الجمهورية الديمقراطية الشعبية الجزائرية

**République Algérienne Démocratique et Populaire**


**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Ecole Supérieure des Sciences Appliquées d'Alger**

وزارة التعليم العالي والبحث العلمي
المدرسة العليا في العلوم التطبيقية بالجزائر


**Département du second cycle**

# Mémoire de Fin d'Etudes

**En vue de l'obtention du diplôme de MASTER**

Filière **: Electrotechnique**

Spécialité : **Traction électrique**

## Thème :

# API et GRAFCET


Présenté par : SIACI Maissa

Encadrée par : M. TEFFAHI Abdelkader


Soutenu le : 28/06/2022,

Devant le jury composé de :

| | | | |
|---|---|---|---|
| Président : | M. ARBID Mahmoud | MCB | ESSA Alger |
| Examinatrice : | Mme. DJELLOUL Imene | MCA | ESSA Alger |
| Encadreur : | M. TEFFAHI Abdelkader | MCA | ESSA Alger |

Monôme N° : 03/Master /TR/2022

# Remerciements

Au terme de ce travail, je désire d'abord et avant tout remercier le bon dieu qui m'a donné le courage, l'aptitude et le sérieux de mener ce travail.

Je tiens aussi à exprimer ma profonde gratitude à mon encadrant, Monsieur TEFFAHI Abdelkader pour tout le temps précieux qu'il m'a consacré dans le but de la réalisation de ce travail, et qui n'a jamais manqué de m'orienter et de me conseiller.

Mes remerciements les plus sincères vont à Madame DJELLOUL Imene et Monsieur ARBID Mahmoud, les membres du jury, qui m'ont fait l'honneur d'accepter de juger ce modeste travail, pour toute l'attention qu'ils vont prêter à l'évaluation de ce travail.

Je suis aussi reconnaissante envers tous les enseignants qui nous m'ont soutenue tout au long de nos études. Qu'ils trouvent ici l'expression de mes sentiments les plus respectueux.

# Dédicaces

*Je dédie ce travail*
*A ma famille qui m'a toujours soutenue et encouragée et à laquelle j'éxprime toute ma gratitude,*
*A ma mère disparue trop tôt mais qui a su m'apprendre beaucoup en peu de temps,*
*A mon père à qui je dois tout pour tous les sacrifices qu'il a consentis.*

# Résumé

Résumé

Dans tous les secteurs et applications, les fabricants utilisent des contrôleurs logiques programmables (API) pour surveiller et automatiser leurs processus de production, dans ce travail nous allons présenter une description détaillée des automates programmables industriels dans un premier temps, puis les différents langages de programmation de ces derniers.

Abstract

In all sectors and applications, manufacturers are using programmable logic controllers (PLC) to monitor and automate their production processes, in this work we will present a detailed description of industrial programmable logic controllers first, then we'll see the different programming languages used for PLC

ملخص

في جميع القطاعات والتطبيقات ، يستخدم المصنعون وحدات التحكم المنطقية القابلة للبرمجة (PLC) لمراقبة وأتمتة عمليات الإنتاج الخاصة بهم ، في هذا العمل سنقدم وصفًا تفصيليًا لوحدات التحكم المنطقية القابلة للبرمجة أولاً ، ثم لغات البرمجة المختلفة لهذه.

# Table des matières

# Table des figures

# Introduction Générale

L'objectif de l'automatisation des systèmes est de produire, en ayant recours le moins possible à l'homme, des produits de qualité et ce pour un coût le plus faible possible. C'est dans cet quête que sont apparu les automates programmables industriels, ils ont permit de répondre aux attentes de l'industrie. Les API ont le grand avantage que le même contrôleur de base peut être utilisé avec une large gamme de systèmes de contrôle. Pour modifier un système de commande, il suffit à un opérateur de saisir un jeu d'instructions différent. Il n'est pas nécessaire de recâbler. Le résultat est un système flexible et économique qui peut être utilisé avec des systèmes de commande dont la nature et la complexité varient assez largement.

les API sont destinés à des ingénieurs sans particulièrement de grandes connaissances en informatique ou en programmation c'est pourquoi le langage à contacts a été développé pour l'écriture des programmes, Ils sont ensuite convertis en code machine par un logiciel et sont utilisés par le microprocesseur d'un API.

Cette méthode d'écriture des programmes a été adoptée par la plupart des fabricants d'API, mais chacun a eu tendance à développer ses propres versions. C'est pourquoi une norme internationale a été adoptée pour les langages de programmation utilisées pour la programmation des automates. La norme, publiée en 1993, est la CEI 1131-3 (Commission électrotechnique internationale). Les langages de programmation CEI 1131-3 sont les schémas à contacts (LAD), la liste d'instructions (IL), les diagrammes fonctionnels séquentiels (SFC), le texte structuré (ST) et les diagrammes de blocs fonctionnels (FBD).

Dans ce travail, nous allons présenter les automates programmables industriels en détails, leur composants, leurs types et les langages utilisés pour la programmation de ces derniers et plus particulièrement le GRAFCET.

# Chapitre 1

# Automates programmables industriels

## 1.1 Introduction

Les progrès technologiques de ces dernières années ont entraîné le développement de l'automate programmable et une révolution conséquente de l'ingénierie de contrôle. De ce fait, aujourd'hui tous les aspects de l'industrie, de la production d'électricité à la peinture automobile en passant par l'emballage alimentaire, utilise des automates programmables pour améliorer la production.

Dans ce chapitre nous allons présenter les API et leur architecture en détail, nous aborderons aussi les APIs modulaires, et enfin les différents langages de programmation des APIs ; langage à contact, liste d'instructions, FDB, SQL et Grafcet.

## 1.2 Présentation

Un automate programmable industriel est une forme spéciale de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker des instructions et pour mettre en œuvre des fonctions telles que la logique, le séquençage,

la synchronisation, le comptage et l'arithmétique afin de contrôler les machines et les processus [1].

Les API sont particulièrement appréciés dans les milieux industriels pour pour leur robustesse, leur réactivité et leur simplicité en ce qui concerne la maintenance. Il sont notamment conçus pour être programmé par un personnel ayant une connaissance limitée des ordinateurs et des langages informatiques ainsi le programme de commande puisse être saisi à l'aide d'un langage simple, plutôt intuitif.

## 1.3 Architecture d'un API

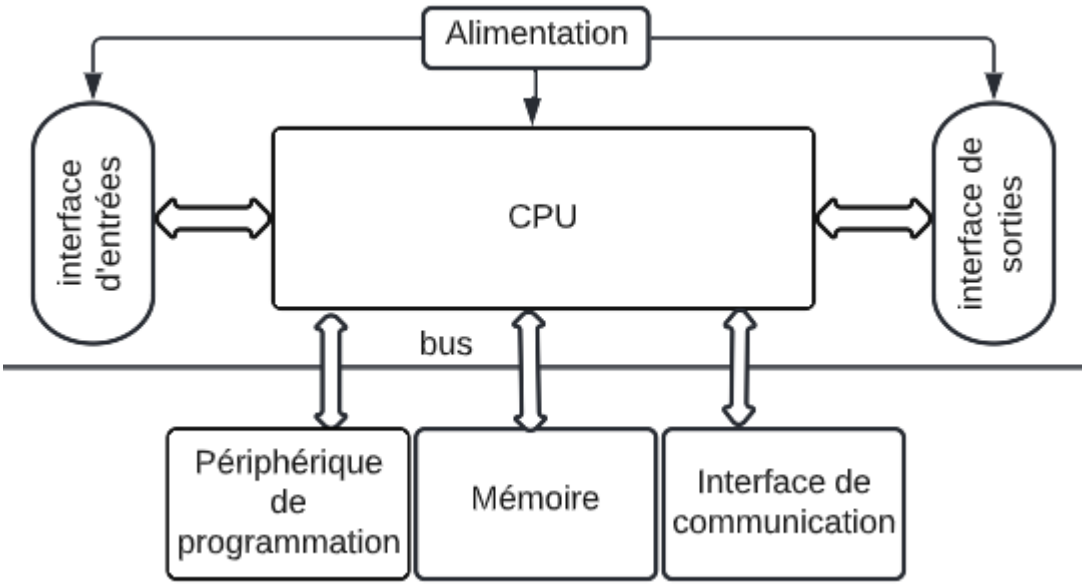


FIGURE 1.1 – Architecture d'un api.

### 1.3.1 CPU

Le processeur commande et régit les activités de l'ensemble de l'API. Ainsi Son rôle consiste à traiter les instructions qui constituent le programme de fonctionnement de l'application (les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul PID, etc..). mais aussi à gérer les entrées et sorties, à surveiller et diagnostiquer l'automate (par des tests lancés régulièrement), à mettre en place un dialogue avec le terminal de programmation. [2]

### 1.3.2 L'alimentation

Elle est indispensable au fonctionnement de l'API, elle est responsable de l'alimentation et de la gestion des besoins en énergie des différents composants de l'automate. Elle convertit une tension alternative en une basse tension continue (24V).

### 1.3.3 La mémoire

Elle est chargée de stocker et de récupérer des données. Deux familles de mémoires sont utilisées dans les automates programmables : les mémoires vives, ou mémoires à accès aléatoire « Random Access Memory (RAM) ». Le contenu de ces mémoires peut être lu et modifié à volonté, mais il est perdu en cas de manque de tension (mémoire volatile), les mémoires vives sont utilisées pour l'écriture et la mise au point du programme, et pour le stockage des données. les mémoires mortes « Read Only Memory (ROM) » sont à lecture seule, les informations ne sont pas perdues lors de la coupure de l'alimentation. Elles sont destinées à la mémorisation du programme après la phase de mise au point.

### 1.3.4 Les interfaces entrées/sorties

Elles permettent au processeur de recevoir et d'envoyer des informations. Ces dispositifs d'entrée et sortie peuvent produire des signaux discrets, numériques

(ce sont des sorties de type tout ou rien) ou analogiques. Les dispositifs qui génèrent des signaux discrets ou numériques sont ceux dont les sorties sont de type tout ou rien. Par conséquent, un interrupteur est un dispositif qui produit un signal discret : présence ou absence de tension. Les dispositifs numériques peuvent être vus comme des dispositifs discrets qui produisent une suite de signaux tout ou rien. Les dispositifs analogiques créent des signaux dont l'amplitude est proportionnelle à la grandeur de la variable surveillée [2].

### 1.3.5 Interface de communication

est utilisée pour recevoir et transmettre des données sur des réseaux de communication qui relient l'I à d'autres API distants ou à des équipements en fonction des protocoles de communication choisis. Elle est impliquée dans des opérations telles que l'acquisition de données, la synchronisation entres des applications et la gestion de la connexion.

### 1.3.6 Interface de programmation

C'est la plate-forme où le programme ou la logique de commande de l'automate sont écrit, il permet de charger le programme dans le contrôleur et surveiller et contrôler l'automate et son programme.

### 1.3.7 Les bus

C'est un ensemble de conducteurs qui réalisent la liaison entre les différents éléments de l'automate. Les informations sont transmises sous forme binaire ; on retrouve plusieurs types de bus :

- Bus de données : Le bus de données transporte les données utilisées dans les traitements effectués par le CPU.
- Bus d'adresses : Le bus d'adresses est utilisé pour transporter les adresses des emplacements de mémoire. Afin que chaque mot puisse être localisé

dans la mémoire, chaque emplacement mémoire reçoit une adresse unique. ainsi les données stockées à un emplacement particulier pourront être consultées par le CPU soit pour lire les données qui s'y trouvent, soit pour y écrire des données.

- Le bus de commande transporte les signaux utilisés par la CPU pour le contrôle.
- Le bus système est utilisé pour les communications entre les ports d'entrée/sortie et l'unité d'entrée/sortie [1].

## 1.4 Les API compact et modulaires

Les systèmes API sont principalement disponibles sous deux formes : en boitier unique et en version modulaire/rack.

### 1.4.1 Les API compacts

Les systèmes non modulaires ou compacts ont un nombre d'entrées sorties fixe (entre 10 et 30 E/S), avec souvent des performances limitées, ce sont les gammes les moins onéreuses, on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...). 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.

### 1.4.2 Les API modulaires

La majorité des installations comporte une solution modulaire illustrée dans la figure 1.2 [2], permettant grâce à des extensions d'étendre les E/S de l'automate

ainsi que les interfaces de communications. Un API modulaire est constitué de modules séparés pour : I'alimentation, le processeur, les entrées/sortie, les interfaces de communication. Les modules sont branchés les uns à la suite des autres dans un rack. Il suffit d'insérer un module sur le rack et de le configurer dans le logiciel pour l'ajouter, le rack de fond fournit le bus de communication et l'alimentation du module [1]. Un automate modulaire possède un module distinct pour

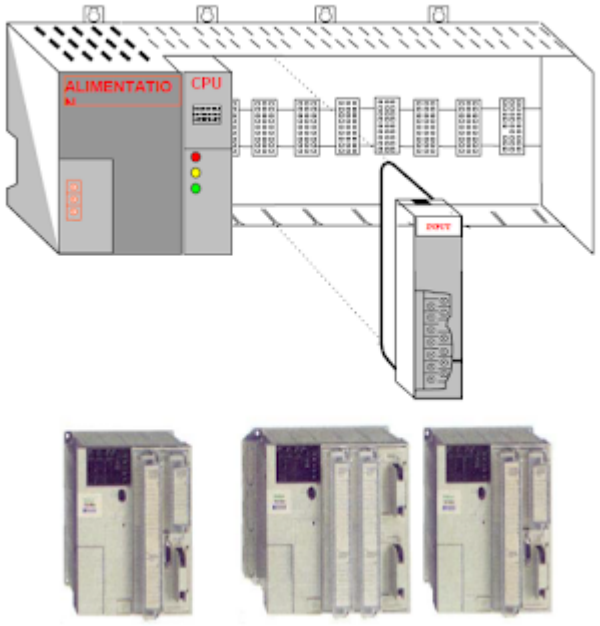


FIGURE 1.2 – API modulaire.

chacun de ses composants matériels. Le module exécute une fonction spécifique conformément à l'architecture de l'API. La conception de ces modules varie selon les fabricants et ne peut généralement pas être échangée entre les fabricants d'API. Les APIs modulaires sont utilisés pour automatiser les applications industrielles où un processeur plus puissant et un grand nombre de dispositifs d'entrée et de sortie sont nécessaires. Ces types d'applications qui utilisent des automates modulaires sont généralement associés à un niveau de complexité plus élevé en ce qui concerne l'exploitation, le contrôle et la surveillance des processus. Les

différents sous-ensembles sont :

- Le rack/Châssis : Certains châssis sont connus pour un montage en fond de panier, tandis que d'autres le sont pour un montage en rack. Il existe aussi des automates modulaires, sans châssis, ou les modules se connectent les uns aux autres.
- Unité centrale : Les processeurs sont disponibles dans plusieurs variantes en termes de capacité d'E/S, de mémoires. Certaines unités peuvent comporter une ou plusieurs interfaces de communication, ainsi que quelque entrée/sortie. Un connecteur de communication permet la programmation de l'automate. L'unité centrale comporte régulièrement un mémoire externe (carte SD) et une pile pour garder la sauvegarde.
- Entrées-sorties TOR : Modules d'entrées/sortie digitale (tout ou rien), il existe en version 8, 16, 32 ou 64 E/S. Il existe avec des variantes de tensions, ou de courant admissibles. Les sorties peuvent être à commande à relais (puissance), ou à transistors.
- Entrées-sorties analogiques : Les modules d'entrées-sorties analogiques réalisent les conversions A/N et N/A, avec une résolution allant jusqu'à seize bits. Nous avons plusieurs gammes de tension/courant disponibles, les plus utilisés étant le 0-10V et le 4-20mA.
- Modules de communication : Des modules de communication peuvent être utilisés pour augmenter le nombre de ports de communication du processeur ou utiliser d'autres protocoles de communications.
- Module métier : les fabricants proposent des modules pour une utilisation plus spécifique ; comme le commande d'axe permettant d'assurer le positionnement avec précision d'élément mécanique selon un ou plusieurs axes, le comptage rapide permettant d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate ; la mesure de température.

- Extensions : il est possible d'étendre son rack par un autre rack, des entrées/sortie déportées par un bus de communication, ou un autre automate en esclaves [2].

## 1.5 Les langages de programmation

Au fur et à mesure que les API se sont développés et étendus, les langages de programmation se sont développés avec eux. Les langages de programmation permettent à l'utilisateur d'entrer un programme de contrôle dans un automate en utilisant une syntaxe établie. Les langages avancés d'aujourd'hui ont de nouvelles instructions plus polyvalentes, qui déclenchent les actions du programme de contrôle [1]. Basés sur la norme « International Electrotechnical Commission (IEC) », les langages de programmation des API sont classés en cinq normes principales.

### 1.5.1 Schéma à contacts (LADDER)

Il ressemble au schéma d'un circuit électrique comme illustré dans la figure 1.3 [3], d'ailleur les schémas à contacts originaux ont été établis pour représenter des circuits logiques câblés utilisés pour contrôler des machines ou des équipements. En raison de leur large utilisation dans l'industrie, ils sont devenus un moyen standard de communiquer les informations de contrôle des concepteurs aux utilisateurs de l'équipement. Les sorties sont obligatoirement à droite du réseau On doit évidemment identifier les E/S, soit directement par leur code (Ia.b / Qa.b), ou avec leur libellé en clair défini dans la table des mnémoniques. On relie les éléments en série pour la fonction ET, en parallèle pour le OU. On peut utiliser des bits internes (peuvent servir en bobines et interrupteurs) leur code étant de type (Ma.b). [4]

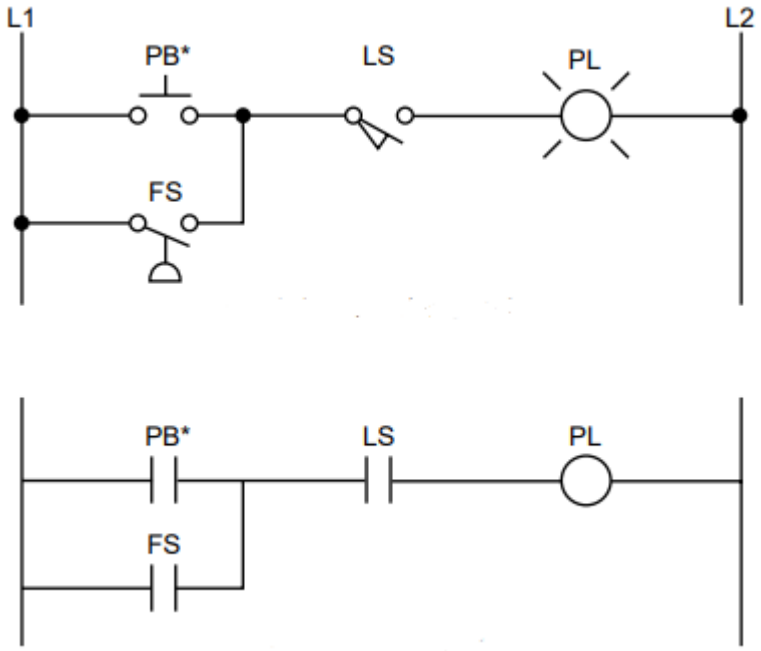


FIGURE 1.3 – Circuit logique câblé et son implémentation en langage à contacts API.

### 1.5.2 Liste d'instructions ou IL

C'est un langage textuel de bas niveau (figure 1.4 [3]), il est utilisé par certains fabricants, il utilise la syntaxe de l'algèbre booléen pour saisir et expliquer la logique de commande, C'est-à-dire qu'il utilise les fonctions logiques AND, OR et NOT pour implémenter les circuits de commande dans le programme de commande, la syntaxe de ce langage de programmation est donc facile à retenir.

Il a pour avantage une vitesse d'exécution élevée et Il occupe moins de mémoire par rapport aux autres langages de programmation.

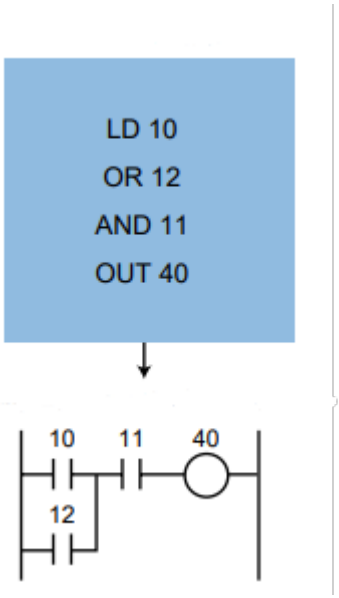


FIGURE 1.4 – Schéma à contacts et sa représentation en liste d'instructions.

### 1.5.3 Schéma par blocs ou FBD

Ce langage permet de programmer graphiquement à l'aide de blocs comme montré dans la figure 1.5 [5], représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.

Un bloc fonctionnel est représenté comme une boîte qui se compose d'un certain nombre de lignes de code pour mettre différentes fonctions de programmation. lorsqu'il est exécuté, il produit une ou plusieurs valeurs de sortie. La Figure ci dessous montre la représentation d'un bloc. Le nom de la fonction est écrit dans la boîte.

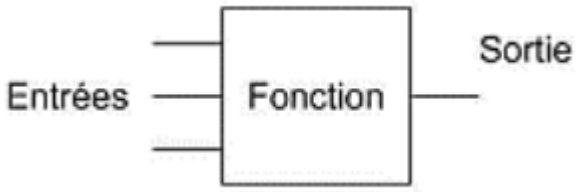


FIGURE 1.5 – Bloc fonctionnel.

### 1.5.4 Texte structuré ou SQL

Ce langage est un langage textuel de haut niveau. Il permet programmation de tout type d'algorithme plus ou moins complexe. Il ressemble énormément au langage Pascal. Les programmes sont écrits sous forme d'une suite d'instructions séparées par des points-virgules. Il s'agit d'instructions prédéfinies et de sous routines qui permettent de modifier des variables, celles-ci étant des valeurs définies, des valeurs mémorisées de manière interne ou des entrées et des sorties.

### 1.5.5 Graphe de fonction séquentielle (grafcet)

Le terme graphe de fonction séquentielle (SFC, Sequential Function Chart) correspond à une représentation graphique du fonctionnement d'un système afin de révéler l'enchaînement des événements qui conduisent à ce fonctionnement. Ce langage de programmation de haut niveau permet la programmation aisée de tous les procédés séquentiels.

Les graphes de fonction séquentielle représentent une technique graphique puissante pour décrire le comportement séquentiel d'un programme [5]. La figure 1.6 [5] représente un Graphe de fonction séquentielle et schéma à contacts équivalent.

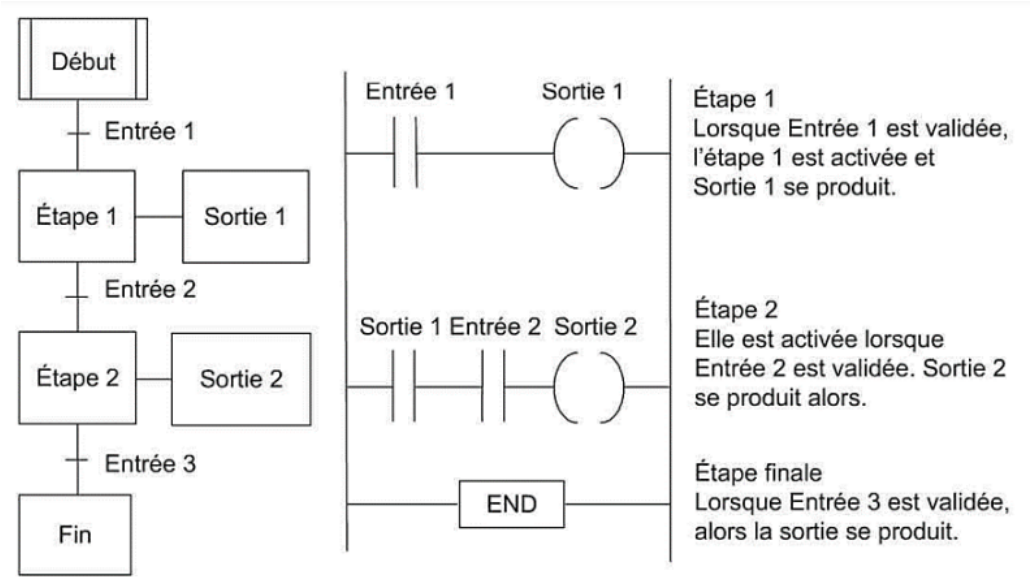


FIGURE 1.6 – Graphe de fonction séquentielle et schéma à contacts équivalent.

## 1.6 Critères de choix d'un automate

Le choix d'un automate programmable se fait en tenant compte de la partie commande, il est généralement basé sur :

- Nombre d'entrées / sorties intégrés
- Type de processeur : la taille mémoire, temps de traitement et les fonctions

spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.

- Fonctions ou modules spéciaux : certaines cartes (commande d'axe, pesage ...) permettront de "soulager" le processeur et devront offrir les caractéristiques souhaitées (résolution, ...).
- Fonctions de communication : l'automate doit pouvoir communiquer avec les autres systèmes de commande et offrir des possibilités de communication avec des standards normalisés (Profibus ...).
- Nombre de compteurs et nombre de temporisateurs Le choix d'un API est fonction de la partie commande à programmer. On doit tenir compte de plusieurs critères.

## 1.7 Types de réseaux d'automates

Un réseau industriel est constitué d'automates programmables, des interfaces hommes/machines, d'ordinateurs, des équipements d'entrées/sorties, reliés entre eux grâce à des lignes de communication, telles que des câbles électriques, des fibres optiques, des liaisons radio et des éléments d'interface, tels que des cartes réseaux, des gateways. L'arrangement physique du réseau est appelé topologie physique ou architecture du réseau.

Lorsque l'on considère la circulation des informations, on utilise la terminologie de topologie logique.

### 1.7.1 Réseau en bus

Cette organisation représentée dans la figure 1.7 [6] est une des plus simples. Tous les éléments sont reliés à une même ligne de transmission par l'intermédiaire de câbles. Le mot bus désigne la ligne physique. Cette topologie est facile

à mettre en oeuvre, la défaillance d'un noeud ou d'un élément ne perturbe pas le fonctionnement des autres organes [6].

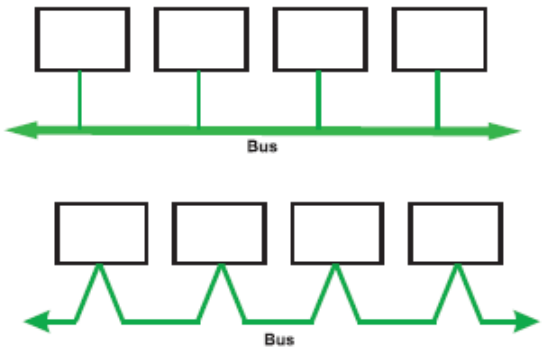


FIGURE 1.7 – Réseau en bus.

### 1.7.2 Réseau en étoile

Dans la configuration en étoile représentée dans la figure 1.7 [4], un centre de traitement commun échange avec chacune des autres stations. Deux stations ne peuvent pas échanger directement entre elles,[4] il présente l'avantage de procurer une grande vitesse d'échange et d'être très souple en matière de gestion et de dépannage. néanmoins il a un coût global élevé.

La défaillance d'un noeud ne perturbe pas le fonctionnement global du réseau, en revanche, l'équipement intermédiaire qui relie tous les noeuds constitue un point unique de défaillance.

### 1.7.3 Réseau en anneau

Dans la configuration en anneau représentée dans la figure 1.8 [4], chaque station peut communiquer avec sa voisine. Cette solution est intéressante lorsqu'une station doit recevoir des informations de la station précédente ou en transmettre vers la suivante.ref4

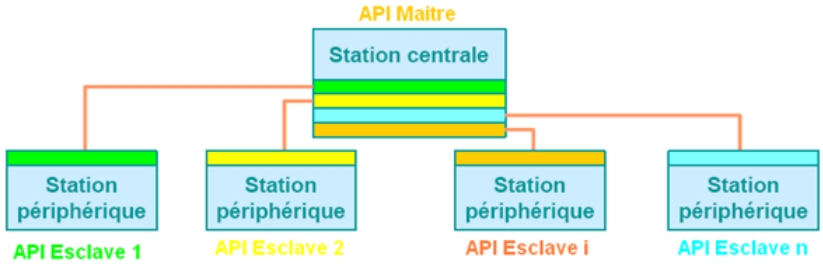


FIGURE 1.8 – Réseau en étoile.


FIGURE 1.9 – Réseau en anneau.

## 1.8 Conclusion

Nous avons étudié dans ce chapitre les APIs, leurs composants, et les différents langages utilisés pour leur programmation. Nous avons vu que les automates programmables industriels sont des contrôleurs industriels matures dont les racines de conception sont basées sur les principes de simplicité et d'application pratique. Garce a différents langages de programmation qui étendent les possibilités de programmation dans des domaines tels que la manipulation de données, la communication réseau, le transfert de données et les contrôles de programme, pour n'en nommer que quelques-uns.

Dans le chapitre suivant nous allons aborder plus en détails le GRAFCET.

# Chapitre 2

# Grafcet

## 2.1 Introduction

Le GRAFCET (acronyme de « GRAphe Fonctionnel de Commande Etapes/Transitions » est un mode de représentation et d'analyse d'un automatisme, particulièrement bien adapté aux systèmes à évolution séquentielle, c'est-à-dire décomposable en étapes. C'est un outil graphique puissant, directement exploitable, car c'est aussi un langage pour la plupart des API existant sur le marché.

Dans ce chapitre nous allons présenter un bref historique du GRAFCET, nous verrons ensuite une description du système automatisé, puis du cahier des charges de la partie commande de ce dernier. On présentera le GRAFCET, ses différents types et éléments, et finalement les règles d'évolution du GRAFCET.

## 2.2 Bref historique du GRAFCET

Les automates programmables sont utilisés pour le contrôle industriel depuis leur développement au début des années 1970. Le langage de programmation API, appelé Ladder, a été conçu pour ressembler à des schémas logiques de commande de relais à deux fils. Cette similitude était nécessaire au début des applications API car les électriciens chargés de la maintenance des systèmes connaissaient les

schémas de relais [7].

En conséquence, l'intégration des automates programmables dans les applications de contrôle industriel a été un tel succès que les automates programmables ont été utilisés dans des usines industrielles de toutes tailles. Cependant, le succès retentissant des automates programmables et du langage LADDER avait quelques inconvénients ; notamment l'absence de normes à l'échelle de l'industrie pour la syntaxe des programmes Ladder, de plus les programmes ne sont pas interchangeables entre les automates de différents fournisseurs.

Ces dernières années, de nombreux outils de description du cahier de charges d'un système automatisé sont apparus. Certains, établis par des chercheurs (Réseau de Petri, etc.) s'appuient sur d'importants travaux théoriques. D'autres créés par des industriels insistent davantage sur la mise en œuvre (diagramme de Girard, organigrammes, norme DIN 40719, etc.) [8].

L'AFCET, (Association Française pour la Cybernétique Économique et Technique) par l'intérimaire de son groupe de travail "Systèmes Logiques", a entrepris depuis 1975 une importante réflexion sur l'unification de la représentation du cahier des charges d'un automatisme logique.

En 1976, le GRAFCET qui est un diagramme fonctionnel dont le but est de décrire graphiquement les différents comportements d'un automatisme séquentiel est créé par l'AFCET, Sa promotion a été faite par l'ADEPA (Agence pour le DEveloppement de la Production Automatisée) puis a été acceptée par les instances internationales de normalisation, notamment par le Comité Électrotechnique International.

## 2.3 Cahier des charges de la partie commande d'un système automatisé

Le cahier des charges d'un automatisme est la description de son comportement en fonction de l'évolution de son environnement.

L'automaticien chargé de la conception et de la réalisation de la partie com-

mande doit rechercher dans le cahier des charges une description claire, précise, sans ambiguïtés ni omission, du rôle et des performances de l'équipement à réaliser. Pour y parvenir, il est souhaitable de diviser la description en deux niveaux successifs et complémentaires ; un premier niveau qui décrit le comportement de la partie commande vis-à-vis de la partie opérative c'est-à-dire les spécifications fonctionnelles, et un deuxième niveau qui ajoute aux exigences fonctionnelles les précisions indispensables aux conditions de fonctionnement des matériels, grâce aux spécifications technologiques et opérationnelles.

### 2.3.1 Niveau 1 : Spécifications fonctionnelles

C'est le premier niveau de la description, les spécifications fonctionnelles caractérisent les réactions de l'automatisme face aux informations issues de la partie opérative, dans le but de faire comprendre au concepteur quel devra être le rôle de la partie commande à construire. Elles doivent donc définir de façon claire et précise les différentes fonctions, informations et commandes impliquées dans l'automatisation de la partie opérative, sans préjuger en aucune façon des technologies. En conséquence, ni la nature ni les caractéristiques des différents capteurs ou actionneurs utilisés n'ont leur place dans ces spécifications. Peu importe, à ce niveau, que l'on effectue un déplacement à l'aide d'un vérin hydraulique ou pneumatique, ou encore d'un moteur électrique, Ce qu'il faut savoir c'est dans quelles circonstances ce déplacement doit s'effectuer [8].

### 2.3.2 Niveau 2 : Spécifications technologiques et opérationnelles

#### 2.3.2.1 Spécifications technologiques

Les spécifications technologiques précisent la façon dont l'automatisme devra physiquement s'insérer dans l'ensemble que constitue le système automatisé et son environnement. Ce sont les précisions à apporter en complément des spécifications fonctionnelles pour que l'on puisse concevoir un automatisme pilotant

réellement la partie opérative. C'est à ce niveau seulement que doivent intervenir les renseignements sur la nature exacte des capteurs et actionneurs employés, leurs caractéristiques et les contraintes qui peuvent en découler [8].

#### 2.3.2.2 Spécifications opérationnelles

Les spécifications opérationnelles sont relatives au suivi de fonctionnement de l'automatisme après sa mise en exploitation, tout au cours de son existence. Notamment sa fiabilité, l'absence de pannes dangereuses, disponibilité, et les possibilités de modification de l'équipement en fonction des transformations de la partie opérative.

## 2.4 Définition du GRAFCET

Le GRAFCET (acronyme de « GRAphe Fonctionnel de Commande Etapes/Transitions » est un diagramme fonctionnel pour la description précise des systèmes séquentiels. Il permet de représenter par un graphe le fonctionnement d'un système automatisé.

Le GRAFCET est un outil de représentation d'un cahier des charges, il normalisé, dépourvu d'ambiguïtés et facile à comprendre et à utiliser contrairement à d'autres langages.

C'est un mode de représentation et d'analyse d'un automatisme, particulièrement bien adapté aux systèmes à évolution séquentielle, c'est-à-dire décomposable en étapes. Un des points forts du Grafcet est la facilité de passer du modèle à l'implantation technologique de celui-ci dans un automate programmable industriel. Le Grafcet passe alors du langage de spécification au langage d'implémentation utilisé pour la réalisation de l'automatisme. On parle ainsi de Grafcet de spécification (niveau 1) et de Grafcet de réalisation (niveau 2).

## 2.5 Types de GRAFCET

Il existe trois types du GRAFCET qui sont :

- GRAFCET de niveau 01

  Ce type de GRAFECT est basé sur la représentation de toutes les parties du système automatisé avant l'existence de ce dernier (système automatisé). Par ailleurs le GRAFCET de niveau 1 est un GRAFCET de coordination des données et des actions [9].

  Il prend en compte la partie fonctionnelle, en faisant abstraction de toute réalisation technologique

- GRAFCET de niveau 02.

  Ce type de GRAFCET est basé sur la technologie des actionneurs (moteurs électriques, vérins, . . .etc.) et capteurs, ces derniers nous permettent de réaliser un diagramme séquentiel qui définit le comportement de la partie commande d'un système automatisé [9].

  Ainsi en s'appuyant sur le GRAFCET fonctionnel, ce GRAFCET intègre les contraintes technologiques et opérationnelles, la figure 2.2 [9] représente le passage d'un GRAFCET de niveau 1 à un GRAFCET de niveau 2.

## 2.6 Éléments du GRAFCET

### 2.6.1 Étape

Une étape correspond à une situation dans laquelle le comportement de tout ou partie du système par rapport à ses entrées et ses sorties est invariant, la figure 2.3 [8] représente une étape active dans un GRAFCET. Tout changement de comportement provoque le passage d'une étape à une autre. Ainsi une étape est soit active ou inactive, et les actions associées à cette étape ne sont effectives que

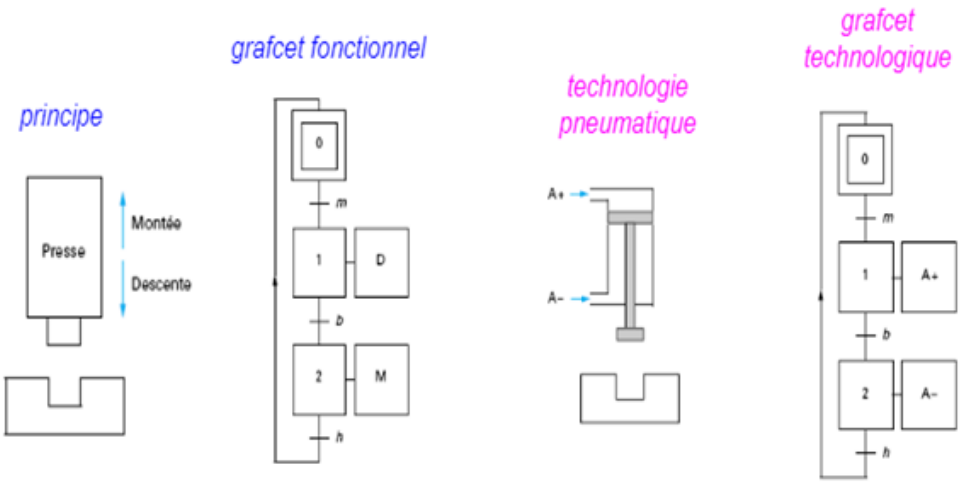


FIGURE 2.1 – Passage d'un GRAFCET de niveau 1 à un GRAFCET de niveau 2.

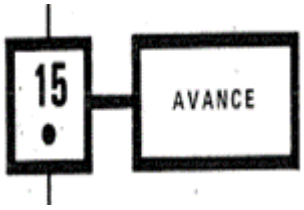


FIGURE 2.2 – Représentation d'une étape active dans un GRAFCET.

lorsque l'étape est active. [10] L'étape est représentée par un carré ou un rectangle repéré numériquement, un nom symbolique peut être adjoint, représentatif de la fonction principale de l'étape (ex : attente, fin, synchronisation, etc.) [8].

Il est possible de montrer que les étapes actives à un instant bien précis en plaçant un point ou un repère quelconque dans la partie inférieure des symboles correspondants.

Au niveau des spécifications technologiques (niveau 2), on devra préciser la façon dont les actions sont réalisées compte tenu du matériel défini pour les capteurs et les actionneurs. La prise en compte de ces nouvelles spécifications peut amener à modifier le GRAFCET de niveau 1.

### 2.6.2 Transition

La transition permet le passage d'une étape à une autre. Elle n'est que logique (dans son sens Vrai ou Faux), sans notion de durée. La condition est définie par une réceptivité qui est généralement une expression booléenne (c.à.d avec des ET et des OU) de l'état des capteurs [10].

On représente une transition par un petit trait horizontal sur une liaison verticale comme illustré dans la figure 2.4 [8]. On note à droite la réceptivité, on peut noter à gauche un numéro de transition (entier positif, indépendant des numéros d'étapes).

La réceptivité, écrite sous forme de proposition logique, est une fonction combinatoire d'informations extérieures (directives de l'opérateur, états de capteurs, de compteurs, de temporisateurs, etc.), de variables auxiliaires ou de l'état actif ou inactif d'autres étapes.

### 2.6.3 Liaisons

Les liaisons orientées modélisent la structure du Grafcet. Elles relient les étapes aux transitions et les transitions aux étapes. Le sens général de parcours est du haut vers le bas. L'arrivée et le départ sur une étape sont représentés verticalement,

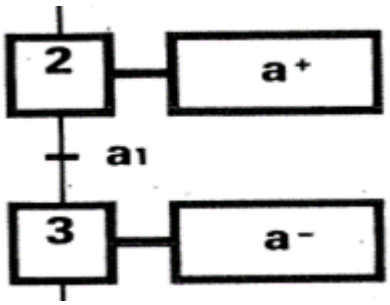


FIGURE 2.3 – Représentation d'une transition de passage de l'étape 1 à l'étape 2 dans un GRAFCET.


FIGURE 2.4 – Représentation de l'étape initiale dans un GRAFCET.

l'arrivée étant à la partie supérieure. Des flèches doivent être utilisées chaque fois qu'une meilleure compréhension pourra en résulter et chaque fois que l'orientation fixée n'est pas respectée. [8]

## 2.7 Règles d'évolution

- Règle N°1 : Condition initiale : A l'instant initial, seules les étapes initiales sont actives. la figure 2.5 [8] représente l'étape initiale dans un GRAFCET.
- Règle N°2 : Franchissement d'une transition : Représentée dans la figure 2.6 [8], ainsi pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives. Le franchissement d'une transition se produit lorsque la transition est validée, et seulement si la réceptivité associée est vraie.
- Règle N°3 : Évolution des étapes actives : Représentée dans la figure 2.7 [8].

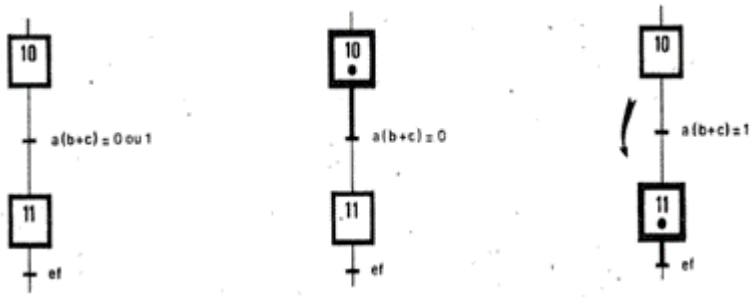


FIGURE 2.5 – Franchissement d'une transition dans un GRAFCET.

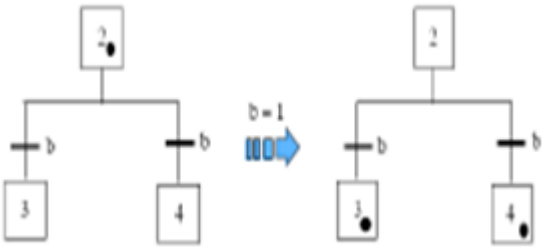


FIGURE 2.6 – Évolution des étapes actives dans un GRAFCET.

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

- Règle N°4 : Franchissement simultané : Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.
- Règle N°5 : Conflit d'activation : Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active. On évite ainsi des commandes transitoires (néfastes à la partie opérative).

## 2.8 Conclusion

Le GRAFCET est un outil largement utilisé pour les automates programmables indutriels.

Après avoir vu une description des automates programmables industriels dans le chapitre précédant, dans ce chapitre nous avons présenté le GRAFCET, son histoire, la manière dont il est élaboré à partir d'un cahier des charges, ses types, éléments et règles d'évolution.

# Conclusion Générale

Ainsi de plus en plus d'entreprises cherchant à automatiser leurs processus de production, poussées par l'essor de la prise de décision basée sur les données et l'évolution des applications automatique, le rôle des API dans la fabrication devient de plus en plus important.

Dans tous les secteurs et applications, les fabricants utilisent des automates programmables industriels pour surveiller et automatiser leurs processus de production. Les automates programmables sont des ordinateurs industriels robustes qui peuvent résister aux conditions d'un environnement industriel.

A travers ce mémoire nous avons pu étudier en détails les automates programmables industriels, et les langages de programmation de ces derniers, plus particulèrement le GRAFCET

# Bibliographie


[1] W. Bolton. *Programmable Logic Controllers, Fourth Edition*. 2006.

[2] Ulysse Develle. Architectures des automates programmables industriels, 28 mai 2020.

[3] E. A. Bryan L. A. Bryan. *PROGRAMMABLE CONTROLLERS, THEORY AND IMPLEMENTATION*. 1997.

[4] https ://www.technologuepro.com/cours-automatismes-industriels/chapitre-4-la-programmation-des api.pdf. chapitre 4 la programmation des api, consulté le 21/06/2022.

[5] *UMC,cours Réseaux, LA PROGRAMMATION DES API, consulté le 21/06/2022*.

[6] *chapitre 9, Réseaux industriels, Manuel Schneider*.

[7] James A. Rehg. Structured plc programming with sequential function charts. 2001.

[8] Agence nationale pour le DEveloppement de la Production Automatisée. Le grafcet. 1967.

[9] Le grafcet g7. grafcet fonctionnel vs grafcet technologique.

[10] https ://www.specialautom.net/automatisme/Grafcet.pdf. Grafcet, consulté le 21/06/2022.