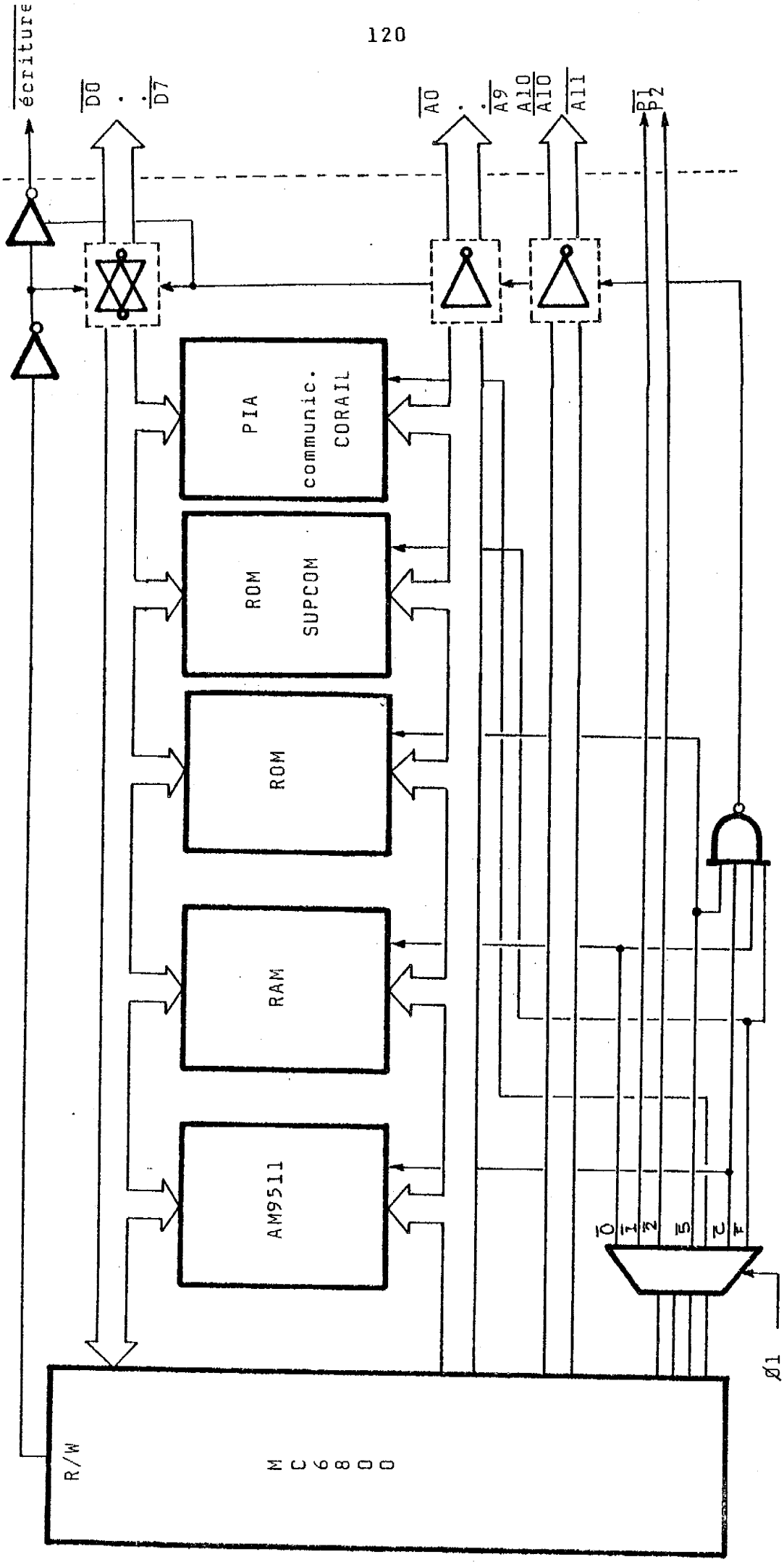


Fig. 4.23 L'architecture du PLP

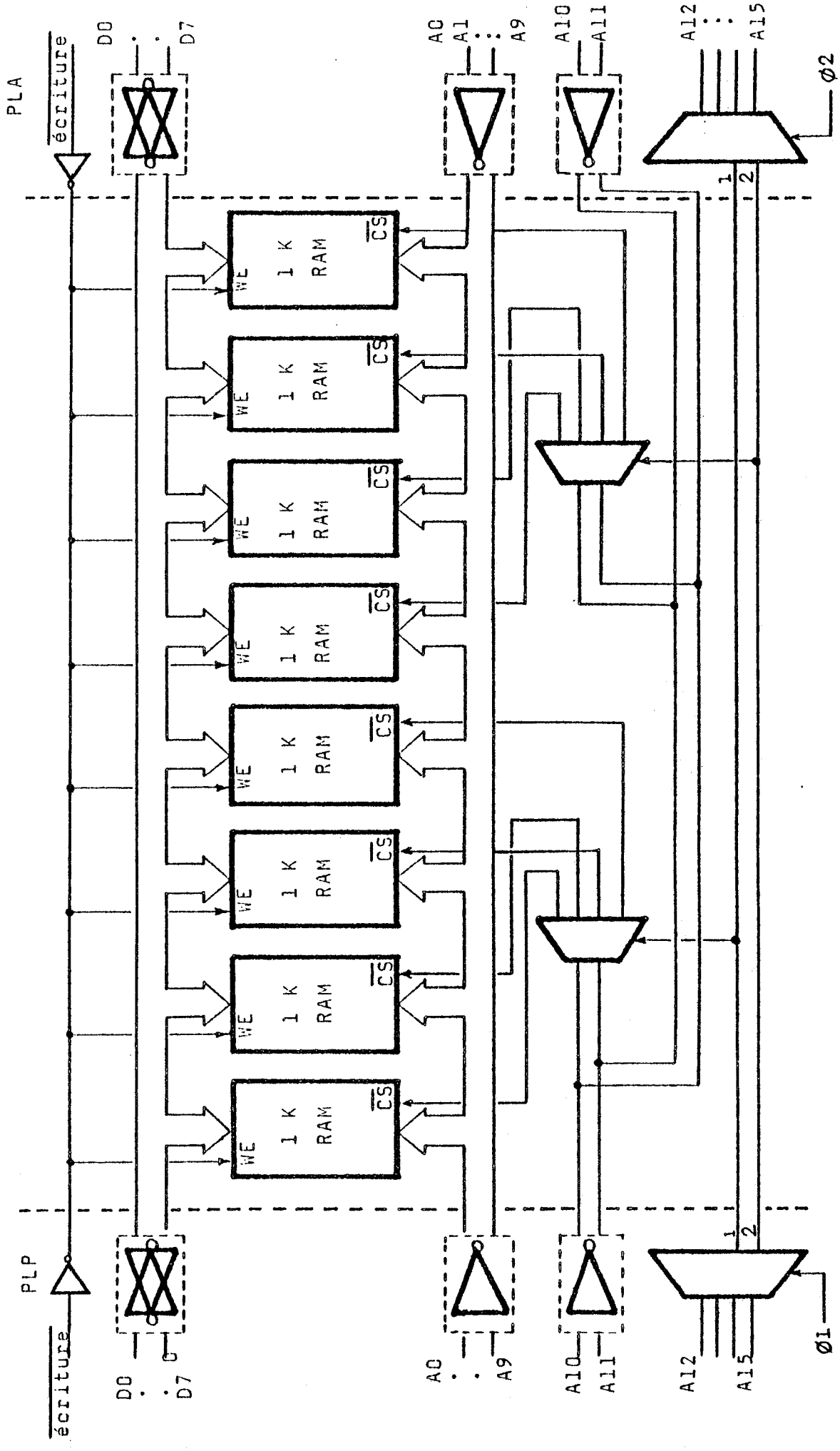


Fig. 4.24 La mémoire partagée

8.1 - Communication avec le PLA.

La communication entre le PLP et les PLA est faite par mémoire commune (cf. II. 8.2). La synchronisation pour l'accès à cette mémoire a été résolue en utilisant des horloges croisées (voir fig. 2.6).

La mémoire commune est divisée en deux zones:

- La première zone va contenir la liste d'affichage virtuelle engendrée par le processus GENDES (cf.IV.4.4).

- La deuxième zone va contenir:
  - . Les variables de synchronisation (cf. masque IV.6.1, sémaphores cf. IV.6.1).
  - . Les variables qui permettront au PLA de contrôler les interactions par exemple le type d'interaction, le nombre de coordonnées à collecter,..etc.
  - . Les informations recueillies par le PLA et qui seront traitées par le PLP avant d'être envoyées aux PLM.

8.2 - Protection matérielle.

Dans le PFG, nous avons limité le pouvoir d'accès à la mémoire commune en implémentant une protection matérielle de la façon suivante:

- La zone mémoire qui contient la liste d'affichage virtuelle sera accédée par le PLP en lecture/écriture, tandis que le PLA ne va l'accéder qu'en lecture. La zone mémoire qui contient les informations destinées à contrôler les interactions, sera aussi accédée de la même façon.

- La zone contenant les variable de synchronisation sera accédée par les deux processeurs en lecture/écriture.

- La zone qui contient les informations à envoyer au PLP, sera accédée par celui-ci en lecture, tandis que le PLA va l'accéder en lecture/écriture.

8.3 - Allocation de la mémoire dans le PLP.

Les procédures qui constituent le PLP, ainsi que les variables de travail, ont été réparties dans l'espace mémoire du microprocesseur comme le montre la fig. 4.25.

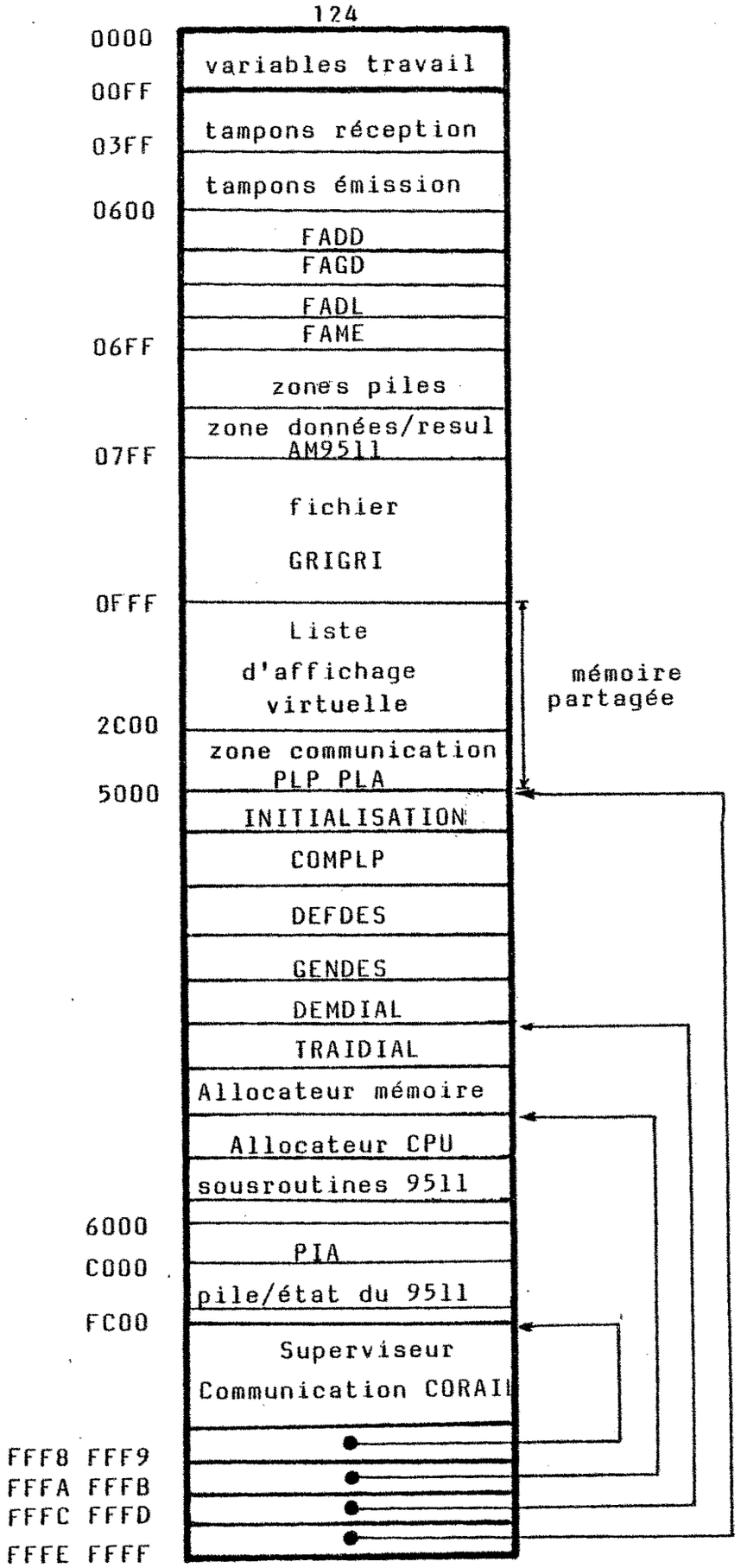


Fig. 4.25 Organisation de la mémoire du microprocesseur qui fait la Préparation à la visualisation.

Les 2K octets de mémoire RAM, comprises entre 0800 et 0FFF vont loger les tableaux qui définisent le fichier GRIGRI. Cette zone de la mémoire a été segmentée en blocs de taille fixe de 256 octets chacun. Ils seront alloués à la demande par le biais d'un allocateur. L'appel à cette allocateur sera fait par le processus DEFDES dès que la place libre dans un bloc sera épuisée.

La liaison entre les blocs mémoire est faite à l'aide de pointeurs localisés dans le bloc lui même. Le format d'un bloc est montré dans la figure 4.26. L'algorithme de l'allocateur de mémoire est montré dans la figure 4.27.

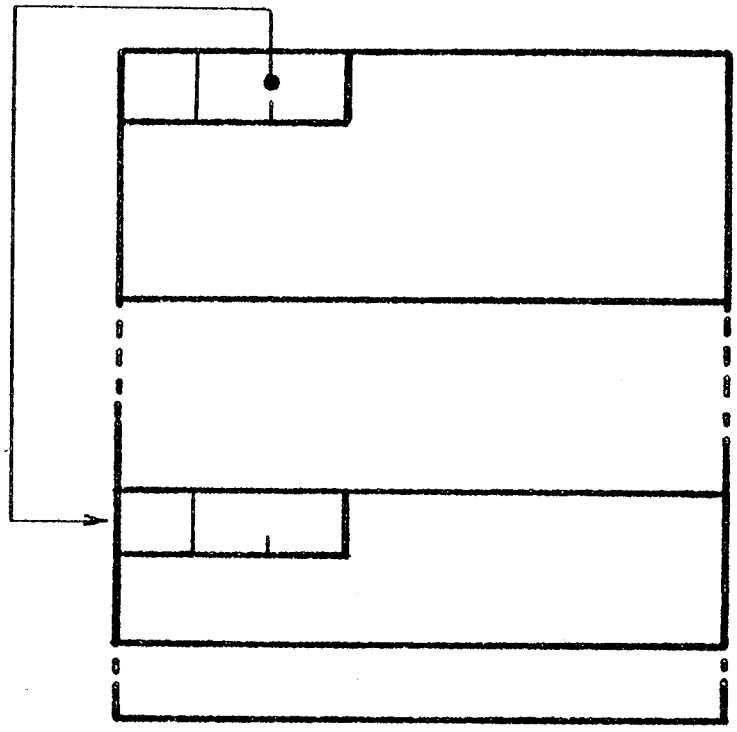


Fig. 4.26 Le bloc mémoire dans le PLP.

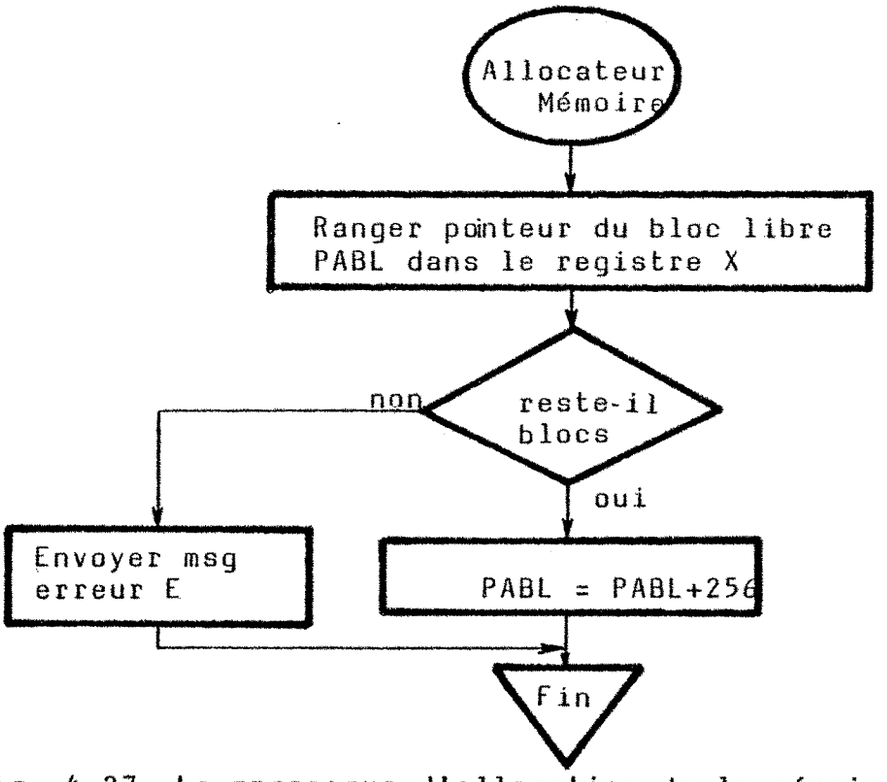


Fig. 4.27 Le processus d'allocation de la mémoire du PLP.

9.- Traitement d'erreurs.

Dans le PLP, nous distinguons trois sources d'erreurs:

- Erreurs provoquées par une fausse manipulation des objets d'un dessin, par exemple lorqu'on veut afficher une section inexistante.

- Erreurs arithmétiques, par exemple un dépassement lors d'un calcul arithmétique.

- Erreurs sémantiques provenant d'une mauvaise utilisation d'un dispositif de dialogue, par exemple l'envoi d'une donnée dont le type ne correspond pas au type attendu.

Dans le traitement des erreurs nous nous contentons de finir le processus courant en envoyant un message à l'utilisateur. Ces messages sont codés de la manière suivante:

El : La place mémoire pour ranger le fichier GRIGRI est épuisée.

E2 : L'entité à afficher n'existe pas.
E3 : L'entité à effacer n'existe pas
E4 : Incompatibilité entre les données envoyées et celles attendues.
E5 : Dépassement lors d'un calcul numérique.

10.- Evaluation de performances du PLP.

Dans la suite, nous allons donner quelques chiffres de performance: La place mémoire occupée par les procédures du PLP ainsi que le temps moyen d'exécution des processus. Dans la programmation des primitives GRIGRI nous nous sommes limités au cas où les paramètres nfig, nsec,... sont supérieurs ou egaux à zéro (voir Annexe 2).

Les chiffres donnés ensuite correspondent au cas mono-utilisateur.

- Place mémoire:

. procédure COMPLP : 319 octets.
. procédure DEFDES : 893 octets.
. procédure GENDES : 537 octets.
. procédure DEMDIAL : 629 octets.
. procédure TRAIDIAL : 418 octets.
. procédure ALLOMEM : 316 octets.
. procédure ALLOCPU : 132 octets.

- Temps d'exécution:

Tout d'abord il faut signaler que le microprocesseur MC6800

utilise en moyenne trois cycles machine pour exécuter une instruction. Le cycle machine est donné par l'horloge du Système CORAIL et correspond à 1,045 microsecondes (921,6 Khz).

Le temps d'exécution pour chacun des processus du PLP n'est pas toujours le même, puisqu'il dépend du nombre de tests à faire, de la longueur de tableaux à analyser, du nombre de données à ranger, ...etc. C'est pour cette raison que nous n'allons donner qu'un temps moyen d'exécution.

- processus COMPLP : 900 µs.
- processus DEFDES : 2700 µs.
- processus GENDES : 2600 µs.
- processus DEMDIAL : 2000 µs.
- processus TRAIDIAL : 2000 µs
- processus ALLOMEM : 600 µs.
- processus ALLOCPU : 350 µs.

Si on prend en compte le temps total mis par le PLA et par le PLP lors de la prise en compte d'une interaction issue du dispositif de dialogue (voir le processus DIAL V.5.2), l'information sera prête en quelques 2050 µsec. pour être envoyée au PLM.

Dans le Système CORAIL, si on considère une configuration de 8 Processeurs, dans le pire des cas il faudra attendre 51,2 msec. pour obtenir le bus CORAIL. Ceci veut dire que au plus tard, quelques 54 msec. aprés la mise en service du dispositif de dialogue, l'information de l'interaction sera envoyée au PLM pour être rangée dans la structure de données de l'application. Ceci ne prendra que quelques milisecondes (voir le processus de ramassage III.3.3). Donc le temps nécessaire pour traiter une interaction est de 58 msec. environ.

CHAPITE V

LE PROCESSEUR LOGIQUE D'AFFICHAGE (PLA)

1.- Introduction.
2.- Les terminaux graphiques.
3.- Contraintes du temps réel.
4.- Conception et réalisation du PLA pour le terminal 30-N de DEC
5.- Les processus du PLA de terminal 30-N de DEC.
6.- Architecture du PLA.
7.- Evaluation de performances du PLA.

V.- LE PROCESSEUR LOGIQUE D'AFFICHAGE (PLA).

1.- Introduction.

Le Processeur Logique d'Affichage PLA est un processeur specialisé conçu pour commander le terminal graphique. Il peut être vu comme la partie intelligente de ce terminal.

Ce processeur assure l'interface entre le Processeur Logique de Préparation (PLP) et le terminal graphique. La liste d'affichage virtuelle engendrée par le PLP constitue le programme que le PLA va interpréter afin de créer les commandes propres au contrôle du terminal graphique.

Le PLA étant la partie intelligente de l'organe d'afichage, aura à sa charge les tâches suivantes:

- Décoder les ordres graphiques contenus dans la liste d'affichage virtuelle afin de: soit créer une liste d'afichage réelle dans le cas de terminaux possédant une mémoire d'entretien, soit engendrer directement les commandes de contrôle propre au terminal graphique. On obtient ainsi le dessin final sur l'écran.

- Faire la gestion des ressources du terminal graphique, telles que mémoire d'entretien, générateurs câblés de fonctions graphiques, dispositifs de dialogue, ...etc.

- Ramasser l'information à envoyer au PLP pendant le dialogue.

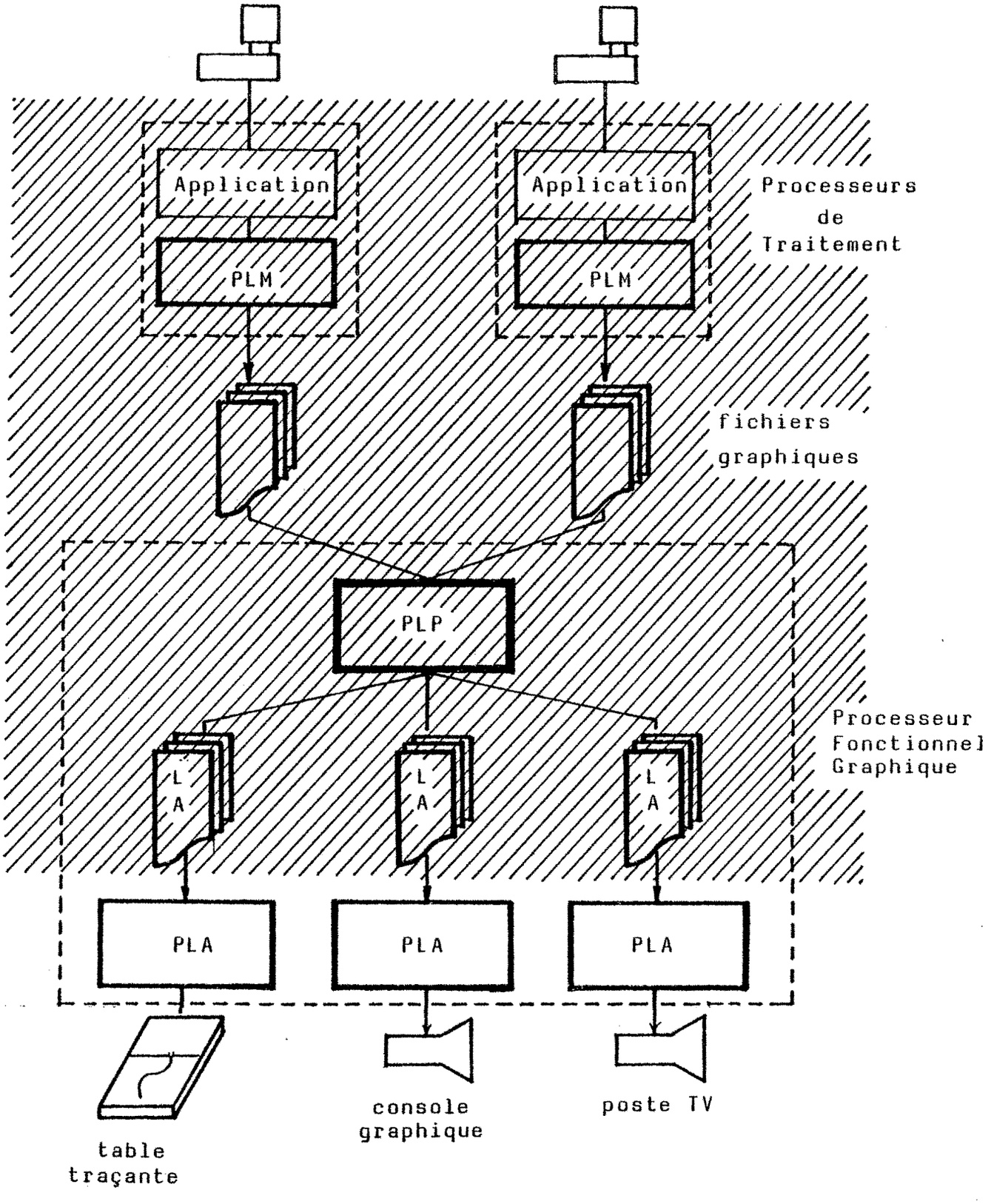


Fig. 5.1 - Topologie du PLA.

2.- Les terminaux graphiques.

Ici, nous allons faire un résumé assez bref des caractéristiques des organes d'affichage, en n'analysant que les parties suivantes:

- La mémoire d'entretien.
- L'écran.
- La partie intelligente, souvent appelée le processeur graphique.
- Les dispositifs de dialogue.

2.1 - La mémoire d'entretien.

Dans (LED-77) on trouve la définition de mémoire d'entretien comme tout dispositif capable d'assurer la permanence de l'image sur l'écran.

2.1.1 - Les types de mémorisation.

Ils peuvent être classés selon deux critères:

- Mémoire vive

- Mémoire synthétique ou mémoire analytique.

Les terminaux à balayage cavalier et à balayage télévision (cf.V.2.2.3) qui ont une mémoire de rafraîchisement, disposent d'une mémoire vive dans laquelle on peut faire des opérations de lecture/écriture.

Les informations contenues dans la mémoire d'entretien sont:

- Soit synthétiques, si elle contient un modèle de l'image constitué de primitives graphiques de haut niveau, par exemple affichage de points, de segments,.. etc. Dans ce cas là, cette représéntation va être capable de fournir un moyen de retrouver ces divers éléments dans la mémoire.

- Soit analytiques ou point par point, si elle contient uniquement des renseignements sur l'état de chaque point de l'écran, par exemple: allumé ou éteint, couleur, ...etc. Cette représentation ne fournit aucun renseignement sur la façon dont l'image a été construite.

Si dans le Système Graphique Interactif on a un terminal à mémoire d'entretien analytique, la liste d'affichage virtuelle engendrée par le PLP sera considérée comme la mémoire d'entretien qui, contenant une description structurée de l'image, facilitera les opérations d'identification et d'effacement sélectif.

2.2 - L'écran.

En général, les écrans des consoles graphiques ne sont pas identiques, les différences essentielles entre eux portent sur les caractéristiques suivantes:

- dimension et forme.
- quantité d'information affichable.
- type de balayage.
- système de coordonnées.
- la couleur et la luminosité.

- l'effacement sélectif.

2.2.1 - La dimension et la forme.

La forme de l'écran n'est pas unique, elle peut être carrée, circulaire, rectangulaire, etc. Dans le SGI, la notion de PLP permet aux utilisateurs d'une application de travailler dans ses propres repères et coordonnées et de ne pas se soucier des dimension et taille de l'écran. C'est donc au PLP d'adapter l'espace utilisateur à l'espace de l'écran réel.

2.2.2 - La quantité d'information affichable.

Ceci est lié à la nature de l'écran et au type de mémorisation (V.2.1.1). Les mémoires analytiques permettent une utilisation maximale de l'écran, par contre les terminaux à mémoire d'entretien sont limités par la taille de celle-ci.

2.2.3. - Le type de balayage.

Les terminaux graphiques, selon le type de balayage, peuvent être classés en deux catégories:

- Balayage cavalier. Ici, le faisceau est asservi en X et Y dans un système cartésien de coordonées rectangulaires et décrit une figure par positionnements successifs.

- Balayage récurent ou ligne par ligne (terminaux à tube télévision) dont l'image résulte de deux opérations simultanées: l'analyse de la surface de l'écran par le faisceau cathodique et la modulation de ce faisceau par des

impulsions appelées signaux d'image. Le faisceau est soumis à des champs orthogonaux variables suivant une loi dite en dents de scie et la trace du faisceau sur l'écran constituée une trame. L'image sera donc constitué par un réseau de lignes parallèles dont chacun des points sera plus ou moins lumineux.

Ce paramètre va jouer un rôle important dans la conception du PLA. En effet, si le terminal est de type balayage ligne par ligne, le PLA devra construire une image analytique (point par point) à partir des informations contenues dans la liste d'affichage virtuelle. Dans le cas du traitement de surfaces, si le terminal est du type cavalier, il faudra implémenter des algorithmes pour le remplisage de lignes.

2.2.4. - Le système de coordonnées.

Le système de coordonnées de divers matériels sont en général fonction de la précision offerte. Celle-ci varie entre 0,3 mm pour les tubes à balayage cavalier, et 0,01 mm pour certaines tables traçantes. Dans le SGI, le problème est résolu au niveau du PLP qui va utiliser la précision maximale acceptée par l'organe d'affichage pour transmettre les coordonnées dans la liste d'affichage virtuelle. Par exemple dans le cas du terminal 30-N on utilise une précision tenant sur 10 bits.

2.2.5 - La couleur et la luminosité.

Les possibilités offertes par le matériel sont ici très diverses:

- monochrome avec un ou plusieurs niveaux de luminosité.

- multichrome sur des traits ou sur des lignes.

Dans le SGI, la solution que nous avons prise est d'offrir à l'utilisateur un maximum d'options. C'est la tâche du PLP d'engendrer une liste d'affichage virtuelle qui soit bien adaptée au terminal. Autrement dit, pendant le codage de cette liste, le PLP va négliger les options que le terminal ne pourra pas prendre en charge et que le PLA ne pourra pas simuler non plus. Par exemple, la couleur dans le cas d'un terminal monochrome.

2.2.6. - L'effacement sélectif.

Ceci est fonction du type de mémorisation. Afin de le rendre possible sur tous les terminaux il suffit d'avoir une représentation structurée de l'image. Dans le SGI, pour les terminaux qui n'ont pas une mémoire d'entretien, la liste d'affichage virtuelle va se comporter comme une mémoire d'entretien.

2.3 - La partie intelligente ou "processeur graphique".

La partie intelligente des terminaux graphiques assure des fonctions diverses. Les instructions qu'elle exécute se divisent en instructions graphiques et en instructions non-graphiques.

Les instructions graphiques composent un répertoire d'ordres de tracé élémentaire tels que: points, traits, caractères,... etc. Ces ordres élémentaires peuvent être précisés par des attributs comme la taille, couleur, texture,... etc.

Les instructions non-graphiques sont en général liées à la présence d'une mémoire d'entretien structurée. Elles peuvent être

classées en deux groupes:

- Le premier groupe comporte des instructions de rupture de séquence et d'appels de sous-programmes. Ces instructions permettent de structurer un modèle de l'image dans la mémoire d'entretien.

- Le deuxième groupe d'instructions se trouve dans les consoles à haute performance et concerne l'ensemble des transformations géométriques.

Dans le SGI, la notion de PLA va permettre de pallier la plupart des défauts d'un terminal peu évolué. Par exemple le manque de générateurs câblés de segments et de caractères, sera pallié par la programmation de ces générateurs.

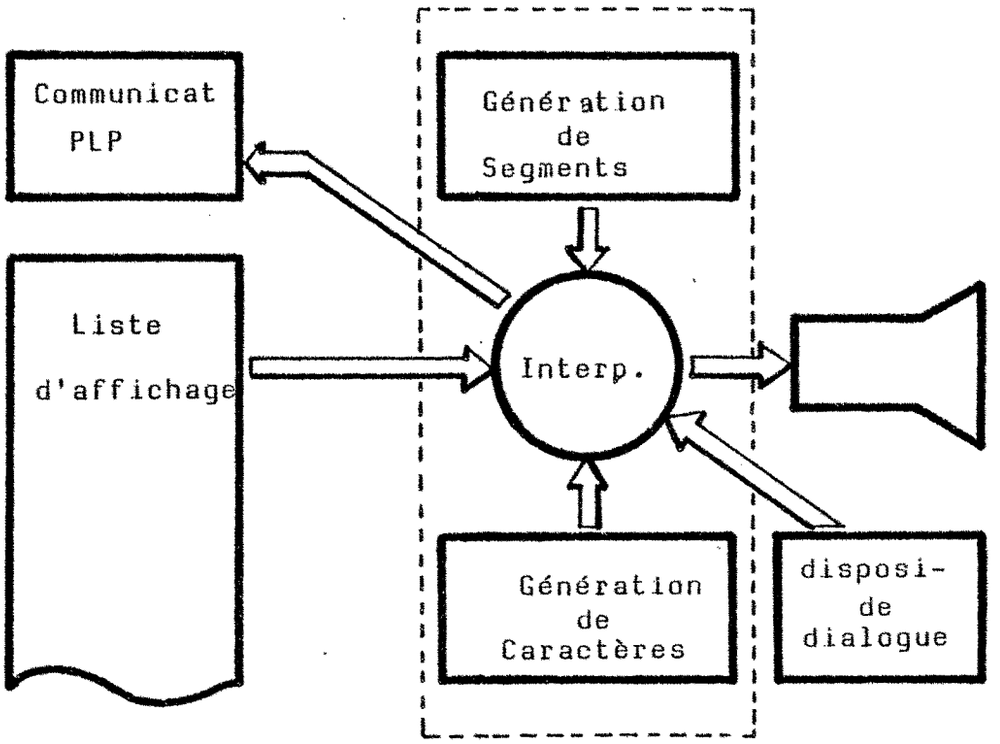


Fig. 5.2 Le PLA d'un terminal peu évolué.

Dans le cas d'un terminal très évolué, le PLA associé sera

conçu de telle façon qu'il va utiliser au maximum les fonctions offertes par le terminal. Par exemple, le PLA va coder, à partir de la liste d'affichage virtuelle, un programme (liste d'affichage réelle) qui sera rangé dans la mémoire d'entretien. Ce programme sera interprété par la "partie intelligente" du terminal afin de produire l'image sur l'écran.

Ainsi, le PLA peut être vu comme un mécanisme d'extension pour l'interprétation de la liste d'affichage virtuelle. La fig. 5.3 montre la place d'un PLA dans le cas d'un terminal évolué.

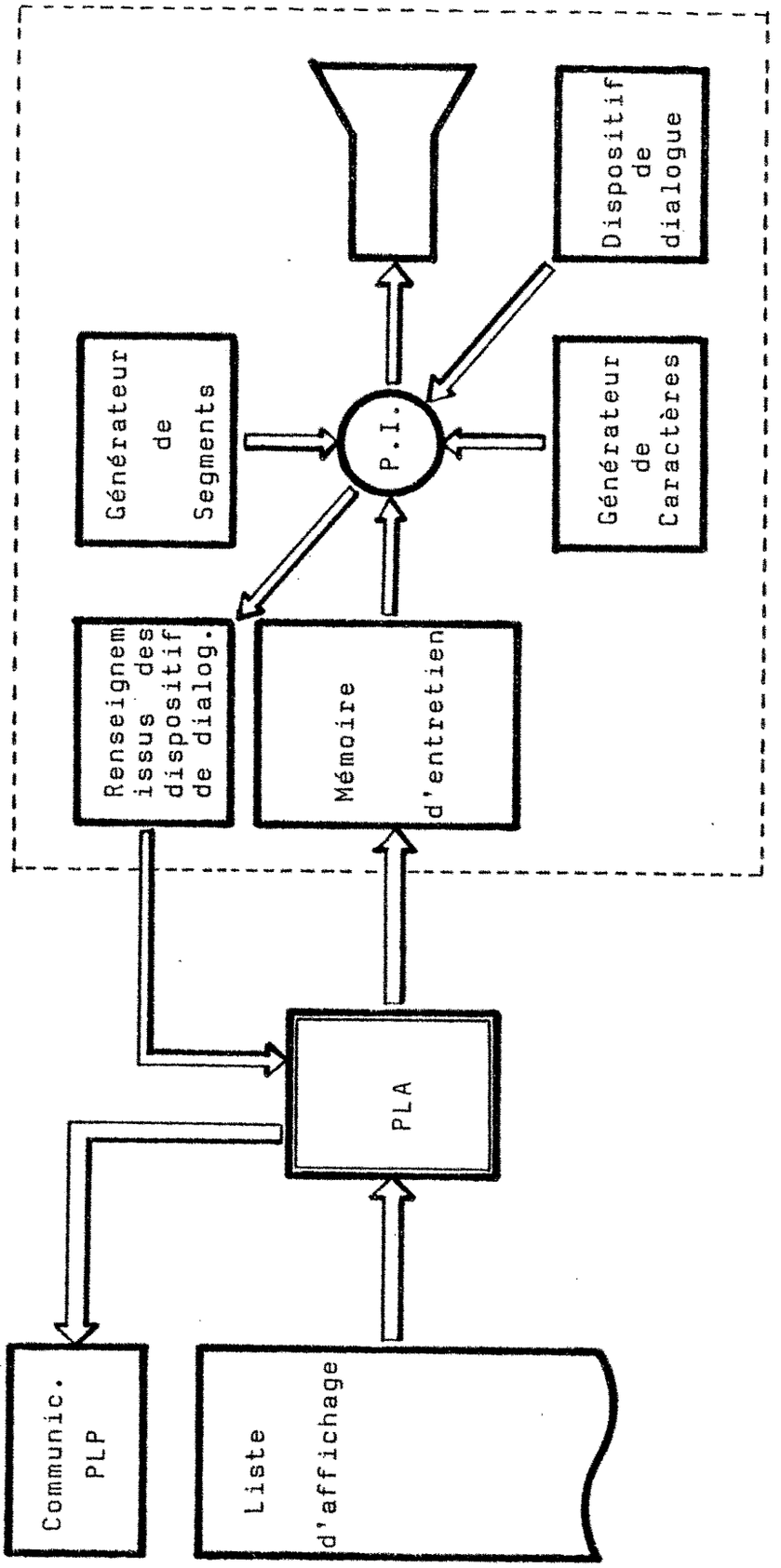


Fig. **5.3** Le PLA d'un terminal évolué.

3.- Contraintes de temps réel.

La notion de temps réel joue un rôle très important dans la conception de tout système graphique interactif puisqu'il faut tenir compte de la vitesse de réaction que l'on veut imposer à la chaîne informatique. En effet cette vitesse de réaction peut varier sensiblement selon qu'il s'agit de visualiser la distribution d'éléments électroniques sur une plaquette, ou de surveiller l'évolution d'un contrôle de processus industriel ou nucléaire.

Un des objectifs visés lors de la conception du SGI a été de répondre le plus vite possible aux demandes d'interactions. C'est pour cette raison que nous avons donné la plus haute priorité d'exécution aux processus du traitement de dialogue (TRAIDIAL dans le PLP et DIAL dans le PLA).

Au niveau de PLA nous proposons de respecter au maximum les caractéristiques des terminaux graphiques. Par exemple dans le cas du PLA pour le terminal DEC 30-N, l'interpréteur de la liste d'affichage va envoyer les coordonnées à la vitesse d'affichage du terminal. Pour éviter d'envoyer les coordonnées à une vitesse supérieure à celle du terminal, nous avons implémenté un mécanisme qui consiste à arrêter le microprocesseur pendant un certain temps, afin de pouvoir aficher tous les points calculés par les algorithmes de génération de segments et de caractères. Une représentation temporelle est montrée dans la fig 5.4. Dans ce cas, la cadence d'affichage est imposée par le terminal lui-même.

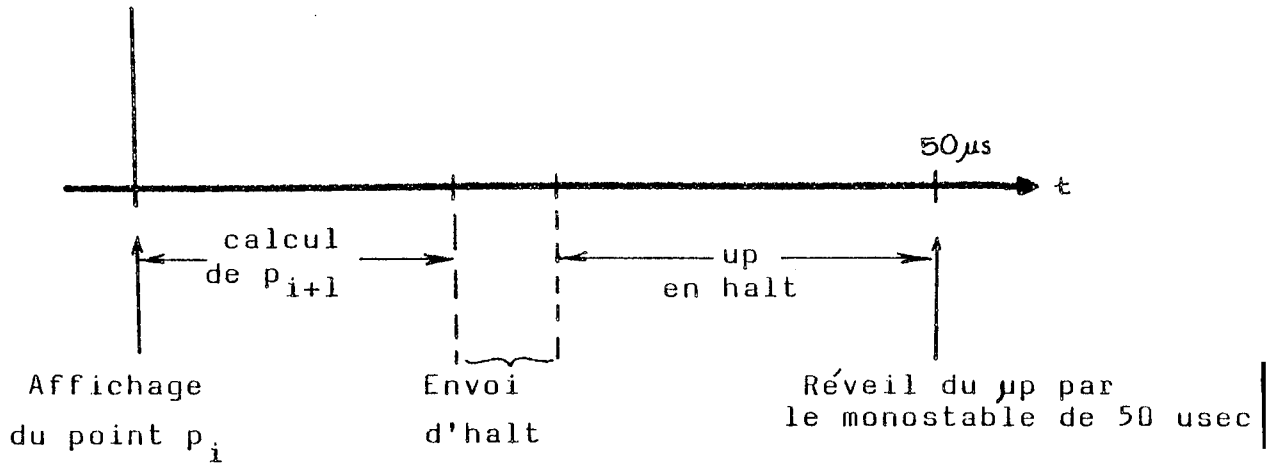


Fig. 5.4 Représentation temporelle du mécanisme d'arrêt pour le PLA du terminal DEC 30-N.

Dans le cas de terminaux très évolués, le PLA à concevoir va se contenter de remplir la mémoire d'entretien et de faire une analyse sémantique des informations issues des dispositifs de dialogue.

Il faut signaler que le grand développement de la technologie des circuits intégrés à grande échelle (LSI) et à très grande échelle (VLSI) a déjà permis la réalisation des organes specialisés dans le contrôle des unités de visualisation. Citons par exemple:

- Le MC6845 de Motorola,
- Le 8275 d'Intel,
- Le DP8350 de National Semiconductor,
- Le 5027 de Standard Mycrosystems Corp,
- Le SFF96365 et SFF96364 de EFCIS,
- Le VT-30 de DEC,

Ce dernier organe permet entre autres:

- une visualisation alphanumérique et graphique sur un écran à balayage ligne par ligne.
- huit couleurs différentes et une couleur de fond.
- clignotement réalisé par matériel.
- contrôle du curseur.
- 128 caractères définis par l'utilisateur.
- définir par logiciel la taille des caractères.
- rafraîchisement de l'écran au rythme de 50 images/seconde.

4.- Conception et réalisation du PLA pour le terminal 30-N de DEC.

Dans la suite nous allons décrire la conception et la réalisation d'un PLA adapté au terminal graphique 30-N de DEC.

Le terminal 30-N, est equipé d'un TRC qui permet un balayage de type cavalier. La définition informatique, c'est-à-dire le nombre de points affichables, est de 1024 x 1024. Il permet huit niveaux de luminosité, et la vitesse d'affichage est limitée à une fréquence de 20 Khz. C'est-à-dire un point tous les 50 usec.

L'inexistence dans ce terminal de mécanismes câblés pour la génération de vecteurs et de caractères, nous a ammené à concevoir une bibliothèque de programmes, afin de pallier ces défauts.

Les hypothèses qui nous ont guidé pendant la conception sont issues de:

- La façon dont le PLP doit structurer les données graphiques dans la liste d'affichage.

- Le mode de visualisation attaché à ces données.

- Les ordres envoyés par l'utilisateur pendant la phase de dialogue.

4.1 Le langage de commande du PLA.

Le niveau de départ de la conception du PLA consiste à définir d'abord le langage de commande de ce Processeur. Ce langage sera caractérisé par un ensemble de règles de syntaxe qui décriront le codage des instructions et le format des données à manipuler. La sémantique de chacune de ces instructions sera décrite dans V.4.2.

Le langage de commande du PLA sera constitué par trois familles d'instructions:

- Instructions graphiques pour l'affichage (point, segment,...).

- Instructions graphiques pour la définition des attributs (luminosité, texture, ...).

- Instructions de commande qui permettront de réaliser les interactions lors du dialogue.

Ces instructions seront reconnues et interprétées par le PLA afin de créer les différentes actions de commande qui permettront de visualiser sur l'écran le dessin codé dans la liste d'affichage virtuelle. Dans cette optique la liste n'est qu'un programme écrit dans le langage de commande du PLA.

La solution retenue dans I.6.3 consiste à structurer la liste

d'affichage virtuelle en accord avec les caractéristiques du terminal graphique, ceci afin de profiter au maximum des qualités de celui-ci. Le format de l'instruction graphique sera donc imposé par les caractéristiques du terminal graphique. Chaque mot de la liste d'affichage virtuelle sera interprété par le PLA d'une façon spécifique.

Le format d'une instruction graphique est en géneral composé de trois champs:

- code opération.
- attributs.
- opérandes.

Le code opération indique le type d'entité graphique dont il s'agit, par exemple, le code 00 peut correspondre à un point, 01 à un vecteur, etc. Le champ des attributs contient les renseignements correspondant au mode de visualisation attaché à l'entité, par exemple, la couleur, la luminosité, etc. Le champ d'opérandes contient les coordonnées ou autres paramètres attachés à l'entité, ce champ en général est de longueur variable, celle-ci étant précisée de deux façons: soit implicitement dans le code d'opération, soit explicitement par l'utilisation d'un mot qui indique le nombre d'opérandes ou en utilisant un délimiteur qui signale la fin des opérandes.

Le niveau de structuration de la liste d'affichage virtuelle pour le PLA du terminal 30-N de DEC, correspond à un niveau intermédiaire composé par les entités graphiques élémentaires suivantes: points, vecteurs et caractères. Les instructions qui permettront de coder la liste d'afichage, sont divisées en trois groupes:

- Instructions graphiques pour l'affichage des entités élémentaires.
- Instructions graphiques de contrôle pour la définition des attributs.

- Instructions de commande qui permettront de reconnaître la fin de la liste d'affichage virtuelle ou de sauter un ensemble d'entités lors de l'interprétation.

Dans le cas du terminal 30-N, la séparation des intructions graphiques pour l'affichage de celles pour la définition des attributs, a l'avantage de pouvoir associer les mêmes attributs à plusieurs niveaux de structuration. Par exemple avec une seule instruction de contrôle on peut définir les attributs d'une entité élémentaire, d'une section, d'une figure et même d'un dessin. Tout ceci se traduit en un gain de place mémoire dans la liste d'affichage virtuelle et de temps lors de l'interprétation des instructions d'affichage.

4.2 - Description des intructions du langage de commande du PLA.

4.2.1. Instructions graphiques pour l'affichage.

Ces instructions sont à opérandes de longueur fixe ou variable.

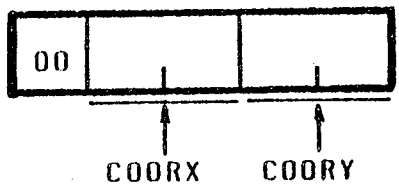
4.2.1.1 Instructions d'affichage à opérandes fixes.

Affichage d'un point:

Sémantique : Cette instruction permet de placer le faisceau allumé sur un point quelconque de la grille de base de l'écran.

Syntaxe : <AFFPOI> ::= <CODPOI><COORX><COORY>
<COORX> ::= 0/1/2...1023
<COORY> ::= 0/1/2...1023
<CODPOI> ::= 00

Format :



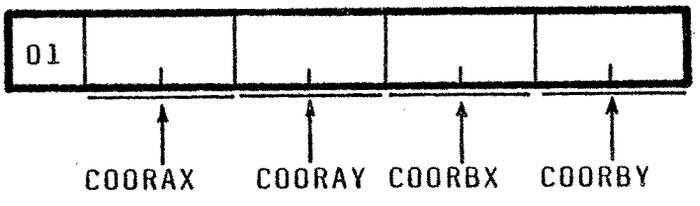
Affichage d'un segment de droite:

Sémantique : Cette instruction permet d'afficher un segment

de droite dont le point de départ est A(X,Y) et le point d'arrivée est B(X,Y).

Syntaxe : <AFFSEG> := <CODSEG><COORA><COORB>
<COORA> := <COORAX><COORAY>
<COORB> := <COORBX><COORBY>
<COORAX> := 0/1/2/3 ... 1023
<COORAY> := 0/1/2/3 ... 1023
<COORBX> := 0/1/2/3 ... 1023
<COORBY> := 0/1/2/3 ... 1023
<CODSEG> := 01

Format:

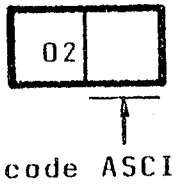


Affichage d'un caractère:

Sémantique : Cette instruction permet d'afficher un caractère sur une matrice de 12 x 8 points, codée dans un format ASCII. Le premier point de la matrice (celui le plus à gauche et en bàs) sera affiché là où se trouve placé le faisceau.

Syntaxe : <AFFCAR> := <CODCAR><CODASCII>
<CODCAR> := 02
<CODASCII> := 31/32/ .....

Format:



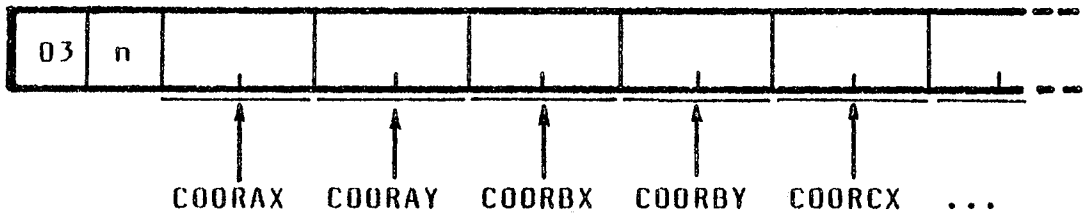
4.2.1.2 Instructions à opérandes variables:

Affichage de segments enchaînés:

Sémantique : Cette instruction permet d'afficher n segments enchaînés. L'origine d'un segment est confondue avec l'extrêmité du précédent.

Syntaxe : <AFFSEGENC> := <CODSGEN><n><COORA><COORB><COORC>...
<CODSGEN> ::= 03
<n> ::= 2/3/... 256

Format:

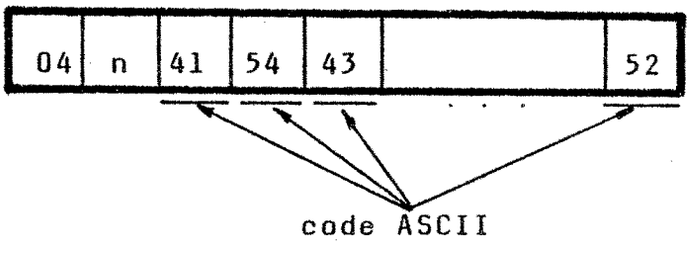


Affichage d'un texte.

Sémantèque : Cette instruction permet d'afficher une suite de de caractères. La distance entre caractères est définie à l'avance dans le code d'opération. C'est-à-dire que pour un code d'opération donné l'interpréteur va appeler le générateur de caractères correspondant.

Syntaxe : <TEXTE> ::= <CODTEX><n><CODASCCI>
<CODTEX> ::= 04

Format:



Ici, le code 04 correspond aux caractères codés dans une matrice de 12 x 8 points. La distance entre caractères est de 8 points.

Il faut indiquer que pendant l'interprétation de ces instructions d'affichage, le niveau de luminosité sera celui qui a été préalablement défini lors de l'interprétation de la dernière instruction de définition de cet attribut. Dans le cas des segments, ils seront affichés avec la texture qui a été définie aussi préalablement.

4.2.2 Instructions de contrôle.

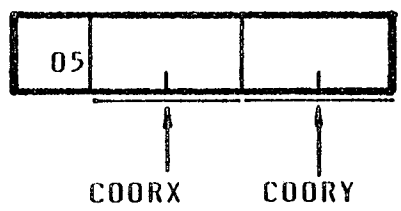
Ces instructions sont à opérande fixe et permettent de placer le faisceau éteint en un point de l'écran ainsi que de définir les attributs de la suite d'entités graphiques d'affichage qui se trouvent définies à la suite de celles-ci, dans la liste d'affichage.

Placement du faisceau éteint:

Sémantique : Cette instruction permet de placer le faisceu éteint sur un point quelconque de la grille de base de l'écran.

Syntaxe : <POSIT> ::= <CODPOS><COORX><COORY>
<CODPOS> ::= 05

Format:

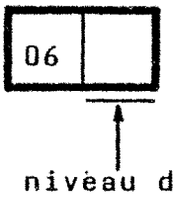


Définition de la luminosité d'affichage:

Sémantique : Avec cette instruction on définit le niveau de luminosité lors de l'affichage des entités graphiques rangées dans la liste d'affichage, après cette instruction.

Syntaxe : <LUMI> ::= <CODLUMI><NIVEAU>
<CODLUMI> ::= 06
<NIVEAU> ::= 0/1/2...7

Format:



Définition de la texture:

Sémantique : Cette instruction permet de définir la texture avec laquelle on va tracer une suite de segments de droite. Dans les paramètres de cette instruction on définit le nombre de points en continu que l'on veut afficher ainsi que le nombre de points en continu que l'on veut sauter.

Syntaxe : <TEXTURE> ::= <CODTXT><NPA><NPS>
<CODTXT> ::= 07
<NPA> ::= 0/1/2 ... 255
<NPS> ::= 0/1/2 ... 255

Format:

nombre de points en continu à sauter

nombre de points en continu à afficher

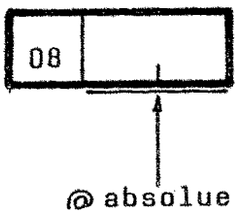
4.2.3 Instructions de commande.

Branchement à une instruction quelconque dans la liste d'affichage.

Sémantique : L'interprétation de cette instruction permet de sauter dans la liste d'affichage virtuelle la zone comprise entre la définition de cette instruction et l'adresse de l'instruction définie dans l'opérande. Donc il n'y aura pas d'interprétation pour les entités définies dans cette zone.

Syntaxe : <SAUT> ::= <CODST><ADRESSE>
<CODST> ::= 08
<ADRESSE> ::= 1000/1001/ ... 4095

Format:

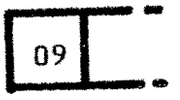


Fin de liste d'affichage.

Sémantique : Lorsque le code d'opération ne correspond pas à ceux définis jusqu'ici, on considère par défaut qu'il s'agit de la fin de la liste d'affichage virtuelle et l'interpréteur doit recommencer au début de la liste afin d'afficher une fois de plus l'image. C'est donc à l'interpréteur de raffraîchir l'image (cf. V.5.1).

Syntaxe : <FINLA> ::= <CODEFIN>
<CODEFIN> ::= 09/10/ ... 255

Format:



5.- Les processus de PLA du terminal 30-N de DEC.

Une fois défini le langage de commande du PLA, nous continuons la conception avec la réalisation de mécanismes pour l'interprétation de la liste d'affichage virtuelle ainsi que pour la transmision des informations issues des oospositifs de dialogue.

5.1 Le processus d'interprétation.

La création de ce processus sera faite lors de l'initialisation du Système. Ce processus est constitué par un ensemble d'instructions MC6800 qui, exécutées dans un ordre prédéfini sur un ensemble fini de données (liste d'affichage virtuelle), engendrent les commandes propres au contrôle du terminal 30-N, afin d'obtenir le dessin sur l'écran.

Ce processus pourra être interrompu par un processus plus prioritaire (cf. V.5.2).

Les procédures de ce processus sont en nombre de trois:

- Dans la première, on a l'algorithme d'interprétation proprement dit.

- La deuxième procédure contient l'algorithme pour la génération de segments de droite.

- Dans la troisième procédure on trouve l'algorithme pour la génération de caractères.

L'organigrame du processus de traduction est montré dans la

figure ci-après.

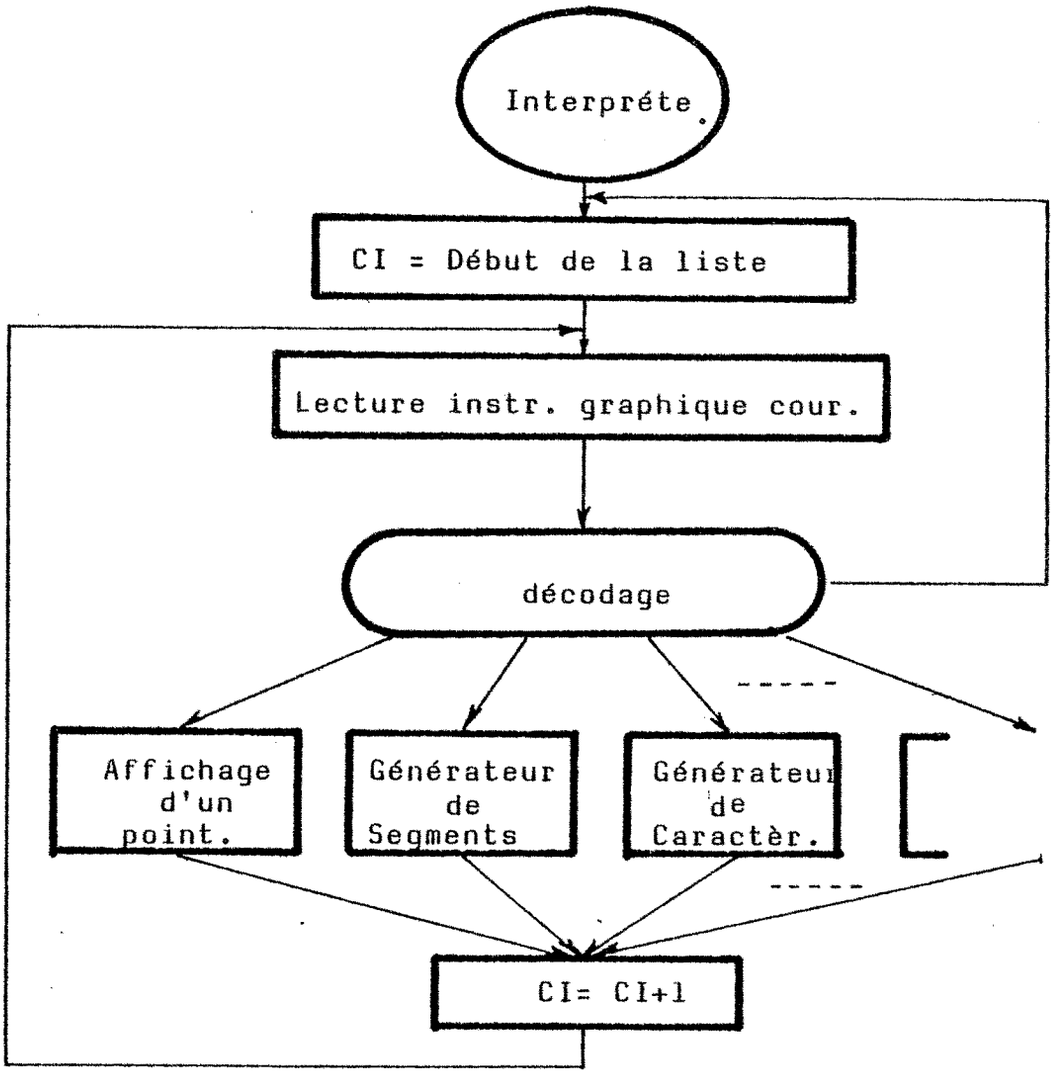


Fig. 5.5 Le processus d'interprétation de la liste d'affcihage.

On remarque que c'est à l'interpréteur de raffraîchir l'image.

5.1.1 - Analyse syntaxique de la liste d'affichage virtuelle.

L'interpréteur de la liste d'affichage virtuelle, fera une analyse syntaxique des instructions afin de les reconnaître et activer les procédures correspondant à l'action demandée.

Dans la réalisation de l'analyseur, le principe utilisé est celui d'un automate régulier dont les transitions d'un état à l'autre sont acompagnées d'une fonction sémantique.

Les états correspondent aux divers champs des instructions de la liste. Le passage d'un état à l'autre est fonction de l'état précédent et de la donnée lue.

Les fonctions sémantiques correspondent à des actions simples telles que la lecture d'une donnée, incrémentation d'un compteur, ... etc.

Pour un état donné, il n'est permis de trouver qu'un type de donnée, par exemple lorsqu'on va afficher un caractère, le mot à lire après le code d'opération doit correspondre à un code ASCII.

5.1.1.2 Graphe d'analyse syntaxique.

Dans la figure 5.6 nous montrons le graphe d'analyse. Dans ce graphe les chiffres entre parenthèses donnent la fonction sémantique determinée par l'état actuel et par la donnée lue.

Etats:

1 : Etat initial
2 : Lecture d'un code d'opération.

3 : Lecture des coordonnées du point.
4 : Lecture de paramètres du segment.
5 : Lecture du code ASCII du caractère.
6 : Lecture du nombre de segments à afficher.
7 : Lecture de nouveaux paramètres.
8 : Lecture du nombre de caractères à afficher.
9 : Lecture du code ASCII du caractère courant.
10 : Lecture de coordonnées du placement.
11 : Lecture de la valeur du niveau de luminosité.
12 : Lecture du nombre de points à afficher en continu et lecture du nombre de points à ne pas afficher.

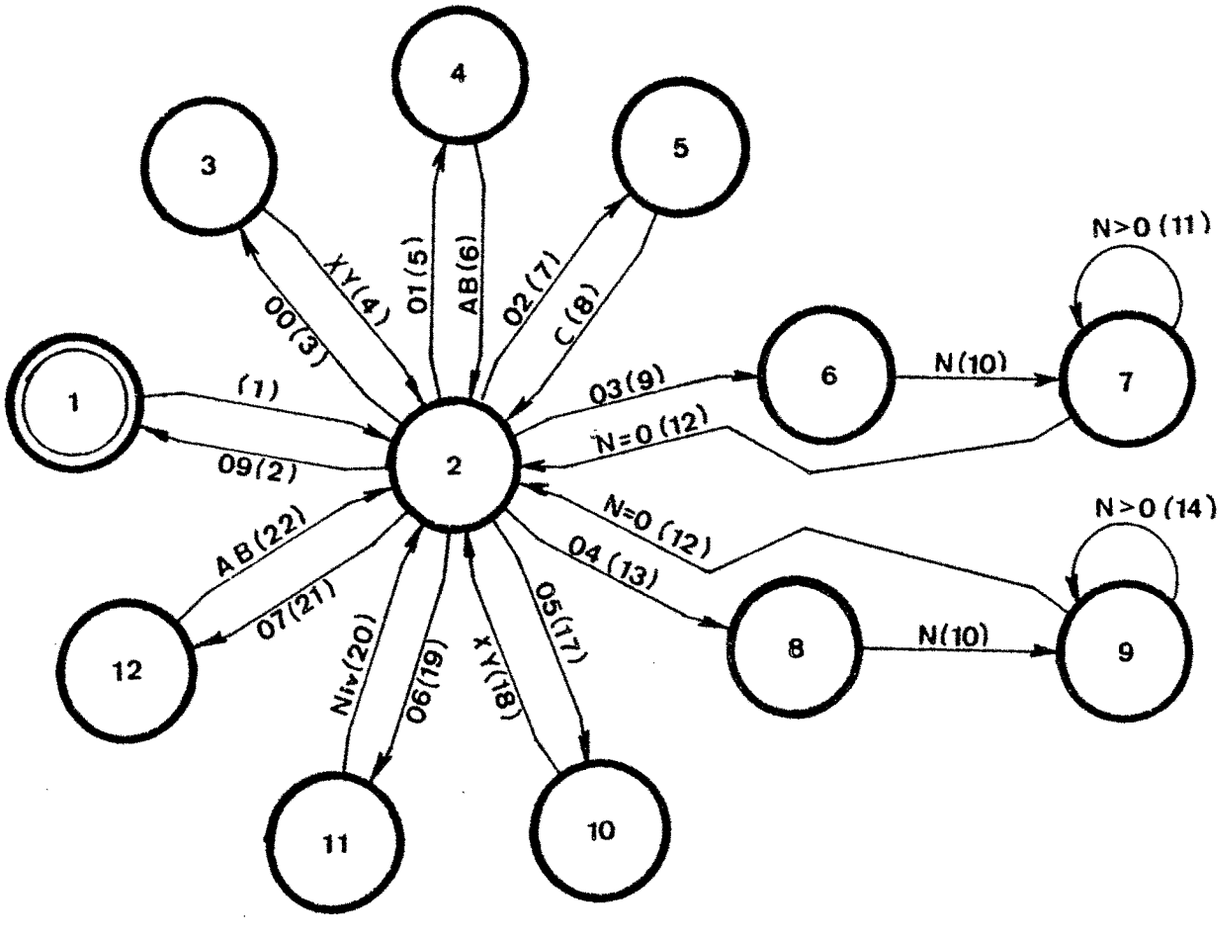


Fig. 5.6 Graphe d'analyse pour la liste d'affichage virtuelle.

Fonctions sémantiques:

(1) : On initialise le compteur ordinal CI de l'instruction graphique, et le pointeur PT de la donnée graphique courante.

(2) : Il s'agit de la fin de liste.

(3) : Il s'agit d'afficher un point. PT := PT+1;

(4) : Chargement des tampons X et Y du 30-N avec la valeur trouvée dans le champ correspondant de l'instruction graphique. Affichage d'un point. CI:= CI+1; PT:=PT+1;

(5) : Il s'agit d'afficher un segment. PT:=PT+1 ;

(6) : Chargement des variables de travail avec la valeur lue dans le champ opérandes. Appel de la procédure pour afficher un segment. CI:=CI+1 ; PT:=PT+4 ;

(7) : Il s'agit d'afficher un caractère. PT:=PT+1 ;

(8) : Chargement de la variable de travail avec le code ASCII lue. Appel de la procédure pour afficher un caractère. PT:=PT+1; CI:=CI+1;

(9) : Il s'agit d'afficher des segments enchaînés. PT:=PT+1;

(10) : Chargement du compteur d'entités à afficher. n:=LA(PT); PT:=PT+1 ;

(11) : Tantque n>0 faire :
Chargement des variables de travail avec les paramètres du segment en cours à afficher; appel de la procédure pour afficher un segment; PT:=PT+4; n:=n-1;

(12) : Mise à jour de pointeurs. PT:=PT+1; CI:=CI+1;

(13) : Il s'agit d'afficher un texte. n:=LA(PT); PT:=PT+1;

(14) : Tantque n>0 faire:
Chargement du code ASCII du caractère à afficher; n:= n-1; Pt:=PT+1;

(15) : Il s'agit de placer le faisceau éteint. PT:=PT+1;

(16) : Chargement des tampons X et Y. PT:=PT+1; CI:=CI+1;

(17) : Il s'agit de définir la luminosité. PT:=PT+1;

(18) : Chargement du registre d'intensité. PT:=PT+1; CI:=CI+1;

(19) : Il s'agit de changer la texture du tracé. PT:=PT+1;

(20) : Chargement de variables de travail qui permettent la définition du nombre de points continus à afficher ainsi que le nombre de points en continu à sauter. PT:=PT+1; CI:=CI+1;

5.2 - Le processus de dialogue DIAL.

La tâche de ce processus consiste simplement à ranger dans la zone de communication avec le PLP, l'adresse de l'instruction de la liste d'affichage qui était en cours d'exécution. Cette adresse est le contenu du compteur d'instructions CI.

Ce processus est plus prioritaire que le processus d'interprétation. Sa création est faite lorsque l'on positionne la ligne IRQ provoquée par la mise en service du dispositif de dialogue (photostyle). Pour l'instant nous positionons la ligne IRQ à l'aide d'un interrupteur astable.

Pour se synchroniser avec le procesus TRAIDIAL du PLP, DIAL va utiliser le mécanisme de "poignée de main" (hand shaking) déjà décrit dans IV.6.1.

Dans la fig. 5.7 nous montrons l'organigramme du processus DIAL.

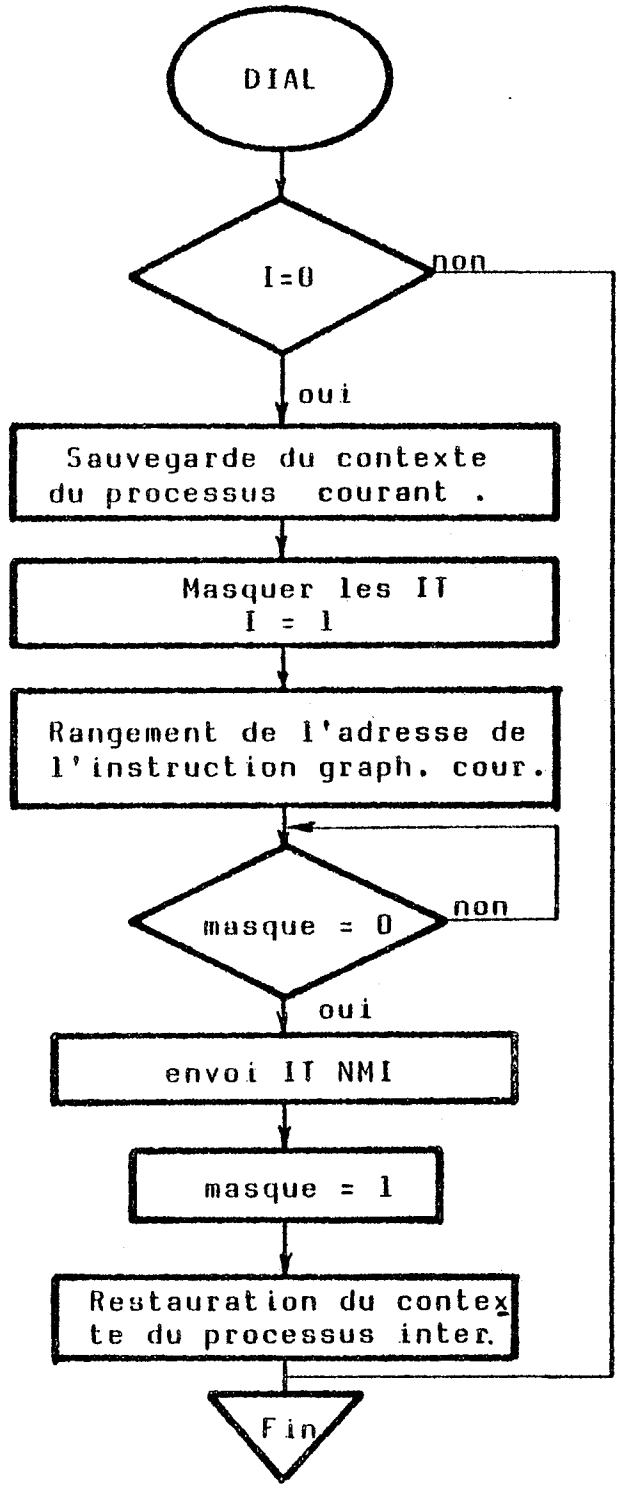


Fig. 5.7 Le processus DIAL du PLA du terminal 30-N de DEC.

6. - Architecture du PLA du terminal 30-N de DEC.

Dans l'implémentation du PLA du terminal 30-N de DEC, nous avons une architecture constituée par les organes suivants:

- Un microprocessuer MC6800

- Un organe spécialisé en les entrées/sorties (PIA) pour commander le terminal 30-N.

- Organes de mémorisation statique (ROM) pour y ranger les procédures de l'interpréteur ainsi que de DIAL.

- Organes de mémorisation dynamique pour y ranger les variables de travail des processus et les informations destinées au PLP pendant le dialogue.

- Des translateurs de niveaux logiques, parce que les niveaux du terminal 30-N sont de 0 et -3 volts, tandis que le microprocesseur travaille avec des niveaux TTL.

6.1 - Répartition de l'espace mémoire.

L'ensemble de procédures du PLA, ainsi que leurs variables de travail ont été répartis dans l'espace mémoire du microprocesseur comme l'indique la figure 5.12.

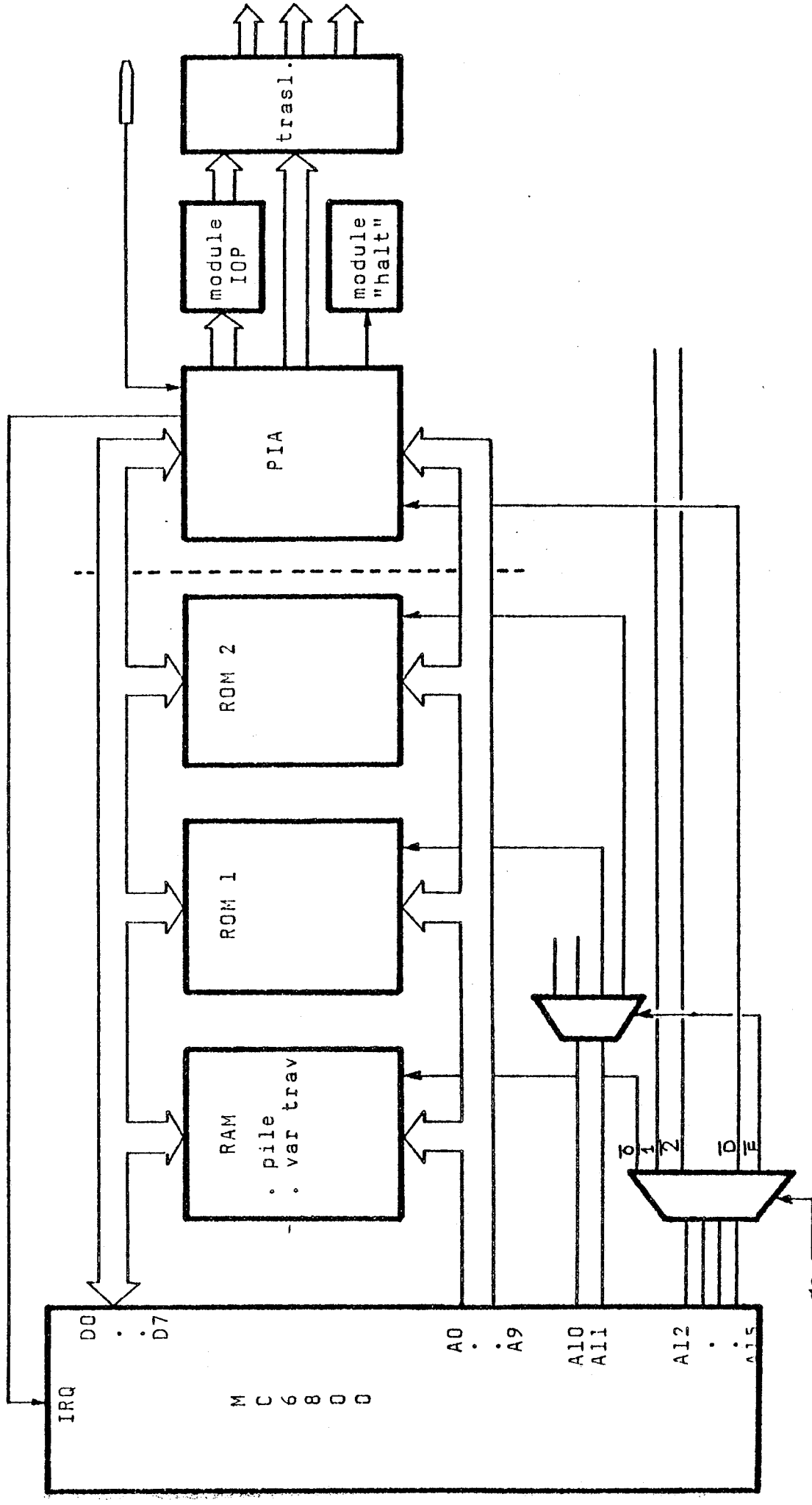


Fig. 5.8 Architecture du PLA du 30 N de DEC

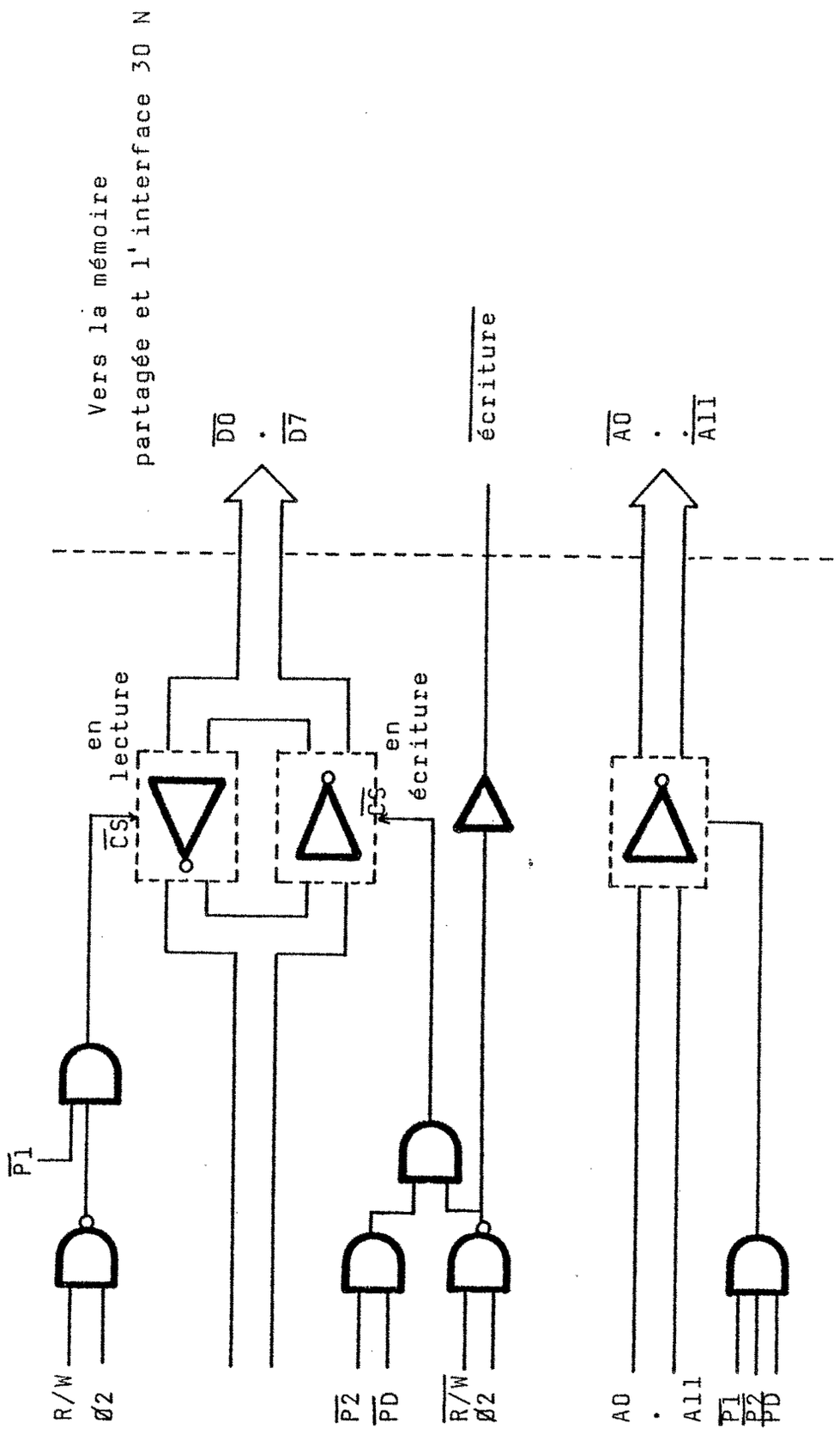


Fig. 5.9 Interface avec l'environnement extérieur

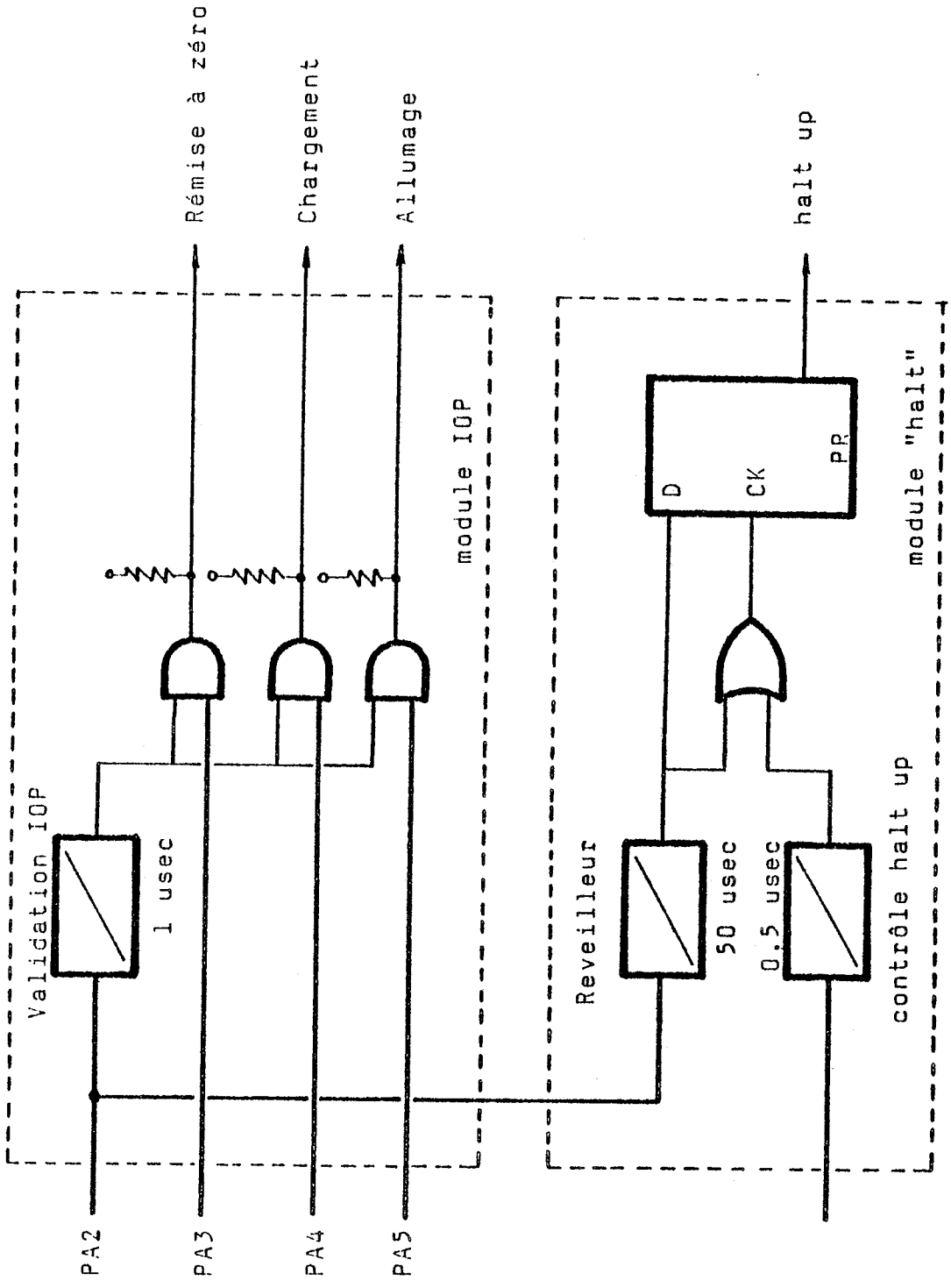


Fig. 5.10 Les module "IOP" et "halt".

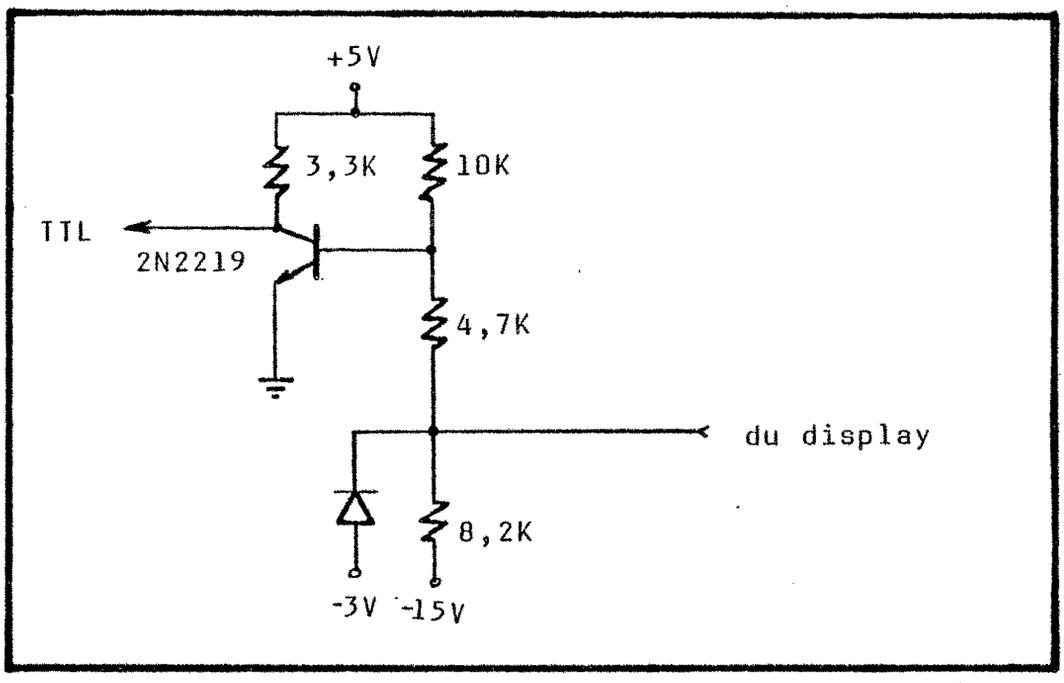


Fig. 5.11a_Translateur en réception.

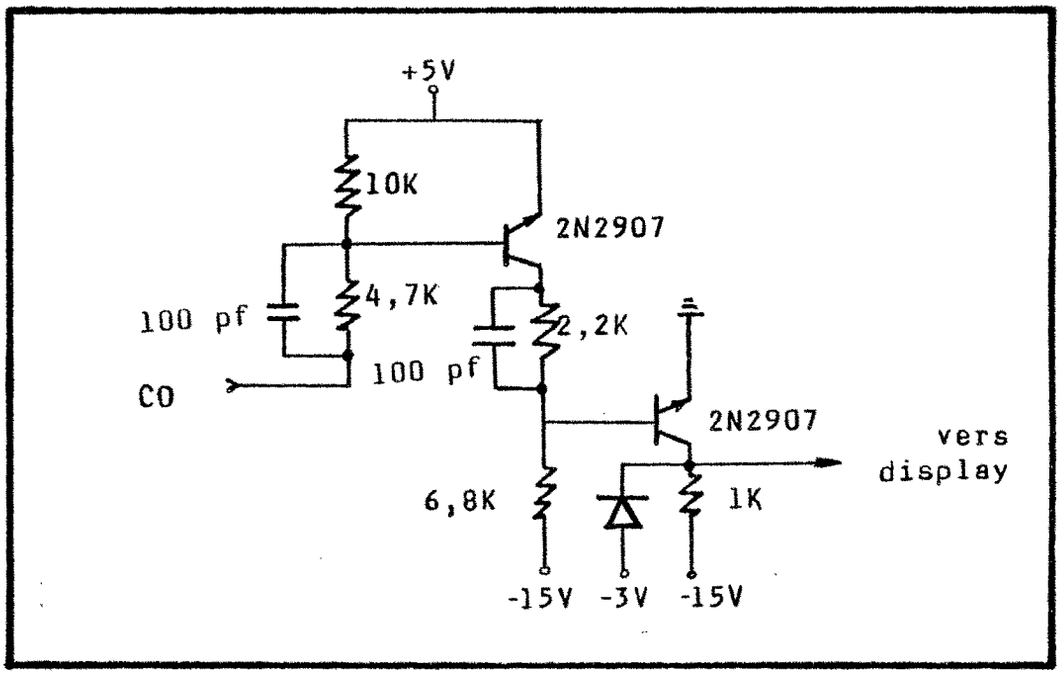


Fig. 5.11b _ Translateur en émission.

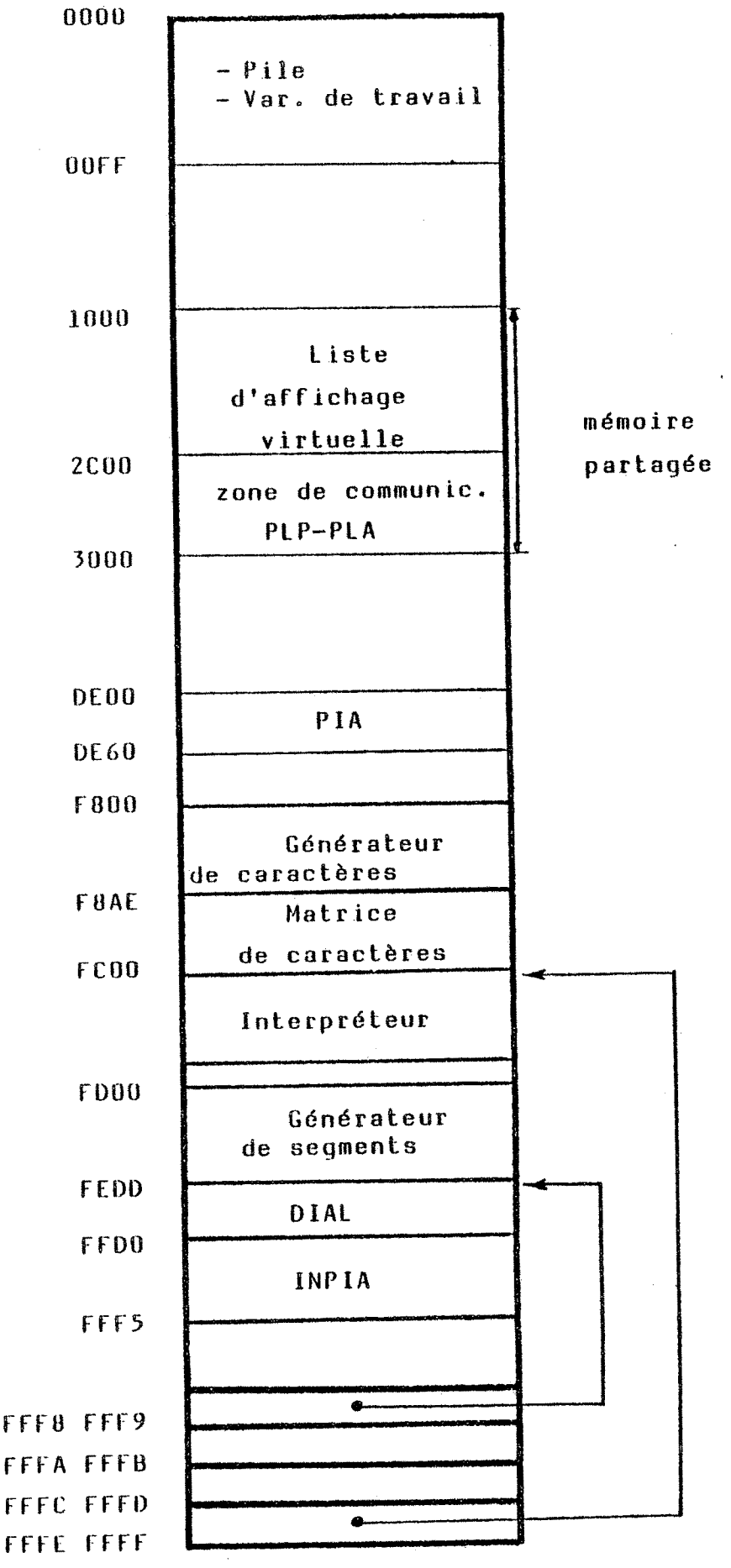


Fig. 5.12 Organisation de la mémoire

7.- Evaluation de performances du PLA du 30-N de DEC.

Dans la suite nous ne donnerons que des chiffres concernant deux paramètres de performance: la place mémoire occupée et le temps d'exécution des processus.

- Place mémoire.

Les chiffres correspondant à ce paramètre sont montrés ci-après.

Procédure INTER : 241 octets.
Procédure GENSEG : 455 octets.
Procédure GENCAR : 183 octets.
Procédure DIAL : 12 octets.

- Temps d'exécution.

Il faut rappeler que la cadence d'affichage permise par le terminal 30-N est de 50 µsec/point. Ceci veut dire que si l'on veut afficher une image à une fréquence de 25 images/seconde afin d'éviter le phénomène de clignotement, l'image sera constituée au plus par 800 points. Compte tenu de la résolution de l'écran qui est de 0.1 mm à peu près, nous ne pourrons afficher plus de 8 cm. sans avoir le phenomène de clignotement!!!.

Lorsqu'on utilise l'algorithme programmé pour la génération de segments de droite, il suffit d'exécuter autour de 25 instructions pour calculer les coordonnées écran du prochain point à afficher. Ceci veut dire que le générateur de segments, va fournir les coordonnées des points à la cadence d'affichage permise par le terminal.

Par contre lorsqu'on va générer un caractère, il suffit d'exécuter moins de dix instructions pour calculer les coordonnées du point suivant à afficher. Dans ce cas là nous allons utiliser le mécanisme de halte prévu pour arrêter le microprocesseur et donner ainsi le temps nécessaire au terminal pour qu'il puisse afficher le point (voir fig. 5.4).

Dans le cas du terminal 30-N, la liste d'afichage virtuelle joue le rôle de mémoire d'entretien de telle sorte que l'interpréteur, à chaque fois qu'il arrive à la fin de cette liste, doit revenir au début de celle-ci afin de la parcourir une autre fois et raffraîchir ainsi l'image sur l'écran.

Il est bien évident qu'au fur et à mesure qu'augmente le nombre d'entités graphiques dans la liste d'affichage virtuelle, le temps mis par l'interpréteur pour raffraîchir l'image, augmente également.

A titre de conclusion, ce que nous pouvons dire, que dans le cas de l'utilisation d'un matériel avec mémoire d'entretien qui permet une fréquence d'affichage plus élevée, l'interpréteur pourra remplir la mémoire d'entretien (création de la liste d'affcihage réelle) en quelques centaines de millisecondes.

CONCLUSION

Bien que nous ne donnions dans ce document que quelques chiffres de performances du Système Graphique Interactif, nous pouvons dire que nous avons atteint le but souhaité.

Il faut d'abord signaler que la méthode de conception suivie nous a permis d'utiliser le langage GRIGRI, développé dans l'Equipe de Techniques Graphiques de l'IMAG, comme langage de commande du Processeur Fonctionnel Graphique.

Une étude des caractéristiques et propiétés des fonctions que doit réaliser le Système Graphique Interactif a été faite afin de chercher le parallélisme lors de l'exécution ainsi que la meilleure répartition de ces fonctions dans le contexte du Système CORAIL.

La modularité de l'architecture fonctionnellement répartie proposée pour l'implémentation du Système Graphique Interatif permet déjà entre autres:

- D'offrir aux utilisateurs des langages évolués ainsi que des logiciels de modélisation (voir sous-systèmes) bien adaptés pour chaque classe de modélisation.

- D'apporter une capacité de s'adapter aux différents contextes d'utilisation. C'est-à-dire qu'ajouter un nouveau sous-système de modélisation pour traiter une application non envisagée au départ ne posera aucun problème. Reciproquement, changer ou ajouter une console graphique n'affectera pas les sous-systèmes de modélisation puisque seuls les niveaux inférieurs du Système Graphique seront modifiés.

Il faut signaler qu'un des récents efforts en Techniques Graphiques est orienté vers la recherche de standardisation des langages de mise en page. Si dans le futur on souhaitait utiliser

ce langage standardisé, celui-ci devrait être implémenté comme langage de commande du Processeur Fonctionnel Graphique (voir Processeur Logique de Préparation).

L'état actuel de la réalisation du notre Système Graphique Interactif au niveau de la Modélisation, ne comporte que quelques séquences de programme destinées à la création de polygones ouverts ainsi que pour la définition de textes.

Au niveau de la Préparation, nous n'avons implémenté que les primitives GRIGRI concernant le cas où les paramètres nfig, nsec, ... sont supérieurs ou égaux à zéro.

Au niveau de l'Affichage, nous avons implémenté tous les mécanismes matériels et logiciels nécessaires à l'exploitation du terminal graphique 30-N de DEC. Des algorithmes programmés pour la génération de segment de droite ainsi que pour la génération de caractères alphanumériques ont été faits.

A titre de conclusion, ce travail contribue à la clarification des fonctions constituant le Système Graphique Interactif et donne une implémentation de ce Système. De par ce découpage fonctionnel, l'architecture qui en découle est à base de microprocesseurs fonctionnellement répartis. La réalisation à base de microprocesseurs montre l'intérêt de cette architecture du point de vue coût et des performances par rapport à une réalisation classique

Tenant compte de grands devéloppements futurs de la technologie en circuits intégrés, nous pouvons prévoir que dans un proche avenir on pourra construire des organes spécialisés dans l'interprétation directe du langage standardisé de mise en page.

BIBLIOGRAPHIE

(ADA-78) G. ADAMS and T. ROLANDER
Design Motivations for Multiple Processor Microcomputer Systems.
Computer Design
Mars 1978

(ALL-77) D. R. ALLISON
A Design philosophy for Microcomputer architectures.
Computer
Fevrier 1977

(ANC-74) F. ANCEAU
Contribution à l'étude des systèmes hiérachisés de ressources dans l'architecture de machines informatiques.
Thèse d'état
Grenoble, le 5 décembre 1974

(ANC-78) F. ANCEAU
Pricipes et mise en oeuvre de microprocesseurs.
E.N.S.I.M.A.G.
Janvier 1978

(AND-75) G. A. ANDERSON and E. D. JENSEN
Computer interconnection structures: taxonomy, characteristics and examples.
Computing Surveys vol. 7 no. 4
Décembre 1975

(ARN-76) R. G. ARNOLD & E. W. PAGE
A hierarchical, restructurable multimicro-processor architecture.
Computer Annual Symposium
AFIPS 1976

(AZE-75) J. AZEMA
Grammaires de graphes, algorithmes d'analyse, Applications.
Thèse de 3ème. cycle.
Grenoble mars 1975

(BAE-74) J. L. BAER
Models for the design, simulation, and performance of Distributed-function architecture.
Computer
Mars 1974

(BAS-77) M. BASIL
A microprocessor-based refreshing buffer for storage tube graphics terminals.
Computer & Graphics vol. 1
Pergamon Press, 1977

(BEB-75) J. BEBB
An interactive environment for scientific development.
Computer & Graphics vol. 1
Pergamon Press, 1975

(BLI-76) J. F. BLINN
The internal design of the IG routines, and interactive graphics system for a large time-sharing environment.
Computer Graphics
SIGGRAPH ACM 1978

(BRA-75) J. C. BRAID
The synthesis of solid Bounded by many faces.
CACM vol 18, no. 4
Avril 1975

(CAJ-74) H. D. CAPLENER and J. A. JANKV
Top-down approach to LSI system design.
Computer Design
AoOt 1974

(CAP-77) O. CAPRANI, H. JENSEN & U. OUGAARD
Microprocessors conected to a common memory.
Euromicro 1977

(CAR-77) B. J. CAREY
Formal methods expedite microprocessor-based system design.
Fevrier 1977

(CDD-77) L.C. CRUTHERES, J. VAN DEN BOS, A. VAN DAM
A device-independent general purpose graphic system for stand-alone and satellite graphics.
Computer Graphics
SIGGRAPH ACM 1977

(COH-75) D. COHEN and E. TAFT
An interactive network graphics system.
Computer & Graphics vol. 1
Pergamon Press, 1975

(COT-75) I. W. COTTON
Standards for network graphics communications.
Computer & Graphics vol. 1
Pergamon Press, 1975

(CRT-67) CRT
Precision display tube 30-N
Digital Equipment Corporation
1967

(CYP-78) R. J. CYPSER
Communication architecture for distributed systems
Addison-Wesley
1978

(DEN-75) E. DENDERT, J. ERNST and WETZEL
GRAPHEX-68 Graphical Language features in Algol 68.
Computer & Graphics, vol. 1
Pergamon Press, 1975

(EAR-74) C. J. EARL
Inovations in heterogeneous and homogeneous distributed-function architectures.
Computer
Mars 1974

(EAS-77) J. F. EASTMAN & D. R. WOOTEN
A general purpose, expandable processor for real-time computer graphics.
Computer & Graphics, vol. 1
Pergamon Press, 1977

(FAR-74) D. J. FARBER
Sotware considerations in distributed-architecture.
Computer
Mars 1974

(FIC-76) M. L. FICHTENBAUN
Top-down design streamlines digital system projects.
Computer Design
Septembre 1976

(FOL-73) J. D. FOLEY
Software for satellite graphics systems.
Proceedings of the IEEE, 1973

(FOL-74) J. D. FOLEY and V. L. WALLACE
The art of natural Graphic man-machine conversation.
Proceedings of the IEEE, vol. 62, no. 4
Avril 1974

(FOL-76) J. D. FOLEY
A tutorial on satellite Graphics Systems.
Computer
Août 1976

(FOR-75) W. S. FORD and V. C. HAMACHER
Low level architecture features for supporting process communications
The Computer Journal.
Août 1975

(FRE-74) H. FREEMAN
Computer processing of line-drawings images.
ACM Computing Surveys vol. 6, no. 1
Mars 1974

(GAM-73) R. C. GAMMIL and D. ROBERTSON
Graphics and interactive system, design considerations of a software system.
National Computer Conference AFIPS 1973.

(GRA-75) M. T. GRAY
Microprocessors in CRT terminal applications: Hardware/Software tradeoffs.
Computer
Octobre 1975

(GSP-77) GSPC State-of-the-art.
State-of-the-art of graphics software packages.
Computer Graphics
SIGGRAPH ACM
Automne 1977

(GUE-76) R. A. GUEDJ
Report on the IFIP W. G. 5.2 Workshop on "Methodology in Computer Graphics".
Avant-projet
Juillet 1976

(GUI-78) S. GUIBOUD-RIBAUD
Architecture de Systemès. Une vue synthétique: Logicielle et Matérielle.
Ecole d'été du Forez
Juillet 1978

(HAA-79) B. HAAS
Single chip video controler.
Byte publications Inc.
Mai 1979

(HAM-76) G. HAMLIN Jr.
Configurable applications for satellite graphics.
Computer Graphics
SIGGRAPH ACM 1976

(HAM-78) G. HAMLIN Jr.
A Microprocessor-assisted graphics system.
Computer Graphics
SIGGRAPH ACM 1978

(HAN-72) B. HANSEN
Structured multiprogramming.
Communications of the ACM 15
AoÛt 1972

(HAN-78) B. HANSEN
Distributed process: A concurrent programming concept.
Communications of the ACM vol. 21, no. 11
Novembre 1978

(HOA-74) C. A. R. HOARE
Monitors: An operating system structurating concept.
Communications of the ACM
Octobre 1974

(HUB-78) R. H. HUBBOLD & P. J BRAMHALL
A flexible, high performance interactive graphics system.
Computer Graphics
SIGGRAPH ACM 1978

(HUS-78) S. S. HUSSON
System development methodology
Proceedings Euromicro
1978

(JER-75) J. C. JERVERT and R. M. DUNN
Multi-console intelligent satellite graphics.
Computer & Graphics vol. 1
Pergamon press, 1975

(KEL-76) R. G. KELLNER & L. D. MAAS
A developmental system for microcomputer based intelligent graphics terminals.
Computer Graphics
SIGGRAPH ACM 1976

(KLI-77) E. E. KLINGMAN
Comparisons and trends in microprocessors architecture.
Computer Design
Septembre 1977

(KOR-77) G. A. KORN
A distributed computer system for laboratory automation.
Computer Design
Juin 1977

(KUN-75) L. KUNII, T. AMANO, H. ARISAWA & S. OKADA
An interactive fashion design system "INFADS".
Computer & Graphics vol. 1
Pergamon Press, 1975

(LAF-75) P. LAFORGUE
Construction de sous-systèmes utilisant une machine abstraite. Réalisation autour du noyeau GEMEAU.
Thèse de Docteur-Ingénieur.
Grenoble, 1er. fevrier 1975

(LAV-77) S.H. LAVINGTON, G. THOMAS & D.B.G. EDWARDS
The MU5 Multicomputer Communication System
IEEE Transactions on Computer
Vol. C-26 No. 1
Janvier 1977

(LED-77) A. LEDUC-LEBALLEUER
Conception et réalisation d'un logicielle graphique de base, indépendant de son contexte. Application au logicielle GRIGRI.
Thèse de Docteur-Ingénieur
Grenoble 30 septembre 1977

(LIS-77) A. M. LISTER and P. J. SAYER
Hierarchical Monitors.
Software-Practice and experience, vol. 7
1977

(LUC-77) M. LUCAS
Contribution à l'étude des techniques de communication graphique avec ordinateur. Eléments de base de logiciels graphiques interactifs.
Thèse d'état
Grenoble, le 16 de décembre 1977

(MAC-78) C. MACHGEELS
IOTA: A non-interactive graphics system using the pseudo-machine concept.
Computer & Graphics vol.3
Pergamon press, 1978

(McK-78) M. S. McKENDRI
The use of monitors in microprocessors software development.
Euromicro journal 4, 1978

(McK-78) K. McKENZI
The evolution of peripheral devices in microprocessor system.
Computer
Juin 1978

(MAR-79) M. MARINESCU
Contribution au problème de la mutuelle exclusion élémentaire et répartie.
R.R. No. 153 ENSIMAG
Février 1979

(MAR-77) N. MAROVAC & W. S. ELLIOT
A network-oriented language. A new approach to network design, using interactive graphics.
Computer & Graphics vol. 2
Pergamon Pres, 1977

(MAT-78) Ph. MATHERAT
A chip for low-cost raster-scan graphic display.
Computer Graphics
SIGGRAPH ACM 1978

(MIC-76) J. MICHEL, A. VAN DAM
Experience with distributed processing on a host/satellite graphics system.
Computer Graphics
SIGGRAPH ACM 1976

(MIF-78) J. C. MICHENER and J. D. FOLEY
Some major issues in the design of the core graphics system.
ACM Computing Surveys vol. 10, no. 4
Décembre 1978

(MIV-78) J. C. MICHENER and A. VAN DAM
A functional overview of the core system with glosary.
ACM Computing Surveys vol.10, no. 4
Décembre 1978

(MOL-76) P. MORVAN M. LUCAS
Images et Ordinateurs. Introduction à l'infographie interactive.
Larouse, collection Sciences Humaines et Sociales, Série informatique.
Septembre 1976

(MUE-77) D. J. MUELLER
Microcomputers decentralize processing in data communications network.
Computer Design
Octobre 1977

(MUN-78) T. MUNTEAN
Spécification de la synchronisation par contraintes.
Thèse Docteur-Ingénieur
Grenoble le 19 juin 1978

(NES-74) W. M. NEWMAN and R. F. SPROULL
An approach to graphics system design.
Proceedings of the IEEE, vol. 62, no. 4
Avril 1974

(NEV-78) W. M. NEWMAN and A. VAN DAM
Recents efforts toward graphics standardization.
ACM Computing Surveys vol. 10, no. 4
Décembre 1974

(NIL-74) R. N. NILSEN
Distributed-function computer arhitectures.
Computer
Mars 1974

(ORG-78) E. ORGANICK
New directions in computer systems architecture.
Proceedings Euromicro 1978

(PAR-72) D. L. PARNAS
Information distribution aspects of the design methodology.
Information Processing 72
North-Holland Publishing Company
1972

(PAR-74) D. L. PARNAS
On a "buzzword" hierarchical structure.
Information Processing 74
North-Holland Publishing Company
1974

(POS-77) J. L. POSDAMER
Concurrent processing for computer graphics.
Computer & Graphics vol. 2
Pergamon Press, 1977

(POS-77) J. L. POSDAMER
Some criteria for the evolution of graphics implementation languages.
Computer & Graphics vol. 2
Pergamon Press, 1977

(POU-76) G. H. POUJOULAT
Machine CORAIL: Présentation générale des éléments du système et exemples d'appliation.
Rapport de recherche no. 53
IMAG novembre 1976

(RIE-75) J. E. RIEBER & A. C. SHAW
Interactive picture generation and manipulation trought formal descriptions.
Computer & Graphics, vol 1
Pergamon Press, 1975

(RUS-77) P. M. RUSSO
Interprocessor communication for multi-microcomputer systems.
Computer
Avril 1977

(SAT-78) S. G. SATTERFIELD, F. RODRIGUEZ & d. ROGERS
A microprocessor display controller for combining refresh and storage tube graphics.
Computer Graphics
SIGGRAPH ACM 1978

(SCH-77) J. P. SCHOELLKOPF
Machine PASC-HLL: Définition d'une architecture pipe-line pour une unité centrale adaptée au langage PASCAL.
Thèse de 3ème cycle
Grenoble le 28 juin 1977.

(SEA-75) B. C. SEARLE and D. E. FREBERG
Tutorial: Microprocessor applications in multiple processor systems.
Computer
Octobre 1975

(SYS-77) Systems reference and data sheets.
Motorola Semiconductor Products Inc.
1977

(TAN-76) A. S. TANENBAUM
Structured Computer Organization
Prentice-Hall Englewood Cliffs
N. J. 1976

(THE-72) F. THERON
Sur le programme EUCLID: création, manipulation et visualisation de formes tridimensionnelles dans un langage géométrique à génération dynamique.
Thèse 3ème cycle, Paris 1972.

(VAS-73) A. VAN DAM and G. M. STABLER
Intelligent satellites for interactive graphics.
National Computer Conference
AFIPS 1973

(WEI-77) S. J. WEISSBERGER
Analysis of multiple-processor system architectures.
Computer Design
Juin 1977

(WIL-72) W. T. WILNER
Burroughs B1700
Burroughs Corportion, Santa Barbara Plant
Goleta, California.
Juin 1972

(ZUR-68) F. W. ZURCHER and B. RANDELL
Iterative multi-level modeling a methodology for computer system design.
Proceedings of the AFIP Congress.
1968.

ANNEXE I

LE SYSTEME CORAIL

1 - Généralités.
2 - Les différents types de Processeurs.
3 - Connexions matérielles entre Processeurs.
4 - Architecture logicielle.
5 - Architecture matérielle.

I.- LE SYSTEME REPARTI CORAIL.

1.- Généalités.

Le Système CORAIL est un Système multi-microproceseur distribué fonctionnellement. Il est conçu pour être totalement modulaire au niveau des processeurs et du système d'exploitation. Il se présente comme une fédération de processeurs totalement indépendants (figure I.1)

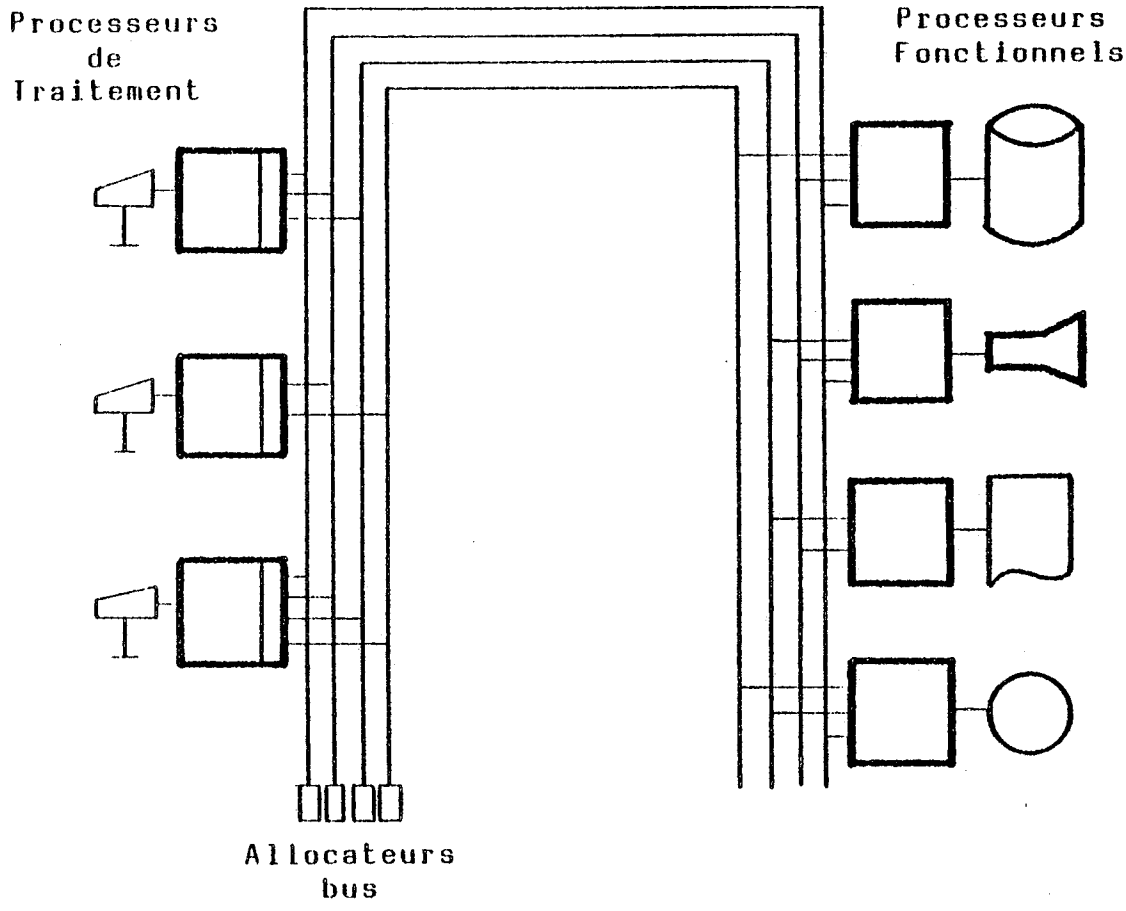


Fig. I.1 - Structure générale du Système CORAIL

2.- Les différents types de Processeurs.

Dans CORAIL on distingue deux types de Processeurs: Les Processeurs de Traitement et les Processeurs Fonctionnels.

2.1 - Les Processeurs de Traitement.

Ces processeurs sont spécialisés dans l'exécution du traitement propre à l'utilisateur. Chaque poste de travail (télé-imprimante, clavier de commande, visualisation, machine outil, ...), peut être associé à un utilisateur et comporte un Processeur destiné à effectuer le travail qu'il sollicite.

Ces processeurs doivent dialoguer avec l'utilisateur et gérer les requêtes destinées aux autres Processeurs du Système.

Ils exécutent des tâches propres à leur poste. Leur puissance est donc déterminée par la complexité de la tâche à effectuer et la vitesse de communication avec l'utilisateur humain à partir de son poste de travail.

Pendant une séance, le Processeur de Traitement aura besoin de fonctions supplémentaires, par exemple, l'accès à une base de données. Dans ce cas-là, il utilisera les Processeurs Fonctionnels appropriés en échangeant des messages avec eux.

2.2 - Les Processeurs Fonctionnels.

Ce sont des Processeurs spécialisés dans l'exécution d'une seule fonction, correspondant à une ressource au niveau du Système. Par exemple le Processeur Fonctionnel Graphique ou le Processeur base de données.

Une grande partie des Processeurs fonctionnels disposent d'un système d'exploitation monoprogrammé. C'est-à-dire qu'ils n'exécutent qu'une action à la fois pour la compte d'un Processeur de Traitement. Cependant, certains Processeurs comme le Processeur de Stockage ou le Processeur Fonctionnel Graphique sont exploités en mode multiprogrammé.

3.- Connexions matérielles entre Processeurs.

Le Système CORAIL dispose de bus parallèles dont chacun comporte 14 lignes (8 lignes pour l'échange de données, plus 6 lignes de contrôle) pour la communication entre les Processeurs. La version actuelle dispose de 4 de ces bus.

L'allocation pour chacun de ces bus est faite par un allocateur centralisé (matériel) qui demande cycliquement (pooling) à tous les Processeurs qui sont attachés à ce bus, s'il désirent l'utiliser.

En général, pour toute configuration, il y a au moins un bus commun à tous les Processeurs du Système et selon les différents débits demandés pour l'échange d'information, les Processeurs Fonctionnels peuvent disposer de bus privés ou partagés selon leur utilisation.

Les Processeurs du Système communiquent par échange de messages. Ces échanges se font en mode synchrone et on arrive à des cadences de 24 usec/octet en utilisant des microprocesseurs MC6800 de Motorola.

Lors de la communication entre Processeurs, l'envoi d'un message par un Processeur expéditeur, provoque le génération et l'envoi d'un "message écho" par le Processeur destinataire afin d'avertir son interlocuteur de la bonne ou mauvaise réception du

message.

4.- Architecture logicielle d'un Processeur.

L'indépendence totale entre les Processeurs du Système CORAIL permet d'avoir un système d'exploitation distribué sur les différents Processeurs.

Le découpage logiciel d'un Processeur comporte les niveaux suivants:

- Communication avec les autres Processeurs.
- Organisation des travaux.
- Gestion de la ressource ou interprétation des requêtes des utilisateurs, selon qu'il s'agit d'un Processeur Fonctionnel ou d'un Processeur de Traitement.

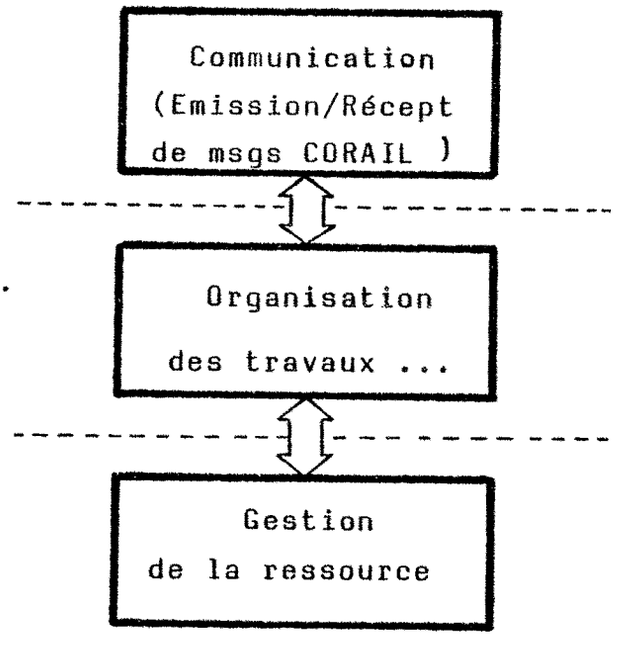


Fig. I.2 - Découpage logiciel dans un Processeur de CORAIL.

Dans ce découpage, le premier niveau est spécifique au protocole physique de communication du Système et il est commun à tous les Processeurs.

Dans le deuxième niveau il y a les fonctions corespondant à la vérification des messages, mise à jour de tables de communication, ... ainsi que des fonctions particulières relatives au rôle du Processeur dans le Système.

Le troisième niveau comporte l'ensemble des mécanismes logiciels qui permettront d'une part la gestion de la Ressource Système, et d'autre part de contrôler le flux de l'information afin d'éviter la saturation des ressources de ce Processeur telles que tampons, files d'attente, ... etc.

4.1 - Le Superviseur de communication SUPCOM.

Les deux premiers niveaux du logiciel d'un Processeur sont implémentés sous la forme d'un superviseur de communication, celui-ci comporte les deux modules suivants:

- Module d'émission/réception de messages.
- Module de contrôle qui a deux tâches principales:
  + Vérification de messages.
  + Fonction d'allocation proprement dit.

4.2 - Les messages CORAIL.

Dans CORAIL il y a quatre sortes de messages pour la communication entre les Processeurs et pour la réalisation des différentes fonctions nécessaires à son fonctionnement.

1).- Les messages de réservation de ressources: demande de réservation d'une ressource, acceptation ou refus de réservation.

2).- Les messages entre Processeurs de Traitement: les Processeurs de Traitement ne communiquent que pour discuter de la réservation d'une ressource; les messages sont alors: demande de cession d'une ressource, acceptation ou refus.

3).- Les messages d'exécution utilisés pour la réalisation d'un travail, les principaux messages sont:

. messages généraux: occupation ou libération de ressources, échanges élémentaires.

. macrorequêtes: ce sont les messages utilisés pour le passage de l'information correspondant au graphe de travail demandé.

4).- Les messages d'erreurs, utilisés pour signaler les différentes erreurs qui peuvent se présenter, par exemple réception d'un mauvais code d'un message.

Un message CORAIL est constitué par un descripteur et un corps. Le descripteur définit:

. Le numéro de l'expéditeur,
. Le numéro du destinataire,
. Le code de la commande,

. Le numéro de la conversation,
. Le numéro de phrase dans la conversation,
. La longueur des données.

Le corps est constitué:

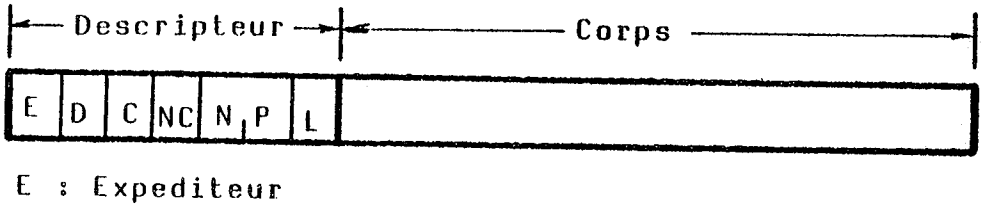
. soit de données transmises (texte, paramètres,...)
. soit d'une liste de paramètres.

La zone contenant les données peut aller jusqu'à 256 octets.

Il y a 4 commandes principales:

. Réservation d'une Ressource: RES
. Lecture de données: LEC
. Ecriture de données: ECR
. Fermeture de la conversation: FER

Le format d'un message CORAIL est montré dans la fig I.3



E : Expediteur
D : Destinataire
C : Code
NC: Numéro de Conversation
NP: Numéro de phrase
L : Longueur

Fig. I.3 Le format d'un message CORAIL

5.- Architecture matérielle d'un Processeur.

Les Processeurs du Système sont construits à partir des organes de la famille Motorola 6800. L'architecture matérielle d'un Processeur est généralement simple et très classique. Tous les Processeurs possèdent un CPU (MC6800), des interfaces pour la communication avec les autres Processeurs (PIA MC6820), une mémoire de travail RAM (Intel 2114), et une mémoire de programme ROM (Intel 2708).

Les Processeurs de Traitement disposent d'un interface avec une téléimprimante (ACIA MC6850, plus l'électronique associée) et les autres types de Processeurs disposent des interfaces correspondant aux dispositifs qui leur sont attachés.

La dimension d'un processeur est d'une à trois cartes électroniques de taille moyenne, selon la mémoire demandée et les circuits d'interface nécessaires.

ANNEXE II

LES PRIMITIVES GRIGRI

1 - Primitives de définition de systèmes de coordonnées.
2 - Primitives de déclaration de données.
3 - Primitives pour l'interprétation graphiques de données.
4 - Primitives pour l'affichage.
5 - Les messages à l'opérateur.
6 - Les primitives de dialogue.

LES PRIMITIVES GRIGRI

1.- Primitives de définition de systèmes de coordonnées.

FENETRE (nfen, igx, igy, sdx, sdy)
CLOTURE (nclo, igx, igy, sdx, sdy)

Ces deux primitives permettent de définir un espace de départ (la fenêtre) et un espace d'arrivée (la clôture). Les numéros (nfen, nclo) servent d'identificateurs et permettent d'utiliser la même fenêtre (clôture) pour des ensembles de données différents. Les coordonnées des coins de la fenêtre sont exprimées dans le système choisi par l'utilisateur. Par contre, les coordonnées des coins de la clôture sont exprimées dans un système de coordonnées arbitarire, permettant de définir la portion d'écran choisie pour l'affichage, indépendamment des systèmes de coordonnées réels. Le Processeur Logique de Préparation (PLP) assure automatiquement la transformation qui affiche le contenue de la fenêtre de la meilleure manière possible dans la clôture.

2 - Primitives de déclaration de données.

POINTS (nfig, tablong, tabx, taby, tablegende)
TEXTES (nfig, tablong, tabtextes, tabx, taby)

Ce deux primitives permettent de prévenir au PLP du type des données qu'il aura à manipuler et de définir la strcture employée.

La primitive POINTS permet de transmettre une suite de coordonnées, suite contenue dans le tableaux à une dimension tabx

et taby. La longueur de chaque section est définie grâce au tableau tablong. Le tableau tablegende correspond à la notion de marquage d'une position. Chaque élément de tablegende contient un groupe de caractères (par exemple 4) qui est destiné à marquer la première position de la section correspondante. Ce tableau permet ainsi d'étiqueter (si necéssaire) les différentes sections d'une figure. Les sections sont repérées au niveau du PLP par leur numéro d'ordre dans le tableau de donées. Par contre, le programmeur de l'application peut attribuer à chaque section un numéro d'identification (corrélation), permettant de la différencier des autres. Les deux niveaux de dénomination correspondent donc:

- Globalment, aux données repérées par le même numéro de figure.
- Au niveau en dessous, aux différentes sections d'une même figure.

Il est intéresant de constater la parfaite symétrie de la primitive TEXTES. Le texte rangé dans le tableau tabtextes est découpé en sections dont la longueur est spécifiée dans tablong. A chaque section peut être associé un couple de coordonnées permettant de spécifier le point de départ du texte. L'omission de ce couple permet de lier un texte au texte de la section précédente.

3.- Primitives pour l'interprétation graphique de données.

MODE (nfig, ncor, nsec, descripteur)

Cette primitive permet d'associer à une section (nsec) un certain nombre de qualifications qui permettront d'obtenir le tracé la représentant. Citons par exemple, pour les points: la luminosité, la couleur, la texture, traites pleins, pointillés, tiretés, etc.; pour les textes: la couleur, la taille, etc... Ici on considère que le numéro de corrélation (ncor) est une caractéristique comme une autre, ce qui explique qu'il soit

attribué à travers la primitive MODE.

4.- Primitives pour l'affichage.

AFICHER (nfig, ncor, nfen, nclo)
EFFACER (nfig, ncor)

La primitive AFFICHER permet d'obtenir sur l'écran la représentation des ensembles de données qui ont été déclarés lors de l'attribution du numéro de figure. Les systèmes de coordonnées sont définis par reférence à une fenêtre et une clôture. Si l'on ne désire pas afficher tout la figure, on peut se servir du numéro de corrélation passé en paramètre. Pour simplifier certains écritures, on propose la convention suivante.

	nfig	ncorrel	nsection
nul	ts les figs.	ts les sect.	ts les sect.
positif	la figure nfig	les sec. corr.ncor.	la section nsec.
négatif	ts fig. sauf nfig	ts sect sauf ncor.	ts secs sauf nsec

L'affichage se fait dès l'invocation de la primitive AFFICHER. La notion d'affichage différé n'existe pas. Par contre on peut noter, que l'on peut construire tout une figure avant de l'afficher, quelque soit le mode de tracé de chaque section. Ce fait est extrêmement intéressant du point de vue du programmeur d'application, car il permet de conserver la logique de construction des objets à représenter, san se soucier des problèmes de tracé proprement dit.

En ce qui concerne la primitive EFFACER, il y a peut de chose à dire, sinon que la combinaison des numéros de figure et de corrélation permet d'obtenir des effacements sélectifs au niveau

de chaque section (ou d'un ensemle de sections de même corrélation).

5.- Les messages à l'opérateur.

5.1 - La description d'un message.

MESSAGE (nmess, descripteur, nclot, x, y)
EFMESS (nmess)

Un message est constitué d'une chaîne de caractères portant des zones de textes et de zones réservées. L'emplacement du message sur l'écran est défini par la donée d'une clôture (nclot) et la position du début du texte dans cette clôture (x,y). Chaque message est repéré par un numéro (nmess). Lors de l'invocation de la primitive MESSAGE, le texte spécifié est affiché. Les zones réservées sont vides lors de la première invocation. Elles contiennent ensuite des valeurs qui ont été soit éditées (voir 5.2), soit introduites lors d'une phase de dialogue (voir 6.1).

5.2 - L'éditon de données alphanumériques.

EDENTIER (nmess, nzone, entier)
EDRREL (nmess, nzone, réel)
EDCHAINE (nmess, nzone, chaîne)

Les zones réservées à l'intérieur d'un message (voir 5.1) servent à l'identification de valeurs alphanumériques de type entier, réel ou chaîne de caractères. L'édition d'une variable se fait en sélectant une zone réservée (nzone) d'un message donné (nmess). Le contenu de la variable est alors affiché dans l'emplacement choisi. Cet emplacement ne sera plus modifié jusqu'à la rencontre d'un ordre d'effacement du message, ou une nouvelle édition avec les mêmes paramètres.

6.- Les primitives de dialogue.

Ici on écarte délibérément la définition de primitives faisant appel explicitement à tel ou tel dispositif de communication, c'est-à-dire qu'on utilise la notion de dispositif logique qui permet de définir les outils de dialogue en termes fonctionnels, sans se préoccuper de leur réalisation effective. Il y a quatre actions de base:

- Introduction d'une valeur alphanumérique,
- Introduction d'un couple de coordonnées,
- Identification d'un élément du dessin,
- Sélection d'une action parmi n proposées (menu).

6.1 - Introduction de données alphanumériques.

RVAL (nb, nmess, tréel)
NVAL (nb, nmess, tentier)
CVAL (nb, nmess, tchaîne)

L'introduction de valeurs alphanumériques se fait par l'intermédiaire des messages. A chaque invocation d'une fonction d'introduction de données, une zone du message spécifié (nmess) est selecté. L'opérateur tape la donnée sur le clavier alphanumérique, et celle-ci est transmise, en fin d'action, au programme d'application par l'intermédiaire d'un tableau (tréel, tentier, tchaîne).

Le nombre de répétitions (nb) est indiqué de la manière suivante: Si nb=0 alors le nombre est inconnu et une indication de fin d'action sera donnée par l'opérateur. Le nombre de données relevé sera alors transmis au retour. Si nb>0 alors le nombre d'entrées est connue à l'avance. L'arrêt se fait soit lors d'une action de "fin" soit lorsque le nombre maximum

est atteint.

6.2 - Collecte de coordonnées.

POSITION (nb, nfig, nfren, nclot)

Le nombre de coordonnées spécifié (nb) est relevé par tout dispositif de communication utilisable et rangé dans les tableaux tabx et taby indiqués lors de la déclaration de données correspondant à nfig. Le tableau des longueurs de section est égalment mis à jour. La donée de la clôture (nclot) et de la fenêtre (nfen) permet de relever les points dans l'espace écran souhaité et de restituer des coordonnées exprimées dans le système utilisateur.

6.3 - Identification.

IDENTIFIER (nb, nfig, tfig, tcorr)

Cette primitive permet d'identifier une suite d'éléments d'un dessin. L'identification (nfig,ncorr) est rangée dans les tableaux tfig, et tcorr. La convention donnée en 4 permet de spécifier quelle figure ou quelle ensemble de figures peut être designé grâce à nfig.

6.4 - Menu

MENU (nb, liste, tretour)

La liste descriptive du menu est formée d'une suite de nombres, chacun d'entre eux pouvant être etiqueté. Le résultat de l'action est une suite de nb nombres, suite rangée dans le tableau tretour. Les valeurs sont introduites par l'un quelconque des moyens de communication disponibles. Tous les nombres portant une étiquette pourront être obtenus par le biais d'un menu affiché sur l'écran et constitué de la liste des étiquettes.

ANNEXE III

RAPPORT DE REALISATION

1 - Introduction.
2 - La maquette.
3 - Le processeur de Préparation à la visualisation.
4 - Le processeur d'Affichage.
5 - Expériences réalisées.

RAPPORT DE REALISATION
DU SYSTEME GRAPHIQUE INTERACTIF

1.- Introduction.

La mise au point de la réalisation du Système Graphique Interactif a été faite en deux étapes:

- La première étape a consistée à faire tourner le Système en mode local, c'est-à-dire indépendamment du Système CORAIL.

- Dans la deuxième étape nous nous sommes branchés (mode périphérique) comme un Processer Fonctionnel du Système CORAIL.

Dans la première étape nous avons implémenté et testé un noyau minimun des fonctions à réaliser par le Système Graphique Interactif.

En ce qui concerne la dernière étape nous sommes arrivés à établir la communication avec le(s) Processeur(s) de Traitement de CORAIL. Il reste encore à implémenter un logiciel de mise en forme ou de modélisation.

2.- La maquette.

La maquette du Processeur Fonctionnel Graphique est contenue dans un chasis standard 19 pouces. Le chasis comporte:

- des entretoises mécaniques,

- les alimentations de 5V-7A, 12V-0.5A et -12V-0.5A,

- trois cartes standard No. 2 de l'Equipe de Recherche en Architecture des Ordinateurs. Deux de ces cartes concretisent le Processeurs Fonctionnel Graphique tandis que la troisième contient l'interface de communication avec le terminal graphique 30-N de DEC.

La face avant de la maquette contient le bouton astable pour l'initialisation du Processeur Fonctionnel Graphique.

La communication entre le Processeur de Prépatation et le Processeur d'Affichage est réalisée par un bus commun matérialisé sur un bus banalisé de fond de pannier auquel sont reliées les cartes au moyen des connecteurs de 2x49 broches. Chacune des trois cartes contient une quarentaine de circuits intégrés à grande échelle plus quelques composants discrets. Les connexions entre les circuits sont effectuées au moyen de la technique de "wire wrapping".

3.- Le Processeur de Préparation.

3.1 Le Logiciel.

Le logiciel du Processeur de Préparation a été écrit en langage assembleur MC6800 et nous n'avons réalisé et mis au point que les procédures concernant le traitement des primitives GRIGRI pour la définition et la déclaration de dessins dont les paramètres nfig, nsec, ncor, ... etc, sont supérieurs ou égaux à zéro. En ce qui concerne les primitives de dialogue, nous sommes en train d'en faire la mise au point.

3.2 Le matériel.

La partie matérielle du Processeur de Préparation est bâtie sur une carte standard No. 2. Cette carte a été réalisée et testée et elle comporte:

- un microprocesseur Motorola MC6800.

- une unité arithmétique AMD9511

- une horloge quartz qui fourni Ø1 et Ø2 (MC6871)

- 4 Koctets de mémoire morte (4 ROM 2708) dont:
  +1 Koctet contient un superviseur pour la mise au point du logiciel et du matériel.
  +1 Koctet contient le superviseur de communication avec le reste du Système CORAIL.
  +2 Koctet contiennent la programmation des primitives GRIGRI.

- 383 octets de mémoire vive statique (3 RAM MC6810) dont:
  + 256 octets contiennent les variables de travail des processus du PLP et du superviseur de communication CORAIL.
  + 127 octets contiennent les variables de travail du superviseur de mise au point.

- 8 Koctets de mémoire vive statique (16 RAM 2114) réparties de la manière suivante:
  + 4 Koctets qui contiennent les tampons, file d'attente, ... etc, ainsi que le fichier GRIGRI.

  + 4 Koctets qui constituent la mémoire partagée qui va contenir la liste d'affichage virtuelle.

- Un interface imprimante pour dialoguer avec l'utilisateur

lors de la mise au point ou lorsqu'on travail en mode local. Cet interface comporte un organe d'entrée/sortie serie/parallèle (ACIA MC6850) plus des portes logiques ainsi que des éléments discrets.

- Un interface CORAIL comportant des portes logiques ainsi qu'un organe d'entrée/sortie parallèle/parallèle (PIA MC6820).

- des amplificateurs trois états pour les données, adresses et quelques lignes de contrôle.

4.- Le Processeur d'Affichage.

4.1 Le logiciel.

Le logiciel du Processeur d'Affichage que nous avons réalisé et mis au point a été écrit en langage assembleur MC6800. Ce logiciel corresponde à la programmation des algorithmes pour:

- l'interpréteur de la liste d'affichage virtuelle,
- le traitement du dialogue,
- la génération de segments de driote,
- la génération de caractères alphanumériques.

4.2 Le matériel.

La partie matérielle du Processeur d'Affichage a été bâtie sur un carte standard No. 2. Cette carte comporte les éléments suivants:

- un microprocesseur Motorola MC6800.

- 2 Koctets de mémoire morte (2 ROM 2708) dont:
  + 1 Koctet contient les algorithmes d'interprétation et génération de segments et caractères.
  + 1 Koctet contient le matrice de 12x8 points des caractères alphanumériques.

- 127 octets de mémoire vive statique (RAM 6810) qui contient la pile et les variables de travail de l'interpréteur.

- des amplificateurs en trois états pour les lignes des données, des adresses et de contrôle.

4.3 - L'interface avec le terminal 30-N de DEC.

L'interface avec le 30-N a été aussi bâtie sur une carte standard No. 2, il comporte:

- un organe d'entrée/sortie parallèle (PIA 6820),

- des translateurs de niveau de TTL à niveaux de 0 et -3V (et vice-versa)

- un connecteur, placé à l'arrière, de 2x49 broches pour la liaison matérielle avec le terminal graphique 30-N.

Il faut signaler que le photostyle n'a pas encore été branché.

5. Expériences réalisées.

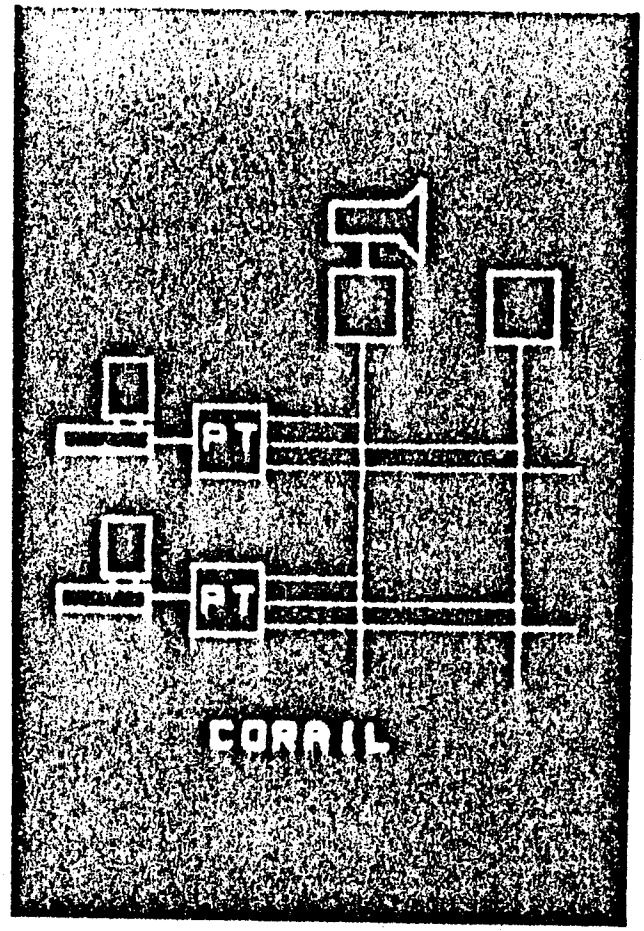
Les expériences que nous avons réalisées ont consisté, tout d'abord, en tester et mettre au point tout le matériel décrit précédemment, ensuite nous avons testé le système en mode local ainsi qu'en mode périphérique.

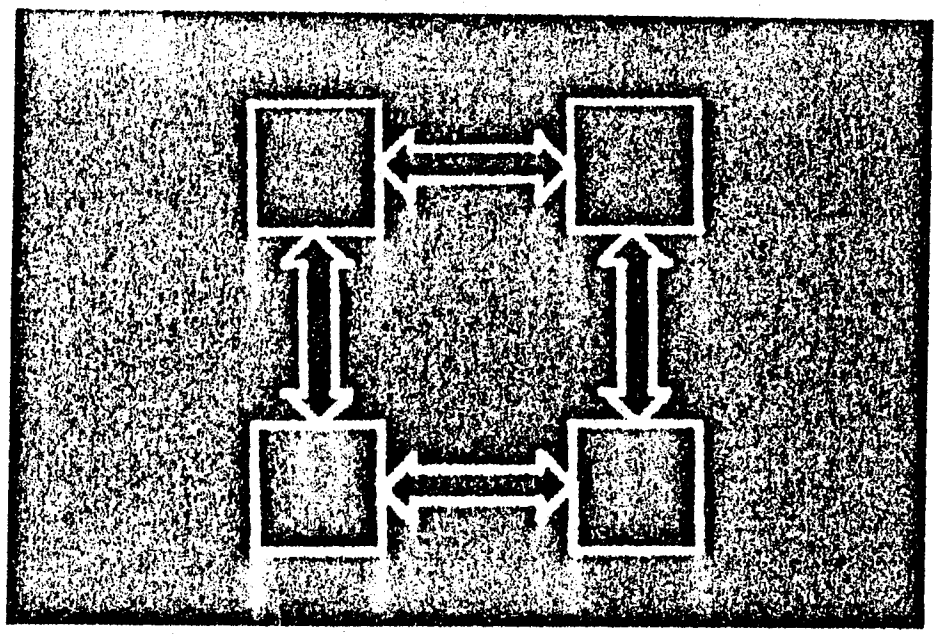
Dans une première étape nous avons câblé et mis au point l'interface avec le terminal 30-N, ensuite nous avons programmé les algorithmes pour la génération de segments de droite et caractères ainsi que pour l'interprétation de la liste d'affichage. Une fois la mise au point de ces algorithmes nous les avons enregistrés dans des mémoires mortes que nous avons placées sur la carte du Processeur d'Affichage.

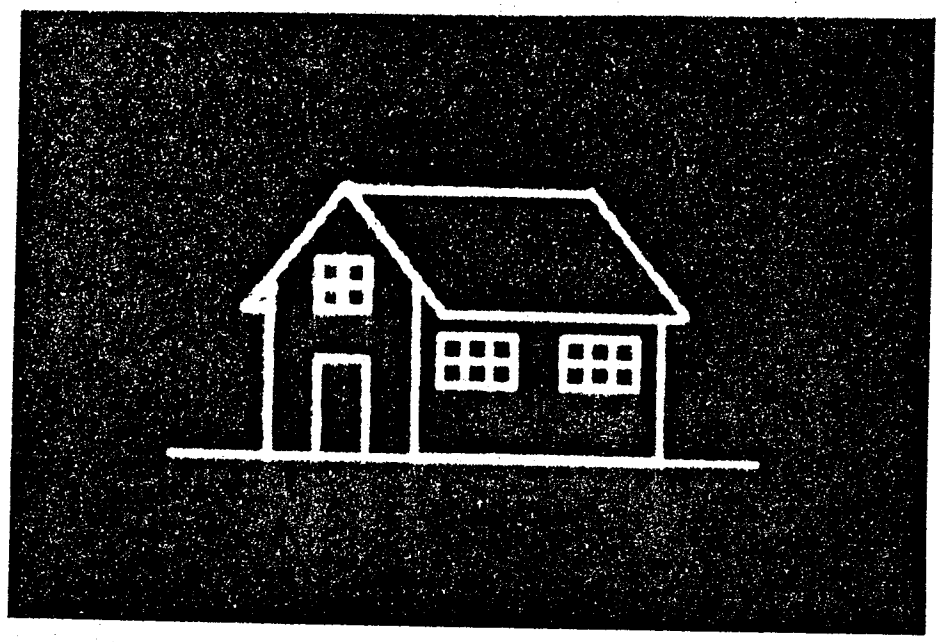
Dans une deuxième étape nous avons programmé les primitives GRIGRI pour la définition et la déclaration des dessins ainsi qu'un ensemble minimum de procédures pour exploiter l'unité arithmétique AMD9511. Nous avons utilisé les fonctions d'entrée/sortie du superviseur de mise au point pour la définition de primitives de définition de textes et de polygones ouverts. Ceci nous a permis de travailler en mode local.

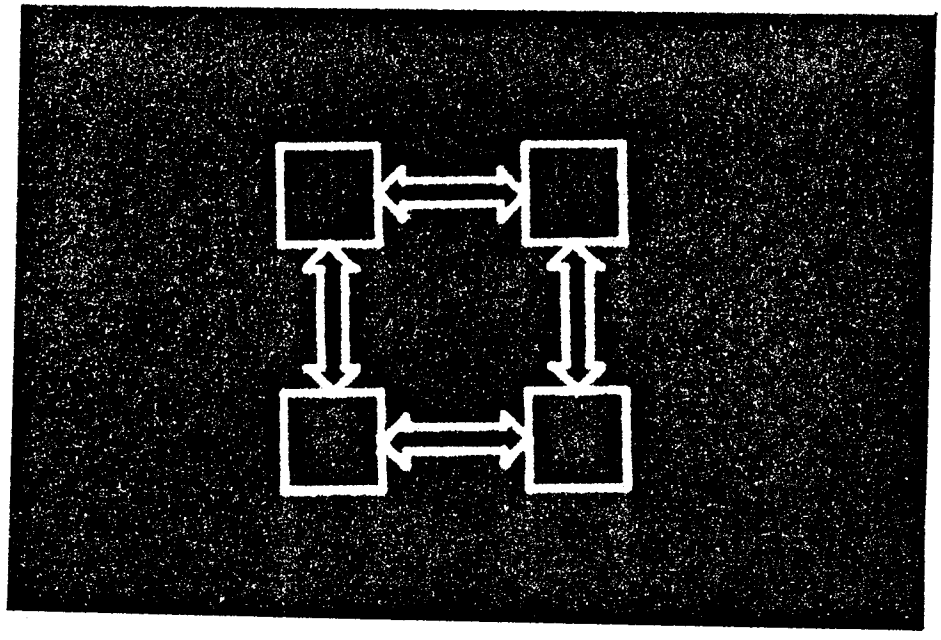
Dans une troisième étape nous avons câblé et mis au point l'interface CORAIL afin de pouvoir travailler en mode périphérique. Actuellement nous arrivons à envoyer et recevoir des messages avec le(s) Processeur(s) de Traitement.

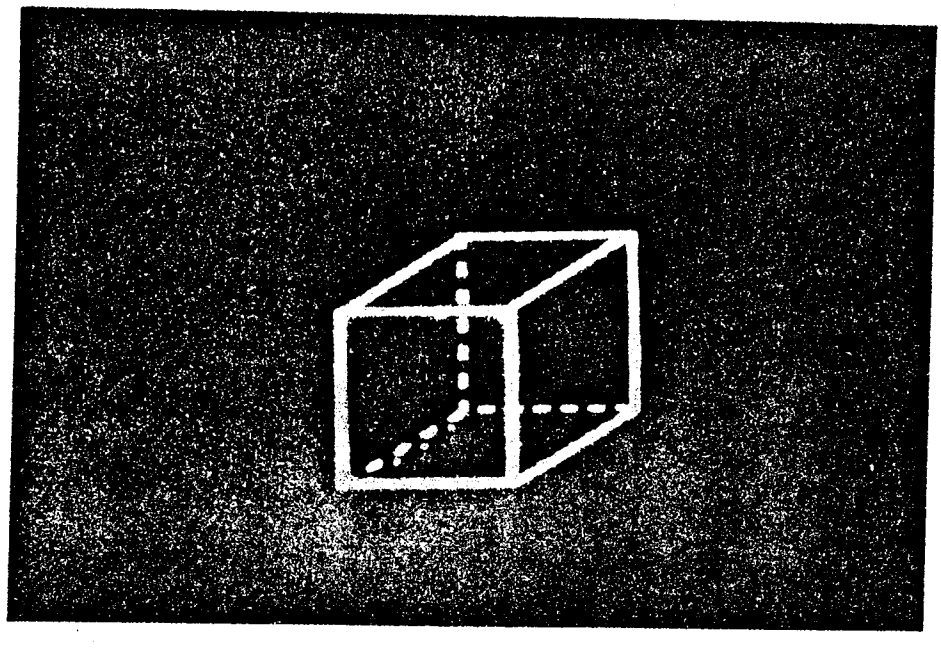
Des photos des exemples de création/modification des dessins que nous avons faits en utilisant le Système Graphique Interactif, sont montrées dans les pages suivantes.


PT
PT
CORAIL









dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- F. ANCEAU, Maître de Conférences à l'Institut National Polytechnique de GRENOBLE

- R.A. GUEDJ, Directeur du Laboratoire de Communication Homme-Machine de la THOMSON-CSF - ORSAY -

Monsieur Alejandro VILLAVICENCIO RAMIREZ

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 2 Août 1979

Ph. TRAYNARD
Président
de l'Institut National Polytechnique

P.O. le Vice-Président,