N° d'ordre : 947

THÈSE

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR TROISIEME CYCLE

Spécialité : Electronique

par

Negib LAHMAR
maitre ès sciences

SUR UN SYSTEME DE GESTION DE PROCESSUS MULTITACHES : Application à la commande et à la simulation

Soutenue le 9 février 1982 devant le Jury d'examen :

MM.	F. LAURENT	Professeur	Président
	J.C.GENTINA	Professeur	Rapporteur
Mme	G. DAUPHIN	Docteur	Rapporteur
MM.	P. BORNE	Professeur	Examinateur
	M. BENREJEB	Professeur	Examinateur
	D. CORBEEL	Docteur	Examinateur
	P. WALRAVE	Directeur Sté «TEREL»	Invité

S O M M A I R E

o

o o

o

o o

o

o o

o

o o

Ce langage de haut niveau, après compilation, est interprété par un logiciel approprié (moniteur interpréteur). L'automate réalisé assure la gestion pseudo-parallèle de 8 tâches au maximum.

Ces différentes tâches évoluent et communiquent par la mise en œuvre de 3 primitives de liaisons normalisées ; il s'agit de la synchronisation, du partage de ressources exclusives, du producteur-consommateur.

Il doit être possible d'assurer sans difficultés les fonctions de calcul complexe permettant la prise en compte de données analogiques et le traitement de texte dans des applications de contrôle interactif.

Le partage du temps entre tâches est obtenu par une scrutation séquentielle des tâches actives. Le traitement des macro-instructions d'une tâche est effectué jusqu'à l'attente d'une condition de transition ou le dépassement d'un temps limite maximal alloué à chaque tâche.

Nous proposons d'utiliser le mode prioritaire et le système de gestion d'interruptions pour assurer une efficacité maximale dans la prise en compte des évènements. Dans ce sens, le processeur se trouve déchargé des tâches de scrutation des évènements, il est directement alerté par le processeur de gestion d'interruptions sur occurence d'un évènement.

Le plan que nous adopterons pour la présentation de ce mémoire comporte quatre chapitres.

Dans un premier chapitre, nous rappellerons les caractéristiques essentielles d'une classe de modèles basés sur l'utilisation des Réseaux de Pétri et du Grafcet.

Dans une deuxième partie, nous présenterons les caractéristiques matérielles et logicielles de l'automate réalisé en le situant par rapport à d'autres systèmes de commande.

Dans un troisième volet, nous décrirons tout d'abord le langage de description d'automatismes puis nous préciserons les caractéristiques de l'interpréteur.

Nous tenterons, dans un dernier chapitre, d'illustrer par deux exemples d'application la facilité de mise en œuvre du logiciel et les possibilités matérielles de l'automate.

La première application concerne la commande d'une machine à usinages multiples alimentée par un robot.

Le deuxième exemple est relatif au contrôle d'un calculateur analogique et concerne la simulation d'un échangeur thermique par convection.

CHAPITRE 1

INTRODUCTION

Nous allons aborder dans ce chapitre, la présentation d'une classe de représentation des processus.

On définit formellement la notion de Processus au sens de l'Automatique et de l'Informatique.

Dans les deux cas, il s'agit d'effectuer, à partir d'un automate, la gestion d'un processus industriel ou d'un système opératoire. La modélisation de l'automate de contrôle fait appel à différents types de représentations mathématiques telles que les équations d'état ou des représentations graphiques du type organigramme, Réseau de Pétri ou Grafcet.

Nous proposons dans ce chapitre, de rappeler les caractéristiques essentielles des Réseaux de Pétri et du Grafcet utilisés dans ce mémoire comme support d'analyse, de synthèse et de représentation d'automates de contrôle.

I.1 - GENERALITES SUR LES RESEAUX DE PETRI

Durant les dernières années, les réseaux de Pétri ont suscité de nombreux travaux en vue de l'analyse et de la synthèse des structures de commande de processus discontinus.

Cet outil permet une représentation graphique du cahier des charges d'un automatisme ; sa décomposition en sous-Réseaux correspond à la décomposition d'un système en sous-machines de commande, ce qui facilite la synthèse des grands ensembles tout en tenant compte de fonctionnements asynchrones parallèles de sous-ensembles du processus à automatiser.

En général, une modification locale du cahier des charges n'introduit qu'une modification locale dans le réseau |1| |2| |3| |4| sans remettre en question la totalité d'une synthèse.

La modélisation par Réseaux de Pétri est parfaitement adaptée à la représentation de programmes ou de tâches parallèles affectés à la gestion des processus industriels. |9|

C'est pourquoi nous proposons de rappeler sommairement les définitions, les caractéristiques principales des Réseaux de Pétri utilisés dans ce mémoire.

I.1.1 - Graphe de Pétri

Définition : Un graphe de Pétri est un triplet G (P, T, Γ)

où P est un ensemble fini de sommets (ou nœuds) appelés places et représentés graphiquement par des cercles.

T est un ensemble fini de sommets appelés transitions et représentés graphiquement par des barres.

Γ est la correspondance associant à un sommet, ses successeurs.
(Γ^{-1}, associe à un sommet, ses prédécesseurs)

$\Gamma(P) \subset T$ et $\Gamma(T) \subset P$

La figure 1 représente un graphe de Pétri.

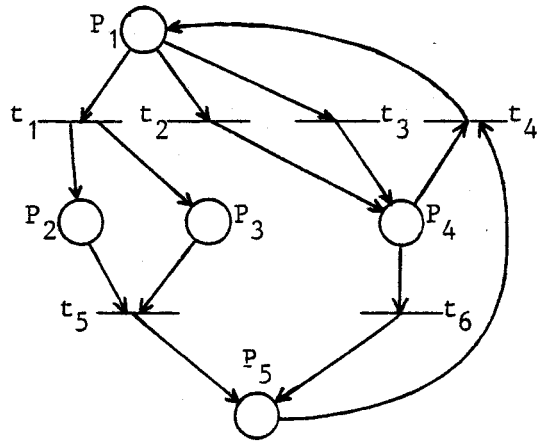


Figure 1

$P = \{P_1, P_2, P_3, P_4, P_5\}$

$T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$

μ	P_1	P_2	P_3	P_4	P_5	t_1	t_2	t_3	t_4	t_5	t_6
$\Gamma(\mu)$	$t_1t_2t_3$	t_5	t_5	t_4t_6	t_4	P_2P_3	P_4	P_4	P_1	P_5	P_5

I.1.2 - Graphe d'état

Un graphe d'état est un graphe de Pétri particulier tel que toute transition a une place d'entrée et une place de sortie.

L'exemple de la figure 1 n'est pas un graphe d'état puisque t_1, t_4, t_5 ont plusieurs places d'entrée et de sortie. Par contre, l'exemple de la figure 2 est un graphe d'état.

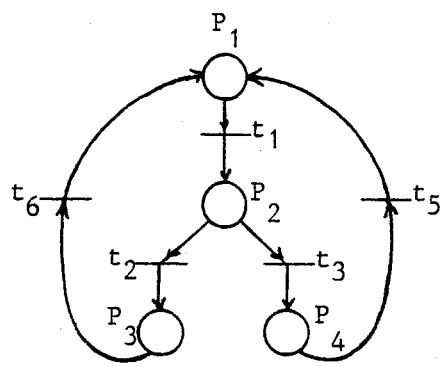


Figure 2

I.1.3 - Réseau de Pétri

Un Réseau de Pétri est un couple R = (G, M_o) où G = G (P, T, Γ) est un graphe de Pétri ; M_o appelé marquage initial du réseau est une application de P dans ℕ.

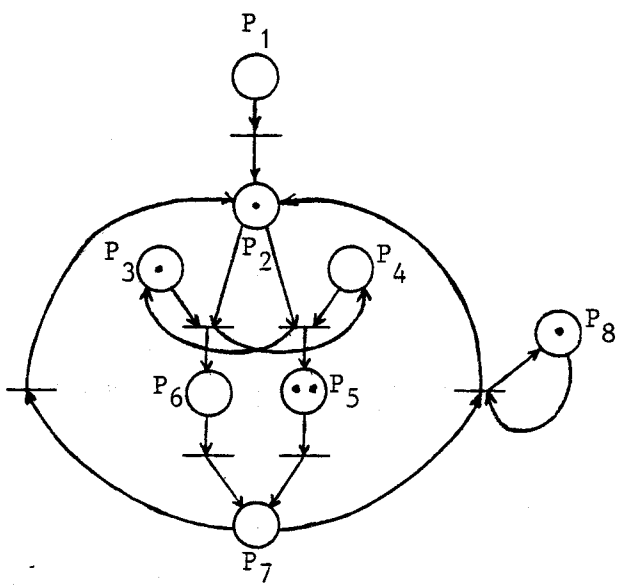
Si n est le nombre de places dans le réseau, un marquage M_o de R est défini par la donnée d'un Vecteur.

$$M_o = \begin{vmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{vmatrix} \quad \text{et} \quad m_i = M_o\ (P_i)$$

I.1.3.1 - Représentation

Sur le graphe associé à R, le marquage M_o peut être représenté par une distribution dans les places, d'objets appelés marqueurs. Une place P contenant K marques est schématisée par K points.

$$M_o\ (P) = K$$



$$M_o = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{vmatrix}$$

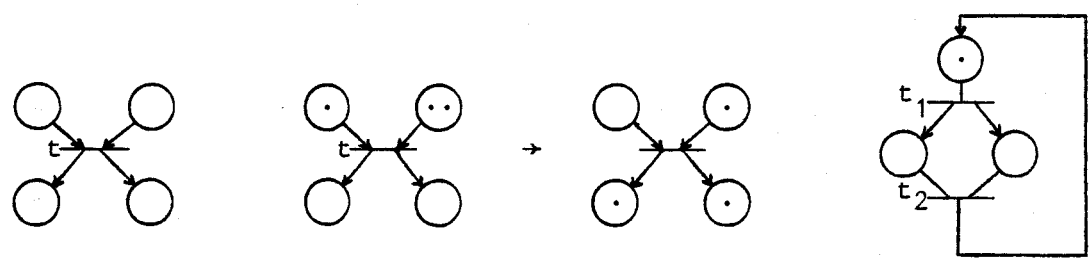
Figure 3

I.1.4 - Règles de déclenchement d'une transition

Le marquage du réseau correspond à l'état du Réseau et doit donc être susceptible d'évoluer ; ceci peut se faire par le déclenchement des transitions.

Définition : Une transition t est dite déclenchable pour un marquage M si et seulement si, pour tout P de Γ^{-1} (t), M (t) $\geq$ 1.

Autrement dit, une transition est validée si chaque place d'entrée de cette transition contient au moins un marqueur.



t non validée — t est validée — après déclenchement

a — b — c — d

Figure 4

Le déclenchement ou le tir d'une transition consiste à enlever un marqueur de chaque place d'entrée de la transition pour en rajouter un à chaque place de sortie.

I.1.4.1 - Notion de conflit

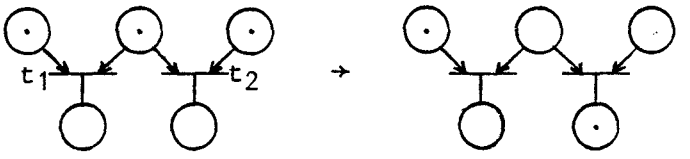
Le tir d'une ou d'un ensemble de transitions validées simultanément peut modifier ou non le nombre total de marqueurs d'un réseau, le tir de t_1 dans le réseau de la figure 4d augmente le nombre de marqueurs.

Lorsqu'il existe une place d'entrée commune à deux ou plusieurs transitions validées simultanément, il n'est pas possible d'appliquer la règle de tir énoncée ci-dessus. Les transitions validées sont dites en conflit.

Il est donc nécessaire de rendre une transition prioritaire par

rapport aux autres, sinon il y a blocage du réseau. Il existe des moyens d'associer des priorités aux transitions.

Le tir de la transition prioritaire modifie le marquage et supprime la validation aux autres.



t_1 et t_2 validées simultanément sont en conflit pour P_2.

Si on rend t_2 prioritaire par rapport à t_1, le conflit est résolu.

I.1.4.2 - Séquence de déclenchement

Nous dirons que M' est un marquage atteint à partir de M_o s'il existe une séquence de déclenchement qui conduit le processus à cet état. On note l'ensemble des marquages atteints à partir de M_o, par ε_{M_o}.

I.1.5 - Propriétés associées aux marquages

I.1.5.1 - Réseau borné

Soit P un Réseau de Pétri et M_o son marquage initial. P est dit borné pour M_o si et seulement si il existe un entier positif donné n_{max} tel que pour tout marquage M_i de l'ensemble ε_M conséquent à M_o, aucune place de P ne contient plus de n_{max} marqueurs.

I.1.5.2 - Réseau vivant

Un Réseau de Pétri P est dit vivant pour M_o si et seulement si pour toute transition t de P et pour tout marquage M_i de M_o, il existe une séquence de tir comprenant t tirable à partir de M_i. Autrement dit, si toute transition du réseau peut être validée et tirée par une séquence finie de tir, alors le réseau est vivant. Ceci entraîne l'absence de blocage et toute partie du réseau sera accessible en fonctionnement dynamique.

I.1.5.3 - Réseau sain ou sauf

Un réseau est dit sauf pour un marquage initial M_o si, quelque soit le marquage obtenu à partir de M_o par une séquence finie de déclenchements, aucune place ne posséde plus d'un marqueur.

Autrement dit, c'est un réseau borné à 1 ($n_{max} = 1$).

I.1.5.4 - Réseau conforme

C'est un réseau vivant et sauf à la fois.

I.1.5.5 - Réseau propre

Un réseau P est propre si et seulement si, pour tout marquage M conséquent à M_o, il existe une séquence de déclenchement telle que $M = M_o$.

On peut donc réinitialiser le réseau.

I.1.6 - Réseaux de Pétri non autonomes

Nous avons jusqu'alors rappelé les définitions et propriétés relatives aux réseaux autonomes. Pour de tels réseaux, aucunes conditions particulières autres que celles définies dans les règles de déclenchements (§ I.1.4) n'est mis en œuvre pour déclencher une transition. Il est cependant très intéressant de pouvoir synchroniser les mises à feu de transitions sur des évènements externes. Chaque transition "validée" devient alors "réceptive" à un évènement qui provoquera la mise à feu et le transfert des marqueurs des places d'entrée aux places de sortie.

I.2 - PROCESSUS ET SCHEMA DE PROCESSUS [5]

I.2.1 - Introduction

On definit formellement la notion de processus au sens de l'automatique lorsqu'il s'agit d'effectuer le contrôle de processus industriel ou au sens de l'Informatique pour décrire le contrôle d'un système opératoire.

I.2.2 - Processus assimilé à un Réseau de Pétri

Un processus correspond à un programme séquentiel donc à un enchaînement séquentiel de tâches (ici prises au sens large c'est à dire activités).

I.2.2.1 - Nombre de marque

Le nombre de marques d'un sous-ensemble de places P' d'un Réseau de Pétri $R = (G, M_o)$ pour un marquage M' de $\mathcal{E}_{M_o}$, représente la quantité $\sum_{P \in P'} M'(P)$ et sera noté $N(P', M')$.

I.2.2.2 - Processus |5|

Un réseau de Pétri $R = (G, M_o)$ est un processus si et seulement si :

- pour toute place p de P : $|\Gamma(p)| \leq 2$ (| | : cardinal)
  : nombre
- pour toute transition t de T : $|\Gamma^{-1}(p)| \leq 2$
  autrement dit G est un graphe d'état.
- $N(P, M_o) = 1$: c'est à dire le nombre de places marquées à l'état initial est égal à 1.
- Le graphe G est fortement connexe : entre tout couple de sommets distincts, il existe un chemin.

I.2.2.3 - Propriétés des processus |5|

Un processus est un Réseau de Pétri propre, vivant, sauf et pour tout marquage M appartenant à $\mathcal{E}_{M_o}$, on a : $N(P, M) = 1$

I.2.3 - Schéma de processus et interprétation d'un processus

I.2.3.1 - Réseau de Pétri interprété

C'est un Réseau de Pétri totalement synchronisé, auquel on associe pour chaque place un opérateur, et à chaque transition, une condition logique de réceptivité non dépendante du marquage.

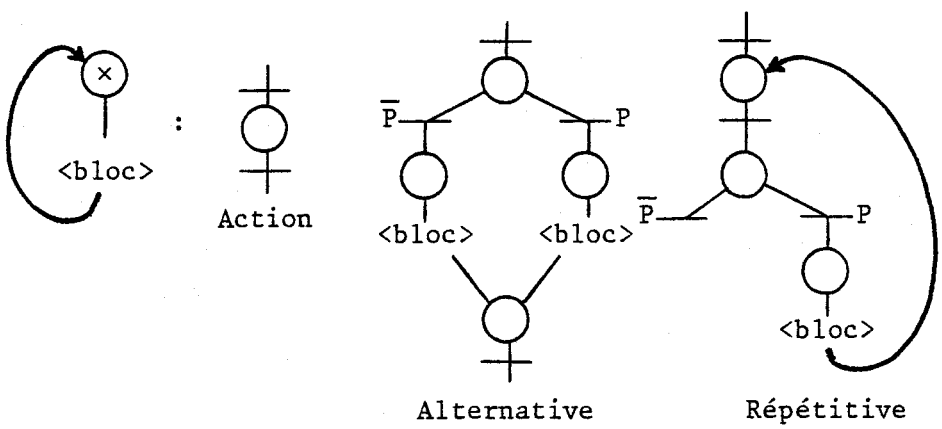
I.2.3.2 - Schéma de processus

Un schéma de processus interprété est tel que l'ensemble d'opérateurs se compose de deux sous-ensembles disjoints d'actions a et de tests P.

I.2.4 - Schéma de processus structuré

On représente le schéma de contrôle d'un processus structuré par un Réseau de Pétri interprété ; le graphe représentant le Réseau de Pétri est engendré par la grammaire suivante :

Action, Alternative, Répétitive : ce qui constitue un bloc.



La classe des schémas de processus structurés suffit pour décrire tout schéma de processus.

L'utilisation de ces trois structures permet d'assurer que le système est sauf, propre et vivant.

I.2.5 - Système de processus et relations entre processus

Un système de plusieurs processus est défini par :

- les données de chacun des processus le constituant
- les liaisons réprésentant les intéractions entre ceux-ci.

La décomposition d'un système industriel en n processus doit tenir compte du parallélisme d'actions, de la cohérence fonctionnelle et de la

minimisation du nombre de processus.

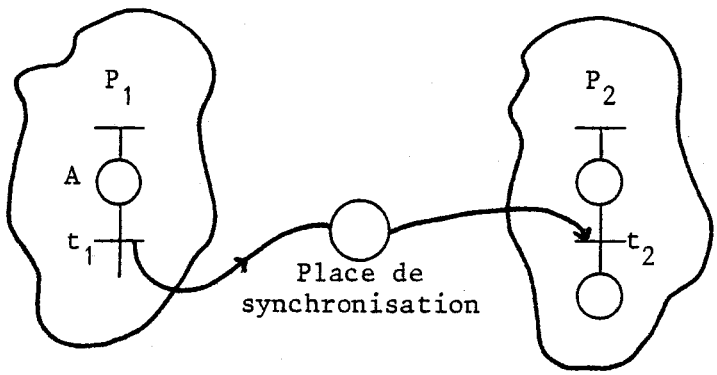
Chaque processus peut être décrit par un Réseau de Pétri structuré (réseau qui ne possède qu'un seul marqueur), puis on fait apparaître les liaisons entre différents graphes.

I.2.5.1 - Les liaisons

Les liaisons sont les intéractions entre les blocs des divers processus. Elles sont de trois types :

- Liaison de synchronisation
- Liaison de type exclusion mutuelle (sémaphore)
- Liaison de type producteur-consommateur

I.2.5.2 - Liaison de synchronisation

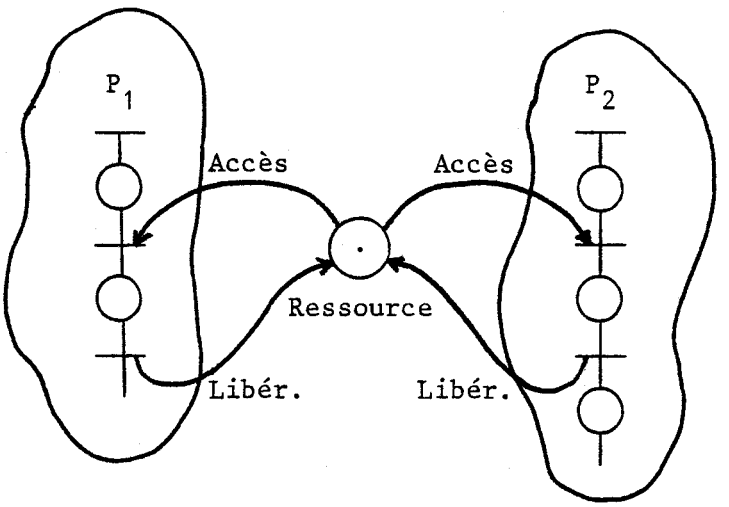


P_1 synchronise P_2 ; la transition t_1 est la transition synchronisante émettrice ; t_2 transition synchronisée est réceptrice.

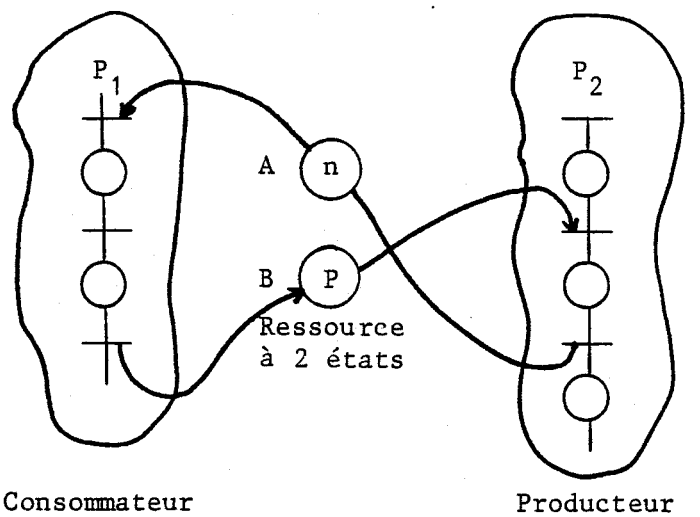
Il n'y a pas d'accumulation de marqueurs dans la place de synchronisation.

I.2.5.3 - Liaison de type d'exclusion mutuelle

Les deux processeurs P_1 et P_2 utilisent la même ressource ; son utilisation doit être exclusive. Elle est restituée après emploi.



I.2.5.4 - Liaison de type producteur-consommateur



Le producteur P_2 utilise la ressource dans son état B. Il la restitue ensuite dans l'état A.

De même, le consommateur P_1 prend la ressource dans son état A ; après utilisation, il la restitue dans l'état B.

I.3 - AUTRE SYSTEME DE REPRESENTATION DE PROCESSUS : LE GRAFCET |6|

Il existe d'autres systèmes de représentation de processus, il faut citer notamment les organigrammes, organiphases et le Grafcet.

Nous proposons de rappeler ici quelques éléments sur le Grafcet, outil puissant d'analyse et de synthèse de processus.

I.3.1 - Rappel des principales règles du Grafcet

I.3.1.1 - Définition

Le Grafcet (Graphe de Commande Etapes-Transitions) est un modèle qui a été présenté par la commission du groupe "Systèmes Logiques" de l'AFCET |7| ; en vue de la normalisation de la représentation du cahier des charges d'un automatisme logique.

C'est un graphe orienté sur lequel est superposée une interprétation.

Comme les Réseaux de Pétri, il comprend un nombre fini de nœuds de deux types :

- Les étapes (Resp. places)
- Les transitions.

Ces nœuds sont connectés entre eux par des arcs orientés, en respectant les deux règles suivantes :

- un arc ne connecte que des nœuds de types différents (étape à transition ou transition à étape)
- deux nœuds ne peuvent être connectés que par deux arcs au plus, un arc dans chaque sens.

I.3.1.2 - Représentation

Si l'on adopte la représentation originale présentée par l'AFCET, un exemple de Grafcet est donné par la figure 5.a (Graphe Original). On remarque la similitude avec les Réseaux de Pétri.

Après la normalisation opérée par l'ADEPA |8| qui a proposé de donner au Grafcet une représentation spécifique, le même exemple de processus de 5.a après la normalisation devient celui de la figure 5.b.

I.3.1 - Rappel des principales règles du Grafcet

I.3.1.1 - Définition

Le Grafcet (Graphe de Commande Etapes-Transitions) est un modèle qui a été présenté par la commission du groupe "Systèmes Logiques" de l'AFCET |7| ; en vue de la normalisation de la représentation du cahier des charges d'un automatisme logique.

C'est un graphe orienté sur lequel est superposée une interprétation.

Comme les Réseaux de Pétri, il comprend un nombre fini de nœuds de deux types :

- Les étapes (Resp. places)
- Les transitions.

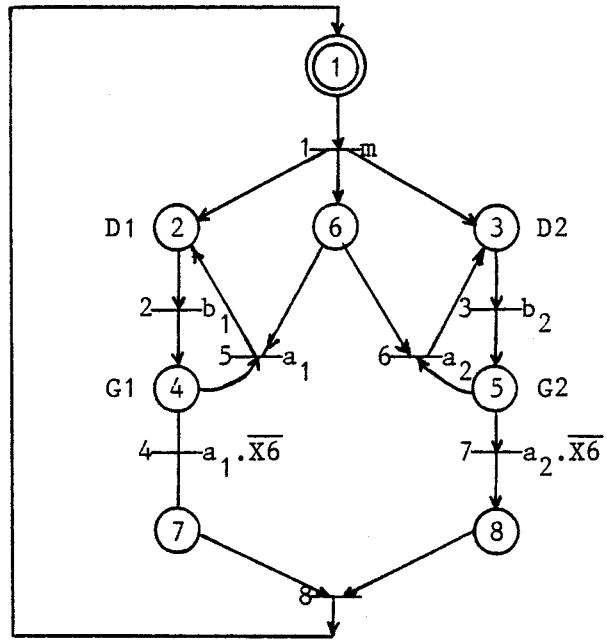
Ces nœuds sont connectés entre eux par des arcs orientés, en respectant les deux règles suivantes :

- un arc ne connecte que des nœuds de types différents (étape à transition ou transition à étape)
- deux nœuds ne peuvent être connectés que par deux arcs au plus, un arc dans chaque sens.

I.3.1.2 - Représentation

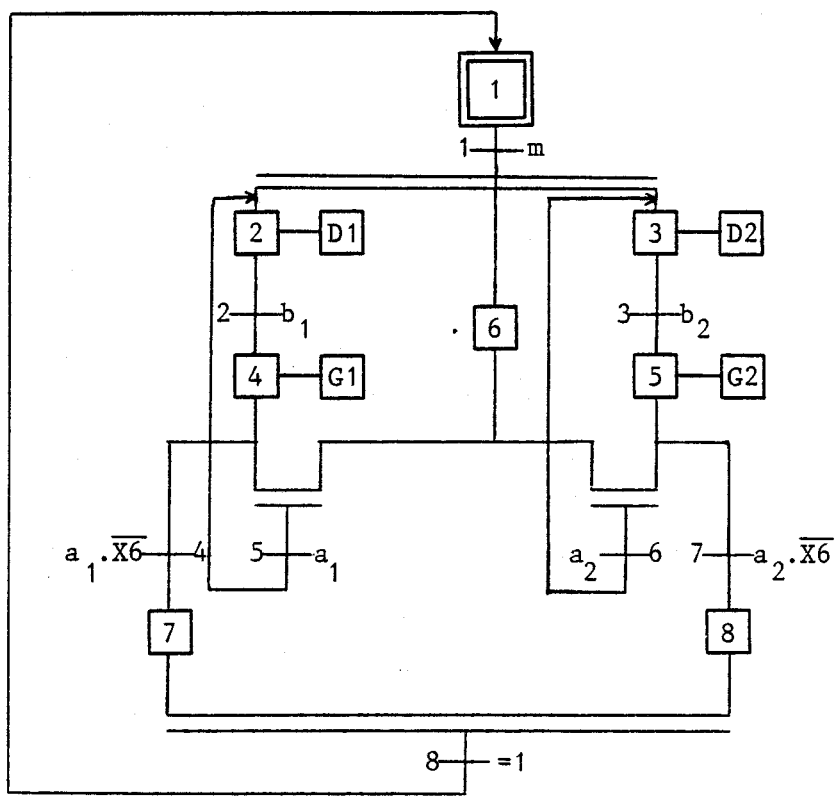
Si l'on adopte la représentation originale présentée par l'AFCET, un exemple de Grafcet est donné par la figure 5.a (Graphe Original). On remarque la similitude avec les Réseaux de Pétri.

Après la normalisation opérée par l'ADEPA |8| qui a proposé de donner au Grafcet une représentation spécifique, le même exemple de processus de 5.a après la normalisation devient celui de la figure 5.b.



GRAPHE ORIGINAL

Figure 5.a



GRAPHE NORMALISE

Figure 5.b

I.3.1.3 - Règles d'évolution

1° Règle : L'initialisation précise les étapes activées au début du fonctionnement. Elles sont représentées par un double contour du symbole correspondant (exemple de l'étape 1).

2° Règle : Une transition est validée lorsque les étapes immédiatement "prédécesseurs" sont actives.

3° Règle : Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement "successeurs" et la déactivation des étapes "prédécesseurs".

4° Règle : Plusieurs transitions simultanément franchissables sont simultanément franchies.

5° Règle : Si, au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste active.

I.3.1.4 - Sur l'interprétation du Grafcet

L'interprétation associe à chaque étape, des actions et à chaque transition, une réceptivité.

- Les actions correspondent aux fonctions à réaliser chaque fois que l'étape est activée. Elles peuvent être conditionnelles.

- La réceptivité précise le sous-ensemble des transitions validées donc réceptives aux conditions de franchissement.

Toutes deux peuvent être conditionnées par l'adjonction d'une fonction combinatoire qui fait intervenir l'état ou le changement d'état des variables d'entrée et éventuellement l'état actif ou inactif de certaines étapes.

L'activation d'une étape déclenche l'exécution des actions associées. Une action peut être à niveau (elle est exécutée en permanence tant que l'étape reste active) et impulsionnelle (elle n'est effectuée qu'une seule fois et pour une durée brève).

On note une action impulsionnelle à l'aide d'un astérisque suivi éventuellement de sa durée.

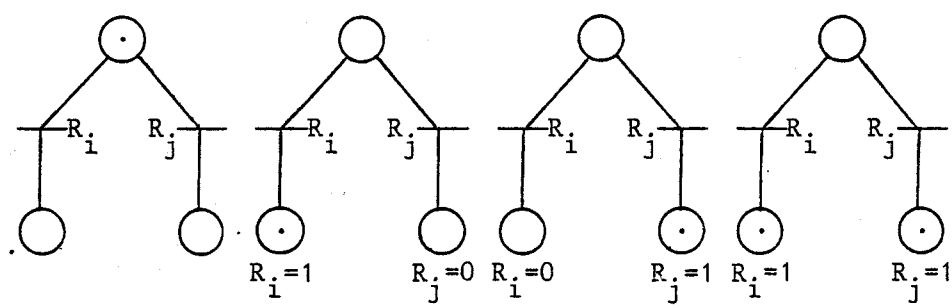
I.3.2 - Comparaison avec les Réseaux de Pétri |9|

Le Grafcet emprunte certains élèments de représentation aux Réseaux de Pétri ; il en diffère cependant par certaines règles d'évolution.

Le Grafcet a un fonctionnement non autonome et ne peut être comparé de ce fait qu'à un Réseau de Pétri non autonome. Le modèle Grafcet se rapproche de la classe des Réseaux de Pétri Interprétés saufs.

On peut dégager les différences essentielles suivantes :

- La notion de conflit dans un Réseau de Pétri ne se pose pas dans le Grafcet puisque la 3° Règle d'évolution permet le franchissement simultané de plusieurs transitions simultanément franchissables (c'est la notion de divisibilité du marqueur qui n'existe pas dans un Réseau de Pétri).



avant franchissement après franchissement

EVOLUTION DU GRAFCET

- Dans les R d PI, les marques s'accumulent dans les places sauf dans les R d PI saufs qui sont conçus de telle façon qu'une place ne contienne jamais plus d'une marque. Dans le Grafcet, plusieurs marqueurs accèdant à une étape seront confondus, au sens de la réunion (Absorption des marqueurs) (Etat d'une étape).

- Les transitions qui contiennent un test sur l'état du système

alourdit très sensiblement la réalisation en R d PI ; le Grafcet permet très facilement ce type de représentation.

- D'autres différences existent indiquant notamment que le Grafcet offre plus de souplesse et de facilité pour la description des systèmes logiques que les R d PI.

Ces facilités permettent une simplification de la représentation mais conduisent généralement à une plus grande difficulté dans la vérification et l'analyse de la description.

Les R d P sont suffisamment formels et bien connus du point de vue théorique pour permettre une analyse fine.

CONCLUSION

Dans ce premier chapitre, nous avons pu rappeler sommairement les règles générales des deux principaux outils de représentation des processus, que sont les R d P et le Grafcet.

Ces modèles de représentation permettent tout aussi bien d'aborder l'analyse que la synthèse des systèmes. Ils traduisent à la fois, de manière simple, le parallélisme du contrôle et la dynamique d'évolution du processus. |17|

Toutefois, la vérification de la validité d'un modèle graphique peut être délicate. En effet, il est possible dans le cas des R d P, d'analyser et de vérifier mathématiquement les propriétés d'invariance et de consistance du graphe, essentiellement liées à la structure et aux conditions de marquage initial. |10|

Si le graphe est structuré (pour chaque tâche), cette vérification ne concerne en fait que le graphe de liaison. Pour un grafcet, de telles vérifications sont beaucoup plus difficiles en raison d'une part, des interprétations et d'autre part, des règles d'évolution plus larges des marqueurs. |17|

CHAPITRE 2

PRESENTATION DE L'AUTOMATE IDN

II.1 - INTRODUCTION

Au cours de ce chapitre, nous proposons de présenter la partie matérielle et logicielle de l'automate utilisé en le situant dans le contexte général des matériels actuellement disponibles.

Dans une première partie, nous aborderons les différents choix de matériels possibles en vue d'effectuer la commande programmée d'un processus.

Dans une seconde étape, nous présenterons le caractéristiques générales de l'automate, d'abord logicielles puis matérielles.

Dans ce sens, nous préciserons ce qui constitue la partie originale de ce travail concernant le traitement multitâches, à partir d'un monoprocesseur basé sur l'utilisation d'un microprocesseur.

II.2 - NOTION DE COMMANDE DE PROCESSUS

En automatique, la commande des processus tient une part importante, en vue de l'automatisation rationnelle d'un ou de plusieurs procédés.

Le schéma fonctionnel de la figure 6 décrit globalement le principe d'une installation automatisée.

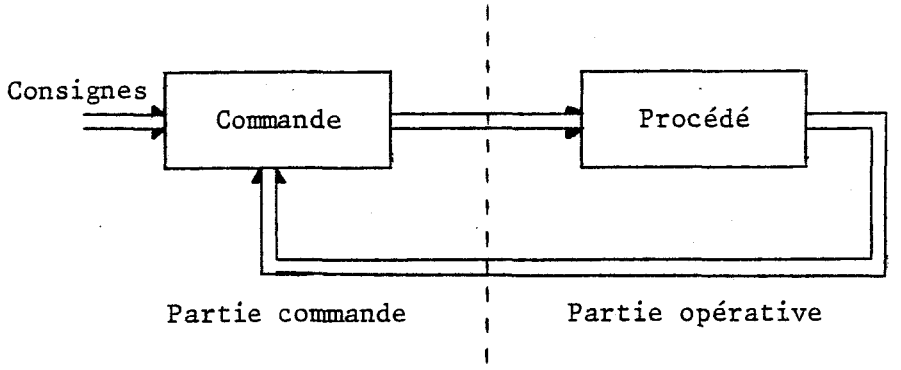


Figure 6

Il est constitué de deux ensembles interconnectés : la partie commande et la partie opérative.

Si l'on prend comme exemple, une machine outil à commande numérique, la partie opérative est la machine y compris ses capteurs et actionneurs (la mécanique, les moteurs, ...), la partie commande est le dispositif de traitement numérique.

La première partie est chargée d'effectuer les tâches fonctionnelles productives (usinage, ...), la deuxième partie est chargée de prendre en considération d'une part les consignes imposées au système (données, côtes, ...) et d'autre part, l'état du procédé (Fin de course, vitesse, pas, emplacement des outils, ...) pour élaborer les commandes susceptibles de faire évoluer l'ensemble d'une manière autonome et prédictive.

Selon la nature et le nombre des signaux échangés entre les deux parties, des classifications de systèmes s'opèrent, pour orienter le choix d'une technologie particulière de commande. Un tel choix doit prendre en considération la complexité du problème posé par l'élaboration des signaux de commande.

Si l'on s'intéresse par exemple, à la classe des systèmes séquentiels

qui représentent une large part des automatismes industriels, les signaux échangés sont du type "tout ou rien".

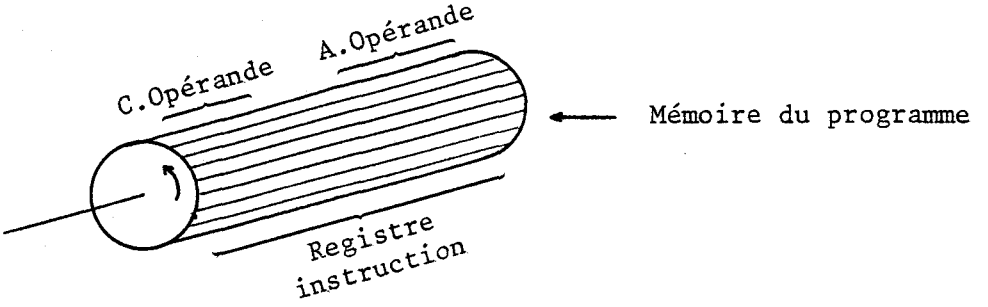
Deux classes de solution peuvent être envisagées selon qu'il s'agit de réalisations de commandes cablées ou programmées.

La technologie cablée utilise un nombre important de circuits discrets, à faible niveau d'intégration (reliés matériellement entre eux). Dans certains cas simples où les tâches à effectuer sont peu nombreuses, où la rapidité est primordiale et le combinatoire peu évolué, alors l'apport d'un matériel sophistiqué programmable n'est pas nécessaire, et la solution cablée peut encore être intéressante.

Elle cède cependant la place à des outils programmables plus flexibles, plus faciles d'accès grâce à de nouvelles méthodologies d'analyse et de synthèse des automatismes.

II.3 - AUTOMATES PROGRAMMABLES INDUSTRIELS (API) |11||12|

Leur première application a concerné le traitement d'équations logiques (combinatoire et séquentielle) assurant la commande en temps réel d'un enchaînement de tâches. L'évolution récente des automates permet d'effectuer du traitement numérique, et parfois, pour certains automates de haut de gamme, de traiter des signaux analogiques. On dispose parfois dans ce sens, de régulateurs PID préprogrammés. La spécificité originale et communes aux API est le fonctionnement cyclique de l'unité centrale.



Chaque mot du programme contient une instruction. Un lecteur permet-par l'intermédiaire d'un pointeur, de transmettre l'instruction à un registre qui la décode et l'exécute. Souvent, la nature des opérations effectuées sont des calculs et des tests sur des bits, ou sur un en-

semble de bits formant un mot.

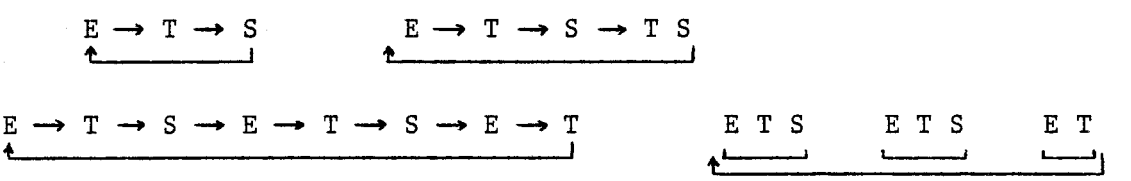
Une incrémentation du pointeur (sauf cas de saut) permet l'exécution séquentielle des instructions et assure le fonctionnement cyclique (ou synchrone) de l'U.C.. Le processeur qui permet de décoder les instructions est cablé et/ou microprogrammé.

Les instructions ont le plus souvent un format de codage sur 16 bits. La puissance de l'unité centrale est liée directement à sa vitesse d'exécution. On appelle période d'un API, le temps d'exécution de 1 K instructions logiques.

Lors d'un cycle, l'automate assure les échanges avec l'extérieur et les traitements spécifiés par les programmes.

Un cycle réel comprend deux phases : Phase d'acquisition des Entrées/sorties (E/S) et Phase de traitement (T).

Il existe plusieurs déroulements de cycle selon les automates.



L'organisation devient plus complexe lorsqu'on se trouve en présence d'instruction de saut particuliers qui peuvent rallonger ou raccourcir le cycle ou parfois bloquer entièrement le cycle à la suite d'une erreur de programmation.

Certaines méthodes permettent de détecter les "boucles infinies". Dans ce sens, l'utilisation d'un dispositif mesurant le temps écoulé entre deux passages par un "Point Obligé" de programme permet d'éviter ces difficultés.

Le "chien de garde" émet une alarme lorsque cette durée est supérieure à un seuil donné.

De plus, les calculs numériques sont traités sur plusieurs cycles avant leur utilisation.

Les éléments spécifiques permettant de caractériser un automate sont notamment :

- sa période (3/50 ms par exemple)
- Sa capacité mémoire
- La facilité de programmation
- La prise en compte éventuelle de temporisation et horloges
- La nature et la puissance des signaux gérés et élaborés par l'automate (24 V, 210 V ~, tensions analogiques, ...)

II.4 - LES MINI ET LES MICRO-ORDINATEURS

II.4.1 - Le mini-ordinateur

Il s'agit d'un outil universel et puissant, adapté à la gestion de systèmes complexes.

La capacité mémoire, la vitesse d'exécution et la richesse de son logiciel, lui permettant de traiter des problèmes de tous ordres.

Le plus souvent, l'utilisation de mini-ordinateurs est requise pour une commande centralisée de plusieurs processus travaillant en parallèle et dont les signaux échangés seront multiplexés sur des interfaces performants et sophistiqués.

II.4.2 - Les micro-ordinateurs

L'excellente fiabilité de ces matériels, leur faible coût, la grande richesse des fonctions possibles et composables, permet l'élaboration rapide et efficace des architectures de contrôle les mieux adaptées à un problème donné. En ce sens, une structure microprocesseur pourra très facilement prendre la place d'un automate voire même d'un minicalculateur.

II.5 - AUTOMATE IDN PROCESS

II.5.1 - Généralités

L'automate IDN Process est basé sur l'utilisation de la série 80 des microprocesseurs 8 bits.

Il a été conçu à partir d'une carte "mère" complétée par des E/S logiques et analogiques, des temporisateurs/compteurs.

Sur cette base, nous avons élaboré un logiciel (moniteur interpréteur) permettant d'assurer la commande pseudo parallèle de processus évoluant simultanément.

Le contrôle simultané de plusieurs tâches est envisagé ici à partir d'un processeur unique.

Nous cherchons essentiellement à obtenir à la fois le traitement le plus rapide possible et une approche de conception de haut niveau. [14] [15] [16]

Dans ce sens, nous proposons de présenter un logiciel de gestion de tâches en temps réel assurant une transcription quasi directe d'un cahier des charges défini sur la base de Grafcet ou de Réseaux de Pétri. [13] [14] [16]

II.5.2 - Spécificités générales du logiciel

L'automate "IDN Process" présenté doit permettre la gestion pseudo parallèle d'un maximum de 8 tâches. Ces dernières communiquent par la mise en œuvre de 3 primitives :

- la synchronisation
- Le partage exclusif de ressources à 1 état
- Le producteur/consommateur

Il convient donc d'assurer le partage du temps du microprocesseur entre les différentes tâches.

Dans ce sens, on considère qu'à un moment donné, les tâches sont

réceptives à un sous-ensemble très restreint d'évènements conditionnant l'exécution de macro-instructions.

Il suffit donc pour chacune des tâches, d'associer à chaque macro-instruction jouant le rôle d'une place/étape, une condition de déclenchement jouant le rôle d'une transition.

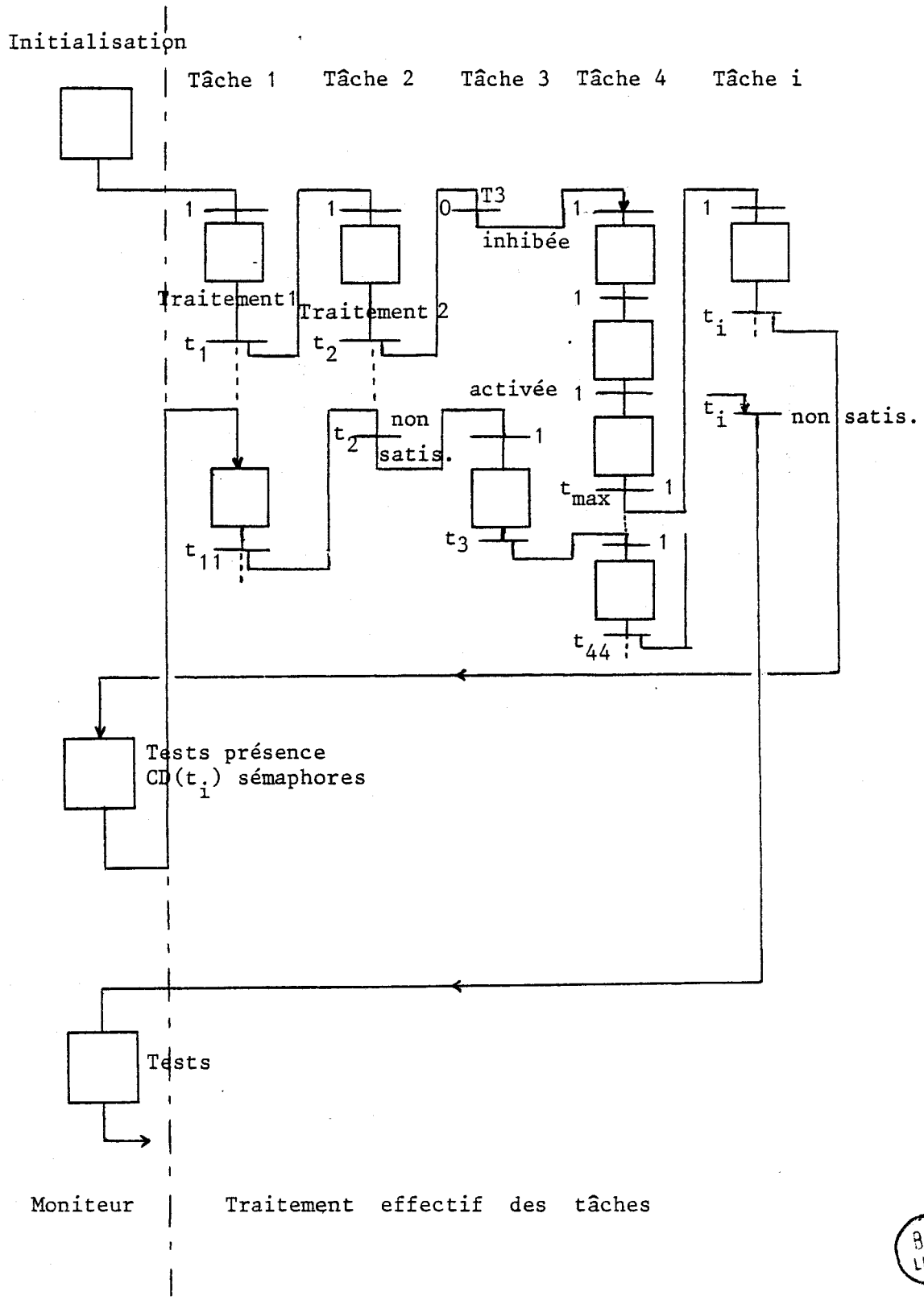
Cette condition est alors validée sur présence de l'évènement attendu. Cette procédure permet ainsi de traduire de manière simple et intéractive, l'enchaînement des fonctions consécutives d'une tâche.

Deux solutions peuvent alors être envisagées :

- soit pour un évènement donné, balayer toutes les tâches réceptives
- soït pour une tâche élémentaire, balayer les évènements enregistrés.

La première solution permet d'assurer facilement la remise à zéro des évènements attendus et acquittés, cette solution sera donc retenue pour notre application.

II.5.2.1 - Le partage du temps en traitement multitâches



Il convient d'allouer pour chaque macro-instruction, d'une tâche et pour l'ensemble des tâches, un temps maximum à ne pas dépasser.

Cette procédure permet d'éviter lors du blocage de l'une d'entre elles, le défaut d'activation des autres tâches.

Automatiquement après la scrutation séquentielle de l'ensemble des tâches activées, les tâches momentanément suspendues à la suite d'un dépassement de temps sont systématiquement relancées.

II.5.2.2 - Fonctionnement en mode interruptible

Les évènements externes attendus par les tâches, sont démasqués (réceptivité) et le restent jusqu'à leur prise en compte. Dès leurs apparitions, ou s'ils sont déjà présents au moment du démasquage, ils sont mémorisés dans une table par interruption du déroulement normal du programme principal.

En fin de scrutation des différentes tâches, on active les tâches en attente enregistrées en file.

Cette façon de procéder laisse plus de temps au processeur pour les traitements effectifs, d'où une plus grande efficacité. A l'encontre, la méthode de scrutation et test de présence des évènements (Polling) ralentirait énormément le processeur.

La prise en compte des évènements fugitifs doit être possible, qu'ils soient sur front ou sur niveau.

II.6 - PARTIE MATERIELLE (HARDWARE)

L'aspect matériel de l'automate est modulaire et extensible. Il est constitué par les cartes suivantes :

- La carte maître IDN Process
- 1 Carte interruption (esclave)
- 1 Carte monostable (adaptation)

- 1 Carte sortie relais
- 1 Carte extension mémoire
- 1 Carte extension timers
- 1 Carte extension E/S
- 1 Carte conversion Analogique Digitale (16 voies)
- 1 Carte conversion Digitale Analogique (4 voies)
- 1 Carte processeur arithmétique

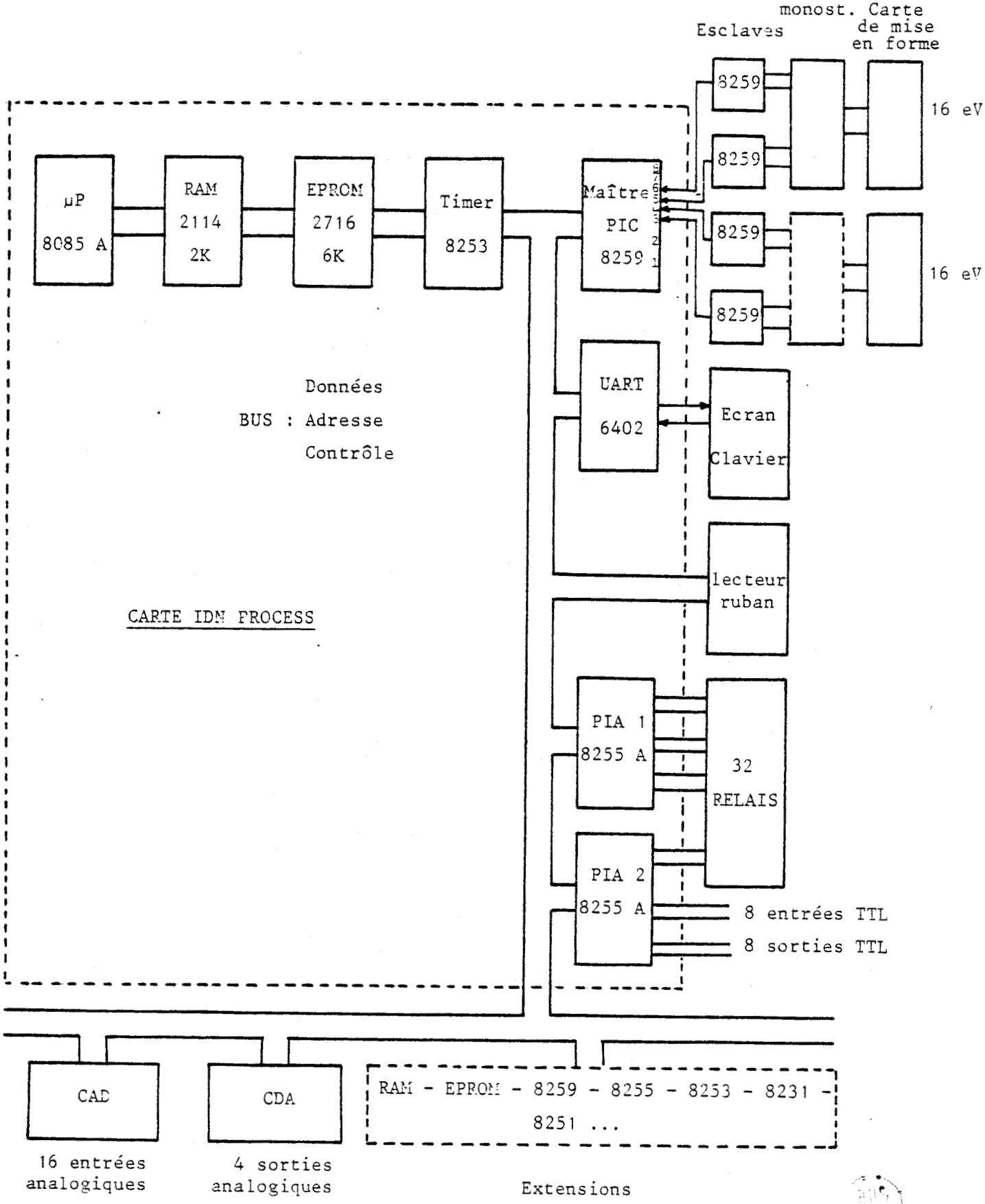
Les périphériques dont on dispose sont un clavier/écran, une liaison série RS 232 (réglage de vitesse - transmission) ou un TTY en boucle de courant, un lecteur de ruban perforé rapide.

D'autres périphériques sont succeptibles d'être raccordés tel que :

- lecteur de cassettes (TU 58)
- Disquettes

La carte IDN comprend sur une carte unique de format SBC :

* 1 Processeur 8085 A de 3 MHz avec des bus complétement bufférisés
* 1 à 2 K de mémoire RAM statiques (2114)
* 6 K de mémoire EPROM (3 * 2716), contenant le moniteur interpréteur
* 48 E/S TTL avec protection (2 * 8255 A)
  USART RS 232
* 1 Entrée série (Clavier/Ecran ou TTY)
* 3 Timers utilisés par l'interpréteur (1 . 8253)
* 3 Entrées d'interruption disponibles (RST 6.5 6.6 6.7)
* 1 Entrée série (RIM) du microprocesseur est disponible
* 1 Circuit de gestion d'interruption programmable (Maître) (8259)
* 1 Connexion DMA
* 1 Entrée pour un clavier hexa décimal
* 1 Connecteur de 86 broches contenant les bus de données d'adresses et de contrôle
* 1 Connecteur de 62 broches disponible pour connecter un clavier hexa décimal
* 3 Connecteurs de 26 broches chacun, correspondant aux timers et PIA.



ARCHITECTURE GENERALE

DE L'AUTOMATE

La carte d'interruption contient 4 circuits de gestion d'interruption 8259. Mis en cascade par rapport au 8259 A maître, ces quatre 8259 A esclaves peuvent gérer 32 interruptions.

Leur mode d'interruption est programmable, dans la phase d'initialisation de l'interrupteur.

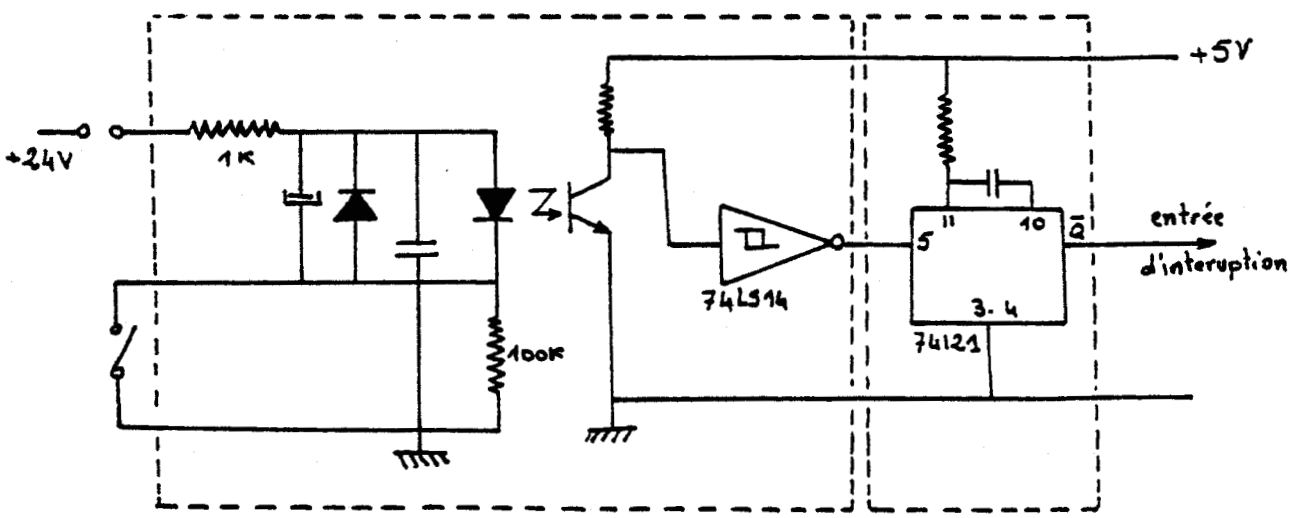
Ils peuvent être programmés pour gérer des interruptions (masquables) aussi bien sur front montant que sur niveau de leurs entrées respectives.

Cette programmation se fait par groupe de 8 niveaux d'interruptions.

La carte monostable optionnelle :

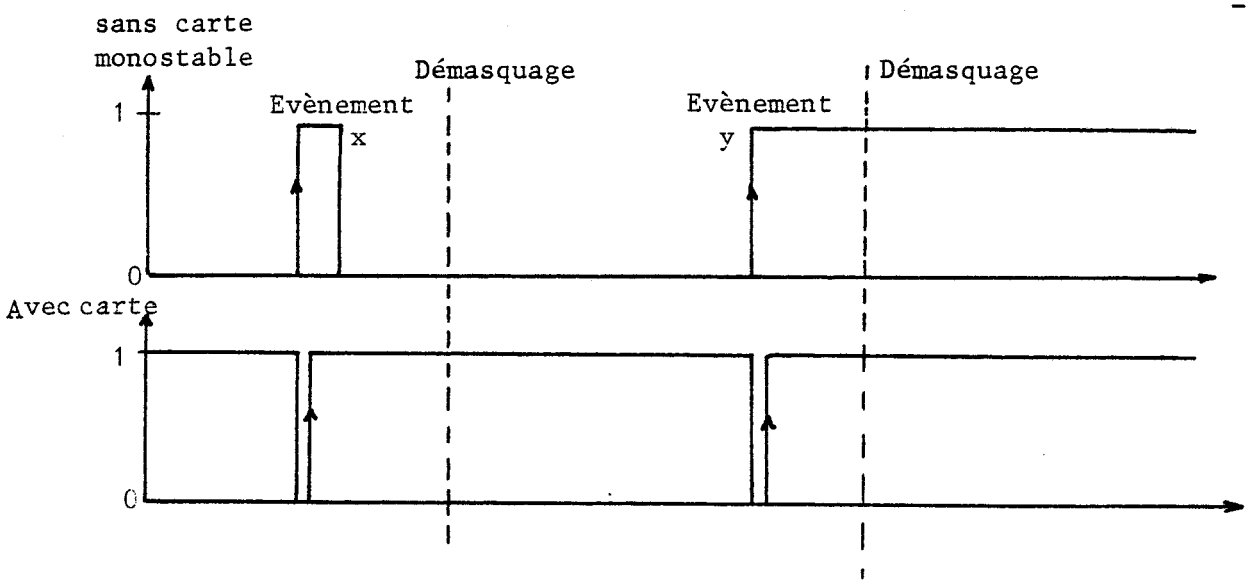
Elle permet la prise en compte d'évènements fugitifs. En effet, les circuits d'interruptions 8259 A ne permettent pas de mémoriser ces interruptions. Cette carte se compose de 16 monostables qui s'introduisent en amont des entrées interruptibles des 8259 esclaves et en aval d'une autre carte de mise en forme des entrées logiques.

La constitution générale pour une entrée décrite sur la figure suivante, comporte une isolation opto-électronique assurant le découpalge du dispositif d'entrée de l'ensemble de l'automate.



CARTE DE MISE EN FORME ET D'ISOLATION

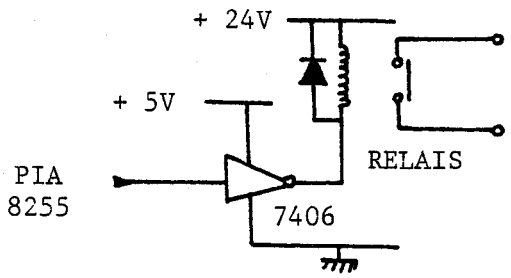
CARTE MONOSTABLE



L'évènement fugitif x n'est pas pris en compte lors de son démasquage si l'on n'utilise pas la carte mono. Il est évidemment traité dans la seconde configuration.

Dans ce cas, l'utilisation du monosatble implique une génération d'interruptions sur front montant.

La carte sortie relais : on dispose des cartes d'interface (12) de sortie pour commander des relais de 24 V.



La carte extension E/S : on dispose de cartes additionnelles pour étendre les entrées-sorties au-delà de 48 des PIA des 2 circuits 8255 A.

La carte d'extension memoire : il existe aussi bien des cartes d'extension mémoire vive à base de 2114 sur chaque 4 K Octets que des cartes d'extension pour des EPROM de 16 K Octets à base de 2716.

La carte extensions timers : on dispose de 3 x 8253 A pour pouvoir faire des comptages et des temporisations externes, nécessaires pour la régulation.

La carte de conversion Analogique Digitale est une carte permettant la saisie de 16 canaux d'entrées analogiques sur 12 bits.

Le gain d'entrée est programmable.

Par contre, la plage de conversion est réglable par cavaliers (± 10 V, de 0 à + 10 V, de 0 à + 5 V, ± 5 V).

La carte de conversion Digitale Analogique (4 canaux) : on dispose de 4 Dacs (8 UP BC) pour conversion sur 8 bits. Le choix de la conversion (en courant ou tension) (0, + 10 V, ± 5 V) se fait par cavaliers.

La carte processeur arithmétique : l'utilisation du processeur 8231 permet d'effectuer des calculs numériques sur des nombres entiers ou en virgule flottante et donc de décharger le processeur de calculs fastidieux et longs.

II.7 - PARTIE LOGICIELLE

Le logiciel est constitué de deux parties bien distinctes :

- Le compilateur/éditeur
- L'interpréteur

II.7.1 - L'édition/compilation

Elle s'effectue (dans la version actuelle) sur un mini-ordinateur PDP 11/34.

La première étape consiste à créer et éditer un programme source décrivant l'automatisme à partir d'un graphe orienté (tel que R d P, Grafcet) et en se servant d'un langage interpréteur de haut niveau détaillé dans le chapitre III.

Après la phase d'édition, apparaît la phase de compilation. Cette dernière permet de détecter les erreurs de syntaxe et d'assemblage et éventuellement d'enregistrer les corrections. Elle aboutit en phase finale à un programme binaire sous deux formes :

- Un listing contennant le détail des différentes instructions utilisées
- Un ruban perforé qui sera chargé sur l'automate

II.7.2 - Interprétation

Une fois le ruban chargé en mémoire de l'automate, le moniteur interpréteur est capable d'interpréter c'est à dire de décoder le langage binaire introduit en paramétrant les sous-routines nécessaires à son exécution.

II.8 - CONCLUSION

Le principe retenu pour la réalisation d'un automate programmé consiste à partager le temps d'un microprocesseur entre différentes tâches évoluant en parallèle. |13|

L'utilisation de processeurs indépendants assurant la gestion des évènements sur le mode prioritaire permet d'obtenir un niveau de performance convenable. En effet, le processeur se trouve dégagé de la surveillance des entrées, ce qui constitue un gain de temps substantiel.

Il reste donc à définir les caractéristiques d'un langage de haut niveau permettant une mise en œuvre aisée de l'automate. Le matériel utilisé assure sans difficulté des fonctions de calcul complexe permettant la prise en compte de données analogiques et le traitement de texte en vue d'applications combinées de gestion et de contrôle fortement intéractif.

CHAPITRE 3

LOGICIEL INTERPRETEUR

III.1 - INTRODUCTION

Le logiciel que nous proposons de décrire dans cette partie a été réalisé selon deux objectifs. Le premier concerne la facilité de mise en œuvre liée essentiellement à l'utilisation systématique de méthodologies de conception d'automatismes basées sur l'utilisation de graphes (R d P, Grafcet). La seconde préoccupation que nous avons conservé tout au long de cette étude, visait à obtenir d'une part une efficacité maximale de traitement sur micro-processeur et d'autre part à permettre, par la création d'un nombre suffisant de primitives, d'assurer la prise en compte de tout type de problèmes de contrôle. Une limite nous est cependant imposée par la taille maximale de l'application prise en compte sur l'ensemble matériel et logiciel proposé dans ce mémoire.

Ce chapitre comporte donc trois parties :

- Dans un premier volet, nous préciserons les différentes primitives réalisées sur le noyau de l'interpréteur.

- Une deuxième partie concerne l'aspect "mise en œuvre" au niveau programmation, et nous présenterons le compilateur.

- Dans une troisième phase, nous présenterons les caractéristiques de l'interpréteur (phase d'exécution).

III.2 - PRIMITIVES

Les primitives élaborées dans ce logiciel visent essentiellement à donner à l'utilisateur la possibilité de programmer plusieurs tâches fictivement simultanées. Dans ce sens, il est donc nécessaire de disposer des 3 types de primitives suivantes :

- Des primitives de calcul, de tests et d'affectations

- Des primitives permettant de réaliser les structures de base de la programmation classique.

- Des primitives assurant la communication entre tâches et la gestion du temps.

III.2.1 - Primitives de calcul, tests et affectations

Il s'agit tout d'abord de permettre le traitement de variables booléennes et nous disposons pour ce faire, des codes instructions de calcul logique : OU, NAND, ...

La calcul arithmétique s'effectue actuellement sur mots de 8 bits sur la base des instructions suivantes : ADD, ...

Les comparaisons sont possibles sur mots de 8 bits et chargent des prédicats.

Les affectations d'entrée ou sortie sont possibles sous les différentes formes associées aux variables prises en compte.

III.2.2 - Primitives de programmation

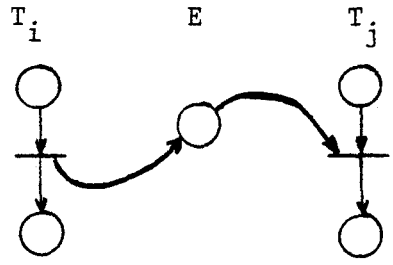
Les instructions de rupture de séquence conditionnelles ou non telles que BRP, TPR, JMP, TES liées aux états des paramètres et/ou prédicats conditionnent l'évolution du pointeur. Elles permettent aussi de réaliser les structures de choix alternatifs. En associant à ces instructions des paramètres incrémentables et testables, il devient possible d'effectuer des boucles sur indice ou sur condition.

Les sous-programmes communs aux différentes tâches sont gérés par les instructions d'appel CPN et de retour RTN.

III.2.3 - Primitives assurant la communication entre tâches et la gestion du temps

La synchronisation des tâches s'effectue ici de manière implicite par l'utilisation d'évènements. La signalisation d'un état prêt pour une

tâche T_i permettant le relancement d'une autre tâche T_j s'effectue simplement en générant un évènement E à partir de T_i et en introduisant une condition de déclenchement pour le même évènement dans T_j.



L'utilisation des ressources communes sera réglée par la mise en œuvre de sémaphores, des instructions de requêtes RQS, de libération RLS et de test TES.

Les tâches déclarées pourront être soit en état inactif, en état actif (attente d'un évènement), ou en état d'exécution (ou en cours).

Les instructions TIN et TEN permettront respectivement de désactiver ou d'activer une tâche à une adresse donnée. La gestion du temps est associée à des compteurs liés à une horloge temps réel fournissant un cycle de base ajustable.

III.2.4 - Autres possiblités

Selon les applications, il doit être possible d'adjoindre aux primitives de base, des fonctions plus sophistiquées d'utilisation courante. Cette possibilité de composition des primitives est possible au niveau du compilateur, c'est notamment le cas pour les communications du type producteur/concommateur.

Des instructions spécifiques d'entrée/sortie peuvent également être créées pour des applications particulières telle la gestion d'un calculateur hybride.

Dans tous les cas, il doit pouvoir être possible d'effectuer des liens avec d'autres programmes déjà assemblés.

III.2.5 - Structures de données

Les données prises en compte dans l'interpréteur sont de 3 types :

- Des paramètres : sur 8 bits
- Des prédicats : sur 1 bit
- Des tableaux à deux indices.

* Les paramètres peuvent représenter des valeurs numériques hexadécimales, décimales, des données d'entrées/sorties, des caractères.

* Les prédicats caractérisent des variables booléennes notamment utilisées en signalisation et en tests.

* Les tableaux à 256 positions mémoires permettent essentiellement des échanges par bloc avec l'extérieur ou entre tâches.

III.3 - STRUCTURE GENERALE DU PROGRAMME UTILISATEUR ET MISE AU POINT DES PROGRAMMES

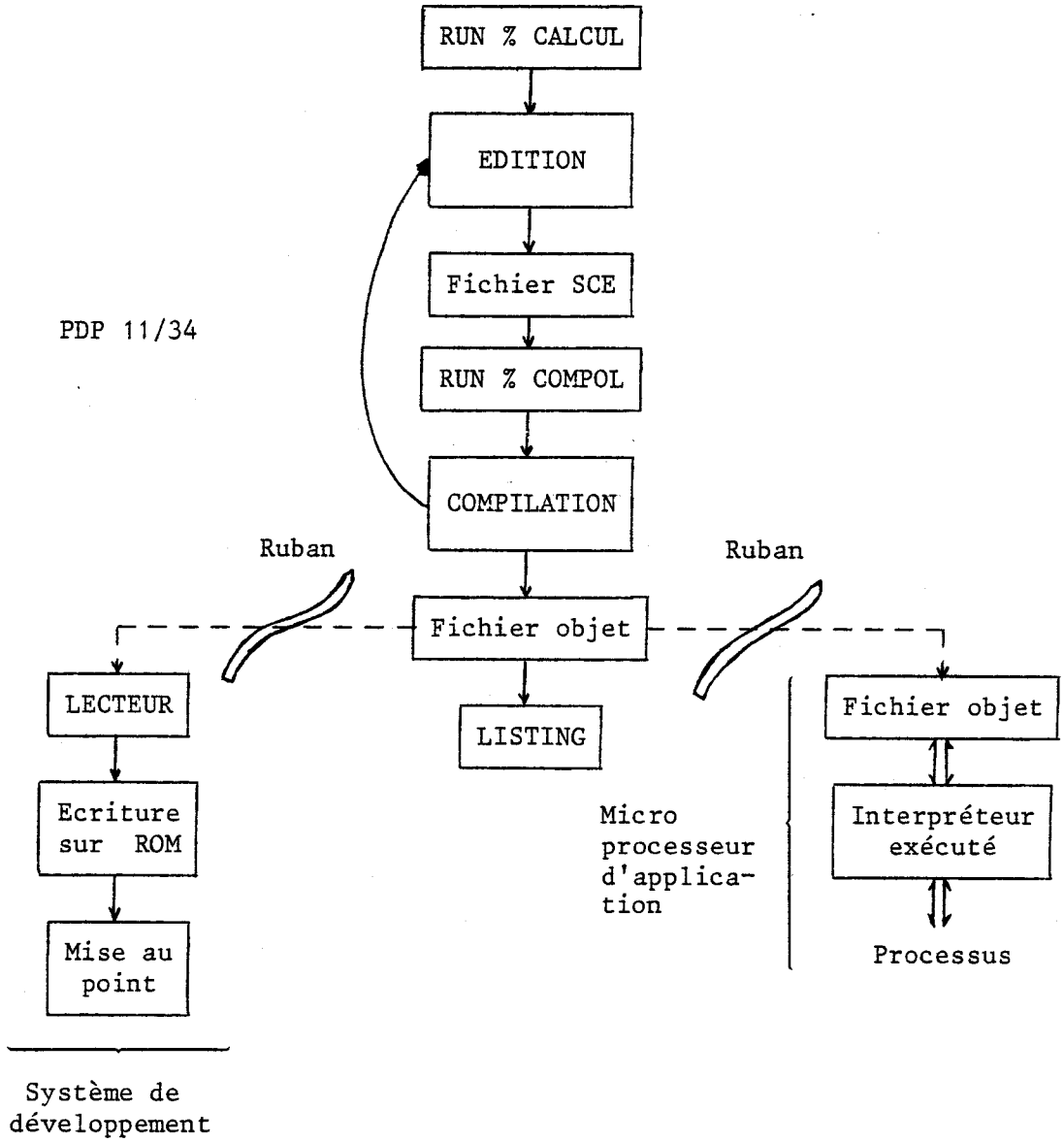
Le programme utilisateur comporte plusieurs tâches (8 maximum). Il conviendra donc de pouvoir écrire dans un premier temps l'ensemble des objets nécessaires à la gestion de ces tâches pour décrire ensuite les actions affectées aux tâches.

La mise au point d'un programme va donc s'effectuer en 3 opérations.

- Tout d'abord un logiciel d'édition (RUN % CALCUL) va permettre la création du fichier SOURCE. Ce travail est effectué de manière performante sur le PDP 11/34.

- Le programme SOURCE est ensuite analysé et compilé sur le même calculateur fournissant un code objet destiné à paramètrer les différentes fonctions de l'interpréteur et à créer les tables nécessaires.

- Le code objet est ensuite transmis au système de développement ou au microprocesseur lui-même par l'intermédiaire d'un support ruban perforé. Il peut alors être exécuté par l'interpréteur résidant en mémoires RAM.



PROCEDURE D'EDITION ET
D'EXECUTION D'UN PROGRAMME SOURCE

III.4 - SYNTAXE GENERALE D'UN PROGRAMME UTILISATEUR

Un programme utilisateur est composé de deux parties distinctes :

- Les déclarations
- Les instructions (desciption des tâches).

III.4.1 - Les déclarations

Cette partie est uniquement composée de directives. Les directives possibles sont au nombre de 7. Deux seulement sont obligatoires. Chaque ligne de directive ne peut contenir qu'une seule directive.

Cette partie est close par la directive obligatoire *DEB (8ème directive possible, cf § Directive *DEB).

III.4.1.1 - Directive PRO

Elle est obligatoire et doit être la première des directives à figurer.

Elle déclare les différentes tâches du programme. Sa structure est la suivante :

PRO ␣ <nom tâche> {, <nom tâche>}$_{0}^{7}$

où <nom tâche> est une chaîne alphanumérique non vide d'au plus 6 caractères. Le nombre de tâches doit être de 1 au moins, 8 au maximum.

Ex : PRO TOTO, QWERT

III.4.1.2 - Directive INI

Elle est obligatoire et placée après PRO.

Elle donne la liste des tâches à activer au lancement du programme et reprend donc tout ou partie des tâches déclarées dans PRO :

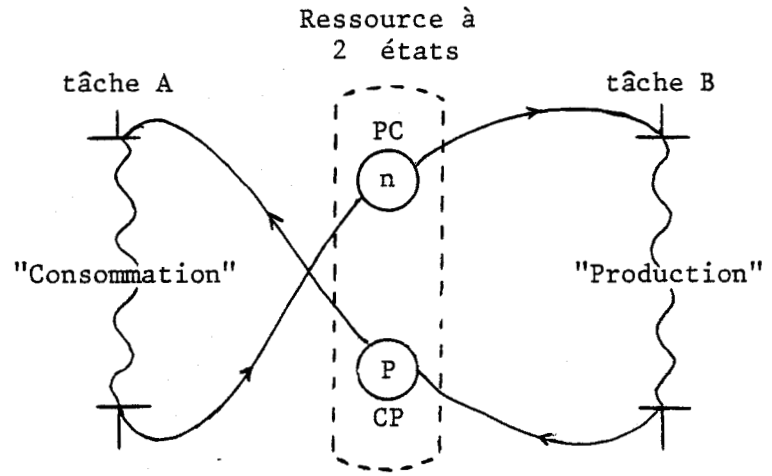
INI ␣ <nom tâche> {, <nom tâche>}$_{0}^{7}$

Ex : INI TOTO

III.4.1.3 - Directive P.C

Elle est optionnelle et placée après PRO.

Elle déclare les ressources (à 2 états) partagées en Production/ Consommation. Le schéma de Pétri équivalent introduit deux sémaphores. Ces sémaphores sont générés automatiquement par le compilateur.



La directive PC a la structure suivante :

P.C ␣ <nom ressouce> {(n ; p)} , {<nom ressource> {(n ; p)}} 7 0

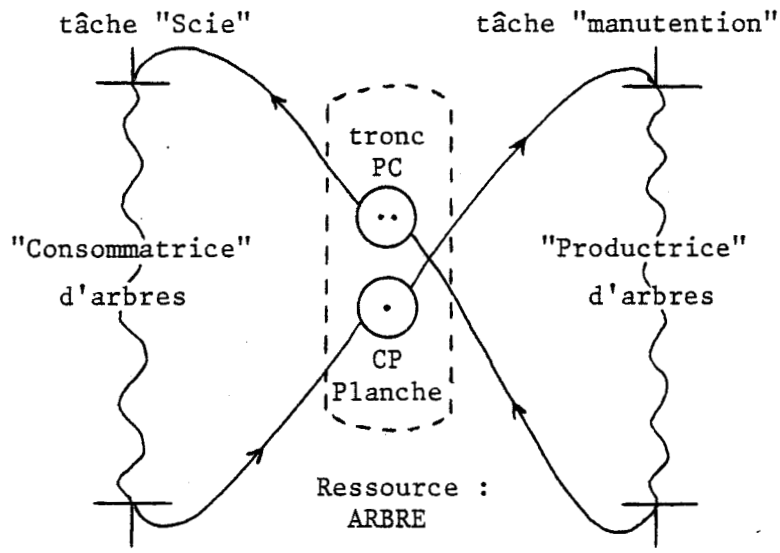
où <nom ressource> est une chaîne alphanumérique non vide, d'au plus 6 caractères. Huit ressources au maximum sont autorisées (cf § Directive SEM).

n et p sont des entiers de |0 , 255| et donnent les valeurs initiales des 2 sémaphores associées PC et CP avec :

n = V.I. du sém. PC (Prod. → Cons.) (1 par défaut)
p = V.I. du sém. CP (Cons. → Prod.) (1 par défaut)

Ex : Scie - Manutention des arbres

.../...



avec PC ARBRE (2 ; 1)

- La tâche SCIE étant considérée comme consommatrice, va consommer la ressource "ARBRE" dans son état "TRONC" (elle va aussi la produire sous la forme "PLANCHE").

Les valeurs initiales des deux sémaphores sont alors données sous la forme : (Tronc ; Planche).

III.4.1.4 - Directive SEM

Elle est optionnelle et placée après PRO.

Elle déclare les sémaphores utilisés dans le programme.

SEM ␣ <nom sém.> {= n} {, <nom sém.> {= n}} (15 / 0)

où <nom sém.> est une chaîne alphanumérique non vide, d'au plus 6 caractères. Les formes commençant par PC ou CP sont interdites pour les <nom sém.>. 16 sémaphores au maximum sont autorisés.

n = V.I. du sémaphore, entier de |0 , 255|
n = 1 par défaut

Ex : SEM RATE = 4, HOURA

Remarque importante : Le nombre total de sémaphores est limité à 16. Ceci comprend les sémaphores déclarées (SEM) et ceux générés implicitement par le compilateur (PC : 2 par ressource déclarée).

III.4.1.5 - Directive DIM

Elle est optionnelle et placée après PRO.

Elle déclare les tables utilisateur.

DIM <nom table> (m) {, <nom table> (m)}$_0^7$

où <nom table> est une chaîne alphanumérique non vide, d'au plus 6 caractères. 8 tables au maximum sont autorisées.

m = longueur de la table : entier de |1 , 32767|

Ex : DIM A$ (20) ; BIC (42) ; T (3)

N.B. : T contient 3 éléments : T(1), T(2), T(3).

III.4.1.6 - Directive HOR

Elle est optionnelle et placée après PRO.

Elle déclare la fréquence utilisée dans les temporisations figurant dans le programme.

HOR ␣ n

n est un entier de |4 , 250.000| représentant, en Hz, la fréquence désirée n doit être un sous-multiple de 250.000 Hz.

Ex : HOR 250 (f = 250 Hz)

Si la ligne HOR ne figure pas dans les déclarartions, le compilateur fait n = 4, soit f = 4 Hz.

III.4.1.7 - Directive FNC

Elle est optionnelle et placée après PRO.

Elle déclare une fonction d'évènements qui pourra servir de code déclenchement dans le programme. Cette fonction est construite à base de "ou" (inclusifs) :

$$\text{év}^t \text{ i "ou" év}^t \text{ j "ou" év}^t \text{ k}$$

Elle a la structure suivante :

FNC ␣ Fi = a + b + ... + z

où i est le numéro de la fonction déclarée. C'est un entier de $|0\ ,\ 7|$.

(a, b, ..., z) est un sous-ensemble des entiers de $|1\ ,\ 32|$ dans un ordre quelconque et sans répétition : seuls les évènements 1 à 32 sont autorisés pour une fonction.

Remarque : Cette directive est répétée sur autant de lignes qu'il est nécessaire pour définir toutes les fonctions souhaitées. Le nombre de ces fonctions est limité à 8.

```
Ex : FNC FO = 31 + 4 + 14 + 1
     FNC F7 = 1 + 12
     FNC F2 = 9 + 8
```

*III.4.1.8 - Directive *DEB*

Elle est obligatoire.

Elle clot la partie des déclarations. Toute déclaration décrite précédemment est interdite à partir de cette directive. Elle s'écrit :

```
*DEB
```

Remarque : Mis à part PRO et *DEB dont les places relatives sont fixées,

les directives décrites jusque-là peuvent figurer dans un ordre quelconque.
Des commentaires peuvent figurer à la fin de chaque ligne après le caractère "!" et la fin de ligne n'est pas prise en compte par le compilateur.

```
Ex : PRO TOTO, QWERT
     P.C ARBRE (2 ; 1)    ! Prod./Cons. Arbre
                          Pro. = 2     Cons. = 1
     INI TOTO
     FNC F2 = 9 + 8
     SEM RATE = 4, HOURA ! 2 sémaphores RATE et HOURA
     DIM A$(20)           ! TABLE à 20 éléments
     HOR 4
     FNC F0 = 31 + 4 + 14 + 1
     *DEB                 ! Fin des déclarations
```

III.4.2 - Les lignes-instructions

Cette partie est composée de directives et d'instructions. Chaque ligne ne peut contenir qu'une directive ou qu'une instruction.

Les commentaires sont possibles en fin de ligne après le caractère "!".

III.4.2.1 - Directive BEGIN

Elle est obligatoire.

Elle délimite le début de description d'une tâche. Elle doit donc figurer au moins une fois : à la ligne suivant *DEB pour délimiter la première (et peut-être unique) tâche du programme.

BEGIN ␣ <nom tâche>

où <nom tâche> doit être l'une des tâches données dans PRO.

Ex : BEGIN QWERT

*III.4.2.2 - Directive *FIN*

Elle est obligatoire.

Elle délimite la fin du programme, c'est à dire la fin des lignes prises en compte par le compilateur. Elle s'écrit :

*FIN

III.4.2.3 - Les lignes-instructions

Elles constituent la description des tâches.

Elles ont la structure suivante :

$$\{\text{ETI :}\}\ \{(\text{COD})\}\ \text{COP}\ \llcorner\!\lrcorner\ \{\text{opérande}\}\ \{\text{,opérande}\}_{0}^{i-1}\ \ \{\text{!commentaire}\}$$

où ETI est optionnelle : étiquette de 6 caractères alphanumériques au plus.
COD est optionnelle : c'est le code de déclenchement. Il figure entre parenthèses et peut être :
- un évènement : Ei, $i \in |1\ ,\ 63|$ Ex : (E3)
- un prédicat : PRi, $i \in |1\ ,\ 63|$ Ex : (PR35)
- une fonction : Fi, $i \in |0\ ,\ 7|$ Ex : (F2)

COP est obligatoire : c'est le code de l'instruction (cf liste des codes possibles)
Opérandes : précise ici l'ensemble des paramètres nécessaires à l'exécution de l'instruction. Ils sont séparés par des virgules
{i = nombre d'opérandes nécessaires}

Ex : ETI1 (F4) RQS RATE, P6 ! REQUETE DE SEMAPHORE

Sur présence de la fonction d'évènement F4, effectuer la requête du sémaphore RATE avec la priorité affectée à la valeur du paramètre P6.

III.4.3 - Les opérandes : structure détaillée

Suivant l'instruction utilisée, les opérandes peuvent être définis comme suit :

a) évènement : soit Ei avec $i \in |1\ ,\ 55|$ Ex : E55
défini sur byte

b) Prédicat : soit PRi avec i ∈ |1 , 63| Ex : PR4
défini sur bit

c) Paramètre : soit Pi avec i ∈ |1 , 55| Ex : P24
défini sur 1 Octet

d) Sémaphore : soit un <nom sémaphore> existant dans la directive SEM
Ex : RATE

e) Tâche : soit un <nom tâche> existant dans la directive PRO
Ex : TOTO

f) Port d'entrée-sortie (In-Out) : soit IOi avec i ∈ |0 , 63|
Ex : IO41

g) Numéro du bit sélectionné : soit n ; n étant un nombre entier
n ∈ |1 , 8|
Ex : 7

h) Table : soit un <nom table> existant dans la directive DIM
Ex : A$

Rappel : un élément de table occupe 8 bits

i) Etiquette : Nom étiquette : présent au début de l'une des lignes-instructions (séparateur) Ex : ETI1

j) Priorité : elle est associée à un sémaphore (simple ou Prod./Cons.)
Elle peut être de deux formes :
- entière de |0 , 127| Ex : 104
- paramètre de P1 à P63 Ex : P45

k) Valeur sur 1 Octet : Cette valeur est entière et peut se présenter sous deux formes :
- Décimale ∈ |-128 , 127| Ex : 104
- Hexadécimale avec le caractère "/" suivi d'une valeur ∈ |0 , FF| Ex : /A3

1) Valeur sur 2 Octets : Cette valeur est entière et peut se présenter sous deux formes :

- Décimale $\in$ |-32768 , 32767|
  Ex : -4651
- Hexadécimale avec le caractère "/" suivi d'une valeur $\in$ |0 , FFFF|
  Ex : /F8B6

Remarque : Le choix d'écriture d'une valeur en décimal ou hexadécimal détermine le placement de l'intervalle de valeurs permises (suivant le nombre d'Octets) :

à partir de 0 en hexadécimal
centré sur 0 en décimal.

III.5 - LES INSTRUCTIONS DISPONIBLES

III.5.1 - Introduction

Un programme utilisateur contient deux types d'instructions :

- Les instructions que comprend l'interpréteur

et dont la liste, suivant le calculateur, est donnée dans les pages suivantes.

- Les macro-instructions qui sont remplacées, au niveau du compilateur, par des séquences d'instructions du premier type.

III.5.2 - Liste des instructions

Les paramètres sont définis sur 8 bits, les prédicats sur 1 bit ; les tables concernent des données sur 8 bits.

III.5.2.1 - Instructions de chargement

a) LDP Paramètre Valeur : Chargement du paramètre avec la valeur (/hexa décimale) ou décimale.

Ex 1 : LDP P12,/FA ! FA → P12

Ex 2 : LDP P20, 13 ! 13 → P20

b) LPR PREDICAT : Mettre le prédicat à 1

Ex : LPR PR15

c) CPR PREDICAT : Mise à zéro du prédicat

Ex : CPR PR15

d) CLP PARAMETRE : Mise à zéro du paramètre

Ex : CLP P12

e) IMP PARAMETRE : Incrémentation du paramètre

Ex : IMP P12

III.5.2.2 - Instructions de calcul logique

a) AND Paramètre i, Paramètre j : Opération ET entre Paramètre i et Paramètre j - Le résultat est rangé dans le Paramètre i.

Ex : AND P11, P12

b) OR Paramètre, Paramètre : Opération ou logique

c) XOR Paramètre, Paramètre : OU exclusif

d) NAN Paramètre, Paramètre : NAND

e) NOR Paramètre, Paramètre : Fonction NI

f) SL Paramètre j : Décalage ouvert d'un pas à gauche du paramètre j.
Le résultat est remis dans le paramètre j.

Ex : SL P12

g) SLC Paramètre j : Décalage ferme d'un pas à gauche du paramètre j.

h) SR Paramètre j : Décalage à droite ouvert d'un pas.

i) SRC Paramètre j : Décalage à droite fermé d'un pas.

III.5.2.3 - Calcul arithmétique

ADD Paramètre i, Paramètre j : Addition sur 1 Octet du paramètre i au paramètre j. Le résultat sur 1 Octet est rangé dans paramètre i.

Ex : ADD P12, P11

III.5.2.4 - Comparaison

a) CME PREDICAT $_k$, Paramètre i, Paramètre j : Mise du prédicat k à 1 si le paramètre i est égal au paramètre j.

Ex : CME PR11, P11, P12

b) CMG PREDICAT $_k$, Paramètre i, Paramètre j : Mise à 1 du prédicat k si :
Paramètre i > Paramètre j

c) CML Prédicat $_k$, Paramètre i, Paramètre j : Mise à 1 du prédicat k si :
Paramètre i < Paramètre j

d) CMN Prédicat $_k$, Paramètre i, Paramètre j : Mise à 1 du prédicat k si :
Paramètre i ≠ Paramètre j

e) CMM Prédicat $_k$, Paramètre i : Mise à 1 du prédicat k si le paramètre i est = < 0

Ex : CMM PR10, P23

f) CMP Prédicat $_k$, Paramètre i : Mise à 1 du prédicat k si le paramètre i est > = 0

g) CMZ Prédicat $_k$, Paramètre i : Mise à 1 du prédicat k si le paramètre i est = 0

III.5.2.5- Dialogue avec un clavier/écran

a) IT Paramètre i : Entrée du paramètre i à partir du clavier (Code caractère)

b) OT Paramètre i : Sortie du paramètre i sur l'écran.

III.5.2.6 - Instructions de branchement

Ces sauts doivent être à l'intérieur d'une même tâche.

a) JMP ETIQUETTE : Branchement inconditionnel à l'étiquette.

Ex : JMP TROIS

b) BRP ETIQUETTE, Paramètre : Branchement à l'étiquette si le paramètre est nul, sinon continuer la suite des instructions.

c) TPR ETIQUETTE, Prédicat : Branchement à l'étiquette si le prédicat est nul, sinon continuer.

d) TES ETIQUETTE, Sémaphore : Branchement à l'étiquette si le sémaphore est nul.

III.5.2.7 - Instructions de parallèlisme

a) TEN TACHE i, ETIQUETTE : Lancer ou relancer la tâche à l'étiquette précisée. Si l'étiquette est absente, la tâche i est relancée à l'adresse actuelle.

Ex 1 : TEN COURSE, FEU

Ex 2 : TEN COURSE

b) TIN TACHE i : Inhibition de la tâche i. La tâche i est annulée dans la table des tâches actives.

Ex : TIN COURSE

III.5.2.8 - Sous-programme

a) CPN ETIQUETTE : Appel d'un sous-programme par son étiquette. Ce sous-programme ne doit pas contenir de codes de déclenchements. Ceci n'est possible que s'il est appelé par la même tâche ou bien par des tâches qui s'excluent mutuellement.

b) RTN : Dernière instruction d'un sous-programme assurant le retour automatique au programme appelant.

III.5.2.9 - Commandes d'entrées/sorties

a) OPØ ENTREE/SORTIE, SELECTION du bit i :

Mise à zéro du bit i sélectionné dans le port de sortie.

Ex : OPØ IO2,3 : Mise à zéro du bit 3, du port 2.

b) OP1 ENTREE/SORTIE, SELECTION du bit i :

Mise à zéro du bit i sélectionné dans le port de sortie.

Ex : OP1 IO1,7

c) OPV ENTREE/SORTIE, VALEUR :

Sortie de la valeur (hexadécimale si précédée par /) ou décimale sur le port de sortie.

Ex : OPV IO3, /FE ! Sortie de la valeur Hexa FE sur le port n° 3.

d) OPP ENTREE/SORTIE, Sélection du bit, Paramètre :

Sortie de 1 sur le bit sélectionné du port de sortie si le paramètre n'est pas nul.

Ex : OPP IOØ, 2, P12

e) OP ENTREE/SORTIE, Paramètre :

Sortie du paramètre sur le port de sortie.

Ex : OP IOØ, P13

f) IP ENTREE/SORTIE, Paramètre :

Entrée du paramètre par l'intermédiaire du port d'entrée.

Ex : IP IO1, P21

III.5.2.10 - Sous-programme écrit en langage machine

EXT Valeur/2 Octets : permet un branchement à un sous-programme écrit en langage machine. Le retour au programme moniteur se fait par JMP RETØ.

III.5.2.11 - Traitement sur évènement

a) ENB EVENEMENT, PREDICAT :

Le code de l'évènement en prédicat représente une mémorisation de l'évènement.

Ex : ENB E30, PR16

Dès l'arrivée de l'évènement 30, le prédicat 16 est mis à 1.

b) MSQ EVENEMENT :

Masque l'évènement et efface le codage de celui-ci en prédicat.

Ex : MSQ E20

c) REE EVENEMENT :

Permet le refus d'un évènement - après son démasquage -

Dès son arrivée, il est pris en compte (remasqué) mais ne sera pas traité.

Ex : REE E30

Remarque : L'utilisation de REE suivi de MQS permet d'effacer la mémorisation des évènements fugitifs.

III.5.2.12 - Instructions de mesures (sur 16 bits)

a) MES __ Table, Valeur 1, Valeur 2 :

Permet la mesure du canal précisé par Valeur 2. Le résultat est rangé dans la table à l'indice Valeur 1.

Ex : MES LEFT, 2, 5

Le canal 5 est converti et rangé à l'adresse 2 de la table LEFT.

b) MTC Table, Valeur 1, Valeur 2 :

Mesure du canal (Valeur 1) jusqu'au canal (Valeur 2). Le résultat des conversions sont rangés dans TABLE.

Ex : MTC RIGHT, 2, 6

RIGHT (1) ← Canal 2

RIGHT (3) ← Canal 3

⋮

RIGHT (9) ← Canal 6

c) CC Table, Valeur 1, Valeur 2 ; (Format de sortie sur 8 bits) :

Conversion Digitale Analogique de l'élément de la table avec l'indice (Valeur 1) sur le canal (Valeur 2).

Ex : CC RIGHT, 8, 3

Sortie de RIGHT (8) sur le canal 3

d) CTC Table, Valeur 1, Valeur 2 :

Conversion Digitale/Analogique des 1ers éléments de table.
Valeur 1 indique le 1er canal sur lequel table 1 doit être convertie
Valeur 2 indique le dernier canal.

Ex : CTC RIGHT, 3, 6 Sortie RIGHT (1) sur canal 3

Sortie RIGHT (3) sur canal 4

⋮

Sortie RIGHT (7) sur canal 6

e) S.C.C Table, Valeur

Sortie convertie en Volt, visualisée sur écran de l'élément de la table avec indexation par l'indice "Valeur". S'il est nul, toute la table est visualisée.

Ex : SCC LEFT, 3

Sortie de la valeur convertie sur l'écran de l'élément LEFT (3).

f) DAM TABLE, Valeur 1, Valeur 2

Sortie de l'élément (Valeur 1) de la table sur le DAM (Digital Analog Multiplier) spécifié par l'index de la Valeur 2.

Ex : DAM LEFT, 10, 4

Sortie de LEFT (10) sur DAM 4

III.5.2.13 - Utilisation de table

a) ETT TABLE 1, TABLE 2

Echange du contenu des deux tables. Elles doivent être de même longueurs spécifiées dans la directrice DIM.

b) FET TABLE

Mise à zéro des éléments de la table.

Ex : FET RIGHT

c) OTT TABLE

Visualiser sur l'écran les éléments de la table.

Ex : OTT RIGHT

d) ITP Paramètre, Valeur, TABLE

Entrée du paramètre dans l'élément de la table Valeur comme indice de la table. 1 < Valeur < Longueur de la table.

Ex : ITP P12, 13, RIGHT

Entrée du paramètre 12 dans la table RIGHT (13)

e) ETP Paramètre i, Paramètre j, Table

Instruction analogue à la précédente où l'indice (Valeur 1) est remplacé par Paramètre j, c'est à dire entrée du paramètre i dans l'élément table (Paramètre j).

Ex : ETP P13, P12, RIGHT

Le paramètre 13 est rangé dans RIGHT (P12)

f) OTP Paramètre, Valeur, Table

Ranger le contenu de l'élément (Valeur) de la table dans le paramètre.

Ex : OTP P12, 14, LEFT

L'élément LEFT (14) est rangé dans le paramètre 12.

III.5.2.14 - Commande d'un calculateur analogique

a) ATT

Instruction d'attente, du calculateur analogique.

b) VI Valeur

Cette instruction déclenche la mise en mode condition initiale.

c) CAL Valeur

Instruction de mise en calcul. Valeur indique le numéro du RACK. S'il est nul, la commande est générale.

III.5.2.15 - Instructions opérant sur les sémaphores

RQS SEMAPHORE PRIORITE

C'est une demande de sémaphore avec priorité (variable de 0 à 255) lors de demandes simultanées de plusieurs tâches pour le même sémaphore.

La priorité peut être directe, sur une valeur ; ou paramètrée par l'intermédiaire d'un paramètre. Si elle est absente, la priorité est nulle.

Ex 1 : RQS BRAS : Demande du Bras avec priorité 0

Ex 2 : RQS BRAS, 7 : Demande du Bras avec la priorité 7

Ex 3 : RQS BRAS, P12 : Demande du Bras avec la priorité contenue dans le paramètre 12.

III.5.2.16 - Diverses instructions

a) NOP : ne rien faire

b) HLT : Arrêt du programme et retour au moniteur - utilisée en cas de détection d'erreur

c) END : Instruction de fin normale de tâche

d) SET : Instruction permettant de visualiser l'état des tâches sur l'écran et de savoir quel code de déclenchement elles attendent.

e) ETA Valeur : Permet de prendre l'état des appels des évènements extérieurs et les ranger dans les 5 premiers paramètres (de P1 à P5). Si la valeur est différente de 0, leur visualisation est faite sur l'écran.

III.5.3 - Macro-instruction

III.5.3.1 - Temporisation

C'est une instruction de la forme : TEMPi ␣ n

où i entier de |0 , 7| précise le numéro de la temporisation.
n est le nombre d'impulsions d'horloge attendues avant continuation du programme. (La fréquence d'horloge a été définie dans la déclaration HOR).

Le compilateur remplace l'instruction {TEMP} par la séquence des deux instructions suivantes :

```
LDP Pk, n
(Ek) NOP
```

où k = i + 56 ; la temporistaion utilise les paramètres et évènements 56 à 63.

l'évènement attendu Ek est associé au passage à 0 du paramètre Pk.

Ex : TEMP 3 50 ; Temporisation n°3 : attendre 50 impulsions d'horloge avant de continuer la suite des instructions.

III.5.3.2 - Production-Consommation

Les instructions de Production-Consommation sont au nombre de 4 :

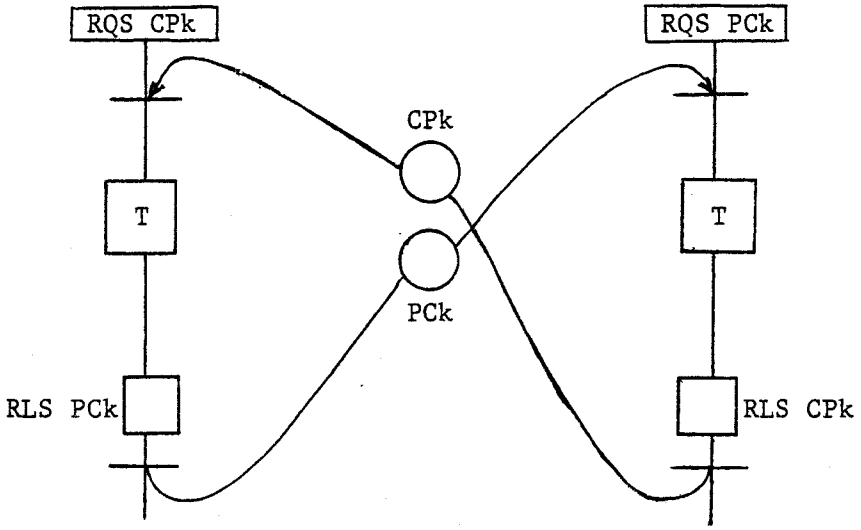
- P.D <ressource>, priorité : début de production
- P.F <ressource> : fin de production
- C.D <ressource>, priorité : début de consommation
- C.F <ressource> : fin de consommation

où <ressource> est un nom de ressource figurant dans la directive P.C.
priorité est un paramètre (P1 à P55) ou une valeur entière de |0 , 127|

Ces 4 instructions sont remplacées dans le compilateur par : respectivement :

```
RQS CPk, priorité
RLS CPk
RQS PCk; priorité
RQS PCk
```

où k est l'indice -ou le rang) de déclaration de la ressource dans la directive P.C (cf P.C).



III.6 - COMPILATION

Lorsque le programme source est généré par le programme RUN % Calcul, ce dernier crée un "fichier - Programme" avec extension ASC

Ex : TESTE ASC

Il suffit alors de lancer la compilation sur le PDP 11/34 par l'appel du programme % Compol.

RUN % Compol. (voir annexe)

La compilation conduit à la détection des erreurs de syntaxe et sémantique.

Certaines erreurs peuvent être corrigées immédiatement, la compilation continue dans ce cas. Par contre, les erreurs fatales arrêtent la compilation.

Si le programme compilé ne présente pas d'erreur, un listing de compilation sera disponible ainsi qu'un ruban perforé contenant le code objet interprétable.

III.7 - DEROULEMENT DE L'INTERPRETATION

L'interpréteur est supposé implanté en mémoire du microcalculateur d'exécution. L'utilisation de la clef "?" permet l'entrée du code objet en mémoire RAM. L'état initial du calculateur et de son processus l'environnant, peut être détecté par la clef I.

L'exécution de l'interpréteur peut s'effectuer selon 4 modes choisis par l'utilisateur, après l'appui sur la touche à :

- pas à pas
- macro-instruction par macro-instruction
- arrêt sur adresse
- exécution en continu

Le déroulement de l'interpréteur s'effectue selon l'organigramme général décrit sur la figure 7 et assure la gestion séquentielle des différentes tâches en prenant en compte les évènements enregistrés et attendus au niveau du code de déclenchement de chaque macro-instruction.

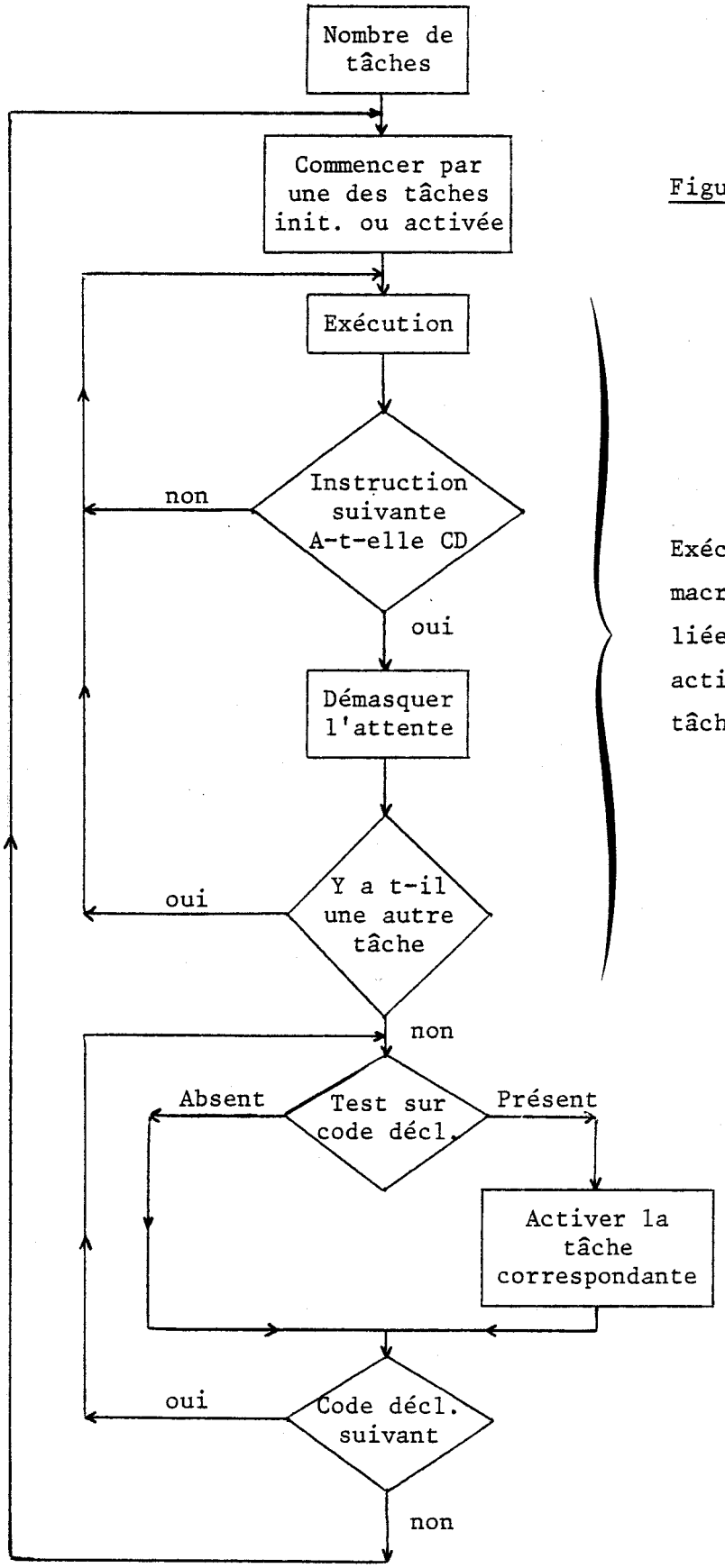


Figure 7

ORGANIGRAMME SIMPLIFIE DE L'INTERPRETEUR
GESTION SEQUENTIELLE DES DIFFERENTES TACHES

L'organigramme décrit sur la figure 8 précise le détail de l'organisation précédente. Notamment il indique l'utilisation des tâches suivantes :

EVE : état des évènements
TTR : liste des états des tâches
TAC : liste des tâches en attente d'un code de déclenchement
CD : codes des déclenchements associés aux tâches en attente
PPRG : pointeurs de tâches
PRIØ : liste des priorités associées aux tâches lors d'une requête d'un même sémaphore
POINTO : table utilisée pour déterminer la tâche la plus prioritaire
POINTS : table des tâches associées aux sémaphores
ETAT : table des états des sémaphores
MAS : masques des (PIC) 8259A (le maître et les esclaves)
PORT : données des ports d'E/S
PARA : table des paramètres
PRE : table des prédicats
FEV : table des évènements associés aux fonctions
REFUS : table des évènements refusés
DEVEFS : liste des fonctions en attente des évènements.

Le détail des tables est donné en annexe.

Les évènements traités sont remis à zéro et la mise à jour des réceptivité est effectuée après chaque balayage d'exécution.

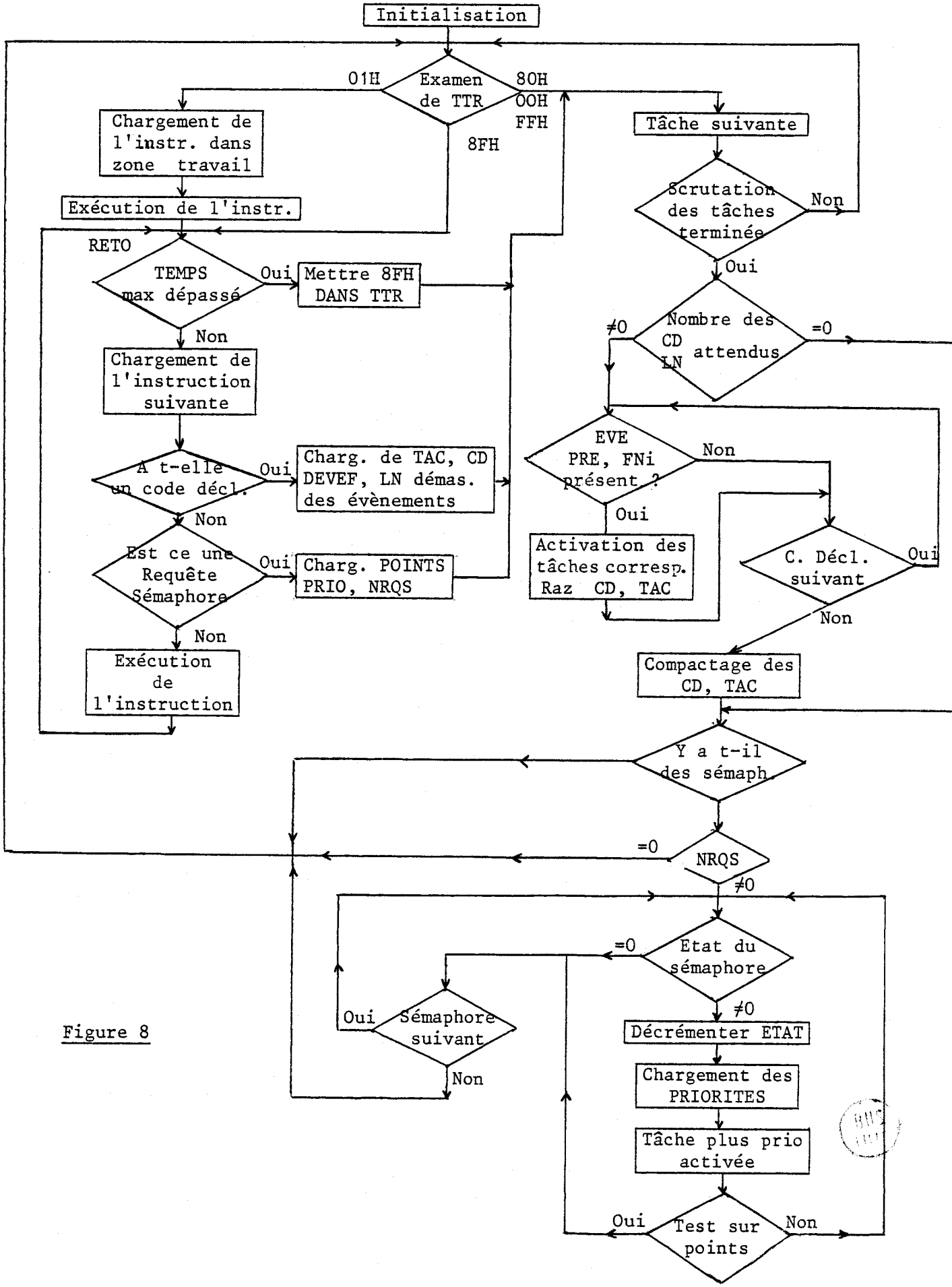

Initialisation
Examen de TTR
01H
80H
00H
FFH
8FH
Chargement de l'instr. dans zone travail
Exécution de l'instr.
RETO
TEMPS max dépassé
Oui
Mettre 8FH DANS TTR
Non
Chargement de l'instruction suivante
A t-elle un code décl.
Oui
Charg. de TAC, CD DEVEF, LN démas. des évènements
Non
Est ce une Requête Sémaphore
Oui
Charg. POINTS PRIO, NRQS
Non
Exécution de l'instruction
Tâche suivante
Scrutation des tâches terminée
Non
Oui
Nombre des CD LN attendus
≠0
=0
EVE PRE, FNi présent ?
Non
Oui
Activation des tâches corresp. Raz CD, TAC
C. Décl. suivant
Oui
Non
Compactage des CD, TAC
Y a t-il des sémaph
NRQS
=0
≠0
Etat du sémaphore
=0
≠0
Sémaphore suivant
Oui
Non
Décrémenter ETAT
Chargement des PRIORITES
Tâche plus prio activée
Test sur points
Oui
Non

Figure 8

Enfin, l'organigramme décrit ci-après décrit le traitement des interruptions ou évènements (Figure 9).

L'interruption n°1 correspond à l'horloge, elle permet la gestion programmée des évènements de temporisation E56 à E63 liés aux paramètres de même indice.

Lorsqu'un évènement est présent, il est, soit chargé dans la table EVE avec le code 80H, soit mis à 1 dans la table PRE mémorisant les évènements dans le prédicat de même rang.

La mise en œuvre de circuits maître-esclave 8259 permet de traduire simplement la notion de réceptivité par le masquage sélectif des entrées d'interruption (tables MAS), il s'agit alors d'un masquage utilisant le matériel.

Lorsqu'une tâche rencontre un code de déclenchement correspondant à un évènement ou une association d'évènements (fonctions), ces évènements sont démasqués sur les 8259.

Dès leur arrivée, on teste s'ils sont refusés (dans la table REFUS) ; si OUI, sans les traiter, on remet à jour la table REFUS. Par contre, s'ils sont acceptés, ils sont mémorisés temporairement dans EVE ou/et dans FNC s'ils sont demandés par une fonction. Ils sont ensuite masqués au niveau des 8259.

Après la fin des scrutations séquentielles des tâches, on active celles dont le code de déclenchement est satisfait (en même temps, on remet à 0 les éléments correspondants acquittés dans EVE et FNC).

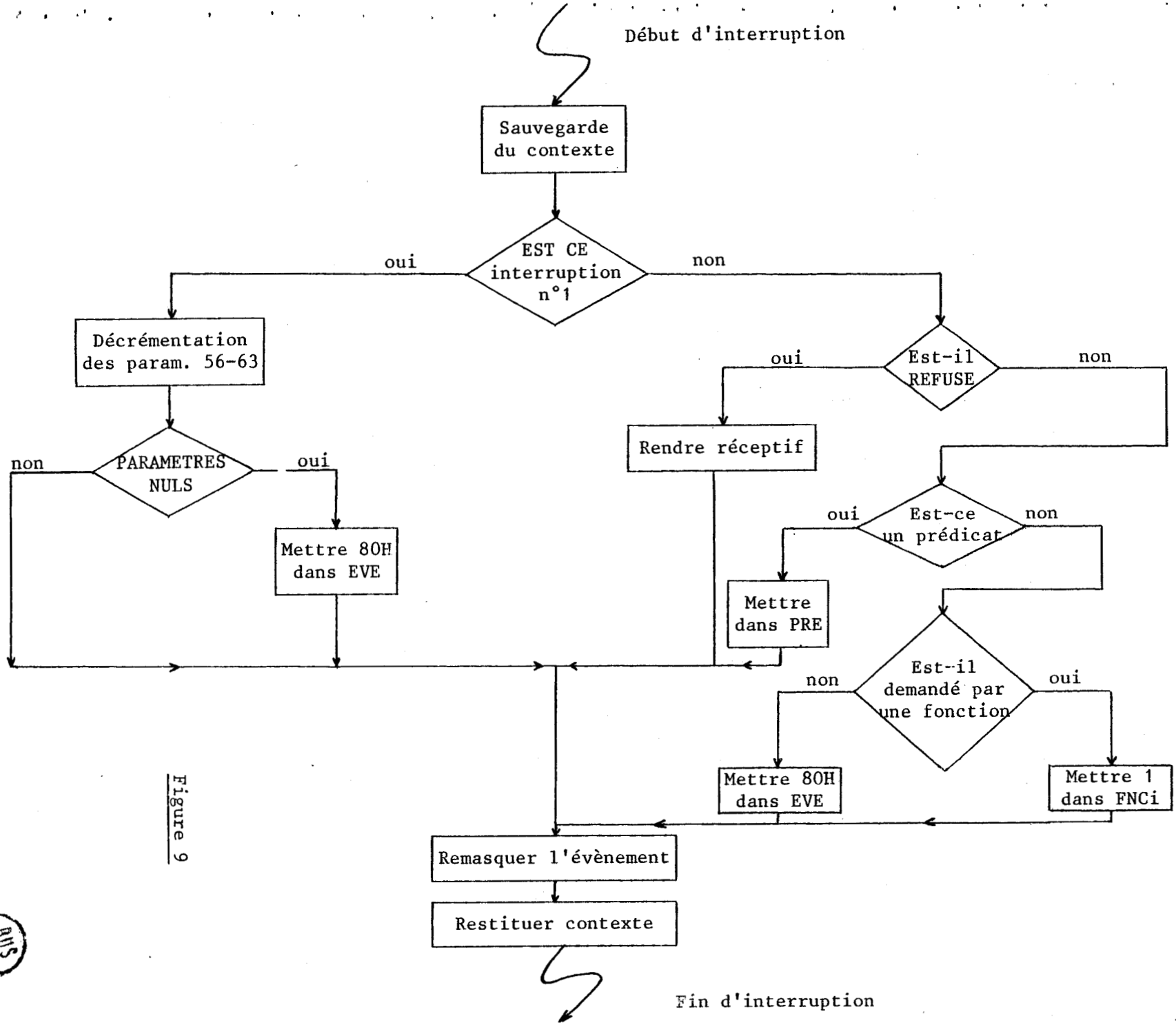

Début d'interruption
Sauvegarde du contexte
EST CE interruption n°1
oui
non
Décrémentation des param. 56-63
PARAMETRES NULS
non
oui
Mettre 80H dans EVE
Est-il REFUSE
oui
non
Rendre réceptif
Est-ce un prédicat
oui
non
Mettre dans PRE
Est-il demandé par une fonction
non
oui
Mettre 80H dans EVE
Mettre 1 dans FNCi
Remasquer l'évènement
Restituer contexte
Fin d'interruption

Figure 9

III.8 - METHODOLOGIE DE CONCEPTION D'AUTOMATISMES

Afin d'illustrer la facilité de mise en œuvre de cet ensemble, nous proposons de donner ici quelques éléments généraux relatifs à la conception d'un automatisme.

III.8.1 - Structures de base

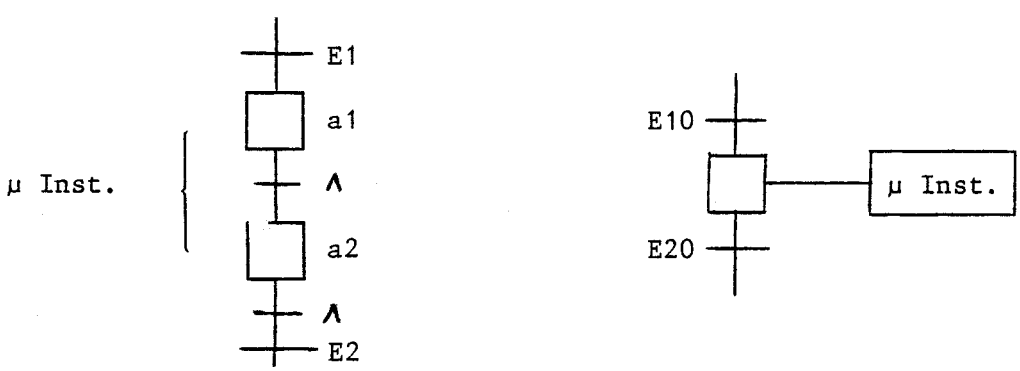
Les structures de programmation classiques sont obtenues à partir des éléments suivants :

a - Enchaînement de séquences

. dans une macro-instruction sans intervention d'évènements externes, l'enchaînement est obtenu selon une progression automatique du pointeur dont la durée ne dépend que du temps d'exécution des opérations réalisées.

. dans une tâche, l'enchaînement des macro - instructions est conditionnée par l'état vrai d'évènements, prédicats, ou fonctions.

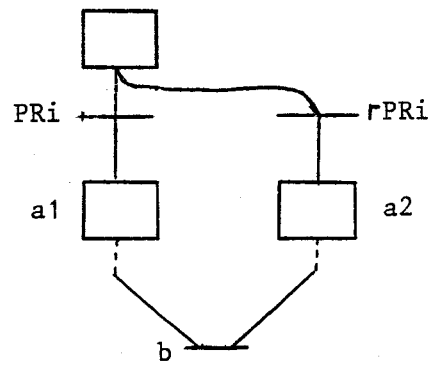
Ex : (E10) LDP P12, /CB
(E20) OT P12



b - Alternative

Elle est réalisée de manière simple par les instructions de rupture de séquence conditionnelle et de tests sur prédicats évalués dans la même tâche.

```
Ex : TPR  a2  PRi
     a1 : OPV IO1, 60
          :
          :   JMP b
     a2 : LDP P13, 13
          :
          :
          :
      b :
```

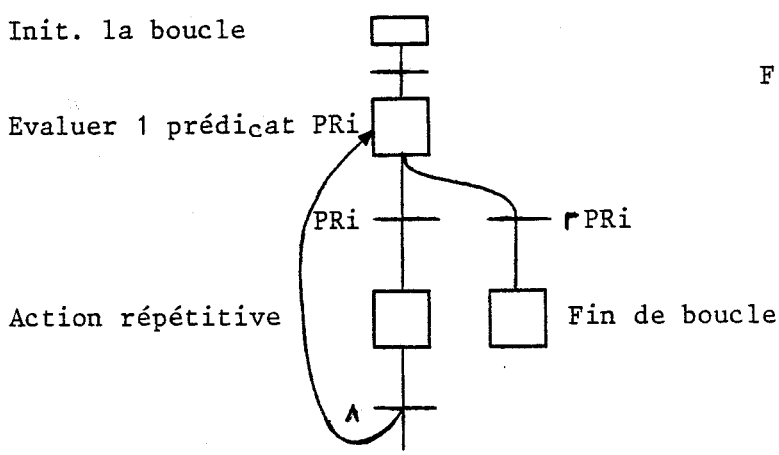


c - Boucles

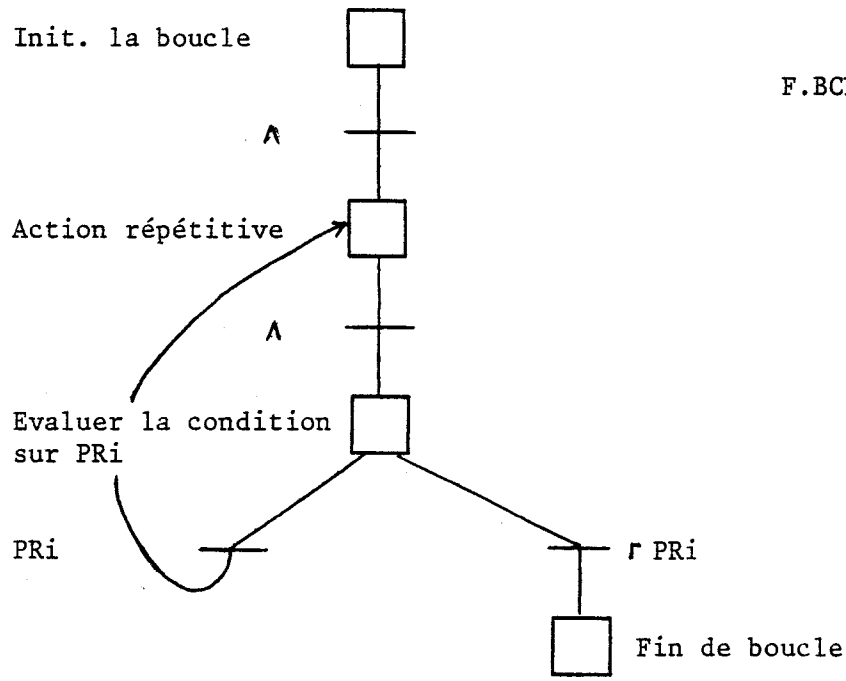
Les instructions de ruptures de séquences liées aux paramètres et prédicats assurent la possibilité de définition des 3 structures itératives classiques.

. boucle tant que

<u>Ex</u> : DEB : TPR F.BCL, PR1
ACTION REPETITIVE
JMP DEB
F.BCL : .
.
.
.

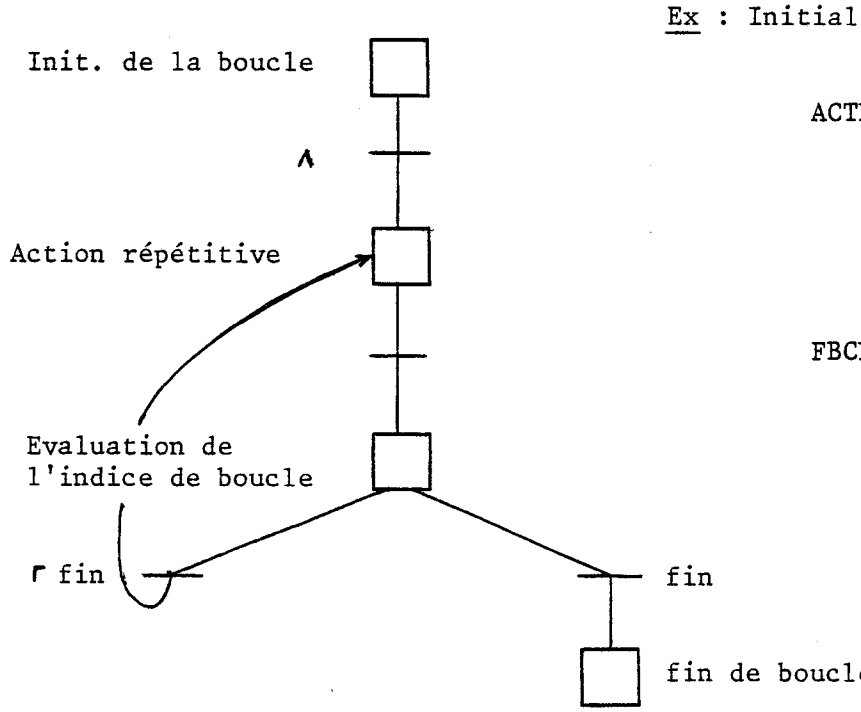


- Boucle jusqu'à



Ex : DEB : ACTION REPETITIVE
TPR F.BCL, PR2
JMP DEB
F.BCL : .....

- Boucle pour I = 1 , N ; faire ...



Ex : Initial. : CLP P1
LDP P2, N
ACTR : ACTION REPETITIVE
IMP P1
CME PR3, P1, P2
TPR F.BCL, PR3
JMP ACTR
FBCL : .

La composition de ces 3 blocs permet donc d'assurer pour les tâches associées un fonctionnement non bloquant.

En effet, les réseaux correspondant sont par définition vivants et

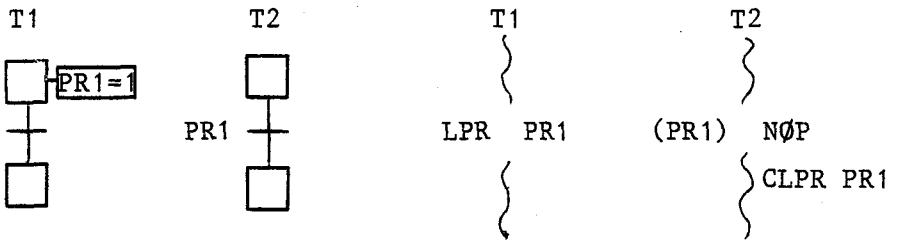
réinitialisables.

III.8.2 - Représentation des communications

III.8.2.1 - Synchronisation

La synchronisation entre tâches programmées peut être obtenue en utilisant un prédicat ou un évènement de signalisation.

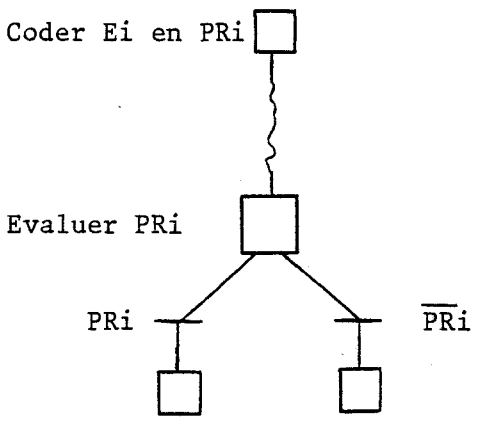
Ex :



La synchronisation avec le processus utilise les évènements gérés en mode prioritaire. La remise à 0 du prédicat de signalisation doit être assurée par la tâche réceptrice qui accuse ainsi réception de la signalisation.

Un évènement est automatiquement remis à zéro lors de la prise en compte par l'interpréteur, il peut être cependant intéressant de le mémoriser en prédicat.

Il devient alors possible de composer les fonctions de choix alternatif et de signalisation.




```
Ex : ENB Ep, PRi
          .
          .
          .
     TPR SAUT, PRi
Suite :   .
          .
          .
SAUT :    .
          .
          .
```

Ce mode peut être extrèmement utile notamment lorsqu'une opération déborde d'un temps limite selon l'exemple ci-dessous :

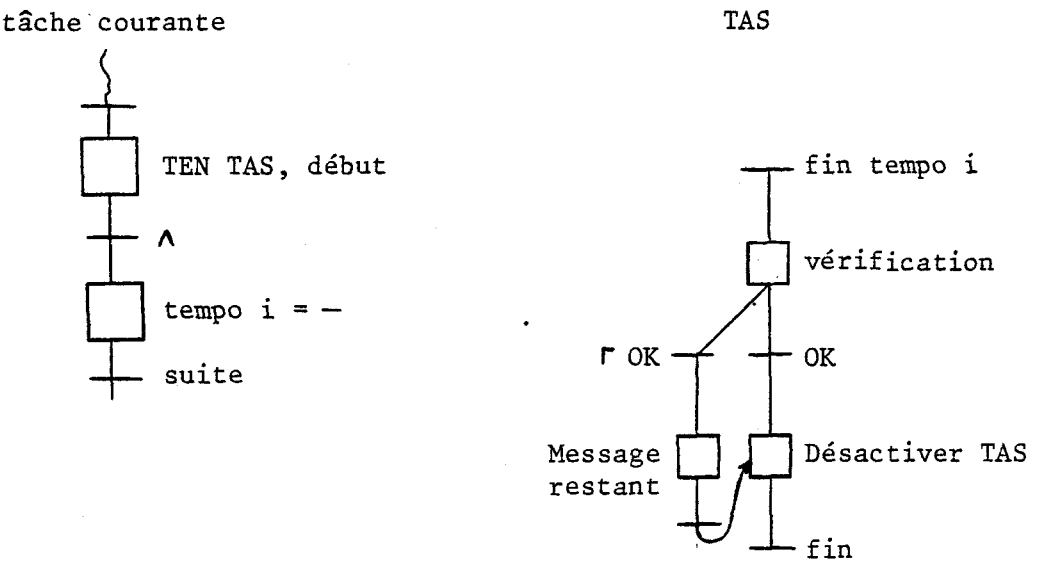
```
Param. P60, —
  ENB  E60, PR—
       :  Action
  TPR  TIMLIM, PR
Suite :    .
           .
           .
TIMLIM :
```

Codage d'un évènement de temps en prédicat

. Les fonctions d'évènements permettent de réaliser des OU logiques. Les intersections seront obtenues par déclenchements successifs de fonctions neutres NOP.

. Activation et désactivation des tâches (TEN, TIN)

Ex : Mise en fonction d'une tâche de surveillance après un certain délai

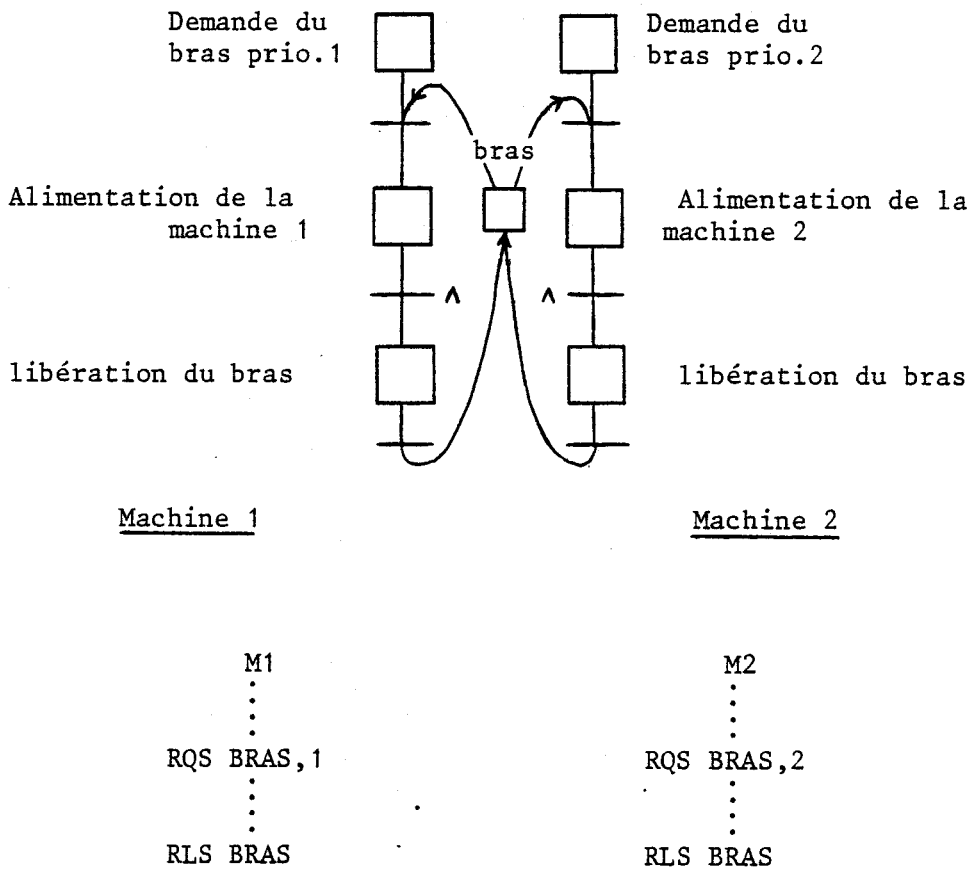


Les lancements de tâches simultanées ou activations de branches peuvent également s'effectuer à partir des instructions précitées.

III.8.2.2 - Utilisation des ressources critiques

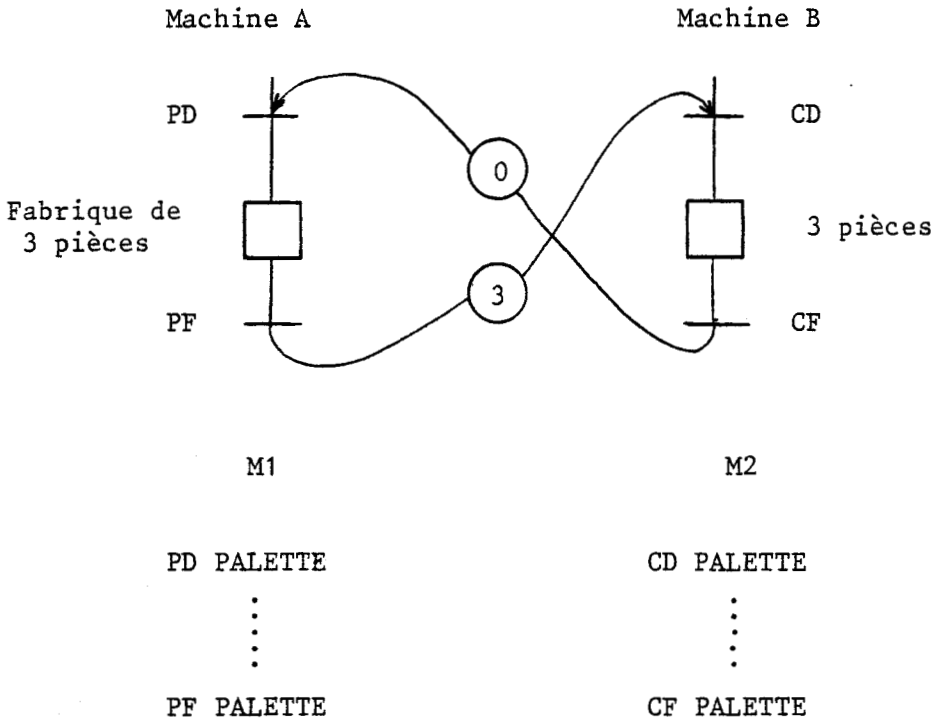
a) Sémaphore

L'utilisation de la ressource critique robot pour 2 machines nécessitant l'emploi du même robot d'alimentation pourra être programmé comme suit :



b) Producteur-Consommateur

Le robot précédent communique 3 pièces usinées par la machine A à la machine B. Le transfert s'effectue sur une palette unique chargée sur A, déchargée vers B séquentiellement. La machine A demande le chargement si elle dispose de 3 pièces usinées, la machine B demande une palette si son stock de pièces à usiner est vide.



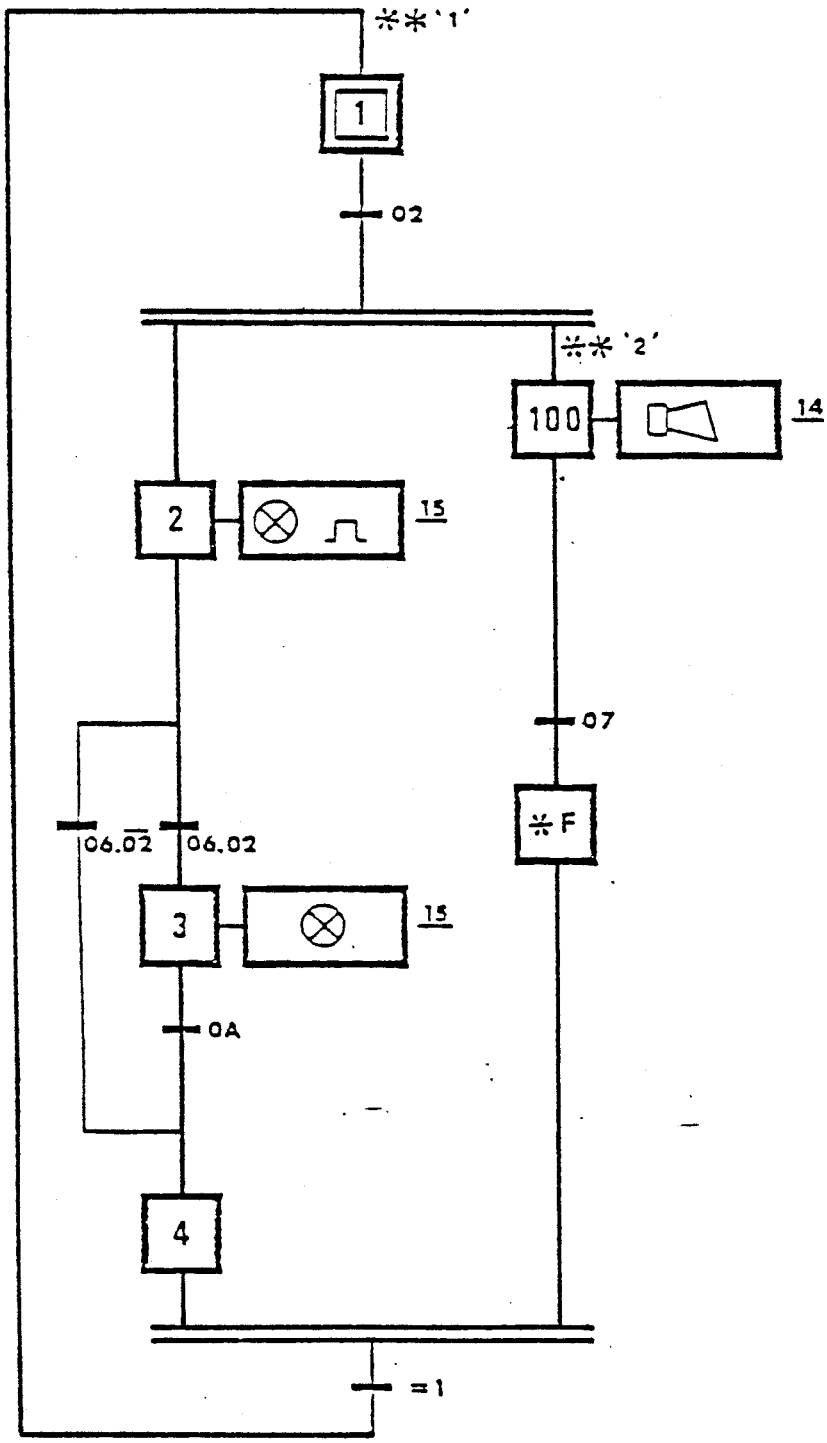
c) Echange de données

Les échanges de blocs de données entre tâches peut également s'effectuer par signalisation en utilisant les signaux de synchronisation sur sémaphores et les tables. De tels échanges de données nécessitent l'utilisation combinée des primitives de synchronisation et de gestion de tables.

III.9 - EXEMPLE DE PROGRAMMATION

Cet exemple est emprunté au manuel du TSX 80 (Télémécanique). Il concerne la signalisation de défauts. Il est présenté par le Grafcet de la figure suivante.

III.9.1 - Grafcet du processus



02 Défaut
06 Acquit. défaut
07 Acquit. signal sonore
0A Effacement du défaut

14 Signal sonore
15 Signal lumineux

* L'action du bouton 02 simule l'apparition d'un défaut et provoque :
  - la mise en marche d'un signal sonore
  - le clignotement d'une lampe
* L'action sur le bouton 07 arrête le signal sonore
* L'action sur le bouton 06 acquite le défaut et provoque :
  - l'allumage de la lampe si le défaut est maintenu
  - l'extinction de la lampe si le défaut a disparu
* L'action sur le bouton 0A confirme la disparition du défaut.

III.9.2 - Programme source

C'est le programme interpréteur du diagramme précédent.

```
PRO TACHE 1, TACHE 2
INI TACHE 1
*DEB
BEGIN TACHE 1
DEPART : NOP
     (E02) TEN TACHE 2, DEB2  ! lancer le tâche 2 en DEB2
     ENB E06, PR6             ! coder E6 en prédicat 6
     ENB E02, PR2             ! coder E2 en prédicat 2
CLIN : OPV I00, 01            ! lampe allumée
       TEMP1 Fa               ! pendant un temps Fa
       OPV I00, 00            ! lampe éteinte
       TEMP1 Fe               ! pendant un temps Fe
       TPR CLIN, PR6          ! test de saut si PR6 = 1
       TPR FIN4, PR2
       OPV I00, 01            ! lampe allumée
       (E0A) OPV I00,00       ! éteindre la lampe
FIN4 : (PR7)                  ! reset de la synchro
       CPR PR07
       CPR PR06
       JMP DEPART             ! saut départ
       END                    ! fin
BEGIN TACHE 2                 ! début tâche 2
DEB2 : OPV I01, 02            ! mise en marche du klaxon
       ENB E7, PR7            ! gérer synchronisation 7
       (PR7) OPV I01, 03 ! Arrêt Klaxon
       END
     *FIN
```

```
NOM DU PROGRAMME SOURCE :
CALCULATEUR UTILISE     : 8080
ADRESSE D'IMPLANTION    :  64080
```

ADRESSES HEXA.	ADRESSES DECI.	OCTETS	SIGNIFICATION
FA50	64080	00000010	NBRE DE TACHES = 2
		00000001	MOT D'ETAT DU PROGRAMME
FA52	64082	00000000	NBRE DE SEMAPHORES = 0
		00000000	NBRE DE TABLES = 0
FA54	64084	00000000	MOT D'ETAT DES FONCTIONS
FA55	64085	11110100	HORLOGE = 62500
		00100100	
FA57	64087	11111010	TACHE = TACHE1
		01011011	
FA59	64089	11111010	TACHE = TACHE2
		10101101	
			TACHE = TACHE1
FA5B	64091	00000000	DEPART:NOP
		00010001	
		00000000	
		00000000	
FA5F	64095	10000010	(E02)TEN TACHE2,DE02
		10001100	
		00000010	
		00000000	

```
                        11111010
                        10101101

FA65    64101           00000000        ENB E06,PR6
                        00001011
                        00000110
                        00000110

FA69    64105           00000000        ENB E02,PR02
                        00001011
                        00000010
                        00000010

FA6D    64109           00000000        CLIN:OPV I00,01     !LAMPE ALLUMEE
                        00011110
                        00000000
                        00000001

FA71    64113           00000000        TEMP1 30
                        00000001
                        00111001
                        00011110

FA75    64117           10111001
                        00010001
                        00000000
                        00000000

FA79    64121           00000000        OPV I00,00          !LAMPE ETEINTE
                        00011110
                        00000000
                        00000000

FA7D    64125           00000000        TEMP1 40            !PENDANT 40
                        00000001
                        00111001
                        00101000

FA81    64129           10111001
                        00010001
                        00000000
                        00000000

FA85    64133           00000000        TFR CLIN,PR06       !TEST DE SAUT SI PR06=1
                        10000100
                        11111010
                        01101101
                        00000110
                        00000000

FA89    64139           00000000        TFR FIN4,PR02
                        10000100
                        11111010
                        10011001
                        00000010
                        00000000
```

```
FA91    64145    00000000                DPV I00,01          !LAMPE ALLUMEE
                 00011110
                 00000000
                 00000001

FA95    64149    10001010                (E10)DPV I00,00     !ETEINDRE LAMPE
                 00011110
                 00000000
                 00000000

FA99    64153    11000111                FIN4:(PR07)CPR PR02 !RESET DE LA SYNCHRO
                 00001010
                 00000010
                 00000000

FA9D    64157    00000000                CPR PR07
                 00001010
                 00000111
                 00000000

FAA1    64161    00000000                CPR PR06
                 00001010
                 00000110
                 00000000

FAA5    64165    00000000                JMP DEPART          !SAUT DEPART
                 00110000
                 11111010
                 01011011

FAA9    64169    00000000                END                 !FIN
                 00001110
                 00000000
                 00000000

                                         TACHE = TACHE2

FAAD    64173    00000000                DEB2:DPV 101,02     !MISE EN MAR KLAXON
                 00011110
                 00000001
                 00000010

FAB1    64177    00000000                ENB E07,PR07        !GENERER SYNCHRO
                 00001011
                 00000111
                 00000111

FAB5    64181    11000111                (PR07)DPV 101,03    !ARRET KLAXON
                 00011110
                 00000001
                 00000011

FAB9    64185    00000000                END
                 00001110
                 00000000
                 00000000
```

TABLE DES ETIQUETTES
====================

NOM	ADRESSE HEXA.	ADRESSE DECI.
CLIN	FA6D	64109
DEB2	FAAD	64173
DEPART	FA5B	64091
FIN4	FA99	64153

III.10 - CONCLUSION

Au cours de ce chapitre, nous avons présenté les différents aspects du logiciel interpréteur constituant le noyau de l'automate.

Les primitives dont on dispose permettent d'une manière simple, de transcrire un diagramme fonctionnel. Les instructions disponibles peuvent facilement être étendues aussi bien au niveau du compilateur, à partir des primitives existantes, qu'au niveau de l'interpréteur.

Enfin, il convient d'insister sur le fait que la syntaxe générale d'un programme utilisateur écrit en langage évolué, est relativement facile à utiliser. Notamment les techniciens de l'atelier de mécanique de l'I. D. N. ont pu, très rapidement, s'initier à la méthodologie de conception d'un automatisme et au langage de traduction que nous avons présenté ici.

CHAPITRE 4

IV.1 - INTRODUCTION

Afin d'illustrer la facilité de mise en œuvre et les possibilités du logiciel, nous proposons dans ce chapitre le traitement de deux exemples d'application.

Le premier est emprunté au domaine de la machine outil et concerne la commande séquentielle multitâches d'une machine à usinage multiple alimentée automatiquement par robot. Cette commande a été effectivement mise en œuvre dans le cadre de l'atelier de mécanique de l'I. D. N. et constitue un test assez complet du logiciel présenté.

La seconde application concerne la simulation hybride, il s'agit de résoudre une simulation analogique/digitale d'un problème de convection du type échange de chaleur. Le graphe de commande du modèle analogique est réalisé à partir du logiciel présenté dans ce mémoire.

IV.2 - EXEMPLE D'APPLICATION : COMMANDE D'UNE MACHINE OUTIL

Le premier exemple d'application présenté au début de ce chapitre est celui de la commande d'un processus industriel consistant à faire un usinage particulier de pièces métalliques par une machine outil. |18| |19| |20| |21|

Cette commande est opérée par l'automate présenté précédemment dans une configuration adaptée au problème spécifique posé.

L'ensemble automatisé et réalisé est en cours d'exploitation dans les ateliers de mécanique de l'I. D. N..

IV.2.1 - Description de la partie opérative

Les fonctions d'usinage disponibles sont présentées sur le schéma général de l'installation (Figure 10).

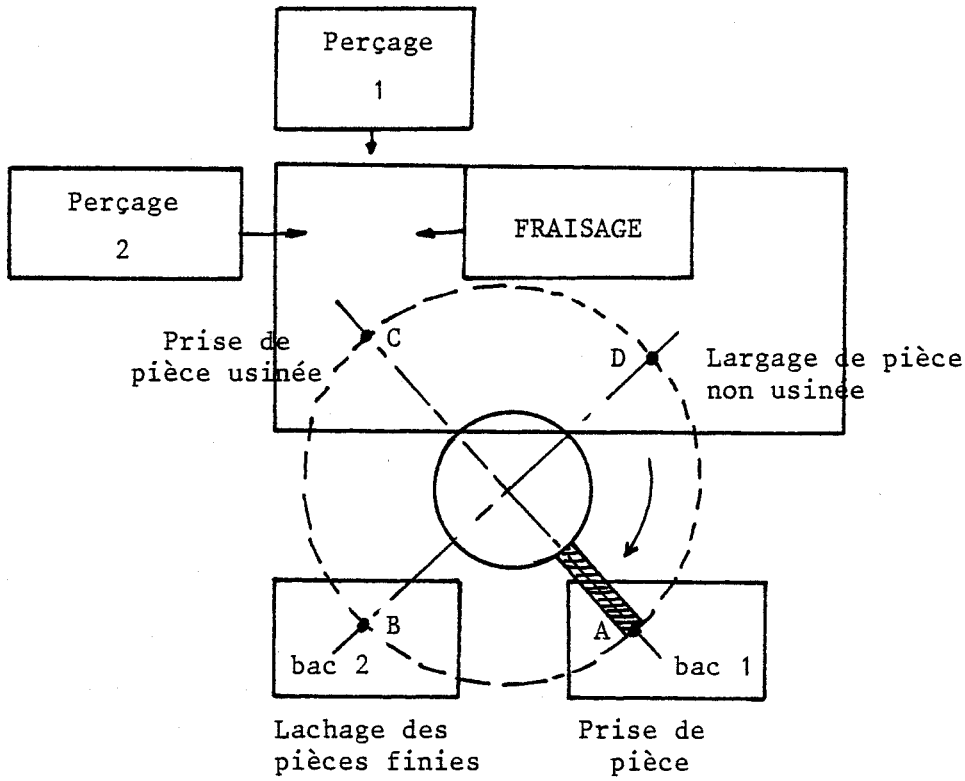


Figure 10

Le bac 1 contient les pièces à usiner.
Le bac 2 contient les pièces finies.

Le robot d'alimentation est constitué d'un bras suspendu sur un plateau tournant. Ce dernier vient chercher les pièces du Bac 1 pour les placer en D. Dès que la pièce est usinée, il la reprend en C pour la mettre dans le Bac 2, en B.

Le schéma détaillé de la machine est donné sur la figure 11.

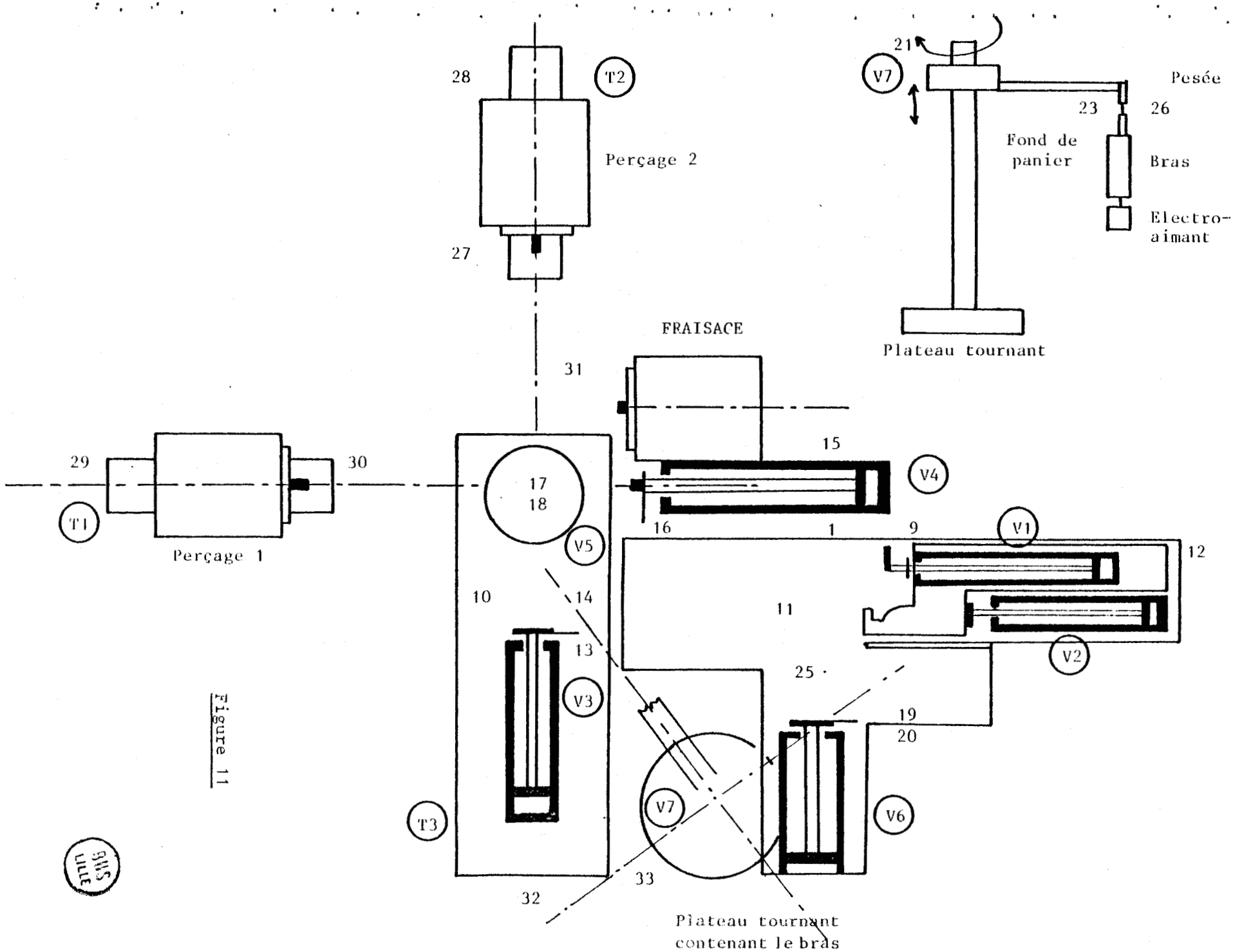
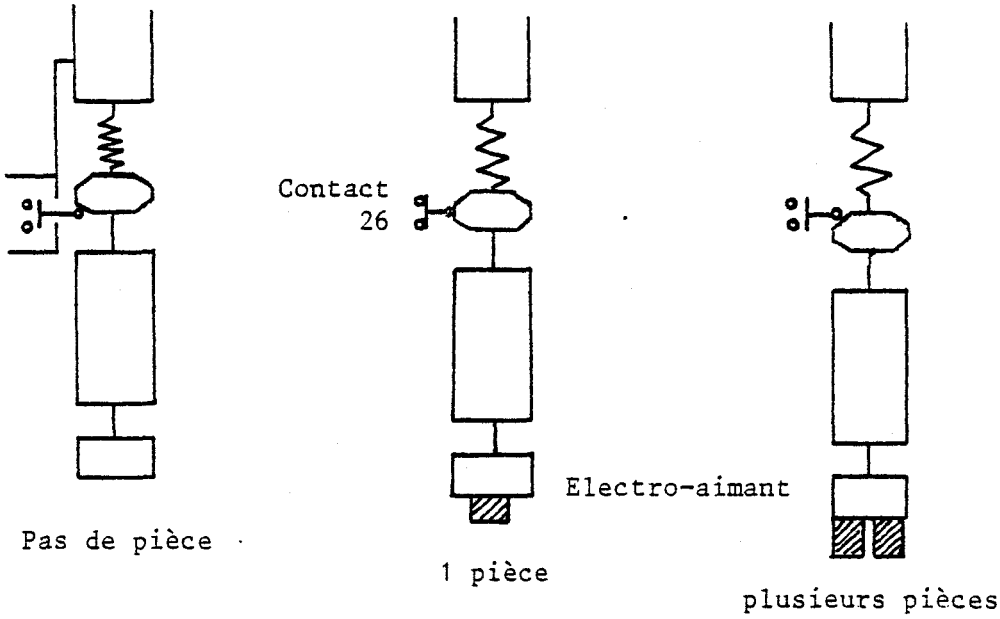

28
T2
Perçage 2
27
21
V7
Pesée
23
26
Fond de panier
Bras
Electro-aimant
Plateau tournant
FRAISACE
31
29
30
T1
Perçage 1
17
18
V5
15
V4
16
1
9
V1
12
10
14
11
13
V3
25
V2
19
20
V7
V6
T3
33
32
Plateau tournant contenant le bras

Figure 11

IV.2.2 - Description des éléments à contrôler : variables de commande

Plateau tournant : on l'a vu précédemment, c'est l'élément qui assure pour une part, l'acheminement des pièces. Il supporte un bras vertical disposant d'un électro-aimant, permettant la prise de pièce.



Un dispositif de pesée permet de savoir si une, aucune ou plusieurs pièces ont été accrochées par l'électro-aimant. Il s'effectue par l'intermédiaire d'un simple contact (n° 26), comme le montre les différentes situations de la figure ci-dessus.

Liste des commandes et contrôles possibles :

V7 : Vérin de montée et descente du bras
V6 : Emplacement intermédiaire de la pièce dès son largage
V2 : Rectifie la position de la pièce tout en la faisant avancer
V1 : Termine la position de la pièce sur la table 3
V3 : Place et retire la pièce dans/de l'étau
V5 : Sert au maintien de la pièce
V4 : Sert d'appui lors du perçage n°1

9 : Entrée du vérin V1
10 : Sortie de V1
11 : Sortie de V2
12 : Entrée de V2
13 : Entrée de V3
14 : Sortie de V3
15 : Sortie de V4 : manostat, contact à chute de pression
16 : Entrée de V4
17 : Entrée de V5
18 : Sortie de V5 : manostat
19 : Sortie de V6 : "
20 : Entrée de V6
21 : Sortie de V7 (bras en position haute)
22 : BP MARCHE
23 : Fond de panier
24 : BP ARRET D'URGENCE
25 : Présentation de pièce sur table
26 : Nombre de pièce sur bras
27 : Fin de perçage 2
28 : Recul de table 2
29 : Recul de table 1
30 : Avance de table 2
31 : Avance de table 3
32 : Recul de table 3
33 : Repère fixe du plateau
34 : Indique 1/4 de tour du plateau

Nature des contacts utilisés : Les contacts (9 - 24) sont fugitifs et associés à des évènements déclenchables sur front positif.

Par contre, les contacts (25 - 34) sont associés aux évènements correspondants. Ils sont utilisés sur front et leur présence doit être assurée pendant un certain temps.

Nature des vérins utilisés : V2, V4, V5, V7 sont des vérins commandés par distributeurs bistables.
V1 et V3 sont des vérins commandés par distributeur monostable (5 orifices, 3 positions, à centre ouvert). Ce sont des distributeurs à position médiane.

IV.2.3 - Cahier des charges de l'automatisme

On veut automatiser l'usinage des pièces ainsi que l'alimentation de la machine. De plus, une optimisation du temps d'usinage conduit à gérer simultanément le plus grand nombre de pièces sur la machine afin de réduire les temps d'attente découlant de l'alimentation ou du retrait des pièces.

Cette formulation conduit à un R d P possible représentant l'automatisme donné par la figure 14, dans lequel on a réparti l'ensemble du procédé en 5 tâches évoluant simultanément.

Tâche P5 : Tâche réservée à la mise en marche du système et à l'arrêt d'urgence

Tâche P1 : tâche d'apport de pièce ; sert à amener les pièces du panier pour les placer sur la machine

Tâche P3 : Positionnement des pièces ; elle transfère les pièces vers la zone d'usinage

Tâche P2 : cette tâche est réservée à l'usinage proprement dit, incluant l'avance des tables, le perçage, la marche des moteurs, ...

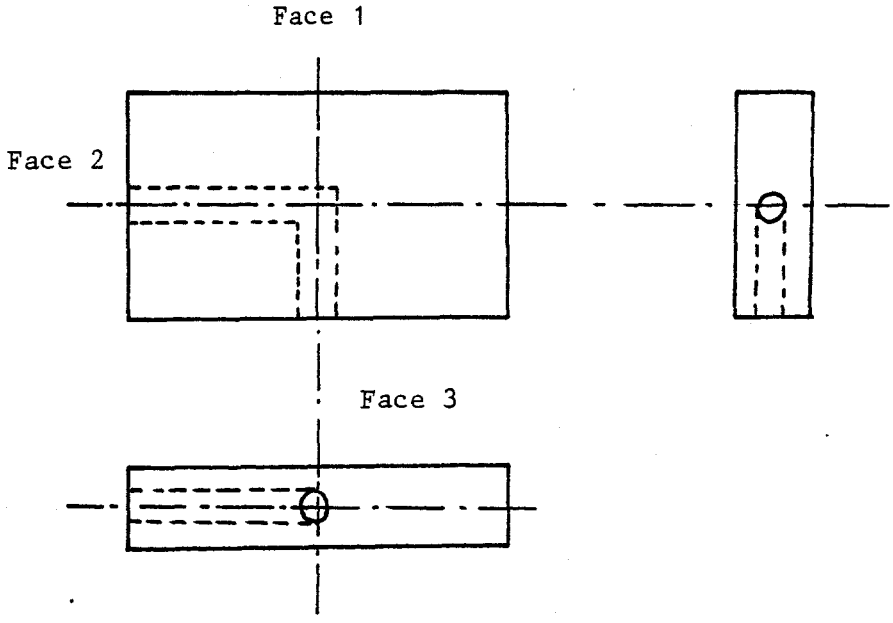
Tâche P4 : cette tâche concerne la reprise des pièces usinées et leur acheminement au bac 2.

IV.2.3.1 - Description de la pièce usinée et des différentes opérations nécessaires

La pièce usinée est parallèlépipédique. L'usinage de cette pièce se compose :

- d'un fraisage en face 1
- d'un perçage en face 2
- d'un perçage en face 3

selon le schéma suivant :



La machine comporte 3 unités d'usinage autonomes :

- 2 unités de perçage orthogonales
- 1 unité de fraisage

IV.2.3.2 - Utilisations des sémaphores

Les deux sémaphores utilisés dans cet exemple sont des sémaphores à valeur initiale égale à l'unité, ils sont donc du type exclusion mutuelle entre tâches (Figure 12).

BRAS : C'est une ressource unique et commune aux tâches P1 (apport de pièces) et P2 (prise de pièce). son utilisation est évidente : si on apporte une pièce pour usinage, on ne peut en même temps, prendre une autre pièce usinée.

SEMA : C'est un sémaphore utilisé pour exclure la largage de pièce tant que la zone réceptrice n'a pas évacué la pièce précédente. Elle est partagée entre P1 et P3.

IV.2.3.3 - Synchronisations entre tâches

Elles s'opèrent sur évènements ou par utilisation des prédicats.

Les différentes liaisons sont représentées sur la figure 13.

Ceci nous conduit au R d P final représentant le processus global et donné sur la figure 14.

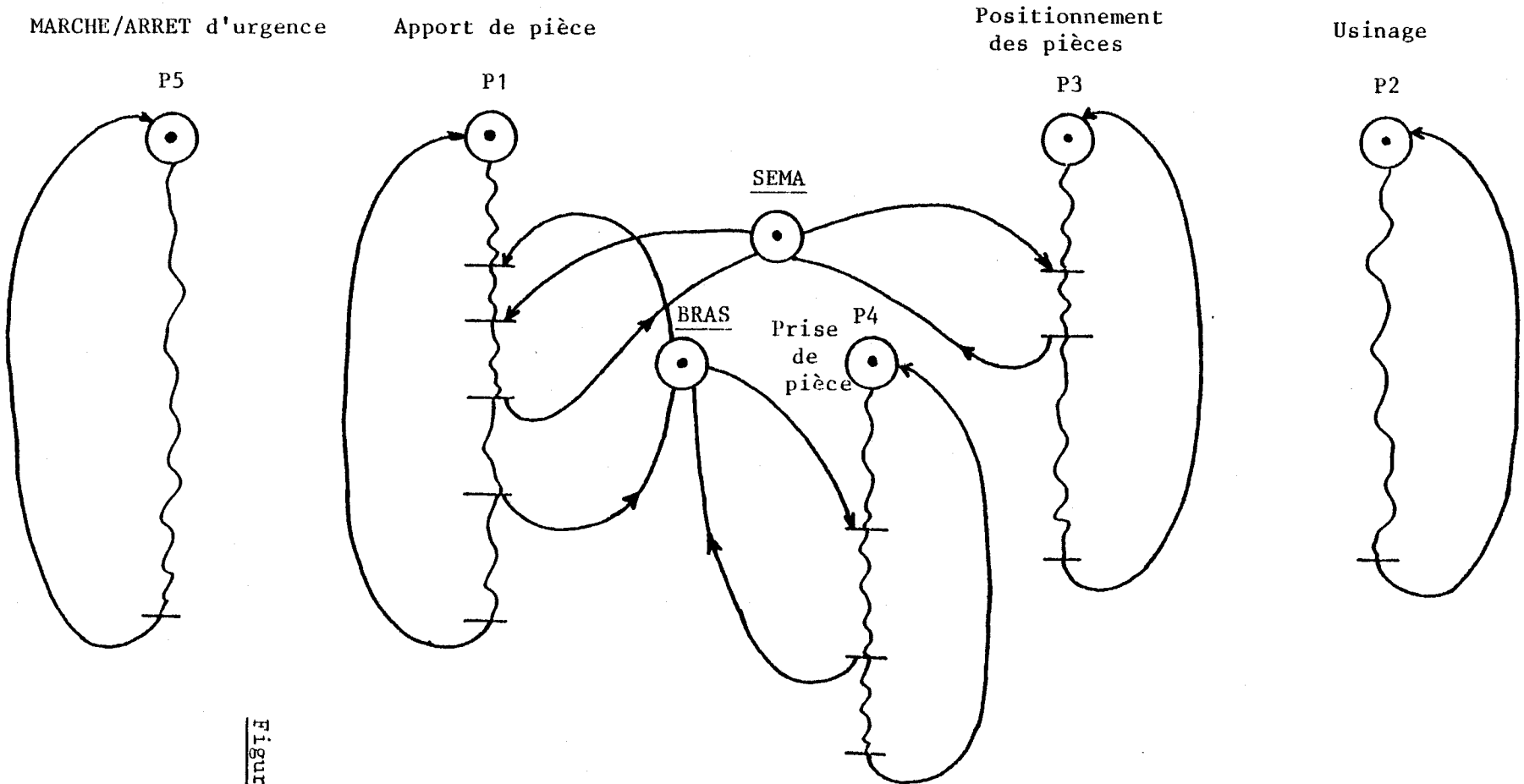


Figure 12

UTILISATION DES SEMAPHORES

IV.2.3.4 - Programme utilisateur de la machine

Ce programme intitulé ROBOT 1 a une longueur de 12(mées en langage évolué, il est donné en annexe. Il tradu l'organigramme à réseau de Pétri décrit précédemment.

Remarque : Un tel problème peut être abordé d'une aut au niveau de la répartition des tâches, des r et des synchronisations.

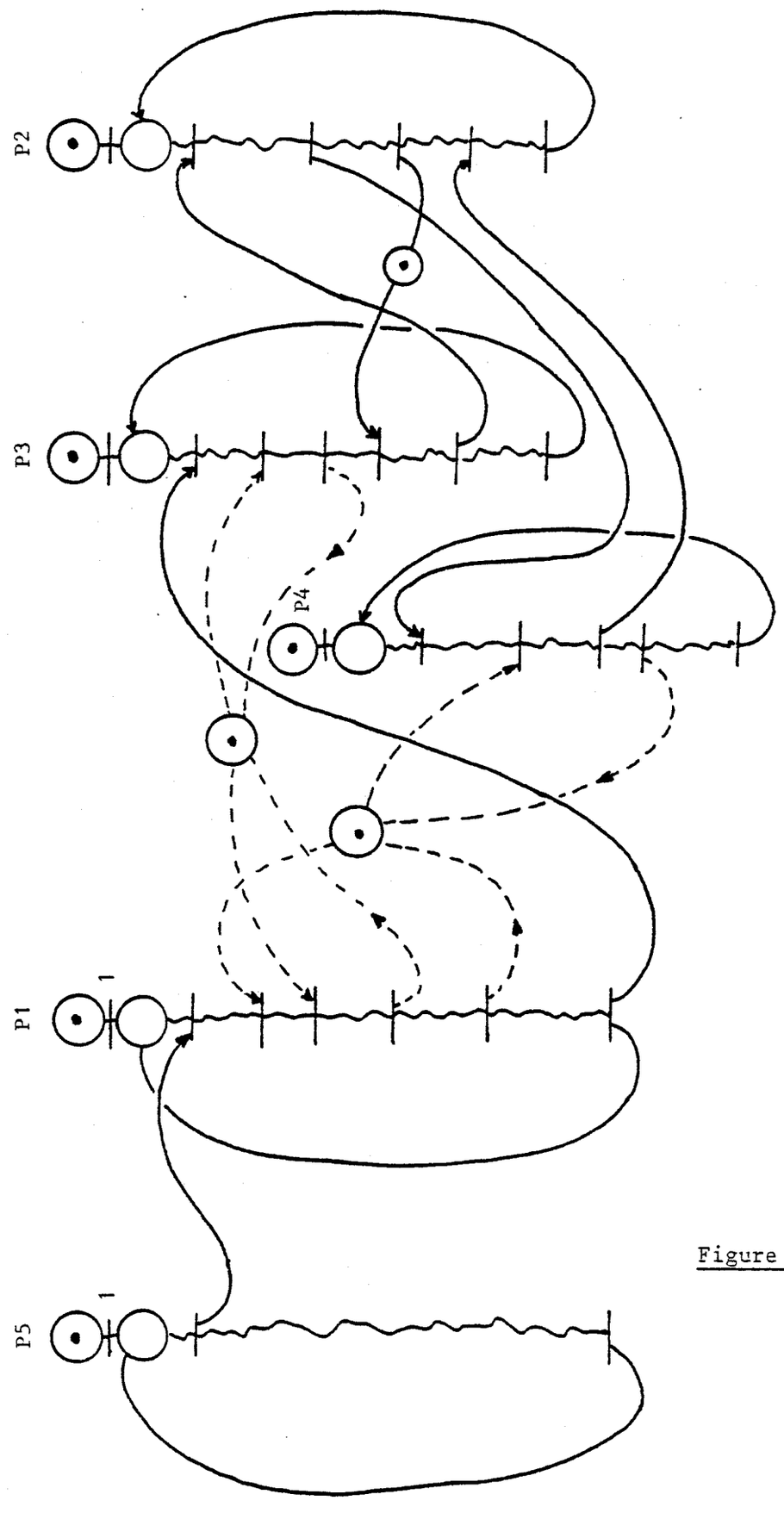


Figure 13

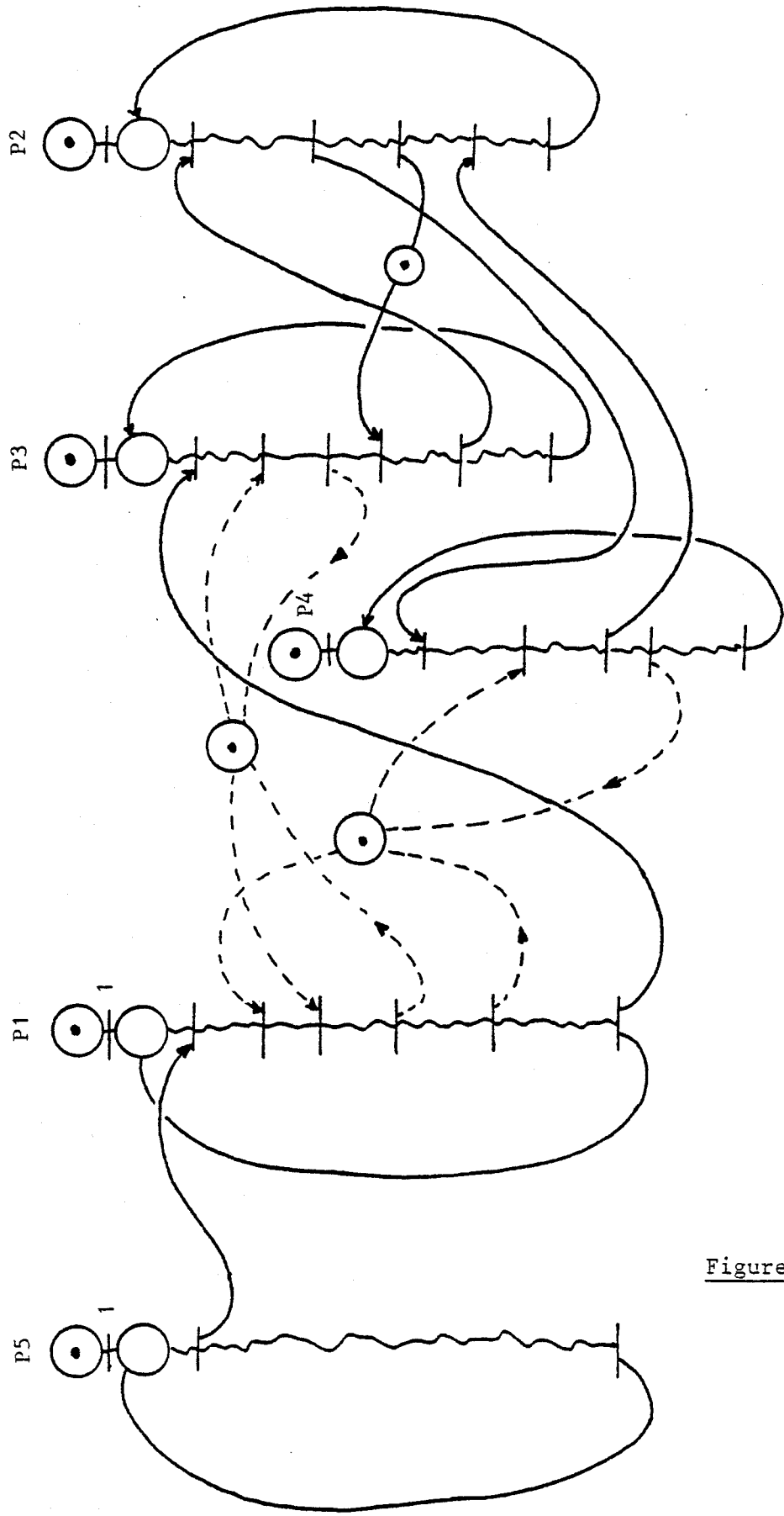


Figure 13

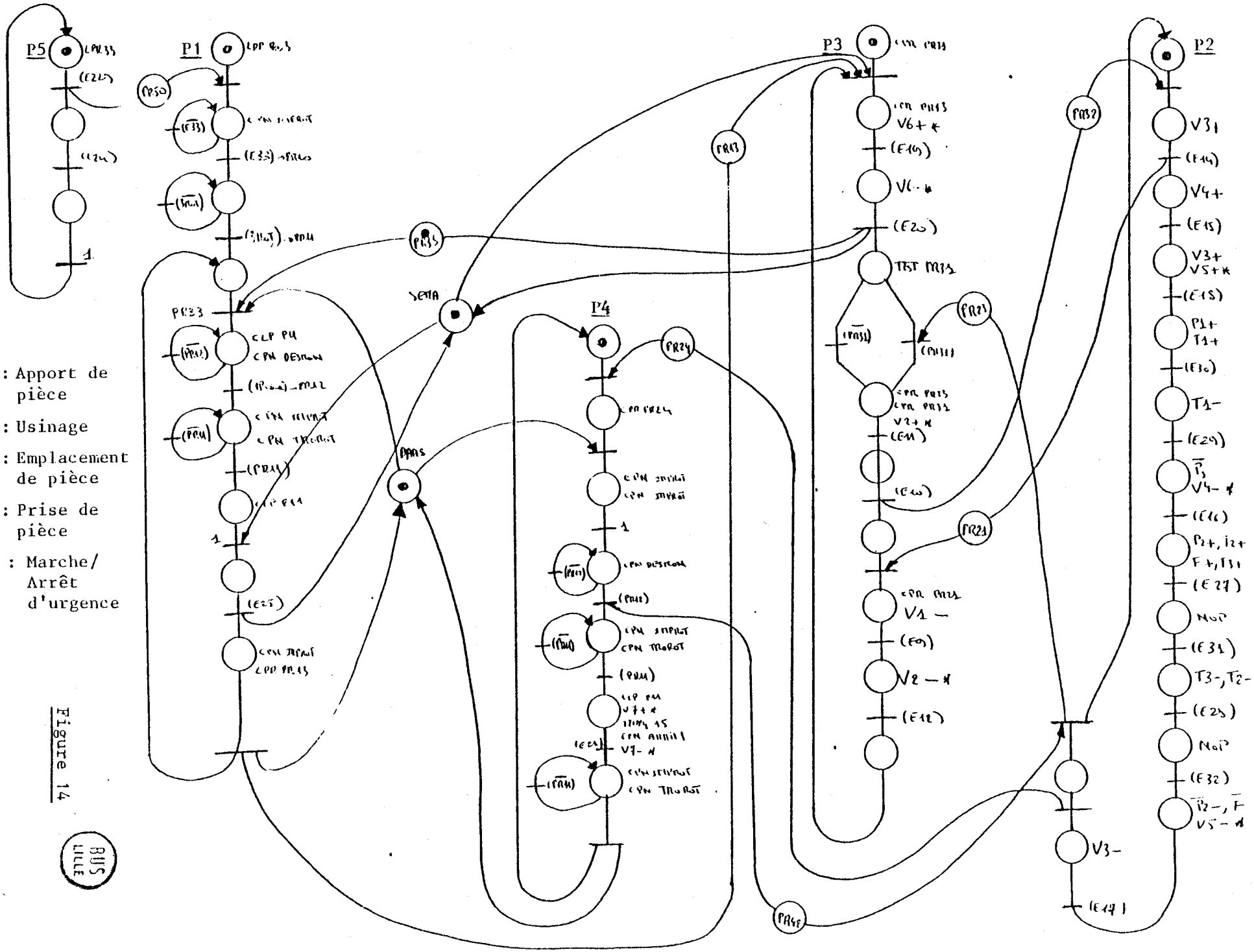


Figure 14

BUS LILLE

IV.2.3.4 - Programme utilisateur de la machine outil

Ce programme intitulé ROBOT 1 a une longueur de 120 lignes programmées en langage évolué, il est donné en annexe. Il traduit fidèlement l'organigramme à réseau de Pétri décrit précédemment.

Remarque : Un tel problème peut être abordé d'une autre manière, même au niveau de la répartition des tâches, des ressources communes et des synchronisations.

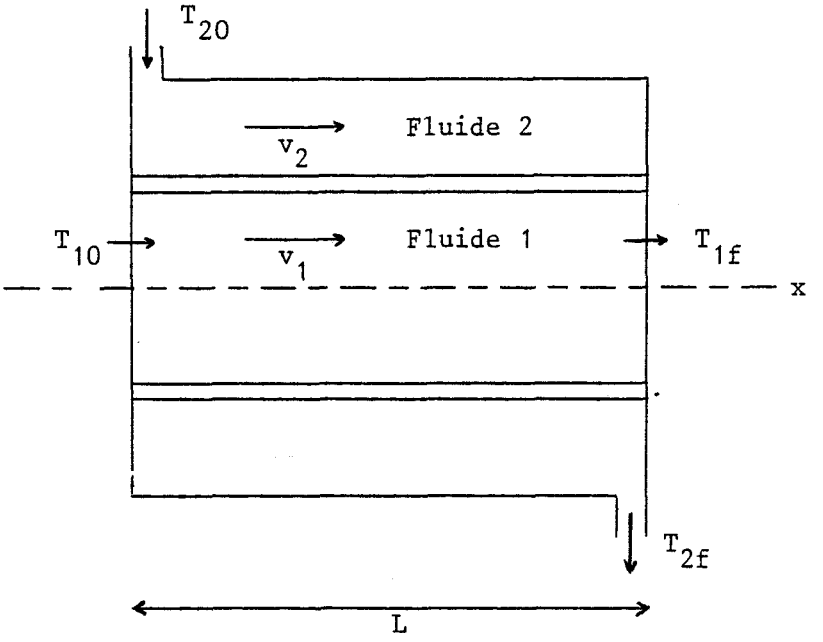
IV.3 - EXEMPLE D'APPLICATION : SIMULATION HYBRIDE D'UN ECHANGEUR THERMIQUE

Nous avons noté précédemment, qu'il était possible de contrôler un calculateur analogique à partir de l'automate présenté dans ce mémoire. Le logiciel comporte toutes les instructions nécessaires au pilotage d'un tel ensemble.

Dans ce sens, nous avons pu coupler un calculateur "TEREL" à l'automate, pour constituer un ensemble de calcul hybride de type II.

Afin d'illustrer la facilité de mise en œuvre du logiciel présenté, nous proposons d'étudier la simulation d'un modèle d'échangeur thermique.

L'échangeur est constitué de deux cylindres coaxiaux contenant chacun un fluide différent, circulant à des vitesses différentes et présenté sur la figure suivante.



On note T_i (x,t) la température du fluide i au point d'abcisse x et à l'instant t.

Nous supposons que la paroi séparant les deux cylindres est d'épaisseur négligeable.

IV.3.1 - Modèle élémentaire

Hypothèse : On considère que le fluide 2 a une vitesse nulle et qu'il se trouve à température uniforme α.
La température du fluide 1 à une abcisse x, à un instant est notée $T_1(x,t)$.

Si k_1 est le coefficient d'échange par convection (relation de COLBURN), la température $T_1(x,t)$ est la solution de l'équation aux dérivées partielles du premier ordre (1) :

$$\frac{\partial T_1(x,t)}{\partial t} + V_1 \frac{\partial T_1(x,t)}{\partial x} = k_1 \ (\alpha - T_1(x,t)) \qquad (1)$$

avec l'utilisation de la méthode des caractéristiques, on démontre (voir annexe 4) que la solution de l'équation (1) est celle du système d'équations différentielles (2) :

$$\begin{cases} \dfrac{dT_1(x,t)}{dt} = k_1 \ (\alpha - T_1(x,t)) \\ \\ V_1 = \dfrac{dx}{dt} \end{cases} \qquad (2)$$

Dans ce cas, la solution $T_1(x,t)$ est obtenue par une simple simulation analogique.

IV.3.2 - Modélisation complète

Hypothèse a : On note V_1 la vitesse du fluide 1 et V_2 la vitesse du fluide 2. Dans un premier temps, nous supposerons qu'elles ont égales ($V_1 = V_2$).

A partir de l'équation (1) et en supposant que la quantité de chaleur émise par le fluide 1 est, en première approximation, totalement récupérée par le fluide 2, il vient :

$$dQ_1 = - dQ_2$$

Donc, les températures $T_1(x,t)$ et $T_2(x,t)$ respectives aux fluides 1 et 2, sont les solutions du système (3) suivant :

$$\begin{cases} \dfrac{dT_1(x,t)}{dt} = k_1\ (\alpha - T_1(x,t)) \\ dT_2(x,t) = k_2\ (\alpha - T_2(x,t)) \\ \dfrac{dT_1(x,t)}{dt} = -\dfrac{dT_2(x,t)}{dt} \\ V_1 = V_2 = \dfrac{dx}{dt} \end{cases} \qquad (3)$$

Effectuons les changements de variables suivants :

$$\theta_1(x,\tau) = \frac{T_{10} - T_1(x,\tau)}{T_{10} - T_{20}} \qquad \theta_2(x,\tau) = \frac{T_{10} - T_2(x,\tau)}{T_{10} - T_{20}}$$

$$K = \frac{k_1 \cdot k_2}{k_1 + k_2} \qquad \tau = Kt \qquad V^* = V_1/K = V_2/K$$

avec : $\theta_1(x,\tau)$: variable réduite déduite de $T_1(x,t)$
$\theta_2(x,\tau)$: variable réduite déduite de $T_2(x,t)$
τ : variable réduite déduite de t
V^* : variable réduite déduite de V

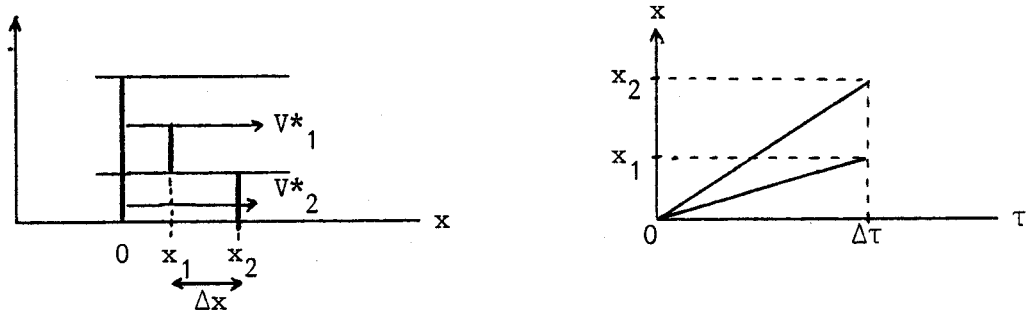
$T_{10} = T_1(0,0)$: valeur initiale de T_1
$T_{20} = T_2(0,0)$: valeur initiale de T_2

Après changement de variables, le système (3) s'écrit sous la forme (4) suivante :

$$\begin{cases} \dfrac{d\theta_1(x,\tau)}{d\tau} = \theta_2(x,\tau) - \theta_1(x,\tau) \\ \dfrac{d\theta_2(x,\tau)}{d\tau} = \theta_1(x,\tau) - \theta_2(x,\tau) \\ V^* = dx/d\tau \end{cases} \qquad (4)$$

Hypothèse b : Si la vitesse V_1 est différente de la vitesse V_2, le système d'équations différentielles (4) n'est plus représentatif du modèle.

Dans ce cas, nous supposerons que le modèle décrit en (4) reste valable pendant un temps $\Delta\tau$ petit. Il convient toutefois de préciser que les deux tranches de fluides 1 et 2 doivent partir de la même origine et qu'en conséquence, elles atteindront les abcisses $x_1 = V^*_1.\Delta\tau$ et $x_2 = V^*_2.\Delta\tau$ (avec, en général, $x_1 \neq x_2$ si $V^*_1 \neq V^*_2$).



Dans ce cas, il convient donc au bout de l'intervalle de temps considéré ($\Delta\tau$), d'effectuer un calcul de correction pour replacer les deux tranches en coïncidence et relancer un nouveau pas.

Le calcul de correction peut s'effectuer au premier ordre en supposant que les recalages successifs s'effectuent sur le fluide 2 par rapport au fluide 1 pris pour référence.

$$\theta_2(x_1,\tau) = \theta_2(x_2,\tau) + \Delta x \frac{d\theta_2(x_2,\tau)}{dx} + \varepsilon.\Delta x^2$$

soit encore le système d'équation suivant :

$$\begin{cases} \theta_2(x_1,\tau) = \theta_2(x_2,\tau).\left|1 - \Delta x/V^*_2\right| + \dfrac{\Delta x}{V^*_2}.\theta_1(x_1,\tau) \\ \dfrac{d\theta_1(x,\tau)}{d\tau} = \theta_2(x',\tau) - \theta_1(x,\tau) \\ \dfrac{d\theta_2(x',\tau)}{d\tau} = \theta_1(x,\tau) - \theta_2(x',\tau) \end{cases} \qquad (5)$$

avec : $x \in [0,x_1]$ et $x' \in [0,x_2]$

$x = V^*_1 \tau$ $\qquad$ $x' = V^*_2 \tau$

IV.3.3 - Simulation

Pour simuler ce dernier modèle, nous avons le choix entre deux méthodes :

- La première consiste à effectuer le calcul de correction en analogique, l'automate ne gère que des variables séquentielles.

- La deuxième solution consiste à effectuer sur l'automate, le calcul de correction sur chaque pas.

Cette simulation a poir objet d'indiquer en régime dynamique, l'effet d'une modification de débits sur les températures de sortie.

Elle permet donc de disposer d'un modèle de processus en vue d'en effectuer le contrôle.

Données numériques :

Si nous nous intéressons au modèle constitué par l'utilisation de l'eau comme fluide 2 et du Benzène comme fluide 1, il vient les données d'application suivantes :

$L = 1$ m	$k_1 = 0,05\ s^{-1}$
$T_{10} = 60$ °C	$k_2 = 0,1\ s^{-1}$
$T_{20} = 20$ °C	$V_2 = 0,1$ m/s
	$V_1 = 0,05$ m/s

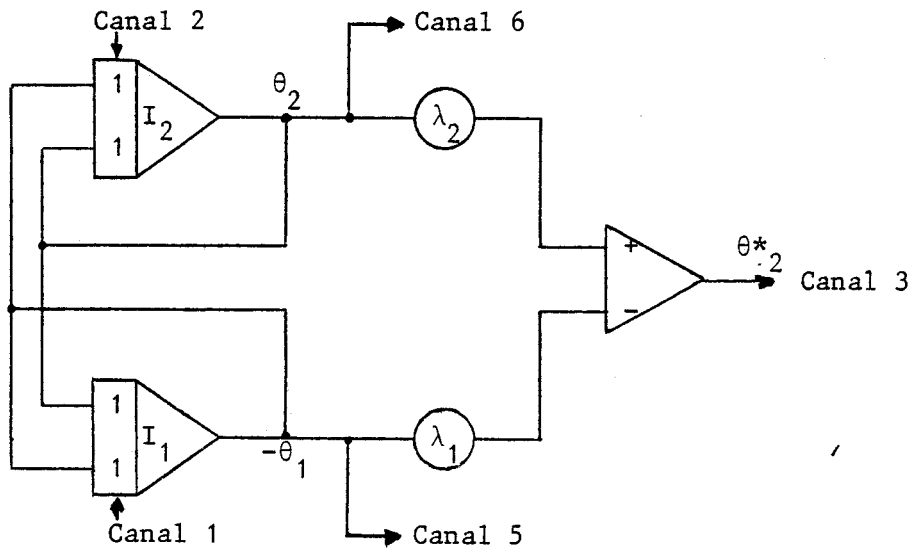
Fixons le nombre de pas (N) à 100. Il vient alors : K = 10/3.

1ère méthode : Les paramètres utilisés dans le système (5) sont préalablement calculés et fixés par cablage. Il en découle la simulation du système d'équations suivantes :

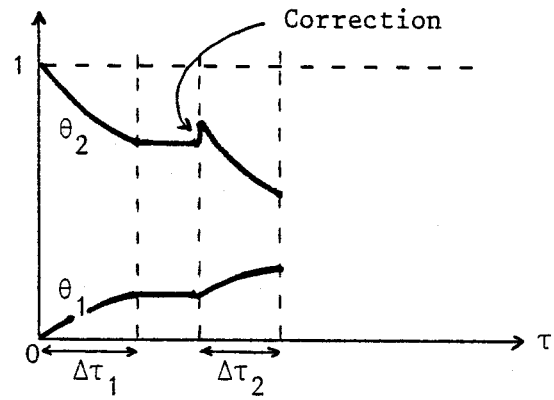
$$\begin{cases} \dfrac{d\theta_1(x,\tau)}{d\tau} = \theta_2(x',\tau) - \theta_1(x,\tau) \\ \dfrac{d\theta_2(x,\tau)}{d\tau} = \theta_1(x,\tau).\lambda_1 + \theta_2(x',\tau).\lambda_2 \\ V^*_2 = \dfrac{V_2}{K} = \dfrac{dx'}{d\tau} \end{cases} \qquad (6)$$

$\lambda_1 = 2/3$ $\qquad\qquad$ $\lambda_2 = 1/3$

et le schéma de cablage suivant :



La simulation s'effectue de la manière suivante :



Au bout d'un temps $\Delta\tau$, l'intégrateur 1 est mis en mode mémoire ; l'intégrateur 2 également. Après correction, l'intégrateur 2 est mis en Conditions Initiales et l'ensemble reprend un nouveau pas de calcul analogique.

Le Réseau de Pétri relatif à la commande de cette simulation est décrit sur la figure 15. Il est constitué de deux tâches différentes :

- La première est affectée à la commande du calculateur analogique puis au traçage des mesures enregistrées au bout des N pas.

- Le seconde consiste à surveiller les saturations et à les visualiser.

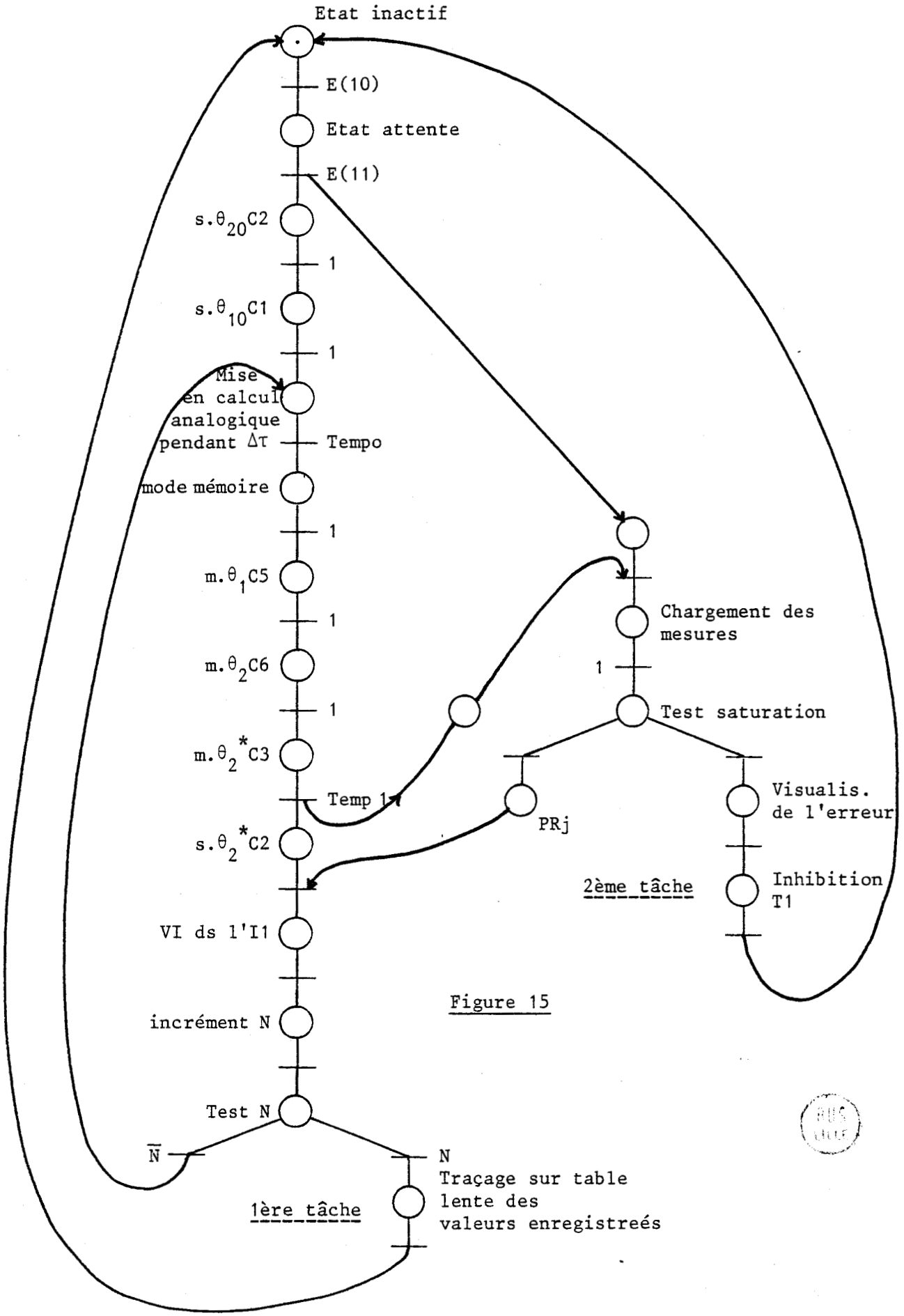
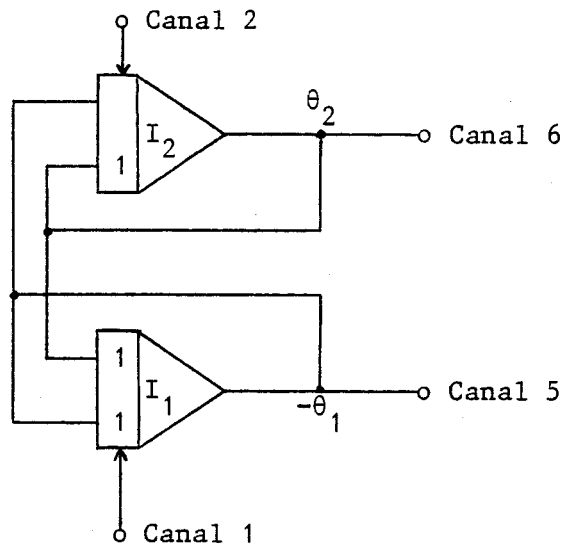

Etat inactif
E(10)
Etat attente
E(11)
s.θ20C2
1
s.θ10C1
1
Mise en calcul analogique pendant Δτ
Tempo
mode mémoire
1
m.θ1C5
1
m.θ2C6
1
m.θ2*C3
Temp 1
s.θ2*C2
VI ds 1'I1
incrément N
Test N
N̄
N
Traçage sur table lente des valeurs enregistreés
1ère tâche
Chargement des mesures
1
Test saturation
PRj
Visualis. de l'erreur
Inhibition T1
2ème tâche

Figure 15

2ème méthode : La seconde solution proposée ici permet directement l'affichage numérique des paramètres V_1 et V_2. Dans ce cas, le calcul de correction s'effectue sur l'automate et le schéma de cablage se réduit à celui de la figure ci-dessous :



La gestion du sous-programme analogique s'effectue à partir de 3 tâches :

- une première tâche est affectée à la commande du calculateur analogique et la gestion des échanges

- une seconde tâche concerne le calcul de correction puis l'enregistrement graphique des résultats.

- Une troisième tâche est relative à la surveillance des saturations pour mise en sécurité du dispositif.

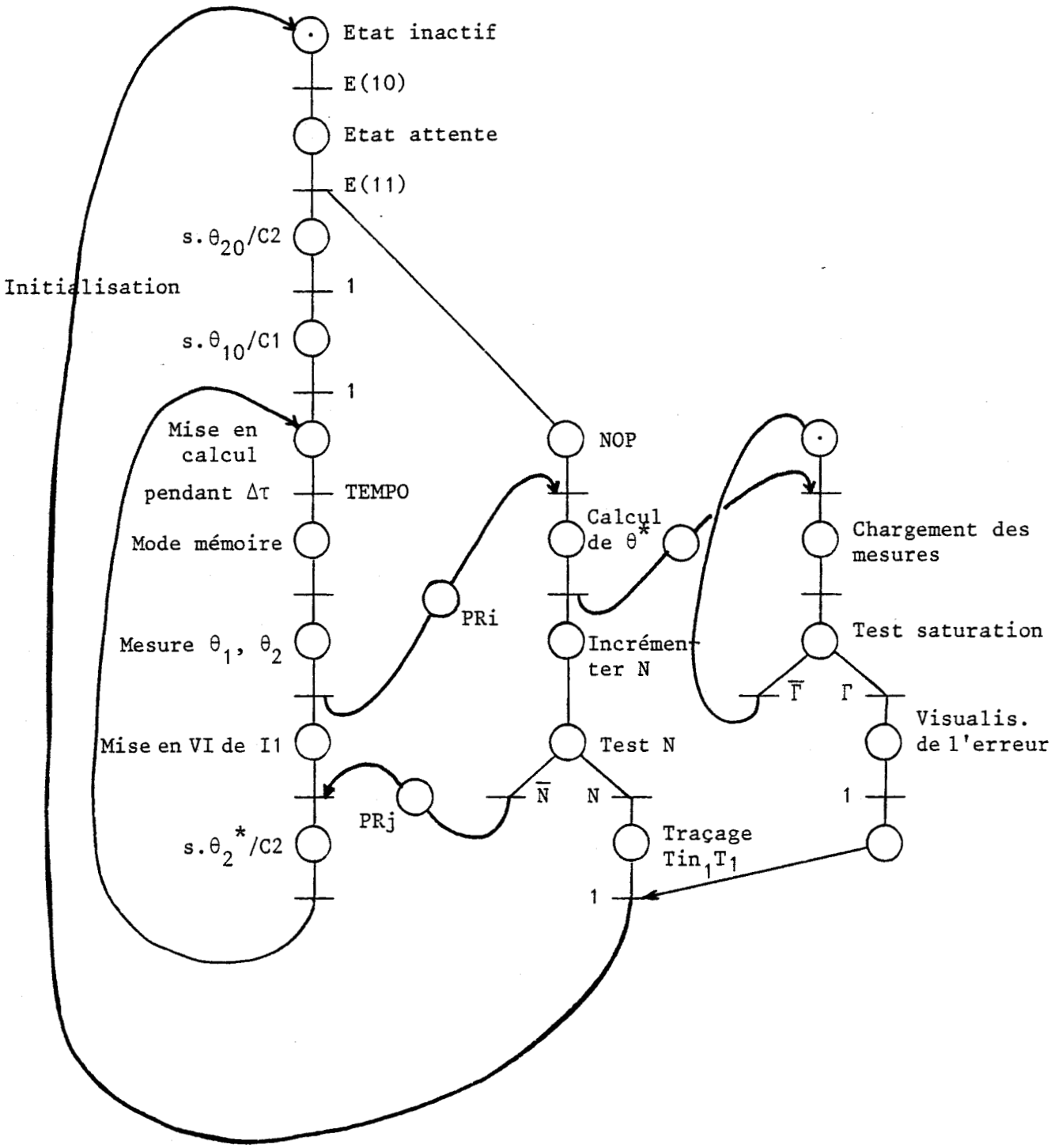


Figure 16

IV.4 - CONCLUSION

Nous avons pu illustrer dans ce dernier chapitre, la mise en œuvre du logiciel présenté dans ce mémoire. Les deux exemples présentés, effectivement réalisés, donnent une idée du large domaine d'application d'un tel logiciel aussi bien en contrôle des processus qu'en simulation.

Dans le premier cas, nous avons essentiellement cherché à montrer que la décompisition en tâches structurées d'une commande séquentielle conduisait à la synthèse d'un automatisme bien décomposé et pleinement efficace sur le plan du temps |22||23|. En effet, la recherche des sous-ensembles à évolutions relativement indépendantes, conduit à définir systématiquement des tâches de contrôles à évolutions simultanées. Dans ce cas, les liaisons entre tâches traduisent très simplement les contraintes de dépendances. La modularité de chaque tâche permet par ailleurs, une vérification et une compréhension plus aisées des programmes.

Dans le second cas, nous avons pu indiquer sur un exemple simple, les possibilités de traitement de variables analogiques dans le sens d'une application en simulation. Il apparaît donc clairement que l'ensemble présenté ici permet à la fois de traiter des applications de commande séquentielles et de régulation. Le langage proposé peut par ailleurs, constituer après quelques adaptations, un logiciel de simulation hybride d'assez haut niveau.

D'autres systèmes de simulation existent ; il faut signaler notamment |14| |15| avec comme application notamment |24|.

CONCLUSION

CONCLUSION GENERALE

Dans ce mémoire, nous avons présenté les caractéristiques d'ensemble d'un logiciel temps réel visant à simplifier l'approche des problèmes de commande de processus. Nous avons tenu à proposer une solution basée sur l'utilisation d'une structure micro-informatique permettant de ce fait, la prise en compte des calculs ou de la gestion de textes.

Il ressort des premières applications de ce travail, que lorsqu'on dispose d'un logiciel permettant réellement d'effectuer du contrôle multitâches, il est possible de traiter de manière structurée et modulaire le contrôle d'un processus industriel. La décomposition du système en sous-systèmes doit être effectuée d'une part en assurant l'indépendance locale des différentes tâches et d'autre part, en traduisant les contraintes de liaison par des relations banalisées.

Il convient également de rechercher le plus haut niveau de parallélisme afin d'éviter les temps d'attente inutiles dus à une mauvaise organisation du contrôle, ce qui est obligatoirement le cas pour une gestion monotâche.

Dans ce sens, certaines améliorations peuvent être apportées en augmentant par exemple le nombre de processeurs chargés de traiter l'ensembles des tâches en cours. Dans ce cas, la parallélisme apparaît dans l'architecture de l'automate et peut conduire, de toute évidence, à une amélioration sensible des temps de réponses à un ensemble de sollicitations extérieures. Un moyen terme relativement facile à réaliser sur la base de ce travail, consisterait à mettre en œuvre deux processus ; l'un se chargeant de le gestion des évènements , des fonctions d'évènements et des affectations d'entrées/sorties, l'autre se chargeant de la gestion des tâches.

Une autre amélioration très sensible, que nous comptons développer prochainement, concerne l'implantation directe du compilateur au niveau de l'automate afin de constituer un ensemble plus autonome.

BIBLIOGRAPHIE

|1| M. HACK
"Pétri nets languages"
M. I. T., TR 159, March 1975.

|2| G. BERTHELOT
"Vérification de réseaux de Pétri"
Doctorat de 3ème Cycle, Paris, 1978.

|3| R. VALETTE
"Sur la description, l'analyse et la validation des systèmes de la commande parallèle"
Thèse d'Etat, Toulouse, 1976.

|4| S. THELLIEZ
"Pratique séquentielle et Réseaux de Pétri"
Editions Eyrolles 1977.

|5| D. CORBEEL
"Schéma de cablage et schéma de contrôle"
Thèse de Docteur Ingénieur, Lille, 1979.

|6| BLANCHARD
"Comprendre, maîtriser et appliquer le Grafcet"
Edition CEPADUES 1979.

|7| GRAFCET
Rapport final de la commission : "Normalisation du cahier des charges d'un automatisme logique"
Août 1976.

|8| ADEPA
"Grafcet, diagramme fonctionnel des automatismes séquentiels"
Mai 1979.

|9| M. MOALLA
"Modèle de représentation du cahier des charges d'un automatisme complexe"
Rapport de contrat D. G. R. S. T., 1977.

|10| L. FLAHAUT
"Logiciel graphique de description de réseau de Pétri"
D. E. A. Electronique, Lille I, Juin 1981.

|11| G. MICHEL, C. LAURGEAU, B.ESPIAU
"Les automates programmables industriels"
Editions DUNOD Technique, 1979.

|12| J. PEYROCAT
"Les automates programmables se banalisent grâce aux bas de gamme"
Mesure Régulation Automatisme, n° 6/7, Juin/Juillet 1981.

|13| J.M. TOULOTTE, P. BRARAT
"L'automate programmable avec implantation directe des graphes fonctionnels est-il une solution d'avenir ?-
Nouvel Automatisme n° 11, Janvier/Février 1980.

|14| M. MARQUETTE
"Sur un système de simulation hybride"
Thèse de Docteur-Ingénieur, 28 Juin 1977, Lille I.

|15| G. DAUPHIN
"Logiciel de gestion de processus P 856"
D. E. A. Electronique, Lille I, 1979.

|16| J. DEFRENNE
"Implantation de réseaux de Pétri sur automate biprocesseur"
Thèse de 3ème Cycle, Lille I, 29 Juin 1979.

|17| D. CORBEEL, J.C. GENTINA
"Analyse et synthèse de commande des processus par réseaux de Pétri"
Congrès MIMI, Juin 1978, Zurich.

|18| D. DALEMAGNE, D. CORBEEL, J.C. GENTINA
"Digital control of Processes by micro-processors. Application to machine tool"
Informatica, Octobre 1979, Ljubljana (Yougoslavie).

|19| D. CORBEEL, C. VERCAUTER, J.C. GENTINA
"Méthodologie de description des systèmes de processus et de gestion d'erreur"
MINI et MICRO Computers, 9/11 Septembre 1980, Budapest.

|20| D. CORBEEL, C. VERCAUTER, J.C. GENTINA
"Formal description of processes system and exception handling"
IASTED International Symposium Modelling, Identification and control, 18/21 Février 1981, Davos.

|21| D. CORBEEL, C. VERCAUTER, J.C. GENTINA
"Méthodologie de description des systèmes de processus et de gestion d'erreur"
Convention Informatique latine, 6/81, Barcelone.

|22| D. CORBEEL, J.C. GENTINA
"Specifications and conception of real time control systems"
IASTED 09/81, Le Caire.

|23| D. CORBEEL, C. VERCAUTER, J.C. GENTINA
'Exécution d'émulation du contrôle des processus en temps réel"
AMSE, 09/81, Lyon.

|24| G. DAUPHIN, J.C. GENTINA, P. BORNE
"On a hybrid simulation software"
IMACS, Simulation of distributed parameters and large scale systems, Patras, Grèce, Octobre 1979.

A N N E X E S

1 - EDITION D'UN PROGRAMME UTILISATEUR

2 - UTILISATION DU COMPILATEUR

3 - AJOUT ET MODIFICATION D'INSTRUCTIONS

4 - ECHANGEUR THERMIQUE

5 - SIGNIFICATION DES TABLES UTILISEES PAR LE MONITEUR INTERPRETEUR

6 - PROGRAMME DE L'EXEMPLE D'APPLICATION DE LOGIQUE SEQUENTIELLE

ANNEXES

1 - EDITION D'UN PROGRAMME UTILISATEUR

Un programme utilisateur peut être généré depuis un numéro quelconque en utilisant le programme d'édition CALCUL. Son appel se fait de la manière suivante :

RUN % CALCUL

La liste des commandes utilisées est la suivante :

NEW <nom> <RC>	Provoque une numérotation automatique des lignes qui cesse quand la dernière ligne rentére est 0 ou FIN.
NUM <RC>	On reprend la numérotation des lignes.
OLD <nom> <RC>	Va chercher en mémoire ce programme. Sort "PRET" une fois l'opération terminée.
LEC <nom> <RC>	Permet la lecture de rubans.
CAT <RC>	Donne l'état du catalogue, puis "PRET".
STO <RC>	Permet de stocker un programme, puis "PRET".
REN <nom> <RC>	Pour changer le nom de votre programme.
DES <nom> <RC>	Supprime ce nom du catalogue, puis "PRET".
LIS <RC>	Liste le programme, puis "PRET".
LIS 3-5, 12 <RC>	Liste les lignes 3, 4, 5, 12, puis "PRET".
DEL 1-4, 7 <RC>	Supprime les lignes 1, 2, 3, 4, 7, puis "PRET".
	Attention : le contenu de la ligne est remplacé par " ".
	FOR, RUB, STO, LST ignorent les lignes "0" et " ".
FOR <RC>	Formate le programme suivant : <Etiquette>, <Code Opération>, <Opérande>, puis "PRET".
LST <RC>	Envoie un listing sur imprimante puis "PRET".
RUB <RC>	Permet de sortir un ruban, répondre aux questions puis "PRET".
INS <nu> <RC>	Permet d'insérer des lignes à partir du numéro donné ; "? RC" détermine la fin d'insertion, puis "PRET"

Dans tous les cas d'erreur, le compilateur reformule sa question.

PDP : Attendez ... Je compile !

La compilation est alors lancée.

Puis, si aucune erreur ne sort :

PDP : Programme compilé : l'utilisateur donne un nom qu'il devra répéter à la question suivante : "nom du programme compilé :"

II - TRAITEMENT DES ERREURS EN COURS DE COMPILATION

Il y a deux types d'erreurs en cours de compilation :

- Les erreurs fatales : à quelques exceptions près, elles provoquent une déconnection immédiate du programme.

- Les erreurs corrigibles : il est possible de modifier la ligne en cause, pour supprimer l'erreur.

II.1 - Les erreurs fatales

C'est l'exemple d'une tentative d'utilisation d'un tableau dans une instruction alors qu'aucune directive DIM n'a été déclarée.

Ce type d'erreur provoque la sortie :

- d'un message d'erreur explicatif

- du message suivant :

"Vous devez corriger votre programme : ça serait trop pénible de le faire sous contrôle du compilateur ... désolé"

- et déconnecte.

Cas particulier : Il existe cependant certaines erreurs fatales corrigibles. Ce type d'erreur ne se rencontre que dans la partie déclarations.

Ex : Présence de 2 lignes de déclaration DIM dans la première partie du programme.

Adresse courante de la ligne		Octets significatifs du ruban	La signification
Hexa.	Déci.		
Ex : F056	61526	00000010	Nombre de tâches = 2

Ce tableau est suivi d'une table des étiquettes.

Remarque : Pour HOR : la valeur occupe 2 octets.
8080 : La valeur imprimée est le nombre qui doit diviser 250 000 pour obtenir la fréquence demandée. (Par défaut, ce nombre est pris égal à 62500).

La séquence générale des octets est la suivante :

	Nombre d'octets utilisés	Signification
FA50 :	1	Nombre de tâches (soit ta)
"	1	Mot d'état du programme (tâches initialisées)
"	1	Nombre total de sémaphores (soit s)
"	1	Nombre de tables (soit t)
"	1	Nombre de fonctions (soit f)
"	2	Horloge
" ta fois :	2	Adresse de début de tâche
" s fois :	1	Valeur initiale du sémaphore
" t fois :	2	Longueur de la table
" f fois :	4	Mot d'état d'une fonction : 32 bits divisés en 4 octets, des poids forts vers les poids faibles soit bit 31 à bit 0. La mise à 1 d'un bit, correspondant à la présence dans la fonction de l'évènement associé (de même numéro). Ex : FNC F3 = 32 + 3 + 16 donne, dans l'ordre, les 4 octets suivants : 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
"	x	x octets correspondants à la description de toutes les tâches.

3 - AJOUT ET MODIFICATION D'INSTRUCTIONS

Au niveau du compilateur, il est possible de modifier des instructions ou d'en ajouter de nouvelles. Ces opérations sont réalisées en utilisant les porgrammes MIAJO et SAIS qui manipulent les fichiers renseignements des instructions.

I.1 - LES FICHIERS INSTRUCTIONS

Une instruction de format général ((COD) COP Opé1, Opé2, ...) est traitée par le compilateur et codée sur le ruban en 4 ou 6 Octets. (La description des tâches, seconde partie des octets significatifs du ruban, est un ensemble de blocs de 4 ou 6 Octets).

Ce moule de 4 ou 6 octets a la forme suivante :

Code Déclenchement
Code Opération
Opérande
Opérande

ou

Code Déclenchement
Code Opération
Opérande
Opérande
Opérande
Opérande

Le choix de l'un ou l'autre varie selon le nombre d'opérandes nécessaires à la définition de l'instruction. Le nombre d'octets réservés aux opérandes est ainsi fixé à 2 ou 4).

Le fichier instructions contient donc tous les renseignements qui servent à structurer ces blocs. (Ces renseignements peuvent être sortis sur imprimante en faisant RUN % LISTE).

Ils comprennent :

		Exemple
n	Numéro de l'instruction : C'est un entier décimal positif (Ordre de création de l'instruction)	8
v	Valeur du code instruction : v = n si Bloc à 4 Octets v = n +128 si Bloc à 6 Octets v indique à l'interpréteur la longueur du bloc de l'instruction que celui-ci lit	136
c	Code, en clair, de l'instruction : Chaîne alphabétique d'au plus 3 caractères	BRP
o	Nombre d'octets de l'instruction : 4 ou 6 sur ruban	6
sp	Séquence des sous-programmes appelés, à l'intérieur du compilateur, par la lecture de l'instruction et correspondant à la séquence des opérandes de la ligne-instruction : Opé1, Opé2 ...	10, 3, 12

Le tableau ci-dessous donne le codage de ces différents sous-programmes.

Code	Opérande associé au sous-programme	Nombre d'octets accupés dans le bloc sur le ruban
1	Evènement	1
2	Prédicat	1
3	Paramètre	1
4	Sémaphore	1
5	Tâche	1
6	Port IO	1
7	Table	2
8	Bit sélectionné	1
9	Valeur sur deux octets	2
10	Etiquette	2
11	Priorité	1
12	...	1 Octet nul
13	Valeur sur 1 octet	1

d'octets "o" ou sur la séquence "sp" des sous-programmes appelés.

Pour cela, faire depuis son numéro propre : RUN % MIAJO, ce qui entraîne la sortie des messages suivants (en majuscules) :

CALCULATEUR UTILISE : 8080
CODE RECH. : Inscrire le code alphabétique de l'instruction à modifier, puis l'état actuel des renseignements apparaît .

Ex :

NUMERO	CODE	VALEUR	OCTETS	SOUS-PROGRAMMES
47	ADD	47	4	3 3

Dans l'ordre n, c, v, o et sp.

MODIFICATION SUR NU, CO, OC, SP : Inscrire la modification désirée (Numéro, Code, Octets ou sous-programmes).
Faire <RC> pour arrêter.

puis effectuer la modification après apparition du message associé :
NUMERO :
ou CODE :
ou NBRE D'OCTETS :
ou S. P. :

Après modification, l'état des nouveaux renseignements apparaît pour vérification.

Lorsque la modification est terminée (avec <RC>, cf plus haut), le programme MIAJO chaîne automatiquement sur un tri du fichier modifié :

DEBUT DU TRI ... annonce le tri sur le code alphabétique.

II.2 - Ajout

L'appel du programme se fait depuis son propre numéro sur PDP en faisant RUN % SAIS. Répondre ensuite aux différents messages :

CALCULATEUR UTILISE :

NUMERO : Vérifier que le numéro choisi n'est pas déjà affecté
(en utilisant LISTE).

NBRE D'OCTETS :

SP :

puis

DEBUT DU TRI ...

TAB : 8 Octets

1	0 0 0 0 0 0 1 0 0 0 0 0 1 0
8	1 0 0 0 0 0 0

Table :

Chaque ligne contient les puissances successives de 2 en binaire.

REC(i) : 6 Octets

1	Code Déclenche.
2	Code opérat.
3	Code opérande
4	"
5	"
6	"

Table zone de travail :

Elle contient les codes instructions de la ligne du programme utilisateur en cours.

TABLSP : 100 Octets

1	
2	
3	
100	

Table des Adresses des S/Programmes :

Chaque ligne contient l'adresse du sous-Programme de l'interpréteur.

Cette table est chargée en mémoire morte.

IDEBU2 : 16 Octets

1	
2	
16	

Table des arrêts modifiables :

Elle contient les adresses d'arrêt lors d'une mise au point.

RTN1, RTN2 : Contient l'adresse de retour à l'interpréteur lors d'un appel d'un sous-programme par l'instruction CPN.

N : Nombre total des tâches.

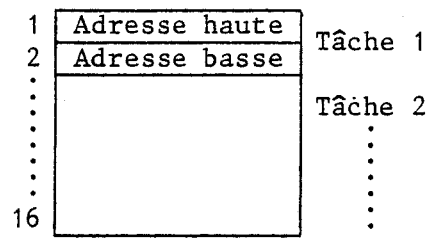
NTAC : Numéro de la tâche actuelle en cours d'exécution.

LN : Nombre des codes déclenchement attendus.

NS : Nombre des sémaphores utilisés.

5 - SIGNIFICATION DES TABLES UTILISEES PAR LE MONITEUR INTERPRETEUR

PPRG : 16 Octets



Pointeur du programme :

Il indique l'adresse de chaque tâche.

TTR : 8 Octets

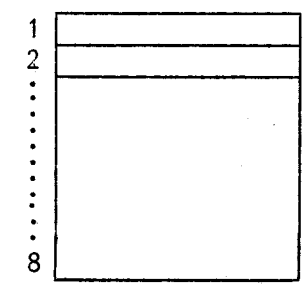


Table des tâches registrées :

Elle indique l'état des tâches
01H : Tâche active et attente d'exécution
80H : Tâche en cours d'exécution ou en attente d'un code déclenchement ou en attente d'un sémaphore.
8FH : Tâche interrompue à la suite d'un dépassement du temps alloué.
FFH : Tâche en attente d'un sémaphore.

EVE : 63 Octets

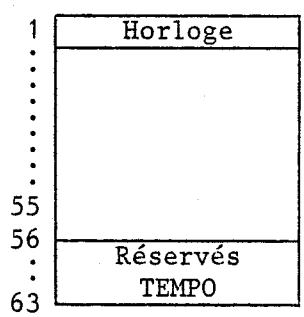


Table des Evènements :

0000.0000 : Evènement absent

1000 0000 : Evènement présent

0,000 0100 : Evènement absent codé en prédicat 4

Evènement 1 réservé pour l'horloge

Evènements 56 à 63 réservés pour les temporisations.

PRE : 8 Octets

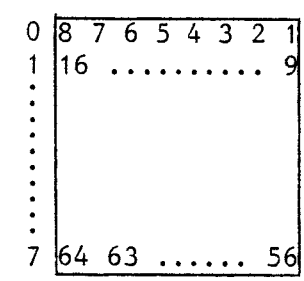


Table des prédicats :

Pour chaque bit de la table, correspond 1 prédicat :

0 : Le prédicat correspondant est nul

1 : Le prédicat correspondant est égal à 1.

PARA : 63 Octets

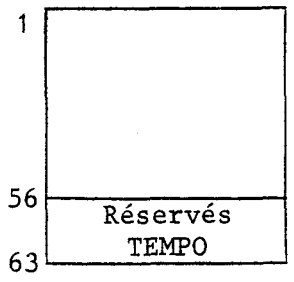


Table des paramètres :

Chaque octet contient la valeur du paramètre correspondant.

Les 8 derniers (56 - 63) sont réservés pour les temporisations.

MAS : 6 Octets

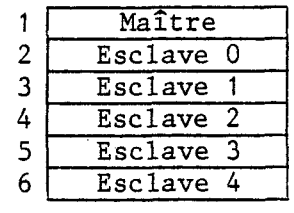


Table des masquages des 8259 :

Chaque bit de chaque port signifie :

1 : Interruption ou évènement associé est masqué

0 : Interruption ou évènement démasqué

PORT : 6 Octets

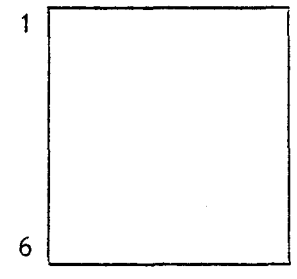


Table des valeurs des ports :

Elle correspond aux valeurs des ports d'entrée/sortie.

CD : 8 Octets

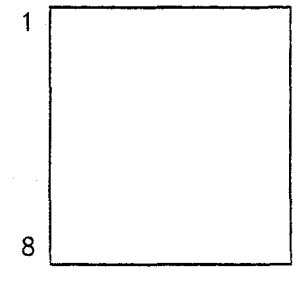


Table des codes des déclenchements attendus :

0000.0000 : pas de code déclenchement

10, n° Ei : Attente d'un évènement i

11, n° PRi : Attente du prédicat i

01, n° Fi : Attente de la fonction i

TAC : 8 Octets

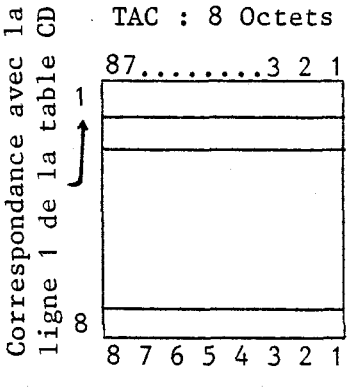
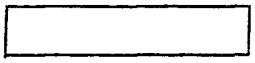


Table des tâches en attente d'un code déclenchement.

Elle va de paire avec la table CD. Pour chaque ligne CD correspond une ligne de TAC contenant la ou les tâches en attente de ce code de déclenchement.

LN : 1 Octet



Contient le nombre des codes de déclenchements attendus à la suite d'une scrutation des tâches.

FEV : 32 Octets

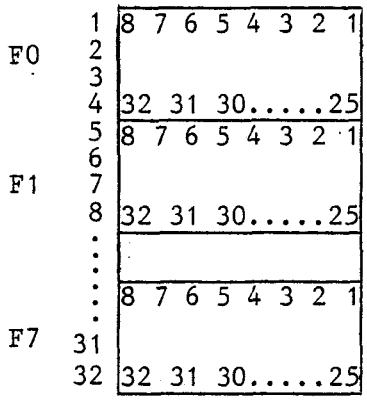


Table Fonction/Evènement :

Table contenant les évènements associés à chaque fonction

REFUS : 8 Octets

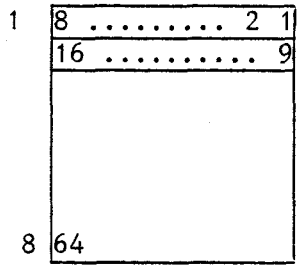


Table de REFUS d'évènements.

Chaque bit correspond à un évènement :

1 : Lors de l'arrivée de l'évènement, il sera masqué et sera remplacé par un 0.

0 : L'évènement est accepté (rangé dans EVE, PRE ou FNC) puis masqué.

DEVEFS : 32 Octets

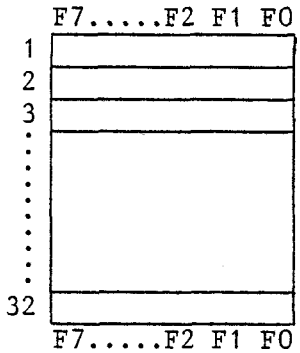


Table des évènements demandés par des fonctions

Chaque ligne correspond à un évènement pour lequel le ou les fonctions demandées sont associées.

FNC : 1 Octet

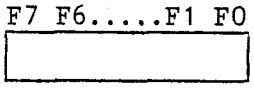


Table de l'état des fonctions :

Si Fi est nul : la fonction correspondante est désactivée

Si Fi = 1 : La fonction correspondante est activée.

ETAT : 16 Octets

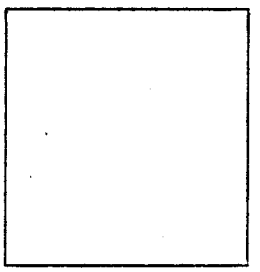


Table des ETATS de sémphores :

Chaque ligne correspond à un sémaphore dont le contenu est l'état du sémaphore.

POINTS : 16 Octets

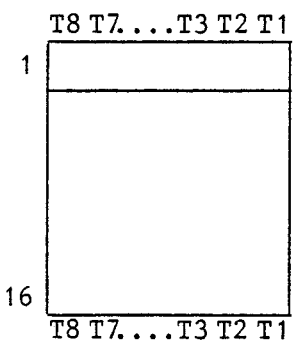
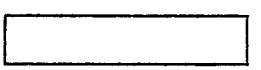


Table pointeur des Sémaphores :

Pour chaque sémaphore, est associé la ou les tâches qui le demandent.

NRQS : 1 Octet



Nombre des requêtes de sémaphores attendues :

PRIO : 8 Octets

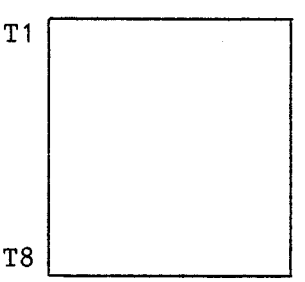


Table des Priorités :

Elle indique les priorités pour chaque tâche lors d'une requête de sémaphore.

POINTØ : 9 Octets

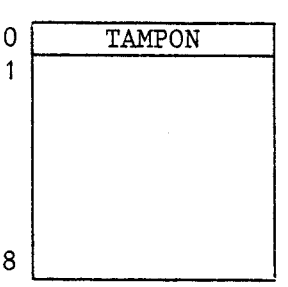


Table de détermination des priorités :

Table utilisée pour déterminer la priorité la plus élevée lors d'une requête d'un même sémaphore.

```
NOM DU PROGRAMME: ROBOT1
001                         PRO P1,P2,P3,P4,P5
002                         INI P1,P2,P3,P4,P5
003                         SEM SEMA,BRAS
004                         %DEB
005                         BEGIN P1
006                         DEB1:LDP P10,3
007                         (PR50)CPR PR50
008                         ENB E33,PR10
009                         RET1:CPN IMPROT
010                         TPR RET1,PR10
011                         REE E33
012                         RET2:CPN IMPROT
013                         CPN TROROT
014                         TPR RET2,PR11
015                         RET3:(PR33)CPR PR33
016                         RQS BRAS,1
017                         CLP P11
018                         RET4:CPN DESMON
019                         TPR RET4,PR12
020                         RET5:CPN IMPROT
021                         CPN TROROT
022                         TPR RET5,PR11
023                         CLP P11
024                         RQS SEMA,1
025                         CPN ARAIM
026                         (E25)RLS SEMA
027                         CPN IMPROT                  !1/4 TOUR
028                         LPR PR13
029                         RLS BRAS
030                         JMP RET3
031                         END
032                         BEGIN P2
033                         DEB2:NOP
034                         (PR32)CPR PR32
035                         OP0 I01,5                   !U3+
036                         (E14)LPR PR21
037                         OP1 I01,5
038                         TEMP2 5
039                         OP0 I01,7                   !U4+
040                         TEMP2 5
041                         OP1 I01,7
042                         (E15)OP0 I01,5              !U3+
043                         TEMP2 5
044                         OP0 I02,1                   !U5+
045                         TEMP2 5
046                         OP1 I02,1
047                         (E18)OP0 I00,1              !MP1
048                         OP0 I03,3                   !T1+
049                         TEMP2 5
050                         OP1 I03,3
051                         (E30)OP0 I03,4              !T1-
052                         TEMP2 5
053                         OP1 I03,4
054                         REE E16
055                         MSQ E16
056                         (E29)OP1 I00,1
057                         OP0 I01,8                     !U4-
058                         TEMP2 5
059                         OP1 I01,8
060                         (E16)OP0 I00,2              !P2+
061                         OP0 I00,3                   !F+
062                         OP0 I03,1                   !T2+
063                         OP0 I03,5                   !T3+
064                         TEMP2 5
```

```
065                     OP1 IO3,1
066                     OP1 IO3,5
067                     (E27)REE E17
068                     MSQ E17
069                     (E31)OP0 IO3,2
070                     OP0 IO3,6               !T3-
071                     TEMP2 5
072                     OP1 IO3,2
073                     OP1 IO3,6
074                     (E28)REE E13
075                     MSQ E13
076                     (E32)OP1 IO0,2          !P2-
077                     OP1 IO0,3               !F-
078                     OP0 IO2,2               !U5-
079                     (E17)OP1 IO2,2
080                     OP1 IO1,5
081                     OP0 IO1,6       !U3-
082                     (E13)OP1 IO1,6
083                     LPR PR24
084                     (PR42)CPR PR42
085                     LPR PR23
086                     REE E14
087                     MSQ E14
088                     REE E15
089                     MSQ E15
090                     REE E18
091                     MSQ E18
092                     JMP DEB2
093                     END
094                     BEGIN P3
095                     DEB30:CPR PR31
096                     DEB31:(PR13)RQS SEMA,1
097                     CPR PR13
098                     OP0 IO2,3               !U6+
099                     TEMP3 5
100                     OP1 IO2,3
101                     (E19)OP0 IO2,4          !U6-
102                     REE E20
103                     MSQ E20
104                      (E20)OP1 IO2,4
105                      LPR PR33
106                      RLS SEMA
107                     TPR DEB32,PR31
108                     (PR23)CPR PR23
109                     DEB32:LPR PR31
110                     OP0 IO1,4               !U2+
111                     REE E11
112                     MSQ E11
113                      REE E10
114                      MSQ E10
115                     (E11)OP1 IO1,4
116                     OP0 IO1,1               !U1+
117                     (E10)OP1 IO1,1
118                     REE E9
119                     TEMP3 10
120                     LPR PR32
121                     (PR21)CPR PR21          !FIN GUIDAGE PIECE
122                     OP0 IO1,2               !U1-
123                     (E9)OP1 IO1,2
124                     OP0 IO1,3               !U2-
125                     REE E12
126                     MSQ E12
127                     TEMP3 5
128                     OP1 IO1,3
129                     (E12)JMP DEB31
130                     END
```

```
131                     BEGIN P4
132                     DEB4:NOP
133                     (PR24)CPR PR24
134                     RQS BRAS,1
135                     CPN IMPROT
136                     CPN IMPROT                  !1/4 TOUR
137                     DEB41:CPN DESMON
138                     TPR DEB41,PR12
139                     LPR PR42
140                     DEB42:CPN IMPROT
141                     CPN TROROT
142                     TPR DEB42,PR11
143                     CLP P11
144                     OPO IO2,5
145                     TEMP4 5
146                     OP1 IO2,5
147                     TEMP4 15
148                     REE E21
149                     CPN ARAIM
150                      TEMP4 5
151                     OPO IO2,6                   !U7+
152                     (E21)OP1 IO2,6
153                     DEB43:CPN IMPROT
154                      CPN TROROT
155                     TPR DEB43,PR11
156                     CLP P11
157                     RLS BRAS
158                     JMP DEB4
159                     END
160                     BEGIN P5
161                     DEB5:LPR PR33
162                     DT P5
163                     (E22)LPR PR50
164                     (E24)OPV IO3,/05            !T1-.T2-.T3-
165                     OPV IO2,/D7                 !U7+.U6-
166                     OPV IO1,/79                 !U2-.U1-.U4-
167                     TIN P1
168                     TIN P2
169                     TIN P3
170                     TIN P4
171                     TEMP5 5
172                     OPV IO3,/FF
173                     OPV IO2,/FF
174                     OPV IO1,/F0                 !U1-
175                     TEMP5 30
176                     OPV IO0,/FF                 !P1-.P2-.F-.A1-
177                     OP1 IO1,2
178                      OPO IO2,2
179                     TEMP5 5
180                      OP1 IO2,2
181                     OPO IO1,6                   !U3-
182                      TEMP5 50
183                     OP1 IO1,6
184                     REE E22
185                     REE E24
186                     JMP DEB5
187                     DESMON:MSQ E26
188                     CPR PR12
189                     OPO IO2,5                    !U7-
190                     TEMP5 5
191                     OP1 IO2,5
192                     TEMP5 15
193                     OPO IO0,6                    !A1-
194                     TEMP5 5
195                     OP1 IO0,6
```

```
197                 REE E21
198                 MSQ E21
199                 (E23)OP0 I00,7          !AI+
200                 OP0 I02,6               !U7+
201                 TEMP5 5
202                 OP1 I02,6
203                 REE E23
204                 (E21)MSQ E23
205                  TEMP5 5
206                 ENB E26,PR12
207                 RTN
208                 IMPROT:OP0 I00,5        !PLA
209                 TEMP5 5
210                 OP1 I00,5
211                 (E34)TEMP5 10
212                 RTN
213                 TROROT:CPR PR11
214                 IMP P11
215                 CME PR11,P11,P10
216                 RTN
217                 ARAIM:OP0 I00,6
218                 TEMP5 5
219                 OP1 I00,6
220                 OP1 I00,7
221                 RTN
222                 END
223                 *FIN
```