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

Université Dr. Tahar Moulay de Saïda
Faculté de Technologie
Département d'Electrotechnique

Mémoire de Fin d'Etudes

Master (LMD)

Spécialité : AUTOMATIQUES &SYSTEMES

Filière : AUTOMATIQUES

Intitulé :

Etude et simulation d'un système automatisé industriel par API (Automgen et Step7)

Présenté par :

BENKHEDDA Ahmed
BENSLIMANE Mohammed Riad

Devant le jury composé de :

Dr.Ch.leban	Président
Dr. M. Sekour	Encadreur
Dr. M. Mostfai	Examinateur
Mr.M.Makkaoi	Examinateur

Soutenu le 07/07/2019
Promotion 2018-2019

REMERCIEMENT

Je remercie « Allah » de nous avoir donné la force et le courage pour réaliser ce modeste travail.

Je remercie mon encadreur ***Dr. M. Sekour*** *qui a suivis de très près ce travail, pour son aide, son orientation, sa disponibilité et tous les conseils qu'il ma prodigué pendant toute la durée de ce travail.*

Je remercie aussi toutes les personnes qui m'ont aidé de près ou de loin. J'adresse mes plus vifs remerciements aux membres du jury pour l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de ma mémoire.

Je tiens à remercier l'ensemble des enseignants de la spécialité automatique qui ont aussi consacré leur temps et leur savoir à m'enseigner cette spécialité.

Enfin, j'exprime ma très grande reconnaissance à ma famille, à la mémoire de mon Père , à ma Mère, mon frère et Mes sœurs à tous mes proches pour leur encouragement, pour tous les soucis que je leur aie causés et surtout pour avoir été toujours auprès de moi par leur conseil et leur soutient.

DÉDICACES

Je dédie ce travail à mes chers parents," Mon père"et

" Ma mère" qui m'ont soutenu avec leurs Douae.

Et aider tout au long de mon parcourt, sans eux

je ne serai jamais arrivé là où je suis.

A mes très chères sœurs

A tout la famille

A mes amis qui ont étudié à mes côtés durant ces quatre dernières années de spécialité qui m'ont toujours encouragé durant mon cursus.

A tout mes amis de la promotion Automatique 2018/2019 Avec tous mes remerciements

Sommaire

CHAPITRE 2 : COMMANDE DES API & PROGRAMMATION STEP7

CHAPITRE 3 : COMMANDE DES API & PROGRAMMATION AUTOMGEN 8.9

Liste des figures

Liste des figures chapitre 1 :

Liste des figures chapitre 2 :

Liste des figures chapitre 3 :

Liste des tableaux

Liste des tableaux chapitre2 :

Liste des tableaux chapitre 3 :

SYMBOLES ET ABSEVIATIONS

AUT : Automatique

MAN : Manuel

FC : Fin Course

TOR : Tout ou Rie

API : Automate Programmable Industriel

PO : Partie Opérative

PC : Partie commande

E/S Entrée/Sortie

CPU : L'unité Centrale

CAN : Conversion Analogique Numérique

PS : Module d'Alimentation

UL : Unité Logique

UAL : Unité Arithmétique et Logique

SM : Module de signaux

LIST : Langage liste

LOG : langage logigramme

CONT : Langage contact

OB : Bloc d'Organisation

FB : Bloc fonctionnel

DB : Bloc de données

FC : Bloc Fonctions

IHM : Interface Homme Machine

MPI : Interface Multi Points

GRAFCET **:** GRAphe de Commande Etape Transition

Introduction générale

L'automatique est devenue indispensable dans l'industrie. Il a pour objectif de concevoir et d'étudier les diverses automatismes en mettant en œuvre les actionneurs électriques et pneumatiques et hydrauliques. Chaque système automatisé possède une partie commande et une partie opérative. Dans la partie commande, l'automate programmable représente l'élément principal de la machine ou de l'installation, car c'est celui qui renferme le programme et doit procéder à son exécution en fonction de l'état des entrées et des sorties, mais la partie opérative représente en général les moteur, les pompes, les agitateurs ou bien les paramètres gérés. L'automatisation a pris un grande place dans le milieu industriel .elle est devenue la nouvelle stratégie de production choisie par les plus grandes entreprises actuelles, en particulier le secteur de la production pétrolier qui joue un rôle très important dans notre pays.

Ce mémoire est composé par trois chapitres :

Le premier chapitre présente des généralités sur l'automatisme industriel, et les différentes structures de base d'un système automatisé.

Dans le deuxième chapitre, nous présenterons les différents aspects d'APIs. Nous nous intéresserons, également à la gamme de produits SIMATIC proposée par SIEMENS dans le cadre de l'automatisation de l'industrie en général, et plus spécialement à l'automate programmable industriel S7-300

Dans le chapitre trois la description et l'étude d'un mélangeur de peinture. Aussi l'architecture et le programme avec un simulateur ITS PLC du système automatisation de cette station utilisant un AUTOMGEN 8.9, sont discutés.

CHAPITRE 1

AUTOMATES PROGRAMMABLES

1. Introduction :

Dans toute la panoplie d'appareils utilisés pour commander les automatismes et les procédés de fabrication , l'automate programmable industriel (API) [1] , occupe une très importante.

La création du premier API remonte à la fin des années 60 , l'industrie automobile en est la principale instigatrice et la première utilisatrice . jusqu'alors , la commande des automatismes industriels était réalisée à l'aide d'armoires de commande à relais. Les changement annuels de modèle de voiture impliquaient des modifications fréquentes des chaines de montage et de leurs armoires de commande . comme ces dernières étaient complexes , leurs modifications étaient difficiles et comportaient un risque élevé d'erreur de branchement. l'industrie automobile a donc amené la création d'un appareil programmable capable de remplacer les armoires de commande .

Ce fut alors le début d'une grande aventure pour plusieurs compagnies. Les ordinateurs qui étaient principalement utilisés pour faire de la comptabilité furent modifiés afin de répondre aux exigences de la commande industrielle.

Petit à petit , la technique s'améliora et gagna plus d'adeptes , il fallu cependant attendre une bonne décennie avant que le concept soit introduit de façon systématique dans l'industrie.

Aujourd'hui , L'API est le principal système de commande utilisé dans l'industrie . on dénombre environ 45 fabricants qui ensemble offrent plus de 200 modèles.

2. Capacités des automates programmables industriels :

Lors de sa création , les capacités de L'API se limitaient au remplacement des relais de commande industriels, évidemment , il offrait déjà des avantages aux utilisateurs , il prenait moins de place que les armoires de commandes conventionnelles et consommait moins d'énergie , il était programmable et était muni d'indicateurs d'état , facilitant la vérification de son bon fonctionnement et l'identification des problèmes.

Aujourd'hui , grâce à l'évolution de l'électronique et de l'informatique , sa performance et ses capacités sont impressionnantes . tous en continuant à remplacer les relais de commande ; l'API peut maintenant effectuer des opérations mathématiques , contrôler et régulariser des procédés industriels (température , débit) , commander la vitesse et le positionnement des moteurs , etc.

De plus , les API peuvent communiquer entre eux , ainsi qu'avec un ordinateur hôte , ce dernier peut faire la collecte de données , modifier les paramètres d'opération des API et même modifier leur programme.

On retrouve sur le marché des API pouvant recevoir plus de 30 000 entrées et sorties . ces API remplacent facilement plus 10 000 relais de commande. Bien que ce ne soit pas une pratique courante , il serait donc possible de commander le fonctionnement d'une usine complète à l'aide d'un seul API.

Dans les sections qui suivent, nous expliquerons d'abord le principe de base d'un automate programmable, en utilisant un modèle très simple ,Ensuite, nous traiterons plus en détail de la composition et la construction d'un API.

3. Les éléments d'un système de commande :

Lors de l'études des circuit de commande au début de ce chapitre , nous avons vu que quelques boutons-poussoirs et contacts auxiliaires de faible puissance pouvaient actionner des gros contacteurs pour démarrer des moteurs . Les diagrammes schématiques se ressemblent. Il contiennent tous des boutons – poussoirs , des contacts auxiliaires et des contacteurs. Si l'on fait exception du nombre de dispositifs utilisés , on constate que la différence fondamentale utilisés les trois circuits de commande réside dans la façon dont les différents dispositifs sont raccordés.

Imaginons donc une « boite noire » à l'intérieur de la quelle on peut réaliser diverses liaisons entre , d'une part , les dispositifs de commande (boutons – poussoirs , contacts auxiliaires) et d'autre part , les dispositifs commandés (bobines de contacteurs , lampes témoins). Cette approche donne pour le montage illustré

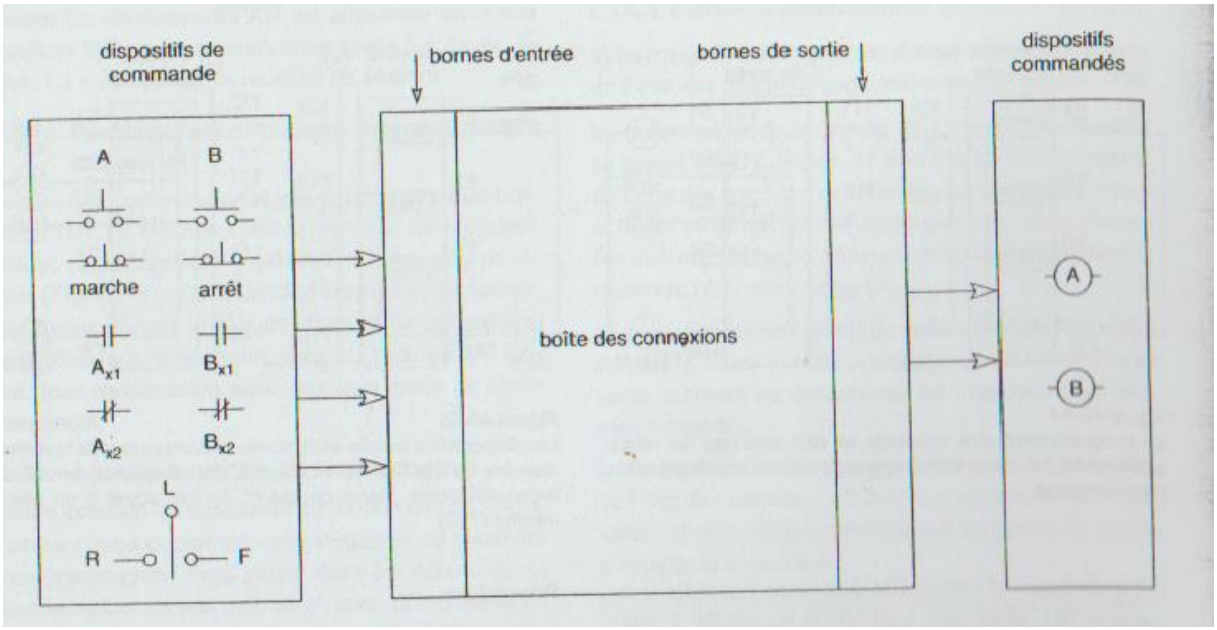


Figure 1.1 : Dans un système de commande , les dispositifs de commande et les dispositifs commandés demeurent essentiellement les mémes .On modifie le système en changeant les connexions entre ces divers dispositifs.

A la fig 1.1 . Les dispositifs de commande sont raccordés aux bornes d'entrée de la boite des connexions. De même, les dispositifs commandés (bobines de maintien A et B) sont branchés aux bornes de sortie. Supposons que la boite de connexions soit un ordinateur , bien que ce dernier consomme très peu d'énergie ; il est capable de simuler les connexions requises , de même que l'action des contacts et des bobines de relais .Cela ouvre des possibilités énormes car il devient alors possible de créer des milliers de contacts et de bobines , dans la mesure ou la mémoire de l'ordinateur est suffisante. Le système de commande peut donc prendre la forme montrée à la fig 1.2 . Le système comprend 5 parties :

1. Une unité centrale de traitement (UCT), soit un ordinateur pouvant simuler les contacts et les bobines de relais requis, ainsi que l'interconnexion.
2. Un module d'entrée qui sert d'interface entre les dispositifs de commande et l'unité centrale de traitement.
3. Un module de sortie qui sert d'interface entre les dispositifs commandés et l'unité centrale de traitement.
4. Une unité de programmation, dotées d'un affichage et d'un clavier pour programmer l'UCT. L'unité de programmation (ou console de programmation) permet de choisir les différents types de relais et de contacts que l'UCT doit simuler, ainsi que la façon de relier ces composants.
5. Un bloc d'alimentation qui fournit la puissance requise par l'UCT , par l'unité de programmation et par les modules d'entrées et de sortie.

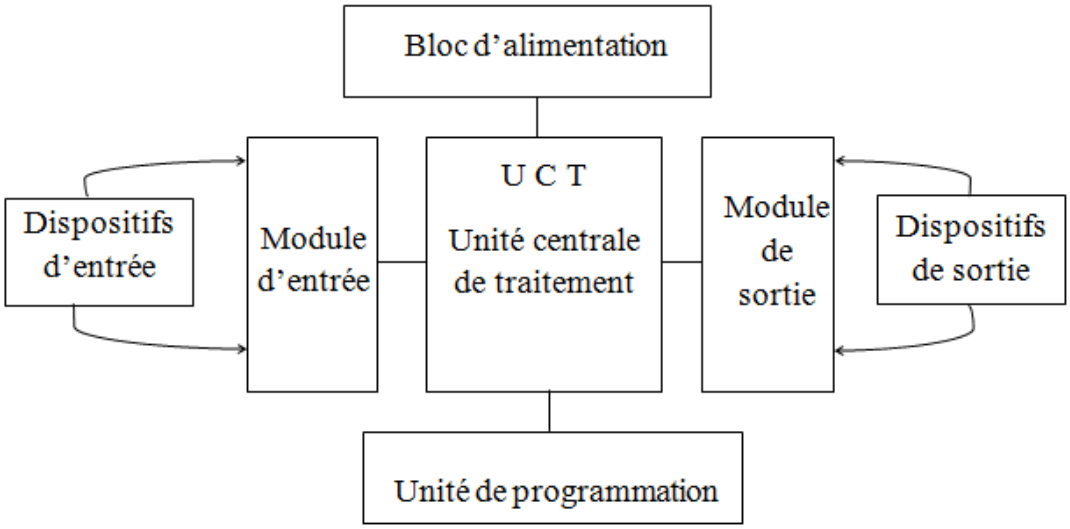


Figure 1.2 : Les cinq paries d'un automate programmable

L'ensemble de ces cinq parties constitue un automate programmable . Lorsqu'il est affecté à un système de commande industrielle , on le nomme automate programmable industriel (abréviation API).

Voyons maintenant la construction et le rôle des quatre premiers éléments de ce système . Pour fins d'explications, nous choisirons un API extrêmement simple ne comprenant que 3 bornes d'entrée et 4 bornes de sortie (la fig 1.3).

Le module d'entrée possède 3 bornes E1, E2, E3 ainsi qu'une borne commune CE. Les dispositifs de commande (appelés dispositifs d'entrée) sont raccordés d'une part individuellement à une borne (E1, E2, E3) et d'autre part à une source d'alimentation à c.c de 24 V, elle-même reliée à la borne CE. Un rectangle, identifié par un numéro de référence, est associé à chaque borne. Par exemple, le numéro 102 associé à la borne E2.

Pour comprendre le fonctionnement de l'API il est utile d'assimiler chaque rectangle à une bobine de relais activée par le dispositif d'entrée qui lui est associé. Par exemple, dans notre modèle, la « bobine » 102 est normalement activée, alors que les « bobines » 101 et 103 ne le sont pas.

La fig. 3montre aussi le module de sortie. Il possède 4 bornes S1, S2, S3, S4 ainsi qu'une borne commune CS. Deux bobines de contacteurs A, B et une lampe témoin sont connectées entre trois de ces bornes er une source à c.a de 120 V. Les 4 bornes sont associées à 4 contacts normalement ouverts, portant chacun un numéro de référence. Ainsi, la borne S1 est associée au contact 111, alors que la borne S4 est associée au contact 114.

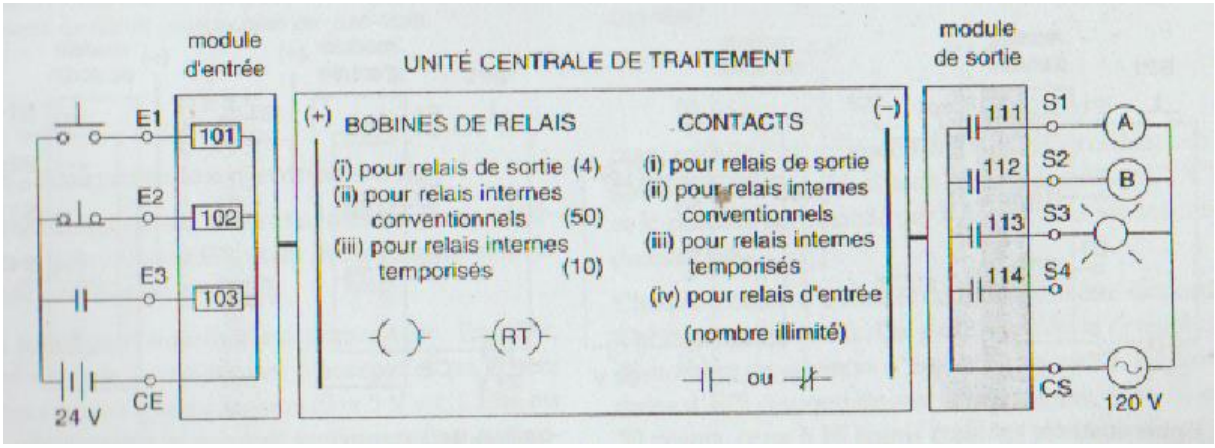


Figure1. 3 : L'unité centrale de traitement contient dans sa mémoire un « stock » de fonction que des bobines de relai, contacts, compteurs, etc.

4. Commande industrielle des moteurs :

Les numéros de référence des modules de sortie et d'entrée sont des « adresses » établies par le fabricant de l'API ; nous verrons bientôt leur utilité .

Afin d'illustrer les capacités et les fonctions de l'unité centrale de traitement de notre exemple, nous pouvons supposer qu'elle contient un important « stock », de contacts et de bobines de relais. Ce stock de « pièces virtuelles » est conservé dans ce qu'on appelle la

mémoire de l'UCT. Le modèle simple présenté à la fig. 40-41 contient dans sa mémoire les composants suivants :

a) Les bobines de relais associées aux 4 contacts du module de sortie. Ces bobines portent les mémés numéros 111,112, 113 et 114 que les contacts qu'elles actionnent. Comme il n'y a que 4 sorties, le nombre de ces bobines est limité à 4.
b) Les bobines de relais internes. Ces bobines de relais fonctionnent exclusivement à l'intérieur d'entrée ni dans le module de sortie.
c) Nous supposons que le stock comprend 50 bobines de relais internes conventionnels, portant les numéros de référence 701 à 750. En plus, la mémoire contient 10 bobines de relais temporisés, portant les numéros 901 à 910. Les délais associés sont déterminés lors de la programmation.
d) Un nombre presque illimité de contacts associés à n'importe quelle bobine de relais mentionnée ci-dessus. Ces contacts portent le même numéro de référence que la bobine de relais qui les actionne.
e) Selon les exigences du système de commande, pour chaque bobine, on peut sortir de la mémoire autant de contacts que l'on désire.

Afin « d'alimenter » les bobines de relais, l'unité centrale de traitement simule aussi une source d'alimentation, représentée par les deux traits verticaux (+) et (-).

Nous présentons maintenant cinq exemples très simples illustrant le principe de fonctionnement de ce modèle d'automate programmable.

5. Exemples d'utilisation d'un automate programmable :

Exemple 1 (fig 1.4) un bouton- poussoir BPI doit allumer et éteindre une lampe témoin L2.

Procédure :

1) Puisque BPI est branché sur la borne E1, la « bobine » 101 est alimentée lorsqu'on appuie sur le bouton - poussoir.
2) Comme la lampe L2 est branchée sur la borne S2, il faut actionner le contact 112 pour la faire allumer.
3) En vertu de (1), l'opérateur de console de programmation doit choisir dans le stock de composants en mémoire un contact normalement ouvert (NO) portant le numéro 101. De la même façon, en vertu de (2), il doit choisir la bobine de relais de sortie numéro 112.

Cette programmation se fait à l'aide des touches appropriées sur la console de programmation. Finalement, toujours à l'aide de la console, l'opérateur doit programmer les connexions entre le contact 101, la bobine 112 et les barres (+) (-) montrées à la fig 1.4.

Lorsqu'on appuie sur le bouton BPI , la « bobine » 101 est alimentée par la source externe de 24 V. Le « contact » 101 se ferme, ce qui alimente la « bobine » 112.

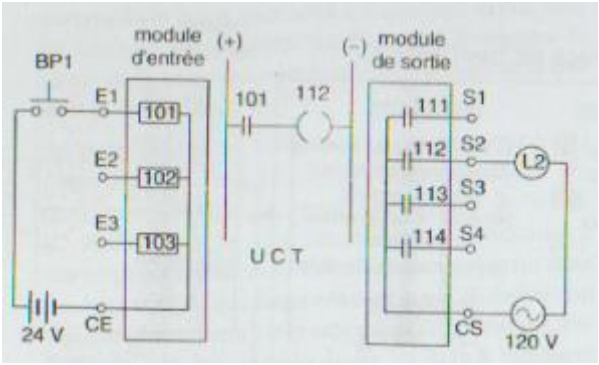


Figure 1.4 : Le contact 101, la bobine de relais 112 et le raccordement de ces deux composants sont programmés. Le rectangle 101 simule une bobine de relais

Remarquer que ces deux derniers composants n'existent pas vraiment. Ce sont plutôt des élément virtuels simulés par l'ordinateur. Lorsque la bobine 112 est « excités », un relais réel est actionné pour fermer le contact réel 112. Par conséquent, la lampe réelle L2 est alimentée par la source réelle de 120 V.

Exemple 2 (fig1. 4) : Le bouton – poussoir BPI doit alimenter la lampe L2, mais cette fois la lampe doit s'éteindre lorsqu'on appuie sur le bouton.

Procédure :

1. Le montage est identique à celui de l'exemple 1 sauf que l'opérateur de la console de programmation doit choisir un « contact » 101 qui est normalement alimentée, et le contact réel 112 est normalement fermé.
   Ce changement peut se faire en moins d'une minute en manipulant quelques touches du clavier. Il suffit d'effacer le contact NO et programmer un contact NF.

Exemple 3 (fig1.5) : Le bouton – poussoir doit alimenter trois lampes L1, L2, L3 de sorte que L1 et L2 s'allument et que L3 s'éteigne lorsqu'on appuie sur le bouton.

Procédure :

1. Comme on doit alimenter 3 lampes branchées aux sorties S1, S2, S3 l'opérateur de la console de programmation doit sélectionner les trois bobines de relais de sortie correspondantes, soient les bobines 111, 112, 113.

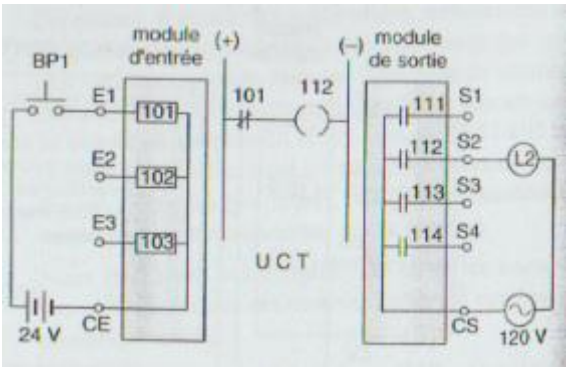


Figure 1. 5 : Ce montage est semblable à celui de la fig1.4 sauf qu'on a programmé un contact NF au lieu d'un contact NO.

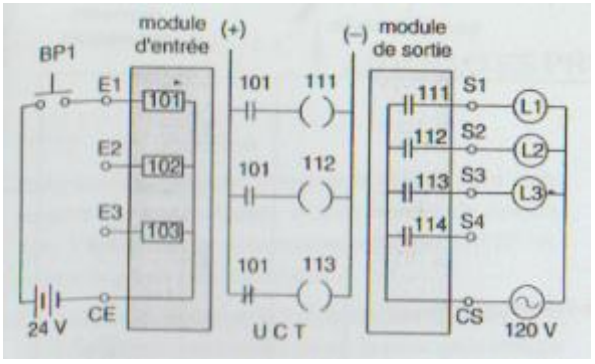


Figure 1. 6 : En programmant des contacts et des bobines de relais additionnels, on peut construire un système de commande plus complexe.

2. La « bobine » d'entrée 101 doit donc posséder trois contacts 101, dont deux sont NO et un est NF. L'opérateur doit programmer les connexions indiquées sur le « circuit » entre les barres (+) (-) de la fig1.6

On constate que l'on peut ajouter des contacts, augmenter le nombre de relais et modifier les connexions, en appuyant tout simplement sur quelques touches de la console de programmation. On n'a jamais besoin de dénuder un fil ou de fixer un relais. Noter que les contact 111, 112, 113 sont des composants réel, de même que la source de 120 V et les trois lampes.

Exemple 4 (fig1.7) : Le fonctionnement de BPI et des lampes doit être identique à celui de la fig1.6, mais l'API doit faire appel à un relais interne (715) pour alimenter les « bobines » des relais 11, 112 , et 113.

Procédure :

1) Parmi les 50 relais internes disponibles, l'opérateur de la console choisit celui portant le numéro 715. De plus, il sélectionne trois contacts associés , dont deux sont NO et un est NF.
2) Ensuit, il programme les connexions indiquées sur le circuit de la fig1.7.

On constate de nouveau que ce changement de relais et de contacts fait seulement intervenir l'ordinateur. Les pièces tangibles demeurent inchangées.

Exemple 5 (fig1.8) : On désire le même mode de fonctionnement que dans l'exemple 4, sauf que la lampe L1 s'allumer 5 secondes après la fermeture de bouton – poussoir.

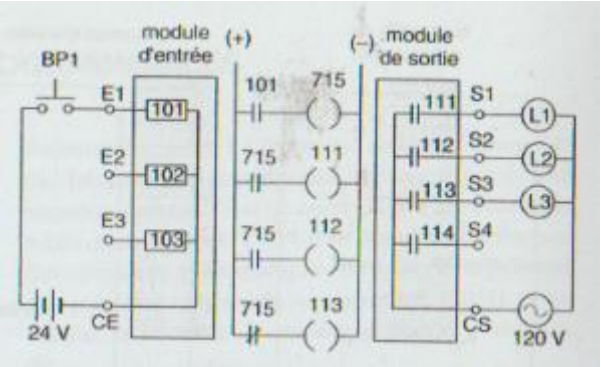


Figure 1.7 : Les Dispositifs d'entrée et de sortie se comportent de la même manière qu'à la fig1.6, mais le circuit est programmé de façon différente. Dans ce cas-ci, on fait appel à un relais interne (715).

Procédure :

1) Dans ce cas, l'opérateur doit ajouter un relais temporisé pour allumer L1. Le reste du montage demeure le même.
2) L'opérateur sélectionne dans la mémoire de l'UCT la bobine de relais portant le numéro de référence RT907. Il y ajoute le contact normalement ouvert portant le numéro 907, et programme les connexions indiquées sur le circuit de la fig1.8. Enfin, il programme le délai requis de 5 secondes.
3) Lorsqu'on appuie sur le bouton BPI, le « contact » 101 se ferme, ce qui alimente la « bobine » du relais 715.
   Aussitôt. Les deux « contact » 715 (NO) se ferment et le contact 715 (NF) s'ouvre. Par conséquent, la lampe L2s'allume immédiatement alors que la lampe L3.

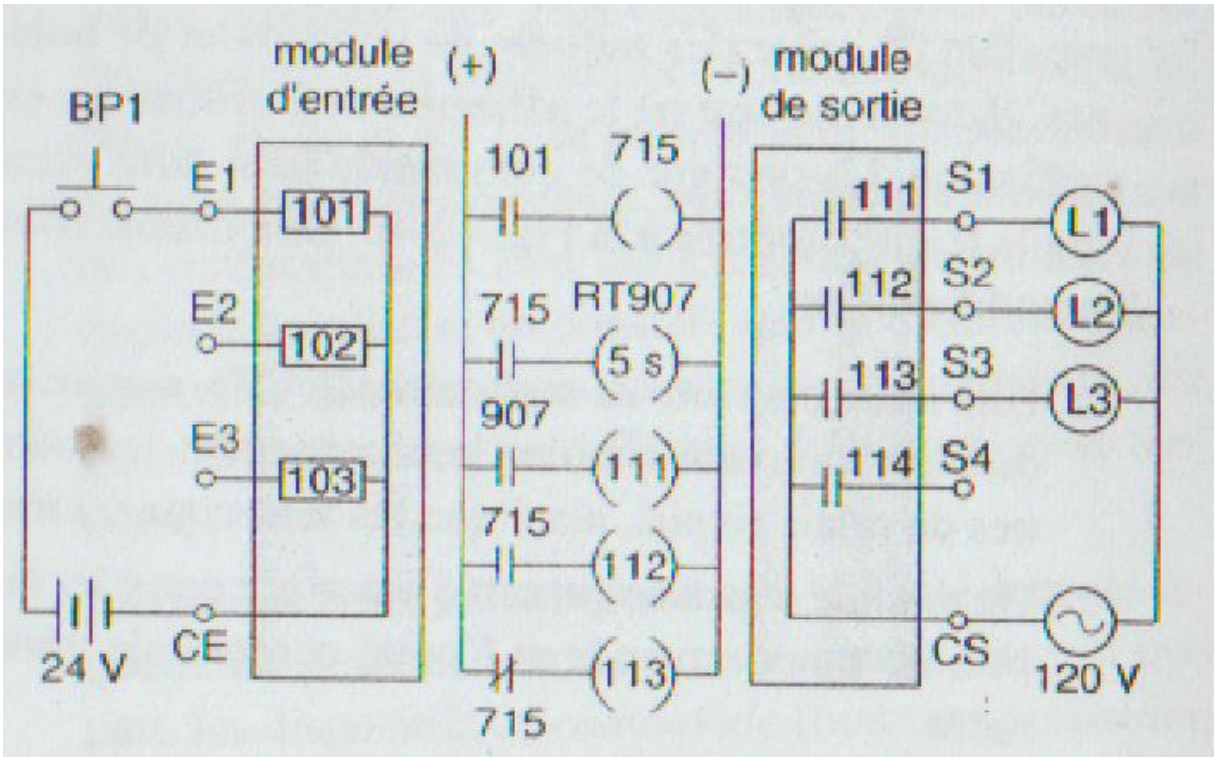


Figure 1.8 : Montage utilisant un relais temporisé interne. Le délai de ce relais est programmé, tout comme le reste du circuit compris entre les barres (+) (-). S'éteint la « bobine » RT907 est alimentée mais son « contact » 907 se ferme seulement après 5 s. Donc, la lampe L1 s'allume après un délai de 5 s.

6. Parties d'un automate programmable industriel :

Tous les API comportent (a) une unité centrale de traitement (UCT), (b) une console ou unité de programmation, (c) un module d'entrée et (d) un module de sortie (fig1.2). Dans les cinq exemples précédents, nous avons illustré le rôle de chacune de ces parties. Nous décrivons maintenant, dans les sections qui suivent, leur construction ainsi que leur mode de fonctionnement.

7. L'unité centrale de traitement :

L'unité centrale de traitement est cerveau de l'API. C'est un circuit complexe, comprenant un ou plusieurs microprocesseurs. Sans entrer dans les détails de sa construction et de son opération, nous décrirons brièvement sa mémoire et l'utilisation qu'il en fait.

Il existe deux types de mémoire. Le premier est la mémoire non volatile, ce qui signifie que son contenu ne peut être ni effacé, ni modifié. Le second est la mémoire volatile, ce qui signifie que son contenu est facilement et rapidement modifiable.

La mémoire non volatile contient toutes les instructions nécessaires à la gestion de l'API. Ces instruction sont utilisées pour interroger les modules d'entrée et connaitre l'état du procédé,

transmettre les ordres aux modules de sortie, interpréter et exécuter les instructions qui parviennent de la console de programmation, exécuter le programme de l'utilisateur, etc.

C'est dans cette mémoire que le fabricant installe la gamme de fonctions exécutables par l'API. En plus des fonctions de type relais, comme la fonction bobine, la fonction contact, etc.., l'API offre une gamme d'au moins 30 autres fonctions, Citons, par exemple, les fonctions de comptage, de commutateur à cames et de registres. En somme, la mémoire non volatile établit tous paramètres d'opération de l'API. Son contenu est défini par le fabricant et ne peut être ni effacé ni modifié par l'utilisateur.

La mémoire volatile de l'API est divisée en plusieurs sections. Trois d'entre elle servent à mémoriser les informations suivantes : (1) l'état des entrées, (2) l'état des sorties et (3) le programme de l'utilisateur.

L'UCT exécute séquentiellement les tache suivantes :

a) Interrogation des modules d'entrée et mémorisation de l'état des dispositifs qui leur sont raccordés.
b) Exécution du programme de l'utilisateur. Pendant ce travail, l'UCT décide, en fonction du programme et de l'état des entrées mémorisées, quelles sorties seront activées ou désactivées. Ses décisions sont inscrites au fur et à mesure de l'exécution du programme dans la mémoire réservée à cette fin.
c) Transmission des résultats mémorisés aux modules de sorties. C'est lors de cette étape que les modules de sortie activent ou désactivent les dispositifs qui leur sont raccordés.

Le cycle d'opération qui consiste à prendre la lecture de l'état des entrées, exécuter le programme de l'utilisateur et affecter les résultats aux modules de sortie, s'appelle la scrutation[2].

Quand l'API est en marche, il exécute continuellement des scrutation. Le temps requis pour exécuter une scrutation complète varie en fonction de la rapidité de l'API et la longueur du programme de l'utilisateur. En générale, il est de l'ordre de quelques millisecondes.

8. Console de programmation :

La console de programmation sert, comme son nom l'indique, à programmer l'API . Mais son rôle ne s'arrête pas la, Elle permet aussi de visualiser ou de modifier l'état des entrées et des sorties de l'API , ainsi que la valeur de certains paramètres. Elle sert aussi d'outil de vérification et de diagnostic pour l'API Finalement, on l'utilise pour sauvegarder les programmes sur des supports magnétiques ou optique (disquette, CD), et pour récupérer ces programmes à partir de ces mêmes supports.

Bien que la console de programmation joue plusieurs rôles, sa présence n'est pas requise lors de l'opération automatique de l'API. Il est donc possible de la débrancher et de la remiser.

La console de programmation est constituée d'un petit boitier muni d'un clavier et d'un affichage simple. Elle peut aussi prendre la forme d'un ordinateur avec écran cathodique et clavier, auquel on a ajouté des touches spéciales . Comme elle est utilisée dans l'industrie, elle est portative et robuste (fig1.9, 1.10, 1.14).

Figure 1.9 : Cet automate programmable industriel a été adapté à des fins didactiques. L'opérateur tient en main la console de programmation qui interagit avec les deux parties de l'API montées sur le tableau vertical, La partie supérieure contient l'unité centrale de traitement, le bloc d'alimentation et des modules d'entrée et de sortie. La partie inférieure est simplement une extension de la première, offrant des modules E/S additionnels. Cet API dispose de 10 points d'entrée et de 6 points de sortie (gracieuseté de Lab-Volf).

9. Les modules d'entrée/sorties :

Les modules d'entrée et les modules de sortie (désignés par l'abréviation E/S) sont des interfaces entre le procédé commandé et l'UCT.

Cette fonction d'interface est primordiale. En effet, l'unité centrale de traitement n'accepte et ne génère que des signaux à basse tension (0 et 5 c.c) . Elle est aussi très sensible et pourrait facilement être endommagée si elle était exposée à des signaux excédant cette gamme de tension, Ainsi , tous les échanges entre l'UCT et le procédé commandé par l'API se font via les modules d'E/S.

Chaque module d'entrée et de sortie peut être raccordé à plusieurs dispositifs. On parle alors de la densité ou du nombre de « points » d'entrée ou de sortie. Les modules d'E/S disponibles sur le marché ont 4, 8, 16 ou 32 points, ceux à 16 points étant les utilisés.

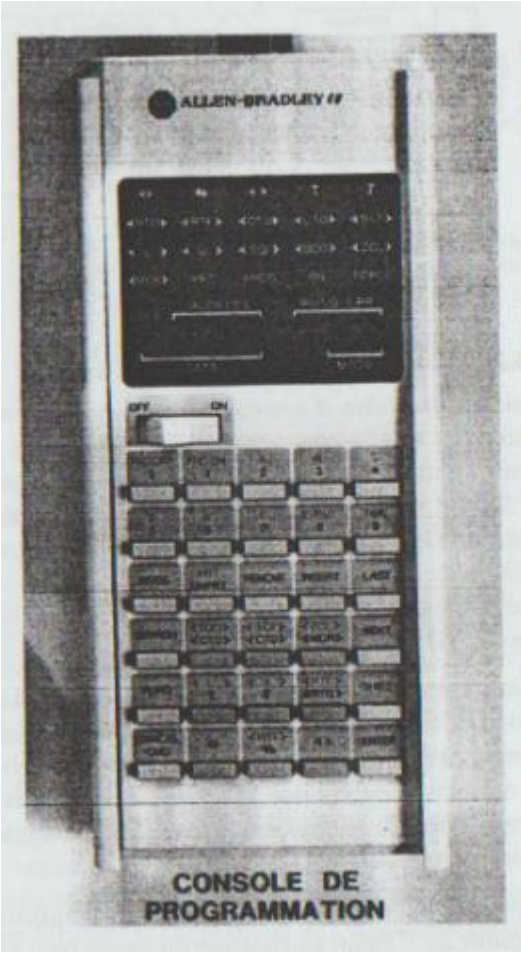


Figure 1.10 : Console de programmation portative montrant les touches de programmation. Le petit écran situé au-dessus du clavier permet de visualiser les contacts, les bobines de relais, les délais, etc., au fur et à mesure de leur programmation en affichant les numéros de référence associés à ces composants. La console sert aussi à vérifier l'état des dispositifs d'entrée et de sortie. Par conséquent, elle constitue un outil de programmation et de diagnostic (gracieuseté de lab-Volf).

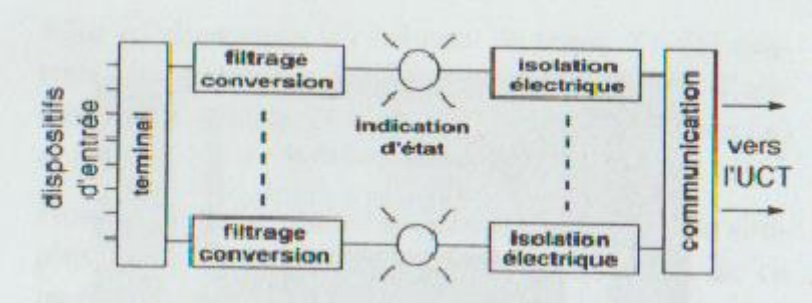


Figure 1.11 : Composants du module d'entrée.

10.Structure des modules d'entrée :

Tous les modules d'entrée ont une architecture qui ressemble à cette montrée à la fig1.11. O n y retrouve, outre le terminal de branchement, une section de filtrage et de conversion, un indicateur d'état, une isolation électrique et une section de communication, cette structure (filtrage- état- isolation) est reproduite autant de fois que le module a de points d'entrée.

Afin d'éviter les fausses activation de l'entrée, la section de filtrage et de conversion élimine les bruits, comme ceux dus à la tension induite ou aux rebonds de contacts. Elle abaisse la tension qui apparait aux bornes d'entrée et. Au besoin, redresse les signaux à courant alternatif.

L'indicateur d'état consiste en un témoin lumineux qui s'allume ou s'éteint en fonction du signal reçu à chacune des bornes d'entrée. Il facilite grandement la vérification du bon fonctionnement des dispositifs d'entrée et de leur raccordement.

L'isolation électrique protégé l'UCT contre les bruits électriques, fréquents dans l'industrie. Dans la majorité des cas, cette protection est assurée par des coupleurs optiques ou les signaux électriques sont remplacés par des signaux lumineux. Ces signaux lumineux ne sont pas affectés par les perturbations de champs électrique et magnétique. Les dispositifs de couplage résistent à des pointes de tension de 1500 volts. Tout en transmettant les signaux. Ils isolent complètement les circuits sensibles de l'API de ceux qui sont directement en contact avec les bornes d'entrée.

La section de communication quant à elle regroupe tous les états de chacun des circuits d'entrée du module et les transmet à l'UCT.

Une caractéristique importante des modules d'entrée est leur impédance. Selon la tension pour laquelle le module est conçu, son impédance est généralement comprise entre 5 kΩ et 12 kΩ. Le courant nécessaire pour activer une entrée est d'environ 10 mA. Ce faible positifs d'entrée une économie sur la robustesse des dispositifs d'entrée et sur le cout du câblage.

Noter que l'utilisateur doit fournir l'alimentation des dispositifs d'entrée. On utilise différentes tensions (soit 24 V à 120 V c.a. ou 10 V à 100 V c.c). Les modules d'entrée abaissent la tension de ces signaux à un niveau acceptable par l'UCT.

11. Structure des modules de sortie :

Tous les modules de sortie, quel que soit leur type, sont construits selon la même architecture (fig1.12). Les sous- ensembles constituant ces modules sont : la section de communication, l'isolation électrique, l'indicateur d'état etle circuit de puissance. Mise à part la section de communication, la structure est reproduite autant de fois que le module a de points de sortie.

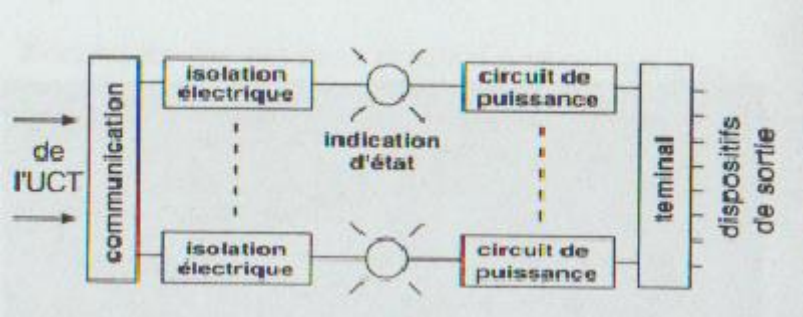


Figure 1.12 : Composants du module de sortie.

La section d'isolation électrique et d'indication d'état jouent des rôles semblables à celles des modules d'entrée, soit : protéger de tension induites, et afficher à l'aide d'un témoin lumineux l'état de la sortie afin d'aider à la vérification du bon fonctionnement de cette dernière.

Finalement, le circuit de puissance amplifie le signal venant de l'UCT afin de pouvoir commander adéquatement le dispositif de sortie qui lui est raccordé.

Comme on l'a vu, chaque point de sortie agit comme un interrupteur. Il applique ou enlève la tension au dispositif qui lui est raccordé.

Remarquer que l'utilisateur doit fournir l'alimentation pour actionner les dispositifs commandés.

Bien que les modules de sortie soient destinés à commander des dispositifs industriels, leur capacité est limitée. La majorité d'entre eux supportent un courant maximum allant de 0.5 A à 2 A par point de sortie. Si un dispositif nécessite un courant supérieur à la capacité des modules, on utilisera un relais intermédiaire (voir le relais B à la fig1.15). Il est aussi recommandé d'installer des fusibles, afin de protéger les équipements contre la surintensité d'un courant accidentelle.

Les pointes de tensions accélèrent l'usure des modules de sortie, et peuvent provoquer leur bris. Afin d'éviter ces ennuis, il est recommandé d'installer des filtres atténuateurs ou des érecteurs lorsque le dispositif de sortie génère des surtensions transitoires.

12.Modularité des automates programmables industriels :

Une caractéristique importante de l4API est sa modularité. Ainsi, l'UCT, les modules d'entrée et ceux de sortie sont montés dans des boitiers individuels (fig1.13).

La modularité conféré à l'API un avantage indéniable sur les système de commandes à relais. Si, lors d'une panne, on soupçonne qu'un module est défectueux, il suffit de le remplacer par un module identique et de réalimenter l'API. Si tout rentre dans l'ordre, le problème est résolu.

Le seul module qui implique une opération supplémentaire lors de son remplacement est celui qui contient la mémoire de l'API. Il faut alors récupérer le programme, préalablement sauvegardé, à l'aide de la console de programmation. Cette opération équivaut à changer toute une armoire de commande à relais. Pourtant, elle se fait en quelques minutes.

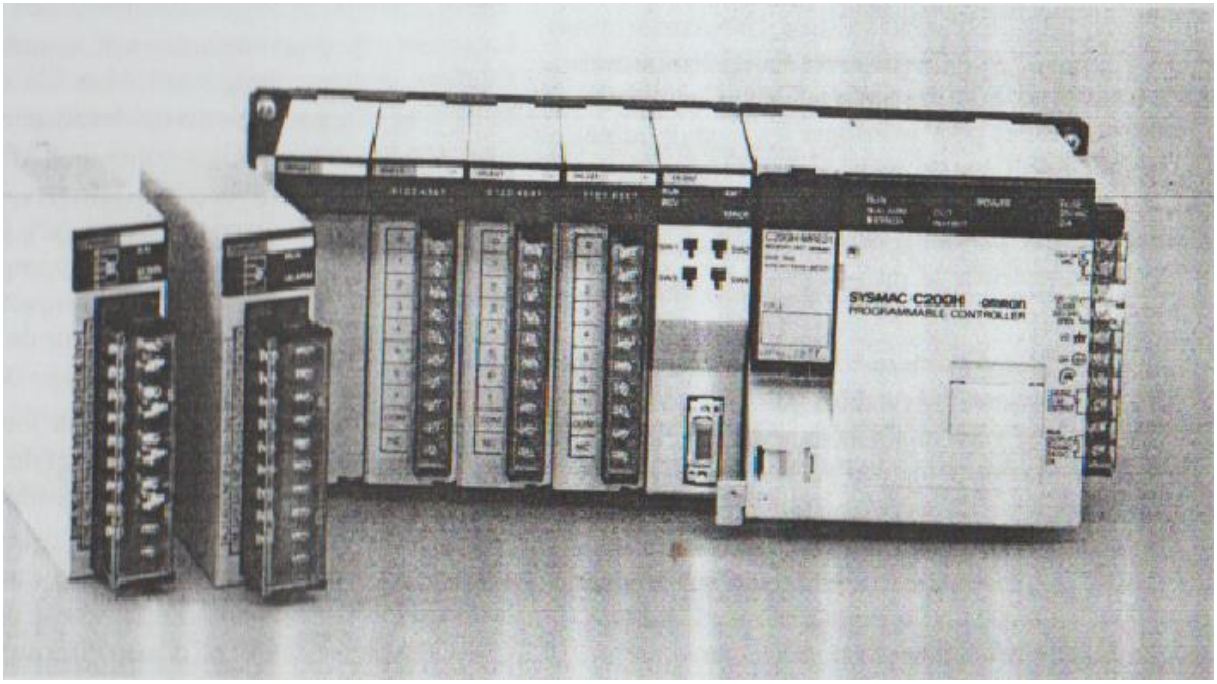


Figure 1.13 : Automate programmable modulaire comportant 5 modules d'E/S. En plus de simuler des circuit de relais, il permet la commande des systèmes asservis. L'UCT est contenue dans la partie de droite portant la marque de commerce. Le module muni de 4 commutateurs sert à communiquer avec ordinateur central qui géré le procédé industriel. Cet API est programmable en plusieurs langages dont le diagramme en échelle et le grafcet (gracieuseté d'Omron Canada Inc.).

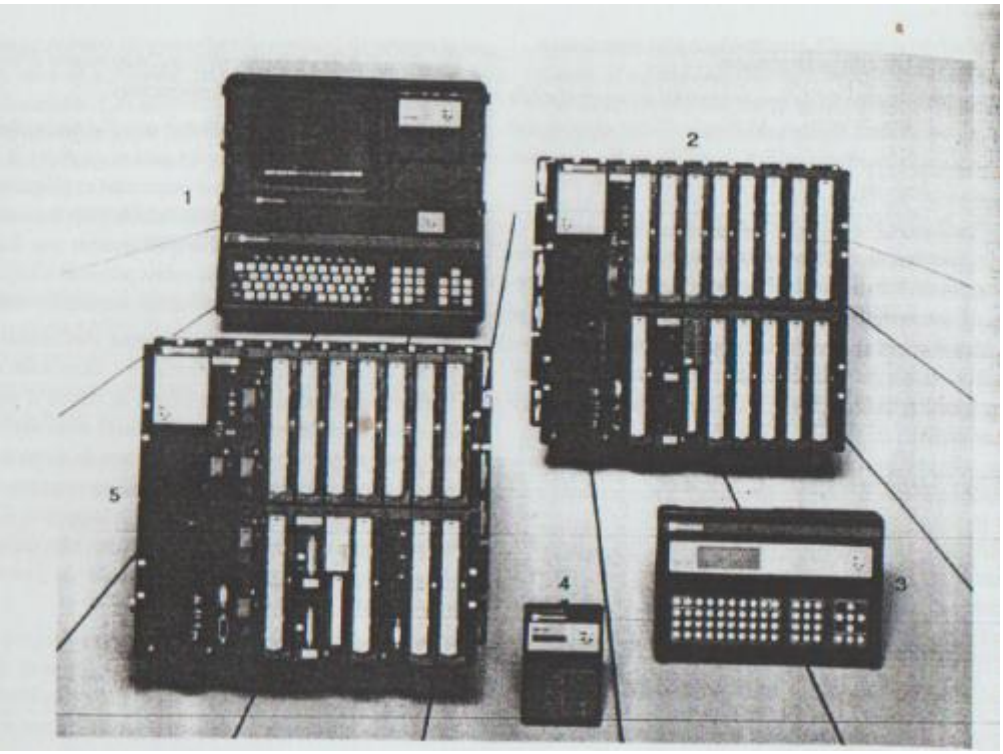


Figure 1.14 : Photo montrant divers modèles d'automates et de consoles de programmation, En progressant dans le sens horaire à partir de 11 heures on remarque :

1)Terminal universel pouvant fonctionner en langage à diagramme en échelle, en littéral en grafcet, totalement mixables .
2)Automate modulaire à 2048 E/S tout ou rien. Les E/S son reliées par fibre optique ou connexion électrique.
3)Terminal d'exploitation pour la lecture, la Modulation et le diagnostic des variables autorisées des automates programmables.
4)Terminal de réglage : Terminal de poche pour la lecture de la modification des variables autorisée.
5)Automate modulaire à 2048 E/S tout ou rien, plus des coupleurs permettant le comptage, le positionnement et le traitement de signaux analogique (gracieuseté de Télémécanique Canada Ltée).

Un autre avantage de la modularité de l'API est sa capacité de s'adapter aux besoins de l'utilisateur. Ainsi, il est possible d'ajouter des modules d'E/S au fur et à mesure que les besoins le justifient. Seule la capacité de l'API limite le nombre maximum de modules d'E/S utilisables.

13.Les entrées et sorties à distance :

Nous venons de voir que l'utilisateur peut configurer l'API selon ses besoins. La modularité des API permet aussi de placer les modules d'E/S dans des enceintes séparées de celle contenant l'UCT. On parle alors d'E/S à distance.

Les modules d'E/S sont alors regroupés en ilots. Ces ilots peuvent être placé assez loin de l'UCT (jusqu'à 3000 mètre). Chaque ilot est alors équipé d'un bloc d'alimentation et d'un module de communication. Un câble, torsadé, coaxial, ou en fibres optiques relie les ilots à l'UCT.

14.Circuit conventionnels et circuits d'automate programmable :

Il est maintenant clair que l'on peut utiliser un automate programmable à la place d'un circuit de commande conventionnel. On doit alors se rappeler que chaque entrée de l4API se comporte comme une bobine de relais dont les contacts sont simulés par le programme.

Exemple 6 la fig1.15 : représente l'application d'un automate programmable à la commande marche/arrêt

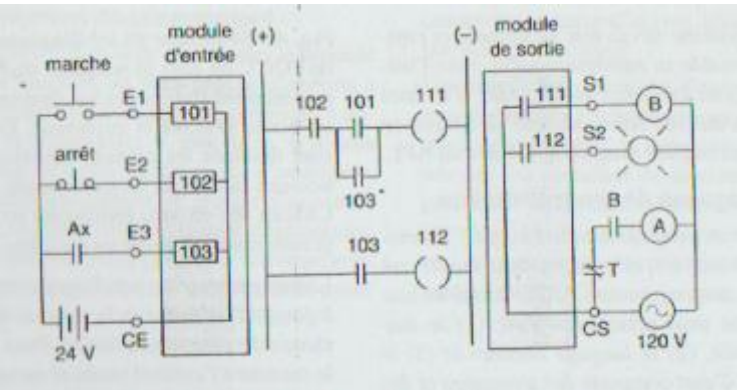


Figure 1.15 : Emploi d'un automate programmable pour réalise la commande du démarreur magnétique illustré

D'un moteur. La version conventionnelle de ce démarreur magnétique est montrée

Remarque que le contact d'arrêt du ajouter un relais auxiliaire B. Les contacts de ce relais sont suffisamment robustes pour porter le courant d'excitation de la bobine du contacteur A.

Exemple 7. Nous réalisons le circuit de commande à l'aide d'un automate programmable le montage est représenté à la Fig1.16.

Seul les contacts NO du bouton – poussoir **marche** et les contacts NF du bouton – poussoir **arrêt** sont branchés sur entrées de l'API le verrouillage mécanique de sécurité des boutons **marche** et **arrêt**. Est maintenant assuré par les contacts NO 102 de l'échelon 1 E et NF 101 de l'échelon 3.

Seuls les contacts NO (A_{x1} et B_{x1}) des contacteurs A et B sont branchés respectivement sur les entrées 103 et 104 de l'API. Les respectivement NC (A_{x2} et B_{x2}) ne sont plus nécessaires. Les contacts NO 103 et NO 104 des échelons 2 et 4 au verrouillage. Les contacts NF 104 et NF 103 des échelons 1 et 3 constituent un système de verrouillage de sécurité.

comme mesure de sécurité additionnelle, les contacts NF 112 et NF 111 des échelons 1 et 3 ont été programmés. Ces permet d'éviter le danger potentiel que représente un bris le raccordement des contacts $A_{x1.}$

Comme on peut le constater, des économies, des économies sont réalisées dans les dispositifs d'entrée et de sortie : les boutons – poussoirs marche et arrêt n'ont qu'ont qu'un seul contacts ; de même, les contacteurs A et B n'ont qu'on contact auxiliaire.

15.Règle de sécurité :

L'utilisation d'un API permet d'inverser l'action des contacts branchés au module d'entrée : un contact réel NO branché sur une d'API peut être programmé en contact NF dans le programme de l'utilisateur (voir exemple 1 et 2). Cette liberté offerte à l'utilisateur doit s'exercer avec prudence, surtout en ce qui a trait au choix du type de contact (NO ou NF) des dispositifs d'entrée. On doit toujours respecter la règle de sécurité suivante.

> Tout contact associé à un dispositif servant à initier une action doit être de type NO : inversement, tout contact associé à un dispositif servant à arrêter une action doit être de type NF.

Si cette règle de sécurité n'est pas respectée, un bris dans les câbles reliant les dispositifs d'entrée à l'API pourrait entrainer le démarrage d'action indues, ou l'impossibilité d'arrêter les actions en cours.

16.La programmation :

Programmer un API, c'est écrire dans sa mémoire la description du travail qu'il aura à accomplir.

Dès sa création, une attention particulière a été portée à la méthode de programmation. Le devis technique stipulait que le système devait être facilement et rapidement programmable et reprogrammable chez l'utilisateur. L'API a donc été conçu avec le souci d'en faire un outil simple à utiliser. Ainsi, aucune formation en informatique n'est requise pour programmer un API.

17.Les langages de programmation :

Le terme langage de programmation désigne l'ensemble des symboles utilisés, et la façon dont ils doivent être agencés afin de programmer l'API. Permis les langages utilisés, les trois principaux sont : (1) le diagramme en échelle, (2) le langage booléen et (3) le Grafcet. Chacun d'eux comporte des avantages et des inconvénients, que nous tenterons d'illustrer dans la section qui suit.

18.Le diagramme en échelle :

Parmi tous les langages de programmation, le diagramme en échelle est le plus facile. C'est d'ailleurs le langage que nous avons utilisé tout au long de cet exposé.

Son nom vient d'une méthode de représentation graphique utilisée pour décrire les circuits à relais des armoires de commande. Ces dernier sont souvent présentés sous la forme d'un diagramme en échelle. La fig1.16 montre un tel diagramme. De chaque côté de l'UCT, on note la présence de deux traits verticaux représentant l'alimentation (borne « vivante » et neutre ou borne positive et commun). Entre ces deux traits, sont dessinés les circuits électriques qui activent les bobines des relais de commande ou des démarreurs. Chacun des circuits représente un échelon, d'où l'expression « diagramme en échelle ».La programmation par diagramme en échelle consiste à dessiner à l'écran de la console de programmation le circuit de commande désiré. Pour ce faire, on déplace le curseur à l'endroit voulu et on appuie sur une touche de sélection de fonction pour faire apparaitre, à l'endroit indiqué, un contact NO ou NF, une bobine de relais interne ou de sortie, etc. Une fois la fonction choisie, il suffit de taper son numéro de référence. Puis on répété les mêmes opérations pour le reste du circuit de commande. La majorité des API utilisant le diagramme en échelle comme langage de langage de programmation peuvent être reliés à une imprimante. On peut ainsi imprimer le diagramme en échelle, ce qui s'avère un outil pratique de vérification du programme.

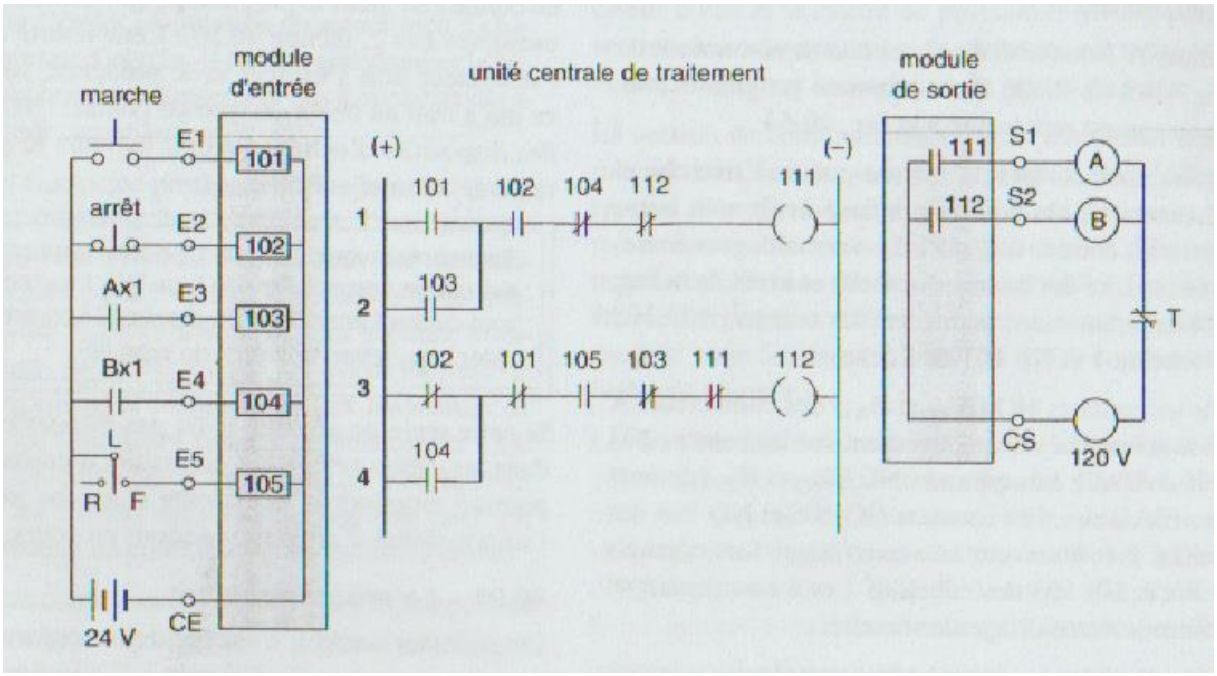


Figure 1.16 : Emploi d'un automate programmable pour réaliser le système de commande de démarrage et de freinage par inversion illustré.

19.Le langage booléen :

Le nom de ce langage de programmation vient du fait qu'il s'inspire de l'algèbre de Boole. Cette algèbre est un outil mathématique utilisé pour résoudre des problème de logique. Il fut inventé au milieu du 19e siècle, par le mathématicien britannique George Boole.

Un inconvénient du langage booléen et des consoles de programmation qui l'utilisent est la difficulté de lecture. Il est facile d'écrire un programme en langage booléen à partir d'un diagramme en échelle. Par contre, il est assez pénible de relire un programme en langage booléen et de reconstituer le diagramme en échelle équivalent.

20.Le Grafcet :

Le mot GRAFCET est l'acronyme de l'expression : GRAphe de commande Etape –Transition [3]. Il représente le fruit des recherches visant à créer une méthode d'étude rigoureuse des automatismes ; et facilement applicable dans l'industrie. C'est un outil très efficace de diagnostic des programmes de l'API et tout l'automatisme.

21.Avantages et inconvénient des automates programmables :

Les raison qui expliquent la popularité croissante des API sont nombreuses. Nous indiquons ici les principales.

L'API est flexible. Comme il est programmable, la modification de sa tâche est facile. Par contre, avec les systèmes de commande à relais réels, toute modification implique l'ajout ou le retrait de relais et la modification des raccordements. Cette opération comporte un risque élevé d'erreurs de branchement.

La flexibilité de l'API est telle que lorsqu'un procédé n'est plus requis, on peut le démonter et réinstaller pour commander un autre procédé complètement différent. Ceci serait impossible avec une armoire de commande à relais.

L'API est beaucoup moins encombrant que l'armoire de commande à relais qu'il remplace. Par exemple, une unité centrale de traitement d'environ 0.1 mètre cube remplace des centaines de relais de commande et tout le câblage qui relie leurs contacts.

De plus, l'API consomme beaucoup moins d'énergie, et son fonctionnement est silencieux.

L'API est beaucoup plus fiable que l'armoire de commande à relais. L'absence de pièces mobiles à l'intérieur de L'API est un facteur important expliquant cette fiabilité. Les relais de l'armoire de commande comportent plusieurs pièces en mouvement qui finissent par s'user. Les contacts des relais peuvent s'oxyder ou se souder, provoquant ainsi des commandes erronées.

De plus, la fermeture et l'ouverture des contacts des relai ; bien que rapides, nécessitent un certain temps, Il n'est pas sur que ce temps reste le même d'un relais à l'autre, surtout lorsque ces derniers sont usés. Dans certaines applications ou la séquence de fermeture des contacts est importante pour la bonne marche du procédé, ceci peut causer des erreurs de séquence. Comme diagnostiquer. Etant donné son mode de fonctionnement. l'API élimine ce problème.

Toute armoire de commande à relais est assemblée à la main. Des centaines, et même des milliers de fils doivent être branchés entre les contacts et les bobines de relais, ce qui implique un très grand risque d'erreur. Ces erreurs sont très difficiles à repérer. Lorsqu'on utilise un API, il suffit essentiellement de « dessiner » le diagramme en échelle, tel que conçu. Là encore, s'il se glisse une erreur, la console de programmation dispose de fonctions utilitaires permettant de la retracer et de la corriger rapidement.

Finalement, le cout d'achat et d'installation d'un API est inférieur à celui d'une armoire de commande à relais, dés que l'API remplace une trentaine de relais de commande. Cette économie croit évidemment avec l'ampleur du systèmes.

Parmi les inconvénients de l'utilisation des API, citons que leur mode de fonctionnement entraine parfois des problèmes du type aléas de séquence. Ainsi, il se peut que l'ordre dans lequel on écrit le programme influence le comportement de la commande.

Finalement, mentionnons que, d'une marque d'API à l'autre, une même fonction n'a pas nécessairement le même effet, ou ne produit pas exactement les mêmes résultats. Cet inconvénient vient du fait que les fabricants n'ont pas encore établi de standards communs. Ainsi, chaque fois que l'on change de marque d'API.

Il est important de consulter le manuel de programmation, afin de s'assurer du comportement des différentes ne sont pas énormes, mais elle impliquent parfois de légères modifications dans le circuit de commande programmé.

22. MODERNISATION D'UNE INDUSTRIE GRACE AUX API :

Depuis l'introduction des automates programmables industriels (API), les industries manufacturières et de service ont subi d'énormes transformations, Comment cette transition de l'ancienne technologie vers la nouvelle a-t-elle pu s'effectuer ?

Pour répondre à cette question, nous prendrons l'exemple d'une grande entreprise portuaire d'arrimage qui a changé le mode de transbordement de l'un de ses produit en installant de l'équipement et un systèmes de commande très sophistiqués, Durant la période de transition, il a fallu intégrer la nouvelle technologie avec l'ancienne, car il était impensable de modifier d'un seul coup toutes les anciennes procédures.

Le processus d'arrimage consiste à décharger d'un bateau un produit en vrac comme du sel, du charbon ou un minerai comme de l'alumine. L'alumine (AI_2O_3) est une poudre blanche et légère qui sert à fabriquer de l'aluminium au moyen d'un procédé électrolytique, Lorsqu'un

bateau chargé d'alumine arrive au port, Le produit doit etre transbordé jusqu'à des wagons, en vue de l'envoyer par train jusqu'aux usines de fabrication d'aluminium. Lors de l'opération d'arrimage, on doit donc aspire la poudre d'alumine du bateau, la transporter par convoyeur et la souffler dans des wagons cylindriques. L'alumine étant extraite des cales à un rythme d'environ 700tonnes à l'heure, chaque wagon se remplit en moins de 10 minutes.

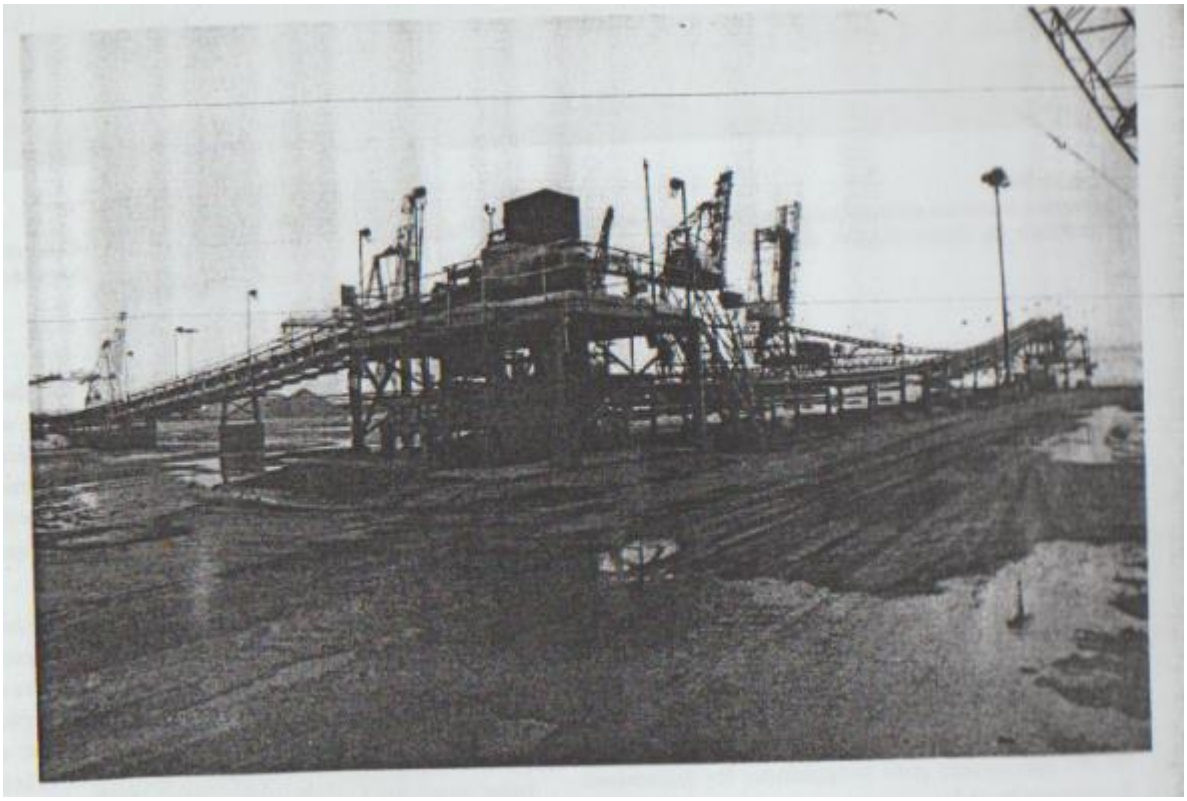


Figure 1.17 : Cette photo montre une des connexions entre deux convoyeurs utilisés pour charger et décharger les navires dans le port de Québec. La longueur totale de ces deux convoyeurs est de plus d'un kilomètre (gracieuseté de la Compagnie d'Arrimage de Québec Ltée)

23.Planification du changement :

Lorsque l'entreprise a décidé de moderniser le procédé d'arrimage de l'alumine, un petit groupe d'experts a été mis sur pied afin de déterminer les méthodes à utiliser et comment automatiser le processus. Il a fallu plus d'un an pour compléter cette étude d'envergure. Avant de produire son rapport , le groupe d'experts a du, entre autres, obtenir l'avis de bureaux d'ingénieurs – conseils, calculer les couts et déterminer comment la main- d'œuvre allant réagir face a cette nouvelle technologie.

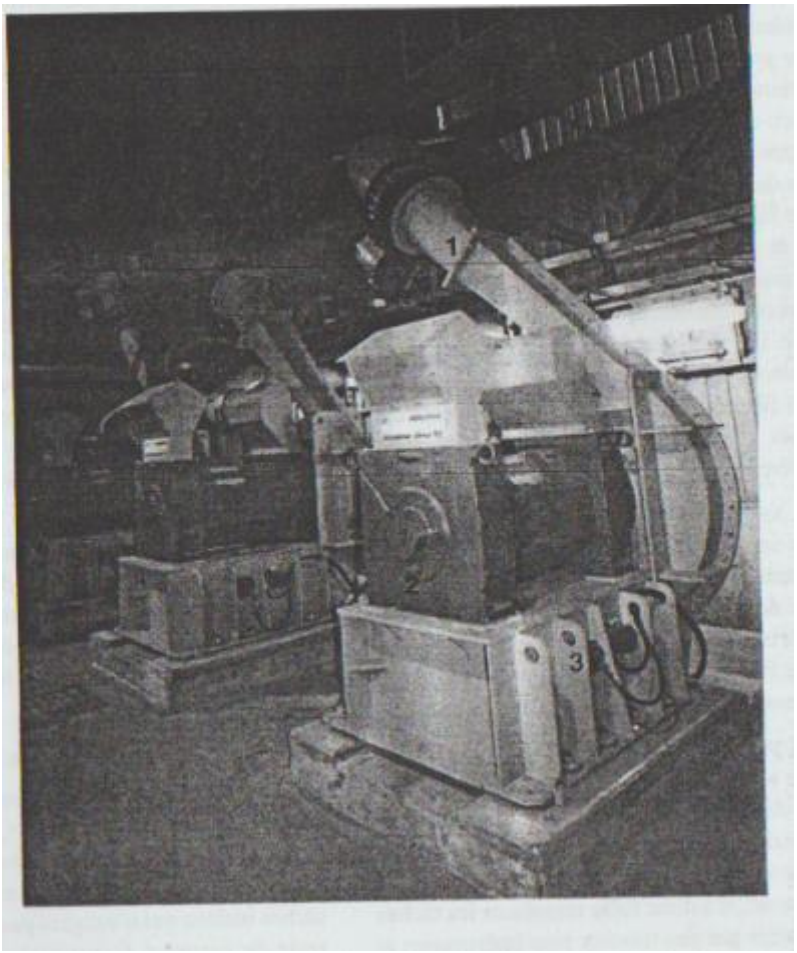


Figure 1.18 : Des moteurs triphasés de 500hp, 4160 V ; 3600r/min actionnent les pompes qui siphonnent la poudre d'alumine des bateaux et la soufflent vers les wagons. La photo montre : (1) une valve pneumatique, (2) un détecteur de température des enroulements du moteur et (3) un détecteur de vibrations (gracieuseté de la Compagnie d'Arrimage de Québec Ltée).

Une fois l'étude terminée, l'entreprise a commencé la transformation graduelle du procédé d'arrimage. Il a ainsi fallu installer de nouveaux équipement entrainés par des centaines de moteurs dont les puissances pouvaient varier entre une fraction de hp et 1500 hp. De plus, il fallait coordonner le fonctionnement normal de tous ces moteurs à partir d'un poste de commande central, comme ces moteurs se trouvaient souvent à des centaines de mètres les uns des autres, la communication de leur état représentait déjà un premier défi. Ainsi, il fallait en tout temps communiquer la vitesse, le courant absorbé, le déséquilibre des phases, la température des paliers, la température des enroulement, la vibration, etc. Si, par exemple, un des moteurs de 500 hp commençait à vibrer, ce système de surveillance afin d'éviter d'endommager les paliers ou les structures.

De plus, il fallait en tout temps maintenir au niveau désire le débit de poudre d'alumine passant dans des tuyaux de 0.5 m de diamètre. Ce débit est contrôlé par des valves actionnés par servo - moteurs et utilisant un système de commande de position.

De toute évidence, l'automatisation d'un tel procédé industriel ne pouvait se réaliser qu'avec des ordinateurs. Mais, durant la transition, il fallait bien que l'équipement ancien continue à fonctionner comme auparavant, y compris les relais traditionnels, le câblage, les interrupteurs de fin de course, les boutons – poussoirs, etc. Il a donc fallu marrer temporairement la nouvelle technologie et l'ancienne. Les ordinateurs devaient commander une nouvelle section alors que les opérateurs humains continuaient à faire fonctionner une section ancienne. De plus, il a fallu intégrer l'ancien câblage avec le nouveau en utilisant des câbles coaxiaux et des liens à fibres optiques. Il a donc fallu installer des dispositifs spéciaux agissant comme interfaces entre ces différents systèmes de communication.

Qui plus est, tous ces changements devaient se faire non pas dans un laboratoire, mais sur un chantier ou des grues géantes et des moteurs puissants risquaient de provoquer des dégâts importants à la moindre erreur. Durant la transition, il ne fallait donc pas négliger la fiabilité du fonctionnement de l'équipement ni la sécurité du personnel.

Autre point à prendre en considération : l'impact de cette nouvelle technologie sur l'emploi et les habitudes de travail des ouvriers. En effet, la modernisation et l'automatisation du processus d'arrimage rendaient plusieurs taches caduques. Pour maintenir de bonnes relation de travail, il a donc fallu remplacer les taches devenuesdésuètes par des travaux plus intéressants et moins routiniers. La migration vers la nouvelle technologie a donc dû se faire à un rythme compatible avec le maintien des bonnes relations de travail.

24.Le personnel apprend à maitrise les API :

Comment a-t-on fait pour installer les nouveaux équipement et qui s'en est chargé ?Soulignons tout d'abord que les techniciens se sont adaptés rapidement à la nouvelle technologie. Les communications verbales avec les représentants des firmes fournissant les API et les dispositifs à fibre optique ont permis aux techniciens plus âgés de comprendre le langage de l'Internet, de l'Ethernet, et autres protocoles de communication. En parallèle avec la modernisation graduelle de l'équipement et des systèmes de commande, on a donc assisté à une transformation semblable au niveau humain. Les connaissances acquises depuis des années se sont graduellement enrichies au contact des nouvelles technologies.

La plupart des ouvriers étaient heureux d'apprendre ces nouveaux concepts, mais certains craignaient de ne pas pouvoir comprendre la nouvelle technologie. Cependant, le temps et le contact journalier avec les nouveaux équipements sont vite venus à bout de ces réticences. Au fur et à mesure que la transitions poursuivait le langage devenait plus familier et les taches plus routinières. Les ordinateurs se montraient môns menaçants lorsque le technicien aux cheveux gris réalisait qu'il faisait maintenant partie d'une équipe « high-tech » Il savait maintenant comment utiliser le clavier, comment interpréter les informations affichées à l'écran, quoi faire lorsqu'une alarme se mettait à sonner, et comment transmettre ces informations à ses collègues (Fig1.19)

Par ailleurs, les jeunes techniciens qui venaient d'être embauchés ont eu l'avantage de travailler avec des collègue cumulant de nombreuses années d'expérience dans un autre domaine. Les échanges techniques ont favorisé le renforcement des liens de camaraderie entre anciens et nouveaux

25.Liaisons entre les API :

L'entreprise d'arrimage est contrôlée par une quinzaine d'automates programmables. Quelques API sont interconnectées par un réseau de communication afin de coordonner les opérations. D'autres API commandent des tache isolées qui n'exigent pas une intégration avec le reste du système. Cependant, on étudie actuellement la possibilité de centraliser tous les API, le but étant de contrôler l'appel de puissance électrique maximale exigée par l'entreprise. En effet, le cout de l'énergie

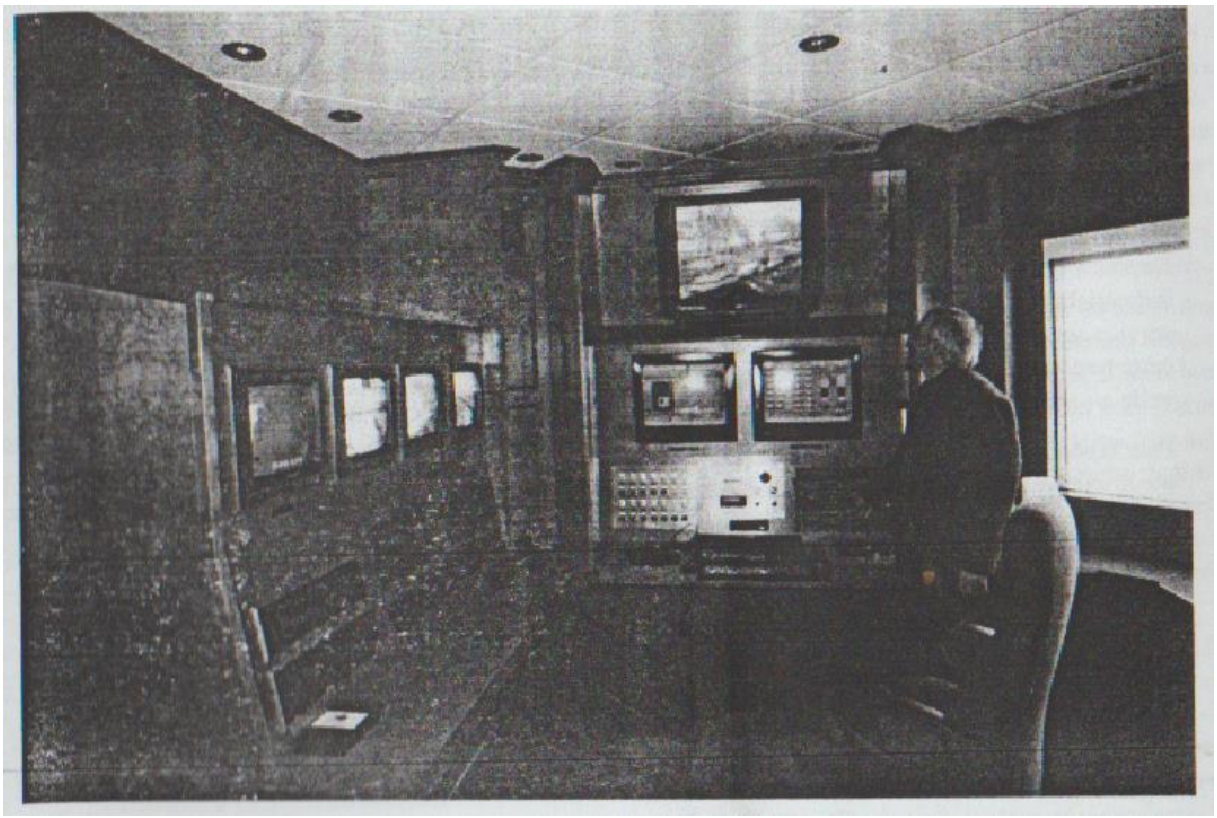


Figure 1.19 : Centre de commande et de surveillance des produits en en vrac . Les réseaux de communication, le contrôle des caméras et l'affichage de l'état des API sont centralisés à cet endroit (gracieuseté de la Compagnie d'Arrimage de Québec Ltée).

Electrique consommée peut être réduit de façon significative en évitant que les gros moteurs fonctionnent en même temps. En échelonnant la mise en marche de ces moteurs, on peut ainsi réduire le cout mensuel de l'électricité de plusieurs milliers de dollars.

L'entreprise d'arrimage couvre un territoire de plus de 50 hectares. L'utilisation de caméras de surveillance joue donc un rôle important pour assurer la sécurité du personnel et de l'équipement.

26.Programmation des API :

Naturellement, il a fallu implanter la commande du procédé industriel sur ordinateur. On a donc embauché des spécialistes pour programmer les techniques de commande et les incorporer dans la mémoire des automates programmables. Cette tâche a été confiée à de nouveaux diplômés des écoles techniques qui ont travaillé en collaboration avec les représentants des fabricants d'API. Les API ont été programmées en utilisant des langages comme le GRAFCET, et plus récemment les schémas blocs ou « Function Block Diagrams »(FBD). Pour un programmeur expérimenté, l'utilisation des schémas blocs est souvent plus efficace que diagramme en échelle.

Lors de la conception d'une nouvelle méthode de commande (disons, d'une grue), le programmeur doit simuler le fonctionnement de la grue afin de déceler des failles éventuelles dans son programme. Cette phase préliminaire de simulation est très programme. Cette phase préliminaire de simulation est très importante car elle permet au programmeur de vérifier la sécurité et la fiabilité du système. Enfin, lorsque le programme est mis à l'épreuve pour la première fois, une équipe technique surveille de près le déroulement du procédé afin de s'assurer que le tout fonctionne tel que prévu. A cette fin, des boutons poussoirs d'urgence sont installés aux points stratégiques afin d'interrompre une activité qui aurait été mal programmée.

Les panneaux de commande des moteurs contiennent les contacteurs usuels utilisés pour démarrer et arrêter les moteurs. Ces panneaux contiennent aussi des dispositifs de surveillance qui vérifient en permanence la condition du moteur. Ces informations sont transmises à l'API concerné par fibre optique. Un seul brin de fibre optique permet de véhiculer toute l'information provenant du moteur (fig1.22).

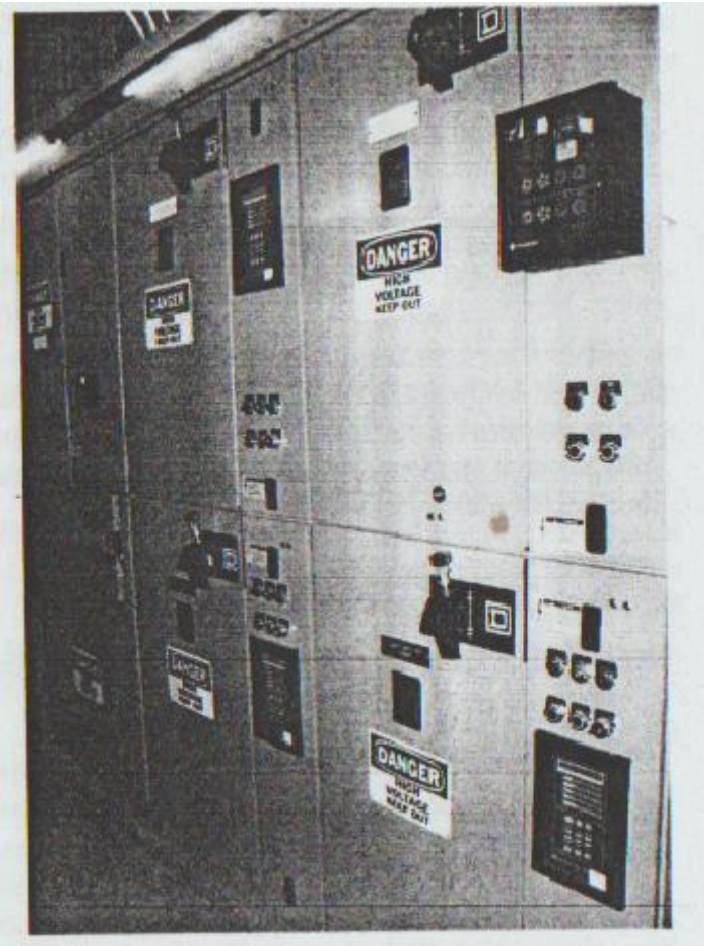


Figure 1.20 : Ce panneau de commande des moteurs triphasés à 600 V est équipé de dispositifs Power logic , qui permettent d'observer la tension. Le courant, les puissances active et réactive, les formes d'ondes, etc. La technologie Powerlogic, permet en outre la communication avec Internet (gracieuseté de Schneider Electric).

De s'assurer que le tout fonctionne tel que prévu . A cette fin , des boutous-poussoirs d'urgence sont installes aux pointes stratégique afin d'interrompre une activité qui aurait ètè mal programmée . les panneaux de command des moteur contiennent les contacteurs usuels utilisés pour démarrer et arrêter les moteurs . Cet panneaux contiennent aussi des dispositifs de surveillance qui vérifient en permanence la condition du moteur. Cet information sont transmises à l' API concerné par fibre optique .Un seul brin de fibre optique permet de véhiculer touts l' information provenant du moteur(fig1.22)

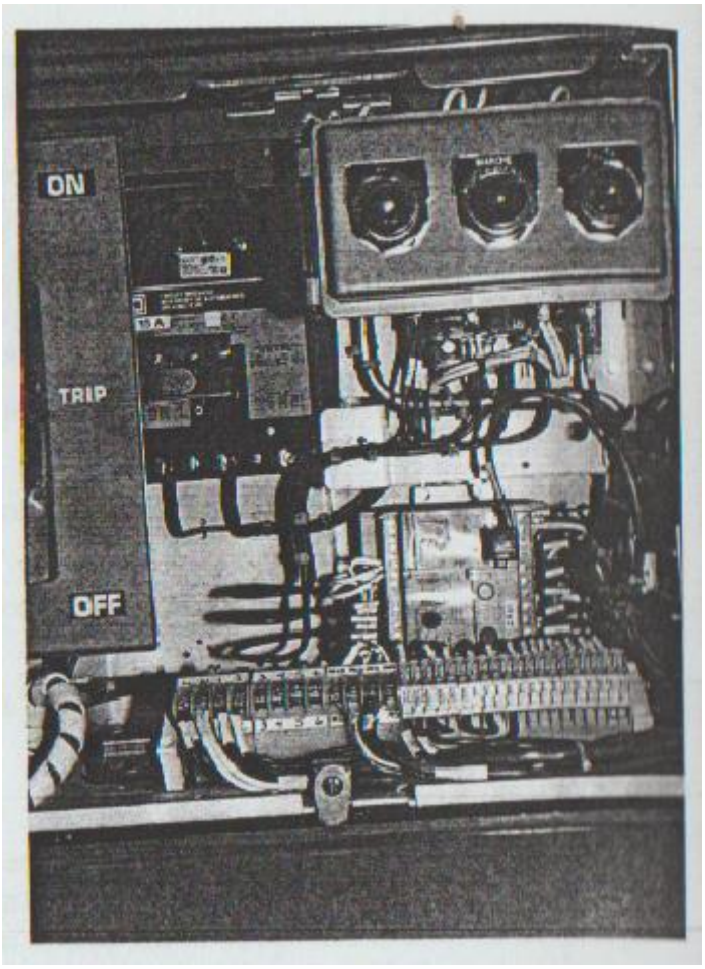


Figure1.21 : Ce démarreur triphasé comprend un disjoncteur et un contacteur .Il contient en plus un système de protection contre les surcharges mécaniques , l'échauffement excessif des enroulements , le déséquilibre des tensions et des courants il permet de surveiller la facteur de puissance , le temps de démarrage ,le courant de démarrage et les défauts de mise à la terre

Toutes ces informations sont transmises par câble coaxial ou par fibre optique à un API ,en utilisant le protocole Modbus (gracieuseté de Schneider Electric).

En conclusion, l'installation d'un système de commande moderne basé sur les API a réduit le cout de l'arrimage. En même temps, ces nouvelles technologies ont stimulé le personnel et créé des emplois plus agréables.

Les photos présentées dans ce texte et leurs légendes (Fig1.18 à 1.24) donnent une vision globale de ce programme typique de la modernisation d'une industrie.

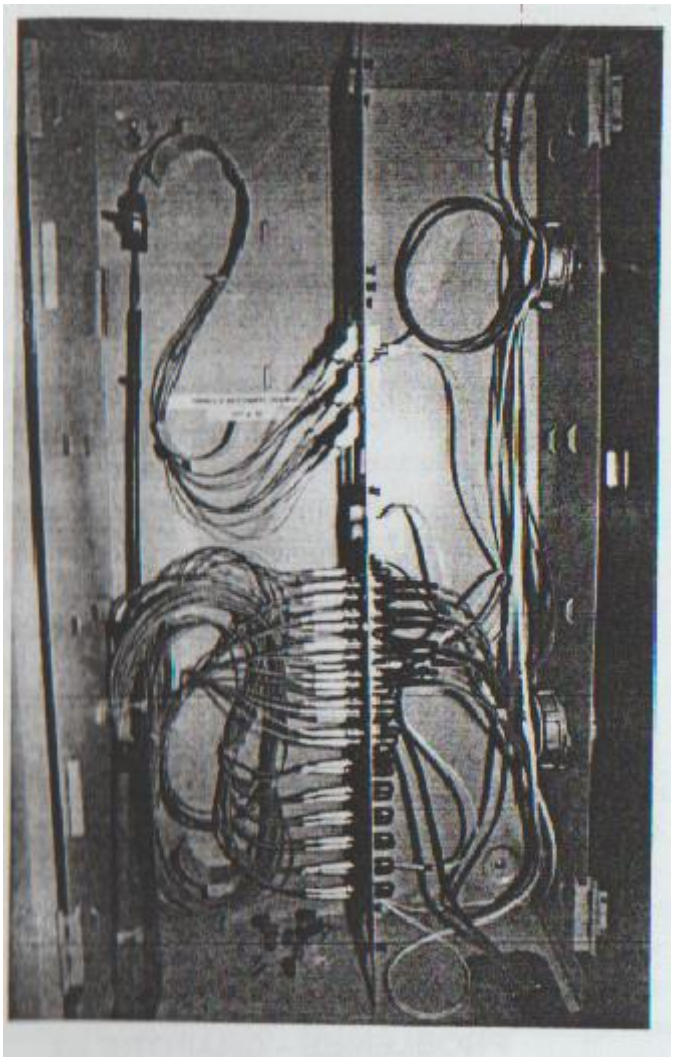


Figure 1.22 : Cette boite de jonction facilite l'interconnexion des divers câbles de commande (gracieuseté de Schneider Electric).

27.Evolution vers une entreprise virtuelle :

Nous venons de voir comment les API ont permis d'automatiser un procédé industriel. Mais la tendance à l'automatisation ne s'arrête pas là. En effet, l'Internet permet maintenant d'automatiser également les transactions commerciales. L'Internet met en communication tous les secteurs concernés depuis la fabrication jusqu'à l'utilisation d'un produit, incluant l'achat des matièrespremières et la vente du produit, fini, et peut même communiquer directement avec les API équipés des ports de communication requis. Pour permettre cette intégration, de grands efforts sont déployés afin de normaliser les modes de communication entre les différents secteurs. Le but ultime est de créer un environnement ou le commerce électronique (« e-business ») remplacera toutes les transactions sur papier.

Il est devenu un des maillons de la chaine de production et de commercialisation qui s'intégré dans le réseau Internet.

28.Résumé :

Dans ce chapitre, nous avons vu qu'avec une dizaine de dispositifs de base, on peut réaliser des systèmes de commande allant du simple démarreur de moteur aux contrôleurs de procédés industriels les plus complexes. Certains de ces dispositifs, comme les sectionneurs, disjoncteurs, contacteurs, relais thermiques, résistances, transformateurs sont introduits directement dans le circuit de puissance pour interrompre, mesurer ou modifier le courant de l'appareil commandé .

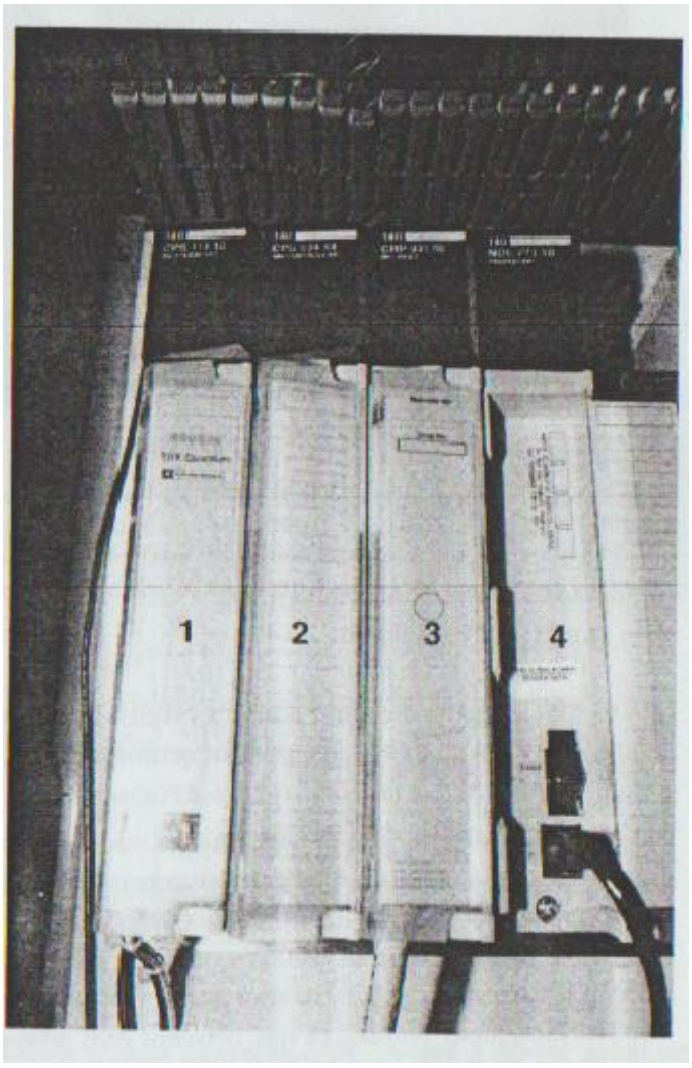


Figure 1.23 : Cet automate programmable de marque Quantum,comprend quatre parties. De gauche à droite m (1à le module des entrées/sorties. (2) l'unité centrale de traitement, (3) le bloc d'alimentation et (4) un serveur web qui établit une connexion avec l'Internet à raison de 100 Mb/s (gracieuseté de Schneider Electric).

D'autres dispositifs sont utilisé dans le circuit de commande à basse tension. Ces composants comprennent les boutons – poussoirs, les interrupteurs spéciaux, les lampes témoins, les relais de commande pouvant comporter plusieurs contacts qui sont normalement ouverts ou

normalement fermés et qui peuvent être temporisés. Ils sont interconnectés de façon à réaliser « l'Intelligence » du système de commande et les fonctions de signalisation.

Nous avons appris comment fonctionnent les systèmes de commande les plus courants. Différents types de démarreurs permettent le démarrage des moteurs asynchrones à pleine tension ou à tension réduite (par résistances, par autotransformateurs, par enroulement partiel ou en étoile – triangle). Ces démarreurs sont parfois associés à des systèmes de commande permettant le démarrage par à-coups, le freinage, et l'inversion du sens de rotation.

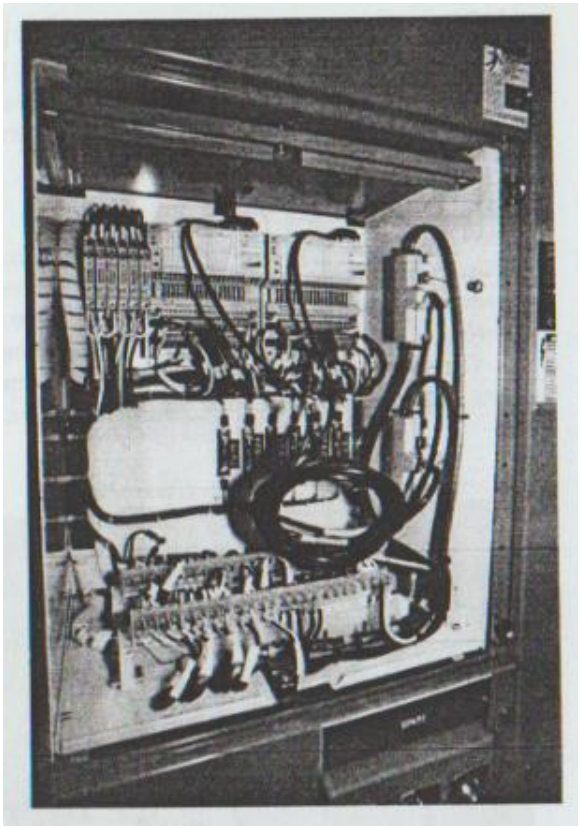


Figure 1.24 : Cet API appartient à la famille Momentum, En plus des entrées et sorties habituelles, il possède aussi un port de communication permettant la réception et l'émission de messages sur le réseau Internet (gracieuseté de Schneider Electric).

Les armoires de commande à relais traditionnelles sont maintenant remplacées par des automates programmables industriels (API). L'API est en effet beaucoup plus flexible car il est programmable. Le cœur de ces API est un ordinateur ou unité centrale de traitement qui simule les contacts et relais de bobines requis pour réaliser les différentes fonctions de commande. Cette unité centrale est interfacée avec un module d'entrée qui lui transmet les états des différents contacts de boutons – poussoirs, interrupteurs spéciaux et contacteurs. Un module de sortie transmet les ordres de l'unité centrale aux bobines des contacteurs à l'aide d'une alimentation et de contacts de sortie. Les fonctions de commande de l'unité centrale sont inscrites dans la mémoire de l'API au moyen d'une console de programmation, en utilisant un langage tel que le diagramme en échelle, le langage booléen ou le Grafcet.

29. **Conclusion** :

Dans ce chapitre nous avons donné une description générale sur les automates programmables ainsi que leur structure interne et leurs fonctionnements.

CHAPITRE 2

COMMANDE DES API

&

PROGRAMMATION STEP7.

1. Introduction :

L'automate programmable industriel API est aujourd'hui le constituant le plus répandu pour réaliser des automatismes. On le trouve pratiquement dans tous les secteurs de l'industrie car il répond à des besoins d'adaptation et de flexibilité pour un grand nombre d'opérations. Cette émergence est due en grande partie, à la puissance de son environnement de développement et aux larges possibilités d'interconnexions.

Le but de ce chapitre est l'étude théorique des systèmes automatisés, précisément l'automate programmable industriel.

Dans ce chapitre, nous détaillons en premier lieu l'historique et la définition de l'API, ensuite nous présenterons la structure générale d'un système automatisé et enfin nous terminerons ce chapitre avec une présentation détaillée sur les automates programmables industriels et la structure modulaire essentiellement S7-300.

2. Historique

Les automates programmables industriels (API) sont apparus à la fin des années soixante aux Etats Unis, à la demande de l'industrie automobile américaine (General Motors en leader), qui réclamait plus d'adaptabilité de leur systèmes de commande.
Les couts de l'électronique permettant alors de remplacer avantageusement les technologies actuelles.

3. Définition

Un automate programmable industriel, ou API, est un dispositif électronique programmable destiné à la commande des processus industriels par un traitement séquentiel. Il envoie des ordres vers les prés actionneurs (partie opérative ou PO côté actionneur) à partir de données d'entrées (capteurs) (partie commande ou PC côté capteur), de consignes et d'un programme informatique.

4. Structure d'un système automatisé

Un API constitué essentiellement **: Figure 2 .1.**

- D'une unité centrale.
- D'un module d'entrée (ou interface d'entrées).
- D'un module de sortie (ou interface de sortie).
- D'un coupleur.

- D'une console de programmation ou autre périphérique.

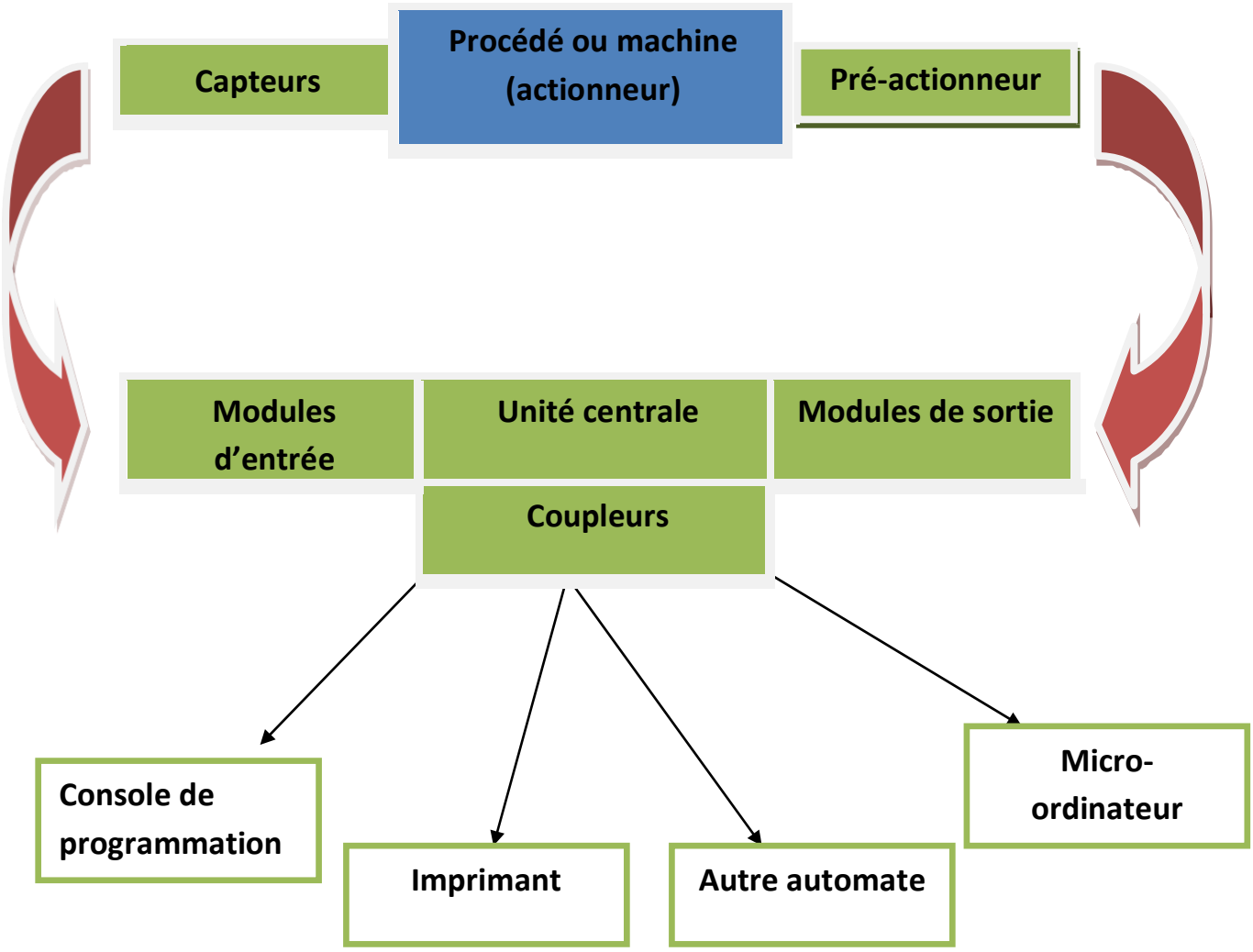


Figure.2 .1 : structure d'un système automatisé

- **L'unité centrale (UC) :**

C'est le cœur de la machine, comporte le(s) processeur(s) (unité de traitement logique ou numérique et la mémoire(s).

- **Le module des entrées (ou interface d'entrées) :**

Il permette de raccorder à l'automate les différents capteurs.

- **Le module des sorties (ou interface de sortie) :**

Il permette de raccorder à l'automate les différents pré-actionneurs.

- **Le Coupleur :**

Ce sont des cartes électroniques qui assurent la communication entre les périphériques (Modules d'E/S ou autres) et l'unité centrale.
En général, les échanges entre l'UC et les modules d'E/S s'effectuent par l'intermédiaire d'un bus interne.

- **Les consoles**

Il existe deux types de consoles :

Console d'exploitation : permet le paramétrage et les relevés d'informations (Modification des valeurs et visualisation). Console de programmation, réglage et exploitation. Cette dernière effectue dans la phase de programmation:

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

5. Les avantages et inconvénients des API

Les a vantages	Les inconvenient
- Il facilite la documentation des applications, donc leur maintenance. - L'API est favorable aux traitements évalues, calculs numérique, régulation etc. - La possibilité d'agir deux paramètres matériel et programme. - Les API permettent d'ajuster la disponibilité du système aux besoins	- l'API ne supprime pas tout le reliage, il reste le câblage du circuit de puissance - sa vitesse peut s'avérer insuffisante. - le déroulement cyclique des programme peut s'avérer un facteur de complexité et limite les possibilités d'organisation des taches.

Tablea2 .1 : les avantages et inconvénients.

6. Nature des informations traitées par l'automate

Les informations peuvent être de type :

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

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

6.3 Numérique : l'information est continue dans des mots codés sous forme binaire ou bien hexadécimale, C'est le type d'information délivrée par un ordinateur.

7. Architecture des automates

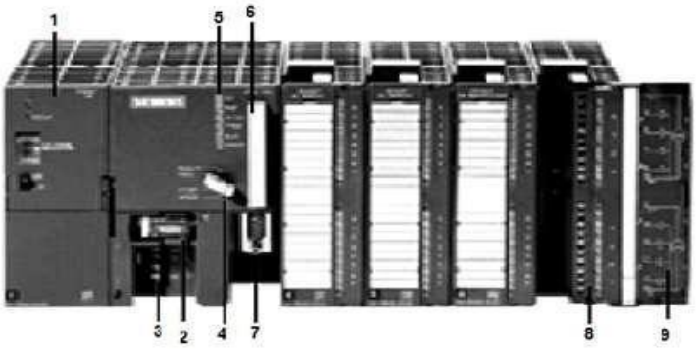
7.1 Aspect extérieur :

Les automates peuvent être de types compact ou modulaire **Figure. 2.2.**

- **De type compact** : on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Grouzet…) des micros automates. 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 analogique…) et recevoir des extensions. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes.
- **De type modulaire :** le processeur, l'alimentation et les interfaces d'entrées/sorties résident dans des unités séparées (**modules**) et sont fixées sur un ou plusieurs **racks** contenant '' le fond de panier'' (bus plus connecteurs).Ces automates sont intégrés dans les automatismes complexes ou de puissance, capacité de traitement et flexibilité sont nécessaire.

Automate modulaire (Modicon)

Automate compact (Allen_Bradley)



Automate modulaire (Siemens)

Figure.2.2. Les types des automates

1- Module d'alimentation
2- Pile de sauvegarde
3- Connexion au 24Vcc
4- Commutateur de mode (à clé)
5- LED de signalisation d'état et de défauts.
6- Carte mémoire
7- Interface multipoint (MPI)
8- Connecteur frontal
9- Volet en face avant

7.2 Structure interne :

Cette partie comporte quatre parties principales :

1- une mémoire.

2- Un processeur.

3- Des interfaces d'Entrées/Sorties.

4- Une alimentation (240Vac-24Vcc).

Ces quatre parties sont reliées entre elles par des bus (ensemble câblé autorisant le passage de l'information entre ces quatre secteur de l'API).ces quatre parties réunies forment un ensemble compact appelé automate **Figure. 2 .3.**

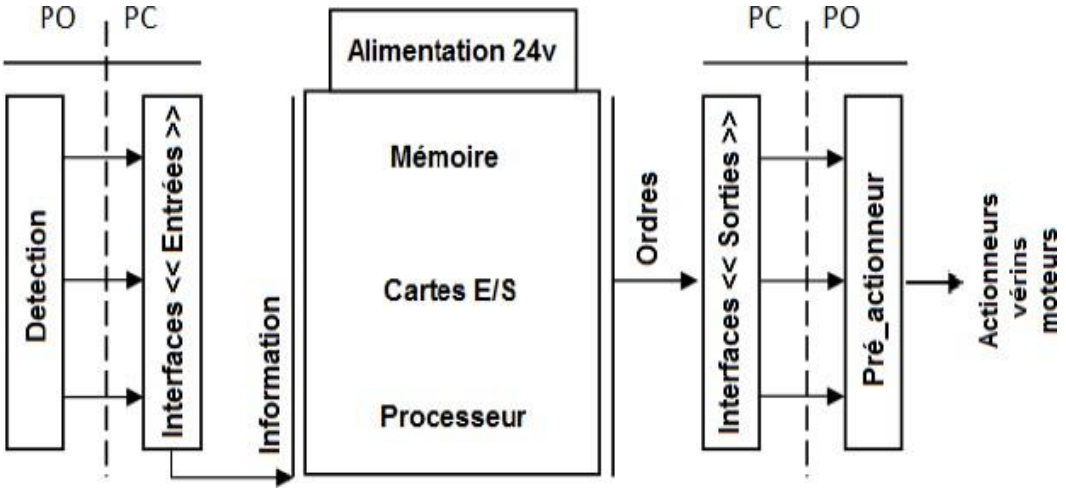


Figure.2.3. Structure interne d'un API

1- Le processeur :

Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'E/S et d'autre part à gérer les

2- Les interfaces :

- L'interface d'entrée comporte des adresses d'entrée,
- L'interface de sortie comporte des adresse de sorties, une pour chaque actionneur.
- Le nombre de d'E/S varie suivant le type automates.

- Les cartes d'E/S ont une modularité de 8, 16 ou 32voies. Elles admettent ou délivrent des tensions continues 0

3- La mémoire :

Elle est conçu pour recevoir, gérer, stocker des informations issues des différents secteurs du système qui sont le terminal de programmation (PC ou console) et le processeur qui lui gèrent et exécute le programme. Elle reçoit également des informations en provenance de capteurs.

4-L'alimentation :

Tous les automates actuels utilisent un bloc d'alimentation alimente 240 V AC et délivrant une tension de 24V CC

7.3 Fonction réalisées :

Les automates compacts permettent de commander des sorties en T.O.R et gèrent parfois des fonctions de comptage et de traitement analogique.
Les automates modulaires permettent de réaliser de nombreuses autres fonctions grâce à des modules intelligents que l'on dispose sur un ou plusieurs racks. Ces modules ont l'avantage de ne pas surcharger le travail de la CPU car ils disposent bien souvent de leur propre processeur.

7. 4 Les types de cartes :

- **Carte d'entrées/sorties :**

Au nombre de 4, 8, 16 ou 32, elles peuvent aussi bien réaliser des fonctions d'entrées, de sorties ou les deux. Ce sont les plus utilisées et les tensions disponibles sont normalisées 24v 48v (continu) ou 110v, 230v (alternatif).
Les voies peuvent être indépendantes ou posséder des "communs". Les cartes d'entrées permettent de recueillir l'information des capteurs, boutons ... qui lui sont raccordés et de la matérialiser par un bit image de l'état du capteur.
Les cartes de sorties offrent deux types de technologies : les sorties à relais électromagnétiques (bobine plus contact) et les sorties statiques (à base de transistors ou de triacs).

- **Carte de comptage rapide :**

Elles permettent d'acquérir des informations de fréquences élevées incompatible avec le temps de traitement de l'automate.
Exemple : signal issu d'un codeur de position.

- **Carte de commande d'axe** :

Elles permettent d'assurer le positionnement avec précision d'élément mécanique selon un ou plusieurs axes. La carte permet par exemple de piloter un servomoteur et de recevoir les informations de positionnement par un codeur. L'asservissement de position pouvant être réalisé en boucle fermée.

- **Cartes d'entrées/sorties analogiques :**

Elles permettent de réaliser l'acquisition d'un signal analogique et sa conversion numérique(CAN) indispensable pour assurer un traitement par le microprocesseur. La fonction inverse (sortie analogique) est également réalisée. Les grandeurs analogiques sont normalisées : 0-10V ou 4-20mA.

8. La programmation des API :

La programmation des API peut s'effectuer de trois manières possibles : sur l'API lui même à l'aide de touche, avec une console de programmation relié par un câble spécifique ou avec un PC et un logiciel approprié.

Dans notre système, la commande des déférents mouvements sont géré par un automate S7-300.

9. L'automate S7-300 :

L'automate S7-300 est un mini automate modulaire de la famille SIMATIC, destiné à des taches d'automatisation moyennes hautes gammes, avec possibilité d'extensions jusqu'à 32 modules et une mise en réseau par l'interface multipoint (MPI), PROFIBUS et Industriel Ethernet **Figure. 2.4 .**

Figure. 2.4. L'API S7-300

- **Les modules S7-300 :**

L'automates programmables S7-300 est d'une forme modulaire, permet un vaste choix de gamme de module suivant :

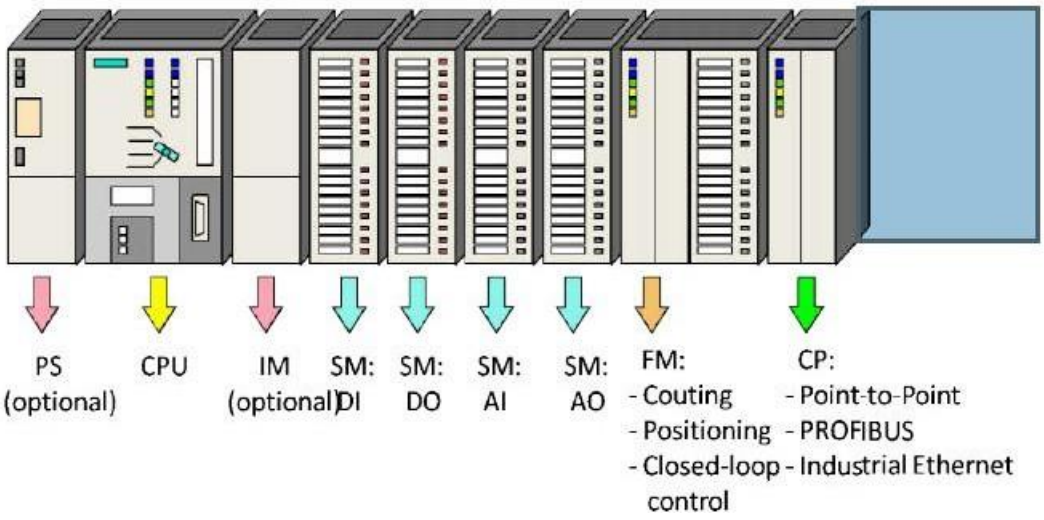


Figure 2 .5 : Les différents modules constituant S7-300

- **Le module d'alimentation (PS)**

Le module d'alimentation (PS) délivre un courant de sortie assignée de 2A ,5A ou 10A sous une tension de 24 volts.

La tension de sortie a séparation galvanique pour la protection de la CPU contre les courts circuits.

- **Unité centrale (CPU) :**

C'est une carte électronique construite autour d'un ou plusieurs processeurs et mémoire.

La CPU possède un système d'exploitation, une unité d'exécution et des interfaces de communication. Essentiellement la CPU lit l'état des signaux d'entrée et exécute le programme utilisateur séquentiellement **Figure . 2 .6.**

Figure 2 .6 : CPU S7-300.

1. Interfaces MPI :

Chaque CPU est équipée d'une interface MPI pour la connexion de la console de programmation (PG) ou un appareil par exemple adaptateur PC.

2. Commutateur de mode de fonction :

Les modes de fonctionnement sont :

- RUN-P : exécution de programme, accès en écriture et en lecture avec la PG.
- RUN : exécution de programme, accès en lecture seule avec la PG.
- STOP : le programme n'est pas exécuté, toutes les fonctions avec la PG sont autorisées.
- MRES : position dans laquelle en effacement générale de la CPU peut être effectué.

3. La carte mémoire :

Une carte mémoire peut être montée à la CPU ; elle conserve le contenu du programme en cas de coupure de courant, même en l'absence de la pile.

4. Le processeur :

C'est le cerveau de l'automate. Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'entrée et de sortie et d'autre part à gérer les instructions du programme. Il est composé :

- D'une Unité Logique (UL) qui traite les opérations ET, OU et la Négation.
- D'une Unité Arithmétique et Logique (UAL) qui traite les opérations de temporisation, de comptage et de calcul.
- D'un Accumulateur qui est un registre de travail dans lequel se range une donnée ou un résultat.

- D'un Décodeur d'instruction qui décode l'instruction a exécuté en y associant les microprogrammes de traitement.
- D'un Compteur Programme ou Compteur ORDINAL qui l'adresse à la prochaine instruction à exécuter et gère ainsi la chronologie de l'exécution des instructions du programme.

5. La mémoire :

Le stockage des données et des programmes s'effectue dans la mémoire.

Ces mémoires peuvent être :

- Des RAM ou des EPROM durant la phase d'étude et de mise au programme.
- Des RAM ou des PROM durant la phase d'exploitation.

➢ **Module de signaux (SM) :**

Ils servent d'interface entre le processus et l'automate. Il existe des modules d'entrée TOR, des modules de sortie TOR ainsi que des modules d'entrée et de sortie analogiques. Les modules d'entrée/sortie sont des interfaces entre les capteurs et les actionneurs d'une machine ou d'une installation **Figure 2.7.**

Figure. 2. 7 : Un module SM de S7-300.

1. Les entrées Tout Ou Rien (TOR) :

Les modules d'entrée tout ou rien permettent de raccorder a l'automate les différents capteurs logiques. Elles assurent l'adaptation, l'isolement, le filtrage et la mise en forme des signaux électroniques. L'état de chaque entrée est donné par une diode électroluminescente

situant sur la carte. Le nombre d'entrées sur une carte est de : 4, 8, 16 32.les tensions d'entrées sont de : 24, 48, 110, 220 volts en courant continu ou alternatif.

2. Les entrées analogiques :

Les cartes d'entrées analogiques permettent de gérer des grandeurs analogiques en variant un code numérique au sein des modules. Il existe 3 types d'entrées analogiques :

- Haut niveau qui accepte une tension de 0 à 10 v et une intensité de 0 à 20 mA ou de 4 à 20mA.
- Thermocouple avec un signal d'entrée de 0 à 20mV, de 0 à 50mV ou de 0 à 100 mV.
- Sender PT 100 avec un signal d'entrée de 0 a100 mV, 0 à 250 mV ou de 0 à 400 mV.

Il existe des modules à 2, 4, 8 voies d'entrées sur le marché ces modules disposent d'un seul convertisseur analogique/numérique, elles sont scrutées les unes à la suite des autre par un multiplexeur a relais.

3. Les sorties Tout Ou Rien :

Les modules de sorties tout ou rien permettent de raccorder a l'automate les différents prés actionneurs.

Les tensions de sorties usuelles sont de 5, 24, 48, 110 ou 220 volts en continu ou en alternatif. Les courants vont de quelque mA a quelque Ampères.

Ces modules possèdent des relais ou bien des triacs des transistors. L'état de chaque sortie est visualisé par une diode électroluminescente.

4. Les sorties analogiques :

Les modules de sorties analogiques permettent de gérer des grandeurs analogiques en faisant varier un code numérique au sein du module. Il existe deux grands types de sorties :

- Avec une résolution de 8 bits.
- Avec une résolution de 12 bits.

Ces sorties peuvent posséder un convertisseur par voie. Le nombre de voies sur ces cartes est de 2 ou 4.

10. Avantage de l'automate S7-300 :

- Une construction compacte et modulaire, libre de contrainte de configuration.
- Une riche gamme de modules adaptés à tous les besoins du marché est utilisable en architecture centralisée.
- Une large gamme de CPU.
- Une large plage de température de -25°C à +60°C.
- Une meilleure tenue aux sollicitations mécaniques.

- Une résistance à la pollution par des gaz nocifs, poussière et humidité de l'air.

11.Fonctionnement de base d'une API :

11.1 Le module central CPU :

La tension du signaleur est connectée sur la barrette de connexion du module d'entrée. Dans la CPU (module central), le processeur qui traite le programme se trouve dans la mémoire, il interroge les entrées de l'appareil pour savoir si elles délivrent la tension ou non. Au même temps, il ordonne au module de sortie de commuter sur le connecteur de la barrette de connexion correspondante en fonction de l'état de tension sur les connecteurs des modules de sorties. Les appareils à positionner et les lampes indicatrice sont connectés ou déconnectés.

11.2. Réception des informations sur les états du système :

Le S7-300 reçoit des informations sur l'état du processus via les capteurs de signaux reliés aux entrées. Il met à jour la mémoire image au début de chaque cycle de programme en transférant l'état des signaux d'entrées des modules vers la mémoire image des entrées qui permet à la CPU de savoir l'état de processus.

11.3. Exécution du programme utilisateur :

Après avoir acquis les informations d'entrée et exécuter le système d'exploitation, la CPU passe à l'exécution du programme utilisateur, qui contient la liste d'instruction à exécuter pour faire fonctionner le procédé. Il est composé essentiellement de bloc de donnée, de bloc d'organisation.

11.4. La commande du processus :

Les consoles de programme ((SIMATIC)) sont des outils pour la saisie, le traitement et l'archivage des données du processus, ainsi que la suppression du programme. Avec l'atelier logiciel ((SIMATIC)), l'utilisateur dispose d'une gamme d'outils complète de chaque tache d'automatisation. Le raccordement entre l'automate et la console est réalisé par l'interface multi points (MPI).

11.5. Mise en œuvre d'un automate :

A partir d'un problème d'automatisme donné, dans lequel on définit les commandes les capteurs, les organes de sortie et le processus à réaliser, il faut établir :

- Le grafcet niveau 1 et le grafcet niveau2.
- Faire le repérage des entrées/sorties.
- Ecrire le programme, le charger dans la mémoire RAM/EPROM et le transférer dans l'unité centrale de l'automate.

- Tester à vide (mise au point).
- Raccorder l'automate a la machine.

12.Programmation STEP7 :

Les automates programmables industriels effectuent des taches d'automatisation traduites sous formes de programme d'application qui définit la manière dont l'automate doit commander le système par une suite d'instructions, le programme doit être écrit dans un langage déterminé avec des règles définies pour que l'automate puisse l'exécuter, pour cela les automates de la famille SIEMENS sont programmés grâce au logiciel STEP7 via une console de programmation ou PC et sous un environnement WINDOWS.

12.1.Définitions du logiciel STEP7:

Step7 est le logiciel de base qui permet la configuration et la programmation des systèmes d'automatisation SIMATIC. Il s'exécute sous un environnement Windows, à partir d'une console de programmation ou d'un PC.

Il existe plusieurs versions : STEP micro/DOS et STEP micro/ Win pour les applications S7-300 et S7-400.

Le logiciel STEP7 offre les possibilités suivantes :

- Configuration et paramétrage du matériel et de communication.
- Création de gestion des projets.
- La création des programmes.
- Gestion des mnémoniques.
- Test de l'installation d'automatisation.
- Le diagnostic lors des perturbations dans l'installation.
- Document et archivage.
- Notre premier objectif est la programmation et le simulation sur STEP7 et la 2eme objectif est la programmation sur **GRAFCET** (voir annexe).

12.2 Création d'un nouveau projet :

Dans le but de créer un nouveau projet sur STEP7, nous devons suivre les étapes suivantes :

1- Double-clique sur l'icône SIMATIC MANAGER qui se trouve dans le bureau

Figure 2. 8.

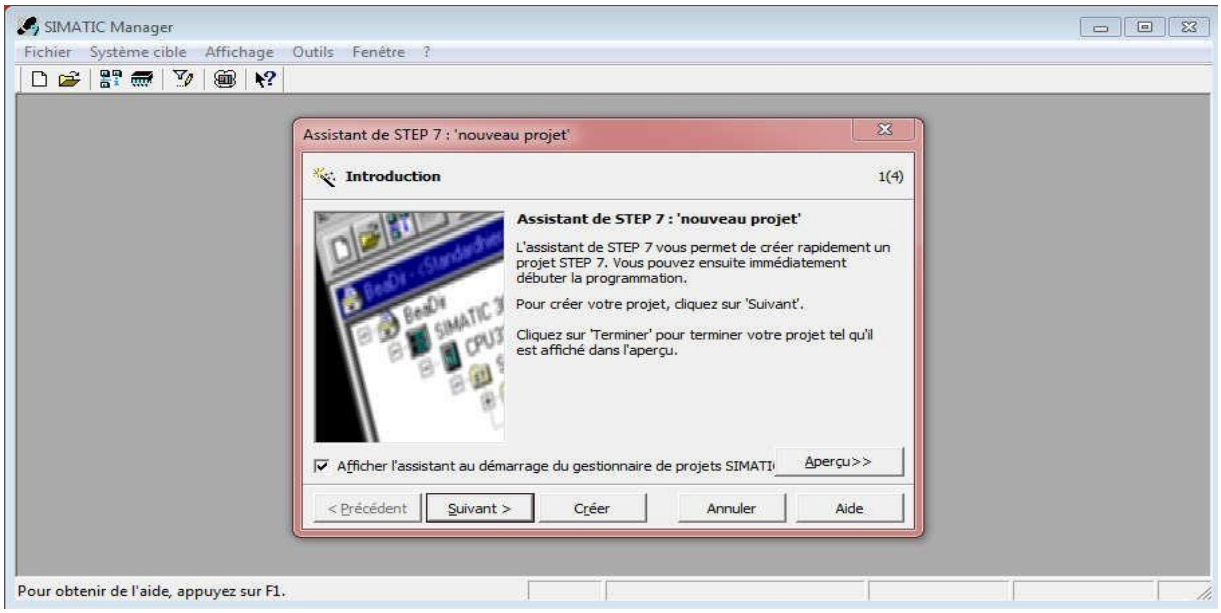


Figure 2. 8: Assistant de STEP7

2- En cliquant sur l'icône « suivant », la fenêtre qui apparaitra nous permettra de choisir la CPU avec la quelle nous voulons travailler. (Dans notre cas nous avons choisi la CPU 313C-2DP) **Figure 2. 9.**

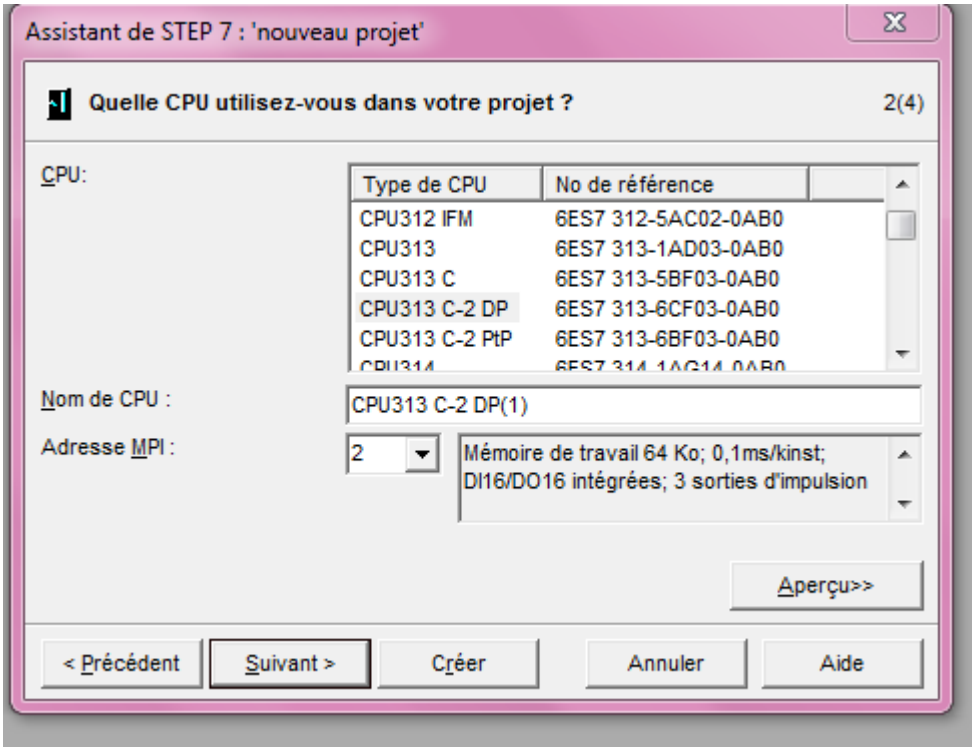


Figure 2. 9 : Choix de la CPU

3- Apres avoir choisit la CPU qui nous convient, la fenêtre qui apparait va nous permettre de choisir les blocs à insérer, ainsi que de choisir le langage de programmation (LIST, LOG ou CONT).

Dans notre cas, nous avons choisi le bloc OB1 (bloc d'organisation) et le langage à contact (CONT) comme langage de programmation **Figure 2. 10.**

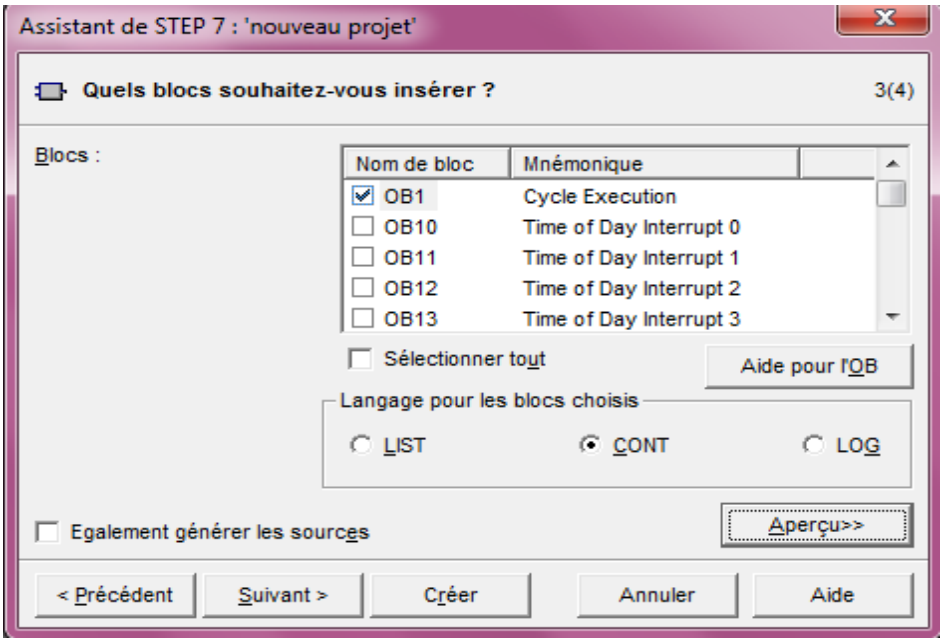


Figure 2. 10 : Choix du bloc à insérer et du langage de programmation utilisé

4- En cliquant sur suivant, l'icône de la création de projet apparait afin de le créer **Figure .2.11**

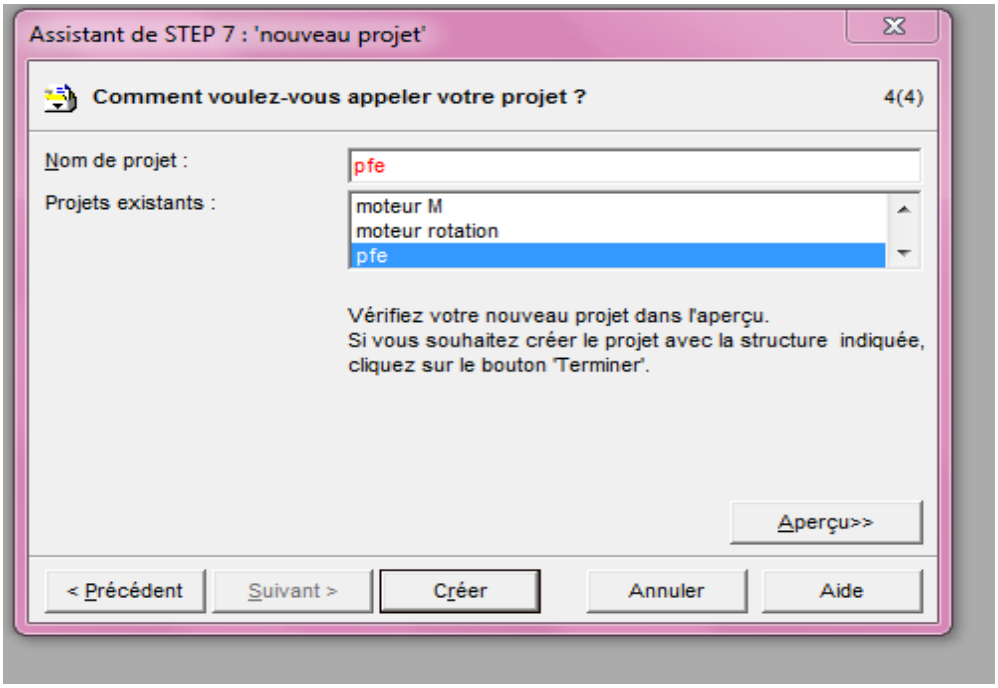


Figure 2. 11 : Choix du nom et création du projet

5-En cliquant sur créer, la fenêtre suivante apparait **Figure 2. 12.**

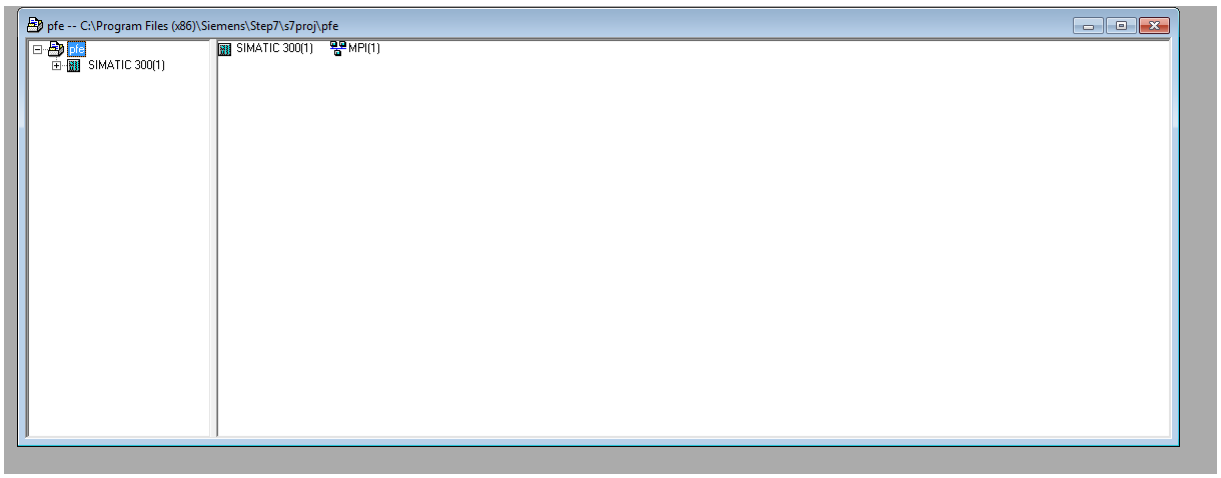


Figure 2. 12 : Fenêtre SIMATIC MANAGER d'un projet

12.3 Configuration matérielle :

La configuration matérielle est une étape importante. Elle consiste à disposer les châssis (rack), les modules et les appareils de la périphérie centralisée.
Les châssis sont représentées par une table de configuration dans la quelle on peut placer un nombre définis de modules comme dans les châssis réels.

Dans notre cas, nous avons choisis une alimentation PS 307 5A, la CPU 313C-2DP, un module d'entré/sortie TOR pour la configuration de notre matériel. (Le choix du nombre d'entrée/sortie doit être fait en fonction des besoins de notre machine). **Figure 2. 13.**

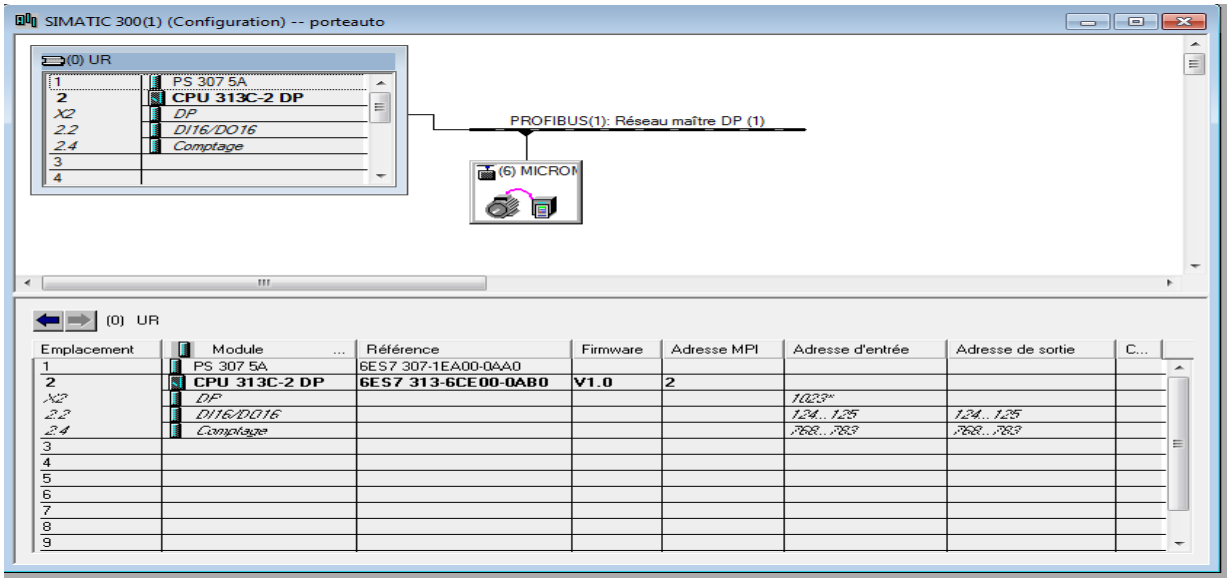


Figure 2. 13 : Configuration du matériel

Nous avons utilisé jusqu'à présent 14 entrées et 6 sorties Tout Ou Rien (TOR).

12.3.1 câblage d'entrée :

Les entrées de l'automate programmable doivent recevoir l'information sous forme de potentiel électrique, dans notre cas c'est **24V**. Une fois que la sortie ait reçu les **24V**, elle s'active. Par ailleurs, l'activation d'une ou de plusieurs entrées peut déclencher une tâche à exécuter au niveau du programme.

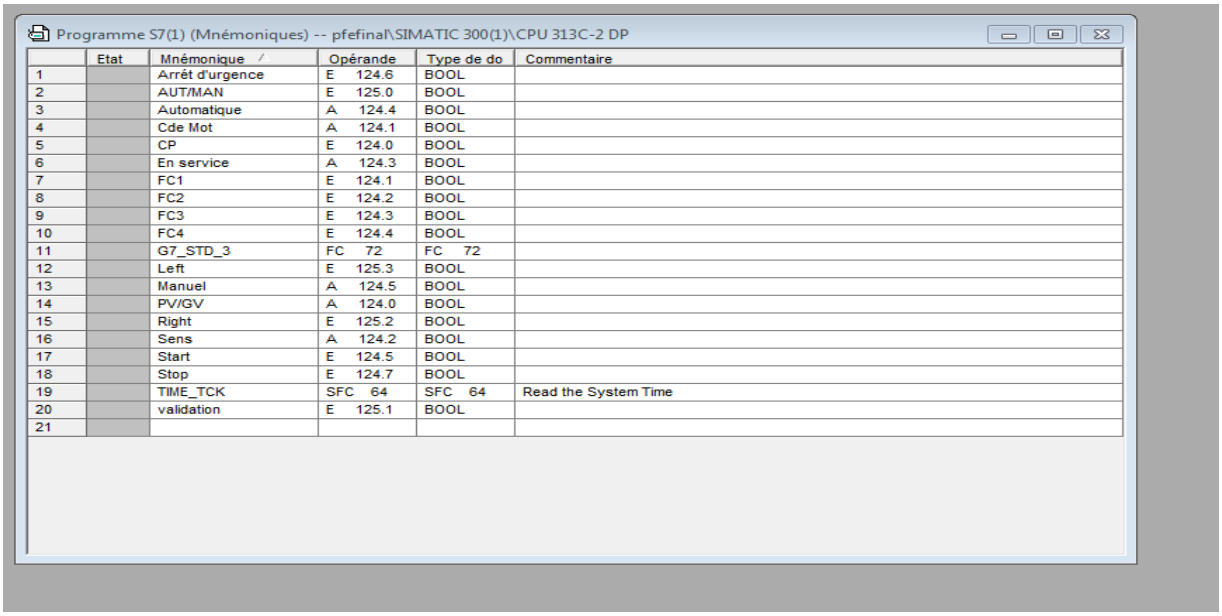
12.3.2 câblage des sorties :

Chaque sortie de l'automate est constituée d'un voyant de signalisation dont la fermeture du contact est commandée par la consigne opérative élaborée par le programme. La fermeture du contact va permettre l'alimentation des voyants. La figure ….. montre un exemple du câblage.

12.4. Table des Mnémoniques :

Une mnémonique est un nom que l'utilisateur définit en respectant les règles de la syntaxe imposée. Il est destiné à rendre le programme lisible et aide donc à gérer facilement le grand nombre de variables couramment rencontrées dans ce genre de programme. Ce nom qu'on a donné à l'adresse pourra être utilisé directement dans le programme une fois les affectations terminées.

La figure suivante **Figure2. 14.** illustre une partie de la table de mnémoniques de notre projet.



Programme S7(1) (Mnémoniques) -- pfefinal\SIMATIC 300(1)\CPU 313C-2 DP

	Etat	Mnémonique	Opérande	Type de do	Commentaire
1		Arrêt d'urgence	E 124.6	BOOL	
2		AUT/MAN	E 125.0	BOOL	
3		Automatique	A 124.4	BOOL	
4		Cde Mot	A 124.1	BOOL	
5		CP	E 124.0	BOOL	
6		En service	A 124.3	BOOL	
7		FC1	E 124.1	BOOL	
8		FC2	E 124.2	BOOL	
9		FC3	E 124.3	BOOL	
10		FC4	E 124.4	BOOL	
11		G7_STD_3	FC 72	FC 72	
12		Left	E 125.3	BOOL	
13		Manuel	A 124.5	BOOL	
14		PV/GV	A 124.0	BOOL	
15		Right	E 125.2	BOOL	
16		Sens	A 124.2	BOOL	
17		Start	E 124.5	BOOL	
18		Stop	E 124.7	BOOL	
19		TIME_TCK	SFC 64	SFC 64	Read the System Time
20		validation	E 125.1	BOOL	
21					

Figure 2. 14 : Table de mnémonique

Dans notre table des mnémoniques nous avons : **Figure2. 14 .**

- Les sorties qui sont adressé avec A (ex : A124.0, A124.1….).
- Les entrées qui sont adressé avec E (ex : E124.0, E125.2….).

12.5 La programmation sur STEP7 [10] :

Le STEP7 dispose de trois langages de programmation, ainsi que d'une méthode utilisant le GRAFCET comme outil.

1. Langage liste (LIST) : image textuelle proche du comportement interne de l'automate.

2. Langage logigramme (LOG) : langage graphique, utilisant les symboles de l'électronique numérique (portes logiques).

3. Langage contact(CONT) : suite de réseaux parcourus séquentiellement dont les entrées sont représentées par des interrupteurs et les sorties par des bobines.

Bobine  **interrupteur**

4. Graph : utilise le GRAFCET comme outil, qui permet de vérifier si le GRAFCET fonctionne correctement, et cela en utilisant la simulation.

12.6 Blocs du programme utilisateur :

Le logiciel STEP7 dans ces différents langages de programmation possède un nombre important de bloc utilisateur, destinés à structurer le programme utilisateur dont on peut citer les blocs importants suivants :

- Bloc d'organisation (OB).
- Bloc fonctionnel (FB).
- Bloc de données d'instance (DB d'instance).
- Blocs de données globales (DB).
- Les fonctions (FC).

12.6.1 Bloc d'organisation (OB) :

Notre projet, contient les blocs que l'on doit charger dans la CPU pour réaliser la tâche d'automatisation, il englobe :

- Les blocs de code (OB, FB, FC, SFB, SFC) qui contiennent les programmes,
- Les blocs de données DB d'instance et DB globaux qui contiennent les paramètres du programme.

On a utilisé le bloc d'organisation **OB1** qui est appelé par le système d'exploitation, il fait appel aux autres blocs qui constituent le programme, lorsqu'on appelle un bloc fonctionnel dans l'OB1 un bloc de donnée associé sera créé automatiquement.

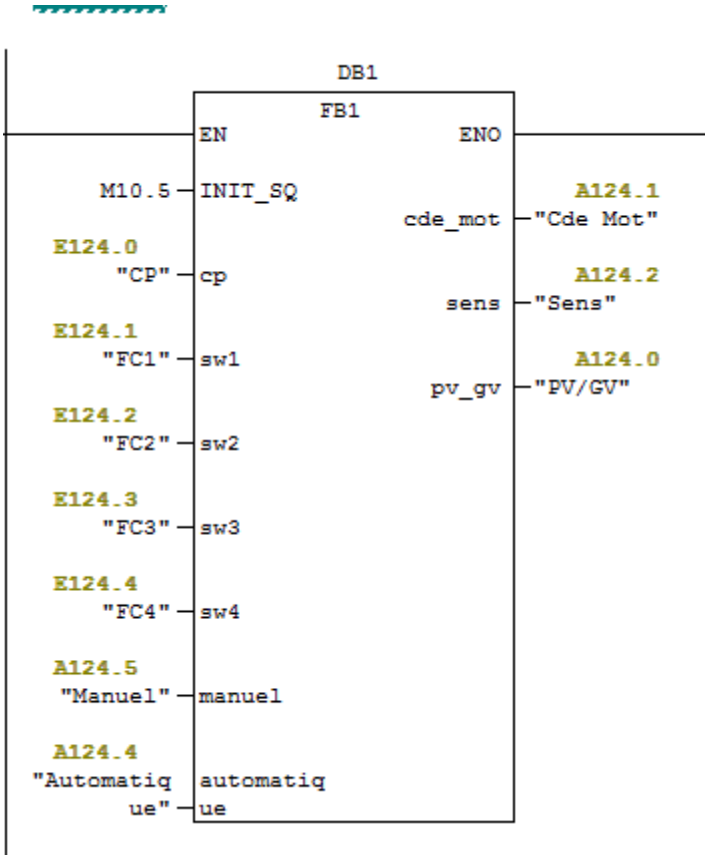


Figure 2. 15 :BLOC OB1**(** l'appel FB1 dans DB1)

12.6.2 Blocs fonctionnelles (FB) :

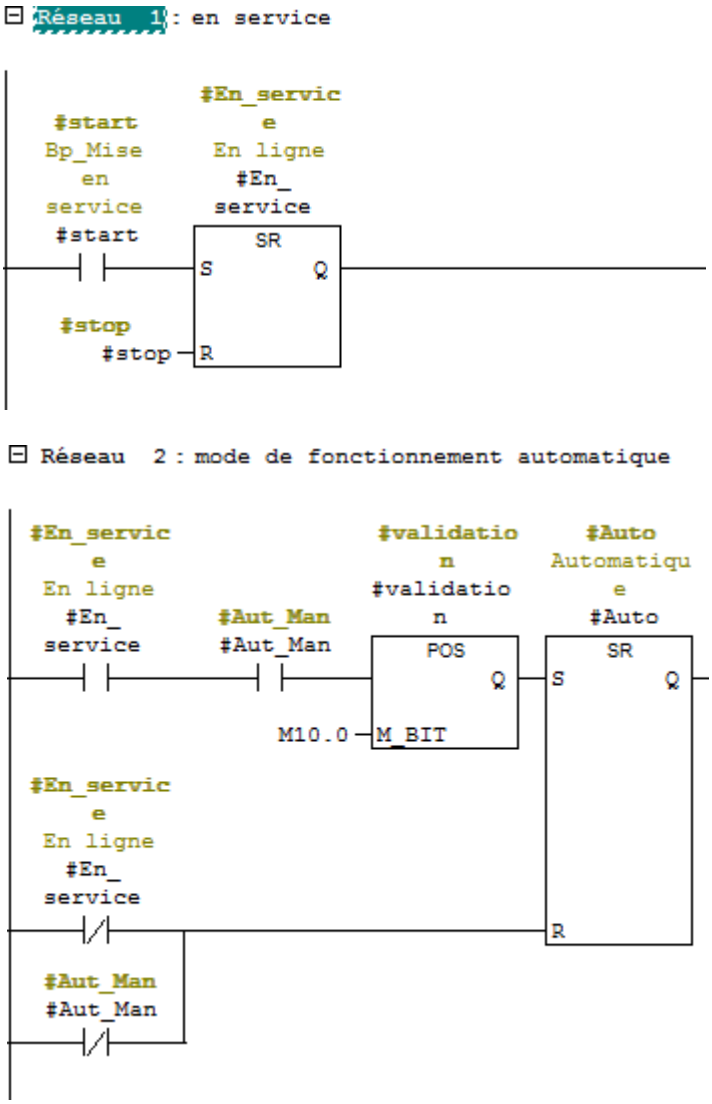
Le FB est un sous-programme écrit par l'utilisateur et exécuté par des blocs de code, On lui associe un bloc de données d'instance DB relatif à sa mémoire et contenant ses paramètres. Pour ce programme on a utilisé deux blocs de ce type(FB1 Mode Automatique, FB2 Mode Manuelle), programmé en langage GRAPH.

12.6.3 Blocs de données (DB) :

Ces blocs de données servent uniquement à stocker des informations et des données mais pas d'instructions, ces données seront utilisées par d'autres blocs.

12.6.4 Bloc fonction (FC) :

La FC contient des routines pour les fonctions fréquemment utilisées. Elle est sans mémoire et sauvegarde ses variables temporaires dans la pile de données locales. Cependant elle peut faire appel à des blocs de données globaux pour la sauvegarde de ses données



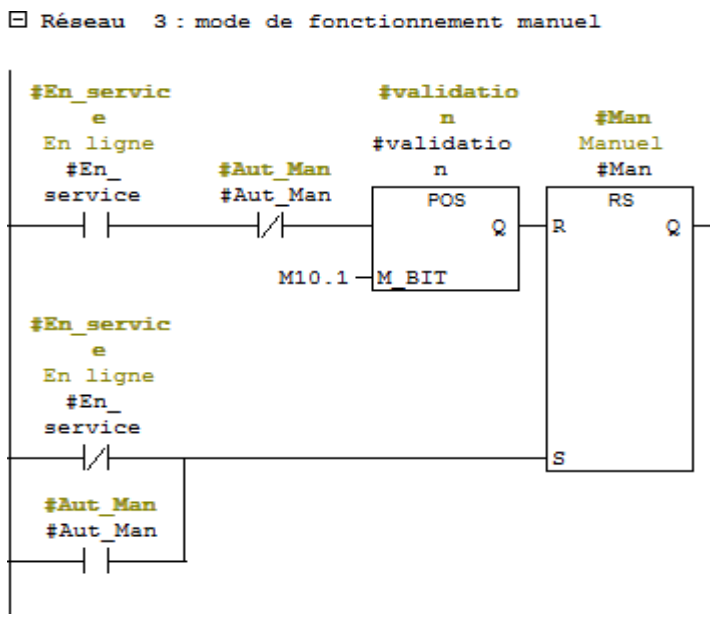


Figure 2. 16 : sélection sur le mode de fonctionnement

12.7 Simulation :

En l'absence de l'automate et des moyens est nous n'avons pas pu réaliser l'armoire électrique et tester réellement l'exécution du programme.

Nous avons utilisé un logiciel optionnel de STEP7, ce logiciel nommé PLCSIM (**Figure 2.17)** permet de simuler un automate de la famille SIEMENS avec tous ces modules. Le simulateur présente une interface simple et accessible, en effet pour changer l'état d'une entrée, il suffit de cocher la case correspondante, les états des sorties changent Automatiquement selon l'évolution du programme. Lors de la simulation et dans la fenêtre de programmation(CONT), chaque contact représentant une variable active est affiché en vert (les contacts non actifs en pointillé).Ceci permet de suivre l'évolution du programme en détails. La simulation nous a permis de tester les différentes situations que peut affronter le système.

Nous concluons à la fin que notre programme répond exactement aux exigences du cahier des charges et qu'il peut donc être transféré du PC vers l'automate qui lui correspond.

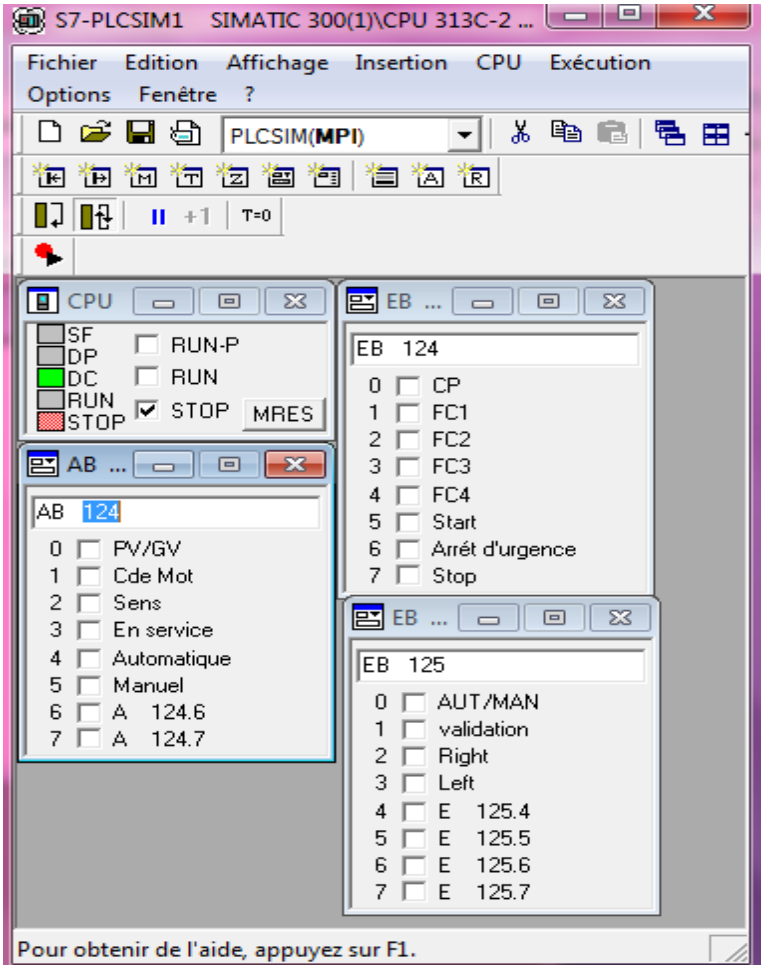
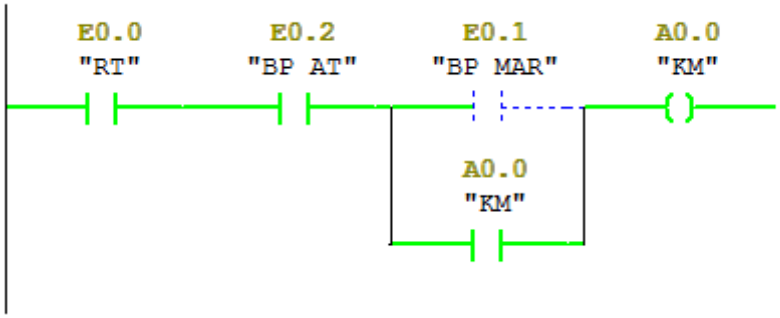
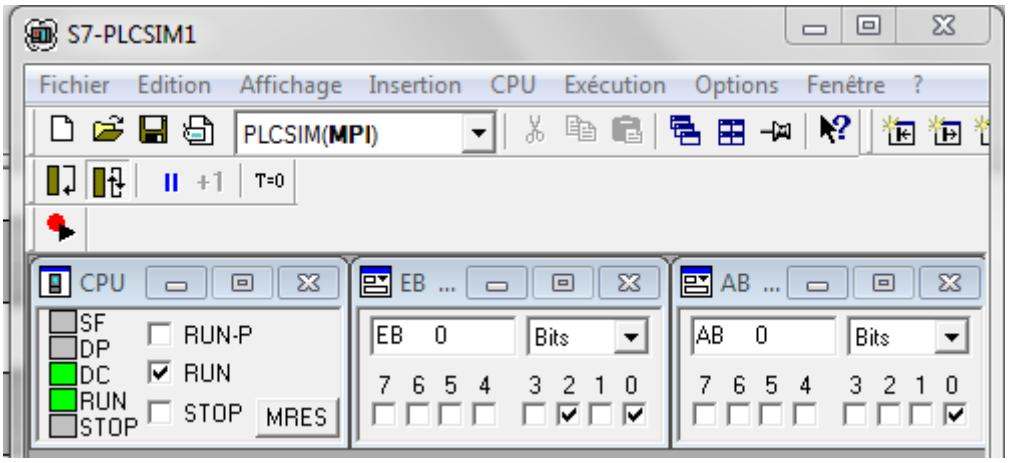


Figure 2. 17 : Simulateur PLCSIM

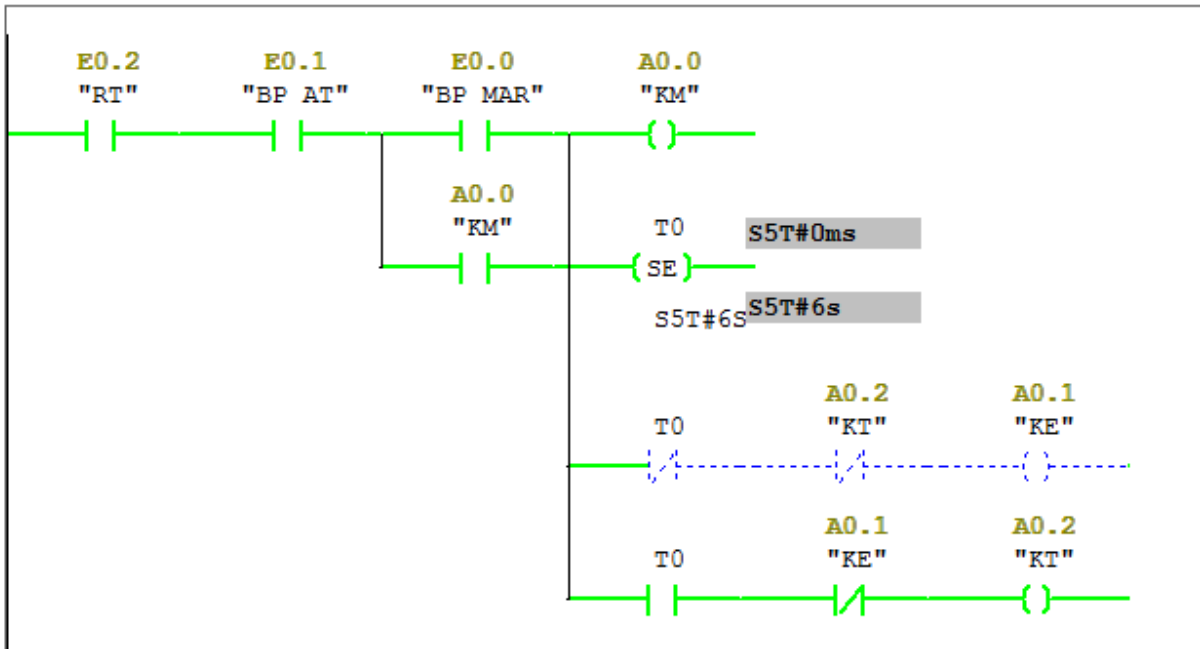
13. Application :

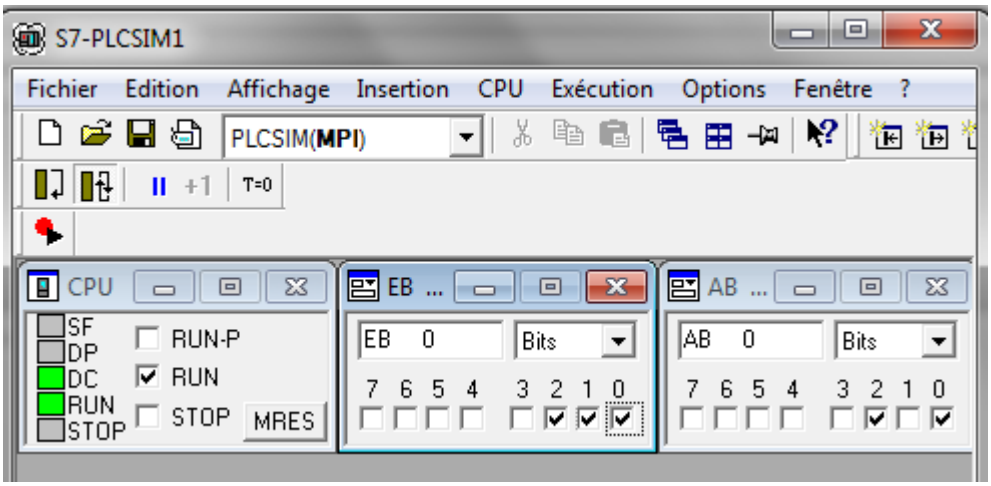
1- Démarrage direct d'un moteur asynchrone triphasé on langage CONT





2- Démarrage étoile triangle d'un moteur asynchrone triphasé on langage CONT





14. Conclusion :

L'automate programmable industriel API est un outil adéquat pour les solutions d'automatisation. C'est l'outil le plus répandu dans les procédés industriel. Le recours au logiciel S7-300 est indispensable pour la simulation du programme et de concepts de commandes automatisées avant leur implantation dans un système réel.

Enfin, nous avons eu l'occasion de présenter le logiciel STEP7 qui nous a permis de programmer le fonctionnement de notre porte automatique, en réalisant d'abord le GRAFCET pour pouvoir simuler notre travail.

CHAPITRE 3

COMMANDE DES API
&
PROGRAMMATION AUTOMGEN 8.9

1. Introduction

Automgen 8.9 est un atelier d'automatisme, de supervision et de simulation de parties opératives 2 D et 3D fonctionnant sur PC sous Windows . il est utilisé dan l'enseignement à partir des classes de secondes pour l'apprentissage des automatismes .et dans l'industrie pour les développement d'applications. l'exceptionnelle convivialité de l'environnement de développement permet aux automaticiens de se consacrer à l'essentiel de leur métier

2. Equipement au laboratoire [12]

2.1. Généralités

L'équipement du laboratoire permet de simuler le fonctionnement d'automatismes industriels et de se familiariser avec la manipulation des différentes variables et des différents procédés reliés aux problèmes d'automatisation industrielle en mode TOUT ou RIEN.

La programmation est facilitée par la saisie et la visualisation graphique des programmes (GRAFCET ou schémas à contacts) sur l'ordinateur du laboratoire. Les corrections et les mises au point sont également facilitées par la visualisation en temps réel des états des entrées/sorties de l'automate.

Le présent document se veut un outil dont la progression dirige l'utilisateur pour lui assurer une compréhension rapide en facilitant les différentes étapes de programmation. Le logiciel offre plusieurs possibilités dont l'utilisation n'est pas nécessaire dans le cadre de la simulation en laboratoire. Des manuels de référence sont disponibles sur l'ordinateur du laboratoire et sur Internet pour toutes consultations supplémentaires.

2.2. Description du Automgne de Télémécanique

L'automate programmable Automgne est essentiellement modulaire, ce qui lui donne une bonne flexibilité. L'automate programmable du laboratoire possède deux modules d'entrées (total de 32 entrées Tout ou Rien) et trois modules de sorties (total de 24 sorties Tout ou Rien)

Configuration des modules :

L'adressage d'une entrée ou d'une sortie est défini par :

1- **I** (input) pour les entrées ou **O** (output) pour les sorties;
2- Son emplacement dans les bacs: 0 ou 1 pour les entrées, 3,6 ou 7 pour les sorties;
3- Une virgule;
4- Le numéro de la voie
   - 0 à 9 ou A à F pour les deux modules d'entrées;
   - 0 à 7 pour les modules de sorties.

Exemple: I0,F :16^{e} voie du module d'entrée situé dans l'emplacement 0 (premier module d'entrée).

O7,3 : 3e voie du module de sortie situé dans l'emplacement 7 (dernie module de sortie).

2.3. Description du panneau de simulation

1- **Automate programmable :**
Différents témoins lumineux permettent de voir l'état général de l'automate (POWER → tension sur l'automate, RUN → programme en marche, MEM → chargement d'un programme dans la mémoire de l'automate). Des témoins lumineux sur chaque module permettent également de voir l'état des entrées/sorties de l'automate.

2- **Écran, clavier et ordinateur :**
Interface de l'usager. Contient, sur le disque dur, le logiciel de programmation AUTOMGEN 7.0 permettant de communiquer avec l'automate. Il permet la sauvegarde des fichiers de programmation sur disquette 3 ½".

3- **Console de commande:**
Cette console comprend : 1 bouton poussoir de démarrage (MA), 1 sélecteur manuel/automatique, 1 arrêt d'urgence (Urg), 4 sélecteurs manuels (numéros 1 à 4) et 10 boutons poussoirs (numéros 5 à 14). Ils permettent de simuler des éléments d'automatismes (encombrement, capteurs, dés alarme, etc.) n'utilisant pas des vérins ou des moteurs hydrauliques et/ou pneumatique.

4- **Témoins lumineux:**
Six (6) témoins lumineux permettent de simuler des éléments d'automatismes (alarmes, capteurs actifs ou inactifs, etc.) n'utilisant pas des vérins ou des moteurs hydrauliques et/ou pneumatique.

5- **Zone de simulation :**
Cette zone comprend : cinq (5) vérins pneumatiques (de A à E) , un moteur (illustration seulement) et les représentations graphiques des six distributeurs correspondants. Des interrupteurs de fin de course, ainsi que des valves de blocage sont associées aux vérins sur le panneau. Des témoins lumineux sont installés sur les symboles des distributeurs pour permettre à l'utilisateur de bien suivre le déroulement d'une séquence de simulation. Les vérins peuvent être forcés manuellement grâce à des boutons poussoirs situés près des différents distributeurs. Pour le moteur, d'autres témoins lumineux permettent de simuler la rotation à gauche ou la rotation à droite du moteur.

La mise sous tension du système se fait avec la barre d'alimentation située à l'intérieur du panneau, en bas à gauche. L'automate et les différentes composantes devraient se mettre automatiquement en marche. Si un problème survient, contacter le technicien du laboratoire.

2.4 Liste des mnémoniques

Le branchement des différentes entrées/sorties est fixe et les tableaux suivants résument les diverses composantes du panneau associées aux variables du logiciel (colonne

Automgen) et aux adresses de l'automate programmable (colonne TSX). Les mnémoniques sont des aide-mémoire qui sont utilisés lors de la programmation. Les symboles des mnémoniques peuvent être modifier pour s'adapter à votre application. Pour savoir de quelle façon on procède, veuillez consulter la section 2.2 Le navigateur.

Tableau 3.1 : Listes des mnémoniques des sorties

LISTE DES SORTIES			
VARIABLES		MNÉMONIQUES	COMMENTAIRES
AUTOMGEN	TSX	Symboles	
o0	O3,0	Ablo	Bloquer vérin A
o1	O3,1	Bblo	Bloquer vérin B
o2	O3,2	Cblo	Bloquer vérin C
o3	O3,3	Dblo	Bloquer vérin D
o4	O3,4	Eblo	Bloquer vérin E
o5	O6,0	Aout	Sortir vérin A
o6	O6,1	Bin	Rentrer vérin B
o7	O6,2	Bout	Sortir vérin B
o8	O6,3	Cin	Rentrer vérin C
o9	O6,4	Cout	Sortir vérin C
o10	O6,5	Dout	Sortir vérin D
o11	O6,6	Ein	Rentrer vérin E
o12	O6,7	Eout	Sortir vérin E
o13	O7,0	D1	Témoin lumineux 1
o14	O7,1	D2	Témoin lumineux 2
o15	O7,2	D3	Témoin lumineux 3
o16	O7,3	D4	Témoin lumineux 4
o17	O7,4	D5	Témoin lumineux 5
o18	O7,5	D6	Témoin lumineux 6
o19	O7,6	RMD	Rotation Moteur Droite
o20	O7,7	RMG	Rotation Moteur Gauche

Tableau 3.2 : Listes des variables booléennes et numériques

LISTE DES VARIABLES BOOLÉENNES ET NUMÉRIQUES			
VARIABLES		MNÉMONIQUES	COMMENTAIRES
AUTOMGEN	TSX	Symboles	
U101 à U130	B101 à B130	REL01 à REL30	Relais pour ladder (bool.)
U200 à U220	B200 à B220	MEM00 à MEM20	Mémoires internes (bool.)
C0 à C15	C0 à C15	C0 à C15	Compteurs (num.)
T0 à T15	C0 à C15	T0 à T15	Temporisations (num. + bool.)

Tableau 3.3 : Listes des mnémoniques des entrées

LISTE DES ENTRÉES			
VARIABLES		MNÉMONIQUES	COMMENTAIRES
AUTOMGEN	TSX	Symboles	
i0	I0,0	URG	Urgence
i1	I0,1	S1	Interrupteur no.1
i2	I0,2	S2	Interrupteur no.2
i3	I0,3	S3	Interrupteur no.3
i4	I0,4	S4	Interrupteur no.4
i5	I0,5	S5	Bouton no.5
i6	I0,6	S6	Bouton no.6
i7	I0,7	S7	Bouton no.7
i8	I0,8	S8	Bouton no.8
i9	I0,9	S9	Bouton no.9
i10	I0,A	S10	Bouton no.10
i11	I0,B	S11	Bouton no.11
i12	I0,C	S12	Bouton no.12
i13	I0,D	S13	Bouton no.13
i14	I0,E	S14	Bouton no.14
i15	I0,F	AUT	Automatique
i16	I1,0	MA	Démarrage
i17	I1,1	da0	Vérin A entrée
i18	I1,2	da1	Vérin A sortie
i19	I1,3	da2	Vérin A mi-course
i20	I1,4	db0	Vérin B entrée
i21	I1,5	db1	Vérin B sortie
i22	I1,6	db2	Vérin B mi-course
i23	I1,7	dc0	Vérin C entrée
i24	I1,8	dc1	Vérin C sortie
i25	I1,9	dc2	Vérin C mi-course
i26	I1,A	dd0	Vérin D entrée
i27	I1,B	dd1	Vérin D sortie
i28	I1,C	dd2	Vérin D mi-course
i29	I1,D	de0	Vérin E entrée
i30	I1,E	de1	Vérin E sortie
i31	I1,F	de2	Vérin E mi-course

3. Programmation avec automgen 8.9

Un fichier nommé *Configuration.agn* est mis à votre disposition sur le disque dur. Il contient toutes les mnémoniques ainsi que les différents paramètres nécessaires à la communication avec l'automate programmable.

Il est aussi possible de télécharger une version du logiciel et les manuels de références (en format pdf) via le site Internet www.irai.com. Cette version est valide pour une durée de 40 jours seulement. De plus, une seule installation par poste est possible, c'est-à-dire que vous ne pouvez plus réinstaller le logiciel sur le même ordinateur une fois les 40 jours expirés.

3.1 L'environnement

Lorsque vous ouvrez le fichier *Configuration.agn*, la fenêtre du projet s'ouvre. Le projet permet de regrouper l'ensemble des éléments d'une application d'AUTOMGEN. Ainsi regroupés, les éléments ne forment plus qu'un fichier portant l'extension « .agn ». On y retrouve les barres d'outils en haut, le navigateur à gauche, l'espace de travail à droite et les fenêtres de messages en bas.

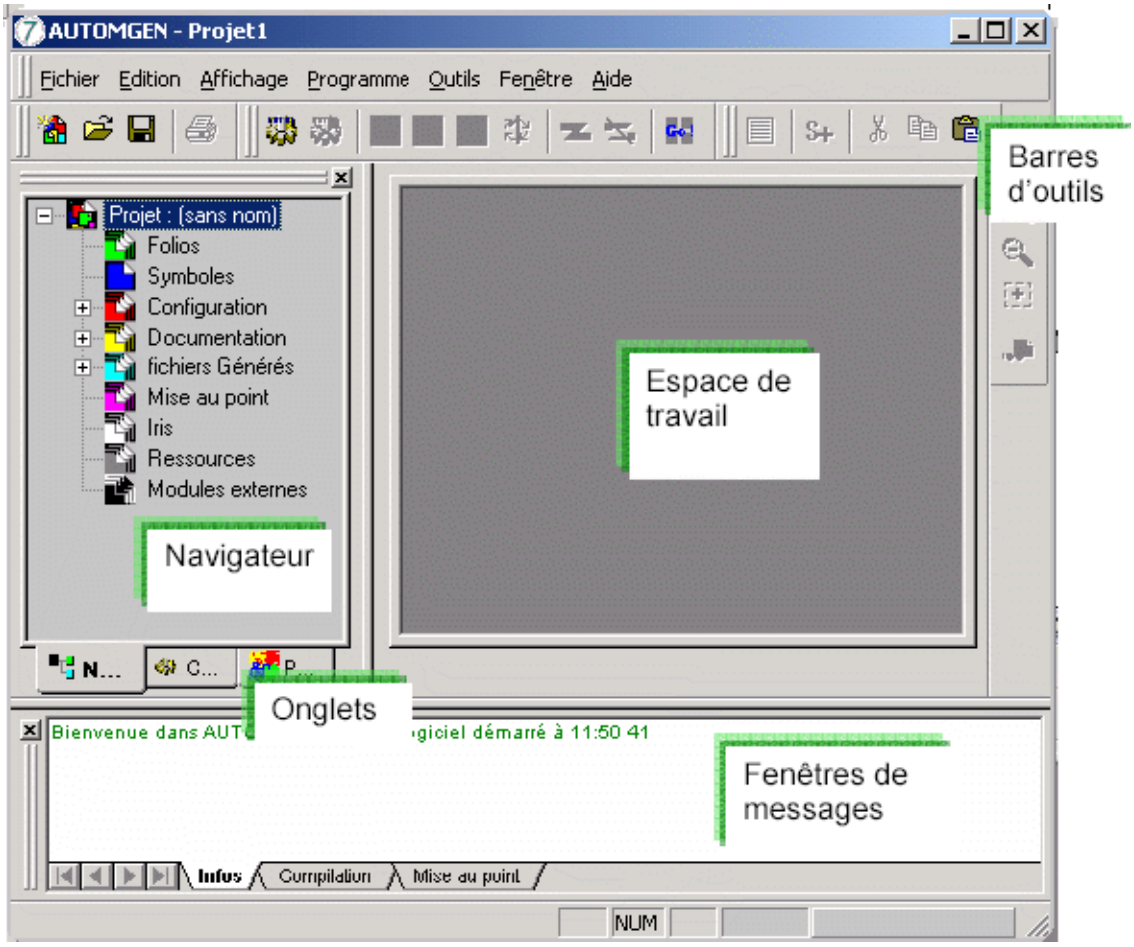


Figure 3.1 : Fenêtre principale d'AUTOMGEN

En bas de la fenêtre du navigateur se trouvent deux autres onglets.

Figure 3.2 : Onglets

L'onglet nommé « Cibles » permet d'accéder à la liste des post-processeurs installés. On en retrouve deux au laboratoire : l'exécuteur PC qui permet d'exécuter le programme sur l'ordinateur et le PL72 qui permet de télécharger le programme dans l'automate programmable. La cible active est marquée d'un crochet vert. Pour activer ou changer de cible, double cliquez sur la cible voulue.

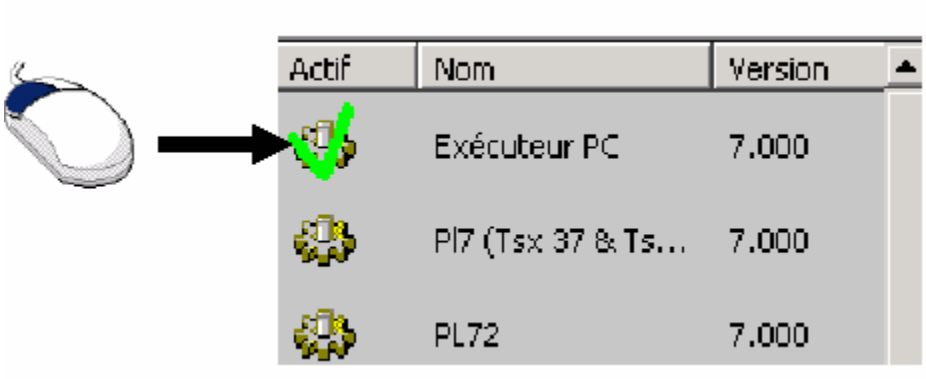


Figure 2. 3: Choix du post-processeur

L'onglet « Palette », pour sa part, permet d'accéder aux éléments de dessin de programmes. On retrouve son contenu plus en détail à la section « 2.3 Saisie de GRAFCET et de schéma à contacts ».

Il est possible d'afficher l'espace de travail en mode plein écran. Pour ce faire, sélectionnez l'option « Plein écran » dans le menu « Affichage ». Pour sortir du mode plein écran, cliquez sur l'icône « Close FullScreen »  située en haut à gauche.

3.2 Le navigateur

Élément central de la gestion des applications, le navigateur permet un accès rapide aux différents éléments d'une application : folios, symboles, etc. Les icônes « + » et « - » permettent de développer ou de rétracter les éléments du projet. Les actions sur le navigateur sont réalisées en double cliquant sur les éléments ou en cliquant avec le bouton droit pour modifier les propriétés. Les deux principaux éléments que vous aurez à utiliser sont les folios et les symboles.

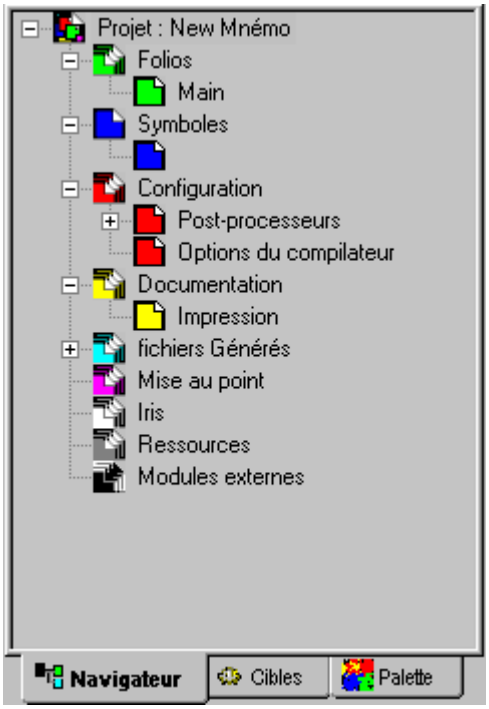


Figure 3. 4 : Le navigateur

Un folio est une page nommée « Main » sur laquelle est conçu un programme ou une partie de programme. Cette page est verte et se situe dans l'espace de travail. On peut y accéder en double cliquant sur « Main » dans le navigateur ou en cliquant sur l'onglet portant le même nom au bas de l'espace de travail.

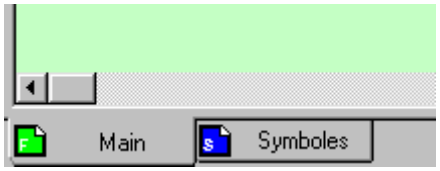
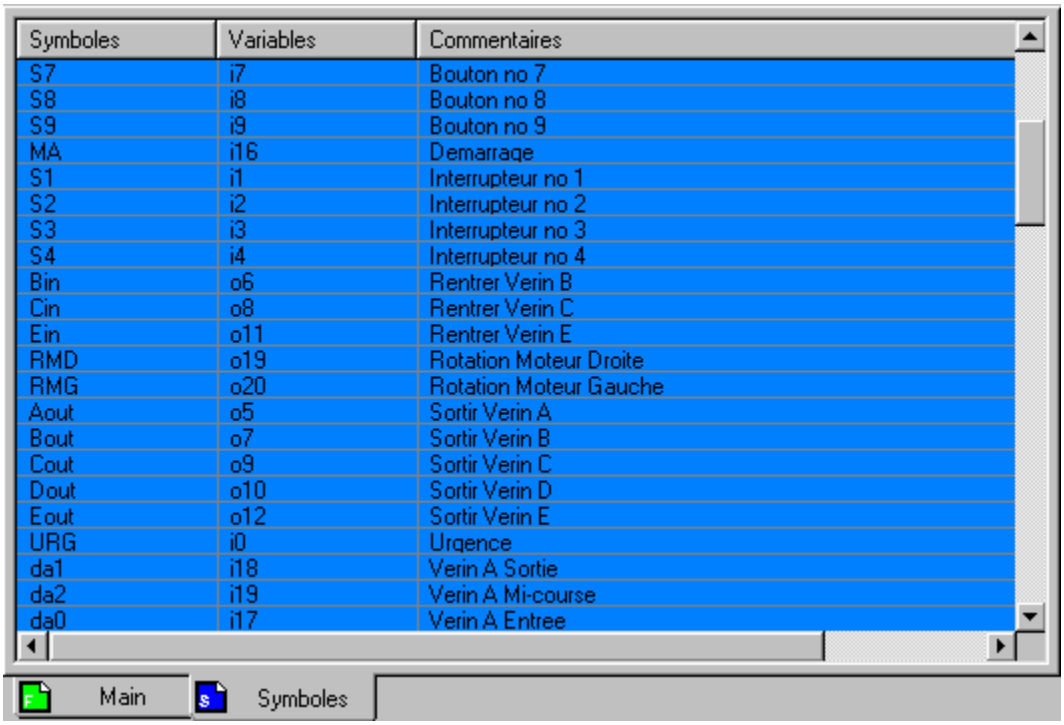


Figure 2. 5 : Accès à la page de travail (en vert)

En double cliquant sur l'icône « Symboles » dans le navigateur, la liste des mnémoniques apparaît en bleu dans l'espace de travail. On peut aussi y accéder en cliquant sur l'onglet portant le même nom au bas de l'espace de travail. Un projet ne peut contenir qu'une seule table de symboles. Si vous avez besoins de mnémoniques supplémentaires pour votre projet, veuillez contacter le technicien.



Symboles	Variables	Commentaires
S7	i7	Bouton no 7
S8	i8	Bouton no 8
S9	i9	Bouton no 9
MA	i16	Demarrage
S1	i1	Interrupteur no 1
S2	i2	Interrupteur no 2
S3	i3	Interrupteur no 3
S4	i4	Interrupteur no 4
Bin	o6	Rentrer Verin B
Cin	o8	Rentrer Verin C
Ein	o11	Rentrer Verin E
RMD	o19	Rotation Moteur Droite
RMG	o20	Rotation Moteur Gauche
Aout	o5	Sortir Verin A
Bout	o7	Sortir Verin B
Cout	o9	Sortir Verin C
Dout	o10	Sortir Verin D
Eout	o12	Sortir Verin E
URG	i0	Urgence
da1	i18	Verin A Sortie
da2	i19	Verin A Mi-course
da0	i17	Verin A Entree

Main Symboles

Figure 3. 6 : Fenêtre de la table de symboles

C'est dans cette fenêtre que vous pouvez modifier les symboles des mnémoniques pour les adapter à votre application. Par exemple, on pourrait faire correspondre l'interrupteur n°1 (S1) à la simulation de la présence de pièces dans le magasin. Il suffit de double cliquer sur la ligne du symbole S1. La fenêtre de la figure 8 apparaîtra. Il reste seulement qu'à donner un nouveau nom au symbole comme « pm » et cliquer sur « OK ». Surtout, ne changez pas la variable associée si vous voulez qu'AUTOMGEN reconnaisse votre symbole et qu'il adresse celui-ci à la bonne place dans l'automate programmable.

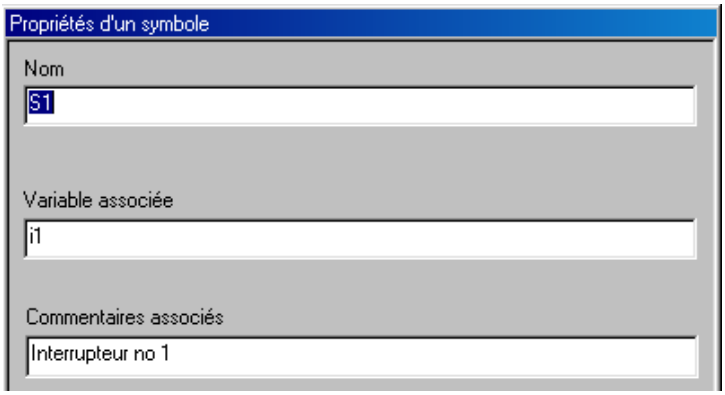


Figure 3. 7 : Changer le nom d'un symbole

3.3 Saisie de GRAFCET et de schémas à contacts

3.3.1 Généralités

Pour saisir des GRAFCET et des schémas à contacts, plusieurs outils sont à votre disposition : l'assistant, le menu contextuel, la palette et les touches du clavier.

Concevoir un programme avec l'assistant est sans doute la façon la plus simple lorsqu'on débute avec AUTOMGEN. Cliquez avec le bouton droit de la souris sur un folio ouvert dans l'espace de travail et choisissez « Assistant » dans le menu.

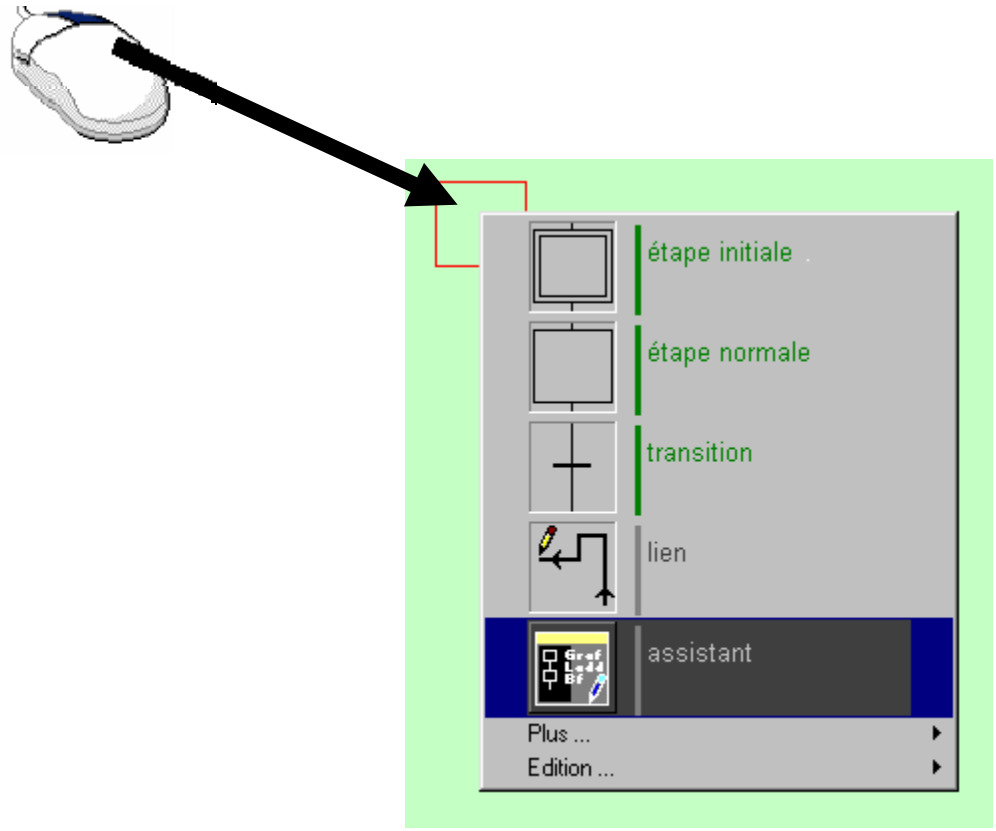


Figure 3. 8 : Accès au menu Assistant

Laissez-vous ensuite guider dans les choix. Lorsque vous avez fini, cliquez sur « OK ».

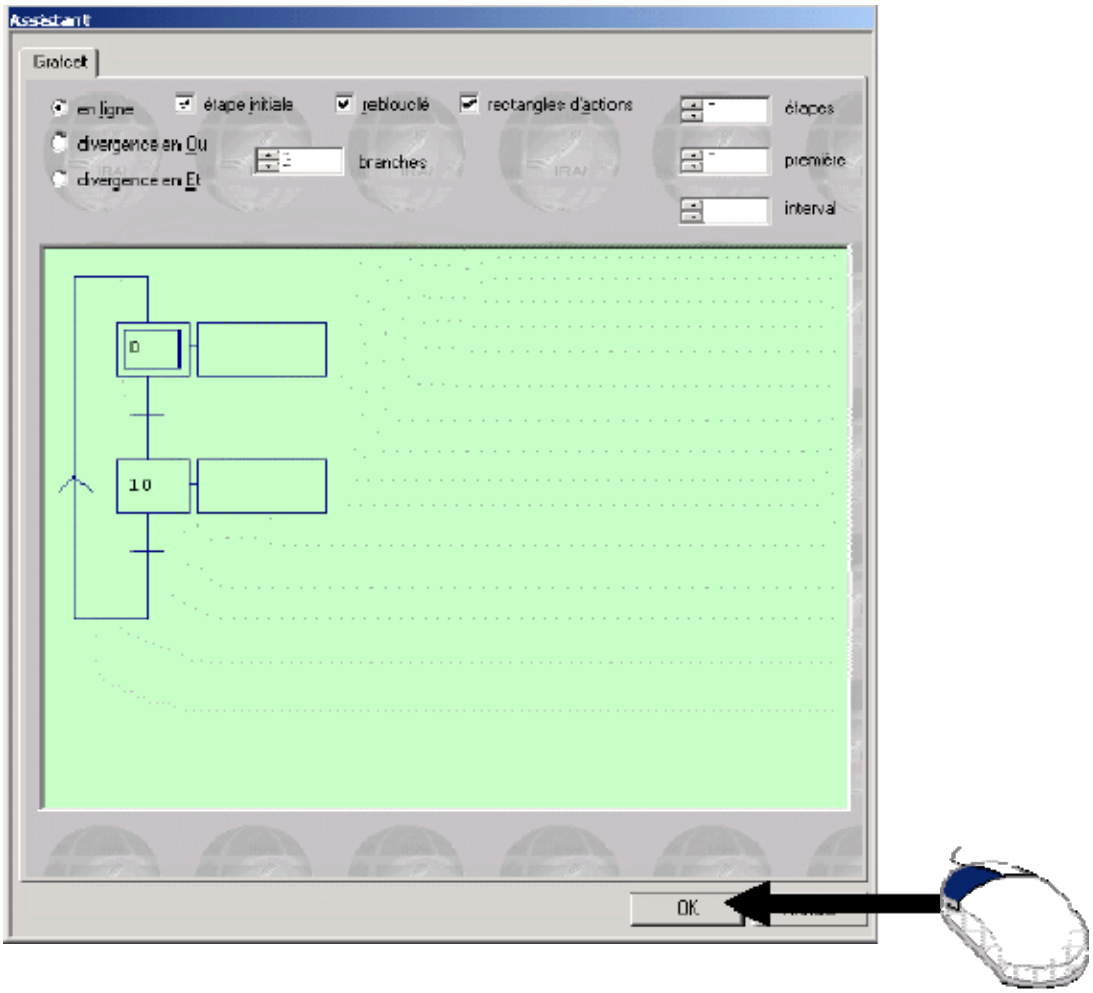


Figure 3.9 : Menu de l'Assistant

Ensuite, poser le GRAFCET sur le folio en cliquant avec le bouton gauche de la souris.

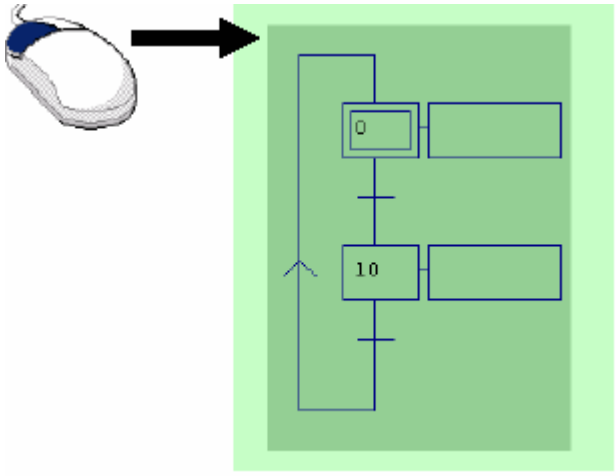


Figure 3.10 : Déposer le GRAFCET sur le folio

Une autre option est d'utiliser le menu contextuel. En cliquant avec le bouton droit de la souris sur un folio ouvert dans l'espace de travail, le menu vous propose une série d'éléments que vous pouvez poser sur le folio. C'est un mode de création instinctif et rapide. Voir la figure 9 pour un exemple des choix disponibles.
En bas de la fenêtre du navigateur se trouve l'onglet « Palette »permettant d'accéder aux éléments de dessin de programmes. En sélectionnant des éléments dans la palette, vous pouvez créer rapidement des programmes à partir d'éléments déjà créés. Pour ce faire, cliquez avec le bouton de gauche une fois sur l'élément de la palette que vous désirez pour que le fond devienne vert foncé. Ensuite, cliquez encore une fois sur l'élément en gardant le bouton de gauche enfoncé. Vous n'avez plus qu'à déposer l'élément sur l'espace de travail à l'endroit voulu en relâchant le bouton de gauche.

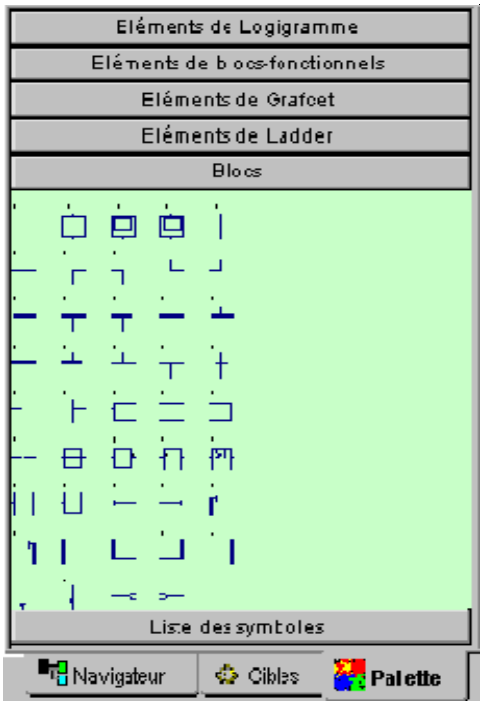


Figure 3. 11 : Éléments généraux

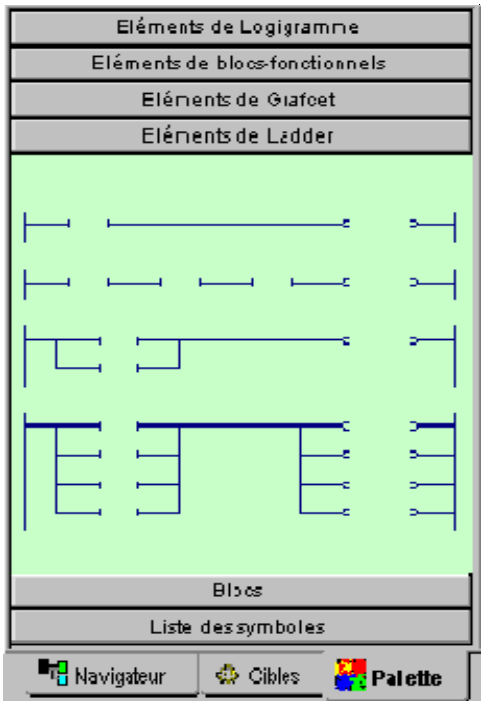


Figure 3. 12 : Éléments de ladder

À chaque élément de la palette correspond une touche du clavier (figure 12). Les associations sont écrites en petits près des éléments de la palette. Cela constitue une autre façon rapide de concevoir un programme. Le tableau 4 de la page suivante donne la liste des éléments, les touches associées et leur utilisation.

Tableau 3.4 : Liste des éléments, des touches associées et leur utilisation

Bloc d'effacement

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[A]	Effacement	Permet de remettre à blanc une cellule	Tous

Blocs de liaison

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[E]	Liaison verticale	Liaison haut vers bas ou bas vers haut	Tous
	[F]	Liaison horizontale	Liaison droite vers gauche ou gauche vers droite	Tous
	[G]	Coin supérieur gauche	Liaison bas vers droite ou gauche vers bas	Tous
	[H]	Coin supérieur droit	Liaison bas vers gauche ou droite vers bas	Tous
	[I]	Coin inférieur gauche	Liaison haut vers droite ou gauche vers haut	Tous
	[J]	Coin inférieur droit	Liaison haut vers gauche ou droite vers haut	Tous
	[Z]	Croisement	Croisement de deux liaisons	Tous

Blocs Grafcet

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[B]	Etape	Etape normale	Grafcet
	[C]	Etape initiale sans activation	Etape initiale sans activation	Grafcet
	[D]	Etape initiale	Etape initiale	Grafcet
		Macro-étape	Accessible dans le menu contextuel uniquement	Grafcet
	[T]	Transition	Transition	Grafcet
	[K]	Bord gauche d'une divergence en « Et »	Obligatoirement à gauche des divergences en « Et »	Grafcet
	[L]	Branche supplémentaire d'une divergence en « Et » ou convergence en « Et »	Ne pas utiliser comme bord gauche ou droit d'une divergence en « Et »	Grafcet

	[M]	Bord droit d'une divergence en « Et »	Obligatoirement à droite d'une divergence en « Et »	Grafcet
	[N]	Prolongation d'une divergence en « Et »	Se place entre les blocs [K], [L], [M], [P] ou [O],[P],[Q], [L]	Grafcet
	[O]	Bord gauche d'une convergence en « Et »	Obligatoirement à gauche d'une convergence en « Et »	Grafcet
	[P]	Branche supplémentaire d'une convergence en « Et » ou divergence en « Et »	Ne pas utiliser comme bord gauche ou droit d'une convergence en « Et »	Grafcet
	[Q]	Bord droit d'une convergence en « Et »	Obligatoirement à droite d'une convergence en « Et »	Grafcet
	[R]	Divergence en « Ou »	Ne pas utiliser comme bord d'une convergence en « Ou »	Grafcet
	[S]	Convergence en « Ou »	Ne pas utiliser comme bord d'une divergence en « Ou »	Grafcet
	[U]	Saut ou reprise d'étape à gauche	Convergence ou divergence en « Ou »	Grafcet
	[V]	Saut ou reprise d'étape à droite	Convergence ou divergence en « Ou »	Grafcet
	[ESPACE] sur un bloc [E]	Liaison orientée vers le haut	Pour les rebouclages et les reprises d'étapes	Grafcet

Blocs Ladder

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[(]	Partie gauche bobine	Débute une action	Ladder
	[)]	Partie droite bobine	Termine une action	Ladder
	[U]	Bord gauche	Termine le schéma	Ladder
	[V]	Bord droit	Débute le schéma	Ladder
	[R]	Connexion	Fonction « Ou »	Ladder
	[S]	Connexion	Fonction « Ou »	Ladder

Blocs Action

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[W]	Bord gauche rectangle d'action	Débute une action	Grafcet et Logigrammes
	[X]	Milieu d'un rectangle d'action	Prolonge une action	Grafcet et Logigrammes
	[Y]	Bord droit d'un rectangle d'action	Termine une action	Grafcet et Logigrammes
	[S]	Divergence Action	Permet de juxtaposer des rectangles d'action verticalement	Grafcet et Logigrammes
	[V]	Divergence Action	Permet de juxtaposer des rectangles d'action verticalement	Grafcet et Logigrammes

Blocs Test

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[7]	Bord gauche d'un test	Débute un test	Logigrammes et Ladder
	[6]	Bord droit d'un test	Termine un test	Logigrammes et Ladder

*** Prenez note qu'il ne faut jamais utiliser la touche du clavier « Supprimer » pour enlever un élément car vous perdrez tout votre programme ou de grandes parties! Il fautplutôt utiliser la touche « A » associée à l'effacement d'un élément. Si vous avez une grande partie de votre programme que vous voulez effacer et que vous ne voulez pas utiliser la touche « A » pour chacun des éléments (ce qui peut être très long et pénible), vous pouvez toujours « Couper » la zone désirée sans toutefois le recopier!

Vous pouvez aussi introduire des commentaires dans l'espace de travail. Il suffit de cliquer à l'endroit voulu avec le bouton de gauche et un curseur clignotant apparaîtra. Il ne reste plus qu'à taper votre texte.

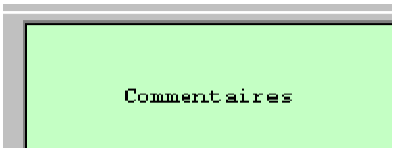


Figure 3.13 : Introduire un commentaire

En tout temps, vous pouvez sélectionner une partie du programme en cliquant sur le bouton de gauche dans l'espace de travail et en le maintenant enfoncé. Un cadre en lignes pointillées délimitera la zone sélectionnée. En relâchant le bouton, le fond de cette zone deviendra vert foncé.

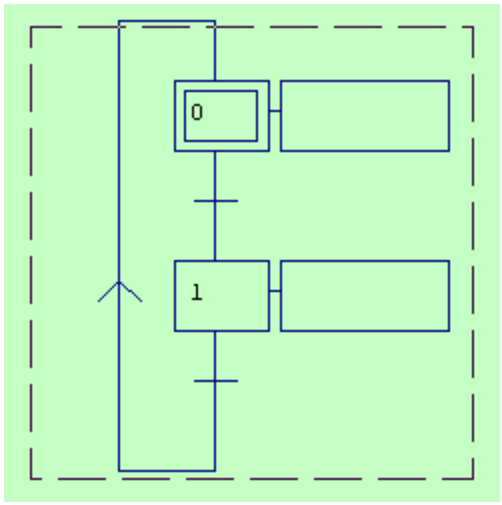


Figure 3.14 : Sélection d'une zone

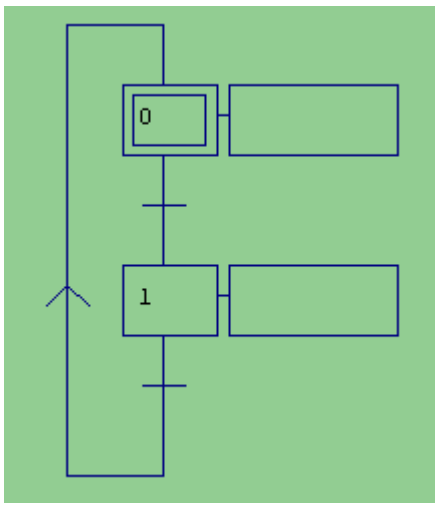


Figure 3.15 : La zone est sélectionnée

Vous pouvez alors déplacer cette zone en cliquant encore une fois dans cette zone et en maintenant le bouton de gauche enfoncé. Vous n'avez plus qu'à placer l'ensemble à l'endroit voulu et à relâcher le bouton. Pour désélectionner la zone et la rendre l'ensemble vert pâle de nouveau, cliquer avec le bouton gauche n'importe où dans l'espace de travail vert pâle.

Vous pouvez aussi Copier/Coller la zone en question en choisissant les commandes appropriées dans le menu « Édition… » après avoir cliqué dans l'espace vert foncé avec le bouton de droite ou en utilisant les raccourcis claviers habituels ou en utilisant les icônes de la barre d'outils.

3.3.2 Les étapes et les actions

3.3.2.1 Pour les GRAFCET

Lors de l'élaboration des étapes de votre GRAFCET, vous pouvez utiliser la divergence et la convergence en « Et » ou en « Ou ». Les divergences peuvent avoir « n » branches. L'important est de respecter l'utilisation des éléments du tableau 4:

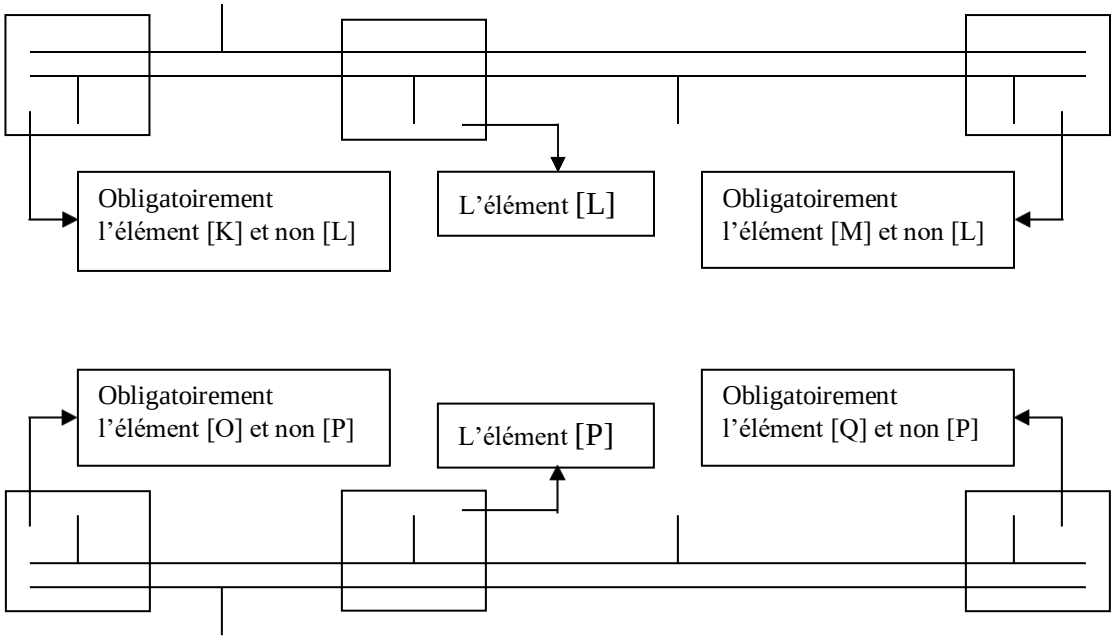


Figure 3.16 : Convention pour la divergence et la convergence en « Et»

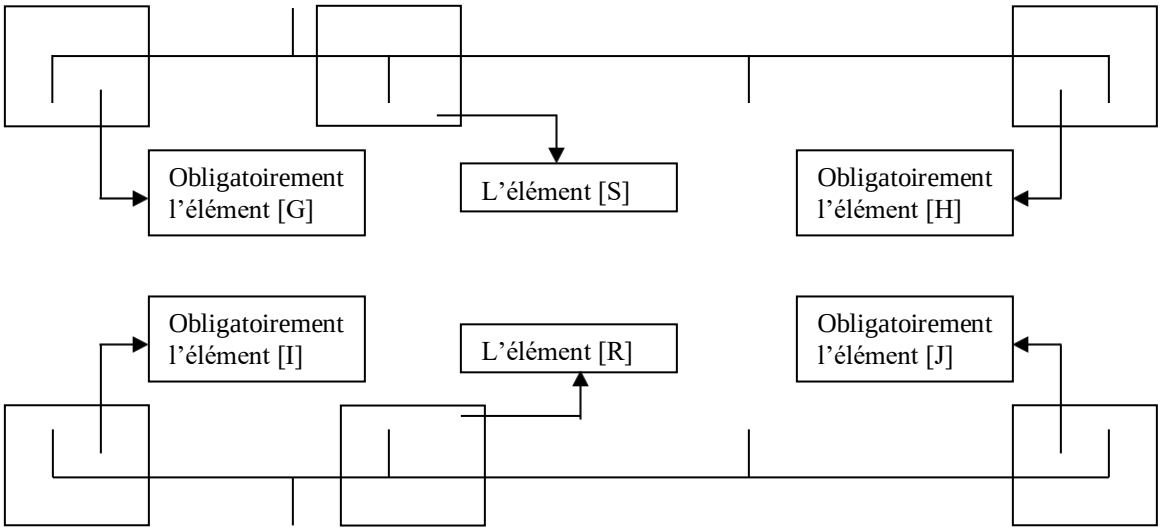


Figure 3.17 : Convention pour la divergence et la convergence en « Ou »

De plus, les divergences en « Ou » doivent obligatoirement se brancher sur des liaisons descendantes. Par exemple :

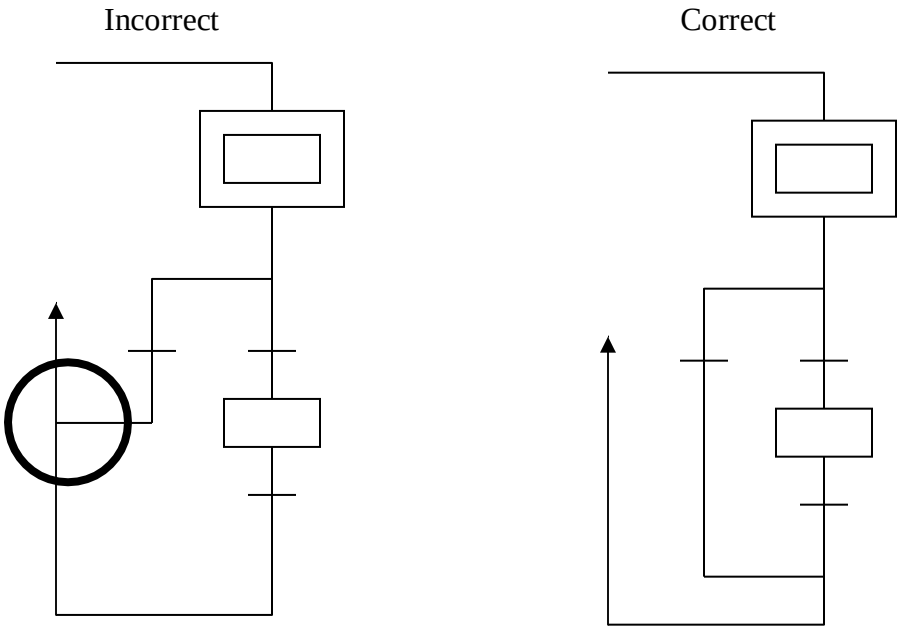


Figure 3.18 : Branchement des divergences en « OU »

On remarque que la liaison de retour est orientée vers le haut. Cela s'avère utile pour s'y retrouver lorsque le GRAFCET devient gros et chargé. Pour identifier une liaison orientée vers le haut, il suffit de cliquer avec le bouton droit sur la partie de la liaison qu'on veut identifier et à choisir l'option correspondante.

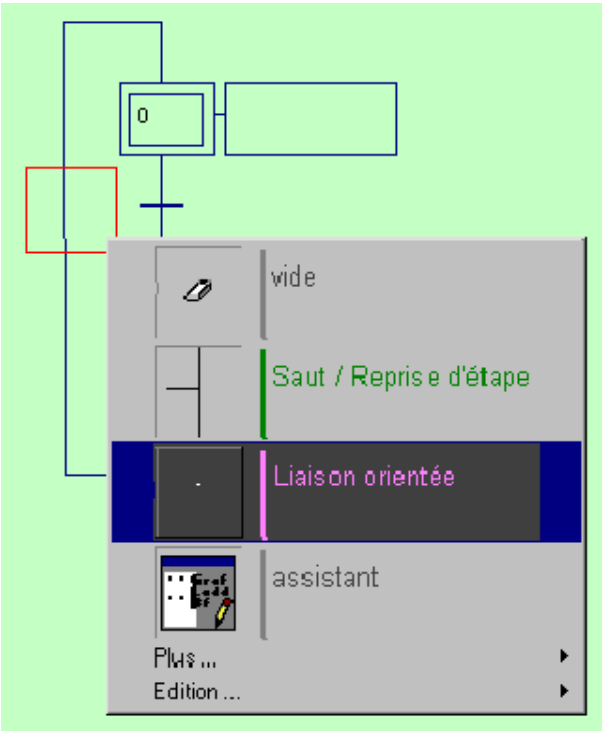


Figure 3.19 : Liaison orientée vers le haut

Les actions sont utilisées dans les rectangles d'action situés à la droite des rectangles d'étapes :

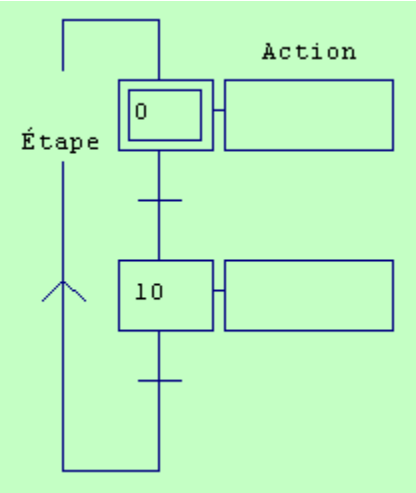


Figure 3.20 : Étapes et actions

1. Pour pouvoir écrire dans ces rectangles, il suffit de cliquer avec le bouton de gauche dans le rectangle en question.:

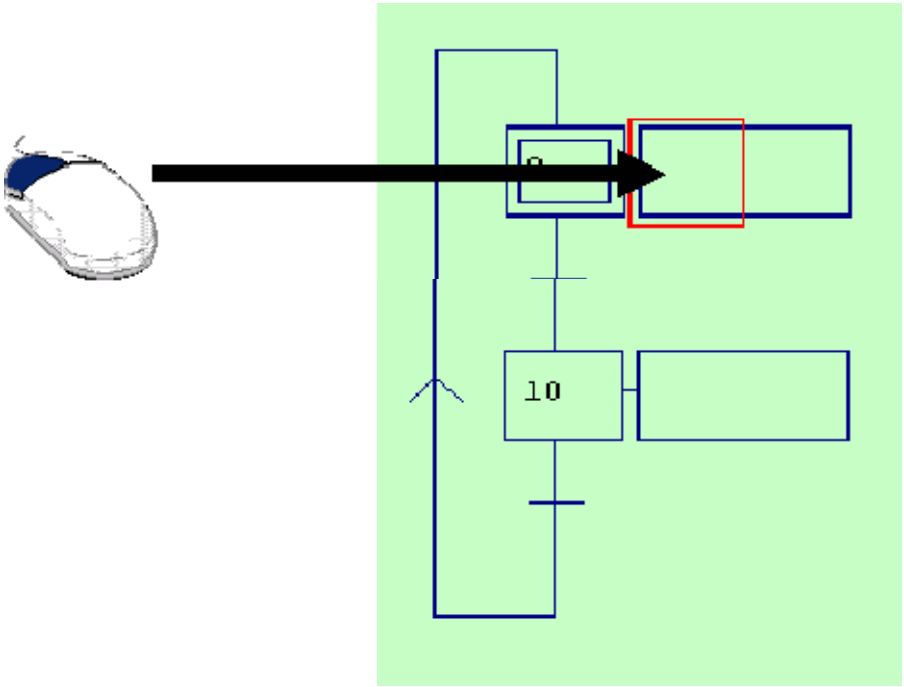


Figure 3.21 : Accès à l'écriture d'une action

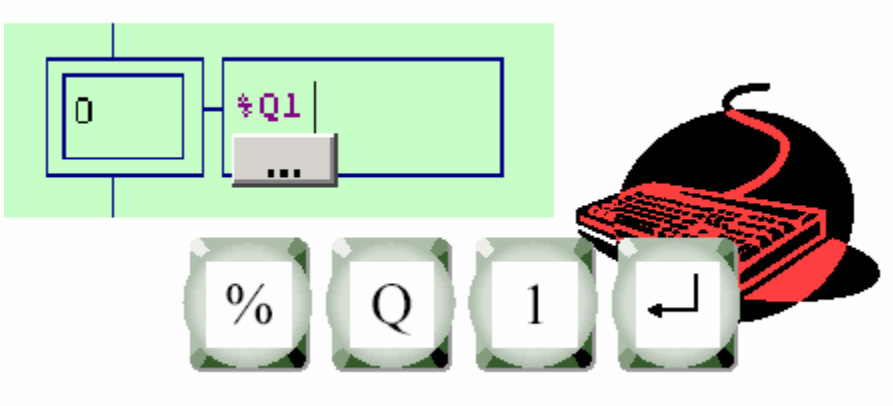


Figure 3.22 : Écriture d'une action

2. Pour les étapes, le curseur clignotant d'écriture apparaîtra et on écrit alors le numéro de l'étape. Dans le cas des actions, le curseur clignotant d'écriture apparaîtra aussi, vous permettant d'écrire le nom des variables vous-même:

3. En plus, l'option « Édition d'une action » représentée par les trois petits points dans le rectangle gris est disponible. Lorsque vous cliquez sur ces trois petits points, une nouvelle fenêtre apparaît:

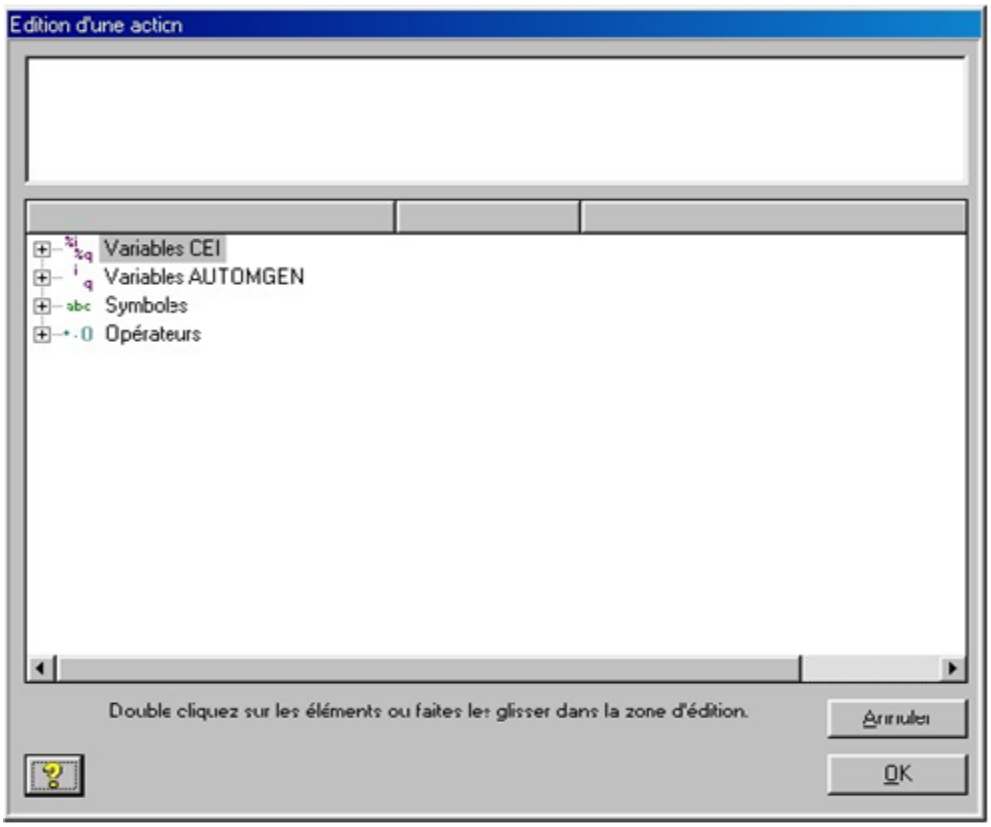


Figure 3.23 : Édition d'une action

4. Vous double cliquez alors sur « Symboles » et la liste des variables ainsi que leurs noms seront disponibles:

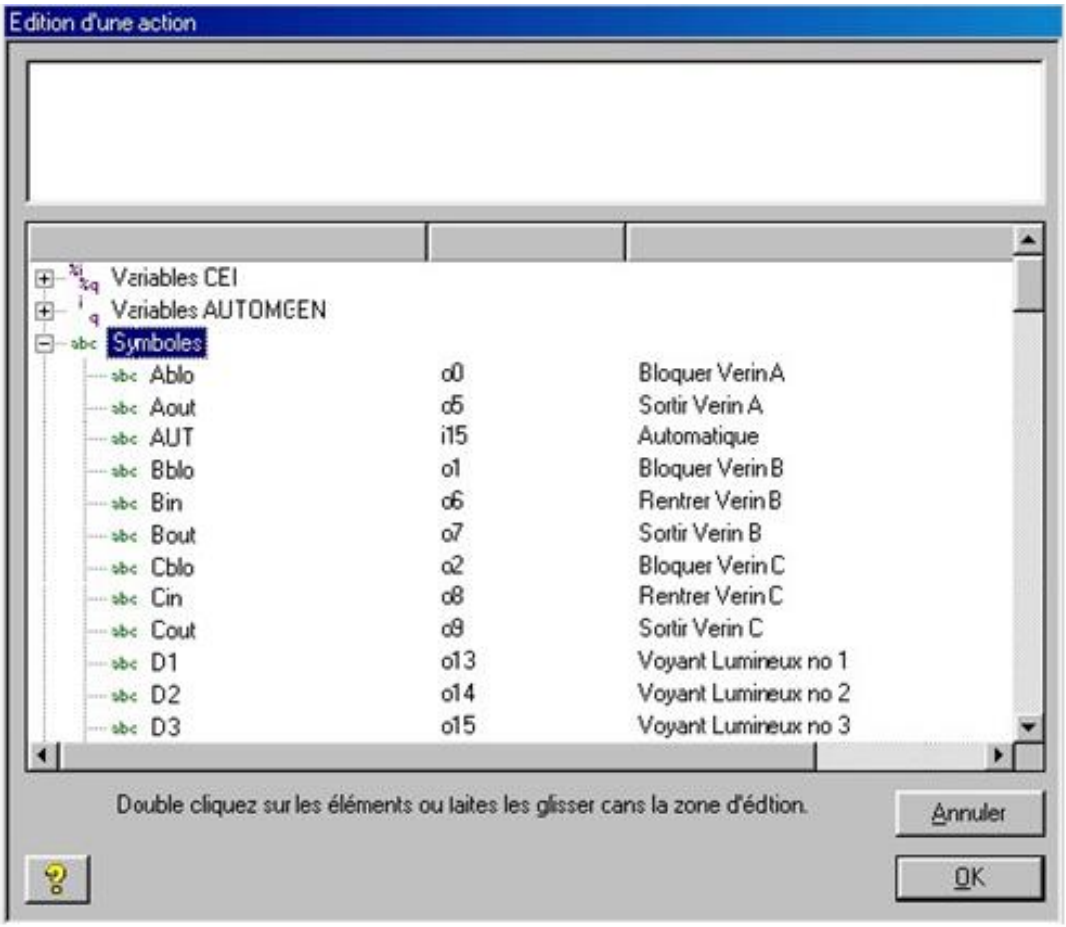


Figure 3.24 : Choix des symboles

5. Il suffit de double cliquer sur la variable voulue et elle sera inscrite dans la partie du haut de la fenêtre d'édition. Vous n'avez plus qu'à cliquer sur « OK » pour valider vos actions.

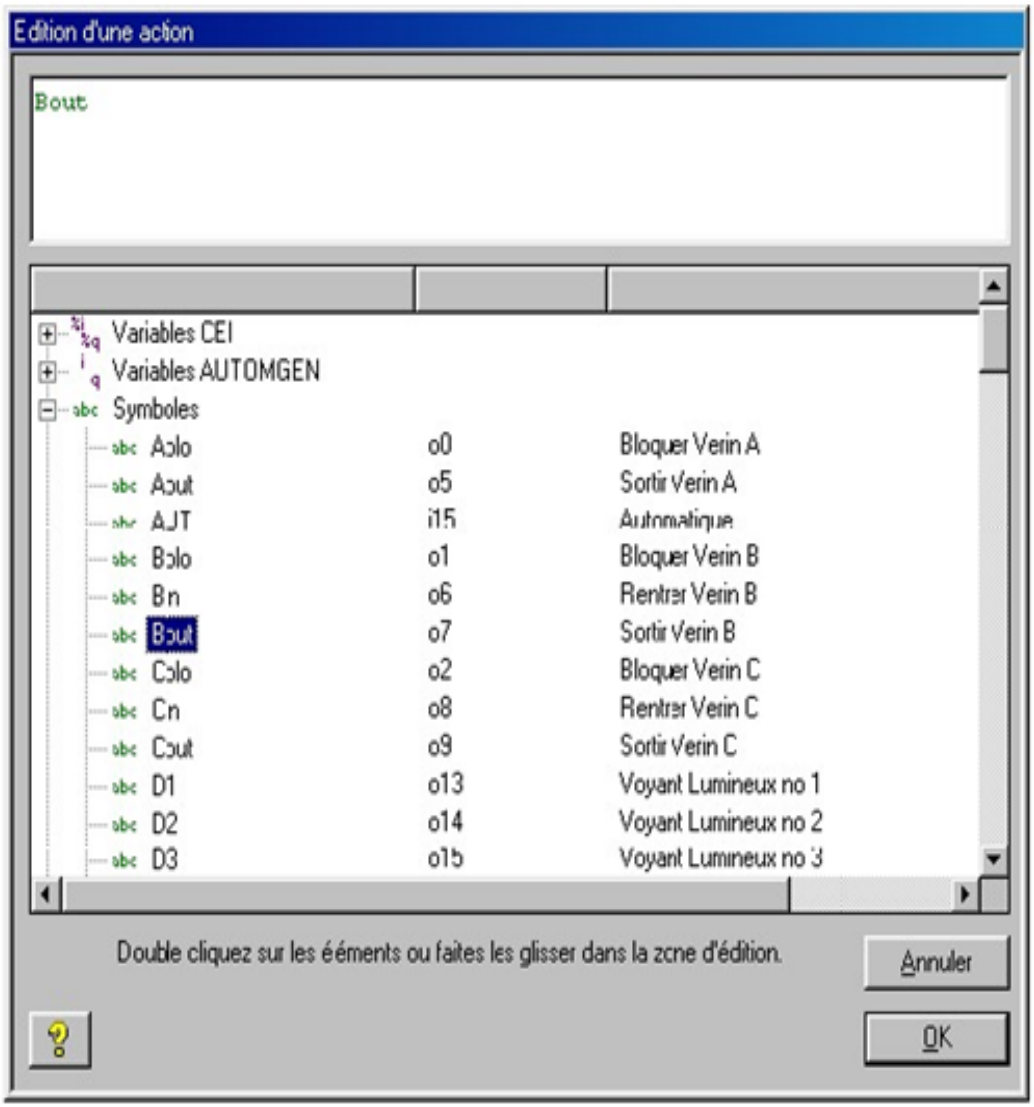


Figure 3.25 : Sélection d'un symbole

Au sein d'un même rectangle d'action, plusieurs actions peuvent être écrites en les séparant par le caractère « , » (virgule) signifiant « ET ». Toutefois, il est préférable d'utiliser la norme des GRAFCET qui est de mettre une action par rectangle. Il est normal que les lignes de connexion des rectangles d'actions supplémentaires ne touchent pas le rectangle précédent.

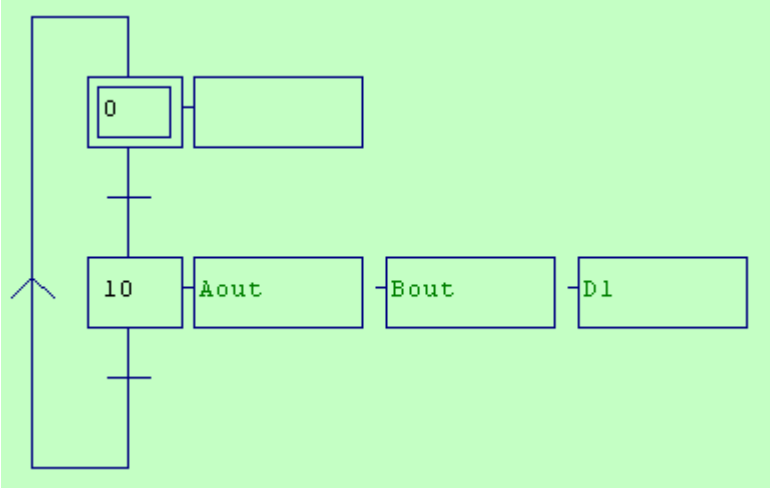


Figure 3.26 : Plusieurs actions à la même étape

De plus, on peut désirer, pour une même étape, plusieurs actions qui seront conditionnées de façon différente. Lorsqu'une condition est ajoutée sur un rectangle d'action, c'est l'ensemble des actions contenues dans ce rectangle qui sont conditionnées. Pour créer une action conditionnelle :

1- Placez le curseur sur le rectangle d'action, cliquez sur le bouton droit de la souris et choisissez « Action conditionnelle » dans lemenu.

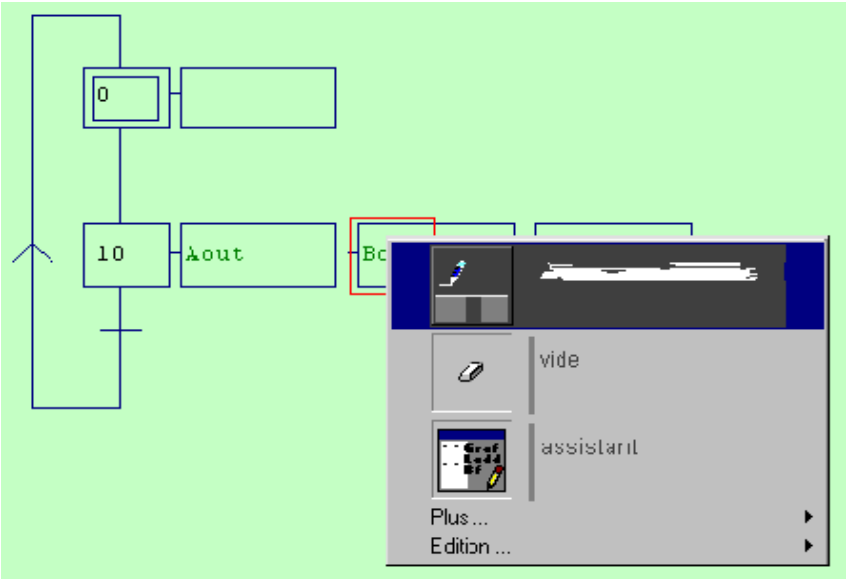


Figure 3.27 : Créer une action conditionnelle

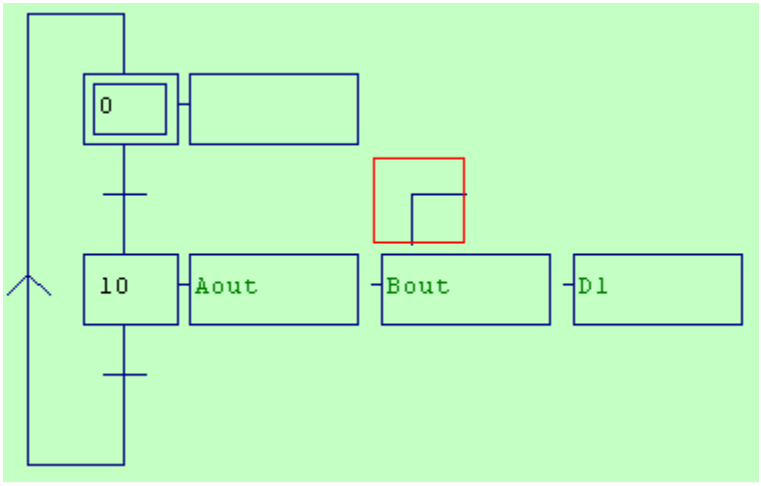


Figure 3.28 : La condition est créée

2- Pour documenter la condition sur l'action, cliquez avec le bouton gauche sur l'élément  et suivez les mêmes étapes que pour écrire dans les rectangles d'action (figure 22, 23, 24, 25). Assurez-vous que vous avez assez suffisamment d'espace au-dessus des rectangles d'actions.

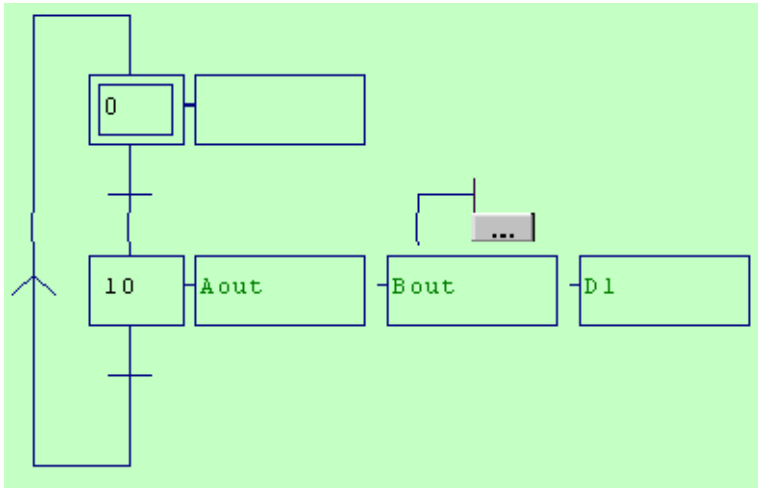


Figure 3.29 : Documenter une condition

*** Faites attention aux caractères lorsque vous écrivez avec AUTOMGEN : le zéro (0) est semblable au « O » majuscule et le un (1) est semblable au « l » minuscule! Ceci est valide en tout temps pour les GRAFCET comme pour les schémas à contacts.

Dans les rectangles d'action, vous pouvez utiliser des mémoires. On peut utiliser les noms suivants pour les mémoires : MEM00 à MEM20. Pour faire la mise à un (Set) d'une mémoire, on utilise la syntaxe suivante : « S variable ».

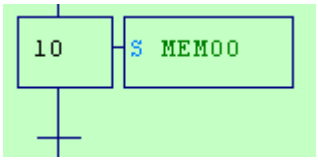


Figure 3.30 : Mise à un d'une mémoire

Pour faire la mise à zéro (Reset) d'une mémoire, on utilise la syntaxe suivante :
« R variable ».

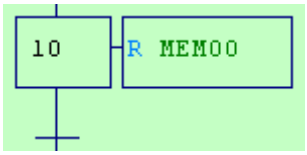


Figure 3.31 : Mise à zéro d'une mémoire

La mémoire s'activera quand l'étape sera active et gardera la valeur 1 (état vrai) jusqu'à l'étape où on fera la mise à zéro. La variable prendra alors la valeur 0 (état faux). Ne pas oublier de laisser un espace entre le « S » ou le « R » et le nom de la variable.
Les compteurs peuvent s'avérer aussi utile lors de la programmation d'un GRAFCET. On peut utiliser les noms suivants pour les compteurs : C0 à C15. La syntaxe de l'action « Incrémentation d'un compteur » est : « +Compteur » sans espace.

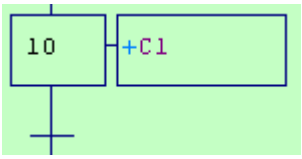


Figure 3.32 : Incrémentation d'un compteur

Donc, si la commande du rectangle d'action est à l'état vrai alors le compteur est incrémenté de 1 à chaque cycle d'exécution. La syntaxe de l'action « Décrémentation d'un compteur » est : « -Compteur ».

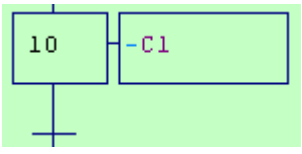


Figure 3.33 : Décrémentation d'un compteur

Donc, si la commande du rectangle d'action est à l'état vrai alors le compteur est décrémenté de 1 à chaque cycle d'exécution.En début de programme ou lorsque l'on a plus besoin du compteur, il faut faire la mise à zéro (Reset) de celui-ci. On utilise alors la syntaxe suivante : « R compteur ».

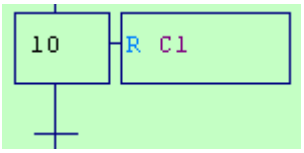


Figure 3.34 : Mise à zéro d'un compteur

On peut aussi incrémenter ou décrémenter un compteur dans une action sur une impulsion sur front montant « P1 » ou sur front descendant « P0 ». L'utilité de ces fonctions vient du fait que le logiciel balaie à une certaine fréquence tout le GRAFCET. Donc, si on a à l'étape qui est active une sortie de tige et l'incrémentation d'un compteur:

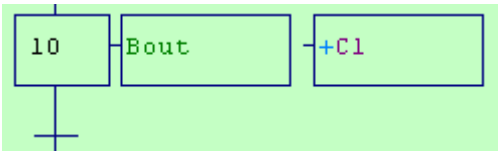


Figure 3.35 : Action et compteur à la même étape

Le logiciel incrémentera le compteur de 1 jusqu'à ce que le vérin B atteigne le détecteur de fin de course. On se retrouverait alors avec une valeur de C1 égale à 20 environ lorsque B aurait sorti qu'une seule fois ce qui est totalement indésirable.

Pour éviter ce problème, on utilise l'incrémentation sur une impulsion sur front montant « P1 », c'est-à-dire que le compteur s'incrémente de 1 lorsque l'impulsion du signal de sortie de B est donnée seulement ou bien l'incrémentation sur une impulsion sur front descendant « P0 », c'est-à-dire que le compteur s'incrémente de 1 lorsque le signal de sortie de B est arrêté seulement. On utilise alors la syntaxe suivante : « Px +compteur».

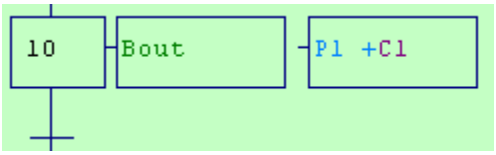


Figure 3.36 : Utilisation du front montant P1

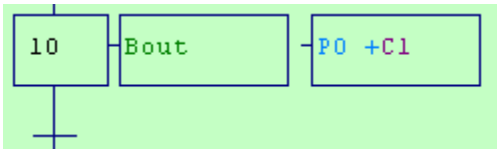


Figure 3.37 : Utilisation du front descendant P0

La temporisation est un autre outil utile en programmation. Les temporisations sont considérées comme des variables booléennes et peuvent être utilisées avec les actions de « Mise à un » et de « Mise à zéro » comme pour les mémoires et les compteurs. On peut utiliser les noms suivants pour les compteurs : T0 à T15. La syntaxe est la suivante : « temporisation (durée)».

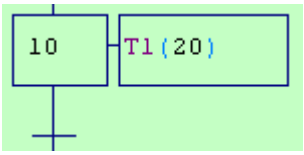


Figure 3.38 : Temporisation temporaire

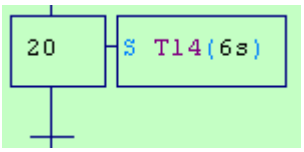


Figure 3.39 : Temporisation permanente

La durée est par défaut exprimée en dixième de secondes. On peut cependant l'exprimer en jours, heures, minutes, secondes et millisecondes avec les opérateurs « d », « h », « m », « s » et « ms ». Par exemple : 1d30s égale 1 jour et 30 secondes.
L'étape 10 de la figure 39 lance une temporisation de 2 secondes qui restera active tant que l'étape le sera. L'étape 20 de la figure 40 arme une temporisation de 6 secondes qui restera active même si l'étape 20 est désactivée. De cette façon, une même temporisation peut être utilisée à plusieurs endroits avec une même consigne de temps et à des instants différents. Il faudra toutefois la mettre à zéro (Reset) lorsque vous en n'aurez plus besoins et aussi en début de cycle.
*** Une autre syntaxe existe pour les temporisations. Elle s'utilise pour les transitions et les conditions seulement. Voir la section 2.3.3 Les tests à la page 31.

3.3.2.2 Pour les schémas à contacts

Dans le cas des schémas à contacts les actions sont utilisées dans les bobines du langage la dder.

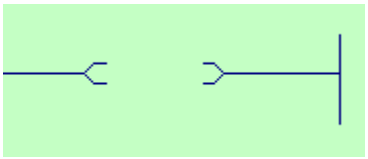


Figure 3.40 : Bobine

Pour pouvoir écrire dans ces bobines, il suffit de cliquer avec le bouton de gauche sur la première parenthèse. Pour documenter l'action, on procède de la même façon que pour les actions du GRAFCET (figure 22, 23, 24,25).

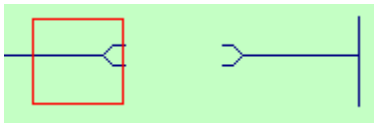


Figure 3.41 : Documenter une bobine

Au sein d'une même bobine, plusieurs actions peuvent être écrites en les séparant par le caractère « , » (virgule) signifiant « ET ». Toutefois, il est préférable d'utiliser la norme des schémas à contacts qui est de mettre une action seulement par bobine. Il suffit de mettre les bobines en parallèle.

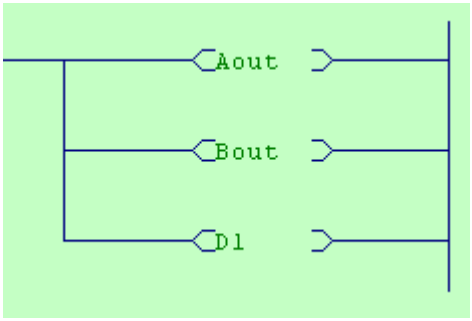


Figure 3.42 : Actions en parallèle

Dans les schémas à contacts, on utilise les relais. On peut utiliser les noms suivants pour les relais : REL01 à REL30. Ils s'utilisent de la même manière que les mémoires :

Figure 3.43 : Les relais dans une action avec le schéma à contacts

La syntaxe et les règles pour les compteurs et les temporisations sont les mêmes que pour les GRAFCET. Ici aussi, le logiciel balaie à une certaine fréquence tout le schéma à contact. On peut donc faire face au même problème d'incrémentation ou de décrémentation erronée du compteur d'où l'utilisation du front montant « P1 » et descendant « P0 ».

Figure 3.44 : Les compteurs dans une action avec le schéma à contacts

Figure 3.45 : Les temporisations dans une action avec le schéma à contacts

3.3.3. Les tests

Un test est une équation booléenne (expression logique) de une ou de « n » variables séparées par les opérateurs booléens « + » (Ou) ou « . » (Et). Par défaut, l'opérateur « . » (Et) a la priorité sur l'opérateur « + » (Ou). Des parenthèses peuvent être utilisées pour définir une autre priorité.
Noter que les lumières (D1 à D6) ne peuvent pas être utilisées comme transition. En effet, les lumières sont des sorties au même titre que les actions sur les vérins (Aout, Bin,etc.).

3.3.3.1. Pour les GRAFCET

Les tests sont utilisés dans les transitions du langage GRAFCET.

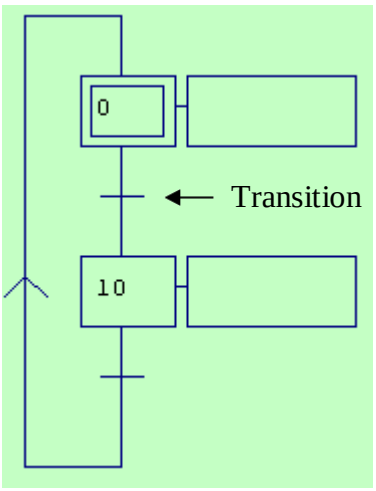


Figure 3.46 : Transition

Pour pouvoir écrire dans ces transitions, il suffit de cliquer avec le bouton de gauche sur la transition en question.

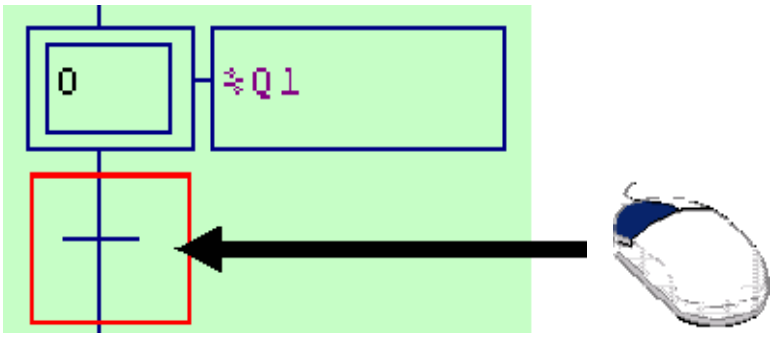


Figure 3.47 : Accès à l'écriture d'un test

Pour documenter la transition, on procède de la même façon que pour les actions du GRAFCET (figure 22, 23, 24, 25).

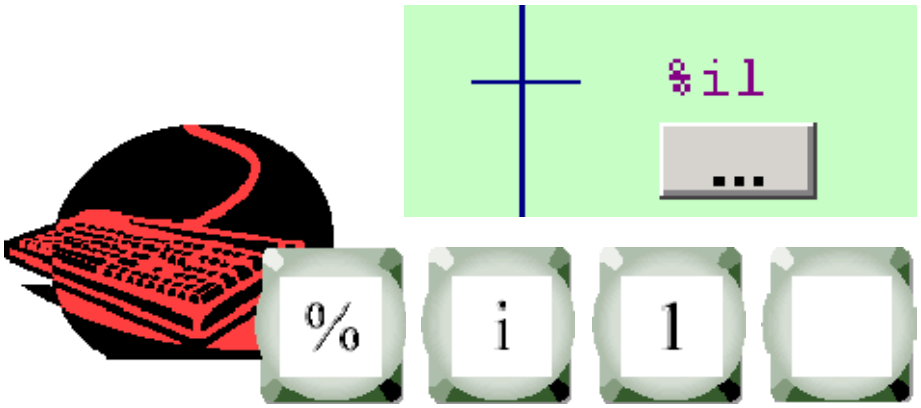


Figure 3.48 : Écriture d'un test

Par défaut, si le seul nom d'une variable est spécifié, le test est « si égale à un » (si vrai). Des modificateurs permettent d'autres états de variables :

1- Le caractère « / » placé devant une variable teste l'état complémenté (sifaux);

Figure 3.49 : État complémenté

2- Le caractère « ↑ » placé devant une variable teste le frontmontant;

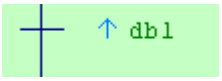


Figure 3.50 : Test sur front montant

3- Le caractère « ↓ » placé devant une variable teste le front descendant.

Figure 3.51 : Test sur front descendant

Pour obtenir les caractères « ↑ » et « ↓ » pendant l'édition d'un test pressez sur les touches du clavier [↑] et [↓] respectivement. N'oubliez pas de laisser un espace entre la flèche et le nom de la variable. Les modificateurs de tests peuvent s'appliquer à une variable ou à une expression entre parenthèses.

Pour obtenir un test toujours vrai, on utilise la syntaxe « » (néant) ou « =1 ».

Figure 3.52 : Test toujours vrai

Il est possible d'utiliser les mémoires (sans le « Set » ni le « Reset »), les compteurs et les temporisations comme transitions. Pour les compteurs, plusieurs tests sont disponibles. La syntaxe est la suivante : « compteur » « type de test » « constante ou variable ». Il n'y a aucun espace entre les trois termes. Le type de test peut être : « = » égale, « ! » ou « <> » différent, « < » inférieur, « <= » inférieur ou égal, « > » supérieur,
« >= » supérieur ou égal.

Figure 3.53 : Exemples de tests pour les compteurs

Pour les temporisations, on peut tout simplement nommer le temporisateur utilisé lors de l'action précédente.

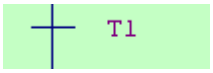


Figure 3.54 : Test avec une temporisation temporaire ou permanente

Une autre syntaxe possible est de la forme suivante : « temporisation / variable de lancement / durée ». Il ne faut pas laisser d'espace entre les termes et le symbole « / ». La variable de lancement est généralement l'étape précédente et s'écrit de la façon suivante : « xnuméro d'étape ».

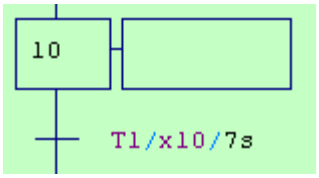


Figure 3.55 : Temporisation utilisée dans la transition seulement

Pour passer à l'étape 11, il faut donc que le temporisateur T1 ait comptabilisé 7 secondes à partir de l'activation de l'étape 10. Lorsque la transition est franchie, T1 est remise à zéro automatiquement.

3.3.3.2. Pour les schémas à contacts

Les tests sont utilisés dans les contacts du langage ladder.

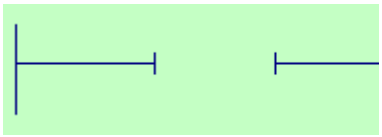


Figure 3.56 : Contact

Pour pouvoir écrire dans ces contacts, il suffit de cliquer avec le bouton de gauche sur le premier bord du contact. Pour documenter le test, on procède de la même façon que pour les actions du GRAFCET (figure 22, 23, 24, 25).

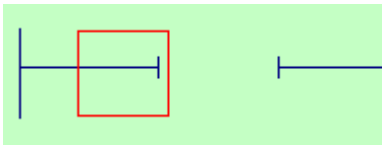


Figure 3.57 : Documenter un contact

Lors de l'élaboration de votre schéma à contacts, vous pouvez utiliser les fonctions « Ou » et « Et » pour les contacts. L'important est de respecter les deux règles suivantes : les contacts sont en parallèle pour le « Ou » et en série pour le « Et ». Vous devez écrire qu'un seul test par contact selon les normes des schémas àcontact.

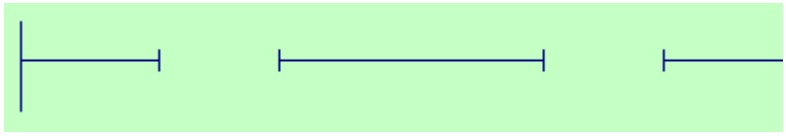


Figure 3.58 : Contacts en série « Et »

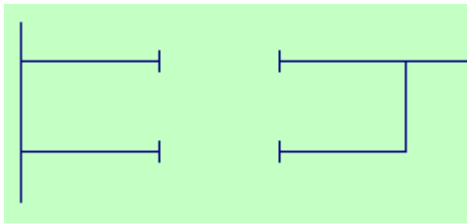


Figure 3.59 : Contacts en parallèle « Ou »

Les modificateurs des figures 49, 50, 51 peuvent aussi être utilisés dans les contacts. Il est possible d'utiliser les relais (sans le « Set » ni le « Reset »), les compteurs et les temporisations comme tests dans les contacts. La syntaxe des tests sur les compteurs est la même que pour les transitions du GRAFCET (voir page31).

Figure 3.60 : Exemples de compteurs dans un contact

Pour les temporisations, on nomme tout simplement la temporisation temporaire ou permanente.

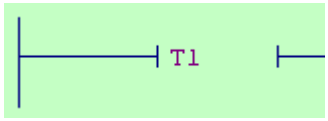


Figure 3.61 : Exemple d'une temporisation dans un contact

Remarquez cependant que vous ne pouvez pas utiliser la syntaxe des temporisations « temporisation / variable de lancement / durée » car il n'y a pas de numéro d'étape dans le langage ladder.

3.4. Le compilateur

Celui-ci vérifie s'il n'y a pas d'erreurs dans votre programme. Il est donc fortement recommandé de l'exécuter avant de tester votre programme sur l'ordinateur ou Sur l'automate .Pour débuter la compilation cliquez sur l'icône «Compile»  situé dans la barre d'outils. Pour l'arrêter cliquez sur l'icône « Arrête la compilation»  .

À la fin de la compilation, l'onglet « Compilation » de la fenêtre des messages donne la liste des éventuelles erreurs. Si la fenêtre est verte, il n'y a aucune erreur.

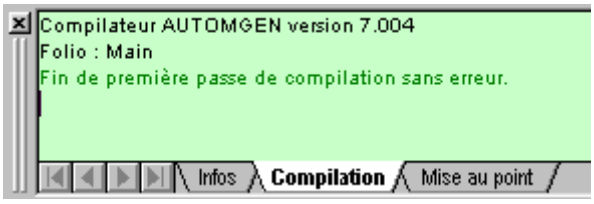


Figure 3.62 : Compilation sans erreur

Si elle est rouge, les différents messages d'erreurs seront affichés. En double cliquant sur les messages d'erreurs, vous pouvez en localiser la source.

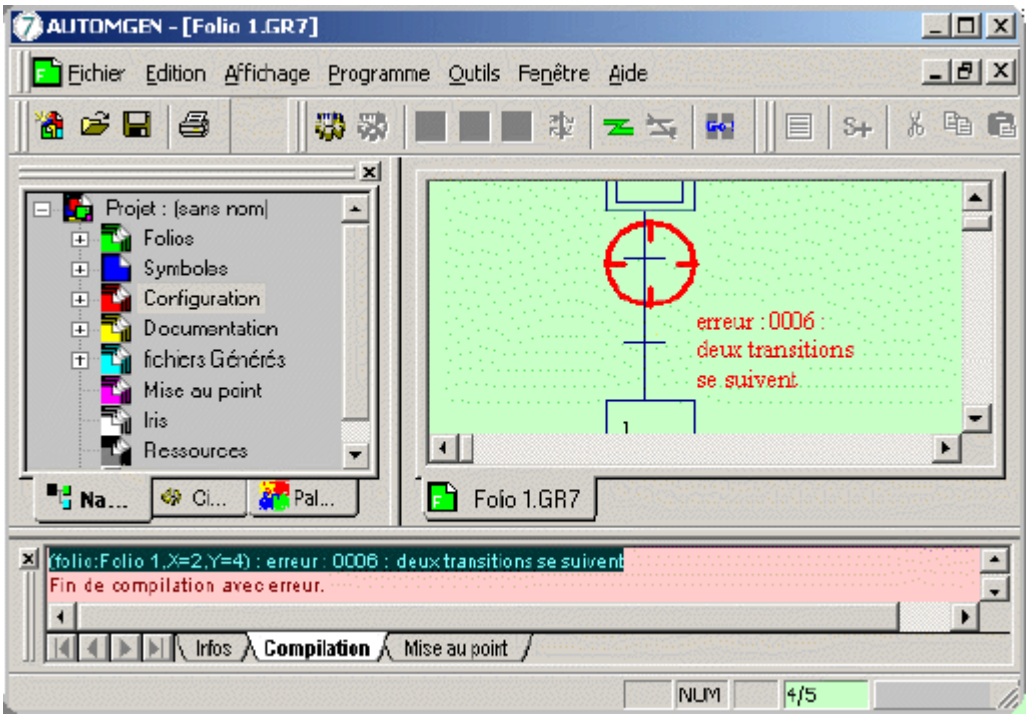


Figure 3.63 : Message d'erreur et localisation de sa source

Vous pouvez aussi retracer l'emplacement de l'erreur en utilisant ses coordonnées cartésiennes données dans le message. Effectivement, lorsque vous promenez le curseur sur l'espace de travail, les coordonnées cartésiennes sont affichées dans un rectangle vert en bas à droite de l'écran. La nomenclature est : X / Y pour ligne / colonne.

3.5. Exécuter une application

Lorsque vous avez compilé votre programme et que vous avez vérifié qu'il ne contenait pas d'erreur, vous pouvez l'exécuter soit sur le PC soit sur le panneau de simulation. Cela dépend de quel post-processeur vous aurez sélectionné dans l'onglet nommé « Cibles ». Vous pouvez voir en tout temps où est rendu le GRAFCET (boule bleue) grâce à la visualisation dynamique en cliquant sur l'icône à la droite de la barre d'outils ou en faisantF2.

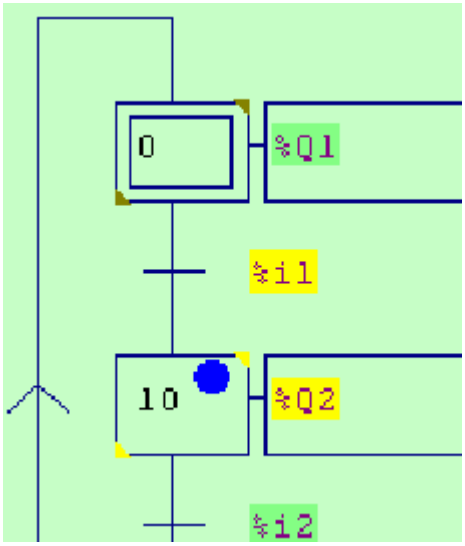


Figure 3.64 : Visualisation dynamique

La méthode la plus rapide pour observer le résultat de l'exécution d'une application est de cliquer sur l'icône « GO » de la barre d'outils après avoir choisi la cible désirée. Ce bouton active :

- La compilation de l'application si elle n'est pas à jour;
- L'installation du module d'exécution (PC ou automate);
- Le passage de la cible en RUN;
- L'activation de la visualisation dynamique.

La méthode plus longue consiste à exécuter chacune des étapes mentionnées ci- dessus une après l'autre :

1- Choisir la cible (PC ou PL72);
2- Compiler le programme en cliquant sur l'icône ;
3- Installerl'exécuteurPCouseconnectersurl'automateencliquantsurl'icône « Connexion » ;
4- Mettre la cible en mode RUN en cliquant sur l'icône ;
5- Activer la visualisation dynamique en cliquant sur l'icône .

Si vous êtes sur l'exécuteur PC, vous pouvez manipuler le GRAFCET ou le schéma à contacts en cliquant sur les différentes conditions des transitions ou des contacts pour changer leur état d'activation. Pour ce faire, il faut que la visualisation dynamique soit activée. Les symboles en jaune sont actifs et ceux en vertinactifs.

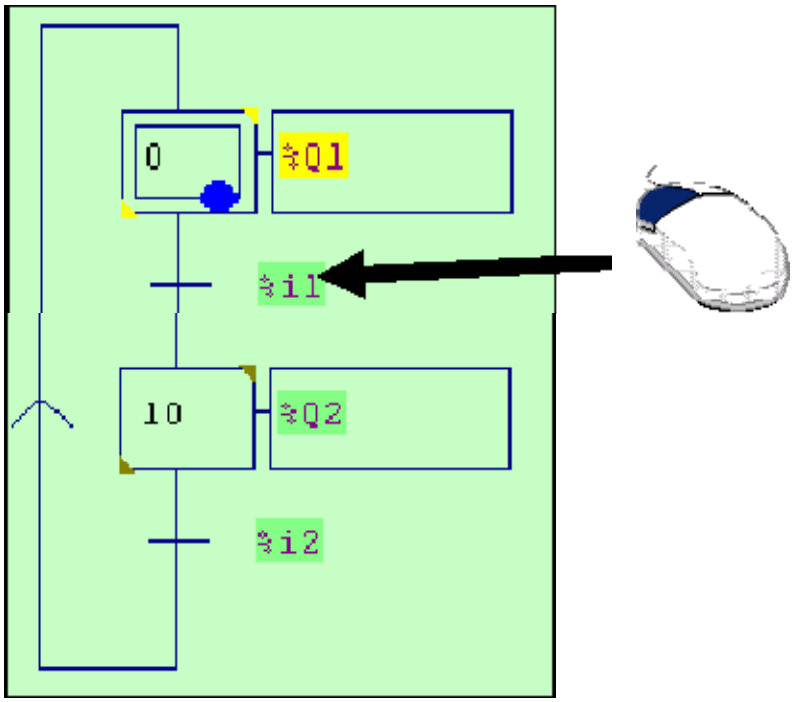


Figure 3.65 : Activation d'une transition

Si vous êtes sur le PL72, le panneau de simulation est prêt à fonctionner. Vous pouvez alors simuler et visualiser de façon concrète votre programme à l'aide des différents boutons du panneau.

Vous pouvez réinitialiser la cible en cliquant sur l'icône .

Pour mettre fin à l'application, cliquez sur l'icône « GO » en foncé . Si vous êtes sur le PL72, il est préférable d'utiliser la méthode suivante:

1- Mettre la cible en mode STOP en cliquant sur l'icône ;
2- Se déconnecter de l'automate en cliquant sur l'icône . De cette façon, l'automate programmable n'est plus en mode « RUN».
3- Vous pouvez alors modifier votre programme.

3.6. Sauvegarder un projet

Pour sauvegarder un projet au complet il faut cliquer sur « Enregistrer sous… » dans le menu « Fichier ». Il suffit ensuite de donnez un nom à votre fichier qui portera l'extension « .agn ». N'oubliez surtout pas de sauvegarder fréquemment votre travail, la fonction « undo » n'étant pas disponible avec AUTOMGEN.
Il est aussi possible de sauvegarder le GRAFCET ou le schéma à contacts seulement. Pour ce faire, il faut cliquer avec le bouton de droite sur le folio « Main » et choisir l'option « Exporter ».

Figure 3.66 : Exporter un folio

Le fichier portera l'extension « .gr7 ». Toutefois, ce fichier est beaucoup plus gros que le fichier « .agn » comportant le projet au complet (environ 700k contre une dizaine pour le projet total!). Pour aller chercher un GRAFCET ou un schéma à contacts « .gr7 », il faut d'abord ouvrir AUTOMGEN et ensuite cliquer avec le bouton de droite sur « Folios » et choisir « Importer un ou plusieurs folios existants ».

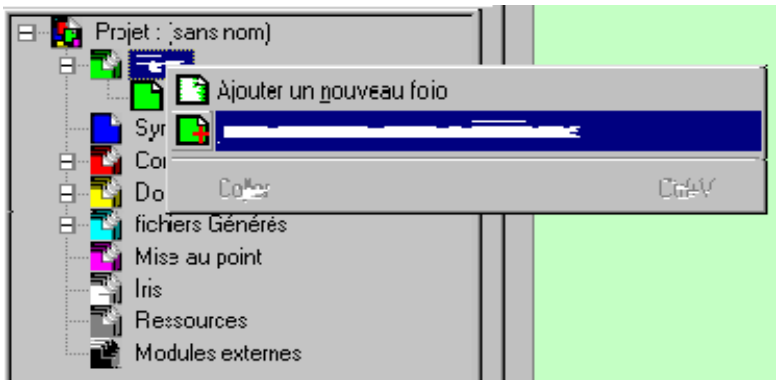


Figure 3.67: Importer un folio

3.7. Impression du GRAFCET et du schéma à contacts

Tout d'abord, vous allez dans le menu « Fichier » et cliquez sur « Aperçu avant impression ». De cette manière, vous voyez si la mise en page de votre programme dans l'espace de travail est correcte. Ensuite, vous n'avez plus qu'à cliquer sur « Imprimer ».

4. EXEMPLES DEPROGRAMMATION

4.1. GRAFCET

Exemple 1 : Construire le GRAFCET de deuxième niveau avec urgence et l'option répétée de la table de machine-outil inspirée de l'exemple 8.1 se trouvant à la page 357 du manuel « Circuits Hydrauliques ».

Dû à la nature des distributeurs des vérins au laboratoire, on choisira le vérin E pour les déplacements DR et GA et le vérin B pour les déplacements AR et AV. On présente deux versions d'urgence différentes. La première se fait en appuyant sur le bouton « URG » qui remet les vérins à leur position initiale via le bouton poussoir S5 et la deuxième en appuyant sur le bouton « S6 » qui bloque les vérins.

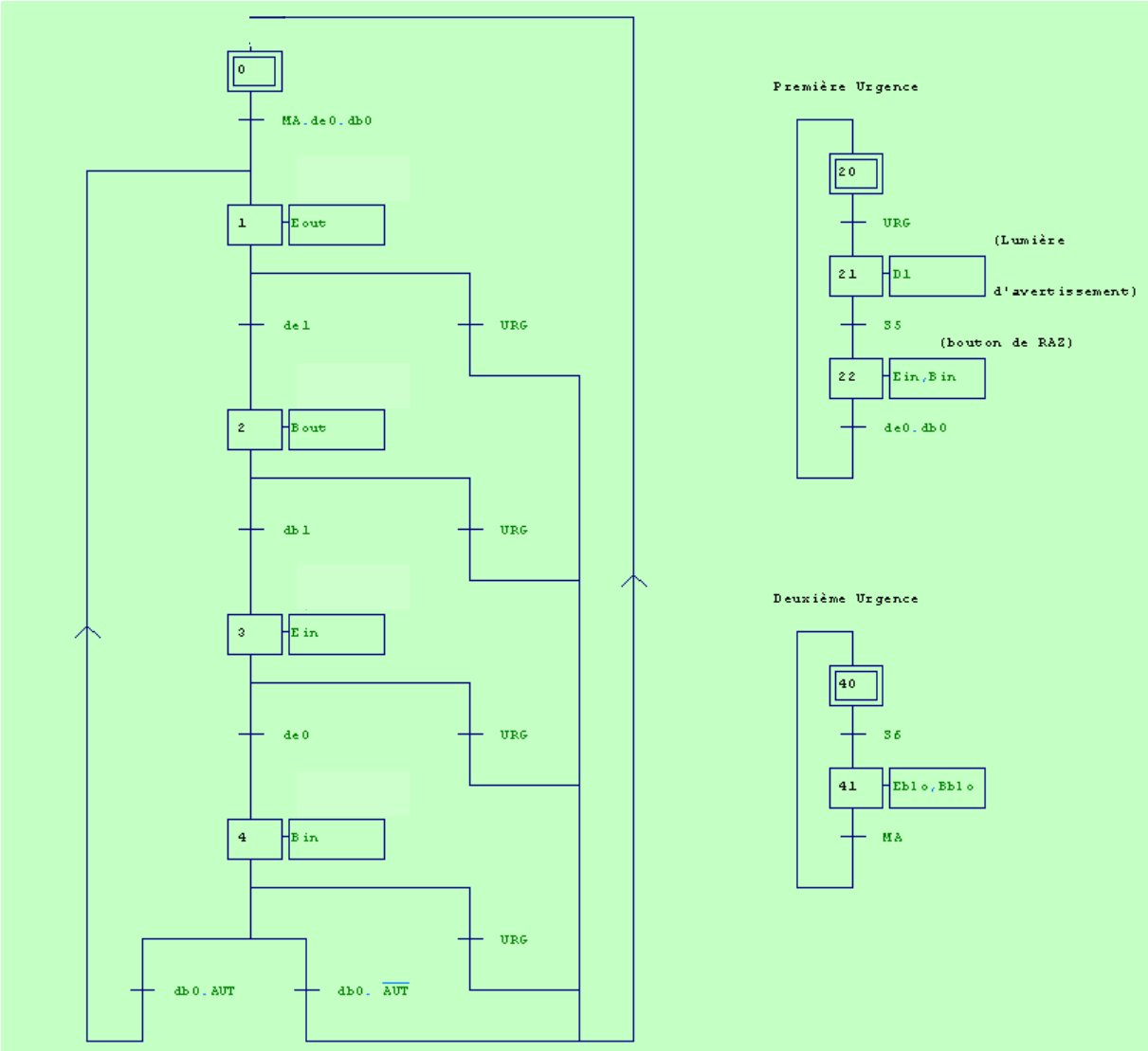


Figure 3.68 : GRAFCET de l'exemple 1

Compteurs :

Dans les deux exemples suivants, on montre deux façons différentes d'utiliser les compteurs. Les deux consistent en 4 cycles de sortie et d'entrée du vérin B.

Exemple 2 : Dans cet exemple, on utilise le compteur en même temps que le vérin B sort. Pour éviter le problème mentionné au cas précédent, on utilise l'incrémentation sur une impulsion sur front montant « P1 », c'est-à-dire que le compteur s'incrémente de 1 lorsque l'impulsion du signal de sortie de B est donnée seulement. On a rajouté l'étape vide 24 car,sinon, l'étape 23 désactive l'étape 22 et pourrait aussi l'activer si le compteur est différent de 4. On se retrouverait alors avec une étape qui désactive et active une autre ce qui est un non-sens!

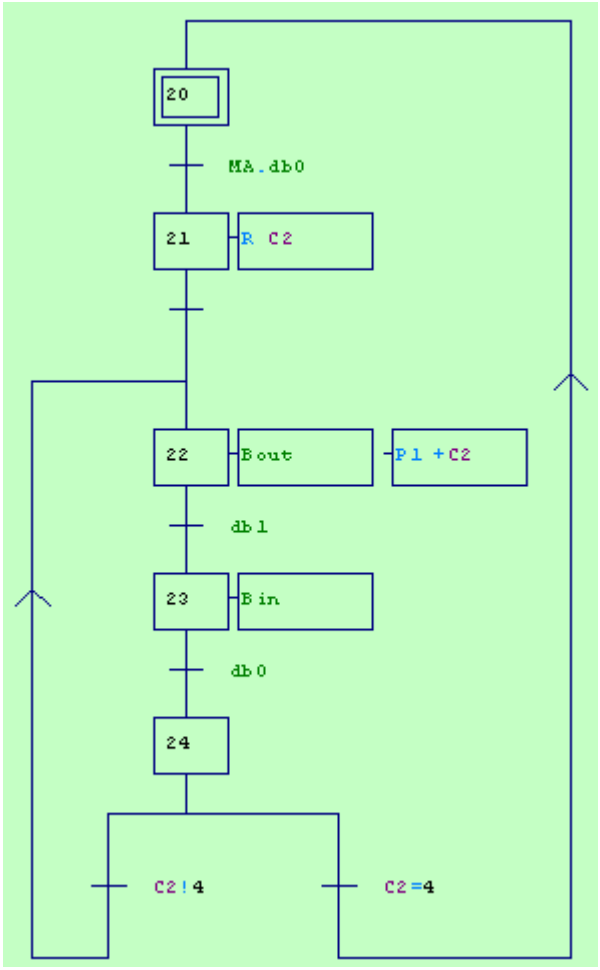


Figure 3.69 : GRAFCET de l'exemple 2

Exemple 3 : Ici, on utilise le compteur seul dans l'étape 4. Si on l'avait mis à l'étape 2 avec la sortie du vérin B par exemple, le compteur aurait incrémenté sa valeur de 1 jusqu'à ce que le vérin B atteigne le détecteur de fin de course. On se retrouverait alors avec une valeur de C1 égale à 20 environ ce qui est totalement indésirable. Pour cette raison, on privilégiera l'utilisation des fronts montant et descendants comme à l'exemple2.

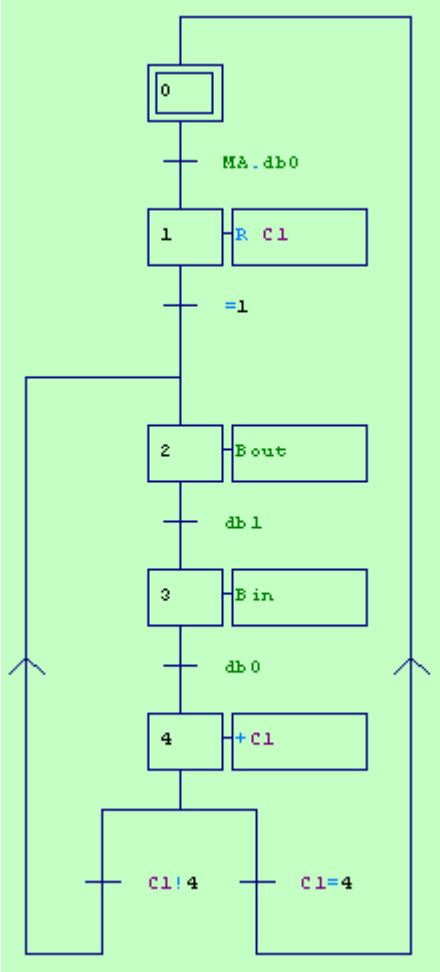


Figure 3.70 : GRAFCET de l'exemple 3

Temporisations :

Dans les trois exemples suivants, on montre les trois façons différentes d'utiliser les temporisations.

Exemple 4 : D'abord, on a une temporisation dont le résultat est utilisé immédiatement après l'étape d'activation. La temporisation 2 compte 10 secondes dès que le vérin B est complètement sortie. Le vérin B reste donc sortie pendant 10s avant de se rétracter. Cette temporisation sert de transition immédiatement après l'étape 21 et se désactive automatiquement quand l'étape 22 devient active. Remarque : avec le type de distributeur associé à B, un front montant (P1 Bout) suffit. Cela nous permet d'éviter d'alimenter le solénoïde durant de longuespériodes.

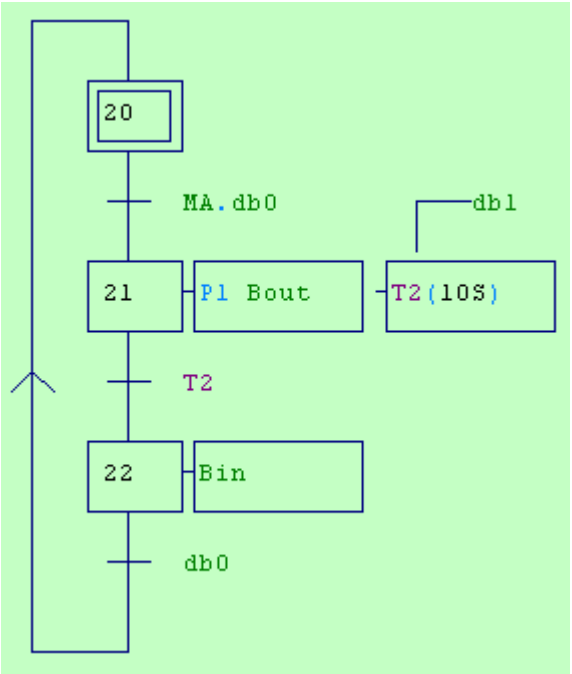


Figure 3.71 : GRAFCET de l'exemple 4

Exemple 5 : Ensuite, on a une temporisation dont le résultat est exploité beaucoup plus tard dans le GRAFCET. La temporisation 1 compte 15 secondes à partir de l'étape 1 [S T1(15s)]. Cette temporisation reste active jusqu'à sa désactivation « Reset » qui a lieu à l'étape 3 [R T1]. On veut que le vérin A sorte à l'étape 3, c'est-à-dire après le retrait de B et que 15s se soit écoulées depuis l'activation de l'étape1.

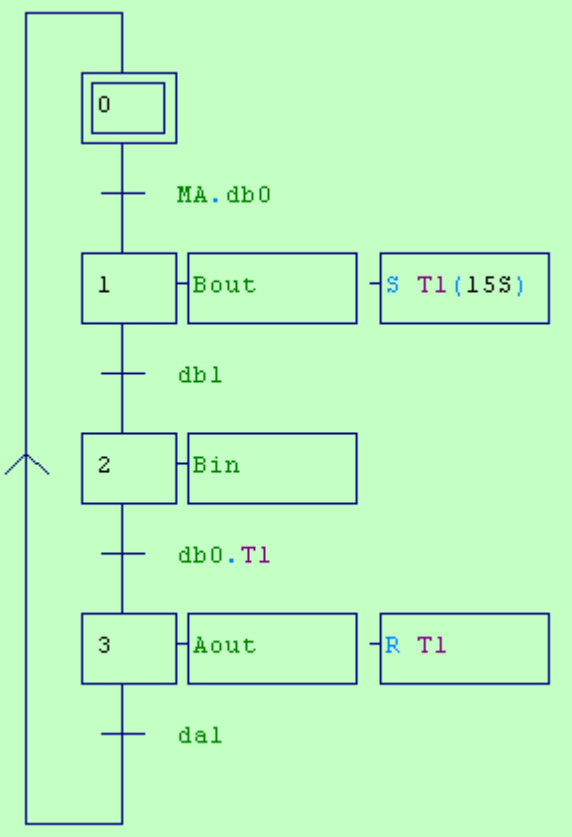


Figure 3.72 : GRAFCET de l'exemple 5

Exemple 6 : Finalement, on a une temporisation qui est définie entièrement dans une transition et dont le résultat est utilisé immédiatement. La temporisation 3 est utilisée comme transition seulement. Elle compte 10 secondes à partir de l'activation de l'étape 31 (x31). Lorsque l'étape 32 devient active après les 10 secondes, T3 se désactive automatiquement.

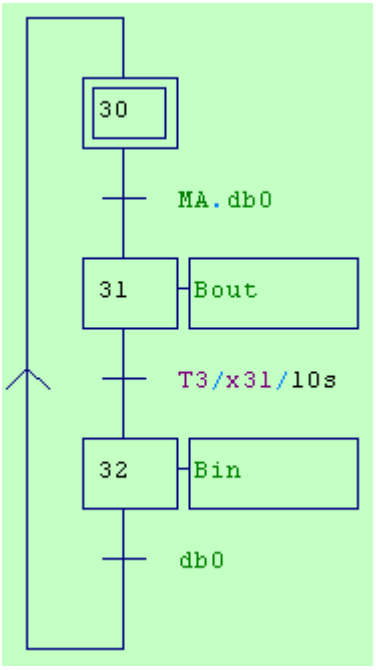


Figure 3.73 : GRAFCET de l'exemple 6

Exemple 7 : Construire le GRAFCET de deuxième niveau avec urgence et l'option répétée de la table de machine-outil de l'exemple 1 avec l'ajout des temporisations.

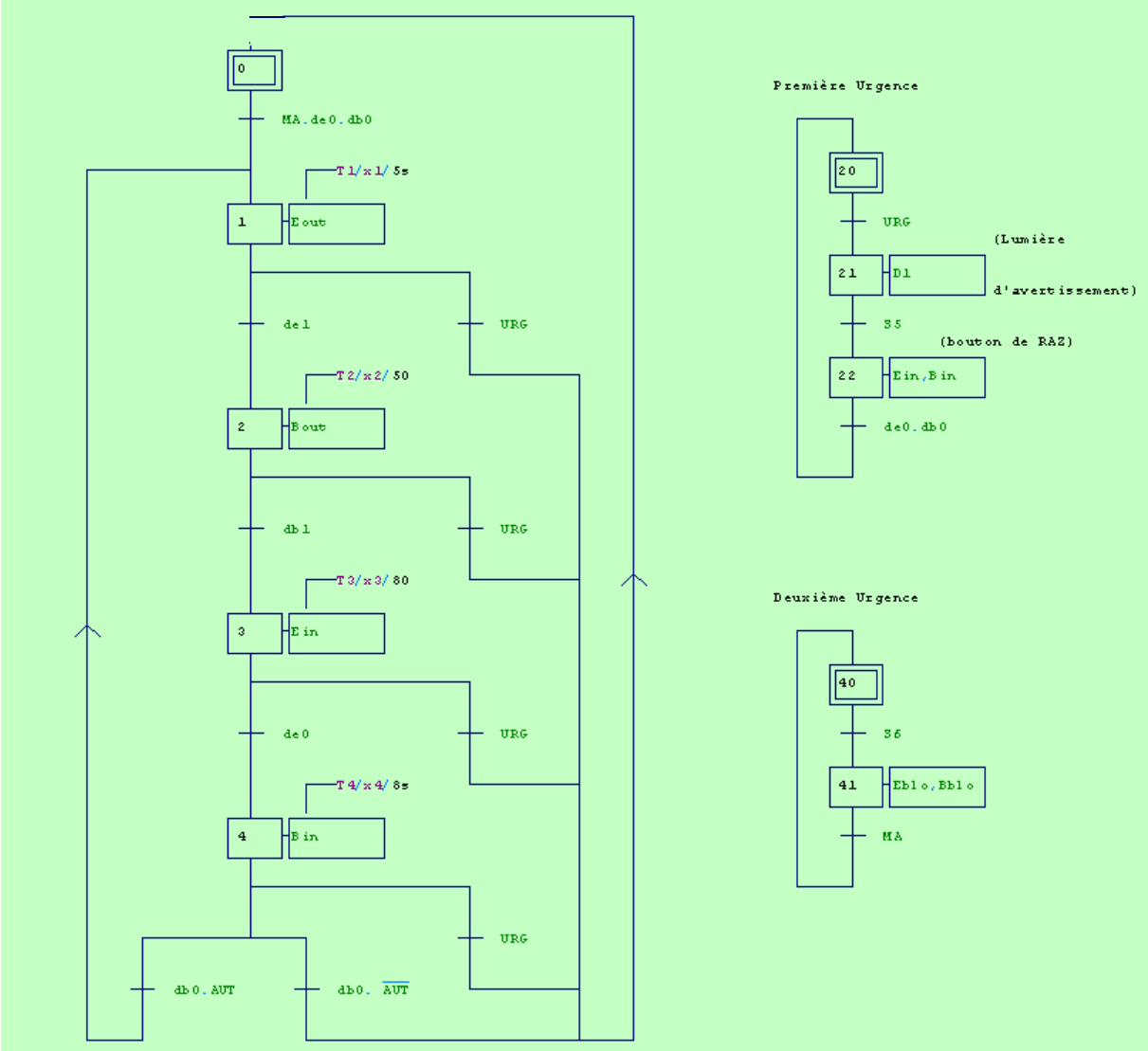


Figure 3.74 : GRAFCET de l'exemple 7

4.2. Schémas à contacts

Exemple8 : Construire le schéma à contacts avec urgence de la table de machine-outil correspondant à l'exemple 1 de la section 3.3 du présentdocument.
Dû à la nature des distributeurs des vérins au laboratoire, on choisira le vérin E pour les déplacements DR et GA et le vérin B pour les déplacements AR et AV.

Remarque: -l'usage de la mise à un des mémoires nécessite leur mise à zéro à l'étape suivante. On a donc plus besoins des contacts $\overline{CR}_{i+1}$ à l'étape « i » pour désengager le relais « i ».

- On a donc plus besoins non plus des contacts $\overline{URG}$ car les mémoires sont mises à zéro lorsque l'urgence est enclenchée.
- C'est une façon de faire, l'autre façon vue en classe est plus proche de la version câblée du montage.

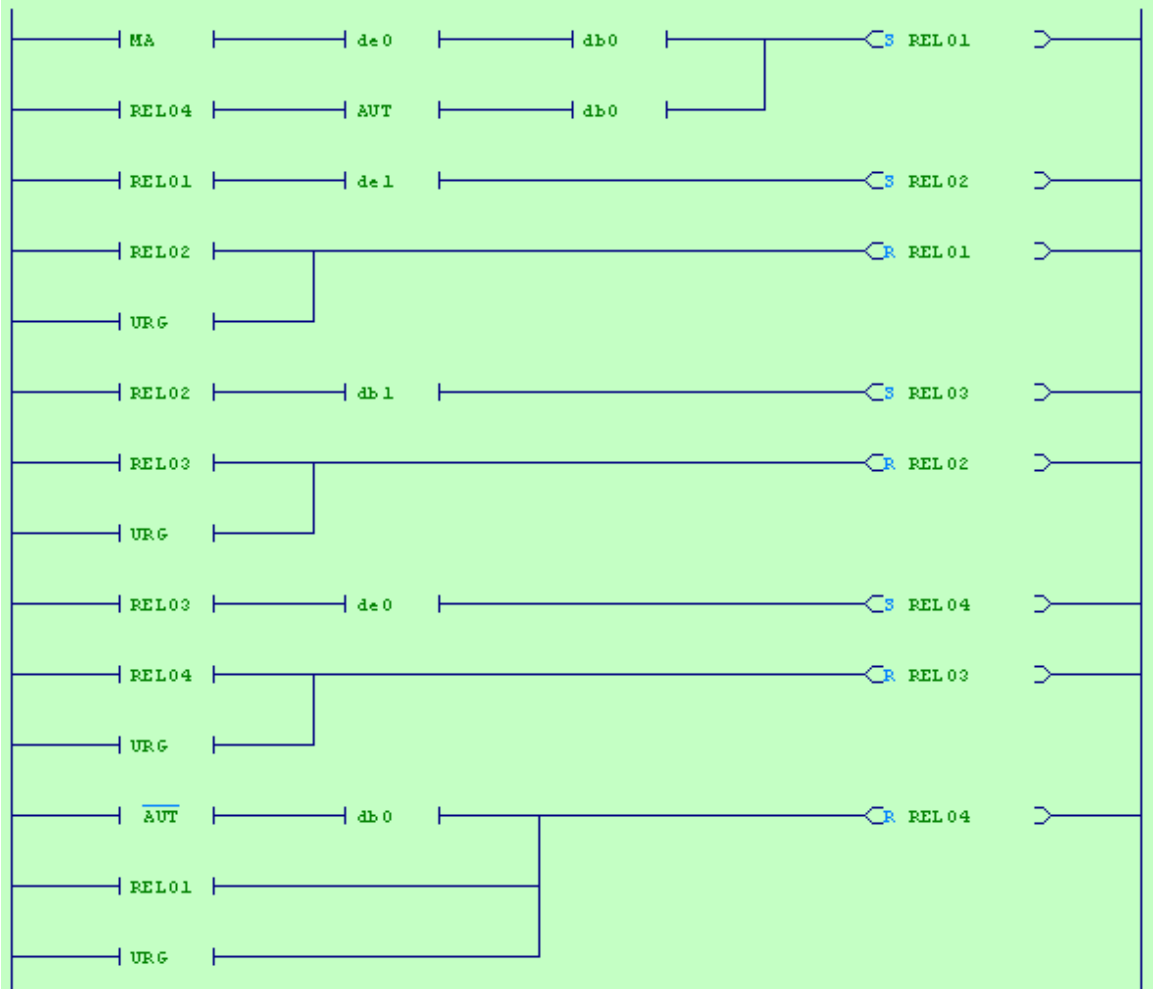


Figure 3.75 : Schéma à contacts de l'exemple 8, première partie

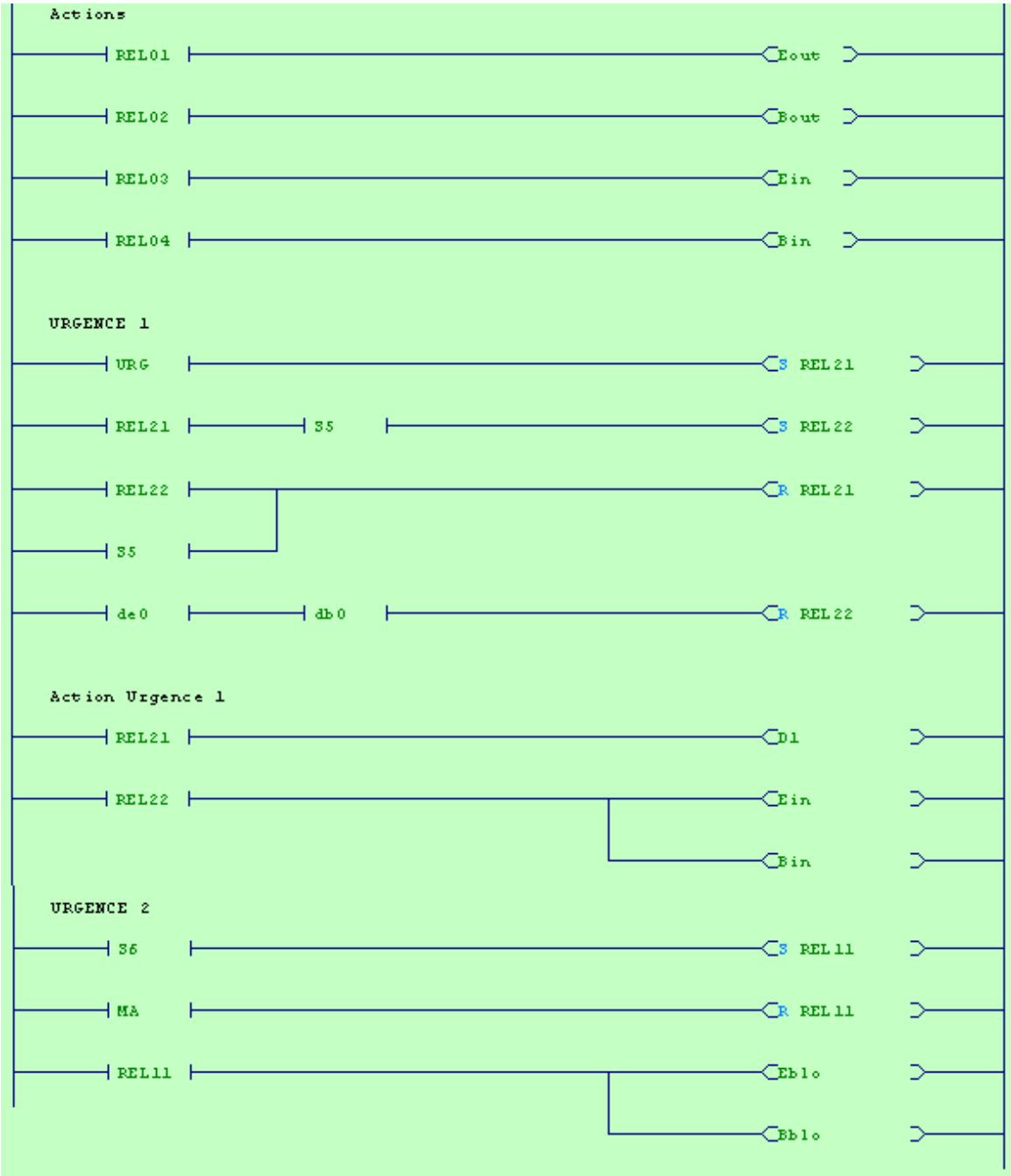


Figure 3.76 : Schéma à contacts de l'exemple 8, deuxième partie

Compteurs :

Dans l'exemple suivant, on montre la façon d'utiliser les compteurs. Elle consiste en 4 cycles de sortie et d'entrée du vérin B.

Exemple 9 : C'est le schéma à contacts de l'exemple 2 à l'exception qu'on a utilisé l'incrémentation sur une impulsion sur front descendant « P0 », c'est-à-dire que le compteur s'incrémente de 1 lorsque le signal de sortie de B est arrêté seulement.

***Notez qu'avec le langage de schéma à contacts vous devez utiliser cette méthode. Il est impossible de faire comme à l'exemple 3 car la fréquence de balayage du programme est trop rapide : le compteur s'incrémentera de 2 dans le temps d'activer un relais et de le désactiver!

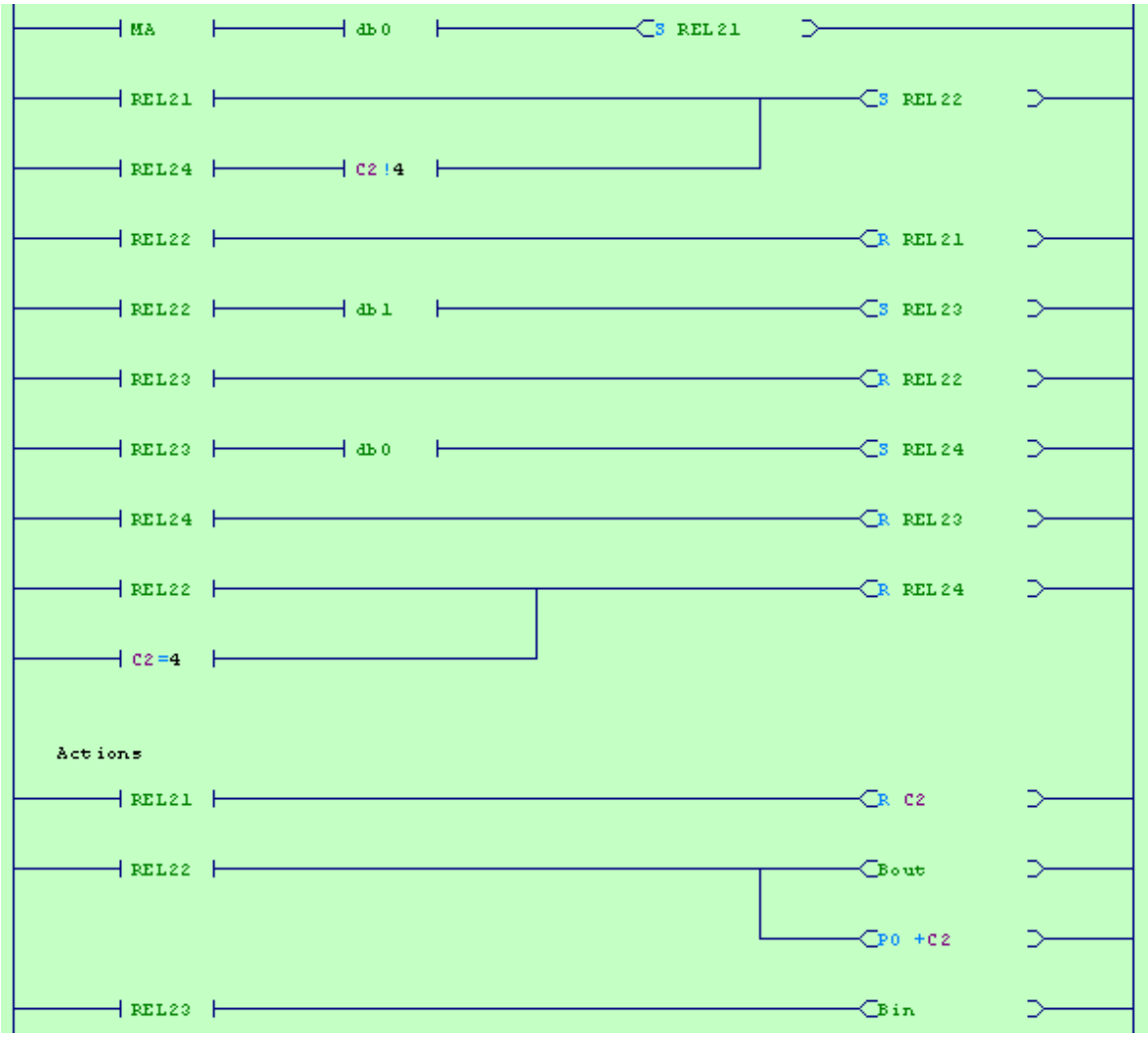


Figure 3.77 : Schéma à contacts de l'exemple 9

Temporisations :

Dans les deux exemples suivants, on montre les deux façons différentes d'utiliser les temporisations.

Exemple 10 : D'abord, on a une temporisation permanente. C'est le schéma à contacts de l'exemple4.

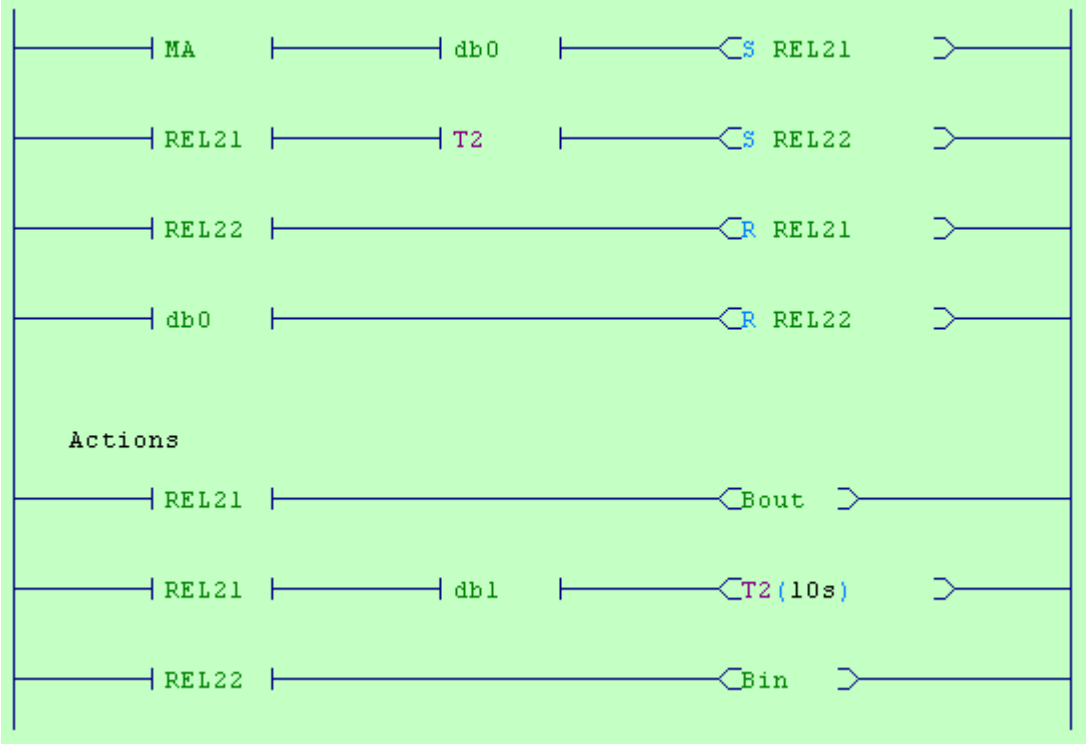


Figure 3.78 : Schéma à contacts de l'exemple 10

Exemple 11 : Ensuite, on a une temporisation temporaire. C'est le schéma à contacts de l'exemple 5.

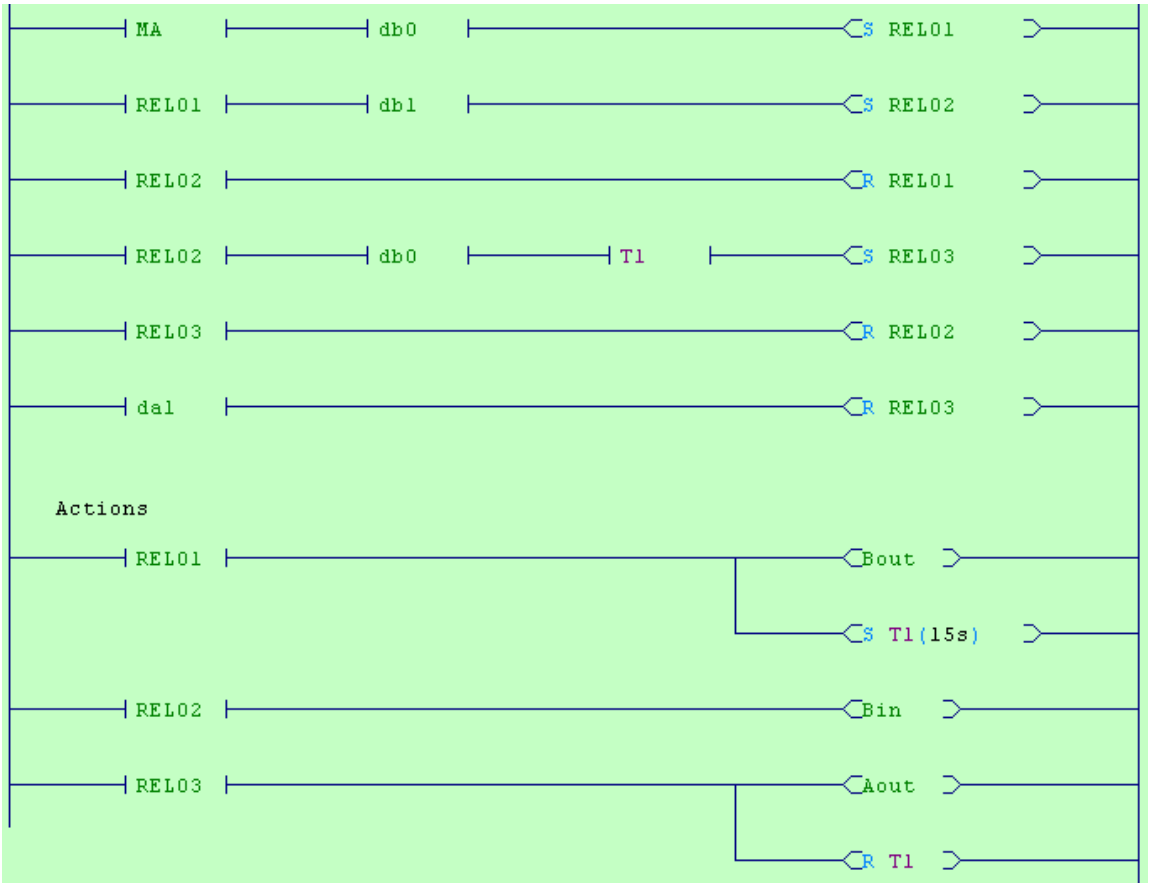


Figure 3.79 : Schéma à contacts de l'exemple 11

Exemple 12 : Ensuite, on a une temporisation temporaire. C'est le schéma à contacts de l'exemple 7.

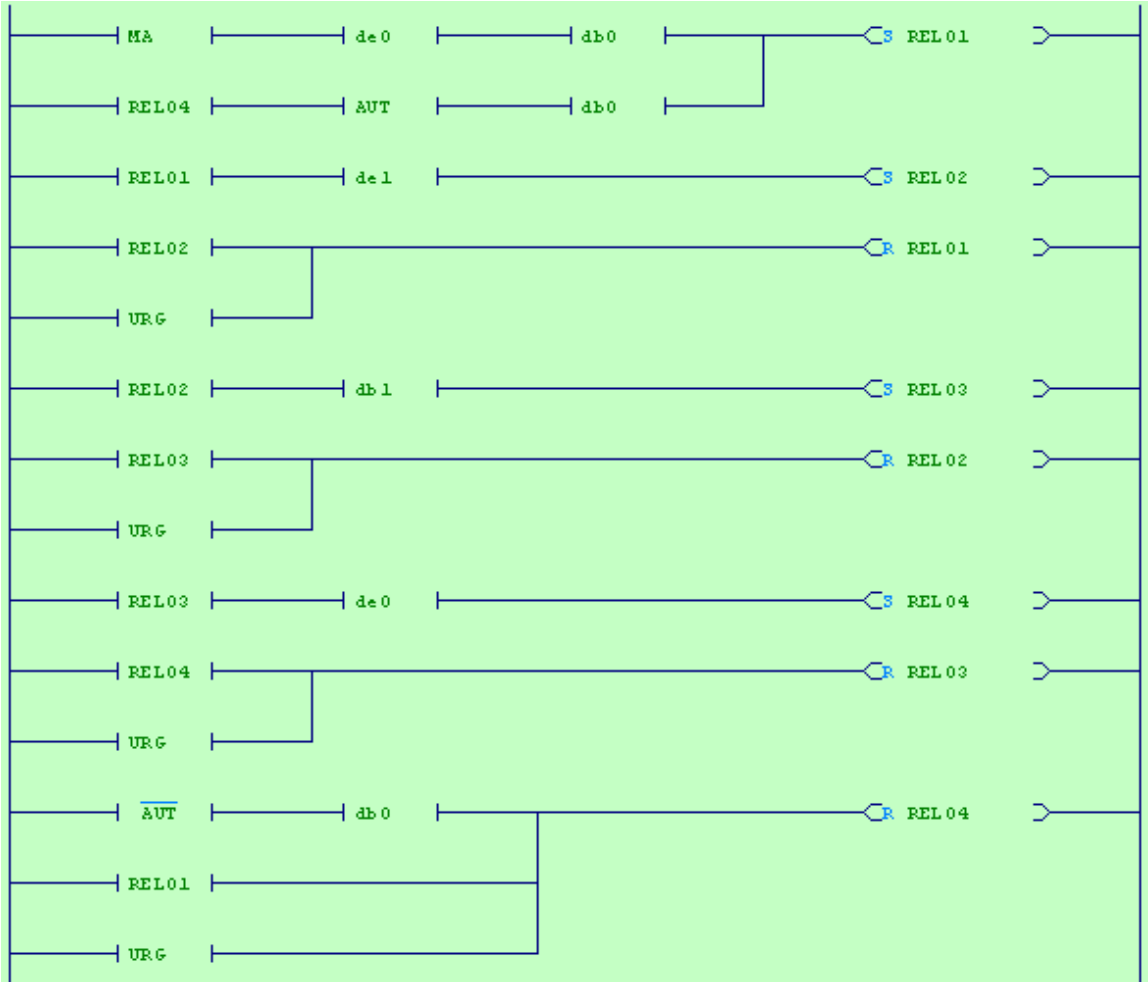


Figure 3.80 : Schéma à contacts de l'exemple 12, première partie

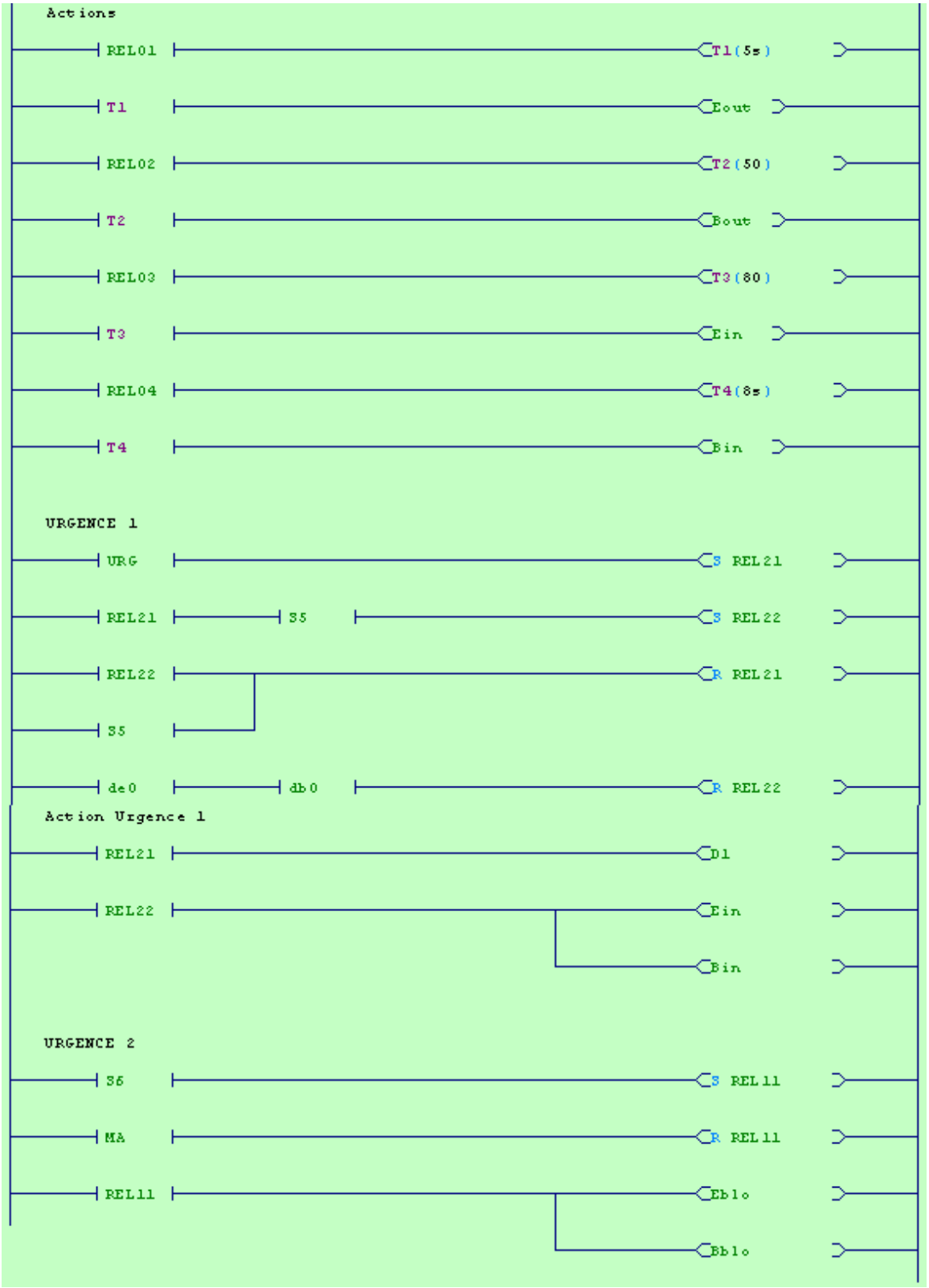


Figure 3.81 : Schéma à contacts de l'exemple 12, deuxième partie

5. Application

1- démarrage direct d'un moteur asynchrone triphasé

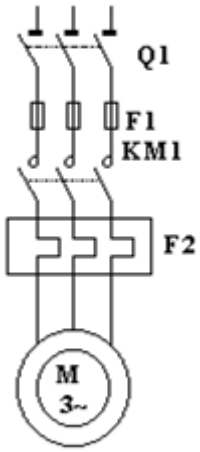


Figure 3.82 : schéma du circuit de puissance

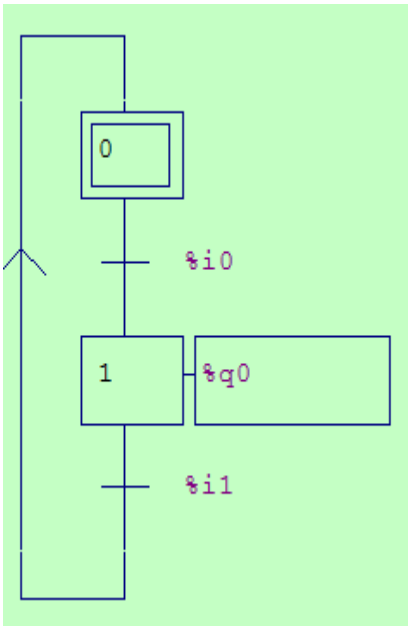


Figure 3.83 : GRAFCET du circuit de puissance

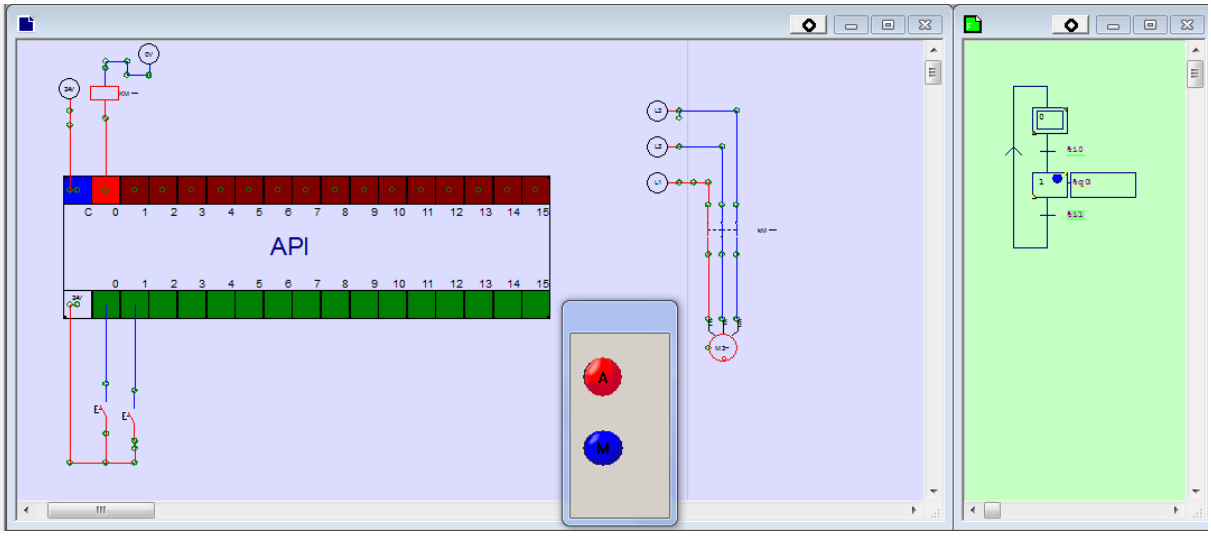


Figure 3.84 : simulation démarrage direct par Automgen

2- démarrage direct à 2 sens de rotation d'un moteur asynchrone triphasé

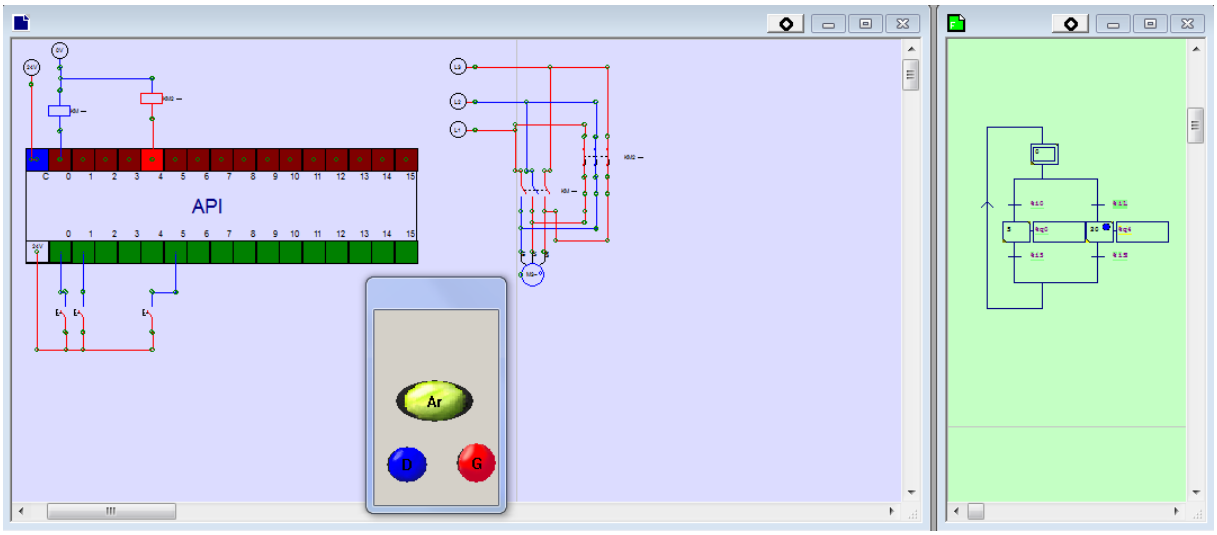
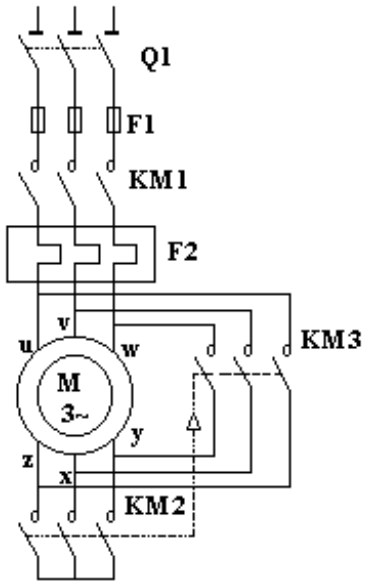


Figure 3.85: simulation démarrage direct à 2 sens de rotation par Automgen

3-Démarrage étoile triangle d'un moteur asynchrone triphasé



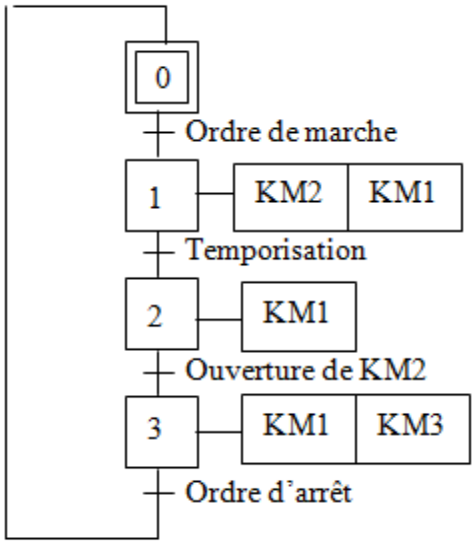


Figure 3.86 : schéma du circuit de puissance **Figure 3.87** : GRAFCET du circuit de puissance

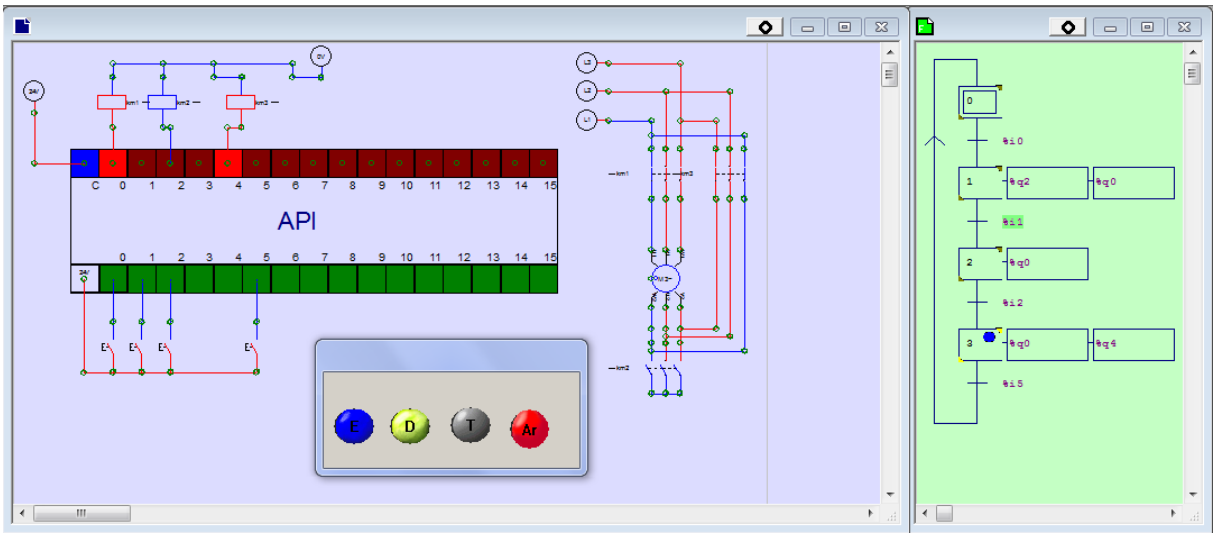


Figure 3.88 : simulation démarrage étoile triangle par Automgen

4-Démarrage statorique d'un moteur asynchrone triphasé

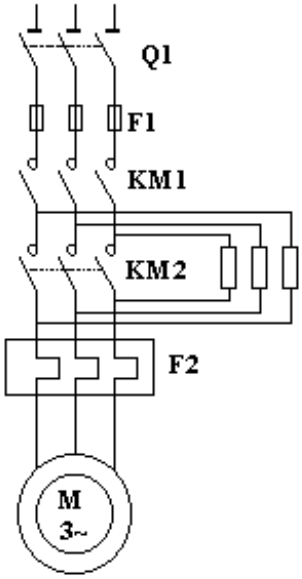


Figure 3.89 : schéma du circuit de puissance

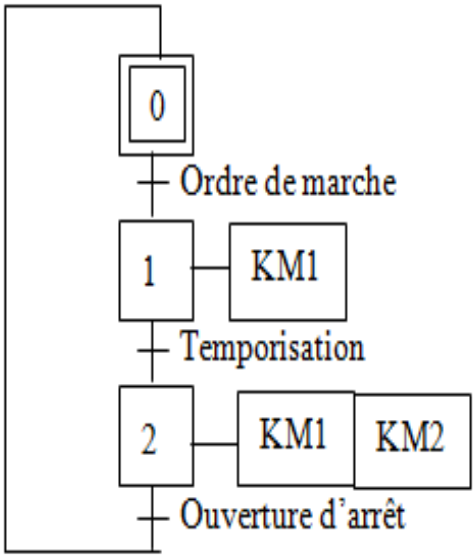


Figure 3.90 : GRAFCET du circuit de puissance

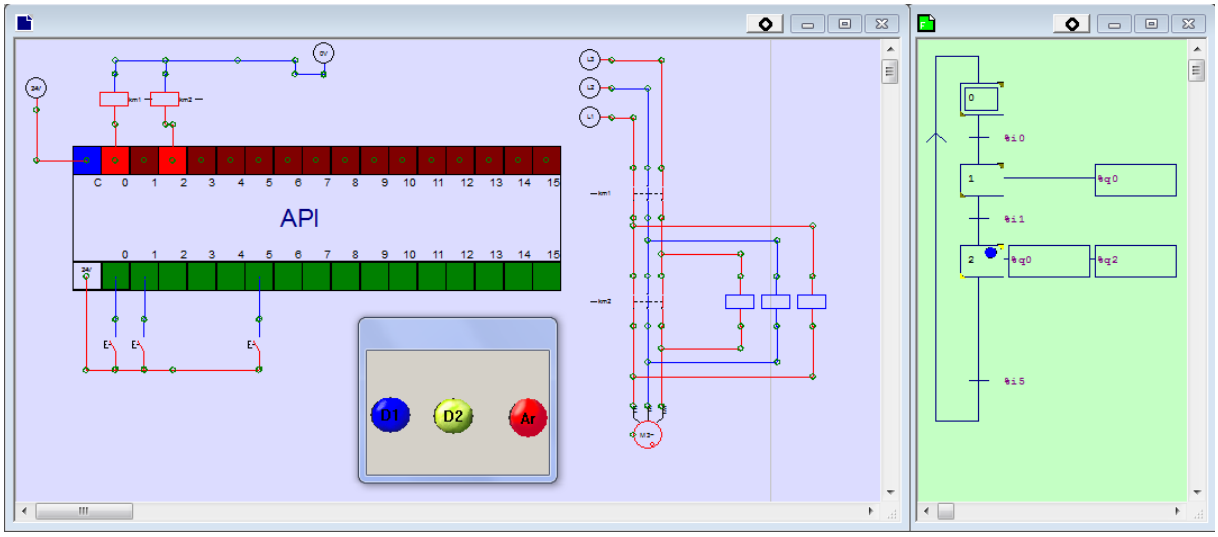


Figure 3.91 : simulation démarrage statorique d'un moteur par Automgen

5- Démarrage Tension réduite par auto-transformateur d'un moteur asynchrone triphasé

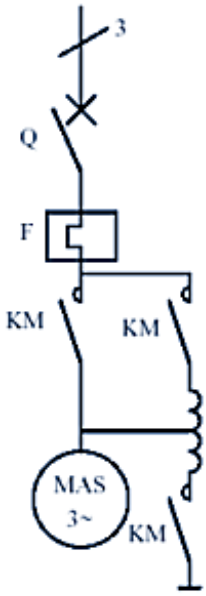


Figure 3.92 : schéma du circuit de puissance

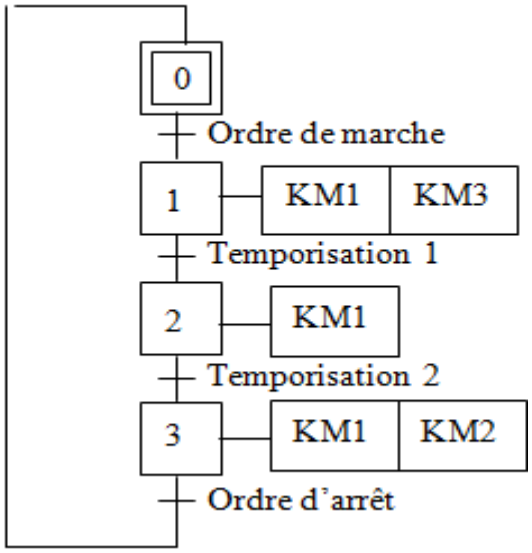


Figure 3.93 : GRAFCET du circuit de puissance

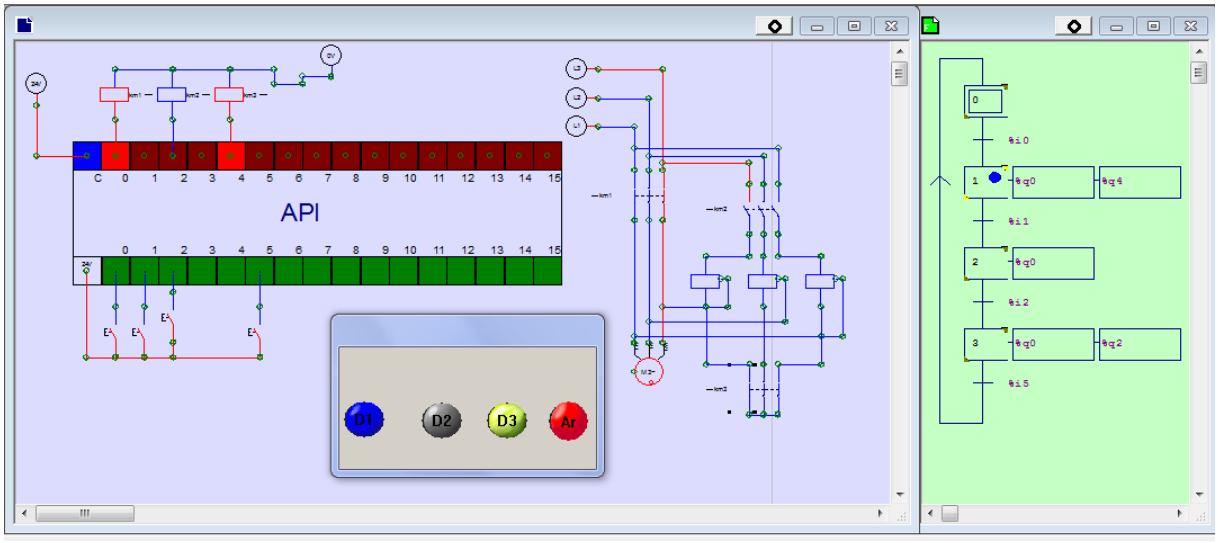


Figure 3.94 : simulation démarrage Tension réduite par auto-transformateur d'un moteur par Automgen

6-Simulation d un système MELANGEUR de peinture

Figure 3.95 : MELANGEUR de peinture

Description du système

Cette partie opérative simule un processus de mélange de peintures. L'objectif est de mélanger les trois couleurs primaires (rouge, vert et bleu) pour obtenir la couleur souhaitée.

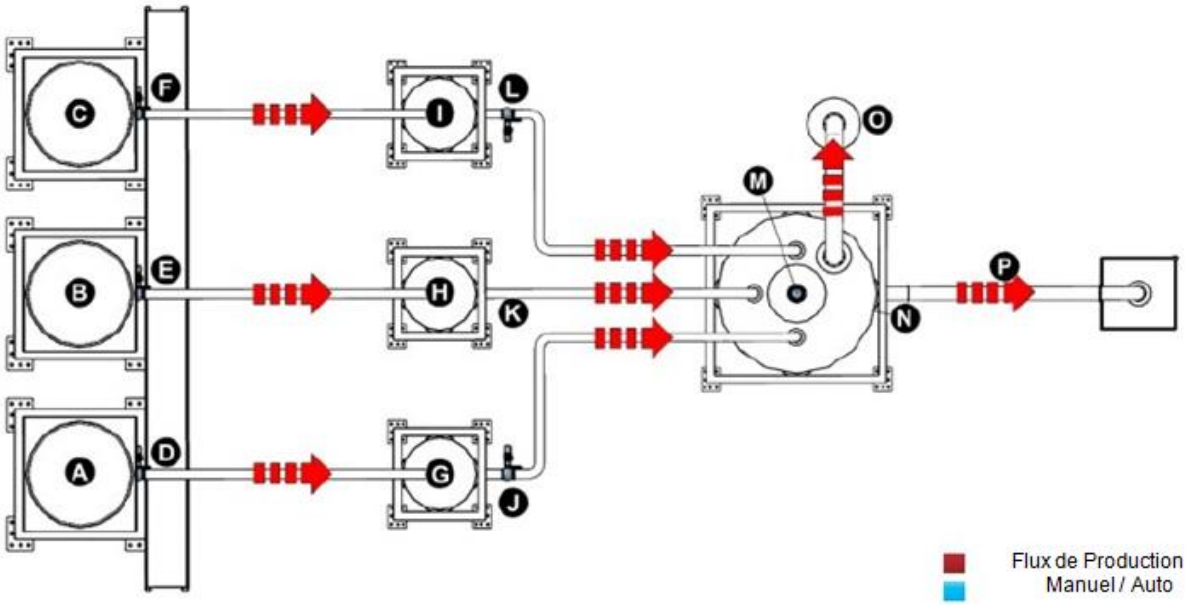


Figure 3.96 :de système

Ce système de mélange est composé de trois réservoirs de peinture, trois cuves de dosage et d'une cuve pour faire le mélange.

Les réservoirs de peinture (A, B et C) contiennent respectivement de la peinture rouge, verte et bleue. Les vannes (D, E, F) permettent de remplir les cuves de dosage (G, H, I) à partir des réservoirs. Chaque cuve de dosage a trois niveaux de mesure. La peinture des cuves de dosage est envoyée à la cuve de mélange (M) via les vannes (J, K, L). Si le volume de peinture ainsi envoyé est plus important que la capacité de la cuve de mélange, le surplus passe par le tuyau de trop plein (O). Le mélange doit durer un minimum de cinq secondes. La peinture obtenue est déchargée par la vanne (N) dans le tuyau de sortie (P).

Les différentes couleurs obtenues en fonction des capteurs de niveaux

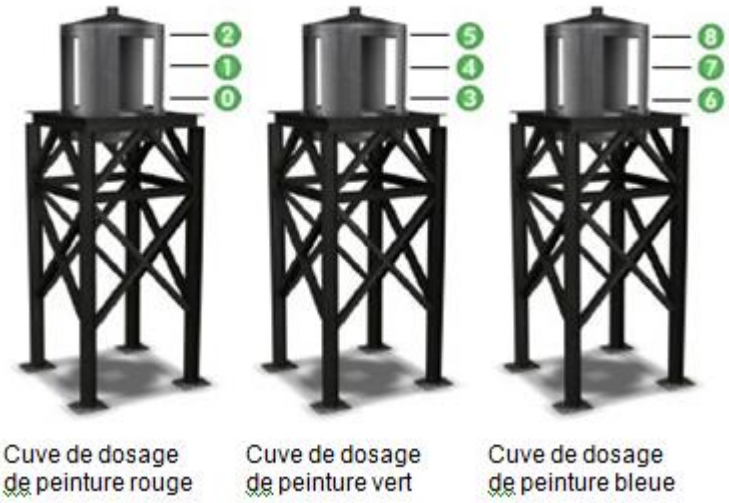


Figure 3.97 : Cuves des dosages

Capteurs de Niveau		
1	4	7
1	5	8
1	4	8
2	4	8
1		8
2	4	7
2		7
1		7
1		
2	5	7
1	4	
2	4	
1	5	7
	5	7
	4	
1	5	
	4	8
	4	7
		7

Figure 3.98 : Capteurs des niveaux

Capteurs

Capteurs	Description
0	Capteur niveau bas de la cuve de dosage de peinture rouge (cuve vide)
1	Capteur niveau milieu de la cuve de dosage de peinture rouge
2	Capteur niveau haut de la cuve de dosage de peinture rouge (cuve pleine)
3	Capteur niveau bas de la cuve de dosage de peinture verte (cuve vide)
4	Capteur niveau milieu de la cuve de dosage de peinture verte
5	Capteur niveau haut de la cuve de dosage de peinture verte (cuve pleine)
6	Capteur niveau bas de la cuve de dosage de peinture bleue (cuve vide)
7	Capteur niveau milieu de la cuve de dosage de peinture bleue
8	Capteur niveau haut de la cuve de dosage de peinture bleue (cuve pleine)
9	Capteur niveau bas de la cuve de mélange
10	Capteur niveau haut de la cuve de mélange

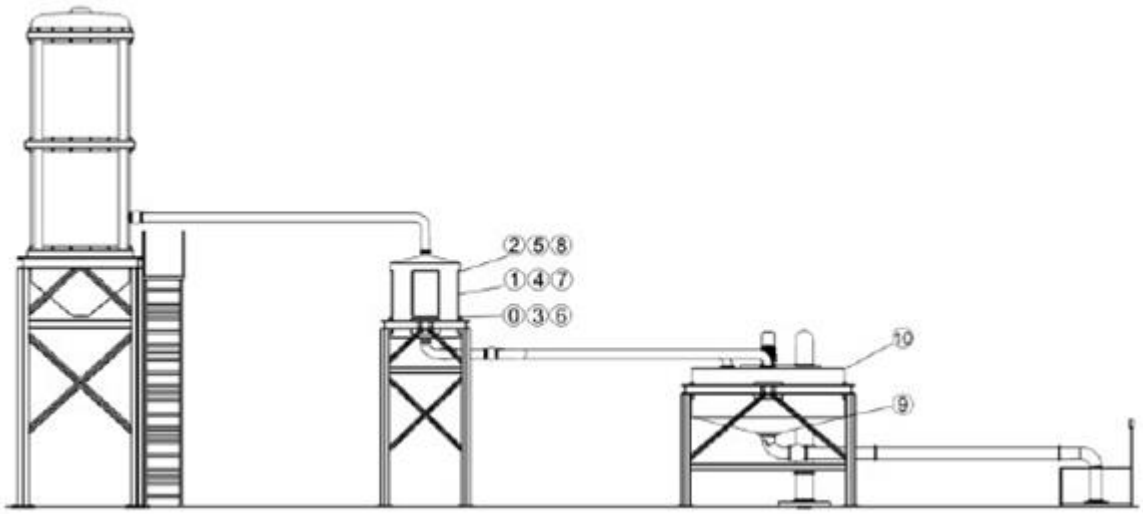


Figure 3.99 :1- position des Capteurs

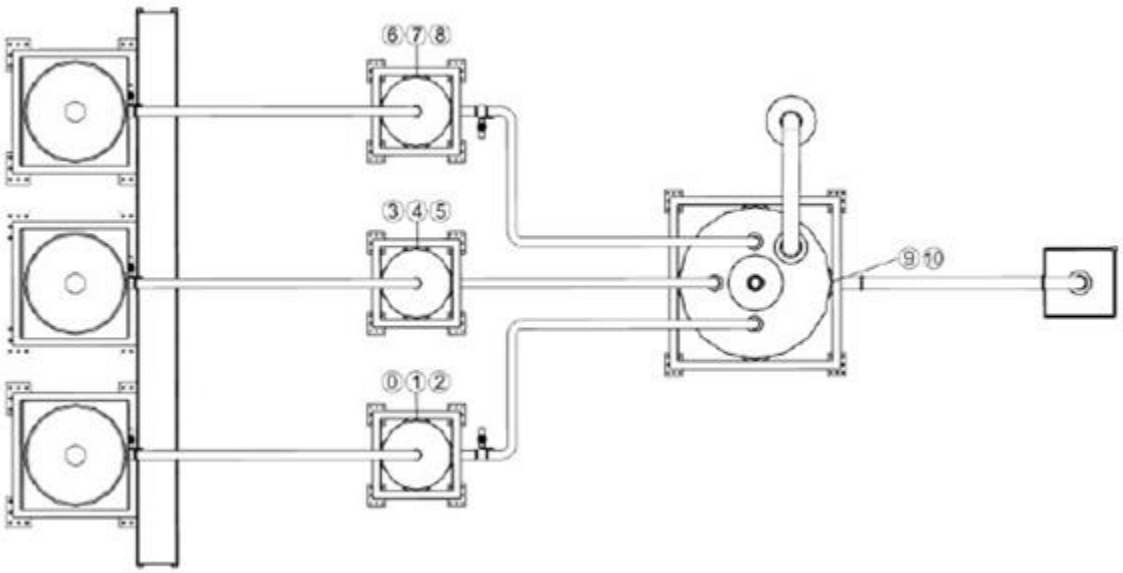


Figure 3.99 :2- position des Capteurs

Actionneur

Actionneur	Description
0	Vanne de vidange du réservoir de peinture rouge
1	Vanne de vidange de la cuve de dosage de peinture rouge
2	Vanne de vidange du réservoir de peinture verte
3	Vanne de vidange de la cuve de dosage de peinture verte
4	Vanne de vidange du réservoir de peinture bleue
5	Vanne de vidange de la cuve de dosage de peinture bleue
6	Mélangeur
7	Vanne de vidange de la cuve de mélange

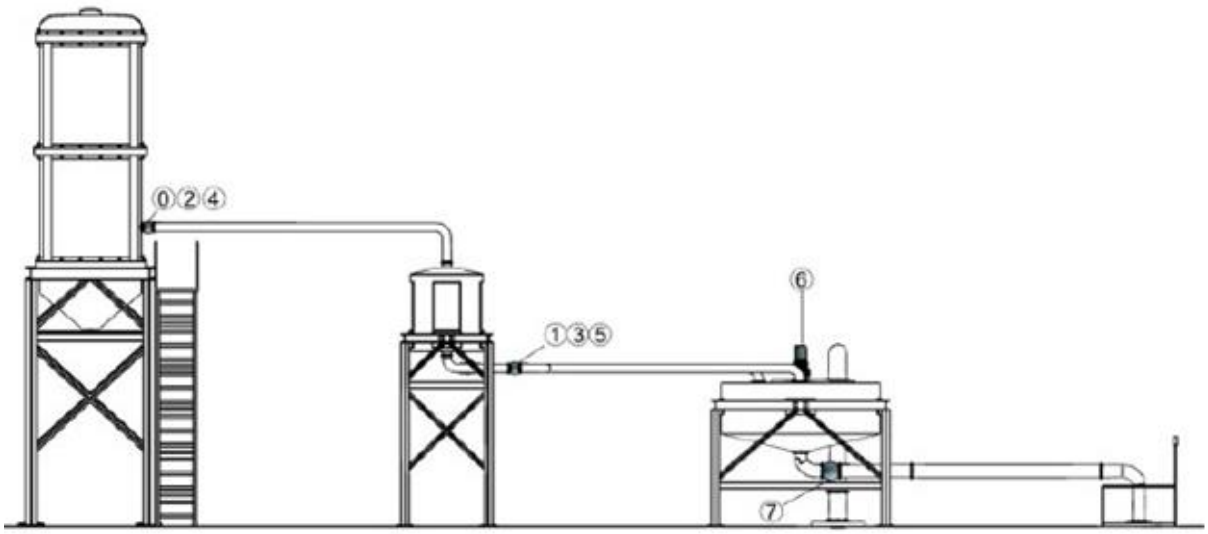


Figure 3.100:1- position des Actionneurs

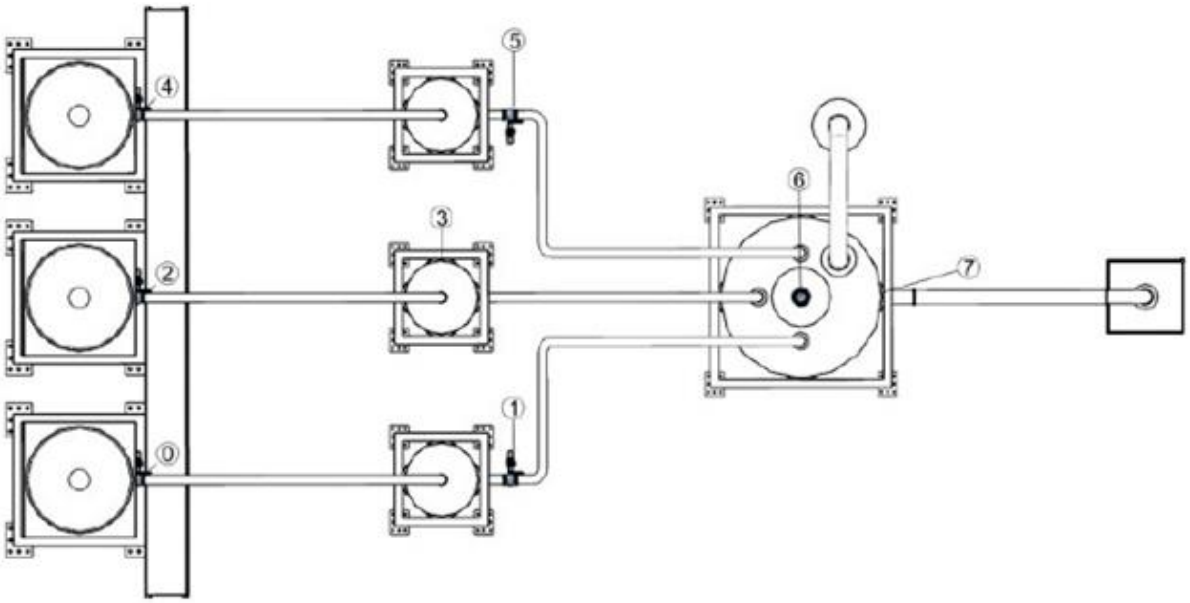


Figure 3.100 :2- position des Actionneurs

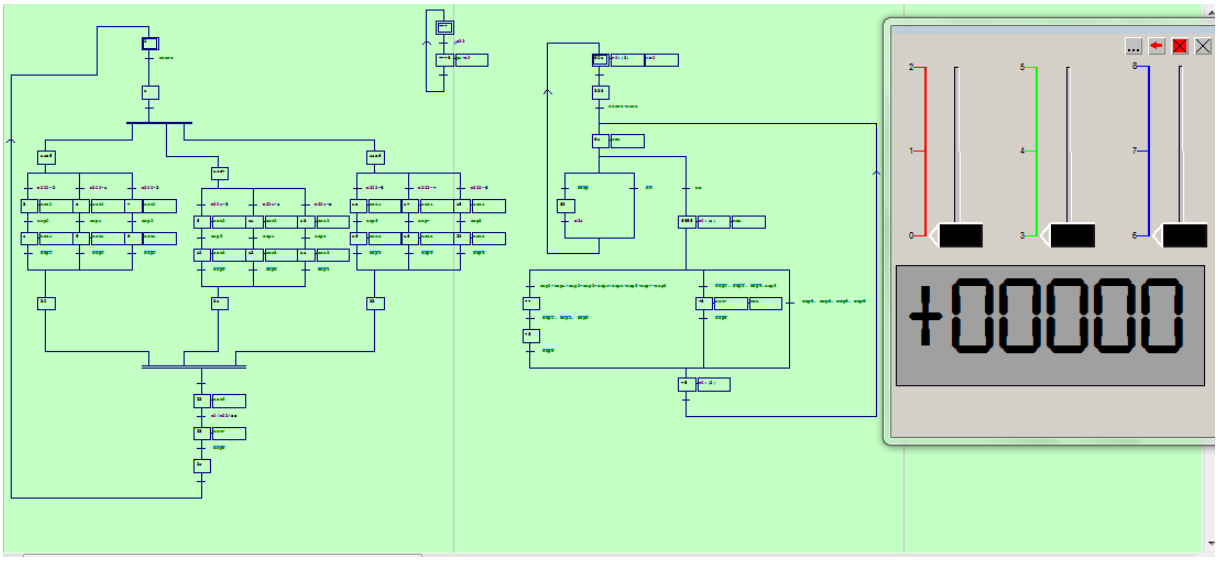


Figure 3.101 : Simulation d un système MELANGEUR de peinture par Automgen

Conclusion :

Un automate programmable (automgen8.9) est un ordinateur numérique industriel utilisé pour le contrôle des dispositifs électromécaniques telles que le contrôle de machines pompes, industrielles, éclairage, etc. Un automate sera connecté aux entrées. L'automate traite les entrées en temps réel basé sur la logique stockée dans sa mémoire et peut ainsi activer ou désactiver des sorties. L'automate est simplement un dispositif électronique c'est une application des microcontrôleurs.

Conclusion Générale

Ce travail nous a permis d'approfondir nos connaissance dans le domaine d'électrotechnique en générale et dans l'automatisme en particulier et de nous familiariser avec les automates programmables industriels .Dans ce travail nous avons étudié et réaliser la modélisation du fonctionnement d'une section pour la préparation du peinture, ensuite nous avons élaboré un programme de commande et de contrôle de la section à l'aide d'un automate programmable automgen8.9,ainsi qu'une supervision de processus à l'aide ITS PLC . Une description du système à automatisé et l'élaboration de l'analyse fonctionnelle de la section et son GRAFCET nous facilitera la tâche pour le bon choix de l'automate et logiciels associés, ainsi que l'élaboration de son programme et sa supervision.

Une étude détaillée de L'automatisation de la section de mélangeur améliore la sécurité de l'opérateur, élimine l'effort physique, augmente la précision et la rapidité de la tâche réalisée.

Donc Le développement scientifique a laissé sa trace sur les systèmes de production donnant naissance au Système Automatisé de Production, qui s'avère être plus ou moins un remède au paradoxe des paramètres coût-qualité visés généralement par la gestion de production (Optimisation du coût, qualité et délai).

En perspective de ce travail, nous souhaiterons faire une étude d'interface humain/machine qui a une animation sur les postes de contrôle, ainsi que l'observation avec un logiciel nommé PCP (Process Control Portal)

Enfin, on termine notre travail par une conclusion générale.

RÉFÉRENCES BIBLIOGRAPHIQUES

Références Bibliographiques

- [1] En anglais : PLC (programmable logic Controller)
- [2] En anglais : « scanning »
- [3] En anglais : Sequential Function Chart (SFC).
- [4] Programmation d'automate ‚Startup' avec STEP 7
- [5] « Structure d'un système automatisé », http://foxi31.ovh.org/dl/2/ISI/04)%20Structure%20d'un%20systeme%20automatise.p df.
- [6]« Les automates programmables », http://www.groupeisf.net/Automatismes/Automatesprogrammables/API_ATTOL/Bas es_automatismes/an9_seq1_Place_et_role_de_l_API.ppt.
- [7]L'Automate Programmable Industriel, http://www.fichier- pdf.fr/2011/03/17/api/api.pdf
- [8] M SERREAU, « Les automates programmables industriels », http://www.larmand.fr/fichiers/Ancien_site/enseigne/ressources/techno/bourse%20co urs/COURS/automate%20programmable%20industriel%20introduction.pdf.
- [9] Dr.Doghmane « cours sur programmation step7 » UBM ANNABA
- [10] Pierre Duysinx, Geoffray Hutsemekers, Henri Lecocq "
- [11] http://www.fichier-pdf.fr/2011/03/17/api/api.pdf
- [12] AUTOMGEN 8.9 et automate TSX47/20 dans le contexte du laboratoire de génie mécanique
- [13] Automatisation et robotisation de la prodution " université de liège 2009-2010.

Résumé :

Dans ce travail de fin d'étude Master en Automatique, l'objectif principal de ce mémoire était l'étude et l'automatisions d'un mélangeur de peinture pour l'optimisation des pertes de temps et d'énergie. La validation du système de choix automatique a été fait par le simulateur d'API «automgen8.9 » et une interface sur ITS PLC. La description et le principe de fonctionnement et la structure d'un système d'automatisme industriel, ont été introduits dans ce mémoire, afin de permettre de comprendre la procédure de la réalisation des différentes parties d'un système automatisé.

***Mots clés** : AUTOGEN8.9, STEP7, ITS PLC, GRAFCET*

ملخص:

الهدف الرئيسي لهذه الأطروحة هو دراسة وأتمتة خلاطة الطلاء لتحسين الوقت وفقدان الطاقة. تم التحقق من صحة نظام الاختيار التلقائي بواسطة محاكي PLC "automgen8.9" وواجهة على ITS PLC. تم تقديم الوصف ومبدأ التشغيل وهيكل نظام الأتمتة الصناعية في هذه المذكرات ، من أجل فهم إجراءات تحقيق الأجزاء المختلفة للنظام الآلي

الكلمات المفتاحية:

AUTOGEN8.9, STEP7, ITS PLC, GRAFCET

Abstract :

In this Master's working memory in Automatic, the main objective of this working memory was the study and automation of a paint mixer for the optimization of time and energy losses. The validation of the automatic choice system was done by the PLC simulator "automgen8.9" and an interface on ITS PLC. The description and the operating principle and the structure of an industrial automation system, have been introduced in this memoir, in order to understand the procedure of the realization of the different parts of an automated system.

Keywords: AUTOGEN8.9, STEP7, ITS PLC, GRAFCET