République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE d'ADRAR
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE
DEPARTEMENT DES SCIENCES ET DE LA TECHNOLOGIE

MEMOIRE DE FIN D'ETUDE en vue de l'obtention du diplôme de

MASTER en Commande des Machines Electriques

Thème

Les langages de programmation de l'automate programmable industriel

(application pilotage d'un ascenseur)

Soutenu le : 24 Mai 2017

Présenté par :	**Membres de jury** :
MAALEM ELHACHEMI	**Président** :
TAOUADJI IBRAHIM	Dr : MAKHLOUFI SALIM Univ. d'Adrar
Encadré par :	**Examinateurs**
Dr : MANSOURI SMAIL univ. d'Adrar	Dr : BELLALI BADREDDINE Univ. d'Adrar

Remerciements

Louange à DIEU le très grand et misécordieux, le seul et uniquequi nous a donné la force et le courage pour terminer nos études et élaborer ce travail.
Avant de commencer la présentation de ce travail, Nous profitons de l'occasion pour remercier toutes les personnes qui ont contribué de près ou de loin à laréalisation de ce projet de fin d'études.Nous tenons à exprimer nos vifs remerciements pour mon grand et respectueux, Dr.MANSOURI SMAIL, d'avoir accepté de nos encadrer pour mon projet de fin d'études,ainsi que pour ses précieux conseils,Nous remercions également
Mr.BERKAT FATHI sur Support et assistance.

Nos remerciements vont aussi à tous enseignants et toutes lespersonnes qui nous soutenus jusqu'au bout, et qui ne cessent de nous donner desconseils très importants en signe de reconnaissance.

DEDICACES

Je dédie ce modeste travail

A ma chère mère

Pour son soutien inconditionnel

Ses sacrifices, sa tendresse,

Son amour infini

A la mémoire de mon père

A mes chers amis

A tous qui m'aiment

Ainsi qu'a tous les camarades de ma section

Et tous mes professeurs.

« MAALEM ELHACHEMI »

DEDICACES

C'est avec une grande émotion,
Je dédier ce modeste travail de fin d'étude
au êtres les plus chères :
Mon père et ma mère qui ont fait de moi
ce qui je suis aujourd'hui et qui ont veillé
de guider mes pas durant tout ma vie
par leurs aides, leur grands émotions et
leur sacrifice

A toute Ma famille
A toute mes amis.

« TAOUADJI IBRAHIM »

Liste des figures

Chapitre 01

Chapitre 02

Chapitre 03

Chapitre 04

Liste des tableaux

Sommaire

Chapitre 03

Programmation par Grafcet

Chapitre 04

application sur l'ascenseur

ملـخـص :

الموضوع المعالج في مذكرة التخرج هذه يتمحور حول التحكم في الأنظمة الآلية باستعمال مختلف اللغات , مما يسمح لنا بتقوية معارفنا في هذا المجال. العمل المقدم يحوي وصف الأنظمة الآلية وكيفية التحكم فيها.

تمكنا في هذه المذكرة من نمذجة دفتر شروط لمصعد آلي باستعمال أداة النمذجة المتمن ''قرافسات'' ,هذه الأخيرة تعتبر أداة قوية لنمذجة نظام معقد كالمصعد الآلي.هذا العمل يحتوي أيضا على محاكاة لهذا النظام باستعمال برنامج '' STEP7''

Résumé :

Le sujet traité dans ce mémoire de fin d'étude porte sur Contrôle des systèmes automatisés par utilisation des différentes langages , ce qui nous permis de creuser nos connaissance dans ce domaine. Le travail présenté constitue une description des systèmes automatisés et comment les contrôler. On a pu modéliser un cahier des charges d'un ascenseur a travers un outil de modélisation graphique ''Grafcet'', ce dernier est un outil puissant pour faire la modalisation d'un système complexe tel que l'ascenseur. Ce travail constitue aussi la simulation de ce système par le logiciel ''STEP7''.

Abstract :

The subject covered in this thesis is on Control of automated systems using different langages, which allowed us to deepen our knowledge in this field. The work presented is a description of the automated systems and how to control them. We were able to model a specification of an elevator through a graphical modeling tool " Grafcet ", the latter is a powerful tool to make the moralization of a complex system such as the elevator. This work also constitutes the simulation of this system by the software " STEP7 ".

Introduction générale

Introduction Général :

L'automatisation sert à remplacer un système à logique câblé par un appareil électronique programmable , adapté à l'environnement industriel, qui réalise des fonctions d'automatisme pour assurer la commande de pré actionneur et d'actionneur à partir d'information logique , analogique ou numérique ,et la surveillance des processus industriel.

Dans le domaine de l'automatisation, comme dans d'autres techniques, l'informatique a révolutionné beaucoup de choses. La connexion d'automates à un ordinateur a permis de franchir une étape de plus dans la voie du progrès technologique.

Ce travail est une contribution à l'étude du système homme-machine, dont l'avènement est le postulat que l'homme est d'une certaine manière, contraint de cohabiter avec un partenaire trop discipliné et algorithmique. L'homme et la machine sont côte à côte pour gérer et contrôler les systèmes que l'on utilise dans la vie de chaque jour, surtout des systèmes de grandes complexités, où l'état d'esprit "homme-machine" est bien clair. Les activités de l'homme, montrant son rôle et saplace, se matérialisent soit par l'accomplissement du travail "homme-machine" est bien clair.

On vise à travers cette étude la présentation de l'API et les différents langages de programmation qu'il utilise. On a pris l'ascenseur comme exemple, Notre projet sera composé de quatre chapitres :

Chapitre 01 : on a donné un aperçu général sur les systèmes automatisés et leurs composants, On a aussi présenté l'API et ses composants internes en donnant les critères dans le choix de l'API et en précisant ces points positifs et négatifs.

Chapitre 02 : on a étudié les différents langages de programmation, en définissant toutes les langages avec une explication des leurs composants et de leurs signes en donnant des exemples explicatifs pour illustrer chaque langage.

Chapitre 03 : dans ce chapitre on a étudié le langage GRAFCET, ce choix est défini pour son application d'ascenseur, On a donné une description détaillée de cette langage en donnant les différentes étapes et les règles nécessaires pour son utilisation et en expliquant les types de Grafcet présents.

Chapitre 04 : dans ce chapitre on a appliqué le langage Grafcet pour savoir la gestion de l'ascenseur en utilisant le logiciel STEP7. Enfin, on à réalisé un schéma en utilisant le langage Grafcet qui illustre le mécanisme de gestion de l'ascenseur selon le cahier des charges.

Chapitre 01

Etat de l’art de déférentes générations des automates programmables

1.1. Introduction :

Depuis toujours l'homme est en quête de bien être". Cette réflexion (qui rejoint la notion de besoin) peut paraître bien éloignée d'un cours de sciences industrielles, pourtant c'est la base de l'évolution des sciences en général, et de l'automatisation en particulier. L'homme a commencé par penser, concevoir et réaliser. Lorsqu'il a fallu multiplier le nombre d'objets fabriqués, produire en plus grande quantité, l'automatisation des tâches est alors apparue : remplacer l'homme dans des actions pénibles, délicates ou répétitives.

Dans ce cadre citons quelques grands hommes, avec les premiers développements de l'ère industrielle au 18ème siècle, Watt, avec ses systèmes de régulation à vapeur, Jacquard et ses métiers à tisser automatiques... Une liste exhaustive serait bien difficile à établir !.

Enfin, le développement des connaissances, et des outils mathématiques, ont conduit à un formidable essor des systèmes automatisés, et des systèmes asservis, dans la deuxième moitié du 20ème siècle. Certains se hasardent à rapprocher l'automatique et la philosophie, Observant d'étranges similitudes entre les processus propres à l'homme et l'approche technologique. Mais au fait qu'est-ce qu'un système ? Bien difficile de répondre à une telle question ! Notre point de vue porte sur les systèmes de production et les systèmes pluri-techniques en général, nous pouvons néanmoins en donner une définition plus large.

Système : toute structure dont la fonction globale est de conférer une valeur ajoutée à un ensemble de matières d'œuvre, dans un contexte donné. Simples ou complexes, les systèmes automatisés sont partout dans notre environnement quotidien.

Ils vont probablement se développer de plus en plus et prendre une place plus importante dans la manière de travailler, tant dans les ateliers de production que dans les divers bureaux des entreprises. Connaître leur fonctionnement permet aussi de mieux comprendre notre environnement.

1.2.Généralités sur les systèmes automatises :

L'automatisation d'un procède (c'est-a-dire une machine, un ensemble de machines ou plus généralement un équipement industriel) consiste a en assurer la conduite par un dispositif technologique. Le système ainsi conçu sait prendre en compte les situations pour lesquelles sa commande a été réalisée. L'intervention d'un operateur est souvent nécessaire pour assurer un pilotage global du procède pour surveiller les installations et prendre en commande manuelle (non automatique) tout ou partie du système. [1]

1.2.1. Frontière d'un système automatisé :

Pour effectuer l'étude d'un système automatisé, ou d'un sous-ensemble du système (une unité de fabrication par exemple) il est nécessaire de délimiter ce système, c'est-a-dire de définir une frontière d'isolement entre :

- d'une part, le système (ou l'unité de production) étudié .
- d'autre part, le milieu extérieur, c'est-a-dire le contexte du système isolé.

Nous dirons, par analogie avec l'étude des parties opératives (mécanique, ...), que nous avons isolé le système.

Le choix de la frontière d'isolement, bien qu'arbitraire, doit rester fonctionnel vis-à-vis de l'obtention de la valeur ajoutée.

Le système isolé peut alors être étudié spécifiquement, à condition d'avoir défini précisément ses interactions avec le milieu extérieur.

Le choix de cette frontière d'isolement système-milieu extérieur permet d'appliquer le concept de système automatisé :

- soit à une machine ou à une machine isolée d'un ensemble de machines d'une unité de production automatisée (machine à embouteiller ou machine à embouteiller isolée d'une chaîne de conditionnement automatisée) .
- soit à une unité de production indépendante (chaîne d'usinage d'une pièce détachée chez un sous-traitant de l'industrie automobile) ou à une unité de production isolée d'un atelier de production automatisée (chaîne d'usinage de carter moteur dans un atelier automatisé de fabrication-assemblage de moteurs électriques).
- soit à un atelier de production automatisée indépendant ou isolé d'une usine.
- soit à une usine de production automatisée .
- soit à un groupe d'usines. ... .[2]

1.2.2. Objectif de l'automatisation :

L'automatisation permet d'apporter des éléments supplémentaire à a la valeur ajoutée par le système.

- Accroitre la productivité du système (augmenter la quantité de produit).
- Améliorer la flexibilité de production.
- Améliorer la qualité du produit.
- adaptation à des contextes particuliers.
- Adaptation à des environnements hostiles pour l'homme (milieu marin, spatial, nucléaire,…).
- Adaptation à des taches physique ou intellectuelles pénibles pour l'homme.
- Augmenter la sécurité, … etc .. .[2]

1.2.3. Nature des informations traitées par l'automate :

Les informations peuvent être de type :

Tout ou rien (T.O.R.) : l'information ne peut prendre que deux états (vrai/faux, 0 ou 1…). C'est le type d'information délivrée par un détecteur, un bouton poussoir …

Analogique : l'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (pression, température …).

Numérique : l'information est contenue dans des mots codés sous forme binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur.[2]

1.2.4. Structure de base d'un système automatisé :

L'analyse structurelle conduit à décomposer tout système automatisé en trois grandes parties :

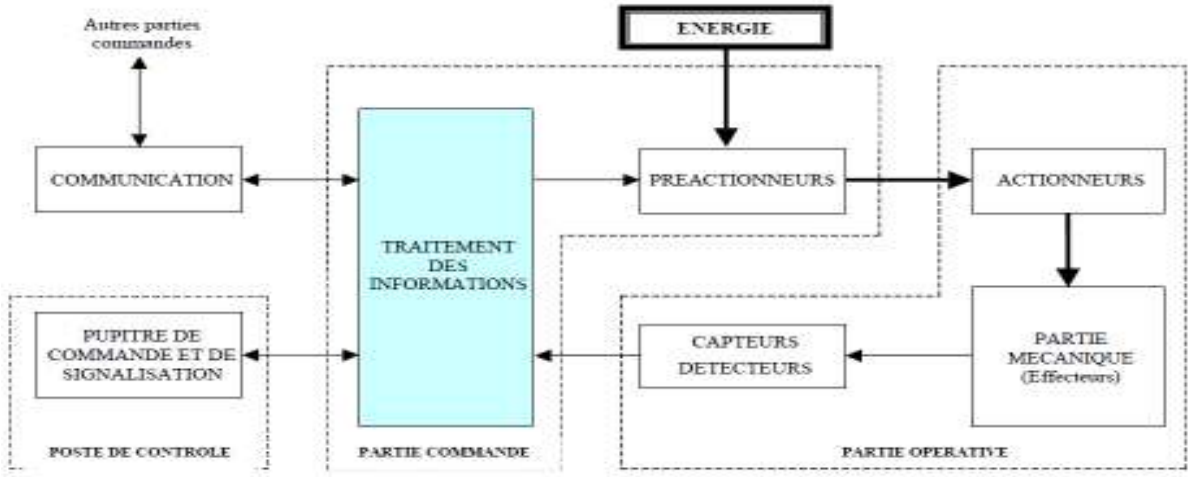


Figure1-1 : Structure de système automatisé.[2]

- La partie opérative (PO) :

Que l'on appelle également partie puissance, c'est la partie visible du système (corps) qui permet de transformer la matière d'œuvre entrante. Elle est composée d'éléments mécaniques, d'actionneurs (vérins, moteurs), de pré-actionneurs (distributeurs et contacteurs) et des éléments de détection (capteurs, détecteurs).

Pour réaliser les mouvements il est nécessaire de fournir l'énergie (Électrique, pneumatique, et hydraulique) à la PO.

- **Les actionneurs :**

Est un élément de la Partie Opérative qui reçoit une énergie « transportable » pour la transformer en énergie « utilisable » par le système. Ils exécutent les ordres reçus en agissent sur le système ou son environnement. Un actionneur est un système dont la matière d'oeuvre est l'énergie et dont la fonction est de transformer l'énergie.

Ces actionneurs appartiennent à trois technologies :

a) Actionneurs pneumatiques (vérins, moteurs).

b) Actionneur hydraulique (vérins).

c) Actionneurs électriques (moteurs électriques).

- **Pré-actionneur :**

Le Pré-actionneur est le constituant qui autorise le passage de l'énergie du milieu extérieur vers l'actionneur. Le Pré-actionneur distribue l'énergie nécessaire à l'actionneur en fonction des ordres reçus.

Le pré-actionneur peut être :

a) Contacteurs pour moteurs électriques.

b) Variateurs de vitesse pour moteurs électriques.

c) Distributeurs pour vérins pneumatiques ou hydrauliques.

- **Les capteurs :**

Les Capteurs permettent de prélever sur la partie opérative, l'état de la matière d'oeuvre et son évolution, il est capable de détecter un phénomène physique dans son environnement

(déplacement, présence, chaleur, lumière, pression...) puis transforme l'information physique en une information codée compréhensible par la partie commande.
Ce qui mène à que les capteurs transforment la variation des grandeurs physiques liées au fonctionnement de l'automatisme en signaux électriques.

- **La partie commande (PC) :**

Elle donne les ordres de fonctionnement à la partie opérative. Les pré-actionneurs permettent de commander les actionneurs ; ils assurent le transfert d'énergie entre la source de puissance (réseau électrique, pneumatique …) et les actionneurs. Exemple : contacteur, distributeur …
Ces pré-actionneurs sont commandés à leur tour par le bloc traitement des informations.
Celui-ci reçoit les consignes du pupitre de commande (opérateur) et les informations de la partie opérative transmises par les capteurs / détecteurs. En fonction de ces consignes et de son programme de gestion des tâches (implanté dans un automate programmable ou réalisé par des relais (on parle de logique câblée), elle va commander les pré- actionneurs et renvoyer des informations au pupitre de signalisation ou à d'autres systèmes de commande et/ou de supervision en utilisant un réseau et un protocole de communication.

- **Poste de contrôle :**

Composé des pupitres de commande et de signalisation, il permet à l'opérateur de commander le système (marche, arrêt, départ cycle …).
Il permet également de visualiser les différents états du système à l'aide de voyants, de terminal de dialogue ou d'interface homme - machine (IHM).[2]

1.2.5. Cahier des charges :

Le cahier des charges décrit :

- Les relations entre la partie commande et la partie opérative.
- Les conditions d'utilisation et de fonctionnement de l'automatisme.

Le fonctionnement d'un automatisme séquentiel put être décomposé en un certain nombre d'étapes .le passage (ou transition) d'une étape à une autre étape se fait à l'arrivée d'un événement particulier (réceptivité) auquel le système est réceptif.[3]

1.3. L'automate programmable industriel (API) :

1.3.1. Historique :

Les automates programmables industriels sont apparus à la fin des années soixante, à la demande de l'industrie automobile américaine (GM), qui réclamait plus d'adaptabilité de leurs systèmes de commande.
Les coûts de l'électronique permettant alors de remplacer avantageusement les technologies actuelles des techniques plus sophistiquées .
Avant d'utiliser la programmation : utilisation de relais électromagnétiques et de systèmes pneumatiques pour la réalisation des parties commandes ⇒ logique câblée
Inconvénients : cher, pas de flexibilité, pas de communication possible.
Solution : utilisation de systèmes à base de microprocesseurs permettant une modification aisée des systèmes automatisés ⇒ logique programmée.

Les ordinateurs de l'époque étant chers et non adaptés aux contraintes du monde industriel, les automates devaient permettre de répondre aux attentes de l'industrie.[2]

1.3.2. Définition :

API (Automate Programmable Industriel) ou en anglais PLC (Programmable Logic Controller) c'est un appareil électronique (matériel, logiciel, processus, un ensemble des machines ou un équipement industriel) destiné à la commande de processus industriels par un traitement séquentiel (Il contrôle les actionneurs grâce à un programme informatique qui traite les données d'entrée recueillies par des capteurs). Qui comporte une mémoire programmable par un utilisateur automaticien (et non informaticien) à l'aide d'un langage adapté (Le langage List, Le langage Ladder…etc) pour le stockage interne des instructions donnée pour satisfaire une objectif donnée. Automate permet de contrôler, coordonner et d'agir sur l'actionneur comme par exemple un robot, un bras manipulateur alors en peut dire API utilisé pour automatiser des processus. L'API est structurée autour d'une unité de calcul (processeur), de cartes d'entrées-sorties, de bus de communication et de modules d'interface et de commande.[4]

1.3.3. Différents types de l'automate programmable industriel :

- **Aspect extérieur :**

Les automates peuvent être de type compact ou modulaire.

- ✓ **Automate de type compact** :

Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes (micro automate).

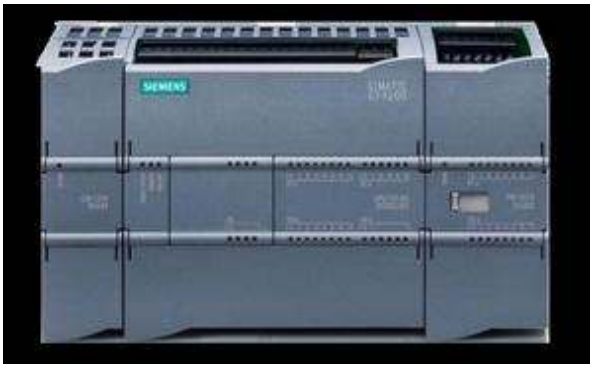


Figure1-2 : API S7-200.[4]

- ✓ **Automate de type modulaire** :

le processeur, l'alimentation et les interfaces d'entrées / sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs).

Ces automates sont intégrés dans les automatismes complexes où de puissance, capacité de traitement et flexibilité sont nécessaires.[4]

Figure1-3 : API S7-300. [4]

Remarque : Les automates compacts permettent de commander des sorties en tout ou rien et gèrent parfois des fonctions de comptage et de traitement analogique mais Les automates modulaires permettent de réaliser de nombreuses autres fonctions grâce à des modules intelligents que l'on dispose sur un ou plusieurs racks. Ces modules ont l'avantage de ne pas surcharger le travail de la CPU car ils disposent bien souvent de leur propre processeur**.** [4]

1.3.4. Architecture interne d'un automate programmable :

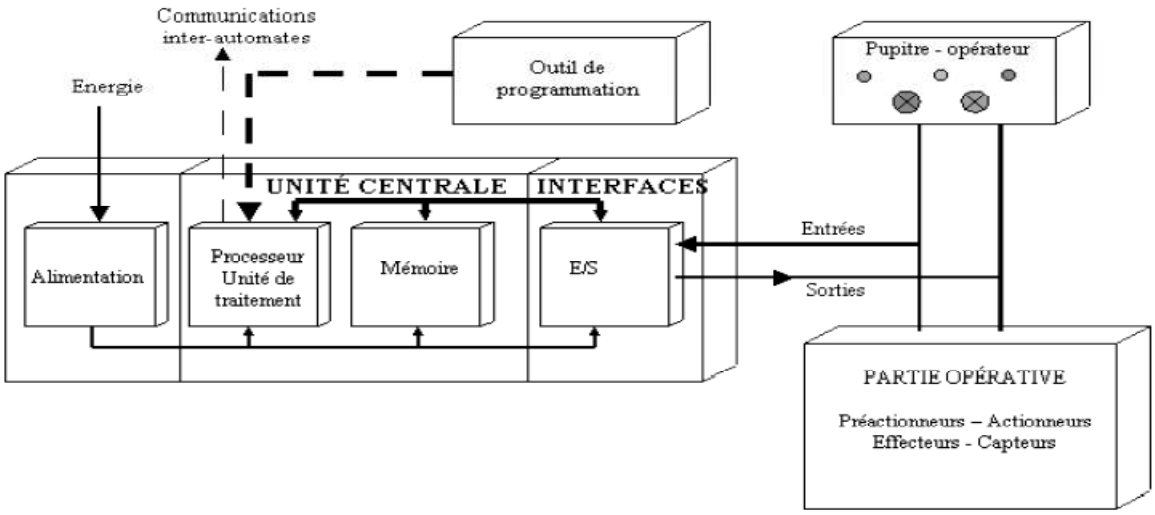


Figure1-4 : structure interne d'API.[1]

Un automate programmable est constitue essentiellement de 5modules :

1. L'unité centrale :

L'unité centrale représente le cœur de la machine, et comprend le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programmes.

Les instructions sont effectuées les unes après les autres, séquencées par une horloge.

a) Le processeur :

Un processeur est l'unité fonctionnelle capable d'interpréter et d'exécuter les instructions du programme. Dans un API le processeur gère l'ensemble des échanges informationnels en assurant :

- La lecture des informations d'entrée.
- L'exécution des instructions du programme mis en mémoire.
- La commande ou l'écriture des sorties.

Pour réaliser ces différentes fonctions, le processeur se compose :

- d'une Unité Logique (UL) qui traite les opérations logiques ET, OU et Négation.
- d'une Unité Arithmétique et Logique (UAL) qui traite les opérations de temporisation, de comptage et de calcul.
- d'un Accumulateur qui est un registre de travail dans lequel se range une donnée ou un résultat.
- d'un Registre d'Instruction qui contient, durant le temps de traitement, l'instruction a exécuter.
- d'un Décodeur d'Instruction qui décode l'instruction a exécuter en y associant les microprogrammes de traitement.
- d'un Compteur Programme ou Compteur Ordinal qui contient l'adresse de la prochaine instruction a exécuter et gère ainsi la chronologie de l'exécution des instructions du programme.

b) La mémoire :

La mémoire centrale est l'élément fonctionnel qui peut recevoir, conserver et restituer. Elle est découpée en zones ou l'on trouve :

- La zone mémoire programme (programme a exécuter) .
- La zone mémoire des données (état des entrées et des sorties, valeurs des compteurs, temporisations) .
- Une zone ou sont stockes des résultats de calcul utilises ultérieurement dans le programme.
- Une zone pour les variables internes.

Ces mémoires peuvent être :

∗ Durant la phase d'étude et de mise au point du programme :

- des mémoires vives RAM (Random Access Memory) volatiles
- des mémoires EAROM (Electrically Alterable Read Only Memory) non volatiles et effaçables partiellement par voie électrique .

∗ Durant la phase d'exploitation :

- des mémoires vives RAM qui imposent un dispositif de sauvegarde par batterie rechargeable pour éviter la volatilité de leur contenu en cas de coupure de courant.
- des mémoires mortes ROM a lecture seulement ou PROM programmables a lecture seulement .
- des mémoires ré-programmables EPROM (Erasable PROM) effaçables par un rayonnement ultraviolet et EEPROM (Electric Erasable PROM) effaçables électriquement.[1]

2. Le module d'entrées :

Un module d'entrées doit permettre a l'Unité Centrale de l'automate, d'effectuer une "lecture" de l'état logique des capteurs qui lui sont associes (module 4, 8, 16 ou 32 entrées). A chaque entrée correspond une voie qui traite le signal électrique pour élaborer une information binaire, le bit d'entrée qui est mémorise. L'ensemble des bits d'entrées forme le "mot" d'entrées. Périodiquement, le Processeur de l'automate programmable vient questionner (adresser) le module: le contenu du mot d'entrées du module est alors recopie dans la mémoire DONNEES de l'automate programmable.

Un module d'entrées est principalement défini par sa modularité (nombre de voies) et les caractéristiques électriques acceptées (tension, nature du courant...).

a) Les cartes d'entrées logiques :

Les cartes d'entrées logiques (cartes d'entrées tout ou rien) permettent de raccorder a l'automate les différents capteurs logiques tels que :

- boutons poussoirs.
- fins de course.
- capteurs de proximité inductifs ou capacitifs.
- capteurs photoélectriques.
- etc.…

Elles assurent l'adaptation, l'isolement, le filtrage et la mise en forme des signaux électriques. Une diode électroluminescente située sur la carte donne l'état de chaque entrée.

b) Les cartes d'entrées analogiques :

Les cartes d'entrées analogiques permettent de gérer des grandeurs analogiques en faisant varier un code numérique au sein du module.

Les entrées analogiques disposent d'un seul convertisseur analogique/numérique, elles sont scrutées les unes a la suite des autres par un multiplexeur a relais .

3. Le module de sorties :

Un module de sorties permet a l'automate programmable d'agir sur les actionneurs. Il réalise la correspondance: état logique signal électrique. Périodiquement, le processeur adresse le module et provoque l'écriture des bits d'un mot mémoire sur les voies de sorties du module. L'élément de commutation du module est soit électronique (transistors, triac) soit électromécanique (contacts de relais internes au module).

a) Les cartes de sorties logiques :

Les cartes de sorties logiques (tout ou rien) permettent de raccorder a l'automate les différents pré- actionneurs tels que :

- Les contacteurs.
- Les voyants.
- Les distributeurs.
- Les afficheurs...

Les tensions de sorties usuelles sont de 5 volts en continu ou de 24, 48, 110, 220 volts en alternatif. Ces cartes possèdent soit des relais, soit des triacs, soit des transistors. L'état de chaque sortie est visualise par une diode électroluminescente.

b) Les cartes de sortie analogiques :

Les cartes de sortie analogiques permettent de gérer des grandeurs analogiques en faisant varier un code numérique au sein du module. Ces modules assurent la conversion numérique/analogique, Les sorties analogiques peuvent posséder un convertisseur par voie. Le nombre de voies sur ces cartes est de 2 ou 4.

4. Le module d'alimentation :

Compose de blocs qui permettent de fournir a l'automate l'énergie nécessaire a son fonctionnement. A partir d'une alimentation en 220 volts alternatif, ces blocs délivrent des sources de tension dont l'automate a besoin : 24V, 12V ou 5V en continu.

En règle générale, un voyant positionne sur la façade indique la mise sous tension de l'automate.

5. Le module de communication :

Comprend les consoles, les boitiers de tests et les unités de dialogue en ligne :

a) Les consoles :

Il existe deux types de consoles. L'une permet le paramétrage et les relèves d'informations (modification des valeurs, et visualisation), l'autre permet en plus la programmation, le réglage et l'exploitation. Cette dernière dans la phase de programmation effectue :

- L'écriture.
- La modification.
- L'effacement.
- Le transfert d'un programme dans la mémoire de l'automate ou dans une mémoire REPROM.

La console peut également afficher le résultat de l'autotest comprenant l'état des modules d'entrées et de sorties, l'état de la mémoire, de la batterie, etc. Les consoles sont équipées souvent d'un écran a cristaux liquides. Certaines consoles ne peuvent être utilisées que connectées a un automate, d'autres peuvent fonctionner de manière autonome grâce a la mémoire interne et a leur alimentation.[1]

b) Les boîtiers de tests :

Destinées aux personnels d'entretien, ils permettent de visualiser le programme ou les valeurs des paramètres. Par exemple :

- Affichage de la ligne de programme a contrôler.

- Visualisation de l'instruction (code opératoire et adresse de l'opérande).
- Visualisation de l'état des entrées.
- Visualisation de l'état des sorties.

c) Les unités de dialogue en ligne :

Elles sont destinées aux personnels spécialistes du procède et non de l'automate programmable, elle leur permet d'agir sur certains paramètres :

- Modification des constantes, compteurs temporisations.
- Forçage des entrées/sorties.
- Exécution de parties de programme.
- Chargement de programmes en mémoire a partir de cassettes.[1]

1.3.5. Domaines d'emploi des automates :

On utilise les API dans tous les secteurs industriels pour la commande des machines (convoyage, emballage ...) ou des chaînes de production (automobile, agroalimentaire ...) ou il peut également assurer des fonctions de régulation de processus (métallurgie, chimie ...). Il est de plus en plus utilisé dans le domaine du bâtiment (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes.[3]

1.3.6. Principe de fonctionnement d'un automate programmable industriel :

L'automate programmable fonctionne par déroulement cyclique du programme. Le cycle comporte trois opérations successives qui se répètent normalement comme suit :

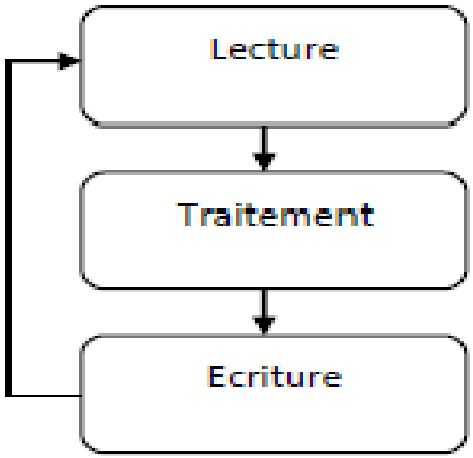


Figure1-5 : Principe de fonctionnement d'un API. [4]

Phase1 : Lecture (Photographie des entrées)

Durant cette phase qui dure quelques microsecondes :

✓ les entrées sont photographiées et leurs états logiques sont stockés dans une zone spécifique de la mémoire de donnée.

✓ Le programme n'est pas scruté.

✓ Les sorties ne sont pas mises à jour.

Phase 2 : Traitement (exécution de programme)

Durant cette phase qui dure quelques millisecondes :

- ✓ Les instructions de programme sont exécutées une à une. Si l'état d'une entrée doit être lu par le programme, c'est la valeur stockée dans la mémoire de données qui est utilisée.
- ✓ Le programme Détermine l'état des sorties et stocke ces valeurs dans une zone de la mémoire de données réservée aux sorties.
- ✓ Les entrées ne sont pas scrutées.
- ✓ Les sorties ne sont pas mises à jour.

Notez que pendant cette phase, seules la mémoire de données et la mémoire programme sont mises à contribution. Si une entrée change d'état sur le module d'entrées, l'API ne voit pas ce changement.

Phase 3 : Ecriture (mise à jour des sorties)

Durant cette phase qui dure quelques microsecondes :

- ✓ Les états des sorties mémorisés précédemment dans la mémoire de données sont reportés sur le module de sorties.
- ✓ Les entrées ne sont pas scrutées.
- ✓ Le programme n'est pas exécuté.[4]

1.3.7. Câblage de l'automate :

1.3.7.1. Alimentation de l'automate :

L'automate est alimenté généralement par le réseau monophasé 230V; 50 Hz, mais d'autres alimentations sont possibles (110 V etc...). La protection est de type magnétothermique (voir les caractéristiques de l'automate et les préconisations du constructeur). Il est souhaitable d'asservir l'alimentation de l'automate par un circuit de commande spécifique (contacteur KM1). De même, les sorties seront asservies au circuit de commande et alimentées après validation du chien de garde. [5]

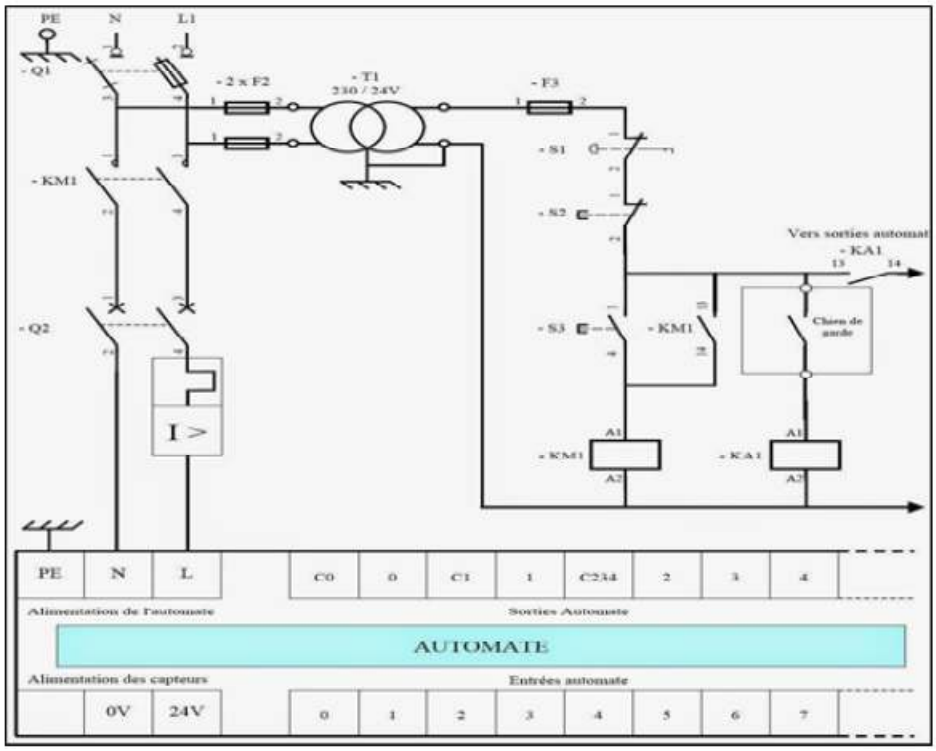


Figure1-6 : Alimentation de l'automate. [5]

1.3.7.2. Alimentations des entrées d'automate :

L'automate est pourvu généralement d'une alimentation pour les capteurs/détecteurs (attention au type de logique utilisée : logique positive ou négative). Les entrées sont connectées au OV (commun) de cette alimentation. Les informations des capteurs/détecteur sont traitées par les interfaces d'entrées. [5]

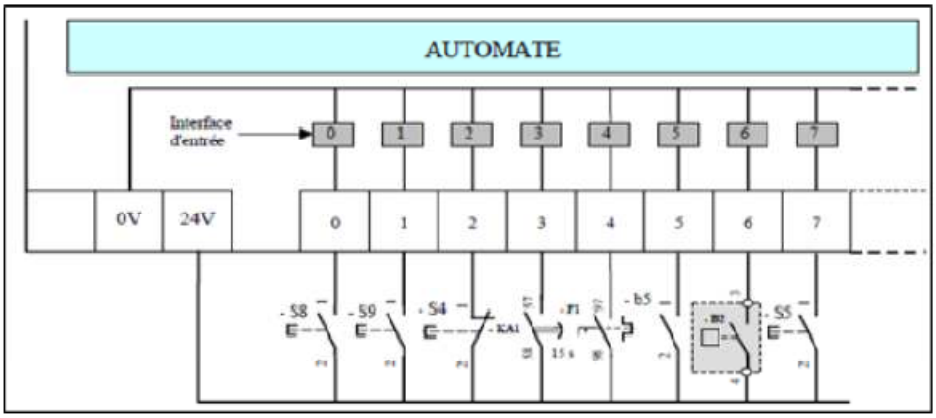


Figure1-7 *:* Alimentations des entrées d'automate. [5]

1.3.7.3. Alimentations des sorties d'automate :

Les interfaces de sorties permettent d'alimenter les divers pré-actionneurs, Il est souhaitable d'équiper chaque pré-actionneur à base de relais de circuits RC (non représentés).[5]

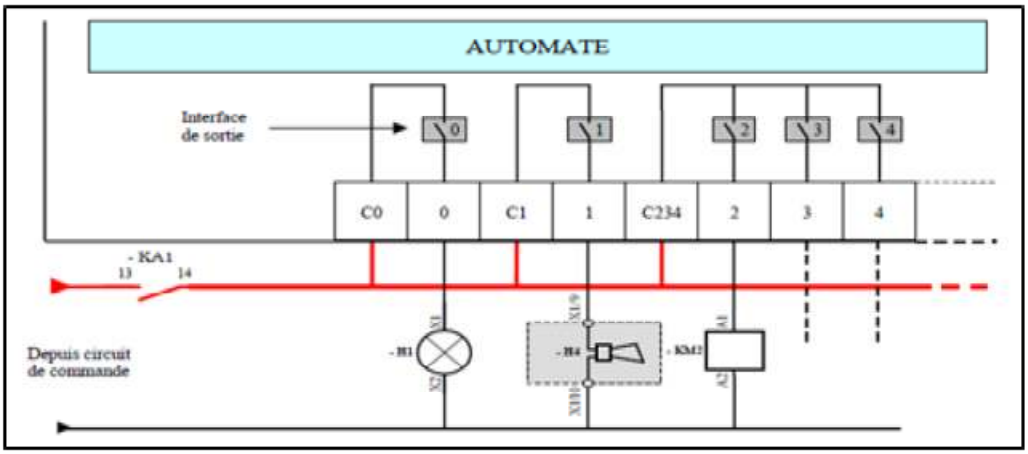


Figure1-8 : Alimentations des sorties de automate. [5]

1.3.8. Protection de l'automate :

La protection des circuits d'entrée contre les parasites électriques est souvent résolue par découplage opto-électrique . Le passage des signaux par un stade de faisceau lumineux assure en effet une séparation entre les circuits internes et externes. Du coté sorties, on doit assurer le même type de protection , mais amplification de puissance, avec au final un courant continu ou alternatif selon les cas. Deux types de sortie sont utilisés :

Sorties statiques :

Relais statiques intégrant des composants spécialisés : transistor bipolaires, thyristors.

Ces composants n'ont aucune usure mécanique et leurs caractéristiques de commutation se maintiennent dans le temps.

Sorties relais électromagnétique :

Où le découplage résulte de l'existence de deux circuits électriques (Bobines d'excitation, circuits de puissance), Ces relais électromagnétique ont l'avantage d'avoir une faible résistance de contact, une faible capacité de sortie et surtout un faible coût, mais une durée de vie et une vitesse de commutation inférieure aux sorties statiques.[5]

1.3.9. Critère de choix d'un API :

Le choix d'un API est fonction de la partie commande à programmer. On doit tenir compte de plusieurs critères :

- ✓ Nombre d'entrées / sorties.
- ✓ Le temps de traitement.
- ✓ La capacité de la mémoire.
- ✓ Le nombre d'étapes ou d'instructions.
- ✓ Le nombre de temporisateurs.
- ✓ Le langage de programmation.[4]

1.3.10. Les avantages et les inconvénients :

Ses avantages sont :

- ✓ Améliorer les conditions de travail en éliminant les travaux répétitifs.
- ✓ Améliorer la productivité en augmentant la production.
- ✓ Améliorant la qualité des produits ou en réduisant les coûts de production.
- ✓ Automates programmables sont programmés facilement et ont un langage de programmation facile à comprendre (logique programmé) alors la Modification du programme facile par rapport à la logique câblée.
- ✓ Simplification du câblage.
- ✓ Puissance et rapidité.
- ✓ Facilité de maintenance (l'API par lui même est relativement fiable et peut aider l'homme dans sa recherche de défauts).
- ✓ Augmenter la sécurité.
- ✓ Possibilités de communication avec l'extérieur (ordinateur, autre API)
- ✓ énorme possibilité d'exploitation.
- ✓ plus économique.

Ses inconvénients sont :

- ✓ Plantage.
- ✓ Il y a trop de travail requis dans les fils de connexion.
- ✓ Besoin de formation.[4]

1.4. Conclusion :

D'après ce qui précède, le développement scientifique a laissé sa trace sur les systèmes de production donnant naissance au Système Automatisé de Production, qui s'avère être plus ou moins un remède au paradoxe des paramètres coût-qualité visés généralement par la gestion de production (Optimisation du coût, qualité et délai). Le rôle de l'automatisme industriel est prépondérant puisque les systèmes automatisés occupent et contrôlent l'ensemble des secteurs de l'économie, il a comme objectif d'améliorer la productivité, la qualité, la sécurité et autres variables qui peuvent influencés les objectifs de l'entreprise.

Chapitre 02

Les langages de programmation

2.1. Introduction :

L'écriture d'un programme consiste à créer une liste d'instructions permettant l'exécution des opérations nécessaires au fonctionnement du système. Il existe différents types de langage de programmation :

Le langage GRAFCET.
Le langage à contact (Ladder).
Le langage LIST.
Le langage booléen (Logigramme).

L'API traduit le langage de programmation en langage compréhensible directement par le microprocesseur. Ce langage est propre à chaque constructeur, il est lié au matériel mis en œuvre.

Chaque instruction du programme est composée :

- de l'opération à effectuer (la nature de l'opération est codée 1 ou 0).
- de la variable sur laquelle l'opération va être effectué (variable de sortie, variable d'entrée, variable interne,…).
- de la nature de la variable (binaire, numérique, texte, …).

Chaque instruction est écrite dans une partie de la mémoire appelée adresse ou label.

2.2. Langage GRAFCET :

2.2.1. Définition :

(Graphe Fonctionnel de Commande Étapes-Transitions). Le grafcet est un diagramme fonctionnel; il représente par un graphe le fonctionnement de la partie opérative, donc les actions effectuées par le système. Il nous servira ensuite à décrire le fonctionnement de la partie commande, c'est-à-dire la technologie employée pour commander les actionneurs.[6]

2.2.2. Domaine d'application :

Le *diagramme* fonctionnel est indépendant des techniques séquentielles "tout ou rien", pneumatique, électrique ou électronique, câblées ou programmées, pouvant être utilisées pour réaliser l'automatisme de commande. Mais l'utilisation de séquenceurs, d'une part, et d'automates à instructions d'étapes d'autre
part, permet une transcription directe du diagramme fonctionnel .

Cette représentation graphique concise et facile à lire est aisément compréhensible par toute personne en relation avec le système automatisé, du concepteur à l'utilisateur sans oublier l'agent de maintenance.

Utilisé industriellement, le GRAFCET est aussi enseigné dans les options techniques et l'enseignement supérieur.

Depuis les premières publications le concernant et surtout depuis la norme française NF C03-190 de 1982, cet outil a été travaillé et enrichi par le groupe systèmes logiques de l'AFCET (Association Française pour la Cybernétique Economique et Technique).

Il existe une documentation et symboles graphiques, diagramme fonctionnel "Grafcet" éditée par l'Union Technique de l'Electricité. UTE C03-190 Nov. 1990.[7]

2.2.3. Principe du grafcet :

Pour visualiser le fonctionnement de l'automatisme, le GRAFCET utilise une succession alternée d'ETAPES et de TRANSITIONS.

A chaque étape correspond une ou plusieurs actions à exécuter. Une étape est soit active, soit inactive. Les actions associées à cette étape sont effectuées lorsque celle-ci est active.

Les transitions indiquent avec les LIAISONS ORIENTEES, les possibilités d'évolution entre étapes.

A chaque transition est obligatoirement associée une condition logique pouvant être vraie ou fausse.

Cette condition de transition est appelée RECEPTIVITE. L'évolution d'une étape à une autre ne peut s'effectuer que par le franchissement d'une transition.

Une transition ne peut être franchie, donc activer l'étape suivante que :

- si elle est validée par l'étape antérieure active.
- et que les conditions de réceptivité soient satisfaites.

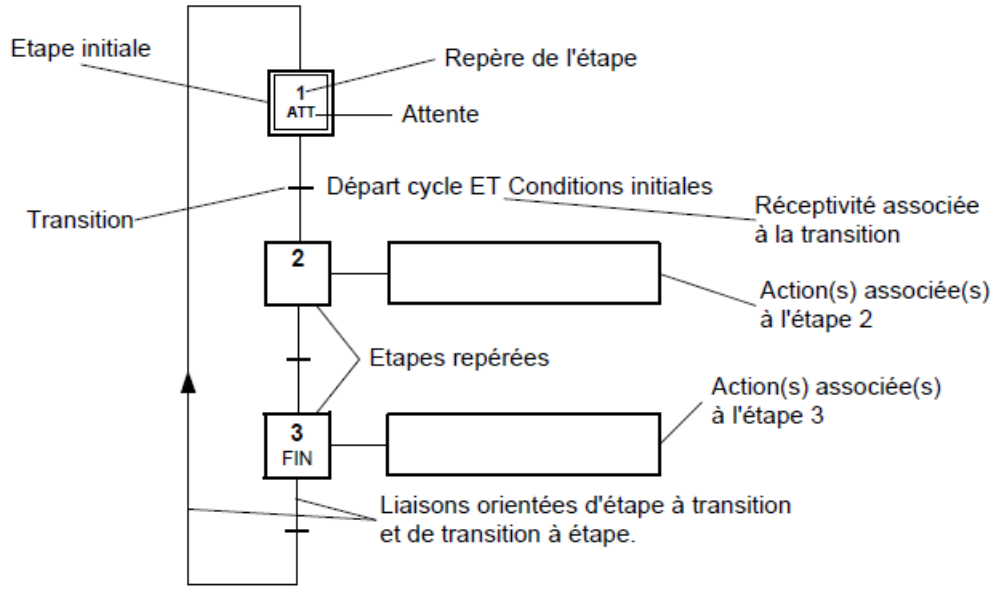


Figure2-1: principe de grafcet.

Etape initiale : représente une étape qui est active au début du fonctionnement.
Elle se différencie de l'étape en doublant les côtés du carré.
Transition : la transition est représentée par un trait horizontal.
Réceptivité : les conditions de réceptivité sont inscrites à droite de la transition.
Etape : chaque étape est représentée par un carré repéré numériquement.
Action(s) : elles sont décrites littéralement ou symboliquement à l'intérieur d'un ou plusieurs rectangles reliés par un trait à la partie droite de l'étape.
Liaisons orientées : indique le sens du parcours.[7]

2.2.4. Règles d'évolution du grafcet :

Règle 1 : L'initialisation précise les étapes actives au début du fonctionnement. Elles sont activées inconditionnellement et repérées sur le GRAFCET en doublant les côtés des symboles correspondants.

Figure2-2 : Etape initiale.

Règle 2 : Une transition est soit validée soit non validée. Elle est validée lorsque toutes les étapes immédiatement précédentes sont activées .
Elle ne peut être franchie que :
- lorsqu'elle est validée.
- et que la réceptivité associée à la transition est vraie.

La transition est alors obligatoirement franchie.

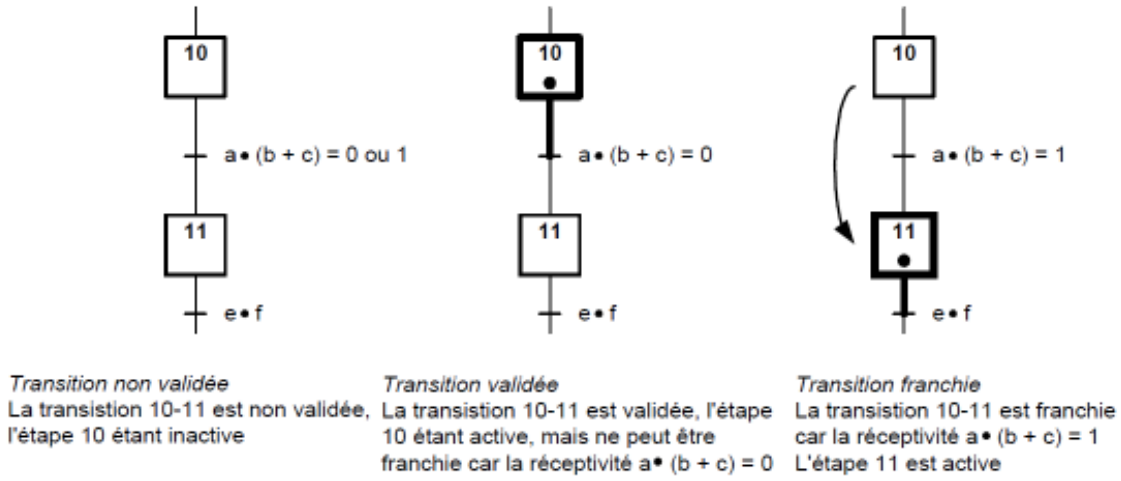


Figure2-3: Franchissement d'une transition.

Règle 3 : Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes . Cette évolution du GRAFCET est donc synchrone.
Il y a évolution asynchrone lorsque le franchissement de la transition entraîne l'activation des étapes suivantes et que c'est la vérification de cette activation qui autorise la désactivation des étapes précédentes.

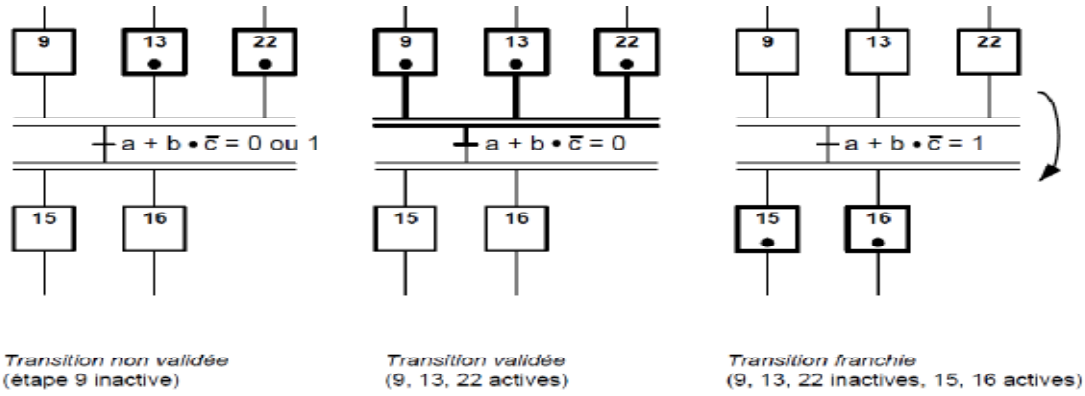


Figure2-4: evolution des étapes actives**.**

Règle 4 : Plusieurs transitions simultanément franchissables sont simultanément franchies.
Règle 5 : Si au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste activée. L'activation doit être prioritaire sur la désactivation au niveau d'une même étape.

Remarque : La durée de franchissement d'une transition ne peut jamais être rigoureusement nulle, même si, théoriquement (règles 3 et 4), elle peut être rendue aussi petite que possible. Il en est de même de la durée d'activation d'une étape. En outre, la règle 5 se rencontre très rarement dans la pratique. Ces règles ont été ainsi formulées pour des raisons de cohérence théorique interne au GRAFCET.[7]

2.3. Langage CONTACT (LADDER) :

2.3.1. Définition :

le langage à contacts (LD: Ladder Diagram) est composé de réseaux lus les uns à la suite des autres par l'automate. Ces réseaux sont constitués de divers symboles représentant les entrées/sorties de l'automate, les opérateurs séquentiels (temporisations, compteurs, …), les opérations, ainsi que les bits systèmes internes à l'automate (ces bits permettent d'activer ou non certaines options de l'automate, telle que l'initialisation des grafcets).[6]

2.3.2. Les symboles utilisés :

Il existe 3 types d'élément de langage :

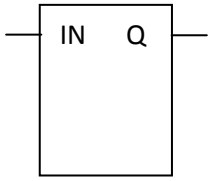
- les entrées (ou contact), qui permettent de lire la valeur d'une variable booléenne,
- les sorties (ou bobines) qui permettent d'écrire la valeur d'une variable booléenne,
- les blocs fonctionnels qui permettent de réaliser des fonctions avancées.[5]

Le tableau 2.1 donne les principaux éléments (contacte et bobines) d'un réseau LD.

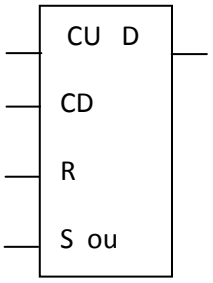
Tableau 2.1 : les principaux éléments (contacte et bobines) d'un réseau LD.

Object graphique	Nom
-| |-	Contact normalement ouvert
-|/|-	Contact normalement fermé
-|P|-	Contact fermé au front montant
-|N|-	Contact fermé au front descendant
-( )-	Bobine normalement ouverte
-(/)-	Bobine normalement fermée
-(S)- ou–(L)-	Bobine Latch (maintenu à 1 une fois actionné)
-(R)-ou –(U)-	Bobine Reset (remise à 0 de la bobine latch)
-(P)-	Bobine active au front montant de son entrée
-(N)-	Bobine active au front descendant de son entrée
|-<return>	Retour inconditionnel (vers le sous-programme appelant)
|-cond-<return>	Retour conditionnel
|->>Label	Saut inconditionnel
|-cond->>Label	Saut conditionnel

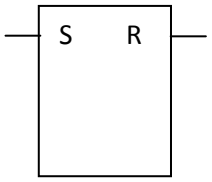
b. Les blocs fonctionnels (Les circuits séquentiels) **:**



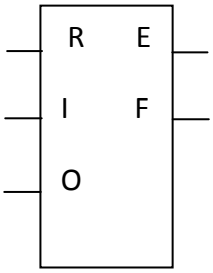
Les blocs temporisations possèdent une entrée I reliée aux éléments graphiques précédents et une sortie activée lorsque le temps écoulé depuis l'activation de la temporisation atteint la valeur prédéfinie.



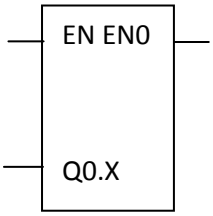
Les fonctions comptage/décomptage peuvent être séparées ou réunies dans un seul bloc selon les marques. CU est l'entrée de comptage sur front montant, CN est l'entrée de décomptage front montant, R est l'entrée de remise à zéro de la valeur courante et S ou LD est l'entrée de chargement de la valeur prédéfinie. D est la sortie lorsque la valeur prédéfinie ou le sont atteints selon que l'on compte ou que l'on décompte.



Les monostables ne sont pas utilisés dans toutes les marques. S est l'entrée d'activation sur front montant du monostable et R sa sortie logique.



Registre TÉLÉMÉCANIQUE. R est l'entrée de remise à zéro du registre, I est l'entrée stockage sur front montant et O est l'entrée déstockage sur front montant. La sortie E indique que le registre est vide et la sortie F qu'il est plein.



Générateur d'impulsions SIEMENS. EN est l'entrée qui permet d'activer le train d'impulsions. ENO est la sortie qui permet de relier plusieurs générateurs en série plutôt qu'en parallèle (ENO = EN). Q0.X est la sortie du train d'impulsions. Ne peuvent être utilisés pour cette fonction que les sorties Q0.0 et Q0.1..[6]

2.3.3. Structure d'un réseau de contacts:

Un réseau de contacts se compose de la manière suivante: étiquette (ou titre) + commentaire + réseau graphique (zone test + zone action).

La zone de test accueille :

- les contacts.
- les blocs fonction (temporisations, compteurs, …).
- les blocs comparaison.

La zone action accueille :

- les bobines.
- les blocs opérations.[6]

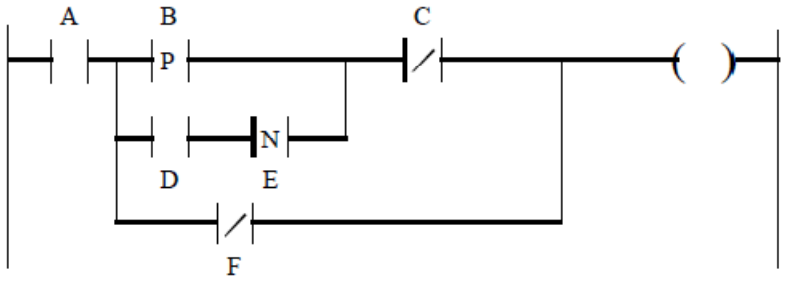
2.3.4. Règles d'évolution d'un réseau de contacts :

La lecture d'un réseau se fait réseau connexe par réseau connexe (de haut en bas), puis de gauche à droite à l'intérieur d'un réseau connexe.

Un réseau connexe est constitué d'éléments graphiques tous reliés entre eux, mais indépendants des autres éléments graphiques du réseau.

Si l'on rencontre une liaison verticale de convergence, on évalue d'abord le sous-réseau qui lui est associé (toujours dans la même logique) avant de continuer l'évaluation du sous-réseau qui l'englobe.

Exemple :



L'ordre d'exécution des éléments de ce réseau est le suivant :

- 1ère phase: lecture des contacts A et B jusqu'à la rencontre de la 1ère liaison verticale de convergence entre les contacts B et C .
- 2ème phase: lecture du premier sous-réseau, contacts D et E .
- 3ème phase: reprise de lecture de la première ligne du réseau connexe, contact E, jusqu'à la rencontre de la deuxième liaison verticale de convergence .
- 4ème phase : lecture du deuxième sous-réseau, contact F.
- 5ème phase : lecture de la bobine.

Remarque : la mise à jour des sorties s'effectue en fin de cycle, après la lecture de tout le programme.[6]

2.4. Langage LISTE :

2.4.1. Définition :

Langage basique des automatismes il représente une liste d'instructions qui met en oeuvre, comme pour le langage booléen, des équations logiques. Il permet également de résoudre quelques calculs numériques. Bien que les différents programmes en liste d'instructions des différents constructeurs d'API possèdent tous leurs spécificités, ils sont néanmoins structurés de la façon suivante.

Chaque instruction est composée d'un CODE INSTRUCTION et d'un OPERANDE.[6]

2.4.2. Les instructions de base :

a) Instructions de test :

LD : contact normalement ouvert.
LDN : contact normalement fermé.
LDR ou EU : contact à front montant.
LDF ou ED : contact à front descendant.
AND : liaison série (ET) à un contact normalement ouvert.

ANDN : liaison série (ET) à un contact normalement fermé.
ANDR : liaison série (ET) à un contact à front montant.
ANDF : liaison série (ET) à un contact à front descendant.
OR : liaison parallèle (OU) à un contact normalement ouvert.
ORN : liaison parallèle (OU) à un contact normalement fermé.
ORR : liaison parallèle (OU) à un contact à front montant.
ORF : liaison parallèle (OU) à un contact à front descendant.

Remarque : Les instructions AND et OR (et leurs dérivés) peuvent utiliser des parenthèses.
Ces parenthèses permettent de traduire des schémas à contact de façon simple. Il est possible d'imbriquer plusieurs niveaux de parenthèses.[6]

b) Instructions d'action :
ST : bobine directe. L'objet bit associé prend la valeur du résultat de la zone test.
STN : bobine inversée. L'objet bit associé prend la valeur inversée du résultat de la zone test.
S : bobine d'enclenchement. L'objet bit associé est mis à 1 lorsque la valeur du résultat de la zone test est à 1.
R : bobine de déclenchement. L'objet bit associé est mis à 0 lorsque la valeur du résultat de la zone test est à 1.

c) Instructions de saut :
JMP : saut de programme inconditionnel.
JMPC : saut de programme si le résultat de l'instruction test précédente est à 1.
JMPCN : saut de programme si le résultat de l'instruction test précédente est à 0.
SRN : branchement en début de sous-programme.
RET : retour de sous-programme inconditionnel.
RETC : retour de sous-programme si le résultat de l'instruction test précédente est à 1.
RETCN : retour de sous-programme si le résultat de l'instruction test précédente est à 0.

d) Instructions d'arrêt :
END : fin de programme inconditionnelle.
ENDC : fin de programme si le résultat de l'instruction test précédente est à 1.
ENDCN : fin de programme si le résultat de l'instruction test précédente est à 0.
HALT : Arrêt de l'exécution du programme.

e) Opérations de transfert (SIEMENS) :
MOVB : transfert d'un octet dans un autre.
MOVW : transfert un mot dans un autre.
MOVD : transfert d'un double mot dans un autre.
MOVR : transfert d'un double mot réel dans un autre.[6]

2.4.3. Programmation des blocs fonction :

Temporisation: le pilotage est réalisé par des instructions et la sortie peut être transféré dans un bit .
Exemple :

```
LD    %I1.1
IN    %TM1
LD    Q
ST    %Q2.0
```

Le bit d'entrée I1.1 pilote la temporisation TM1.
La sortie Q de la temporisation est chargé dans le bit de sortie Q2.0.

Compteur/décompteur : les pilotages sont réalisés par des instructions et la sortie est directement disponible sous forme de bit.

Exemple :

```
LD      %I1.1
R       %C8
LDN     %I1.2
ANDN    %M0
CU      %C8
LD      %C8.D
ST      %Q2.0
```

Le bit d'entrée I1.1 pilote la fonction remise à zéro R du compteur C8.
Le bit d'entrée inversé I1.2 et le bit mémoire inversé M0 pilotent la fonction comptage CU du compteur C8.
Le bit de sortie C8.D est chargé dans le bit de sortie O2.0.

Monostable: le pilotage est réalisé par des instructions et la sortie est directement disponible sous forme de bit.

Exemple :

```
LDN     %I1.1
AND     %M10
S       %MN0
LD      %MN0.R
ST      %Q3.0
```

Le bit d'entrée inversé I1.1 et le bit mémoire M10 pilotent l'entrée S du monostable MN0.
Le bit de sortie MN0.R est chargé dans le bit de sortie Q3.0.

2.4.4. Structure d'une phrase :

Chaque phrase d'instructions commence par un point d'exclamation généré automatiquement. Comme pour le langage à contacts, elle peut comporter un commentaire et être repérée par une étiquette.

Exemple :

```
! (*Attente de séchage*)          Commentaire entre (*  *).
%L2:                              Etiquette de repérage de la phrase.
LD      %I1.0     Instruction  ┐
AND     %M10      Instruction  ├ Phrase
ST      %Q2.5     Instruction  ┘

Code instruction  Opérande
```

Remarque : il faut bien faire attention aux priorités des instructions lors de la rédaction d'un programme en LIST.[6]

2.4.5. Règles d'exécution d'un réseau :

L'exécution d'un réseau en LIST s'effectue séquentiellement instruction par instruction.
La première instruction d'une séquence d'instructions doit toujours être LD ou une instruction inconditionnelle (ex: JMP). Toutes les autres instructions utilisent le résultat booléen précédent.

Remarque : le séquence ment des instructions peut être modifié par les instructions de saut et d'appel à un sous-programme.[6]

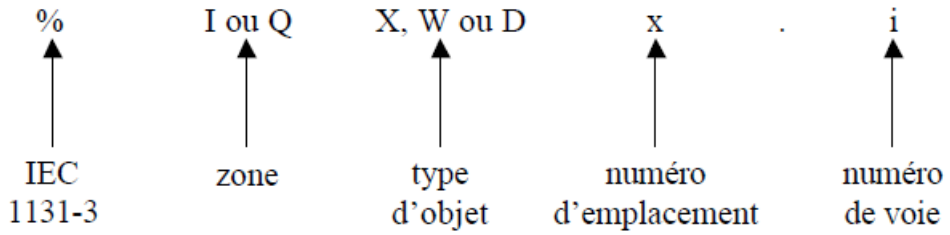
2.4.6. Priorités d'exécution du programme :
Un programme est divisé en plusieurs parties. Celles-ci diffèrent selon les marques (se référer à la documentation constructeur pour les priorités d'exécution).[6]

2.4.7. Les objets langage :
Il existe cinq principaux adressages pour les objets langage :
- la zone mémoire (M).
- la zone des entrées (I).
- la zone des sorties (Q).
- la zone des constantes (K).
- la zone système (S).

Il existe quatre principaux objets langage :
- l'objet bit (X), facultatif pour un adressage direct.
- l'objet mot (W).
- l'objet octet (B).
- l'objet double mot (D).

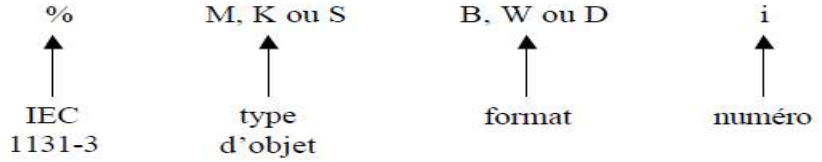
- L'adressage direct :
Il existe deux éditeurs de programme :
- SIMATIC (SIEMENS),
- IEC 1131-3 (TELEMECANIQUE et SIEMENS).

1) Objets d'entrées/sorties :



Remarque : Pour les objets bits, le X n'est pas obligatoire sauf s'il est extrait d'un mot (ex: %MW10:X4: désigne le bit de rang 4 dans le mot numéro 10 de la mémoire interne) .

2) Objets mot :



M : mémoire interne servant à stocker des valeurs tout au long du programme.
K : mots constants écrits en même temps que le programme aux emplois divers.
S : mots systèmes assurant plusieurs fonctions (modes de marche, temps de

fonctionnement, …). Les mots double longueur (D) sont l'association de deux mots simple longueur. Ainsi, %MD0 = %MW0 + %MW1.

Remarque : les mots peuvent prendre une valeur positive ou négative. Le signe est donné par le bit de poids le plus fort (1=négatif et 0=positif) .[6]

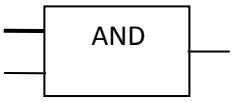
2.5. Langage LOGIGRAMME :

2.5.1. Définition :

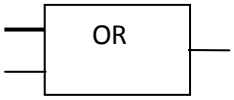
Un réseau LOG est composé d'une ou plusieurs boîtes d'opérations LOG. Au lieu d'utiliser des contacts, on affecte une ou plusieurs valeurs binaires comme entrées à une boîte d'opération LOG. Vous utilisez les sorties de l'opération pour connecter cette dernière à une opération consécutive ou pour achever le réseau. Ainsi, une seule opération LOG peut représenter la même fonction qu'un ensemble de contacts, bobines ou boîtes en schéma à contacts. Le réseau est achevé lorsque vous avez procédé à l'affectation de tous les paramètres de l'opération ou que vous les avez connectés à une autre opération.[6]

2.5.2. Les symboles utilisés :

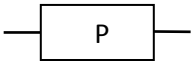
a) Les opérations sur bits :



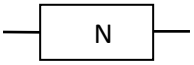
Cette boîte représente la fonction ET en associant deux bits que l'on peut inverser à l'entrée de la boîte. Il est possible de rajouter jusqu'à 32 entrées.



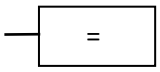
Cette boîte représente la fonction OU en associant deux bits que l'on peut inverser à l'entrée de la boîte. Il est possible de rajouter jusqu'à 32 entrées.



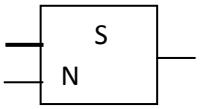
Contact à front montant



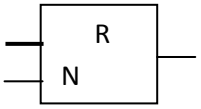
Contact à front descendant.



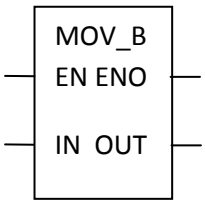
L'opération Sortie écrit la nouvelle valeur du bit de sortie dans la mémoire image.



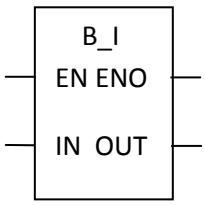
L'opération SET met à 1 un nombre N de sorties à partir de l'adresse bit indiquée.



L'opération RESET met à 0 un nombre N de sorties à partir de l'adresse bit indiquée.

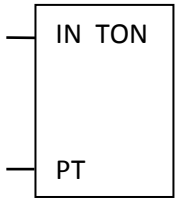


Les opérations de transfert permettent de transférer l'octet d'entrée (IN) dans l'octet de sortie (OUT). Cette fonction est utilisable avec des mots (MOV_W), des doubles mots (MOV_DW), des réels (MOV_R), …

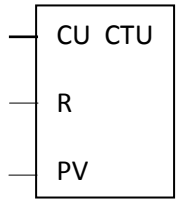


Les blocs de conversion permettent de convertir un nombre d'une base donnée dans une autre base (vu dans le module 2). Il est également possible de tronquer ou arrondir un nombre, de l'encoder ou le décoder.

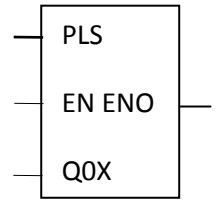
b) Les circuits séquentiels :



Il existe trois types de temporisations: TON (retard à la montée), TONR (retard à la montée temporisé), TOF (retard à la descente). IN est l'entrée de validation et PT le temps prédéfini. La base de temps dépend du numéro de la temporisation.



Il existe trois principaux types de compteurs/décompteurs: CTU (compteur incrémental), CTD (compteur décrémental), CTUD (compteur incrémental/décrémental). PV est la valeur prédéfinie et R la remise à zéro. L'entrée d'incrémentation s'appelle CU et l'entrée de décrémentation CD.



Le bloc PLS permet de générer des trains d'impulsions de rapport cyclique 50% (PTO) ou bien modulable (PWM).

c) Les blocs comparaison :
Ces blocs permettent de comparer des nombres, des bits, des octets ou des mots en supériorité, infériorité ou égalité.
d) Les opérations d'exécution :
Ces blocs regroupent plusieurs fonctions établissant l'ordre d'exécution du programme: saut de programme, retour de sous-programme, fin de programme, … [6]

2.5.3. Structure d'un programme LOG:

Un réseau LOG se compose de la manière suivante:
étiquette (ou titre) + commentaire + réseau graphique. Voici les quelques règles permettant de créer un réseau LOG :

- on peut créer un réseau contenant une seule opération LOG si cela correspond au contexte du programme.
- il n'y a pas de limite maximale fixe pour les opérations d'un réseau. On peut considérer la fenêtre de l'éditeur de programme LOG comme une grille divisée en cellules. Les cellules sont les zones dans lesquelles on place une opération, on affecte une valeur à un paramètre ou on trace un segment de ligne. Dans cette grille, un réseau individuel ne peut pas s'étendre sur plus de 32 cellules horizontalement ou 32 cellules verticalement.
- si une boîte d'opération comporte des sorties >>, il faut soit fournir une connexion à une autre boîte, soit affecter des valeurs aux paramètres de sortie. S'il s'agit d'une sortie ENO>|, on peut la laisser vide,
- il n'est pas possible de relier directement les sorties de plusieurs opérations entre elles. Pour connecter plusieurs sorties, il faut connecter chacune d'elles à un paramètre d'entrée d'une boîte AND ou OR, pour en faire une sortie unique.[6]

2.6. Conclusion :

Dans ce chapitre nous avons présenté les langages de programmation de l'API. Les langages destinés à la programmation des automates programmables industriels ont pour objectifs d'être facilement mis en œuvre par tout technicien après une courte formation. L'écriture d'un programme consiste à créer une liste d'instructions permettant l'exécution des opérations nécessaires au fonctionnement du système.

Chapitre 03

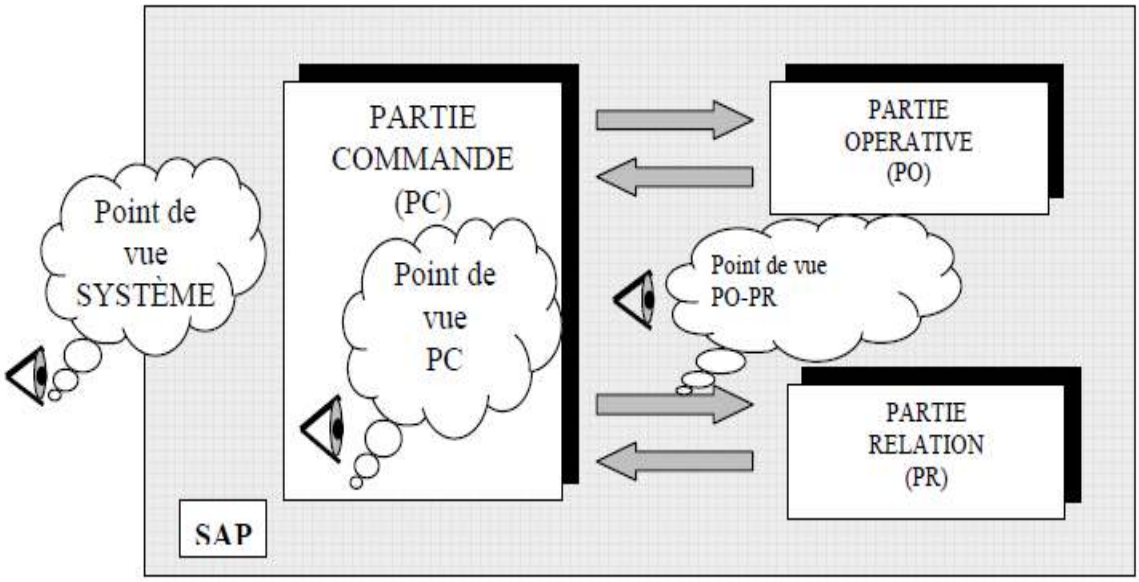
Programmation par Grafcet

3.1. Introduction :

Le Grafcet est un outil graphique de définition pour l'automatisme séquentiel, en tout ou rien. Mais il est également utilisé dans beaucoup de cas combinatoires, dans le cas où il y a une séquence à respecter mais où l'état des capteurs suffirait pour résoudre le problème en combinatoire. Il utilise une représentation graphique.

C'est un langage clair, strict mais sans ambiguïté, permettant par exemple au réalisateur de montrer au donneur d'ordre comment il a compris le cahier des charges. Langage universel, indépendant (dans un premier temps) de la réalisation pratique (peut se "câbler" par séquenceurs, être programmé sur automate voire sur ordinateur).

3.2. Notion de point de vue :



Figeur3-1: Notion de point de vue. [8]

3.2.1. Point de vue système (procédé et processus) :

Description faite par un observateur se situant d'un point de vue externe au SAP.

Le point de vue système décrit le comportement du système vis à vis du produit.

Le procédé est l'ensemble des fonctions successives exécutées sur un même produit au cours de sa fabrication.

Le processus est l'organisation du procédé. C'est la succession des fonctions simultanées réalisées sur tous les produits présents dans le système automatisé.

Le GRAFCET du point de vue système permet le dialogue entre le client et le concepteur pour la spécification du système automatisé.

3.2.2. Point de vue partie opérative :

Description du comportement du système faite par un observateur se situant d'un point de vue interne au SAP et externe à la PC. Les choix technologiques de la PO sont effectués.

Le point de vue partie opérative décrit les actions produites par les actionneurs à partir des informations acquises par les capteurs.

Le GRAFCET du point de vue partie opérative permet le dialogue entre le concepteur de la partie opérative et le concepteur de la partie commande.
La notation, à ce niveau peut être littérale (ex : fermeture de la porte) ou symbolique en utilisant les repères du dossier technique.[8]

3.2.3. Point de vue partie commande :

Description du comportement du système par un observateur se situant d'un point de vue interne à la PC.
Ce GRAFCET prend en compte **les choix technologiques** et l'ensemble des échanges **PC↔ PO** et **PC↔ Opérateur**. Il décrit dans un premier temps la marche normale et peut évoluer en fonction des modes demarches et d'arrêts imposés par le cahier des charges du système automatisé.
C'est le GRAFCET du point de vue du réalisateur de la Partie Commande
La notation retenue à ce niveau est la notation symbolique utilisant les repères du dossier technique. [8]

3.3. Le modèle GRAFCET :

L'AFCET (**A**ssociation **F**rançaise pour la **C**ybernétique **E**conomique et **T**echnique) et l'ADEPA (**A**gence nationale pour le **Dé**veloppement de la **P**roduction **A**utomatisée) ont mis au point et développé une représentation graphique qui traduit, sans ambiguïté, l'évolution du cycle d'un automatisme séquentiel.
Ce diagramme fonctionnel: le GRAFCET (**Gra**phe **F**onctionnel de **C**ommande, **E**tapes **T**ransitions) permet de décrire les comportements attendus de l'automatisme en imposant une démarche rigoureuse, évitant ainsi les incohérences dans le fonctionnement.

3.3.1. Définitions :

Le modèle est défini par un ensemble constitué :
- d'éléments graphiques de base comprenant : les étapes, les transitions, les liaisons orientées.
- d'une interprétation traduisant le comportement de la partie commande vis-à-vis de ses entrées et de ses sorties, et caractérisée par les réceptivités associées aux transitions et les actions associées aux étapes.
- de 5 règles d'évolution définissant formellement le comportement dynamique de la partie commande.
- d'hypothèses sur les durées relatives aux évolutions.[8]

3.3.2. Eléments graphiques de base :

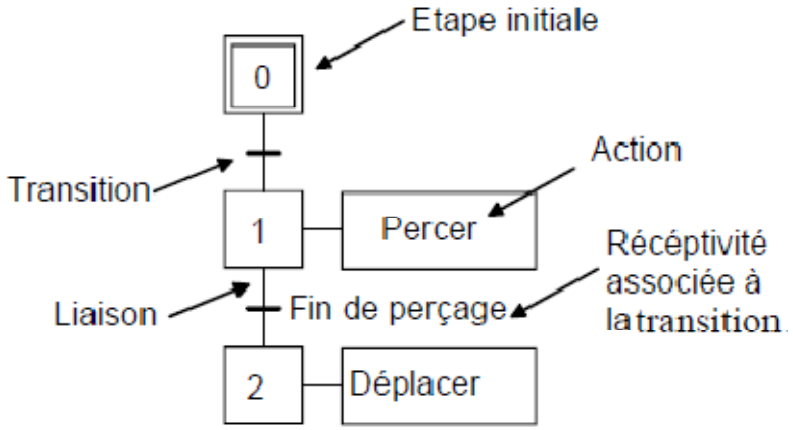


Figure 3-2 : Eléments de base en GRAFCET.

Étape: une étape représente une situation stable de la PC.
Une étape est soit active soit inactive. On peut associer à chaque étape i une variable Xi image de son activité.
Étape initiale: étape active au début du fonctionnement. Elle se représente par un double carré.
Liaisons orientées: Elles relient les étapes aux transitions et les transitions aux étapes. Le sens général d'évolution est du haut vers le bas. Dans le cas contraire, des flèches doivent être employées.
Transitions: une transition indique une possibilité d'évolution d'activité entre deux ou plusieurs étapes. Cette évolution s'accomplit par le franchissement de la transition.
Réceptivité: La réceptivité associée à une transition est une fonction logique :
- des entrées (capteurs, commande opérateur).
- des activités des étapes.
- des variables auxiliaires.

Action: L'action indique, dans un rectangle, comment agir sur la variable de sortie, soit par assignation (action continue), soit par affectation (action mémorisée).

3.3.3. Règles d'évolution :

Règle 1: Situation initiale
La situation initiale est la situation à l'instant initial, elle est donc décrite par l'ensemble des étapes actives à cet instant.
Règle 2: Franchissement d'une transition
Une transition est validée lorsque toutes les étapes, immédiatement précédentes reliées à cette transition, sont actives. Le franchissement d'une transition se produit :
- lorsque la transition est VALIDÉE .
- ET QUE la réceptivité associée à cette transition est VRAIE.

Règle 3: Évolution des étapes actives
Le franchissement d'une transition provoque simultanément :
- L'activation de toutes les étapes immédiatement suivantes.
- La désactivation de toutes les étapes immédiatement précédentes.

Règle 4: Évolutions simultanées
Plusieurs transitions simultanément franchissables sont simultanément franchies.

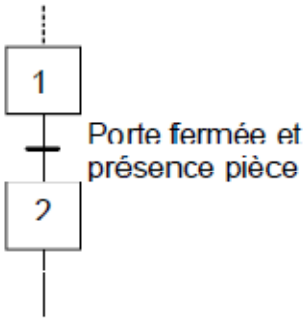
Règle 5: Activation et désactivation simultanée d'une même étape
Si au cours d'une évolution, une même étape se trouve être à la fois activée et désactivée, elle reste active.

3.3.4. Règle de syntaxe :
L'alternance étape-transition et transition-étape doit toujours être respectée quelle que soit la séquence parcourue.

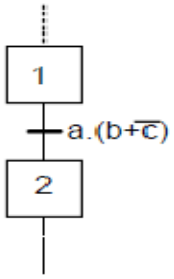
3.3.5. Les réceptivités :

- Les réceptivités associées aux transitions :
Une proposition logique, appelée réceptivité, qui peut être vraie ou fausse est associée à chaque transition.

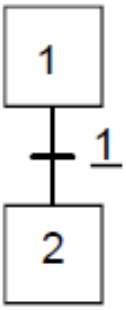
Description d'une réceptivité par un texte



Description d'une réceptivité par une expression booléenne



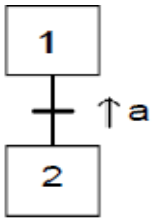
- Réceptivité toujours vraie :
La notation 1 (1 souligné) indique que la réceptivité est toujours vraie.

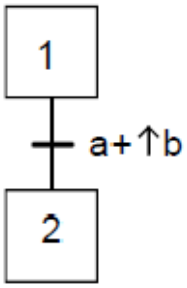


Dans ce cas, l'évolution est dite toujours fugace le franchissement de la transition n'est conditionné que par l'activité de l'étape amont.[8]

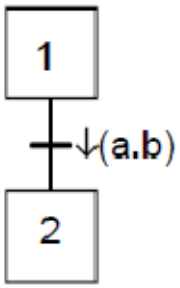
- Front montant et descendant d'une variable logique :

- **Front montant :**
  La notation ↑ indique que la réceptivité n'est vraie que lorsque la variable passe de la valeur 0 à la valeur 1.
- **Front descendant :**
  La notation ↓ indique que la réceptivité n'est vraie que lorsque la variable passe de la valeur 1 à la valeur 0.





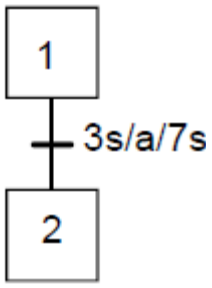
La réceptivité n'est vraie que lorsque a est vraie ou que b passe de l'état 0 à l'état 1

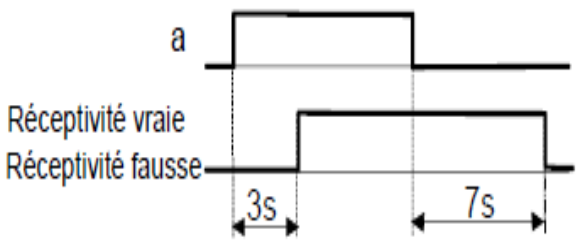


La réceptivité n'est vraie que lorsque le produit logique « a.b » passe l'état 1 à l'état 0

- **Réceptivité dépendante du temps :**

La notation est de la forme « **t1/variable/t2** ».Dans l'exemple ci contre, la réceptivité n'est vraie que 3s après que « a » passe de l'état 0 à l'état 1, elle ne redevient fausse que 7 s après que « a » passe de l'état 1 à l'état 0.

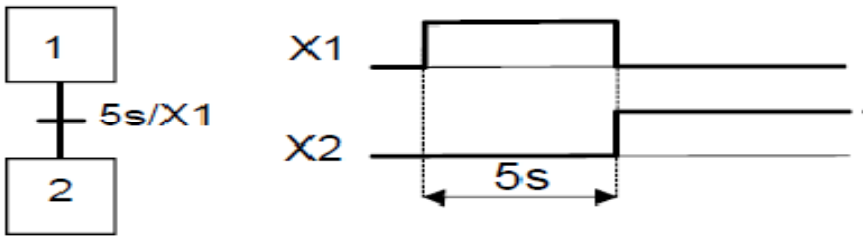




- **Simplification usuelle :**

L'utilisation la plus courante est la temporisation de la variable d'étape avec un temps t2 égal à zéro :

Dans ce cas la durée d'activité de l'étape 1 est de 5 s.



Remarque : Il est possible d'utiliser cette notation lorsque l'étape temporisée n'est pas l'étape amont de la transition.

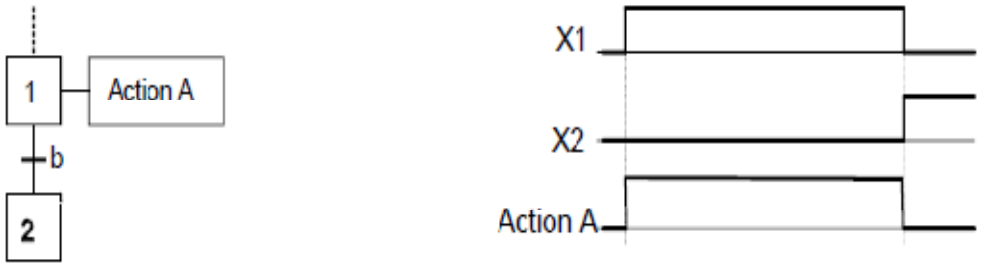
3.3.6. Les actions associées :

Une ou plusieurs actions élémentaires ou complexes peuvent être associées à une étape. Les actions traduisent ce qui doit être fait chaque fois que l'étape à laquelle elles sont associées est active. Il existe 2 types d'actions :

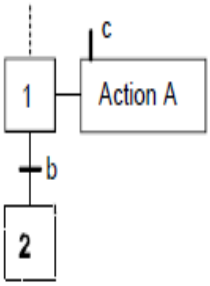
- les actions continues.
- Les actions mémorisées.

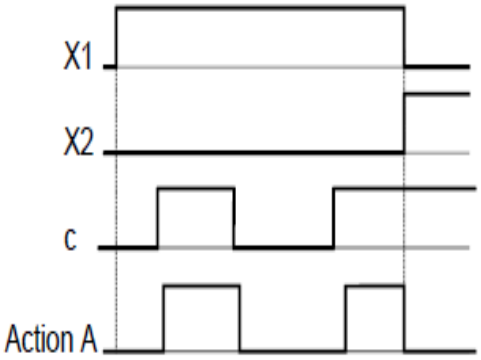
3.3.6.1. Action continue :

L'exécution de l'action se poursuit tant que l'étape à laquelle elle est associée est active et que la condition d'assignation (expression logique de variables d'entrées et/ou de variables internes) est vérifiée. En l'absence de condition l'action s'effectue tant que l'étape à laquelle elle est associée est active.

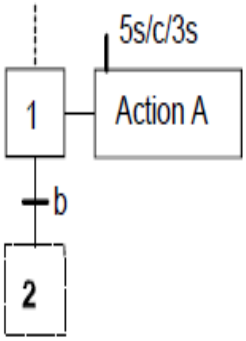


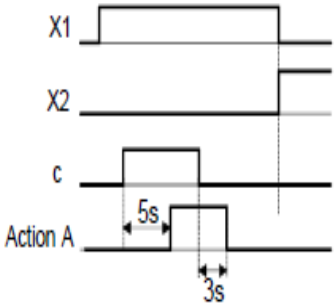
Une proposition logique, appelée condition d'assignation, qui peut être vraie ou fausse, conditionne l'action continue. La condition d'assignation ne doit jamais comporter de front de variables d'entrées et/ou de variables internes.



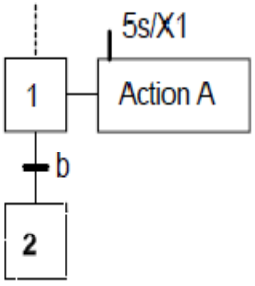


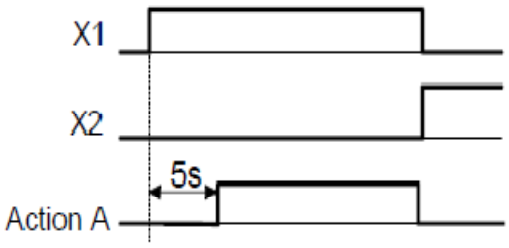
La condition d'assignation n'est vraie que 5 secondes après que « **c** » passe de l'état 0 à l'état 1 (front montant de c) ; elle ne redevient fausse que 3 secondes après que « **c** » passe de l'état 1 à l'état 0 (front descendant de c).[8]



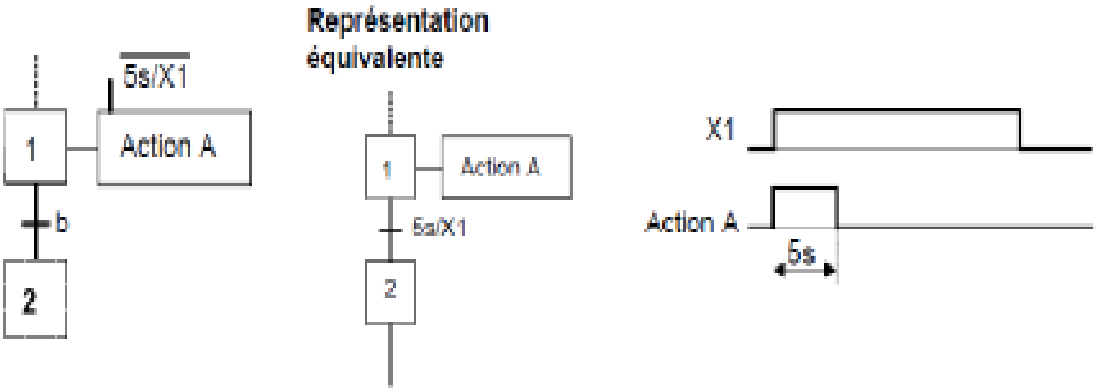


L'action retardée est une action continue dont la condition d'assignation n'est vraie qu'après une durée t1 spécifiée depuis l'activation de l'étape associée. Dans l'exemple ci-dessous, l'action A sera exécutée 5s après l'activation de l'étape 1.





L'action limitée dans le temps est une action continue dont la condition d'assignation n'est vraie que pendant une durée t1 spécifiée depuis l'activation de l'étape à laquelle elle est associée.



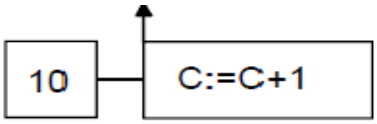
3.3.6.2. Action maintenue ou mémorisée :

Pour qu'une action reste maintenue lorsque l'étape qui l'a commandée vient d'être désactivée, il faut utiliser une action mémorisée.

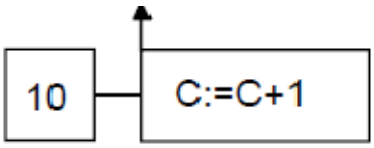
En mode mémorisé c'est l'association d'une action à des événements internes qui permet d'indiquer qu'une variable de sortie prend et garde la valeur imposée si l'un des événements se produit.

3.3.6.3. Action à l'activation et à la désactivation :

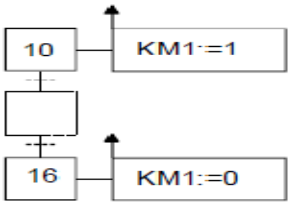
- **Une action à l'activation :** est une action mémorisée lors de l'activation de l'étape liée à cette action.



Une action à la désactivation est une action mémorisée lors de la désactivation de l'étape liée à cette action.



Mise à 0 du compteur C à la désactivation de l'étape 10.



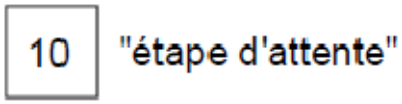
KM1=1 dès l'activation de l'étape 10 et reste à 1 jusqu'à l'activation de l'étape 16.

Une action sur évènement est une action mémorisée conditionnée à l'apparition d'un événement, l'étape à laquelle l'action est reliée étant active. Il est impératif que l'expression logique associée à l'évènement comporte un ou plusieurs fronts de variables d'entrées.

Incrémentation du compteur C sur le front montant de « **a** », l'étape 10 étant active.

3.3.7. Commentaires :

Un commentaire relatif aux éléments graphiques d'un GRAFCET peut être placé entre guillemets.



3.3.8. Les structures de base :

3.3.8.1. Séquence linéaire :

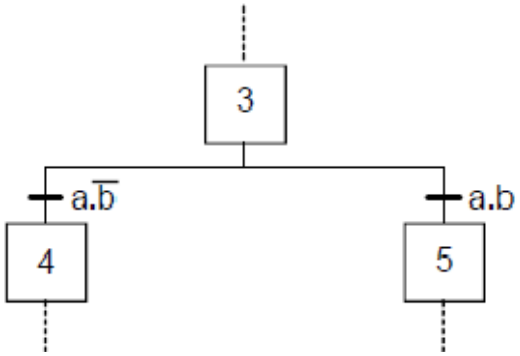
Une séquence linéaire est composée d'une suite d'étapes qui peuvent être activées les unes après les autres.

3.3.8.2. Sélection de séquence :

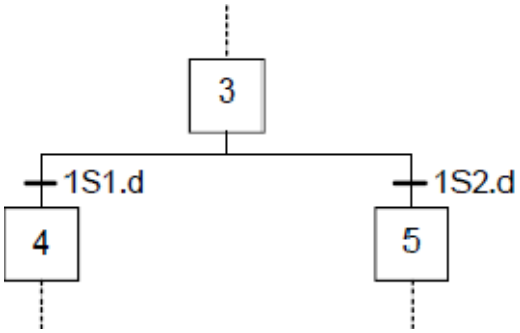
Une sélection de séquence est un choix d'évolution entre plusieurs séquences à partir d'une ou plusieurs
étapes. Elle se représente graphiquement par autant de transitions validées en même temps qu'il peut y avoir d'évolution possibles. L'exclusion entre les séquences n'est pas structurelle. Pour l'obtenir, il faut s'assurer soit de l'incompatibilité mécanique ou temporelle des réceptivités, soit de leur exclusion logique.[8]

- Exclusivité logique :

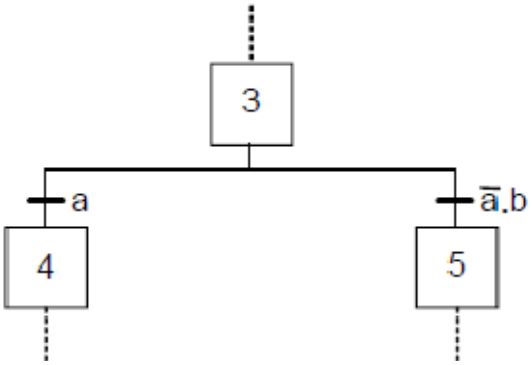
Les réceptivités a.b et a.b sont logiquement exclusives.



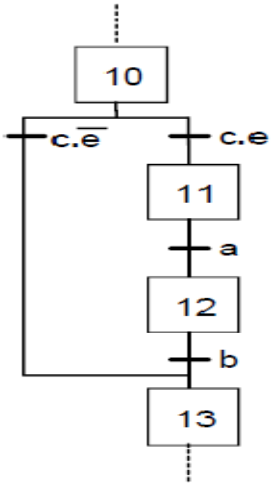
- Exclusivité technologique :
Les réceptivités 1S1.d et 1S2.d sont technologiquement exclusives par les capteurs fins de course 1S1 et 1S2 du vérin 1A.



- Exclusivité avec priorité :
Les réceptivités a et /a.b sont exclusives avec priorité à l'évolution 3→4 sur l'évolution 3→5 si a=1 et b=1.

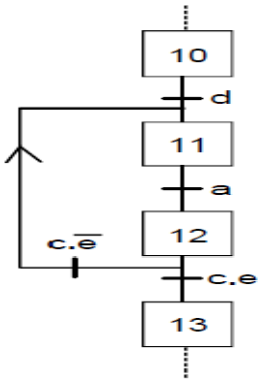


3.3.8.3. Saut d'étapes et reprise de séquence :
- Saut d'étapes :
Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées à ces étapes deviennent inutiles.



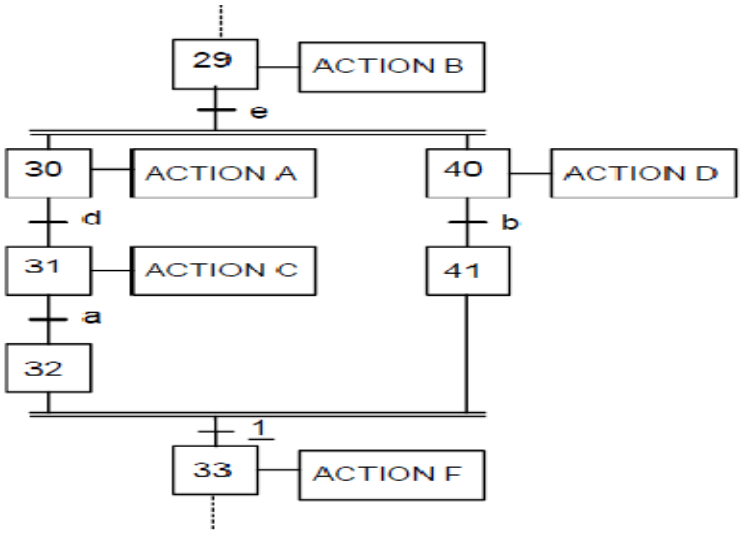
- Reprise de séquence :

La reprise de séquence permet de recommencer plusieurs fois la même séquence tant qu'une condition n'est pas obtenue.



3.3.8.4. Séquences simultanées (séquences parallèles) :

Si le franchissement d'une transition conduit à activer plusieurs étapes en même temps, ces étapes déclencheront des séquences dont les évolutions seront à la fois simultanées et indépendantes.



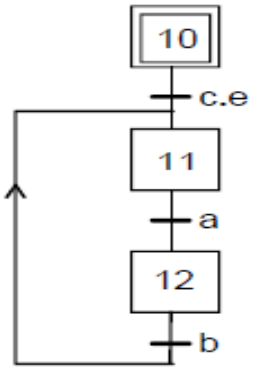
Si l'étape 29 est active, la réceptivité « **e** » provoque, lorsqu'elle est vraie, l'activation simultanée des étapes 30 et 40. Les deux séquences évoluent alors indépendamment l'une de l'autre.
Les étapes 32 et 41 sont des étapes d'attente; dès qu'elles sont actives, la transition 32,41→33 est franchie (1 : réceptivité toujours vraie) ce qui entraîne simultanément, l'activation de l'étape 33 et la désactivation des étapes 32 et 41.
On remarque :
- que l'activation de l'étape 32 permet d'éviter que l'action C se poursuive lorsque a est vraie et que b ne l'est pas encore.
- que l'activation de l'étape 41 permet d'éviter que l'action D se poursuive lorsque b est vraie et que a ne l'est pas encore.[8]

3.3.9. Les structures particulières :

3.3.9.1. Étape et transition source :

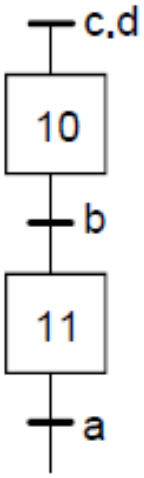
Étape source :

Une étape source est une étape qui ne possède aucune transition amont. Dans l'exemple ci-dessous, l'étape source initiale 10 n'est active qu'à l'initialisation (et tant que la réceptivité c.e n'est pas vraie).



-Transition source :

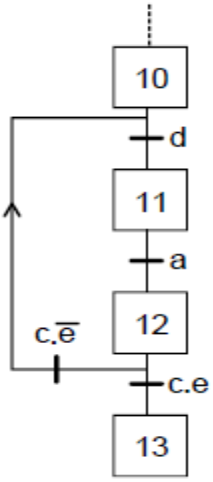
Une transition source et une transition qui ne possède aucune étape amont. Par convention, la transition source et toujours validée et est franchie dès que sa réceptivité est vraie. Dans l'exemple ci-dessous, l'étape 10 est activée dès que la réceptivité « **c.d** » est vraie.



3.3.9.2. Étape et transition puits :

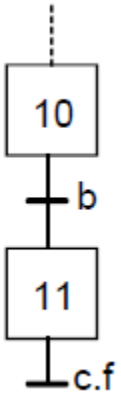
- Étape puits :

Une étape puits est une étape qui ne possède aucune transition aval ; sa désactivation est possible par un ordre de forçage d'un GRAFCET de niveau supérieur.



- Transition puits :

Une transition puits et une transition qui ne possède aucune étape aval. Dans l'exemple ci-dessous, lorsque la transition puits est validée et que « **c.d** » est vraie, le franchissement de cette transition a pour unique conséquence de désactiver l'étape 11.

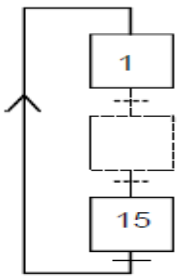


3.3.10. Remarques sur les liaisons orientées :

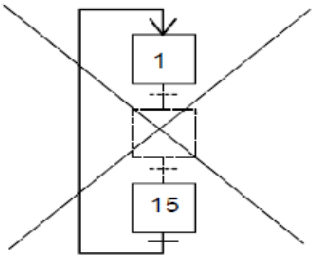
3.3.10.1. Liaison orientée de bas en haut :

Par convention le sens d'évolution est toujours de haut en bas. Des flèches doivent être utilisées si cette Convention n'est pas respectée ou si leur présence peut apporter une meilleure compréhension.

Solution conseillée

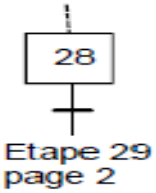


Solution à éviter



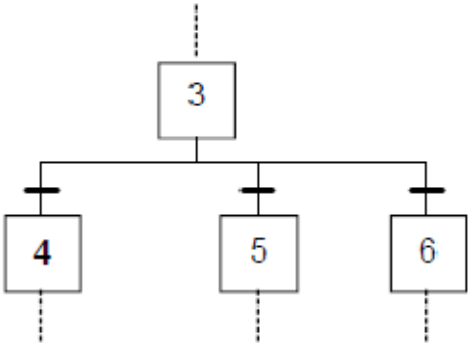
3.3.10.2. Repère de liaison :

Lorsqu'une liaison orientée doit être interrompue, (dans des dessins complexes ou dans le cas de représentation sur plusieurs pages), le repère de l'étape de destination ainsi que le repère de la page à laquelle elle apparaît doivent être indiqués.[8]

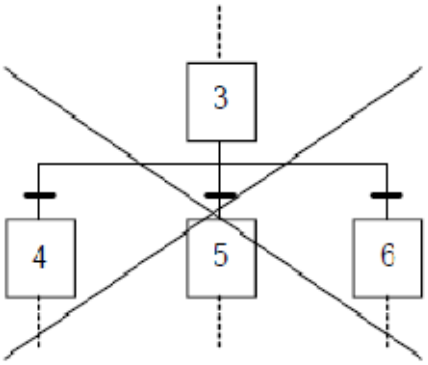


3.3.10.3. Cas de la sélection de séquence :

Solution conseillée



Solution à éviter



3.4. Conclusion :

En ce qui concerne les domaines d'application le Grafcet est bien adapté à la programmation d'un automate directement en interaction avec les capteurs et les actionneurs (commande locale), le Grafcet est plus proche de la mis en œuvre et on spécifie les réceptivités et les actions en même temps que l'on construit le Grafcet.

Chapitre 04

application sur l’ascenseur

4.1. Introduction :

La production d'une machine automatique exige la connaissance du cahier des charges par le client et le fabricant qui propose des solutions, mais le dialogue entre elles n'est pas toujours facile, le client ne peut pas décrire avec précision le problème et le fabricant ne peut pas soulever toutes les circonstances En raison du langage du dialogue entre eux et voici le rôle de GRAFCET, qui donne la solution au problème.

Après avoir analysé les engagements de temps d'un système automatisé et sa représentation par GRAFCET du point de vue de la partie de contrôle vient l'analyse physique et peut passer facilement du GRAFCET au mode de réalisation avec la liberté de choix de la logique.

4.2. L'ascenseur :

4.2.1. Définition :

Appareil élévateur installe a demeure, desservant des niveaux définis, comportant une cabine, dont les dimensions et la constitution permettent manifestement l'accès des personnes, se déplaçant, au moins partiellement, le long de guides verticaux ou dont l'inclinaison sur la verticale est inferieure a 15°.[1]

4.2.2. Description :

Les ascenseurs a traction a câbles sont les types d'ascenseurs les plus fréquemment utilises, notamment dans les bâtiments tertiaires. Ils se différencient entre eux selon le type démotorisation :

- à moteur-treuil a vis sans fin.
- à moteur-treuil planétaire.
- à moteur a attaque directe (couramment appelé "Gearless" ou sans treuil).

Un ascenseur a treuil se compose essentiellement d'une cabine, d'un contre-poids, des câbles reliant la cabine au contre-poids, des guides, et d'un système de traction au dessus de la cage de l'ascenseur.[1]

4.2.3. Cahier des charges :

-L'ascenseur à 4 étages :

Obligations en matière de mécanique :

- À chaque étage, un bouton pourvu d'une LED doit permettre d'appeler l'ascenseur.
- Dans la cabine (zone grisée dans la partie inférieure de la maquette), des boutons pourvus de LED doivent permettre de choisir un étage.
- Le sens de déplacement de la cabine doit être indiqué à chaque étage par des diodes. L'affichage est lié en interne à la commande moteur.
- L'affichage des étages est codé en bits sur deux entrées.

Obligations pour le programmateur :

- En cas d'appel par pression sur un bouton, la cabine doit se rendre à l'étage correspondant.
- L'ascenseur ne peut se déplacer à l'étage suivant que lorsque toutes les portes sont
- fermées.
- En cas d'appel depuis un étage, tous les boutons d'appel de l'étage et de la cabine doivent être allumés.
- L'appel ne s'éteint que lorsque la cabine a atteint l'étage correspondant et que les portes commencent à s'ouvrir.
- L'étage où se trouve la cabine est indiqué sur le panneau d'affichage des étages.

- En présence de plusieurs appels, la porte de la cabine doit rester ouverte au moins 2 s au même étage avant de se refermer et que la cabine n'aille à l'étage suivant.
- Si la cabine se trouve à l'étage central et est appelée à la fois à l'étage du bas et à l'étage du haut, l'ascenseur se rend d'abord à l'étage qui a appelé le premier.
- La cabine ne doit pas dépasser les étages inférieur et supérieur.
- Une pression sur le bouton d'arrêt d'urgence entraîne l'arrêt de tous les moteurs (montée/descente de l'ascenseur, ouverture/fermeture des portes).
- Une seconde pression sur le bouton d'arrêt d'urgence entraîne la reprise de l'action interrompue.
- En cas d'urgence, il doit être possible de commander la cabine de l'extérieur.

- Ici, nous mettrons un GRAFCET théorique pour l'ascenseur selon le Cahier des charges données.

➢ **Tableau4.1 :** Tableau mnémonique théorique.

mnémonique	**Commentaire**
B0	Bouton pour sélection de l'étage 0.
B1	Bouton pour sélection de l'étage 1.
B2	Bouton pour sélection de l'étage 2.
B3	Bouton pour sélection de l'étage 3.
Dcy	Bouton d'arrêt d'urgence.
E0	capteur Cabine à l'étage 0.
E1	capteur Cabine à l'étage 1.
E2	capteur Cabine à l'étage 2.
E3	capteur Cabine à l'étage 3.
KM1-	La cabine descend.
KM1+	La cabine monte.
KM2-F	Ferme la porte.
KM2-O	Ouvrir la porte.
L E0	Signal du capteur Cabine à l'étage 0.
LE1	Signal du capteur Cabine à l'étage 1.
LE2	Signal du capteur Cabine à l'étage 2.
LE3	Signal du capteur Cabine à l'étage 3.
PF	Porte Cabine sélectionnée se ferme.
PO	Porte Cabine sélectionnée s'ouvre.
POE0	Le porte de étage 0 sélectionnée s'ouvre.
POE1	Le porte de étage 1 sélectionnée s'ouvre.
POE2	Le porte de étage 2 sélectionnée s'ouvre.
POE3	Le porte de étage 3 sélectionnée s'ouvre.
PFE0	Le porte de étage 0 sélectionnée se ferme.
PFE1	Le porte de étage 1 sélectionnée se ferme.
PFE2	Le porte de étage 2 sélectionnée se ferme.
PFE3	Le porte de étage 3 sélectionnée se ferme.

4.2.4. Diagramme grafcet de l'ascenseur :

- étape initial :

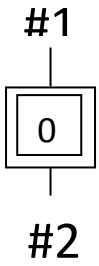


Figure4-1: étape initial de l'ascenseur

- étage 0 :

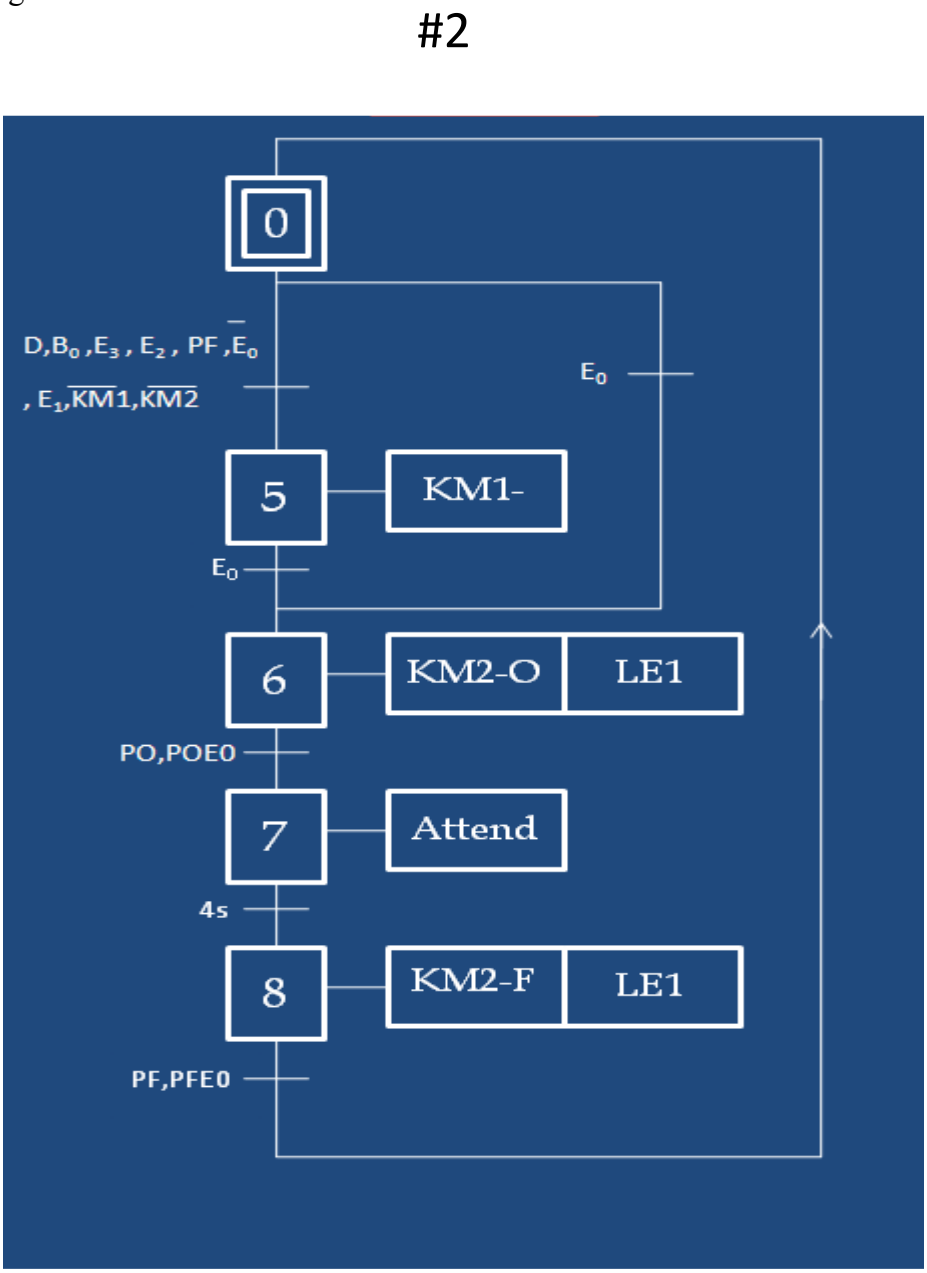


Figure4-2: l'étage 0 de l'ascenseur

- étage 1 :

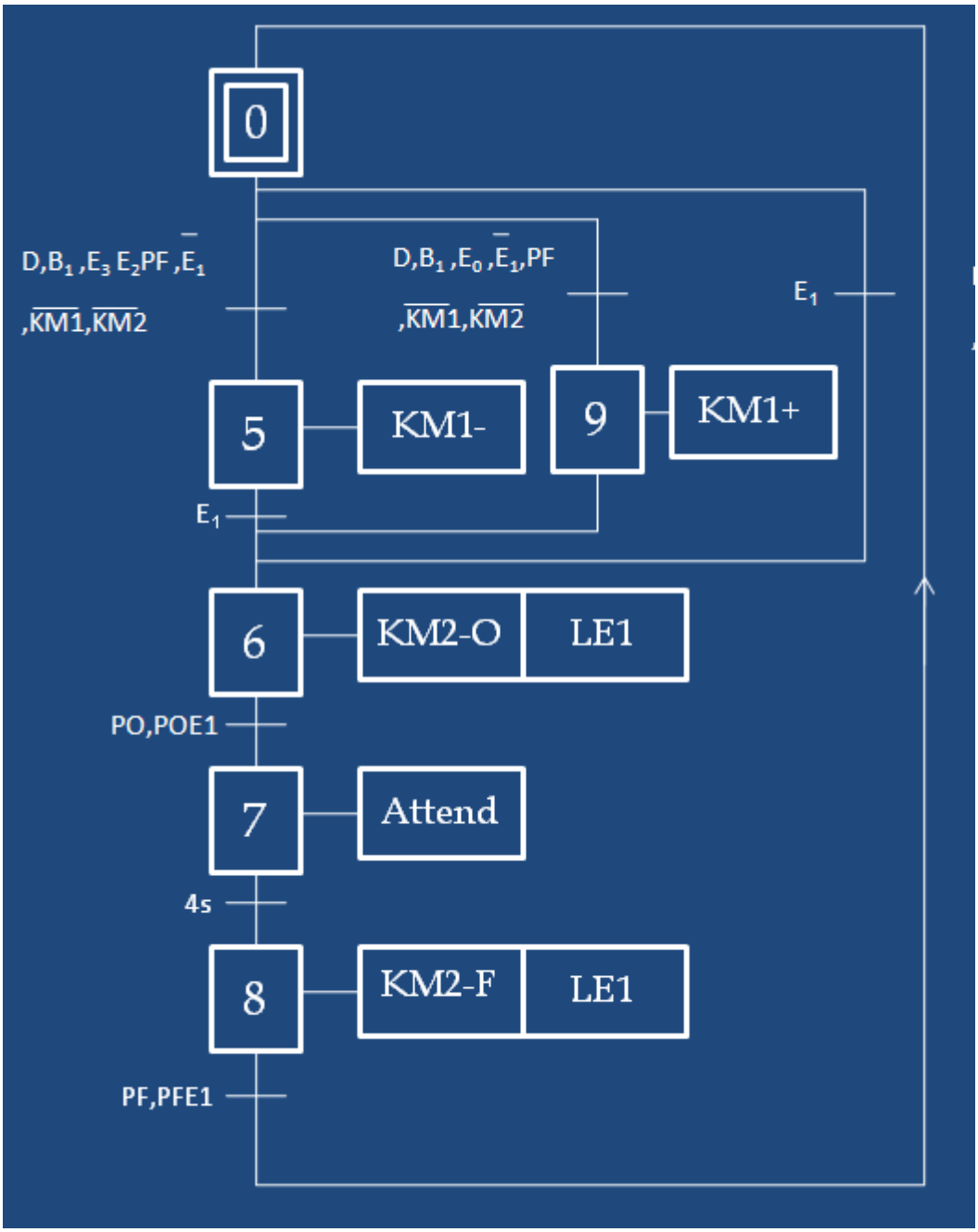


Figure4-3: l'étage 1 de l'ascenseur

- étage2 :

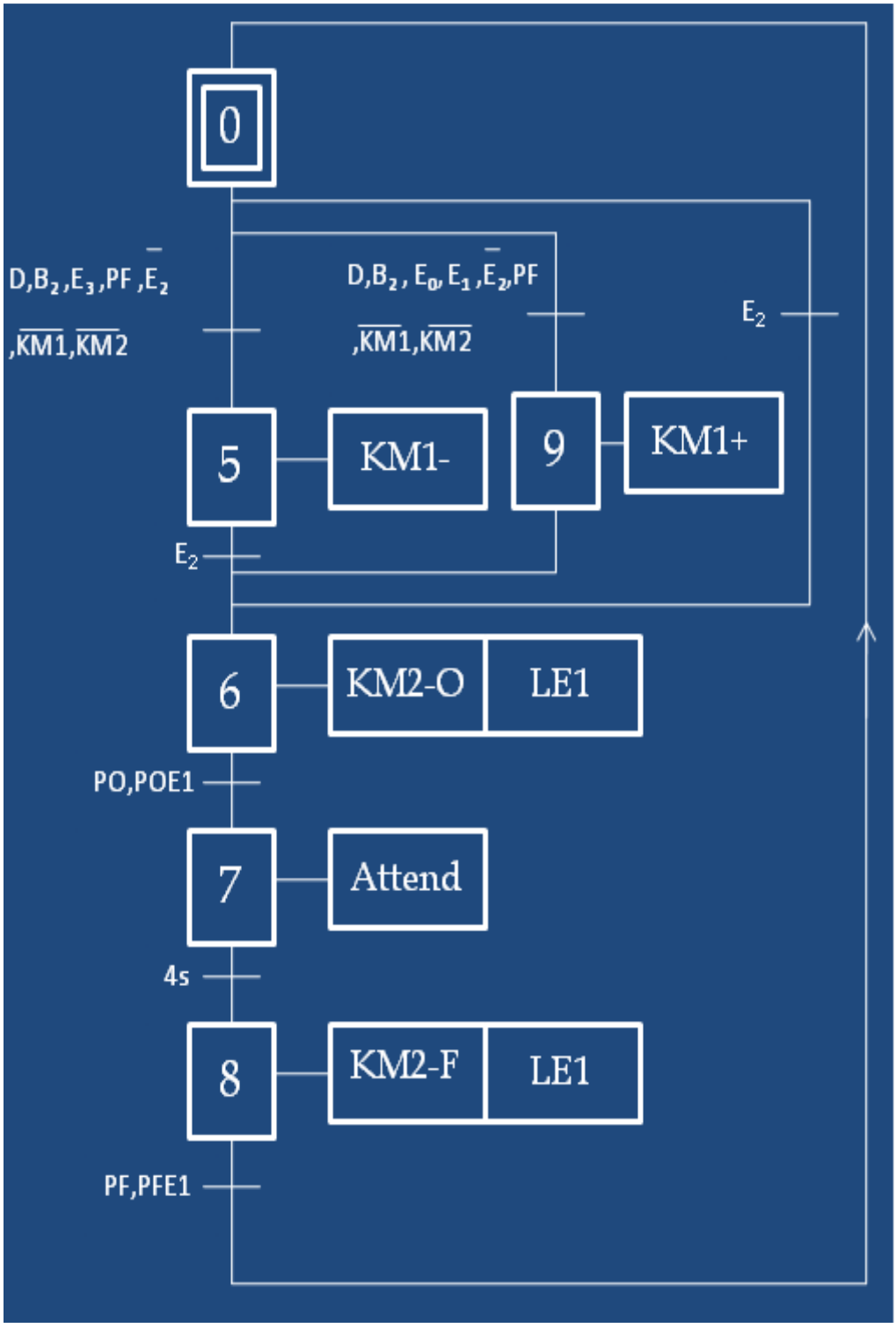


Figure4-4: l'étage 2 de l'ascenseur

- étage3 :

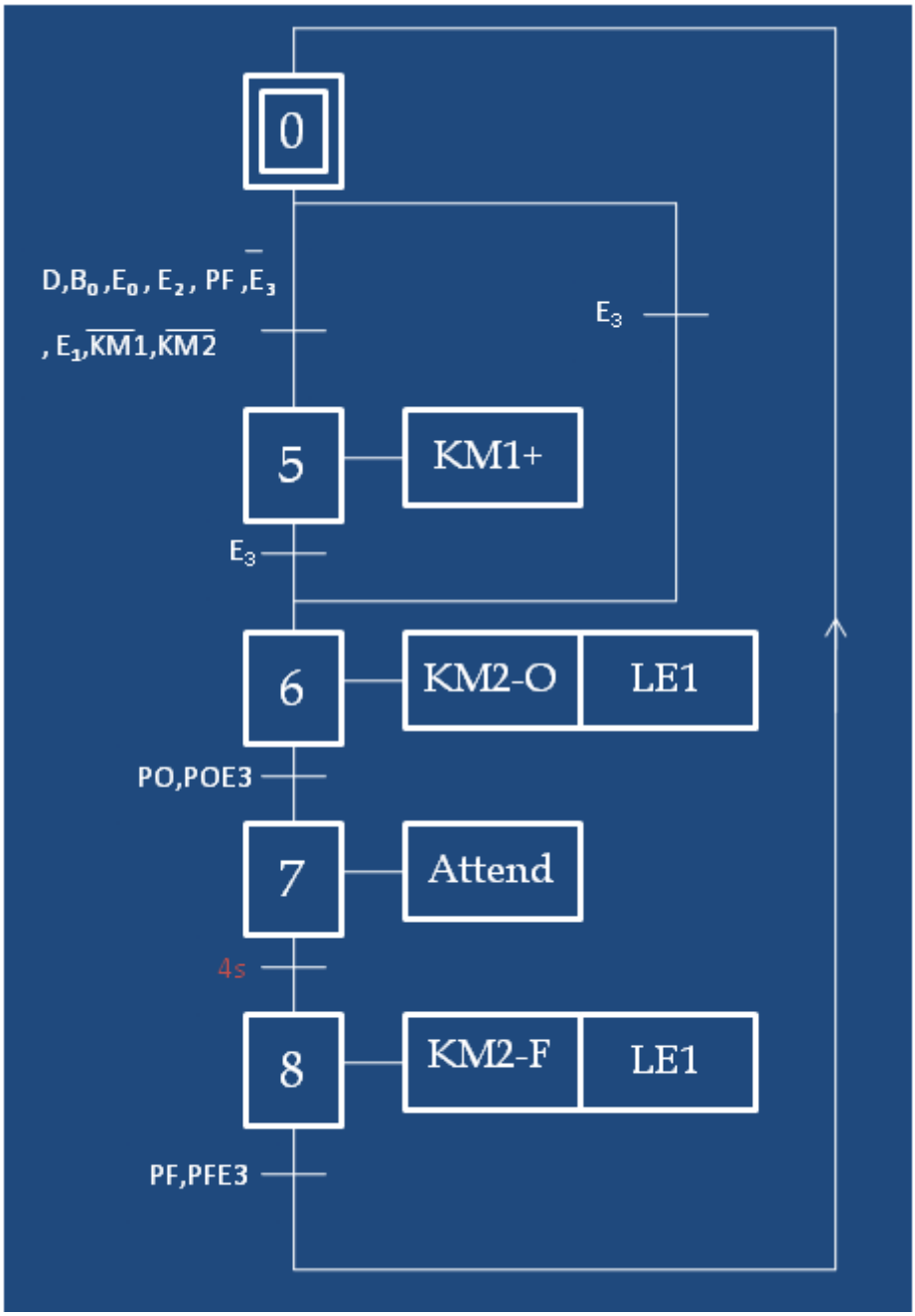


Figure4-5: l'étage 3 de l'ascenseur

4.3. Logiciel STEP7 :

4.3.1. Description :

STEP 7 est le progiciel de base pour la configuration et la programmation de systèmes d'automatisation SIMATIC. Il fait partie de l'industrie logicielle SIMATIC. Le progiciel de base STEP 7 existe en plusieurs versions :

* STEP 7-Micro/DOS et STEP 7-Micro/Win pour des applications autonomes simples sur SIMATIC S7 - 200.

* STEP 7 pour des applications sur SIMATIC S7-300/400, SIMATIC M7-300/400 et SIMATIC C7 présentant des fonctionnalités supplémentaires :

- Possibilité d'extension grâce aux applications proposées par l'industrie logicielle SIMATIC (voir aussi Possibilités d'extension du logiciel de base STEP 7).
- Possibilité de paramétrage de modules fonctionnels et de modules de communication.
- Forçage et fonctionnement multiprocesseur.
- Communication par données globales.
-Transfert de données commandé par événement à l'aide de blocs de communication et de blocs fonctionnels.
- Configuration de liaisons.[9]

4.3.2. Structure :

La programmation structurée permet la rédaction claire et transparente de programmes. Elle permet la construction d'un programme complet à l'aide de modules qui peuvent être échangés et/ou modifiés à volonté.

Pour permettre une programmation structurée confortable, il faut prévoir plusieurs types de modules : les modules d'organisation (OB), de programmes (FB), fonctionnels (FC), de pas de séquences (SB), de données (DB).

Les modules de programmes (FC) servent à subdiviser le programme en parties fonctionnelles et/ou orientées vers le "processus".

Les modules de données (DB) contiennent des données variables, textes, valeurs de temporisations ou de comptage, résultats de calculs, etc. et sont acessibles et actualisables à tout moment.

Les modules séquentiels (SB) sont spécialement utilisés pour effectuer des séquences selon Grafcet.

Les paramètres d'entrées y seront les conditions d'avancement d'un pas de séquence et les paramètres de sorties, les ordres à exécuter lorsque ces conditions seront vérifiées.

Les modules d'organisation (OB) sont, comme leur nom l'indique, utilisés pour l'organisation interne du programme et forment ainsi un moyen puissant et essentiel pour la programmation structurée.

Ils servent par exemple au déroulement cyclique du programme principal, à l'exécution de programmes d'interruption par des fonctions d'alarmes ou de temps, ou par des fonctions diagnostic interne autant du point de vue hardware que software du système complet.

Ce dernier point est surtout un élément essentiel pour des systèmes complexes.

Ainsi, une chute de tension, une défectuosité des cartes d'entrées/sorties, un dépassement du temps de cycle, des erreurs d'adressage, etc. peuvent être détectés, signalés et la réaction du système suite à ces défauts, peut être librement programmée.

Les modules fonctionnels (FB) sont librement paramétrables spécialement conçus pour la standardisation de fonctions complexes et revenant souvent.
S'il faut commander par exemple une cinquantaine de vannes à l'aide d'un automate, on ne programmera qu'une fois ce programme de commande et de surveillance de vannes avec des paramètres symboliques dans un module fonctionnel. Ensuite, on appellera 50 fois ce module dans le programme principal et à chaque fois on y adjoindra d'autres entrées et d'autres sorties étant donné le caratère de substition des paramètres.
En résumé on distingue plusieurs types de blocs :

- **Les blocs d'organisation : OB**

On retiendra principalement l'OB1 qui est examiné à chaque cycle d'automate. C'est donc à partir de ce bloc que l'on fera les appels aux différents blocs de programmes.
L'OB100 et l'OB101 sont uniquement appelés aux démarrage (respectivement à chaud et à froid). On y appellera donc les blocs traitant les initialisations.

- **Les fonctions : FC**

C'est dans ces blocs que l'on va mettre les instructions à exécuter. La numérotation est libre (de 0 à 255). Ces blocs n'ont pas de mémoire.

- **Les blocs de fonction : FB**

Ces blocs sont paramétrables. On peut passer des données en créant des DB d'instance associés à un seul FB pour le passage de paramètres. La numérotation est libre (de 0 à 255).
Ils peuvent être très utiles pour réduire le code en créant des DB d'instance associés à un seul FB avec passage de paramètres.

- **Fonctions systèmes SFC, les blocs fonctionnels systèmes SFB, les blocs fonctionnels de communication CFB.**[10]

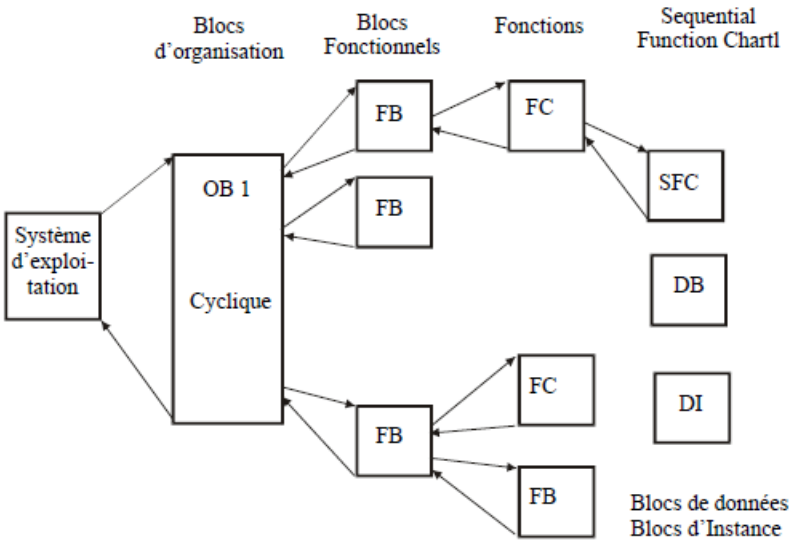


Figure4-6: Architecture des programmes en S7

Remarque : Pour mettre plus de clarté dans un programme, on le découpe en plusieurs sections affectées chacune à une fonction technologique. On est donc amené à programmer différents blocs (FB ou FC). L'ordre chronologique d'appel et de traitement des différents blocs est défini dans le bloc d'organisation (OB 1). Chaque bloc ainsi appelé peut lui-même contenir une instruction de saut vers un autre bloc. A la fin de l'exécution du bloc ainsi appelé, le traitement se poursuit automatiquement au lieu de départ du saut. De cette manière, il est possible d'obtenir une "imbrication" de 15 blocs.[10]

4.3.3. programmation command d'ascenseur par grafcet dans STEP7 :

Ici, nous allons incorporer le précédent GRAFCET de l'ascenseur à l'aide du programme STEP7 avec le choix de la logique programmée

Tableau4.2 : Tableau mnémonique en STEP7 .

Mnémonique	Opérande	Commentaire
BOT-ETAG0	E 124.0	Bouton pour sélection de l'étage 0
BOT-ETAG1	E 124.1	Bouton pour sélection de l'étage 1
BOT-ETAG2	E 124.2	Bouton pour sélection de l'étage 2
BOT-ETAG3	E 124.3	Bouton pour sélection de l'étage 3
Dcy	E 124.6	Bouton d'arrêt d'urgence
ETAGE0	E 125.0	capteur Cabine à l'étage 0
ETAGE1	E 125.1	capteur Cabine à l'étage 1
ETAGE2	E 125.2	capteur Cabine à l'étage 2
ETAGE3	E 125.3	capteur Cabine à l'étage 3
KM1-	A 124.0	La cabine descend
KM1+	A 124.1	La cabine monte
KM2-F	A 124.3	Ferme la porte
KM2-O	A 124.2	Ouvrir la porte
LED ETAG0	A 125.0	Signal du capteur Cabine à l'étage 0
LED ETAG1	A 125.1	Signal du capteur Cabine à l'étage 1
LED ETAG2	A 125.2	Signal du capteur Cabine à l'étage 2
LED ETAG3	A 125.3	Signal du capteur Cabine à l'étage 3
LED –attend	A 752.0	LED –attend
PORT-F	E 125.5	Porte Cabine sélectionnée se ferme
PORT-O	E 125.4	Porte Cabine sélectionnée s'ouvre
PORT,E0-O	E 126.0	Le porte de étage 0 sélectionnée s'ouvre
PORT,E1-O	E 126.1	Le porte de étage 1 sélectionnée s'ouvre
PORT,E2-O	E 126.2	Le porte de étage 2 sélectionnée s'ouvre
PORT,E3-O	E 126.3	Le porte de étage 3 sélectionnée s'ouvre
PORT,E0-F	E 126.4	Le porte de étage 0 sélectionnée se ferme
PORT,E1-F	E 126.5	Le porte de étage 0 sélectionnée se ferme
PORT,E2-F	E 126.6	Le porte de étage 0 sélectionnée se ferme
PORT,E3-F	E 126.7	Le porte de étage 0 sélectionnée se ferme.

Grafcet d'un ascansseur à 4 étages par Step 7 version 5.5

- étape initial :

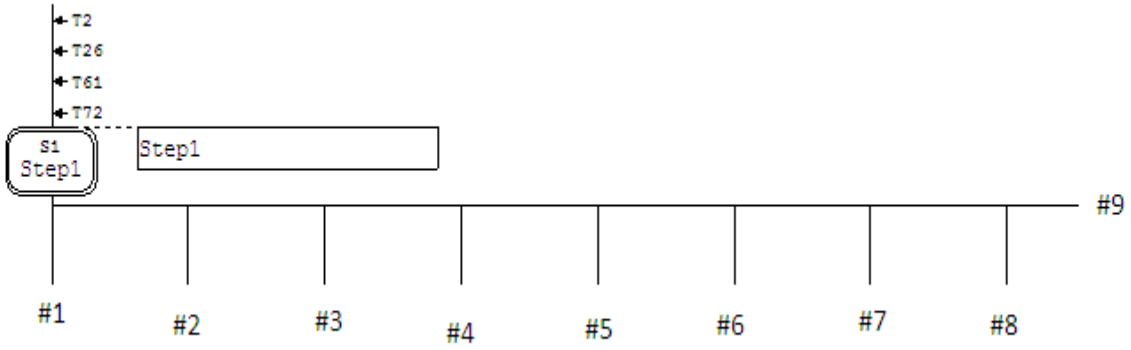


Figure4-7: GRAFCET d'étape initial (par STEP7)

- étage 0 :

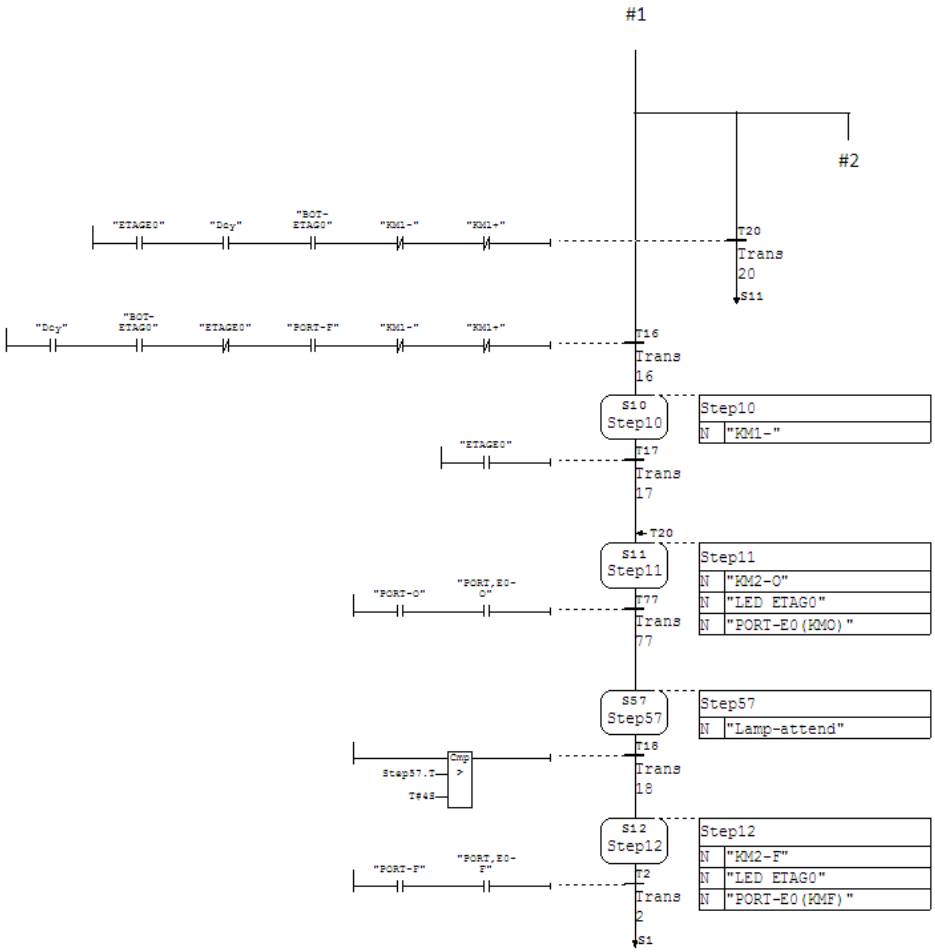


Figure4-8: GRAFCET d'étage 0 (par STEP7)

- étage1 :

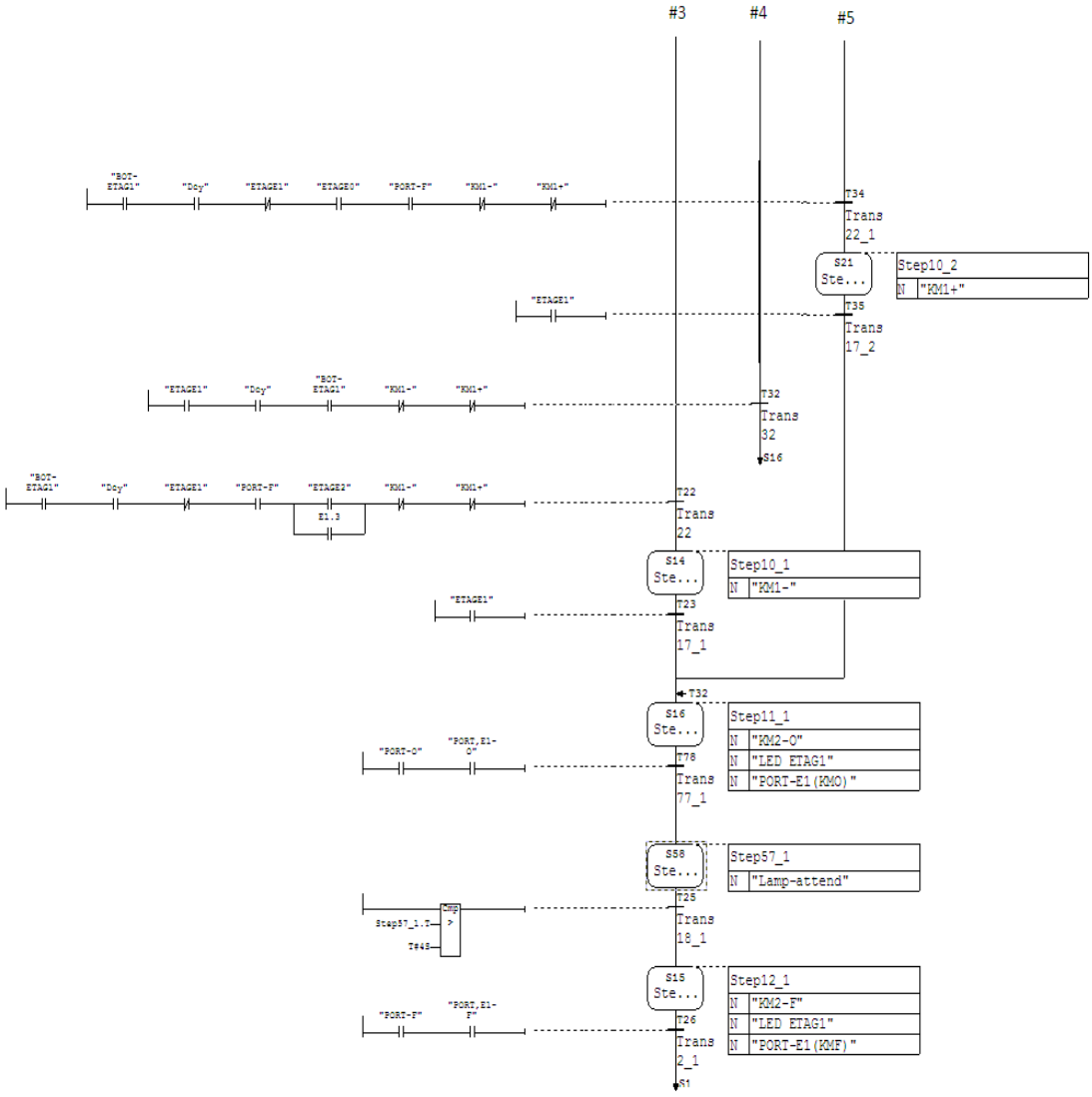


Figure4-9: GRAFCET d'étage 1 (par STEP7)

- étage2 :

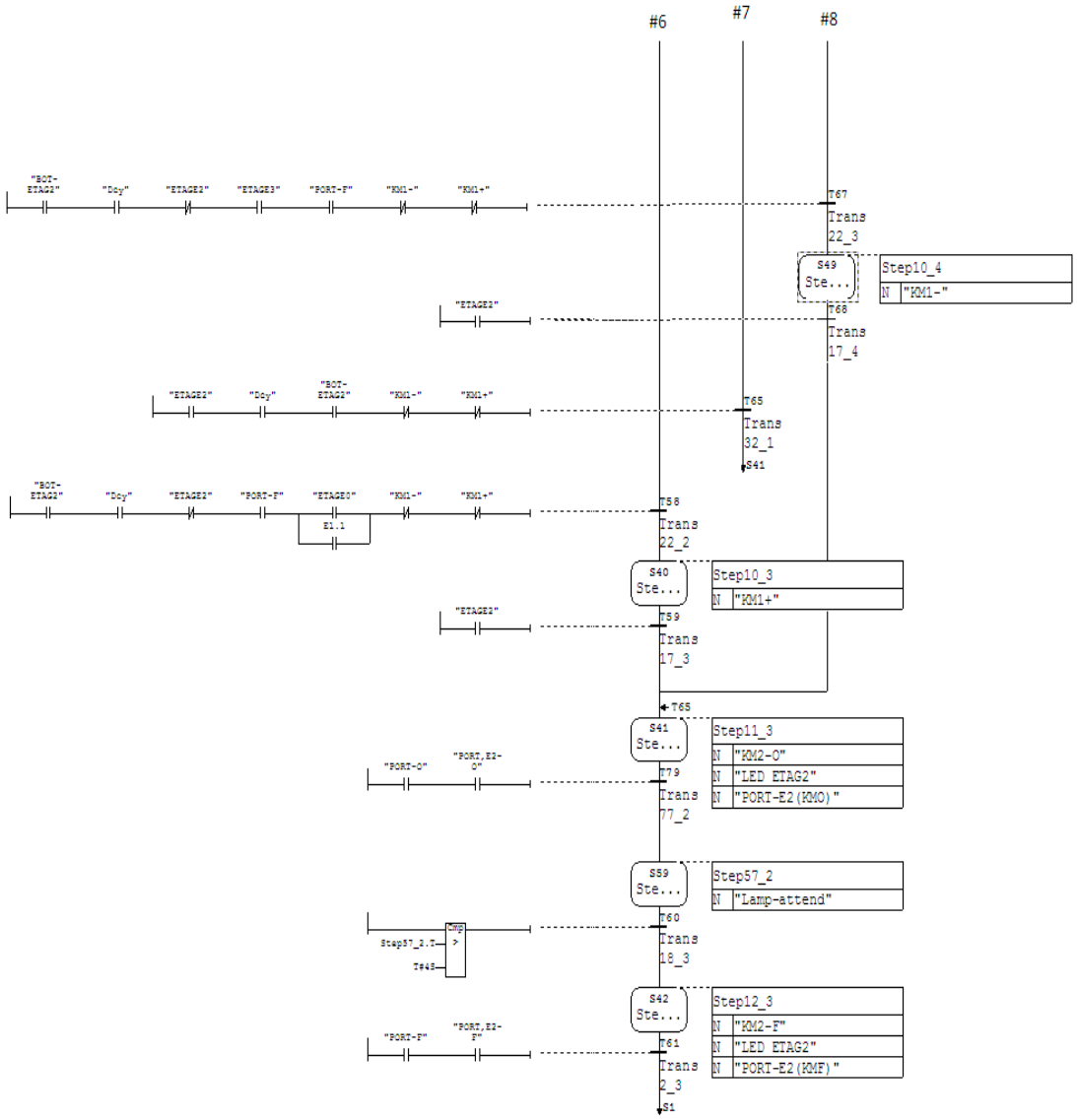


Figure4-10: GRAFCET d'étage 2 (par STEP7)

- étage3 :

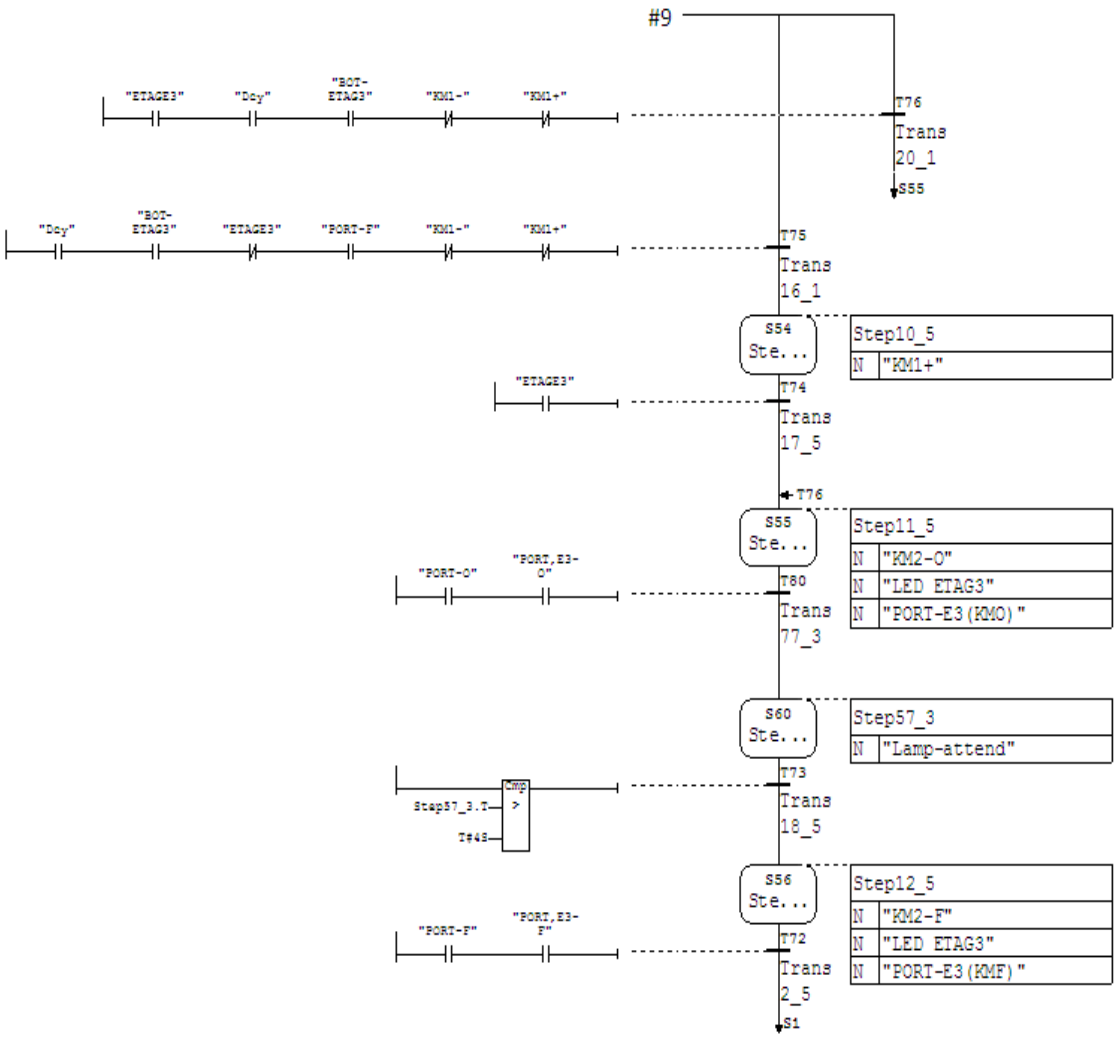


Figure4-11: GRAFCET d'étage 3 (par STEP7)

4.4. Conclusion :

Dans ce chapitre, nous avons développé un plan de contrôle pour un ascenseur de 4 étages utilisant le langage GRAFCET. Cette langage est la mieux adaptée pour contrôler des systèmes complexes tels que des ascenseurs et d'autres par rapport à d'autres langages car il se caractérise par une grande précision et une grande facilité de contrôle.

Conclusion générale

Conclusion Général :

L' API est un bon produit s'il est bien choisi et bien employé. Ce qui peut apparaître comme une lapalissade nous a amené à attirer l'attention sur des aspects parfois jugés triviaux, tels les types d'E/S, le dimensionnement des alimentations électriques, les modes d'exécution d'un programme, les limites des divers types de communication, car ce sont des points où sont parfois commises des erreurs qui entraînent des surcoûts d'installation ou limitent fâcheusement les performances obtenues.

Le choix d'un API est lié à l'environnement. Plus ce dernier est perturbé, plus les exigences en termes de sûreté de fonctionnement sont grandes, plus l'API s'impose face à des solutions concurrentes.

Dans cette étude, nous avons trouvé plusieurs résultats :

- Comme il est un système automatisé intéressant, pour adoption dans diverses technologies de l'information et de la communication dans de nombreux domaines pour cela, il est jugé utile, en plus de son usage général et complet, ce qui rend son développement à augmenter pour améliorer la sécurité des usagers.
- La langue d'un système automatisé particulier est sélectionnée selon le cahier de charge.
- parce que le chaire des charges est la meilleure solution au problème de la communication entre le client et le programmeur est sélectionné langage de programmation par la complexité.
- Nous avons pu réaliser une simulation de la partie commande de l'ascenseur à l'aide du programme STEP 7.

Ce projet de graduation nous a apporté une grande contribution à l'apprentissage de la maîtrise des nouvelles machines électriques et des systèmes automatisés et à la compréhension des langages de programmation généraux qui résolvent les sujets actuels et futurs.

Rappelez-vous, nous n'avons qu'un simulation, qui est dû au manque de tous les moyens, malgré nos tentatives, d'attente pour une mise en œuvre future.

Références Bibliographiques

[1] : " ETUDE D'ASCENSEUR COMMANDE PAR AUTOMATE PROGRAMABLE " , Université sidi Mohammed ben Abdallah Fasse , Année universitaire : 2006/2007.

[2]: sitelec.org/download.php?filename=cours/automates...industriels.pdf.

[3] : Alain GONZAGA , LES AUTOMATES PROGRAMMABLES INDUSTRIELS.

[4] : Mémoire de Master "Système de Contrôle Distribué (DCS) avec l'exploitation de l'automate programmable AC800 F (ABB) ", Université Mohamed Khider Biskra , Juin 2012.

[5] : Mémoire de Master "automatisation et realisation a petite echelle (maquette) d'une chaine transporteuse de briques ", Universite Hassiba Benbouali De Chlef , Juin 2016.

[6] : AUTOMATISME, REPRODUCTION INTERDITE ASSOCIATION OUVRIERE DES COMPAGNONS DU DEVOIR DU TOUR DE France , FÉVRIER 2004 .

[7] : Cours Complet sur le Grafcet & Exercices Corrigés

[8] : Sciences et Techniques Industrielles , Automatique et Informatique Industrielle

Génie Mécanique – Terminale.

[9] : SIEMENS SIMATIC Programmer avec STEP 7Manuel, Edition 03/2006 A5E00706945-01 .

[10] : "AUTOMATISATION ET ROBOTISATION DE LA PRODUCTION " , UNIVERSITÉ DE LIÈGE , Année académique 2009-2010 .