PARTIE A : ELEMENTS DE BASE

I . PRESENTATION DU CONTEXTE

1 - NECESSITE D'UN LANGAGE COMMUN

Cet ouvrage est essentiellement consacré à la conception des machines de production. La réalisation d'un équipement industriel fait intervenir plusieurs services d'une même entreprise (méthodes, atelier de production, maintenance, service commercial, ergonomie...) et souvent plusieurs entreprises (le client, l'intégrateur, les sous-traitants...) Or, chacun a son expérience, ses habitudes, son savoir-faire et surtout son propre jargon.

En voici une illustration très simple. Pour utiliser un vocabulaire courant, on peut dire que la figure A-1 représente un *objet*. Un concepteur y voit cependant un *cylindre* en référence à sa géométrie, ou un *axe*, un *manchon* ou encore un *arbre* en référence à sa fonction. Pour un fabricant, c'est une *pièce*. Dans le domaine de la gestion de production, on parle d'*article* (élément de base d'une nomenclature). C'est pourtant le même *objet* pour tous : des malentendus peuvent donc nuire au bon déroulement d'un projet.

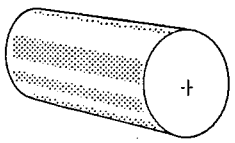


figure A-1

De plus, vers les années 70, la complexité croissante des automatismes industriels nécessite de nouveaux outils de modélisation. Sous la direction de Michel Blanchard, le groupe *Systèmes Logiques* de l'AFCET a répondu à ce besoin en créant en 1975 le Grafcet, dont le principe et le graphisme sont inspirés des Réseaux de Petri (1962, par Carl Adam Petri, mathématicien allemand). Dès alors, Jean-Paul Frachet a constitué le groupe de travail *Equipements de Production Automatisée* au sein de l'ADEPA afin de développer le Grafcet et de le promouvoir dans l'enseignement et l'industrie. Une démarche similaire a eu lieu simultanément en Allemagne pour aboutir à la norme DIN 40719.

A l'origine, le Grafcet est donc un outil de communication qui s'attache exclusivement à la description fonctionnelle des automatismes.

- Il a été normalisé en France en juin 1982 (NF C 03-190). En 1985, le GREPA a publié de nouveaux concepts.
- Une norme européenne a été consacrée au Grafcet (CEI 848 de 1988) sous la dénomination *diagramme fonctionnel pour systèmes de commande* ou *function chart for control systems*.
- La norme française a été revue suite à l'approbation internationale en donnant lieu à la norme UTE C 03-190 de novembre 1990, puis au complément UTE C 03-191 de juin 1993.
- La version française de septembre 1995 (NF C 03-190) est la plus récente.

Le concept Grafcet a été intégré dans la norme européenne sur les langages de programmation des automates sous la dénomination *SFC : diagramme fonctionnel en séquence* ou *sequential function chart* (CEI 61131-3 de juillet 1993, version française NF EN 61131-3 de novembre 1993).

Avant cette normalisation, les constructeurs avaient déjà proposé leurs propres solutions, très spécifiques, pour permettre la programmation sous représentation Grafcet. Maintenant, tous les automates récents et à venir se réfèrent à la norme CEI 61131-3. Les interfaces utilisateurs et les caractéristiques intrinsèques des automates sont toujours spécifiques à chaque constructeur. Les applications restent incompatibles d'une marque à l'autre.

Décliné en plusieurs niveaux, à la fois outil de modélisation et support pour la programmation, le Grafcet garantit aujourd'hui la transmission correcte des informations entre toutes les parties prenantes d'un projet d'automatisme, de la spécification à l'exploitation et à la maintenance, en passant par la réalisation.

2 - DECOUVERTE DU GRAFCET

Pour les lecteurs qui découvrent l'existence du Grafcet, voici une première approche de cet outil à travers la modélisation progressive d'un exemple simple. On suppose que *le système existe déjà* et qu'on peut donc en *observer sa structure matérielle et aussi son fonctionnement.*
Le système étudié est un cas industriel classique : un cycle de trempe (figure A-2).

2 - 1. Description de la machine

a) Description par une figure

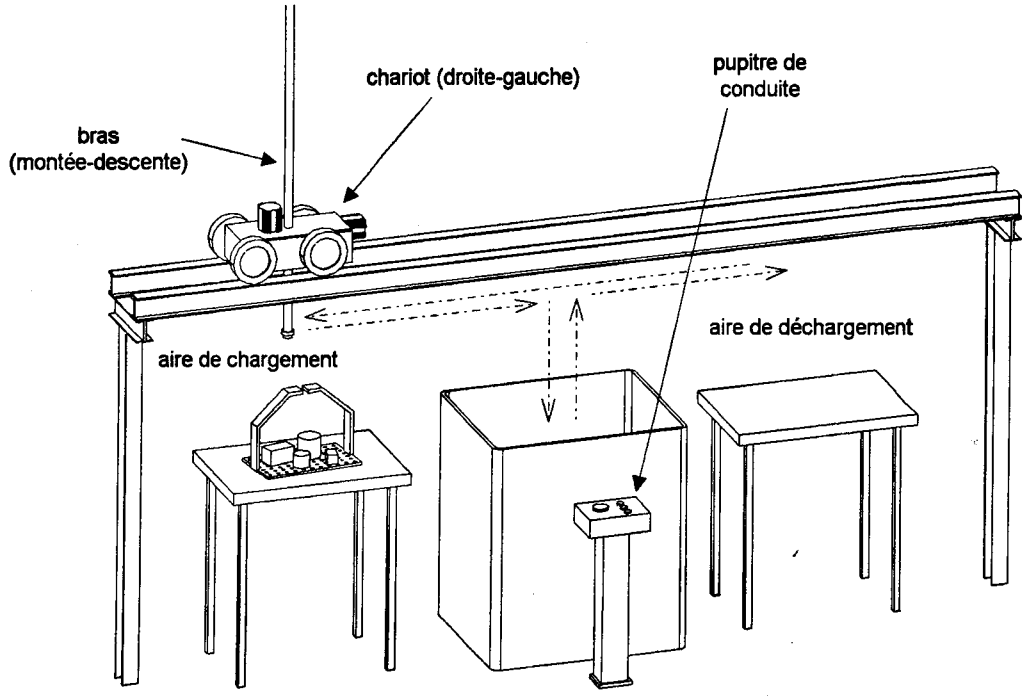


figure A-2

b) Description par un texte

Un opérateur installe un plateau qui contient des pièces à tremper sur le bras d'un chariot mobile. Il appuie ensuite sur le bouton *départ* installé sur un pupitre de conduite. Le chariot se déplace vers la droite jusqu'au bac de trempe. Le bras descend et plonge les pièces dans le liquide qui opère le traitement souhaité. Au bout de 8 minutes, le bras remonte. Puis le chariot se déplace de nouveau vers la droite et s'immobilise au-dessus de l'aire de déchargement. Un voyant clignote pour avertir l'opérateur que le cycle de trempe est terminé (l'opérateur peut avoir quitté ce poste pour effectuer un autre travail). Dès que l'opérateur

a déposé le plateau de pièces traitées, il appuie sur le même bouton *départ*, ce qui provoque cette fois le retour du chariot en position de chargement. Un nouveau cycle peut être demandé.

c) Description par une liste succinte

- L'opérateur installe le plateau et appuie sur le bouton *départ*.
- Le chariot se déplace jusqu'au bac.
- Le bras descend.
- Au bout de 8 minutes, le bras remonte.
- Le chariot se déplace jusqu'à l'aire de déchargement.
- Un voyant clignote, l'opérateur enlève le plateau et appuie sur le bouton *départ*.
- Le chariot revient en position de chargement.

Conclusion

Aucun des moyens de représentation ci-dessus ne suffit à lui seul à décrire le système.

- La figure ne décrit que très globalement l'installation. Elle peut être développée en un schéma technologique pour une connaissance plus fine des organes mis en oeuvre. Les dessins techniques de même que les schémas électro-pneumatiques et les nomenclatures apportent tous les détails quant à sa structure matérielle. Mais les mouvements ne peuvent être que devinés, et son fonctionnement n'est exprimé d'aucune manière.
- Le texte précise le fonctionnement du système, mais sa compréhension repose entièrement sur la figure. De plus, il doit être rédigé dans un langage qui ne laisse place à aucun doute. Cette rédaction est un exercice bien plus difficile que ce qu'on peut penser à priori.
- La liste est plus simple à rédiger *tant que l'évolution du système est linéaire*. Mais l'efficacité d'un tel document peut être compromise si plusieurs séquences sont exécutées simultanément, ou si des séquences sont exécutées dans certaines conditions seulement.

2 - 2. Construction intuitive à travers l'exemple

Malgré ses limites, la liste est un bon moyen de présenter simplement l'aspect dynamique du système. Mais pour que ce type de représentation puisse être généralisé, certaines règles doivent être observées. Ce chapitre permet de les découvrir progressivement.

a) Liste détaillée

La liste précédente, assez succinte, peut être complétée pour que tout ce qui se passe y figure d'une manière explicite.

(Le lecteur est vivement invité à rédiger sa propre liste et ensuite seulement à la comparer à celle proposée ci-dessous : on remarquera la difficulté à bien scinder, et quelquefois à combiner, les différentes « choses » qui se produisent, même dans un cycle aussi simple que celui-ci).

- L'opérateur installe un plateau.
- L'opérateur appuie sur le bouton *départ*.
- Le chariot se déplace vers la droite.
- Le chariot arrive au-dessus du bac.
- Le bras descend.
- Le bras arrive en position basse.
- Une temporisation de 8 minutes commence.
- Les 8 minutes sont écoulées.
- Le bras remonte.
- Le bras arrive en position haute.
- Le chariot se déplace vers la droite.
- Le chariot arrive au-dessus de l'aire de déchargement.

- Un voyant clignote, l'opérateur enlève le plateau.
- L'opérateur appui sur le bouton *départ.*
- Le chariot se déplace vers la gauche.
- Le chariot arrive au-dessus de l'aire de chargement.

b) Frontière entre opérateur et système

Les gestes de l'opérateur ne doivent pas tous être considérés de la même manière. Par exemple, la proposition *l'opérateur installe un plateau* n'a en soi aucune influence sur le fonctionnement du système alors que la proposition *l'opérateur appuie sur le bouton départ* constitue une cause de son évolution.
Puisque la liste représente le fonctionnement du système, on ne prend pas en compte les gestes de l'opérateur qui n'ont pas d'influence sur son évolution : dans cet exemple, il s'agit de la mise en place et de l'enlèvement du plateau.

c) Distinction entre action et réceptivité

L'étude porte sur la machine et non sur l'opérateur. Par conséquent, une action est l'effet d'un ordre émis par la machine et non par l'opérateur : le chariot se déplace *parce que* le système en a donné l'ordre (il s'agit bien d'une action produite par le système), alors que le fait d'appuyer sur le bouton départ n'est *pas* une action du point de vue du système (bien au contraire, il s'agit d'une information qu'il reçoit de son environnement) .
Une telle information constitue une condition d'évolution. Elle permet au système de changer d'état (exemple : *si* le chariot arrive au-dessus du bac, *alors* il s'arrête et le bras descend). Du fait que les conditions d'évolution peuvent être très variées, le terme générique de *réceptivité* permet de les désigner quelle que soit leur nature.

d) Liste complète

On propose, provisoirement, de distinguer les actions et les réceptivités par une typographie différenciée (par exemple : actions en gras, réceptivités en italiques fins).

- *L'opérateur appuie sur le bouton départ.*
- **Le chariot se déplace vers la droite.**
- *Le chariot arrive au-dessus du bac.*
- **Le bras descend.**
- *Le bras arrive en position basse.*
- **Une temporisation de 8 minutes commence.**
- *Les 8 minutes sont écoulées.*
- **Le bras remonte.**
- *Le bras arrive en position haute.*
- **Le chariot se déplace vers la droite.**
- *Le chariot arrive au-dessus de l'aire de déchargement.*
- **Un voyant clignote.**
- *L'opérateur appuie sur le bouton départ.*
- **Le chariot se déplace vers la gauche.**
- *Le chariot arrive au-dessus de l'aire de chargement.*

Le point essentiel est de constater l'alternance entre les actions et les réceptivités.

e) Apport d'éléments graphiques et d'un vocabulaire approprié

• Les actions sont encadrées et les tirets qui les précèdent sont supprimés. Par contre, les tirets en face des réceptivités sont renforcés et appelés *transitions*. Le texte peut être condensé (figure A-3-a).

• A chaque action est associée une *étape*, représentée par un carré. Les étapes peuvent être numérotées afin de faciliter leur repérage et elles sont reliées entre-elles par des *liaisons orientées* (figure A-3-b).

L'alternance entre les actions et les réceptivités se traduit par l'alternance entre les étapes et les transitions. Cette caractéristique est en partie à l'origine du mot Grafcet (GRAphe Fonctionnel de Commande Etapes-Transitions).

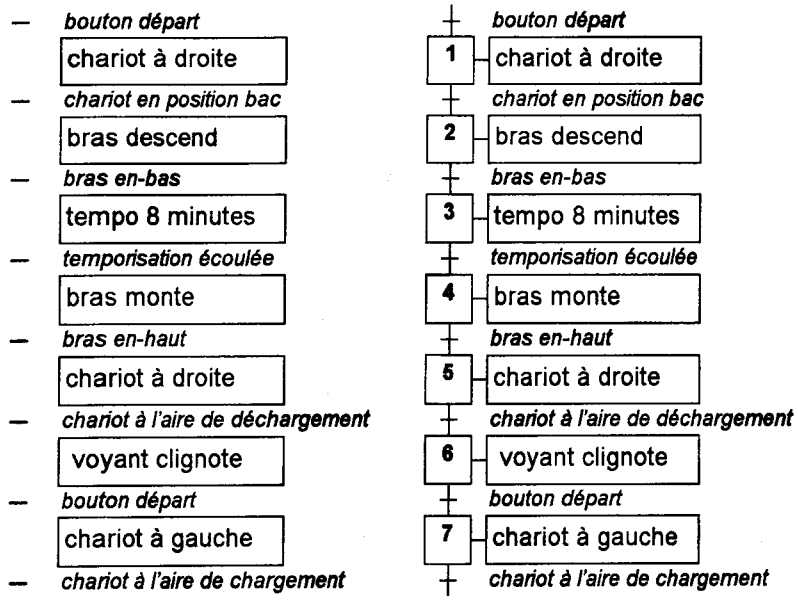


figure A-3

La figure A-3-b est cependant incomplète. En particulier, elle n'exprime pas le fait que les séquences peuvent se succéder les unes aux autres, et suggère *implicitement* le moment du début de la séquence (en-haut de la figure), ce qui est imprécis. Une analyse plus détaillée permettra de compléter ce diagramme afin d'aboutir à un grafcet complet.

f) Analyse de l'évolution du grafcet

L'action qui est produite à un instant donné est celle qui est associée à l'étape dite *active*. Le changement d'état du système est donc représenté par le changement de situation du grafcet, la situation d'un grafcet caractérisant l'ensemble des étapes qui sont actives à l'instant considéré. Ainsi, le passage d'une étape vers l'étape suivante se fait par franchissement de la transition qui les sépare.

En fait , *une transition est franchie lorsque l'étape qui la précède est active et que la réceptivité qui lui est associée est vraie*.

Par conséquent, le grafcet de la figure A-3-b doit comporter une étape *au-dessus* de la première transition pour que celle-ci puisse être franchie.

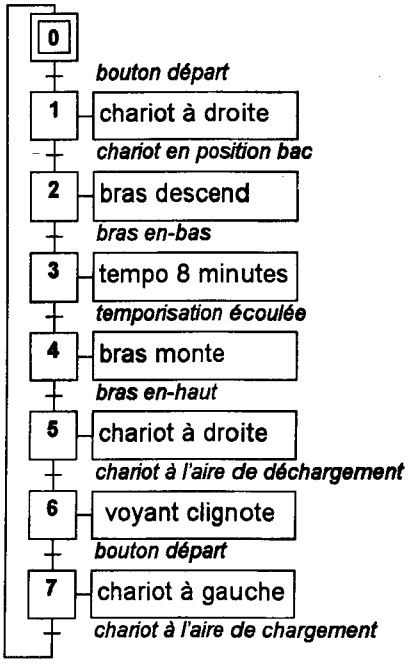


figure A-4

Par ailleurs, si aucune étape n'est active à la mise sous tension, aucune transition ne peut être franchie : aucune étape ne peut être activée et à fortiori, aucune action ne peut être produite. L'activation d'une étape supplémentaire (0 sur la figure A-4) peut être obtenue automatiquement dès la mise sous tension : il s'agit d'une étape initiale, représentée par un carré double. La possibilité de répétition du cycle est symbolisée par la liaison entre les étapes 7 et 0. Les règles d'évolution du Grafcet sont détaillées au § II.1 - 1.

g) Notion d'automatismes combinatoires

Soit une action produite exclusivement par une combinaison de variables : par exemple une sonnerie déclenchée par l'un des deux boutons poussoirs situés aux deux entrées d'une maison. Le relâchement du bouton arrête la sonnerie. L'équation logique de la sonnerie est :

sonnerie = bouton 1 ***ou*** bouton 2

On parle d'automatismes combinatoires car la sonnerie est produite par *combinaison* de variables.

h) Notion d'automatismes séquentiels

- ***Au niveau des actions***

Dans l'exemple du poste de trempe, l'action *chariot à droite* ne dépend pas seulement des variables, mais dépend également de l'état courant du système : *chariot à droite* est produit par la variable *bouton départ* si le chariot se trouve en situation initiale (étape 0) ou bien par la variable *bras en-haut* si le bras vient de monter (étape 4). Cette proposition logique ne décrit toutefois que le début de l'action, ce qui peut s'écrire d'une manière symbolique :

début chariot à droite = [bouton départ ***et*** étape 0**] *ou* [**bras en-haut ***et*** étape 4**]**

L'action se poursuit jusqu'à satisfaction de :

fin de chariot à droite = [chariot en position bac ***et*** étape 1**] *ou* [**chariot à l'aire de déchargement ***et*** étape 5**]**

En fait, l'action dure tout le temps d'activation de l'étape 1 *ou bien* tout le temps d'activation de l'étape 5. L'équation de *chariot à droite* peut donc s'écrire plus simplement :

chariot à droite = [étape 1**] *ou* [**étape 5**]**

- ***Au niveau des réceptivités***

Toujours dans l'exemple du poste de trempe, on constate par ailleurs que le bouton *départ* n'a pas le même effet selon l'état du système.

- Si le chariot est en position de chargement, le fait d'appuyer sur le bouton *départ* provoque le déplacement du chariot vers la droite.
- S'il est en position de déchargement et qu'on appuie sur ce *même* bouton, le chariot se déplace vers la gauche.
- Dans toutes les autres situations, si le bras est au fond du bac par exemple, l'appui sur le bouton en question ne provoque *aucun* effet.

On parle d'automatismes séquentiels car les situations sont organisées en séquences : les actions sont le résultat de situations successives et le rôle des réceptivités dépend de la situation courante.

i) Ecriture des équations des sorties

On appelle *équations des sorties* les propositions logiques qui définissent la production des actions.
Dans l'exemple de la figure A-5, l'action A1 est produite si l'étape 2 *ou* si l'étape 3 est active. On écrit donc :

A1 = [étape 2**] *ou* [**étape 3**]**

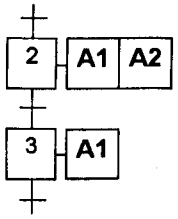


figure A-5

Il est incorrect par contre de combiner toutes les actions qui sont produites par une même étape. Exemple :

étape 2 = [action A1] et [action A2]

- En effet, les fonctions logiques permettent de combiner des variables d'entrée mais ne sont pas destinées à effectuer des affectations multiples.

- Par ailleurs, le signe = peut prêter à confusion car il n'indique pas explicitement le sens de lecture d'une équation logique, qui, comparativement à l'algèbre des nombres réels, est significatif. De ce fait, on préfère souvent le signe ← qui est sans ambiguïté :

A1 ← [étape 2] *ou* [étape 3]

Cette expression montre très clairement que l'activation des étapes 2 ou 3 ***entraîne*** la production de l'action A1. Par convention, l'écriture d'une telle équation se fait plus rarement de la gauche vers la droite.

3 - INDICATIONS POUR LA MISE EN OEUVRE

Ce paragraphe situe succintement la place du Grafcet dans l'univers complexe que constitue un système automatisé de production afin que le lecteur puisse ensuite aborder les manuels de programmation des automates.

3 - 1. Architecture générale d'un système automatisé

Le cycle de trempe décrit plus haut ne peut se dérouler d'une manière automatique que si la machine :

- comporte un certain nombre de constituants,
- gére elle-même les interactions des constituants,
- dialogue avec son environnement.

On appelle cet ensemble *système de production automatisée* (figure A-6).

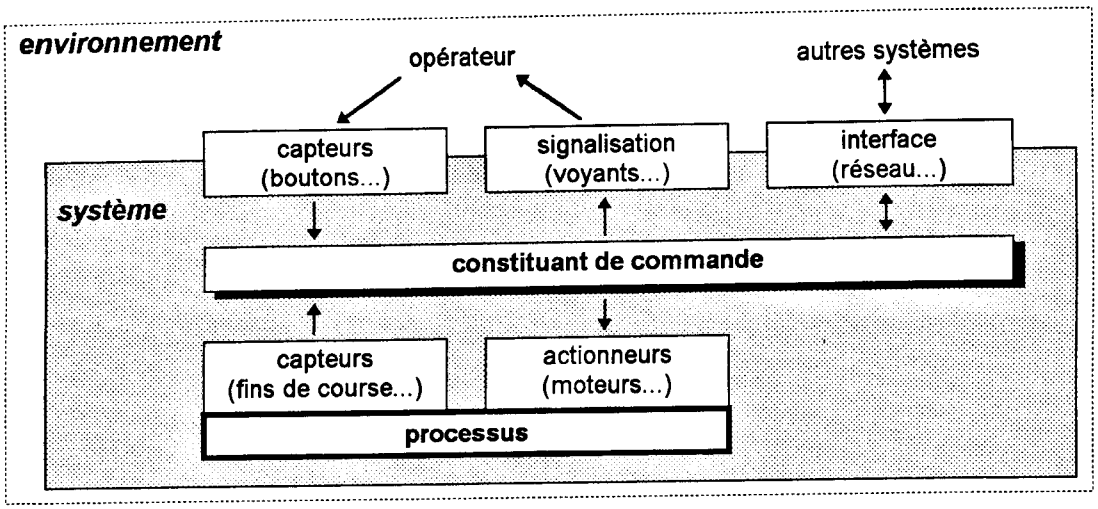


figure A-6

3 - 2. Technologie du constituant de commande

a) Solutions câblées

Les fonctions de commande peuvent être réalisées par des composants logiques élémentaires de type électrique, pneumatique, électronique ou plus rarement fluidique. Ces

composants sont à relier entre-eux par des liaisons physiques (fils, tubes, circuit imprimé) selon le problème à résoudre. Chaque problème nécessite donc un schéma spécifique. Celui-ci peut être élaboré par des méthodes d'analyse diverses : tableaux de Karnaugh, méthode en cascade, etc. La reconfiguration du coffret de commande est une opération longue et délicate si des modifications ou des ajouts s'avèrent nécessaires. Composants évolués, les séquenceurs permettent de simplifier l'étude et le câblage des systèmes séquentiels en permettant en outre une meilleure évolutivité des machines. Ils sont disponibles dans les différentes technologies. Les solutions câblées ne représentent plus qu'une part infime des réalisations mais restent performantes dans des domaines particuliers : les temps de réponse sont très courts, et en ambiance explosible la sécurité est optimale pour la technologie pneumatique.

b) Solutions programmées

La souplesse de la programmation permet d'adapter très rapidement le comportement des machines aux contraintes changeantes de la production, même en phase d'exploitation. La profusion des fonctions disponibles grâce à l'évolution de la micro-informatique permet en outre d'améliorer considérablement la qualité générale des machines : dialogues évolués avec l'opérateur, horodatage des événements, dialogues avec les parties commandes d'autres machines et prise en charge du positionnement précis d'un axe de déplacement ne sont que quelques exemples. Il existe 3 familles de solutions programmées, adaptées chacune à un type particulier d'application : le micro-contrôleur, l'automate programmable et le micro-ordinateur.

- ***Le micro-contrôleur***

D'un prix d'achat très attractif, il nécessite cependant un temps (donc un coût) de développement assez conséquent. Son intérêt réside de ce fait dans les applications de série. Sa programmation est proche de celle du micro-processeur.

- ***L'automate programmable***

D'un prix plus élevé, l'automate programmable est cependant d'un rapport performance-prix de plus en plus intéressant. Contrairement au micro-contrôleur, il ne requiert pas de développement matériel supplémentaire. Sa programmation ne nécessite qu'une formation d'environ une semaine pour un technicien. Par ailleurs, il est robuste et est conçu pour offrir une vaste gamme d'interfaces pour communiquer avec son environnement : capteurs et actionneurs, dialogues opérateurs... Pour ces raisons, il est bien adapté au milieu industriel mais est également très utilisé dans la gestion technique des bâtiments (ascenseurs, climatisation, sécurités,etc.) ou des voies de communication (la signalisation active des autoroutes par exemple est souvent pilotée par des automates industriels).

- ***Le micro-ordinateur***

Il offre l'avantage de pouvoir traiter et de stocker de grandes quantités d'informations avec des outils standards (plate-forme Windows par exemple). Mais sa constitution ne lui permet pas de survivre dans l'ambiance agressive d'un atelier (vibrations, pollution électro-magnétique, etc.). Des modèles industriels sont disponibles, mais leur prix reste encore élevé.

- ***Solutions hybrides***

L'interpénétration des domaines de l'automatique et de l'informatique conduit aux solutions qui combinent automate et micro-ordinateur :

• soit le micro-ordinateur est complété par des cartes d'entrées-sorties industrielles et d'une carte pour automatismes séquentiels (un automate en fait),
• soit c'est l'automate qui est complété par une unité centrale de type PC ou tout autre élément issu du monde informatique : carte graphique, carte réseau Ethernet, etc.
• certains constructeurs proposent depuis peu des solutions qui intègrent des fonctions d'automatismes et des fonctions d'informatique sans plus faire référence ni à l'automate, ni au micro-ordinateur.

3 - 3. Structure matérielle d'un automate programmable

Notre domaine d'étude étant limité aux systèmes industriels, c'est l'automate programmable qui retiendra plus particulièrement notre attention. *Voir illustrations en fin de chapitre.*

a) Les entrées-sorties

Revenons au poste de trempe décrit au chapitre précédent. L'expression des actions et des réceptivités du grafcet se limitait volontairement à des observations très superficielles. Afin de pouvoir piloter le système d'une manière automatique, il est nécessaire d'une part de *recueillir les informations* qui constituent les réceptivités et d'autre part, de *transmettre des ordres* qui permettront aux moteurs, voyants... de fonctionner.

Exemples : la proposition *chariot présent à l'aire de déchargement* est une information issue d'un capteur de position installé sur le rail ; *bras descend* doit d'abord être un ordre dont l'effet est la rotation du moteur adéquat dans le sens adéquat.

L'ensemble des informations constitue le vecteur des *entrées*. L'ensemble des ordres est le vecteur des *sorties*.

Les informations transitent par le coupleur d'entrée dont le rôle est d'assurer l'adaptation des signaux, le filtrage des parasites ainsi que l'isolement de l'automate du milieu extérieur grâce à un circuit optoélectronique.

Entre deux mises à jour (voir § 3 - 4), les ordres fournis par le programme sont mémorisés par le coupleur de sortie qui comporte également un circuit d'isolement optoélectronique ainsi qu'un circuit d'amplification des signaux ou des relais électromagnétiques.

b) Les variables internes

Le cycle de trempe a, de plus, fait intervenir une temporisation, qui est une variable interne.
Les variables internes sont à la fois entrées *et* sorties selon le besoin mais sont stockées dans l'unité centrale. Elles ne transitent donc jamais par les coupleurs d'entrées-sorties.
Lors du démarrage de la temporisation, la variable interne associée peut être *considérée* comme une sortie. Lors du test de son état, elle est *considérée* comme une entrée.
Les variables internes sont également : les compteurs, les registres, les files d'attente, etc.

c) Des périphéries supplémentaires

Les entrées-sorties et variables internes sont souvent complétées par des composants facultatifs spécialisés :
• des coupleurs de communication : dialogues avec d'autres parties commandes, avec des entrées-sorties déportées, avec un superviseur, etc.

• des cartes intelligentes qui comportent leur propre micro-processeur : comptage rapide, fonctions de régulation, imagerie, gestion d'axes numériques, etc.
• des interfaces pour disques durs, imprimantes, etc.

d) L'unité centrale

L'unité centrale est constituée de différents éléments permettant de répondre à différentes fonctions :

• *le jeu d'instructions* : c'est l'ensemble des instructions disponibles,
• *la mémoire programme utilisateur* (RAM) : elle contient le programme d'application écrit par l'utilisateur ; ce programme peut être transféré en mémoire morte reprogrammable (EEPROM par exemple) après sa mise au point,
• *la mémoire programme système* (ROM) : le programme qu'elle contient est implémenté par le constructeur et n'est pas accessible à l'utilisateur ; ce programme définit comment fonctionne l'automate,
• *la mémoire données utilisateur* : ce sont des adresses disponibles pour l'utilisateur afin qu'il puisse stocker des informations liées à la gestion du processus,
• *la mémoire données système* : adresses en liaison avec le fonctionnement interne de l'automate ; certaines sont à lecture seule pour l'aide au diagnostic, d'autres en lecture-écriture pour la configuration de l'automate,
• *des registres spécialisés* : tempos, compteurs, constantes... à paramétrer et à utiliser dans le programme d'application,
• *des registres temporaires* : le résultat logique (bit) et les accumulateurs (mots), manipulés par le programme système et quelquefois par l'utilisateur pour assurer des fonctions évoluées,
• *l'unité arithmétique et logique, A.L.U.* : elle permet au programme de réaliser des calculs arithmétiques et des combinaisons logiques,
• *les mémoires images des entrées et des sorties* : cette zone mémoire est le *quasi*-reflet des états des entrées et des sorties (d'où le nom d'*image*) ; ces mémoires sont accessibles à la fois par le programme système et par le programme utilisateur (voir § 3 - 4),
• *l'horloge interne* : elle cadence l'exécution du programme.

D'autres fonctions peuvent être disponibles comme le stockage des commentaires ou la gestion horaire des événements.

e) Les bus

• Le *bus d'entrées-sorties*, aussi appelé bus fond de panier, relie les cartes (ou coupleurs) d'entrées-sorties à l'unité centrale ; des standards, comme VME, permettent à des matériels spécialisés fournis par différents constructeurs de cohabiter dans le même châssis.
• Le *bus système* est interne à l'unité centrale : il permet au processeur d'accéder aux différentes ressources.

f) La console de programmation

La console, bien que séparée physiquement de l'automate, doit être considérée comme un de ses éléments constitutifs du fait de l'importance des fonctions qu'elle assure :
• en phase de programmation,
• en phase de réglage des paramètres,
• en phase de diagnostic et de mise au point grâce aux fonctions primordiales de visualisation et de forçage dynamique des variables, de pose de points d'arrêt, etc.

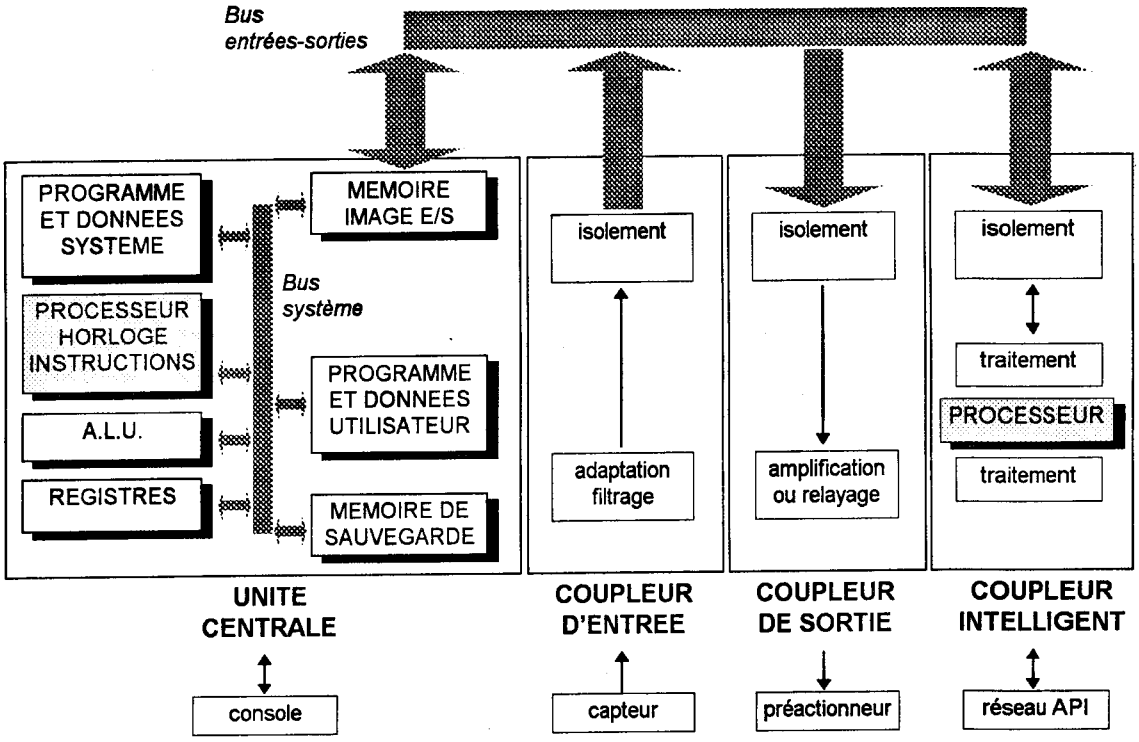
• en phase de constitution de la documentation et enfin en phase d'exploitation pour la lecture ou la modification de valeurs : durées de temporisations, seuils de comptage, paramètres divers, etc.

La console peut être du type PC. Dans ce cas, elle contient le logiciel de programmation complet, ce qui permet d'accéder à toutes les fonctionnalités. Elle sera mise en oeuvre lors des phases de programmation et de maintenance par un personnel qualifié. Mieux adaptées aux opérateurs d'exploitation, il existe des mini-consoles de type « pocket » qui permettent uniquement l'accès aux fonctions non critiques.

Attention à ne pas confondre :

• *logiciel de programmation* : il réside dans la console et propose des outils pour la configuration de l'automate, la définition de la notation symbolique, le réglage des paramètres, la saisie et la modification du programme, le diagnostic...

• *langage de programmation* : il est constitué du jeu d'instructions et de sa syntaxe et est uniquement exploitable par l'automate.



Note : chaque coupleur dispose en général de plusieurs voies.

figure A-7 : exemple de configuration

g) Configuration multi-processeurs

On a vu que les cartes intelligentes disposaient de leur propre micro-processeur pour réaliser des tâches spécialisées, ce qui déleste d'autant le processeur de l'unité centrale. De plus, les modèles moyenne et haute gamme permettent en général l'installation de plusieurs unités centrales dans le même châssis pour le traitement de tâches en parallèle.

3 - 4. Principe de fonctionnement de l'automate programmable

a) Rôle du programme

La *mémoire programme* utilisateur contient la liste des instructions saisie par le programmeur. Ce programme définit le comportement du système :

- en fonction des sollicitations extérieures,
- en fonction des états des variables internes.

b) Notion de scrutation

Afin de permettre des réactions en temps réel (au temps de réponse près), le déroulement du programme repose sur le principe suivant :

- *les instructions sont automatiquement exécutées les unes après les autres*
  - si une condition logique est vraie, alors l'instruction suivante est exécutée et le déroulement du programme se poursuit
  - si une condition logique est fausse, alors l'instruction suivante n'est pas exécutée, sauf si elle est inconditionnelle, et le déroulement du programme se poursuit de toute façon

- *lorsque toutes les instructions du programme ont été lues, le programme se déroule de nouveau depuis le début*
  des instructions de saut permettent de structurer l'application et d'organiser le programme afin d'en accélérer l'exécution

- *cette relecture cyclique est ininterrompue, on parle de cycle de scrutation*
  - ce mode de fonctionnement est celui par défaut de tous les automates ; certains modèles autorisent une scrutation périodique (la période doit évidemment être plus longue que la durée normale du cycle)
  - le déroulement normal du programme peut être brièvement interrompu si des tâches prioritaires s'avèrent nécessaires (cette interruption peut être générée soit par occurrence d'un événement, soit périodiquement), après traitement prioritaire le programme est repris à l'endroit où il a été interrompu

c) Durée de cycle

Il est important de se rendre compte des ordres de grandeur des durées mises en jeu dans le déroulement du programme :

- la durée d'exécution d'une instruction se chiffre en *nano* ou *microsecondes*
- selon la longueur du programme, une scrutation a une durée de l'ordre de *quelques dizaines de millisecondes*
- le traitement d'une tâche prioritaire ne dépassera pas *quelques millisecondes.*

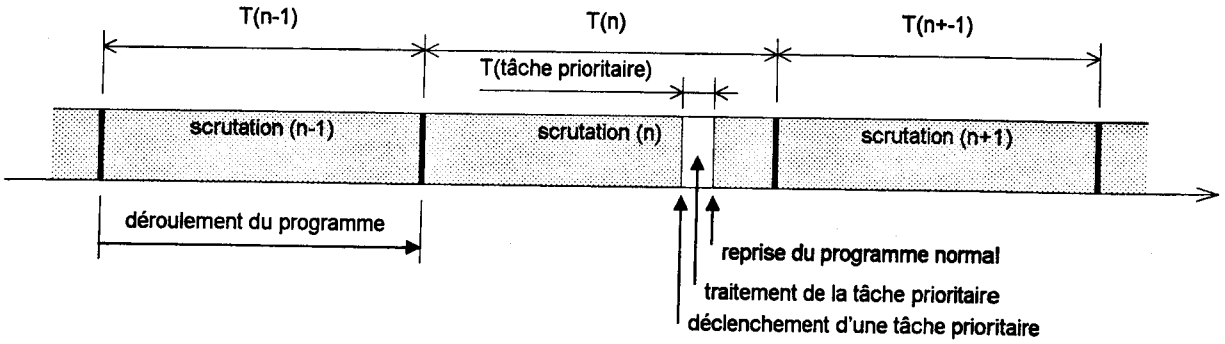


figure A-8

d) Mémoires images des entrées (notées MIE)

L'apparition et la disparition des signaux délivrés par les capteurs ont évidemment lieu indépendamment de l'automate : les états des entrées basculent tout au long du déroulement du programme. Cette instabilité peut générer des aléas de fonctionnement car lorsqu'on développe une application, une entrée est considérée comme étant soit vraie, soit fausse à

un moment donné et non les deux à la fois. Pour la durée d'exécution d'un seul cycle, les états des entrées sont rendus stables par les mémoires images des entrées. Le mécanisme en est le suivant (voir la figure A-9) :

• *en début de scrutation, les états des entrées physiques telles qu'elles sont vues par les coupleurs d'entrées sont mémorisés dans les mémoires-images correspondantes*
- à chaque entrée correspond une mémoire-image (un bit interne en fait)
- ce bit interne conserve son état jusqu'au prochain rafraîchissement, même si l'entrée physique correspondante change d'état pendant la scrutation

• *pendant l'exécution du programme, toute instruction qui requiert l'état d'une entrée provoque en réalité la lecture de l'image de cette entrée*
- en général, le bit interne porte le même nom que l'entrée qu'il représente
- de ce fait, le programmeur distrait peut penser traiter les entrées physiques alors que le programme qu'il écrit traite leurs images

• *les mémoires-images des entrées sont rafraîchies en début de chaque nouvelle scrutation*
- si une entrée physique change d'état, son nouvel état ne sera pris en compte par le programme que lors de la scrutation suivante
- si l'état vrai d'une entrée physique a une durée si brève qu'il apparaît puis disparaît pendant une même scrutation, cette entrée est considérée par le programme comme étant restée fausse.

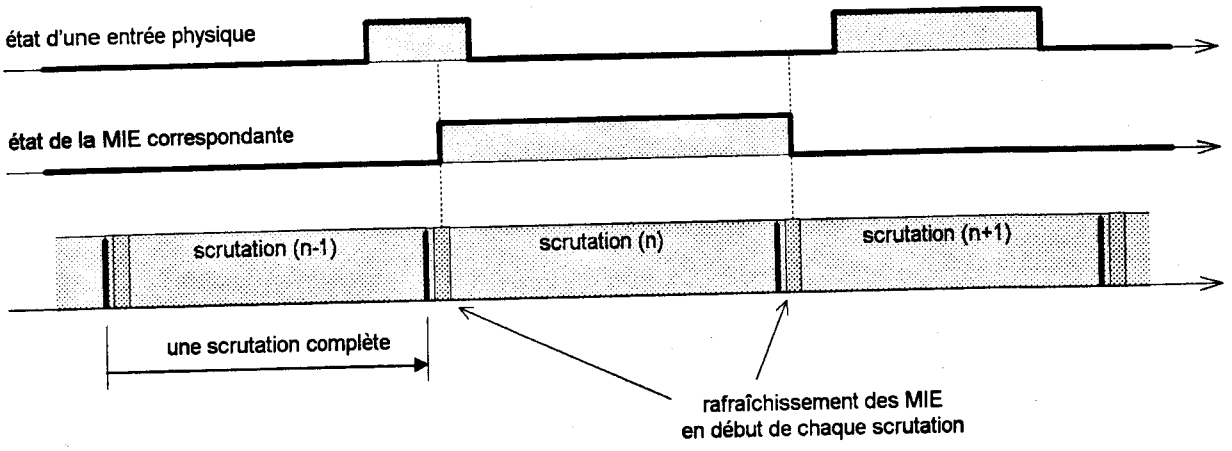


figure A-9

Dans l'exemple de la figure A-9, la première impulsion délivrée par l'entrée est prise en compte par le programme alors que la deuxième ne l'est pas malgré un temps de présence plus long.

e) Mémoires images des sorties (notées MIS)

Un mécanisme similaire met en oeuvre les mémoires images des sorties, interface logique avec les sorties physiques. Là aussi, le programmeur peut penser assigner directement les sorties, alors qu'il assigne en fait des bits internes qui portent simplement les mêmes noms que les sorties.

Le mécanisme est le suivant (voir également la figure A-10) :

• *chaque instruction du programme qui positionne une sortie dans un état déterminé a en réalité comme effet de positionner le bit interne, image de la sortie en question*
la sortie n'est pas affectée par cette instruction, le programme se poursuit

• *chaque nouvelle instruction qui positionne une sortie met la mémoire image correspondante à jour*

l'état de la mémoire-image change d'état au fur et à mesure du déroulement du programme

• *après déroulement complet du programme, les états des mémoires images sont transférés globalement vers les coupleurs de sortie*

l'une des fonctions des coupleurs de sortie est de mémoriser ces états pendant toute la durée du cycle de scrutation suivant, les sorties sont ainsi maintenues dans leur état jusqu'à la fin du prochain cycle

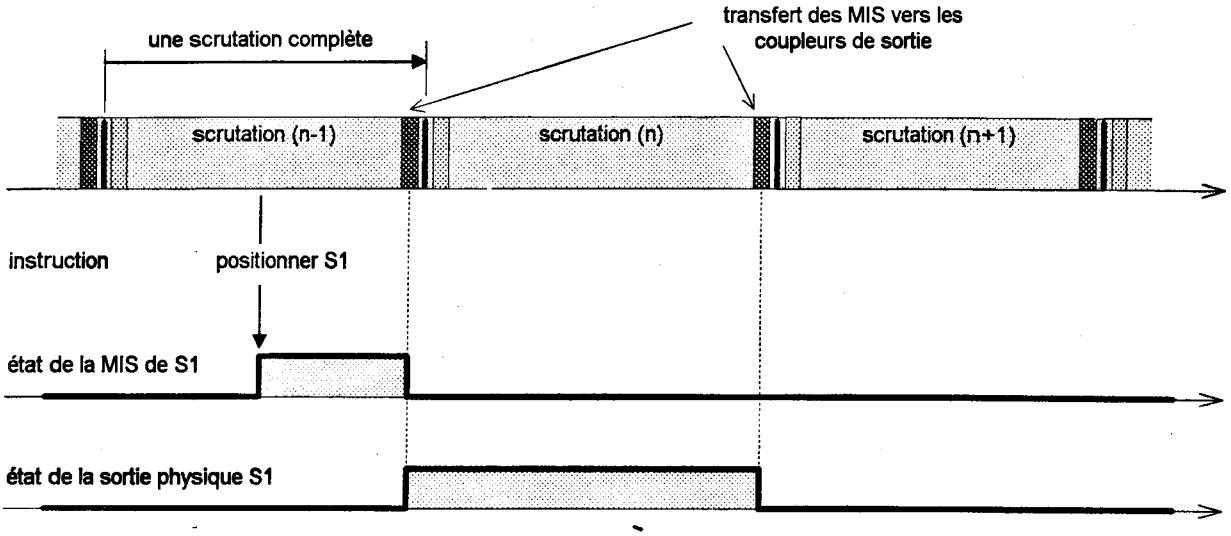


figure A-10

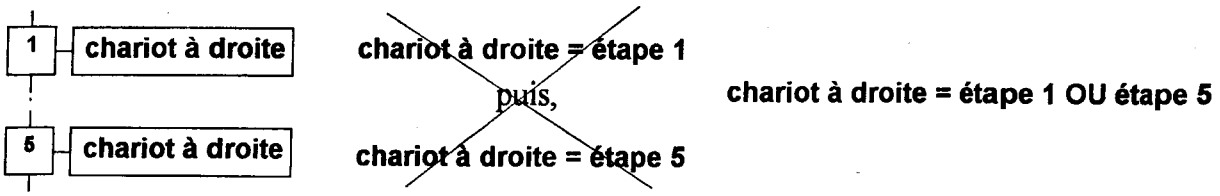
- ***Instructions monostables et bistables***

Dans l'exemple ci-dessus, l'instruction *positionner S1* est de type non mémorisée (monostable). Il existe des instructions avec mémorisation du résultat (bistables) : la mémoire image conserve alors son état pendant toutes les scrutations jusqu'à vérification de l'ordre contradictoire.

- ***Conséquence pour la programmation***

L'étude de ce mécanisme montre qu'une sortie ne doit être positionnée que par une seule équation. Si plusieurs équations se présentaient pendant la scrutation, seule la dernière d'entre-elles serait prise en compte car l'automate traite toutes les équations avant d'effectuer la mise à jour de la MIS.

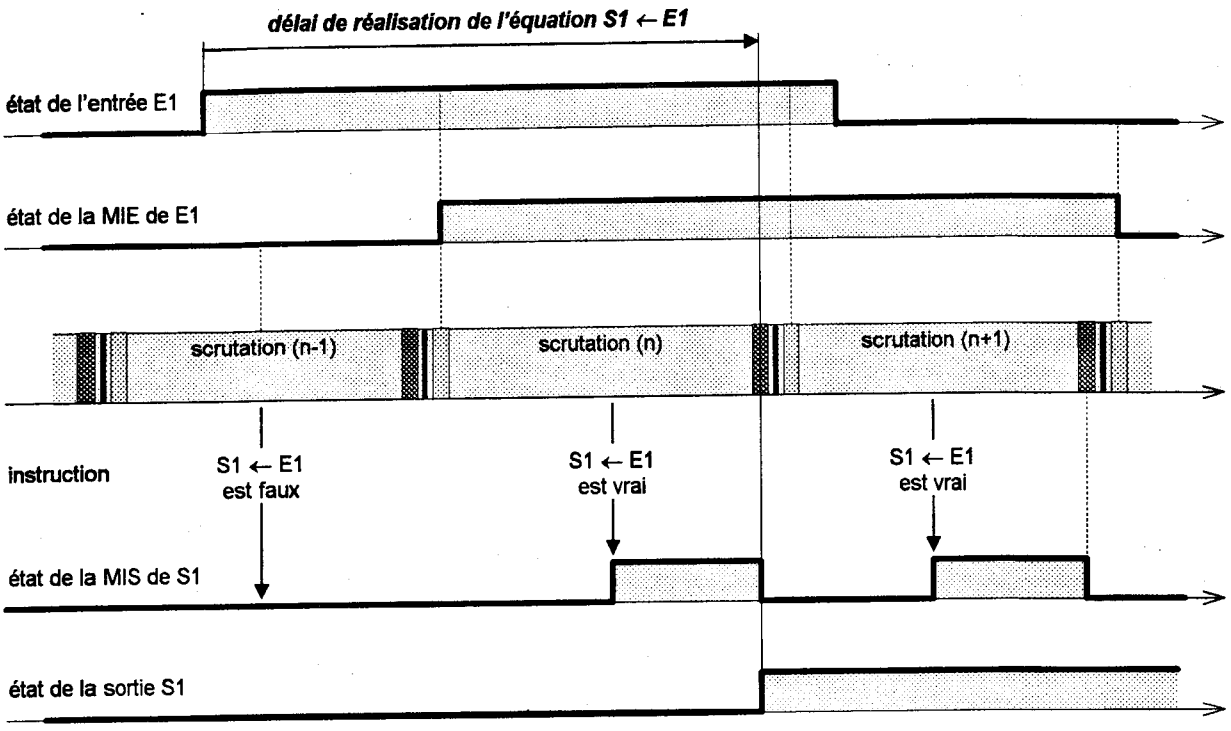
Dans l'exemple suivant, il ne faut pas écrire : ... mais :



Ceci est d'ailleurs conforme à la figure A-5, page 12.

f) Temps de réaction

La figure A-11 illustre le temps de réaction d'une équation combinatoire qui fait intervenir une entrée et une sortie. Ce délai varie de 1 à 2 fois le temps de scrutation, augmenté des temps de traitement des coupleurs, non représentés sur la figure.



Pendant la scrutation (n+2), l'équation S1←E1 sera fausse, la sortie retombera donc à zéro dès la fin de cette scrutation.

figure A-11

3 - 5. La programmation des automates

a) Organisation d'un programme

L'ensemble des instructions saisies par l'utilisateur lors de la programmation s'appelle le *programme*. Selon le choix personnel de chacun, il peut être structuré en *sous-programmes* développés séparément, ce qui facilite sa rédaction et sa mise au point. Ces instructions réalisent des *fonctions* de base : calcul logique sur bit ou sur mot, calcul arithmétique, comparaison, gestion de tableaux... Des *blocs fonctionnels* paramétrables sont habituellement disponibles dans les automates afin de répondre à des fonctions plus évoluées (compteurs, temporisateurs, files d'attente, etc.) voire complexes (régulation, asservissements, communication, etc.)

D'une importance capitale, des *variables* (ou données) sont associées au programme. Elles permettent :

- de paramétrer l'application pour une meilleur flexibilité de la machine,
- d'accéder facilement aux valeurs d'initialisation ou de réglage (sans modifier le programme lui-même),
- de stocker des valeurs issues du processus afin de les analyser *à postériori*.

Exemples :
Le cycle de trempe fait apparaître l'utilisation des *fonctions* logiques ET et OU. Cette application peut être structurée en deux *sous-programmes*, dédiés l'un à l'évolution séquentielle du cycle, et l'autre aux équations des sorties. Le programme comprend également le *bloc fonctionnel standard* de temporisation. La durée de cette temporisation peut être mémorisée dans une *variable*.

b) Les langages de programmation

Le programme doit être saisit dans un langage compréhensible par l'automate. Plusieurs représentations peuvent simultanément être disponibles :

- on distingue d'une part la représentation Grafcet pour l'aspect séquentiel,
- d'autre part, les réceptivités et les actions ainsi que les fonctions qui ne dépendent pas du grafcet sont représentées dans l'un des langages suivants :

langages littéraux :
- liste d'instructions
- littéral structuré

langages graphiques :
- schéma à contacts
- logigramme

Si la représentation Grafcet n'est pas disponible, il est facile de transcrire les grafcets manuscrits dans l'un de ces langages (voir l'exemple au § II.3)

La figure A-12 montre schématiquement la même fonction logique S=(a./b)+c dans les différentes représentations.

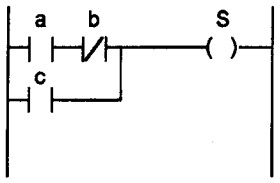
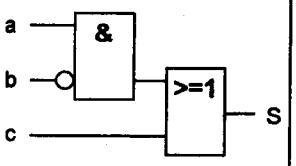
Liste d'instructions	Littéral structuré	Schéma à contacts	Logigramme
(LD a ANDN b) OR c ST S ou O(U - a UN - b) O - c = -S	S := (a ANDNOT b) OR c		
langage puissant, nécessite une certaine expérience	permet les structures habituelles des langages informatiques : IF, CASE, FOR, WHILE, REPEAT	très courant, facile à mettre en oeuvre grâce à sa ressemblance avec les schémas électriques, également appelé LADDER	permet de programmer rapidement des fonctions complexes par paramétrage de blocs fonctionnels
Noms des langages équivalents dans la norme CEI 61131-3 (SFC, issu du Grafcet, est réservé aux fonctions séquentielles)			
IL	**ST**	**LD**	**FBD**

figure A-12