Le langage GRAFCET des automates programmables TELEMECANIQUE TSX27 et TSX47 : saisie graphique et interprétation

Sylvain Korzeczek

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE

DE GRENOBLE (C.U.E.F.A)

MEMOIRE

Présenté en vue d'obtenir

le diplôme d'ingénieur

du Conservatoire National des Arts et Métiers

spécialité

Informatique

par

Sylvain KORZECZEK

LE LANGAGE GRAFCET DES AUTOMATES PROGRAMMABLES TELEMECANIQUE TSX27 ET TSX47 : SAISIE GRAPHIQUE ET INTERPRETATION.

SOUTENU LE : 17 mars 1986

JURY

Président : J.Y. RANCHIN

Membres : L. BOLLIET

R. DOUCET

B. LEBLANC

à mes parents,

à ma femme,

à mes amis ....

Je remercie chaleureusement,

Monsieur RANCHIN, professeur au Conservatoire National des Arts et Métiers, qui m'a fait l'honneur d'accepter de présider le jury pour la soutenance de ce mémoire,

Monsieur BOLLIET, professeur à l'université de Grenoble II, pour ses judicieux conseils et ses constants encouragements lors de la rédaction de ce mémoire,

Monsieur DOUCET, responsable des études et développements logiciels de la société TELEMECANIQUE, pour l'intérêt qu'il a toujours manifesté à l'égard de mon travail, ainsi que pour ses fructueux conseils techniques,

Monsieur LEBLANC, chef de produit - marketing opérationnel à la division automatismes programmables de la société TELEMECANIQUE, qui a bien voulu m'honorer de sa participation à ce jury,

J'assure de ma gratitude les enseigants du CNAM (Conservatoire National des Arts et Métiers) ainsi que ceux du CUEFA (Centre de Formation Universitaire d'Education et de Formation des Adultes) qui m'ont permis l'accès à ce niveau de connaissances.

Je veux enfin assurer de mon amitié tous ceux qui ont contribué de près ou de loin au bon déroulement de ce travail.

S O M M A I R E .

DEUXIEME PARTIE.

TROISIEME PARTIE.

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

Le GRAFCET (GRAphe Fonctionnel de Commande Etape Transition) est un outil de description du cahier des charges des automatismes logiques. Cet outil est un diagramme fonctionnel c'est-à-dire une représentation graphique, donc concise et facile à lire, permettant de décrire tous les comportements attendus d'un automatisme de commande face aux réactions du processus automatisé. De plus, il permet une spécification rigoureuse et progressive du fonctionnement du système automatisé.

Aussi, dès sa présentation, le GRAFCET a été reconnu par l'industrie et l'enseignement comme un outil adapté, pratique et facile à mettre en oeuvre.

Cependant, malgré ce consensus au niveau langage de spécification, très peu d'efforts avaient été consentis par les constructeurs de constituants d'automatismes pour utiliser le GRAFCET comme langage de programmation.

Dans sa nouvelle gamme d'automates programmables TSX7, TELEMECANIQUE a décidé de proposer comme langage de programmation, le GRAFCET. Le développement des automates TSX27, TSX47 et de la console TSX-T407 s'est effectué au sein de la Direction Recherche et Développement de la société TELEMECANIQUE.

Les travaux présentés dans ce mémoire se situent dans ce cadre d'activité et décrivent les études relatives à la mise en oeuvre du GRAFCET en tant que langage de programmation.

A cet effet, ce mémoire est structuré de la manière suivante :

- une première partie est consacrée à une présentation générale de la gamme TSX7, des principaux constituants et des langages de programmation,
- la seconde partie présente les différents problèmes qu'il a fallu aborder pour mettre en oeuvre le GRAFCET en tant que langage de programmation,
- la troisième partie est une description des logiciels réalisés :
  - . au niveau des automates TSX27 et TSX47 pour l'aspect interprétation,
  - . au niveau de la console TSX-T407 pour l'aspect saisie graphique.

P R E M I E R E P A R T I E .

I PRESENTATION.

I.1 L'offre TELEMECANIQUE en automatismes "bas de gamme".

Les constituants d'automatismes proposés par TELEMECANIQUE permettent de couvrir un large éventail d'applications allant des automatismes séquentiels simples aux automatismes multifonctions plus évolués.

L'offre TELEMECANIQUE, au niveau qualifié de "bas de gamme" en termes d'étude de marché, se compose des éléments suivants : les automates TSX27 et TSX47, la console T407, les terminaux T307 et T107, et le réseau TELWAY 7.

Le TSX27 est un automate séquentiel compact destiné à concurrencer et à se substituer à des équipements électromécaniques comportant une vingtaine de contacteurs. Il convient parfaitement aux automatismes simples de petites machines autonomes et d'installations unitaires et répétitives. Il est proposé en 4 configurations de 20, 40, 60, ou 80 entrées/sorties.

Le TSX47 est un automate séquentiel modulaire et évolutif dimensionné pour piloter des applications comportant de 32 à 256 entrées/sorties. Il peut être équipé d'une gamme complète de modules d'entrées/sorties Tout Ou Rien et analogiques et peut être connecté au réseau TELWAY 7. Grâce à sa modularité et sa diversité d'interfaçages, le TSX47 est parfaitement adapté aux automatismes de machines et de petites installations fixes, et de plus il permet d'envisager l'évolution de l'automatisme.

Les automates TSX27 et TSX47, de part leurs capacités respectives en nombre d'entrées/sorties, ont une apparence externe différente : compacte et simple bac pour le TSX27, extensible, simple ou double bac pour le TSX47.

En fait, ces deux automates sont issus d'une même conception : leurs architectures internes, leurs puissances de traitement et les fonctionnalités qu'ils offrent sont très voisines. De plus, les langages de programmation qu'ils acceptent sont identiques. Selon le type de l'application, chacun d'eux pourra être programmé dans l'un ou l'autre des langages suivants :

- le langage à contacts (LADDER DIAGRAM) pour les aspects combinatoires de l'application,
- le GRAFCET pour les traitements de nature séquentielle de l'application.

Les deux langages sont bien sûr proposés en saisie graphique directe respectant les aspects de la norme en ce qui concerne le Grafcet et offrant de nombreux blocs fonctions préprogrammés en ce qui concerne le langage à contacts.

Le terminal portable TSX-T407 est l'outil permettant la saisie et la mise en oeuvre de ces deux langages. Ce terminal offre un ensemble complet de services pour développer, mettre au point et exploiter les programmes d'application des automates TSX27 et TSX47. Il possède une grande souplesse d'utilisation qui favorise l'établissement des multiples relations nécessaires entre le processus automatisé et les différentes catégories d'intervenants, tels que l'automaticien, l'exploitant ou le technicien de maintenance.

Les relations AUTOMATISME - AUTOMATICIEN sont facilitées

- par l'existence de touches dynamiques permettant à l'utilisateur une auto-formation rapide pour la saisie des programmes d'applications,

- la visualisation, l'animation et la mise au point interactive et progressive des programmes d'application.

Les relations AUTOMATISME - EXPLOITANT sont favorisées grâce à un accès permanent aux informations décrivant la situation complète de l'automatisme. La possibilité de modifier en temps réel certaines données du programme facilite grandement les tâches de réglage et de surveillance.

Enfin, les relations AUTOMATISME - TECHNICIEN DE MAINTENANCE sont rendues plus aisées grâce à la centralisation des diagnostics de défaillance. En cas de défaut de fonctionnement, que celui-ci se situe au niveau de l'automate ou au niveau des constituants qui lui sont connectés, le terminal affiche en clair le résultat des auto-tests déclenchés. Une investigation plus poussée de l'incident peut alors être menée grâce à des cartouches de tests spécifiques.

Le TSX T107 et le TSX T307 sont des terminaux de poche. Le premier permet d'effectuer des opérations de réglage et de dialogue opérateur. Le second offre en outre une possibilité de programmation simple en langage booléen.

Le dernier élément de l'offre TELEMECANIQUE "bas de gamme" est le réseau inter-automates transparent TELWAY 7. Ce réseau offre la possibilité de connecter jusqu'à 16 stations TSX47. Cela permet d'envisager une conduite décentralisée des procédés automatisés au moyen d'échanges de variables globales et de messages entre automates, ainsi qu'une supervision à distance de tels procédés puisque l'intégralité des fonctions offertes par un terminal connecté localement sont accessibles par le réseau.

I.2 Architecture matérielle des principaux constituants.

I.2.1 Automates programmables TSX27 et TSX47.

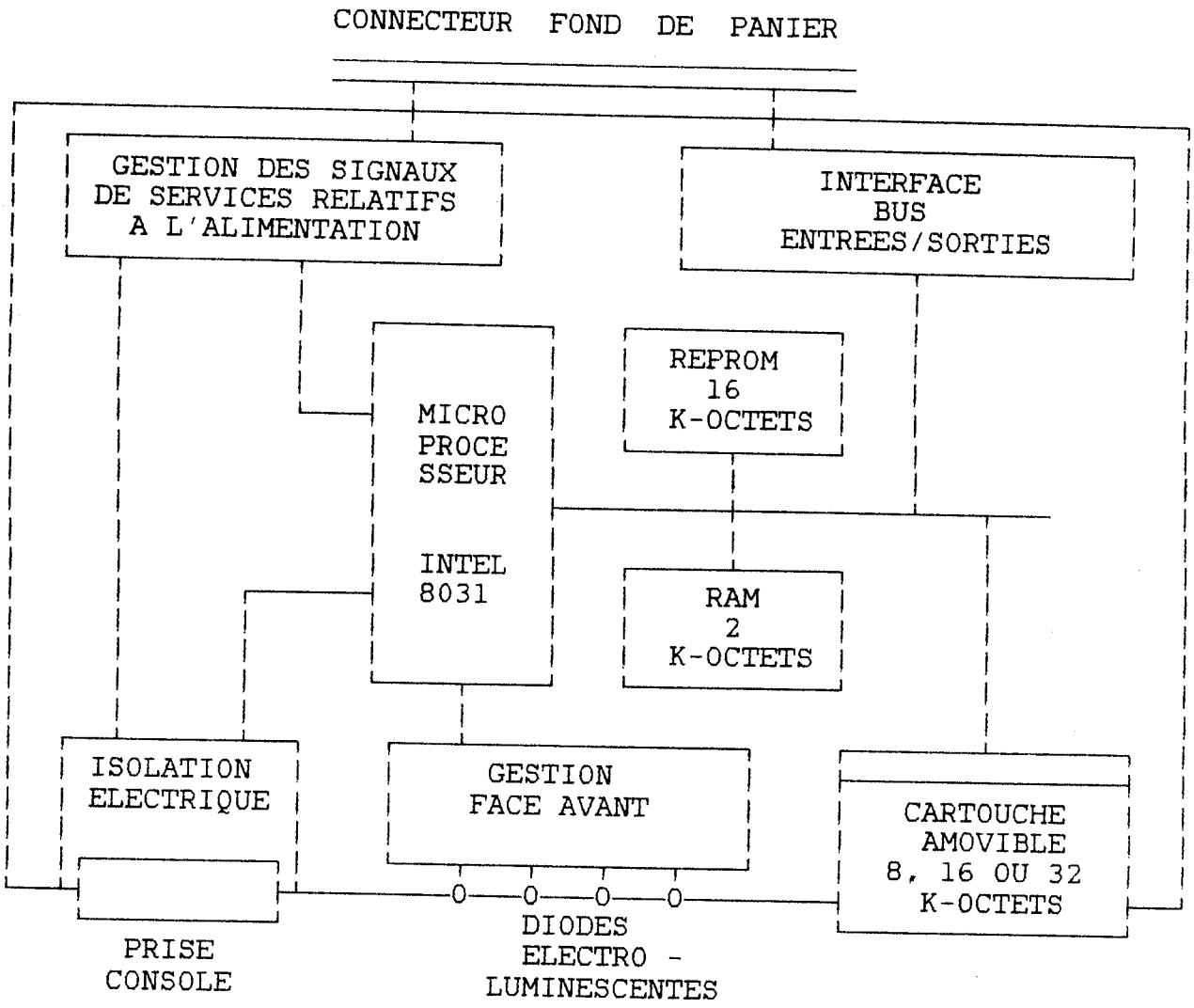
L'architecture des deux automates est pour la plus grande part commune. Elle s'organise autour d'un microprocesseur INTEL 8031, choisi pour ses nombreuses ressources internes. Ses instructions qui travaillent sur bits sont extrêmement commodes pour traiter les équations booléennes propres aux automates programmables. La RAM interne au processeur offre, elle aussi, cet accès aux bits et cet accès est très rapide. Une interface avec une liaison série, elle aussi intégrée, permet de gérer la ligne de la console. Les timers facilitent la programmation des fontions temporelles. Les ports d'entrées-sorties bidirectionnels sont très adaptés à la prise en compte des signaux de service et à l'affichage des voyants.

Ce processeur est connecté à trois mémoires :

- une mémoire programme en PROM, de 16 Ko,
- une mémoire de données en RAM, de 8 Ko,
- une mémoire de taille et de nature (RAM ou ROM) variables: la cartouche utilisateur. Elle contient le programme du système automatisé et est amovible. Cette cartouche est embrochable aussi bien dans l'automate que dans la console afin de favoriser les échanges de programmes. L'utilisateur peut ainsi changer, à son gré, son automatisme.

La différence essentielle d'architecture entre les modèles "27" et"47" se situe au niveau de l'interface avec les entées-sorties Tout Ou Rien. Le modèle "47" est prévu pour gérer jusqu'à 256 ES; il comporte pour cela un bus 4 bits parallèle. Le modèle "27" ne gère que 20, 40, ou 60 ES. Un bus série lui est suffisant comme interface.

Les deux automates peuvent être programmés par la même console "T407".



Architecture commune aux automates TSX27 et TSX47.

I.2.2 **Console de programmation TSX-T407.**

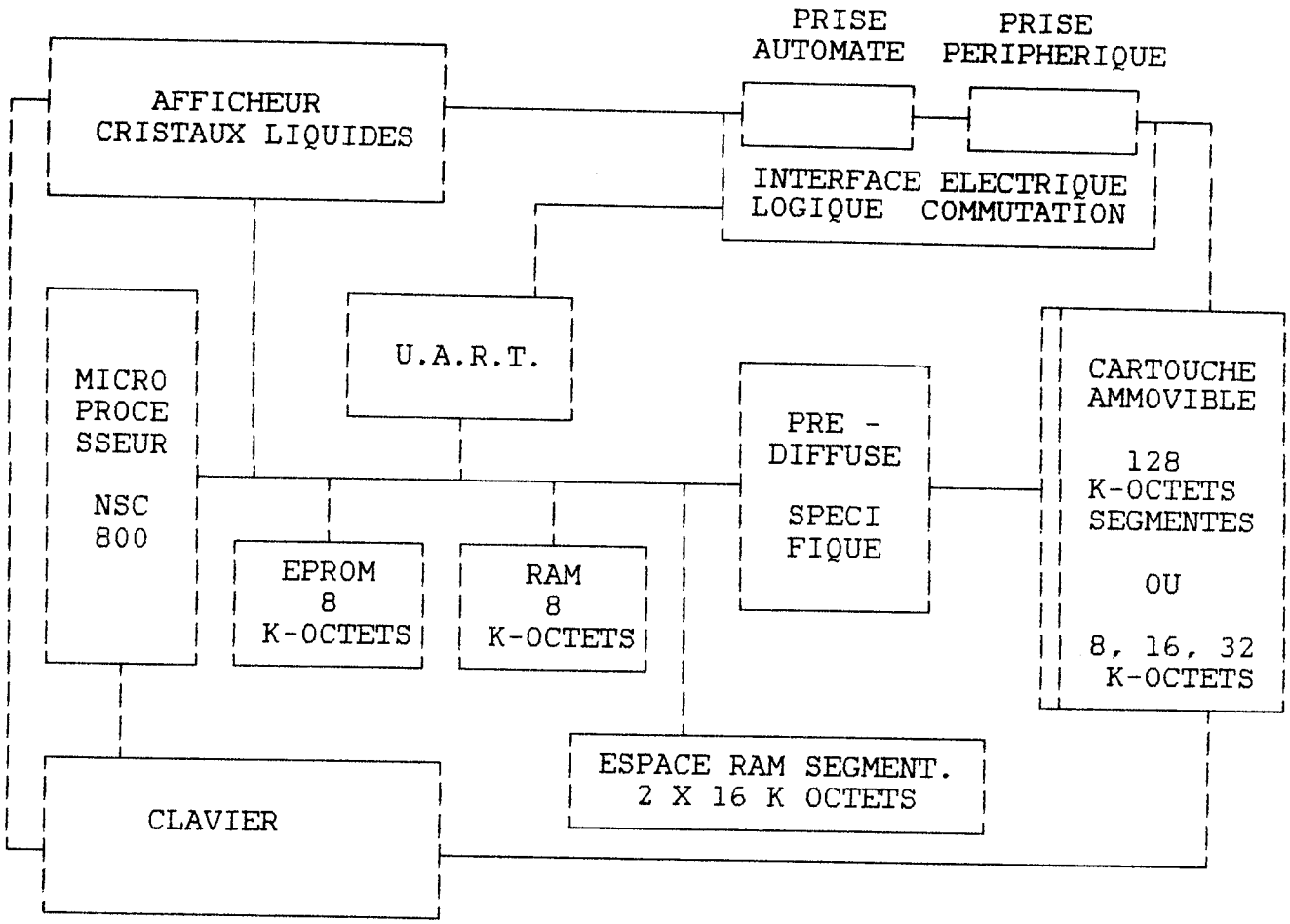
La console présente à l'utilisateur un clavier et un écran à cristaux liquides. Le clavier possède quatre types de touches : les littérales, les numériques, les fonctions fixes, et les dynamiques. Les dynamiques sont disposées juste sous l'afficheur; une signification variable leur est attribuée, en correspondance avec un libellé mnémonique affiché au dessus de chacune.

Le clavier et l'afficheur sont des périphériques de l'unité centrale. Celle-ci est un microprocesseur (le NSC 800) choisi pour sa puissance (le jeu d'instructions est celui du Z80) et aussi pour sa consommation très réduite.

Son espace mémoire est découpé en plusieurs sous-ensembles :

- une PROM de 8 Ko contient le logiciel de base,
- une RAM de 8 Ko contient toutes les données (système et utilisateur),
- le reste de l'espace d'adressage peut être alloué à la cartouche utilisateur. De plus une partie en est aussi accessible grâce à une commutation: cette RAM supplémentaire permet de simuler, sur la console, la mémoire de l'automate. De cette manière, un utilisateur peut programmer dans son bureau, et transférer plus tard som programme dans un automate.

Deux prises autorisent des liaisons vers l'extérieur, l'une en direction d'un automate bien sûr, l'autre en direction de périphériques divers: imprimante, magnétophone, modem,ou même autre calculateur. Ces deux liaisons sont pilotées par un interface commun.



Architecture console TSX-T407.

I.3 Les langages de programmation.

I.3.1 Langage à contacts.

Nous donnons ici les principaux éléments composant le langage à contacts, Ce langage, familier aux électriciens, s'adresse aux traitements logiques simples de type combinatoire.

I.3.1.1 Eléments de base.

Un programme en langage à contacts se compose d'une suite de RESEAUX exécutés séquentiellement par l'automate.

Dessiné entre deux barres de potentiel, un RESEAU est un ensemble d'éléments graphiques tels que CONTACTS, BOBINES, BLOCS FONCTIONS. Ces éléments sont reliés entre eux par des CONNEXIONS horizontales ou verticales.

Les symboles essentiels sont les suivants :

les CONTACTS à FERMETURE —| |— ou à OUVERTURE —|/|— symbolisent la présence ou l'absence d'information et référencent les variables d'entrées du RESEAU,

les BOBINES d'excitation de relais —( )— symbolisent le résultat d'équations logiques et référencent les variables de sorties du RESEAU,

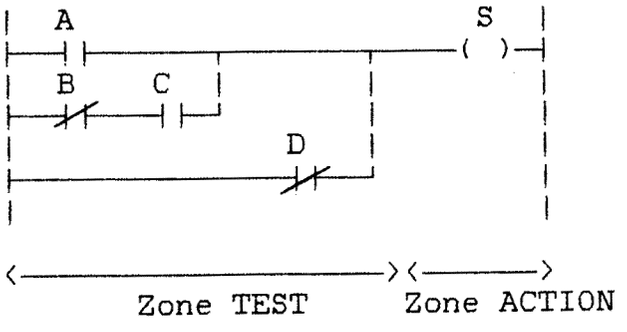
les CONNEXIONS électriques horizontales —— ou verticales | symbolisent les relations logiques ET, OU.

On distingue généralement deux zones dans un RESEAU :

- une zone TEST, destinée à recevoir les CONTACTS et les CONNEXIONS et déterminant des valeurs logiques,
- une zone ACTION, destinée aux BOBINES qui reçoivent ces valeurs.

Exemple.

On traduit $S = A + (\overline{B} \cdot C) + \overline{D}$ en langage à contacts de la manière suivante :

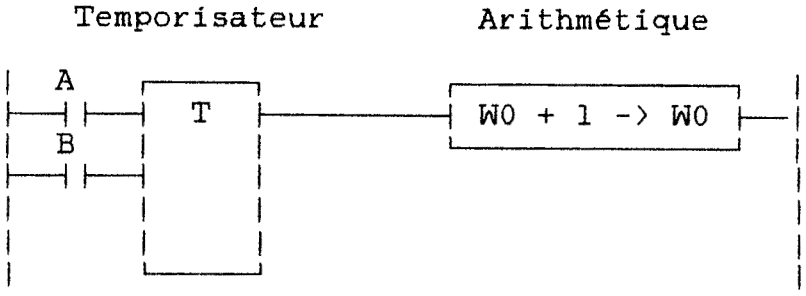


Les BLOCS FONCTIONS sont représentés sous la forme de rectangles :

- verticaux symbolisant l'accès à des fonctions séquentielles prédéfinies (temporisateurs, compteurs, registres à décalages),
- horizontaux symbolisant l'accés à des fonctions numériques programmables (arithmétique, conversions).

Les blocs verticaux s'insèrent en zone TEST, les blocs horizontaux se placent en zone ACTION.

Exemple.



I.3.1.2 **Règle de syntaxe.**

Il doit toujours exister au moins un chemin permettant de joindre la barre de potentiel gauche à la barre de potentiel droite.

I.3.1.3 **Règle d'évaluation.**

Un RESEAU est évalué du haut vers le bas et de la droite vers la gauche, en commençant par le coin en haut à gauche et en terminant dans le coin en bas à droite.

I.3.2 **Langage GRAFCET.**

Nous résumons ici les éléments principaux de la norme NF C 03-190. Cette norme définit le GRAFCET en tant que langage de spécification.

I.3.2.1 **Eléments de base.**

Les différents éléments composant un GRAFCET sont :

- des ETAPES auxquelles on associe des ACTIONS à effectuer,
- des TRANSITIONS auxquelles on associe des RECEPTIVITES,
- des LIAISONS ORIENTEES reliant les étapes aux transitions et les transitions aux étapes.

Une ETAPE caractérise un comportement invariant d'une partie ou de la totalité de la partie commande. Une étape est soit ACTIVE, soit INACTIVE. L'ensemble des étapes actives définit la SITUATION de la partie commande.

Une ACTION associée à une étape caractérise ce qui doit être fait chaque fois que cette étape est active.

Une TRANSITION indique la possibilité d'évolution entre étapes. Cette évolution s'accomplit par le FRANCHISSEMENT de la transition. Une transition est soit VALIDEE, soit NON VALIDEE. Elle est dite validée lorsque toutes les étapes en amont de la transition sont actives.

Une RECEPTIVITE associée à une transition est une proposition logique qui peut être soit VRAIE, soit FAUSSE.

Les LIAISONS ORIENTEES indiquent les voies d'évolution.

Symboles utilisés.

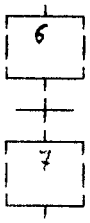
Une ETAPE est représentée par un carré repéré à la partie supérieure.

L'ENTREE et la SORTIE d'une étape sont figurées respectivement à la partie inférieure et à la partie supérieure.

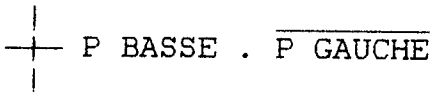
On repère une étape ACTIVE en plaçant un point dans la partie inférieure du symbole.

Une ETAPE INITIALE est représentée par un double carré.

Une ACTION est décrite à l'intérieur d'un ou plusieurs rectangles reliés au symbole de l'étape à laquelle elle est associée.

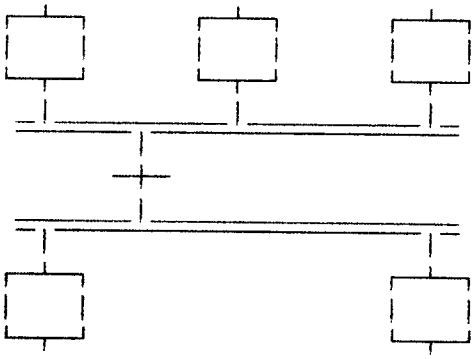


Une TRANSITION entre deux étapes se représente par une barre perpendiculaire aux liaisons orientées et est repérée de préférence à gauche de la barre.



Une RECEPTIVITE est inscrite, de façon littérale ou symbolique, de préférence à gauche de la barre.

Les LIAISONS orientées se représentent par des lignes horizontales ou par des lignes verticales.



Lorsque plusieurs étapes sont reliées à la même transition, les liaisons orientées correspondant à ces étapes sont regroupées en amont et en aval sur deux traits parallèles horizontaux.

I.3.2.2 **Règle de syntaxe.**

L'alternance ETAPE-TRANSITION et TRANSITION-ETAPE doit toujours être respectée quelle que soit la séquence parcourue.

I.3.2.3 **Règles d'évolution.**

Règle n° 1 : Situation initiale.

La situation initiale définit le comportement initial de l'automatisme et correspond aux étapes actives au début du fonctionnement. Dans les automatismes cycliques, les étapes initiales caractérisent la situation initiale.

Règle n° 2 : Franchissement d'une transition.

L'évolution d'un Grafcet correspond au franchissement d'une transition qui ne peut se produire que :

- lorsque cette transition est validée (étapes précédentes actives),
- et que la condition de transition est vraie (condition logique vérifiée).

Quand une transition devient franchissable, elle est obligatoirement franchie.

Règle n° 3 : Evolution des étapes actives.

Le franchissement d'une transition entraîne simultanément

- l'activation de toutes les étapes en aval de la transition,
- la désactivation de toutes les étapes en amont de la transition.

Règle n° 4 : Evolutions simultanées.

Plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle n° 5 : Activation et désactivation simultanées.

Si, au cours du fonctionnement de l'automatisme, une même étape doit être activée et désactivée, elle reste active.

I.3.3 **Complémentarité de ces langages.**

L'intérêt du langage GRAFCET réside dans le fait que la quasi-totalité du fonctionnement d'un système automatisé peut être exprimée par des évolutions séquentielles, même lorqu'elles sont de nature combinatoire. C'est aussi sa faiblesse car la nature combinatoire des évolutions n'apparaît pas sur le GRAFCET. (M. BLANCHARD).

C'est tout le contraire pour un langage tel que le langage à contacts, bien que ce dernier présente tout de même un certain caractère de séquentialité de par son exécution réseau par réseau.

La possibilité d'atténuer les faiblesses du premier par les forces du second sera une aide précieuse pour la mise en oeuvre du GRAFCET en tant que langage de programmation .

D E U X I E M E P A R T I E .

II MISE EN OEUVRE DU LANGAGE GRAFCET.

II.1 Contraintes et choix.

Le langage GRAFCET est avant tout un outil de spécification. C'est un langage simple et rigoureux pour définir le cahier des charges d'un système automatisé, quel que soit le degré de complexité du processus. Le GRAFCET étant bien adapté à la description d'un automatisme logique, il apparaît naturel de l'utiliser comme langage de programmation pour la réalisation de l'automatisme.

Le passage du GRAFCET du niveau langage de spécification au niveau langage de programmation soulève les problèmes suivants :

- limites dues au support matériel,
- contraintes liées au fonctionnement de l'automatisme.

Pour le premier point, nous examinerons successivement les difficultés rencontrées pour la saisie et l'exécution du programme. Pour le second point, nous donnons une solution pour prendre en compte les modes de marches et l'aspect sécurité du processus automatisé.

II.1.1 Au niveau de la saisie.

Le Grafcet étant un outil graphique, sa saisie doit nécessairement être graphique si l'on veut conserver les aspects clarté et facilité de description du langage de spécification GRAFCET.

II.1.1.1 Saisie graphique.

La saisie graphique a été réalisée :

- en tenant compte des limites de l'écran (6 lignes de 24 caractères) ainsi que sa faible résolution (matrices de 6 x 8 points),
- en respectant les aspects graphiques de la norme NF C 03-190,
- en guidant l'opérateur lors de l'écriture de son programme et en facilitant la relecture.

La faible taille de l'écran ne permet pas de visualiser simultanément l'ossature du graphe (étapes, transitions, liaisons) et la définition des actions et des réceptivités. Ce problème est résolu en utilisant la fonction ZOOM qui permet par un effet de grossissement de donner le détail d'une action ou d'une réceptivité.

Pour respecter la norme, il faut pouvoir saisir tous les schémas de base du GRAFCET. Cependant, compte-tenu de la faible résolution de l'écran, Il sera quelquefois nécessaire d'étirer certaines liaisons verticales. De même, on évite le croisement des liaisons en généralisant l'utilisation des renvois.

L'utilisateur est assisté dans sa saisie grâce à l'existence de touches interactives. Lors de la relecture du programme, le graphisme est restitué tel que l'utilisateur l'a saisi. Cette facilité est obtenue en gérant des informations graphiques décrivant la topologie du programme. De plus, le graphe peut défiler de manière continue sur l'écran, ce qui permet sa saisie directe et sa relecture complète.

II.1.1.2 **Saisie interactive - en ligne.**

La saisie interactive en ligne a été retenue pour obtenir, dès la fin de la saisie, un programme prêt à être exécuté. Cet objectif impose :

- d'effectuer les vérifications syntaxiques chaque fois qu'un élément est introduit. Cette vérification systématique et l'utilisation des touches interactives conduisent à un programme global syntaxiquement correct,

- d'introduire directement les éléments dans la mémoire programme de l'automate.

De plus, on a décidé que la saisie des éléments serait libre : les éléments sont introduits dans un ordre quelconque, aucune précédence n'existant d'un symbole à l'autre. Ainsi, par exemple, on saisit indifféremment une étape puis ses liens, ou les liens puis l'étape.

Ceci implique :

- de stocker les éléments dans la mémoire avec ou sans leurs liens,

- de résoudre les liens au fur et à mesure de leur apparition; dans certains cas, on est amené à résoudre en une seule fois une grande quantité de liens,

- d'assurer un temps de réponse rapide en minimisant les échanges d'informations entre la console et l'automate.

A tout moment de la saisie interactive en ligne, peuvent survenir une rupture accidentelle de la liaison console - automate ou la disparition du secteur. Ces incidents peuvent entraîner des incohérences au niveau du contenu de la mémoire programme de l'automate. Pour se protéger contre ces risques, on choisit de sacrifier l'élément en cours de saisie afin de ne pas perdre les éléments précédemment introduits.

II.1.2 **Au niveau de l'exécution.**

Au niveau de l'exécution du programme, il s'agit de choisir une méthode conciliant au mieux la simplicité de mise en oeuvre et la rapidité d'exécution.

Deux manières de procéder sont envisageables :

- exécuter du code machine,
- interpréter des structures de données.

Comparons ces deux méthodes.

II.1.2.1 **Compilation ?**

Le principal avantage d'une méthode langage compilé se situe au niveau du temps d'exécution. Par contre, pour générer efficacement un code machine dont l'exécution satisfait rigoureusement les cinq règles d'évolution du GRAFCET, il est nécessaire de connaître tous les liens reliant un élément à ses voisins. Ceci impose de maintenir à bord de la console une copie complète du programme source, donc de disposer d'une taille mémoire résidente importante.

Cette solution a été écartée pour deux raisons :

- taille mémoire insuffisante au niveau de la console,
- difficulté de mise en oeuvre de l'interactivité.

II.1.2.2 **Interprétation ?**

La solution langage interprété permet de définir progressivement les structures de données et facilite ainsi la mise en oeuvre de l'interactivité. De plus, les cinq règles d'évolution du GRAFCET sont centralisées dans un interpréteur résidant dans l'automate. On évite ainsi la gestion d'un programme source complet par la console. Cependant, un inconvénient de cette méthode apparaît au niveau temps d'exécution. Mais en fait, l'exécution d'un GRAFCET est essentiellement liée aux parties actives du graphe. Or les GRAFCET même complexes présentent simultanément peu d'étapes actives et de transitions validées. En interprétant uniquement les parties actives du graphe, on peut raisonnablement espérer un temps d'exécution court.

Pour ces raisons, la solution langage interprété a été retenue.

II.1.3 **Au niveau du système automatisé.**

II.1.3.1 **Nécessité d'un traitement préliminaire.**

La prise en compte des spécifications de modes de marches et d'arrêts et leur intégration sur le GRAFCET de base, représentant la marche de production normale, peut être menée de deux façons.

Une première façon consiste à "enrichir" le GRAFCET de base. Pour cela, on rajoute des transitions à tous les endroits où une réceptivité aux évènements considérés est nécessaire. Une telle méthode conduit rapidement à l'établissement d'un GRAFCET inextricable et peu lisible. De plus, le grand nombre de transitions utilisées risque de conduire rapidement vers des limites de mise en oeuvre.

Une seconde façon sera de procéder à une découpe en tâches. On associera à chaque évènement à traiter un GRAFCET autonome, représentant le comportement attendu de la partie opérative à l'apparition de l'évènement. Une telle découpe nécessitera l'existence d'un pilote en hiérarchie, exécuté en préambule du traitement séquentiel.

Ce pilote, ou traitement PRELIMINAIRE influera sur l'évolution de chacun des GRAFCET au moyen :

- de test de situations,
- de forçage de situation.

II.1.3.2 **Nécessité d'un traitement postérieur.**

Pour ce qui concerne l'aspect "sécurité de fonctionnement", un consensus des concepteurs en automatismes s'oriente vers une centralisation de leur traitement. Ces sécurités concernent les ordres élaborés au cours du traitement séquentiel émis vers la partie opérative. Pour cela, deux manières de procéder sont également possibles :

- "enrichir" la description des actions directement associées aux étapes par une logique combinatoire conditionnant leur exécution,
- ne pas décrire ces actions au niveau des étapes, mais reporter leur description, et donc leur exécution en "conclusion" du traitement séquentiel.

Cette conclusion, ou traitement POSTERIEUR, favorisera l'expression des relations entre situation du GRAFCET, commandes et sécurités relatives à ces commandes.

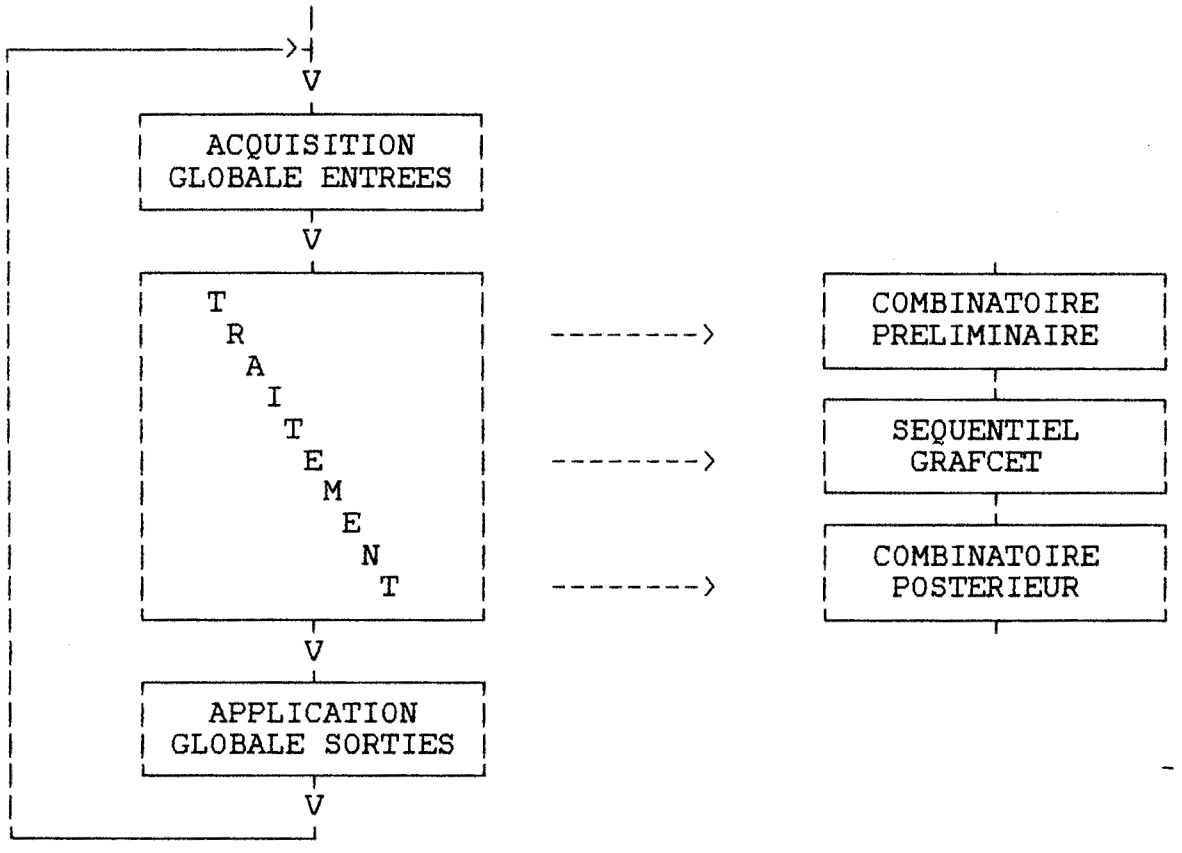
II.2 GRAFCET et cycle de traitement.

II.2.1 Description du cycle.

Selon la nature des problèmes à traiter une description de la partie commande d'un système automatisé utilisant uniquement le langage GRAFCET risque de conduire à une formulation trop complexe, et donc difficile à maîtriser, voire impossible à mettre en oeuvre.

Pour aider à la résolution de certains points délicats, on offrira la possibilité de décrire, dans un langage de type combinatoire (LADDER DIAGRAM), tout ce qui n'est pas idéalement adapté à une formulation uniquement séquentielle .

Lorsqu'un utilisateur choisit d'exprimer par ce biais le fonctionnement de son application, le cycle de traitement de l'automate se schématise de la manière suivante :



De façon générale, on pourra considérer :

- le combinatoire préliminaire comme un pré-traitement des entrées du processus automatisé les réexprimant dans un mode plus adapté à un traitement séquentiel,

- le combinatoire postérieur comme une interface entre les commandes élaborées par le traitement séquentiel et les commandes réelles du procédé.

II.2.2 **Usages du combinatoire préliminaire.**

Au vu d'une réalisation "entièrement" GRAFCET, et selon leurs implications sur la structure du traitement séquentiel, on classera les différentes opérations que l'on pourra effectuer au cours du combinatoire préliminaire en deux familles :

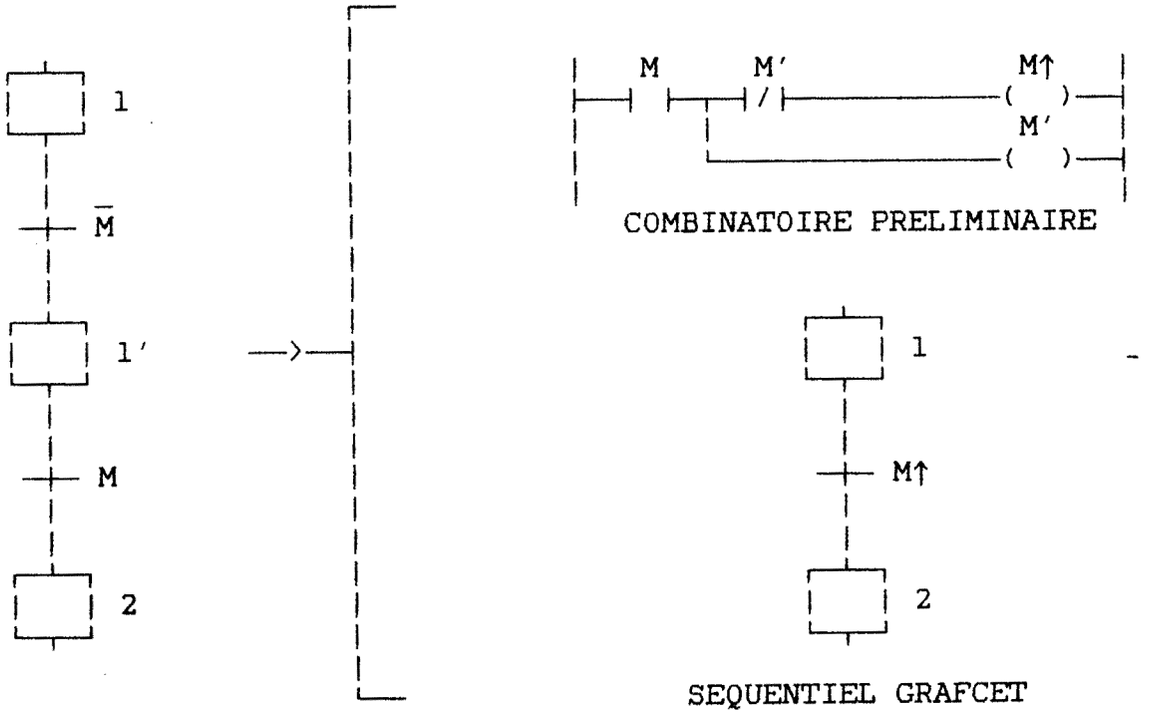
- une première famille regroupera les opérations permettant certaines simplifications tout en laissant inchangée la structure globale du GRAFCET,

- une seconde famille réunira celles qui conduiront à une structuration différente (éclatement d'un GRAFCET unique et complexe en plusieurs GRAFCET plus petits et de moindre complexité).

Parmi les opérations appartenant à la première de ces familles on citera, de façon non limitative :

- l'évaluation au préalable de sous-expressions communes à plusieurs réceptivités ou des expressions conditionnant l'exécution des actions associées aux étapes lorsqu'une même condition se retrouve plusieurs fois, apportant ainsi une meilleure compacité du programme d'application,

- la détection du front montant ou descendant d'une variable conduisant, comme l'illustre la figure ci-dessous, à une utilisation de ressources (en termes de nombre d'étapes et de transitions) optimum.

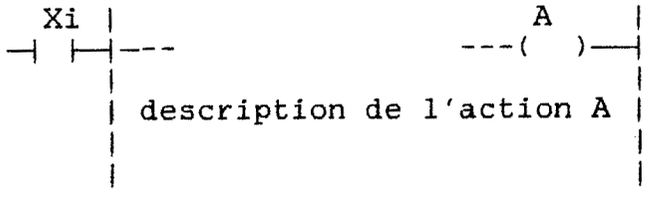


Le traitement d'évènements exceptionnels (reprise secteur, arrêt d'urgence, ...) par complément du GRAFCET décrivant la marche normale de l'application conduit rapidement à une trop forte complexité du GRAFCET résultat. Une structuration en tâches, une tâche prenant en compte un évènement, sera alors nécessaire. Une tâche étant traduite par un GRAFCET automome, l'ensemble des opérations qui permettront l'activation et la désactivation de chacune d'elles constituera la seconde famille d'opérations et déterminera le rôle majeur du combinatoire préliminaire : celui de l'introduction des modes de marche de l'application.

II.2.3 **Usages du combinatoire postérieur.**

Un premier rôle du combinatoire postérieur consistera à permettre une centralisation des aspects "sécurité de fonctionnement". Plutôt que de surcharger l'expression des actions associées aux étapes par des conditions prenant en compte les sécurités relatives à une commande donnée, on pourra préférer leur regroupement en un lieu unique. Outre une meilleure compacité de description, une telle centralisation amènera une formulation plus structurée des commandes : le programme GRAFCET fournit des ordres bruts, le combinatoire postérieur les affine, introduit les sécurités et résoud les contradictions éventuelles.

Avant d'indiquer quel sera un autre rôle du combinatoire postérieur, il est nécessaire d'anticiper quelque peu sur la manière dont seront exécutées les actions directement associées aux étapes lors de l'interprétation du programme GRAFCET. Fonctionnellement, on aimerait pouvoir considérer une action comme un réseau LADDER dont la valeur de la barre d'alimentation gauche serait déterminée par l'état d'activité Xi de l'étape à laquelle cette action est associée.



Malheureusement, le fait même qu'une action ne soit exécutée que lorsque l'étape à laquelle elle est associée est active fera que :

- pendant le temps d'activité de l'étape, l'ordre décrit par l'action sera bien effectué,
- mais l'effet de cet ordre restera maintenu lors du passage de l'étape de l'état actif à l'état inactif puisque le réseau LADDER décrivant l'action n'est alors plus exécuté.

Pour cette raison, il ne sera possible de décrire au niveau des étapes GRAFCET que les actions à effet maintenu (mise à 1 ou mise à 0 d'une variable), les actions dites à niveau étant obligatoirement décrites dans le combinatoire postérieur par reprise des variables Xi.

Notons également comme avantage de cette reprise des variables Xi celui de permettre l'unicité de commande pour une action lorsque cette action est associée à plusieurs étapes.

II.3 **Marches d'exploitation : principe.**

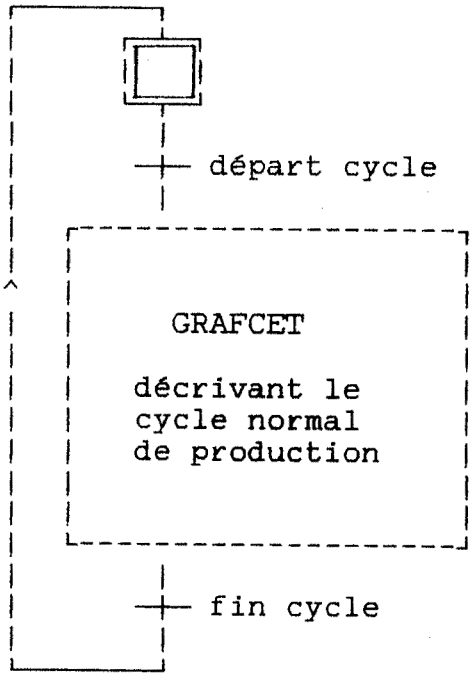
Ce paragraphe décrit quelles sont les fonctionnalités offertes à l'utilisateur pour l'aider à exprimer de manière structurée les différentes marches d'exploitation de son application (marche normale, marches d'arrêt et de mise en route, marches de défaillance, ..). Il précise également, au travers d'un exemple, comment ces différentes marches peuvent être mises en oeuvre par l'intermédiaire du combinatoire préliminaire.

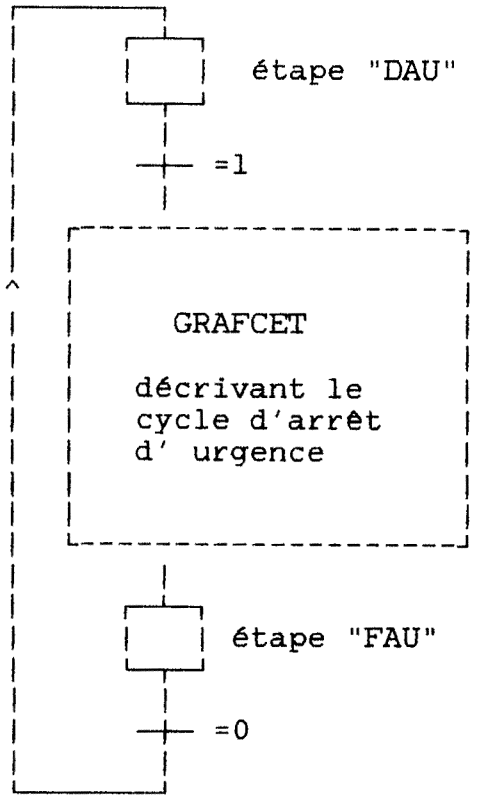
L'outil proposé est un ensemble de trois primitives permettant d'intervenir sur le déroulement de l'algorithme d'interprétation du GRAFCET :

- GRAFCET_INIT qui aura pour effet de forcer le GRAFCET en situation initiale,
- GRAFCET_RAZ_GENERALE forçant le GRAFCET dans la situation "vide",
- GRAFCET_REPRISE le plaçant dans une situation donnée.

L'activation de ces primitives se traduira par la mise à 1 d'une variable "système" associée à chaque primitive. La mise à 0 de cette variable sera prise en charge par l'interprèteur GRAFCET et signifiera que ce dernier a accompli le travail demandé . Une situation donnée, autre que les situations initiale et vide, se communiquera par l'intermédiaire de mise à 1 et de mise à 0 des variables Xi.

Pour illustrer une mise en oeuvre possible de ces mécanismes, considérons le problème de la prise en compte d'un ordre d'arrêt d'urgence. Une structuration en tâches conduira naturellement à décrire séparément, d'une part le cycle normal de production, d'autre part les opérations à effectuer lors de la détection de l'ordre d'arrêt d'urgence.



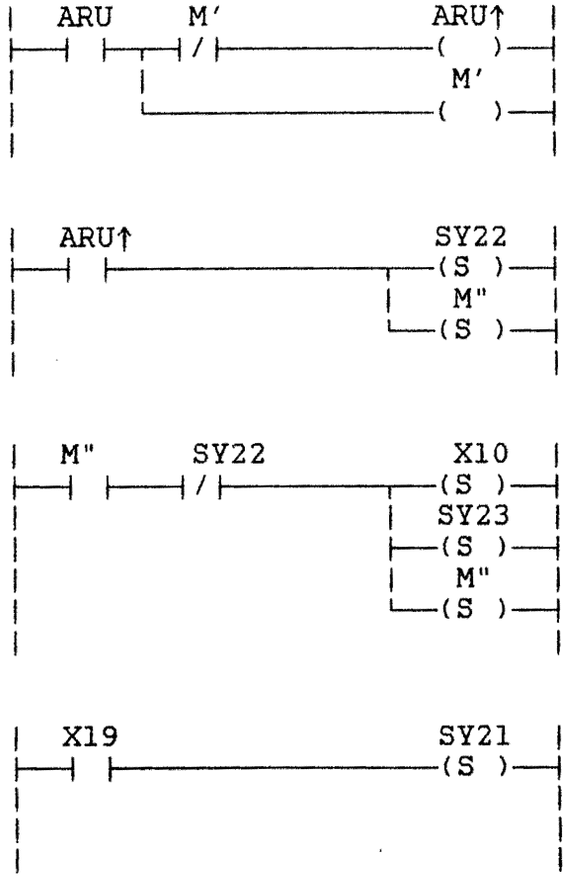


A l'apparition de l'ordre d'arrêt d'urgence, il suffira alors de commander la succession suivante d'opérations au moyen du combinatoire préliminaire :

- désactiver toutes les étapes du GRAFCET assurant la marche normale de production,
- activer l'étape "DAU" (Début Arrêt d'Urgence),
- attendre l'activation de l'étape "FAU" (Fin Arrêt d'urgence),
- remettre en place la situation initiale afin de permettre de commencer un nouveau cycle de production,

dont la programmation en langage LADDER, compte tenu des éléments suivants, est indiquée par la figure ci-après :

. ARU représente l'évènement "arrêt d'urgence",
. SY21 active GRAFCET_INIT,
. SY22 active GRAFCET_RAZ_GENERALE,
. SY23 active GRAFCET_REPRISE,
. —(S)— symbolise une mise à 1,
. M' et M" désignent des variables intermédiaires.



II.4 **Mise au point du GRAFCET : principe.**

Une marche spécifique sera utilisée lorsque les raisons d'un mauvais fonctionnement du programme GRAFCET devront être déterminées. Bien souvent, le caractère fugitif des effets constatés empêchera une mise en évidence du problème par la simple observation du phénomène. Il sera alors nécessaire de mener une exécution pas à pas (étape par étape) du programme GRAFCET.

L'introduction d'une marche de mise au point consistera en une succession d'interventions sur l'évolution "naturelle" du GRAFCET. L'utilisateur pourra :

- poser un point d'arrêt sur une étape,
- déplacer ce point d'arrêt,
- supprimer le point d'arrêt.

La prise en compte d'un tel point d'arrêt n'aboutira pas à stopper complètement l'évolution GRAFCET, mais à l'empêcher localement. Plutôt que de parler d'arrêt, on préfèrera alors parler de "blocage", blocage qu'on peut schématiser ainsi :

Lorsque l'évolution du GRAFCET conduira à activer une étape sur laquelle un point d'arrêt est posé, cette étape sera effectivement activée, les actions associées exécutées, mais les évolutions possibles à partir de cette étape seront provisoirement suspendues.

Par ce biais, il sera ainsi possible :

- soit de détecter l'activation fugitive d'une étape,
- soit de permettre l'observation du comportement d'une partie de la commande tout en laissant évoluer le reste normalement (cas de plusieurs GRAFCET indépendants).

II.5 **Résumé des caractéristiques.**

II.5.1 **Caractéristiques générales.**

Le logiciel réalisé tentera de répondre aux multiples "souhaits" des utilisateurs du GRAFCET en proposant,

globalement :

- une découpe MODULAIRE des traitements afin d'utiliser au mieux la complémentarité des langages proposés,

- une exécution cyclique des traitements enchaînant :

  . un traitement PRELIMINAIRE, décrit en langage à contacts, destiné à la réalisation des marches d'exploitation et à la prise en compte d'évènements exceptionnels,

  . un traitement SEQUENTIEL, décrit en langage GRAFCET,

  . un traitement POSTERIEUR, décrit en langage à contacts, permettant d'assurer la centralisation des sécurités relatives aux ordres émis vers la partie opérative,

au niveau de la console de programmation :

- une saisie GRAPHIQUE, afin d'offrir un langage GRAFCET sous une forme la moins éloignée possible de celle que les automaticiens manipulent quotidiennement,

- une saisie INTERACTIVE, supprimant toute notion de phases de compilation et d'édition de liens explicites,

- une saisie EN LIGNE, de manière à permettre une prise en compte immédiate des modifications de programme et faciliter certaines interventions rapides - éventuellement effectuées sur site,

- une saisie conduite de façon à garantir un maximum de SECURITE au niveau de la cohérence des programmes générés face aux perturbations de natures variées engendrées par les milieux industriels ou plus simplement face aux manoeuvres malheureuses de l'utilisateur lui-même,

- un service COMPLET en termes d'aides à la mise au point (animations en temps réel, marches pas à pas, ...) et d'édition automatique du dossier de programmation,

puis au niveau de l'automate programmable :

- une interprétation algorithmique respectant rigoureusement les cinq règles d'évolution préconisées par la norme NF C 03-190.

II.5.2 **Caractéristiques et limites statiques.**

Les GRAFCET décrits pourront comporter un seul graphe ou plusieurs graphes indépendants et devront respecter les limitations suivantes :

- comporter un nombre maximum de 96 étapes dont au plus 16 initiales,
- comporter un nombre maximum de 128 transitions,
- chaque étape sera liée à au plus 4 transitions en aval et au plus 4 transitions en amont,
- chaque transition sera liée à au plus 4 étapes en aval et au plus 4 étapes en amont,
- utiliser un nombre maximum de 84 renvois (42 paires).

Le contenu des actions et des équations de réceptivité sera décrit en langage à contacts.

II.5.3 **Caractéristiques et limites dynamiques.**

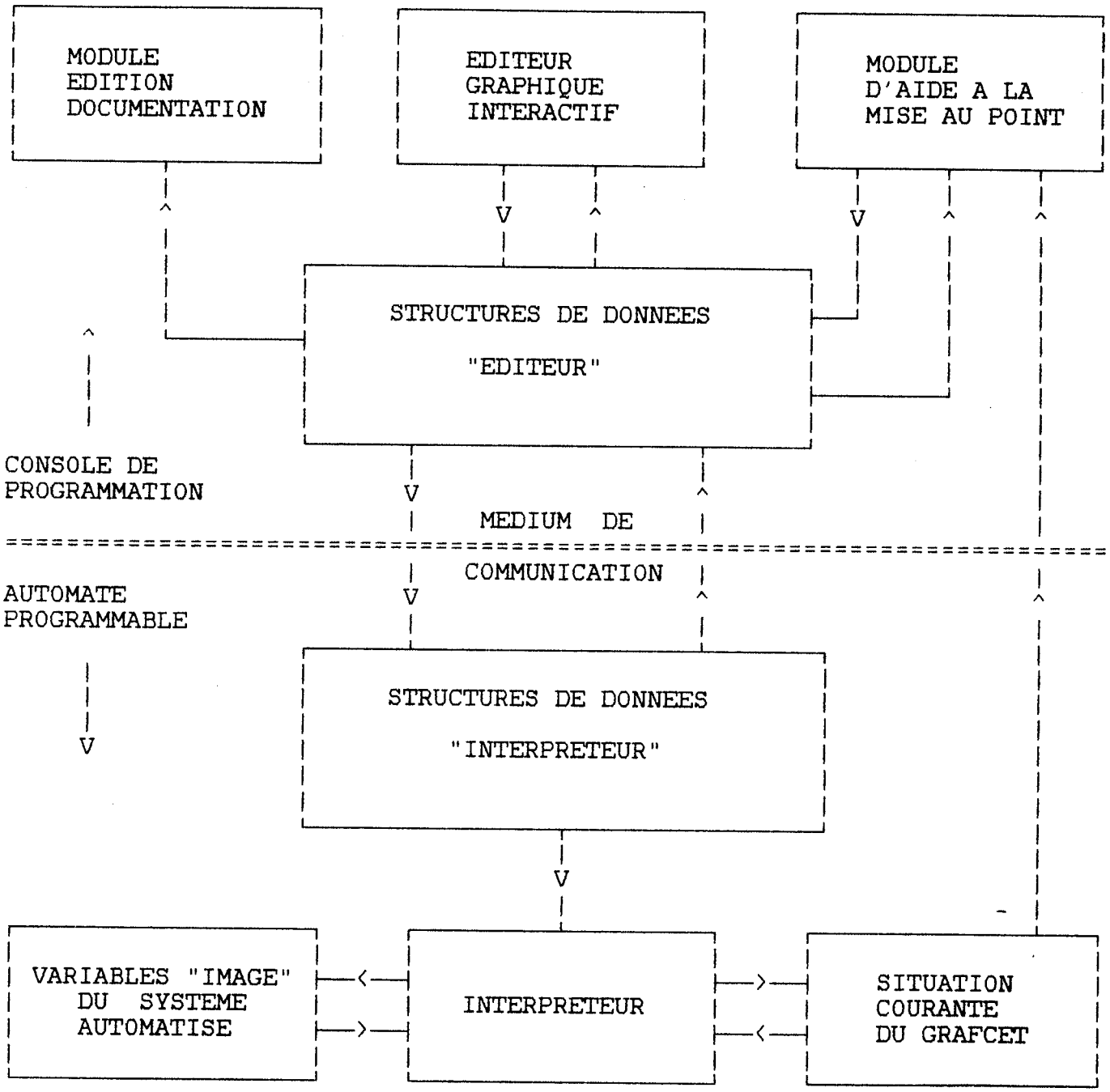
Les capacités de traitement de l'interprèteur ont été dimensionnées de façon à assurer au cours d'un cycle d'évolution :

- un ensemble maximum de 16 étapes actives simultanément,
- un ensemble maximum de 24 transitions validées simultanément.

T R O I S I E M E P A R T I E .

III **LE LOGICIEL REALISE.**

La figure ci-dessous schématise l'architecture globale du logiciel GRAFCET et précise les différentes relations nécessaires entre structures de données et modules logiciels.

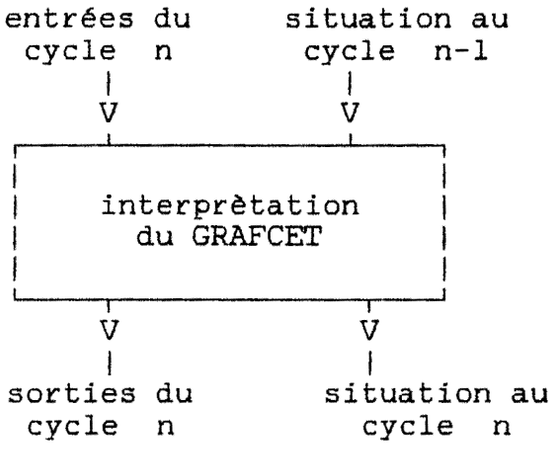


III.1 Le logiciel GRAFCET : partie automate programmable.

III.1.1 L'interprétation du GRAFCET.

III.1.1.1 Interprétation algorithmique et règles d'évolution.

L'objectif était de réaliser une interprétation algorithmique synchrone vis à vis des flux d'informations en entrée et en sortie qui respecte les 5 règles d'évolution du GRAFCET.



règle n° 1 : initialisation
règle n° 2 : franchissement d'une transition
règle n° 3 : évolution des étapes actives
règle n° 4 : évolutions simultanées
règle n° 5 : activation et désactivation simultanées

Le respect des régles n° 1, 2 et 3 ne pose pas de problème particulier. Par contre, une application rigoureuse des 2 dernières va impliquer :

- pour la régle n° 4, la nécessité d'évaluer les réceptivités - lesquelles peuvent faire intervenir l'état d'activité d'une étape - à partir d'une image figée de la situation de GRAFCET,

- pour la régle n° 5, de déterminer séparément l'ensemble des étapes à activer et celui des étapes à désactiver, puis de les recombiner dans un deuxième temps pour obtenir l'image -de la nouvelle situation.

Remarquons également qu'une action peut être conditionnée par l'état d'activité d'une étape différente de celle à laquelle elle est associée, et qu'ainsi l'exécution des actions nécessite aussi une image figée de la situation du GRAFCET.

L'exécution synchrone du cycle de traitement vis à vis des entrées/sorties nous garantissant l'élimination de la majorité des aléas de fonctionnement, il reste néanmoins quelques précautions à prendre lors de l'éxecution des actions. Supposons une sortie positionnée à 1 par l'activation d'une étape i et cette même sortie positionnée à 0 par une étape j activée au cours du même cycle : de l'ordre dans lequel ces actions seront considérées dépendra la valeur de cette sortie. On offre un moyen pour résoudre cet aléa en choisissant d'exécuter les actions dans l'ordre croissant des numéros des étapes auxquelles elles sont associées.

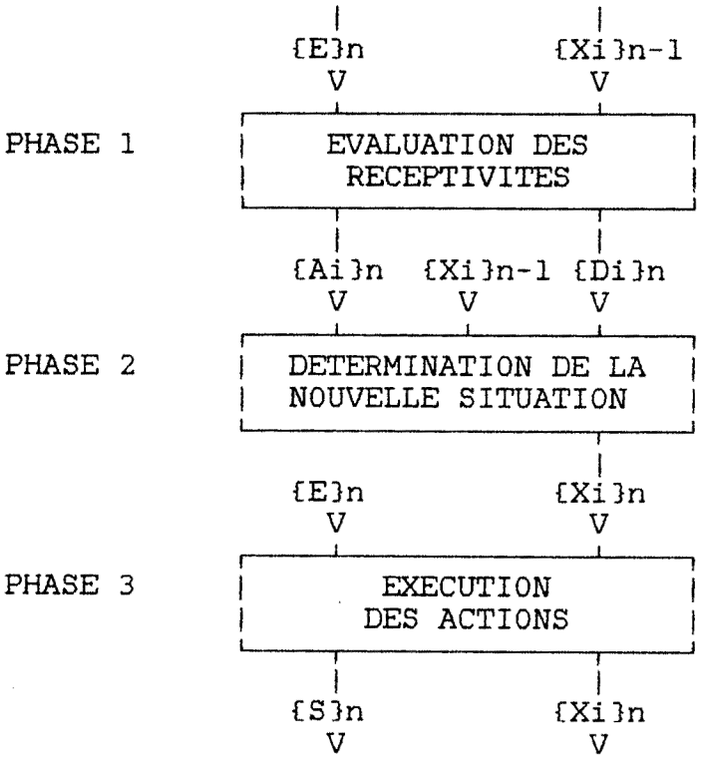
III.1.1.2 **Choix de l'algorithme d'interprétation.**

Première possibilité.

Compte tenu des notations suivantes :

- {E}n l'image des entrées au cycle n
- {S}n l'image des sorties au cycle n
- {Xi}n la situation du GRAFCET au cycle n
- {Ai}n l'ensemble des étapes activées au cycle n
- {Di}n l'ensemble des étapes désactivées au cycle n

et de l'hypothèse que le temps de cycle maximum soit suffisamment court pour qu'une situation stable soit atteinte entre deux variations des entrées - sauf cas de situation stationnaire -, l'algorithme, SANS RECHERCHE DE STABILITE, décrit ci-après satisfait à l'objectif du respect des cinq règles d'évolution.



La PHASE 1 évalue les réceptivités associées aux transitions présentement validées. En cas de franchissement d'une transition, on marque "à activer" les étapes aval à cette transition, "à désactiver" les étapes amont. En fin de phase 1 sont ainsi entièrement déterminés, par application de la règle n° 3, l'ensemble des étapes à activer et l'ensemble des étapes à désactiver au cours d'un cycle. Comme ni les entrées, ni l'état courant de la situation du GRAFCET ne sont modifiés, on satisfait rigoureusement à la règle n° 4.

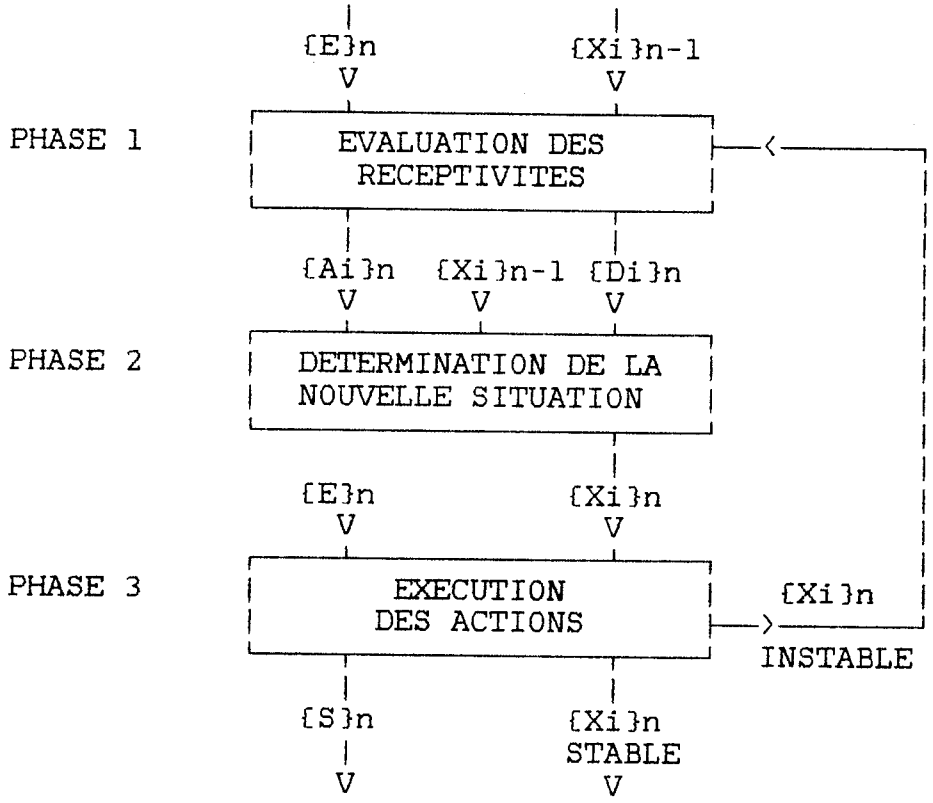
La PHASE 2 calcule la situation future en fonction des étapes actuellement actives, des étapes marquées "à activer", des étapes marquées "à désactiver",permettant ainsi une application aisée de la règle n° 5. Parallèlement, au cours de cette même phase, on procédera à la détermination du nouvel ensemble des transitions validées en accord avec la règle n° 2.

Finalement, la PHASE 3 exécute les actions associées aux étapes actives.

La PHASE 0, ou phase d'initialisation, au cours de laquelle sera appliquée la règle n° 1 est décrite ultérieurement.

Seconde possibilité.

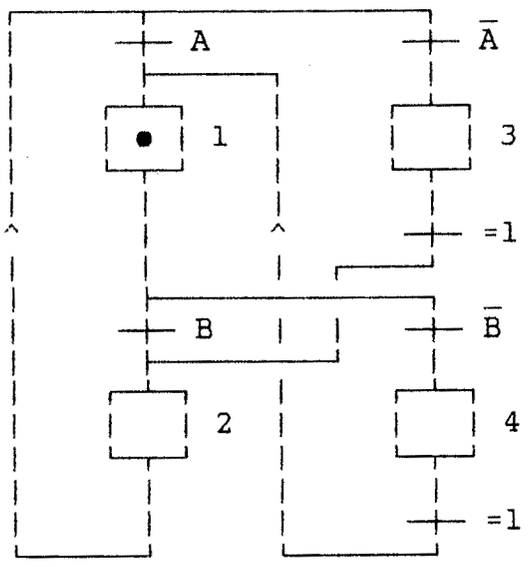
Supposons maintenant le temps de cycle quelconque de sorte que l'occurence des variations d'entrées puisse provoquer, en cas de situations instables, une situation finale non déterministe. Un algorithme différent, AVEC RECHERCHE DE STABILITE, peut alors être envisagée.



Les traitements effectués par chacune des trois phases restent inchangés, l'unique différence consiste en l'introduction d'une boucle de recherche de stabilité. La situation sera dite "stable" lorsque deux itérations successives aboutissent à une même image de situation. Cette recherche de stabilité s'effectue bien sûr à partir d'une image figée de l'état des entrées.

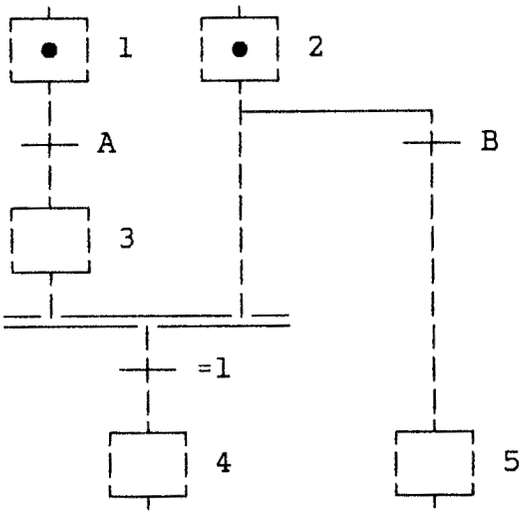
Comparaison.

Chacune de ces méthodes possède ses limites, avantages et inconvénients. Donnons-en une illustration par l'intermédiaire des exemples suivants :



Exemple n° 1.

Considérons dans un premier temps l'exemple n° 1, en supposant active l'étape 1 et les entrées A et B toutes les deux à l'état 1. Dans cette configuration, on assistera à l'établissement d'une situation stationnaire (cycle des situations instables (1)->(2)->(1)-> etc.) pouvant évoluer vers les situations (3) ou (4) sur changement d'état de l'entrée A ou de l'entrée B si l'algorithme utilisé est l'algorithme sans recherche de stabilité. Par contre, l'utilisation de l'algorithme avec recherche de stabilité conduira à l'établissement d'une "boucle" infinie, aucune variation des entrées ne pouvant être prise en compte.



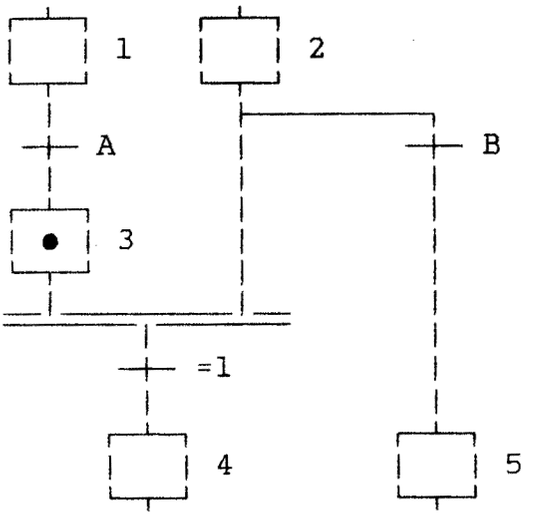
Exemple n° 2.

Dans ce second exemple, on suppose les étapes 1 et 2 actives, les entrées A et B toutes deux à l'état 0 et supposées non corrélées (une variation simultanée de A et de B est impossible). On considère les évolutions possibles lorsque l'entrée A passe de l'état 0 à l'état 1, suivie de peu par l'entrée B.

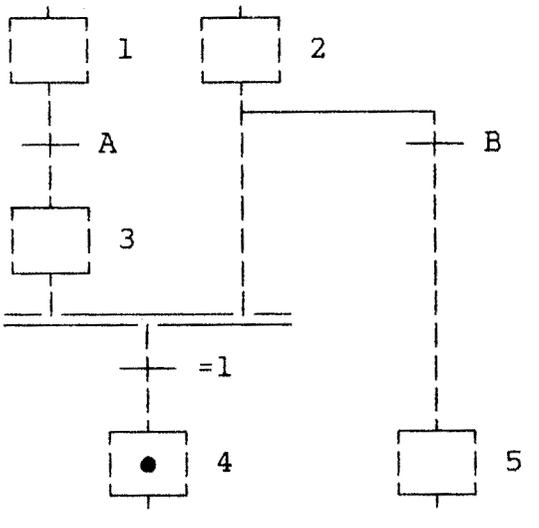
1) **Cas le l'algorithme avec recherche de stabilité.**

Premier cas de figure

A : 0 -> 1



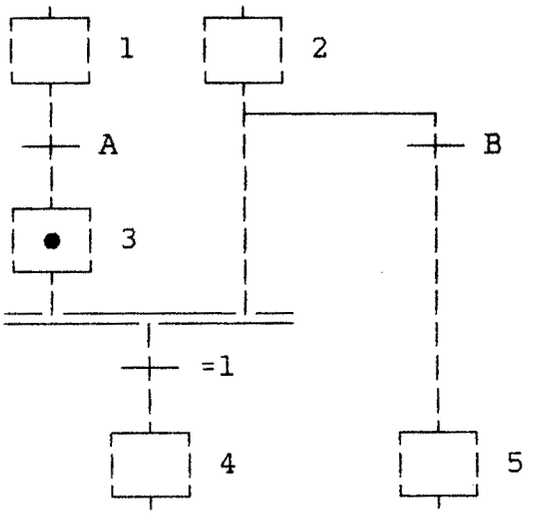
B : inchangé



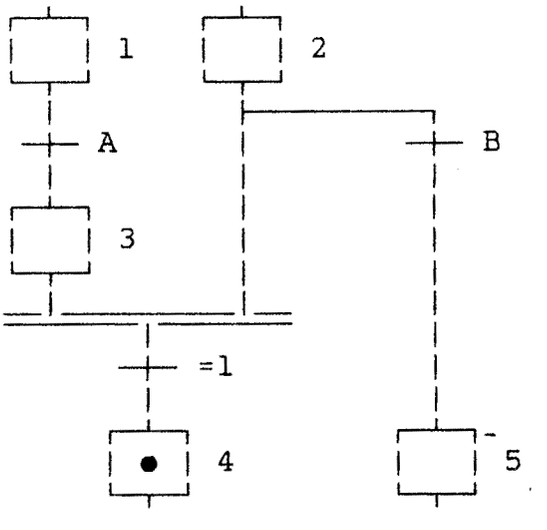
B : 0 -> 1

Second cas de figure

A : 0 -> 1



B : 0 -> 1

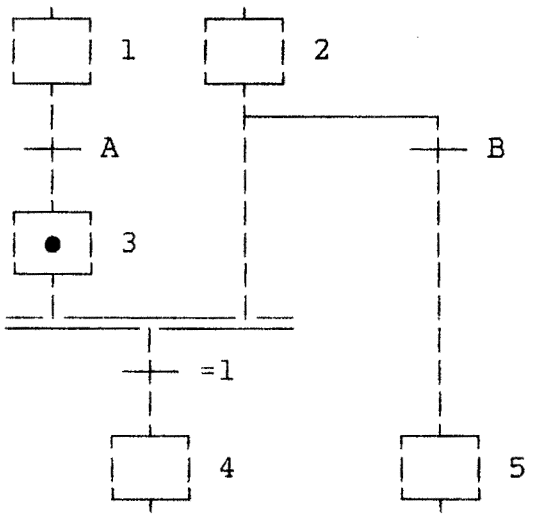


La prise en compte de la variation de l'entrée A fait évoluer le système de la situation (1,2) vers la situation (3,2). Comme cette dernière situation est à caractère instable, on atteint la situation stable (4) sans effectuer de relecture des entrées : l'évolution est déterministe quelle que soit la variation de l'entrée B.

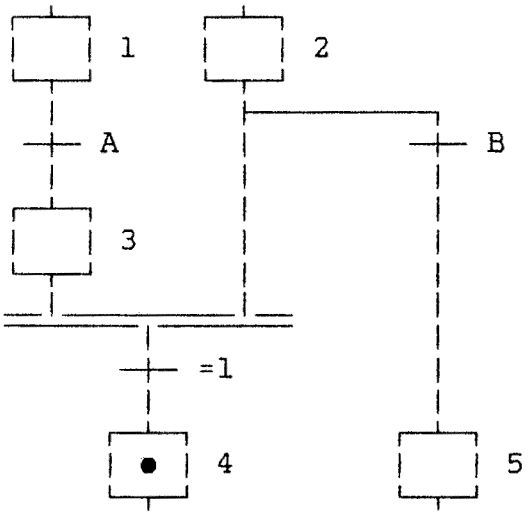
2) **Cas le l'algorithme sans recherche de stabilité.**

Premier cas de figure

A : 0 -> 1



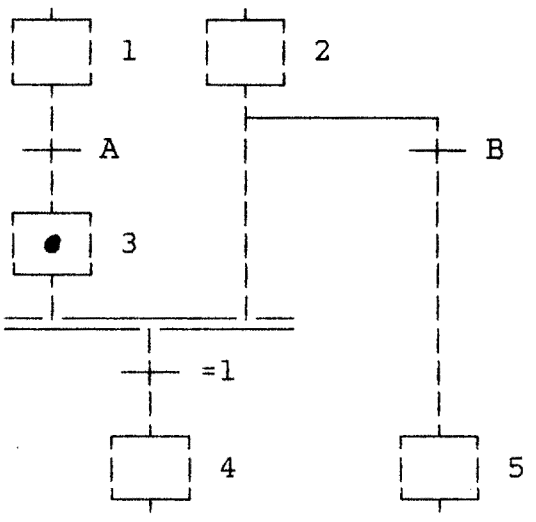
B : inchangé



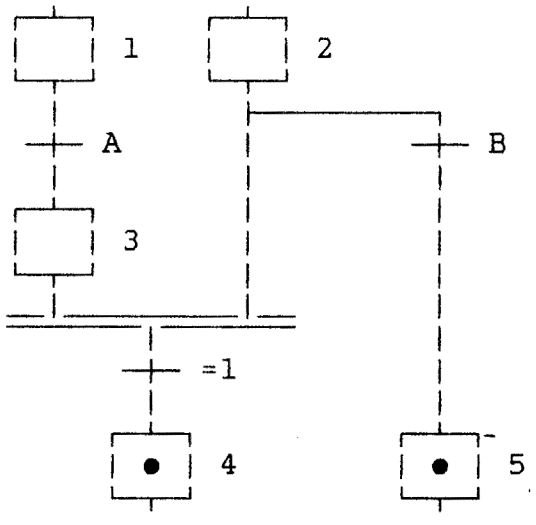
B : 0 -> 1

Second cas de figure

A : 0 -> 1



B : 0 -> 1



Comme précédemment,une variation de l'entrée A fait évoluer le système de la situation (1,2) vers la situation (3,2). Par contre, selon les instants où la variation de l'entrée B sera prise en compte, on évoluera soit vers la situation (4) soit vers la situation (4,5) : l'évolution est indéterministe lorsque le temps de cycle de l'automate est plus court que l'intervalle de temps séparant la variation de l'entrée A de celle de l'entrée B.

Conclusion et choix.

Il n'existe donc pas de solution miracle :

- l'algorithme "sans recherche de stabilité" permet la description de GRAFCET présentant des situations stationnaires, mais une variation rapide des entrées lors d'une situation instable peut conduire à une indétermination,

- l'algorithme "avec recherche de stabilité" est déterministe car insensible aux variations rapides des entrées en cas de situation instable mais réagit par une boucle infinie face à un GRAFCET possédant des situations stationnaires.

Le premier de ces algorithmes posséde néanmoins un léger avantage : celui de permettre le calcul de la borne maximum du temps d'un cycle d'interprétation pour un GRAFCET donné qui se détermine dans ce cas uniquement par le nombre d'étapes actives et de transitions validées à un même "instant".

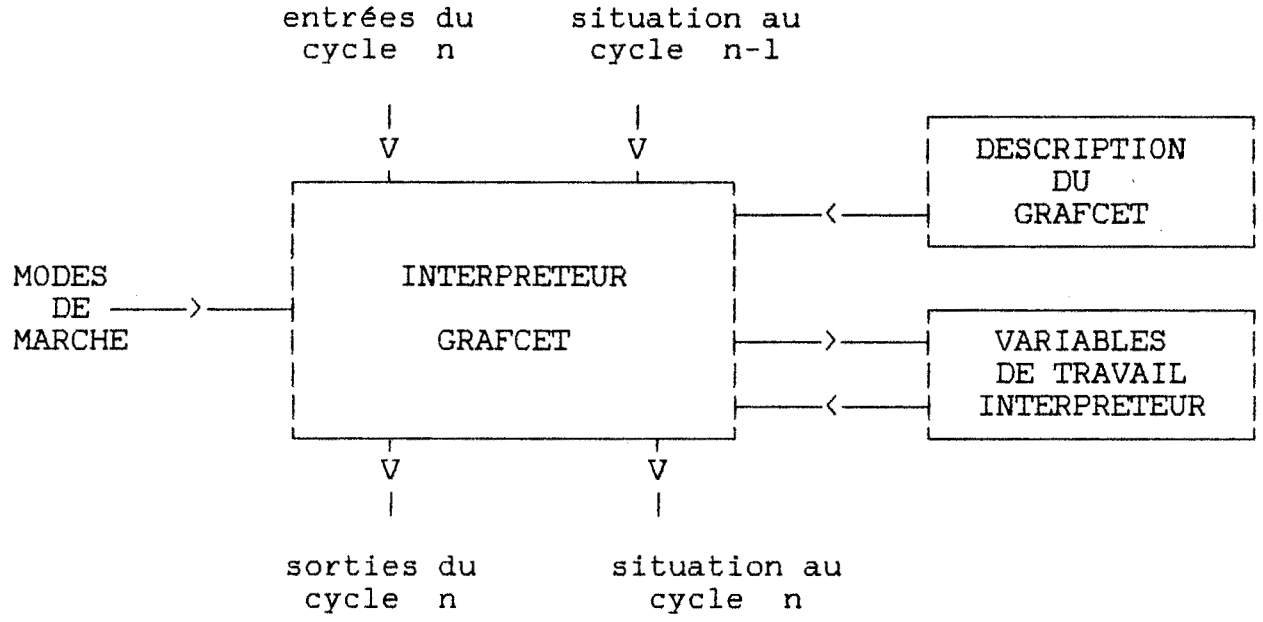
Il est, de plus, difficile d'écarter toute faculté de description de GRAFCET avec situation stationnaire sans interdire l'expression de schémas d'attente d'évènements tels que celui de l'exemple n° 1 et de perdre ainsi une certaine clarté de formulation. Par contre, il semble plus envisageable de prendre le risque d'indéterminismes possibles qu'une formulation attentive et consciente du probléme permettra la plupart du temps d'éviter.

L'algorithme choisi sera donc celui "SANS RECHERCHE DE STABILITE".

III.1.1.3 Structures de données, modes de marches et interprèteur.

L'objectif de respecter les cinq règles d'évolution du GRAFCET nous a déjà conduit à mettre en évidence une interprétation en trois phases.

Il nous reste maintenant à introduire dans l'algorithme les aspects "modes de marches" (initialisation, forçage de situation, marche étape/étape) et de dégager, au vu de l'éternel compromis compacité/vitesse d'exécution, la meilleure organisation possible des données traitées (description du GRAFCET, variables de travail) par l'interprèteur.



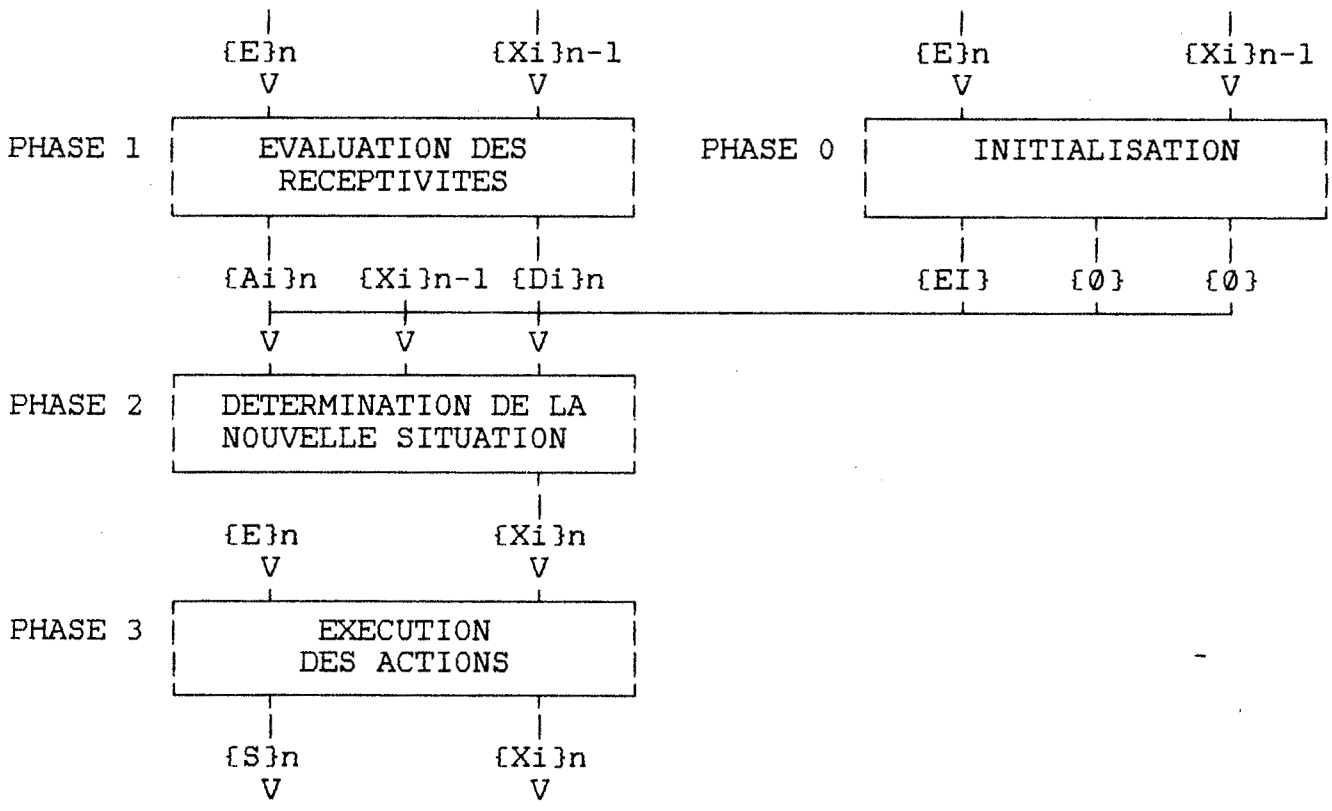
L'initialisation.

Mettre un GRAFCET en position initiale c'est :

- fournir une situation où les seules étapes actives sont les étapes initiales (règle n° 1),
- exécuter une première fois les actions associées à ces dernières.

Une façon commode de procéder est de substituer à l'enchaînement PHASE 1 - PHASE 2 - PHASE 3 (régime permanent) un enchaînement PHASE 0 - PHASE 2 - PHASE 3 (régime transitoire) lors des cycles où une initialisation du GRAFCET est demandée. Il suffira, pour que cela fonctionne correctement, de fournir en fin d'exécution de la PHASE 0 :

- un ensemble des étapes "à activer" égal à la situation initiale,
- une situation au cycle n-1 égale à l'ensemble vide,
- un ensemble des étapes "à désactiver" également vide.



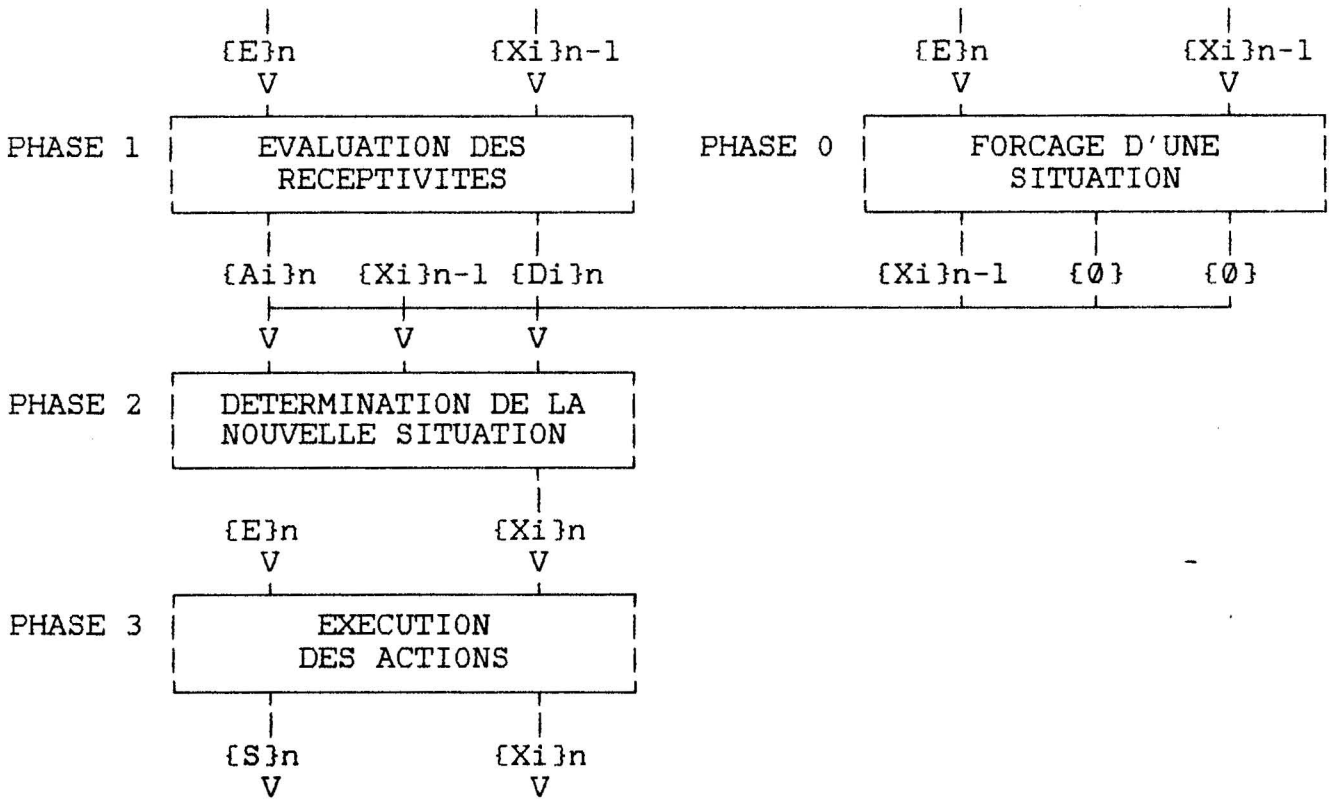
Le forçage de situation.

Le forçage d'un GRAFCET dans une situation donnée peut se traduire en termes très voisins de ceux de l'initialisation. Mettre un GRAFCET dans une situation donnée c'est :

- fournir une situation où les seules étapes actives sont les étapes que l'on désire voir actives,
- exécuter une première fois les actions associées à ces dernières.

On procédera alors de manière analogue en substituant à l'enchaînement PHASE 1 - PHASE 2 - PHASE 3 (régime permanent) un enchaînement PHASE 0 - PHASE 2 - PHASE 3 (régime transitoire) lors des cycles où un forçage de situation du GRAFCET est demandé. Il suffira, pour que cela fonctionne correctement, de fournir en fin d'exécution de la PHASE 0 :

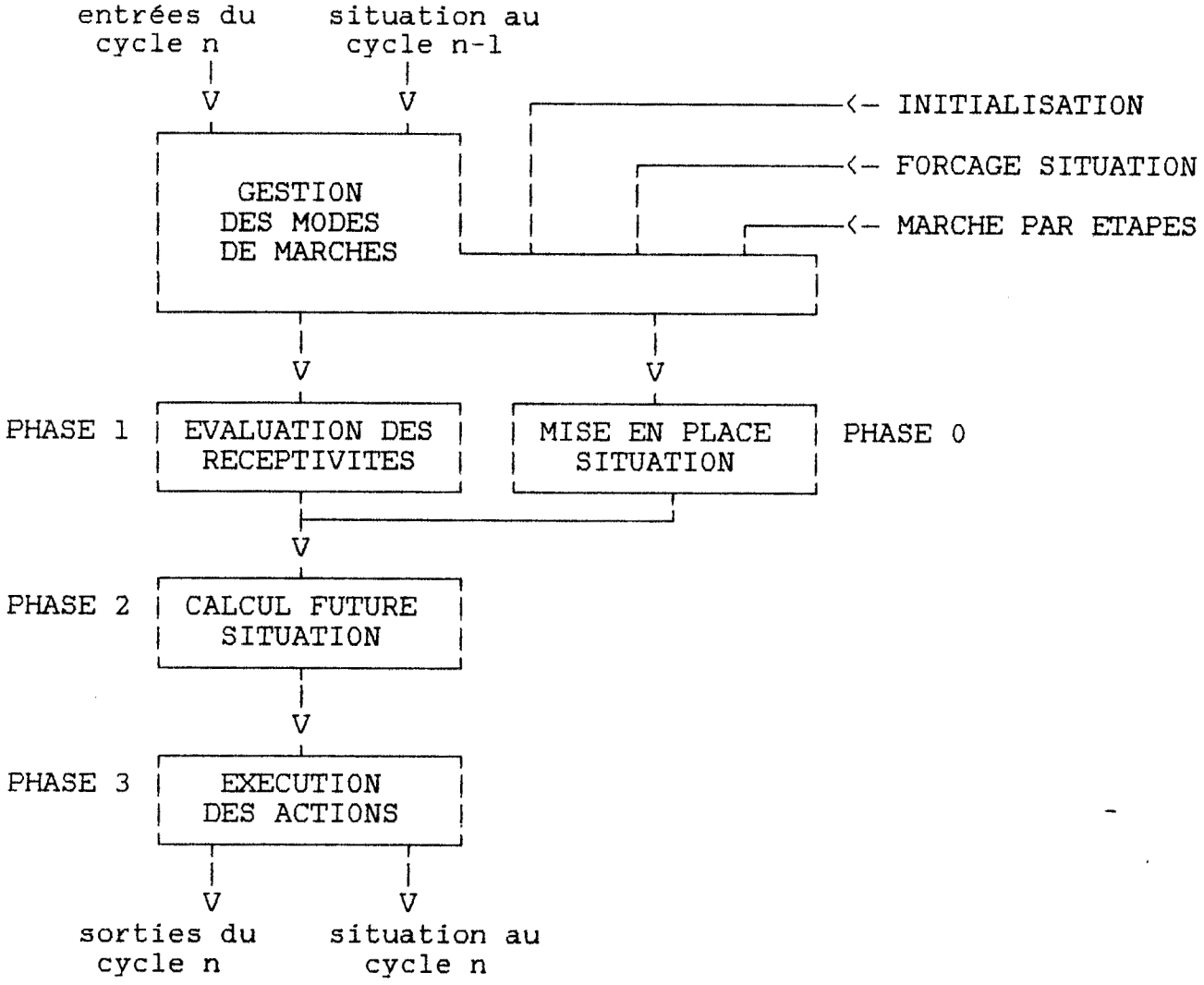
- un ensemble des étapes "à activer" égal à la situation désirée que l'on communiquera à l'interprèteur par l'intermédiaire de {Xi}n-1,
- une situation au cycle n-1 égale à l'ensemble vide,
- un ensemble des étapes "à désactiver" également vide.



Marche étape par étape.

La marche étape par étape - suite de blocages et de déblocages d'étapes - est réalisée par un mécanisme d'inhibition des réceptivités associées aux transitions . On bloquera une étape en inhibant les réceptivités associées à ses transitions aval, on débloquera l'étape en les libérant. Ce mécanisme d'inhibition se traduira par un ET logique du complément d'un indicateur Hj (inHibition transition j) avec le résultat de l'évaluation de la réceptivité associée à la transition j. L'ensemble des réceptivités inhibées au cours d'un cycle sera élaboré par le module "GESTION des MODES de MARCHES" et pris en compte lors de l'exécution de la phase 1 : "EVALUATION des RECEPTIVITES".

La formulation générale de l'algorithme sera finalement :



Détermination des structures de données.

Reconsidérons à nouveau l'enchaînement PHASE 1 - PHASE 2 - PHASE 3 du régime permanent de façon à mettre en évidence, au vu des traitements effectués au cours de chacune des trois phases, quelles sont, pour une phase donnée, les informations nécessaires avant son exécution puis les informations fournies en fin d'exécution.
A partir des informations statiques nous pourront déduire une organisation possible pour la représentation du GRAFCET. De manière analogue l'examen des informations dynamiques conduira à déterminer les variables de travail.

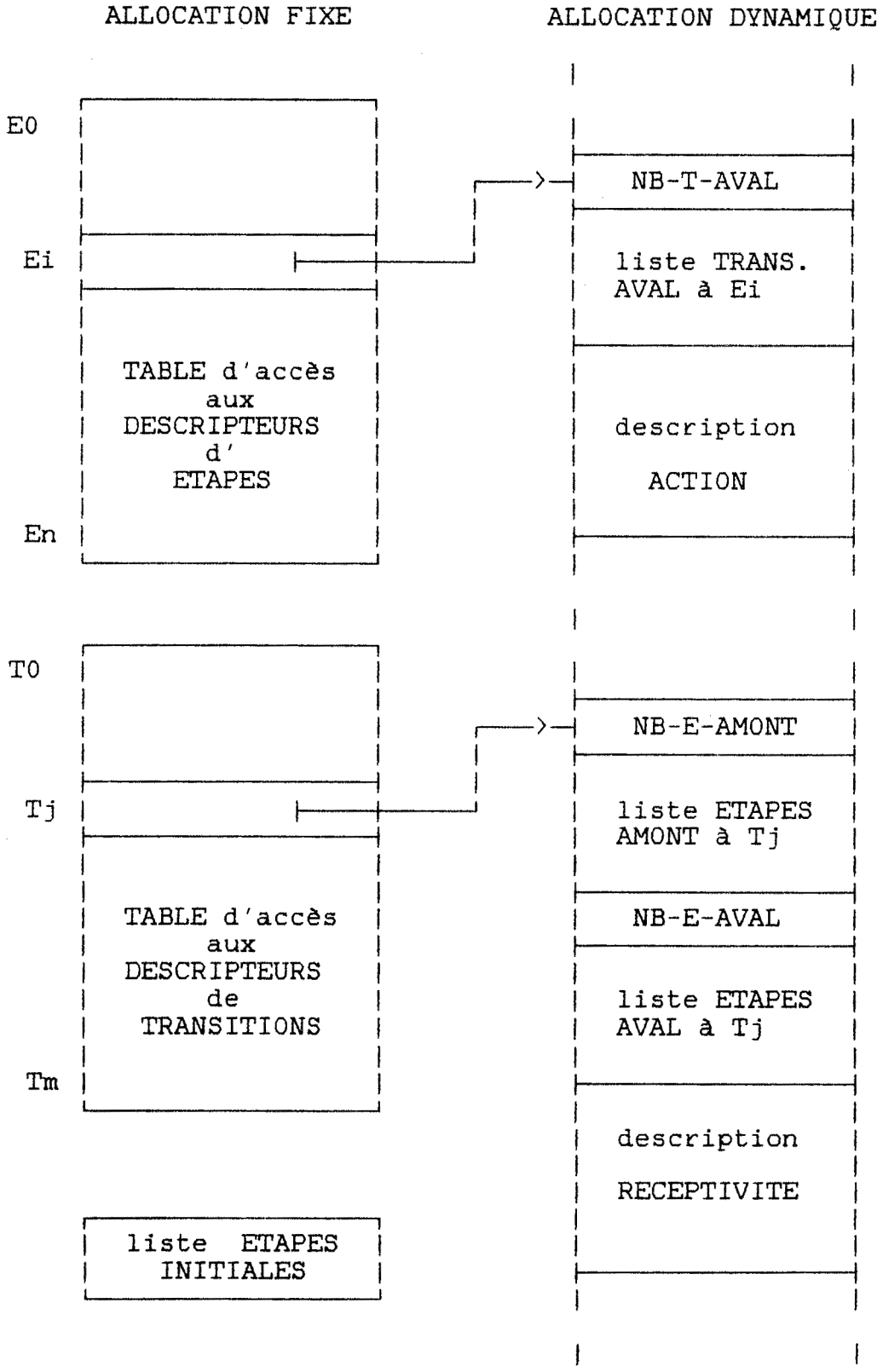
INFO. DYNAMIQUES NECESSAIRES	INFO. STATIQUES NECESSAIRES
- Tj valides cycle n-1	- liste Ei amont à Tj - liste Ei aval à Tj - réceptivités associées à Tj
PHASE 1	- évaluer réceptivités Tj valides - marquer "à désactiver" Ei amont à Tj - marquer "à activer" Ei aval à Tj
- Ei marquées cycle n	
! V	
- Ei marquées cycle n - Ei actives cycle n-1 - Tj valides cycle n-1	- liste Tj aval à Ei
PHASE 2	- déterminer Ei activées - déterminer Ei désactivées - valider Tj aval à Ei activées - invalider Tj aval à Ei désactivées
- Ei actives cycle n - Tj valides cycle n	
! V	
- Ei actives cycle n	- action associée à Ei
PHASE 3	- exécuter actions Ei actives

Pour ce qui concerne la description du GRAFCET (informations statiques), on aperçoit que :

- en phase 1, on a besoin uniquement d'informations structurelles relatives aux transitions (liens sur étapes amont et aval, réceptivités),
- en phases 2 et 3 d'informations relatives aux étapes (liens sur transitions aval, actions) que l'on peut regrouper.

1) Structures de données statiques.

On peut alors envisager une organisation de ces données statiques (structurelles) telle qu'elle est décrite ci-dessous. On place à adresse fixe des tables d'accès et on alloue dynamiquement l'espace nécessaire à la description de chacune des étapes et transitions composant le programme GRAFCET de façon à assurer la meilleure compacité possible.



1) Structures de données dynamiques.

Regroupons maintenant en deux ensembles, d'une part les données dynamiques relatives aux transitions, d'autre part celles relatives aux étapes, ensembles que l'on concrétisera sous formes de listes chainées (LISTE_ACTIVES pour les étapes, LISTE_VALIDES pour les transitions).

Décrivons alors sommairement les traitements (en termes d'opérations sur listes) des phases 1, 2 et 3 en précisant en début et fin de chacune d'elles les caractéristiques des éléments contenus dans les listes,

afin de montrer que cette organisation convient.

```
(*
LISTE_ACTIVES contient les étapes actives au cycle n-1
LISTE_VALIDES contient les transitions validées au cycle n-1
*)
```

PHASE 1 :

```
    Pour chaque (TRANSITION j dans LISTE_VALIDES) faire

        Evaluer RECEPTIVITE

        Si (PASSANTE et non INHIBEE) alors

            Considérer les ETAPES i aval à j
                Marquer "à activer" ces ETAPES i
                Placer ces ETAPES i dans LISTE_ACTIVES

            Considérer les ETAPES i amont à j
                Marquer "à désactiver" ces ETAPES i
        Fsi

    Fpour
```

```
(*
LISTE_ACTIVES contient les étapes actives au cycle n-1 éventuellement
marquées ("à activer" ou "à désactiver") et des étapes non actives au
cycle n-1 marquées "à activer"
LISTE_VALIDES contient les transitions valides au cycle n-1
*)
```

Selon l'état actuel d'activité et les marques d'une étape, on caractérise chacune d'elles de la façon suivante :

```
non active,     marquée activée, non marquée désactivée --> activée
    active, non marquée activée, non marquée désactivée --> active
    active, non marquée activée,     marquée désactivée --> désactivéee
    active,     marquée activée, non marquée désactivée --> suractivée
    active,     marquée activée,     marquée désactivée --> réactivée
```

```
(*
LISTE_ACTIVES contient les étapes actives au cycle n-1 éventuellement
marquées ("à activer" ou "à désactiver") et des étapes non actives au
cycle n-1 marquées "à activer".
LISTE_VALIDES contient les transitions valides au cycle n-1
*)
```

PHASE 2 :

```
    Pour chaque (ETAPE i dans LISTE_ACTIVES) faire

        Cas (ETAPE i) dans

            (activée) :

                Considérer parmi les TRANSITIONS j aval à ETAPE i
                celles qui parviennent à l'état validé
                    Placer ces TRANSITIONS j dans LISTE_VALIDES

            (désactivée) :

                Considérer les TRANSITIONS j aval à ETAPE i
                    Supprimer ces TRANSITIONS j dans LISTE_VALIDES

                Supprimer ETAPE i dans LISTE_ACTIVES

            (réactivée) :

            (suractivée) :

            (active) :

        Fcas

        Effacer les marques

    Fpour

(*
LISTE_ACTIVES contient les étapes actives au cycle n
LISTE_VALIDES contient les transitions valides au cycle n
*)
```

PHASE 3 :

```
    Trier LISTE_ACTIVES en ordre croissant

    Pour chaque (ETAPE i dans LISTE_ACTIVES) faire

        Exécuter ACTION

    Fpour

(*
LISTE_ACTIVES contient les étapes actives au cycle n
LISTE_VALIDES contient les transitions valides au cycle n
*)
```

III.1.2 **Les fonctions annexes.**

Le présent chapitre décrit quelques aspects des fonctionnalités qu'il a fallu definir pour créer, au niveau de l'automate programmable, l'environnement logiciel nécessaire à la mise en oeuvre des programmes d'application. Le but n'est pas de donner une liste exhaustive de ces fonctionnalités mais :

- d'une part, d'insister sur l'aspect sécurité des programmes générés en présentant, au travers des procédures de gestion de la mémoire-programme, le mécanisme permettant d'assurer cette sécurité,

- et d'autre part d'illustrer comment ce même mécanisme a été facilement utilisé pour réaliser la photographie d'un ensemble de variables.

III.1.2.1 **Gestion de la mémoire programme.**

La volonté d'offrir une saisie interactive en ligne des programmes d'applications a conduit à imaginer :

- une organisation de la mémoire-programme qui permette l'introduction des différents éléments constitutifs du programme dans un ordre quelconque,

- un mécanisme qui, allié aux procédures de gestion de cette mémoire , assure que son contenu soit toujours cohérent quels que soient les incidents survenant au cours de la saisie. Les principaux incidents sont la disparition du secteur et la rupture accidentelle de la liaison console - automate.

Principe.

L'espace mémoire destiné à contenir le programme utilisateur est exploité par blocs indépendants, de tailles variables et chaînés entre eux. Chaque bloc concrétise ce qui est présenté au programmeur comme étant une entité élémentaire et indissociable (un réseau de contacts en langage LADDER, une étape ou une transition en langage GRAFCET).

Toute intervention sur le contenu du programme se traduira, au vu de cette organisation, en termes de primitives :

- d'insertion de bloc, matérialisant la création d'unē telle entité,
- de libération de bloc (tranformation en bloc de type libre), pour sa suppression,

ou d'enchaînement de ces deux primitives en cas de remplacement d'un bloc par un autre bloc (modification d'une entité existante).

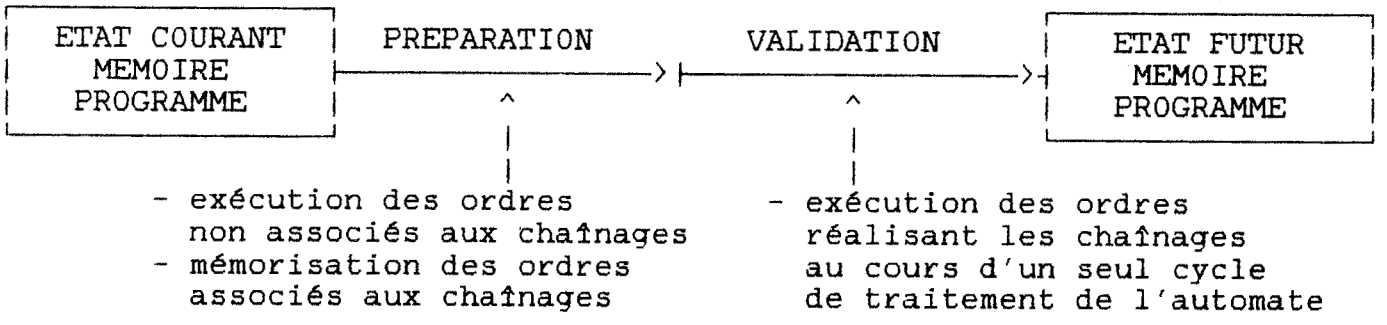
Mécanisme nécessaire.

Chacune de ces primitives sera concrétisée par une succession d'ordres d'écriture dans la mémoire programme élaborés par la console et executés par l'automate. Parmi tous ces ordres d'écriture, certains seront destinés à créer les contenus de blocs, d'autres à réaliser les chaînages entre les blocs. Ces ordres seront communiqués à l'automate au moyen de messages : le nombre d'ordres transporté par un même message sera forcément limité. Afin d'assurer un état cohérent entre deux modifications de programme, il sera nécessaire d'exécuter d'un seul tenant les ordres associés aux chaînages. Un message perdu peut entraîner la rupture d'un chainage et provoquer la perte irrémédiable du programme.

Il faudra procéder en deux temps :

- un premier temps, où on exécutera les ordres d'écriture non relatifs aux chaînages et où on mémorisera, sans les exécuter, ceux qui seront destinés à réaliser les chaînages (phase de PREPARATION),

- un second temps où on exécutera d'un seul tenant les ordres d'écriture réalisant les chaînages (phase de VALIDATION).



Si la liaison est interrompue ou si le secteur disparaît pendant la phase de PREPARATION, le programme restera dans l'état courant : on ne perdra que la modification en cours.

Si la liaison est interrompue ou si le secteur disparaît pendant la phase de VALIDATION, on sera capable de garantir l'achèvement de la modification.

Dans ce dernier cas, le rupture de la liaison sera sans -effet : l'ensemble des ordres nécessaire étant mémorisé à bord de l'automate. Le seul point délicat restant à traiter sera celui de la terminaison en cas de disparition du secteur. La réserve d'énergie disponible lors de la détection d'une coupure secteur étant seulement suffisante à la sauvegarde du contexte d'exécution de l'automate, on assurera cette terminaison à la réapparition du secteur, lors du cycle de reprise.

Mise en oeuvre de ce mécanisme.

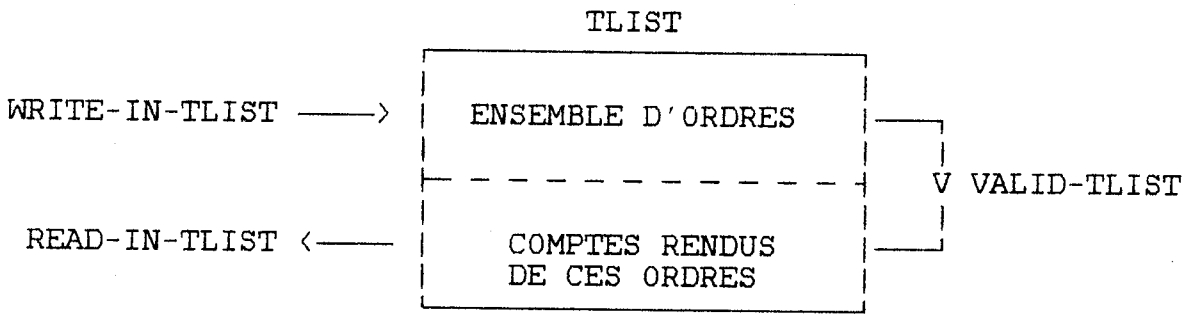
Ce mécanisme est constitué d'un espace mémoire (baptisé TLIST : Traitement-LIST) et d'un ensemble d'opérations sur cet espace mémoire.

TLIST est partagée en deux zones :

- une première zone destinée à mémoriser les ordres à exécuter,
- une seconde zone recevant les comptes rendus de ces ordres.

Les opérations sont :

- WRITE-IN-TLIST <adresse dans TLIST><ordre>,
- VALID-TLIST sans paramètre,
- READ-IN-TLIST <adresse dans TLIST>.



III.1.2.2 **Photographie d'un ensemble de variables.**

Lors des phases de mise au point des programmes d'application, la console présente des "écrans" comportant des informations montrées simultanément. Pour que ces informations reflètent un état cohérent du système automatisé, il n'est pas possible de les acquérir indépendamment.

Un mécanisme en deux temps sera donc encore une fois nécessaire :

- un premier temps où on communiquera à l'automate, qui les mémorisera, les ordres de lecture associés à chaque information,

- un second temps où ces ordres de lecture seront exécutés d'un seul tenant.

Ce mécanisme est tout à fait à celui décrit précedemment : il en recevra d'ailleurs la même mise en oeuvre. On aura donc :

- un premier temps réalisé par une succession d'opérations WRITE-IN-TLIST,

- un second temps déclanché par une opération VALID-TLIST.

Les comptes rendus de ces ordres de lecture retournent les valeurs des informations voulues avec la cohérence souhaitée, informations que l'on exploitera par un succession d'opérations READ-IN-TLIST.

III.2 Le logiciel GRAFCET : partie console de programmation.

III.2.1 L'éditeur graphique.

La fonction, ou rôle essentiel d'un éditeur GRAFCET graphique pourrait être exprimé de la façon suivante :

> "permettre la synthèse d'un programme GRAFCET, en transformant en une forme interne propre à être exécutée par un automate programmable, une forme externe la plus proche possible de celle que l'on pourrait obtenir à l'aide d'un papier et d'un crayon, sans oublier l'inévitable gomme".

On choisira :

- de restreindre l'ensemble des graphismes possibles ainsi que leur localisation mais d'autoriser leur introduction dans un ordre quelconque, éliminant ainsi toute contrainte de précédence d'un élément par rapport à un autre,

- de ne pas tout permettre à un instant donné mais uniquement ce qui est possible au regard des règles de construction du GRAFCET.

Ces choix permettent, outre une relative simplification du logiciel de l'éditeur, de conserver tout au long de la saisie, un programme GRAFCET syntaxiquement correct rendant ainsi possible la génération en parallèle de la forme interne et donc, en fin de saisie, d'un programme prêt à l'exécution.

Fonctionnellement, un programme GRAFCET se décompose en :

- un ensemble d'étapes auxquelles sont associées des actions, éventuellement vides,

- un ensemble de transitions auxquelles sont associées des réceptivités,

- un ensemble de liaisons orientées reliant entre-elles les étapes et les transitions.

Deux remarques peuvent à priori être formulées :

- pour un GRAFCET donné, l'agencement entre-elles des liaisons étape - transition est unique, mais la représentation de cet agencement peut revêtir de multiples formes,

- l'existence d'une action, respectivement d'une réceptivité, est liée à l'existence de l'étape, respectivement la transition, à laquelle elle est associée.

De ces deux remarques on obtient quelques premières idées directrices pour la conception des structures de données de représentation interne du programme GRAFCET :

- il est possible d'opérer une dissociation très nette entre "informations sémantiques", celles qui décriront les étapes, les transitions et leurs liaisons réciproques, et "informations graphiques", qui décriront elles la manière (forme, localisation) dont ces différents éléments seront dessinés,

- il est souhaitable que la représentation d'une étape et de ses actions associées forment un bloc de données unique; il en va de même pour une transition et sa réceptivité associée.

III.2.1.1 **Ergonomie du programme GRAFCET.**

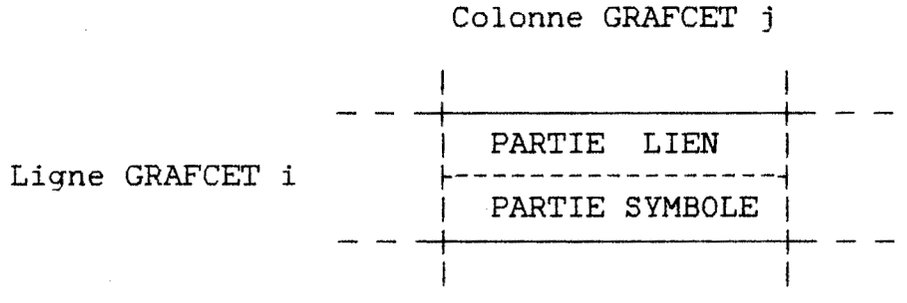
La partie graphe d'un programme GRAFCET est construit à partir d'un ensemble d'éléments graphiques que l'on classera en deux familles, une première constituée par les symboles GRAFCET (étapes, transitions, renvois), une seconde réunissant les liaisons GRAFCET (liens verticaux, liens horizontaux simples ou doubles). Ces éléments graphiques seront assemblés à l'intérieur de plans graphiques ou pages GRAFCET.

Ces pages GRAFCET peuvent être indépendantes entre-elles - chacune d'elles décrivant un GRAFCET complet -, ou être "logiquement" reliées au moyens de renvois - un même GRAFCET occupant plusieurs pages.

Chaque page GRAFCET a été dimensionnée de façon à pouvoir être imprimée au format 21 x 29,7 sur une imprimante de caractéristiques courantes (66 lignes de 80 caractéres ASCII standards),

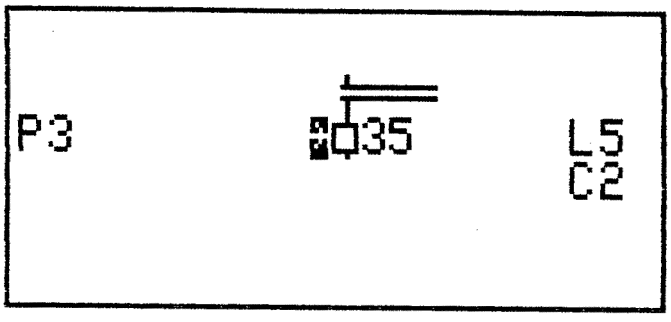
Le contenu d'une page GRAFCET sera accédée par l'intermédiaire d'une fenêtre mobile de taille compatible avec la capacité de l'afficheur à cristaux liquides (6 lignes de 24 caractères) de la console de programmation TSX T407.

Une page GRAFCET est partagée dans un sens en 14 bandes horizontales (lignes GRAFCET), dans l'autre en 8 bandes verticales (colonnes GRAFCET). Chaque intersection entre une ligne GRAFCET et une colonne GRAFCET délimite un emplacement dans une page donnée et constitue l'unité de base à laquelle on s'adressera au travers de l'éditeur GRAFCET. Chaque emplacement contiendra dans sa partie basse un symbole GRAFCET et dans sa partie haute le lien GRAFCET amont de ce dernier.

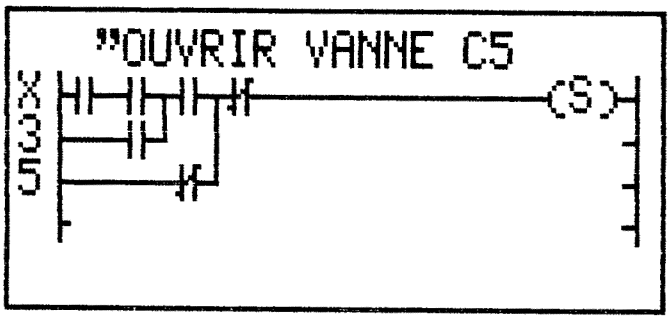


Ce principe de répartition des éléments permettra de travailler, après désignation d'un emplacement donné, indépendamment sur un symbole ou sur son lien amont de même sur l'action associée à une étape, ou sur la réceptivité associée à une transition, tout en demeurant localisé sur cet emplacement.

L'emplacement courant (i.e. celui sur lequel l'éditeur opère à un instant donné) sera repéré d'une part à l'aide d'une flèche (curseur GRAFCET) et d'autre part par l'affichage de ses coordonnées "topologiques" (N° de page, ligne et colonne GRAFCET). Une illustration en est donnée par la figure ci-dessous où l'on a uniquement représenté le contenu de l'emplacement courant : ici une étape et son lien amont (de type "distribution" ou divergence en ET) situés en page 3, ligne 5, colonne 2.



Un simple appui sur la touche ZOOM du clavier permet l'accès à l'action associée à l'étape n° 35.

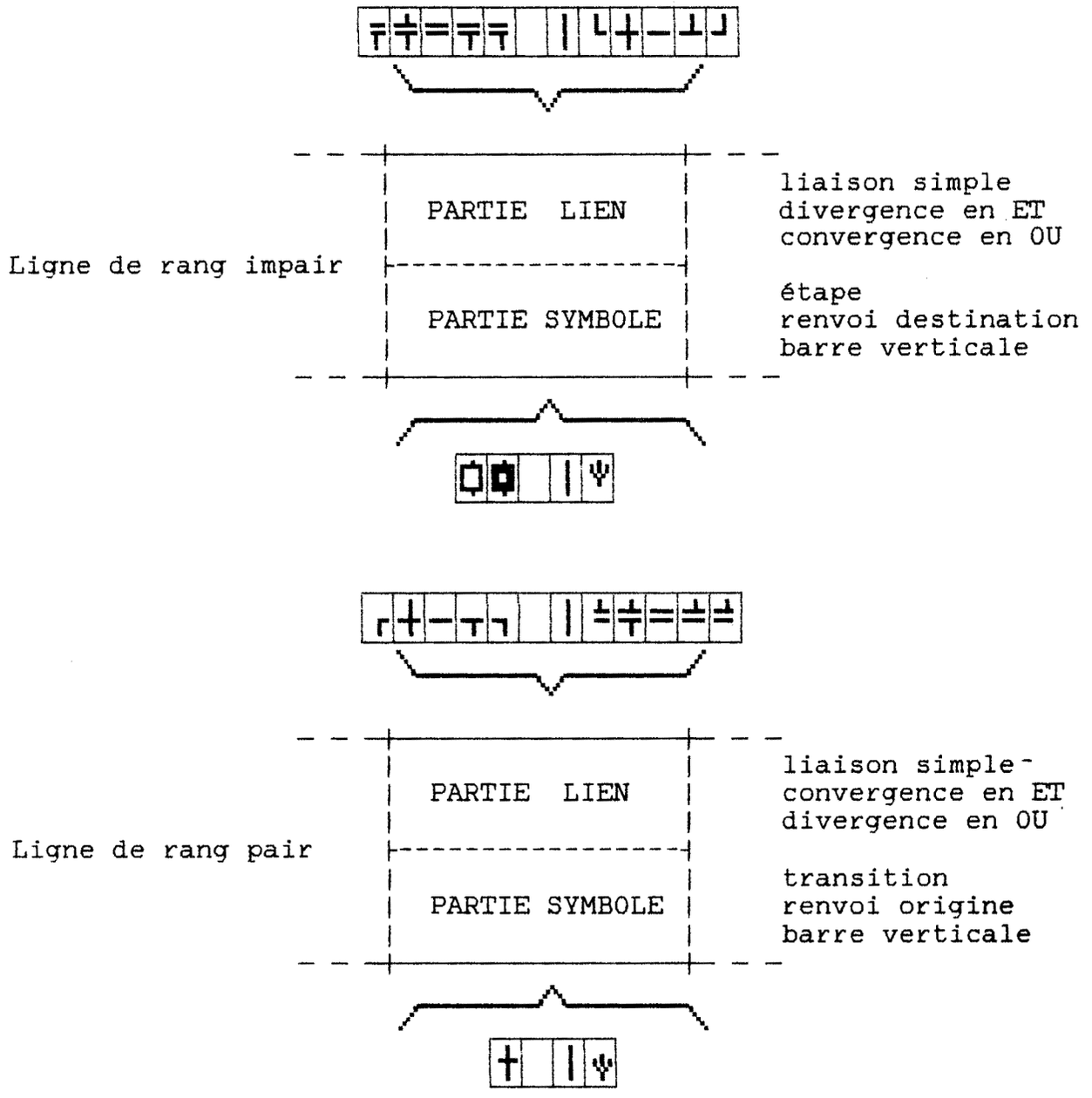


L'existence des variables Xi associées aux étapes et décrivant leur état d'activité conduit à adopter un système de numérotation centré sur les étapes, seuls éléments explicitement numérotés. Ainsi :

- une transition sera repérée par le couple n° étape amont / n° étape aval en privilégiant, en cas de pluralité de l'une ou l'autre, celle située le plus à gauche,
- un renvoi de destination portera le n° de l'étape destination,
- un renvoi de provenance référencera l'étape amont à la transition auquel il est connecté, choix du n° de celle la plus à gauche en cas de pluralité.

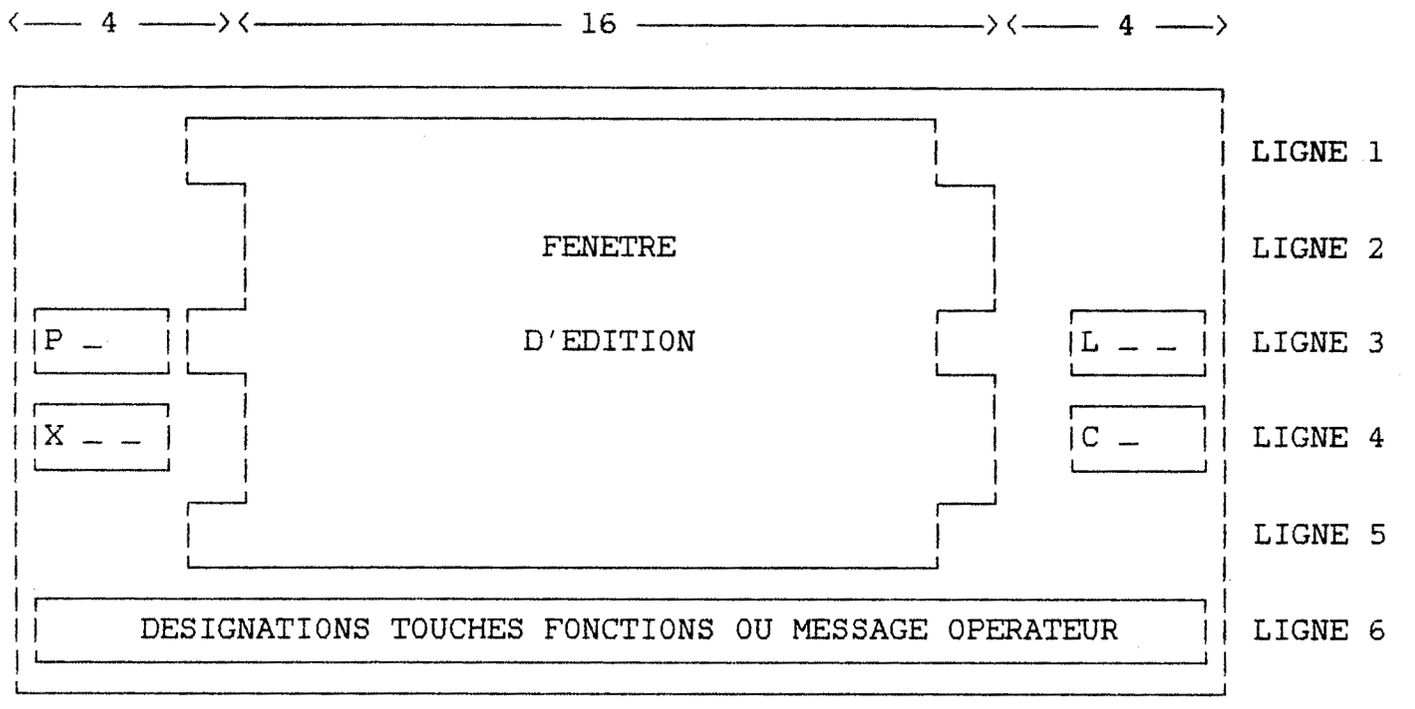
De façon à alléger les procédures de contrôle du respect des règles de construction du GRAFCET, et en particulier celle de l'alternance étape - transition, certaines contraintes ont été introduites sur la localisation des différents éléments graphiques. Ainsi une étape ne pourra être saisie que sur une ligne de rang impair, une transition uniquement sur une ligne de rang pair (le choix pair-impair est arbitraire, le choix inverse aurait été équivalent). Pour la même raison, les "poursuites" de liaisons verticales s'effectueront par portions de longueur égale à deux lignes. De manière analogue, on diminuera la complexité des procédures d'appariement des renvois en ne permettant d'interrompre que les liaisons transition - étape.

La figure suivante fait la synthèse entre les contraintes de localisation des constituants d'un GRAFCET et les graphismes nécessaires et suffisants à sa construction.



III.2.1.2 **Organisation de l'écran d'affichage.**

Les 6 lignes de 24 caractères disponibles sur l'afficheur sont réparties de la manière suivante :

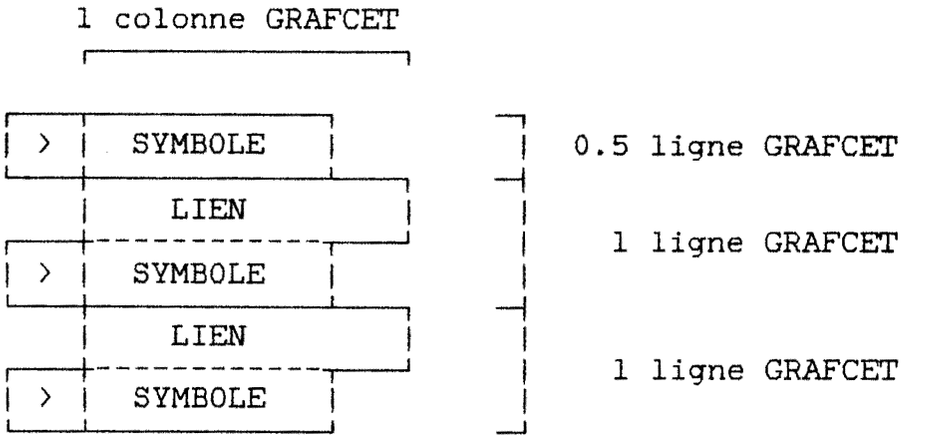


On distingue :

- les champs P_ L__ C_ de localisation topologique de l'emplacement courant,
- le champ X_ n'apparaissant qu'aux moments où l'opérateur est invité à saisir le N° d'une étape,
- la fenêtre d'édition dans la page GRAFCET courante,
- les désignations des différentes touches fonctions valides à un instant donné; ces désignations pouvant être ponctuellement remplacées par un message destiné à l'opérateur, généralement un message d'erreur.

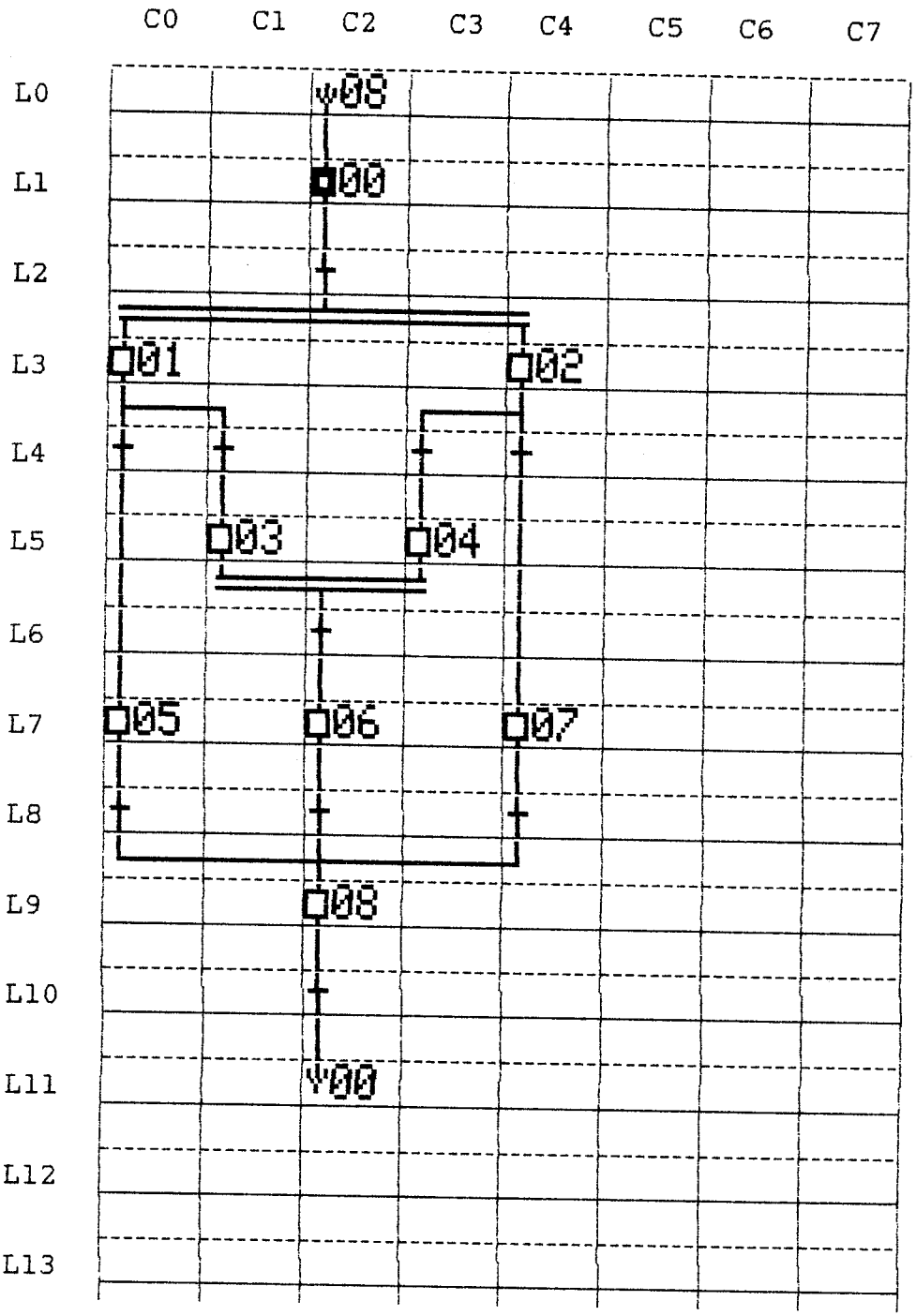
La forme quelque peu bizarre de la fenêtre d'édition est dûe aux emplacements réservés pour le curseur; cette impression n'est nullement conservée en visualisation réelle.

La fenêtre d'édition visualise en permanence une portion de page GRAFCET de 4 colonnes en largeur et de 2,5 lignes en hauteur, en ce sens que l'on ne verra pas apparaître les parties liens des emplacements situés sur la ligne GRAFCET du haut, permettant néanmoins un regard sur 2 alternances étape - transition complètes.

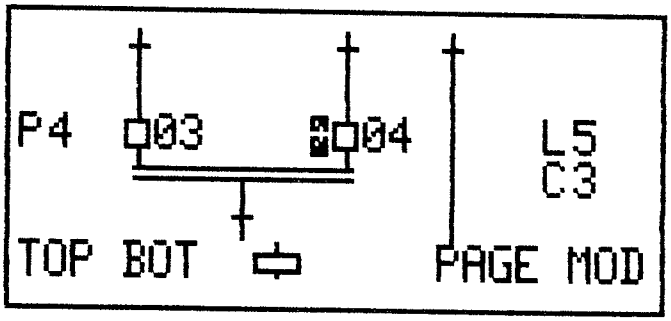


Les mouvements du curseur sont entièrement libres sur les 12 emplacements possibles de la fenêtre : en cas de butée contre l'un des 4 bords, la fenêtre sera repoussée soit d'une ligne soit d'une colonne dans la direction considérée, compte tenu des limites physiques attribuées à une page GRAFCET.

Un exemple de page GRAFCET



et sa visualisation au travers de la fenêtre d'édition

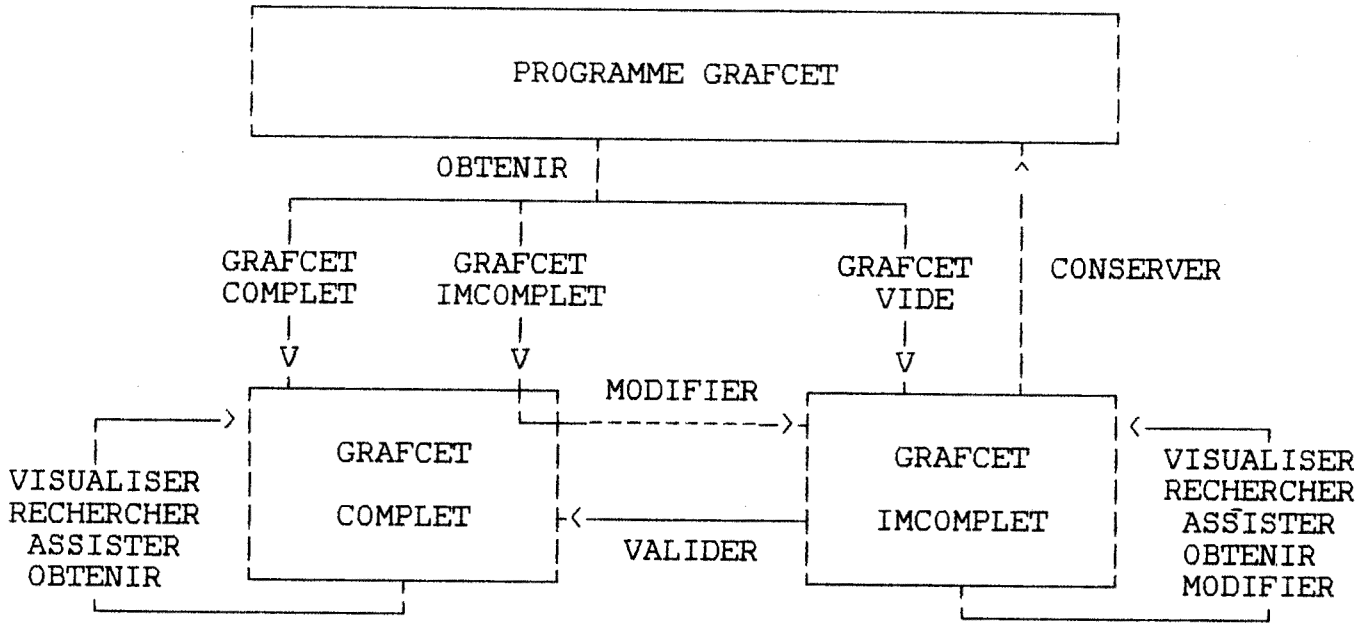


III.2.1.3 **Enchaînement des fonctions d'édition.**

Avant d'entrer dans le détail de la réalisation des différentes fonctions de l'éditeur, il semble auparavant utile d'en présenter une liste, de préciser leur portée, leurs interactions réciproques ainsi que les instants où elles doivent être disponibles.

- OBTENIR une page GRAFCET,
- VISUALISER/IMPRIMER l'état courant d'une page GRAFCET,
- RECHERCHER/DESIGNER un élément, un endroit précis,
- MODIFIER (créer, effacer, remplacer, déplacer) un élément,
- VALIDER l'ensemble du programme GRAFCET,
- CONSERVER les modifications du programme GRAFCET,
- ASSISTER/GUIDER l'opérateur.

Si l'on considère le fait que le programme GRAFCET est stocké en un "lieu" différent du terminal de programmation, on obtient le graphe des disponibilités des fonctions suivant :



RECHERCHER, VISUALISER, ASSISTER et OBTENIR doivent être disponibles à tout moment.

MODIFIER, VALIDER, CONSERVER seulement quand le GRAFCET est incomplet, inexistant, ou qu'une modification est nécessaire.

Ces quelques remarques nous permettent de mettre en évidence deux modes principaux :

- un mode visualisation - modification interdite, correspondant à l'état "complet" du GRAFCET. C'est un mode de consultation dans lequel il est possible de visualiser, eventuellement d'imprimer, le GRAFCET contenu dans une page, une ou plusieurs actions et réceptivités.

- un mode visualisation - modification autorisée, correspondant à l'état "incomplet" du GRAFCET. C'est le mode majeur dans lequel il est possible de saisir, modifier, effacer, déplacer les éléments GRAFCET, les actions et réceptivités.

Selon leur fréquence d'utilisation, leur spécificité par rapport au langage GRAFCET, on accèdera aux différentes fonctions soit par des touches clavier à désignation fixe, soit par les touches à désignation dynamique ou touches "menu".

Les touches à désignation fixe décrites ci-après sont toujours validées et ne changent pas de sens tout au long de la session d'édition.

← : pointer la colonne GRAFCET de gauche

→ : pointer la colonne GRAFCET de droite

↑ : pointer la ligne GRAFCET du dessus

↓ : pointer la ligne GRAFCET du dessous

L : pointer une ligne GRAFCET donnée

C : pointer une colonne GRAFCET donnée

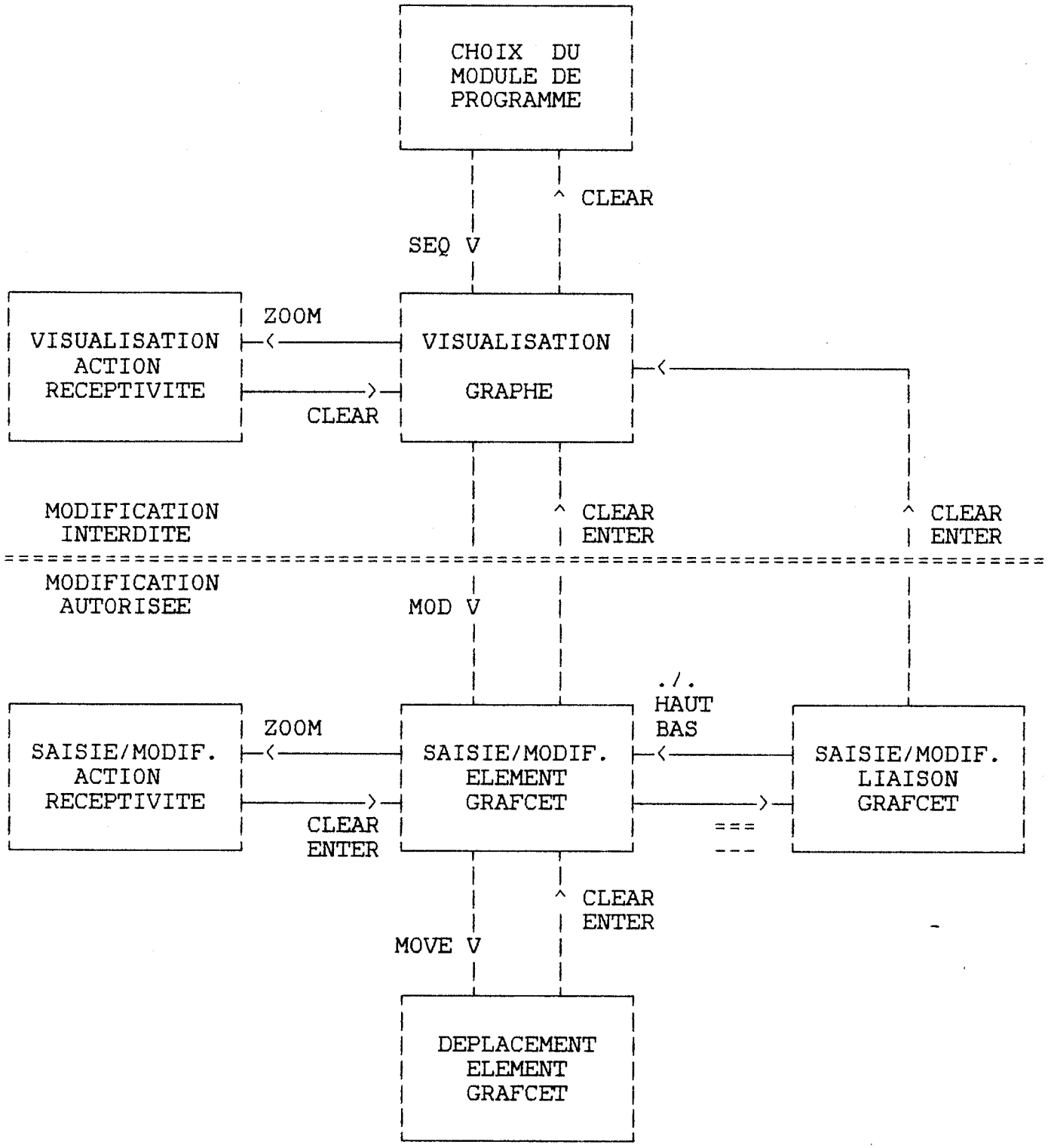
P : pointer une page GRAFCET donnée

X : pointer une étape GRAFCET donnée

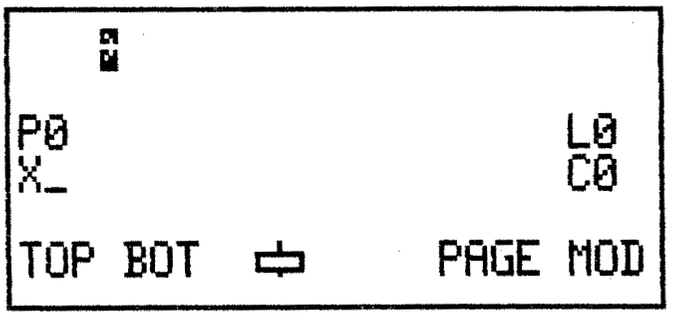
PRINT : imprimer la page GRAFCET affichée

QUIT : abandon et retour à l'écran de connexion

La figure ci-dessous indique l'organisation globale de l'arbre d'enchainemenent des écrans de l'éditeur; les pages suivantes donneront des indications plus détaillées sur les fonctions associées à chacun de ces écrans.



1) **Visualiser le programme GRAFCET.**



TOP : accès à la première page GRAFCET non vide

BOT : accès à la dernière page GRAFCET non vide

⇨ : accès à une étape donnée dans une page GRAFCET quelconque

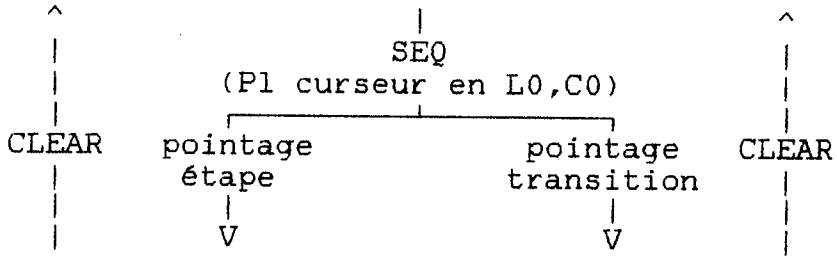
PAGE : accès à une page GRAFCET donnée

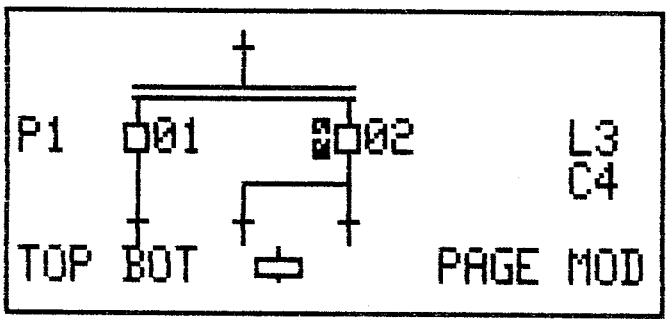
MOD : accès en modification dans la page GRAFCET courante

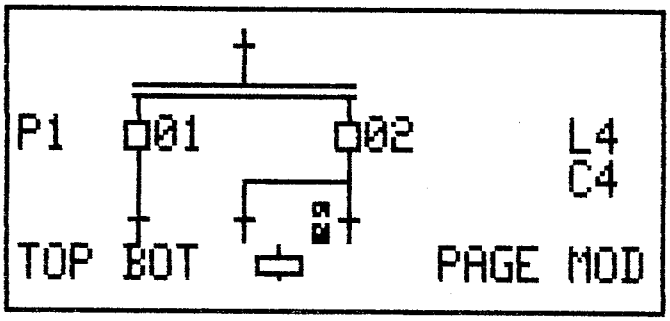
ZOOM : visualisation d'une action, d'une réceptivité

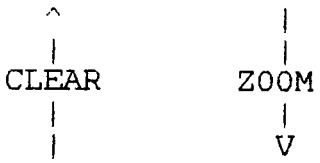
CLEAR : retour à l'écran de sélection des modules programmes

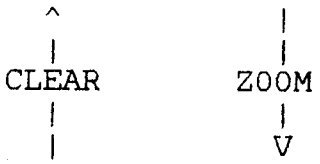

PRE: trait. preliminaire
SEQ: lancase GRAFCET
POS: trait. posterieur
PRE SEQ POS ./.

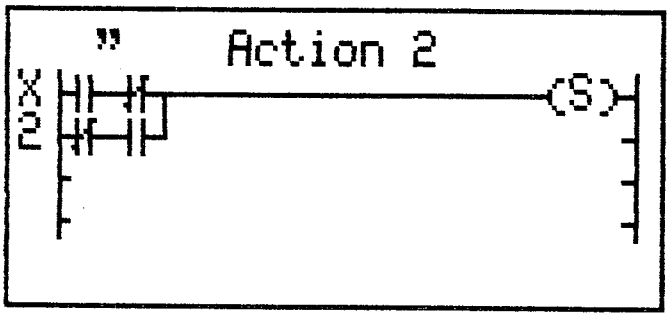

SEQ
(P1 curseur en L0,C0)
CLEAR
pointage étape
pointage transition
CLEAR
V
V

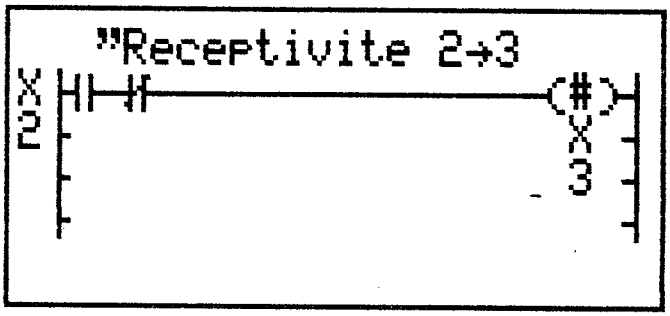

P1
01
02
L3
C4
TOP BOT
PAGE MOD


P1
01
02
L4
C4
TOP BOT
PAGE MOD

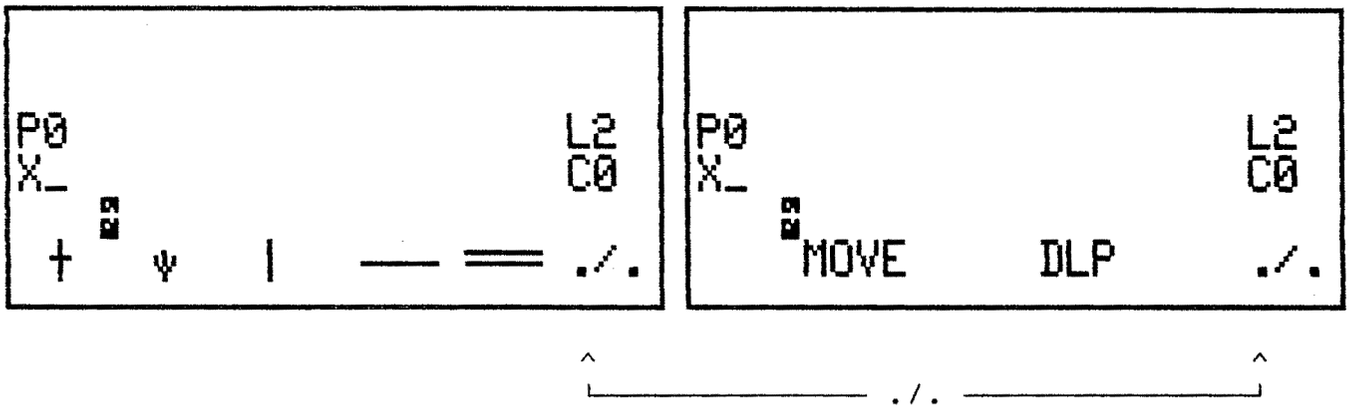

CLEAR
ZOOM
V


CLEAR
ZOOM
V

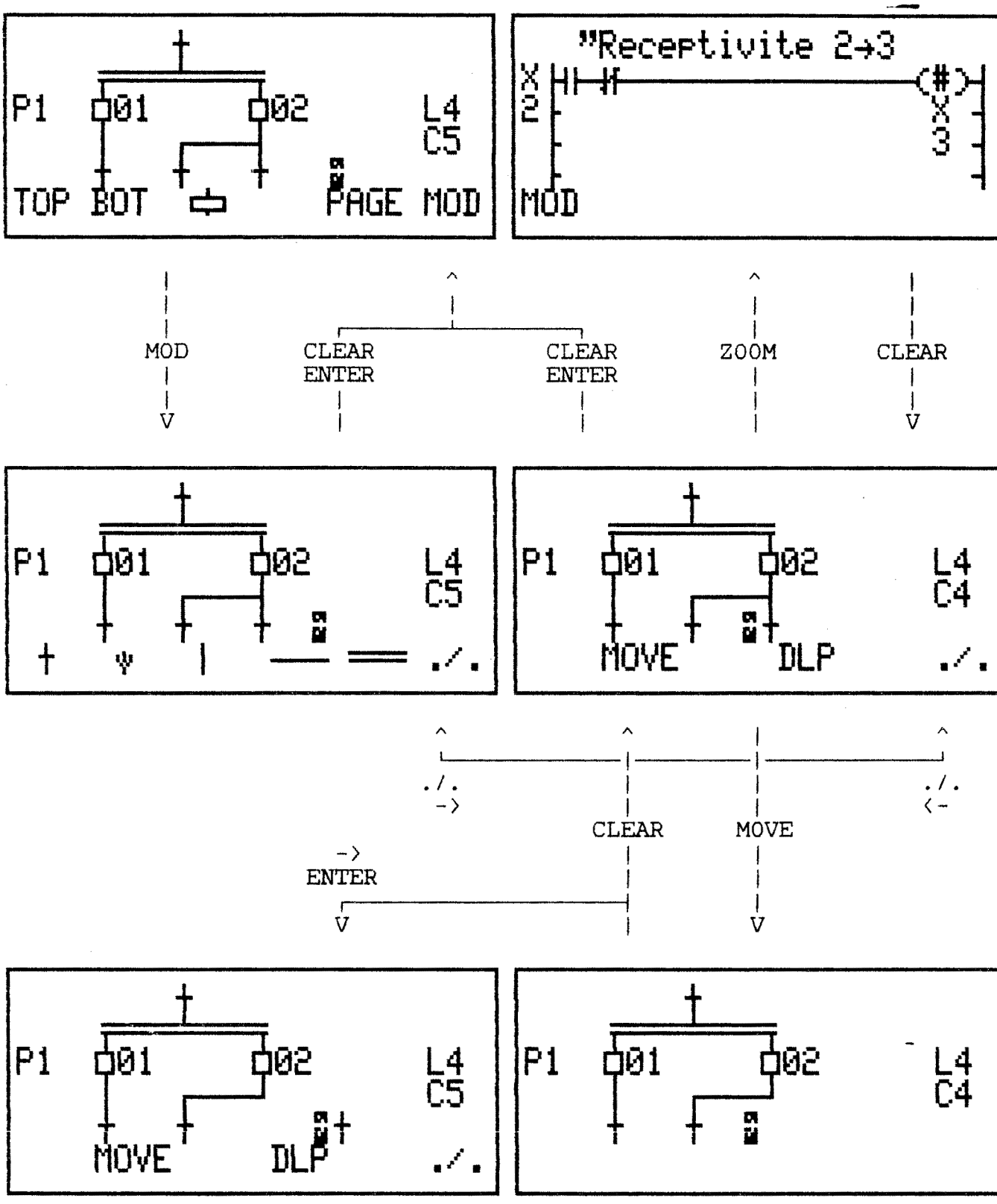

" Action 2
X 2
(S)


"Receptivite 2→3
X 2
(#)
X 3

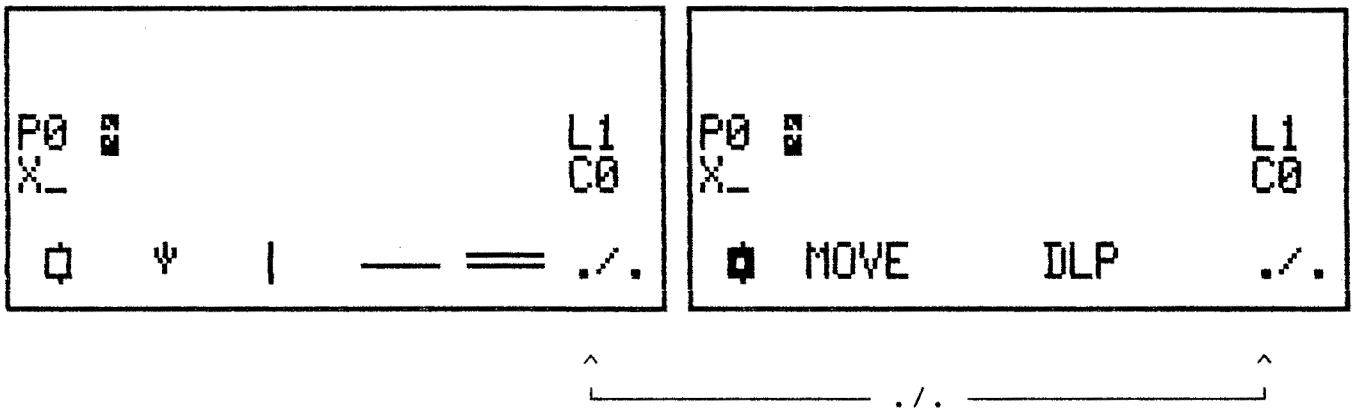
2) **Saisir/modifier/déplacer/effacer élément GRAFCET, lignes paires.**



†	: saisie d'une transition
ψ	: saisie d'un renvoi de provenance
|	: poursuite d'une liaison verticale vers le bas
——	: entrée en mode saisie de divergence en OU
═══	: entrée en mode saisie de convergence en ET
MOVE	: déplacement d'une transition ou d'un renvoi de provenance
DLP	: effacement complet de la page GRAFCET courante
ZOOM	: modification d'une réceptivité (transition pointée)
DEL	: effacement du symbole ou de la liaison verticale pointé
ENTER	: validation du programme GRAFCET
CLEAR	: retour à l'écran de visualisation du GRAFCET


P1
01
02
L4
C5
TOP BOT
PAGE MOD
"Receptivite 2→3
X
2
X
3
MOD
MOD
CLEAR
ENTER
CLEAR
ENTER
ZOOM
CLEAR
L4
C4
MOVE
DLP
./.
CLEAR
MOVE
ENTER

3) **Saisir/modifier/déplacer/effacer élément GRAFCET, lignes impaires.**



 : saisie d'une étape

 : saisie d'un renvoi de destination

 : poursuite d'une liaison verticale vers le bas

 : entrée en mode saisie de convergence en OU

 : entrée en mode saisie de divergence en ET

 : saisie d'une étape initiale

MOVE : déplacement d'une étape ou d'un renvoi de destination

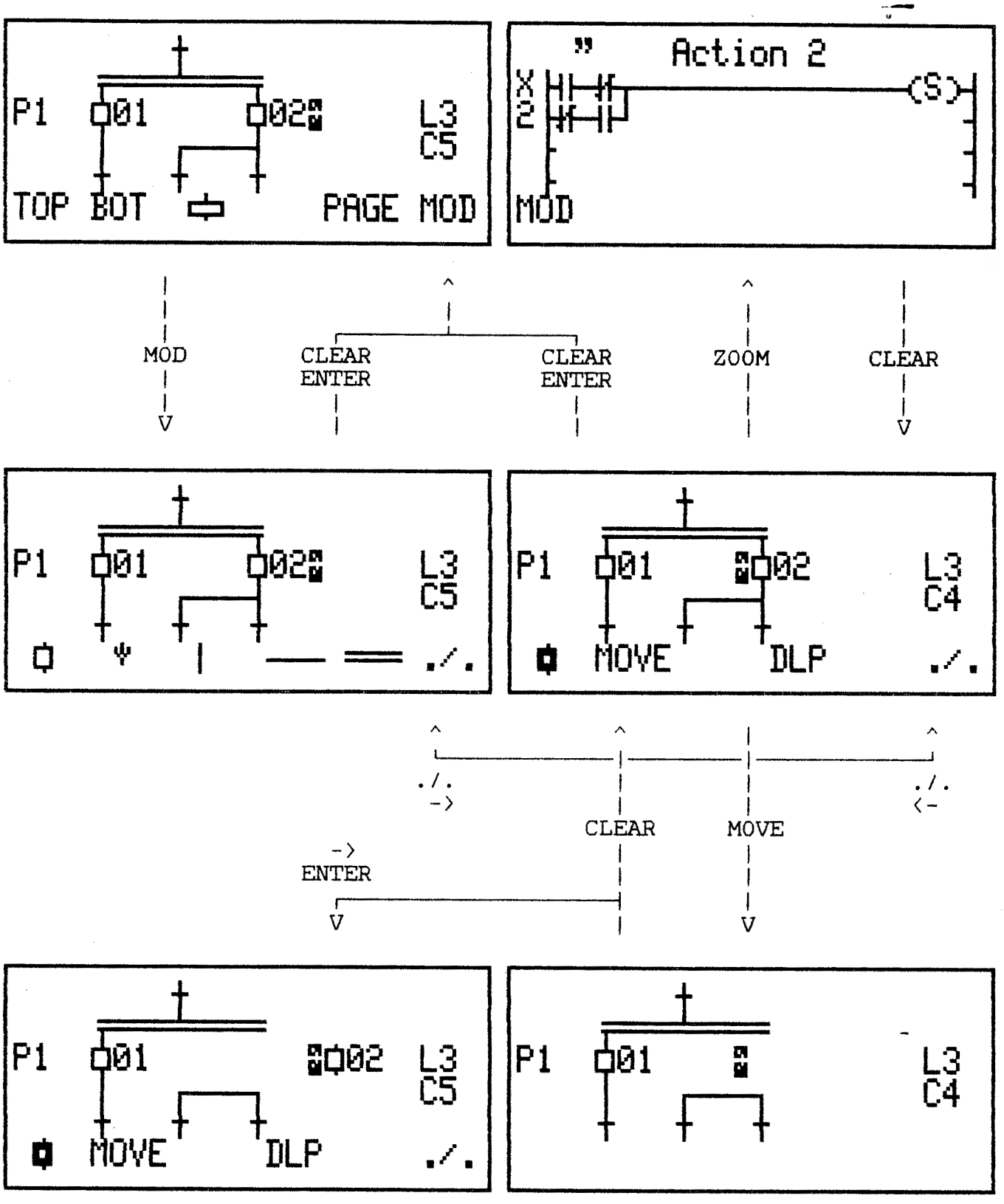
DLP : effacement complet de la page GRAFCET courante

ZOOM : modification d'une action (étape pointée)

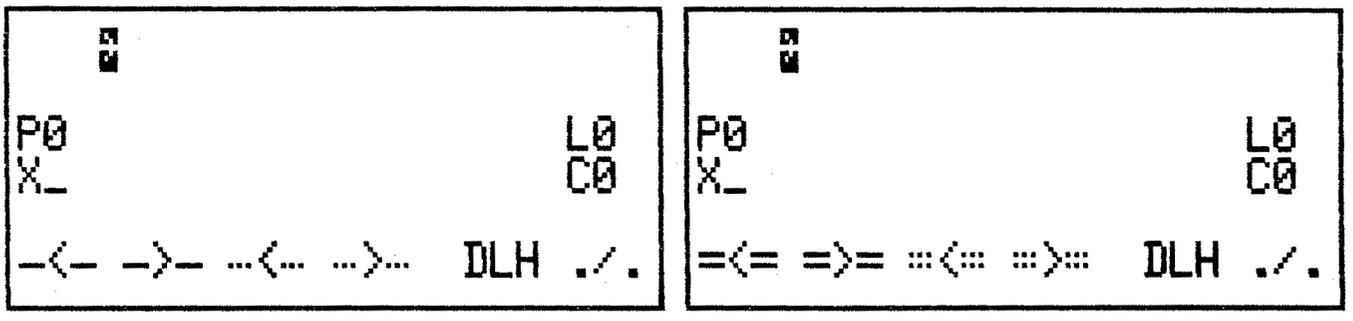
DEL : effacement du symbole ou de la liaison verticale pointé

ENTER : validation du programme GRAFCET

CLEAR : retour à l'écran de visualisation du GRAFCET


P1
01
02
L3
C5
TOP BOT
PAGE MOD
"
Action 2
X
2
(S)
MOD
MOD
CLEAR
ENTER
CLEAR
ENTER
ZOOM
CLEAR
L3
C4
MOVE
DLP
./.
->
<-
ENTER
CLEAR
MOVE

4) **Saisir/modifier/effacer une liaison horizontale.**



—(— : créer un pas de divergence, convergence en OU vers la gauche

—)— : créer un pas de divergence, convergence en OU vers la droite

...(... : effacer un pas de divergence, convergence en OU vers la gauche

...)... : effacer un pas de divergence, convergence en OU vers la droite

=(= : créer un pas de convergence, divergence en ET vers la gauche

=)= : créer un pas de convergence, divergence en ET vers la droite

:::(::: : effacer un pas de convergence, divergence en ET vers la gauche

:::):::: : effacer un pas de convergence, divergence en ET vers la droite

DLH : effacer entièrement le lien horizontal pointé

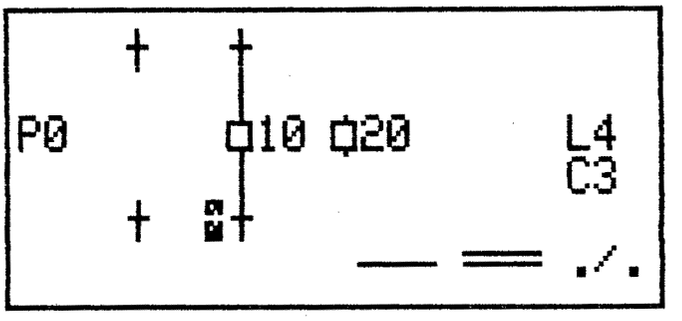
./. : retour à l'écran de modification des symboles

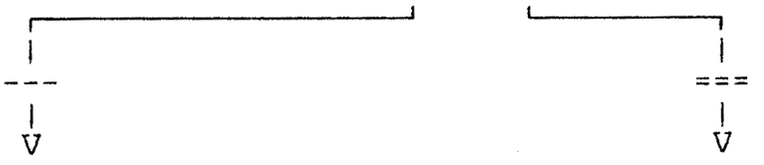
ZOOM : modification d'une action, d'une réceptivité

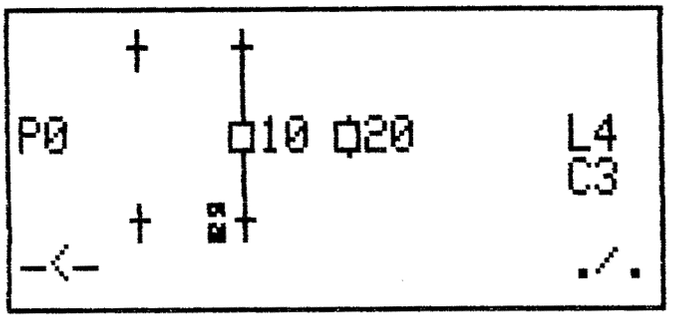
DEL : effacement du symbole ou de la liaison verticale pointé

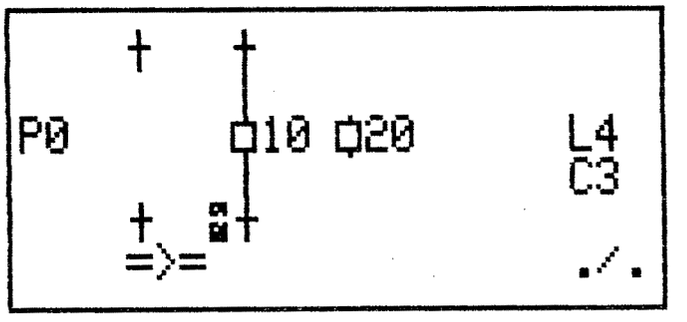
ENTER : validation du programme GRAFCET

CLEAR : retour à l'écran de visualisation du GRAFCET



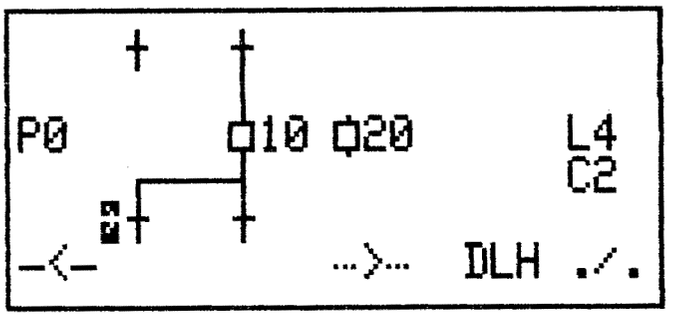


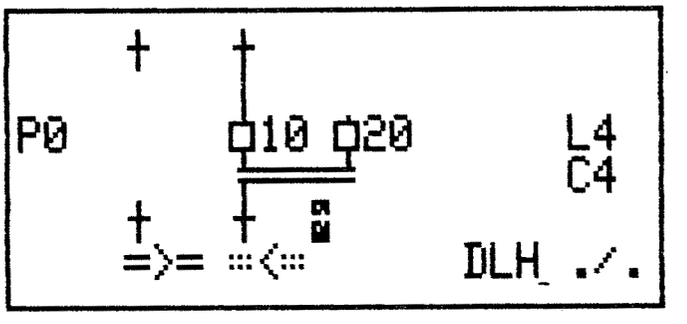


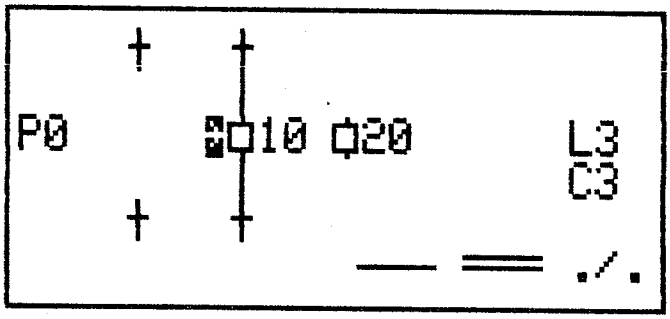


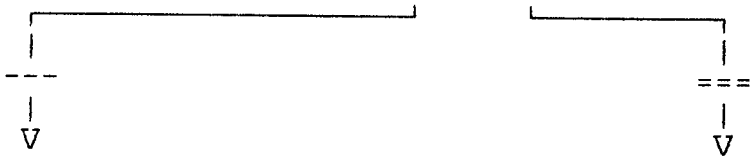
-<- saisie d'une divergence en OU

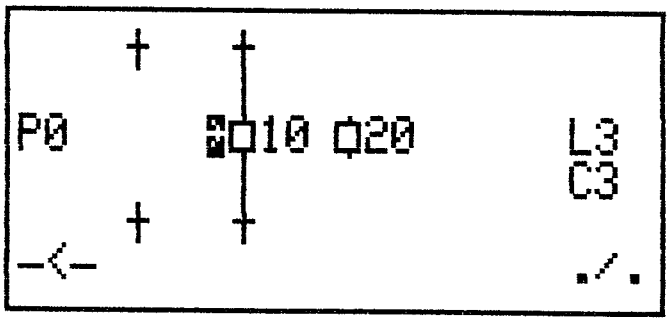
=>= saisie d'une convergence en ET

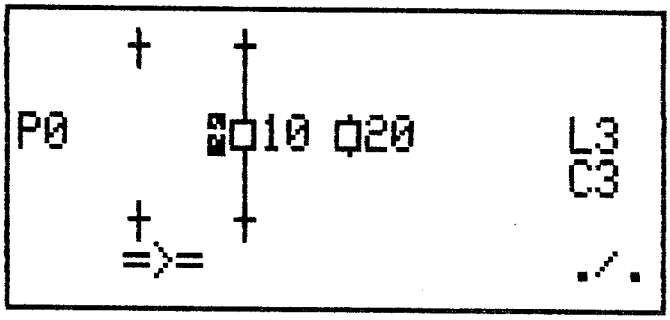






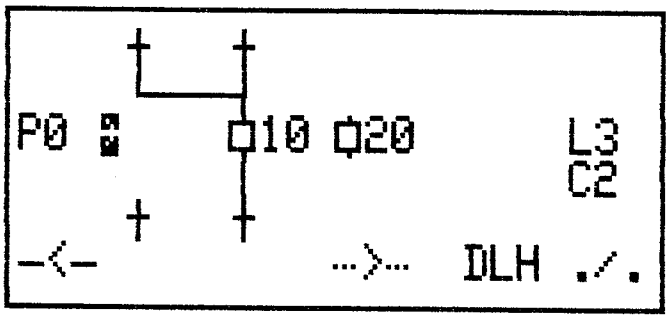


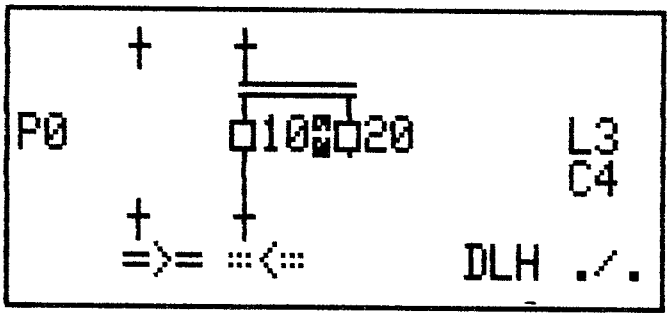




-<- saisie d'une convergence en OU

=>= saisie d'une divergence en ET



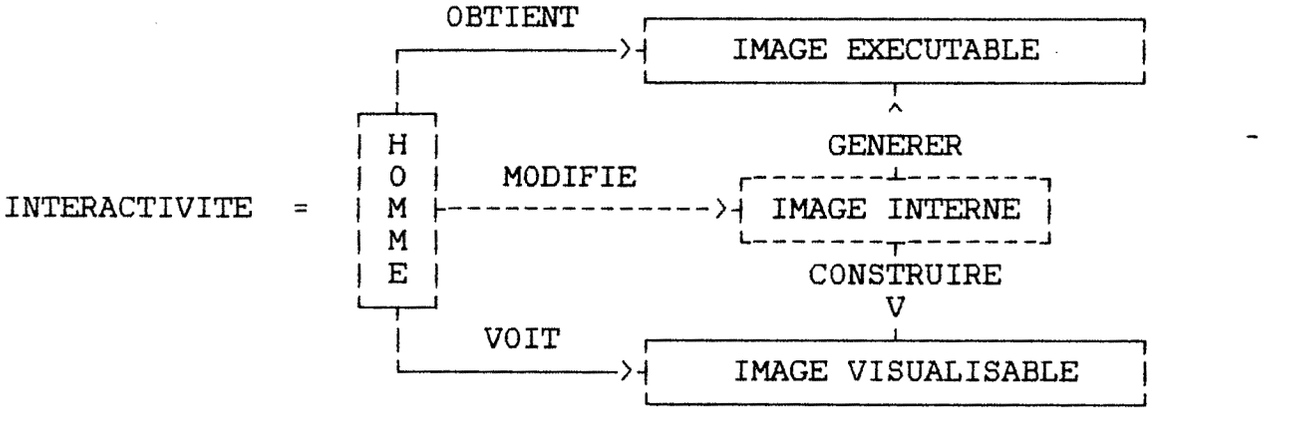


III.2.1.4 **Organisation du logiciel.**

Converties en termes de réalisation logicielle, les fonctions de l'éditeur se reformulent de la façon suivante :

- OBTENIR c'est "rapatrier" de l'automate vers la console les informations attachées à une page GRAFCET,
- VISUALISER c'est montrer l'emplacement courant et son environnement immédiat,
- RECHERCHER c'est changer la localisation de l'emplacement courant,
- CREER, EFFACER c'est modifier le contenu de l'emplacement courant,
- DEPLACER c'est transférer le contenu de l'emplacement courant vers un endroit différent, cette nouvelle localisation devenant le nouvel emplacement courant,
- CONSERVER c'est "mémoriser" le nouvel état du ou des emplacements modifiés
- VALIDER c'est vérifier qu'il existe au moins une liaison entre chaque étape et chaque transition, que chaque renvoi de provenance est apparié à un renvoi de destination,
- enfin, ASSISTER c'est permettre ou interdire une modification au vu des règles de construction du GRAFCET; c'est aussi signaler une liaison omise, un renvoi non apparié, demander confirmation lors des manoeuvres dangereuses (effacement total d'une page GRAFCET, sortie de l'éditeur sans validation préalable, ...).

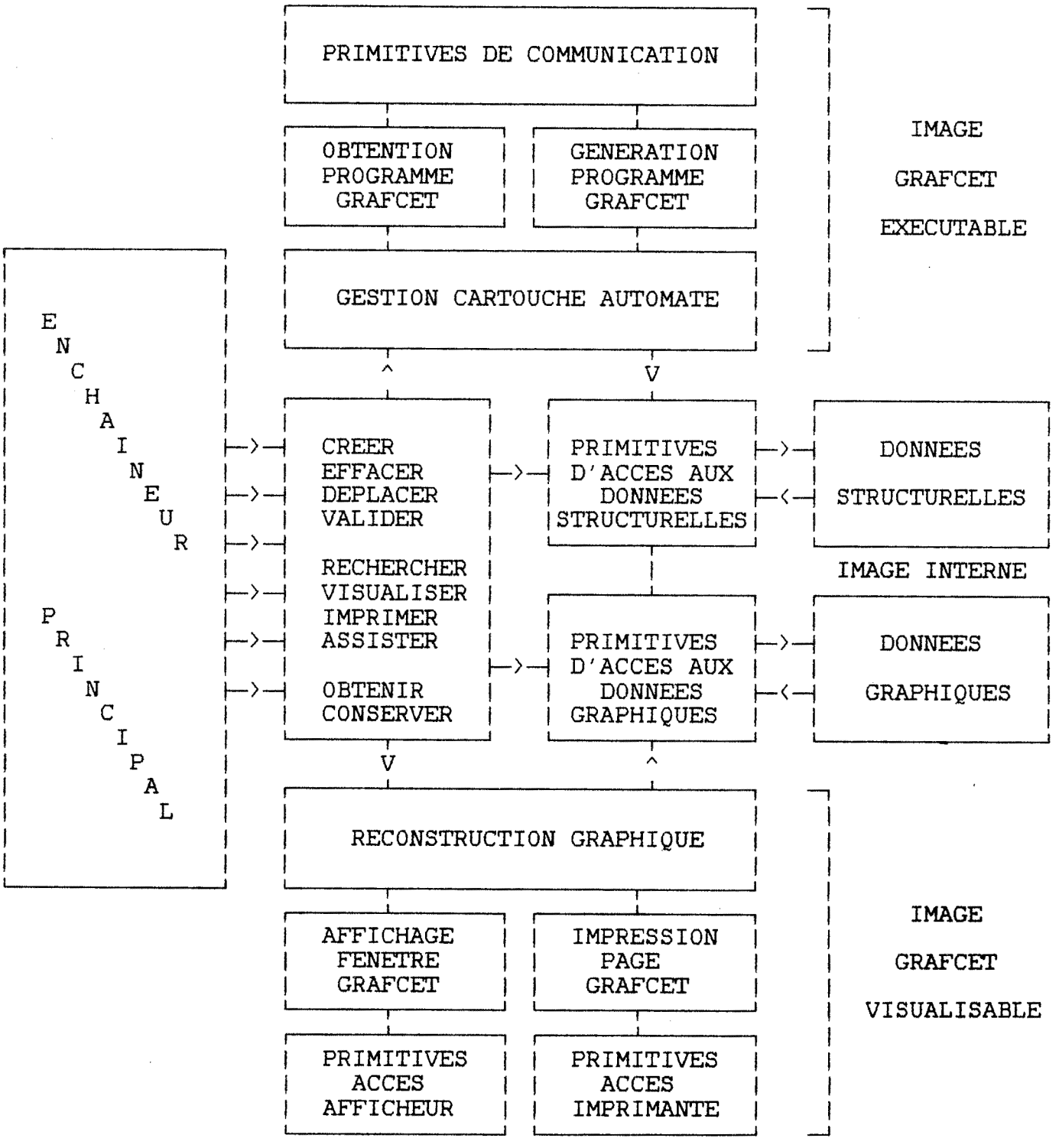
En terme d'INTERACTIVITE, le fonctionnement global de l'éditeur se schématise ainsi :



L'éditeur se doit de maintenir en permanence une stricte équivalence entre deux formes différentes du programme GRAFCET, en ce sens que l'automate sera capable d'exécuter sans délai ni autre manoeuvre intermédiaire ce que l'opérateur aura dessiné sur son terminal de programmation.

Une découpe modulaire du logiciel de l'éditeur distinguera, articulés autour d'un ensemble de primitives d'accès à une image interne du programme GRAFCET (données structurelles + données graphiques) :

- une interface opérateur, en écoute permanente du clavier, traduisant les différentes sollicitations de ce dernier en termes d'interventions sur l'image interne,
- un module transformant cette image interne en image visualisable,
- un module la convertissant en image exécutable.



III.2.2 **Les outils de mise au point du GRAFCET.**

Quelles sont les questions que se pose l'utilisateur pendant la mise au point de son programme :

- qu'est-ce qu'il peut bien se passer ?
- pourquoi une étape ne s'active pas ?
- pourquoi une réceptivite ne passe pas ?
- pourquoi une action conditionnelle n'est pas effectuée ?

Il a besoin autant d'une vision globale de la situation du GRAFCET que d'une vision sur un élément plus localisé. La taille réduite de l'afficheur ne permettait pas de proposer chacun de ces deux niveaux d'observation sous une forme graphique. Ainsi :

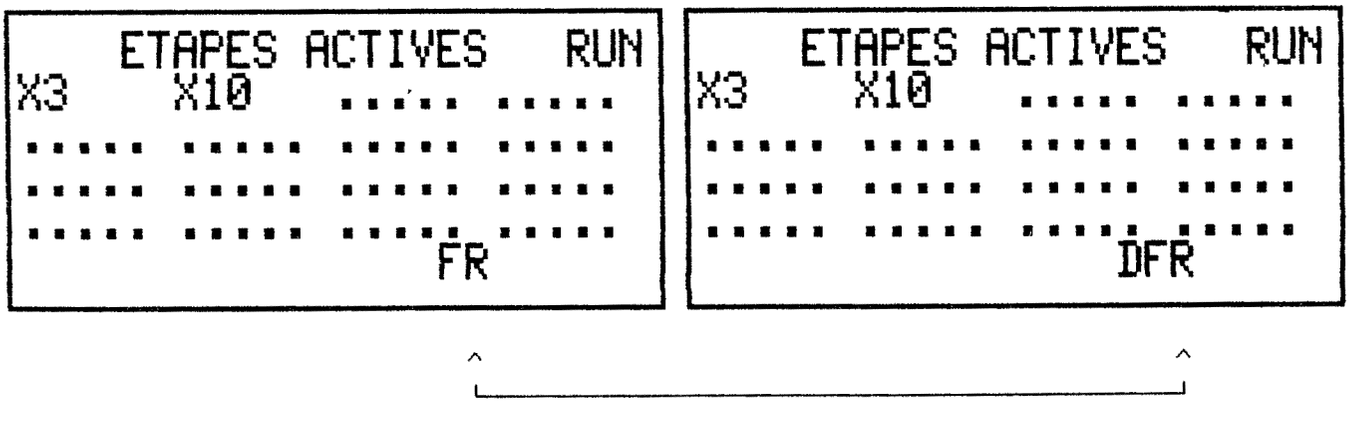
- le niveau d'observation global de la situation du GRAFCET sera matérialisé sous la forme d'une liste : la liste des étapes actives,
- le niveau d'observation locale sera par contre graphique : la structure du GRAFCET étant alors visible, ce niveau servira de support pour l'accès aux fonctions commandant la marche étape par étape.

III.2.2.1 **Visualisation de la situation du GRAFCET.**

La situation du GRAFCET est présentée par la liste des numéros des étapes actives, ordonnée selon un ordre croissant, l'affichage de cette liste étant réactualisée en permanence.

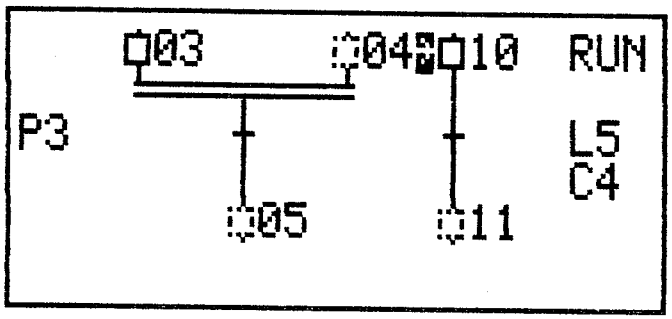
L'animation de la liste pourra être suspendue et reprise à tous moments par l'appui sur une touche dont la signification changera alternativement de FR pour FREEZE à DFR pour DEFREEZE. La lecture de cette liste se trouvera ainsi facilitée lorsque l'évolution est trop rapide vis à vis des capacités de perception humaine.

Le fait de figer l'animation de la liste n'entraînera pas l'arrêt de l'évolution du GRAFCET.



III.2.2.2 **Animation du GRAFCET.**

Le GRAFCET est animé sous la forme graphique par laquelle il a été saisi. Les étapes actives sont représentées en trait plein, les étapes inactives en trait pointillé. La partie de GRAFCET présente sur l'afficheur sera rafraîchie continuellement en fonction de l'état actif ou inactif des étapes visibles.



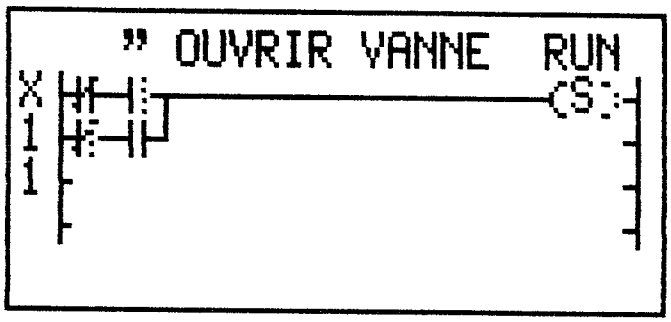
L'intégralité des fonctions (déplacement, recherche, pointage direct) destinées à changer l'espace visualisé par la fenêtre, fonctions définies au chapitre "éditeur graphique", sont également disponibles ici.

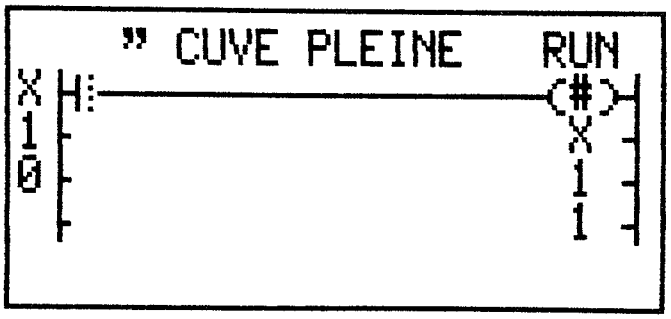
De la même manière qu'en saisie, un appui sur la touche ZOOM permettra l'accès aux actions et aux réceptivités sous une version animée. Le langage utilisé à ce niveau étant le langage LADDER, ce seront alors les symboles des contacts et des bobines que l'on représentera en traits pleins ou pointillés. Les significations que l'on attribuera à chaque type de trait dépendront de la nature du contact ou de la bobine considérée :

- pour un contact à fermeture (symbole —| |—), un pointillé signifiera que la variable associée est à la valeur 0, un trait plein à la valeur 1; ce sera l'inverse pour un contact à fermeture (symbole —|/|—),

- des conventions identiques seront adoptées pour les bobines :
  - une bobine de "mise à 1" (symbole —(S)—|) sera vue en pointillé lorsque la variable associée vaudra 0,
  - une bobine de "mise à 0" (symbole —(R)—|) sera vue en pointillé lorsque la variable associée vaudra 1.

Il n'a pas été possible d'animer les bobines —(#)- symbolisant les conditions de franchissement des transitions. Ces bobines sont "fictives", au sens où aucune variable ne leur est associée, et au sens où elles ne sont présentes que pour respecter le formalisme du langage LADDER.

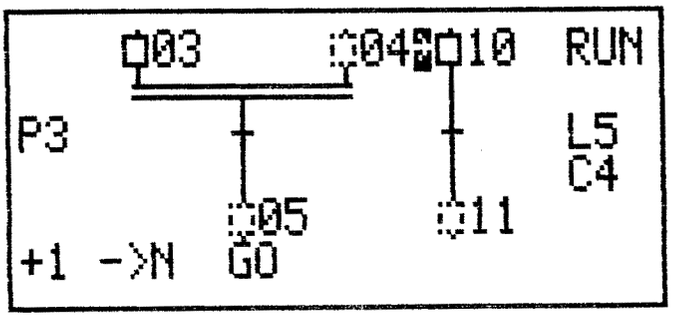




III.2.2.3 **Marche étape par étape.**

Une marche étape par étape sera introduite lorsque l'écran présentera une fenêtre GRAFCET animée. A cet effet, trois touches dynamiques (+1, ->N, GO) sont proposées à l'utilisateur. Chacune de ces touches réalisera une ou plusieurs fonctions.

Un seul point d'arrêt sera possible afin de simplifier l'utilisation. Le mode pas à pas permettra de suivre naturellement l'évolution sur une séquence linéaire d'étapes et de transitions.

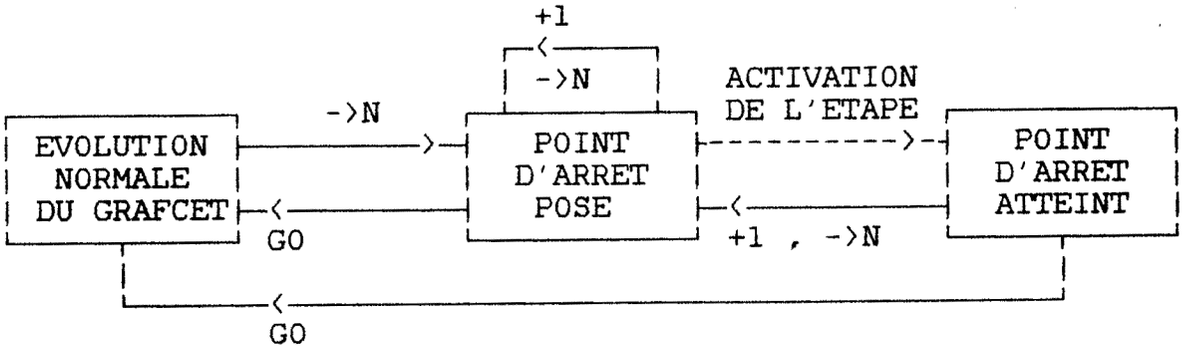


Touche "+1" :
- reporte automatiquement le point d'arrêt sur l'étape aval; la fenêtre de visualisation sera éventuellement déplacé de façon à ce que l'étape portant le point d'arrêt soit toujours visible,
- n'est validée que lorsque cette étape aval est unique : au cas où un choix se présente, on utilise "->N".

Touche "->N" :
- initialise le passage en mode pas à pas étape,
- permet de reporter le point d'arrêt n'importe où ailleurs,
- permet d'indiquer le choix lorsque plusieurs évolutions sont possibles (plusieurs étapes aval).

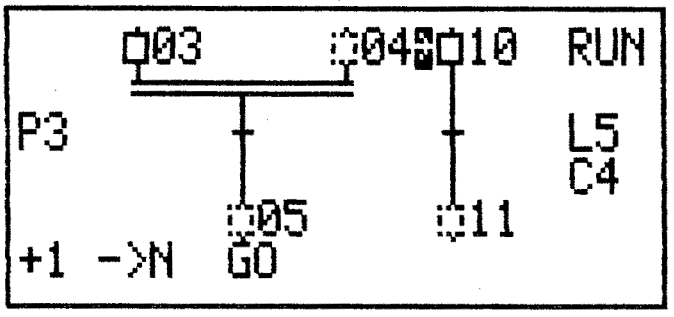
Touche "GO" :
- supprime le point d'arrêt, le GRAFCET évolue alors normalement.

La figure ci-dessous présente de manière synthétique les différents états possible du GRAFCET lors d'une marche étape par étape ainsi que les enchainements possibles entre ces états :



Chacun de ces états sera matérialisé dans le coin supérieur droit de l'écran :

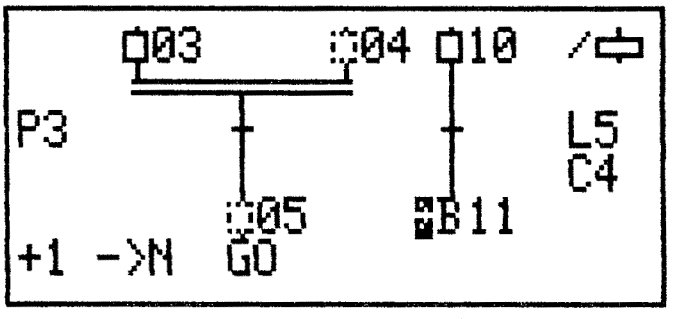
évolution
normale
V



Mise en place d'un point d'arrêt sur l'étape n° 11 :
"->N", "11"

V

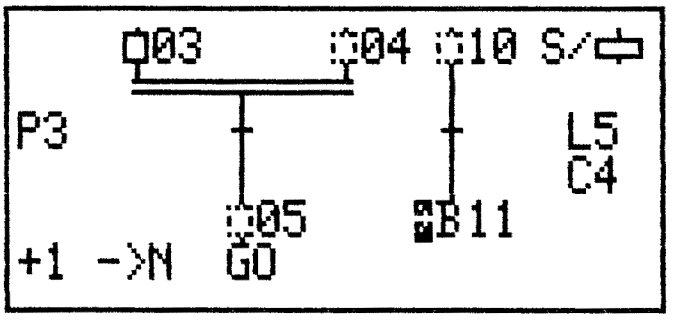
point d'arrêt
posé
V



Après activation
de l'étape N° 11

V

point d'arrêt
atteint
V



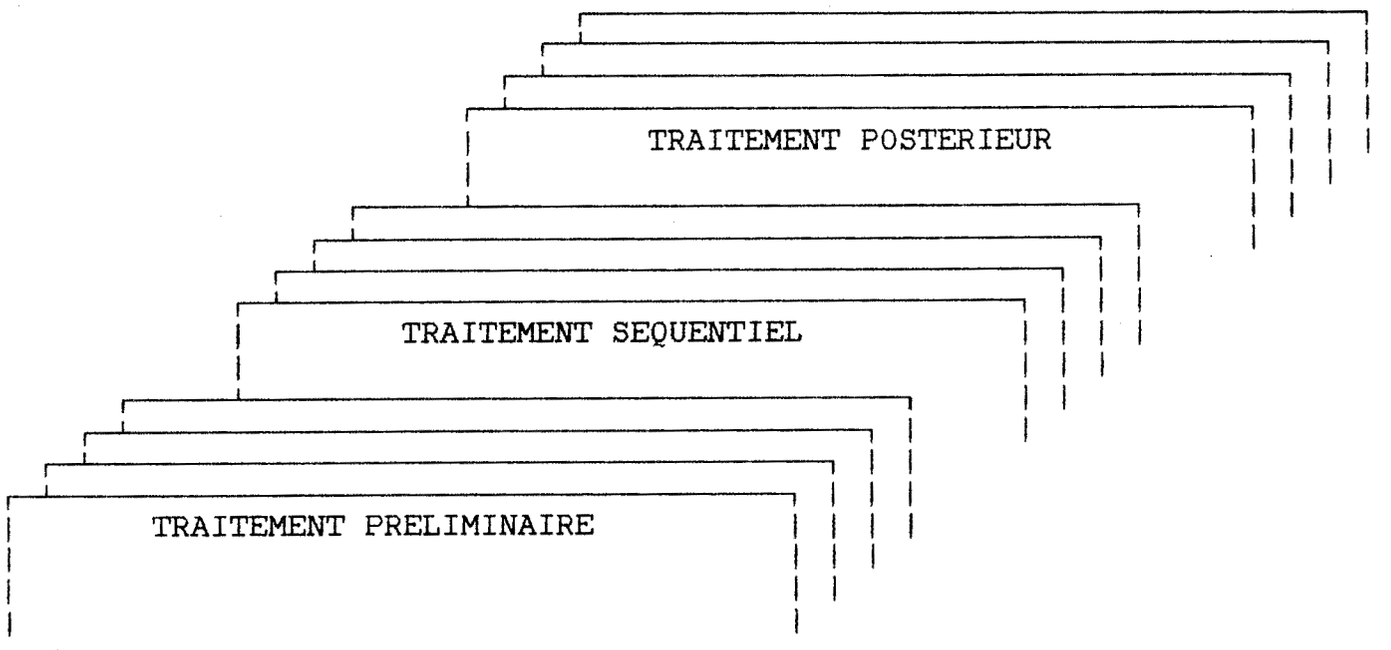
Remarque :

Lorsqu'un point d'arrêt est posé sur une étape, on remplace son symbole par la lettre B (Break-point). Un seul point d'arrêt étant possible, l'état actif ou non actif de l'étape est déterminé sans ambiguité à l'aide du contenu du coin supérieur droit.

III.2.3 **La documentation du GRAFCET.**

De manière générale, on obtiendra après une impression complète du programme utilisateur, une suite de feuillets contenant :

- la suite des réseaux LADDER décrivant le combinatoire préliminaire,
- les différentes pages GRAFCET du traitement séquentiel, suivies chacune par les réseaux LADDER représentant les actions et réceptivités associées aux étapes et transitions apparaissant sur une même page,
- la suite des réseaux LADDER du combinatoire postérieur.

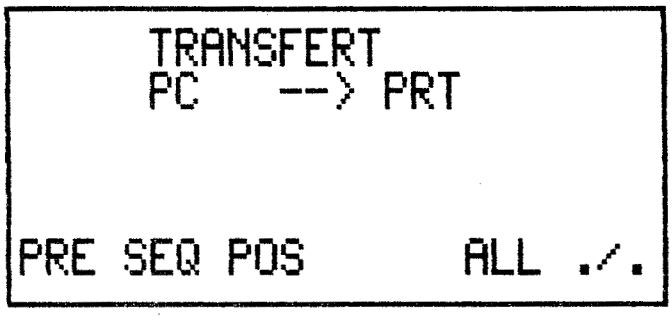


Chaque feuillet sera constitué :

- d'une en-tête indiquant :
  - . le type d'automate TELEMECANIQUE sur lequel le programme est destiné à être exécuté,
  - . le type de traitement auquel ce feuillet est attaché,
  - . le nom du programme,
  - . le n° d'ordre du feuillet,
- d'un contenu dépendant du traitement décrit :
  - . une suite de réseaux LADDER s'il s'agit du traitement préliminaire ou du traitement postérieur,
  - . soit le dessin d'une page GRAFCET, soit la suite de réseaux LADDER décrivant les actions et réceptivités associées à une même page GRAFCET s'il s'agit du traitement séquentiel.

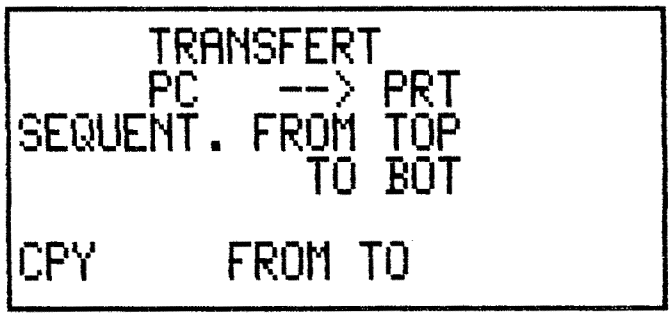
L'accès aux fonctions d'édition du dossier de programmation s'effectue à partir du mode TRANSFERT du terminal de programmation : une impression de programme étant considérée comme un "transfert vers imprimante". Le programme utilisateur étant contenu dans une cartouche mémoire amovible, la "source" d'un tel transfert pourra être soit l'automate, soit le terminal lui-même.

Supposons le cas d'une localisation de la cartouche sur l'automate. L'utilisateur pourra faire apparaître sur son terminal l'écran ci-après:



Plusieurs possibilités s'offrent alors à lui :

- commander l'impression de l'intégralité du dossier de programmation (appui sur ALL),
- commander l'impression de tout ou partie des feuillets relatifs à l'un des trois modules de traitement en provoquant l'apparition (appui sur PRE, SEQ ou POS) de l'écran ci-dessous (module choisi : traitement séquentiel) :



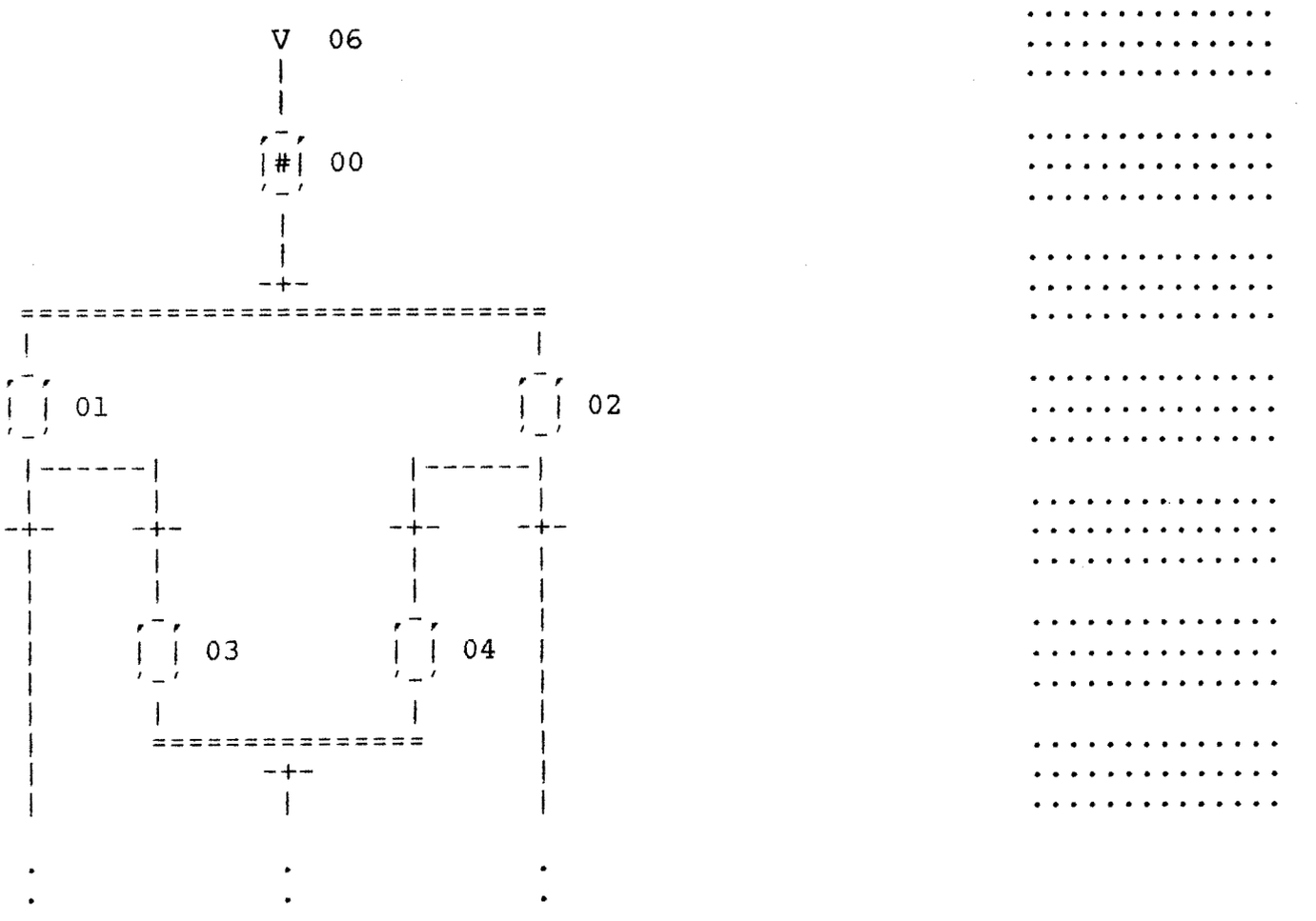
un appui sur CPY déclanchera alors l'impression de la suite des feuillets relatifs au traitement séquentiel, soit la totalité, soit une partie dont les "bornes" seront choisies à l'aide des touches FROM et TO.

La figure ci-dessous montre l'allure générale d'un dossier de programmation.

. . . . .

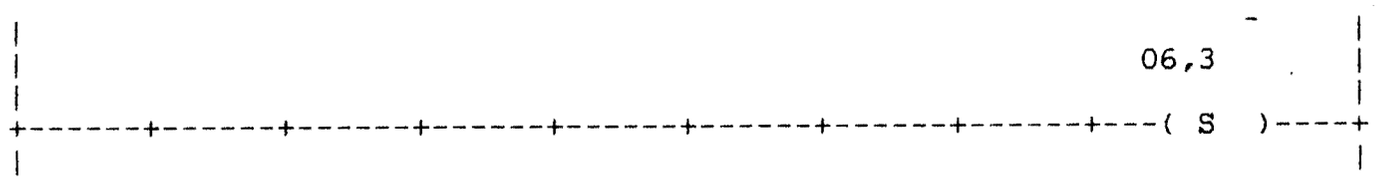
TELEMECANIQUE : TSX47 T.SEQUENTIEL "EMBOUTISSAGE N5 " PAGE 15

PAGE GRAFCET : 3

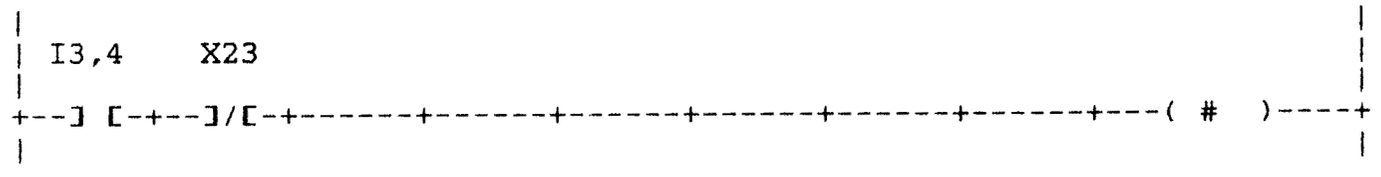


TELEMECANIQUE : TSX47 T.SEQUENTIEL "EMBOUTISSAGE N5 " PAGE 16

LABEL : X0 "VOYANT MARCHE "



LABEL : X0 -> X1 "DEPART CYCLE "



LABEL : X1 "DESCENTE BROCHE "

. . . . .

Extrait du dossier d'un programme d'emboutissage.

C O N C L U S I O N .

Les logiciels qui constituent le produit "langage GRAFCET" viennent d'être décrits. Pour permettre d'évaluer leur facilité d'emploi, leur aptitude à représenter et à exécuter du GRAFCET, et aussi leur efficacité, rien ne remplace les exemples de mise en pratique. Le concepteur, relativement éloigné des utilisateurs de son produit, n'a malheureusement que peu d'avis de ces derniers. Néanmoins, nous avons de bonnes raisons de penser que la réalisation proposée peut être appréciée.

Cette réalisation montre plusieurs points positifs :

- il est possible de conserver l'aspect graphique d'une description GRAFCET. L'éditeur graphique nécessaire, bien qu'ayant un certaine complexité, garde une taille tout à fait compatible avec les exigences du bas de gamme,

- il est possible de respecter dans une large mesure la norme du GRAFCET sans devoir introduire de mécanismes supplémentaires, et sans devoir non plus sacrifier des constructions essentielles.

- il est enfin possible d'implanter un interpréteur GRAFCET sur un automate aux caractéristiques restreintes sans problème technique majeur et surtout sans coût déraisonnable.

Cette étude met aussi en évidence quelques points délicats :

- La petitesse du format de l'écran de la console est un inconvénient. Toutefois cette limite n'empêche pas de proposer un outil facilement utilisable. A l'avenir, le progrès technologique en matière d'afficheur à cristaux liquides permettra vraisemblablement de disposer d'écrans beaucoup plus vastes et à faible coût. Cette tendance d'ailleurs se constate déjà. Sur la console du haut de gamme, cette limite disparaît naturellement par l'emploi d'un écran vidéo.

- Le GRAFCET seul n'est pas suffisant pour faire un langage de programmation complet. Il est nécessaire de lui adjoindre d'une part des outils de structuration, d'autre part un couplage avec un autre langage pour obtenir un outil général et exploitable en pratique.

C'est précisément dans ces deux derniers domaines que des progrès au niveau des normes restent à accomplir. En effet, il n'existe de norme ni pour définir les modes de marche, ni pour préciser les outils de structuration (par exemple des "macro-étapes"), ni enfin pour fixer le couplage avec un autre langage de programmation. Ce dernier point conduit actuellement à des réalisations différentes d'un constructeur à l'autre.

Comme d'autres constructeurs sur leurs automates de haut de gamme, TELEMECANIQUE a son offre GRAFCET sur les siens. Elle complète cette offre par la réalisation de ces travaux sur ses automates du bas de gamme.

Puisque ceux-ci sont destinés à une clientèle d'étendue mondiale, nous espérons que nos travaux contribuerons ainsi à une large diffusion du GRAFCET en France et à l'étranger.

ANNEXES.

A L'algorithme d'interprétation.

Structures de données statiques (description du GRAFCET).

```
MAX_ETP          constante  (* nombre maximum d'étapes *)
MAX_TRA          constante  (* nombre maximum de transitions *)
MAX_AMONT        constante  (* nombre maximum d'éléments en amont *)
MAX_AVAL         constante  (* nombre maximum d'éléments en aval *)

ETAPES_INITIALES          tableau [0..MAX_ETP] de bits

ACCES_ETAPES              tableau [0..MAX_ETP] de pointeurs
ACCES_TRANSITIONS         tableau [0..MAX_ETP] de pointeurs

BLOC_ETAPE                structure composée de

        LOCALISATION              information_graphique
        NOMBRE_TRANSITIONS_AVAL   nombre
        TRANSITIONS_AVAL          tableau [1..MAX_AVAL] de No transitions
        CODE_ACTION               code_ladder

BLOC_TRANSITION           structure composée de

        LOCALISATION              information_graphique
        NOMBRE_ETAPES_AMONT       nombre
        ETAPES_AMONT              tableau [1..MAX_AMONT] de No étapes
        NOMBRE_ETAPES_AVAL        nombre
        ETAPES_AVAL               tableau [1..MAX_AVAL] de No étapes
        CODE_RECEPTIVITE          code_ladder
```

Structures de données dynamiques (variables INTERPRETEUR).

```
MAX_ACT          constante  (* nombre maximum étapes "activables" *)
MAX_VAL          constante  (* nombre maximum transitions "validables" *)

LISTE_INITIALES           tableau [0..MAX_ACT] de No étapes
LISTE_ACTIVES             tableau [0..MAX_ACT] de No étapes
LISTE_ACTIONS             tableau [0..MAX_ACT] de No étapes

LISTE_VALIDEES            tableau [0..MAX_VAL] de No transitions

A       tableau [0..MAX_ETP] de bits  (* étapes "à activer" *)
D       tableau [0..MAX_ETP] de bits  (* étapes "à désactiver" *)
X       tableau [0..MAX_ETP] de bits  (* étapes "actives" courantes *)
X'      tableau [0..MAX_ETP] de bits  (* étapes "actives" futures *)
V       tableau [0..MAX_TRA] de bits  (* transitions "validées" *)

H       tableau [0..MAX_TRA] de bits  (* transitions "inhibées" *)

CPT     tableau [0..MAX_TRA]          (* compteurs de "validation" *)
```

Algorithme.

```
Si (GRAFCET_INITIALISATION)
    Alors
        (* Initialisation à partir des ETAPES_INITIALES *)
        Reset GRAFCET_INITIALISATION
        LISTE_INITIALES := [ ETAPES_INITIALES ]
        PHASE_0
        PHASE_2
        PHASE_3
    Sinon
        Si (GRAFCET_RAZ_GENERALE)
            Alors
                (* Initialisation toutes ETAPES inactives *)
                Reset GRAFCET_RAZ_GENERALE
                LISTE_INITIALES := [ vide ]
                PHASE_0
                PHASE_2
                PHASE_3
            Sinon
                Si (GRAFCET_REPRISE)
                    Alors
                        (* Initialisation à partir du vecteur [Xi] *)
                        Reset GRAFCET_REPRISE
                        LISTE_INITIALES := [ Xi ]
                        PHASE_0
                        PHASE_2
                        PHASE_3
                    Sinon
                        (* Détermination situation suivante *)
                        PHASE_1
                        PHASE_2
                        PHASE_3
                Fsi
        Fsi
Fsi
```

! PHASE 0 ! Initialisation - forçage de situation

```
LISTE_ACTIVES := vide

Pour chaque ETAPE i faire

     Reset Xi
     Reset Xi'
     Reset Di

     Si (i est dans LISTE_INITIALES)

         Alors

             Set Ai
             Placer i dans LISTE_ACTIVES

         Sinon

             Reset Ai
     Fsi
Fpour

LISTE_VALIDEES := vide

Pour chaque TRANSITION j faire

     Reset Vj
     CPTj := nombre ETAPES amont à j
Fpour
```

```
-----------   Evaluation RECEPTIVITES associées TRANSITIONS validées
! PHASE 1 !   Marquages ETAPES amont et aval TRANSITIONS franchies
-----------   Elimination TRANSITIONS invalidées au cycle précédent

Pour chaque TRANSITION j dans LISTE_VALIDEES faire

     Cas Vj dans

          ( 0 ) : Supprimer j dans LISTE_VALIDEE

          ( 1 ) : Si (Hj = 0) Alors     (* TRANSITION j non inhibée *)

                       Evaluer RECEPTIVITE associée à TRANSITION j

                       Si (TRANSITION j passante) Alors

                            Pour chaque ETAPE i amont à j faire

                                 Set Di
                            Fpour

                            Pour chaque ETAPE i aval à j faire

                                 Cas Xi dans

                                      ( 0 ) : Set Ai

                                      ( 1 ) : Si (Ai = 0 ) Alors

                                              Set Ai
                                              Placer i dans LISTE_ACTIVES
                                              Fsi

                                 Fcas
                            Fpour
                       Fsi
                  Fsi
     Fcas
Fpour
```

```
-----------   Mise a jour variables Xi' ETAPES marquées
! PHASE 2 !   Validation TRANSITIONS aval ETAPES activées
-----------   Invalidation TRANSITIONS aval ETAPES désactivées

Si (au moins une TRANSITION franchie en PHASE 1) Alors

    Pour chaque ETAPE i dans LISTE_ACTIVES faire

        Cas (Xi',Ai,Di) dans

            ( 0,1,0 ) : (* ETAPE activée *)

                Set Xi'
                Reset Ai

                Pour chaque TRANSITION j aval à i faire

                    CPTj := CPTj - 1

                    Si (CPTj = 0) Alors

                        Set Vj
                        Placer j dans LISTE_VALIDEES
                    Fsi
                Fpour

            ( 1,0,1 ) : (* ETAPE désactivée *)

                Reset Xi'
                Reset Di

                Pour chaque TRANSITION j aval à i faire

                    Reset Vj
                    CPTj := CPTj + 1
                Fpour

                Supprimer i dans LISTE_ACTIVES

            ( 1,1,0 ) : (* ETAPE suractivée *)

                Reset Ai

            ( 1,1,1 ) : (* ETAPE réactivée *)

                Reset Ai
                Reset Di

            ( 1,0,0 ) : (* ETAPE déjà active *)

                -------

        Fcas
    Fpour
Fsi
```

```
-----------   Recopie variables Xi' dans variables Xi
! PHASE 3 !   Exécution des ACTIONS associées aux ETAPES actives
-----------   (dans l'ordre croissant de leurs numéros d'ordre)

LISTE_ACTIONS := vide

Pour chaque ETAPE i faire

     Xi := Xi'

     Si (Xi = 1) Alors

         Placer i dans LISTE_ACTIONS
     Fsi
Fpour

Pour chaque ETAPE i dans LISTE_ACTIONS faire

     Exécuter ACTION associée à ETAPE i
Fpour
```

B I B L I O G R A P H I E .

Documents de base.

ADEPA Guide d'Etude des Modes de Marches et d'Arrêt.

M. BLANCHARD Comprendre, maitrîser et appliquer le Grafcet. Editions Cepadues - octobre 1979.

J.C. BOSSY, P. BRARD, P. FAUGERES, C. MERLAUD Le Grafcet, sa pratique et ses applications. Editions Casteilla - décembre 1984.

GREPA Le grafcet, de nouveaux concepts. Editions Cepadues - décembre 1985.

S. THELLIEZ J.M. TOULOTTE Grafcet et logique industrielle programmée. Editions Eyrolles - octobre 1980.

Normes et actes de congrès.

NF C 03-190 Diagramme fonctionnel Grafcet pour la description des systèmes logiques de commande. Union Technique de l'Electricité - novembre 1981.

AFCET - ADEPA Automatismes logiques : recherche et applications industrielles. Paris - décembre 1976.

Articles de presse.

GS1 AFCET Grafcet : interprétations algébriques et algorithmiques et temporisations. Le Nouvel Automatisme - septembre 1983.

Thèses et mémoires.

M. MOALLA Spécification et conception sûre d'automatismes-discrets complexes, basées sur l'utilisation du Grafcet et des réseaux de Petri. Thèse de doctorat d'état. Grenoble juillet 1981.

R. JAY Réalisation du module "commande séquentielle" dans le système MASK16. Mémoire d'ingénieur C.N.A.M. Grenoble - décembre 1979.