الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DEL'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF DE M'SILA
FACULTE DE TECHNOLOGIE
DEPARTEMENTDE GENIE ELECTRIQUE

وزارة التعليم العالي و البحث العلمي
جامعة محمد بوضياف بالمسيلة
كلية التكنولوجيا
قسم الهندسة الكهربائية

الرقم: 338/2023 المسيلة في: 19 نوفمبر 2023

مستخلص اللجنة العلمية للقسم بتاريخ: 14-11-2023

بخصوص مطبوعة الدروس

بخصوص مطبوعة الدروس المنجزة من طرف الأستاذ عبدو عبد الحق أستاذ محاضر قسم "أ" بجامعة باتنة تحت عنوان: « Microprocesseur et API »: فقد اطلعت اللجنة على التقارير الواردة من طرف لجنة الخبراء المكونة من الأستاذ بوقرة عبد الرحمان أستاذ بجامعة محمد بوضياف بالمسيلة ، الأستاذ غلاب محمد زين العابدين أستاذ محاضر "أ" بجامعة محمد بوضياف بالمسيلة و الأستاذة بن عالية ليلى أستاذة محاضرة –أ- بجامعة مصطفى بن بولعيد باتنة 2 والتي كانت كلها ايجابية، لهذا فان اللجنة لا ترى مانعا أن تتخذه سندا في تدريس طلبة السنة الثانية ماستر كهروميكانيك في ميدان علوم و تكنولوجيا و أن تعتمد في أي تقييم للمسار العلمي للأستاذ المعني.

رئيس اللجنة العلمية
أ.د بوقرة عبد الرحمان

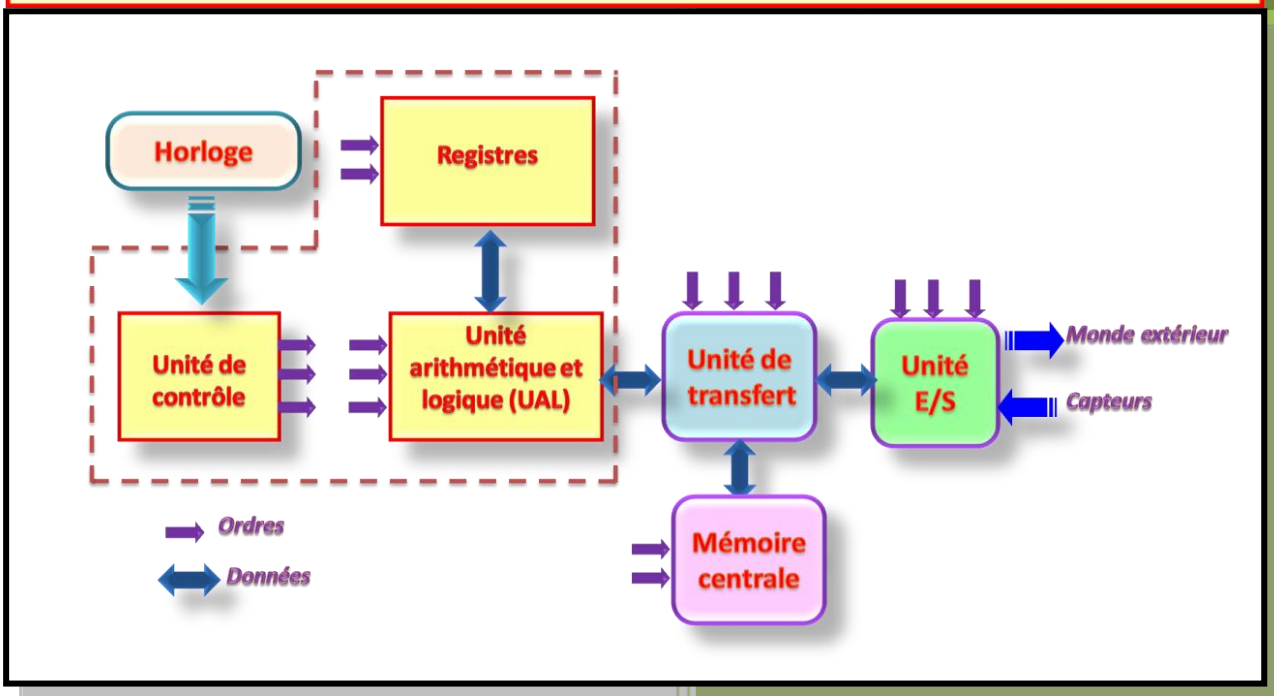
جامعة محمد بوضياف المسيلة
رئيس اللجنة العلمية
قسم الهندسة الكهربائية
كلية التكنولوجيا

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

Université Mohamed BOUDIAF - M'sila

Faculté de Technologie
Département de Génie Electrique

Support de Cours
Microprocesseur et API



Polycopie de Cours Expertisé par :

Nom & prénom	Grade	Etablissement
BENALIA Leila	Prof	Université Batna 2
BOUGUERRA Abderrahmen	Prof	Université de Msila
GHELLAB *Mohamed Zinelabidine*	MCA	Université de Msila

Préparé par :

Dr . Abdelhak ABDOU

Support de cours destiné aux étudiants :
2ème année Master électromécanique
Spécialité : électromécanique

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

Université Mohamed BOUDIAF - M'sila

Faculté de Technologie
Département de Génie Electrique

Support de cours

Microprocesseur et API

Support de cours destiné aux étudiants
2ème année Master : filière électromécanique
Spécialité : électromécanique

Préparé *par :*

Dr Abdelhak ABDOU

abdelhak.abdou@univ-msila.dz

Ce document constitue un support du cours
et ne prétend donc ni à l'originalité, ni à l'exhaustivité.
Ces notes doivent beaucoup aux emprunts faits à de nombreux ouvrages
et à différents travaux de collègues.

Contenu de la matière

Semestre : 3
Unité d'enseignement : UEF 2.1 2
Matière 1 : *Microprocesseur et API*
VHS : 37h30 (Cours: 1h30, T.P : 1h00)
Crédits : 4
Coefficient : 2

Objectifs de l'enseignement:
Connaître le fonctionnement et la mise en œuvre des Microprocesseurs et des Automates Programmables Industriels (API) en vue de développer des circuits de commande.

Connaissances préalables recommandées:
Logiques combinatoire et séquentielle, automatismes.

Contenu de la matière:

Partie 1. Microprocesseurs (07 semaines)
- Architecture
- Microprocesseurs à usage général
- Microcontrôleurs
- Mémoires
- Dispositifs d'entrées/sorties
- Modes d'échanges d'informations
- Microcontrôleurs
- Processeurs de traitement numérique du signal
- Programmation
- Exemples de processeurs disponibles sur le marché

Partie 2. Automates Programmables Industriels (API) (08 semaines)
- Architecture des API : Organisation, entrée-sortie, mémoire, Bus.
- Choix et câblages des API : caractéristiques, environnement, Evaluation
- Logiciels de programmation des API : GRAFCET, Langages de base, de calcul et séquentiel,
- Applications :.

Table des matières

Table des Figures

Partie I Microprocesseur

Partie II API

Avant propos

Description générale

Ce cours est destiné aux étudiants de niveau Master 2 électromécanique option électromécanique semestre 3. Il s'adresse également aux personnes voulant s'initier ou avoir un aperçu général sur systèmes a base de microprocesseur et les systèmes commandés par Automate Programmable Industriel.

Objectifs

Ce cours à pour objectif de :

- Découvrir l'architecture et le fonctionnement du µ-processeur

Les objectifs de ce cours sont :

- Découvrir l'architecture et le fonctionnement du µ-processeur et présenter les notions de base nécessaires à la compréhension des systèmes utilisant des microprocesseurs.
- Être capable de mettre en œuvre des applications d'automatisation conçue autour d'automates programmables industriels.

Plan du cours

Dans notre environnement quotidien, on utilise de plus en plus des systèmes automatisés ou à base de microprocesseur dont la complexité exige une démarche d'étude structurée fondée sur la théorie de ces systèmes.

Pour aborder de tels systèmes, il faut :

- Un minimum de connaissances ou une culture technologique de base des systèmes automatisés;
- Et des compétences pluridisciplinaires impliquant une compréhension approfondie des différents éléments qui composent les systèmes électroniques

Ce support de cours, dans sa structure, suit le cheminement depuis les généralités sur les systèmes automatisés et l'informatique industrielle , en passant par les systèmes basés sur l'utilisation des circuit à base de microprocesseur.

La structure de ce support de cours est conforme aux directives et programmes officiels. Il est axé principalement sur deux grandes parties :

Partie 1. : Généralités sur les systèmes microprocesseurs et microcontrôleurs

Partie 2. : Automates Programmables Industriels (API).

Introduction générale

L'apparition des microprocesseurs date du début des années 1970. A cette époque, deux événements favorables sont apparus :

- *le concept de "LSI (Large Scale Integration)" permettant d'intégrer plusieurs milliers de portes sur un même substrat.*
- *l'arrivée à maturité de la technologie MOS caractérisée par sa faible consommation.*

La conjugaison de ces événements a permis de regrouper une unité centrale d'ordinateur dans un seul circuit intégré appelé "microprocesseur". Depuis, une multitude de composants de ce type sont apparus au sein de familles provenant essentiellement de grands constructeurs américains : Intel, Motorola, Advanced Micro Devices (AMD), Texas Instruments,... et japonais : NEC, Mitsubishi,...Grâce aux progrès de l'intégration, l'augmentation des performances a porté sur :

- *la vitesse de fonctionnement.*
- *la largeur des mots traités (8, 16, 32, 64 bits).*
- *le nombre et la complexité des opérations réalisables.*

L'intégration a également permis de rassembler le microprocesseur et les éléments associés (mémoire, organes d'entrée-sortie,...) au sein d'un seul circuit appelé "microcontrôleur". Ce type de composant s'est répandu dans un très grand nombre de domaines (télécommunications, télévision, électroménager, hifi...

Dans cette optique, les Automates Programmables Industriels ou API (PLCs Programmable Logic Controllers) sont des appareils électroniques conçus autour de microprocesseurs. Ils sont destinés spécialement pour piloter des processus industriels. Leur fonctionnement diffère de celui des systèmes à microprocesseurs usuels. Leurs emplois permettent de rendre les machines employées autonomes et totalement indépendantes de l'intervention d'opérateurs humains. L'API peut alors être défini comme étant un dispositif électronique programmable adapté à l'automatisation des systèmes de production. L'architecture, la programmation et l'exploitation des API répondent aux exigences de son milieu d'utilisation et au personnel qui va l'exploiter..

L'unité de traitement ou Unité centrale constitue le noyau central de l'automate programmable. Le microprocesseur définit l'élément principal d'une unité centrale. C'est à ce niveau que s'effectuent les traitements et les prises de décisions suivant l'interprétation et l'exécution d'un programme écrit par un utilisateur. Dans le cas général, l'exécution d'un

programme se fait instruction par instruction d'une façon séquentielle ; de la première instruction jusqu'à la dernière. Si on veut qu'un programme soit exécuté d'une façon cyclique continue, la dernière instruction devra être du type saut vers le début du programme principal. Le temps réservé à l'exécution de chacune des instructions est fixé par un nombre de cycles de l'horloge du système.

Partie I *Microprocesseurs*

I.1 Introduction

Un système numérique, intégrant de l'électronique, fait souvent apparaître des fonctions ayant pour rôle le traitement d'informations : opérations arithmétiques (addition, multiplication...) ou logiques (ET, OU...) entre plusieurs signaux d'entrée permettant de générer des signaux de sortie. Ces fonctions peuvent être réalisées par des circuits intégrés analogiques ou logiques. Mais, lorsque le système devient complexe, et qu'il est alors nécessaire de réaliser un ensemble important de traitements d'informations, il devient plus simple de faire appel à une structure à base de microcontrôleur ou microprocesseur. Le développement de ces composants programmables a été rendu possible grâce l'essor considérable qu'a connu la microélectronique et notamment les techniques d'intégration. Cette évolution a permis en 1971, la fabrication du premier microprocesseur par la société INTEL. Ce microprocesseur, le « 4004 », comportait déjà 2300 transistors et fonctionnait avec un bus de données de 4 bits.

Depuis, l'intégration du nombre de transistors dans les microprocesseurs n'a cessé d'évoluer, parallèlement à la puissance de calcul et la rapidité d'exécution. Aujourd'hui, un microprocesseur Pentium IV comporte a peu près 24 millions de transistors et peut traiter des données de 8, 16, 32, 64 bits en même temps. La puissance des microprocesseurs d'aujourd'hui a orientée leur utilisations vers le traitement des informations de masse (Gestion d'une base de donnée, Gestion des périphériques bloc, ...), le calcul scientifique ainsi que tout ce qui est interface homme machine réactif (clavier, souris, écran, ...).

Nous pouvons le constater, le domaine d'application des microprocesseurs reste vaste. C'est pourquoi nous les classons dans la catégorie des composants programmables généralistes, cela signifie qu'ils peuvent tout faire, mais ils ne sont optimisés pour rien.

I.2 Introduction à l'étude des calculateurs

I.2.1 Historique

Un calculateur est une machine effectuant des calculs arithmétiques, algébriques ou logiques.
Exemple : une calculatrice, un ordinateur, un calculateur analogique, etc.
La machine d'Anticythère, un mécanisme d'engrenage capable de calculer la date et l'heure des éclipses solaires et lunaires, est le plus ancien calculateur connu. Les premiers calculateurs étaient mécaniques, ils ont été ensuite électromécaniques, aujourd'hui l'électronique numérique a supplanté toutes les autres technologies.

Le terme calculateur s'applique aussi aux systèmes électroniques qui gèrent les fonctions d'une automobile moderne. Plusieurs types de calculateurs peuvent être présents, chacun spécialisé dans un domaine : gestion du moteur, du freinage, de la traction ou même de l'alarme et de l'air conditionné. Les désignations ECU (Engine Control Unit) ou ECM (Engine Control Module) sont fréquemment employées.

I.2.2 Structure d'un calculateur

L'élément de base d'un calculateur est constitué par l'unité centrale de traitement UCT (CPU : Central Processing Unit en anglais), Fig I. 1..

- L'UCT est constituée :
  - ✓ d'une unité arithmétique et logique UAL (ALU : Arithmetic and Logic Unit) : c'est l'organe de calcul du calculateur ;
  - ✓ de registres : zones de stockage des données de travail de l'UAL (opérandes, résultats intermédiaires) ;
  - ✓ d'une unité de contrôle UC (CU : Control Unit) : elle envoie les ordres (ou commandes) `a tous les autres éléments du calculateur afin d'exécuter un programme.

- La mémoire centrale contient :
  - ✓ le programme à exécuter : suite d'instructions élémentaires ;
  - ✓ les données à traiter.

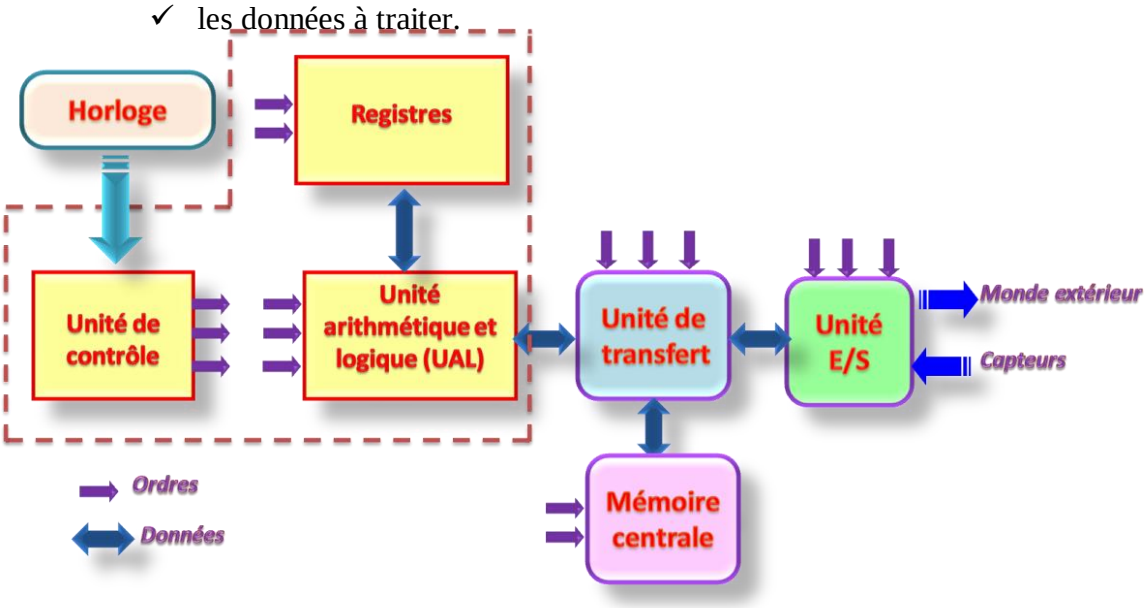


Fig I. 1: Structure d'un calculateur.

I.3 Étude architecturale des microprocesseurs

I.3.1 Définition

Un microprocesseur est un circuit intégré complexe (sous la forme d'un boîtier). Il résulte de l'intégration sur une puce, de fonctions logiques combinatoires (logiques et/ou arithmétique) et séquentielles (registres, compteur, etc…).

Il permet d'interpréter et d'exécuter les instructions, stockées en mémoire, d'un programme.

- il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution.
- il est prend en considération les informations extérieures au système et assurer leur traitement.

Son domaine d'utilisation est donc très vaste.

.

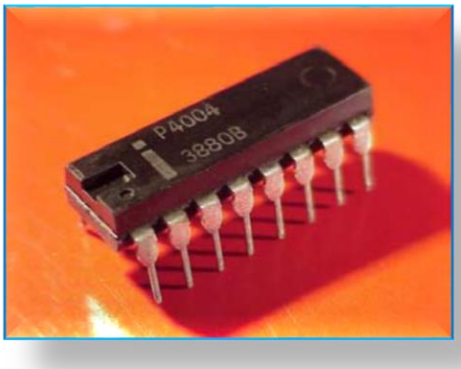


Fig I. 2: Premier microprocesseur commercialisé, un Intel 4004 dans son boîtier à 16 broches,

Les applications des systèmes à base de microprocesseurs sont très variées :

- ✓ ordinateur,
- ✓ console de jeux,
- ✓ calculatrice,
- ✓ télévision,
- ✓ téléphone portable,
- ✓ distributeur automatique d'argent,
- ✓ robotique,
- ✓ automobile.

I.3.2 Architecture de base d'un microprocesseur

Un système à base de microprocesseur est formé des trois éléments, Figure I.3:

- ✓ Un processeur, ou unité centrale de traitement, UCT (Central Processing Unit CPU)

- ✓ Une mémoire (ROM et RAM)
- ✓ Des ports d'entrées/sorties.

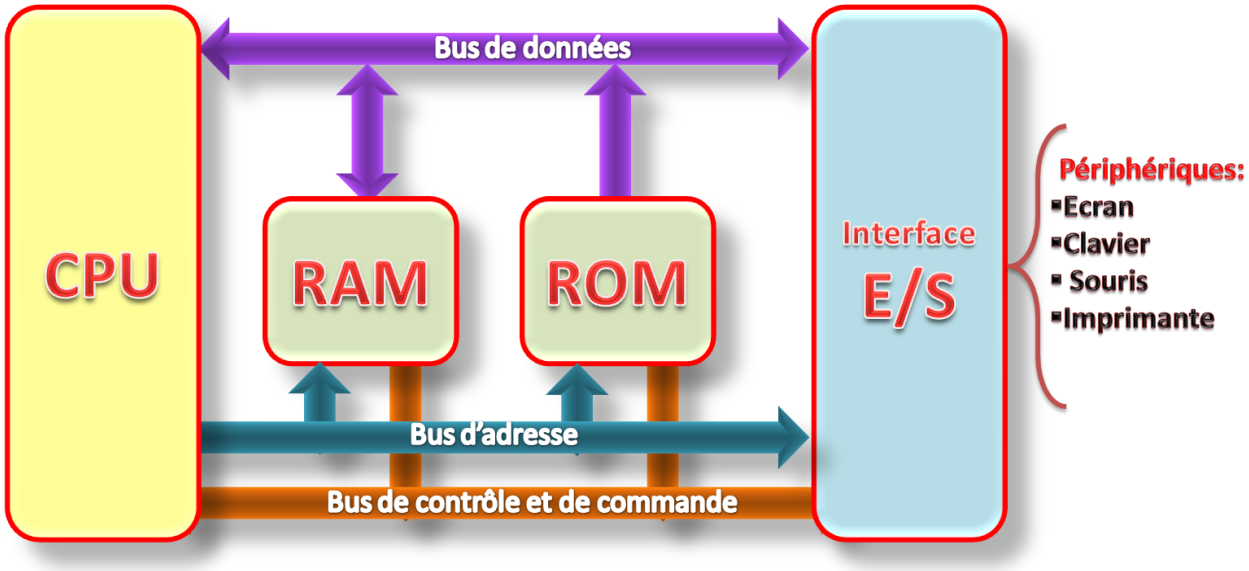


Fig I. 3: Schéma Architectural d'un système à base de microprocesseur.

I.3.2.a. Mémoire

Un circuit intégré de type mémoire permet de stocker et de restituer une très grande quantité d'informations correspondant à N mots de n bits. Une mémoire est constituée par un assemblage de cellules mémoires. Dans chaque cellule, il est possible d'écrire, de conserver et d'extraire un élément d'information. Pour gérer cet ensemble de cellules mémoires, le circuit comprend également des sélecteurs d'adresse, des amplificateurs, des commandes de modes de fonctionnement…

Il existe deux familles de mémoires :

- **les mémoires vives, RAM (Random Access Memory)** : mémoires qui peuvent être lues et écrites en permanence. Elle contient habituellement les données pendant le traitement d'un programme. Elle est effacée en cas de coupure d'énergie, - les mémoires mortes, ROM (Read Only Memory) : Avec ces mémoires, seule la lecture est possible. Les données qu'elle contient sont enregistrées par le concepteur du système. En fonction de leur construction, il existe différents types :

- ✓ PROM, Programmable ROM : ROM programmable,
- ✓ EPROM, Erasable PROM : PROM effaçable,
- ✓ EEPROM, Electric Erasable PROM : PROM effaçable électriquement.

I.3.2.b. Interface

L'interfaçage est l'ensemble matériel qui permet de transférer les données entre le système de traitement de l'information et les périphériques. On distingue les interfaces série et les interfaces parallèles.

I.3.2.c. Périphériques

Les périphériques sont les matériels extérieurs aux systèmes informatiques. On y retrouve :

- ✓ tous les matériels qui servent à la communication de l'homme avec le système informatique : écran, clavier, souris…
- ✓ tous les systèmes de stockage des informations : lecteur de disquettes et CDROM, disque dur, lecteur de bandes magnétiques…
- ✓ tous les appareils qui servent à traduire les données sur papier : imprimante à aiguilles, jet ou bulles d'encre, laser…
- ✓ tous les appareils qui servent à acquérir des informations extérieures : lecteur de cartes magnétiques, scanner, appareil photo numérique, carte d'acquisition son et vidéo…

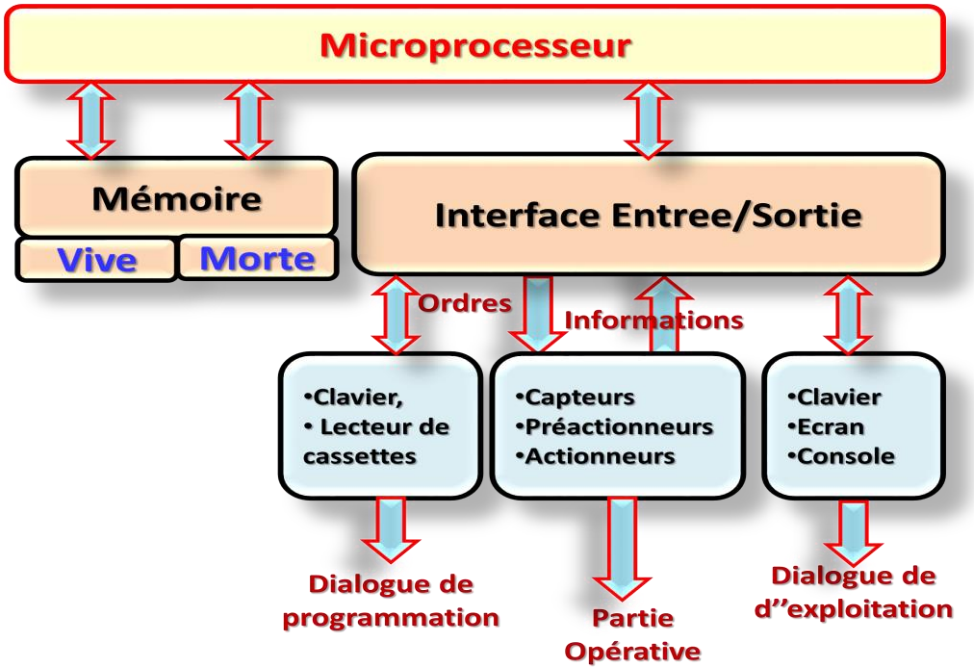


Fig I. 4: Environnement du microprocesseur.

I.3.2.d. Interconnexions

Les trois modules sont interconnectés comme le montre la figure I.3, autour de trois bus : bus de données, bus d'adresses et bus de contrôles et commandes

Bus : Il s'agit de plusieurs pistes électroniques qui sont reliées au microprocesseur. Ces bus assurent la communication interne et externe du microprocesseur.

Traditionnellement, ces éléments sont intégrés dans des circuits distincts. D'où la nécessité de prévoir l'interconnexion de ces composants (bus, câblage, nappes de connexion); ceci va se traduire par une:

- ✓ augmentation de la taille occupée ;
- ✓ augmentation de la consommation énergétique ;
- ✓ augmentation de la chaleur dégagée ;
- ✓ augmentation du coût de fabrication.

I.3.3 Circulation de l'information dans un ordinateur

Le microprocesseur échange des informations avec la mémoire et l'unité d'E/S, sous forme de mots binaires, au moyen d'un ensemble de connexions appelé ***bus***. Un bus permet de transférer des données sous forme ***parallèle***, c'est-`a-dire en faisant circuler *n* bits simultanément.
Les microprocesseurs peuvent ˆêtre classés selon la longueur maximale des mots binaires qu'ils peuvent échanger avec la mémoire et les E/S : microprocesseurs 8 bits, 16 bits, 32 bits,

Le bus peut être décomposé en trois bus distincts :

- le ***bus d'adresses*** permet au microprocesseur de spécifier l'adresse de la case mémoire à lire ou à écrire ;
- le ***bus de données*** permet les transferts entre le microprocesseur et la mémoire ou les E/S ;
- le ***bus de commande*** transmet les ordres de lecture et d'écriture de la mémoire et des E/S.

Remarque :Les bus de données et de commande sont ***bidirectionnels***, le bus d'adresse est ***unidirectionnel*** : seul le microprocesseur peut d'délivrer des adresses, comme l'indique la figure (I.3) :

I.4 Architecture d'un microprocesseur

Le microprocesseur (C.P.U. : Central Processing Unit) est un circuit intégré complexe (ensemble de millions de transistors) capable d'effectuer séquentiellement et automatiquement des suites d'opérations élémentaires (programmes).

Un microprocesseur est qualifié par sa puissance qui est la capacité de traiter un grand nombre d'opérations par seconde sur de grands nombres et en grande quantité. Cette puissance est principalement régie par les critères suivants :

- La longueur des mots : données et instructions (largeur du bus des données).
- Le nombre d'octets que le microprocesseur peut adresser (largeur du bus des adresses).
- La vitesse d'exécution des instructions liée à la fréquence de fonctionnement de l'horloge de cadencement exprimée en MHZ.

Le microprocesseur remplit deux fonctions essentielles : le traitement des données et le contrôle du système.

- Le traitement de données concerne la manipulation des données sous formes de transfert (déplacement), opérations arithmétiques, opérations logiques....

On distingue généralement trois éléments logiques principaux :

  - ✓ Une Unité Arithmétique et Logique (U.A.L.)
  - ✓ Un Accumulateur.
  - ✓ Les Registres

- Le contrôle et la commande du système se traduit par des opérations de décodage et d'exécution des ordres exprimés sous forme d'instruction. Cette unité est constituée principalement de :
  - ✓ Horloge

- ✓ Pointeur d'instruction (Compteur Ordinal)
- ✓ Registre d'instruction
- ✓ Le décodeur
- ✓ Le séquenceur

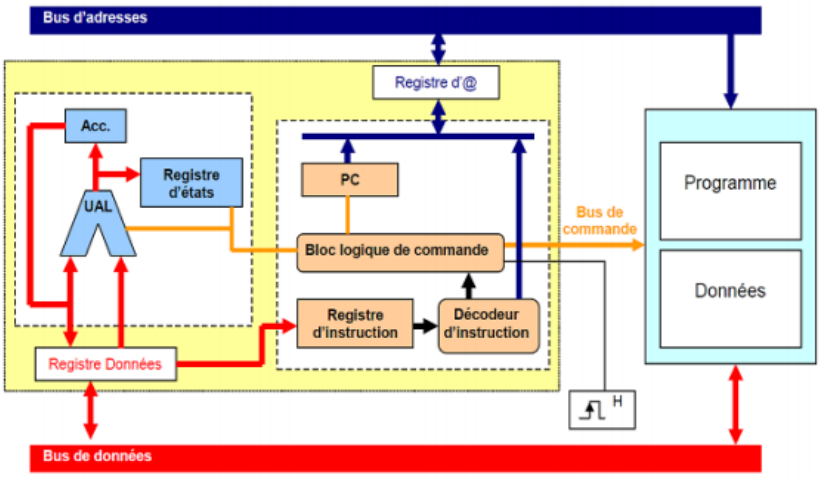


Fig I. 5: Architecture d'un microprocesseur.

I.5 Fonctionnement d'un système à base de microprocesseur

I.5.1 Schéma fonctionnel

Un microprocesseur se présente sous la forme d'un circuit intégré muni d'un nombre important de broches. Exemples :

• Intel 8085, 8086, Zilog Z80 : 40 broches, DIP (Dual In-line Package) ;

• Motorola 68000 : 64 broches, DIP ;

• Intel 80386 : 196 broches, PGA (Pin Grid Array).

Technologies de fabrication : NMOS, PMOS, CMOS.

On peut représenter un microprocesseur par son **schéma fonctionnel** :

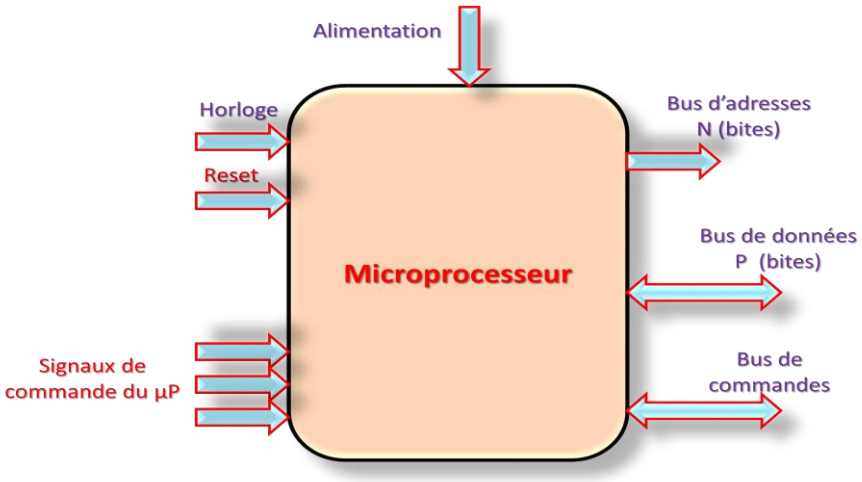


Fig I. 6: Schéma fonctionnel du microprocesseur.

I.5.2 Fonctionnement d'un microprocesseur

Un microprocesseur exécute un ***programme***. Le programme est une suite d'instructions déposées dans la mémoire. Une instruction peut être codée sur ***un ou plusieurs octets***.

Format générale d'une instruction :

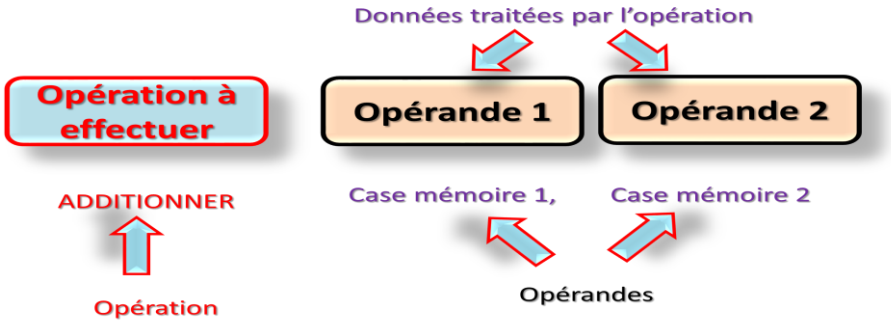


Fig I. 7: Exemple du format générale d'une instruction du microprocesseur.

I.5.3 Rangement en mémoire

Pour exécuter les instructions dans l'ordre confirmé par le programme, le microprocesseur doit savoir à chaque instant l'adresse de la prochaine instruction à exécuter. Le microprocesseur utilise un registre contenant cette information. Ce registre est appelé ***compteur ordinal*** (IP : Instruction Pointer).

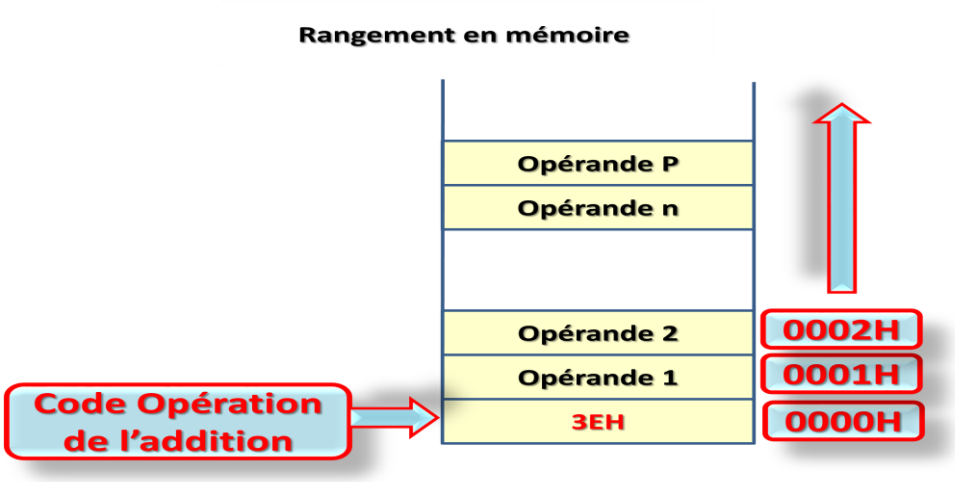


Fig I. 8: Exemple du rangement en mémoire.

Pour exécuter les instructions dans l'ordre établi par le programme, le microprocesseur doit savoir à chaque instant l'adresse de la prochaine instruction à exécuter. Le microprocesseur utilise un registre contenant cette information. Ce registre est appelé pointeur d'instruction (IP : Instruction Pointer) ou compteur d'instructions ou compteur ordinal.

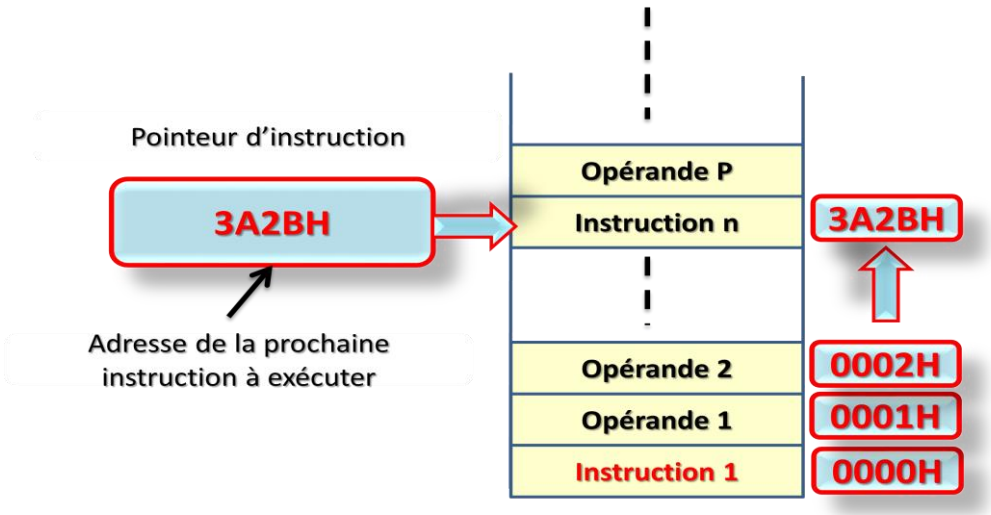


Fig I. 9: Exemple du pointeur d'instruction.

I.5.4 Recherche de l'instruction

Le contenu de PC (compteur ordinal) est placé sur le bus d'adresse (c'est l'unité de commande qui établit la connexion). L'unité de commande (UC) émet un ordre de lecture (READ=RD=1). Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la

case mémoire sélectionnée est disponible sur le bus des données. L'unité de commande charge la donnée dans le registre d'instruction pour décodage. Exemple : Le microprocesseur place le contenu de PC (10000H) sur le bus d'adresse et met RD à 1 (cycle de lecture). La mémoire met sur le bus de données le contenu de sa mémoire n° 10000H (ici 89D9H qui est le code de MOV C,B). Le microprocesseur place dans son registre d'instruction le contenu du bus de données (89D9H). L'unité de commande décode et prépare l'exécution de l'instruction MOV C,B.

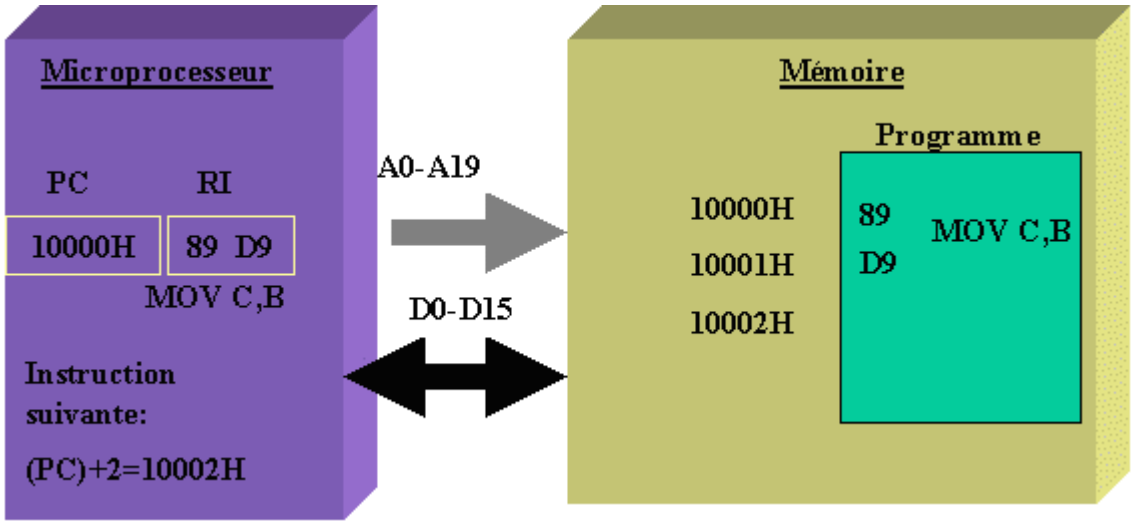


Fig I. 10: Exemple d'une instruction du microprocesseur.

I.6 Étude architecturale des microcontrôleurs

Les microcontrôleurs (en notation abrégée µc) sont apparus suite au progrès considérable de l'intégration des composants.

Un microcontrôleur est un composant réunissant sur un seul et même silicium un micro-processeur, divers dispositifs d'entrées/sorties et de contrôle d'interruptions ainsi que de la mémoire, notamment pour stocker le programme d'application. Dédié au contrôle, il embarque également un certain nombre de périphériques spécifiques des domaines ciblés (bus série, interface parallèle, convertisseur analogique numérique, ...).

Les microcontrôleurs améliorent l'intégration et le coût (lié à la conception et à la réalisation) d'un système à base de microprocesseur en rassemblant ces éléments essentiels dans un seul circuit intégré. On parle alors de "système sur une puce" (en anglais : "System On chip"). Il existe plusieurs familles de microcontrôleurs, se différenciant par la vitesse de leur processeur et par le nombre de périphériques qui les composent. Toutes ces familles ont un point commun c'est de réunir tous les éléments essentiels d'une structure à base de microprocesseur sur une même puce. Voici généralement ce que l'on trouve à l'intérieur d'un tel composant :

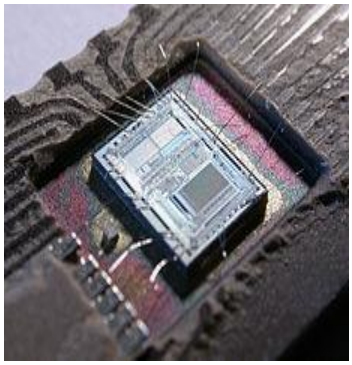
- ✓ un processeur CPU,

- ✓ la mémoire vive (RAM) pour stocker les données et variables ;
- ✓ la mémoire morte (ROM) pour stocker le programme ;
- ✓ une horloge (un oscillateur) pour le cadencement. Il peut être réalisé avec un quartz, un circuit RC;
- ✓ des périphériques, capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :
  - les convertisseurs analogiques-numériques (CAN) (donnent un nombre binaire à partir d'une tension électrique),
  - les convertisseurs numériques-analogiques (CNA) (effectuent l'opération inverse),
  - les générateurs de signaux à modulation de largeur d'impulsion (MLI, ou en anglais, PWM),
  - les compteurs (compteurs d'impulsions d'horloge interne ou d'événements externes),
  - les comparateurs (comparent deux tensions électriques),

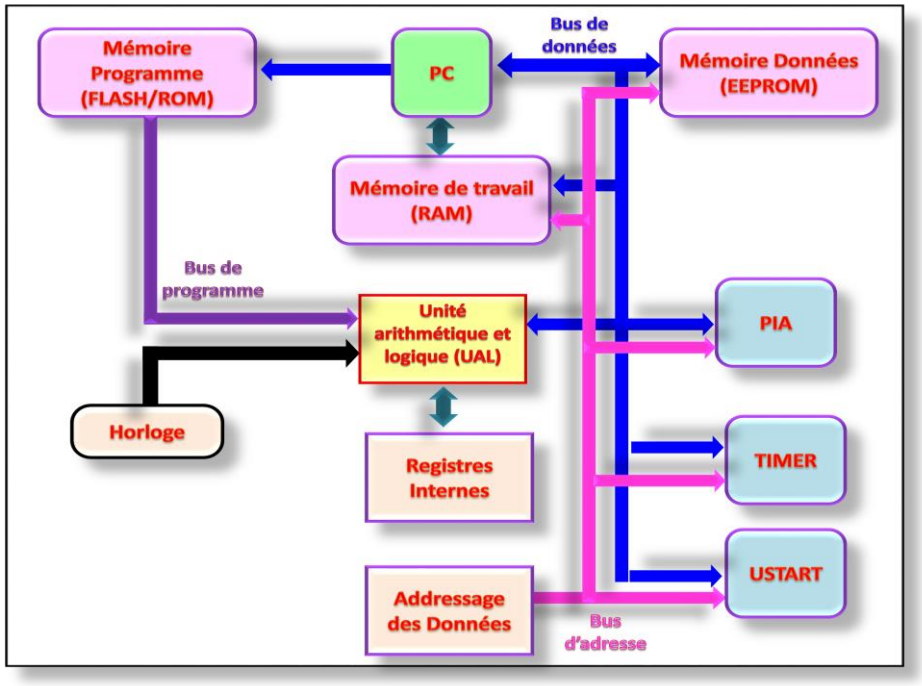
Les microcontrôleurs permettent de diminuer

- ✓ la taille,
- ✓ la consommation électrique
- ✓ le coût des produits.

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc



a. Photo du circuit intégré d'un microcontrôleur



b. Schéma architectural d'un microcontrôleur

Fig I. 11: Structure d'un microcontrôleur.

I.7 . Choix d'un microcontrôleur

Nous n'avons pour l'instant évoqué que des généralités applicables à tous les microcontrôleurs du marché, sans citer de marque précise. En effet, toute la difficulté du choix d'un microcontrôleur pour une application donnée réside dans la sélection du "bon" circuit adapté pour cette application. Le choix du microcontrôleur est surtout dicté par deux critères principaux :

- l'adaptation de son architecture interne aux besoins de l'application (présence de convertisseurs A/N par exemple ou d'un timer disposant d'un mode particulier, …);
- le fait de posséder déjà ou non un système de développement.

En effet, si l'on ne possède rien, on peut se laisser guider par le premier critère en comparant toutefois les investissements de développement à prévoir. Si l'on est déjà équipé, mieux vaut choisir un circuit un peu moins bien adapté, quitte à lui adjoindre des circuits externes, que le circuit qui va bien mais qui impose un changement de système. Pour simplifier un peu ce deuxième dilemme, les fabricants ont essayé de développer non pas des microcontrôleurs isolés mais des familles de circuits, plus ou moins compatibles entre eux tant au niveau de l'architecture qu'au niveau de la programmation et des outils de développement. Il existe plusieurs familles de microcontrôleurs dont les plus connues sont :

- la famille Atmel AT91
- la famille Atmel AVR
- le C167 de Siemens/Infineon
- la famille Hitachi H8
- la famille Intel 8051, qui ne cesse de grandir ; de plus, certains processeurs récents utilisent un cœur 8051, qui est complété par divers périphériques (ports d'E/S, compteurs/temporisateurs, convertisseurs A/N et N/A, chien de garde, superviseur de tension...)
- l'Intel 8085, à l'origine conçu pour être un microprocesseur, a en pratique souvent été utilisé en tant que microcontrôleur

I.8 Fonctionnement d'un µ-contrôleur

Le fonctionnement des microcontrôleurs et des périphériques qui lui seront interfacés est basé sur la connaissance de leurs architectures et les langages utilisés. Ce fonctionnement est résumé dans :

I.8.1 Bus de données

Un bus de données est constitué de connexions électriques reliant les différents composants du microcontrôleur :

a) Le micro-processeur,

b) La mémoire et

c) Les périphériques.

Il leur permet d'échanger des mots binaires.

Le transfert de données est cadencé par un signal d'horloge : à chaque cycle d'horloge, un octet est échangé dans le sens imposé par les lignes de contrôle.

I.8.2 Mémoires du microcontrôleur

Un microcontrôleur contient trois types de mémoires:

a) La mémoire programme ;

b) La mémoire de données non volatiles, conservée même en cas de rupture de l'alimentation ;

c) La mémoire de données volatiles, perdue en cas de coupure d'alimentation

Les informations sont échangées entre la mémoire et les autres composants par un bus de données

I.8.3 Langage machine

Un micro-processeur n'est pas capable de comprendre directement un programme écrit en langage Python ou C. Il ne comprend que du langage machine, c'est-à-dire du code écrit en binaire.

En effet, le micro-processeur ne peut exécuter que les instructions "câblées" dans l'UAL, l'unité de calcul du micro-processeur. Sur un ordinateur classique, il existe un peu plus de mille instructions disponibles

Programmer dans le langage interne du micro-processeur, appelé "Assembleur", est possible, mais plutôt fatigant. La programmation se fait donc généralement dans un langage de plus haut niveau (comme Python ou C), proposant des instructions plus élaborées. Ce type de programme doit être traduit en langage machine pour être exécuté : il s'agit de la compilation.

Pour compiler un programme en langage machine, le logiciel utilisé doit tenir compte des caractéristiques de la machine sur laquelle sera exécuté le programme, en particulier les instructions disponibles et la configuration des registres. S'il existe presque toujours un compilateur C pour n'importe quel microcontrôleur, c'est beaucoup plus rare pour d'autres langages : c'est la raison pour laquelle le langage C est incontournable dans le milieu des microcontrôleurs.

Par ailleurs, le langage machine étant compilé pour un type de microcontrôleur précis, il ne peut pas s'exécuter sur d'autres microcontrôleurs. Par contre un même code C peut être

compilé pour différents microcontrôleurs "cibles". L'étape de compilation se fait sur un ordinateur. Le programme est ensuite transféré dans la mémoire du microcontrôleur.

Partie II *Automates programmables industriels (api)*

Automate Programmable Industriel API

II.1.1 Introduction

Les Automates Programmables Industriels (API) ou encore PLC (programmable logic controller) sont apparus aux Etats-Unis vers 1969 où ils répondaient aux désirs des industries de l'automobile de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués.

Un Automate Programmable Industriel (en abrégé : API) est une ensemble électronique programmable par un personnel non informaticien et destiné commander un processus industriel (agro-alimentaire, fonderie, centre de tri, etc...). l'API (logique programmée) s'est substitué aux armoires à relais (logique câblée) en raison de sa souplesse dans la mise en œuvre, mais aussi parce que dans les coûts de câblage et de maintenance devenaient trop élevés.

II.1.2 Fonctionnement

L'automate programmable reçoit les informations relatives au système, il traite ces informations en fonction du jeu d'instruction et modifie l'état de ses sorties qui commandent les pré-actionneurs. Les fonctions que l'API doit remplir sont, Figure (II.1) :

- ✓ **Recevoir** : nécessité d'informations d'entrées.
- ✓ **Traiter** : notion de programme et de microprocesseur.
- ✓ **Jeu d'instructions** : notion de stockage donc de mémoire.
- ✓ **Commander** : notion de sortie afin de donner des ordres.

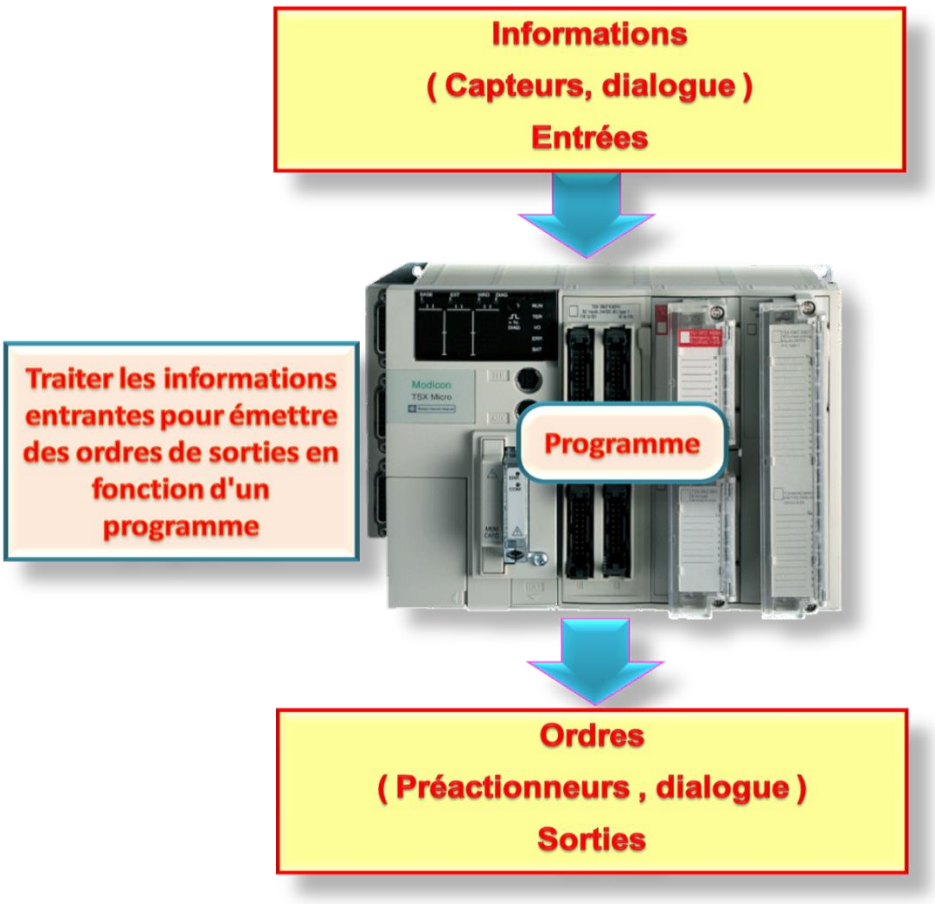


Fig II. 1: Principe d'Automate.

II.1.3 Aspect extérieur des API

Les automates programmables industriels peuvent être de type compact ou modulaire.

I.9 *II.1.3.a.* Type compact

Le processeur, l'alimentation, les cartes d'entrées et sorties sont intégrés dans un seul boitier (rack) et peuvent recevoir des extensions en nombre limité. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces API sont caractérisés par un fonctionnement simple, et leurs utilisations est réservée à la commande de petits automatismes.

a. Siemens LOGO *b. Crouzet MILLENIUM* *c. Schneider ZELIO* *d. Moeller PS*

Fig II. 2: API compacts.

I.10 *II.1.3.b.* Type modulaire

Ce type d'API est formé un ensemble de modules fonctionnels. Généralement, chacun des éléments : processeur, l'alimentation et les interfaces d'entrées / sorties est intégré dans des modules séparés et sont fixés sur un ou plusieurs coffret appelé racks .La communication entre ces différents modules est assurée un bus interne.

Ce type d'automate est intégré dans les automatismes complexes qui sont caractérisés par une grande : puissance, capacité de traitement et flexibilité, figure(II.3).

a. Siemens S7-300 *b. Schneider TSX 37* *c. Moeller* *d. Modicon*

Fig II. 3: API modulaires.

II.1.4 Insertion de l'API dans un système automatisé

L'insertion de l'API dans milieu automatisé est donnée par la figure (II.4), l'automate programmable reçoit les informations relatives à l'état du système et puis commande les pré-actionneurs suivant le programme inscrit dans sa mémoire.

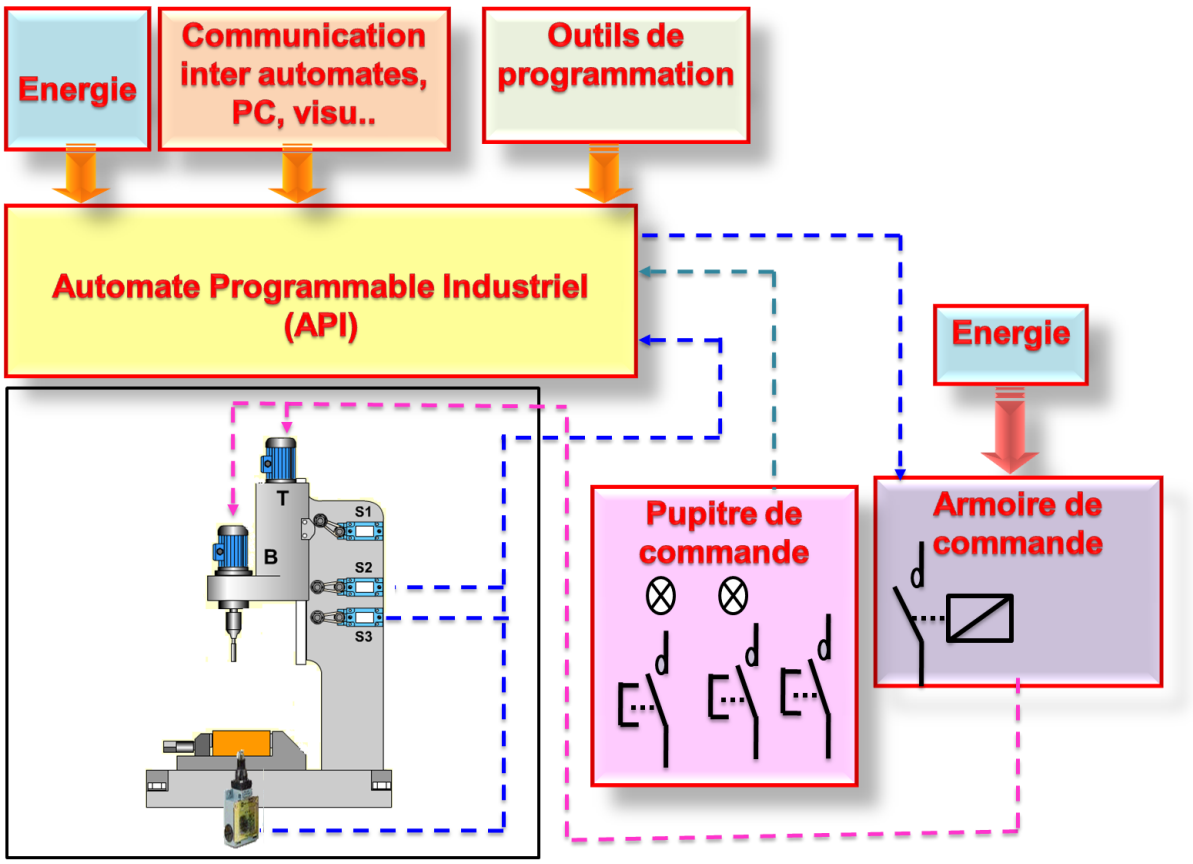


Fig II. 4: API dans un milieu industriel.

II.1.5 Structure d'un A.P.I

I.11 *II.1.5.a.* Structure externe

L'aspect externe d'un API est composé, titre d'exemple, pour un API modulaire Siemens :

1 Module d'alimentation
2 Pile de sauvegarde
3 Connexion au 24V cc
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

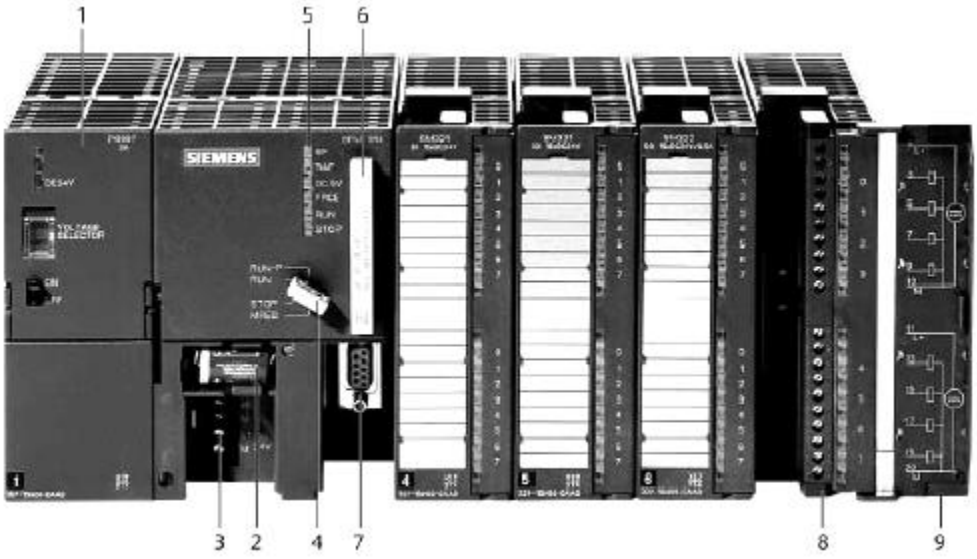


Fig II. 5: Structure externe d'API lieu industriel.

I.12 *II.1.5.b.* Structure interne

La structure interne d'un automate programmable industriel (API) est assez voisine de celle d'un système informatique simple, Fig (II.6).

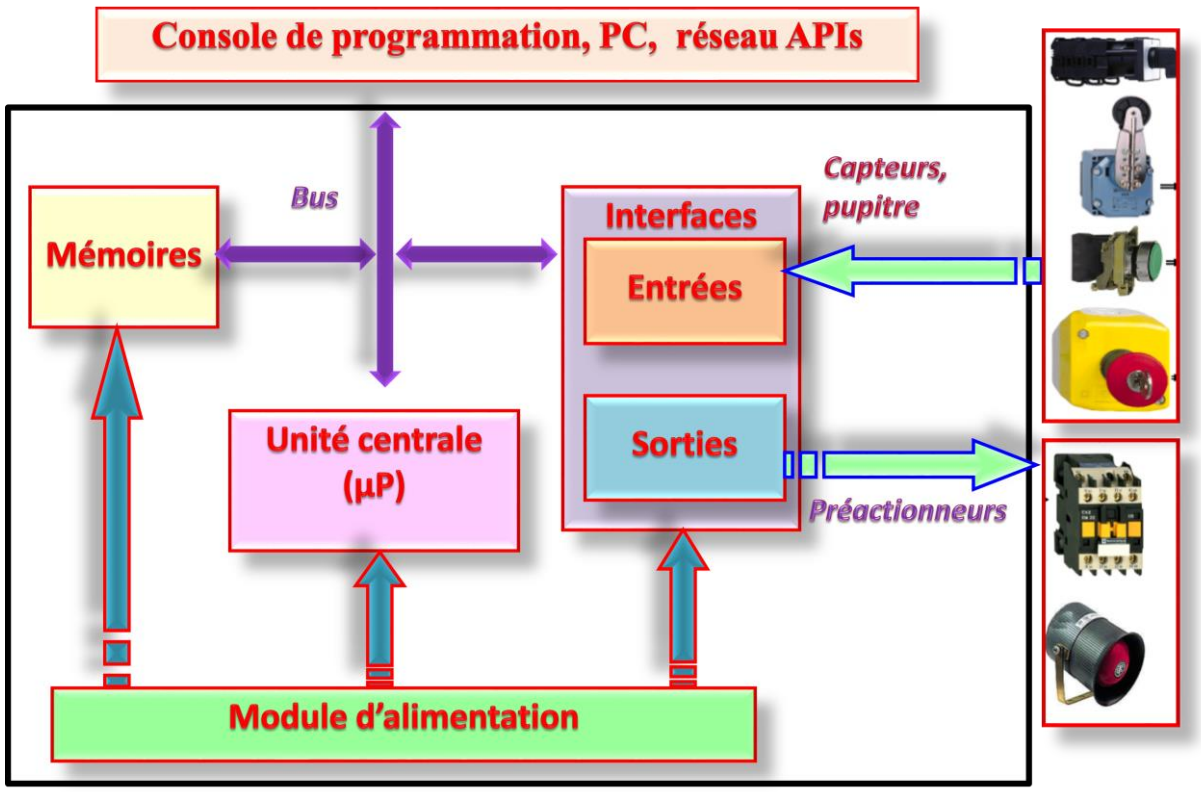


Fig II. 6: Structure interne d'un API.

Quatre parties principales composent l'architecture interne d'API :

- ✓ Une mémoire
- ✓ Un processeur

- ✓ Des interfaces d'Entrées/Sorties
- ✓ Une alimentation

Ces quatre parties, réunies forment un ensemble compact, sont reliées entre elles par des bus (ensemble câblé autorisant le passage de l'information entre ces 4 secteurs de l'API),

I.13 *II.1.5.c.* Description des éléments d'un A.P.I

- ➢ ***Interfaces d'Entrées/Sorties***

C'est la partie se trouvant entre les deux faces partie opérative (PO) et partie commande (PC) Traduisant les ordres et les informations.

Interface d'entrée:

Partie de l'API sur laquelle sont raccordés, les éléments de détection et les organes de commande installés sur le pupitre opérateur, qui donnent les informations à l'unité de traitement (capteurs , détecteurs, bouton poussoirs : départ cycle , arrêt d'urgence, réglage....)

Interface de sortie:

Partie de l'API sur laquelle sont raccordés les préactionneurs ou les éléments de signalisation du pupitre.qui reçoivent des ordres de l'unité de traitement.

Affectation & écriture des entrées / sorties

L'affectation des entées et des sorties permet de faire l'adressage entre le matériel et l'API en fonction de sont câblage. D'un point de vue de la programmation on travaillera sur l'adressage des E/S.

Les entrées et les sorties analogiques sont notées par un mot de la façon suivante : % xy.z

x : Les sorties sont notées par la lettre Q (Output) pour le PL7 et S pour le Step7

Les entrées analogiques seront notées par la lettre I (Input) pour le PL7 et A pour le Step7

y : c'est l'emplacement physique du module analogique, numéro du rack '(boitier)

z : c'est le numéro de la voie utilisée dans le même rack

Exemple :

On a câblé un bouton poussoir nommé DCY sur l'entrée n°5 de la carte d'entrée du module 3 et un voyant sur la sortie n°0 de la carte de sortie du module 2.

L'écriture de l'entrée et de la sortie sont comme suit.

- ✓ Écriture de l'entrée: % I3.5
- ✓ Écriture de la sortie: % Q2.0

➢ ***Mémoire***

La mémoire centrale est l'élément fonctionnel qui peut recevoir, conserver et restituer. Son rôle est de:

- ✓ De recevoir les informations issues des capteurs d'entrées via l'interface d'entrée ;
- ✓ De recevoir les informations générées par le processeur et destinées à la commande des sorties (valeur des compteurs, des temporisations, …) ;
- ✓ De recevoir et conserver le programmable du processus qui peut être introduit :
  - soit de la console (clavier) de programmation
  - soit à partir d'un micro-ordinateur (PC)
  - évidement du processeur de l'API, qui lui gère et exécute le programme.

Il existe dans les automates deux types de mémoires qui remplissent des fonctions différentes:

- ✓ La mémoire morte en lecture seulement (ROM), appelée mémoire Langage où est stocké le langage de programmation.
- ✓ La mémoire vive utilisable en lecture-écriture (RAM) appelée mémoire Travail .Elle s'efface automatiquement à l'arrêt de l'automate (nécessite une batterie de sauvegarde).

➢ ***Processeur***

Il est connecté aux autres éléments (mémoire et interface E/S) par des liaisons parallèles appelées ' BUS ' qui véhiculent les informations sous forme binaire, son rôle est de :

- ✓ Gérer les instructions du programme ;
- ✓ Organiser les différentes relations entre la zone mémoire et les interfaces d'E/S.

➢ ***Module d'alimentation***

Il assure la distribution d'énergie aux différents modules Composé de blocs qui permettent de fournir à l'automate l'énergie nécessaire à son fonctionnement. A partir d'une alimentation en 220 volts alternatifs, ces blocs délivrent des sources de tension dont l'automate a besoin : 24V, 12V ou 5V en continu. En règle générale, un voyant positionné sur la façade indique la mise sous tension de l'automate

II.1.6 Critères de choix d'un API

Les critères techniques de choix essentiels d'un API sont :

- ✓ Le nombre et la nature des E/S ;
- ✓ Les capacités de traitement du processeur (vitesse, données, opérations, temps réel, capacité de la mémoire...).
- ✓ Fonctions ou modules spéciaux
- ✓ Les moyens de dialogue et le langage de programmation ;
- ✓ La communication avec les autres systèmes ;

- ✓ Les moyens de sauvegarde du programme ;
- ✓ La fiabilité, robustesse, immunité aux parasites ;
- ✓ La documentation, le service après vente, durée de la garantie, la formation.

II.1.7 Interfaces et cartes d'Entrées / Sorties

Les interfaces d'Entrée/Sortie (E/S), circuits électroniques, se présentent généralement sous forme d'interfaces modulaires qu'on ajoute selon le besoin, ils ont une modularité de 8, 16 ou 32 voies. Les tensions disponibles sont normalisées (24, 48, 110 ou 230V continu ou alternatif ...).

Ils doivent non seulement transmettre les ordres aux pré-actionneurs ou de recevoir les informations des capteurs, mais aussi garantir un isolement galvanique ou un découplage optoélectronique assurant ainsi la protection de l'automate contre les signaux parasites, d'où l'utilisation d'opto-coupleurs ou phototransistors à chaque entrée des cartes d'entrée et sorties.

L'interface d'entrée a pour fonction de :

- ✓ Recevoir les signaux logiques en provenance des capteurs et du pupitre opérateur ;
- ✓ Traiter ces signaux en les mettant en forme, en éliminant les parasites d'origine industrielle et en isolant électriquement l'unité de commande de la partie opérative (isolation galvanique) pour la protection.

L'interface de sortie a pour fonction de :

- ✓ Commander les pré-actionneurs et éléments de signalisation du système ;
- ✓ Adapter les niveaux de tension de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières.

Les automates permettent de commander des sorties en T.O.R et gèrent parfois des traitements analogique ou fonctions spéciales de comptage rapide , de pesage

II.1.7.a Cartes d'entrées TOR

Elles sont destinées à recevoir l'information en provenance des capteurs et adapter le signal en le mettant en forme, en éliminant les parasites et en isolant électriquement l'unité de commande de la partie opérative, Fig (II.7).

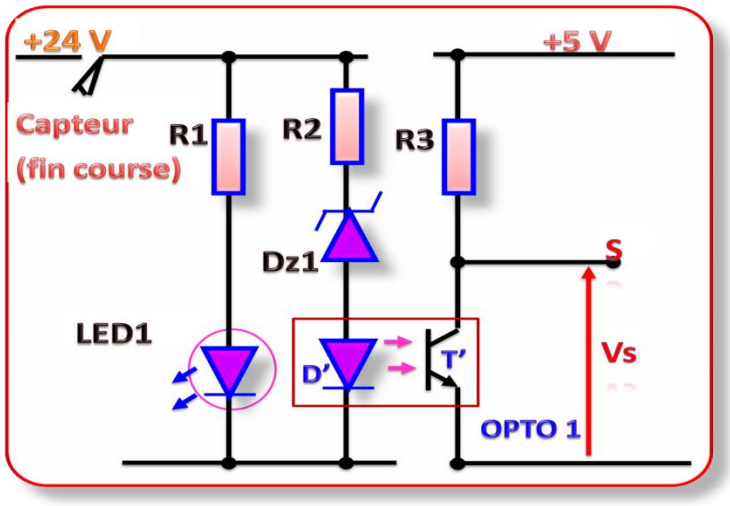


Fig II. 7: Exemple d'une carte d'entrées TOR d'un API.

Pour le schéma ci-dessus, représentant une carte d'entrée TOR, Lors de la fermeture du capteur ;

- LED1 signal que l'entrée automate est actionnée
- La led D' de l'optocoupleur s'éclaire
- Le photo transistor T' de l'optocoupleur devient passant
- La tension Vs=0V

Donc lors de l'activation d'une entrée automate, l'interface d'entrée envoie un **0 logique** à l'unité de traitement et un 1 logique lors de l'ouverture du contact du capteur (entrée non actionnée).

II.1.7.b Cartes de sorties TOR

Elles sont destinées à commander les pré-actionneurs et éléments des signalisations du système et adapter les niveaux de tensions de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières, Fig(II.8).

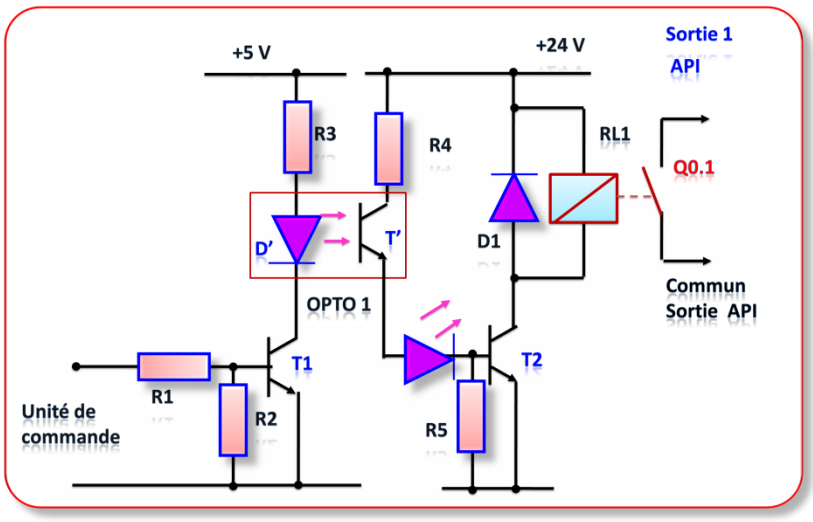


Fig II. 8: Exemple d'une carte de sortie TOR d'un API.

Lors de la commande d'une sortie TOR d'un API ;

- ✓ L'unité de commande envoie un 1 logique (5V)
- ✓ T1 devient passant, donc D' s'éclaire
- ✓ Le photo transistor T' de l'optocoupleur devient passant
- ✓ LED 1 s'éclaire et nous informe de la commande de la sortie Q0,1
- ✓ T2 devient passant
- ✓ La bobine RL1 devient sous tension et commande la fermeture du contact de la sortie O0,1

Donc pour commander une sortie automate l'unité de commande doit envoyer :

- ✓ Un 1 logique pour actionner une sortie API,
- ✓ Un 0 logique pour stopper la commande d'une sortie API.

➢ **Exemple des cartes E/S spéciales**

- ✓ **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.
- ✓ **Cartes de comptage rapide :** elles permettent d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate. (signal issu d'un codeur de position).
- ✓ **Cartes 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 de communication (RS485, Ethernet ...)** : Ils permettent d'établir des communications à distance avec d'autres systèmes de traitement par lignes séries: paires téléphoniques, fibres optiques, ...
- ✓ **Cartes d'entrées / sorties déportées**: ils permettant de décentraliser des châssis entrées / sorties industrielles sur des distances importantes (ordre du km). Cette possibilité de décentralisation permet, dans de nombreux cas, de réduire substantiellement le volume de câblage entre le processus et l'automate.

➢ **Autres cartes**

- ✓ Cartes de régulation PID.
- ✓ Cartes de pesage.
- ✓ Cartes de surveillance et de contrôle.

II.1.8 Traitement du programme automate

Un automate exécute son programme de manière cyclique comme suit, Figure (II.19) :

- **Traitement interne** : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).

- **Lecture des entrées** : L'automate reçoit des données par ses entrées et les recopie dans la mémoire image des entrées.

- **Traitement du programme** : L'automate traite les données entrantes par un programme défini dans sa mémoire et écrit les sorties dans la mémoire image des sorties.
- **Ecriture des sorties** : L'automate bascule les différentes sorties aux positions définies dans la mémoire image des sorties pour commander les actionneurs et dialoguer avec l'opérateur et les autres systèmes automatisés.

On appelle scrutation l'ensemble des quatre opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps varie selon la taille du programme et la puissance de l'automate et il est de l'ordre de la dizaine de millisecondes pour les applications standards.

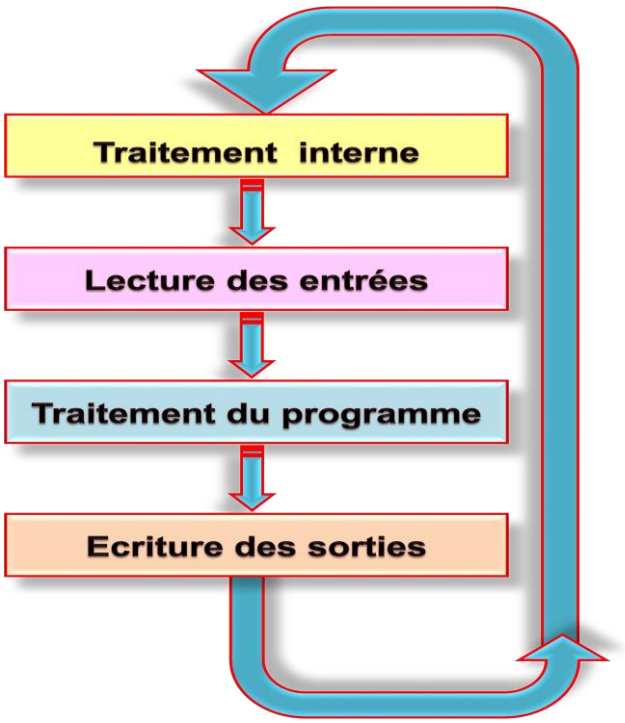


Fig II. 9: Cycle de travail API.

II.1.9 Outils graphiques et textuels de programmation d'un API

Chaque automate possède ses propres langages de programmations. Mais par contre, les constructeurs proposent tous une interface répondant à la norme CEI 11313. Cette norme définit cinq langages de programmation utilisables, ils sont de natures diverses étant donné la diversité, des utilisateurs pouvant les utiliser. Ils peuvent être classés en deux familles :

I.14 *II.1.9.a* Langages graphiques

Ils permettent une transcription graphique aussi directe que possible des modèles afin de faciliter les tâches de programmation et de réduire les sources d'incertitudes. On distingue les trois langages suivants : Schéma à contact, Schéma par blocs et Diagramme fonctionnel de séquence.

- **Schéma à contacts ou relais, Ladder Diagram (LD)**: il utilise des symboles électriques qui assemblés forment le programme. Ce type de programmation à l'avantage de pouvoir être utilisé par du personnel électricien ou ayant une connaissance de la schématique électrique sans pour autant apprendre un langage spécifique. Il est adapté au traitement combinatoire (dédié à la programmation d'équations booléennes).
- **Schéma par blocs Fonctionnel, ou function block diagram (FBD): Le FBD** ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables. Les blocs sont programmés (bibliothèque) ou programmables.

- **Langage GRAFCET ou Diagramme fonctionnel de séquence (Sequential Function Chart :SFC) :** il permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel. Il est dérivé du GRAFCET.

I.15 *II.1.9.b.* Langages textuels

Il s'agit de décrire le fonctionnement du processus par un programme sous forme un texte. Deux langage existent : Liste d'instruction et littéral structuré.

- **Liste d'instructions ou Instruction List (IL)** : C'est un langage "machine" qui permet l'écriture de traitements logiques et numériques. Il peut être comparé au langage assembleur. Très peu utilisé par les automaticiens.

- **Texte structuré ou Structured Text (ST)** : Le **ST** est un langage de type "informatique" permettant l'écriture structurée (algorithmes) de traitements logiques et numériques. Il est de même nature que le Pascal. Peu utilisé par les automaticiens.

Pour programmer l'automate, l'automaticien peut utiliser :

Elle peut s'effectuer de trois manières différentes :

- ✓ Sur l'A.P.I. lui-même à l'aide de touches.
- ✓ Avec une console de programmation reliée par un câble spécifique à l'A.P.I.
- ✓ Avec un PC et un logiciel approprié, (exemple : Step 7 pout les API Siemens ou PL7 pour Schneider, millenium pour les API Crouet…),.

II.1.10 Mise en œuvre d'un automate programmable industriel

I.16 *II.1.10.a.* Alimentation de l'automate

Généralement l'API est alimenté par le réseau monophasé 220V ; 50 Hz via un contacteur (contacteur KM1) et avec une protection du type magnéto-thermique.

De même, les sorties seront asservies au circuit de commande et alimentées après validation du chien de garde dont le rôle est l'enclenchement d'une procédure d'alarme en cas de dépassement du temps de scrutation à chaque cycle, Figure(II.10).

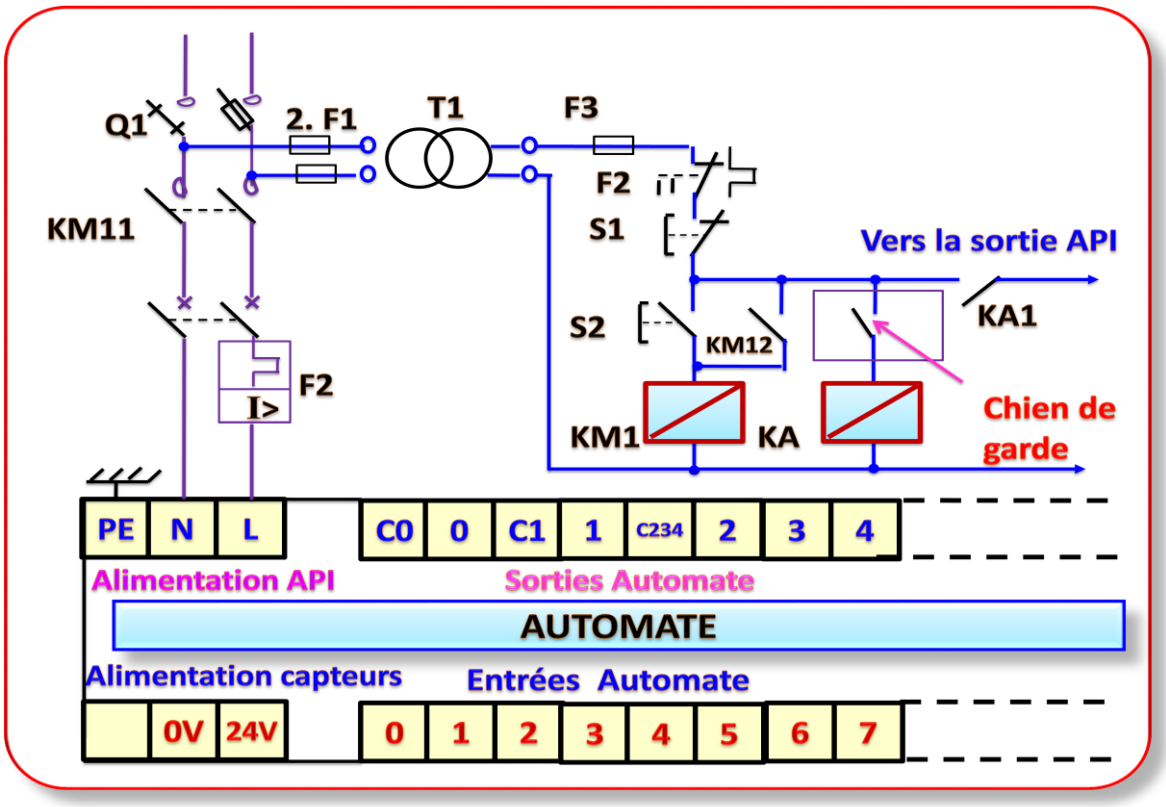


Fig II. 10: Alimentation de l'API.

I.17 *II.1.10.b.* Alimentation des entrées de l'automate

L'automate est pourvu généralement d'une alimentation pour les capteurs/détecteurs. L'alimentation électrique peut être fournie par l'automate (en général 24V continu) ou par une source extérieure, Figure(II.11).

L'alimentation des entrées et sorties d'un automate programmable peut être à :

- ✓ **Logique positive** où le commun interne des entrées est relié au 0V
- ✓ **Logique négative** où le commun interne des entrées est relié au 24V

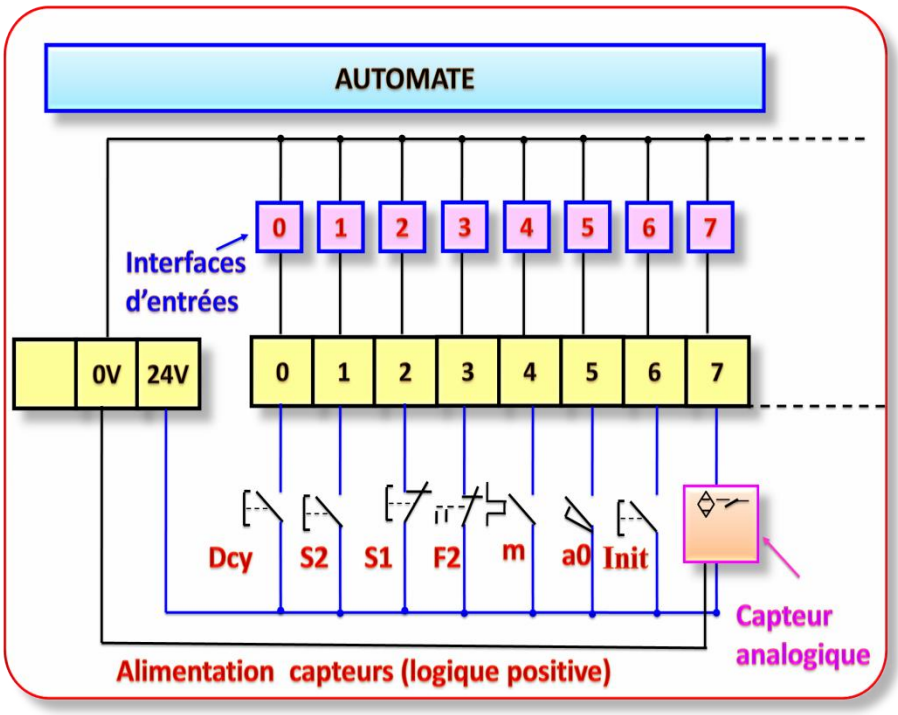


Fig II. 11: Alimentation des entrées de l'API.

I.18 *II.1.10.c.* Alimentation des sorties de l'automate

Les interfaces de sorties permettent d'alimenter les divers préactionneurs, les communs des sorties sont séparés ou groupé en petit nombre afin de séparé les alimentations (alimentation des différentes valeurs des préactionneurs) ou utilisé uniquement celles dont a besoin, Figure(II.12).

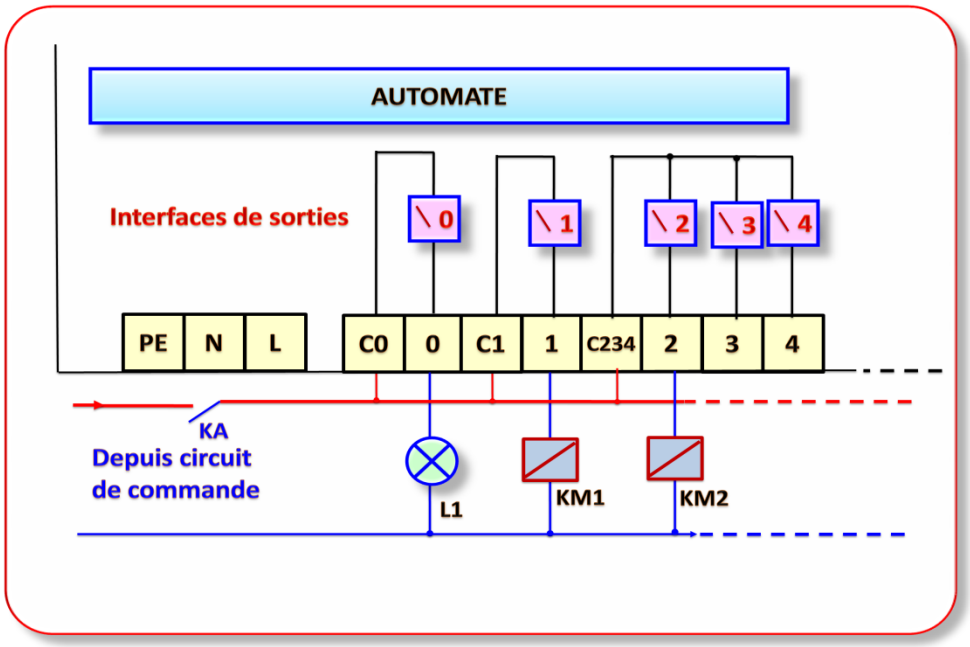


Fig II. 12: Alimentation des sorties de l'API.

II.1.11 Introduction aux Bus de communication et principes des réseaux d'automates

I.19 *II.1.11.a.* Bus de communication

- **Principe**

Avec le développement des systèmes automatisés et de l'électronique, la recherche de la baisse des coûts et la nécessité actuelle de pouvoir gérer au mieux la production et a partir du moment où tous les équipements sont de type informatique, il devient intéressant de les interconnecter à un mini-ordinateur ou à un automate de supervision (Figure II.13).

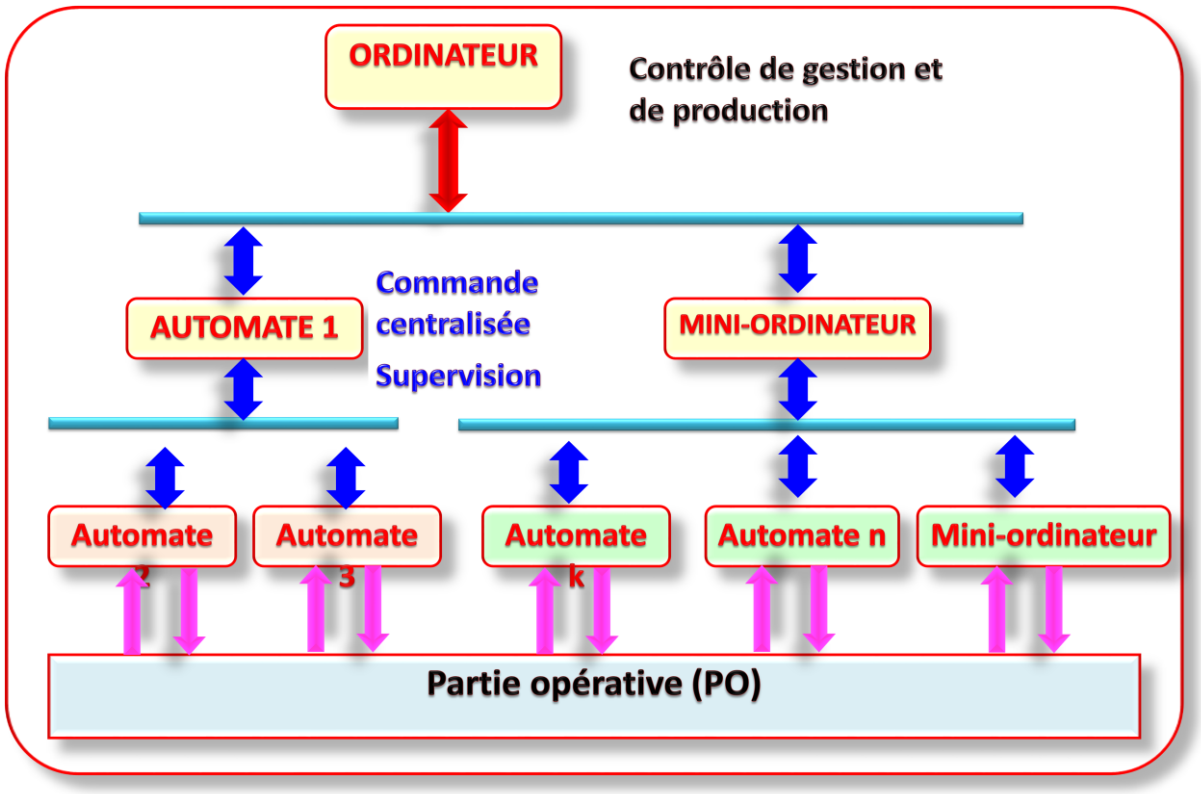


Fig II. 13: Exemple d'une structure de contrôle et gestion de production.

Avant de développé cette partie, il utile de définir des termes couramment utilisés dans le domaine de la communication entre automates.

- Terrain : espace géographique limité.
- Bus : ensemble de conducteurs commun à plusieurs circuits permettant d'échanger des données. Les échanges sont régis par un protocole.
- Réseau : bus ou ensemble de bus répartis sur un terrain.
- Protocole : un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau.
- Les bus de terrain sont les réseaux qui permettent de communiquer avec les équipements (API) de terrain unité de production), c'est à dire avec les capteurs et avec les actionneurs tout ou rien ou analogiques, mais aussi avec les automates

programmables, les terminaux opérateurs ou les applications de supervision de procédé.

I.20 *II.1.11.b.* Réseau d'automate

Les réseaux industriels apportent une grande souplesse aux systèmes de contrôle / commande, ils diminuent les coûts de câblage, ils offrent des possibilités nouvelles pour le contrôle et la supervision des installations, tant pour les équipes d'exploitation que de maintenance, de production ou de gestion.

Un *réseau d'automates* est un ensemble d'automates reliés entre eux de telle sorte que les sorties des uns soient les entrées des autres.

L'architecture d'un réseau comprend deux composantes:

- ✓ La topologie : Elle caractérise la configuration des voies de transmission existant entre les différentes stations (API) d'un réseau via le support physique (câble, nappe .)
- ✓ La répartition des fonctions du protocole : Elle définit si toutes les stations ont les mêmes fonctions ou, si une remplit le rôle de maître tandis que les autres sont des esclaves.

Les Modes de propagation des données dans le réseau peuvent classée en deux groupes :

- ✓ Mode de diffusion (bus ou anneau) :

Utilisation d'un seul support de transmission. Le message est envoyé sur le réseau, toute les unités du réseau sont capable de voir le message et d'analyser selon l'adresse du destinataire si le message lui est destiné ou non.

- ✓ Mode point à point (étoile ou maillée) :

Le support physique ne relie que deux unités. Pour que toutes les unités communiquent ensemble, elles ne passent pas par des points intermédiaires.

- ➢ **Topologie en anneau**

Toutes les unités sont reliées entre elles dans une boucle fermée. Les données circulent dans une direction unique, d'une unité à l'autre .Une unité n'accepte une donnée en circulation sur l'anneau que si elle correspond à son adresse, sinon la donnée est transférée à l'unité suivante, Fig(II.14).

- Problème si une unité est en panne ;
- En pratique il y a deux anneaux en contre-rotation.

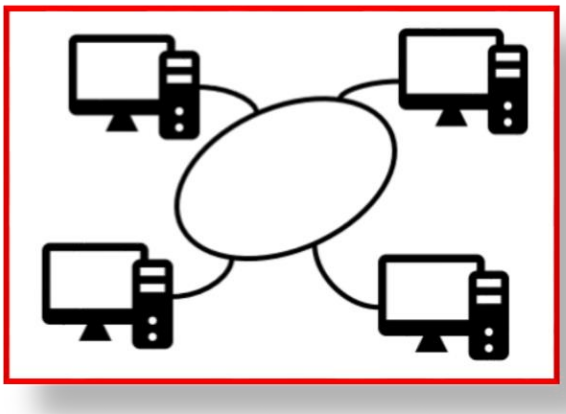


Fig II. 14: Réseau en anneau.

- **Topologie en étoile**

L'ensemble des équipements du réseau sont reliés à un système matériel central (nœud). Ce système a pour rôle d'assurer la communication entre les différents équipements. L'équipement centrale peut être, Fig(II.15):

- ✓ commutateur, « router » : routeur ;
- ✓ Facilite l'ajout de matériel ;
- ✓ Facilite la localisation des défaillances ;
- ✓ Branchement/débranchement à chaud ne pose pas de problème ;
- ✓ Très répandu
  - Relativement onéreux ;
  - Concentrateur défectueux = réseau en panne

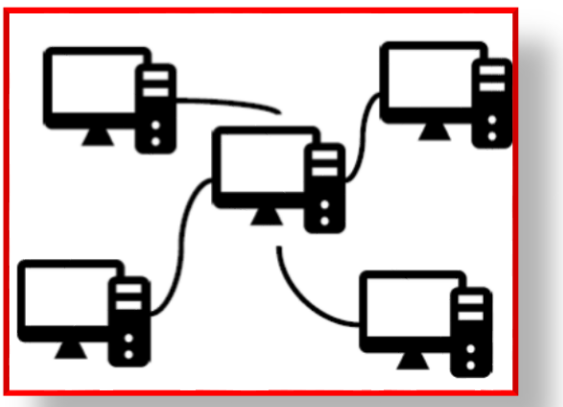


Fig II. 15: Réseau en étoile.

- **Topologie en hiérarchie:**

Cette topologie est organisée en niveaux. Le sommet de la hiérarchie est connecté à des nœuds, niveaux inférieurs. Ces nœuds peuvent également être connectés à d'autres nœuds. Le tout dessine une arborescence, Fig(II.16).

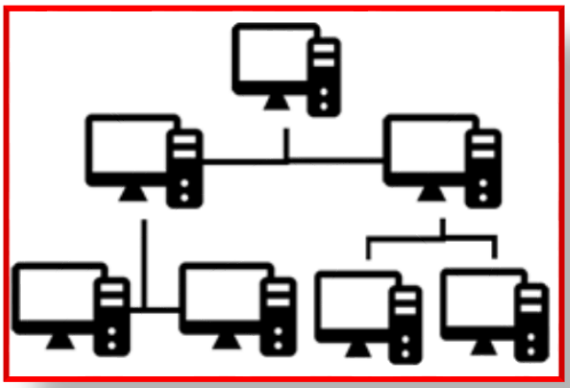


Fig II. 16: Réseau en hiérarchie.

Ce type d'architecture est dépendante du système « père », s'il tombe en panne alors il est impossible d'établir une communication entre les deux branches du réseau.

- **Topologie en bus:**

Un câblage unique permet d'établir la communication entre l'ensemble des nœuds, Fig (II.17).

- ✓ Le réseau n'est pas perturbé lorsqu'une station est défectueuse,
- ✓ Dans le cas d'une communication bidirectionnelle, l'ensemble des stations connectées reçoivent les signaux émis sur le bus,
- ✓ Coût d'installation de l'infrastructure peu élevé,
- ✓ Nombre d'unités limités / la longueur du support.

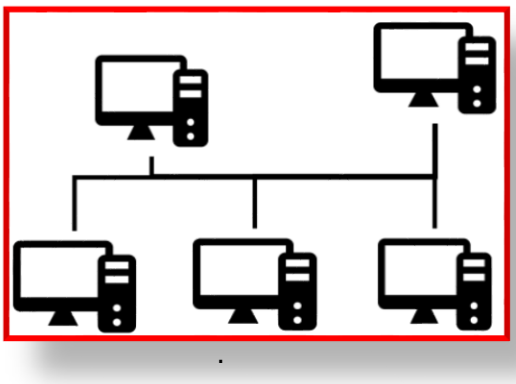


Fig II. 17: Réseau en bus.

- **Topologie en maillé:**

Il s'agit d'une topologie point à point multiple. Chaque nœud peut avoir (1,N) connexions avec les autres nœuds. Les autres nœuds sont reliés à tous les autres, Fig(II.18).

- ✓ Ce type d'architecture devient vite très complexe à gérer lorsqu'il y a beaucoup de stations ;

✓ Cette topologie se rencontre dans les grands réseaux de distribution (ex: Internet).

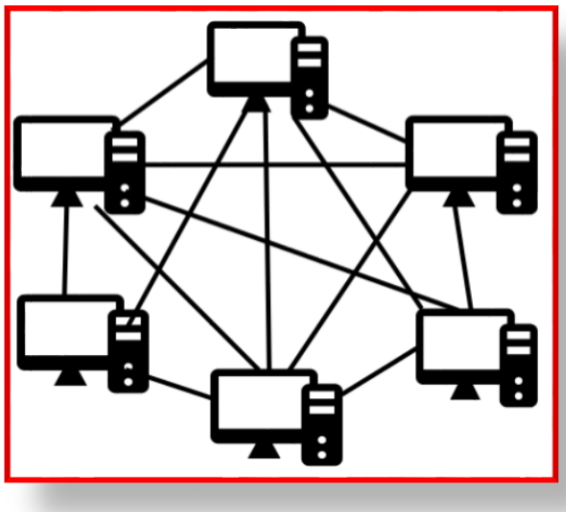


Fig II. 18: Réseau en maillé.

Partie III *Langages de Programmation : Norme IEC 1131-3*

III.1 Introduction

III.1.1 La logique câblée

L'automatisme est obtenu en reliant entre eux les différents constituants de base ou fonctions logiques par câblage. La logique câblée correspond donc à un traitement parallèle de l'information. Plusieurs constituants peuvent être sollicités simultanément.

Elle est étudiée et réalisée une fois pour toutes sur un schéma donné : Les fonctions sont réalisées par voie matérielle.

- ***Inconvénients***
  - ✓ Volume du contrôle proportionnel à la complexité du problème,
  - ✓ Des modifications de la commande impliquent des modifications de câblage,
  - ✓ Exige un grand nombre de composants et rend les montages encombrants,
  - ✓ La durée des études pour réaliser un montage donné (et donc pour le modifier le cas échéant) est longue
- ***Avantages***
  - ✓ Moins chère pour les cas simple et non complexe.

I.21 *III.1.1.a* Logique programmée

Elle correspond à une démarche séquentielle, seule une opération élémentaire est exécutée à la fois, c'est un traitement série. Le schéma électrique est transcrit en une suite d'instruction qui constitue le programme.

- ***Inconvénients***
  - ✓ Coût de l'API est relativement cher,
  - ✓ Investissement non rentable si le montage est simple.
- ***Avantages***
  - ✓ En cas de modification des équations avec les mêmes accessoires, l'installation ne comporte aucune modification de câblage seul le jeu d'instructions est modifié,
  - ✓ Utilisation réduite de composants puisque ils sont remplacés directement les fonctions logiques désirées,
  - ✓ Un circuit ayant moins de composants sera habituellement moins coûteux à concevoir, réaliser et distribuer, et simplification de la maintenance,
  - ✓ La réduction du nombre de composants électroniques tend aussi à augmenter la fiabilité des circuits et à réduire la consommation énergétique,
  - ✓ L'automate simplifie grandement le schéma de la logique câblée prenant en compte tout ce qui est extérieur à la programmation, comme les voyants.

- ✓ Réduire les coûts d'ingénierie
- ✓ Réduire les coûts de maintenance

➢ ***Les limites :***

Le choix du type d'une logique pour résoudre un problème, dépend de plusieurs critères : Complexité ; coût ; évolutivité ; rapidité.

- ***Limite inférieure :***

Si la fonction à réaliser est trop simple, il est plus économique de conserver une logique câblée.

- ***Limite supérieure :***

Si le nombre d'unités à réaliser est très important, il est plus économique de la fabriquer en circuits intégrés à la demande ou en logique câblée pour des fonctions simples.

III.1.2 Besoin de la normalisation

Les besoins d'une normalisation des langages pour API s'exprime, pour les industriels, en termes de:

- ✓ Faciliter la formation des réalisateurs de configurations d'automates programmables,
- ✓ Obtenir un bon niveau de portabilité des programmes,
- ✓ Favoriser la création de bibliothèques de blocs fonctionnels fiables,
- ✓ Faciliter les configurations en réseau d'automates,
- ✓ Améliorer la qualité des applications (sûreté de fonctionnement, maintenabilité, extensibilité),
- ✓ Réutiliser les outils de configuration et de programmation,
- ✓ Produire des dossiers d'applications homogènes,
- ✓ Faciliter la maintenance du logiciel d'application.

III.1.3 Historique

Le CENELEC (Comité Européen de Normalisation Electrotechnique) a adopté en 1993 le texte (65B) de la CEI (Commission Electrotechnique Internationale) comme norme EN 61131 pour la commande des processus industriels. Cette norme traite des automates programmables en 5 parties :

- ✓ CEI 1131-1 définitions, , informations générales.
- ✓ CEI 1131-2 spécifications et essais matériels,
- ✓ CEI 1131-3 langages de programmations
- ✓ CEI 1131-4 documentations
- ✓ CEI 1131-5 communications

L'originalité des langages de programmation pour automates est qu'ils sont généralement imagés par rapport à l'expression de la commande des machines automatisées. Ils sont

souvent graphiques et depuis 20 ans des formes de langages se sont dégagés et sont mise à disposition par les constructeurs d'API (liste d'instruction mnémonique, réseau à contacts, GRAFCET).

III.1.4 Norme ICE 1131

La norme CEI 1131 s'applique aux automates programmables industriels et à leurs périphériques. Les objectifs sont:

- ✓ ***Donner les définitions et identifier*** les principales caractéristiques permettant de sélectionner et utiliser les automates programmables et leurs périphériques associés.
- ✓ ***Déterminer les prescriptions minimales*** relatives aux caractéristiques fonctionnelles, aux conditions de service, aux caractéristiques constructives, à la sécurité générale ainsi qu'aux essais applicables aux automates programmables et à leurs périphériques.
- ✓ ***Définir pour les langages de programmation*** les principaux champs d'applications, les règles syntaxiques et sémantiques ainsi que des ensembles de base simples mais exhaustifs d'éléments de programmation.
- ✓ ***Fournir à l'utilisateur*** des informations générales didactiques et des recommandations quant à son application.

La partie 3 de cette norme, 1131-3, définit :

- ✓ ***Les langages de programmation,***
- ✓ ***Les modules logiciels ou unités d'organisation de programmes.***

➢ ***Remarque***

L'automaticien peut introduire son programme dans l'API de trois manières différentes :

- ✓ A l'aide de touches sur l'A.P.I. lui-même,
- ✓ Avec une console de programmation reliée par un câble spécifique à l'A.P.I,
- ✓ Avec un PC et un logiciel approprié, (exemple : Step 7 pout les API Siemens ou PL7 pour Schneider, millenium pour les API Crouet…) et le câble reliant le PC à l'API.

III.2 Langages de programmation

La norme IEC 1131-3 définit cinq langages qui peuvent être utilisés pour la programmation des automates programmables industriels. Ces cinq langages sont :

- ✓ ***LD*** (« Ladder Diagram », ou schéma à relais): ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (vraie/faux).
- ✓ ***IL*** (« Instruction List », ou liste d'instructions): ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.

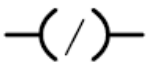
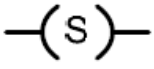
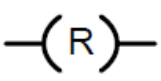
- ✓ ***FBD*** (« Function Block Diagram », ou schéma par blocs): ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.
- ✓ ***SFC*** (« Sequential Function Char »): issu du langage GRAFCET, ce langage, de haut niveau, permet la programmation aisée de tous les procédés séquentiels.
- ✓ ***ST*** («Structured Text » ou texte structuré): ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

III.2.1 Langages graphiques

Ils permettent une transcription graphique aussi directe que possible des modèles afin de faciliter les tâches de programmation et de réduire les sources d'incertitudes. On distingue les trois langages suivants : Schéma à contact, Schéma par blocs et Diagramme fonctionnel de séquence.

I.22 *III.2.1.a* Schéma à contacts ou relais, Ladder Diagram (LD)

Il utilise des symboles électriques qui assemblés forment le programme. Ce type de programmation à l'avantage de pouvoir être utilisé par du personnel électricien ou ayant une connaissance de la schématique électrique sans pour autant apprendre un langage spécifique. Il est adapté au traitement combinatoire (dédié à la programmation d'équations booléennes).

Désignation	Schéma à contact	Fonction	Symbole API
Contact à Ouverture		Passant à l'état de repos quand il n'est pas actionné.	
Contact à Fermeture		Ouvert à l'état de repos, continuité électrique non assurée	
Connexion horizontale		Relie des éléments en série	
Connexion verticale		Relie des éléments en parallèle	
Bobine directs		La sortie prend la valeur du résultat logique.	
Bobine inverse		La sortie prend la valeur inverse du résultat logique.	
Bobine d'enclenchement		Le bit interne est mis à **1** et garde cet état	
Bobine de déclenchement		Le bit interne est mis à **0** et garde cet état	

Le langage à relais est basé sur un symbolisme très proche de celui utilisé pour les schémas de câblage classiques. Les symboles les plus utilisés sont:

➢ ***Remarque***

- Un bit étant une mémoire interne logique prenant la valeur 0 ou 1.
- Une bobine d'enclenchement S « Set » et une bobine de déclenchement R « Reset » correspondent à un relais bistable.
- En plus des blocs fonctions logiques d'automatisme , il existe les blocs de temporisation de comptage….
-

I.23 *III.2.1.b* Schéma par blocs Fonctionnel, ou function block diagram (FBD)

Le langage FBD permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet d'une part la construction d'équations complexes à partir des opérateurs standards, de fonctions ou de blocs fonctionnels et d'autre part de manipuler tous les types de variables. Les blocs sont programmés (bibliothèque) ou programmables.

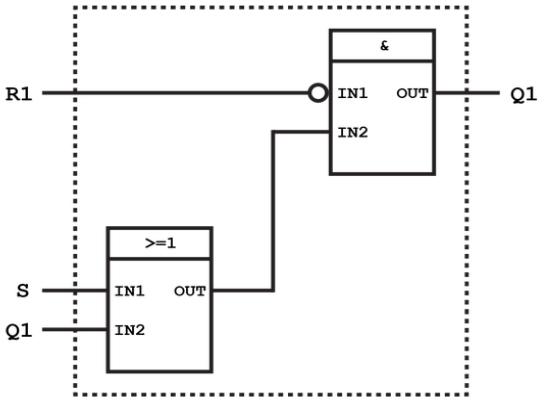


Fig III. 1: Schéma de principe d'une fonction bloc.

➢ ***Les principales fonctions sont***

- ✓ L'énonce RETURN (peut apparaître comme une sortie du diagramme, si liaison connectée prend l'état booléen TRUE, la fin du diagramme n'est pas interprétée.
- ✓ Les étiquettes et les sauts conditionnels sont utilisés pour contrôler l'exécution du diagramme. Aucune connexion ne peut être réalisée à droite d'un symbole d'étiquette ou de saut.
- ✓ Saut à une étiquette (le nom de l'étiquette est « LAB ») :
  - Si la liaison à gauche du symbole de saut prend l'état booléen TRUE, l'exécution du programme est déroutée après l'étiquette correspondante.
  - L'inversion booléenne est représentée par un petit cercle

I.24 *III.2.1.c* Langage GRAFCET (Sequential Function Chart :SFC)

Connu sous aussi sous le nom de Diagramme fonctionnel de Séquence, il permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel. Il est dérivé du GRAFCET. Le procédé est représenté comme une suite connue d'étapes (états stables), reliées entre elles par des transitions, une condition booléenne est attachée à chaque transition. Les actions dans les étapes sont décrites avec les langages ST, IL, LD ou FBD.

- ***Les principales règles graphiques sont :***
  - ✓ Un programme SFC doit contenir au moins une étape initiale,
  - ✓ Une étape ne peut pas être suivie d'une autre étape,
  - ✓ Une transition ne peut pas être suivi d'une autre transition.
- ***Les composants de base (symboles graphiques) du graphique SFC sont :***
  - ✓ Etapes et étapes initiales,
  - ✓ Transitions,
  - ✓ Liaisons orientées,
  - ✓ Renvoi à une étape.
- ***Les différents types d'action sont :***
  - ✓ Action booléenne (Elle est forcée à chaque fois que le signal d'activité de l'étape change d'état.),
  - ✓ Action impulsionnelle programmée en ST, LD ou IL (c'est une liste d'instructions ST, IL ou LD, exécutée à chaque cycle pendant toute la durée d'activité de l'étape),
  - ✓ Action normale programmée en ST, LD ou IL,
  - ✓ Action SFC (Une action SFC est une séquence fille SFC, lancée ou terminée selon les évolutions du signal d'activité de l'étape. Elle peut être décrite avec les qualificatifs d'action N (non mémorisée), S (set), ou R (reset),

Plusieurs actions (de même type ou de types différents) peuvent être décrites dans la même étape. Un appel de fonctions ou de blocs fonctionnels permet d'intégrer des traitements décrits dans d'autres langages (FBD, LD, ST ou IL).

III.2.2 Langages textuels

Il s'agit de décrire le fonctionnement du processus par un programme sous forme un texte. Deux langage existent : Liste d'instruction et littéral structuré.

I.25 *III.2.2.a* Liste d'instructions ou Instruction List (IL)

Le langage IL (instruction list), est un langage textuel de bas niveau. Il est particulièrement adapté aux applications de petite taille . C'est un langage "machine" qui permet l'écriture de traitements logiques et numériques. Il peut être comparé au langage assembleur. Très peu utilisé par les automaticiens.

L'opérateur indique le type d'opération à effectuer entre le résultat courant et l'opérande. Le résultat de l'opération est stocké à son tour dans le résultat courant.

Un programme IL est une liste d'instructions. Chaque instruction doit commencer par une nouvelle ligne, et doit contenir un opérateur, complété éventuellement par des modificateurs et, si c'est nécessaire pour l'opération, un ou plusieurs opérandes, séparés par des virgules (','). Une étiquette suivie de deux points (':') peut précéder l'instruction. Si un commentaire est attaché à l'instruction, il doit être le dernier élément de la ligne. Des lignes vides peuvent être insérées entre des instructions. Un commentaire peut être posé sur une ligne sans instruction.

I.26 *III.2.1.b* Texte structuré ou Structured Text (ST)

Le **ST** est un langage textuel de haut niveau de type "informatique" permettant l'écriture structurée (algorithmes) de traitements logiques et numériques. Il est de même nature que le Pascal. Peu utilisé par les automaticiens.

C'est un langage qui peut être utiliser pour la programmation des actions dans les étapes et des conditions associées aux transitions du langage SFC.

Un programme ST est une suite d'énoncés. Chaque énoncé est terminé par un point virgule (« ; »). Les noms utilisés dans le code source (identificateurs de variables, constantes, mots clés du langage...) sont délimités par des séparateurs passifs ou des séparateurs actifs, qui ont un rôle d'opérateur. Des commentaires peuvent être librement insérés dans la programmation.

- ***Les types d'énoncés standard sont :***
  - ✓ Assignation (variable **:=** expression;) ;
  - ✓ Appel de fonction ;
  - ✓ Appel de bloc fonctionnel ;
  - ✓ Enoncés de sélection (if, then, else, case) ;
  - ✓ Enoncés d'itération (for, while, repeat) ;
  - ✓ Enoncés de contrôle (return, exit) ;
  - ✓ Opérateurs booléens (not, and, or, xor) ;
  - ✓ Enoncés spéciaux pour le lien avec le langage sfc.

Il est recommandé de respecter les règles suivantes quand on utilise les séparateurs passifs, pour assurer une bonne lisibilité du code source :

- ✓ Ne pas écrire plusieurs énoncés sur la même ligne ;
- ✓ Utiliser les tabulations pour indenter les structures de contrôle ;
- ✓ Insérer des commentaires.

Le langage liste d'instruction permet de transcrire sous forme de liste :

- ✓ Un schéma à contact ;
- ✓ Un logigramme, équations booléennes ;
- ✓ Un GRAFCET.

Il réalise aussi des fonctions d'automatisme telles que temporisation, comptage, pas à pas ...

Instructions de test : Instructions d'entrée	
Désignation	**Fonctions**
LD	Le résultat est égal à l'opérande (load : lire la valeur).
LDN	Le résultat est égal à l'inverse de l'opérande (contact ouverture).
AND	ET logique entre le résultat et précèdent et l'état de l'opérande.
ANDN	ET logique entre le résultat et précèdent et l'état inverse de l'opérande.
OR	OU logique entre le résultat et précèdent et l'état de l'opérande.
ORN	OU logique entre le résultat et précèdent et l'état inverse de l'opérande.
XOR, XORN	OU exclusif.

Instructions de test : Instructions d'action	
Désignation	**Fonctions**
ST	L'opérande associé prend la valeur de la zone de test.
STN	L'opérande associé prend la valeur inverse de la zone de test.
S	L'opérande associé est mis à 1 lorsque le résultat de la zone de test est à 1.
R	L'opérande associé est mis à 1 lorsque le résultat de la zone de test est à 1.

III.3 Instructions de base en langages : LD, IL et ST

Les instructions booléennes et les blocs fonctions ont des représentations différentes suivant le langage.

III.3.1 Instructions de chargement

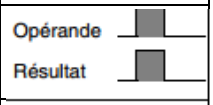
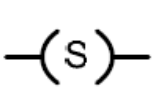
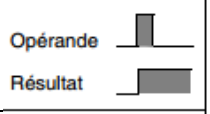
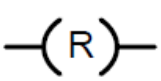
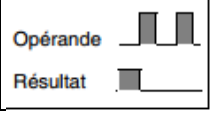
Le tableau suivant décrit le rôle de chacune des instructions de chargement en langage LD, IL et ST.

Langage LD	Langage IL	Langage ST	Fonction	Chronogramme
	LD	**:=**	Contact à fermeture : résultat **1**, quand l'objet bit qui le pilote est à **1.**	Opérande / Résultat
	LDN	**:=NOT**	Contact à ouverture : résultat **1**, quand l'objet bit qui le pilote est à **0.**	Opérande / Résultat
P	**LDR**	**:=RE**	Contact à front montant : détection du passage de **0 à 1**de l'objet bit qui le la mise à	Opérande / Résultat
N				Opérande / Résultat

			1 du résultat s'effectue pendant 1 cycle.	
	LDF	**:= FE**	Contact à front descendant : détection, du passage de **1 à 0** de l'objet bit qui le la mise à **1** du résultat s'effectue pendant 1 cycle.	

III.3. 2 Instructions d'affectation

Le tableau suivant décrit le rôle de chacune des instructions d'affectation en langage LD, IL et ST.

Langage LD	Langage IL	Langage ST	Fonction	Chronogramme
	ST	**:=**	aux bobines directes : l'objet bit associé prend la valeur du résultat de l'équation **.**	
	**STN**	**:=NOT**	aux bobines inverses : l'objet bit associé prend la valeur de l'inverse du résultat de l'équation.	
	**S**	**SET**	aux bobines à enclenchement : l'objet bit associé est mis à **1** lorsque le résultat de l'équation est à **1**.	
	**R**	**RESET**	aux bobines à déclenchement : l'objet bit associé est mis à **0** lorsque le résultat de l'équation est à **1**.	

Partie IV *Exemples et problèmes de programmations*

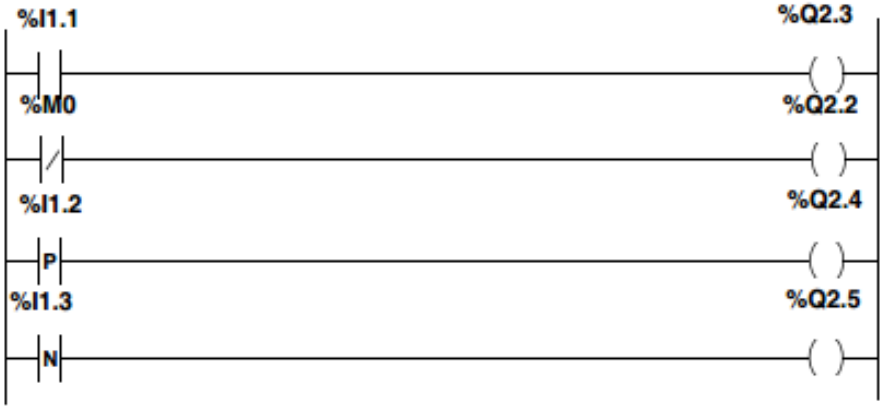
IV.1 Exemples à programmer

IV.1.1 Exemple 1 : Programmation en LD, IL et ST

Exemple en langage Ladder LD :

- Traduisez le schéma ci-dessous en langage liste d'instruction et texte structuré :

I.27



- ***Exemple en langage liste d'instructions IL***

N° de ligne	Instruction	Opérande
00	LD	% I1.1
01	ST	% Q2.3
02	LDN	% M0
03	ST	% Q2.2
04	LDR	% I1.2
05	ST	% Q2.4
06	LDF	% I1.3
07	ST	% Q2.5

➢ ***Exemple en langage littéral structuré ST***

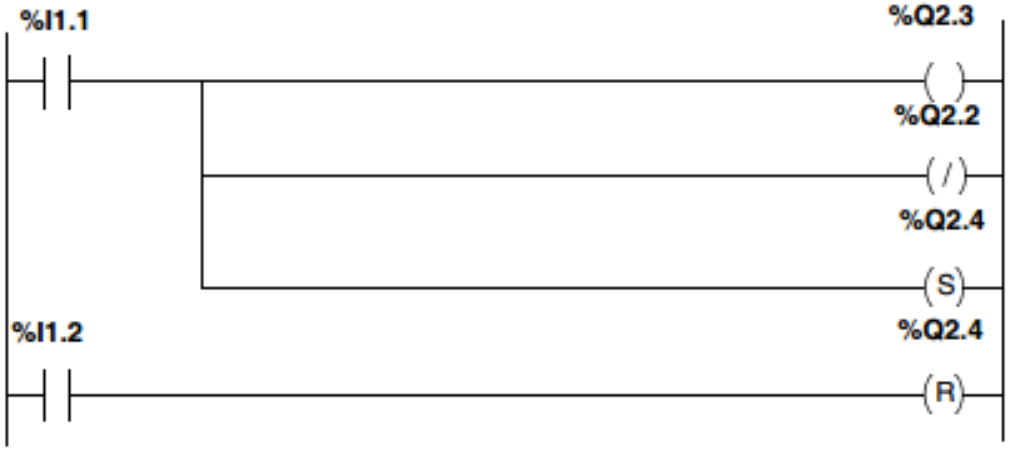
```
% Q2.3 := % I1.1

% Q2.2 :=NOT   % M0

% Q2.4 := RE  % I1.2

% Q2.5 := FE  % I1.3
```

IV.1.2 Exemple 2 : Programmation en LD, IL et ST

➢ ***Exemple en langage Ladder LD :***

- Traduisez le schéma ci-dessous en langage liste d'instruction (PL7) :

I.2



➢ ***Exemple en langage liste d'instructions IL***

N° de ligne	Instruction	Opérande
00	LD	% I1.1
01	ST	% Q2.3
02	STN	% Q2.2
03	S	% Q2.4
04	LD	% I1.2
05	R	% Q2.4

➢ ***Exemple en langage littéral structuré ST***

```
% Q2.3 := % I1.1;

% Q2.2 :=NOT   % I1.1;

IF % I1.1   THEN

      SET  % Q2.4
END_IF;

IF % I1.2   THEN

      RESET  % Q2.4;

END  IF:
```

IV.1.3 Exemple 3 : Programmation de temporisation

➢ ***Cahier des charges :***

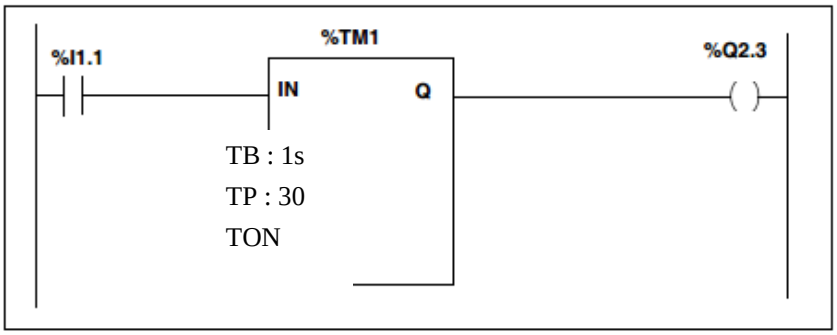
La programmation des blocs fonction temporisateur est identique quel que soit le mode d'utilisation sélectionné.

Le choix du fonctionnement TMi , TON, TB et TP s'effectue dans l'éditeur de variables.

Mode	TON
TB	1s,
TP	30
%TMi	1

➢ ***Exemple en langage Ladder LD***

I.29



➢ ***Exemple en langage liste d'instructions IL***

N° de ligne	Instruction	Opérande
00	LD	% I1.1
01	IN	% TM1
02	LD	% TM1.Q
03	ST	% Q2.3

➢ ***Exemple en langage littéral structuré ST***

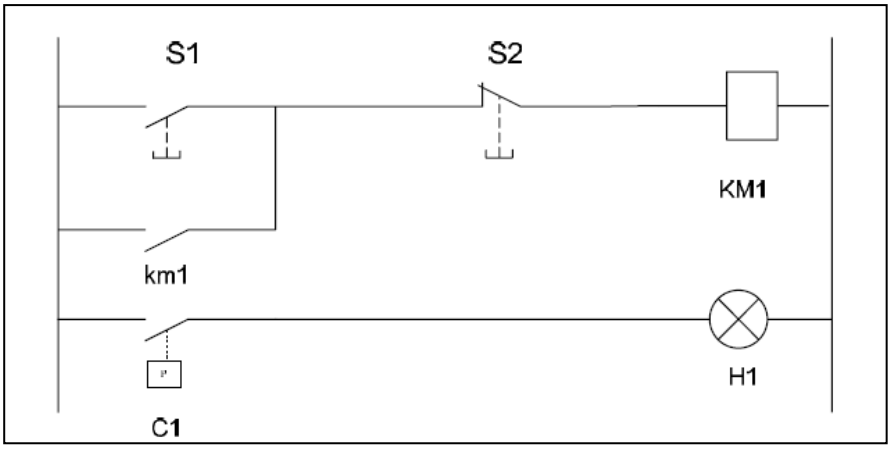
```
IF   RE  % I1.1   THEN
     START   % TM1;
ELSIF  FE % I1.1   THEN
DOWN  % TM1;
     END_IF;
% Q2.3 := % % TM1.Q
```

- L'instruction START %TMi, génère un front montant sur l'entrée IN du bloc temporisateur.
- L'instruction DOWN %TMi, génère un front descendant sur l'entrée IN du bloc temporisateur.

IV.1.4 Exemple 4 : Commande d'une pompe

➢ ***Cahier des charges :***

- Traduisez le schéma ci-dessous en langage liste d'instruction (PL7) :



➢ ***Solution*** :

Soit le tableau des entrées et sortie avec l'adressage API :

Elément d'entrée	désignation	Adressage API	Elément de sortie	désignation	Adressage API
S1	Bouton d'arrêt	% I1.0	KM1	Contacteur de pompe	% Q2.0
S2	Bouton Marche	% I1.1	H1	Voyant de pression	% Q2.1
C1	Capteur de pression	% I1.2			

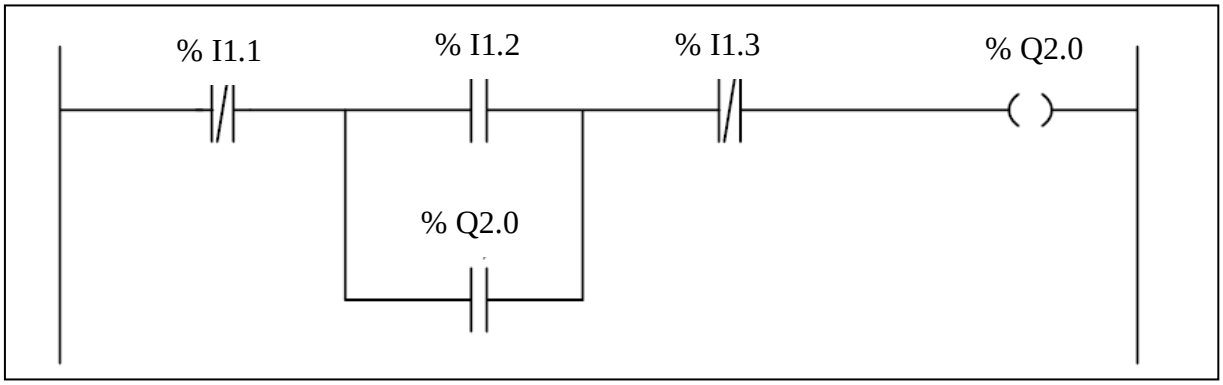
➢ ***Programme en langage IL du schéma de commande de la pompe***

N° de ligne	Instruction	Opérande	Commentaire
00	LD	% I1.0	Tester le bouton marche S1
01	OR	% Q2.0	Exécuter un OU avec KM1
02	AND	% I1.1	Exécuter un ET avec Le bouton marche S2
03	ST	% Q2.0	Activer la sortie du contacteur KM1 (Q2.0)
04	LD	% I1.2	Tester la capteur de pression C1
05	ST	% Q2.1	Activer la sortie voyant H1

IV.1.5 Exemple 5 : Transcription du langage Ladder en langage IL

➢ ***Cahier des charges :***

- A partir du programme Ladder ci-dessous déterminer la liste des instructions.



➢ ***Programme en langage IL***

N° de ligne	Instruction	Opérande	Commentaire
00	LDN	% I1.1	Lire l'entrée inverse
01	AND(	% I1.2	Exécuter un ET , On imbrique une parenthèse.
02	OR	% Q2.0	Exécuter un OU avec la ligne précédente
03	)		On ferme la parenthèse
04	ANDN	% I1.3	Exécuter un NON ET
05	ST	% Q2.0	Activer la sortie

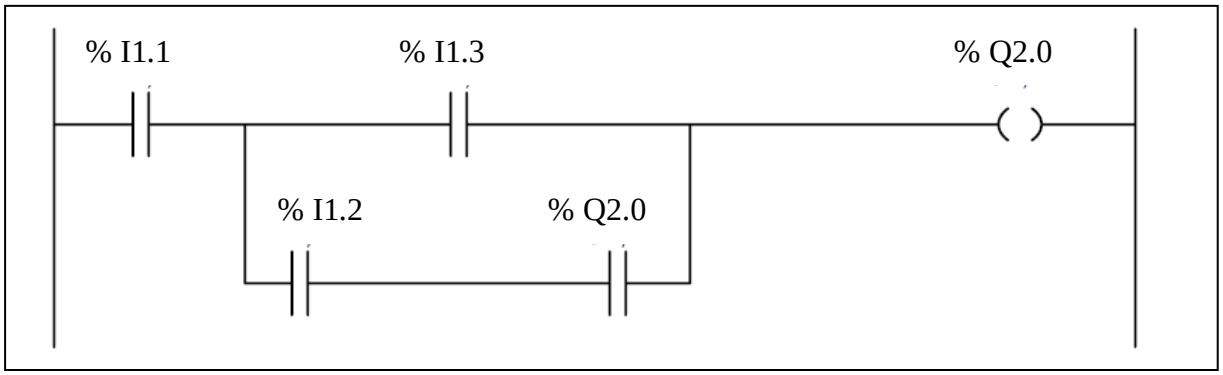
IV.1.6 Exemple 6 : Transcription du langage IL vers le langage LD

➢ ***Cahier des charges :***

- A partir de la liste des instructions du programme Ladder ci-dessous déterminer.

N° de ligne	Instruction	Opérande	Commentaire
00	LD	% I1.1	Lire l'entrée
01	AND(	% I1.3	Exécuter un ET , On imbrique une parenthèse.
02	OR(	% I1.2	Exécuter un OU avec la ligne précédente
03	AND	% Q2.0	Exécuter un NON ET
04	)		On ferme la parenthèse
05	)		On ferme la parenthèse
06	ST	% Q2.0	Activer la sortie

➢ ***Schéma en langage LD de la transcription du programme IL .***



IV.1.7 Problèmes à programmer

IV.1.7.1 Problème 1 : Démarrage d'un moteur asynchrone

➢ ***Cahier des charges :***

On se proposer de programmé le démarrage d'un moteur asynchrone un seul sens de rotation, dont le circuit de commande est donné par la Figure (II.19) .

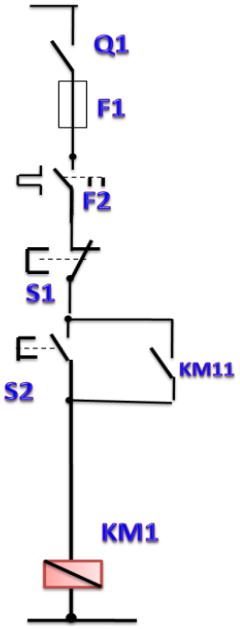


Fig II. 19: Circuit de commande (logique câblée).

✓ Les entrées du système sont : S1 son adresse API est %I1.0
S2 son adresse API est %I1.1

✓ La sortie du système est : KM1 son adresse API est %Q2.0Le contact auxiliaire KM11 d'auto maintien sera pris comme bit mémoire (image) de la sortie KM1, son adresse API est %Q2.0, Le fusible ainsi que le contact Q du sectionneur, le contact du relais

thermique ne seront pas utilisés comme des entrées directes, mais ils seront câblés en série avec le commun de l'alimentation de l'interface de sortie.

- ***Programmation en langage à contact ou Ladder***

Le réseau à contact s'inscrit entre deux barres verticales représentant la tension d'alimentation Exemple de schéma à contact programmable (sous millenium), Figure (II. 3).

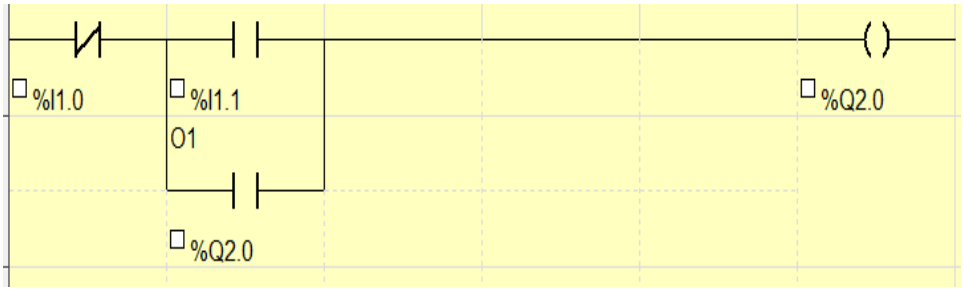


Fig II. 20: Programme en langage LD du circuit de commande du moteur asynchrone.

- ***Programmation en langage liste d'instructions***

L'adresse ou le code opérande est précédé de %.

Exemple d'écriture N° de ligne	Instruction	Opérande	Commentaire
00	LDN	% I1.0	tester le bouton S1 (entrée d'adresse %I1.1)
01	AND(	% I1.1	Exécuter un ET avec Le bouton marche S2 (%I1.1) on imbrique une parenthèse .
02	OR	% Q2.0	Exécuter un OU avec KM (mémoire Q2.0)
03	)		En ferme la parenthèse
04	ST	% O 2.0	Activer la sortie du contacteur KM1 (Q2.0)

- ***Programmation en langage FBD***

Il est similaire aux circuits électroniques à base de portes logiques, Figure (II. 4).

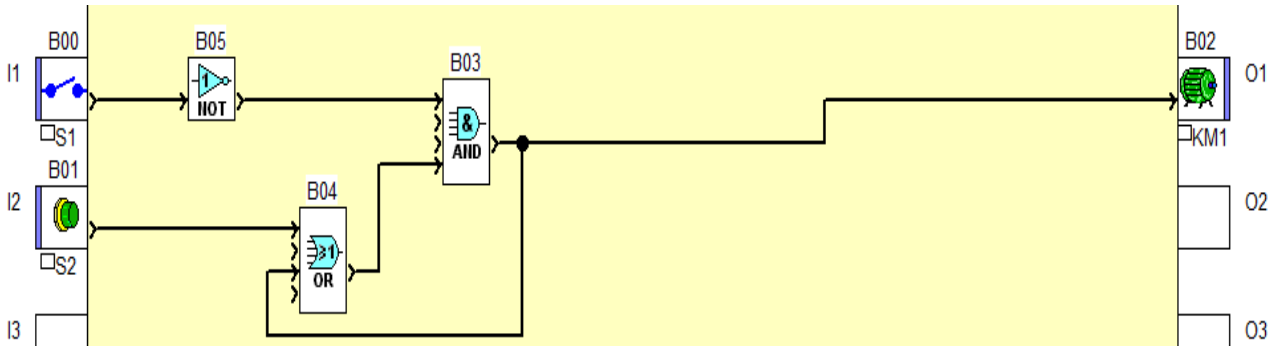


Fig II. 21: Programme en langage FBD du circuit de commande du moteur asynchrone.

IV.1.7.2 Problème 2 : Etude et commande par API d'un poste de perçage

- ***Cahier des charges***

On se propose de programmé, sous langage SFC, la solution GRAFCET d'un système automatisé « Poste de perçage », Figure (II. 22).

Fig II. 22: Système de perçage automatique.

➢ ***Fonctionnement :***

L'appui sur le bouton Départ cycle (Dcy) lance le cycle suivant :

- ✓ Serrage de la pièce à percer ;
- ✓ Perçage de la pièce ;
- ✓ Desserrage de la pièce.

▪ ***Remarque :***

Le cycle ne peut être lancé que si la pièce à percer est présente, à l'arrêt les fins de courses **s0** et **t0** sont actionnés.

➢ ***Fonctionnement en tenant compte des solutions technologiques utilisées :***

- Le capteur (**Cp**) indique la présence de la pièce à percer;
- L'appui sur le bouton Départ cycle (**Dcy**) lance le cycle ;
- Le vérin de serrage (**S**) déplace la pièce pour la serrer; le capteur (**s1**) indique que la pièce
- est serrée ;
- Le moteur supportant le foret (**MT**) commence à tourner et le vérin (**T**) pousse le moteur
- vers le bas ;
- L'action sur le capteur (**t1**) indique que la pièce est percée, d'où la remonté du vérin (**P**) ;
- Quand le capteur (**t0**) est actionné, le moteur (**MT**) et le vérin (**T**) sont arrêtés , par contre Le vérin (**S**) retourne dans l'autre sens ;
- L'action sur le capteur (**s0**) indique que la pièce est desserrée, et le cycle est terminé (on revient à l'état initial).

- Afin d'améliorer le fonctionnement on doit prévoir le fonctionnement cycle par cycle (Cy/cy)et cycle continu ou automatique (AQ), ainsi qu'un bouton d'initialisation de l'étape initiale.

On demande en premier lieu le GRAFCET du point de commande et par la suite la programmation de la solution trouvée en langage SFC.

➢ ***Solution graphique par GRAFCET***

La solution GRAFCET proposée, Figure (II.6) :

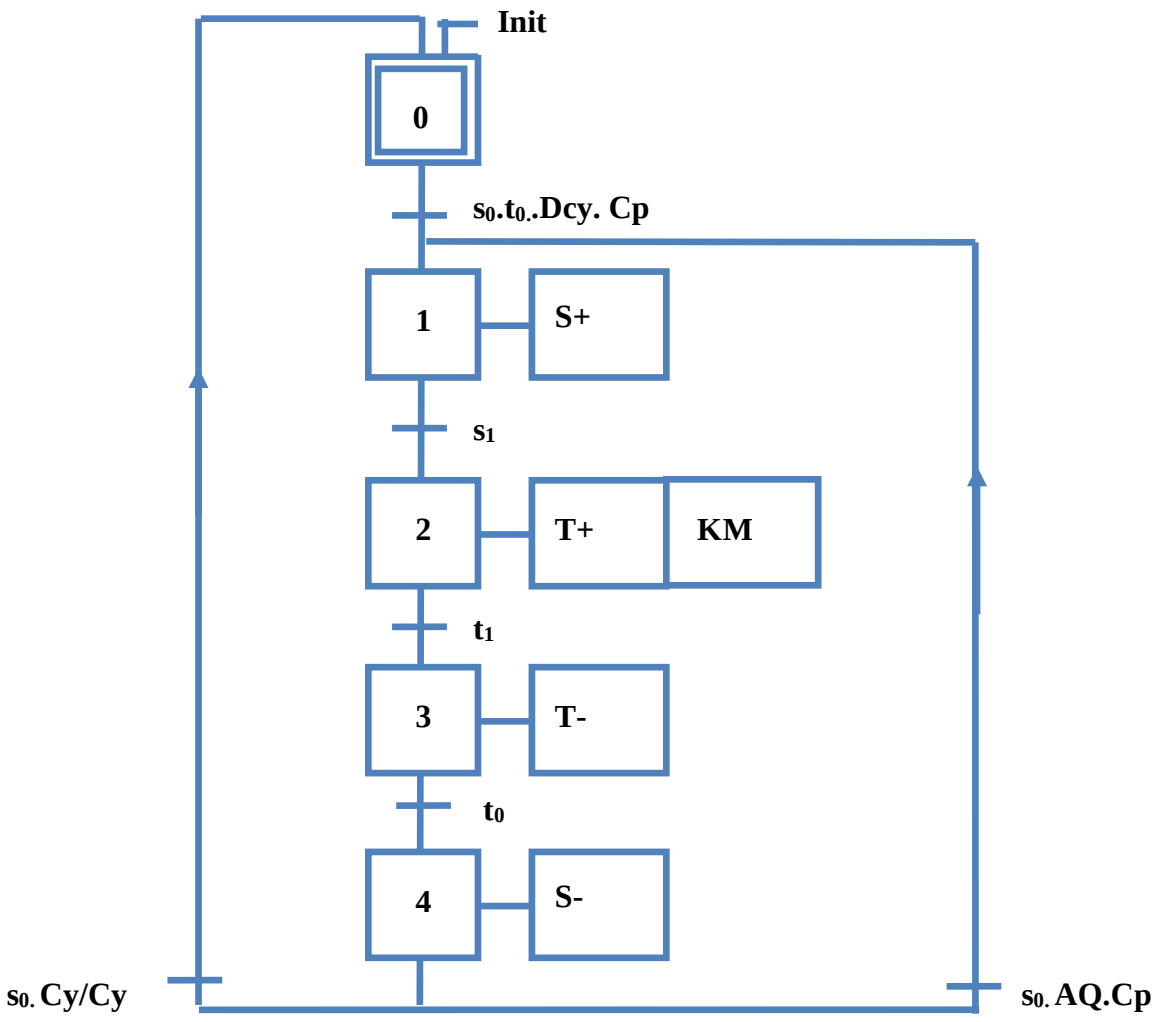


Fig II. 23: Solution GRAFCET du système de perçage automatique.

➢ ***Programmation de la solution en langage SFC***

La Figure (II.24) illustre la programmation de la solution GRAFCET proposée en langage SFC sous le logiciel Millénium de API Crouzet,

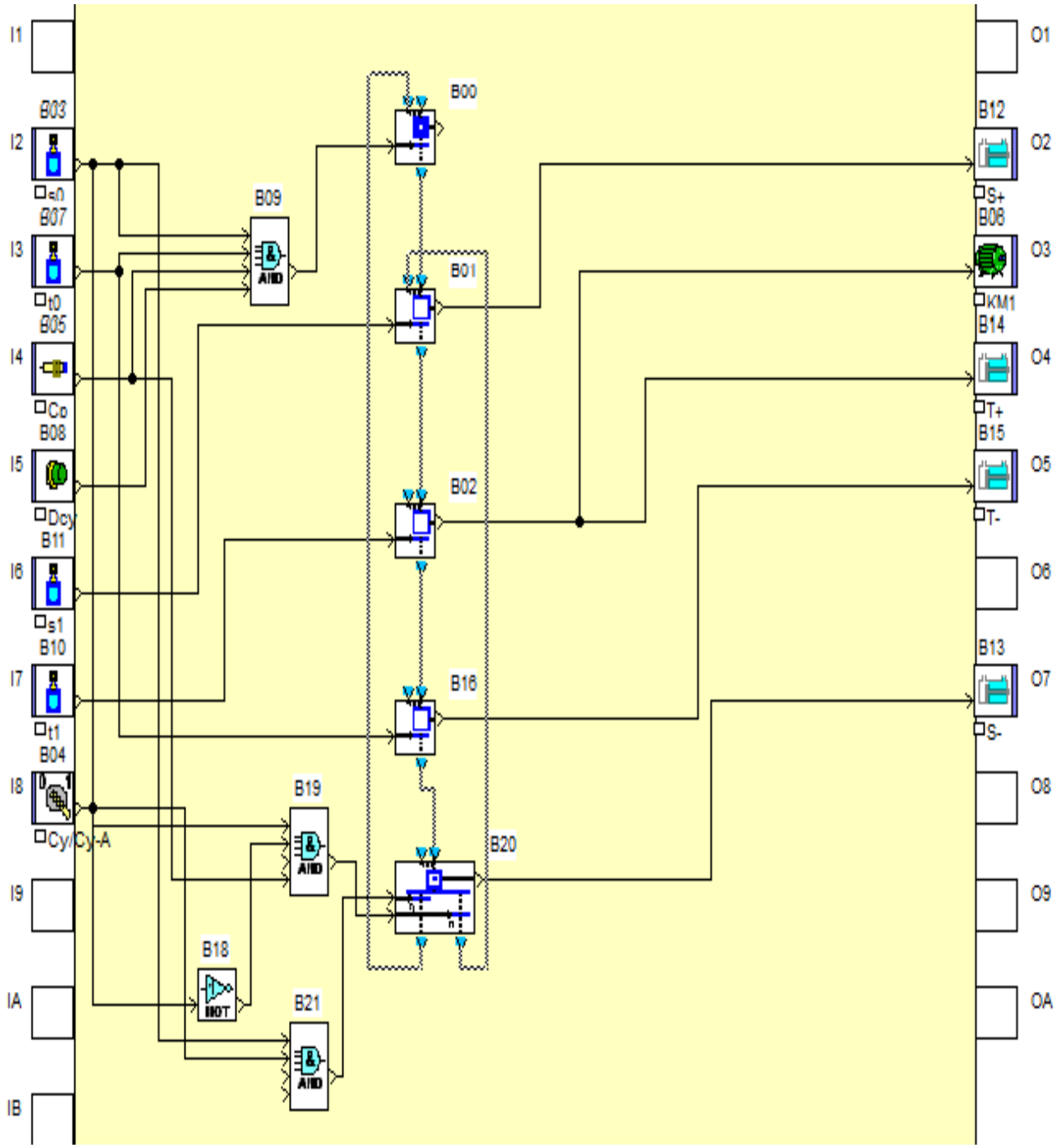
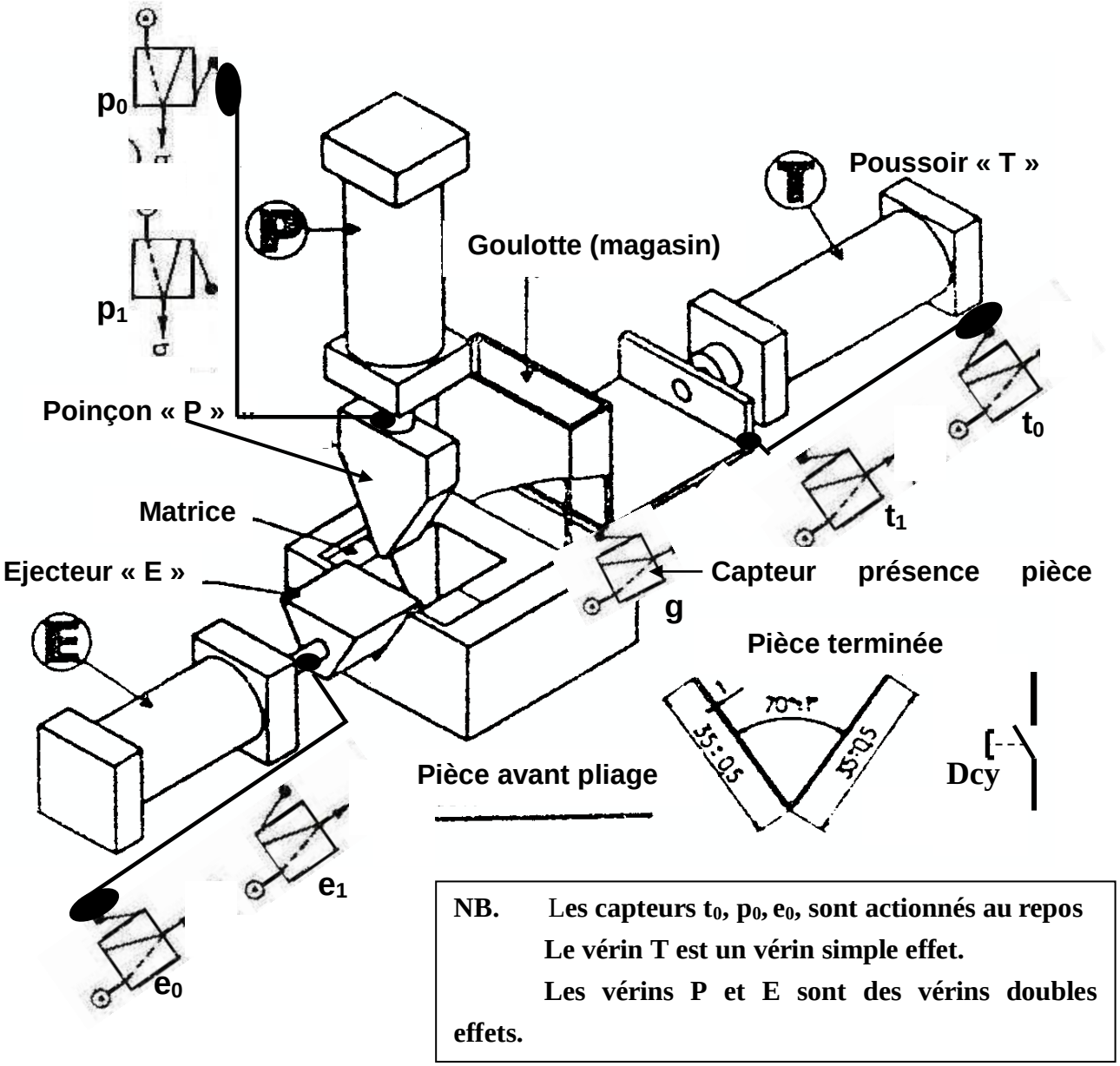


Fig II. 24: Programmation en SFC de la Solution GRAFCET du système de perçage automatique.

V.1.7.3 Problème 3 : Etude et commande par API d'un appareil à plier les tôles

- ***Cahier des charges***



NB. **Les capteurs t_0, p_0, e_0, sont actionnés au repos**
Le vérin T est un vérin simple effet.
Les vérins P et E sont des vérins doubles effets.

Cet appareil sert à plier des tôles.

Des tôles non pliées sont empilées dans une goulotte (magasin). Un capteur **(g)**, détecte la présence de tôles dans la goulotte.

➢ ***Fonctionnement :***

Lorsque l'opérateur actionne le bouton poussoir **(Dcy)**, avec présence de pièce **(g),** Les actions suivantes se produisent :

- Sortie de la tige du vérin **T** (Le poussoir **(T)** pousse la tôle sous le poinçon).
- Action sur t_1, la tige du vérin **P** sort pour plier la tôle.
- Action p_1, les tiges des vérins **P** et **T** rentrent.
- Action sur t_0 et p_0, la tige du vérin **E** sort pour éjecter la tôle pliée.
- Action sur e_1, la tige du vérin **E** rentre.

- Action sur $\mathbf{e_0}$, le cycle se termine.

➢ ***Travail demandé :***

a) Complétez le tableau des entrée et sorties avec les adresse API correspondantes, Donnez le GRAFCET du point commande et du point de vue automate;

b) Ecrire les équations de sorties (préactionneurs),;

c) Programmez le GRAFCET (chart) , les réceptivités et les actions (programmation postérieur), (Remarque pas de programmation préliminaire) ,

d) Complétez le schéma de câblage de l'automate.

a) Tableau Affectation des entrées / sorties

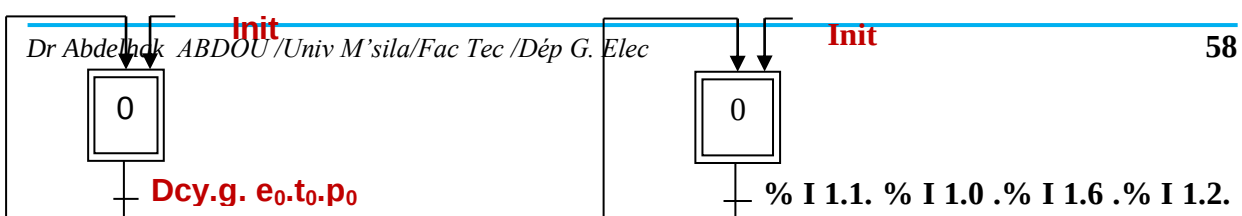
Désignation	Repère	Affectation adresse API	
Présence pièce	**g**		Boutons Poussoirs et capteurs
Bouton Poussoir départ cycle	**Dcy**		
Fin de course rentrée de tige vérin T	$\mathbf{t_0}$		
Fin de course sortie tige vérin T	$\mathbf{t_1}$		
Fin de course rentrée tige vérin P	$\mathbf{p_0}$		
Fin de course sortie tige vérin P	$\mathbf{p_1}$		
Fin de course rentrée tige vérin E	$\mathbf{e_0}$		
Fin de course sortie tige vérin E	$\mathbf{e_1}$		
P : Vérin de pliage de la tôle	**P+**		Pré actionneurs
	P-		
E : Vérin éjecteur de la tôle pliée	**E+**		
	E-		
T : Vérin de translation de la tôle	**T**		

➢ ***Solution graphique par GRAFCET***

a) Tableau d'adressage des entrées et sorties de l'API.

Désignation	Repère	Affectation adresse API	
Présence pièce	g	% I 1.0	Boutons Poussoirs et capteurs
Bouton Poussoir départ cycle	Dcy	% I 1.1	
Fin de course rentrée de tige vérin T	t_0	% I 1.2	
Fin de course sortie tige vérin T	t_1	% I 1.3	
Fin de course rentrée tige vérin P	p_0	% I 1.4	
Fin de course sortie tige vérin P	p_1	% I 1.5	
Fin de course rentrée tige vérin E	e_0	% I 1.6	
Fin de course sortie tige vérin E	e_1	% I 1.7	
P : Vérin de pliage de la tôle	P+	%Q 2.0	Pré actionneurs
	P-	%Q 2.1	
E : Vérin éjecteur de la tôle pliée	E+	%Q 2.2	
	E-	%Q 2.3	
T : Vérin de translation de la tôle	T	%Q 2.4	

b) GRAFCET point de vue partie commande (PC) : c) GRAFCET point de vue API



d) Ecrire les équations des sorties :

P+ .= %X2 **P-** = %X3

E+ .= %X4 **E-** = %X5

T .= %X1 + %X2

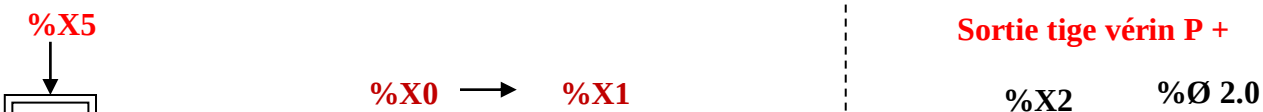
On peut aussi écrire directement les sortie selon leurs adresse API

% Ø 2.0 .= %X2 **% Ø 2.1** = %X3

% Ø 2.2 = %X4 **% Ø 2.3** = %X5

% Ø 2.4 = %X1 + %X2

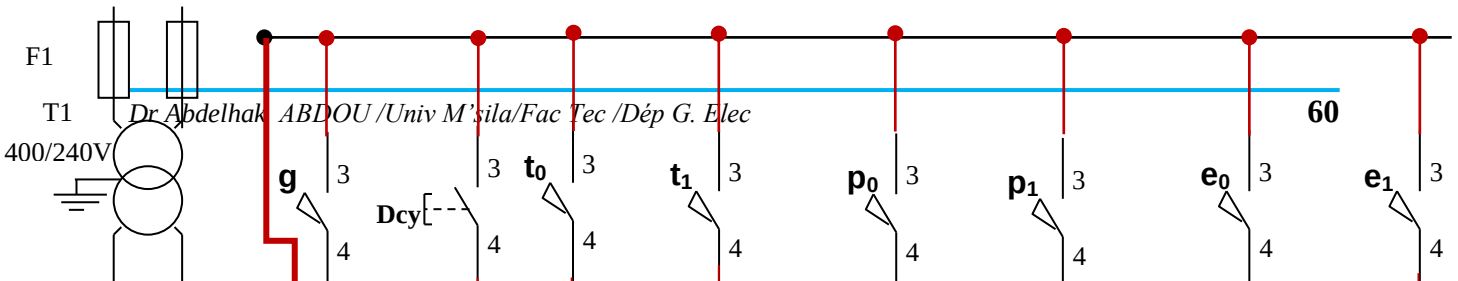
e) Document réponse programmation de réceptivités et action en langage GRAFCET sous PL7,



Réceptivités (entrées) Actions (sorties)

Programmation Postérieur

e) Schéma de câblage de l'API en logique positive



IV.1.7.3 Annexe 1 : Adressage des Objets Bits et Mots

PRESENTATION GENERALE

QU'EST QU'UN OBJET

Un objet est une entité pouvant être manipulée par programme, ce peut être une image d'entrée, un élément d'un temporisateur, un élément du système, un élément de communication etc.

REPRESENTATION DES OBJETS

Les objets sont représentés par le symbole « % » suivit d'une ou deux lettres précisant leurs type puis d'une lettre précisant leur format (bits, octet, mots , double, réel etc.)

Exemples :	**%IW**	% : objet	**I**	: Type image d'entrée	**W**	: Format mot
	%MB	% : objet	**M**	: Type mémoire interne	**B**	: Format octet
	%Q ou %Qx	% : objet	**Q**	: Type Image de sortie	**X**	: Format bit

LISTE DES DIFFERENTS TYPES D'OBJETS

♦ OBJETS D'ENTREES	**%I**	*Images des entrées process*
♦ OBJETS DE SORTIES	**%Q**	*Images des sorties process*
♦ VARIABLES INTERNES	**%M**	*Mémoire utilisateur*
♦ VARIABLES GRACETS	**%X**	
♦ CONSTANTES	**%K**	*Mémoire constante ou de configurations*
♦ VARIABLES SYSTEME	**%S**	*Etats ou actions sur le système*
♦ VARIABLES RESEAUX (FIPWAY)	**%N**	*Mots communs échangés automatiquement*

Les types des objets des blocs fonctions prédéfinis (Pas de précision de format)

♦ TEMPORISATEURS	**%TM**	*Repos, travail*
♦ TEMPORISATEURS SERIE 7	**%T**	*Compatible série 7*
♦ MONOSTABLES	**%MN**	*Monostables « retriggerables »*
♦ COMPTEURS	**%C**	*Comptage, décomptage*
♦ REGISTRES	**%R**	*Pile FIFO ou LIFO*
♦ PROGRAMMATEURS tambour	**%DR**	Programmateurs cycliques à

Les types DFB (Pas de précision de format)

LES FORMATS DES OBJETS

BITS	**X** ou rien	0 - 1
OCTETS	**B**	Code ASCII uniquement
MOTS	**W**	16 bits signé ou pas
MOTS DOUBLES	**D**	32 bits signé
FLOTTANT	**F**	-3.402824E+38 et -1.175494E-38 et 1.175494E-38 et 3.402824E+38

15/77

LES ELEMENTS DES OBJETS

La plupart des objets sont constitués de plusieurs éléments ou sous éléments, par exemple, un compteur contient une valeur courante, une valeur de présélection, deux bits de dépassements et un bit de présélection atteinte

Les cartes d'entrées TOR disposent des bits représentant les états physiques des capteurs, mais aussi d'un bit de défaut du module et un bit de défaut par voie, certaines cartes disposent aussi de mots ou de tableaux de mots représentant leurs états ou leurs configurations.

On accède à un élément d'un objet en mettant un point derrière le numéro de l'objet puis l'identificateur de l'élément (un numéro ou un symbole réservé), puis si l'élément contient d'autre sous éléments on ajoute un point suivi de l'Identificateur etc.

Exemples :	**%I0.1**	Module à l'emplacement **0**, la voie **1**
	%I0.1.ERR	Module à l'emplacement **0**, la voie**1**,**défaut module**
	%C5.V	Le compteur **5**, la valeur courante
	%X3	L'étape **3**
	%X3.T	L'étape **3**, sont temps d'activité
	%IW0.12.2	Module à l'emplacement **0**, voie **12**, mot **2**

OBJETS DES CARTES

Pour identifier un objet d'une carte, il faut préciser son numéro de module puis le numéro de voie.
Si plusieurs objets sont associés à une voie, il faudra préciser le numéro d'objet. (sauf pour l'objet N°0)

Exemples:	**%IW0.12.2**	Mot **2** de la voie **12** du module **0**
	%IW0.12	Mot **0** de la voie **12** du module **0**

LES OBJETS STRUCTURES

(bits extraits de mots, tableaux de bits, tableaux de mots)

BIT EXTRAIT DE MOTS

mot : Xi		i : numéro du bit
Exemples :	**%MW5 :X4**	Le mot interne **5**, le bit **4**
	%IW0.12.2 :X3	Le module **0**, la voie **12**, le registre **2**, le bit **3**

TABLEAUX DE BITS OU CHAINES DE BITS

bit de départ : nombre		nombre compris entre 1 et 32
Exemples :	**%I0.0 :16**	Module **0**, voie **0**, **16** bits (voie 0 à 15)
	%M5 :32	Bit **5**, **32** bits (%M5 à %M36)

TABLEAU DE MOTS

mot de départ : nombre		nombre compris entre 1 et maximum mémoire
Exemples :	**%MW10 :50**	Mot interne **10**, **50** mots (%MW10 à %MW59)
	%KW25 :100	Mot constant **25**, **100** mots (%KW25 à %KW124)
	%MD20 :10	Mot double **20**, **10** mots (%MD20 à % MD38)

Partie V *INITIATION AU LOGICIEL MILLENIUM 3 API CROUZET*

V.1 Introduction

Le logiciel Millenium 3 sert à programmer l'Automate Programmable Industriel (API) Crouzet.

On nomme **Automate Programmable Industriel**, (**API**, en anglais **P**rogrammable **L**ogic **C**ontroller, **PLC**) un dispositif similaire à un ordinateur, utilisé pour automatiser des processus comme la commande des machines sur une chaîne de montage dans une usine.

Là où les systèmes automatisés plus anciens emploieraient des centaines ou des milliers de relais et de cames, un simple automate suffit. On nomme automaticiens les programmeurs de ces Automates Programmables Industriels.

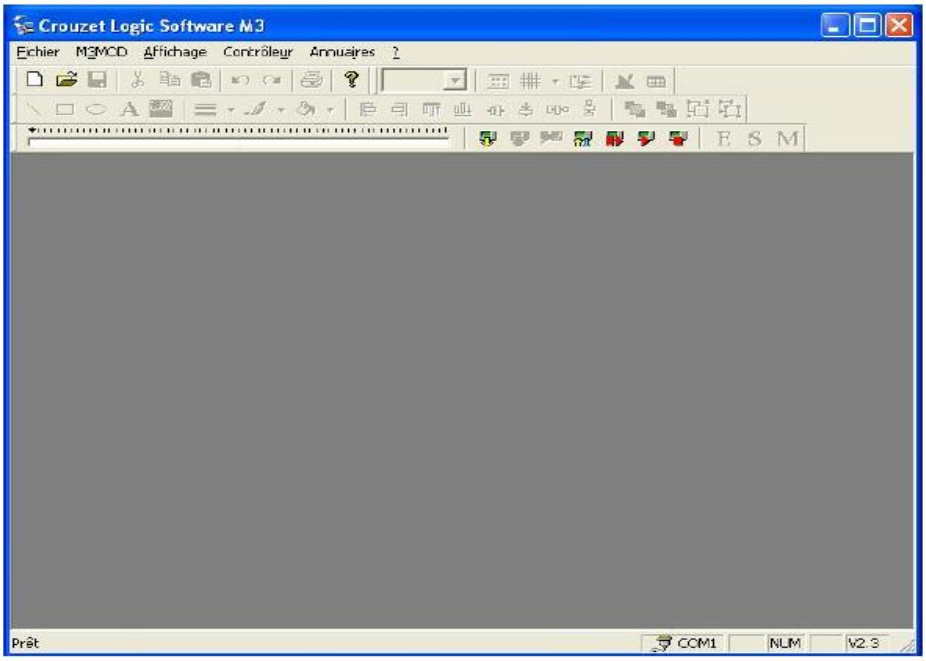
L'API est structuré autour d'une unité de calcul ou processeur (en anglais Central Processing Unit, CPU), d'une alimentation (depuis des tensions alternatives - AC ou continues - DC) et, de modules suivant les besoins de l'application, tel que:

- des cartes d'entrées - sorties (en anglais **I**nput - **O**utput, **I/O**) numériques (Tout ou rien)
- ou analogiques
- des cartes d'entrées pour brancher des capteurs (un capteur est un dispositif qui transforme l'état d'une grandeur physique observée en une grandeur utilisable,...), boutons poussoirs, ...
- des cartes de sorties pour brancher des actionneurs, voyants, vannes, ...
- des modules de communication pour dialoguer avec d'autres automates, des entrées/sorties déportées, des supervisions ou autres interfaces homme-machine
- des modules dédiés métiers, tels que de comptage rapide, de pesage...
- des modules d'interface pour la commande de mouvement, dits modules Motion, tels que démarreurs progressifs, variateur de vitesse, commande d'axes.

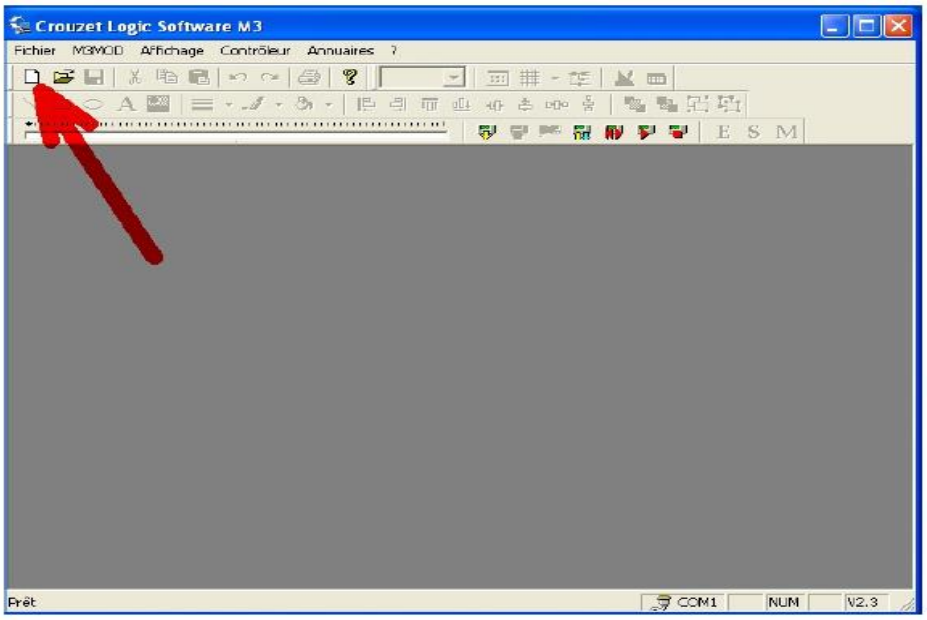
Nous allons étudier le fonctionnement du logiciel de programmation de cet automate. Une fois le programme élaboré et testé sur l'ordinateur, il pourra être transféré dans l'automate pour y être exploité.

V.2 CREATION D'UN NOUVEAU DOCUMENT DE TRAVAIL

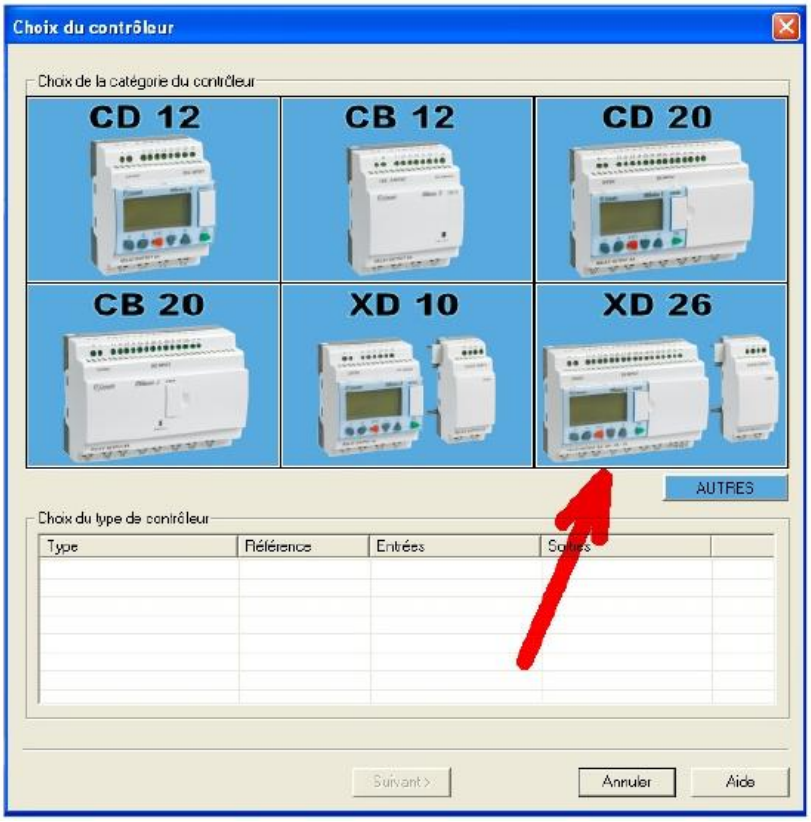
a. Lancez le logiciel "Millenium 3 AC".



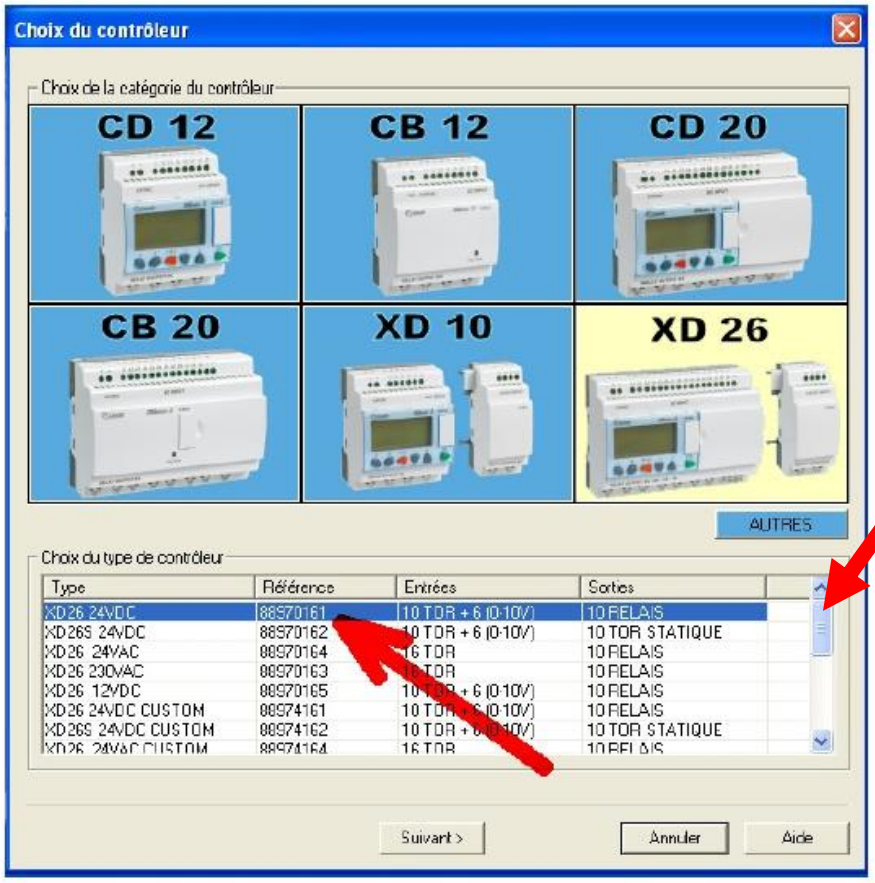
b. Cliquez sur "Nouveau document".



c. Vous devez choisir le type d'Automate : Exemple : cliquez sur XD26.

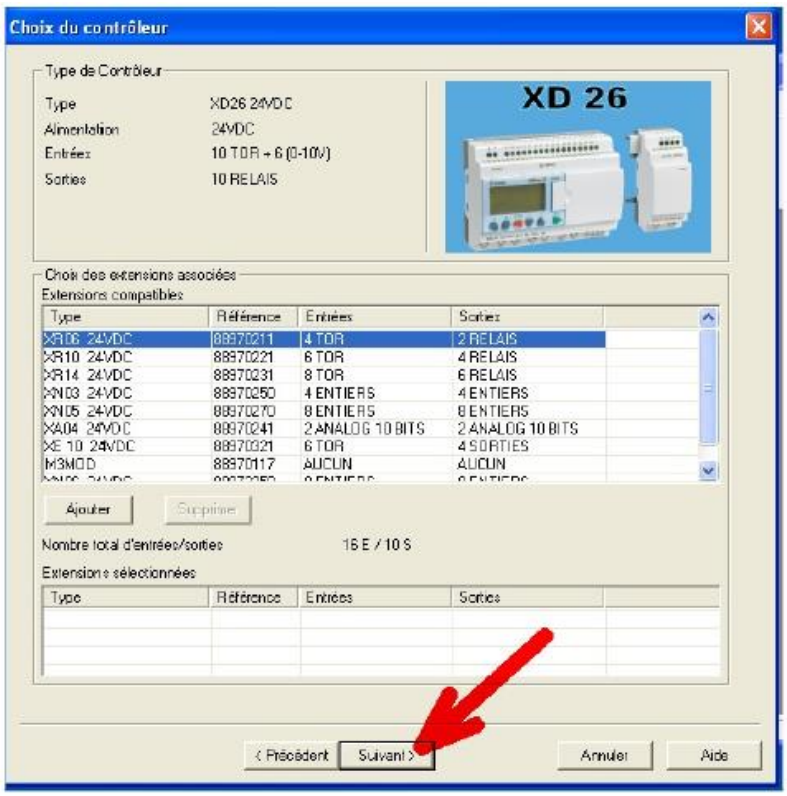


d. Précisez la référence exacte de l'Automate : Exemple : choisissez "88 970 161".

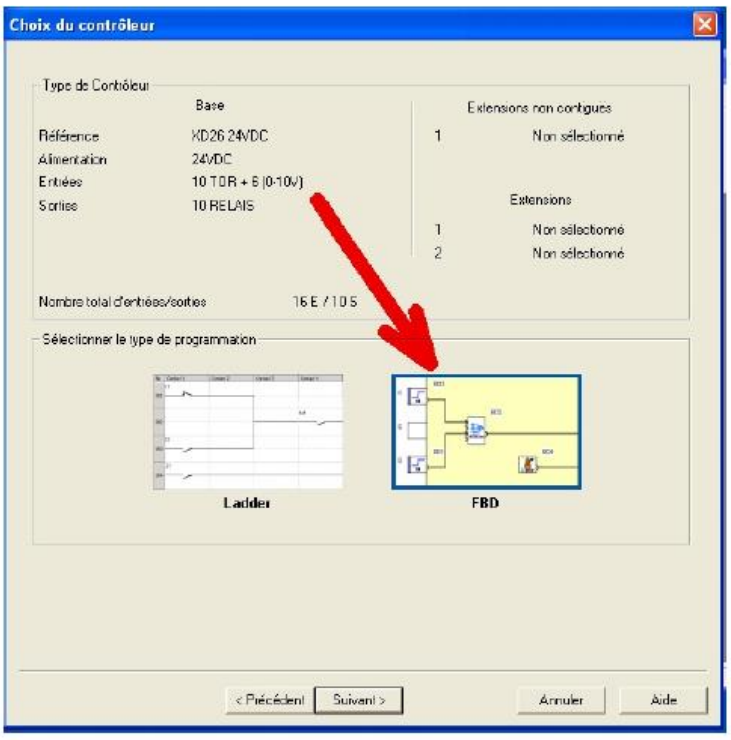


Ce type est caractérisé par :
- 10 entrées Tout Ou Rien (TOR) et 6 entrées analogiques,
- 10 sorties Tout Ou Rien (TOR).

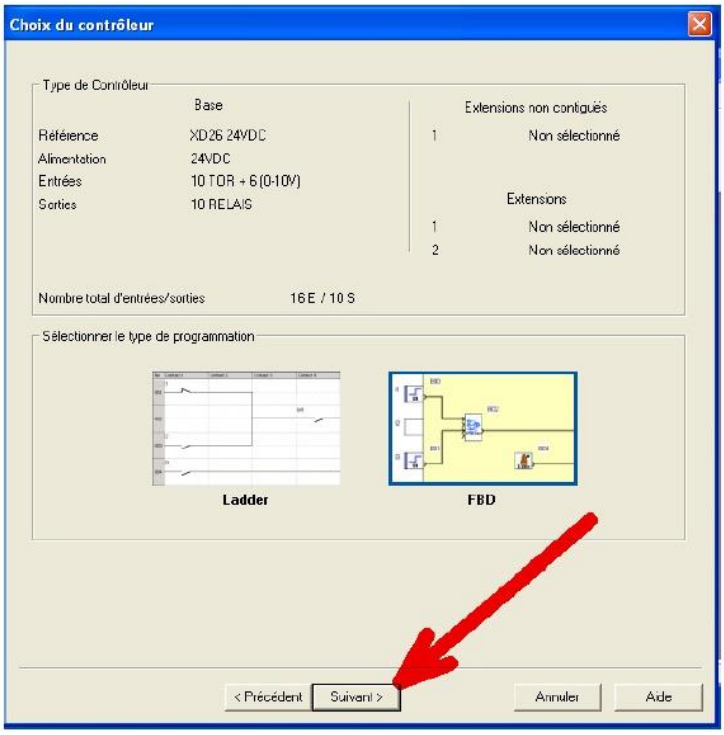
e. Il est possible d'ajouter des extensions sur l'Automatique (communication, entrées, sorties, etc..). Pour cette application, nul besoin d'ajouter d'extension : cliquez sur "**Suivant**".



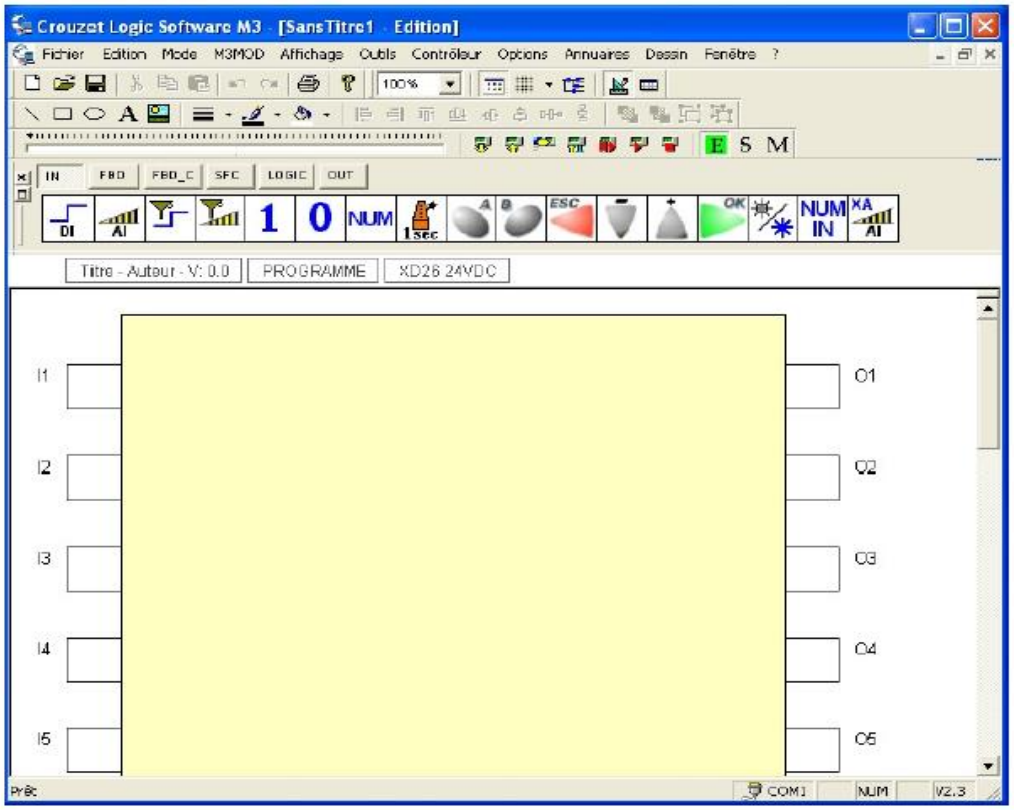
f. Sélectionnez le type de langage de programmation - langage à contact (Ladder) ou FBD (Functional Block Diagram). Pour cette initiation, cliquez sur "**FBD**".



g. Validez ce choix en cliquant sur "Suivant".

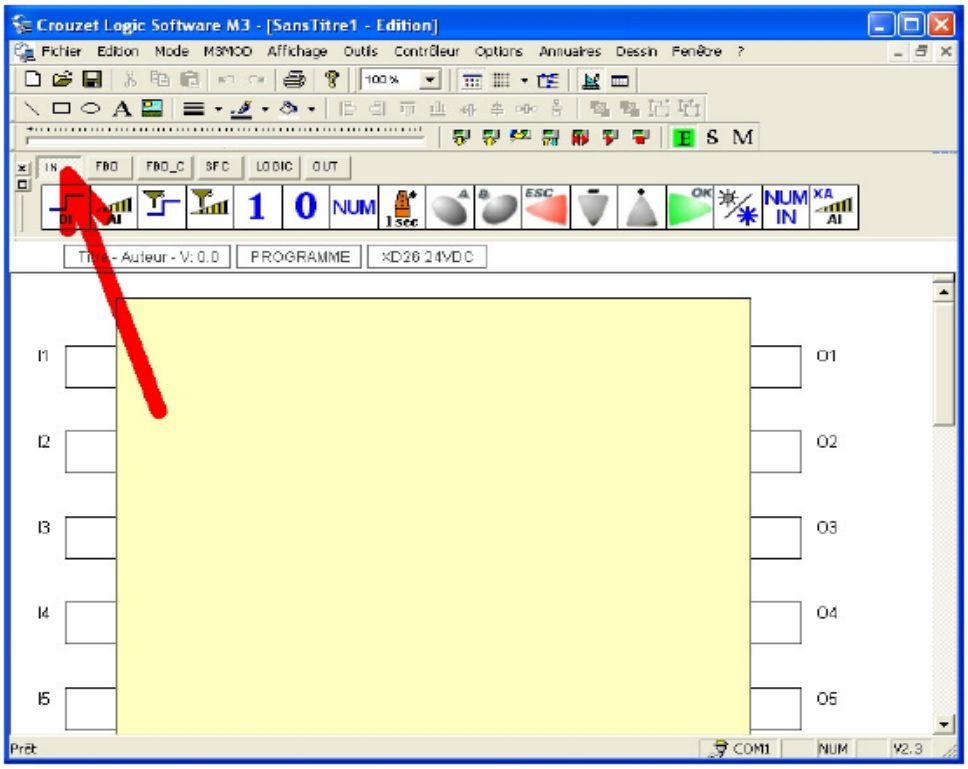


h. Voici le nouveau document configuré pour votre automate.

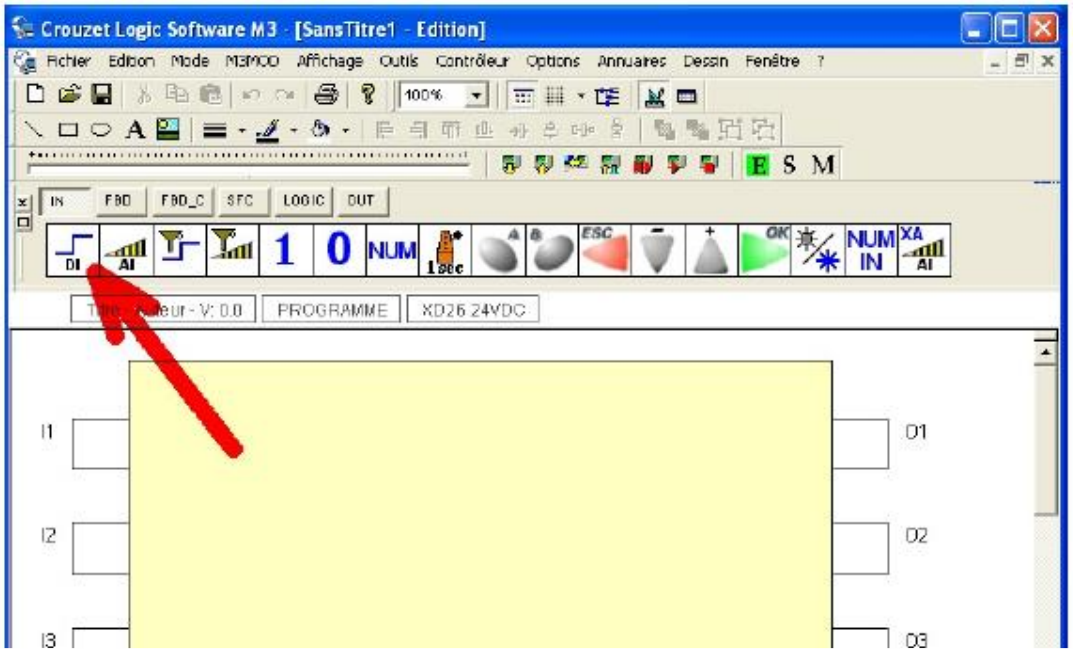


V.3 PROGRAMMATION DE BASE

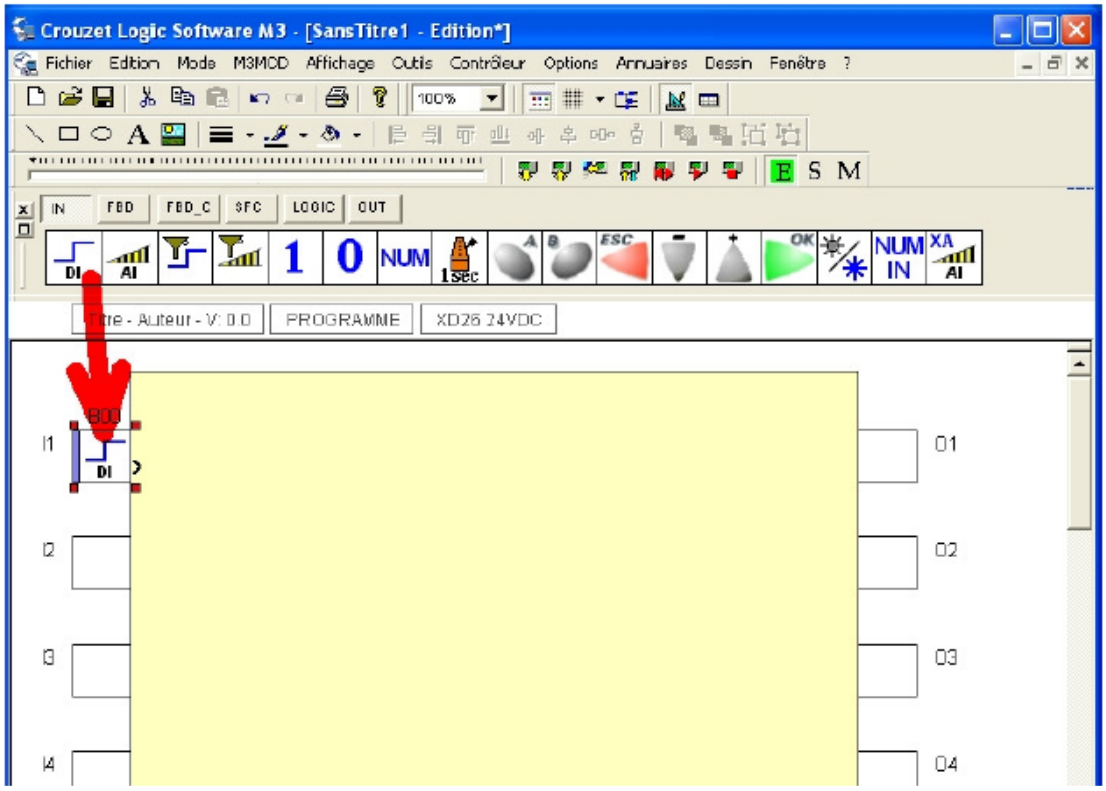
a. Une barre d'outils nommée "Barre de fonctions" permet d'insérer dans le document des éléments d'entrées (IN), de fonctions (FBD), de grafcet (SFC), de logique (LOGIC) ou de sorties (OUT). Choisissez pour commencer le menu de fonctions "IN".



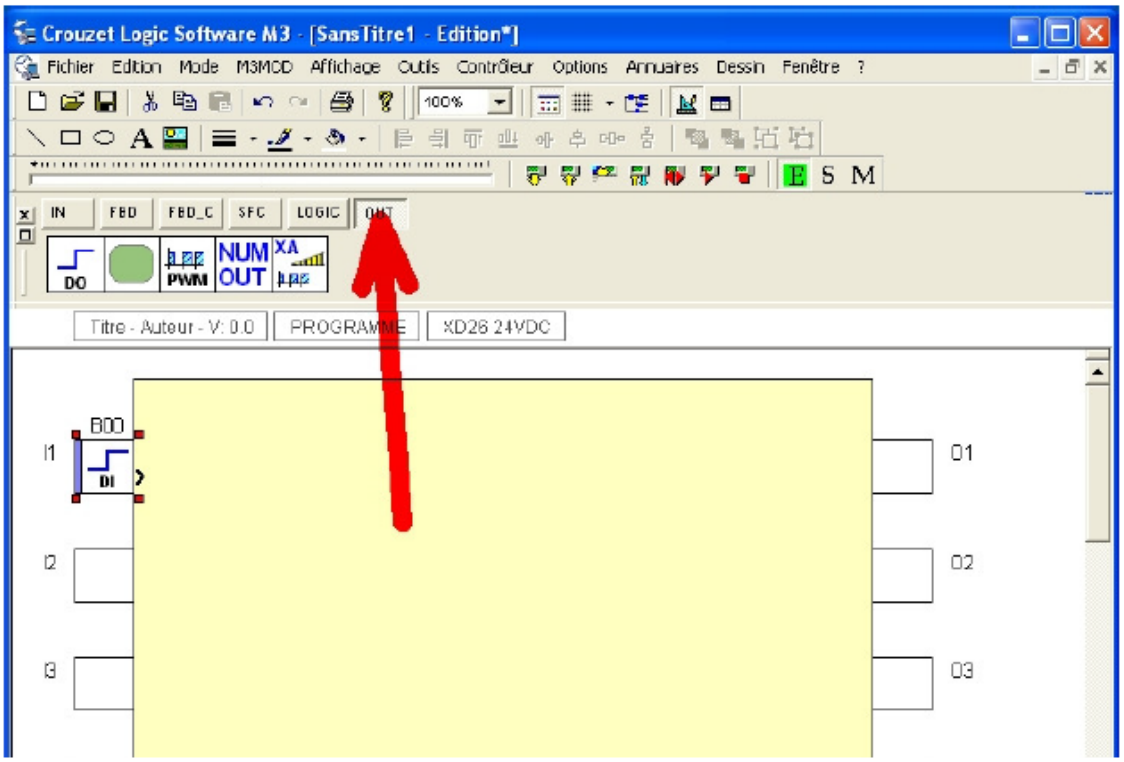
b. Nous allons placer une entrée binaire (de type Tout Ou Rien - TOR) sur notre document : elle se nomme "DI" (Digital Input).



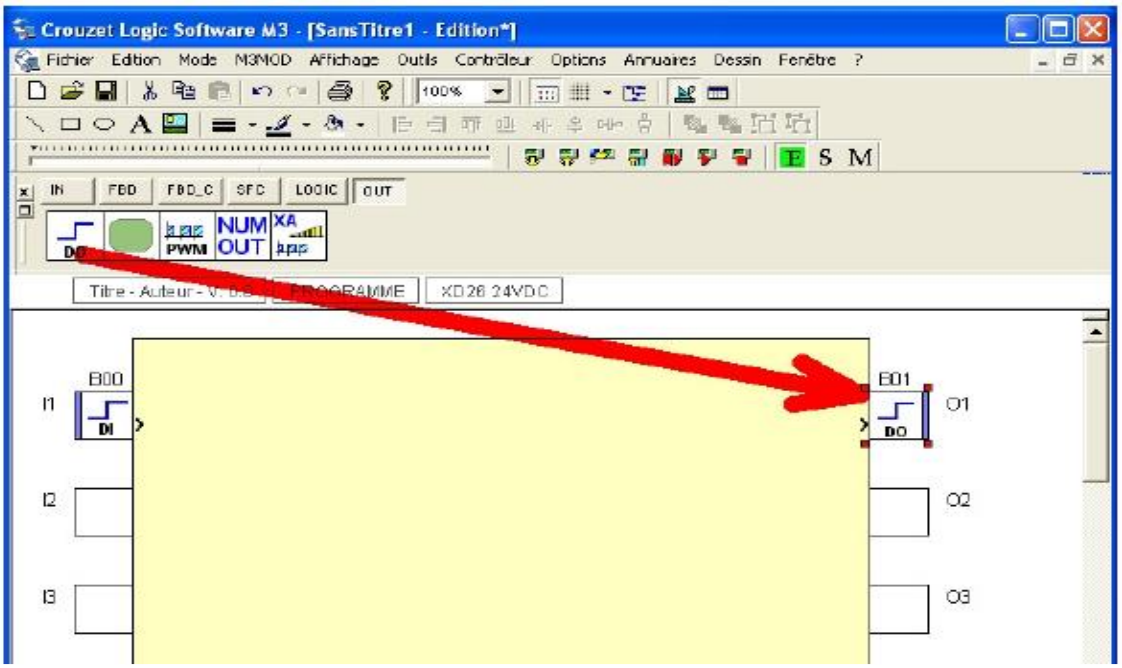
c. Faites glisser l'icône sur une des bornes d'entrées de l'automate (I1 à IA).



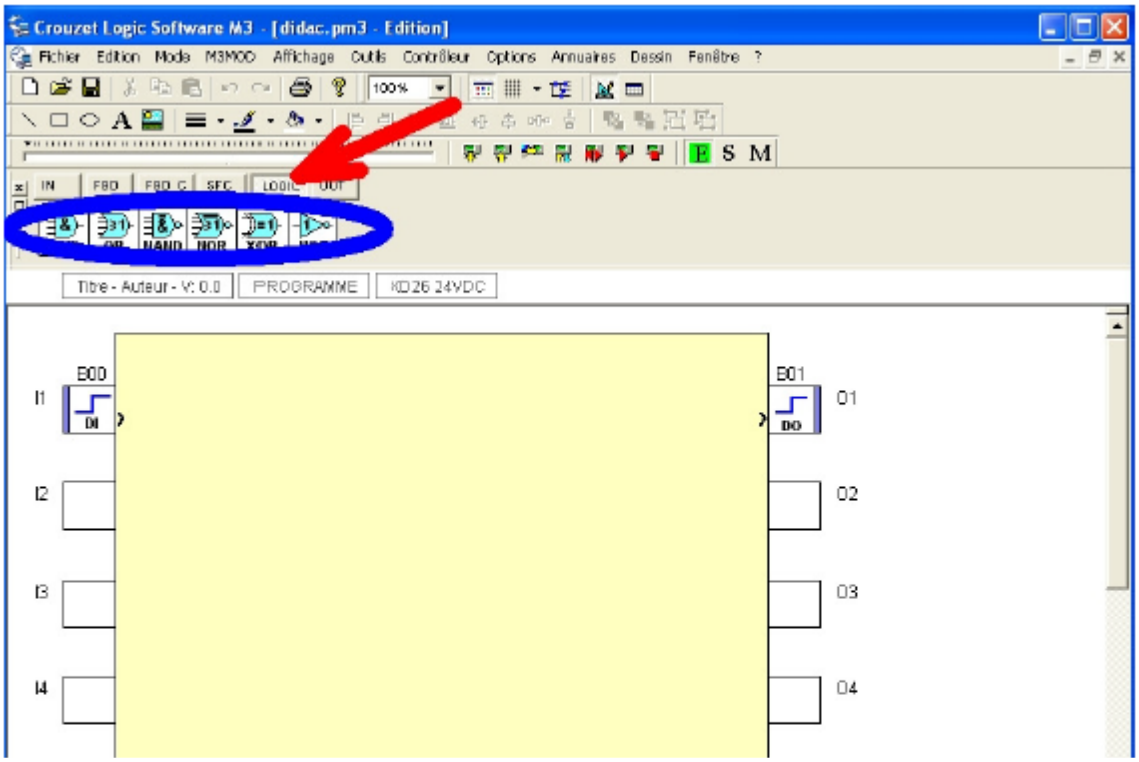
d. Choisissez le menu de fonctions "OUT" (sorties).



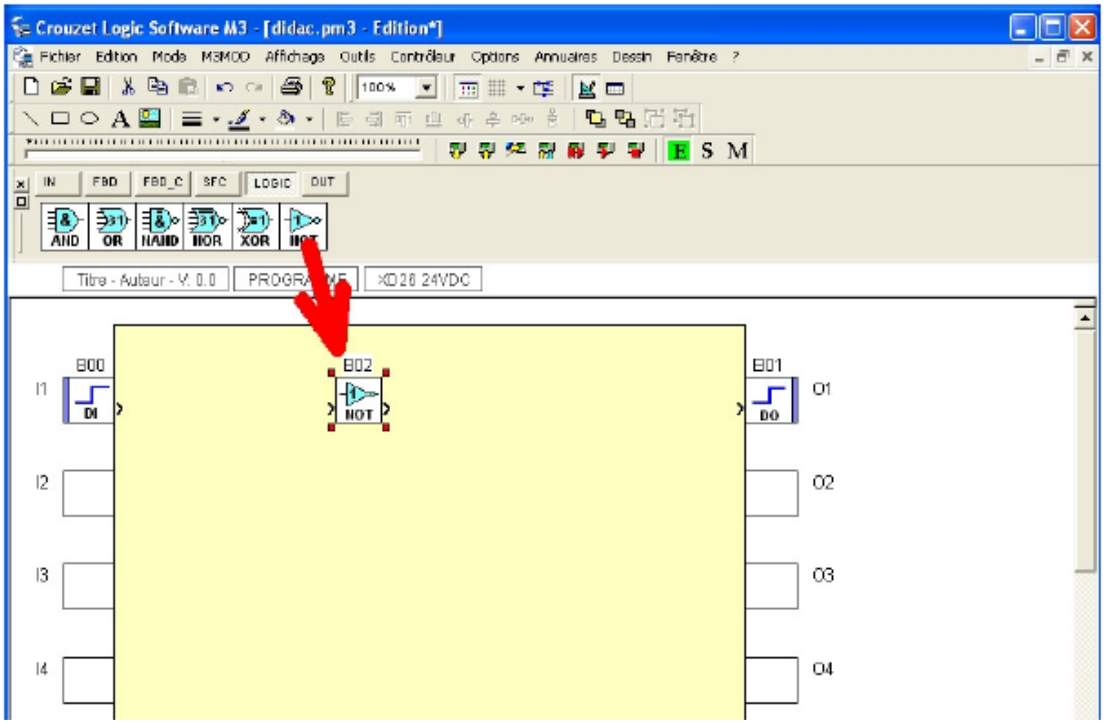
e. Faites glisser l'icône de sortie "DO" (Digital Output) sur une des bornes de sortie de l'automate (O1 à OA).



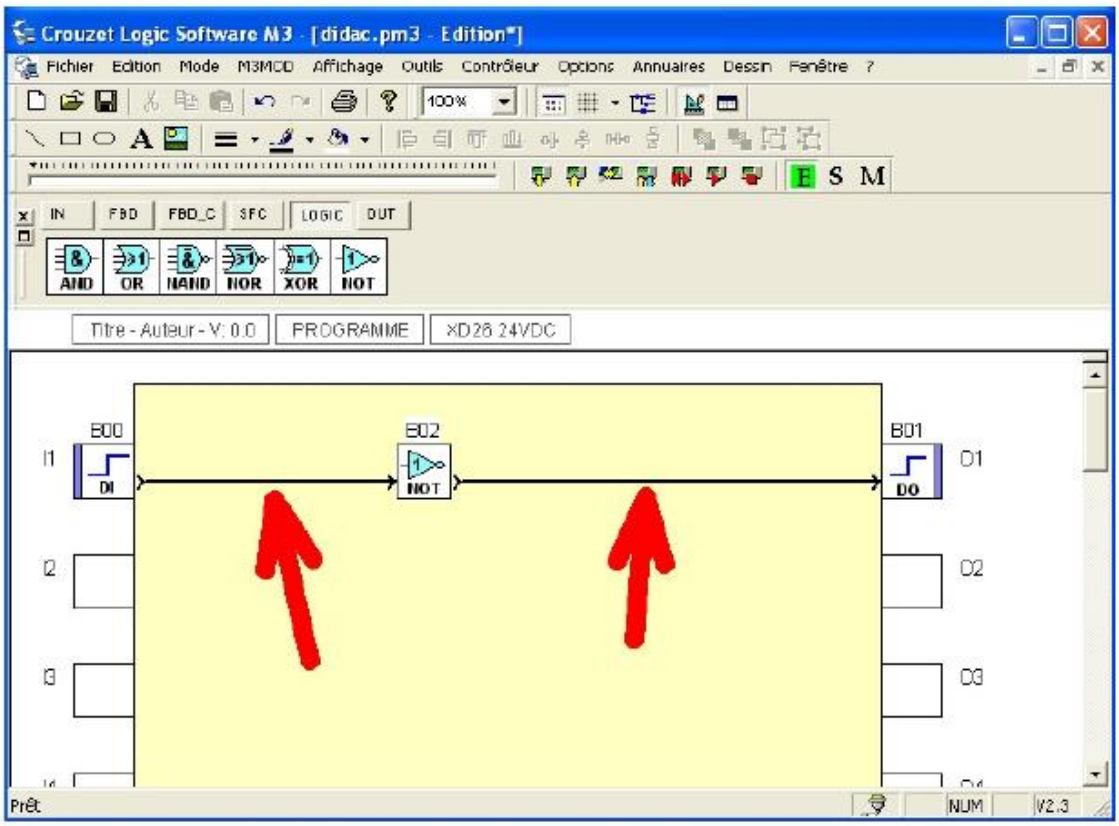
f. Choisissez le menu de fonctions "LOGIC".



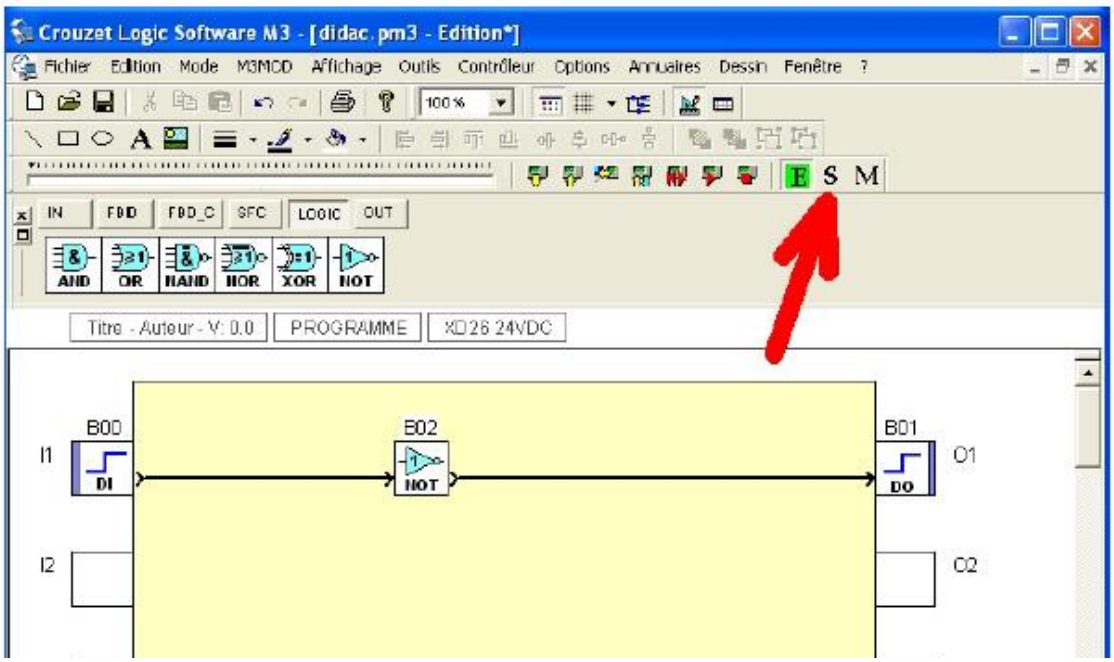
g. Faites glisser l'icône de l'inverseur logique (NOT) sur l'espace de travail.



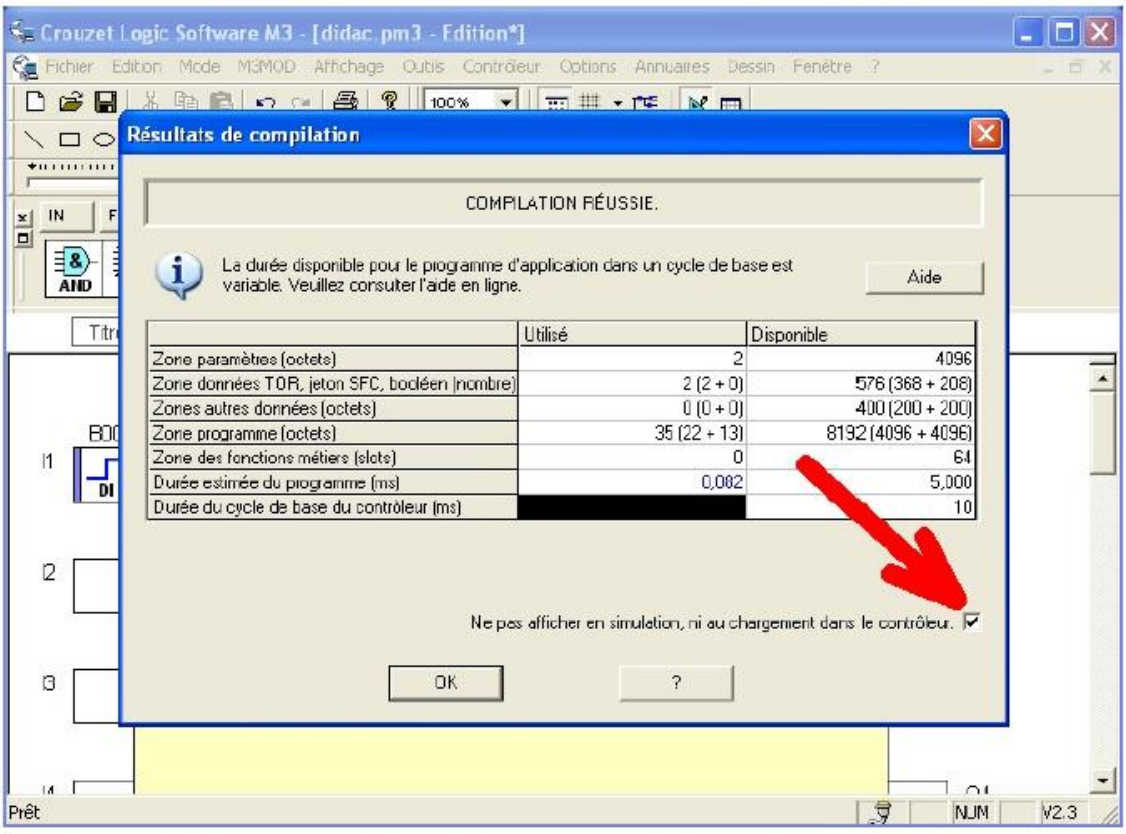
h. Effectuez les liaisons en cliquant-déplaçant votre souris depuis les bornes d'entrée/sortie vers les connexions de l'inverseur logique.



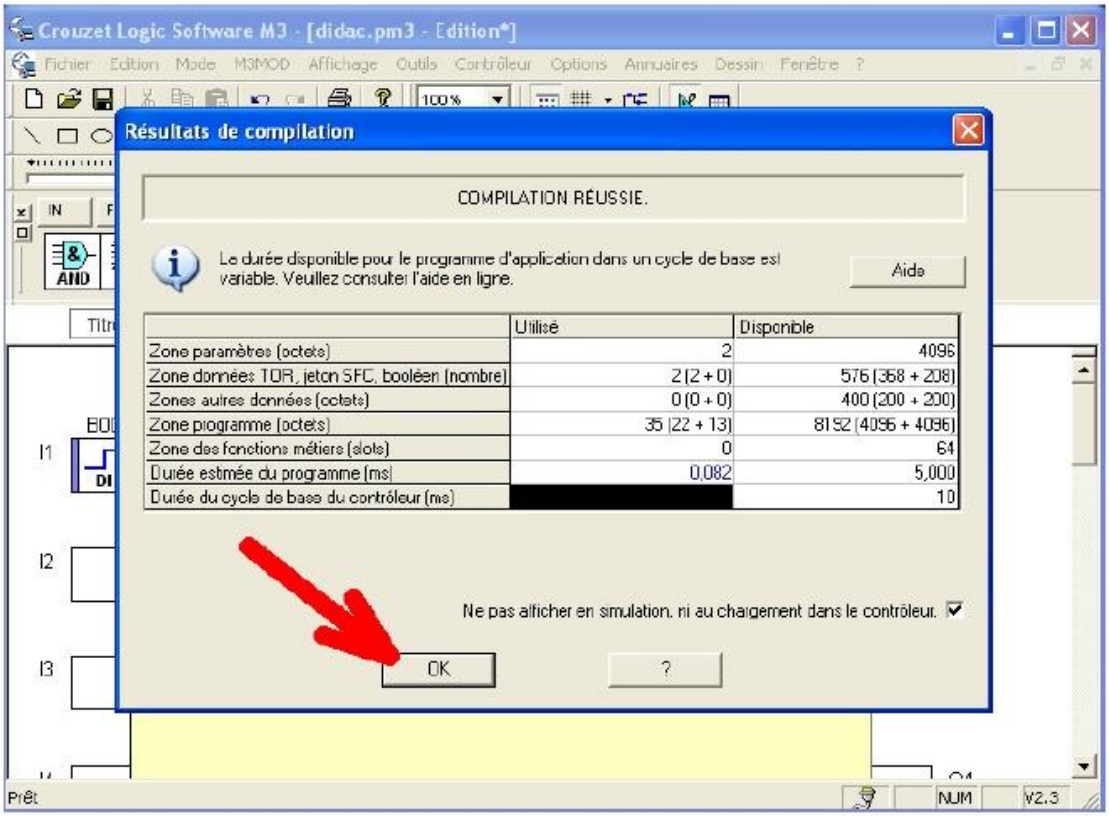
i. Votre circuit de base est terminé.. Il faut maintenant vérifier son fonctionnement à l'aide du simulateur (bouton "S").



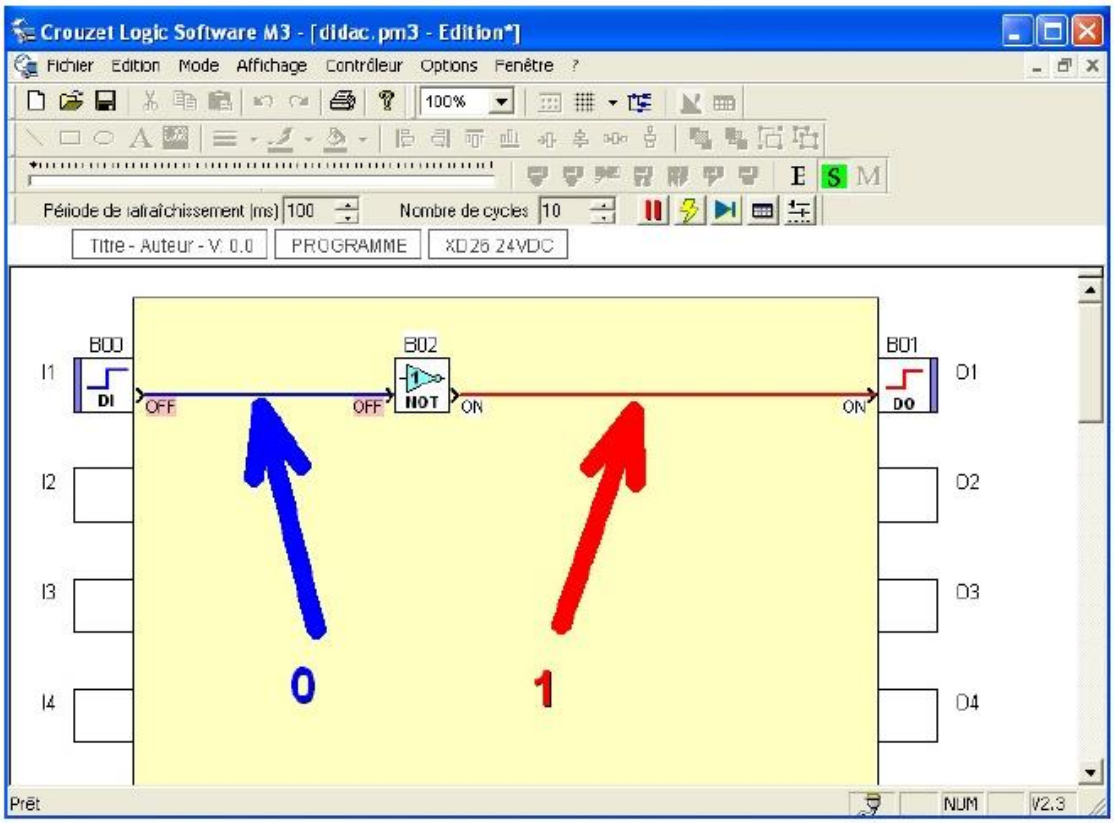
j. Lorsque vous lancez la Simulation, il se peut que le résultat de la compilation du programme apparaisse : il atteste que le programme écrit a été correctement "traduit" pour être exploité par le simulateur. Si vous ne voulez plus voir cette fenêtre, cochez l'option correspondante :



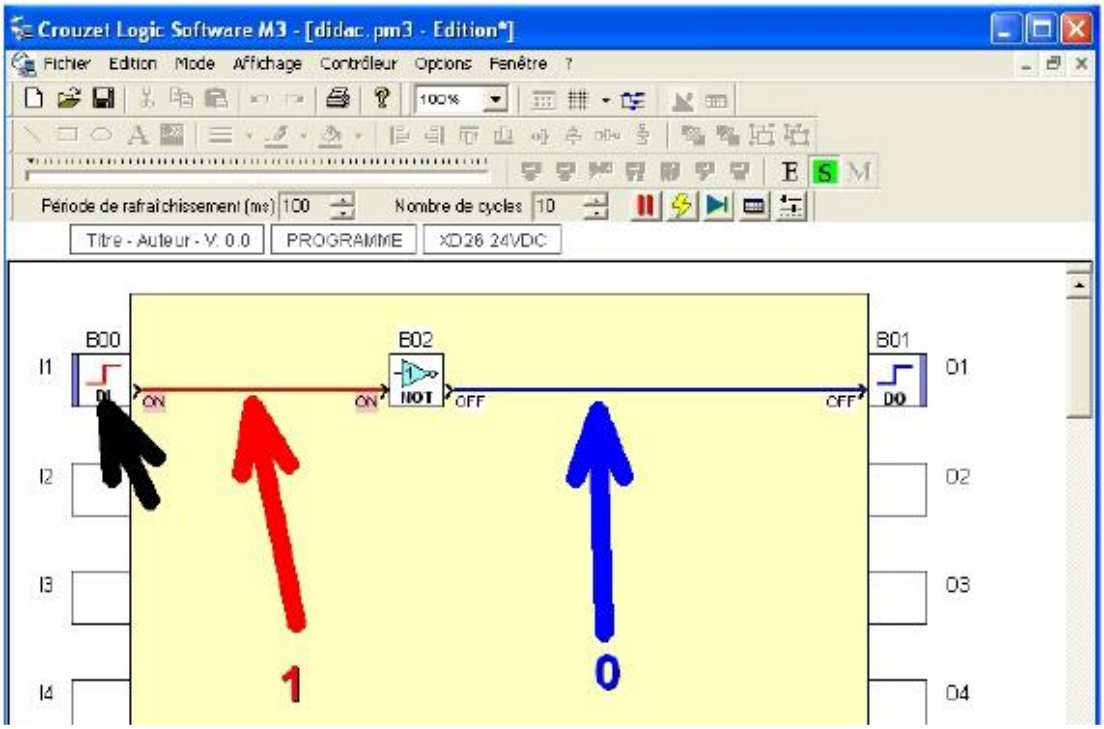
k. Validez ensuite en cliquant sur "OK".



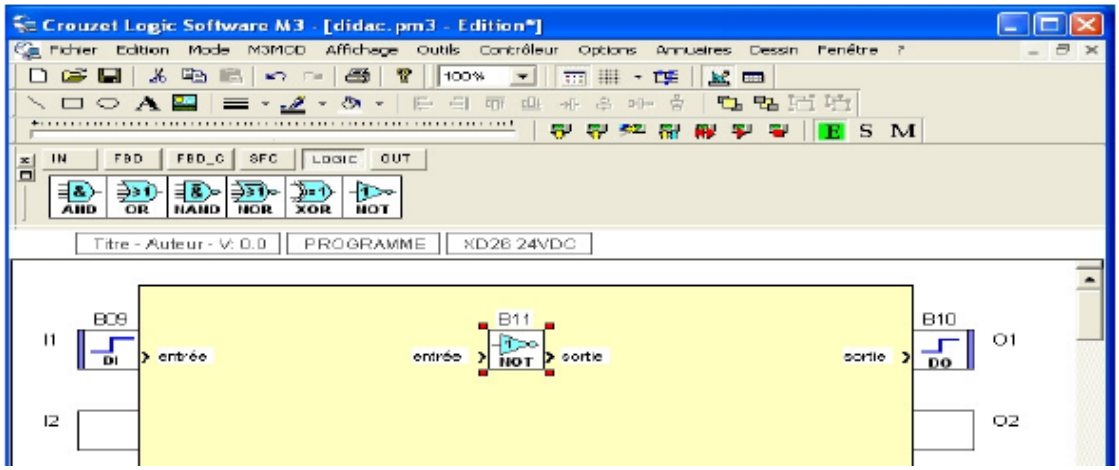
l. La fenêtre montre alors le circuit dessiné en fonctionnement : une liaison en bleu indique un état logique 0 ; une ligne en rouge indique un état logique 1.



m. Pour changer l'état d'une entrée, il suffit de cliquer dessus :



n. 14. On peut aussi utiliser du texte à la place des fils de liaison. Il suffit de cliquer avec le bouton droit sur une liaison et de choisir : "Type de câblage / Texte" et de remplacer dans un second temps le texte par défaut par celui de votre choix (entrée 1, etc..)



V.3 EXERCICES D'APPLICATION 1 : CIRCUIT LOGIQUE

On vous demande de réaliser le schéma ci-dessous et de remplir la table de vérité ci-après à partir des essais effectués en mode Simulation.

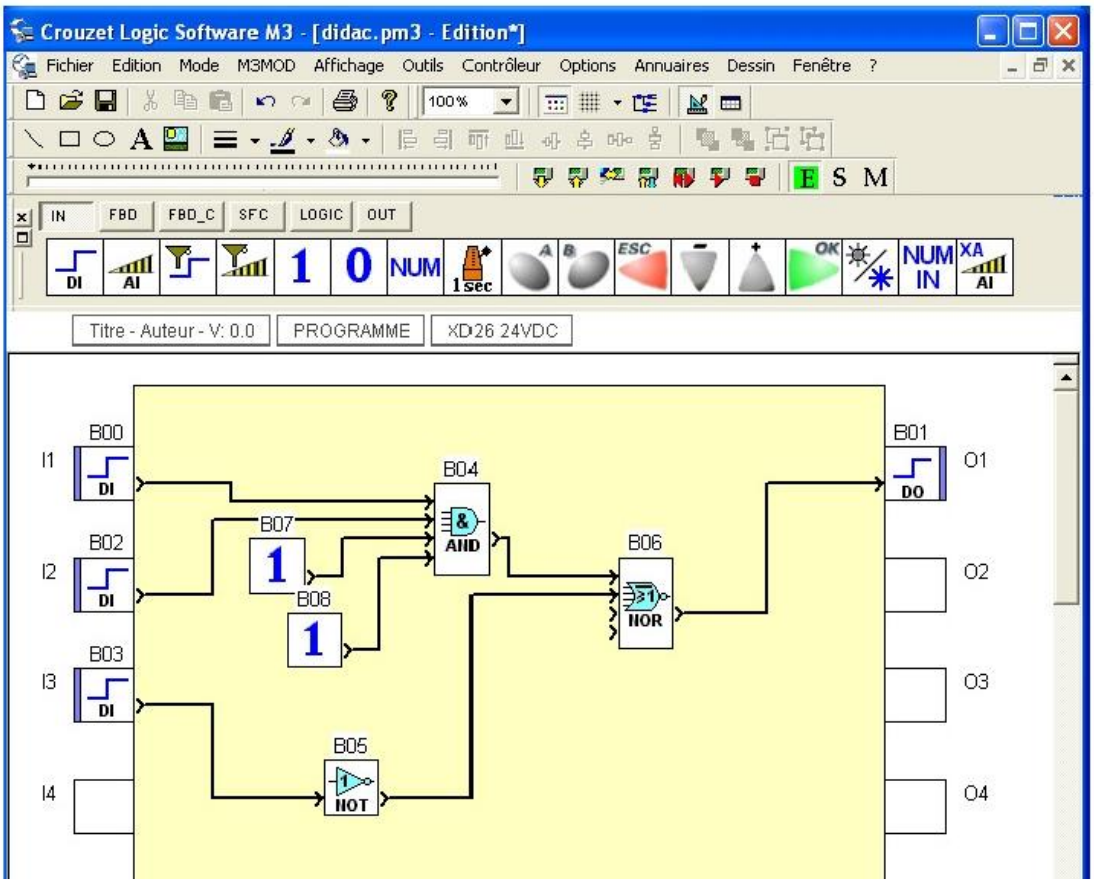
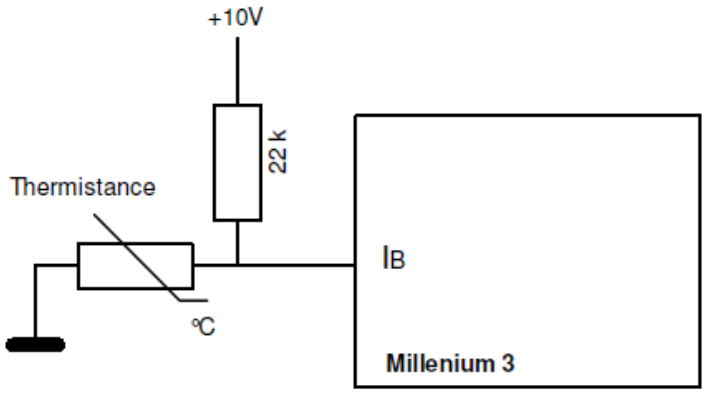


Table de vérité :

I1	I2	I3	Q1
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

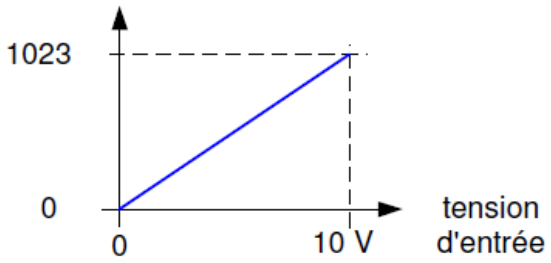
V.4 EXERCICES D'APPLICATION 2 : DETECTION DE SEUIL

Le modèle d'automate Millenium 3 étudié peut aussi gérer des entrées de type analogique (entrées IB à IG), variant entre 0 et 10 V, issue par exemple d'un capteur extérieur :



Exemple de capteur de température placé sur l'entrée analogique IA du Millenium

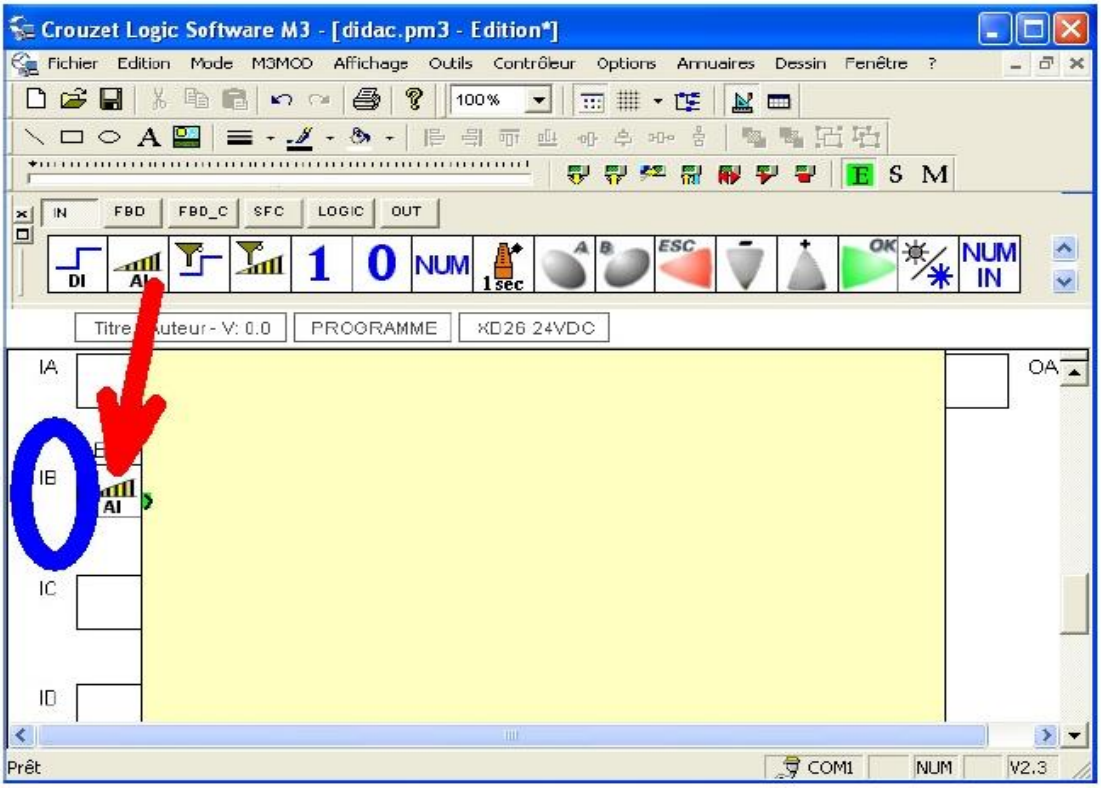
Dans cet exemple, la tension présente sur l'entrée IB du Millenium sera fonction de la valeur de la résistance du capteur, elle-même dépendante de la température. L'automate va effectuer une conversion de cette tension pour la transformer en une valeur numérique comprise entre 0 et 1023 (Conversion Analogique Numérique sur 10 bits) :



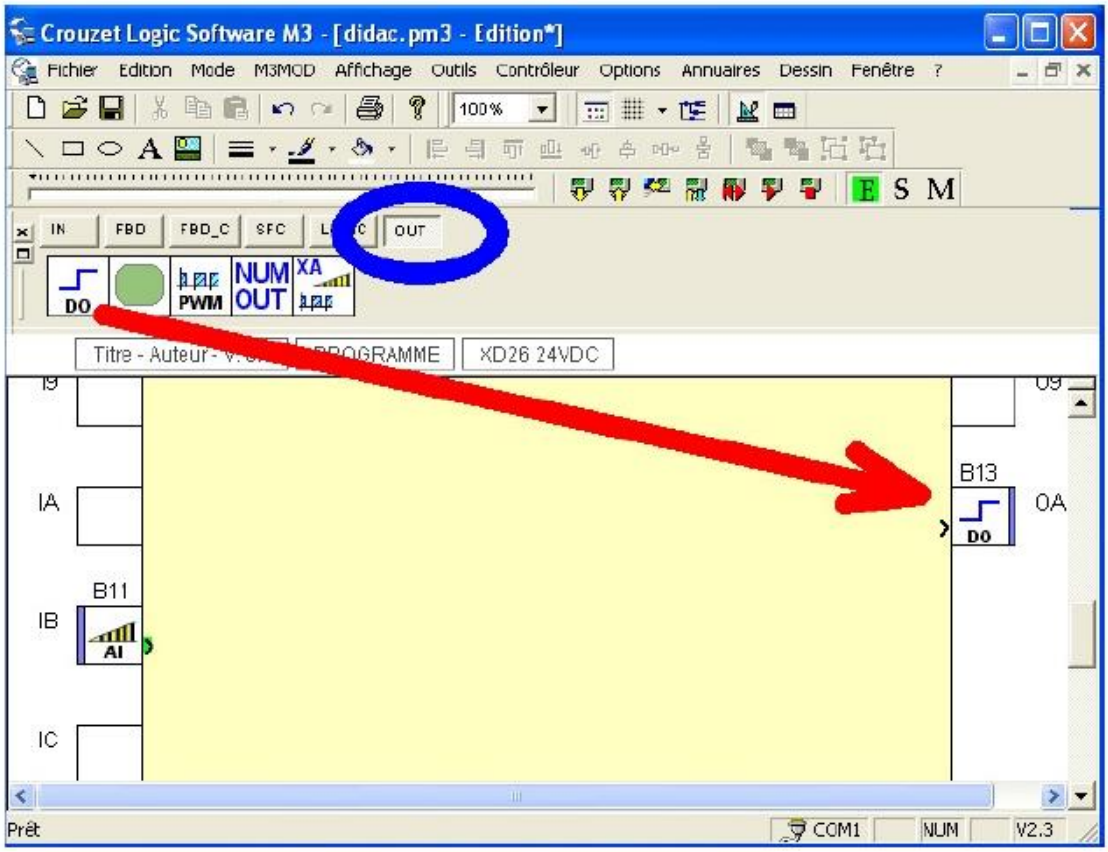
Il est alors possible d'exploiter cette information pour détecter un seuil de tension correspondant dans cet exemple à un seuil de température.

On choisira dans cet exercice de commander une sortie lorsque le nombre issu de la conversion de tension sera supérieur à 314.

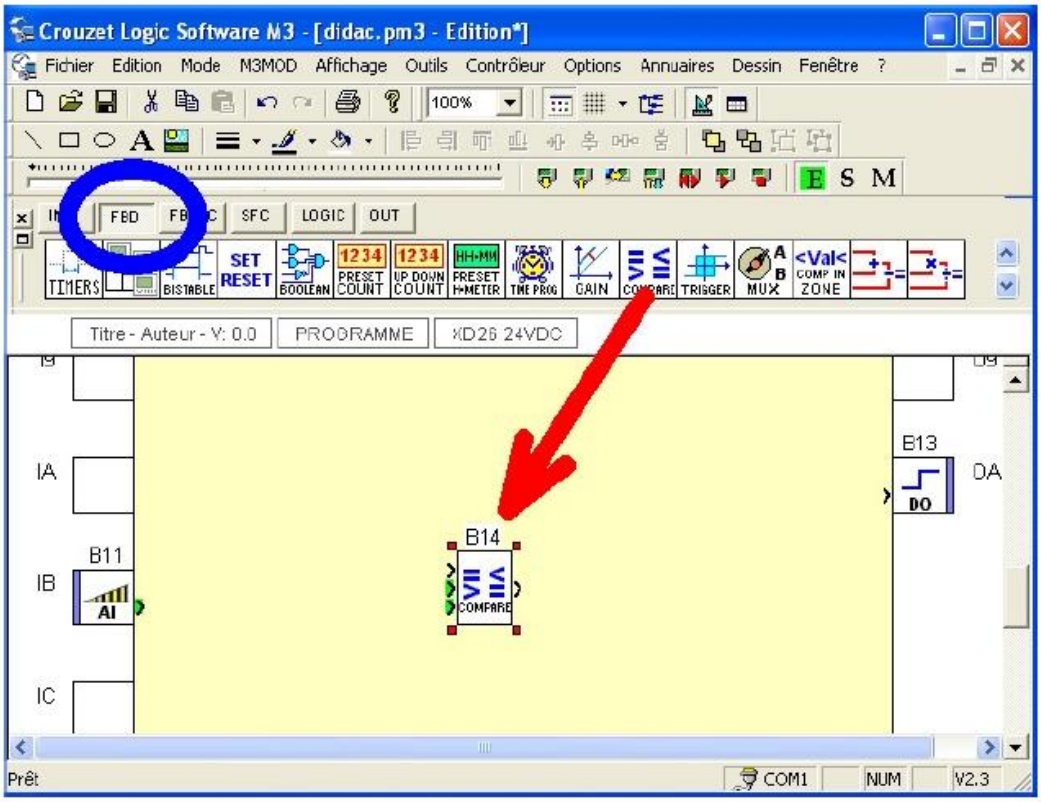
a. Dans le menu de fonction "IN", déplacez l'entrée "AI" (Analog Input) sur l'entrée IB.



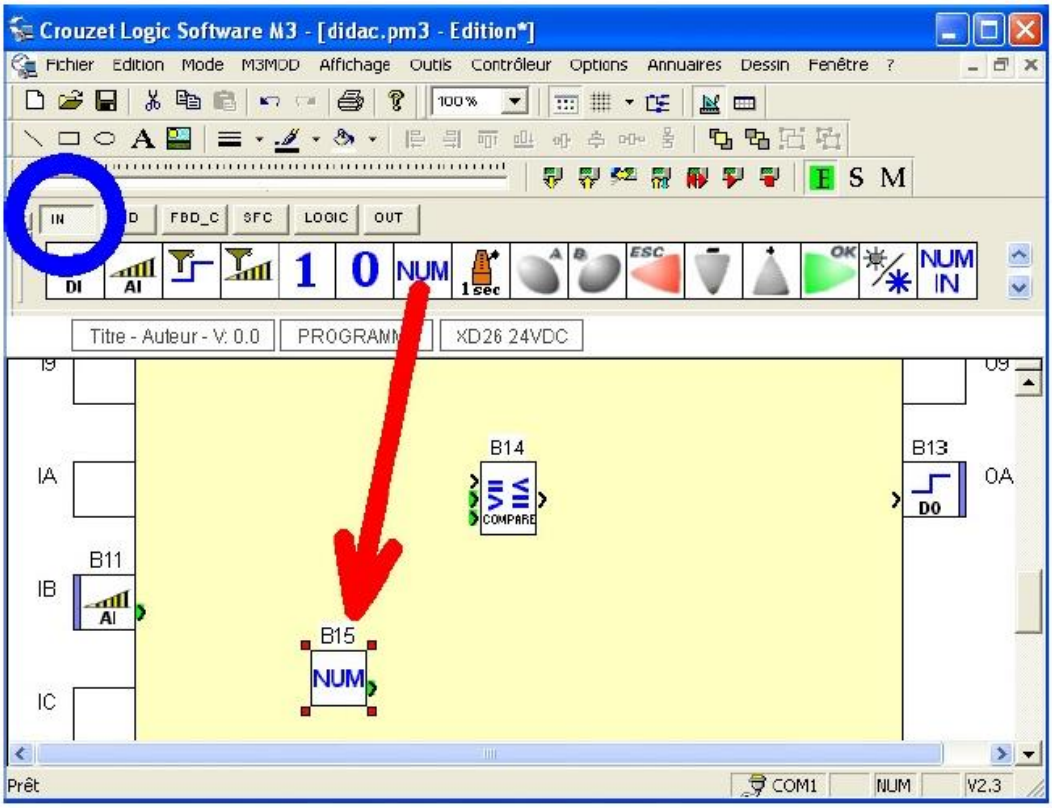
b. Dans le menu de fonction "OUT", déplacez la sortie "DO" (Digital Ouput) vers la sortie OA.



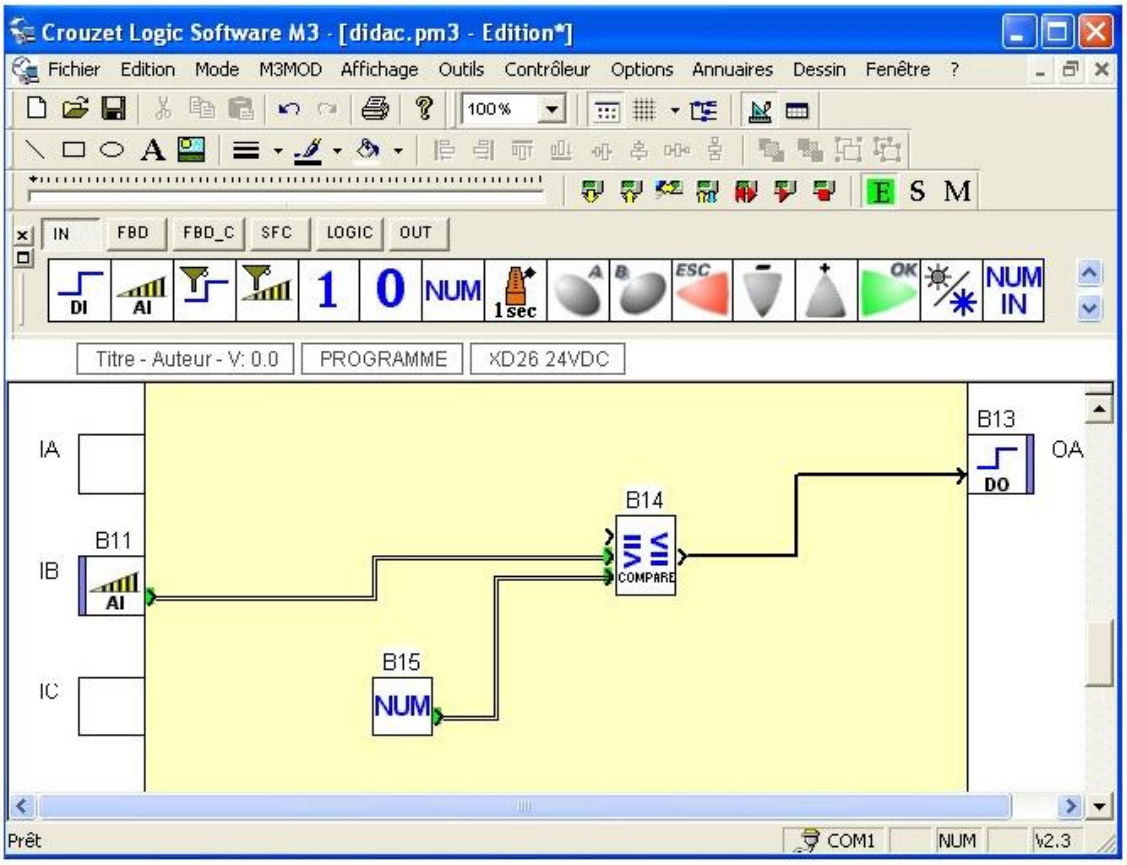
c. Dans le menu de fonction "FBD", déplacez l'élément "Compare" sur la feuille de travail (cet opérateur compare 2 valeurs numériques).



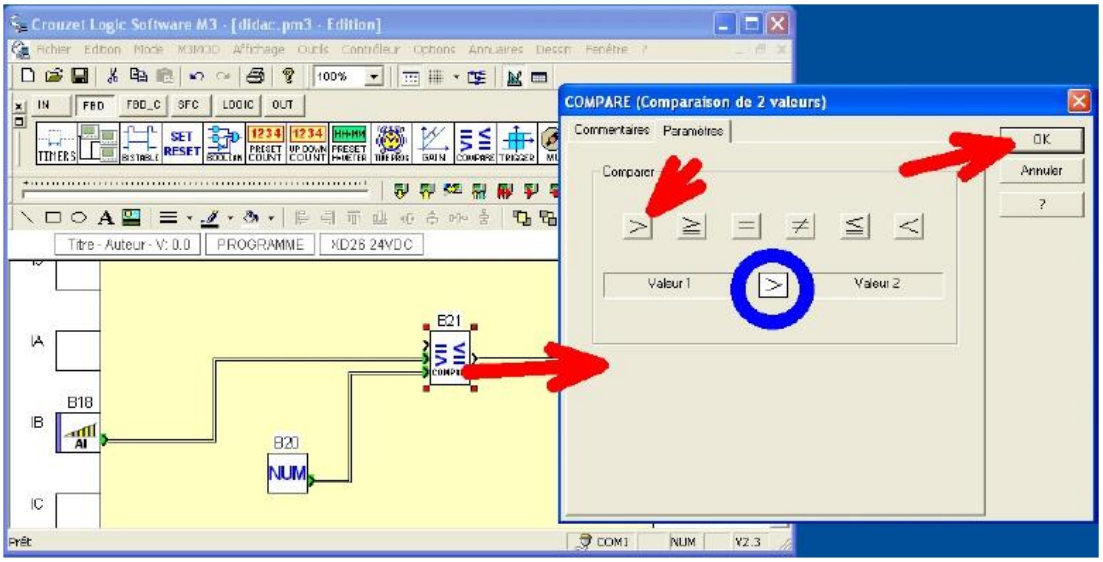
d. Dans le menu de fonction "IN", déplacez l'élément "NUM" sur la feuille de travail. Cette entrée permet de délivrer un nombre compris entre -32768 et +32767.



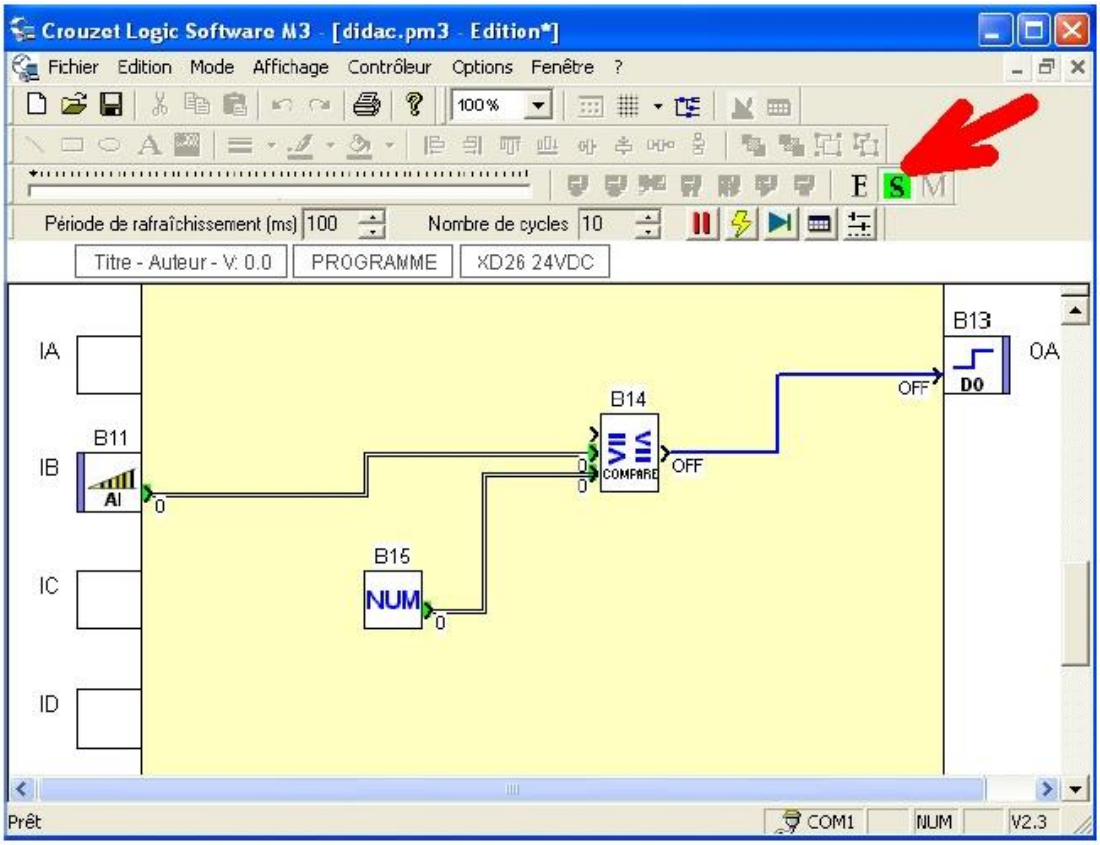
e. Effectuez les liaisons entre les entrées-sortie et les divers éléments de la feuille. Vous remarquerez que les liaisons "numériques" (entre IB et "Compare" ainsi qu'entre "NUM" et "Compare") n'ont pas la même allure que les liaisons "binaires" (entre "Compare" et "OA").



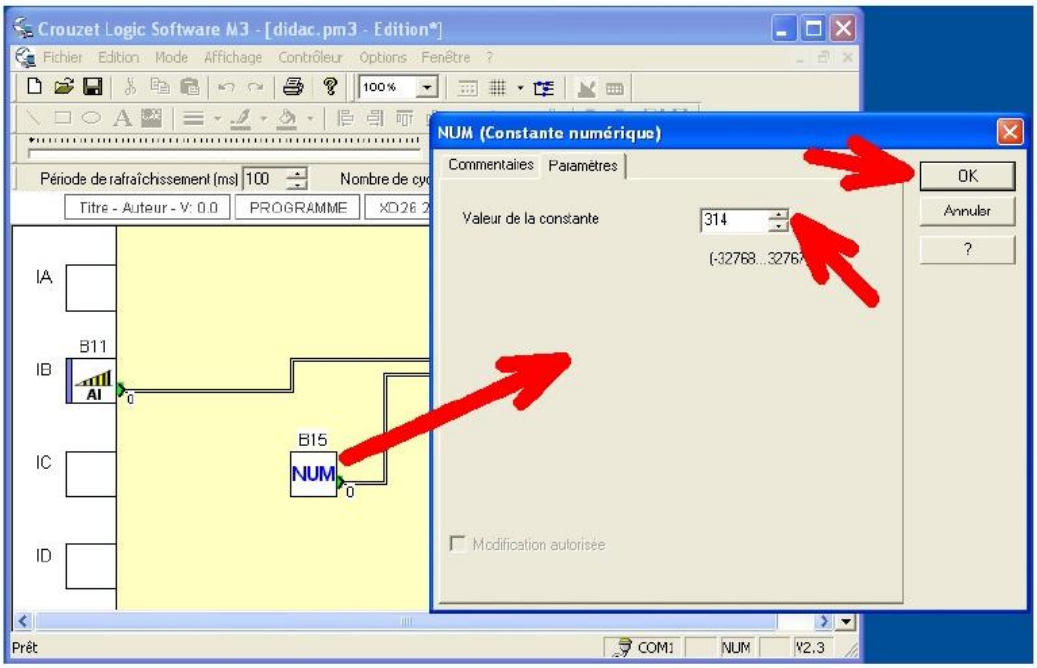
f. En double-cliquant sur l'élément "Compare", vous ouvrez une fenêtre qui vous permet de choisir le type de comparaison recherché. Optez pour que la sortie soit validée quand "Valeur 1" (l'entrée IB) est supérieure à "Valeur 2" (entrée "NUM").



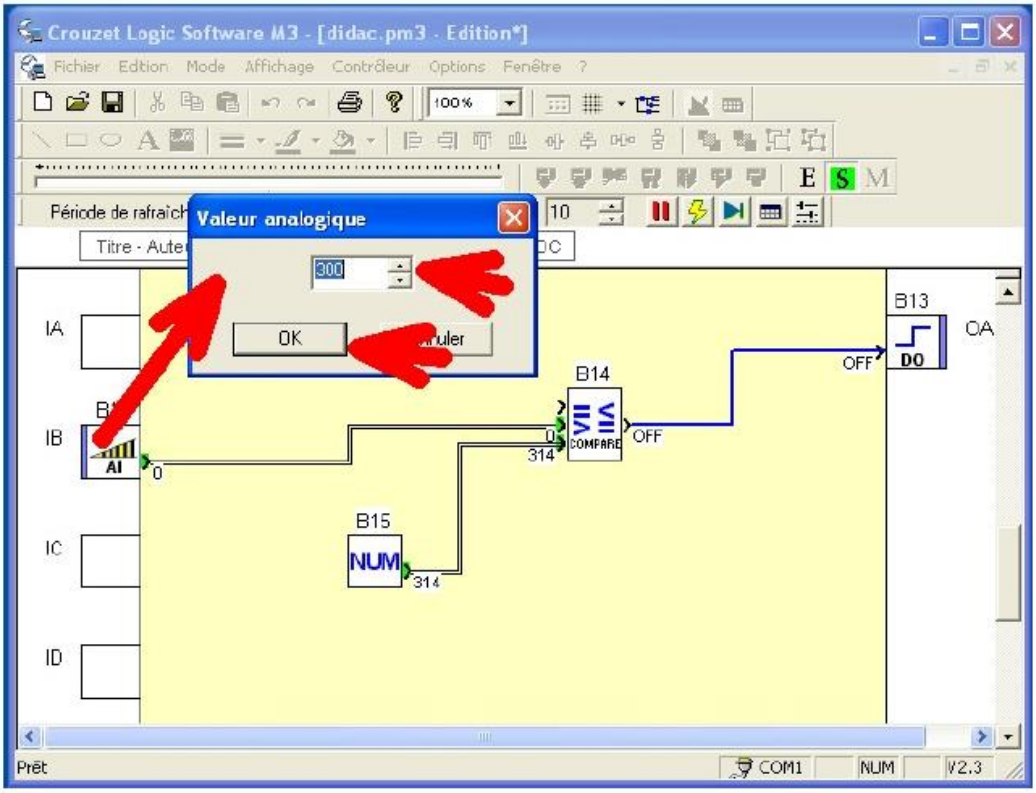
g. Lancez la simulation (bouton "S"), les valeurs numériques des entrées sont indiquées à côté des liaisons (valeurs par défaut = 0). Les 2 valeurs étant égales, la sortie est sur "OFF".



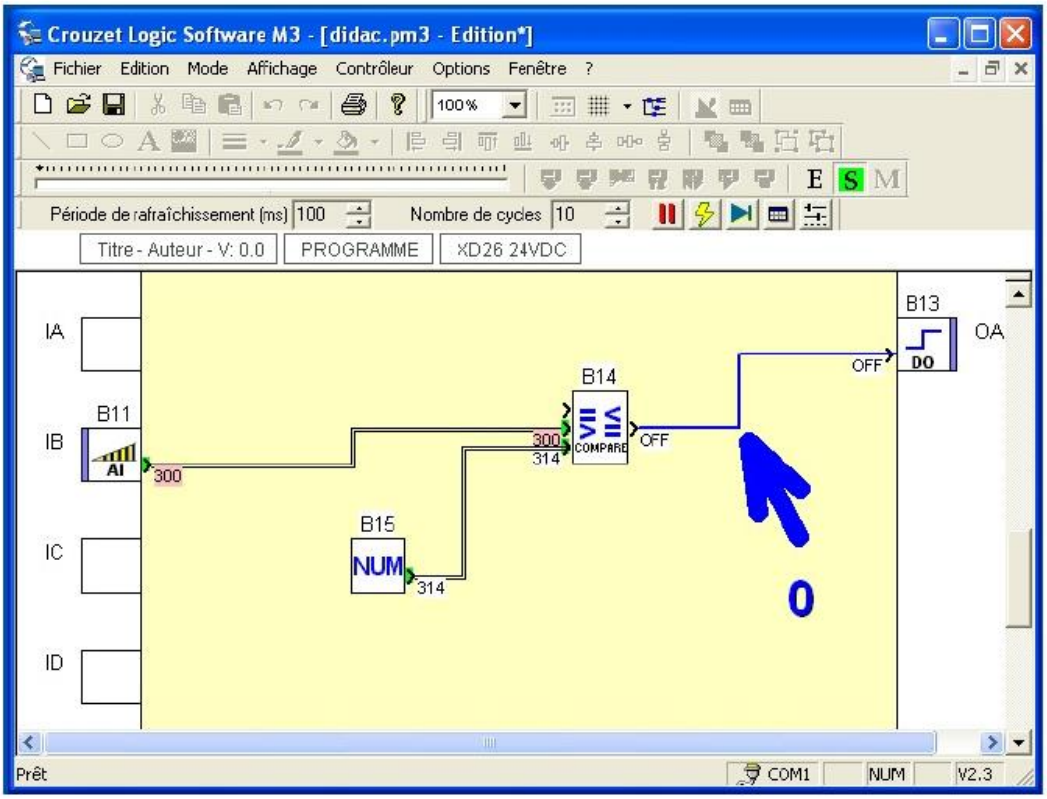
h. En double-cliquant sur l'élément "NUM", vous ouvrez une fenêtre qui vous permet de choisir la valeur "numérique" délivrée par cet élément. Saisissez "314" puis validez.



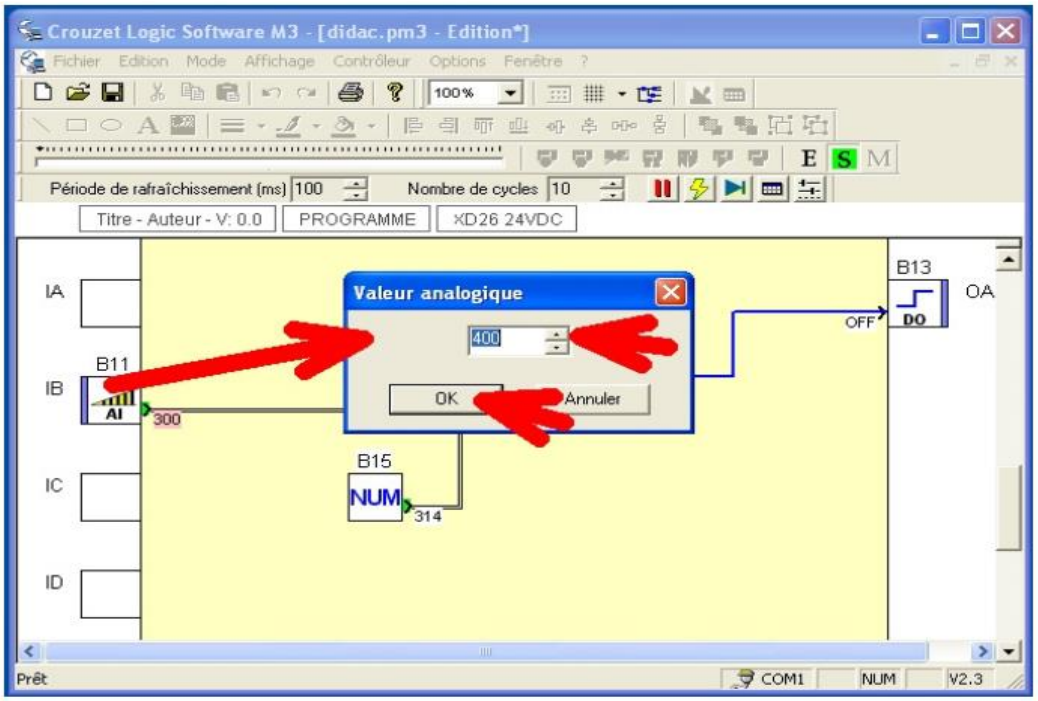
i. Cliquez sur l'entrée analogique "IB" et imposez une valeur de 300 à cet élément puis validez.

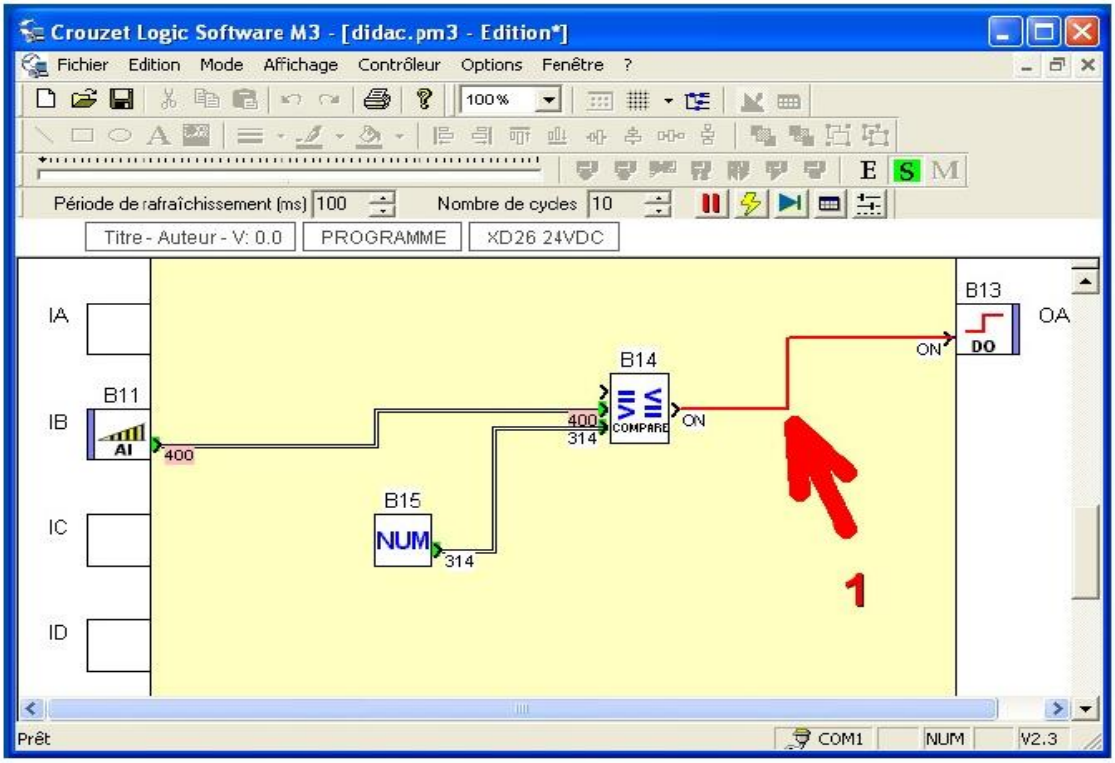


j. On constate que la sortie reste à 0 car 300 < 314.



k. Choisissez maintenant une valeur supérieure à 314 (400 pour notre exemple) :





l. On constate que la sortie est maintenant validée.

V.5 EXERCICES D'APPLICATION 3 : CIRCUIT LOGIQUE

On vous demande de réaliser le schéma de commande, en langage FBD de trois actionneurs (moteur **M**, vérin **V** et électrovannes **Ev**) :

a. commande séparée (bouton poussoir " m" pour commander le moteur **M**, le fin de course " **c1**" pour commander la sortie de la tige du vérin **V** et " **c2**" pour commander la rentrée de la tige du vérin **V**

b. commande alternée après de trois 3 secondes (moteur seul puis après 3 seconde lampe sera allumée et 3 seconde après le vérin se déplace)

c. commande de deux actionneurs (moteur et électrovanne) avec action conditionnée par un compteur.

Partie VI *Initiation au logiciel PL7-PRO*

VI.1 Introduction

L'objectif de cette partie est de connaître le minimum requis pour l'utilisation du logiciel PL7 vis à vis d'un TSX 37/57. Les langages PL7 sont des langages graphiques (Langage à Relais, Grafcet) et textuels (Structured Text, Instruction List) destinés à programmer les automates TSX. Ils offrent un environnement de programmation complet permettant à l'utilisateur de développer son application dans un langage graphique simple (éditeur), de la traduire automatiquement en langage automate (compilateur) et enfin de la charger dans la mémoire centrale de 1' A.P.I. (téléchargement).

Suivant le TSX dont vous disposez (Micro ou Premium), une certaine quantité de mots et de bits sont disponibles (pour le TSX3722 Micro, on a 128 bits d'étapes %Xi, 256 bits internes %Mi, et un nombre de mots limité par la taille mémoire de l'automate.).

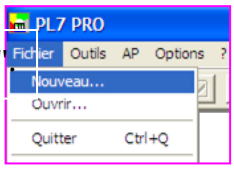
VI.2 Ouverture d'une session

Cliquer sur l'icône dans le bureau 

ou utiliser le menu Démarrez-->Programmes-->Modicon Télémécanique-->PL7 PRO V4.4.

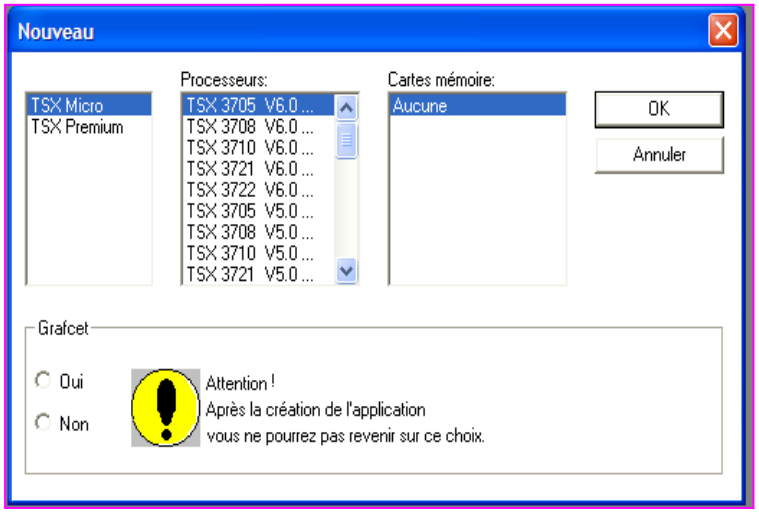
1)- Créer un nouveau projet

Créer une nouvelle application dans le menu "Fichier", sous menu " Nouveau..."



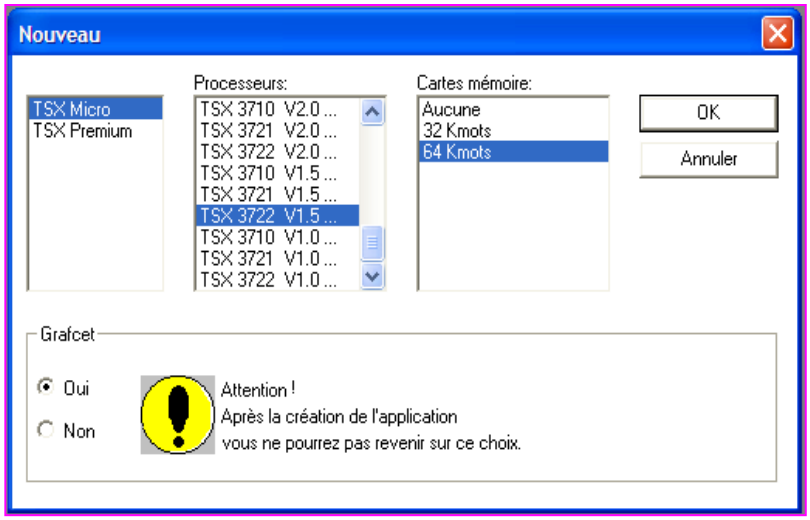
2)- Sélectionner la version d'automate que vous utilisé

Apres creation du nouveau projet , on obtient cette fenêtre.



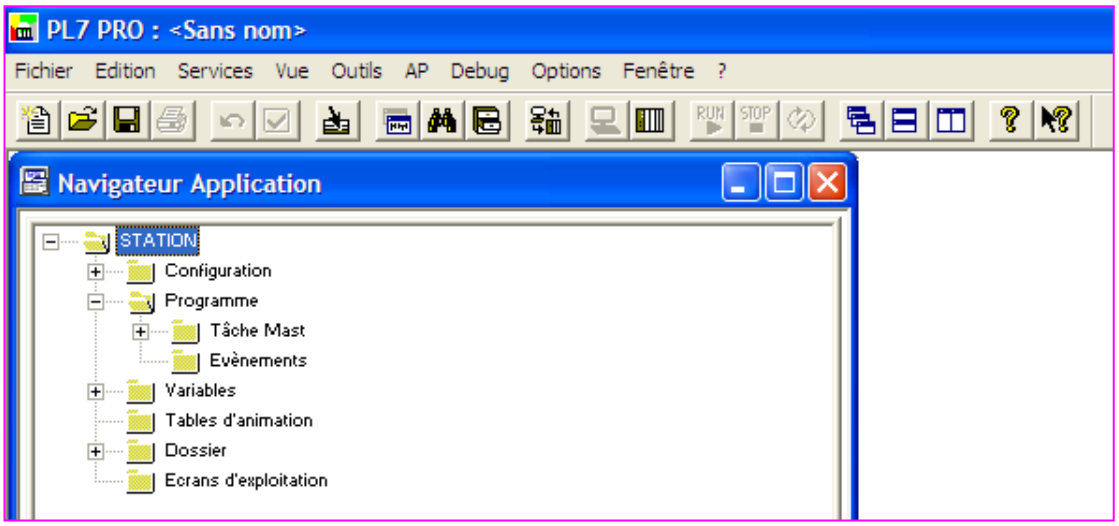
Où on doit choisir en premier le nom de l'automate utilisé soit TSX Pemium (pas Langage Grafcet) ou TSX Micro (avec Langage Grafcet) , puis en second on choisit le type de processeur et enfin la carte mémoire utilisée (si elle existe) choisir la programmation Grafcet oui ou non selon la solution à proposée

Choisir l'automate utilisé par exemple TSX3722 V3.0 , cartes mémores « 64 Kmots » et « oui » pour la programmation Grafcet.
Valider votre choix par l'appui sur le bouton "OK".



3)- Ouverture « Navigateur Application »

L'application est crée et la fenêtre "Navigateur Application" s'ouvre alors.



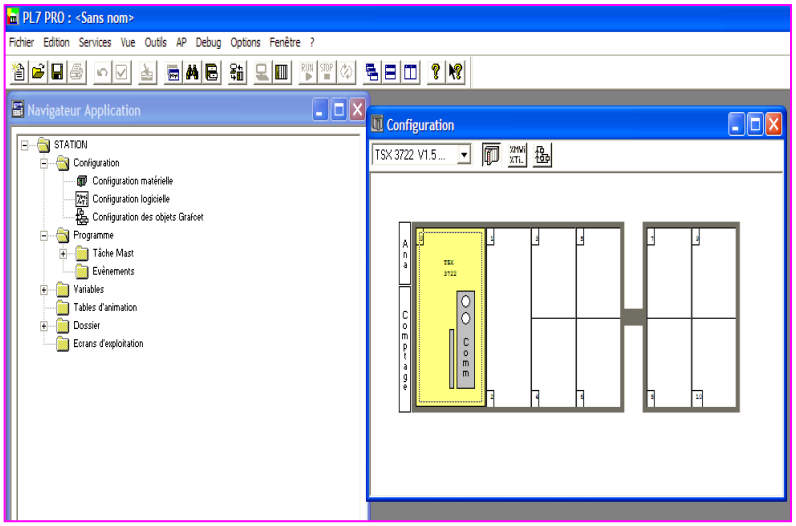
Le Navigateur "Application" présent sur la gauche de l'écran permet d'accéder aux différentes fonctionnalités de l'outil de développement : configuration, programmation, accès aux données, transfert et mise en application du programme, dossier de l'application en cours.

3.1) *Configuration matérielle de l'Automate*

a) Configuration (automatique) du processeur

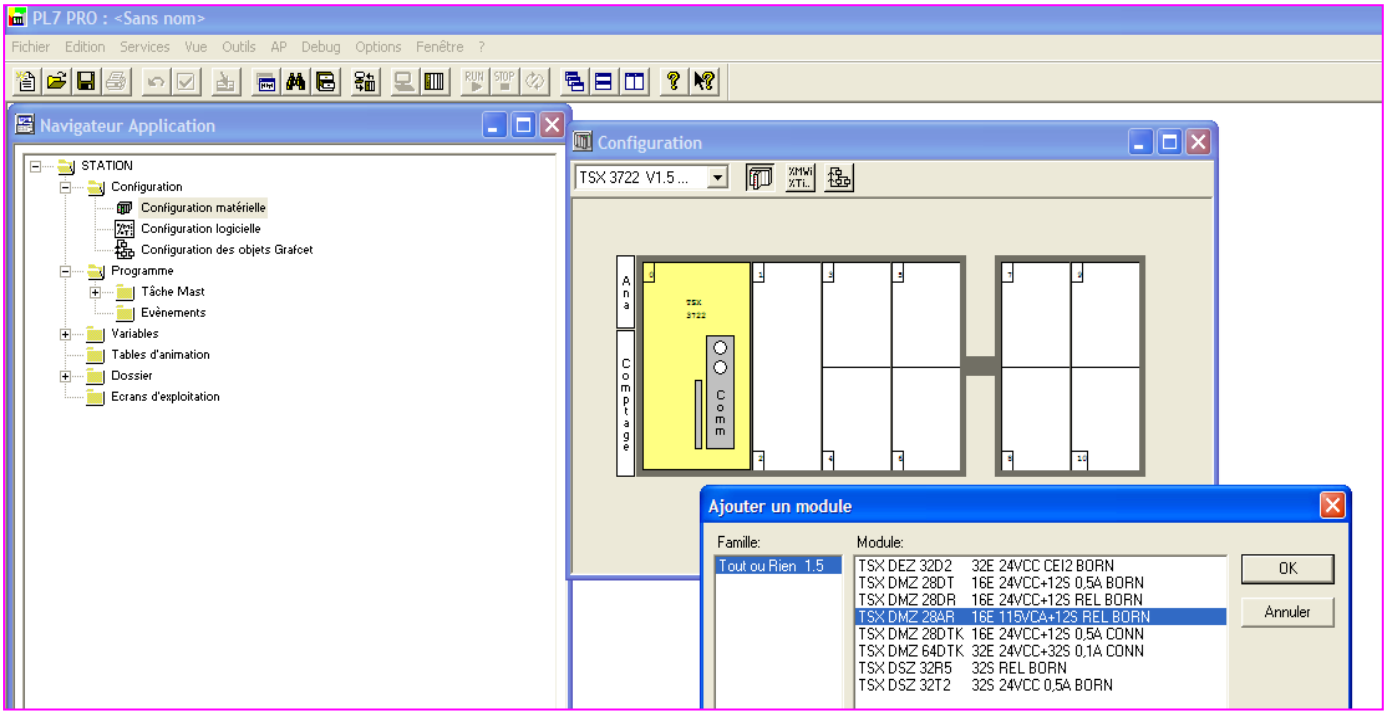
La première étape est de définir sur l'outil de développement la configuration matérielle de l'automate sur lequel le programme sera transféré puis exécuté. Sélectionner dans le dossier "Configuration"--> "Configuration matérielle".

On obtient une image du TSX choisi avec par défaut l'ensemble de ses emplacements vides. Le module de communication "Com" ou module « 0 » est par défaut correctement configuré il correspond au processeur choisit.

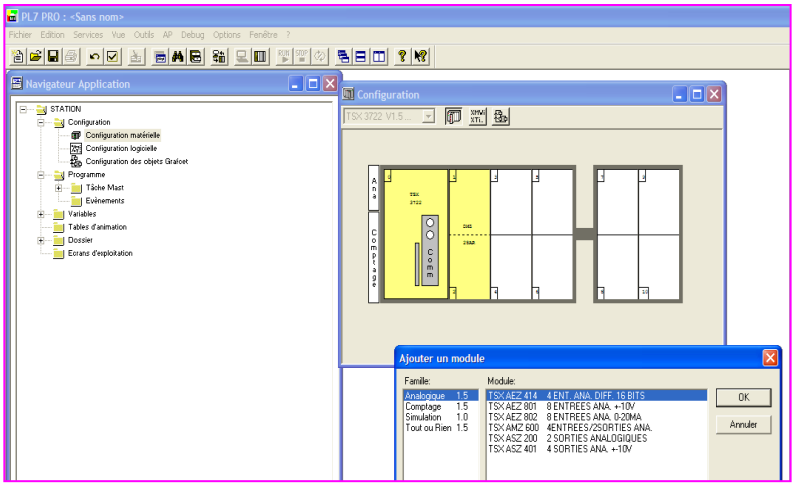


b) *Configuration module entrée / sortie*

Pour déclarer les entrées sorties du TSX, cliquer sur l'emplacement 2 et ajouter le module TSXDMZ28DR . Ce module comporte 16 entrées TOR sur la position 1 et 12 sorties sur la position 2 du rack. On valide par OK.



On obtient cette fenêtre, si le nombre variable d'entrée (ou le type d'entrée) ou sortie , on peut ajouter d'autre module on double cliquant sur le module 3 et procédé de la même manière précédente

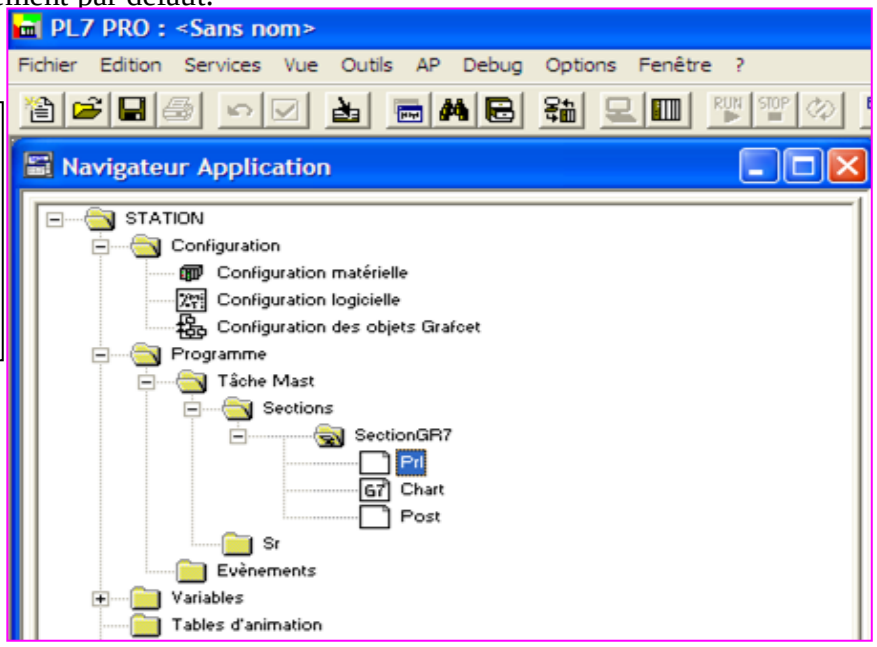


En quittant, valider la "reconfiguration matérielle"

4) Programmation d'une application

Par défaut, la programmation se fait en mode "déconnecté" ou "local". Vous ne communiquez pas avec le TSX. Le navigateur d'application permet d'accéder à tous les modules de programme qui constituent l'application complète. Si vous avez choisi le mode "Grafcet" à la création de l'application, le navigateur montre les éléments de programme prédéfinis : le Grafcet en lui-même (Chart) , la partie de programme exécutée avant l'exécution du Grafcet (Pré) et celle exécutée après (Post) . Le langage des sections Pre et Post doit être défini par l'utilisateur lors du premier appel de la section. Vous pouvez rajouter une ou plusieurs sections de programme dans le langage de votre choix à votre application. Toutes les sections s'exécutent séquentiellement par défaut.

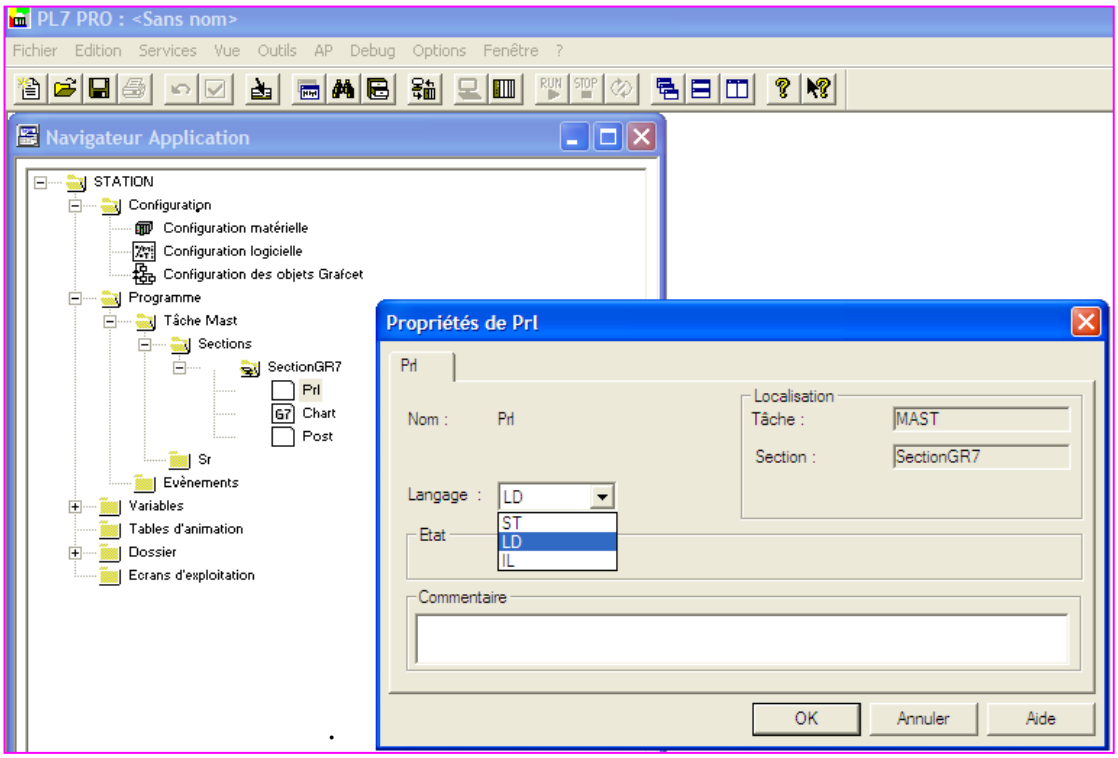
Pour commencer la programmation, on ouvre le dossier Programme, Tâche Maste, Sections, SectionGR7



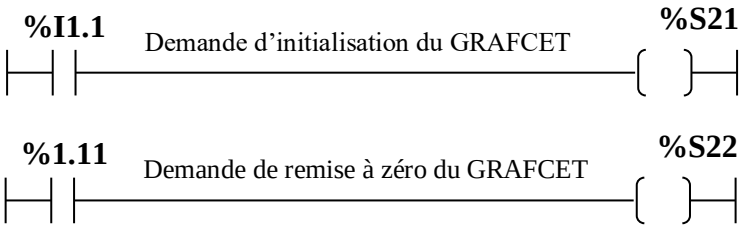
4-1) Traitement préliminaire

Pour une programmation d'un Grafcet, le traitement préliminaire dépend du cahier de charge.

Ouvrir le traitement préliminaire en double-cliquant sur « PRL ». Il peut être programmé en Ladder (LD), structuré (ST), ou instruction list (IL), choisir « LD »



%S21 : initialisation du GRAFCET %S22 : remise à zéro du GRAFCET



Une fois le programme terminé, il faut valider, la couleur des traits et des adresses change et devient bleue.

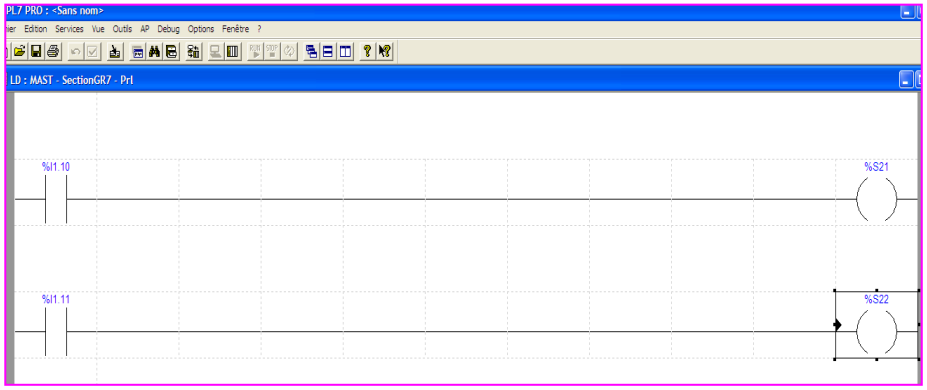


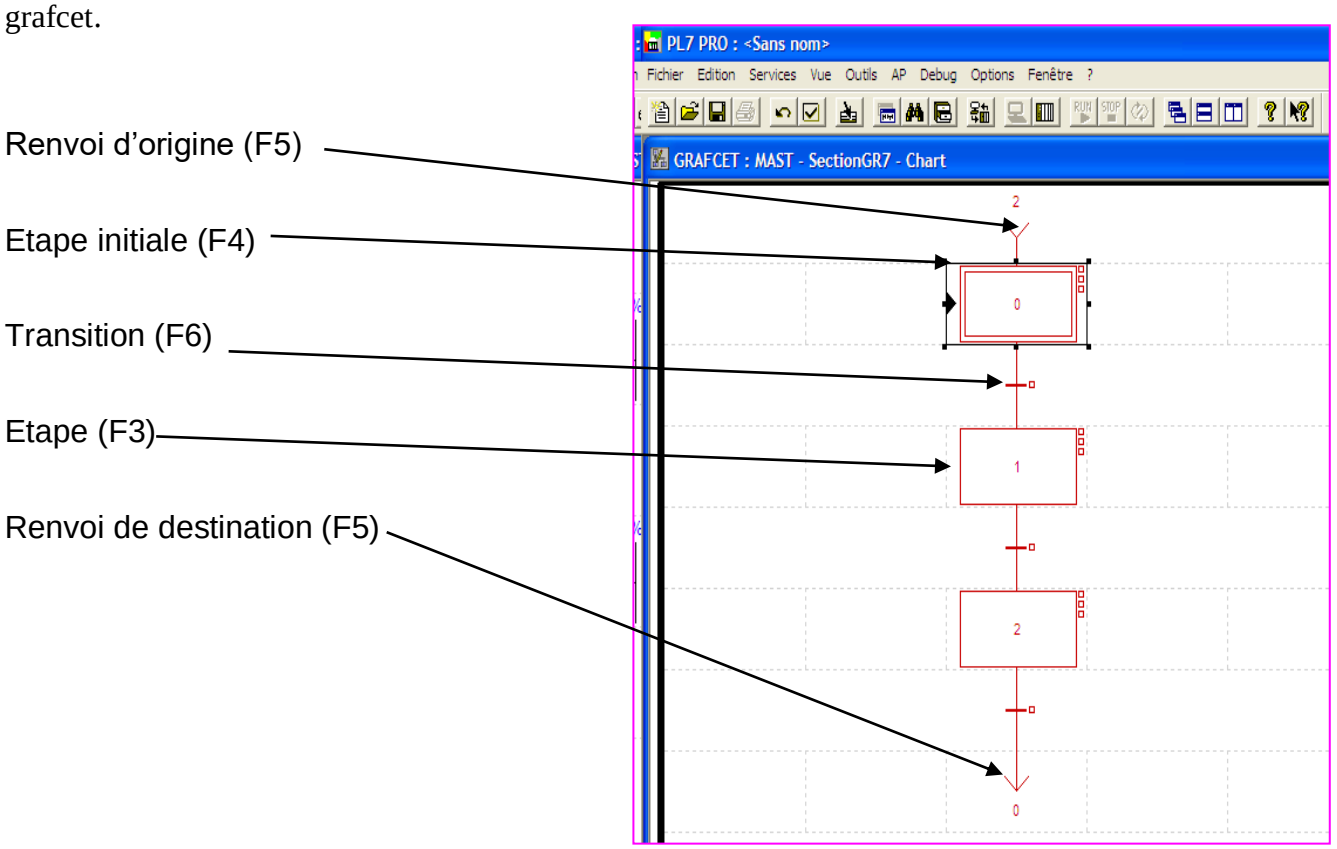
Fig A1-12 Création d'un module de programmation en langage Ladder.

4-2) Traitement séquentiel

La programmation le traitement séquentiel se fait en deux étapes, la construction l'architecture du Grafcet et la programmation des transitions.

a) Construction de l'architecture du graphe.

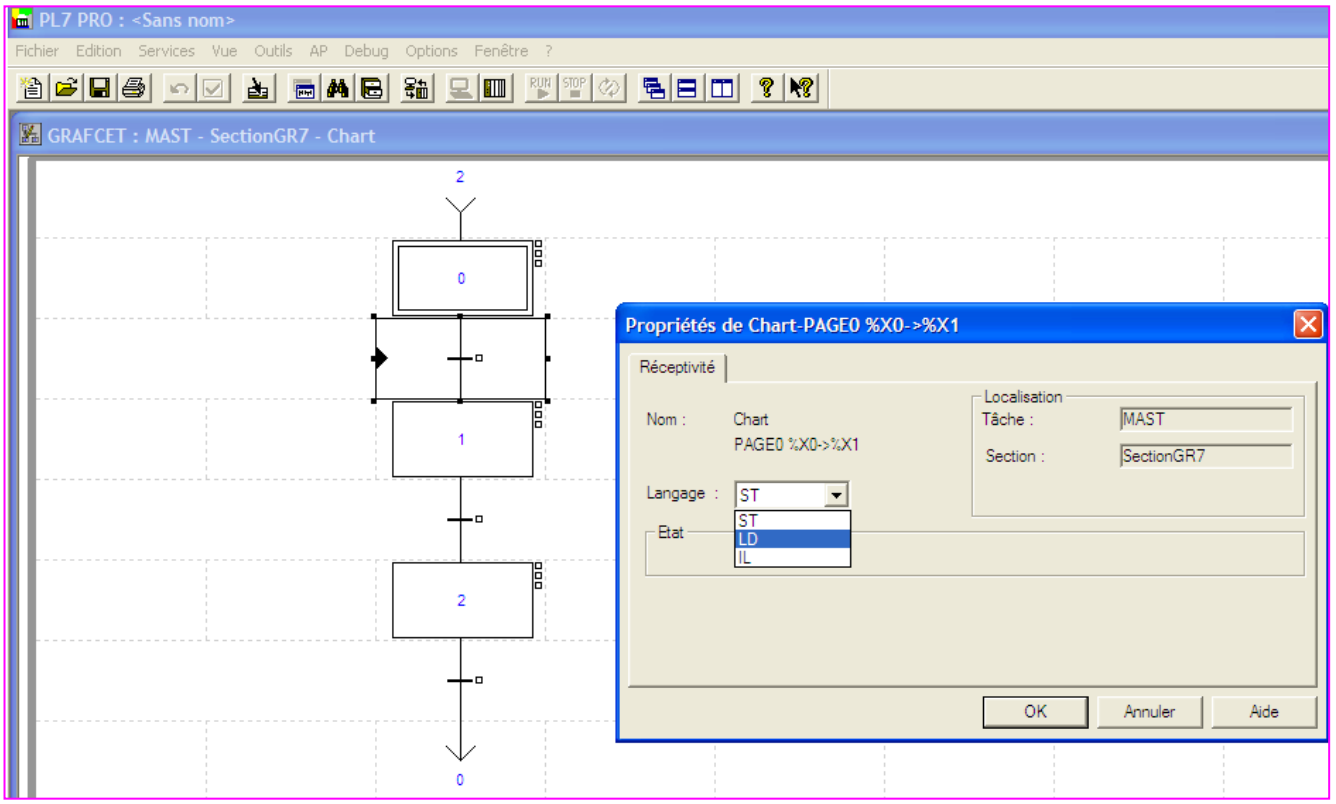
Il faut Cliquer sur **chart** (Le navigateur d'application, Programme, Tâche Maste, Sections, SectionGR7, **chart**), la page suivante s'ouvre. Compléter cette dernière en fonction du grafcet.

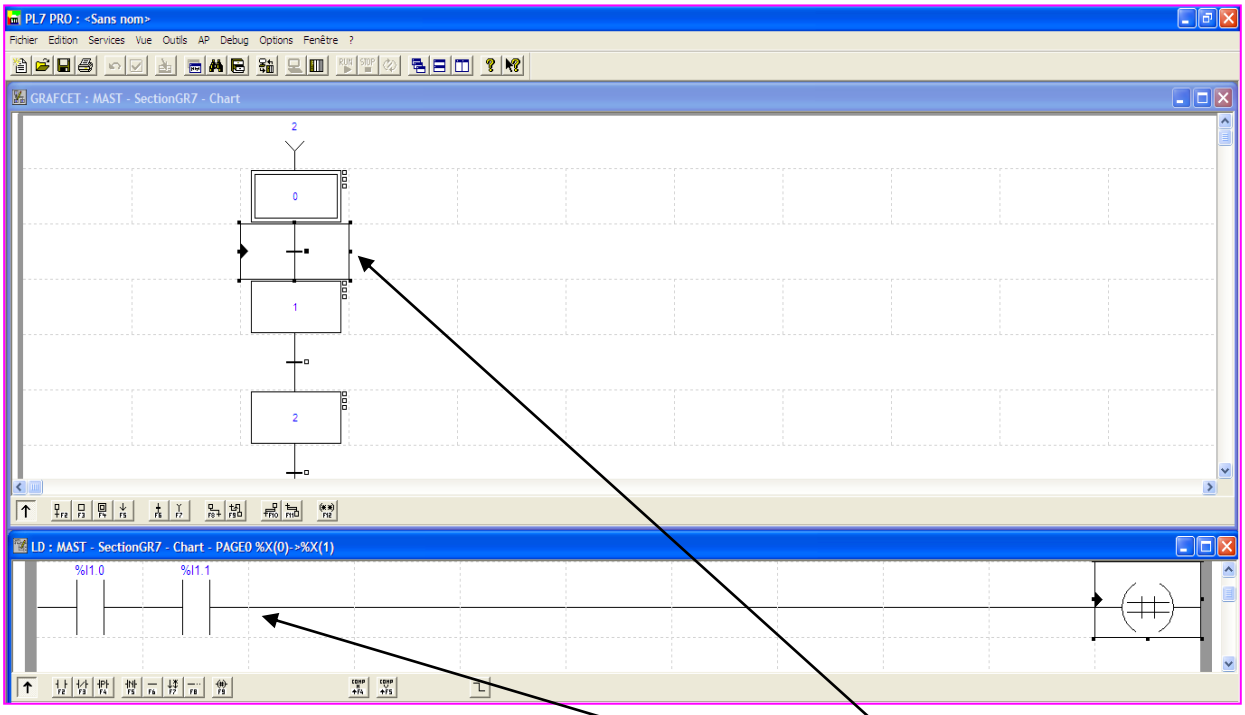


Une fois terminé, il faut valider, la couleur du Grafcet devient noire.

b) Programmation des transitions

Pour la programmation des transitions, double-cliquez sur celle qui vous intéresse et sélectionnez le langage de spécification formelle de la réceptivité associée (LD pour LADDER) puis éditer la fonction combinatoire de transition (la transition agit sur la sortie implicite "#" qui est la réceptivité de l'étape suivante) .Vous validez et fermez la fenêtre de la réceptivité que vous venez de saisir afin d'en saisir une autre de la même manière.

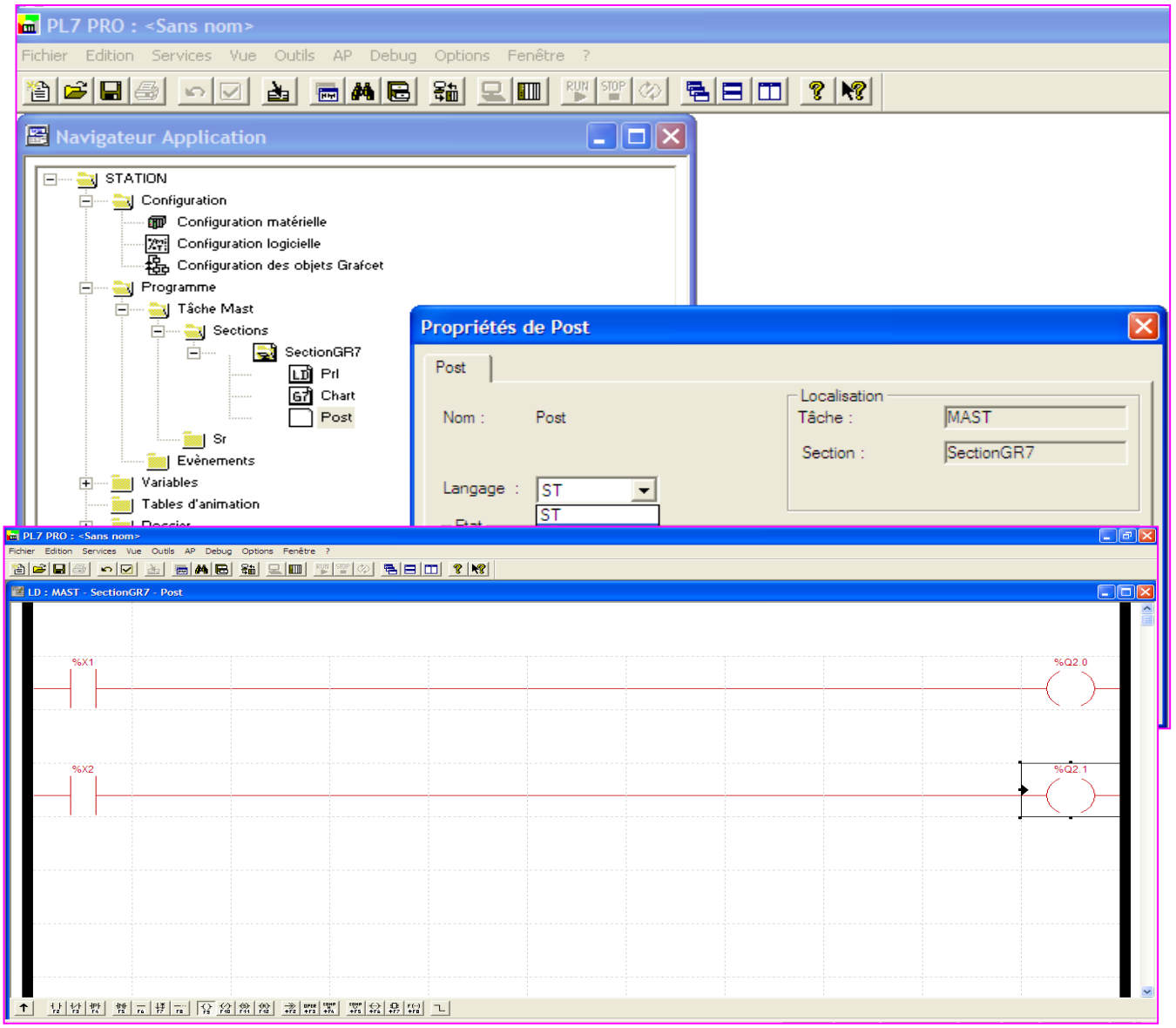




Apres validation l'écriture de l'adresse devient bleue et le carré de la transition devient noir complètement.

4-3) Traitement postérieur

C'est la programmation **actions associées aux étapes .** Pour traiter les actions Grafcet, ouvrir le traitement postérieur Post (Le navigateur d'application, Programme, Tâche Maste, Sections, SectionGR7, **Post**), . Sélectionner le langage de programmation (LD pour LADDER) et écrire les fonctions de sortie en utilisant les activités d'étapes %Xi .



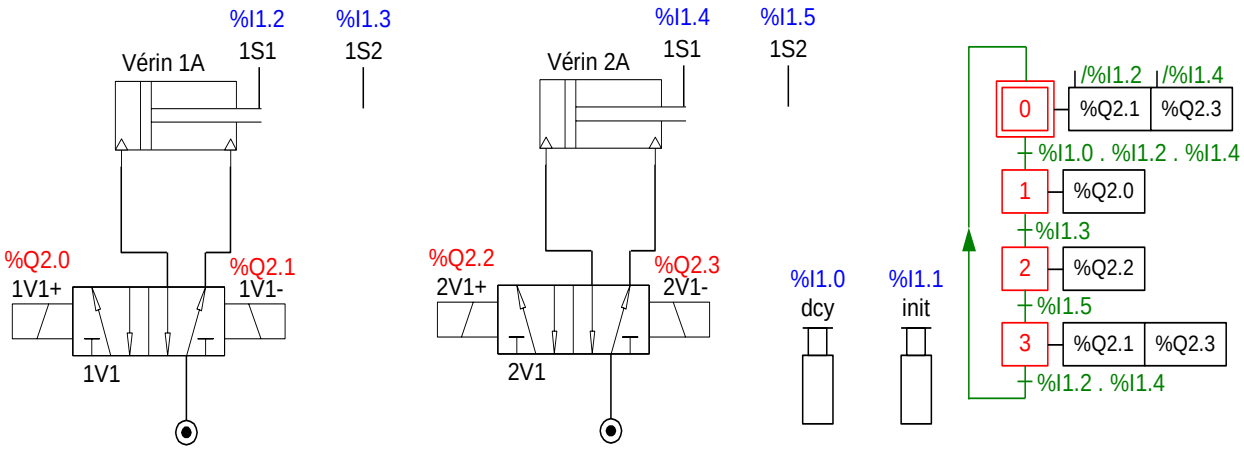
Apres la programmation postérieur, il faut valider , le réseau devient noir , l'écriture des adresses bleue.

Pour terminer il faut enregistrer le programme.

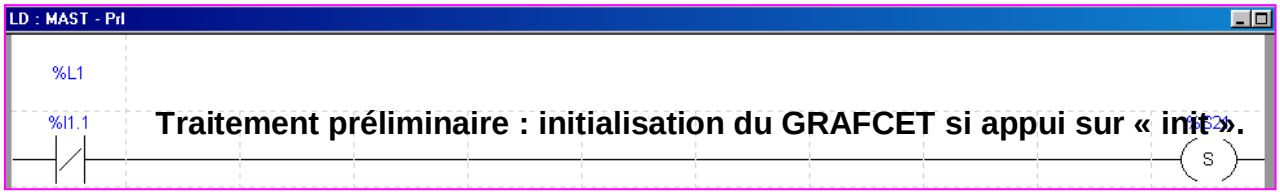
Exemple 1 : programmation de grafcet

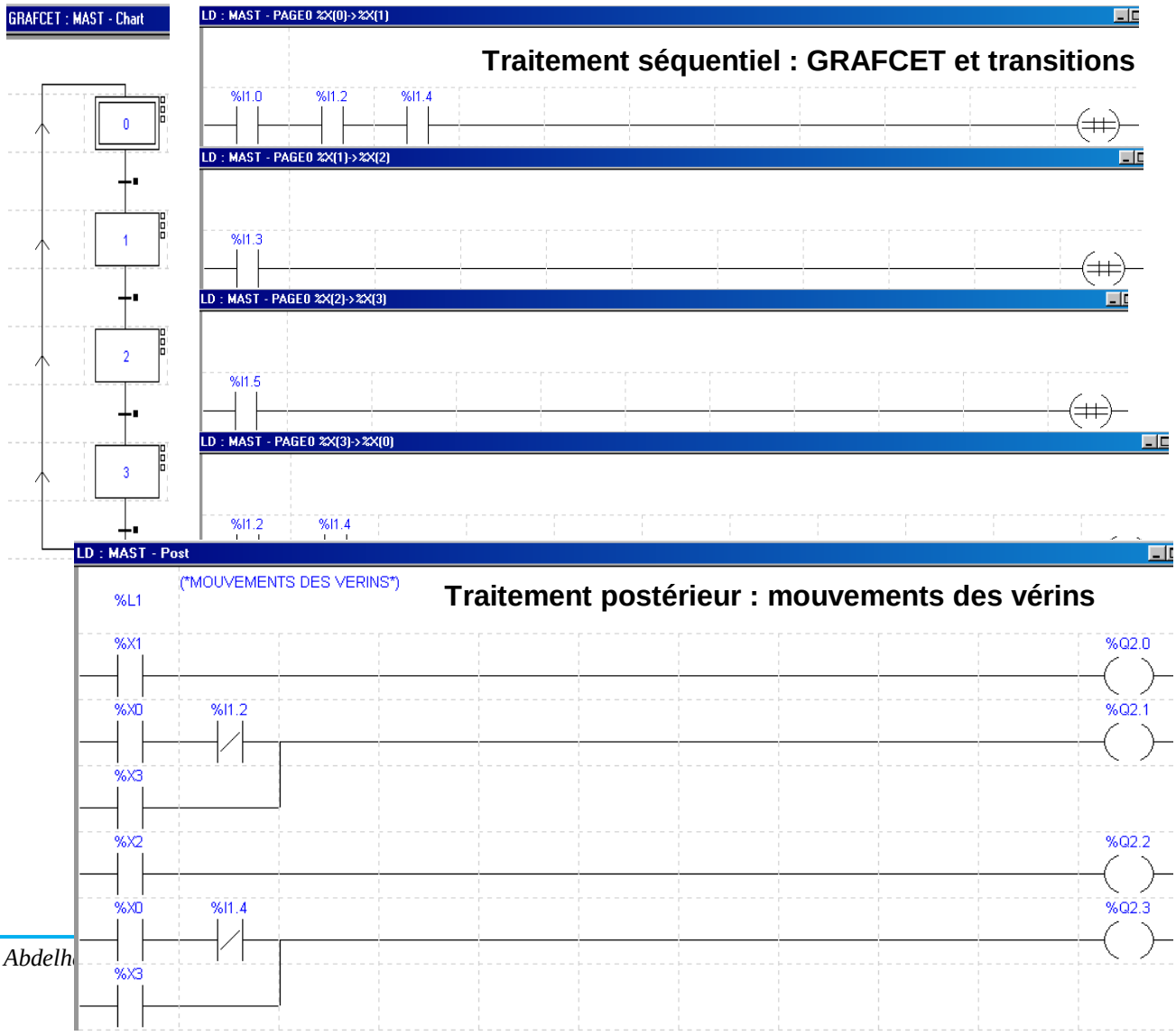
SCHEMA PNEUMATIQUE

GRAFCET AUTOMATE



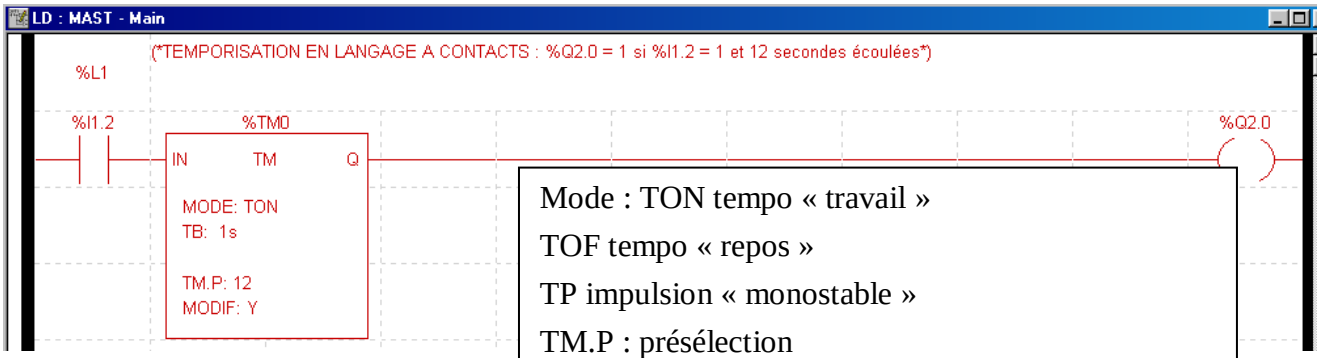
PROGRAMME CORRESPONDANT





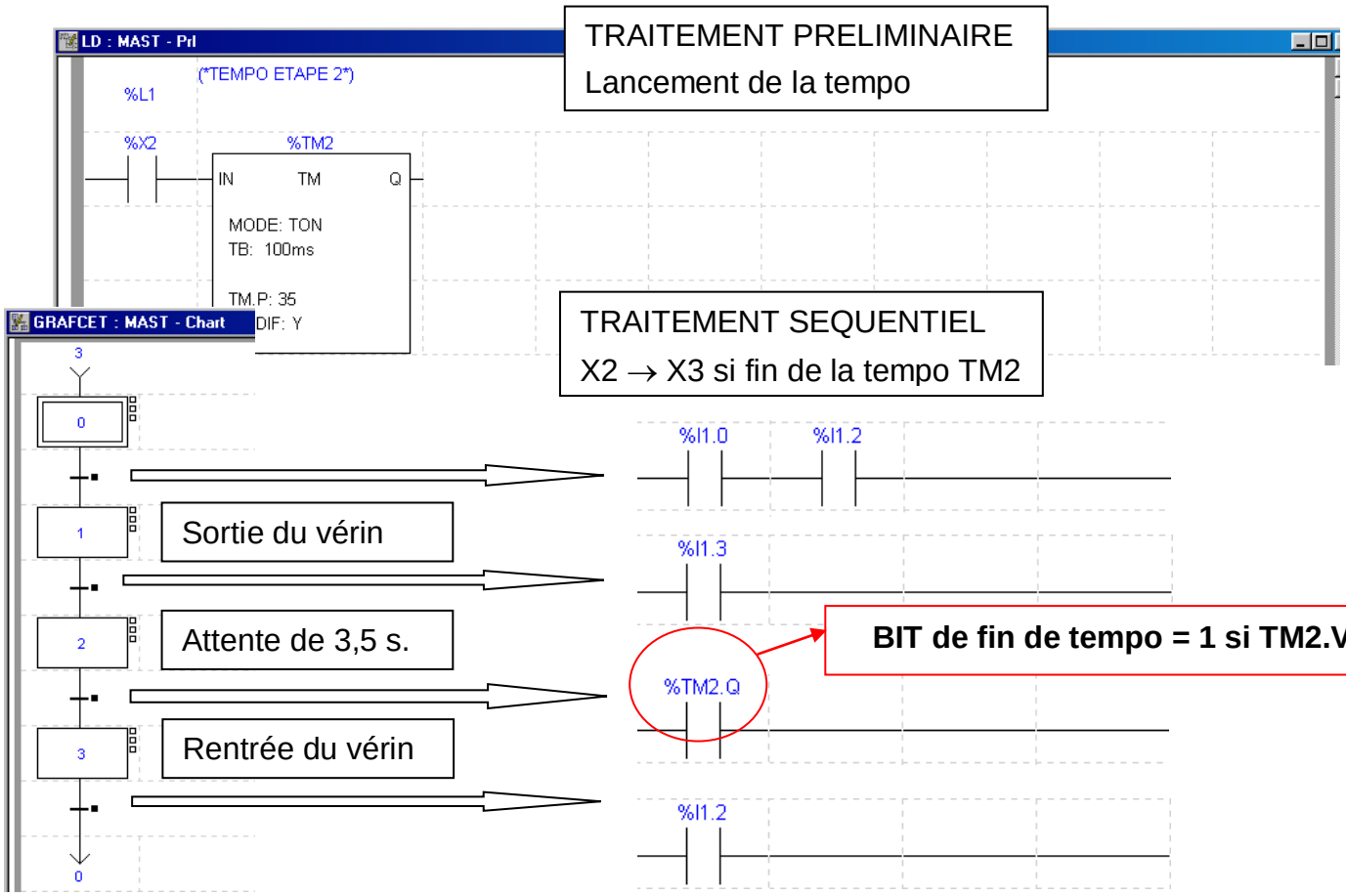
Exemple 2 : programmation de temporisation

1/ PROGRAMMATION EN SCHEMA CONTACT



Mode : TON tempo « travail »
TOF tempo « repos »
TP impulsion « monostable »
TM.P : présélection
TB : base de temps

2/ PROGRAMMATION EN GRAFCET



TRAITEMENT PRELIMINAIRE
Lancement de la tempo

TRAITEMENT SEQUENTIEL
X2 → X3 si fin de la tempo TM2

Sortie du vérin

Attente de 3,5 s.

Rentrée du vérin

BIT de fin de tempo = 1 si TM2.V =

TRAITEMENT POSTERIEUR
Pilotage des mouvements

TD N° 1 :
MINI AUTOMATE MILLENIUM 3

1. Travail demandé :

1 Schémas et symboles aux normes
2 Ecriture du programme
3 Programmation du module
4 Schéma de raccordement du module

2. Objectif :

Découvrir et mettre en œuvre le mini automate MILLENIUM et son logiciel de programmation (Crouzet Logic Software).

3. Cahier des charges :

Un particulier désire que l'accès à son domicile soit contrôlé par un portail automatisé équipé d'un moteur à double sens de rotation (ouverture ou fermeture).

Ouverture :

Que le portail soit fermé ou qu'il soit en position intermédiaire, le signal de la télécommande provoque l'ouverture complète du portail.

Durant l'ouverture, chaque nouvelle action sur la télécommande stoppe ou relance le moteur.

Dés que le portail est complètement ouvert, une temporisation de 4 secondes retarde sa fermeture.

Fermeture :

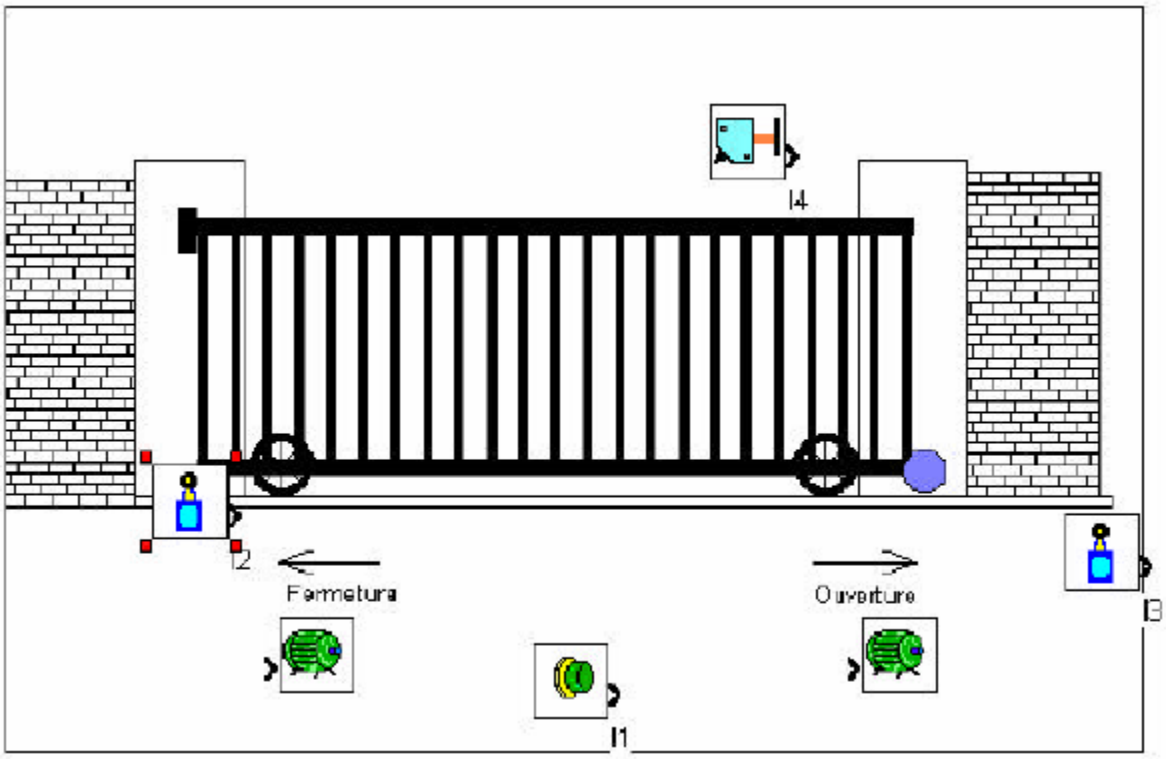
Pendant la fermeture, un capteur provoque l'ouverture complète si un passage est détecté ou si la télécommande est actionnée.

Tant que ce détecteur est activé (véhicule arrêté dans le passage par exemple), le portail reste complètement ouvert.

Signalisation :

Une balise clignote durant toute la phase d'ouverture et de fermeture du portail.

4. Disposition de la partie opérative :



5. Tableau des entrées. / sorties :

A chaque bouton poussoir il sera affecté une adresse correspondant à une entrée sur le module.

A chaque bobine de contacteur ou voyant il sera affecté une adresse correspondant à une sortie sur le module.

Comme l'indique le tableau ci-contre:

ENTREES		SORTIES	
I1	Télécommande	**Q1**	Ouverture du portail
I2	Portail position fermé	**Q2**	Fermeture du portail
I3	Portail position ouvert	**Q3**	Balise clignotante
I4	Détecteur de passage		

6. Programmation de l'automate Millénium :

Lancez le logiciel :Crouzet Logic Software, puis vous allez créer un nouveau programme.

Vous devrez ensuite choisir votre automate.

Vous pouvez maintenant programmer. Ce logiciel permet de tester votre programme, il vous faut lancer le mode simulation (S).

7. Schéma et câblage du module avec son environnement :

Complétez les phrases suivantes :

8. Les entrées et les sorties:

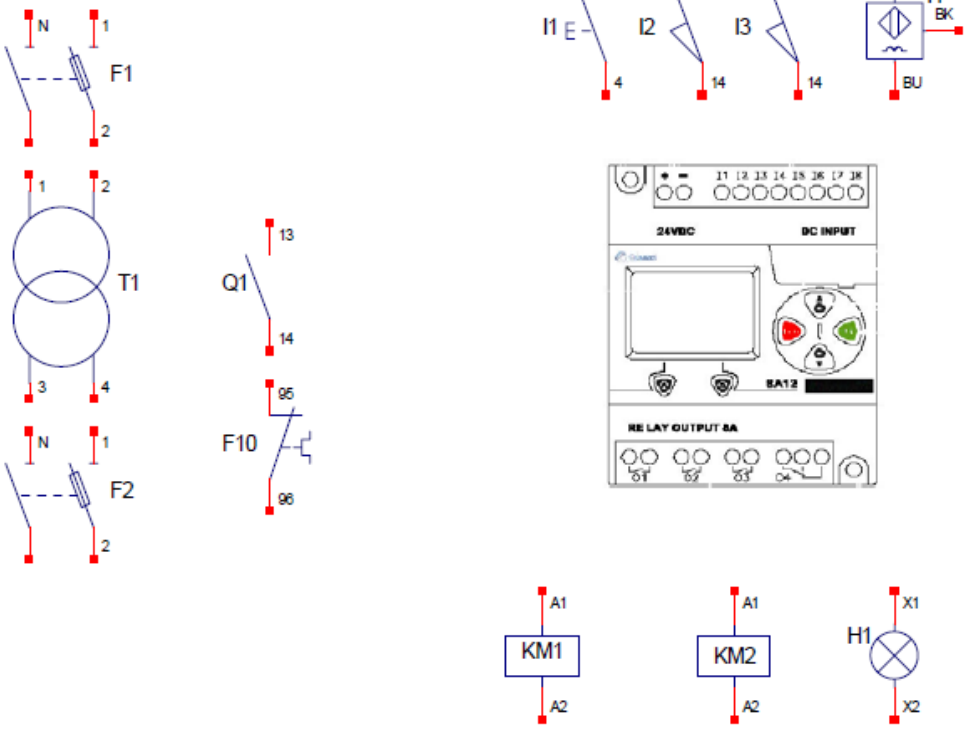
Afin de faciliter la vérification du Câblage, vous devrez utiliser des Cordons bleu pour l'alimentation, **Rouge** pour les entrées et **noir** pour les sorties.

Réalisez le schéma de câblage des entrées sorties sur la feuille ci-dessous en respectant les consignes suivantes :

- Partie alimentation : Les conducteurs d'alimentation seront tracés en bleu.

- Partie entrées : Les entrées seront tracées en rouge.

- Partie sorties: Les sorties seront tracé en noir.



TD N° 2 :
PROGRAMMATION EN LANGAGE LD et SFC SOUS LE LOGICIEL PL 7 et Step 7

1. Cahier de charge: étude d'un chariot automatise

Le dessin ci-après fig1 représente un système de transfert de sable.

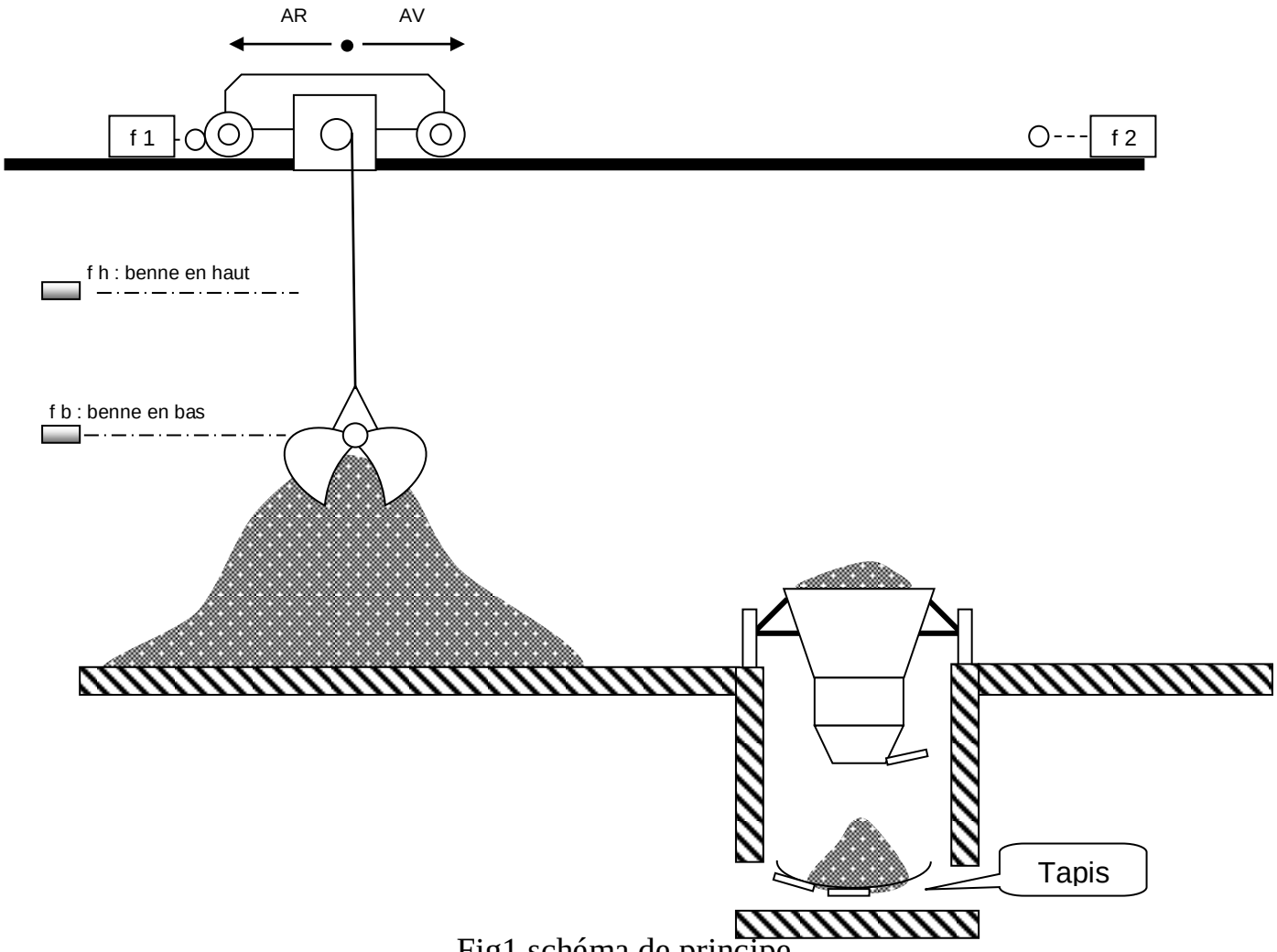


Fig1 schéma de principe

2. Description du cycle de fonctionnement

Au départ le chariot est au-dessus du tas de sable, en position haute et la benne est ouverte. L'appui sur un **bouton poussoir marche-cycle** provoque la descente de la benne sur le tas de sable, sa fermeture, puis sa remontée. En fin de montée, le chariot se déplace jusqu'au dessus

de la trémie. Dans cette position, il y a descente de la benne jusqu'a la position basse, ouverture, puis la remontée (en position haute). Enfin, le chariot repart en arrière à sa position d'origine au-dessus du tas de sable, et le cycle s'arrête.

- **Capteurs :**

f1 : chariot au dessus du tas.

f2 : chariot au dessus de la trémie.

fh : benne en position haute.

fb : benne en position basse.

fo : benne ouverte.

ff : benne fermée.

- **NB :** les capteurs **fh**, **fb** sont des détecteurs photoélectriques d'une portée de 10m type reflex, alors que les détecteurs **fo** et **ff** sont des capteurs inductifs placés directement sur la benne (capteurs non représentés sur le schéma).

- **Moteurs :**

T : moteur translation (avant / arrière)

L : moteur de levage (montée / descente)

B : moteur d'ouverture et fermeture benne.

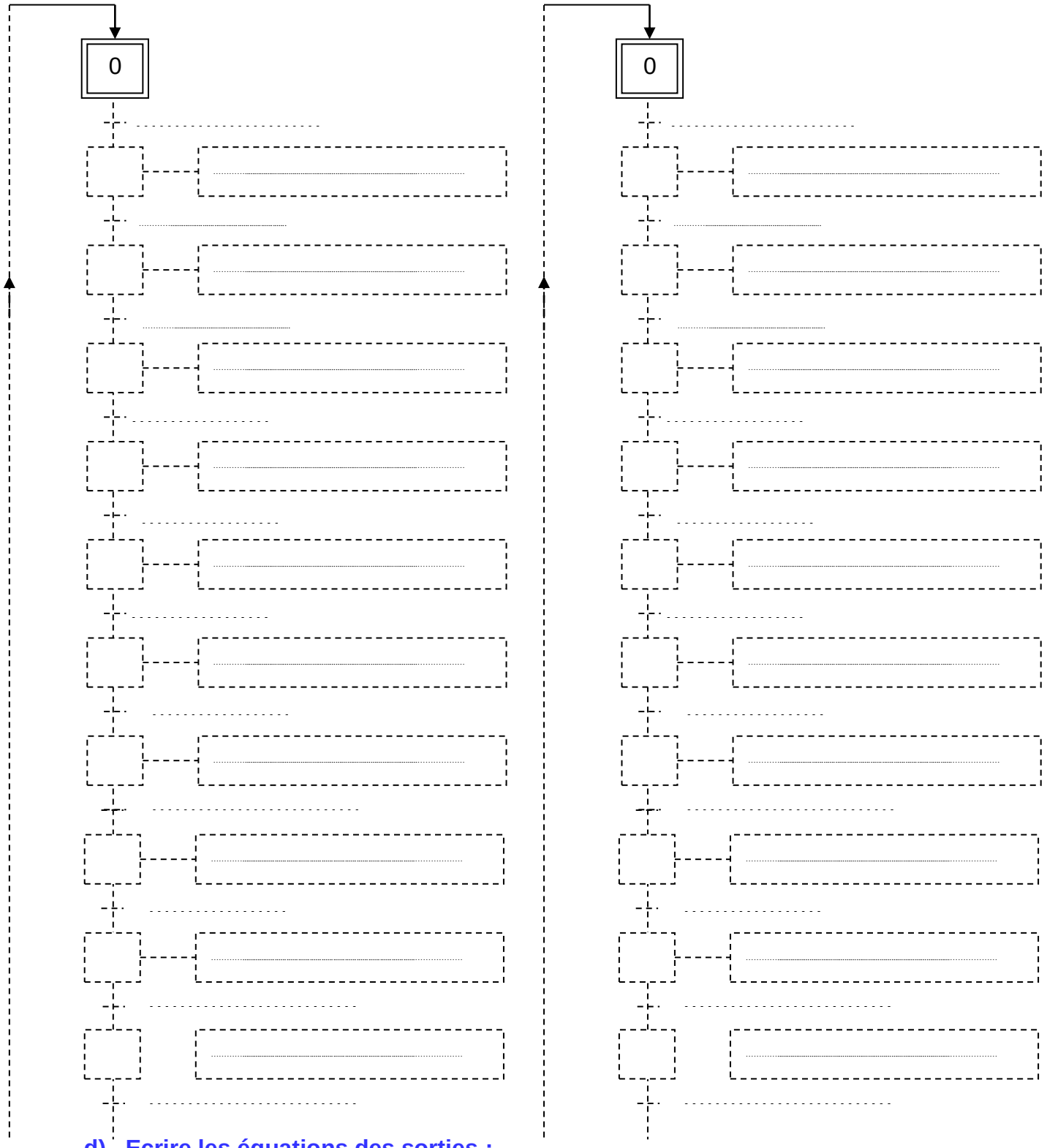
3. Travail demandé :

a) Complétez le tableau des entrée et sorties avec les adresse **API correspondantes**;
b) Donnez le **GRAFCET** du point commande et du point de vue automate, fig2 ; fig3;
c) Ecrire les équations des sorties (préactionneurs);
d) Programmez selon le **logiciel PL7** le GRAFCET (**chart**) , les réceptivités et les actions (**programmation postérieur**), ainsi que la **programmation préliminaire** avec **Initialisation de l'étape initiale et remise à zéro du système** ;
e) Programmez selon le **logiciel Step7**
f) Complétez le schéma de câblage de l'automate **en logique négative**.

b) Tableau Affectation des entrées / sorties

Désignation	Repère	Affectation adresse API		
		PL7	Step7	
.....	.....	.....	.....	Bouton Poussoirs et capteurs
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	Pré actionneurs
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	
.....	.....	.....	.....	

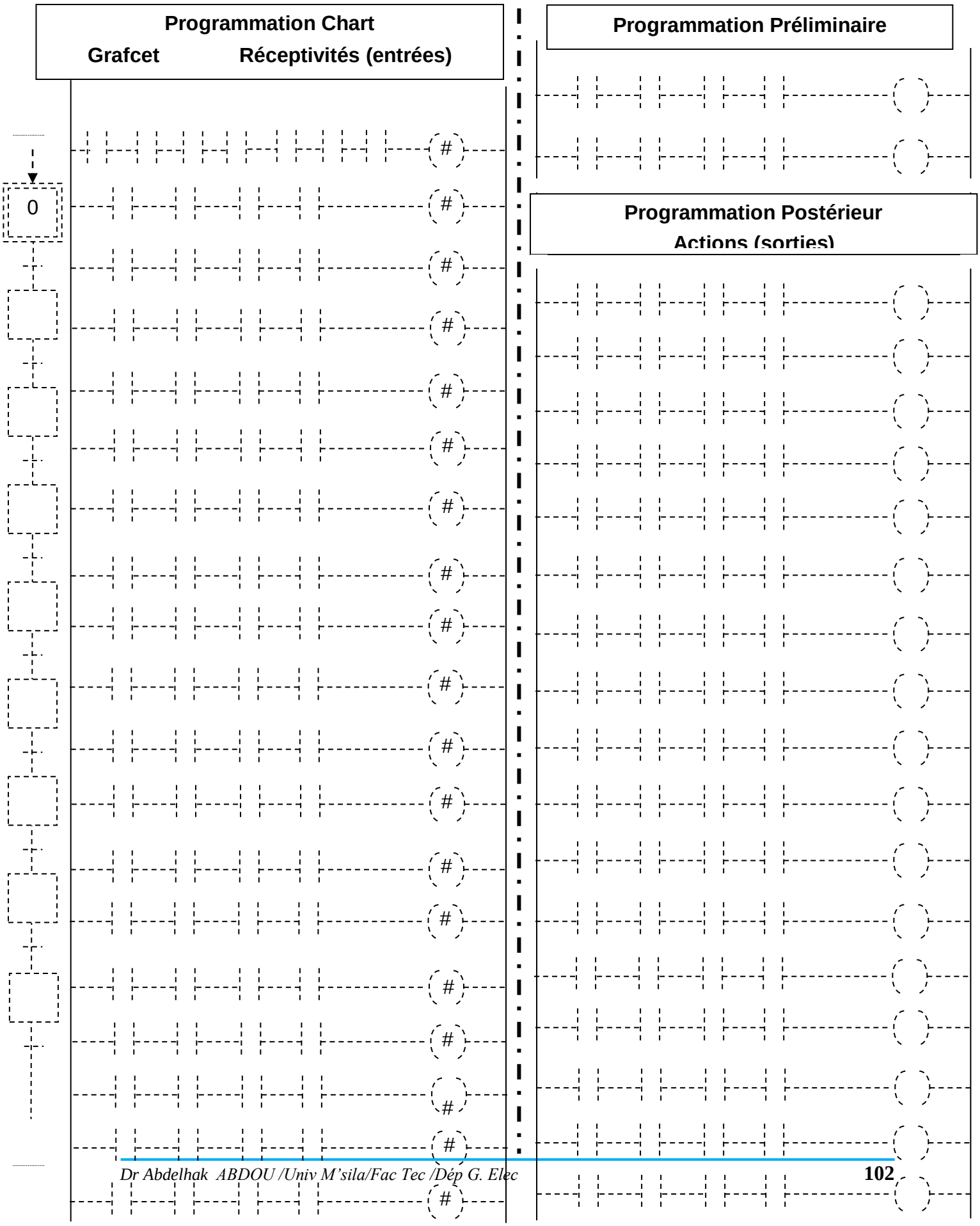
b) GRAFCET point de vue commande (PC) ; c) GRAFCET point de vue automate (API):



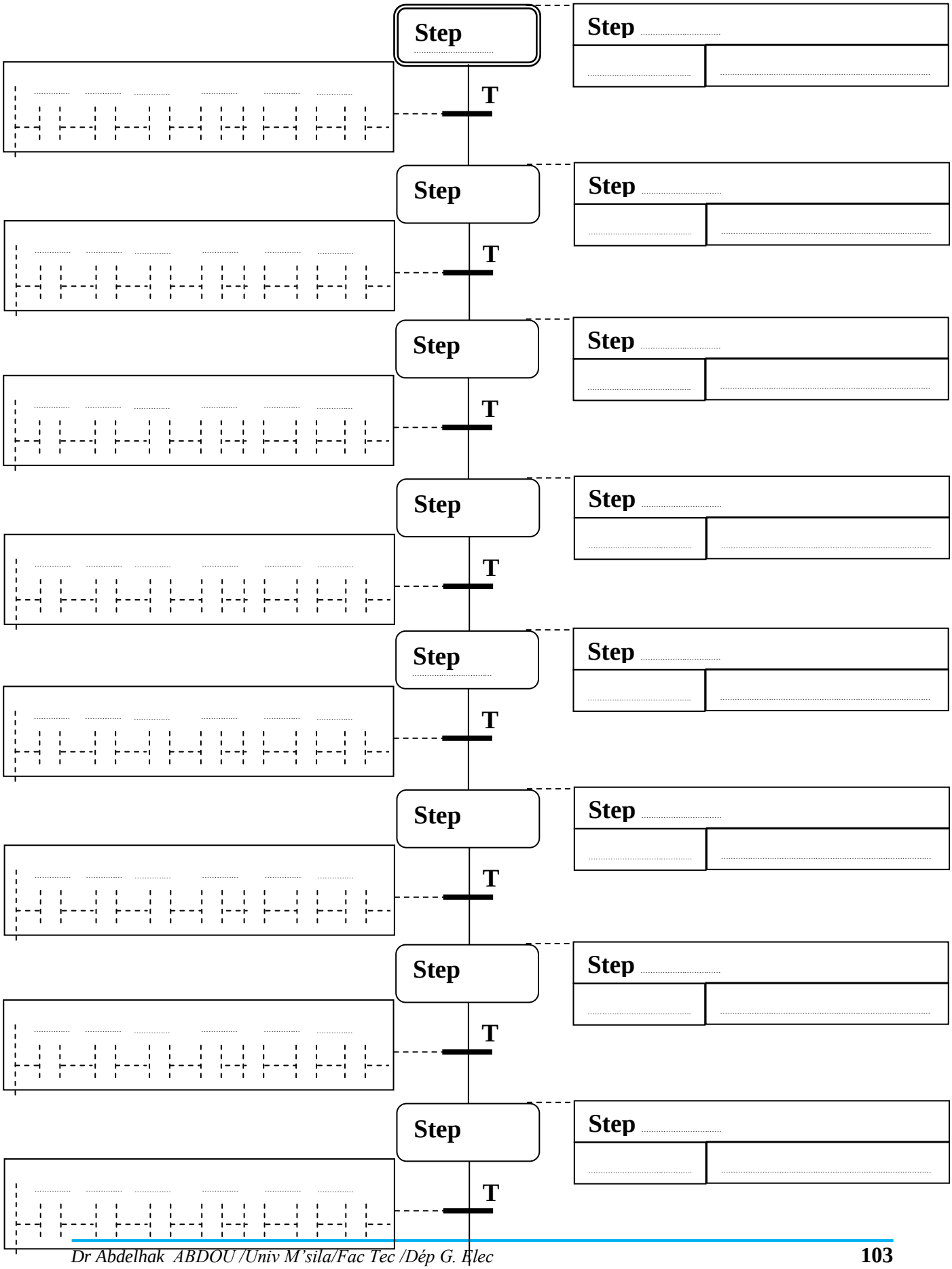
d) Ecrire les équations des sorties :

…… = …… …… = ……
…… = …… …… = ……
…… = …… …… = ……

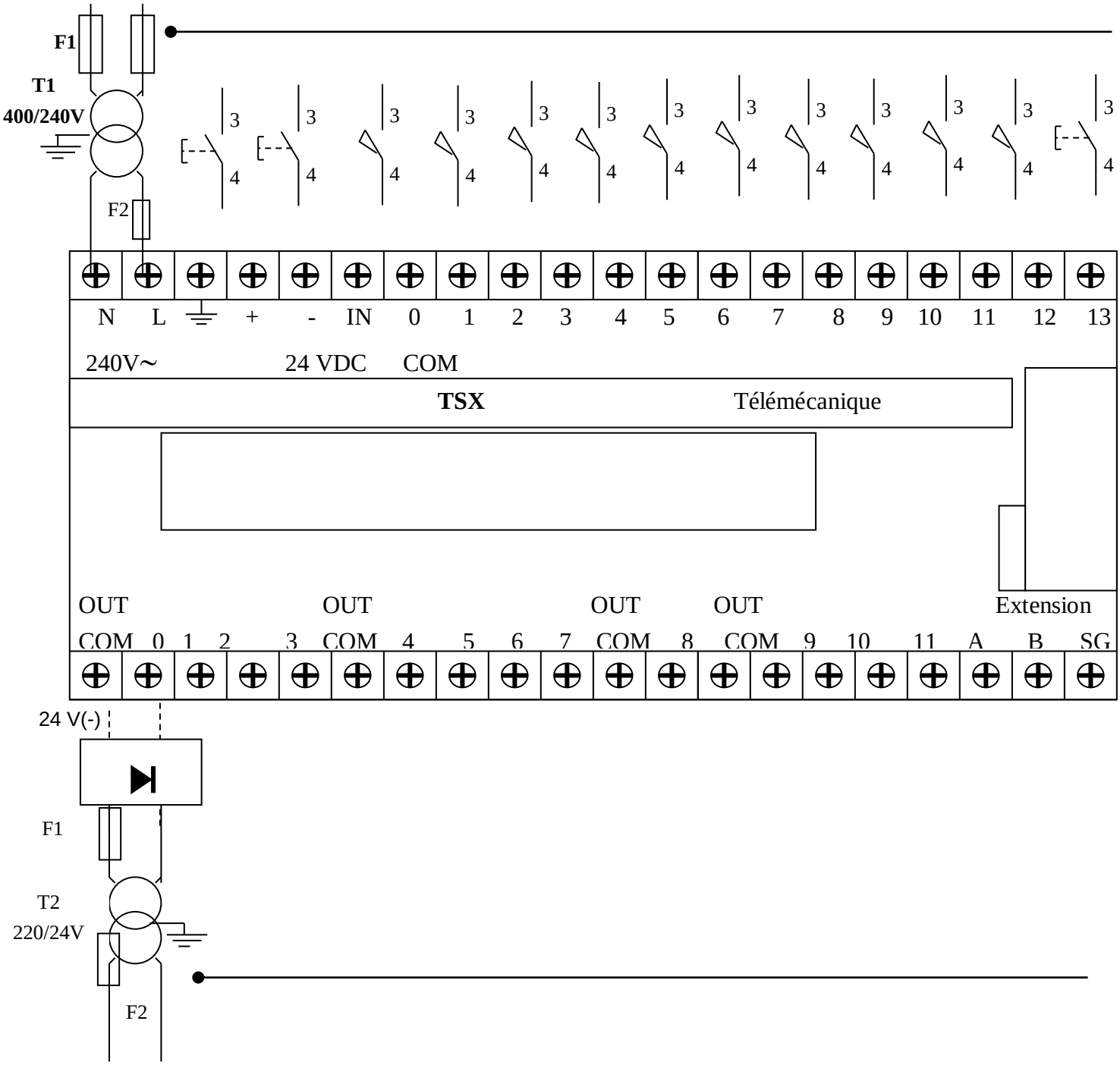
e) Document réponse programmation de réceptivités et actions en langage SFC sous PL7,



f) Document de réponse ; programmation sous Step7

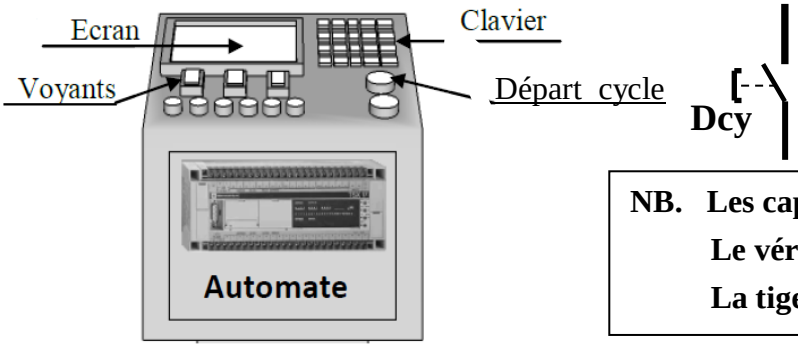


g) Schéma de câblage de l'automate

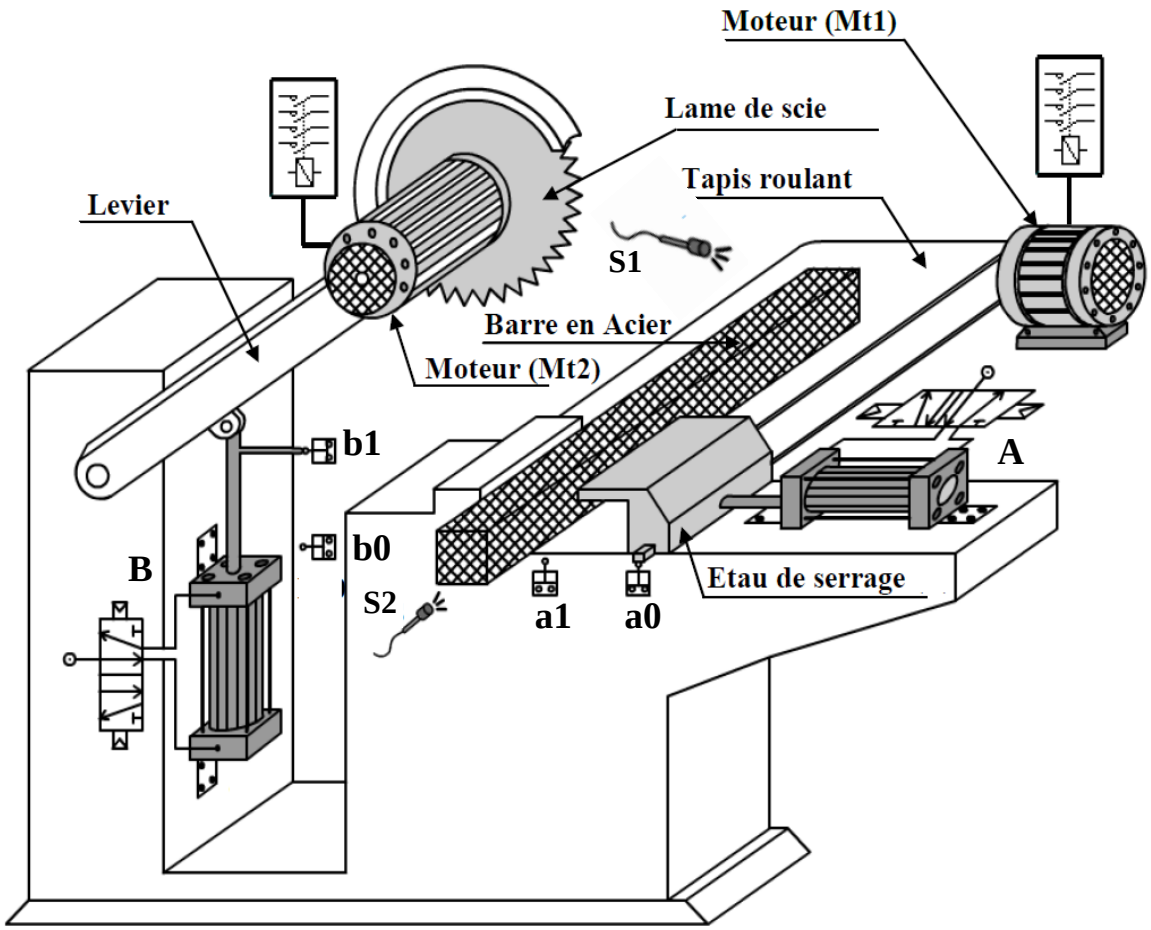


TD N° 3 :

Système technique : " Poste automatique de sciage de barre en acier ":



NB. Les capteurs a_0, et b_1, sont actionnés au repos
Le vérin A, B sont des vérins doubles effets.
La tige du vérin B est sortante au repos,



a) Donnez le GRAFCET du point commande et du point de vue automate;

b) Ecrire les équations de sorties (préactionneurs) ;

c) Programmez le GRAFCET (chart) , les réceptivités et les actions (programmation postérieur) ;

d) Complétez le schéma de câblage de l'automate, (EN LOGIQUE POSITIVE).

I- Description du fonctionnement :

Le système automatisé proposé est un poste automatique de sciage de barre en acier.

Des barres en acier arrivent par gravité sur le tapis roulant. Un capteur **(S1)**, détecte la présence des barres.

Lorsque l'opérateur actionne le bouton poussoir **(Dcy**), avec présence de caisse **(S1),** les actions suivantes se produisent :

- Rotation du moteur **Mt** , pour déplacer la barre vers le poste de sciage ;
- Action **S2**, arrêt du moteur Mt1 du tapis roulant , sortie de la tige des vérin **A** pour serrer (immobiliser) la barre ;
- Action sur $\mathbf{a_1}$, démarrage du moteur de scie **Mt2** et rentrée la tige du vérin **B** pour faire descendre la scie vers la barre ;
- Action sur $\mathbf{b_0}$, la tige du vérin **B** sorte et le moteur **Mt2** toujours en marche ;
- Action sur $\mathbf{b_1}$, arrêt du moteur de scie **Mt2** et la tige du vérins **A** rentre pour libérer la barre ;
- Action sur $\mathbf{a_0}$, **le cycle se termine**.

<u>Travail demandé :</u>

a) Complétez le tableau des entrées et sorties avec les adresse API correspondantes ;

b) Donnez le GRAFCET du point commande et du point de vue automate;

c) Ecrire les équations des sorties (préactionneurs) ;

d) Programmez le GRAFCET (chart) , les réceptivités et les actions (programmation postérieur) ;

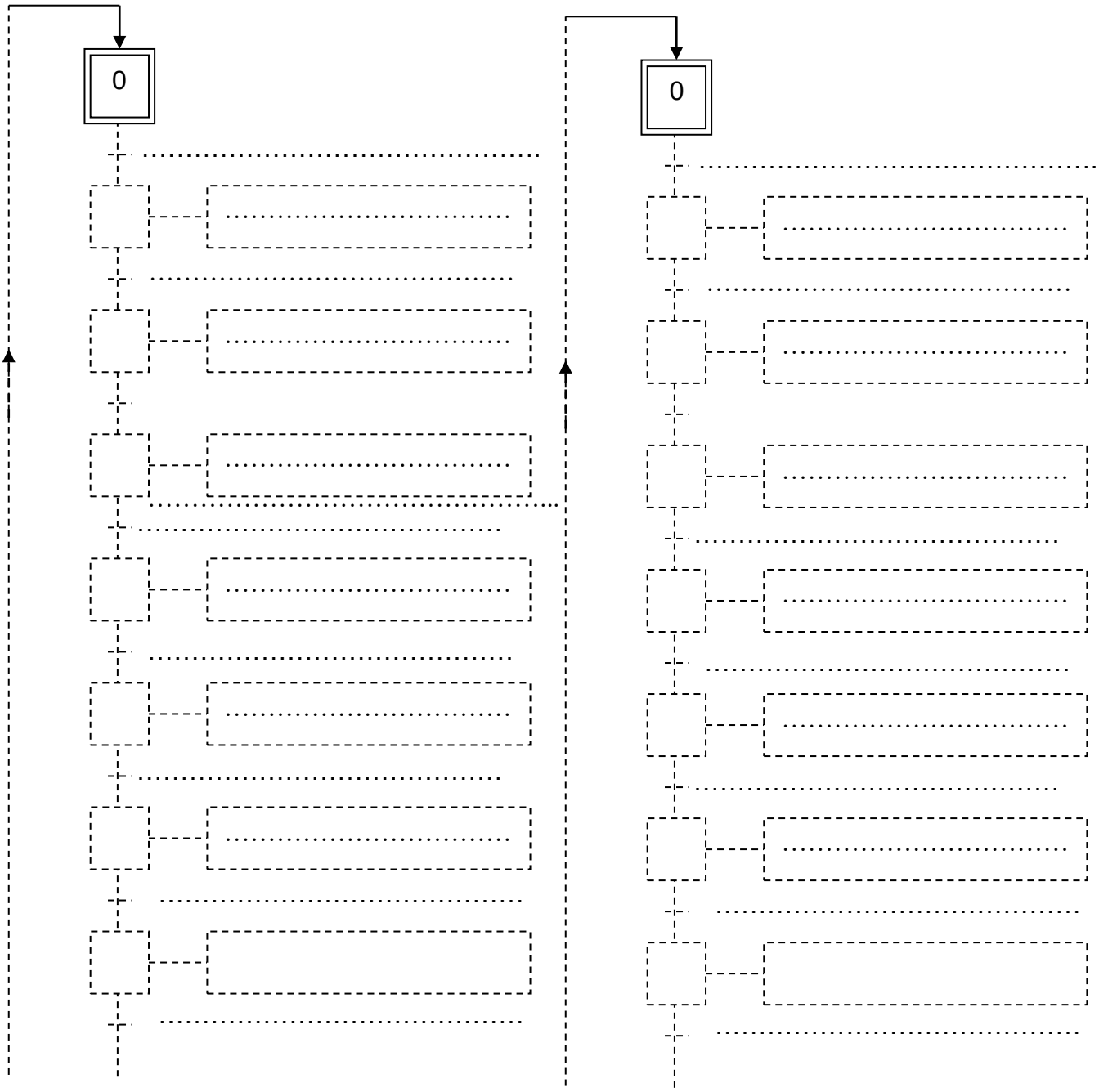
e) Complétez le schéma de câblage de l'automate, (EN LOGIQUE POSITIVE).

a) Tableau des affectations des entrées / sorties

Numéroter les adresses API a partir de numéro « 0 » et selon cet ordre

Désignation	Repère	Affectation adresse API	
Bouton Poussoir départ cycle	**Dcy**	.....	**Bouton Poussoirs et capteurs**
Capteur Présence caisse sur tapis roulant	**S1**	.....	
Capteur Présence caisse au poste d'agrafage	**S2**	.....	
Fin de course rentrée de tige vérin A	**a_0**	.....	
Fin de course sortie tige vérin A	**a_1**	.....	
Fin de course rentrée de tige vérin B	**b_0**	.....	
Fin de course sortie tige vérin B	**b_1**	.....	
Sortie tige Vérin A	.....	.....	**Pré actionneurs**
Rentrée tige Vérin A	.....	.....	
Sortie tige Vérin B	.....	.....	
Rentrée tige Vérin B	.....	.....	
Moteur du tapis roulant	**Mt1**	.....	
Moteur de scie	**Mt2**	.....	

b) Le GRAFCET point de vue partie commande (PC) : **c) Le GRAFCET point de vue partie API**



d) Ecrire les équations des sorties :

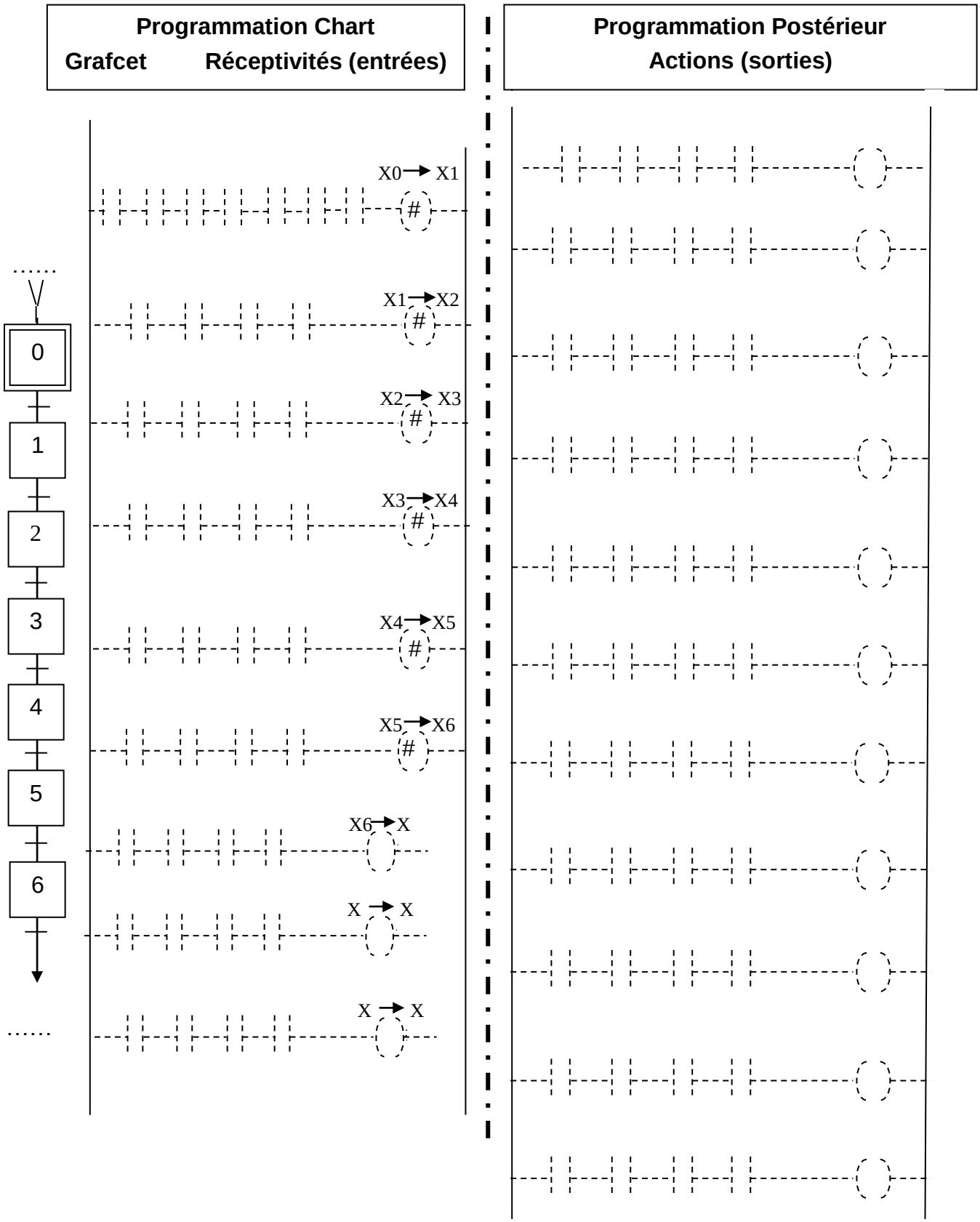
…….= ……. …….= …….

…….= ……. …….= …….

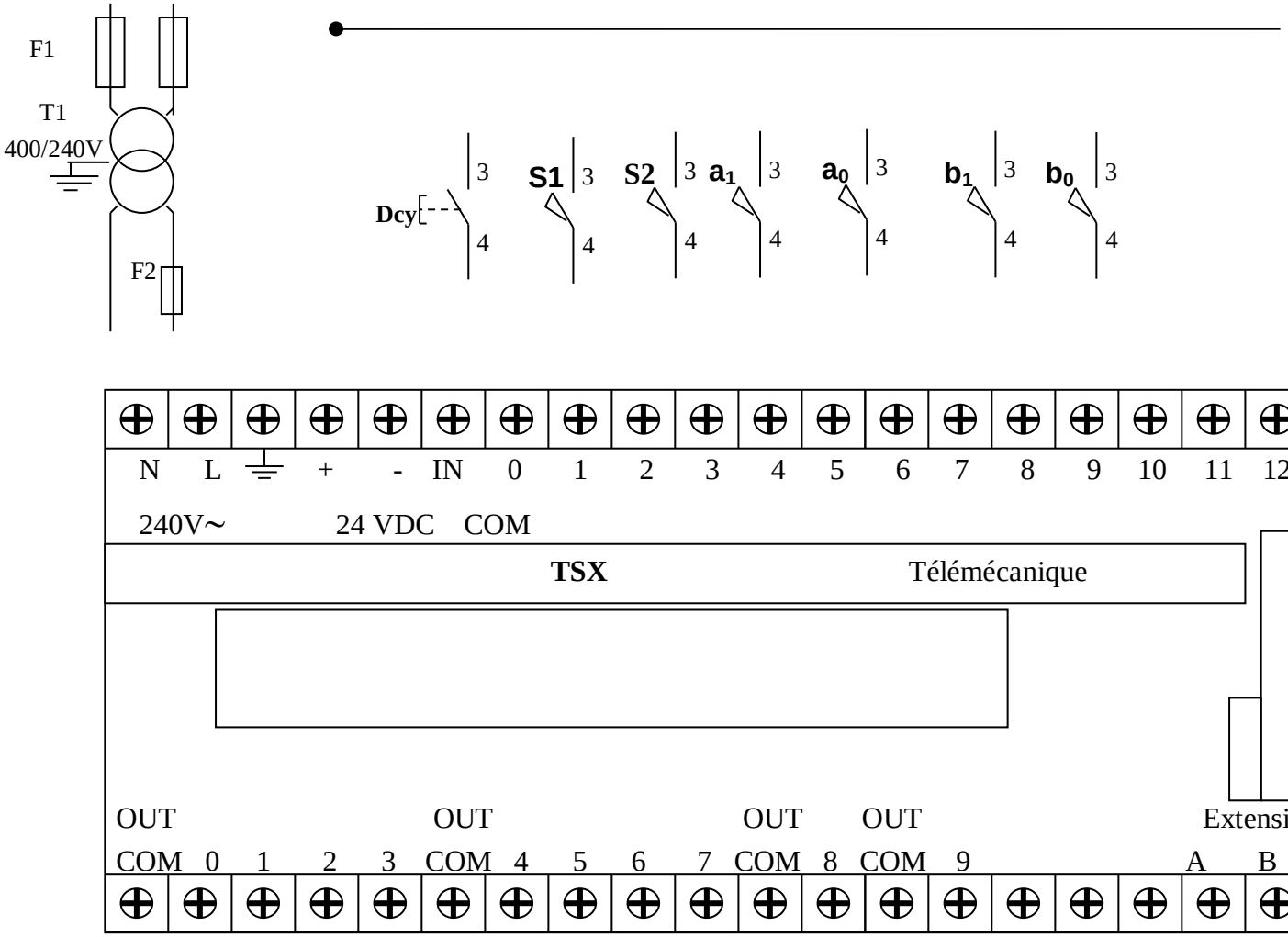
…….= ……. …….= …….

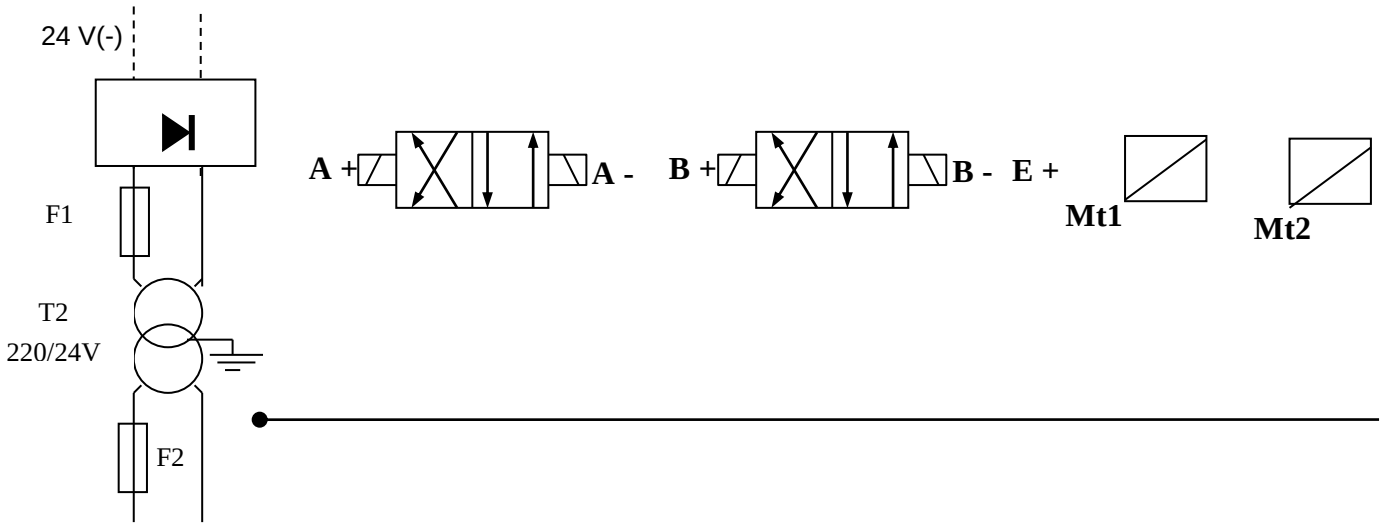
…….= ……. …….= …….

e) Document réponse programmation de réceptivité et action en langage GRAFCET



f) le schéma de câblage de l'automate





TD N°4:
PROGRAMMATION EN LANGAGE LD et SFC SOUS LE LOGICIEL PL 7

1. Cahier de charge : étude d'un système automatisé qui fabrique et embouteille des bidons de liquide lave glace pour les voitures:

À la fin de la ligne d'embouteillage, les bidons sont emballés manuellement par groupes de deux (2) dans des petits cartons ou par groupes de quatre (4) dans des grands cartons, pour être expédiés à différents distributeurs, Fig 1.

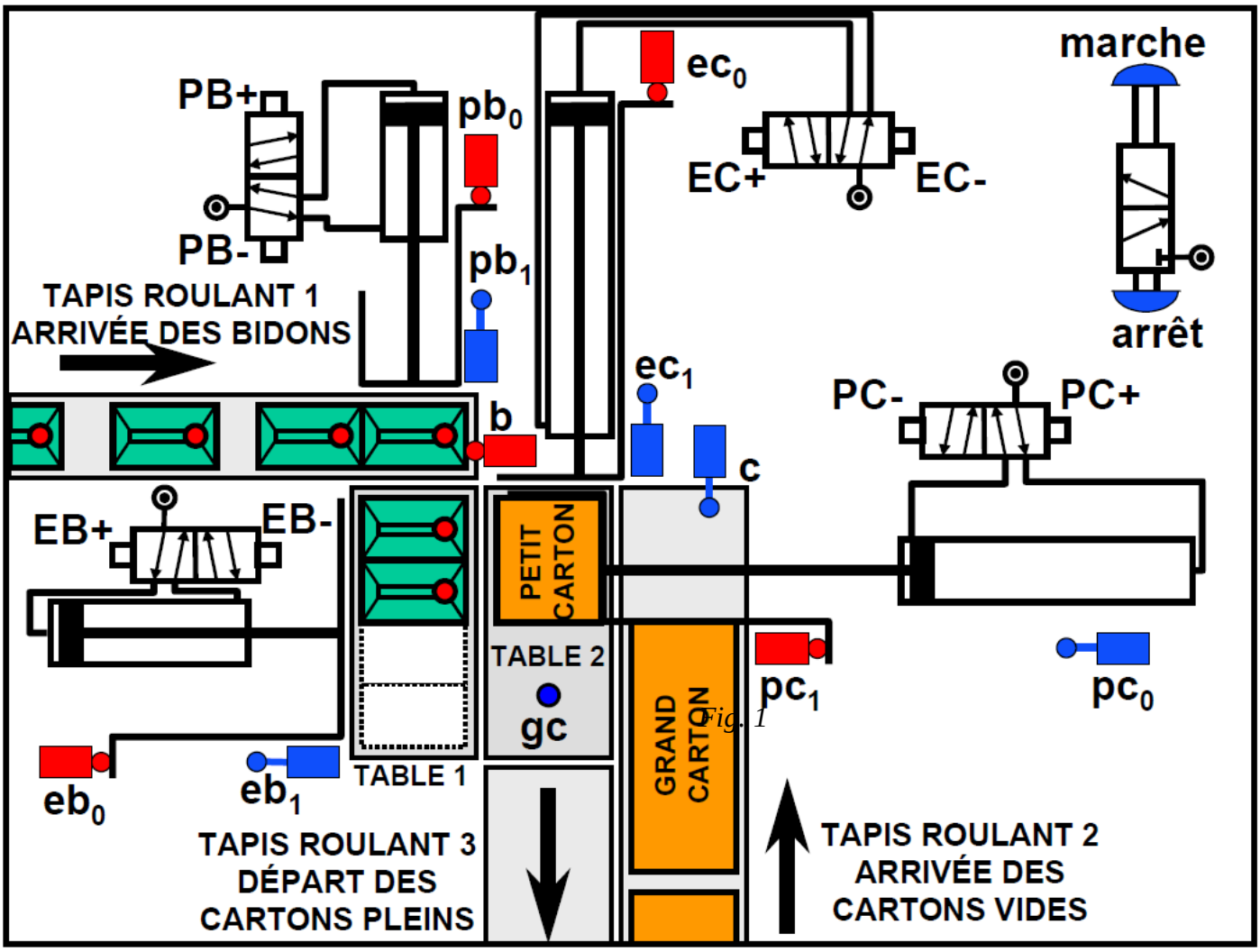


Fig. 1

2. Description de la machine :

- Trois (3) tapis roulants, qui ne font pas partie du système à étudier :
  - tapis 1 : fonctionne en continu et amène les bidons à la machine d'emballage;
  - tapis 2 : fonctionne en continu et amène les cartons vides à la machine d'emballage;
  - tapis 3 : fonctionne en continu et évacue les cartons pleins vers l'étiquetage et l'expédition.

- Deux (2) tables :
  - table 1 : reçoit les bidons à emballer, par groupe de deux (2) ou de quatre (4);
  - table 2 : reçoit le carton d'emballage, et permet de reconnaître le type de carton à remplir grâce à la cellule photo-électrique « gc », qui est active (=1) lorsque c'est un grand carton qui est sur la table, et qui est inactive (=0) lorsque c'est un petit carton.

- Quatre (4) **vérins à double effet** pilotés par des **distributeurs 5/2 bi-stables** :
  1) vérin « **P**oussoir des **B**idons » muni de deux (2) capteurs de fin de course « **pb0** » (poussoir rentré) et « **pb1** » (poussoir sorti) : ce vérin ne peut sortir (**PB+**) que s'il y a un bidon au bout du tapis roulant 1 (capteur « **b** » actif) et chaque fois que ce vérin sort (**PB+**), il pousse un bidon sur la table 1. Ce vérin rentre sur l'ordre « **PB-**».
  2) vérin « **E**mballage des **B**idons » muni de deux (2) capteurs de fin de course « **eb0** » (vérin rentré) et « **eb1** » (vérin sorti) : chaque fois que ce vérin sort (**EB+**), il pousse deux (2) ou quatre (4) bidons dans le carton. Ce vérin rentre sur l'ordre «**EB-** ».
  3) vérin « **P**oussoir des **C**artons » muni de deux (2) capteurs de fin de course « **pc0** » (poussoir rentré) et « **pc1** » (poussoir sorti) : ce vérin ne peut sortir (**PC+**) que s'il y a un carton au bout du tapis roulant 2 (capteur « **c** » actif) et chaque fois que ce vérin sort (**PC+**), il pousse un carton sur la table 2. Ce vérin rentre sur l'ordre « **PC-**».
  4) vérin « **E**vacuation des **C**artons pleins » muni de deux (2) capteurs de fin de course « **ec0** » (vérin rentré) et « **ec1** » (vérin sorti) : chaque fois que ce vérin sort (**EC+**), il pousse le carton contenant deux (2) ou quatre (4) bidons sur la tapis roulant 3 pour l'évacuation. Ce vérin rentre sur l'ordre « **EC-** ».
     a. Un bouton **bi-stable** de mise en route du cycle « **marche** », et d'arrêt du cycle
        i. « **arrêt** ».
     b. Une cellule photo-électrique « **gc** » sur la table 2, qui permet de reconnaître un **g**rand **c**arton, lorsque la cellule vaut « **1** ».

     • Un capteur de présence de **b**idon « **b** » au bout du tapis roulant 1, qui autorise sortie du vérin poussoir de bidons **PB**. Si le capteur « **b** » n 'est pas actif, le vérin **PB** ne doit pas sortir.

     • Un capteur de présence de **c**arton « **c** » au bout du tapis roulant 2, qui autorise la sortie du vérin poussoir de cartons **PC**. Si le capteur « **c** » n 'est pas actif, le vérin **PC** ne doit pas sortir.

3. Description du cycle de fonctionnement :

La fig1 représente le système à étudier dans son **état initial**, avant la mise en route du cycle :

- Le vérin PB est rentré;
- Le vérin EB est rentré;
- Le vérin PC est sorti;
- Le vérin EC est rentré;
- Il y a deux (2) bidons sur la table 1;
- Il y a un (1) carton (petit ou grand) sur la table 2.

Si la machine est bien dans son état initial et que le carton sur la table 2 est un grand carton, dès que le bouton marche est activé, le poussoir PB sort et rentre deux fois (il sort à condition qu'il y ait un bidon au bout du tapis 1 à chaque fois = capteur « b »), afin d'amener deux (2) bidons de plus sur la table 1.

Lorsque le poussoir PB est revenu en position rentrée la deuxième fois, les bidons de la table 1 sont emballés dans le carton de la table 2, à l'aide du vérin EB.

Ensuite, le vérin EB et le vérin poussoir de cartons PC rentrent simultanément.

Dès que tous les deux sont arrivés en position rentrée, deux séquences se produisent en même temps :

1) On attend qu'il y ait un bidon au bout du tapis 1 (capteur b);
Dès que c'est le cas, le vérin poussoir de bidons PB sort et rentre deux fois (à condition qu'à la deuxième sortie il y ait un bidon sur le capteur « b »), afin d'amener deux (2) nouveaux bidons sur la table 1.

Une fois qu'il est rentré la deuxième fois, on attend la fin de la séquence 2.

2) Le vérin EC évacue le carton rempli de bidons pleins sur le tapis roulant 3, puis revient en position rentrée.

Dés qu'il est revenu en position rentrée et qu'il y a un carton au bout du tapis roulant 2 (capteur « c »), le vérin poussoir de carton PC pousse le carton sur la table 2 et reste sorti.

Quand c'est fait, on attend la fin de la séquence 1.

Dès que les deux séquences sont terminées, le système est à nouveau à l'état initial et le cycle est fini.

Si, lors de la commande de mise en route, le carton sur la table 2 est un petit carton (au lieu d'en être un grand), alors le cycle commence directement par l'emballage (vérin EB) des deux (2) bidons de la table 1.

4. Travail demandé :

a) Complétez le tableau des entrée et sorties avec les adresse **API correspondantes**;

b) Donnez le **GRAFCET** du point commande, fig2 ;

c) Donnez le **GRAFCET** du point de vue automate, fig3;

d) Ecrire les équations des sorties (préactionneurs);

e) Programmez selon le **logiciel PL7** le GRAFCET (**chart**) , les réceptivités et les actions (**programmation postérieur**), ainsi que la **programmation préliminaire** avec **Initialisation de l'étape initiale et remise à zéro du système** ;

Complétez le schéma de câblage de l'automate **en logique positive**.

Références bibliographiques

[1] : Henry NEY, Automatique et informatique industriel, éditions Nathan, Paris 1998 ;

[2] : Pièrre BOYE André BIANCIOTTO, Le schéma en électrotechnique, éditions Delagrave Paris 1982 ;

[3] : Henry NEY, Eléments d'automatismes, éditions Nathan, Paris 1985 ;

[4] : F DEGOULANGE, R LEMAITRE, D PERRIN,Automatismes , éditions Technique et vulgarisation ;

[5] : J MONTAGNAC, Cours de schémas automatisme – électricité, éditions Dunod, Paris 1985 ;

[6] : Philippe RAYMOND, Les Automates Programmables Industriels (API), Notes de cours 2005 ;

[7] : R. CHAPPERT A. CAMPA L. THIBERVILLE, L'automatique par les problèmes, tome2, éditions Foucher Paris 1979 ;

[8] : Robert STRANDH Irène DURAND, Architecture de l'ordinateur, éditions Dunod, Paris, 2005 ;

[9] : T. DUMARTIN ; Architecture des ordinateurs, Note de cours, 2004-2005 ;

[10] : D. BLIN J. DANIC R. Le GARREC F. TROLE J.C SEITE, Les automatismes, éditions Casteilla, Paris 1984 ;

[11] : C. BOURBOUNNE J. COJEAN, Les systèmes automatisés de connaissance à la conception, Tome 2, éditions Foucher Paris 1981 ;

[12] : A. RIDEAU A. BIANCIOTTO P. BOYE, La technologie des systèmes automatisés, éditions Delagrave Paris 1997 ;

[13] : D. BOUTEILLE N. BOUTEILLE S. CHANTREUIL R. COLLOT A. SFAR, Les automatismes programmables, éditions Cepadues, Toulouse 1988 .