

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



Université de Ghardaïa

N° d'ordre :
N° de série :

Faculté des Sciences et Technologies
Département des Sciences et Technologies

Projet de fin d'étude présenté en vue de l'obtention du diplôme de

LICENCE

Domaine : Sciences et Technologies

Filière : Génie Electrique

Spécialité : Maintenance en Instrumentation Industrielle

Thème

Etude des Automates Programmables Industriels

Par :

- **SEBROU Abdelaziz**
- **GUETIB Hamza**

Jury :

M. A.BENCHAABANE

Maître Assistant B

Univ.Ghardaïa

Encadreur

M. R.SADOUNI

Maître Assistant B

Univ.Ghardaïa

Examineur

Année universitaire 2012/2013

DEDICACES

Je dédie ce travail à mes parents, qui ont employés tous leurs moyens et efforts, pour m'élever, pour me former, et pour me faire arriver à ce niveau. A tous les professeurs qui ont contribués à ma formation du primaire a l'université.

Je dédie aussi ce travail à mes frères et sœurs, A tous ceux que j'aime et qui m'aiment, A ma famille et à mes amis.

Abdelaziz

A ma tendre mère et à mon cher père.

A tous ceux que j'aime et qui m'aiment.

A ma famille et à mes amis

Hamza

REMERCIEMENTS

Nous remercions en premier lieu **Mr A.Benchaabane**, notre encadreur, pour son aide et disponibilité tout au long de ce projet.

Nous tenons à remercier toute personne ayant contribué de près ou de loin à l'aboutissement de nos efforts.

Merci à nos professeurs et formateurs du primaire à l'université.

Résumé

Ce travail que nous avons effectué consiste d'étude sur l'automate programmable industriel (API), il est forme particulière d'automate à base de microprocesseur qui se fonde sur une mémoire programmable pour enregistrer les instructions et mettre en œuvre des fonctions, qu'elles soient logiques, de séquençement de temporisation, de comptage ou arithmétiques, pour contrôler des machines et des processus. Le logiciel utiliser pour la simulation d'API s'appelle STEP7 qui constitue des solutions efficaces pour décrire graphiquement des systèmes de commande séquentiels. Nous pouvons ainsi évaluer et corriger le programme avant son installation sur le matériel physique.

Mots-clés: automate programmable industriels, STEP7.

Abstract

This work we have done is to study programmable logic controller (PLC), it is a special form of microprocessor-based controller that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic to control machines and processes.

The software used for the simulation PLC is called STEP 7 which provides an efficient way to describe sequential control systems graphically, and we can enabling testing and refining prior to physical hardware installation.

Keywords: programmable logic controller, STEP 7.

قمنا في هذا العمل بدراسة جهاز التحكم المنطقي القابل للبرمجة، وهو شكل خاص من وحدة تحكم المعالجات الدقيقة القائمة على استخدام ذاكرة قابلة للبرمجة لتخزين التعليمات وتنفيذ مهام مثل المنطق، العد، والحساب من أجل التحكم في الآلات والعمليات .
لمحاكاة جهاز التحكم المنطقي القابل للبرمجة قمنا باستعمال برنامج STEP 7 الذي يسمح لنا بمراقبة و تتبع أنظمة التحكم بشكل بياني مع إمكانية التجربة بدون القيام بتوصيل الأجهزة.

الكلمات المفتاحية : جهاز التحكم المنطقي القابل للبرمجة ، STEP 7،

Liste des Figures

<i>Figure 1.1: exemples de tâches de commande et de capteurs d'entrée</i>	<i>5</i>
<i>Figure 1.2: un circuit de commande.....</i>	<i>5</i>
<i>Figure 1.3:un automate programmable industriel.....</i>	<i>8</i>
<i>Figure 1.4: Structure d'un API.</i>	<i>12</i>
<i>Figure 1.5: les signaux : (a) discrets, (b) numériques et (c) analogiques. ...</i>	<i>12</i>
<i>Figure 1.6: Modèle de base des communications.....</i>	<i>13</i>
<i>Figure 1.7: Architecture d'un API.</i>	<i>15</i>
<i>Figure 1.8: Un photocoupleur.....</i>	<i>18</i>
<i>Figure 1.9: Niveaux des entrées.</i>	<i>19</i>
<i>Figure 1.10:Niveaux des sorties.....</i>	<i>19</i>
<i>Figure 1.11: les entrées : (a) à fourniture et (b) à absorption de courant. ..</i>	<i>20</i>
<i>Figure 1.12: les sorties : (a) à fourniture et (b) à absorption de courant....</i>	<i>21</i>
<i>Figure 1.13: Bloc fonctionnel du langage LD</i>	<i>39</i>
<i>Figure 1.14: Structure du langage FBD.....</i>	<i>40</i>
<i>Figure 1.15: Fonction élémentaire du langage FBD</i>	<i>41</i>
<i>Figure 1.16: les éléments constituant d'un programme SFC.....</i>	<i>43</i>
<i>Figure 1.17: Divergences et convergences simple</i>	<i>44</i>

<i>Figure 1.18: Divergences et convergences double</i>	44
<i>Figure 1.19: Saut a une étape dans un programme SFC</i>	45
<i>Figure 1.20: Macros-étape dans un programme SFC</i>	45
<i>Figure 1.21: Interaction du logiciel et du matériel</i>	49
<i>Figure 1.22: Structure du projet dans SIMATIC Manager</i>	50
<i>Figure 1.23: choix de la station de travail</i>	52
<i>Figure 1.24: configuration de matériel</i>	52
<i>Figure 1.25: Sélections des modules</i>	54
<i>Figure 1.26: édition des mnémoniques</i>	55
<i>Figure 1.27: L 'éditeur de programme CONT</i>	56

Liste des tableaux

<i>Tableau 1: Opérateur logique du langage ST</i>	<i>27</i>
<i>Tableau 2: Opérateur arithmétique du langage ST</i>	<i>28</i>
<i>Tableau 3: Opérateur de Comparaison du langage ST</i>	<i>28</i>
<i>Tableau 4: Instructions conditionnelles du langage ST</i>	<i>29</i>
<i>Tableau 5: Instruction de choix du langage ST</i>	<i>30</i>
<i>Tableau 6: Instructions répétitives du langage ST</i>	<i>31</i>
<i>Tableau 7: Structure du langage IL</i>	<i>33</i>
<i>Tableau 8: Opérateurs d'affectations du langage IL</i>	<i>34</i>
<i>Tableau 9: Opérateurs logique du langage IL</i>	<i>34</i>
<i>Tableau 10: Opérateurs arithmétiques du langage IL</i>	<i>35</i>
<i>Tableau 11: Opérateur de comparaison du langage IL</i>	<i>35</i>
<i>Tableau 12: Branchement dans le langage IL</i>	<i>36</i>
<i>Tableau 13: Les contacts du langage LD</i>	<i>38</i>
<i>Tableau 14: Les relais du langage LD</i>	<i>38</i>
<i>Tableau 15: Composants graphique du langage SFC</i>	<i>42</i>
<i>Tableau 16: Comparaison des langages</i>	<i>46</i>

Liste des abréviations

PC : Personal Computer.

MMS : Manufacturing Message Specification.

CEI : Commission Electrotechnique Internationale.

ISO : Organisation Internationale de Normalisation.

API : Automate Programmable Industriel.

E/S : Entrées/Sorties.

ST : Structured Text (langage Texte Structuré).

LD : Ladder Diagram (Diagramme à Echelle).

IL : Instruction List (langage Liste d'Instructions).

FBD : Function Block Diagram (Diagramme de Blocs Fonctionnels).

SFC : Sequential Function Chart (Diagramme de Fonctions Séquentielles).

FC : Flow Chart (Diagramme de Flux).

GRAFCET : GRAPhe Fonctionnel de Commande Etapes/Transitions.

HMI : Human Machine Interface (Interface homme/machine).

PLC : Programmable Logic Controller (Automate programmable industriel).

DCS : Distributed Control System (Système de contrôle distribué).

PCI : Peripheral Component Interconnect

DLL : Dynamic Link Library.

TTL : Transistor-Transistor Logic

PIB : Peripheral Interface Bus (Bus interface de périphériques).

Sommaire

INTRODUCTION GENERALE	1
Chapitre I : Automates Programmables Industriels.....	3
I) Introduction :.....	4
II) Systèmes de commande :	4
III) Limitations des solutions PC :.....	6
III.1) Sur le plan matériel :.....	6
III.2) Sur le plan logiciel :	6
IV) Automates programmables industriels	7
V) Conclusion :.....	9
Chapitre II : Matériel ET Architecture interne d'un API.....	10
I) Introduction :.....	11
II) Matériel :.....	11
III) Architecture interne :.....	13
III.1) Le CPU (Central Processing Unit)	14
III.2) Les Bus.....	14
III.3) La Mémoire :.....	16
III.4) Unité d'entrées-sorties :.....	17
III.5) Fourniture et absorption de courant :	20
IV) Conclusion.....	22
Chapitre III : Les Langages de la Norme CEI 61131-3.....	23
I) Introduction :.....	24
II) Les normes pour les systèmes automaties :.....	24
II.1) Présentation du standard CEI 61131:	24

II.2) Les parties de la norme CEI 61131 :.....	25
III) Les langages de la norme CEI 61131-3 :.....	27
III.1) Langages Textes:.....	27
III.1.1) Texte structuré (ST, Structured Text)	27
III.1.2) Listes d'instructions (IL, Instruction list) :.....	32
III.2) Langages graphiques :	37
III.2.1) Le langage à contact (LD, Ladder Diagram) :.....	37
III.2.2) LANGAGE Logigramme (FBD, Function Block Diagram) :....	40
III.2.3) Langage séquentiel (SFC, Sequential Function Chart) :.....	42
IV) Comparaison des langages	46
Chapitre IV : SIMATIC Step7	47
I) Introduction :.....	48
II) Interaction du logiciel et du matériel.....	49
III) Structure du projet dans SIMATIC Manager.....	50
IV) Création d'un projet STEP7.....	51
III.1) Configuration du matériel.....	53
III.2) Définition des mnémoniques.....	54
III.3) Edition des programmes	55
III.3.1) Blocs d'organisation	57
III.3.2) Fonctions et blocs fonctionnels	58
III.3.3) Bloc de données	58
III.4) Programmation des blocs	58
CONCLUSION GENERALE	60
BIBLIOGRAPHIE	61

INTRODUCTION GENERALE

Le développement massif des techniques de l'automatisme a permis le passage de la machine automatisée à celui des systèmes automatisés de production, qui gèrent l'alimentation en énergie et qui permettent d'avoir une meilleure qualité des produits en plus de la sécurité et de la flexibilité des processus, mais cela entraîne un accroissement des besoins, en particulier la manipulation d'un grand nombre de variables et la gestion de véritables flux de communication.

Cela explique que les systèmes câblés deviennent trop volumineux et trop rigides pour de telles applications, et que l'on se tourne donc vers des solutions utilisant les techniques de traitement de l'information par processeurs programmables ou les automates programmables industriels occupent une place de choix.

Un automate programmable est un système électronique, destiné à être utilisé dans un environnement industriel, il utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en œuvre de fonctions spécifiques, telles que des fonctions de logique, de mise en séquence, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties divers types de machines ou de processus.

Les API (ou Programmable Logic Controller PLC) sont aujourd'hui les constituants les plus répandus des automatismes. On les trouve non seulement dans tous les secteurs de l'industrie, mais aussi dans les services (gestion de parkings, d'accès à des bâtiments, contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes) et dans l'agriculture (composition et délivrance de rations alimentaires dans les élevages) et cela est due surtout à leurs performances de sécurité.

Dans tous les secteurs où ils sont utilisés, les API doivent remplir des tâches de commande en élaborant des actions suivant une algorithmique appropriée, à partir des informations données par des détecteurs (Tout ou Rien) ou des capteurs (analogiques ou numériques), et des tâches de communication avec des opérateurs humains ou avec d'autres processus (automates, calculateurs de gestion de production,...).

Bien que les automates programmables incluent des modules et des options permettant le contrôle ou la commande d'un système continu, beaucoup d'utilisateurs pensent que les APIs ne sont destinées qu'à la commande séquentielle.

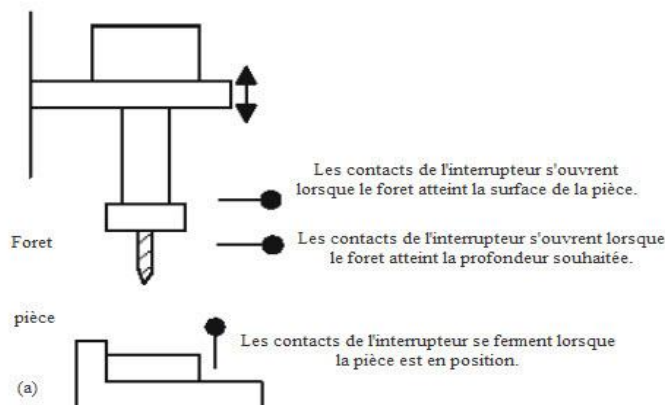
Chapitre I : Automates Programmables Industriels

I) Introduction :

Un Automate Programmable Industrielle (API) est une machine électronique programmable par un personnel non informaticien, il est destiné à piloter en ambiance industrielle et en temps réel des procédés appelés encore partie opérative. On programme l'API pour effectuer des opérations cycliques qui reçoit des données par ces entrées, ensuite ces derniers sont traitées par un programme définis et les résultats obtenu sont délivrées par ses sorties pour commander le système.

II) Systèmes de commande :

Quelles tâches un système de commande peut-il prendre en charge ? il peut être chargé de contrôler une séquence d'événements, de maintenir constante une certaine variable ou de suivre un changement prévu. par exemple, le système de commande d'une perceuse automatique (voir figure 1.1a) peut démarrer la descente du foret lorsque la pièce est en position, démarrer le perçage lorsque le foret arrive sur la pièce, stopper le perçage lorsque la profondeur du trou souhaitée est atteinte, retirer le foret, arrêter la perceuse, puis attendre que la pièce suivante arrive en position avant de répéter le processus. Un autre système de commande (voir figure 1.1b) pourrait être utilisé pour vérifier le nombre d'articles convoyés par une bande transporteuse et les diriger vers une caisse. les entées de ces systèmes de commande peuvent provenir d'interrupteurs, qui sont ouverts ou fermés. Par rupteur, qui se ferme, ou par d'autres capteurs, par exemple de température ou de débit. Le contrôleur peut être chargé de commander un moteur pour déplacer un objet vers un certain emplacement ou pour tourner une vanne, ou encore d'allumer ou d'éteindre un radiateur.



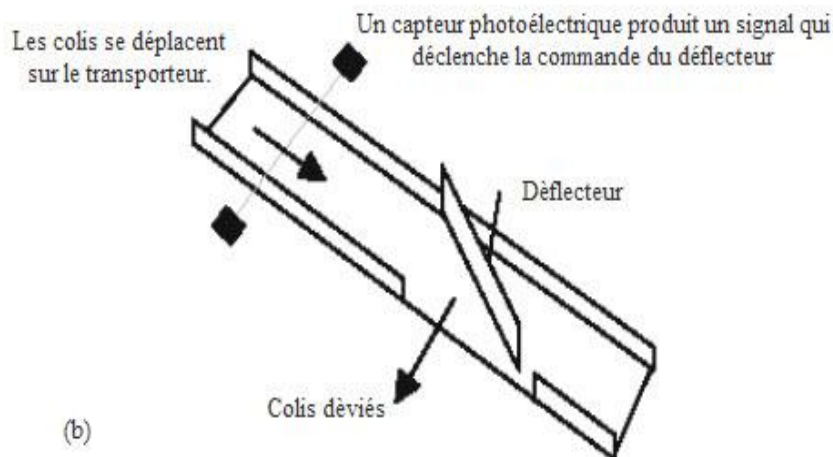


Figure 1.1: exemples de tâches de commande et de capteurs d'entrée : (a) une perceuse automatique et (b) un système de conditionnement.

Quelles formes un système de commande peut-il prendre ? Pour la perceuse automatique, il est possible de câbler des circuits électriques dans lesquels l'ouverture ou la fermeture de l'interrupteur démarrent des moteurs ou déclenchent des vannes. Ainsi, la fermeture d'un interrupteur peut être utilisée pour activer un relais et mettre sous tension une vanne pneumatique ou hydraulique, qui met sous pression un vérin afin de pousser dans la position requise. De tels circuits électriques sont spécifiques à la perceuse automatique. Pour contrôler le nombre d'articles placés dans un carton, nous pouvons également concevoir des circuits électriques qui mettent en œuvre des capteurs et des moteurs. Cependant, le circuit de chacun de ces systèmes de commande sera différent. Dans un système de commande « traditionnel », les règles de fonctionnement et les actions exécutées sont déterminées par le câblage. Lorsque les règles qui déterminent les actions de commande sont modifiées, le câblage doit également être revu.

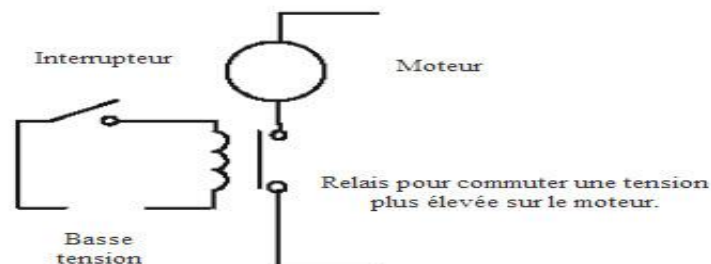


Figure 1.2: un circuit de commande.

III) Limitations des solutions PC :

III.1) Sur le plan matériel :

La vitesse de renouvellement de la technologie du PC est extrêmement rapide au regard de l'échelle de temps d'un procédé de fabrication (10-15 ans). La durée de vie d'un procédé industriel apparaît difficilement compatible avec les évolutions très rapides du PC qui pose inmanquablement le problème de la pérennité de la solution du contrôle du procédé. 18 mois après la commercialisation d'un modèle de PC, il n'est plus disponible sur le marché, par voie de conséquence, la gestion du stock de pièces de rechange n'est pas chose aisée.

La sensibilité aux contraintes des environnements industriels (vibrations, chocs, températures, corrosions...) se traduit par un accroissement très important du prix des PC pour obtenir un « durcissement » de ceux-ci ou requiert une mise en armoire de protection.

Les contraintes industrielles (rayonnements) sont souvent incompatibles avec l'utilisation intensive de supports d'informations magnétiques (disque dur...). Des unités de stockage statiques de données sont certainement mieux adaptées aux contraintes industrielles.

III.2) Sur le plan logiciel :

La cible initiale du PC, le marché de la bureautique et le marché grand public, n'a pas originellement imposé de doter le PC de capacité de traitement temps réel. Un système d'exploitation temps réel efficace est une condition nécessaire pour le contrôle de procédés industriels.

Lors du redémarrage du PC, le procédé industriel se trouve momentanément sans aucun contrôle, suite à un cas d'exception non géré par le système d'exploitation, et/ou l'application écarte actuellement le PC de bon nombre d'applications de contrôle commande.

Les virus, véritable maladie du PC, constituent une infection présentant un caractère périlleux à l'utilisation d'un PC pour assurer le contrôle d'une machine ou procédé industriel.

Un virus peut en effet rendre indisponible une ressource (donnée, fichier...) indispensable à la tâche de contrôle et provoquer ainsi des perturbations, voire l'arrêt de l'outil de production.

Sous les systèmes d'exploitations tels que Windows, il n'est pas possible de garantir le déterminisme temporel des événements et des tâches en exécution. Ce déterminisme temporel que nous appelons caractéristique temps réelle d'un système d'exploitation n'est pas garanti par Windows.

IV) Automates programmables industriels

Un automate programmable industriel (API) est une forme particulière de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker les instructions et que implémente différentes fonctions, qu'elles soient logiques, de séquençement, de temporisation, de comptage ou arithmétiques, pour commander les machines et les processus (Voir figure 1.3).

Il est conçu pour être exploité par des ingénieurs, dont les connaissances en informatique et langages de programmation peuvent être limitées. La création et la modification des programmes de l'API ne sont pas réservées aux seuls informaticiens les concepteurs de l'API l'ont préprogrammé pour que la saisie du programme de commande puisse se faire à l'aide d'un langage simple et intuitif. La programmation de l'API concerne principalement la mise en œuvre d'opérations logiques et de commutation, par exemple, si A ou B se produit, alors allumer C, ou si A et B se produisent, alors allumer D. les dispositifs d'entrée, c'est-à-dire des capteurs. Comme des interrupteurs, et les dispositifs de sortie, c'est-à-dire des moteurs, des vannes, etc...., du système sont connectés l'API. L'opérateur saisit une séquence d'instructions, le programme, dans la mémoire de l'API. L'automate surveille ensuite les entées et les sorties conformément aux instructions du programme et met en ouvre les règles de commande définies.

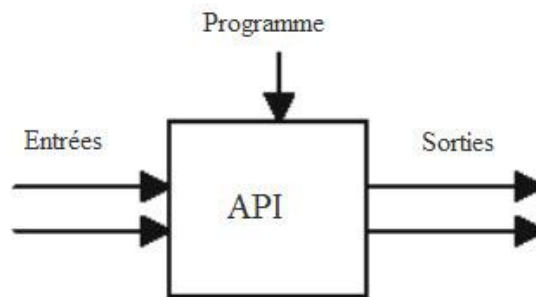


Figure 1.3: *un automate programmable industriel.*

Les API présentent un avantage majeur : le même automate de base peut être employé avec une grande diversité de système de commande. Pour modifier un système de commande et les règles appliquées, un opérateur doit simplement saisir une suite d'instructions différente. On obtient ainsi un système flexible et économique, utilisable avec des systèmes de commande dont la nature et la complexité peuvent varier énormément.

Les API sont comparables aux ordinateurs. Toutefois, alors que les ordinateurs sont optimisés pour les tâches de calcul et d'affichage, les API le sont pour les tâches de commande et les environnements. Voici ce qui caractérise les API :

- ils sont solides et conçus pour supporter les vibrations, les températures basses ou élevées, l'humidité et le bruit ;
- les interfaces des entrées et des sorties sont intégrées à l'automate ;
- ils sont faciles à programmer et leur langage de programmation facile à comprendre est principalement orienté sur les opérations logiques et de commutation.

Le premier API a été conçu en 1969 ; ils sont à présent largement utilisés. Ils prennent la forme de petites unités autonomes pour environ vingt entrées-sorties numériques ou de systèmes modulaires qui peuvent être employés pour des entrées-sorties très nombreuses, analogiques ou numériques, et qui disposent des modes de régulation PID (proportionnel, intégral et dérivé).

V) **Conclusion** :

La venue des automates programmables industriels (API), avec leur solution programmée, réduit de beaucoup l'espace requis pour l'installation, simplifie le filage et élimine complètement le bruit; les modifications de l'automatisme deviennent presque un jeu d'enfant. Les électriciens, selon la majorité des experts, ont alors vu leur tâche simplifiée.

Chapitre II : Matériel ET Architecture interne d'un API

I) Introduction :

Cet ensemble électronique gère et assure la commande d'un système automatisé. Il se compose de plusieurs parties et notamment d'une mémoire programmable dans laquelle l'opérateur écrit, dans un langage propre à l'automate, des directives concernant le déroulement du processus à automatiser.

Son rôle consiste donc à fournir des ordres à la partie opérative en vue d'exécuter un travail précis comme par exemple la sortie ou la rentrée d'une tige de vérin, l'ouverture ou la fermeture d'une vanne.

II) Matériel :

De manière générale, un API est structuré autour de plusieurs éléments de base que sont l'unité de traitement, la mémoire, l'unité d'alimentations, les interfaces d'entrées-sorties, l'interface de communication et le périphérique de programmation (voir figure 1.4) :

- **Le processeur ou unité centrale de traitement** (CPU, central processing unit) contient le microprocesseur. Le CPU interprète les signaux d'entrée et effectue les actions de commande conformément au programme stocké en mémoire, en communiquant aux sorties les décisions sous forme de signaux d'action.
- **L'unité d'alimentation** est indispensable puisqu'elle convertit une tension alternative en une basse tension continue (5v) nécessaire au processeur et aux modules d'entrées-sorties.
- Le périphérique de programmation est utilisé pour entrer le programme dans la mémoire du processeur. Ce programme est développé sur le périphérique, puis transféré dans la mémoire de l'API.
- **La mémoire** contient le programme qui définit les actions de commande effectuées par le microprocesseur. Elle contient également les données qui proviennent des entrées en vue de leur traitement, ainsi que celles des sorties.
- **Les interfaces d'entrées-sorties** permettent au processeur de recevoir et d'envoyer des informations aux dispositifs extérieurs. Les entrées peuvent être des interrupteurs, comme dans le cas de la perceuse automatique (voir figure 1.1a), ou d'autres capteurs, comme des cellules photoélectriques dans le cas du mécanisme

de comptage (voir figure 1.1b), des sondes de température, des débitmètres, etc. les sorties peuvent être des bobines de moteur, des électrovannes, etc. nous reviendrons sur les interfaces d'entrées-sorties. les dispositifs d'entrées-sorties peuvent être classés en trois catégories, selon qu'ils produisent des signaux discrets, numériques ou analogiques (voir figure 1.5). 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. Par exemple, un capteur de température peut produire une tension proportionnelle à la température.

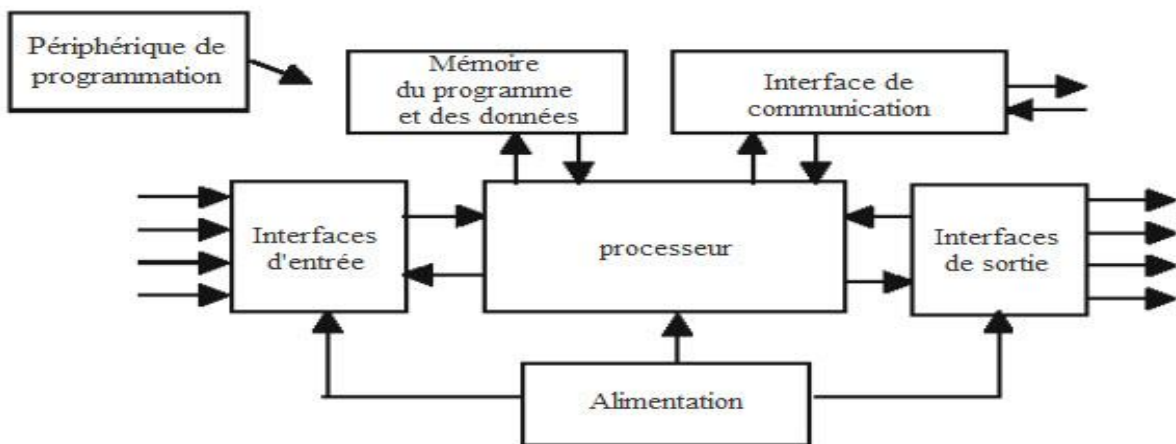


Figure 1.4: Structure d'un API.

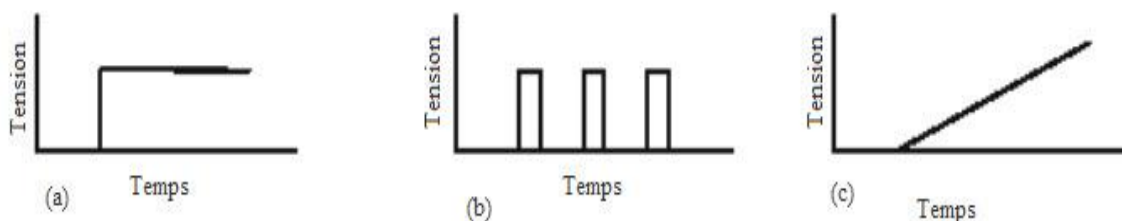


Figure 1.5: les signaux : (a) discrets, (b) numériques et (c) analogiques.

- **L'interface de communication** est utilisée pour recevoir et transmettre des données sur des réseaux de communication qui relient l'API à d'autres API distants (voir figure 1.6). Elle est impliquée dans des opérations telles que la vérification d'un périphérique. L'acquisition de données, la synchronisation entre des applications et la gestion de la connexion.

III) Architecture interne :

La figure 1.7 illustre l'architecture interne de base d'un API. Il est constitué d'une unité centrale de traitement (CPU, central processing unit) qui comprend le microprocesseur, la mémoire et les entrées-sorties du système. Le CPU, ou processeur, contrôle et exécute toutes les opérations de l'automate. Il est muni d'une horloge dont la fréquence se situe généralement entre 1 et 5 MHz. Cette fréquence détermine la rapidité de fonctionnement de l'API et sert de base au minutage et à la synchronisation pour tous les éléments du système. Au sein de les chemins par lesquels passent ces signaux sont appelés bus. Au sens physique, un bus n'est qu'un ensemble de conducteurs sur lesquels circulent les signaux électriques. Il peut s'agir de pistes sur un circuit imprimé ou de fils dans un câble plat.

Le CPU utilise le bus de données pour l'échange des données entre les composants, le bus d'adresse pour la transmission des adresses des emplacements permettant d'accéder aux données stockées et le bus de contrôle pour les signaux qui concernent les actions de contrôle internes le bus système sert aux communications entre les ports d'entrées-sorties et l'unité d'entrées-sorties.

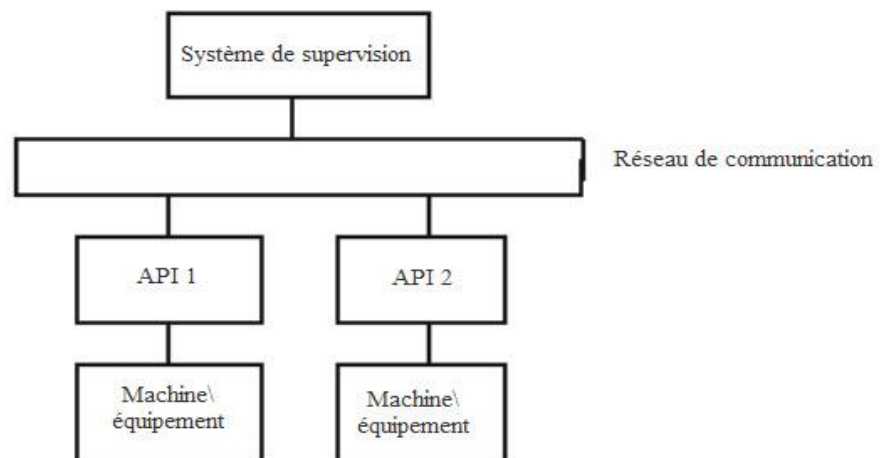


Figure 1.6: *Modèle de base des communications.*

III.1) Le CPU (Central Processing Unit)

L'architecture interne du CPU dépend du microprocesseur employé. En général, les CPU sont constitués des éléments suivants :

- *L'unité arithmétique et logique* (UAL) est responsable de la manipulation des données, ainsi que de l'exécution des opérations arithmétiques d'addition et de soustraction et des opérations logiques ET, OU, NON et OU exclusif.
- La mémoire, appelée *registres*, se trouve à l'intérieur du microprocesseur et sert à stocker les informations nécessaires à l'exécution d'un programme.
- Une *unité de commande* est utilisée pour gérer le minutage des opérations.

III.2) Les Bus

Les bus représentent les chemins de communication au sein de l'API. Les informations sont transmises en binaire, c'est-à-dire sous forme de groupes de bits. Un bit est un chiffre binaire qui vaut 1 ou 0 et indique les états marche/arrêt (ON/OFF). Un mot est un groupe de bits qui constitue une information. Ainsi, un mot de huit bits peut être le nombre binaire 00100110. Chaque bit est envoyé simultanément sur son propre fil. Le système comprend quatre bus :

- *Le bus de données* transporte les données utilisées dans les traitements effectués par le CPU. Un microprocesseur huit bits dispose d'un bus de données interne qui permet de traiter des nombres de huit bits. Il peut donc réaliser des opérations entre des nombres de huit bits et fournir des résultats sous forme de valeurs sur huit bits.
- *Le bus d'adresse* transporte les adresses des emplacements mémoire. Pour que chaque mot puisse être localisé en mémoire, chaque emplacement possède une adresse unique. À l'instar des maisons d'une ville qui disposent d'une adresse distincte de manière à pouvoir être situées, chaque emplacement de mot possède une adresse que le CPU utilise pour accéder aux données enregistrées à cet emplacement, que ce soit pour les lire ou pour les y écrire. C'est le bus d'adresse qui fournit les informations stipulant l'adresse à laquelle le CPU doit accéder. Si le bus d'adresse est constitué de huit lignes, le nombre de mots huit bits, et, par

conséquent, le nombre d'adresses distinctes, est égal à 28, c'est-à-dire 256. Avec seize lignes d'adresse, il est possible d'accéder à 65 536 emplacements.

- **Le bus de contrôle** transporte les signaux utilisés par le CPU pour le contrôle. Il sert, par exemple, à informer les dispositifs mémoire s'ils vont recevoir des données à partir d'une entrée ou s'ils vont envoyer des données, et à transmettre les signaux de minutage qui permettant de synchroniser les opérations.
- **Le bus système** sert aux communications entre les ports d'entrées-sorties et l'unité d'entrées-sorties.

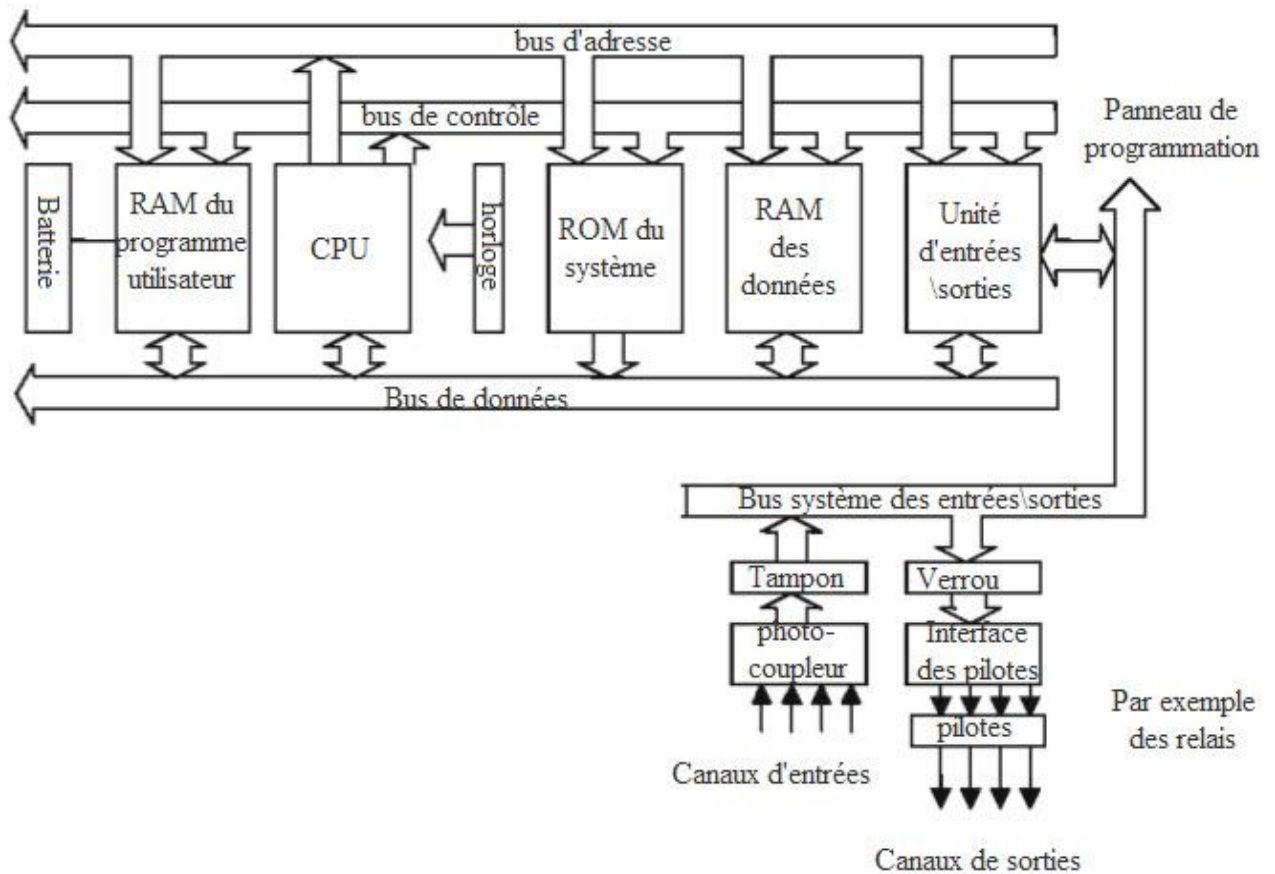


Figure 1.7: Architecture d'un API.

III.3) La Mémoire :

Pour que l'API effectue son travail, il doit accéder aux données à traiter et aux instructions. C'est-à-dire au programme, qui lui expliquent comment traiter ces données. Ces informations sont stockées dans la mémoire de l'API, qui est composée de plusieurs éléments :

- **La mémoire morte** (ROM, Read Only Memory) du système représente un espace de stockage permanent pour le système d'exploitation et les données figées utilisées par le CPU.
- **La mémoire vive** (RAM, Random Access Memory) est utilisée pour les données. C'est là que sont stockées les informations sur l'état des entrées et des sorties, ainsi que les valeurs des temporisateurs, des compteurs et des autres dispositifs internes. La RAM des données est parfois appelée tableau de données ou tableau de registres. Une partie de cet espace mémoire, c'est-à-dire un bloc d'adresses, est réservée aux adresses d'entrées et de sorties, ainsi qu'à leur état. Une autre partie est réservée aux données prédéfinies, et une autre encore, au stockage des valeurs des compteurs, des temporisateurs, etc.
- **Une mémoire morte reprogrammable** (EPROM, Erasable and Programmable Read Only Memory) est parfois employée pour stocker de manière permanente les programmes.

Les programmes et les données en RAM peuvent être modifiés par l'utilisateur.

Tous les API disposent d'une quantité de RAM pour stocker les programmes développés par l'utilisateur et les données de ces programmes. Cependant, pour éviter la perte des programmes lorsque l'alimentation est coupée, une batterie complète l'API. En cas de coupure de l'alimentation, elle permet de conserver le contenu de la RAM pendant un certain temps. Lorsque le développement du programme en RAM est terminé, il peut être chargé dans une puce de mémoire EPROM, souvent un module ajouté à l'API, pour être rendu permanent. Par ailleurs, les canaux d'entrées-sorties disposent également de tampons temporaires.

La capacité de stockage d'une unité de mémoire est déterminée par le nombre de mots qu'elle peut enregistrer. Si la taille d'une mémoire est de 256 mots, elle peut stocker $256 \times 8 = 2\,048$ bits, pour des mots de huit bits, et $256 \times 16 = 4\,096$ bits, pour des mots de seize bits. La taille de la mémoire est souvent indiquée en fonction du nombre disponibles. 1 k représente le

nombre 210, c'est-à-dire 1 024. Les fabricants fournissent des puces de mémoire dont les emplacements sont des groupes de un, quatre ou huit bits. Une mémoire de type $4k \times 1$ dispose de $4 \times 1\,024 \times 1$ emplacements de bit. Une mémoire de type $4k \times 8$ dispose de $4 \times 1\,024 \times 8$ emplacements de bit. Le terme octet désigne un mot de huit bits. Par conséquent, une mémoire $4k \times 8$ peut enregistrer 4 096 octets. Avec un bus d'adresses de seize bits, nous disposons de 216 adresses différentes. Par conséquent, si nous stockons des mots de huit bits à chaque adresse, nous avons 216×8 emplacements de stockage et nous utilisons donc une mémoire de taille $216 \times 8/210 = 64\,k \times 8$, qui peut être constituée de quatre puces de mémoire $16\,k \times 8$ bits.

III.4) Unité d'entrées-sorties :

L'unité d'entrées-sorties apporte l'interface entre le système et le monde extérieur. Au travers de canaux d'entrées-sorties, elle permet d'établir des connexions avec des dispositifs d'entrée, comme des capteurs, et des dispositifs de sorties, comme des moteurs et des solénoïdes. C'est également par l'intermédiaire de cette unité que se fait la saisie des programmes depuis un terminal. Chaque point d'entrée-sortie dispose d'une adresse unique, que le CPU peut utiliser. Le principe est comparable à une rangée de maisons le long d'une rue ; le numéro 10 peut correspondre à la « maison » pour l'entrée d'un capteur particulier, tandis que le numéro 45 peut correspondre à la « maison » utilisé pour la sortie vers un moteur spécifique.

Puisque les canaux d'entrées-sorties mettent en place les fonctions d'isolation et de traitement des signaux, il est possible de connecter directement des capteurs et des actionneurs aux canaux, sans passer par un autre circuit d'interface.

L'isolation électrique avec le monde extérieur est généralement réalisée par des photocoupleurs (également appelés optocoupleurs), dont le principe est illustré à la (figure 1.8). Lorsque la diode électroluminescente (LED, Light Emitting Diode) est traversée par une impulsion numérique, elle produit un rayonnement infrarouge.

Ce rayonnement est détecté par le phototransistor, qui fait naître une tension dans son circuit. L'espace qui sépare la LED et le phototransistor crée une isolation électrique, mais une impulsion numérique dans le premier circuit permet néanmoins de produire une impulsion numérique dans le second circuit.

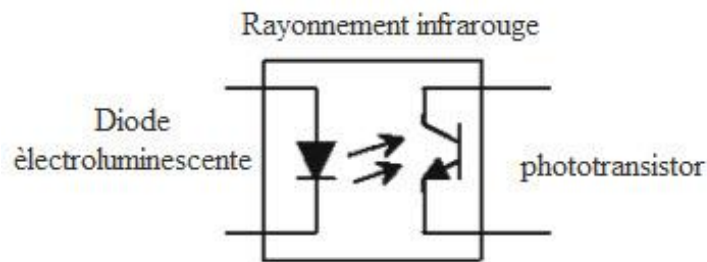


Figure 1.8: *Un photocoupleur.*

Pour être compatible avec le microprocesseur de l'API, le signal numérique utilise généralement une tension continue de 5 V. Toutefois, le traitement du signal réalisé au niveau du canal d'entrée. Un API élaboré peut ainsi accéder à des entrées dont les signaux numériques/discrets (c'est-à-dire tout ou rien) utilisent des tensions de 5V, 24V, 110 V et 240 V (voir figure 1.9). Un API de base sera généralement en mesure d'utiliser une seule forme d'entrées, par exemple des signaux 24 V.

Une unité d'entrées-sorties peut produire des sorties numériques avec un niveau de 5 V. Toutefois, après traitement du signal par des relais, des transistors ou des triacs, il est possible d'obtenir en sortie un signal de commutation 24 V et 100 mA, un courant continu de 110 V et 1A ou un courant alternatif de 240 V et 1 A ou 240 V et 2 A (voir figure 1.10). Avec un API de base, toutes les sorties seront probablement d'un seul type, par exemple un courant alternatif de 240 V et 1 A. En revanche, un API modulaire pourra disposer de différentes sorties en installant les modules correspondants.

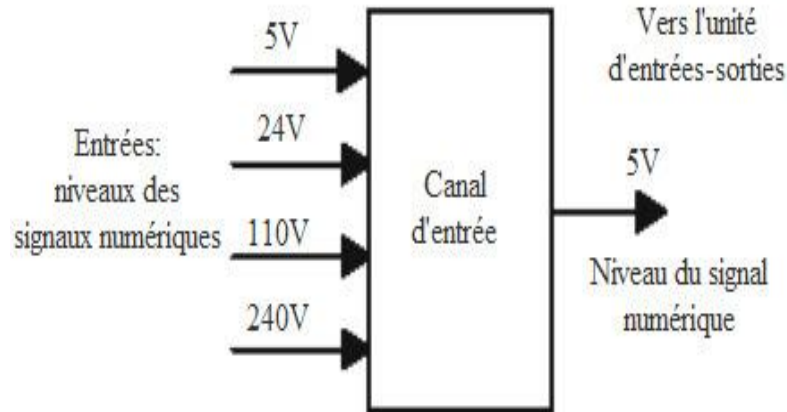


Figure 1.9: Niveaux des entrées.

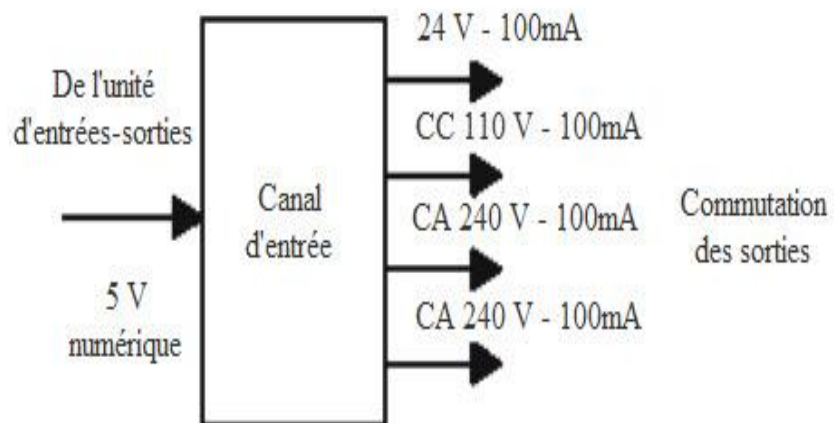


Figure 1.10: Niveaux des sorties.

Les sorties sont de type relais, transistor ou triac :

- Avec le type *relais*, le signal issu d'une sortie de l'API est utilisé pour déclencher un relais et peut activer des courants de quelques ampères dans le circuit externe. Grâce au relais, non seulement des courants faibles peuvent commuter des courants forts, mais l'isolation entre l'API et le circuit externe est également assurée à la commutation de courants alternatifs et continus. Ils peuvent supporter des surtensions transitoires et des courants de choc élevés.

- Avec le type *transistor*, la sortie se fonde sur un transistor pour commuter le courant dans le circuit externe. L'opération de commutation est donc extrêmement rapide. Toutefois les sorties de ce type sont adaptées uniquement à la commutation d'un courant continu et sont détruites par les surintensités et les tensions inverses élevées. Pour leur protection, il faut utiliser un fusible ou un système électronique. Les photocoupleurs sont employés à des fins d'isolation.
- Avec le type *triac*, et des photocoupleurs pour l'isolation, les sorties peuvent servir à contrôler les charges externes connectées à une alimentation en courant alternatif. Elles prennent en charge uniquement les courants alternatifs et sont très facilement détruites par les surintensités. De manière générale, les sorties de ce type sont toujours protégées par des fusibles.

III.5) Fourniture et absorption de courant :

Les termes fourniture (sourcing) et absorption (sinking) décrivent la manière dont les appareils à courant continu sont connectés à un API. Dans le cas de la fourniture, en supposant le sens conventionnel du module d'entrée. Autrement dit, le module d'entrée est une source de courant et le fournit au dispositif d'entrée (voir figure 1.11a). Dans le cas de l'absorption, un dispositif d'entrée fournit le courant au module de d'entrée reçoit et absorbe le courant (voir figure 1.11b). Si le courant va du module de sortie vers une charge de sortie, le module de sortie fournit le courant (voir figure 1.12a). en revanche, si le courant va de la charge de sortie vers le module de sortie, celui-ci absorbe le courant (voir figure 1.12b).

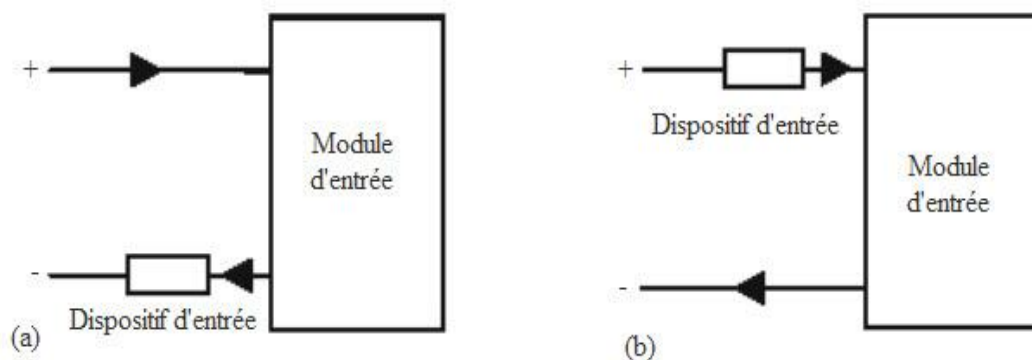


Figure 1.11: les entrées : (a) à fourniture et (b) à absorption de courant.

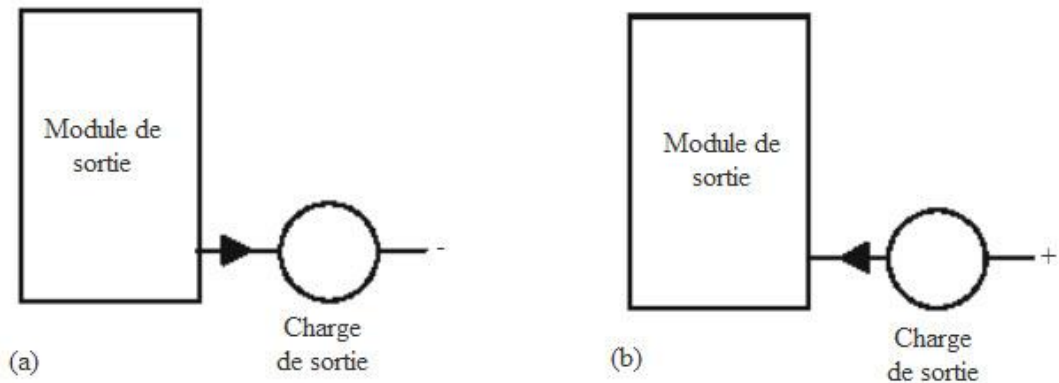


Figure 1.12: les sorties : (a) à fourniture et (b) à absorption de courant.

Il est important de connaître le type de la sortie ou de l'entrée afin qu'elle soit branchée correctement à l'API. Ainsi, les capteurs avec sorties à fourniture de courant doivent être connectés aux entrées à absorption de courant de l'API, tandis que les capteurs avec sorties à absorption de courant doivent être connectés aux entrées à fourniture de courant. Si ces directives ne sont pas suivies, l'interface avec l'API ne fonctionnera pas et risque d'être endommagée.

IV) Conclusion

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.

Ensuite, il faut trouver un API adapté aux besoins. Il existe des API à 8 E/S, d'autres à plus de 1 000... La multiplicité des modèles, des configurations, des fonctions métier, permettent de trouver le matériel qui convient. L'offre est presque trop large, et des critères non quantifiables mais importants (habitudes de l'entreprise ou de l'intégrateur, autres matériels à associer, etc.) viennent en resserrer la palette. Transformé partiellement ou totalement en outil spécialisé, l'API retenu, grâce à ses capacités de communication, prendra...

Chapitre III : Les Langages de la Norme CEI 61131-3

I) Introduction :

Tandis que le programmeur de langages évolués s'appuyait depuis longtemps sur un environnement de programmation normalisé, le programmeur d'automates programmables devait réapprendre une programmation spécifique pour chaque nouveau système d'API. Non seulement le langage de programmation différait, mais également les possibilités de structurer un programme. Ainsi, la transposition de programmes existants d'un système à l'autre demandait des efforts considérables.

La complexité sans cesse croissante des applications à base d'automates programmables a fait naître la nécessité d'une harmonisation, autrement dit la normalisation de la programmation des automates programmables.

II) Les normes pour les systèmes automaties :

Les normes permettent de remplacer aisément un produit par un équivalent quand on rencontre une difficulté d'approvisionnement quelconque, ou d'échange. De plus, elles permettent une interopérabilité des systèmes et produits industriels entre eux. Quoique volontaires par nature, elles sont donc devenues indispensables.

On peut rencontrer la norme CEI 61131 avec l'appellation CEI 1131, le chiffre 6 avant 1131 est apparu, ceci pour harmoniser les notations locales des pays, et les réunir dans une numérotation plus large à caractère international.

II.1) Présentation du standard CEI 61131 :

La diversité des aspects intervenant dans la conception des systèmes de contrôle se reflète dans celle des langages utilisés pour leur programmation. Elle comporte des aspects historiques, comme la diversité de cultures techniques (automatique, électromécanique, informatique...), l'héritage de volumineuses spécifications, le caractère éprouvé vis-à-vis de la sûreté.

Ces langages sont liés à l'utilisation de plate formes spécifiques pour l'exécution des contrôleurs typiquement les automates programmables industriels (API). Ils présentent des

traits directement liés à un schéma d'exécution, et qui repose explicitement sur des fonctionnements cycliques, et sur des architectures matérielles représentées explicitement, parfois proches des circuits électroniques. Bien que la programmation évolue vers une séparation du matériel, par des couches logicielles de plus en plus élaborées, ces langages continuent d'être pratiqués, au moins pour les motivations historiques évoquées ci-dessus, et même dans les cas où l'architecture matérielle utilisée est à base de PC.

Des efforts de normalisation tendent à promouvoir l'unification de ces langages et de ces architectures, et de la communication entre elles. En particulier, la norme CEI 61131 (la première édition date de 1993) définit la programmation des automates, et propose un cadre qui s'étend de la spécification aux architectures. Cette norme comporte à l'origine cinq parties (Introduction, Matériel, Langages de programmation (CEI-61131-3), Manuel et conseils à l'utilisateur, Communications.), et huit dans sa dernière version. PLCOpen, est une organisation fondée en 1992 dédiée au développement de la norme CEI 61131 en particulier à la troisième partie 61131-3

II.2) Les parties de la norme CEI 61131 :

La norme CEI 61131 est maintenant divisée en 8 parties

➤ CEI 61131-1 General information

Etablit les définitions et identifie les caractéristiques principales relevant de la sélection et de l'application des contrôleurs programmables et leurs périphériques associés.

➤ CEI 61131-2 Equipment requirements and tests

Spécifie besoins des équipements et les tests liés pour les API et leurs périphériques associés.

➤ CEI 61131-3 *Programming Languages*

définit comme un ensemble minimal, les éléments de programmation de base, les règles syntaxique et sémantique pour les langages de programmation le plus communément utilisés, y compris les langages graphiques comme Ladder Diagram Diagramme bloc fonctionnel, et les langages textuels Instruction List et de Structured Text, ainsi que les domaines d'application les plus important, les tests et les moyens par lesquels les fabricants peuvent étendre ou d'adapter ces ensembles de base à leurs propres implémentations.

➤ CEI 61131-4 User Guidelines

Un rapport technique qui fournit une vue d'ensemble informations et applications des grandes lignes du standard, pour l'utilisateur final de la solution automatisée.

➤ CEI 61131-5 Messaging service spécification

Définit les données de communication entre les automates et les autres systèmes électroniques utilisant la spécification de la messagerie industrielle (Manufacturing Message Specification MMS) selon la norme ISO/CEI 9506.

➤ CEI 61131-7 Fuzzy control programming

Définit les éléments de base pour le contrôle par logique floue, comme utilisé dans l'API

➤ CEI 61131-8 Grandes lignes pour l'application et l'implémentation des langages de programmation offrant aux développeurs un guide de langages de programmation définies dans la partie 3

La section 6 est réservée pour un usage ultérieur. Et la troisième partie, CEI 61131-3, est celle qui nous intéresse particulièrement ici, elle décrit les langages de programmation et leur sémantique. Elle définit les standards des langages et se présente sous la forme de 4 sections et d'une série de 62 tables de conformité. Lorsque l'on décrit la compatibilité d'un progiciel à ce standard, on compare généralement son jeu d'instructions avec ces tableaux de conformité. Parmi les produits qui ont été approuvés par l'organisation PLCopen, nous pouvons

mentionner : Altersys ISAGRAF, de C.J. International, AnchorPAW, Performance Software Associates, Inc, Siemens Step 7, et bien d'autres.

III) Les langages de la norme CEI 61131-3 :

III.1) Langages Textes:

III.1.1) Texte structuré (ST, Structured Text)

Le « Structured Text » (ST), est un langage de programmation structuré de la norme CEI 1131-3. C'est un langage adapté aux systèmes d'automatisation. La structure générale d'une instruction est la suivante :

<Variable> := <opérande> <opérateur> <opérande> ;

Chaque instruction se termine par le caractère « ; »

III.1.1.1) Les Operateurs :

1) Opérateur d'affectation

En « Structured Text » l'opérateur d'affectation est le symbole « := ».

2) Opérateurs logiques

Tableau 1: *Opérateur logique du langage ST*

Opérateur	Description	Opérateur	Description
NOT	Négation logique	AND	ET logique
OR	OU logique	XOR	OU logique exclusif

3) Opérateurs arithmétique

***Tableau 2:** Opérateur arithmétique du langage ST*

Opérateur	Description	Opérateur	Description
+	Addition	-	Soustraction
*	Multiplication	/	Division
mod	Reste de la division		

4) Opérateurs de comparaison

***Tableau 3:** Opérateur de Comparaison du langage ST*

Opérateur	Description	Opérateur	Description
=	Egal à	<>	Différent de
>	Supérieur à	>=	Supérieur ou égal à
<	Inférieur à	<=	Inférieur ou égal à

III.1.1.2) Les Structures de contrôle :

1) Instructions conditionnelles :

***Tableau 4:** Instructions conditionnelles du langage ST*

pseudo langage	Langage ST	Langage C
<p>Si (<expression logique>)</p> <p> Alors</p> <p> Bloc action 1</p> <p>Sinon Si (expression logique)</p> <p> Alors</p> <p> Bloc action 2</p> <p> </p> <p> </p> <p> Sinon</p> <p> Bloc action 3</p> <p> Fin de Si</p>	<p>IF <expression logique></p> <p> THEN</p> <p> Bloc action 1 ;</p> <p> ELSIF <expression logique></p> <p> THEN</p> <p> Bloc action 2 ;</p> <p> </p> <p> </p> <p> ELSE</p> <p> Bloc action 3 ;</p> <p> END_IF</p>	<p>if (<expression logique>)</p> <p> {</p> <p> Bloc action 1 ;</p> <p> }</p> <p> else if (<expression logique>)</p> <p> {</p> <p> Bloc action 2 ;</p> <p> }</p> <p> </p> <p> </p> <p> else</p> <p> {</p> <p> Bloc action 3 ;</p> <p> }</p>

2) Instruction de choix :

Tableau 5: Instruction de choix du langage ST

pseudo langage	Langage ST	Langage C
Décider sur (<expression>)	CASE <expression>	switch (<expression>)
Entre	OF	{
Val_1 :	Val_1 :	case Val_1 :
Bloc action	Bloc action	Bloc action ; break ;
.....
.....
Val_2, Val_3, ... :	Val_2, Val_3, ... :	case Val_2 :
Bloc action	Bloc action	case Val_3 :
.....	case
.....	Bloc action ; return(n) ;
Val_i à Val_j :	Val_i.Val_j :
Bloc action	Bloc action
.....	Pas de correspondance en C
.....
default :	ELSE
Bloc action	Bloc action	default :
Fin de Décider	END_CASE ;	Bloc action ;
		}

3) Instructions répétitives :

Tableau 6: Instructions répétitives du langage ST

pseudo langage	Langage ST	Langage C
<p>Tant que (<expression logique>)</p> <p>Bloc action</p> <p>Répéter</p>	<p>WHILE <expression logique></p> <p>DO</p> <p>Bloc action ;</p> <p>END_WHILE ;</p>	<p>While (<expression logique>)</p> <p>{</p> <p>Bloc action ;</p> <p>}</p>
<p>Répéter</p> <p>Bloc action</p> <p>Tant que (<expression logique>)</p>	<p>REPEAT</p> <p>Bloc action ;</p> <p>UNTIL <expression logique></p> <p>END_REPEAT ;</p>	<p>do</p> <p>{</p> <p>Bloc action ;</p> <p>}</p> <p>While (<expression logique>);</p>
<p>Pour variable = valeur initiale</p> <p>Jusqu'à valeur final</p> <p>Par incrément</p> <p>Faire</p> <p>Bloc action</p> <p>Fin de Pour</p>	<p>FOR variable := valeur initiale</p> <p>TO valeur final</p> <p>BY incrément</p> <p>DO</p> <p>Bloc action ;</p> <p>END_FOR ;</p>	<p>For (variable = valeur initiale;</p> <p>variable <> valeur final;</p> <p>variable += incrément)</p> <p>{</p> <p>Bloc action ;</p> <p>}</p>

4) Instruction « EXIT » :

L'instruction « EXIT » permet d'interrompre une boucle répétitive.

Langage ST	Langage C
<i>Début de boucle</i>	<i>Début de boucle</i>
<i>Bloc action ;</i>	<i>Bloc action ;</i>
EXIT ;	break ;
<i>Bloc action ;</i>	<i>Bloc action ;</i>
<i>Fin de boucle ;</i>	<i>Fin de boucle ;</i>

5) Instruction « RETURN » :

L'instruction « RETURN » permet d'interrompre une unité d'organisation de programme comme une fonction, un bloc fonctionnel ou un programme. Contrairement au langage C l'instruction « RETURN » ne permet pas de retourner une valeur.

III.1.2) Listes d'instructions (IL, Instruction list) :

Le langage IL (Instruction List), est un langage textuel de bas niveau proche de L'assembleur.

Il est particulièrement adapté aux applications de petite taille, ou à l'optimisation de portions réduites d'une application. Les instructions travaillent toujours sur un *résultat courant* (ou registre IL).L'opérateur indique le type d'opération à effectuer entre le résultat courant et l'opérande. Le résultat de l'opération est stocké à son tour dans le résultat courant.

Tableau 7: Structure du langage IL

Label	Opérateur	Opérande	Commentaires
<Etiquette>:	<Opérateur>(<Modifieur>)	<Opérande>	(* Accumulateur = Accumulateur <Opérateur>(<Modifieur>) <Opérande> *)
TOTO :	ADD	25	(* Accumulateur = Accumulateur + 25 *)

Le label ou étiquette se termine toujours par le caractère « : ». Il est optionnel et Il précède une ligne d'instruction. Le label est utilisé lors de sauts conditionnels ou non.

L'opérateur correspond à l'opération à réaliser entre l'accumulateur et l'opérande. Le résultat est stocké dans l'accumulateur. Un modificateur peut être associé à l'opérateur.

L'opérande est une constante ou une variable. Il est possible d'associer à chaque ligne d'instruction un commentaire. Le début de commentaire se compose des deux caractères « (* » et la fin de commentaire se compose des deux caractères « *) ».

III.1.2.1) Les Operateurs :

1) Opérateurs d'affectation :

Tableau 8: Opérateurs d'affectations du langage IL

Opérateur	Description	Opérateur + Modificateur	Description
LD	Accu=Opérande(Op)	LDB	Accu= not Op
ST	Op=Accu	STN	Op= not Accu
S	Si Accu=1 alors Set Op		
R	Si Accu=1 alors Reset Op		

2) Opérateurs logiques :

Tableau 9: Opérateurs logique du langage IL

Opérateur	Description	Opérateur+ Modificateur	Description
AND	Accu=Accu AND Op	ANDN	Accu=Accu AND not Op

OR	Accu=Accu OR Op	ORN	Accu=Accu OR not Op
XOR	Accu=Accu XOR Op	XORN	Accu=Accu XOR not Op
NOT	Accu=not Accu		

3) Opérateurs arithmétiques :

Tableau 10: Opérateurs arithmétiques du langage IL

Opérateur	Description	Opérateur	Description
ADD	Accu=Accu + Op	SUB	Accu=Accu -Op
MUL	Accu=Accu *Op	DIV	Accu=Accu /Op
MOD	Accu=Rest de Accu/Op		

4) Opérateurs de comparaison :

Tableau 11: Opérateur de comparaison du langage IL

Opérateur	Description	Opérateur	Description
GT	Accu=Vrai si Accu>Op	GE	Accu=Vrai si Accu>=Op
LT	Accu=Vrai si Accu<Op	LE	Accu=Vrai si Accu<=Op
EQ	Accu=Vrai si Accu=Op	NE	Accu=Vrai si Accu<>Op

Branchements :

Tableau 12: Branchement dans le langage IL

Opérateur	Description	Opérateur	Description
JMPC	Saut à Label si Accu <>0	JMPCN	Saut à Label si Accu =0
JMP	Saut à Label	CAL	Appel d'un bloc de fonction
RET	Instruction « return »		
RETC	« return » si Accu <>0	RETCN	« return » si Accu=0

Modificateur « (» et Opérateur «) » :

Le modificateur « (» peut être associé aux opérateurs logiques, arithmétiques et de comparaison (par exemple « ADD(»), Il provoque l'ouverture d'une parenthèse par la création d'un deuxième accumulateur. L'opérateur «) » provoque la fermeture de parenthèse.

Exemple :

Res = (A + B) * (C + D)

LD A (* Accu = A *)

ADD B (* Accu = Accu + B = A + B *)

MUL(C (* Ouverture de parenthèse, Accu2 = C *)

ADD D (* Accu2 = Accu2 + D = C + D *)

) (* Fermeture de parenthèse Accu = Accu * Accu2 = (A + B) * (C + D) *)

ST Res (* Res = Accu *)

III.2) Langages graphiques :

III.2.1) Le langage à contact (LD, Ladder Diagram) :

Le langage LD est une représentation graphique d'équations booléennes combinant des contacts (en entrée) et des relais (en sortie). Le langage LD permet la manipulation de données booléennes, à l'aide de symboles graphiques organisés dans un diagramme. Les symboles du diagramme LD sont organisés comme les éléments d'un schéma électrique à contacts. Les diagrammes LD sont limités à gauche et à droite par des barres d'alimentation.

III.2.1.1) Bases du langage LD

Un programme LD est une suite d'équations appelées *échelles* où les contacts et les bobines de relais sont arrangés. Voici les composants de base d'un diagramme LD:

Début d'échelle (barre d'alimentation à gauche)

Chaque échelle commence par une barre d'alimentation à gauche, qui représente l'état VRAI.

L'éditeur LD crée automatiquement la barre d'alimentation à gauche quand le premier contact de l'échelle est placé par l'utilisateur. Chaque échelle peut avoir un nom logique qui peut être utilisé comme une étiquette dans les instructions de saut.

Fin d'échelle (barre d'alimentation à droite)









Une échelle se termine par une barre d'alimentation à droite, l'éditeur LD ajoute automatiquement la barre d'alimentation à droite de chaque bobine de relais posée par l'utilisateur.

Les contacts

Un contact modifie le flux de données booléen, selon la valeur de la variable booléenne qui lui est associée. Le nom de la variable est affiché en dessus du dessin du contact.

Les types de contacts suivants sont supportés par l'éditeur LD:




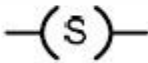

Tableau 13: Les contacts du langage LD

Graphe	Désignation	Fonction	Schéma à contact
	Contact à fermeture	contact passant quand il est actionné	
	Contact à ouverture	contact passant quand il n'est pas actionné	
	connexion horizontale	permet de relier les éléments action série	
	connexion verticale	permet de relier les éléments action en parallèle	

Les relais

Une bobine de relais représente une action. La variable associée au relais est forcée selon l'état de l'échelle à gauche du relais. Le nom de la variable est affiché en dessus du dessin du relais. Les types de relais suivants sont supportés par l'éditeur LD:

Tableau 14: Les relais du langage LD

Graphe	Désignation	Fonction	Schéma à contact
	bobine directe	la sortie prend la valeur du résultat logique	
	bobine inverse	la sortie prend la valeur inverse du résultat logique	
	bobine d'enclenchement	le bit interne est mis à 1 et garde cet état	
	bobine déclenchement	le bit interne est mis à 0 et garde cet état	

Blocs fonctionnels

Un bloc dans un diagramme LD peut représenter un opérateur, une fonction, un sous programme ou un bloc fonctionnel. Sa première entrée et sa première sortie sont toujours connectées sur l'échelle. Les autres paramètres en entrée et en sortie sont écrits en dehors du rectangle du bloc.

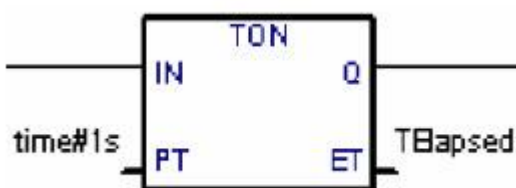


Figure 1.13: Bloc fonctionnel du langage LD

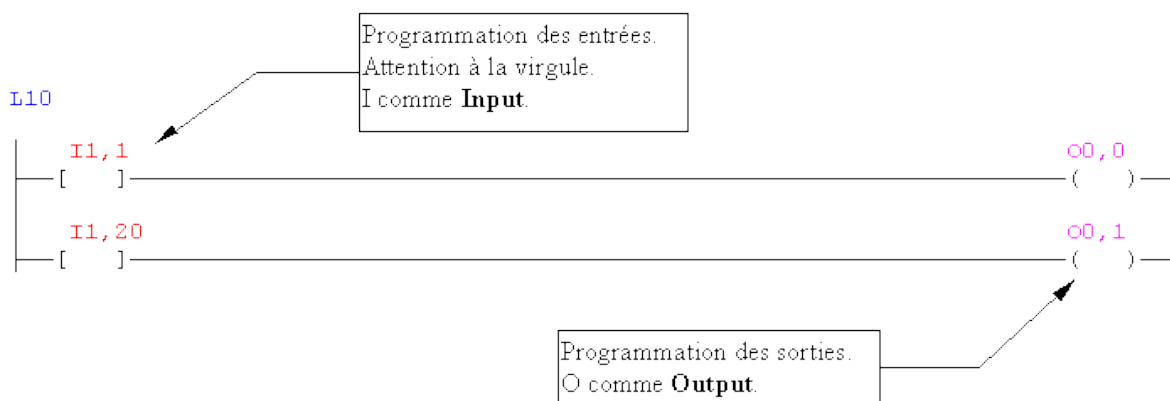
Symbole de saut

Un symbole de saut se réfère toujours à une étiquette (nom d'échelle) définie ailleurs dans le même diagramme LD. Le symbole de saut est posé à la fin d'une échelle. Quand l'état de la liaison à gauche du symbole est TRUE, l'exécution du diagramme est dérivée vers l'échelle identifiée par l'étiquette associée au saut. Notez que les sauts en arrière sont dangereux car ils peuvent bloquer le cycle automate.

Symbole "Return"

Un symbole "Return" est placé à la fin d'une échelle. Il indique que l'exécution du programme doit être abandonnée si l'état de la liaison est TRUE. Un symbole "Return" est équivalent à un saut après la dernière échelle du diagramme.

Exemple :



Le fonctionnement de ce réseau ladder est le suivant :

SI j'ai l'entrée I1,1 ALORS j'active la sortie O0,0 (la sortie O0,0 = 1).

SI j'ai l'entrée I1,20 ALORS j'active la sortie O0,1.

III.2.2) LANGAGE Logigramme (FBD, Function Block Diagram) :

Le FBD (Function Block Diagram) est un langage graphique. Il permet la construction d'équations complexes à partir des fonctions élémentaires.

Le diagramme FBD décrit une fonction entre des *variables d'entrée* et des *variables de sortie*. Une fonction est décrite comme un réseau de *fonctions élémentaires*. Les variables d'entrée et de sortie sont connectées aux boîtes fonctions par des arcs de liaison. Une sortie d'une boîte peut être connectée sur une entrée d'une autre boîte.

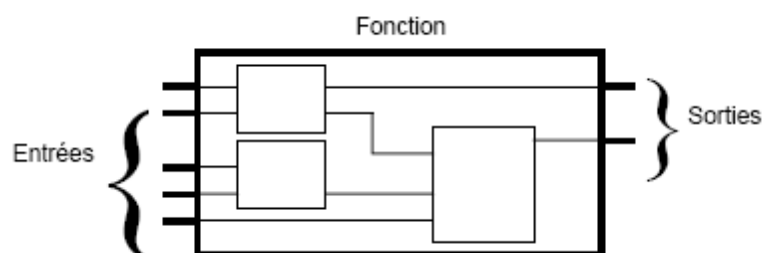


Figure 1.14: Structure du langage FBD

Chaque boîte fonction élémentaire a un nombre prédéfini de *points de connexion* en entrée et en sortie. Une boîte fonction est représentée par un *rectangle*. Ses entrées sont connectées sur le bord *gauche* du rectangle, et Ses sorties bord *droit*.

Une boîte fonction élémentaire réalise une fonction élémentaire entre ses entrées et ses sorties. Le nom de la fonction réalisée est inscrit dans le rectangle de la boîte. Chaque entrée ou sortie de la boîte est caractérisée par un *type*.

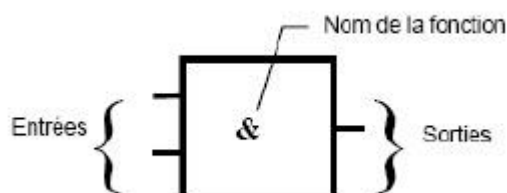


Figure 1.15: Fonction élémentaire du langage FBD

Les variables d'entrée du diagramme FBD doivent être connectées aux entrées des boîtes fonctions. Le type de chaque variable doit être cohérent avec le type de l'entrée de la boîte correspondante. Une entrée pour le diagramme FBD peut être une expression *constante*, toute variable *interne*, d'*entrée* ou de *sortie*.

Les variables de sortie du diagramme FBD doivent être connectées aux sorties des boîtes fonctions. Le type de chaque variable doit être cohérent avec le type de la sortie de la boîte correspondante. Une sortie pour le diagramme FBD peut être une variable *interne* ou de *sortie*. Des lignes sont utilisées pour représenter les liaisons entre deux points du diagramme :

- Une variable d'entrée et une entrée de boîte
- Une sortie d'une boîte et une entrée d'une autre boîte
- Une sortie de boîte et une variable de sortie

Les liaisons sont *orientées*: une liaison transporte une information depuis son extrémité gauche vers son extrémité droite. Les extrémités gauche et droite doivent être connectées à des

éléments de même type. Des liaisons multiples à droite sont utilisées pour transmettre une information depuis une extrémité à gauche vers plusieurs points à droite. Toutes les extrémités de la liaison doivent avoir le même type.

III.2.3) Langage séquentiel (SFC, Sequential Function Chart) :

Le « Sequential Function Chart » (SFC) est un outil de programmation. Ce langage est inspiré du GRAFCET (norme CEI 60848), mais il n'y a aucune identité entre les deux représentations, graphiques et sémantique des deux langages.

Le langage SFC est un langage graphique, et donc visuel. Il permet de bien représenter les différents états d'un système séquentiel. Associé aux autres langages (IL, ST, LD), il permet d'obtenir une programmation claire et lisible.








Le SFC propose un moyen de structuration, et d'organisation interne, d'une unité d'organisation de programme, à travers un ensemble de « steps », de « transitions », et de liaisons orientées. Un ensemble d'actions est associé aux étapes, et des conditions sont associées aux transitions.

III.2.3.1) BASES DU LANGAGE SFC :

Le langage SFC décrit des comportements séquentiels. Il divise le procédé en un nombre d'étapes connues (situations stables), séparées par des transitions.

Les symboles d'un graphe SFC sont reliés par des *liaisons orientées*. L'orientation par défaut est *du haut vers le bas*. Voici les composants graphiques utilisés pour la construction d'un graphe SFC:

Tableau 15: Composants graphique du langage SFC

Symbole	Signification	Symbole	Signification
	Etape initiale		Etape
	Transition		Renvoi à une étape
	Macro-étape		Etape de début d'une macro
	Etape de fin d'une macro		

La programmation SFC se décompose en deux niveaux. Le *niveau 1* montre le graphe, les numéros de référence et les commentaires attachés aux étapes et aux transitions. Le *niveau 2* est la programmation en ST ou IL des actions dans les étapes et des conditions attachées aux transitions. Les actions et conditions peuvent appeler des *sous-programmes* écrits dans d'autres langages (FBD, LD, ST ou IL). Voici un exemple de programmation de niveau 1 et de niveau 2.

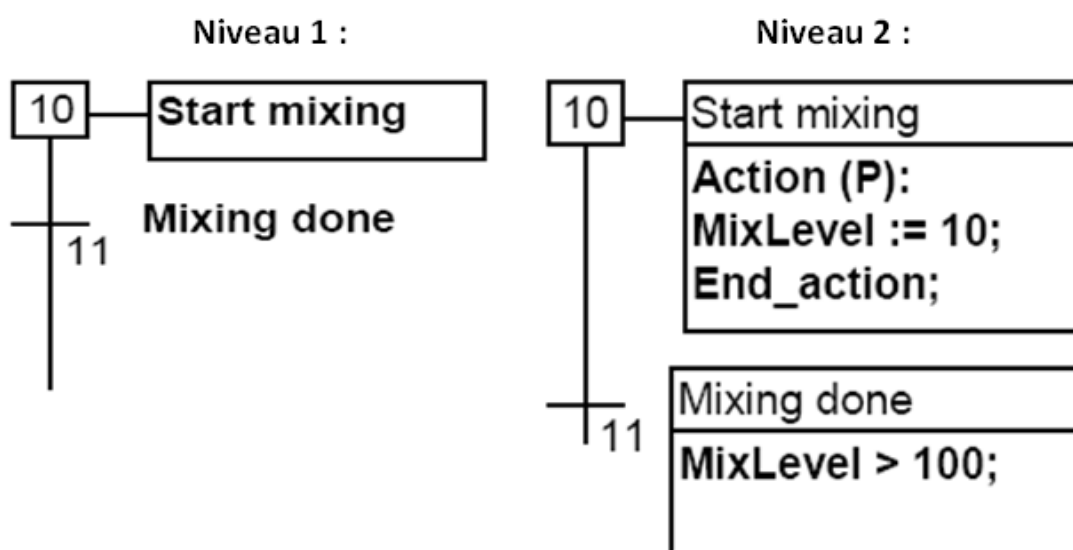


Figure 1.16: les éléments constituant d'un programme SFC

Le niveau 2 d'une étape est entré à l'aide d'un éditeur de textes. Il contient des blocs d'action écrits en ST ou IL. Le niveau 2 d'une transition peut être saisi soit en langage texte IL ou ST, soit avec l'éditeur LD (schéma à contacts).

Divergences et convergences

Les divergences et les convergences sont utilisées pour représenter des liaisons multiples entre les étapes et les transitions. Les divergences et convergences simples représentent plusieurs possibilités (inclusives) dans l'exécution du procédé.

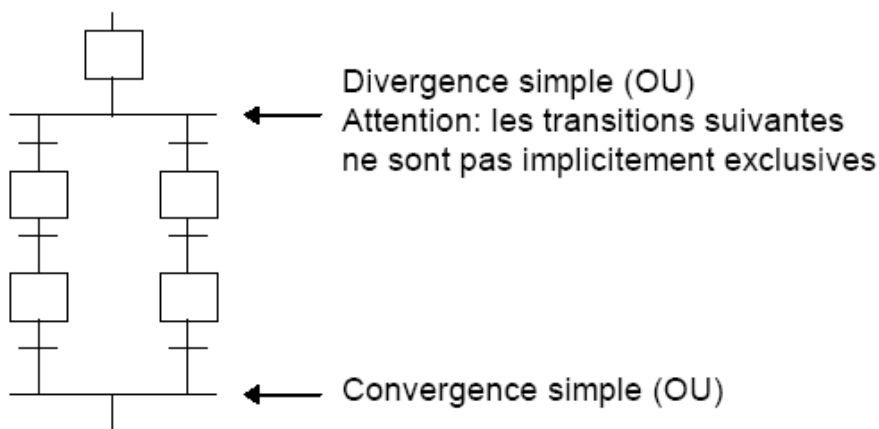


Figure 1.17: Divergences et convergences simple

Les divergences doubles représentent des procédés **parallèles**.

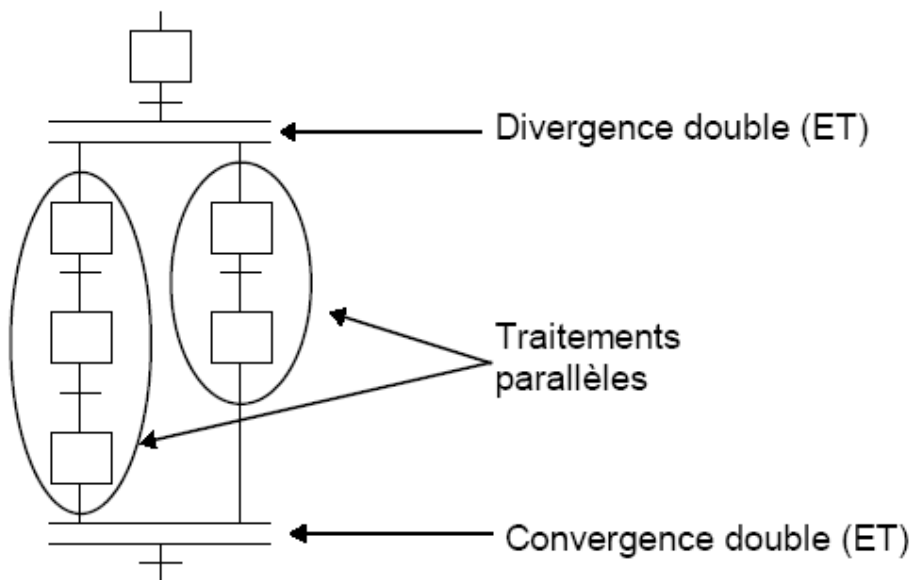


Figure 1.18: Divergences et convergences double

Saut à une étape

L'éditeur SFC ne visualise que les liaisons tracées du haut vers le bas. Un *renvoi* à une étape peut être utilisé pour représenter un renvoi vers une partie antérieure du graphe.

Exemple:

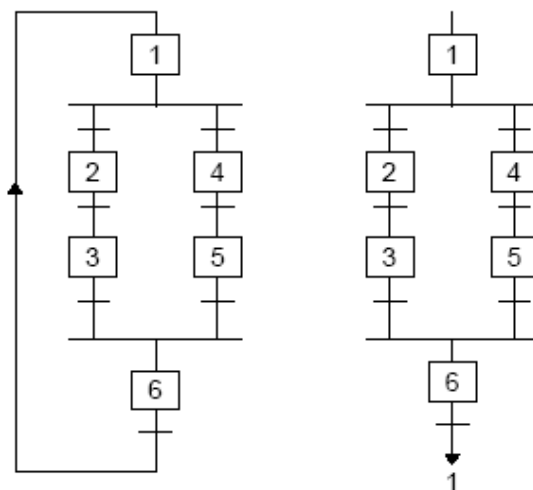


Figure 1.19: Saut a une étape dans un programme SFC

Un renvoi à une transition est interdit, et doit être explicitement représenté comme une convergence double.

Macro-étape

Une macro-étape est la représentation sous la forme d'un symbole unique d'un groupe *unique* et *connexe* d'étapes et de transitions. Le corps d'une macro-étape commence par une *étape de début* et se termine par une *étape de fin*.

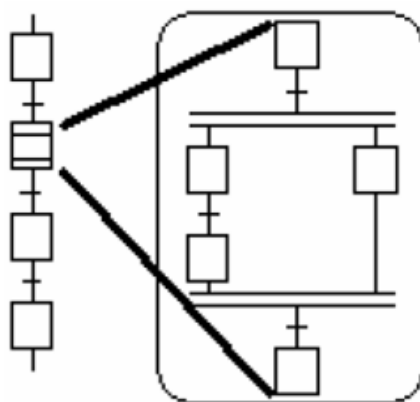


Figure 1.20: Macros-étape dans un programme SFC

La représentation de la macro-étape doit être décrite dans le même programme SFC. Le symbole de la macro-étape et son étape de début doivent avoir le même *numéro de référence*. Le corps d'une macro-étape peut contenir le symbole d'une autre macro-étape.

IV) Comparaison des langages

Tableau 16: Comparaison des langages

LANGAGE	AVANTAGES	INCONVENIENTS
LD	facile à lire et à comprendre par la majorité des électriciens langage de base de tout PLC	suppose une programmation bien structurée
FBD	Très visuel et facile à lire	Peut devenir très lourd lorsque les équations se compliquent
ST	Langage de haut niveau (langage pascal) Pour faire de l'algorithmique	Pas toujours disponible dans les ateliers logiciels
IL	langage de base de tout PLC type assembleur	très lourd et difficile à suivre si le programme est complexe Pas visuel.
SFC	Description du fonctionnement (séquentiel) de l'automatisme. Gestion des modes de marches Pas toujours accepté dans l'industrie...	Peu flexible

Chapitre IV : SIMATIC Step7

I) Introduction :

STEP7 est le logiciel de base pour la programmation et la configuration de systèmes d'automatisation SIMATIC. Il permet : la création et la gestion de projets, la configuration et le paramétrage du matériel et de la communication, la gestion des mnémoniques, la création de programmes.

Il inclut 6 applications :

1. Gestionnaire de projets SIMATIC : il gère toutes les données relatives à un projet d'automatisation. Il démarre automatiquement les applications requises pour le traitement des données sélectionnées.

2. Editeur de mnémoniques : il permet de gérer toutes les variables globales. C'est-à-dire la définition de désignations symboliques et de commentaires pour les signaux du processus (entrées/sorties), mémentos et blocs, l'importation et l'exportation avec d'autres programmes Windows.

3. Diagnostic du matériel : il fournit un aperçu de l'état du système d'automatisation. Dans une représentation d'ensemble, un symbole permet de préciser pour chaque module, s'il est défaillant ou pas. De plus permet l'affichage d'informations générales sur le module et son état, l'affichage d'erreurs sur les modules de la périphérie centrale et des esclaves DP et l'affichage des messages de la mémoire tampon de diagnostic.

4. Langages de programmation : trois langages de programmation sont inclus dans le logiciel de base : CONT (LD Ladder Diagram), LIST (IL Instruction List) et LOG (FBD Function Bloc Diagram), d'autre langage de programmation peuvent être procurés sous forme de logiciel additionnel : le SCL (ST Structured Text) et le GRAPH (GRAFSET).

5. Configuration matérielle : il permet de configurer et paramétrer le matériel d'un projet d'automatisation. Il suffit juste de sélectionner le châssis (Rack) dans un catalogue électronique et leurs affecter les modules sélectionnés aux emplacements souhaités dans les racks (CPU, SM, FM...). De plus il permet le paramétrage de la CPU (comportement à la mise

en route, surveillance du temps de cycle), des modules fonctionnels (FM) et de processeurs de communication (CP).

6. NetPro : il permet le transfert de données via MPI tout en offrant les possibilités de choisir les participants à la communication et de définir les liaisons de communication.

II) Interaction du logiciel et du matériel

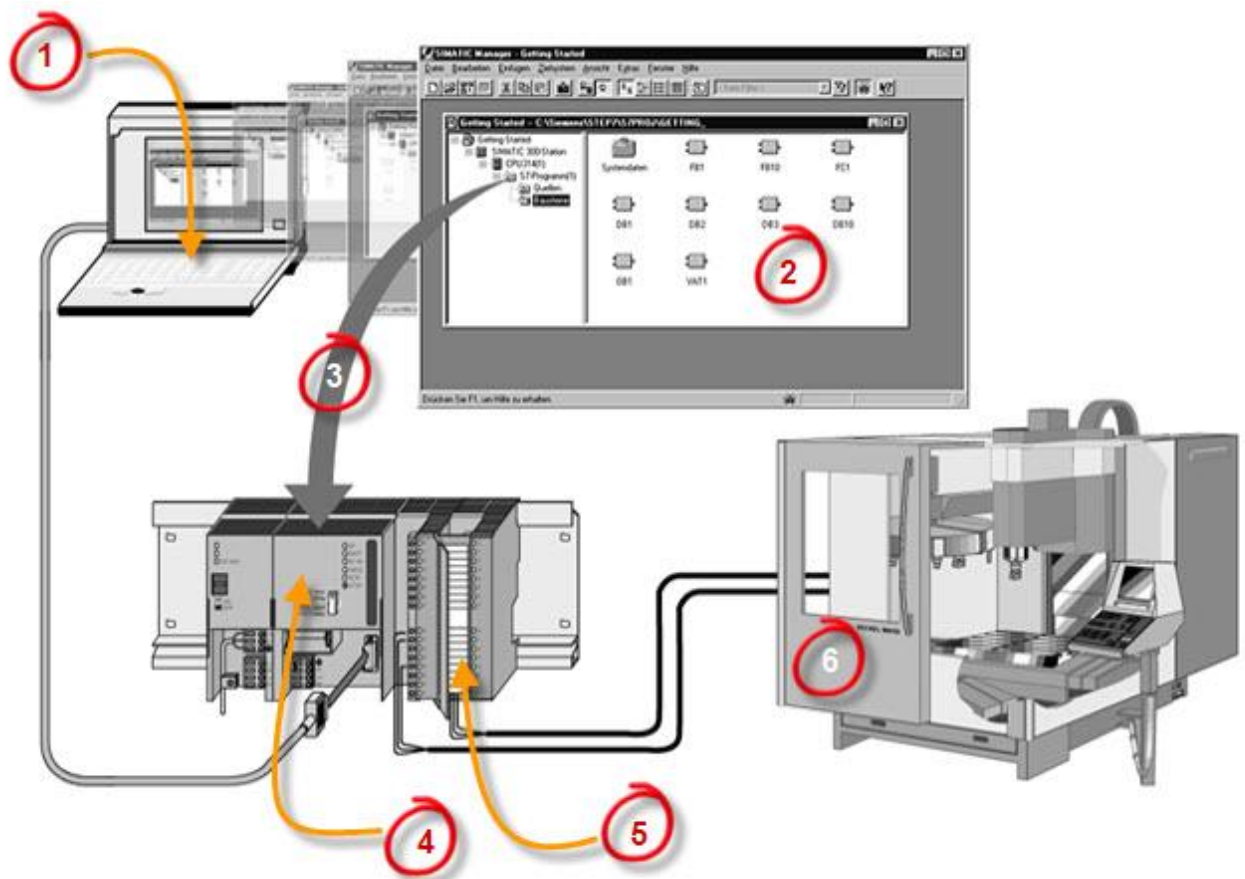


Figure 1.21: Interaction du logiciel et du matériel

- | | |
|--------------------------------|----------------------------------|
| 1) Console de Programmation | 4) CPU d'api |
| 2) Logiciel STEP 7 | 5) Module de sorties |
| 3) Transfert du programme créé | 6) Machine devant être commandée |

III) Structure du projet dans SIMATIC Manager

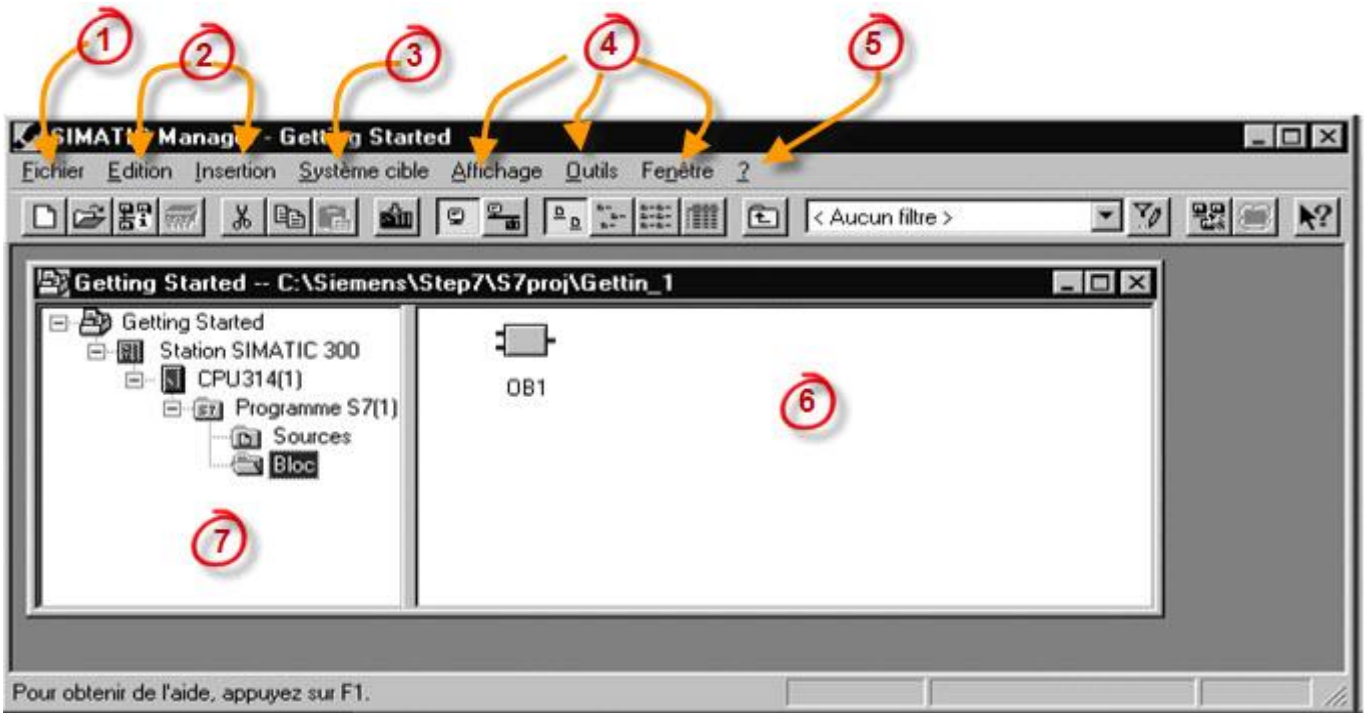


Figure 1.22: Structure du projet dans SIMATIC Manager

- 1) Ouvrir, organiser et imprimer les projets
- 2) Editer les blocs et insérer les éléments de programme
- 3) Charger le programme Et surveiller le matériel
- 4) Choisir la représentation et la disposition des fenêtres, choisir la langue et sélectionner diverses options pour les données du processus
- 5) Appeler l'Aide de STEP 7
- 6) Le contenu de la fenêtre de droite affiche les objets et les dossiers du dossier sélectionné à gauche.
- 7) Le contenu de la fenêtre de gauche affiche la structure du projet.

IV) Création d'un projet STEP7

Un projet comprend deux données essentielles, les programmes et la configuration du matériel, on peut commencer par définir l'une ou l'autre, mais tout d'abord il faut démarrer le programme SIMATIC Manager. Ce programme est l'interface graphique qui permet la manipulation du projet et l'accès aux autres programmes de STEP7.

Pour en créer un nouveau, il suffit de cliquer sur le bouton '*Nouveau projet*', attribuer un nom et valider. Ensuite il faut choisir une station de travail.

Une station SIMATIC représente une *configuration matérielle S7* comportant un ou plusieurs modules programmables. Il existe différents types:

SIMATIC 400 : Automate à performances extrêmes, adapté à l'exécution de programme de lourds calculs.

SIMATIC 300 : Automate à extensibilité modulaire.

SIMATIC H : Automate insensible aux défaillances, il se compose de 2 CPU du même type, en cas de problème elle commute de l'une vers l'autre sans perte de données.

SIMATIC PC : ou Station PC, représente un PC ou une station OS contenant des composants SIMATIC : des applications (WinCC, par ex.), un automate logiciel ou une carte CPU enfichée dans le PC.

Autres stations : se sont soit des appareils d'autres fabricants ou bien des stations de SIMATIC S7 contenus dans un autre projet.

SIMATIC S5 : liaison vers un projet S5.

PG/PC : Outils de programmation pour contrôleurs SIMATIC, c'est une console de programmation compatible avec le milieu industriel.

Par exemple on va choisir une station SIMATIC 300

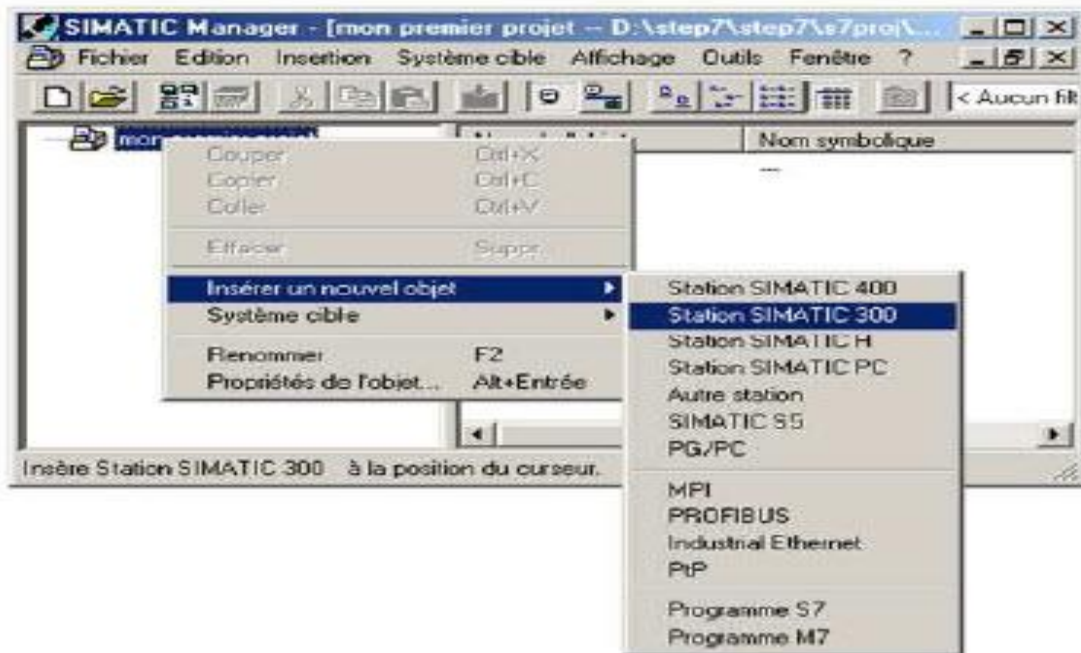


Figure 1.23: choix de la station de travail

Pour commencer, le plus simple est de configurer le matériel, d'éditer les programmes puis les charger dans la CPU. Double clique sur 'Matériel' démarre l'application HW Config.

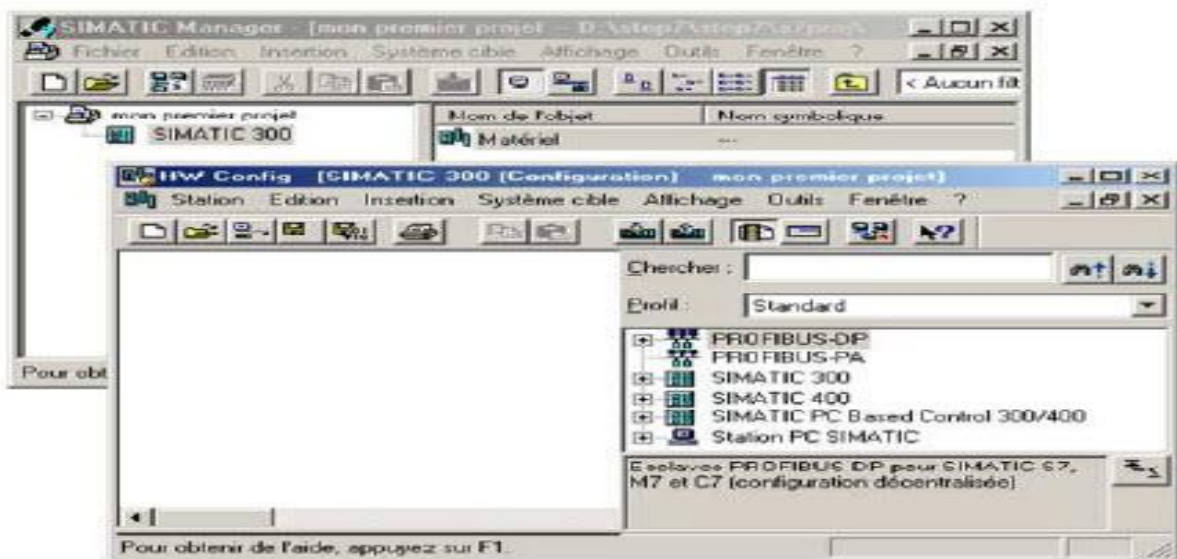


Figure 1.24: configuration de matériel.

III.1) Configuration du matériel

Pour configurer le matériel, il suffit de faire glisser des éléments du catalogue dans l'emplacement approprié, on choisit le 'Rack', l'alimentation, la CPU et les E/S...

Dans le catalogue on trouve les modules qu'on peut affecter à chaque type de station, on distingue:

C7 : Système intégré compact qui regroupe automate programmable et interface homme machine (pupitre opérateur) pour la réalisation de commandes de machines sous encombrement réduit.

CP : Communication Processor, module de communication (PROFIBUS, Industriel Ethernet, Peer to Peer...).

FM : Function Module, il regroupe les modules de fonctions (regulation, comptage...).

IM : Coupleurs d'extension, il permet l'ajout d'autres modules.

M7 : Modules d'extension et cartouches interface pour SIMATIC M7.

PS : Module d'alimentation.

Rack : Support mécanique.

Routeur : Relie Industriel Ethernet à PROFIBUS.

SM : Signal Module, c'est le module d'E/S, il contient le AI module d'entrées analogiques, le AO module de sorties analogiques, le DI module d'entrées TOR et le DO module de sorties TOR.

CPU : L'Unité Centrale, noté CPU xxx a b. xxx est la famille de la CPU a, b sont les propriétés de la CPU (éléments additionnels, port de communication...). Par exemple :

C : compact, la CPU intègre des modules E/S ainsi que des fonctions spécialisées.

PtP : Peer to Peer, la CPU intègre un port de communication Point to Point.

H : Fault-tolerant, des unités de traitements insensibles aux défaillances

DP : Decentralized Periphery, la CPU intègre un port de communication PROFIBUS.

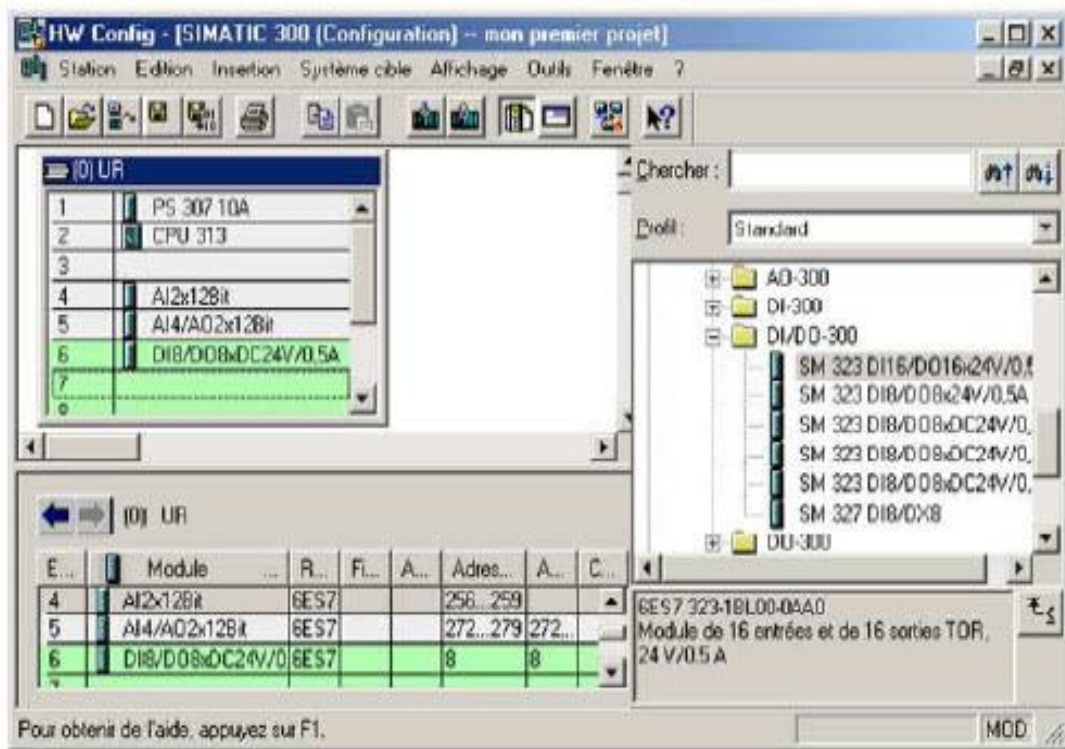


Figure 1.25: Sélections des modules

Si l'insertion de l'élément choisie est possible dans le Rack, la case appropriée devient verte. Une fois le matériel choisi on sauvegarde, on compile et on charge dans la CPU.

III.2) Définition des mnémoniques

De retour dans le SIMATIC Manager, on trouve de nouveaux éléments. On commence par créer les mnémoniques dans la section programmes.

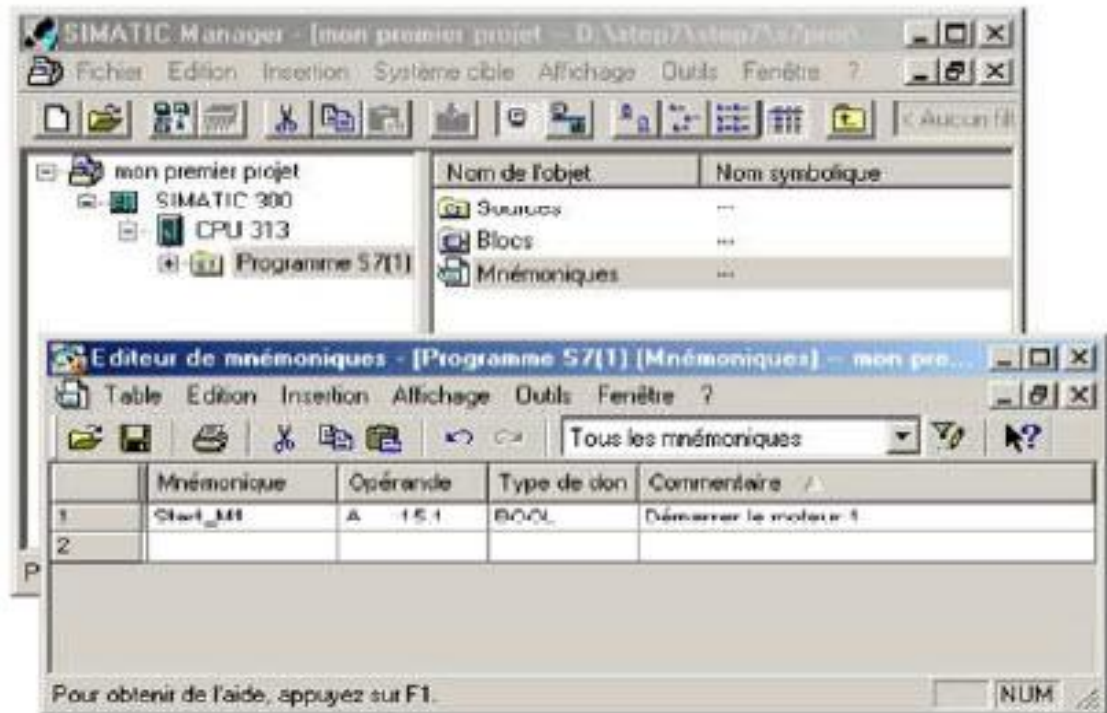


Figure 1.26: édition des mnémoniques

En affectant des noms symboliques aux adresses absolues, les programmes deviennent plus lisibles, faciles à corriger et à mettre à jour.

Il y a quatre différents types d'opérande : le bit, l'octet, le mot et le double mot. Ces types définissent l'accès à une zone mémoire.

III.3) Edition des programmes

Dans la section 'bloc' du SIMATIC Manager, on trouve par défaut le bloc d'organisation 1

'OB1' qui représente le programme cyclique. On peut rajouter d'autres blocs à tout moment par une clique droite dans la section Bloc de SIMATIC Manager.

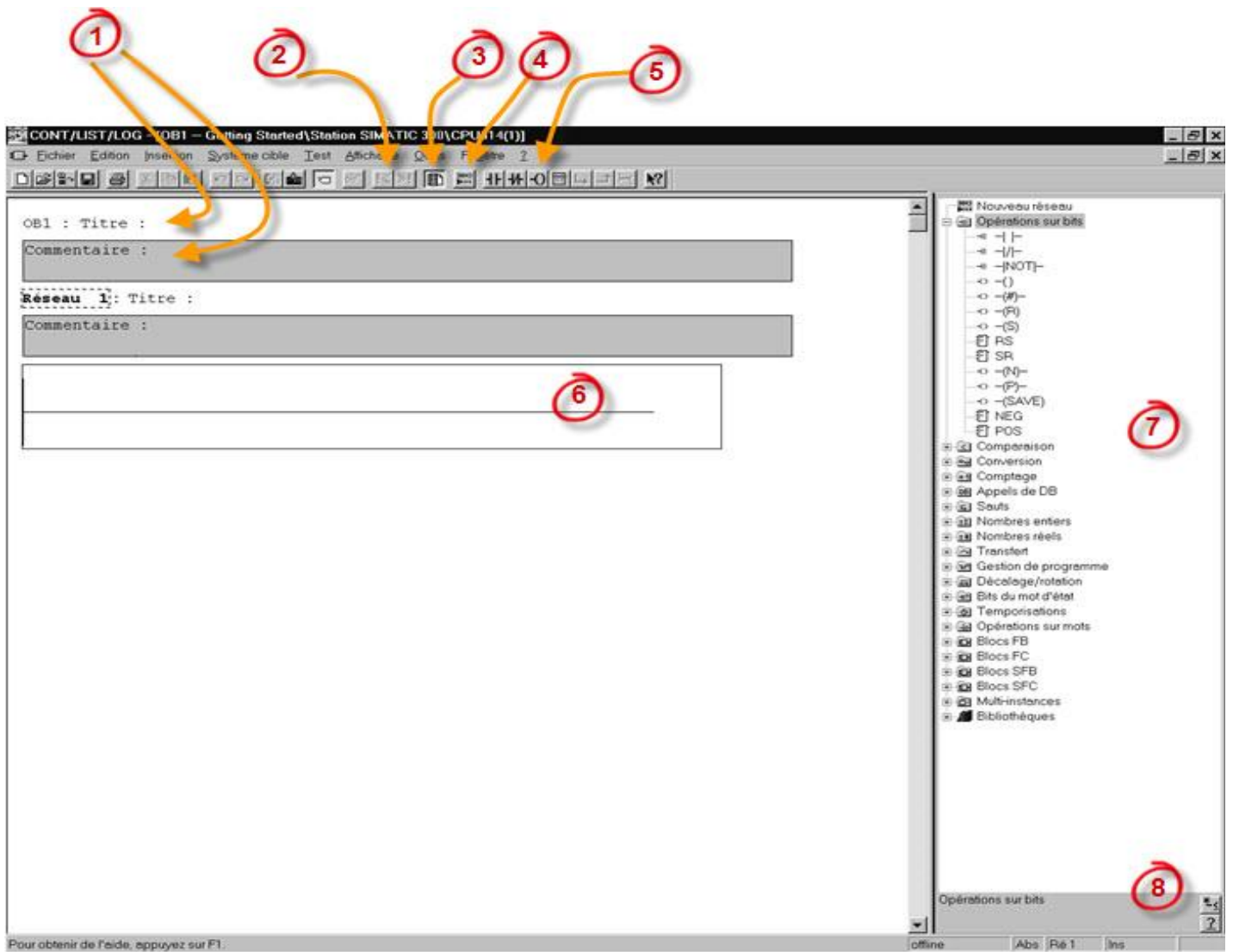


Figure 1.27: L'éditeur de programme CONT

- 1) Modifier la vue du langage De programmation
- 2) Afficher, masquer le catalogue des éléments de programme
- 3) Insérer un nouveau réseau
- 4) Principaux éléments de programme CONT et LOG
- 5) Titre et zone de commentaire du bloc ou du réseau
- 6) Ligne de saisie du programme (encore appelée Réseau ou Branche de courant)
- 7) Catalogue des éléments de programme, ici CONT
- 8) Description succincte de l'élément de programme sélectionné

Deux programmes différents s'exécutent dans la CPU: le système d'exploitation et le programme utilisateur.

Le système d'exploitation, organise toutes les fonctions et procédures dans la CPU qui ne sont pas liées à une tâche d'automatisation spécifique. Il gère le déroulement du démarrage à chaud et du redémarrage, l'actualisation de la mémoire image des entrées et l'émission de la mémoire image des sorties, l'appel du programme utilisateur, la gestion des zones de mémoire l'enregistrement des alarmes et l'appel des OB d'alarme...

Le programme utilisateur contient toutes les fonctions nécessaires au traitement des tâches d'automatisation spécifique. Ce programme doit être créé et chargé dans la CPU par l'utilisateur.

Il détermine les conditions pour le démarrage à chaud et le redémarrage de la CPU (par exemple, initialiser des signaux), il traite les données du processus (par exemple, combiner des signaux binaires, lire et exploiter des valeurs analogiques), il doit réagir aux alarmes et traiter les perturbations dans le déroulement normal du programme.

Le STEP 7 permet de structurer le programme utilisateur en le subdivisant en différentes parties autonomes ou dépendantes. Ceci permet d'écrire des programmes importants mais clairs, simples à tester et à modifier.

III.3.1) Blocs d'organisation

Les blocs d'organisation (OB) constituent l'interface entre le système d'exploitation et le programme utilisateur. Ils sont appelés par le système d'exploitation et gèrent le traitement du programme cyclique et des programmes déclenchés par alarmes, ainsi que le comportement à la mise en route de l'automate programmable et le traitement des erreurs.

Les blocs d'organisation définissent l'ordre (événements de déclenchement) dans lequel les différentes parties du programme sont traitées.

III.3.2) Fonctions et blocs fonctionnels

On peut programmer chaque bloc d'organisation en tant que programme structuré en créant des fonctions (FC) et des blocs fonctionnels (FB)

· Les blocs fonctionnels (FB) sont des blocs de code associés à des blocs de données d'instance, dans les quels sont sauvegardés les paramètres effectifs et les données statique des blocs fonctionnels.

Les fonctions (FC) sont des blocs de code sans rémanence, c'est-à-dire qu'ils ne sont pas associée à des blocs de données, les paramètres effectifs ne sont pas sauvegardés automatiquement.

De plus il existe les blocs fonctionnels système (SFB) et les fonctions système (SFC), qui sont des fonctions préprogrammés. Ils peuvent être appelés à partir du programme utilisateur. On trouve des fonctions système pour la copie de blocs de données, le contrôle du programme utilisateur, la gestion des alarmes horaires et temporisées...

III.3.3) Bloc de données

Les blocs de données (DB) servent à l'enregistrement de données utilisateur. Les blocs de données globaux servent à l'enregistrement de données qui peuvent être utilisées par tous les autres blocs. Les blocs de données d'instances sont affectés à des blocs fonctionnels.

Les différents blocs cités ci-dessus peuvent être édités avec l'application 'CONT LIST LOG'.

III.4) Programmation des blocs

La programmation des blocs de codes peut se faire à l'aide de trois applications:

1) **CONT/ LIST /LOG** : elle permet de programmer des blocs d'organisations 'OB', des blocs fonctionnels 'FB' et des fonctions 'FC'.

2) **GRAPH** : elle permet de programmer des blocs fonctionnels 'FB'.

3) **SCL** : elle permet de créer des sources de code. Une source de code est un fichier texte, qui contient une suite d'instructions, une fois compilé il peut être transféré dans la CPU.

On peut trouver dans un même fichier source tout le programme utilisateur, c'est-à-dire les blocs d'organisations, les blocs fonctionnels et les fonctions.

Note : les deux dernières applications GRAPH et SCL n'existent pas dans la version de base du STEP7, il faut les installer séparément.

On va présenter comment éditer les Blocs de code avec ces trois applications.

CONCLUSION GENERALE

L'automate est un bon produit, facile à programmer, à connecter, adapté aux conditions industrielles. L'expansion considérable de ses possibilités, et celle corrélative de son marché, le prouvent. Il ne faut pas vouloir en faire une solution miracle. Dans tous les cas :

- une bonne analyse du problème à résoudre
- le respect des règles d'installation

Un léger surdimensionnement pour préserver des marges de modification, sont les conditions d'une implantation réussie, dont la durée de vie dépassera largement celles habituelles dans le monde informatique, dont l'API est pour partie issu.

BIBLIOGRAPHIE

- WILLIAM BOLTON. « Les automates programmables industriels », Traduit par Hervé Soulard, 5^{ème} édition.
- G.DÉCHENAU, « API et PC : solutions concurrentes ou complémentaires» Techniques de l'ingénieur, Vol. R8022.
- P.JARGOT, « Langages de programmation pour API. Norme IEC 1131-3 » Techniques de l'ingénieur, Vol. S8030.
- A.ABRICHE, « Réalisation et gestion d'un prototype de station de pompage à base d'automates programmables industriels SIEMENS » projet de fin d'étude Ecole Nationale Polytechnique, 2007.
- A.ABRICHE, « Réalisation et gestion d'un prototype de station de pompage à base d'automates programmables industriels SIEMENS » projet de fin d'étude Ecole Nationale Polytechnique, 2007.
- S.BENALIA, «Conception et mise en œuvre d'automatisation basee sur un pc» projet de fin d'étude Ecole Nationale Polytechnique, 2008.
- ETT, « User Manual For ET-PCI8255 V3 Card ».
- SIEMENS « Mise en route STEP7 V5.3 », Réf 6ES7810-4CA06-8CA0, SIMATIC 2004.
- SIEMENS « Langage CONT pour SIMATIC S7-300/400 », Réf 6ES7810 4CA06-8CR0,SIMATIC 2004
- SIEMENS « Langage LOG pour SIMATIC S7-300/400 », Réf 6ES7810-4CA06 8CR0,SIMATIC 2004
- SIEMENS « S7-300 and M7-300 Programmable Controllers Module Specifications »,Réf 6ES7- 3988-AA03-8BA0

- SIEMENS « Caractéristiques des CPU, CPU 312 IFM – 318-2 DP »
Réf 6ES7 312-5AC02-0AB0
- SIEMENS «SIMATIC STEP 7 V5.1 Getting Started», Réf 6ES7 810-4CA04-8CA0

Internet

- www.siemens.com
- www.PLCopen.org
- www.techniques-ingenieur.fr
- www.institut-numerique.org
- www.technologuepro.com