الجمهورية الجزائرية الديمقراطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعليم العالي و البحث العلمي
جامعة محمد بوضياف بالمسيلة
كلية التكنولوجيا
قسم الهندسة الكهربائية
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY OF M'SILA
FACULTY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
جامعة محمد بوضياف - المسيلة

رقم:416/ق هـ ك /2024 المسيلة في : 24 نوفمبر 2024

شهادة إدارية

بخصوص مطبوعة الدروس الخاصة بالأستاذ

زوقار الوليد

بناءً على محضر اللجنة العلمية لقسم الهندسة الكهربائية تحت رقم: 365/ق هـ ك/2024المنعقد بتاريخ 06نوفمبر2024 والمتضمن تعيين الخبراء: الأستاذ خطاب خثير أستاذ بجامعة المسيلة ،الأستاذ حريزي عبد الغفورأستاذ محاضر –أ-بجامعة المسيلة والأستاذ عبدو عبد الحق أستاذ محاضر -أ- بجامعة باتنة2 وذلك لتقييم مطبوعة الدروس الخاصة بالأستاذ زوقار الوليد أستاذ محاضر "ب" بقسم الهندسة الكهربائية لجامعة المسيلة تحت عنوان:

" **Microprocesseurs et Automates programmables Industriels (API)** " وبعد إطلاع رئيس اللجنة العلمية ورئيس القسم على التقارير الواردة و التي كانت كلها ايجابية، وعليه فإن اللجنة لا ترى مانعا أن تتخذه سندا في تدريس طلبة السنة الثانية ماستر كهروميكانيك، شعبة كهروميكانيك ميدان علوم و تكنولوجيا و أن تعتمد في أي تقييم للمسار العلمي للأستاذ المعني.

رئيس اللجنة العلمية
أ .د بوقرة عبد الرحمان

رئيس القسم

ملاحظة: سلمت هذه الشهادة للمعني(ة) لاستعمالها في حدود ما يسمح به القانون.

الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF DE M'SILA
FACULTE DE TECHNOLOGIE

DEPARTEMENT DE GENIE ELECTRIQUE

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

Microprocesseurs et Automates Programmables Industriels (API)

[Support de cours et travaux dirigés]

Destiné aux étudiants, 2ème Année Master **Électromécanique.**

Élaboré par :

Dr. El Oualid ZOUGGAR

(Maitre de Conférences Classe B, Département de Génie-Électrique, Université de M'sila).

Eloualid.zouggar@univ-msila.dz

Année universitaire : 2024/2025

Avant-propos

Ces dernières années, les avancées technologiques ont conduit au développement des automates programmables industriels (API) et à une révolution importante dans l'automatique.

Ce présent travail, issu d'une longue expérience pratique au niveau industriel et une dizaine d'expérience pédagogique acquise au sein de département de Génie-Électrique de l'université de M'sila, est dédié particulièrement aux étudiants d'électromécanique inscrits en deuxièmes année Master qui suivent leurs études dans la filière : Électromécanique. Il est enseigné en tant que cours intégré en semestre 3, a pour objectif de faciliter le travail des étudiants qui font leurs premiers pas dans le domaine des automates programmables industriels.

Ce support de cours permettra à l'étudiant d'acquérir des connaissances de base pour identifier les éléments technologiques permettant de piloter le fonctionnement et de faire un suivi d'un système automatisé de production, ainsi que sur l'utilisation des outils de spécification d'un automatisme industriel en vue de prévoir une durée de cycle ou une cadence de production.

Le présent document peut être scindé en deux parties essentielles :

La première partie sera consacré au microprocesseur, on va commençais par une présentation générale de l'historique des microprocesseurs et des systèmes de numération suivis d'un résumé sur les opérations arithmétiques des nombres signés et les opérations booliennes. Nous traitons ensuite en détail les architectures et les fonctions de base des microprocesseurs, tout en induisant une explication des différents types de mémoires. Une fois les connaissances de bases assimilées, nous continuant avec l'étude et la programmation du microcontrôleur ainsi que la manière dont ce dernier s'interface avec les modules externes. Tous les modules composant ce microcontrôleur sont introduits avec détail à savoir, le compteur de programme, les temps, les mémoires, les ports d'entrée et de sortie, la pile du système etc.

Une deuxième partie sera consacrée aux automates programmables industriels (API). Voici les axes abordés :

- L'architecture de base des API et les caractéristiques des entrées-sorties souvent employées avec ces systèmes ;
- Choix et câblages des API : caractéristiques, environnement, Évaluation ;
- Logiciels de programmation des API : GRAFCE, Langages de base, de calcul et séquentiel,

- Applications : Automatisations des ascenseurs, des pompes, des systèmes de ventilation, des compresseurs, des mécanismes de transport continu, des machines outils.

Des exemples et exercices complètent le présent ouvrage. Ils provoquent la réflexion du lecteur, et à ce titre, nous pensons que ce document intéressera non seulement l'étudiant mais aussi bien les esprits curieux de s'initier au nouveau concept de système.

A la fin, on trouve des indications bibliographiques permettant un approfondissement général de l'étude et analyse des systèmes de production automatisés.

J'espère que ce travail sera apprécié par mes collègues et les étudiants et je serais très content de recevoir avec reconnaissance leurs éventuelles remarques, critiques et suggestion.

Sommaire

I. Partie 1 : Microprocesseur et microcontrôleurs

I.1. Introduction :

Dans ce premier chapitre, il est question de commencer avec une présentation de l'historique des microprocesseurs. En premier lieu, il s'agira de la présentation des éléments constitutifs du microprocesseur avec le traitement en détail des architectures et les fonctions de base des microprocesseurs, tout en induisant une explication des différents types de mémoires. Ensuite, et une fois les connaissances de bases assimilées, en basant sur l'étude et la programmation du PIC16F84 de Microchip ainsi que la manière dont ce dernier s'interface avec les modules externes. Tous les modules composant ce microcontrôleur sont introduits avec détail à savoir, le compteur de programme, les timers, les mémoires, les ports d'entrée et de sortie, la pile du système etc.

Nous terminons ce cours avec l'étude d'un microprocesseur de quatre bits de toutes les instructions permettant la programmation du 16F84 en langage assembleur.

I.2. Historique des microprocesseurs :

Les ordinateurs sont apparus vers la fin de la seconde guerre mondiale. À l'époque, ils étaient immenses car chaque transistor était un tube cathodique. C'est aussi vers la fin de la seconde guerre mondiale qu'est apparue l'architecture **Von Neumann**, une architecture de base des ordinateurs modernes. L'apparition des transistors vers la fin des années 1960 (1954, Bipolar JunctionTransistor, Texas Instrument, en 1960, Metal-Oxyde Semiconductor, Bell Labs) a permis de réduire considérablement la taille des circuits avec microprocesseurs. Les transistors ont ensuite été intégrés dans de petits circuits électroniques simples (portes logiques ou amplificateur opérationnel par exemple).

Puis, ils ont été intégrés dans des circuits de plus en plus complexes (VLSI = Very Large Scale Integration). Les premiers microprocesseurs avec transistors étaient donc constitués de plusieurs circuits intégrés simples. Ils étaient conçus avec une logique câblée (le microprocesseur exécute chaque instruction avec du matériel unique) ou avec des micro instructions (le microprocesseur découpe chaque instruction en sous-instructions qui peuvent être communes à des instructions différentes).

En 1971 est apparu le premier microprocesseur entièrement contenu sur un seul die : le 4004 d'Intel. Ce précurseur a rapidement été suivi d'autres microprocesseurs, de plus en plus puissants.

Voici les jalons importants de l'histoire du microprocesseur

- ✓ Fairchild Semiconductors a inventé le premier IC (circuit intégré) en 1959.
- ✓ En 1968, Robert Noyce, Gordan Moore et Andrew Grove fondent leur propre société Intel.
- ✓ Intel est passée d'une start-up composée de trois personnes en 3 à un géant industriel en 1968.
- ✓ En 1971, INTEL a créé le microprocesseur 4004 de première génération qui fonctionnerait à une vitesse d'horloge de 108 kHz.
- ✓ De 1973 à 1978, des microprocesseurs 8 bits de deuxième génération ont été fabriqués, comme les Motorola 6800 et 6801, INTEL-8085 et Zilog's-Z80.
- ✓ En 1978, le processus Intel 8008 de troisième génération est arrivé sur le marché.
- ✓ Au début des années 80, Intel a lancé des processeurs 32 bits de quatrième génération.
- ✓ En 1995, Intel a lancé des processeurs 64 bits de cinquième génération.

I.3. Définition d'un microprocesseur

Le microprocesseur (C.P.U. : Central Processing Unit) est un circuit intégré complexe (ensemble de millions de transistors) appartenant à la famille des Very Large Scale Integration (VLSI) capable d'effectuer séquentiellement et automatiquement des suites d'opérations élémentaires (programme). Le microprocesseur remplit les fonctions suivantes :

- ❖ Il permet d'effectuer des opérations arithmétiques (additions, soustraction, multiplications, …) et logiques (Non, Ou, Et, OU Exclusif, …) et d'organiser des transferts de données entre les différents éléments du système.
- ❖ Il est chargé de décoder et d'exécuter les ordres exprimés sous forme d'instruction d'un programme.
- ❖ Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement.

Le traitement de données se fait par l'unité de traitement (U.A.L. : Unité Arithmétique et Logique ou ALU : Arithmetic and Logic Unit en anglais). Le traitement concerne la manipulation des données sous formes de transfert, opérations arithmétiques, opérations logiques, etc.

Le contrôle du système se traduit par des opérations de décodage et d'exécution des ordres exprimés sous forme d'instruction, c'est donc le séquençage des opérations et c'est le rôle de l'unité de commande.

A l'heure actuelle, un microprocesseur regroupe sur quelques millimètres carrés des fonctionnalités toujours plus complexes. Leur puissance continue de s'accroître et leur encombrement diminue régulièrement respectant toujours, pour le moment, la fameuse loi de Moore (Moore : co-fondateur de la société Inetl).

Figure I-1: Microprocesseur 4004

I.4. Rôle d'un microprocesseur

Un microprocesseur est capable de lire les instructions constituant le programme situé dans la mémoire centrale puis les décoder, et en fin exécuter ce programme.

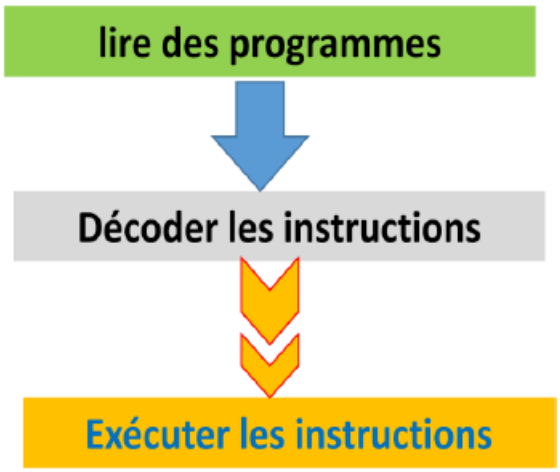


Figure I-2: Cycle d'un microprocesseur

I.5. Architecture interne d'un microprocesseur

Un microprocesseur est construit autour des éléments principaux suivants :

- ✓ Une unité de traitement.
- ✓ Une unité de commande.
- ✓ Les Bus internes

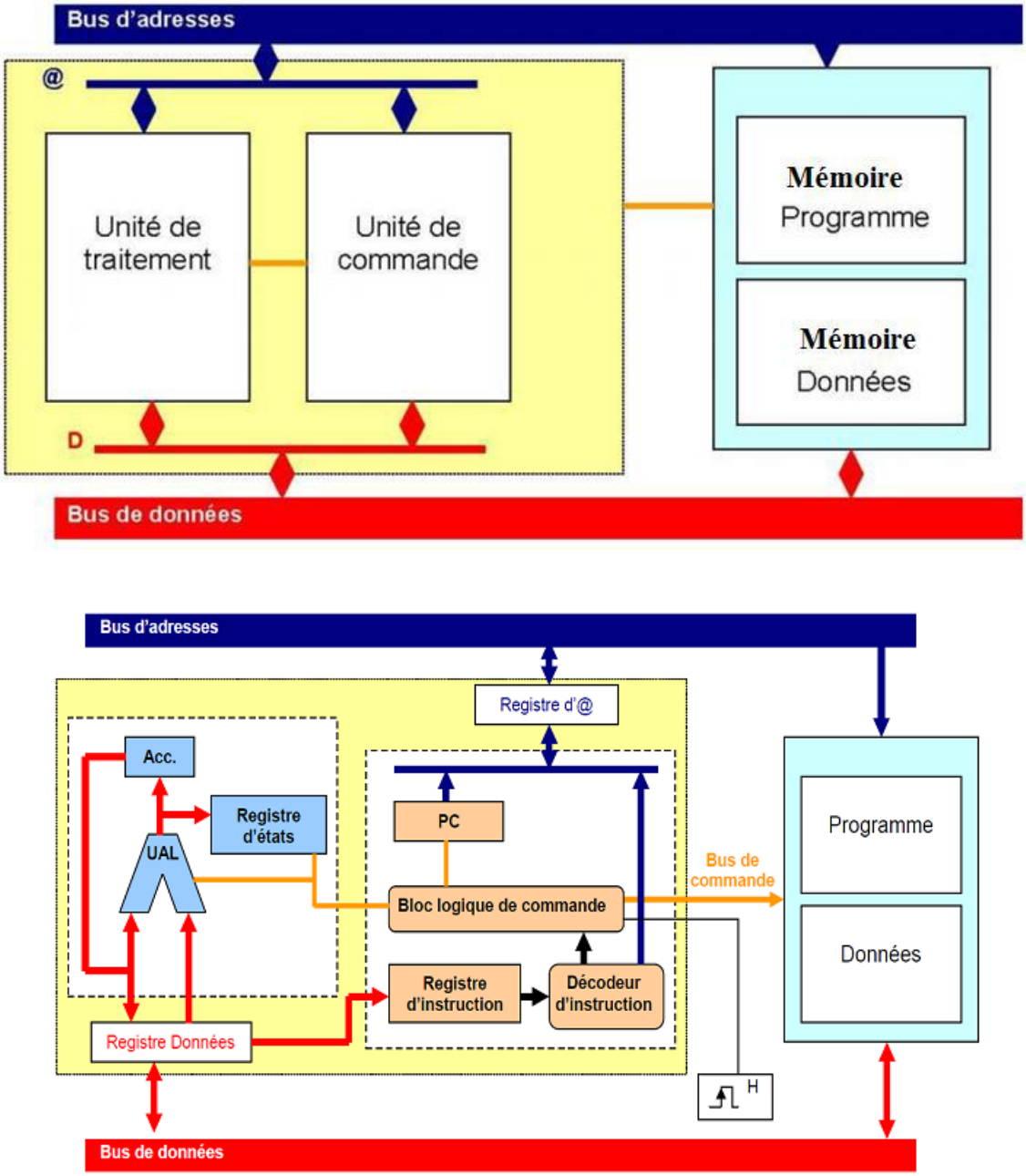


Figure I-3: Schéma fonctionnel d'un microprocesseur

I.5.1. Unité de commande (ou unité de contrôle)

Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire de l'instruction, en suite le décodage de l'instruction codée (Le décodeur de l'instruction) sous forme binaire. Enfin elle pilote l'exécution de l'instruction (Le bloc logique de commande (ou séquenceur)) puis effectue la préparation de l'instruction suivante. Pour cela, elle est composée par des registres. Un registre est une zone mémoire à l'intérieur du microprocesseur de faible taille, qui permet de mémoriser des mots mémoires ou des adresses d'une façon temporaire lors de l'exécution des instructions.

I.5.1.a. Le compteur de programme :

Compteur programme (en anglais Program Counter PC) appelé aussi compteur ordinal (CO). Il est constitué par un registre dont le contenu représente l'adresse de la prochaine instruction à exécuter. Il est donc initialisé avec l'adresse de la première instruction du programme. Puis il sera incrémenté automatiquement pour pointer vers la prochaine instruction à exécuter.

I.5.1.b. Le registre d'instruction :

Contient le code de l'instruction en cours de traitement (lu en mémoire via le bus de données).

I.5.1.c. Le registre d'adresse :

C'est un registre tampon qui assure l'interfaçage entre le microprocesseur et son environnement.

I.5.1.d. Le décodeur d'instruction :

Il décode les instructions.

I.5.1.e. Le bloc logique de commande (ou séquenceur) :

Organise l'exécution des instructions au rythme d'une horloge. Il élabore tous les signaux de synchronisation internes ou externes (bus de commande) du microprocesseur en fonction de l'instruction qu'il a à exécuter.

I.5.2. Unité de traitement

C'est le cœur du microprocesseur. Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions. L'unité de traitement est ccomposé de trois principales unités d'exécution, la première est l'unité arithmétique et logique (UAL) puis deux autres ont été ajoutés qui sont l'unité de calcul en virgule flottante et l'unité multimédia pour des raisons d'optimisation des performances des microprocesseurs.

I.5.2.a. L'Unité Arithmétique et Logique (UAL), ou ALU (Arithmetic and Logic Unit)

C'est un circuit complexe qui assure les fonctions logiques (ET, OU, Comparaison, Décalage, etc…) ou arithmétique (Addition, division, soustraction…). Comme l'objectif est de développer un programme qui gère une application voulue, toute instruction qui modifie une donnée fait toujours appel à l'ALU. Il permet donc de traiter et tester les données. Comme indiqué sur la figure I-4, l'entrée de L'UAL est connectée au bus interne par un ensemble de registres "temporaires" et aussi par un registre particulier appelé "accumulateur" ou

plus particulièrement le registre de travail (Working Register, le Registre W). Alors que la sortie de l'UAL est connectée uniquement à l'entrée de l'accumulateur. Il est à noter que les deux entrées sont précédées par une mémoire tampon appelée registres tampons ou verrous. Ces registres, qui ne peuvent être manipulés par le programmeur et le sont totalement transparents, permettent de stocker des octets aux entrées de l'U.A.L. comme l 'UAL est constitué d'une logique combinatoire, elle est dépourvue de moyens propres de stockage.

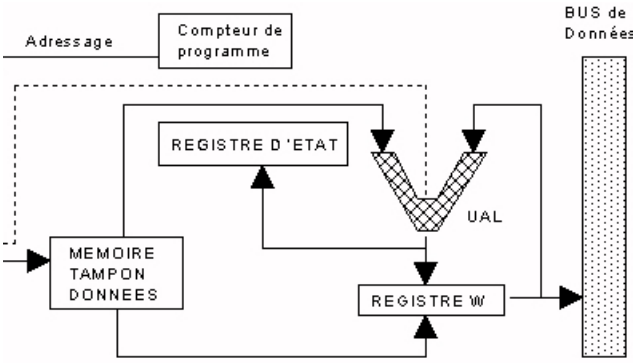


Figure I-4: Schéma de l'unité arithmétique et logique (ses entrées et ses sorties).

I.5.2.b. Unité de calcul en virgule flottante :

C'est une unité qui est capable de réaliser les opérations de calcul pour les réels ainsi que les calculs mathématiques et scientifiques complexes. A l'origine la tâche de cette unité était réalisée par tout un processeur à part, en 1989 elle a été intégrée dans les microprocesseurs afin d'optimiser les calculs.

I.5.2.c. Unité multimédia :

C'est une unité qui est chargée d'accélérer l'exécution des programmes multimédia comportant des vidéos, du son, graphisme en 3D etc.…

Cette unité porte le nom de MMX pour les premiers pentium (MutiMedia eXtensions) intégrants des fonctions de gestion du multimédia, de même la technologie 3DNOW pour les AMD et SSE pour les pentium III. Ces unités ont été créées vu la grande tendance vers la multimédia dans tous les types des programmes informatiques (jeux, logiciels sur Internet, encyclopédies…).

I.5.2.d. L'accumulateur ou le registre de travail

Le processeur utilise toujours des *registres*, qui sont des petites mémoires internes très rapides d'accès utilisées pour stocker temporairement une donnée, une instruction ou une adresse. Chaque registre stocke 8, 16, 32 ou 64 bits. Le nombre exact de registres dépend du type de processeur et varie typiquement entre une dizaine et une centaine. Parmi les registres, le plus important est le registre *accumulateur*, qui

est utilisé pour stocker les résultats des opérations arithmétiques et logiques. L'accumulateur intervient dans une proportion importante des instructions.

C'est le registre le plus important du microprocesseur, il sert systématiquement lorsque ce dernier aura besoin de "manipuler" des données. La plupart des opérations logiques et arithmétiques sur les données font appel au couple "UAL - accumulateur" selon la procédure établie, figure suivante :

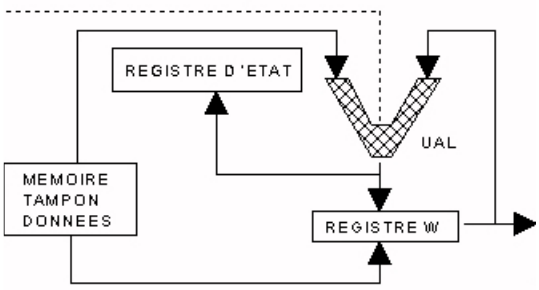


Figure I-5: Procédure de fonctionnement de l'ALU.

I.5.2.e. Le registre d'état :

C'est un registre pour lequel chacun de ses bits est un indicateur dont l'état dépend du résultat de la dernière opération effectuée par l'UAL. On les appelle indicateur d'état ou flag ou drapeaux. Dans un programme, le résultat du test de leur état conditionne souvent le déroulement de la suite du programme. Par exemple l'indicateur Z indique quand il est positionné que le résultat de l'opération est égal à Zéro. L'indicateur C indique que l'opération a généré une retenue. Le bit N indique que le résultat est négatif …

On peut citer comme indicateur :

- Retenue (carry : C)
- Retenue intermédiaire (Auxiliary-Carry : AC)
- Signe (Sign : S)
- Débordement (overflow : OV ou V)
- Zéro (Z)
- Parité (Parity : P)

Il existe d'autres types de registres à savoir les registres de travail (µP 8086 : AX, BX, CX, DX), les registres de segmentation (µP 8086 : CS, DS, SS et ES), etc.

I.5.3. Les bus internes

Ce sont des liaisons électriques, sous forme de pistes de circuits, qui relient le microprocesseur avec son environnement interne (Mémoire centrale, unité de commande et unité de traitement) ainsi que son environnement externe, à savoir les périphériques (à partir des interfaces E/S) et les mémoires. Il existe trois bus :

I.5.3.a. Le bus de donnée :

Il assure le transfert des informations entre le microprocesseur et son environnement, et inversement. Son nombre de lignes est égal au format des mots de données du microprocesseur (bus bidirectionnel).

I.5.3.b. Le bus d'adresses (bus d'adressage) :

Il transporte des adresses mémoires (bus unidirectionnel). Il permet la sélection des informations à traiter dans un espace mémoire (ou espace adressable) qui peut avoir 2_n emplacements, avec n = nombre de conducteurs du bus d'adresses.

I.5.3.c. Le bus des commandes (bus de contrôle) :

Il transporte des commandes provenant du microprocesseur vers les divers composants matériels (bus bidirectionnel car il récupère des accusés de réception).

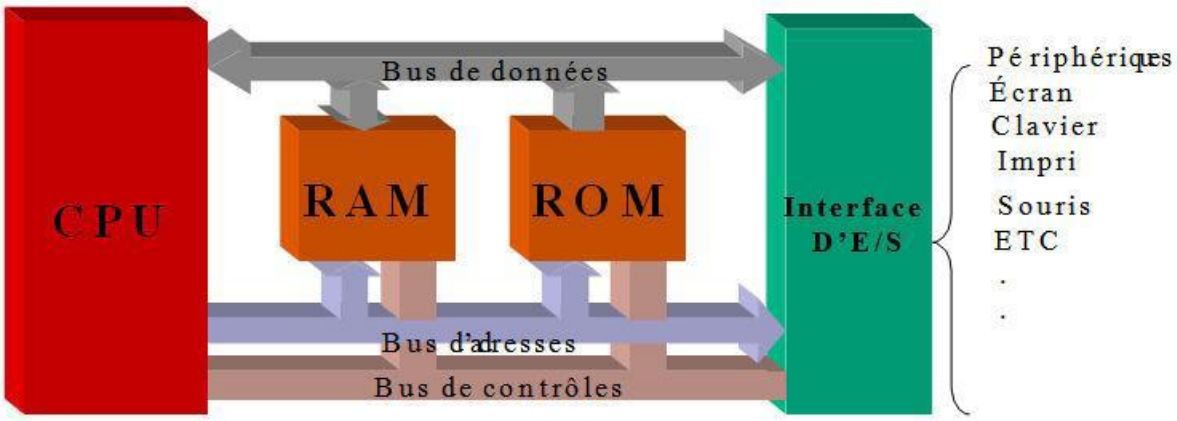


Figure I-6: Différents bus dans un microprocesseur.

I.6. Jeu d'instructions

I.6.1. Définition

La première étape de la conception d'un microprocesseur est la définition de son jeu d'instructions. Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur pourra exécuter. Il va donc en partie déterminer l'architecture du microprocesseur à réaliser et notamment celle du séquenceur. A un même jeu d'instructions peut correspondre un grand nombre d'implémentations différentes du microprocesseur.

I.6.2. Type d'instructions

Les instructions que l'on retrouve dans chaque microprocesseur peuvent être classées en 4 groupes :

- ❖ Transfert de données pour charger ou sauver en mémoire, effectuer des transferts de registre à registre, etc…
- ❖ Opérations arithmétiques : addition, soustraction, division, multiplication
- ❖ Opérations logiques : ET, OU, NON, NAND, comparaison, test, etc…
- ❖ Contrôle de séquence : branchement, test, etc…

I.6.3. Codage

Les instructions et leurs opérandes (paramètres) sont stockés en mémoire principale. La taille totale d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type d'instruction et aussi du type d'opérande. Chaque instruction est toujours codée sur un nombre entier d'octets afin de faciliter son décodage par le processeur. Une instruction est composée de deux champs:

- ✓ Le code instruction, qui indique au processeur quelle instruction réaliser
- ✓ Le champ opérande qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).

Exemple :

Code instruction	Code opérande
1001 0011	0011 1110

Le nombre d'instructions du jeu d'instructions est directement lié au format du code instruction. Ainsi un octet permet de distinguer au maximum 256 instructions différentes.

Jeu d'instructions simulées en assembleur				
Langage machine	Assembleur		Mnémonique	Explication
	Op code	opérande		
00	**NOP**		*No operation*	Ne rien faire
01	**LDA**	adr	*Load A from adr*	A ← contenu de la cellule **adr**
02	**STA**	adr	*Store A in adr*	Contenu la cellule **adr** ← A
03	**MOV A,B**		*Move A in B*	B ← A
04	**MOV B,A**		*Move B in A*	A ← B
05	**ADD A**		*Add A to A*	A ← A + A
06	**ADD B**		*Add B to A*	A ← A + B
07	**INC A**		*Increment A*	A ← A + 1
08	**INC B**		*Increment B*	B ← B + 1
09	**DCR A**		*Decrement A*	A ← A - 1
10	**DCR B**		*Decrement B*	B ← B - 1
11	**SUB B**		*SUB B from A*	A ← A - B
12	**MVI A**	val	*Move immediate Value*	A ← val
13	**MVI B**	val	*Move immediate Value*	B ← val
14	**JMP**	adr	*Jump*	IP ← adr
15	**JP**	adr	*Jump on positive*	Si positif alors (IP ← adr) sinon ne rien faire
16	**JN**	adr	*Jump on negative*	Si négatif alors (IP ← adr) sinon ne rien faire
17	**JZ**	adr	*Jump on zero*	Si Z=1 alors (IP ← adr) sinon ne rien faire
18	**JNZ**	adr	*Jump on non zero*	Si Z=0 alors (IP ← adr) sinon ne rien faire

Figure I-7: Quelques jeu d'instructions

I.7. Définition d'un microcontrôleur

Un microcontrôleur est un système qui ressemble à un ordinateur : il a une mémoire, un processeur, des interfaces avec le monde extérieur. Les microcontrôleurs ont des performances réduites, mais sont de faible taille et consomment peu d'énergie, les rendant indispensables dans toute solution d'électronique embarquée. Le microcontrôleur est une unité de traitement de l'information de type microprocesseur à la quelle on a ajouté un interfaçage avec le monde extérieur sans nécessiter l'ajout

des composants à jeu d'instructions réduits. Ils se caractérisent essentiellement par un plus haut degré d'intégration, une plus faible consommation d'énergie électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels. Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateurr dont : processeur (CPU), mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'entrées-sorties.

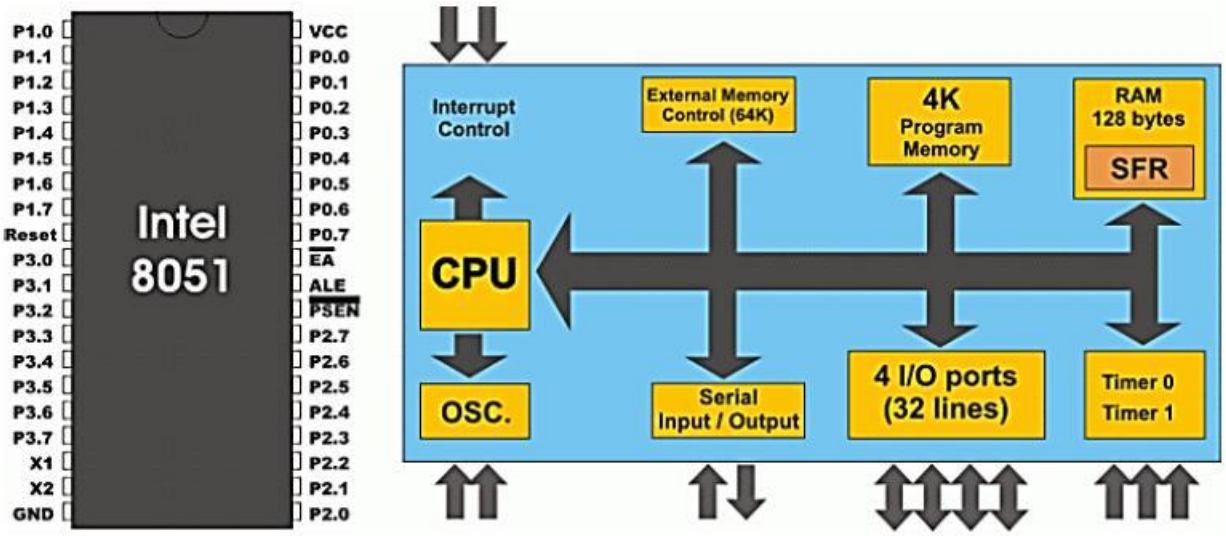


Figure I-8: Les éléments clés d'un microcontrôleur.

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

I.8. Caractéristiques essentielles des microcontrôleurs

Le microcontrôleur est plus adapté aux applications embarquées, par contre il est généralement moins puissant en termes de rapidité ou de taille de mémoire adressable et le plus souvent cantonné aux données de 8 ou 16 bits, mais il existe des microcontrôleurs 32 bits pour certaines applications qui le nécessitent. 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. Les microcontrôleurs 32 bits présentent les avantages suivants faces au microcontrôleur 8 bits. Ses caractéristiques principales sont :

- Séparation des mémoires de programme et de données (architecture Harvard) ;
- Communication avec l'extérieur seulement par des ports ;
- Utilisation d'un jeu d'instructions réduit, d'où le nom de son architecture : RISC (Reduced

Instructions Set Construction). Les instructions sont ainsi codées sur un nombre réduit de bits, ce qui accélère l'exécution (1 cycle machine par instruction sauf pour les sauts qui requirent 2 cycles).

En revanche, leur nombre limité oblige à se restreindre à des instructions basiques, contrairement aux systèmes d'architecture CISC (Complex Instructions Set Construction) qui proposent plus d'instructions donc codées sur plus de bits mais réalisant des traitements plus complexes.

Il existe trois familles de PIC : -Base-Line : Les instructions sont codées sur 12 bits, -Mid-Line : Les instructions sont codées sur 14 bits et, -High-End : Les instructions sont codées sur 16 bits.

Un PIC est identifié par un numéro de la forme suivant : **xx (L) XX yy – zz**

- **xx** : Famille du composant (12, 14, 16, 17, 18)
- **L** : Tolérance plus importante de la plage de tension,
- **XX** : Type de mémoire de programme. XX prend la lettre C pour une EPROM ou EEPROM, les lettres CR pour une PROM et F pour une mémoire de type Flash.
- **yy** : Identification **,**
- **zz** : Vitesse quartz de cadencement du microcontrôleur.

A titre d'exemple, un PIC 16F84 –10, s'identifie comme : 16 signifie que le PIC est de la famille Mid-Line ; F le PIC a une mémoire programme de type FLASH ; son type est 84 ainsi que 10 veut dire utilisation d'un Quartz à 10 MHz au maximum.

I.8.1. Architecture des microcontrôleurs

Les microcontrôleurs comme les microprocesseurs sont formés des modules suivants :

- ✓ **Les bus** assurent la communication (série ou parallèle) du microcontrôleur avec les autres composants. Il s'agit de :
- **Bus d'adresse :** permettant au microprocesseur de choisir la case mémoire ou le périphérique auquel il veut accéder pour lire ou écrire une information (instruction ou donnée) ;
- **Bus de données** permettant le transfert des informations entre les différents éléments ;

- **Bus de contrôle** indiquant la nature de l'opération en cours ou si un périphérique nécessite une interruption envoyer une information au processeur, …

✓ **L'unité centrale (CPU)** exécute les instructions et décide du fonctionnement du programme ;

✓ **Les mémoires (EPROM et RAM)** conservent les instructions ainsi que les données ;

✓ **Les entrées /sorties (E/S)** servent d'interface de communication entre le processeur et les périphériques.

Selon leurs architectures internes on distinguera les microcontrôleurs à architecture Harvard de ceux à architecture Von-Neumann. Mentionnons, tout de même que la quasi-totalité des microcontrôleurs actuels à l'instar des processeurs d'ordinateurs utilisent, en interne, la deuxième architecture. L'architecture Von-Neumann admet une mémoire de programme renfermant les instructions et données. Ici, on dispose que du bus de données véhiculant tour à tour les codes des instructions et qui ne sont que des données. L'architecture Von-Neumann pose problème en cas de fonctionnement rapide.

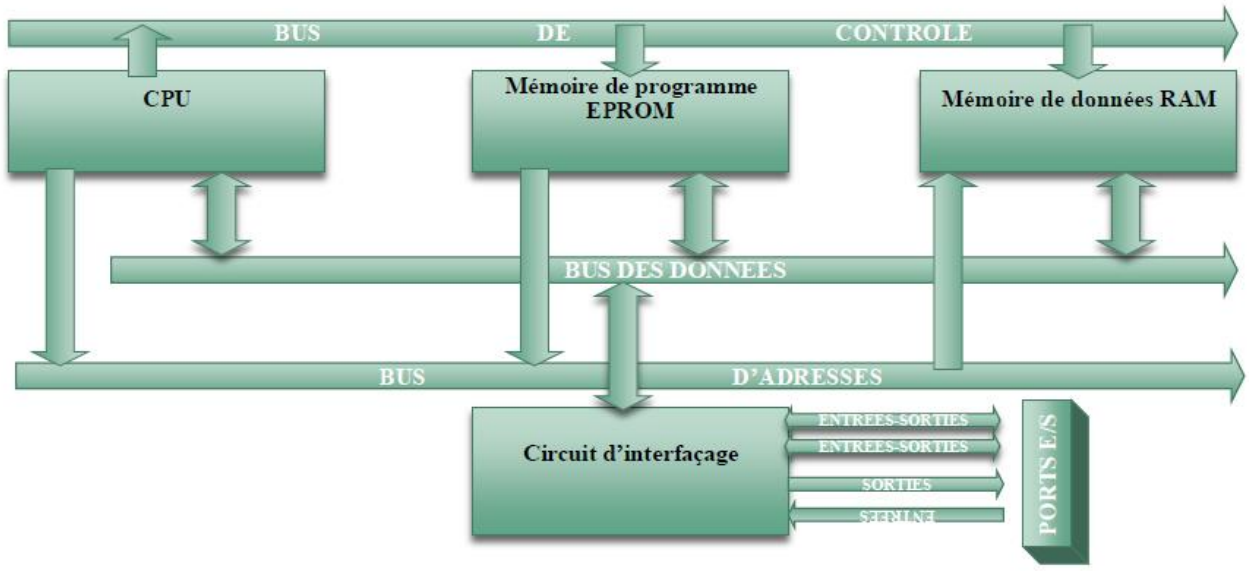


Figure I-9: Constitution d'un système à microcontrôleur avec différents bus.

II. Partie 2 : Automates Programmables Industriels (API)

Dans cette partie, on va commencer par une introduction aux automates programmables industriels (API), puis nous aborderons les points suivants :

- Architecture interne et description des éléments d'un API ;
- Choix de l'unité de traitement ;
- Choix d'un API ;
- Câblages des entrées et sortie d'un API ;
- Langage de programmation d'un API ;
- Applications

II.1. Introduction

Les Automates Programmables Industriels (API) 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 est une machine électronique programmable et destiné à piloter en ambiance industrielle et en temps réel des procédés industriels. Un automate programmable est adaptable à un maximum d'application, d'un point de vue traitement, composants, langage.

Un automate programmable industriel est une forme particulière de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker les instructions et qui implémente différente fonction, qu'elles soient logiques de séquencement, de temporisation de comptage ou arithmétique pour commander les machines et les processus.

Les API sont pour les taches de commande dans les environnements industriels, voici ce qui caractérise les API :

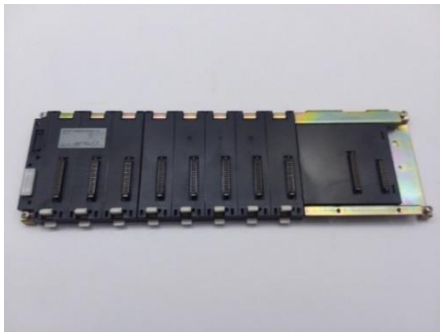
- Ils sont solides et conçus pour supporter les vibrations, les températures basses ou élevées, l'humidité et le bruit.
- Les interfaces des entrés et des sorties sont intégrées à l'automate.
- Ils sont faciles à programmer et leurs langages de programmation sont facile à comprendre est principalement orienté sur les opérations logiques et de communication.

II.2. Architecture des APIs :

II.2.1. Structure générale des API :

Les caractéristiques principales d'un automate programmable industriel sont :

- Coffret, rack ou cartes



- Compact ou modulaire

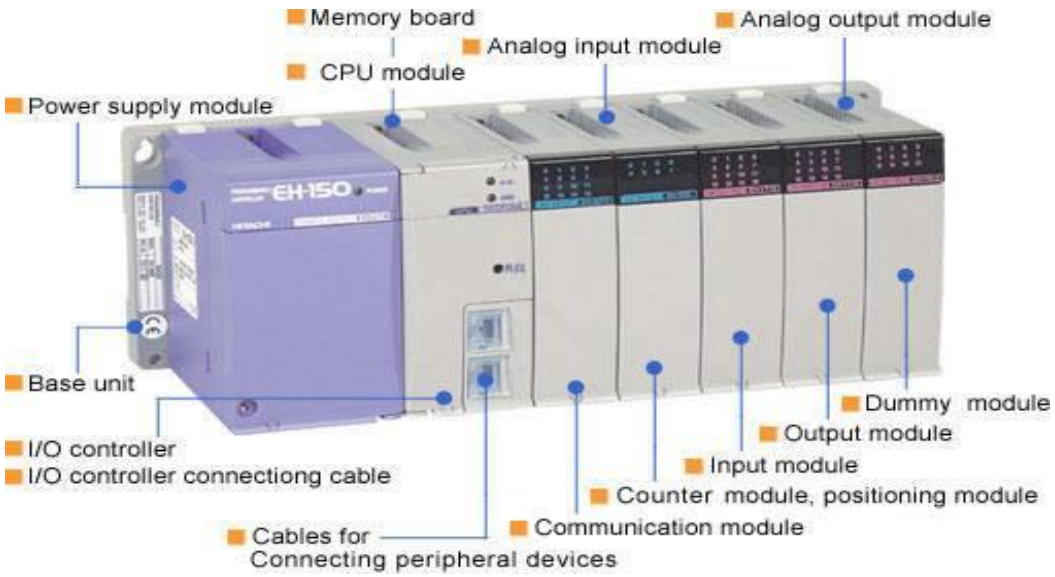


Figure II-1: API Compact et modulaire

- Tension d'alimentation
- Taille mémoire
- Sauvegarde (EPROM, EEPROM, pile, …)
- Nombre d'entrées / sorties
- Modules complémentaires (analogique, communication, ...)
- Langage de programmation.

Les API en boîtier étanche sont utilisés pour les ambiances difficiles (température, poussière, risque de projection ...) supportant ainsi une large gamme de température, humidité ... L'environnement industriel se présentent sous trois formes :

- Environnement physique et mécanique (poussières, température, humidité, vibrations);
- Pollution chimique ;
- Perturbation électrique. (Parasites électromagnétiques).

II.2.2. Architecture interne d'un automate programmable industriel (API) :

Les API comportent quatre principales parties (Figure II-2) :

- Une unité de traitement (un processeur CPU) ;
- Une mémoire ;
- Des modules d'entrées-sorties ;
- Des interfaces d'entrées-sorties ;
- Une alimentation 230 V, 50/60 Hz (AC) - 24 V (DC).

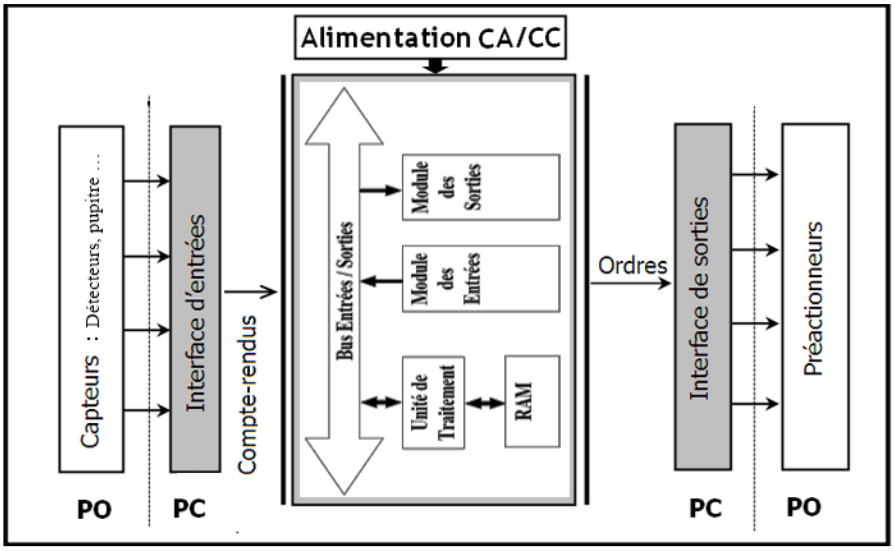


Figure II-2: Architecture d'un API

La structure interne d'un API est assez voisine de celle d'un système informatique simple, L'unité centrale est le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programme. Les instructions sont effectuées les unes après les autres, séquencées par une horloge. Deux types de mémoire cohabitent :

- La mémoire Programme où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement. (ROM : mémoire morte) ;
- La mémoire de données utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive). Elle fait partie du système entrées-sorties. Elle fige les valeurs (0 ou 1) présentes sur les lignes d'entrées, à chaque prise en compte cyclique de celle-ci, elle mémorise les valeurs calculées à placer sur les sorties.

II.2.3. Description des éléments d'un API

II.2.3.a. Le processeur

Le processeur, ou unité centrale (UC), a pour rôle principal le traitement des instructions qui constituent le programme de fonctionnement de l'application (les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul PID, etc..). Mais en dehors de cette tâche de base, il réalise également d'autres fonctions :

- Gestion des entrées/sorties.
- Surveillance et diagnostic de l'automate par une série de tests lancés à la mise sous tension ou cycliquement en cours de fonctionnement.
- Dialogue avec le terminal de programmation, aussi bien pour l'écriture et la mise au point du programme qu'en cours d'exploitation pour des réglages ou des vérifications des données.

Un ou plusieurs processeurs exécutent ces fonctions grâce à un micro logiciel préprogrammé dans une mémoire de commande, ou mémoire système. Cette mémoire morte définit les fonctionnalités de l'automate. Elle n'est pas accessible à l'utilisateur.

II.2.3.b. La mémoire

Elle est destinée au stockage des instructions qui constituent le programme de fonctionnement de l'automatisme, ainsi que des données qui peuvent être :

- Des informations susceptibles d'évoluer en cours de fonctionnement de l'application. C'est le cas par exemple de résultats de traitements effectués par le processeur et rangés dans l'attente d'une utilisation ultérieure. Ces données sont appelées variables internes ou mots internes.
- Des informations qui n'évoluent pas au cours de fonctionnement, mais qui peuvent en cas de besoin être modifiées par l'utilisateur : textes à afficher, valeurs de présélection, etc.. Ce sont des mots constants.
- Les mémoires d'état des entrées/sorties, mises à jour par le processeur à chaque tour de scrutation du programme.

Deux familles de mémoires sont utilisées dans les automates programmables :

- Les mémoires vives, ou mémoires à accès aléatoire « Random Access Memory (RAM) ». Le contenu de ces mémoires peut être lu et modifié à volonté, mais il est perdu en cas de manque de tension (mémoire volatiles). Elles nécessitent par conséquent une sauvegarde par batterie. Les mémoires vives sont utilisées pour l'écriture et la mise au point du programme, et pour le stockage des données. Elles sont à lecture seule, les informations ne sont pas perdues lors de la coupure de l'alimentation des circuits. On peut citer les types suivants :

- **ROM « Read Only Memory » :** Elle est programmée par le constructeur et son programme ne peut être modifié.
- **PROM « Programmable ROM » :** Elle est livrée non enregistrée par le fabricant. Lorsque celle-ci est programmée, on ne peut pas l'effacer
- **EPROM « Erasable PROM » :** C'est une mémoire PROM effaçable par un rayonnement ultraviolet intense.
- **EEPROM « Electrically EPROM » :** C'est une mémoire PROM programmable plusieurs fois et effaçable électriquement.
- **Mémoire Flash :** C'est une mémoire EEPROM rapide en programmation. L'utilisateur peut effacer un bloc de cases ou toute la mémoire.

La mémoire morte est destinée à la mémorisation du programme après la phase de mise au point. La mémoire programme est contenue dans une ou plusieurs cartouches qui viennent s'insérer sur le module processeur ou sur un module d'extension mémoire.

II.2.3.c. Les interfaces entrées/sorties :

Les entrées/sorties TOR (Tout ou Rien) assurent l'intégration directe de l'automate dans son environnement industriel en réalisant la liaison entre le processeur et le processus. Elles ont toutes, de base, une double fonction :

*Une fonction d'interface pour la réception et la mise en forme de signaux provenant de l'extérieur (capteurs, boutons poussoirs, etc.) et pour l'émission de signaux vers l'extérieur (commande de pré-actionneurs, de voyants de signalisation, etc.). La conception de ces interfaces avec un isolement galvanique ou un découplage opto-électronique assure la protection de l'automate contre les signaux parasites.

*Une fonction de communication pour l'échange des signaux avec l'unité centrale par l'intermédiaire du bus d'entrées/sorties.

Le fonctionnement de l'interface d'entrée (figure 3) peut être résumé comme suit :

Lors de la fermeture du capteur ;

- La « **Led 1** » signale que l'entrée de l'API est actionnée.
- La « **Led D'**» de l'optocoupleur « **Opto 1** » s'éclaire.
- Le phototransistor « **T'**» de l'optocoupleur « **Opto 1** » devient passant.
- La tension Vs=0V.

Donc lors de l'activation d'une entrée de l'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).

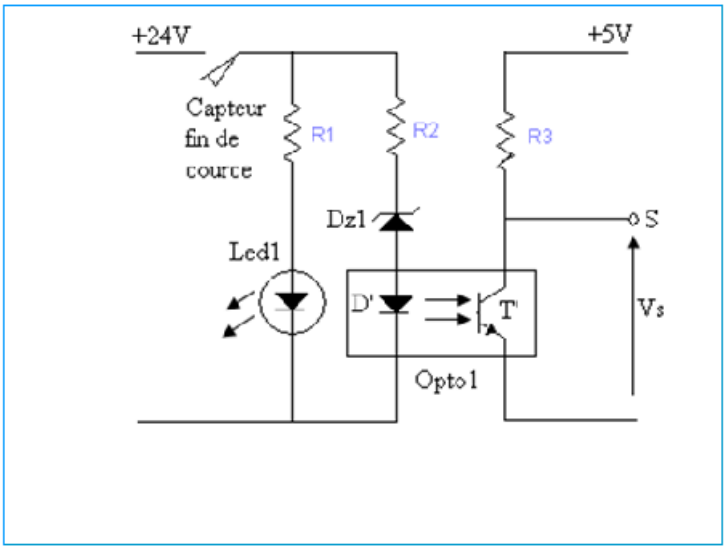


Figure II-3: Principe de fonctionnement de l'interface d'entrée

Le fonctionnement de l'interface de sortie (figure 4) peut être résumé comme suit :

Lors de commande d'une sortie automate ;

- L'unité de commande envoi un « **1** » logique (5V).
- « **T1** » devient passant, donc la « **Led D'** » s'éclaire
- Le photo-transistor « **T'** » de l'optocoupleur « **Opto1** » devient passant.
- La « **Led1** » s'éclaire.
- « **T2** » devient passant.
- La bobine « **RL1** » devient sous tension et commande la fermeture du contact de la sortie « **Q0.1** ».

Donc pour commander un API, 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

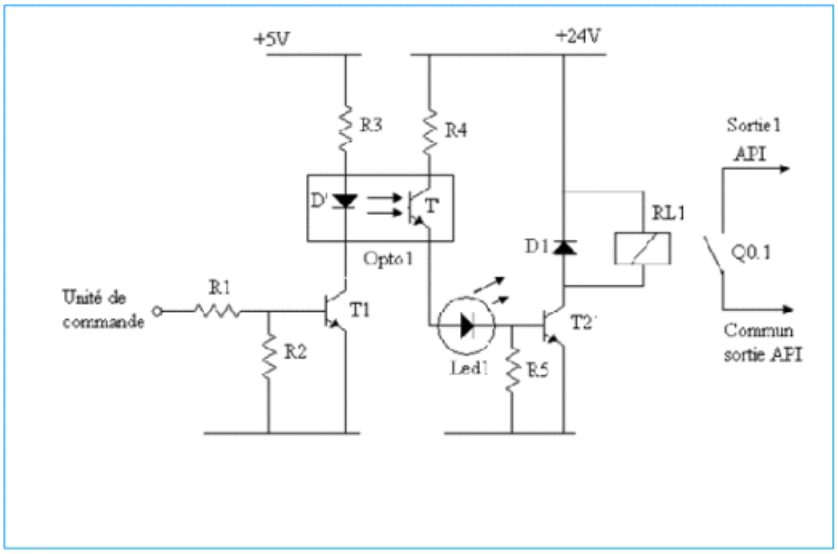


Figure II-4: Principe de fonctionnement de l'interface de sortie

La figure 5 donne une idée concrète sur un module TOR industriel.

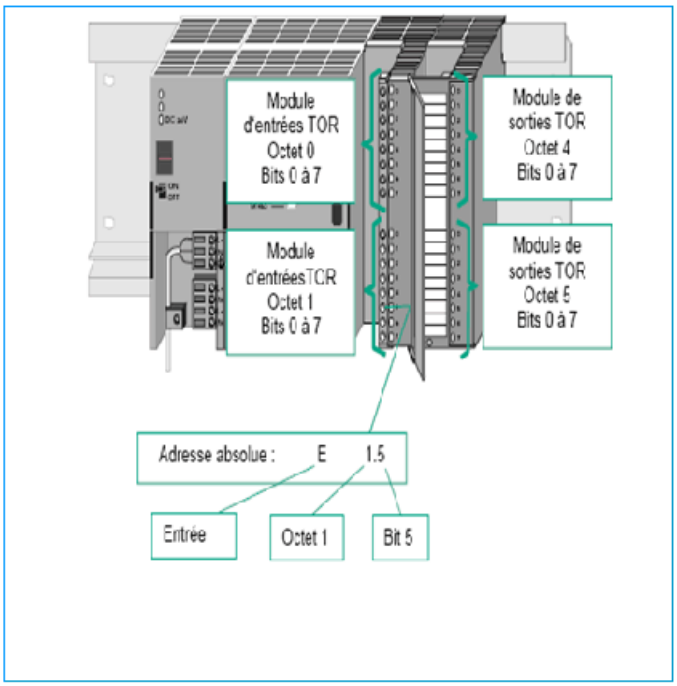


Figure II-5: Schéma de principe d'un module TOR

II.2.3.d. Le Bus

C'est un ensemble de conducteurs qui réalisent la liaison entre les différents éléments de l'automate. Dans un automate modulaire, il se présente sous forme d'un circuit imprimé situé au fond du bac et supporte des connecteurs sur lesquels viennent s'enficher les différents modules : processeur, extension mémoire, interfaces et coupleurs.

Le bus est organisé en plusieurs sous ensembles destinés chacun à véhiculer un type bien défini d'informations :

- Bus de données.
- Bus d'adresses.
- Bus de contrôle pour les signaux de service tels que tops de synchronisation, sens des échanges, contrôle de validité des échanges, etc.

❖ Bus de distribution des tensions issues du bloc d'alimentation.

II.2.3.e. Alimentation

Elle élabore à partir d'un réseau 220V en courant alternatif, ou d'une source 24V en courant continu, les tensions internes distribuées aux modules de l'automate.

A fin d'assurer le niveau de sûreté requis, elle comporte des dispositifs de détection de baisse ou de coupure de la tension réseau, et de surveillance des tensions internes. En cas de défaut, ces dispositifs peuvent lancer une procédure prioritaire de sauvegarde.

Les schémas de principe et raccordement sont, respectivement, illustrés par la figure II-6.

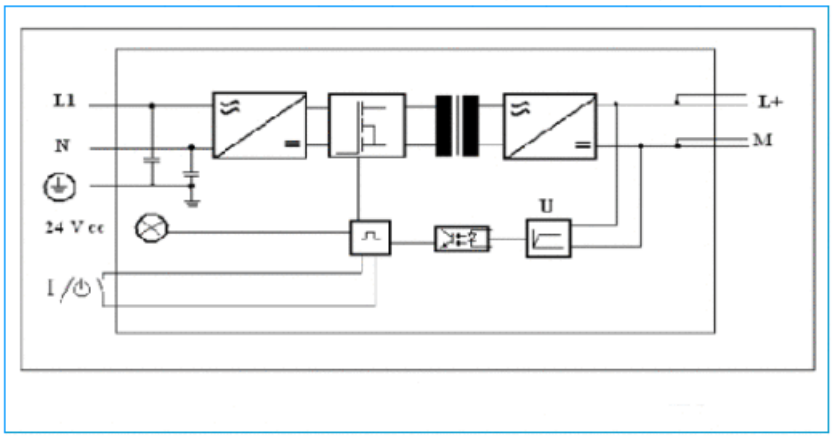


Figure II-6: Schéma de principe d'une alimentation d'un API.

II.2.4. Principaux automates programmables industriels :

La programmation de ces automates se fait soit à partir de leur propre console, soit à partir du logiciel de programmation propre à la marque.

II.2.4.a. OMRON :

- **CQM1 – CPU 11/21/41**

- E - 192 Entrées/Sorties (à relais, à triac, à transistors ou TTL) ;
- 32 K RAM data on Board;
- Structure multifonction ;
- Structuration multitâche ;
- SYSWIN 3.1, 3.2 … 3.4 et CX Programmer (Littéral, Ladder) ;
- Communication sur RS 232 – C ;
- Programmation sur IBM PC/PS.

II.2.4.b. TELEMECANIQUE :

- **TSX 17/20 :**

- Nombre d'entrées et de sorties variable : 20 à 160 E/S.
- Microprocesseur 8031.
- Langage de programmation PL7.2.

- **TSX 67.20 :** La compacité d'un automate haut de gamme, à E/S déportables par fibre optique :

- 1024 E/S en six bacs de huit modules ;
- Extension de bacs à distance par fibre optique à 2000 m ;
- 16 coupleurs intelligents ;
- 24 K RAM data on Board;
- 32 K RAM / EPROM cartouche utilisateur ;
- Structure multifonctions ;
- Structuration multitâche ;
- Langage PL7.3 (Grafcet, Littéral, Ladder);
- Programmation sur IBM PC/PS.

II.2.4.c. FESTO :

Architecture modulaire : carte de base ; carte processeur ; carte de mémorisation ; carte E/S.

- FPC 202 :
- 16 entrées 24 V DC ;

- 16 sorties 24 V DC - 1 A ;
- 8 K RAM, 8 K EPROM ;
- Interface série, 20 mA boucle de courant pour imprimante ;
- Console de programmation externe : console ou IBM PC ;
- Programmation : grafcet, langage Festo, schéma à relais.

II.2.4.d. SIEMENS :

- **S7 – 200.**

- 64 entrées 24 V DC ;
- 64 sorties 24 V DC - 1 A ;
- 8 Entrées analogiques AEW0
- AEW14 ; - 8 Sorties analogiques AAW0
- AAW6 ; - interface série,
- Console de programmation externe : PG 702 ;
- Programmation STEP7 : schéma à relais, Ladder.

II.3. Choix et câblages des APIs

II.3.1. Critères de choix :

Le point de départ dans le choix d'un automate programmable est en premier lieu le choix d'un groupe de contacts commerciaux (**sociétés ou firmes)** où l'expérience vécue avec est certaine. IL est souhaitable de designer deux, ou plus, de fabricants pour faire jouer la concurrence entre eux.

Un automate utilisant des langages de programmation de type GRAFCET est également préférable pour assurer les mises au point et dépannages dans les meilleures conditions.

Il faut ensuite quantifier les besoins :

- **Le premier paramètre** à prendre en compte pour choisir un automate est le nombre d'entrée et de sortie nécessaire. Il pourra y avoir un bloc de base et des extensions, ou une unité centrale et des cartes d'entrée ou de sortie. On commencera donc par faire le bilan des entrées et des sorties.
- **Amplitude** des entrées/sorties,
- **Type des entrées/sorties** Les entrées et les sorties peuvent être :
  - **i) Logique** : entrées et sorties tout ou rien,

ii) **Analogique** : liaison avec génératrice tachymétrique en entrée et variateur de vitesse en sortie par exemple.

iii) **Numérique** : comptage rapide sur un codeur incrémental. Chaque entrée ou sortie devra être adaptée au capteur ou au **pré-actionneur.**

- Les cartes assurent **l'isolation galvanique** entre l'unité centrale et le système.
- **Les cartes de sortie** peuvent être à **relais** ou à **transistor**. Celles à relais assurent une coupure entre l'alimentation et le pré-actionneur mais sont relativement lentes. Celles à transistor commutent plus rapidement mais n'assurent pas de séparation électrique.
- **Unité centrale :** C'est le cœur de l'automate. Elle comporte un microprocesseur et de la mémoire qui permettent de définir sa puissance. La capacité mémoire de certain automate peut être augmentée.
- Type de processeur : la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.

Nombre de compteurs.

- Nombre de temporisateurs
- Fonctions de communication : l'automate doit pouvoir communiquer avec les autres systèmes de commande (API, supervision ...) et offrir des possibilités de communication avec des standards normalisés (Profibus ...).
- **Alimentation :** Elle doit couvrir les besoins énergétiques de l'unité centrale et de toutes les extensions. Quand elle existe sur l'automate de base, elle ne couvre pas les besoins d'un nombre important d'extension et il faudra rajouter une deuxième alimentation.

II.3.2. Câblage des entrées/sorties d'un automate :

L'automate est alimenté généralement par le réseau monophasé 230V/50Hz mais d'autres alimentations sont possibles (110V,24vdc, …, etc.).

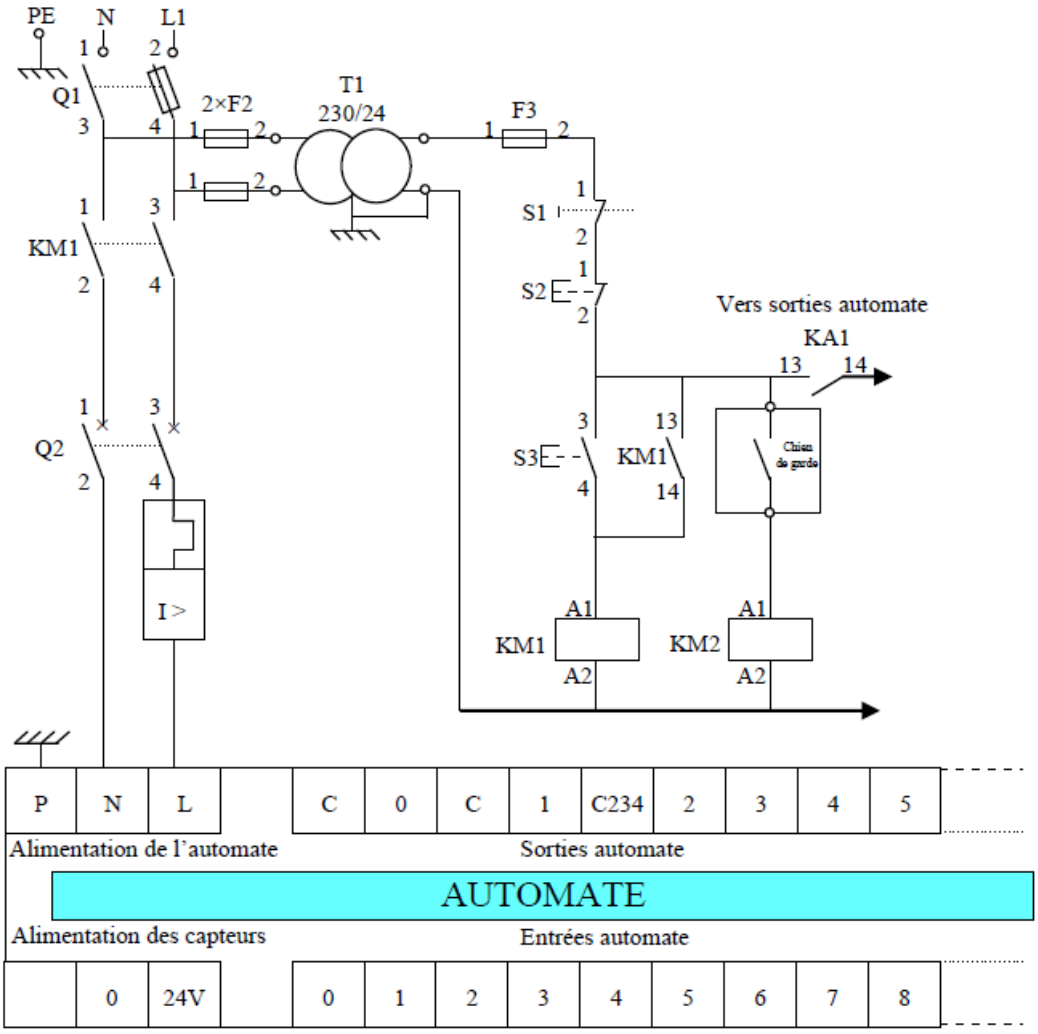


Figure II-7: Raccordement d'un automate.

II.3.2.a. Alimentation des entrées de l'automate :

L'automate est pourvu généralement d'une alimentation pour les capteurs/détecteurs (attention au type de logique utilisée : logique positive ou négative).

Les entrées sont connectées au OV (commun) de cette alimentation. Les informations des capteurs/détecteurs sont traitées par les interfaces d'entrées.

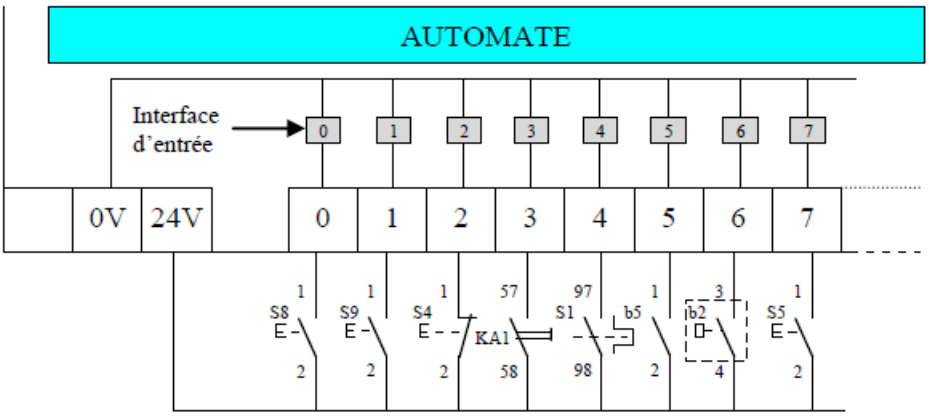


Figure II-8: Câblage des entrées de l'automate.

II.3.2.b. Alimentation des sorties de l'automate :

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

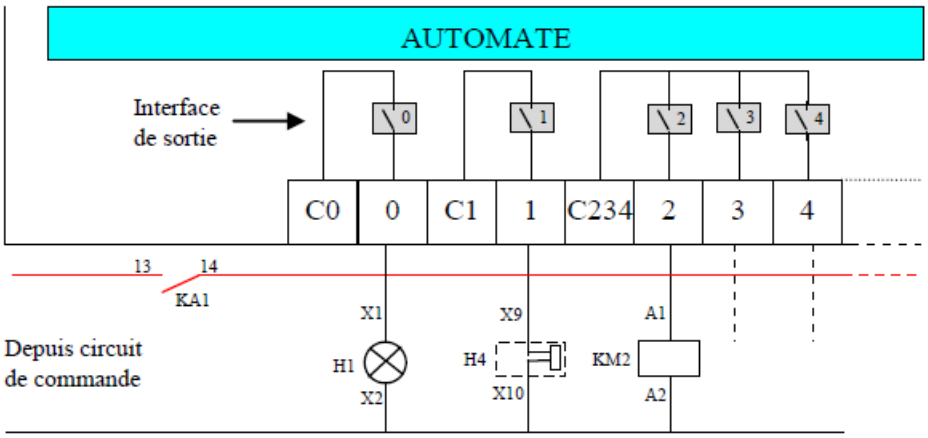


Figure II-9: Câblage des sorties de l'automate.

II.3.2.c. Traitement du programme automate :

Tous les automates fonctionnent selon le même mode opératoire :

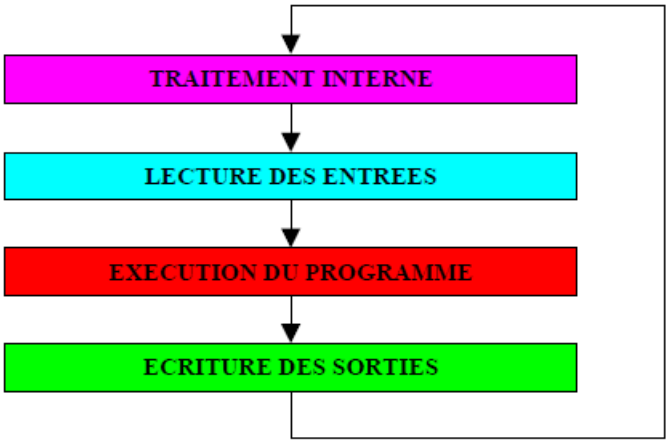


Figure II-10: Cycle de fonctionnement d'un API.

- 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 lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées.
- Exécution du programme : L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.
- Ecriture des sorties : L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties.

Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique).

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 est de l'ordre de la dizaine de millisecondes pour les applications standards.

Le temps de réponse total (TRT) est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante :

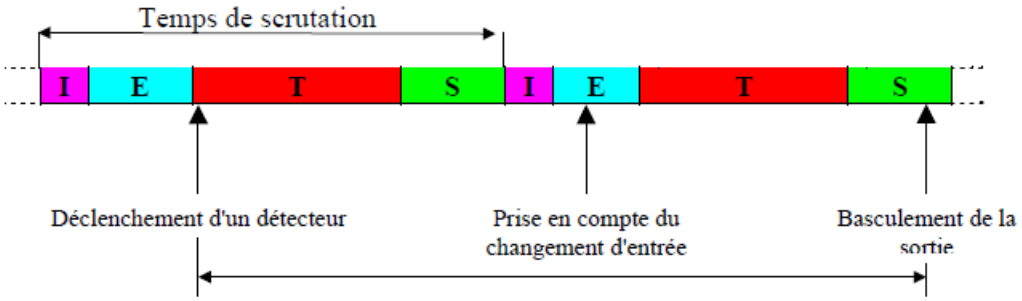


Figure II-11: Temps de réponse total

Le temps de réponse total est au plus égal à deux fois le temps de scrutation (sans traitement particulier).

Le temps de scrutation est directement lié au programme implanté. Ce temps peut être fixé à une valeur précise (fonctionnement périodique), le système indiquera alors tout dépassement de période.

Dans certains cas, on ne peut admettre un temps de réponse aussi long pour certaines entrées : ces entrées pourront alors être traitées par l'automate comme des événements (traitement événementiel) et prises en compte ***en priorité*** (exemples : problème de sécurité, coupure d'alimentation ...).

Certains automates sont également pourvus d'entrées rapides qui sont prises en compte avant le traitement séquentiel mais le traitement évènementiel reste prioritaire.

II.4. Langages de Programmation

II.4.1. Logique câblée et programmée

II.4.1.a. 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.

II.4.1.b. 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 puisqu'ils ont 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 et
- Réduire les coûts de maintenance.

II.4.2. Besoin de la normalisation

Les programmes utilisés avec les API peuvent être écrits dans différents formats. La plupart des fabricants d'automates ont adopté le langage à contacts pour l'écriture des programmes. Toutefois, puisque chacun a eu tendance à développer ses propres versions, une norme internationale a été établie pour le langage à contact et, par voie de conséquence, pour toutes les méthodes de programmation

employées avec les API. Cette norme, publiée en 1993 par la Commission Electrotechnique internationale (CEI), est désignée par la référence CEI 61131-3. La dernière version, qui date de 2013, est une extension qui reste compatible avec la version antérieure.

Les langages de programmation définis par la norme CEI 61131-3 sont le langage à contacts (LAD, Ladder Diagram), les listes d'instruction (IL, Instruction List), les graphes de fonction séquentielle (SFC, Sequential Function Charts), le texte structuré (ST, Structured Text) et les diagrammes de schémas fonctionnels (FBD, Function Block Diagram). La norme comprend une bibliothèque de fonctions préprogrammées et des bloc fonctionnels. La norme CEI donne une définition formelle de chaque paramètre d'entrée et de sortie afin que des blocs fonctionnels conçus par différents programmeurs puissent être facilement interconnectés. N'importe quel API conforme à la norme CEI prend en charge ces fonctions sous forme d'une bibliothèque, le code étant écrit dans la partie PROM d'une mémoire flash de l'appareil.

Parmi ces langages, deux sont textuel, le texte structuré et les listes d'instructions, seront saisis sur l'appareil de programmation à partir d'un clavier, une ligne à la fois. Les autres langages, le langage à contact, les graphes de fonction séquentielle et les diagrammes de schémas fonctionnels, sont graphiques et un programme peut donc être développé à partir d'éléments graphiques sur l'écran de l'appareil de programmation.

II.4.3. La norme CEI 61131

La norme CEI 31131 couvre l'intégralité du cycle de vie des API :

- Partie 1 : Définition générale de la terminologie et des concepts.
- Partie 2 : Exigences sur le matériel électronique et mécanique et tests de contrôle des API et des équipements associés.
- Partie 3 : Langages de programmation.
- Partie 4 : Conseil de sélection, d'installation et de maintenance des API.
- Partie 5 : fonctions logicielles pour la communication avec d'autres appareils selon la norme MMS (Manufacturing Messaging Specification).
- Partie 6 : Communications via les fonctions logicielles de bus de terrain.
- Partie 7 : Programmation par la logique floue.
- Partie 8 : Conseils d'implémentation des langages de programmation.

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).

II.5. Programmer les API

La norme IEC 1131-3 définit entre autres choses, cinq langages qui peuvent être utilisés pour la programmation d'applications d'automatisme. Les cinq langages sont :

- SFC (« sequential function char ») : issu du langage GRAFCET, ce langage, de haut niveau, permet la programmation aisée de tous les procédés séquentiels ;
- 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 ;
- LD (« ladder diagram », ou schéma à relais) : ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (true/false) ;
- 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 ;
- 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.

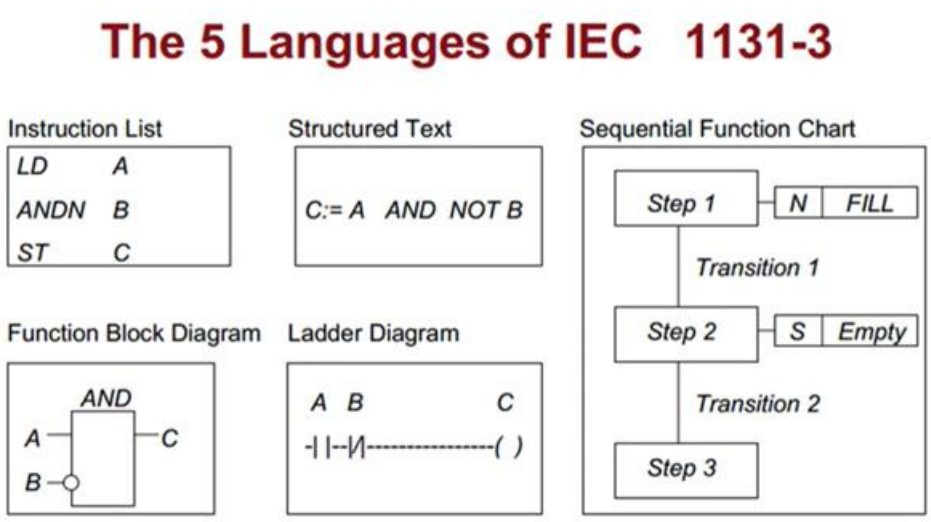


Figure II-12: Les cinq langages de programmation

II.5.1. Les langages graphiques :

II.5.1.a. Langage Ladder :

Le langage LD (ladder diagram) est une représentation graphique d'équations booléennes combinant des contacts (les entrée) et des relais (les sortie). Il permet la manipulation de données booléennes, à l'aide de symboles graphiques organisés dans un diagramme comme les éléments d'un schéma électrique à contacts.

L'idée initiale du Ladder est la représentation de fonction logique sous la forme de schémas électriques. Cette représentation est originalement matérielle : quand l'Automate Programmable Industriel n'existait pas, les fonctions étaient réalisées par des câblages. Par exemple, pour réaliser un ET logique avec des interrupteurs, il suffit de les mettre en série. Pour réaliser un OU logique, il faut les mettre en parallèle. Partant de ces principes, le Ladder a été créé et normalisé dans la norme IEC 1131-3. Il est, depuis, très utilisé dans la programmation des Automates Programmables Industriels. Un programme Ladder se lit de haut en bas et l'évaluation des valeurs se fait de gauche à droite.

Les valeurs correspondent en fait, si on le compare à un schéma électrique, à la présence ou non d'un potentiel électrique a chaque nœud de connexion. En effet, le Ladder est basé sur le principe d'une alimentation en tension représentée par deux traits verticaux reliée horizontalement par des bobines, des contacts et des blocs fonctionnels. Il est composé de réseaux lus les uns à la suite des autres par l'automate. Ces réseaux sont constitués de divers symboles représentant les entrées/sorties de

l'automate, les opérateurs séquentiels (temporisations, compteurs, …), les opérations, ainsi que les bits systèmes internes à l'automate

Les composants du langage

Il existe 3 types d'élément de langage :

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

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

Object graphique API	Nom
-| |-	Contact normalement ouvert
-|/|-	Contact normalement fermé
-|P|-	Contact fermé au front montant
-|N|-	Contact fermé au front descendant
-( )-	Bobine normalement ouverte
-(/)-	Bobine normalement fermée
-(S)- ou–(L)-	Le bit interne maintenu à 1 une fois actionné
-(R)-ou –(U)-	Le bit interne remise à 0 une fois actionné

Tableau 1: les principaux éléments (contacte et bobines) d'un réseau LD.

- Les blocs fonctionnels qui permettent de réaliser des fonctions avancées (temporisation, comptage, etc.).

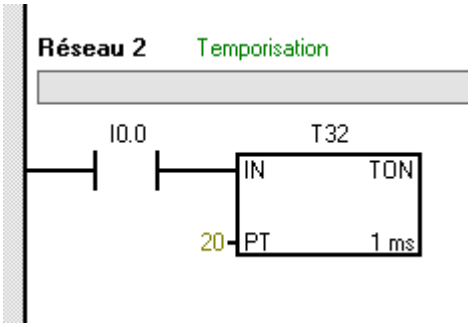


Figure II-13: Temporisation

L'opération Démarrer temporisation sous forme de retard à la montée s'écoule jusqu'à la valeur maximale lorsqu'elle est activée. Lorsque la valeur en cours « T » est supérieure ou égale à la valeur prédéfinie PT, le bit de temporisation T est activé. La temporisation « retard à la montée » est remise à zéro lors de sa désactivation. Cette temporisation s'arrête lorsque sa valeur maximale est atteinte.

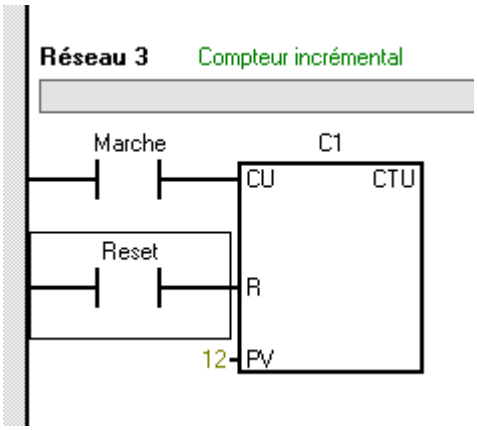


Figure II-14: Compteur incrémental

L'opération Compteur incrémental incrémente jusqu'à la valeur maximale en cas de front montant à l'entrée d'incrémentation CU. Lorsque la valeur en cours « C » est supérieure ou égale à la valeur prédéfinie PV, le bit de compteur C est activé. Le compteur est remis à zéro lorsque l'entrée de remise à zéro R est activée.

II.5.1.b. Structure d'un réseau de contacts :

Un réseau de contacts se compose de la manière suivante : étiquette (ou titre) +

Commentaire + réseau graphique (zone test + zone action).

II.5.1.c. La zone de test accueille :

- les contacts.

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

- les blocs comparaison.

II.5.1.d. La zone action accueille :

- Les bobines.
- Les blocs opérations.

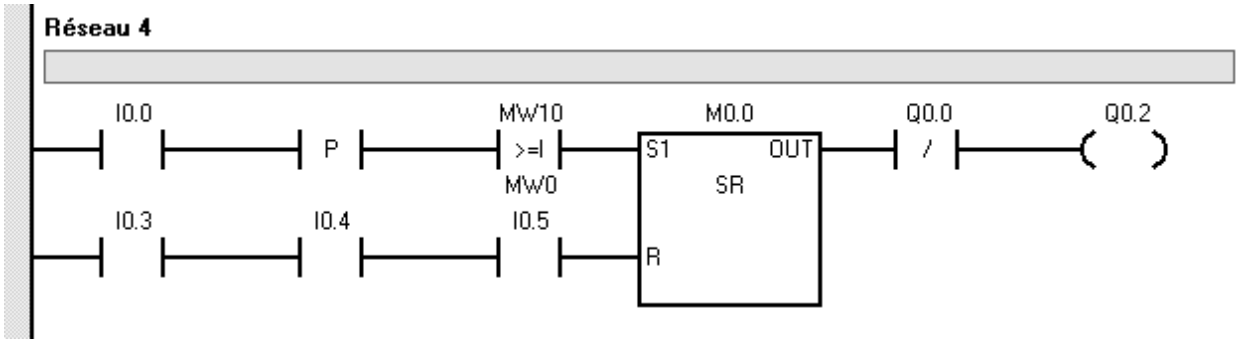


Figure II-15 : Exemple d'une structure d'un réseau de contacts

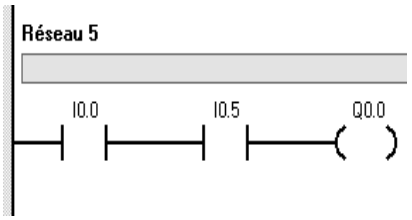
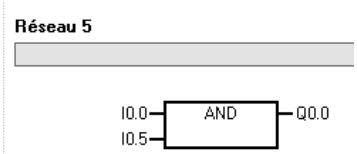
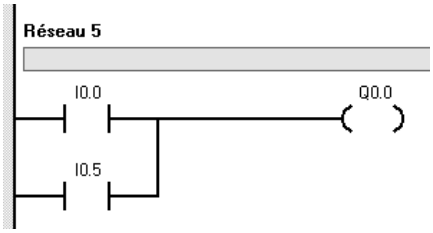
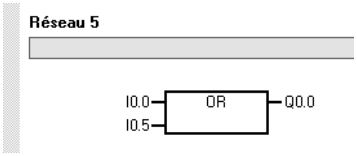
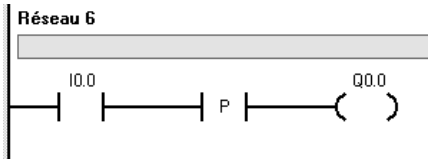
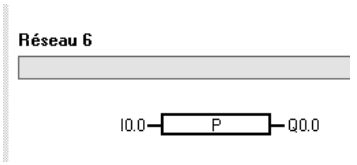
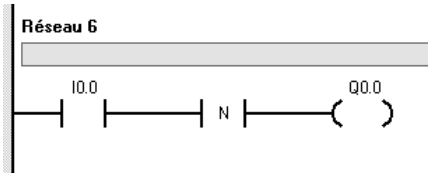
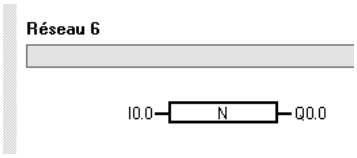
II.5.2. Langage FBD (Function Block Diagram ou Logigramme) :

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

II.5.2.a. Les symboles utilisés :

Les opérations sur bits :

Langage LD	Langage LOG	Fonction

		Cette boîte représente la fonction ET en associant deux bits que l'on peut inverser à l'entrée de la boîte.
		Cette boîte représente la fonction OU en associant deux bits que l'on peut inverser à l'entrée de la boîte.
		Contact à front montant
		Contact à front descendant.

Les circuits séquentiels :

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

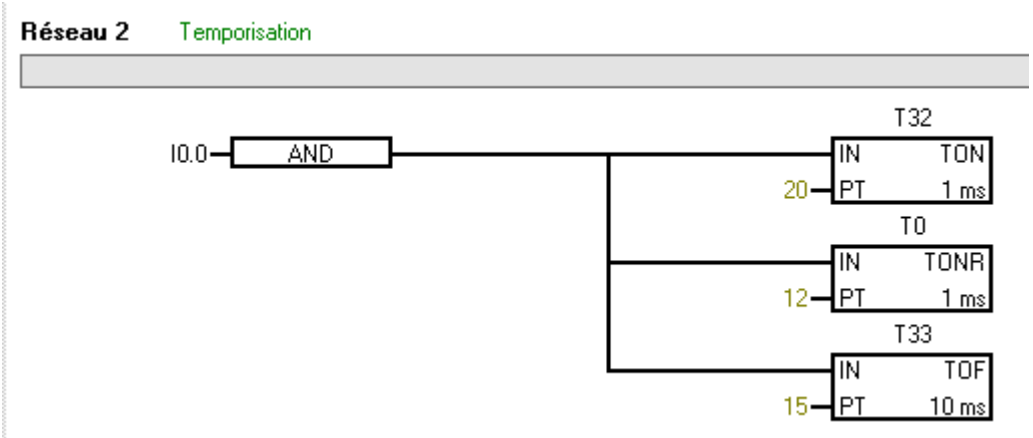


Figure II-16 : Temporisation (Siemens)

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

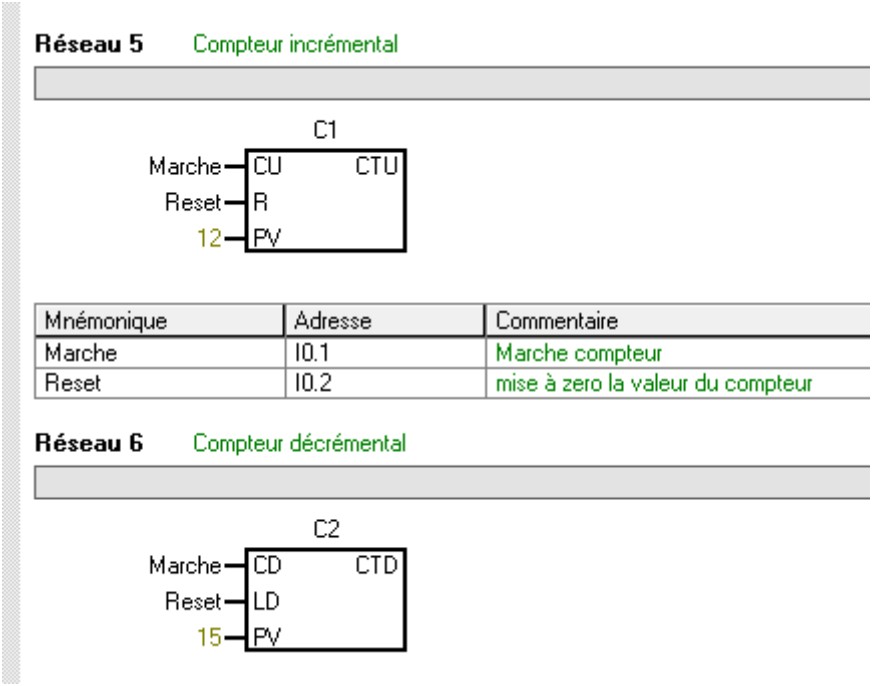


Figure II-17 : Compteurs (Siemens)

II.5.2.b. Les blocs comparaison :

Ces blocs permettent de comparer des nombres, des bits, des octets ou des mots en supériorité, infériorité ou égalité.

II.5.2.c. Les opérations d'exécution :

Ces blocs regroupent plusieurs fonctions établissant l'ordre d'exécution du programme : saut de programme, retour de sous-programme, fin de programme, …

II.5.3. Langage SFC (Sequential function char) : Le Grafcet :

Le GRAFCET c'est un outil graphique de description du comportement déterministe de la Partie Commande, il décrit les interactions informationnelles à caractère déterministe à travers la frontière

d'isolement entre la partie commande et la partie opérative d'un système isolé, il est utilisé pour la description de procédures séquentielles à séquences alternatives ou parallèles.

Les procédures se configurent et se programment clairement et rapidement dans un mode de visualisation standardisé, le processus se décrit graphiquement en se décomposant en différentes étapes à fonctions élémentaires.

- Structure flexible des chaînes séquentielles : Branchements simultanés ou alternatifs, sauts à l'intérieur des chaînes séquentielles, activation et désactivation d'étapes.
- Exécution sélective des étapes : Le temps d'exécution d'une chaîne séquentielle est ainsi fonction du nombre d'étapes.
- Synchronisation des modes automatique et manuel : le processus n'est plus alors synchronisé quand il a été amené dans un autre état en mode manuel. GRAPH supporte la recherche de points de synchronisation en vue de la reprise du mode automatique. Les étapes pertinentes sont à cet effet repérées. Les critères définissables peuvent être des transitions ou des verrouillages.
- Visualisation étape par étape donnant un aperçu de tous les détails d'une étape Gain de temps important par rapport à la programmation en CONT/LOG/LIST.

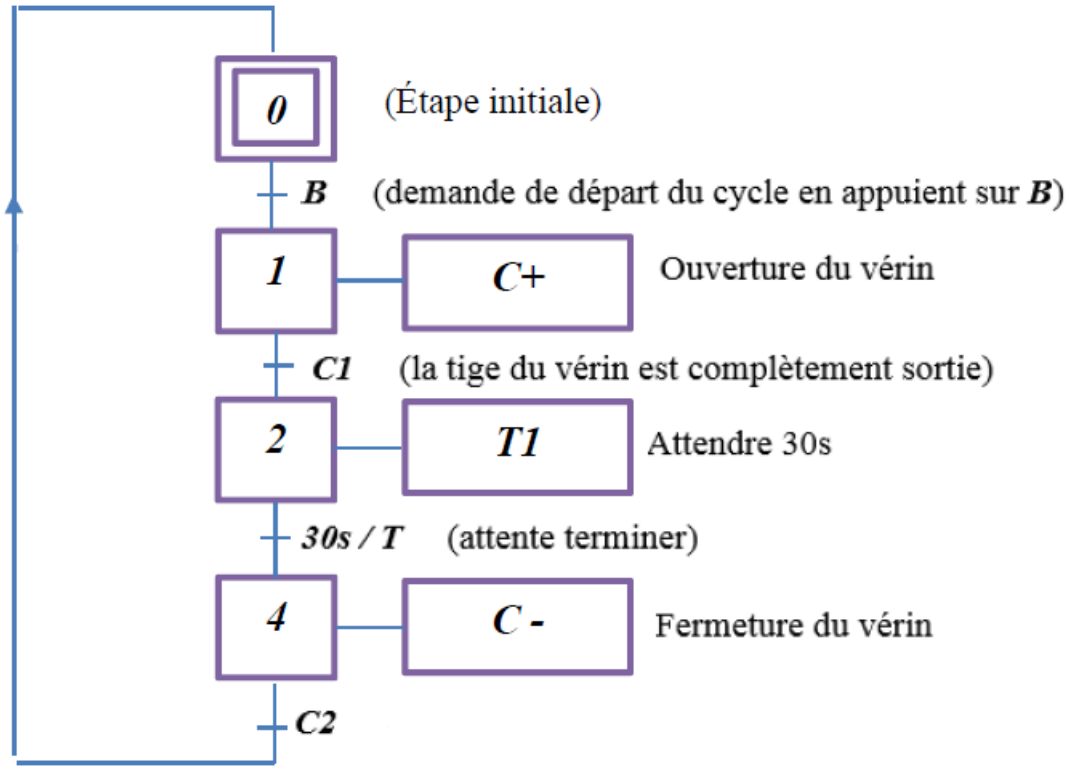


Figure II-18: exemple d'un programme GRAFCET du fonctionnement d'un vérin

II.6. Les langages textuels :

II.6.1. Langage IL (LIST) :

II.6.1.a. Définition :

Le langage de programmation textuelle LIST permet de créer des programmes d'applications à un niveau proche du matériel et en optimisant le temps d'exécution et la place en mémoire.

Il est l'abréviation de liste d'instructions. C'est un langage de liste d'instructions ***(LIST)*** est un langage de programmation textuel proche de la machine.

Dans un programme ***LIST***, les différentes instructions correspondent, dans une large mesure, aux étapes par lesquelles la ***CPU*** traite le programme. Pour faciliter la programmation, ***LIST*** a été complété par quelques structures de langage évolué (comme par exemple, des paramètres de blocs et accès structurés aux données).

Tout programme écrit en CONT ou en LOG peut être réécrit en LIST.

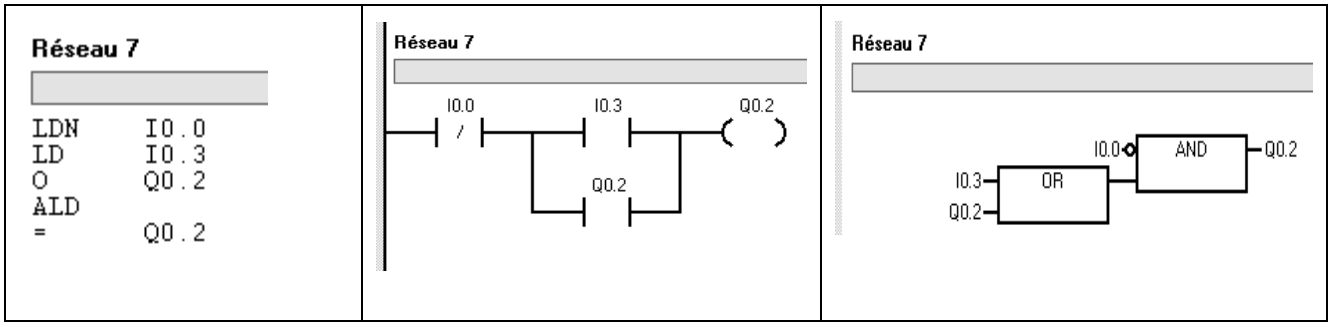


Figure II-19: Exemple d'allumage d'une LED sous langage à List

II.6.1.b. Les instructions de base :

Instructions de test :

- LD : contact normalement ouvert.
- LDN : contact normalement fermé.
- LDR ou EU : contact à front montant.
- LDF ou ED : contact à front descendant.

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

Instructions d'action :

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

II.6.2. SCL (Structured Control Language) :

Programmation d'algorithmes complexes

Ce langage est un langage textuel de haut niveau ST (Structured Text), dédié aux applications d'automatisme. SCL convient notamment à la programmation rapide d'algorithmes complexes et de fonctions mathématiques ou à des missions relevant du domaine du traitement des données difficilement modélisables avec les langages graphiques.

Le code SCL est plus simple, plus court et plus clair, ce qui en facilite et en accélère l'écriture et la manipulation.

II.7. Le GRAFCET

Le GRAFCET (GRAphe Fonctionnel de Commandes des Etapes - Transitions) est un outil graphique de description temporelle du fonctionnement d'un système séquentiel (automatisé).

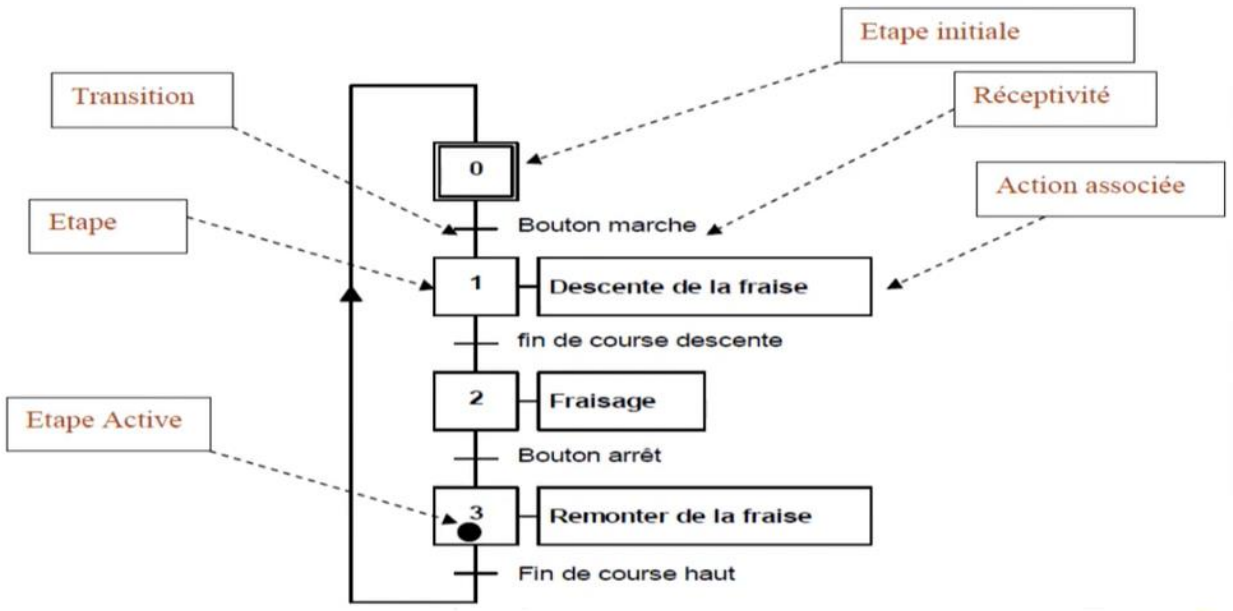
Un système est dit automatisé, s'il exécute toujours, de façon séquentielle, le même cycle de travail conçu sans l'intervention de l'opérateur.

La fin de chaque tâche autorise le début de la suivante.

II.7.1. Éléments graphiques de base

Le GRAFCET est défini par un ensemble constitué d'éléments graphiques de base :

- Les étapes
- Les actions associées aux étapes
- Les transitions
- Les réceptivités associées aux transitions
- Les liaisons orientées.



Étape initiale : l'étape initiale caractérise l'état du système au début du fonctionnement.

Étape : une étape correspond à un comportement stable du système. Les étapes son numérotées dans l'ordre croissant. A chaque étape en peut associer une ou plusieurs actions.

Transition : les transitions indiquent les possibilités d'évolutions du cycle, à chaque transition est associée à une réceptivité.

Réceptivité : la réceptivité est la condition logique pour évolution du grafcet. Si la réceptivité est vraie (=1) le cycle peut évoluer. Les réceptivités proviennent du pupitre de commande, des fins de courses ou d'information provenant de la partie opérative.

Liaisons orientées : le Grafcet se lit de haut en bas, autrement il est nécessaire d'indiquer son évolution avec des liaisons orientées constituées de flèche indiquant le sens.

Action : L'action est associée à une étape, elle est active lorsque le cycle est arrivé sur l'étape. Il est possible de définir les actions conditionnelles, temporisé… (électrovanne, enclenchement d'un contacteur…).

Étape active : le point indique l'étape est active.

II.7.2. Différent type de Grafcet

- Grafcet point de vue système ;
- Grafcet point de vue partie opérative ;
- Grafcet point de vue partie commande ;
- Grafcet point de vue automate.

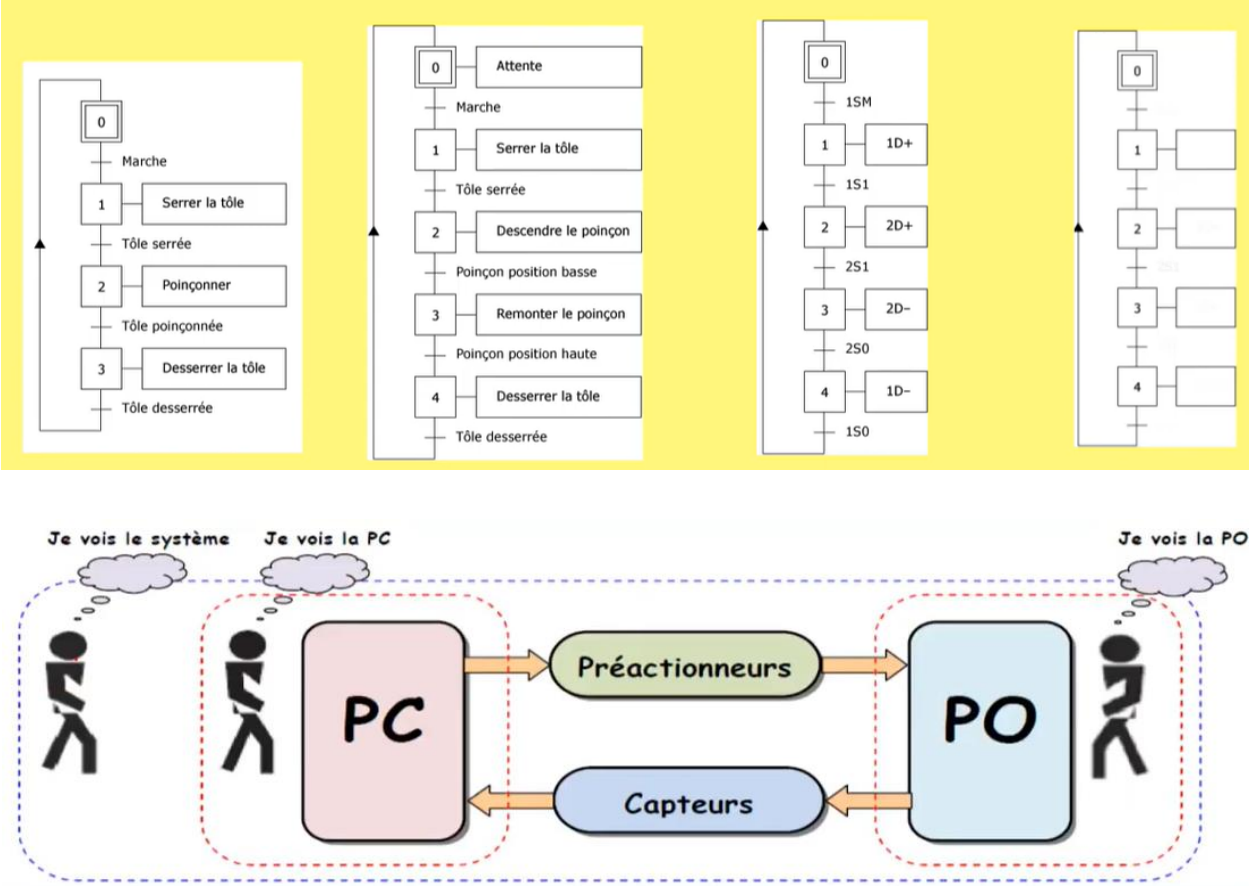


Figure II-20: Différent type de Grafcet

II.7.3. Règles d'évolution du Grafcet

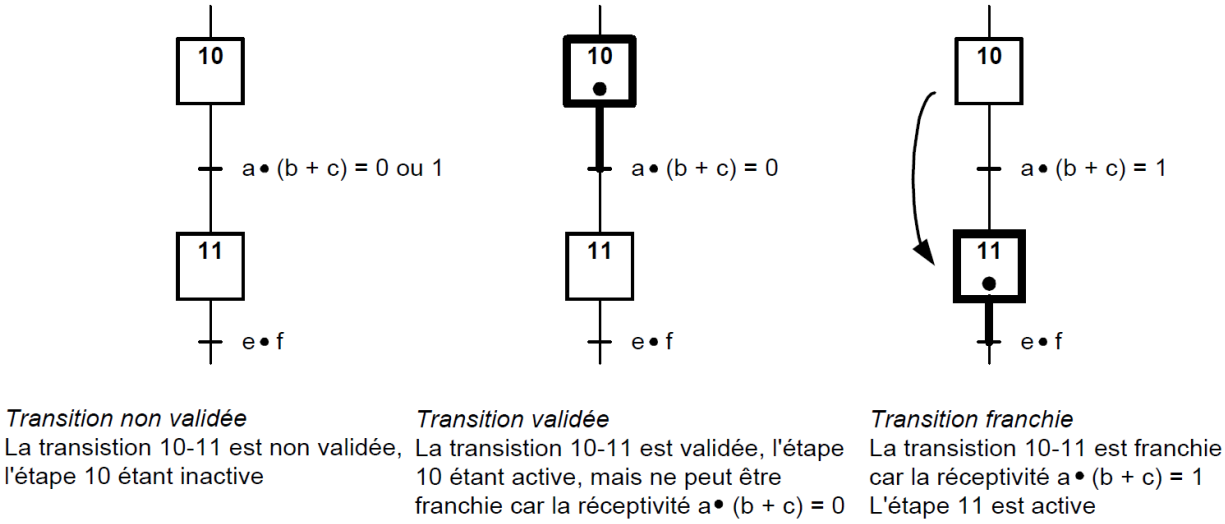
II.7.3.a. Règle 1 : Situation initiale

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

Figure II-21: Étape initiale

II.7.3.b. Règle 2 : Franchissement d'une transition

Une transition est soit validée ou non validée, elle est valide lorsque toutes les étapes immédiatement précédentes sont actives. Lorsque la transition est valide et que la réceptivité associée est vraie elle est alors obligatoirement franchie.



Transition non validée
La transistion 10-11 est non validée, l'étape 10 étant inactive

Transition validée
La transistion 10-11 est validée, l'étape 10 étant active, mais ne peut être franchie car la réceptivité a • (b + c) = 0

Transition franchie
La transistion 10-11 est franchie car la réceptivité a • (b + c) = 1
L'étape 11 est active

Figure II-22: Franchissement d'une transition

II.7.3.c. Règle 3 : Évolution des étapes actives :

Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes. Cette évolution du GRAFCET est donc synchrone.

Il y a évolution asynchrone lorsque le franchissement de la transition entraîne l'activation des étapes suivantes et que c'est la vérification de cette activation qui autorise la désactivation des étapes précédentes.

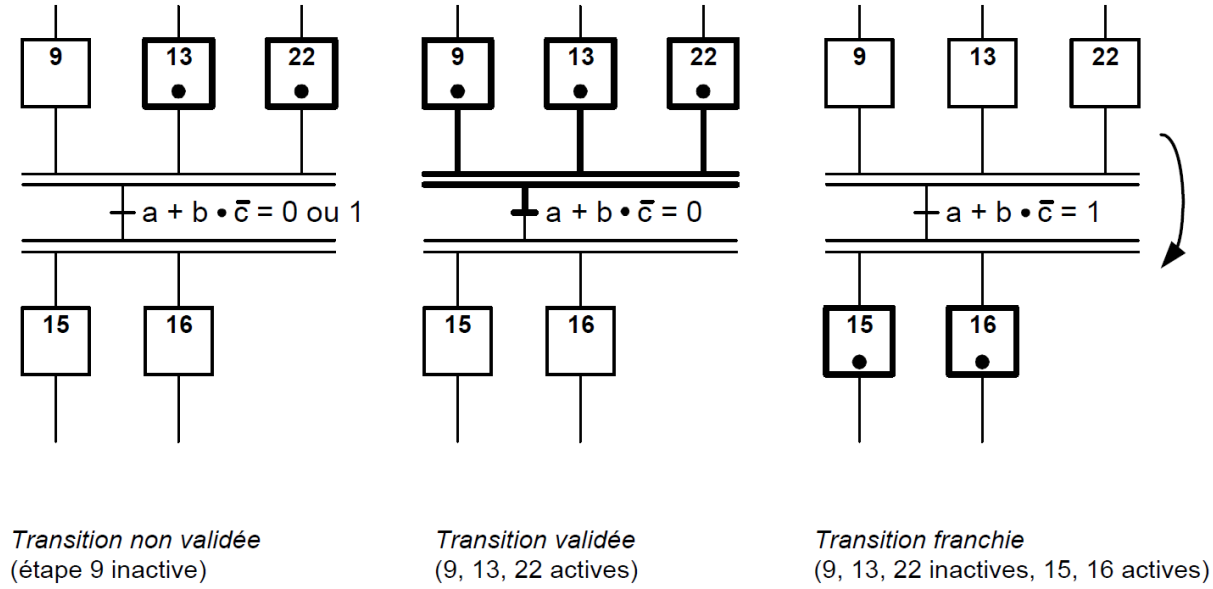


Figure II-23: Evolution des étapes actives

II.7.3.d. Règle 4 : Transitions simultanées

Plusieurs transitions simultanément franchissables sont simultanément franchies.

II.7.3.e. Règle 5 : Activation et désactivation simultanées

Si au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste activée. L'activation doit être prioritaire sur la désactivation au niveau d'une même étape.

Remarque : La durée de franchissement d'une transition ne peut jamais être rigoureusement nulle, même si, théoriquement (règles 3 et 4), elle peut être rendue aussi petite que possible.

Il en est de même de la durée d'activation d'une étape. En outre, la règle 5 se rencontre très rarement dans la pratique. Ces règles ont été ainsi formulées pour des raisons de cohérence théorique interne au GRAFCET.

II.7.4. Actions associées aux étapes.

Rappelons que l'ordre conditionne l'action.

II.7.4.a. Ordre continu

L'ordre est émis de façon continue tant que l'étape à laquelle il est associé est active.

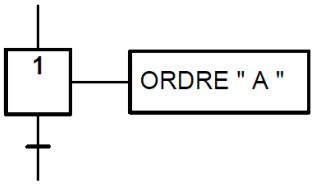


Figure II-24: ordre continu

II.7.4.b. Ordre conditionnel

L'ordre est émis lorsqu'en plus de l'activité de l'étape à laquelle il est associé, une condition logique spécifiée doit être satisfaite.

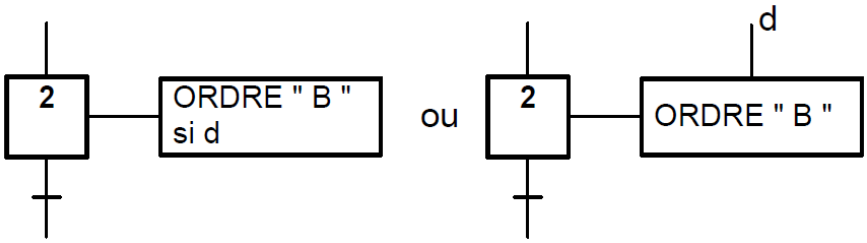


Figure II-25: ordre conditionnel

II.7.4.c. Ordre de mémorisation de l'action

Ces deux ordres, de mémorisation et d'effacement permettent d'élaborer l'action de sortie du composant : ordre de début d'action, noté : "action = 1" (set) ordre de fin d'action, noté : "action=0" (reset).

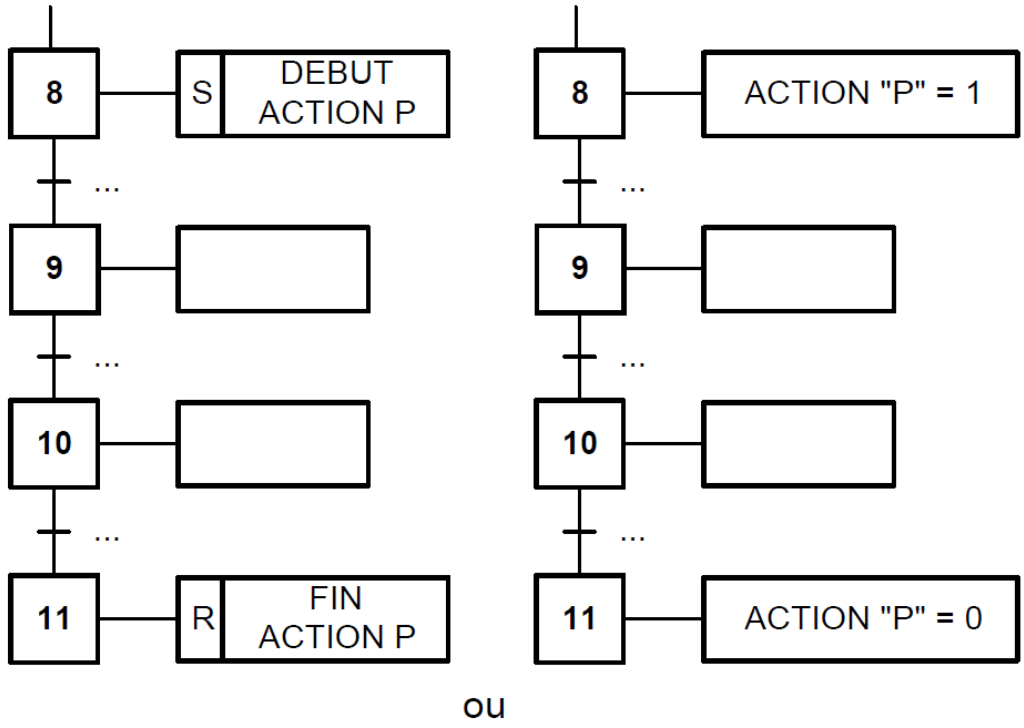


Figure II-26: ordre de mémorisation

II.7.4.d. Ordre retardé (D)

C'est un cas particulier d'un ordre conditionnel ou le temps intervient comme condition logique

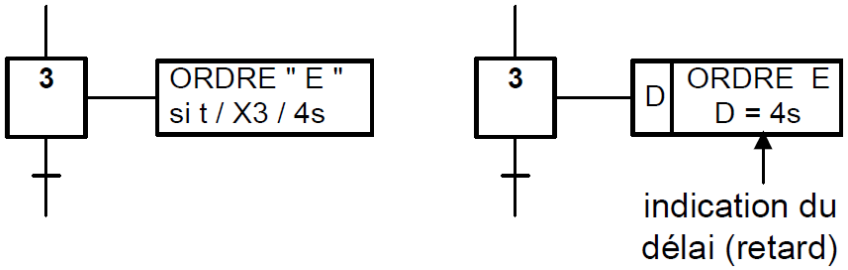


Figure II-27: Ordre retardé

II.7.4.e. Ordre de durée limitée (L)

L'ordre est émis immédiatement dès l'activation de l'étape à laquelle il est associé, mais sa durée est limitée à la valeur spécifiée.

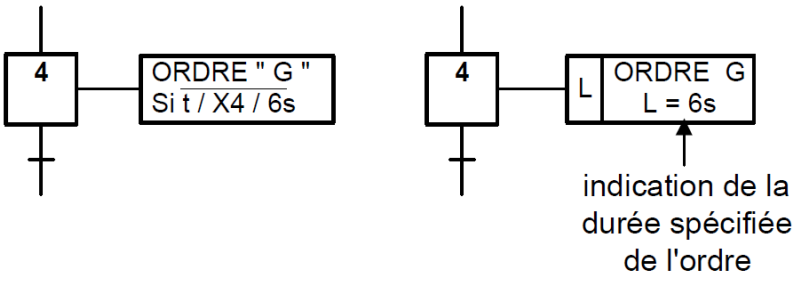


Figure II-28: ordre de durée limitée

II.7.5. Décompteur

Il faut prévoir une séquence d'initialisation (ou de remise à zéro dans le cas d'un compteur).

Après l'action, on établit une séquence de décrémentation du décompteur suivi d'une reprise de séquence en fonction de la valeur de celui-ci

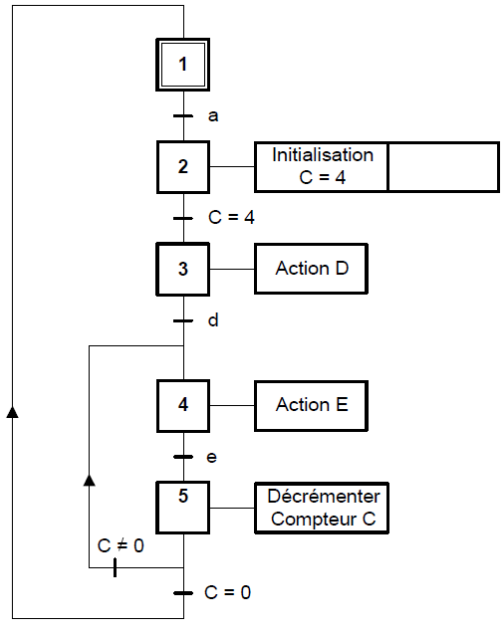


Figure II-29: Décompteur

II.7.6. Transitions

II.7.6.a. Transition

Les transitions indiquent les possibilités d'évolution entre étape. On associe à chaque transition une condition logique appelée réceptivité.

II.7.6.b. Réceptivité

La réceptivité est écrite sous forme de proposition logique, c'est une information simple ou une fonction combinatoire d'informations extérieures (capteur, compteur, fin de courses, etc. …)

II.7.7. Liaison orientée

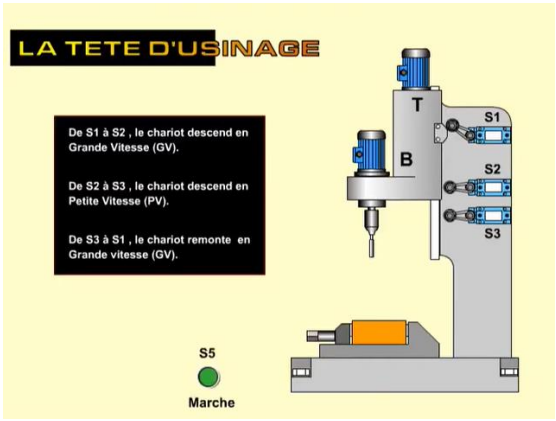
Les liaisons indiquent l'évolution de l'état du grafcet.

Les liaisons sont horizontales ou verticales.

II.7.8. Séquence de base

II.7.8.a. Grafcet linéaire (séquence unique)

Le Grafcet linéaire ci-dessous, représente un cycle fonctionnel d'une perceuse, c'est une succession d'étape et de transitions.



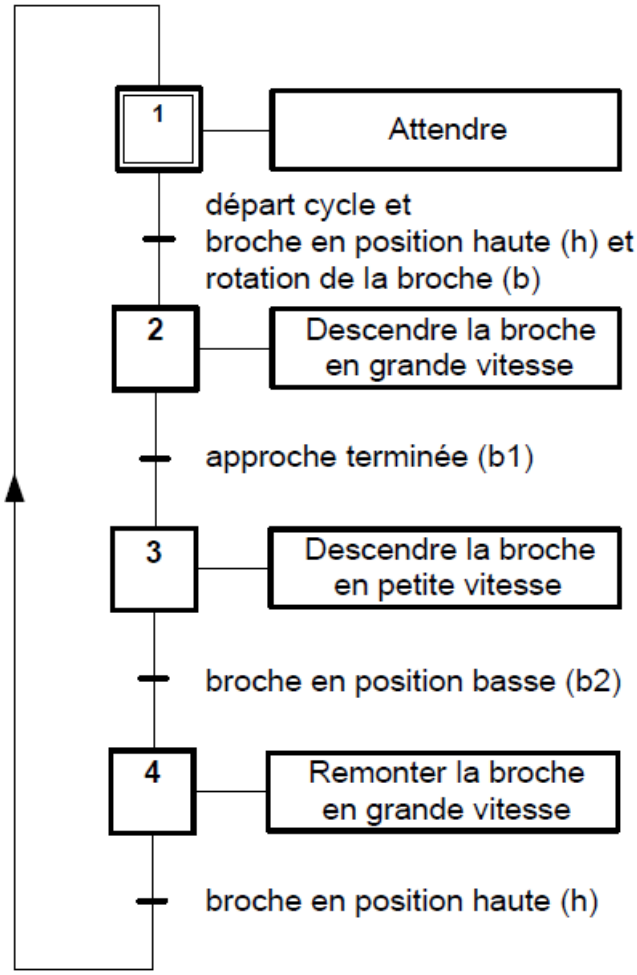


Figure II-30: Grafcet linéaire

II.7.8.b. Divergence et convergence en « OU »

Choix conditionnel entre plusieurs séquences.

Un GRAFCET est généralement constitué de plusieurs séquences, c'est-à-dire de plusieurs suites d'étapes à exécuter les unes après les autres et est souvent nécessaire d'effectuer une sélection exclusive d'une de ces séquences.

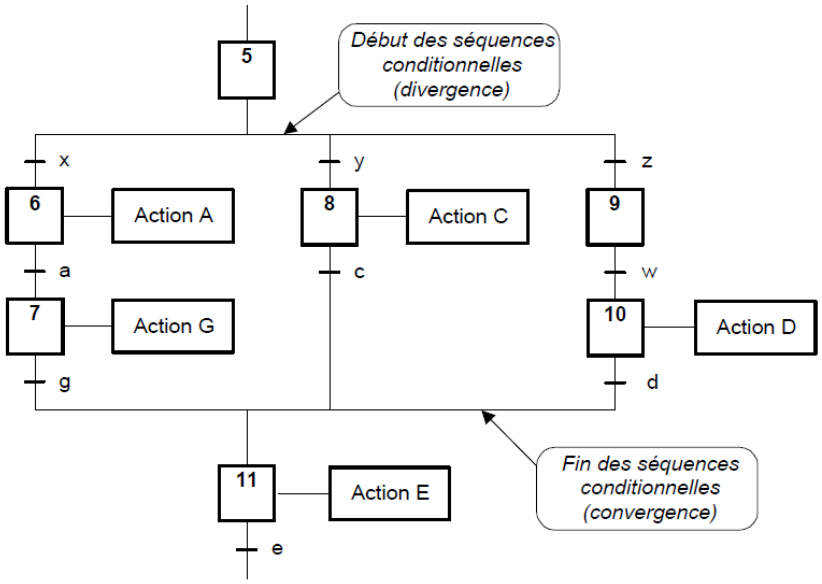


Figure II-31: Divergence et convergence en "OU"

Remarque : Une divergence entraîne automatiquement une convergence.

Dans l'aiguillage formé par le choix de la séquence à réaliser, les différentes transitions correspondant aux réceptivités x, y et z étant simultanément validées par la même étape 5, pourrait d'après la règle de simultanéité, être franchies simultanément. En pratique l'automaticien est souvent amené à rendre des réceptivités exclusives. Il est possible également d'introduire des priorités.

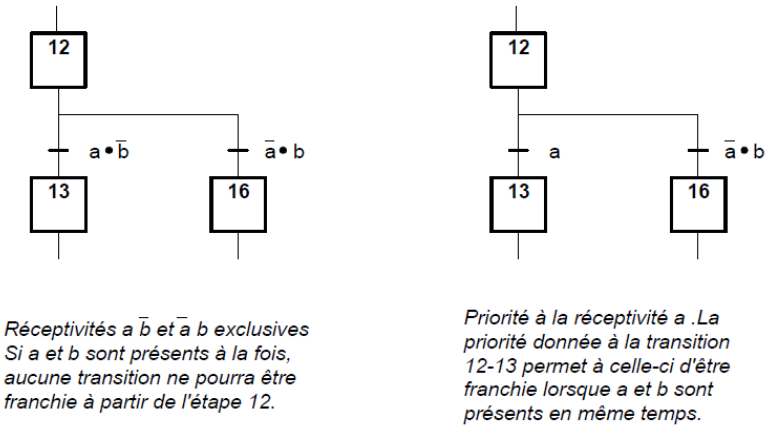


Figure II-32: réceptivités exclusives.

II.7.8.c. Divergence et convergence en "ET"

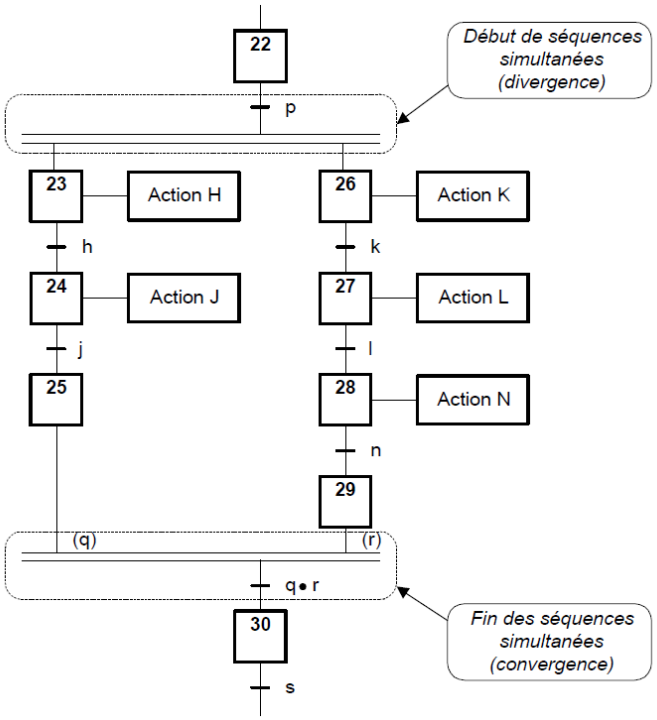


Figure II-33: divergence et convergence en ET.

Un GRAFCET peut comporter plusieurs séquences s'exécutant simultanément mais dont les évolutions des étapes actives dans chaque branche restent indépendantes.

Pour représenter ces fonctionnements simultanés, une transition UNIQUE et deux traits parallèles indiquent le début et la fin des séquences, c'est-à-dire l'activation simultanée des branches ainsi réalisées et leur attente réciproque vers une séquence commune.

A partir de l'étape 22 de la figure 13, la réceptivité p provoque l'activation simultanée des étapes 23 et 26.

Ces des séquences 23-24-25 et 26-27-28-29 évolueront alors de façon totalement indépendante et ce n'est que :

- lorsque les étapes de fin de branche 25 et 29 sont actives

- lorsque la réceptivité est vraie (q . r = 1),

Que la transition sera franchie. L'étape 30 devient alors active et les étapes 25 et 29 inactives.

II.7.9. Sauts d'étapes et reprise de séquences

Le saut conditionnel est un aiguillage particulier permettant de sauter une ou plusieurs étapes lorsque les actions à réaliser deviennent utiles, tandis que la reprise de séquences permet au contraire de reprendre une ou plusieurs fois la même séquence tant qu'une condition fixée n'est pas obtenue

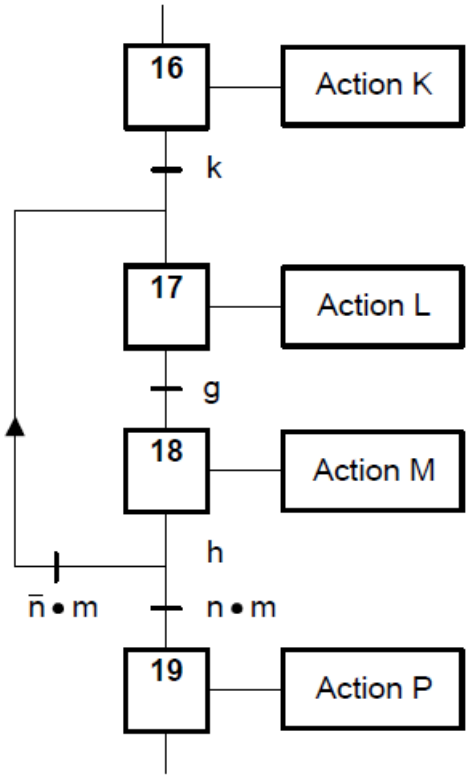


Figure II-34: Saut d'étape

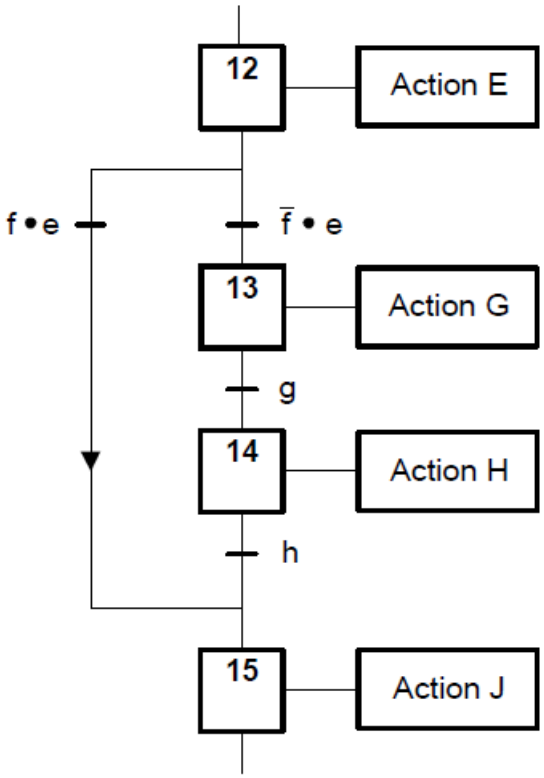


Figure II-35: reprise de séquence

II.8. Principe de la programmation sous STEP7 :

Les fonctions suivantes peuvent être utilisées avec STEP7 pour l'automatisation d'un dispositif :

Création et la gestion du projet.

- La configuration et le paramétrage du matériel et de la communication.
- La gestion des mnémoniques.
- La création du programme.
- Le chargement du programme dans les systèmes cible.
- Le teste et l'installation d'automatisation.
- Le diagnostic lors de perturbations de l'installation.

II.8.1. Structure du programme :

La programmation structurée permet la rédaction claire et transparente de programmes, elle la construction d'un programme complet à l'aide de modules qui peuvent être échangés et/ou modifiés à volonté. Pour permettre une programmation structurée confortable, il faut prévoir plusieurs types de modules : (OB) Bloc d'organisation, (FB) Bloc fonctionnel, (FC) Fonction, (SFB) Bloc fonctionnel système, (SFC) Fonction système, (DB) Bloc de données.

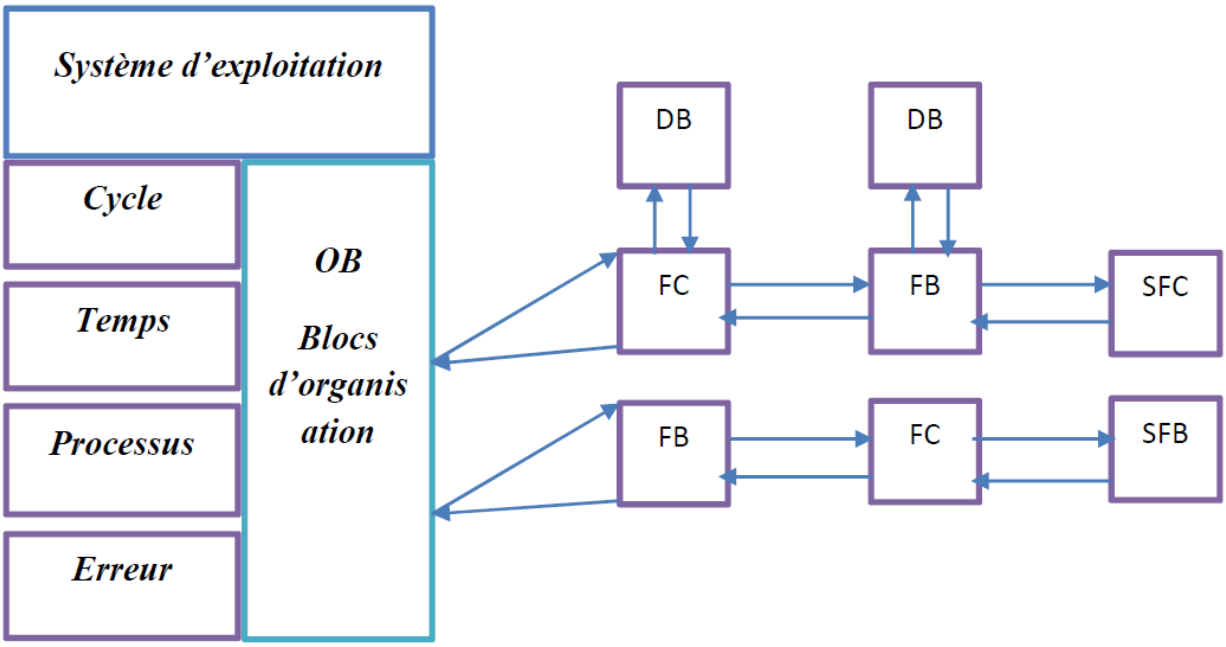


Figure II-36: Blocs de programme.

Blocs **:** Le système d'automatisation utilise différents types de blocs dans lesquels peuvent être mémorisés le programme utilisateur et les données correspondantes. Selon les exigences du processus, le programme peut être structuré en différents blocs.

Bloc d'organisation : Les blocs d'organisation (***OB***) constituent l'interface entre le système

OB d'exploitation et le programme utilisateur. L'ensemble du programme peut être concaténé dans un seul bloc OB1 (programme linéaire) appelé de manière cyclique par le système d'exploitation ou être structuré dans plusieurs blocs (programme structuré).

Fonction : Une fonction (***FC***) assure une fonctionnalité spécifique du programme.

FC, SFC Les fonctions peuvent être paramétrables. Dans ce cas, des paramètres sont transmis à la fonction lorsqu‘elle est appelée. Les fonctions conviennent donc pour la programmation de fonctionnalités récurrentes et complexes, par exemple pour effectuer des calculs.

Les fonctions système (***SFC***) sont des fonctions paramétrables, intégrées au système d‘exploitation de la CPU, dont le numéro et la fonctionnalité sont définis de manière fixe. Pour de plus amples informations, se reporter à l‘aide en ligne.

Bloc fonctionnel : Du point de vue du programme, les blocs fonctionnels s‘apparentent aux

FB, SFB fonctions, mais ils disposent en plus de zones mémoire spécifiques, sous forme de blocs de données d‘instance. Les blocs fonctionnels conviennent donc pour la programmation de fonctionnalités récurrentes encore plus complexes, par exemple pour assurer des tâches de régulation.

Blocs fonctionnels système (***SFB***) ***:*** sont des blocs fonctionnels paramétrables, intégrés au système d‘exploitation de la CPU, dont le numéro et la fonctionnalité sont définis de manière fixe.

Pour de plus amples informations, se reporter à l‘aide en ligne.

Blocs de données : Les blocs de données (***DB***) sont des zones de données du programme **DB** utilisateur.

III.2.4. Les opérandes et types de données autorisés dans la table des mnémoniques : [12]

Saisir les mnémoniques est très utile, il vaut mieux saisir un programme entièrement en symbole qu’en adressage absolu, c’est beaucoup plus lisible et compréhensible.

Il suffit d’aller dans la table des mnémoniques et y entrer les différents éléments, le nom du symbole, sont adresse réel, sont type et son commentaire.

Les paramètres peuvent être des types simples tel que : BOOL, WORD, INT, REAL, …ect, ou de types complexes tel que : TIMER, COUNTER, ect

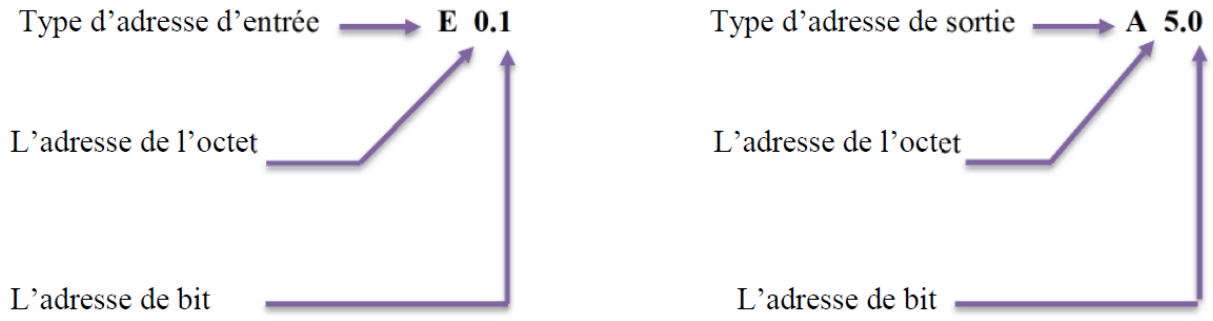
Les différents types de variables sont donnés dans le tableau qui suit, dans le cadre de cette première approche, on fera plus particulièrement attention aux variables suivantes :

Opérandes	*Désignation*	*Type de données*	*Plage d'adresse*
E	*Bit d'entré*	***BOOL***	***0.0 → 127.7***
A	*Bit de sotie*	***BOOL***	***0.0 → 127.7***
M	*Bit de mémento*	***BOOL***	***0.0 → 255.7***
MB	*Octet de mémento*	***BYTE, CHAR (8bit)***	***0 → 255***
MW	*Mot de mémento*	***WORD, INTEGER (16bit)***	***0 → 254***
MD	*Double mot de mémento*	***WORD, REAL (32bit)***	***0 → 254***
T	*Temporisation*	***TIMER***	***0 → 127***
Z	*Compteur*	***COUNTER***	***0 → 127***
VAT	*Table de variable*	***- - -***	***0 → 127***

Table II-1: Les différents types de variable

II.8.2. L'adressage des modules E/S :

Une entrée ou une sortie est désignée dans le programme à l'aide d'une adresse qui indique clairement quel est son emplacement sur l'automate. Pour adresser une entrée ou une sortie il faut entrer une adresse comme suit :



Utilisation du front montant :

Pour cela, il faut utiliser le bloc POS, il utilise un memento de front et l'entrée sur laquelle on veut détecter le front.

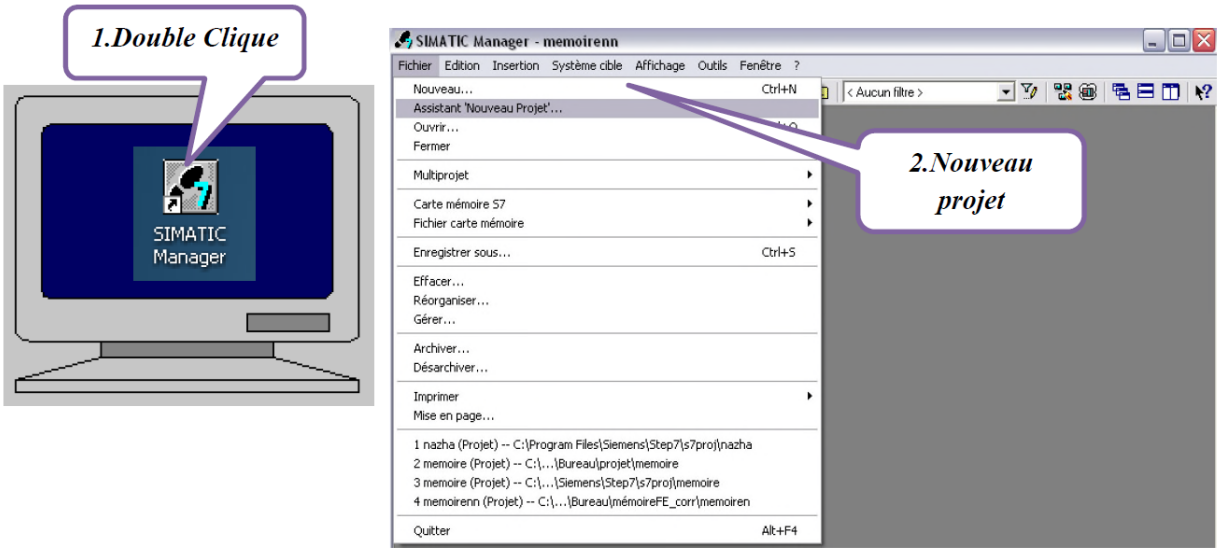
Dans notre programme on l'a utilisé pour le remplissage et le soutirage du sucre dans les trémies, la détection du front montant pour déclencher l'additionneur à incrémenter une valeur désirer.

II.8.3. Création d'un projet STEP7 :

Une fois Windows démarré, on trouve dans l'interface Windows une icône pour SIMATIC

Manager qui permet d'accéder au logiciel STEP 7.

Afin de crée un nouveau projet STEP7 il nous est possible d'utiliser « l'assistant de création de projet » ou bien crée le projet soi-même et le configurer directement, cette dernière est un peu plus complexe mais nous permet aisément de gérer notre projet. En sélectionnant l'icône SIMATIC Manager on affiche la fenêtre principale pour sélectionner un nouveau projet et le valider comme nous montre les figures suintantes :

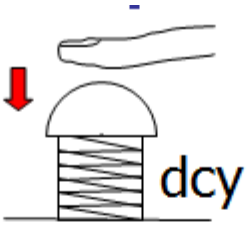


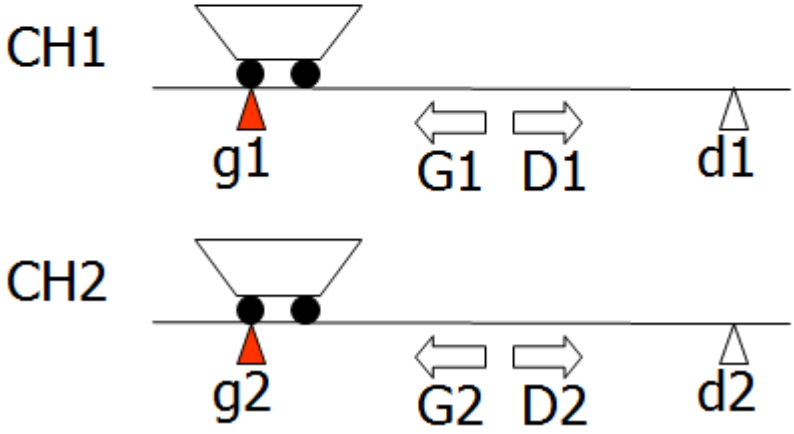
III. Travaux Dirigés

III.1. Exercice 01 :

III.1.1. Cahier des charges :

Après l'appui sur le départ cycle « dcy », les chariots partent pour un aller-retour. Un nouveau départ cycle ne peut se faire que si les deux chariots sont à gauche.





CH1, CH2 : chariot 1, 2

g : capteur « position gauche »

d : capteur « position droite »

G : action « aller à gauche »

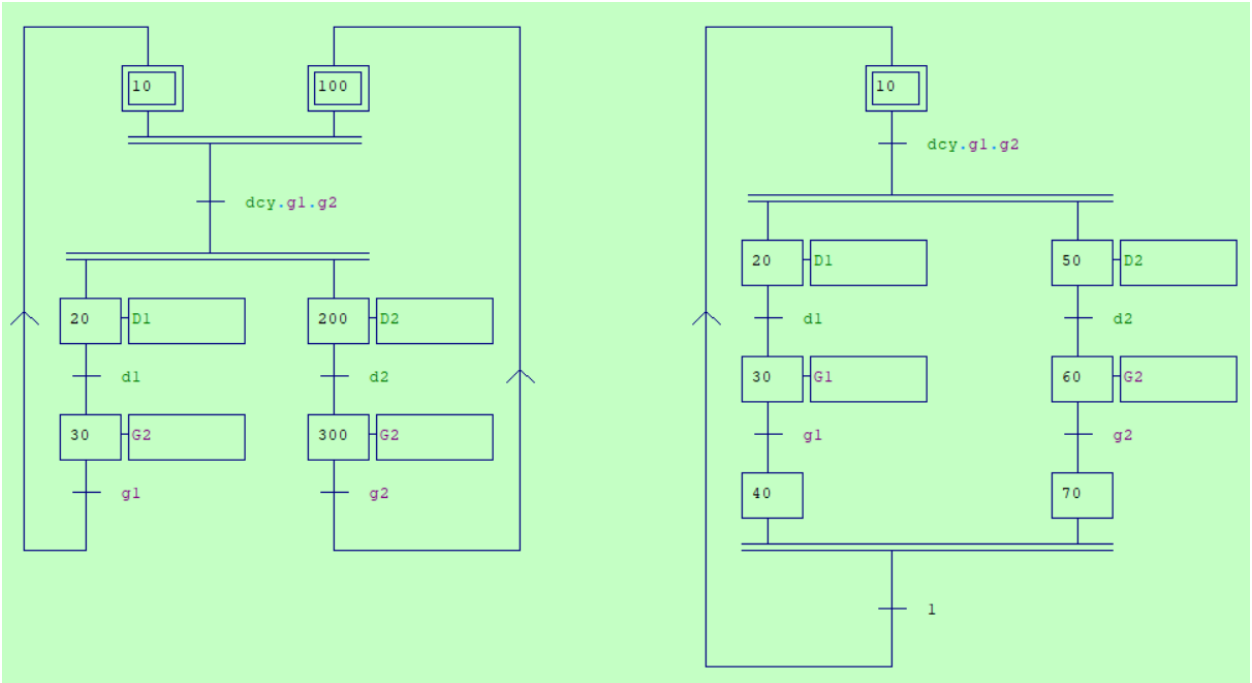
D : action « aller à droite »

Donner :

- Le grafcet.
- Traduisez le grafcet en langage LADERR

III.1.2. Solution :

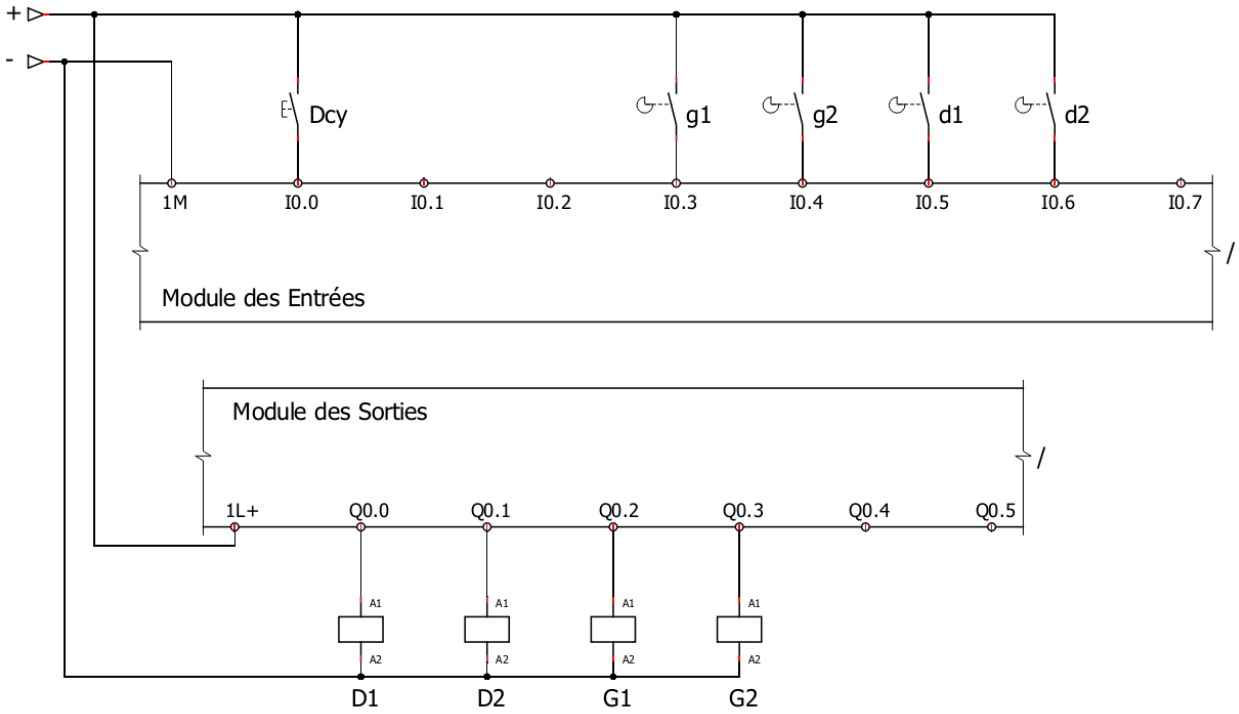
III.1.2.a. Le grafcet



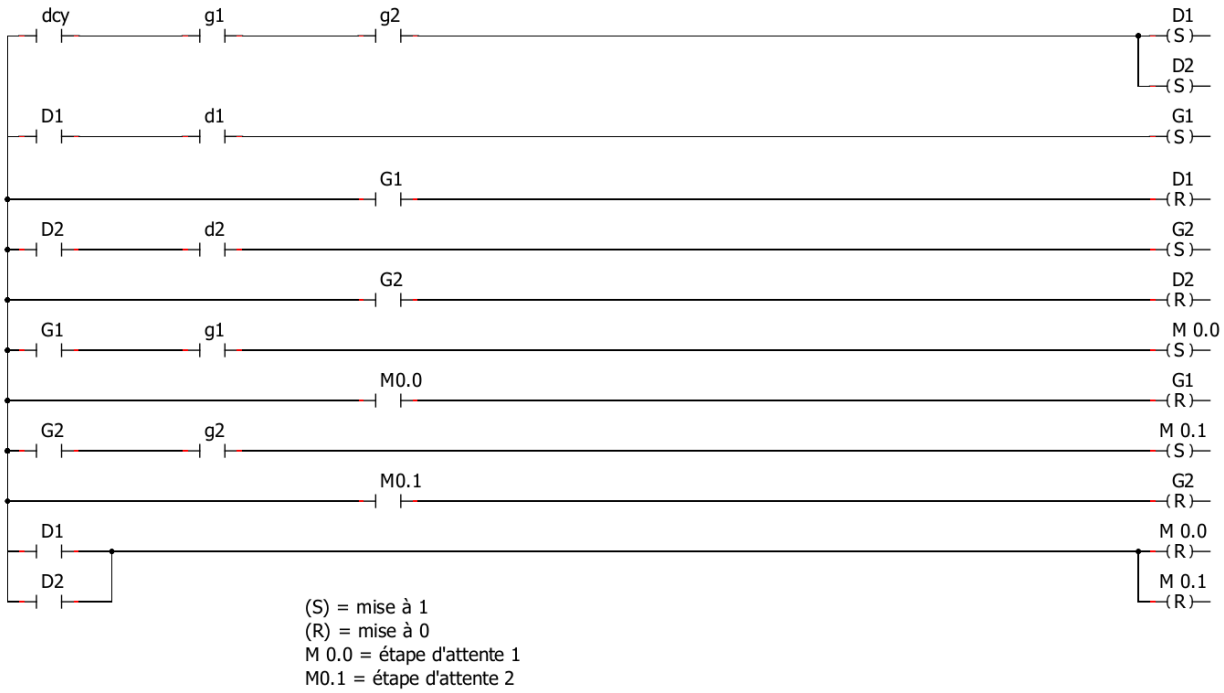
III.1.2.b. Tableau des adresses des entrées et des sorties

Entrées	Sorties
Dcy ----------> I 0.0	**D1 ----------> Q 0.0**
g1 ----------> I 0.3	**D2 ----------> Q 0.1**
g2 ----------> I 0.4	**G1 ----------> Q 0.2**
d1 ----------> I 0.5	**G2 ----------> Q 0.3**
d2 ----------> I 0.6	

III.1.2.c.Câblage des entrées et des sorties

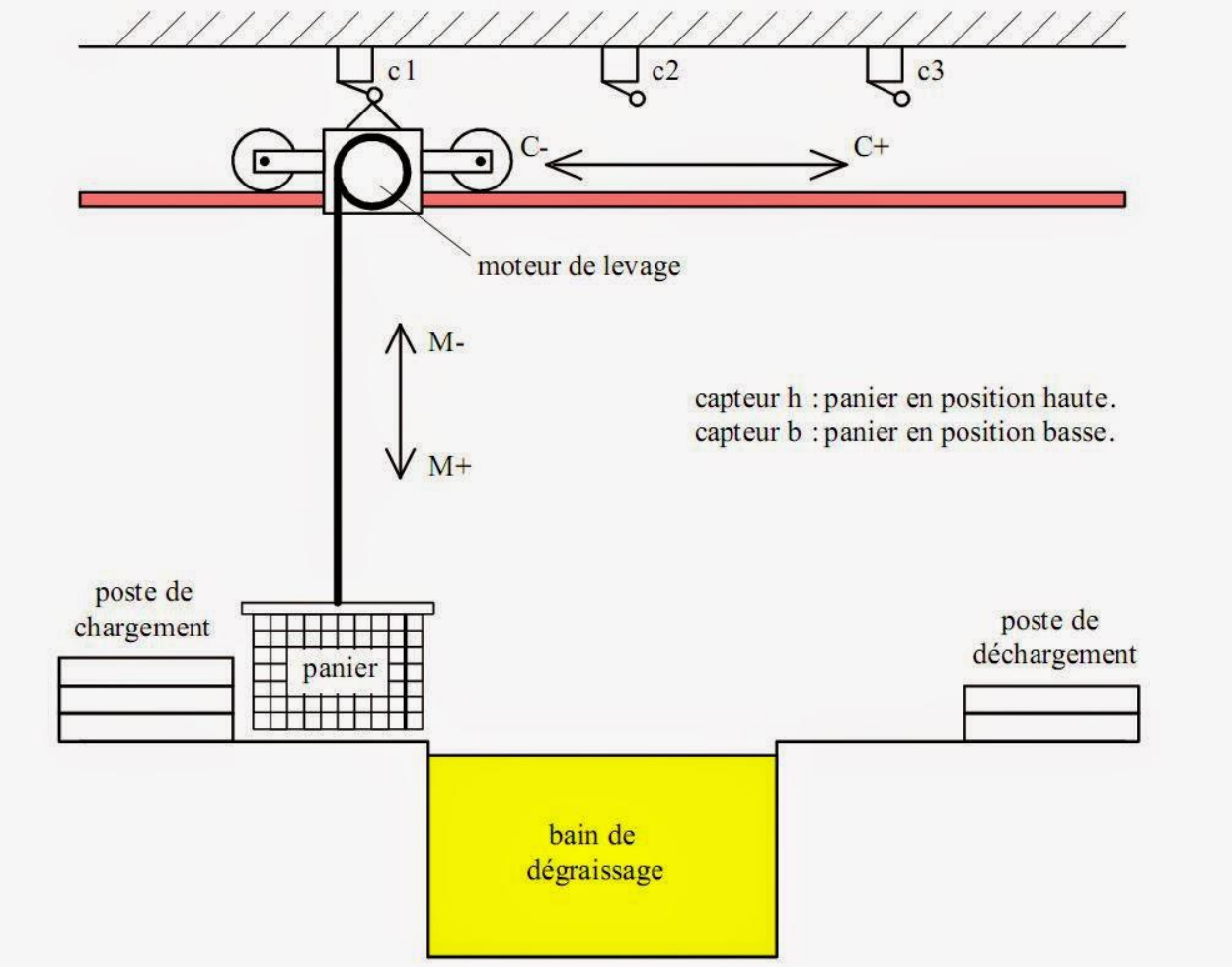


III.1.2.d. Programme Ladder



III.2. Exercice 02 : BAIN DE DÉGRAISSAGE

III.2.1. Cahier des charges :



III.2.2. Fonctionnement :

Un chariot se déplace sur un rail et permet, en se positionnant au-dessus d'une cuve, de nettoyer des pièces contenues dans un panier en les trempant dans un bac de dégraissage. Cycle détaillé :

- Quand le chariot est en haut à gauche et que l'on appuie sur le bouton de départ du cycle (dcy), le chariot va au-dessus du bac de dégraissage.
- Le panier descend alors dans ce bac où on le laisse 30 secondes.
- Après cette attente, le panier remonte.
- Après cela, le chariot va jusqu'à l'extrême droite où il sera déchargé.

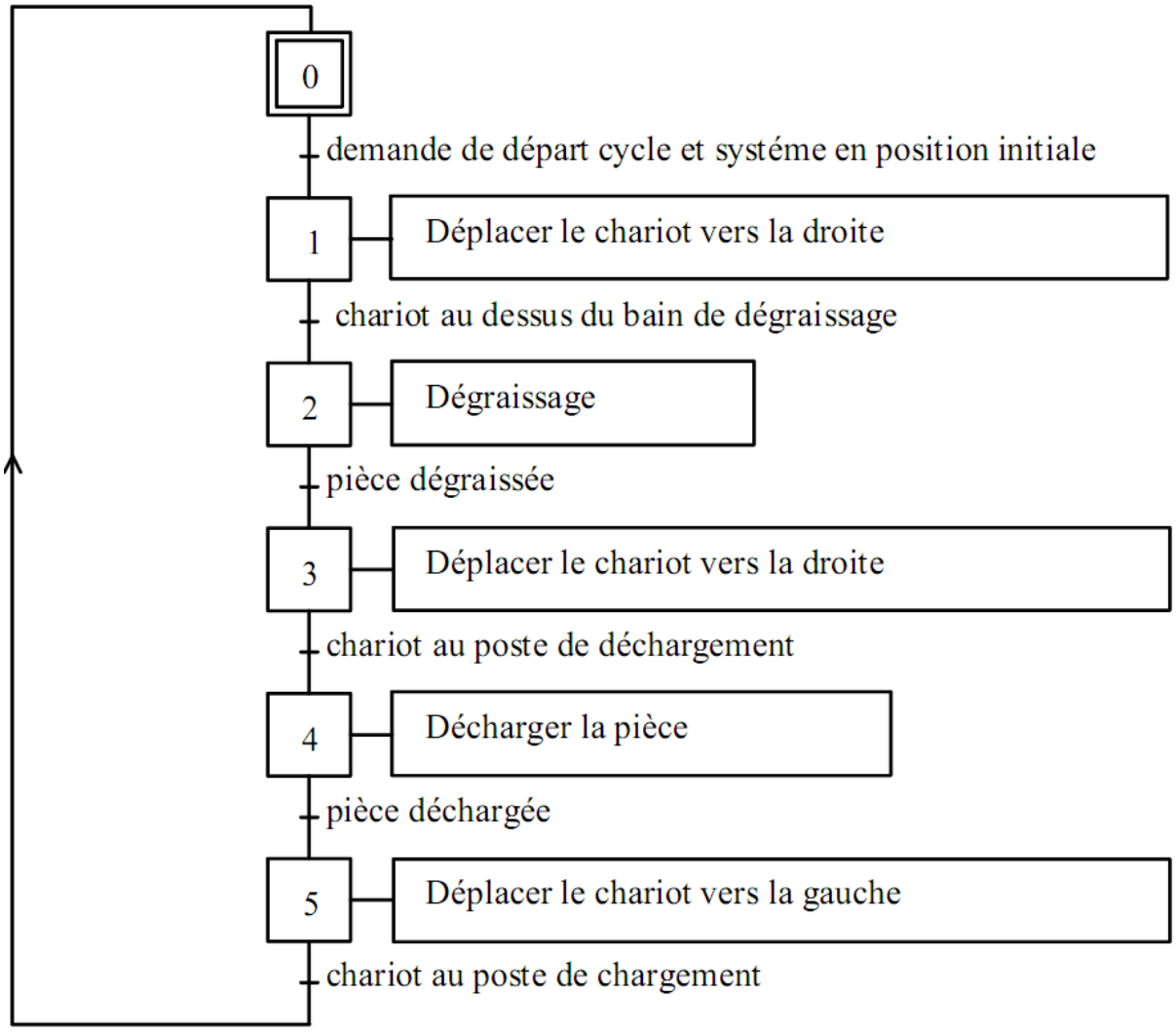
- Quand le déchargement est terminé, le système revient dans sa position de départ. **Remarque :** Le chargement et le déchargement du panier s'effectuent manuellement. Le contrôle du fait que le panier est déchargé sera donc validé par un bouton poussoir d.

Donner :

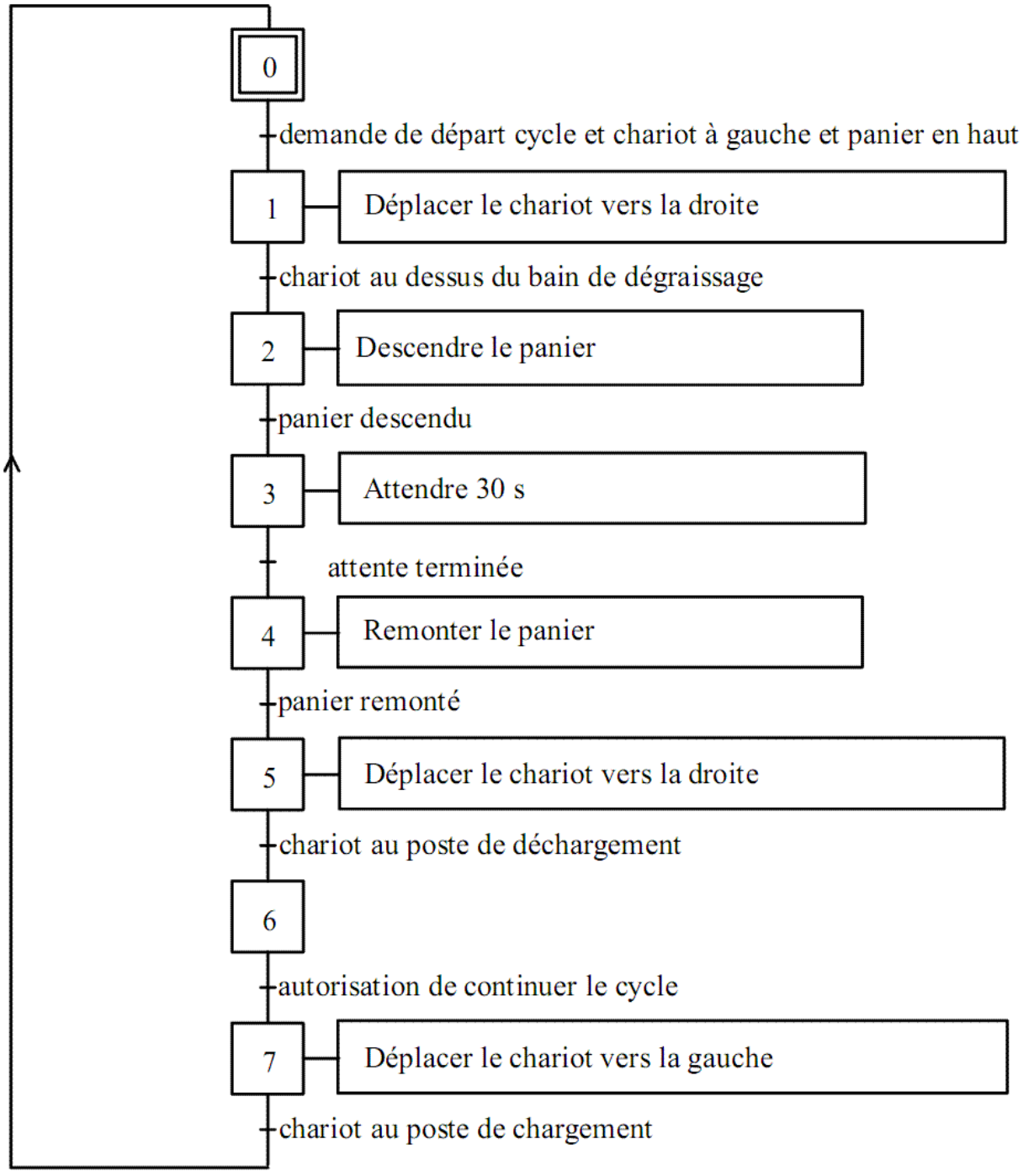
1. Le grafcet point de vue système.
2. Le grafcet point de vue Partie Opérative.
3. Le grafcet point de Partie commande.

III.2.3. Solution :

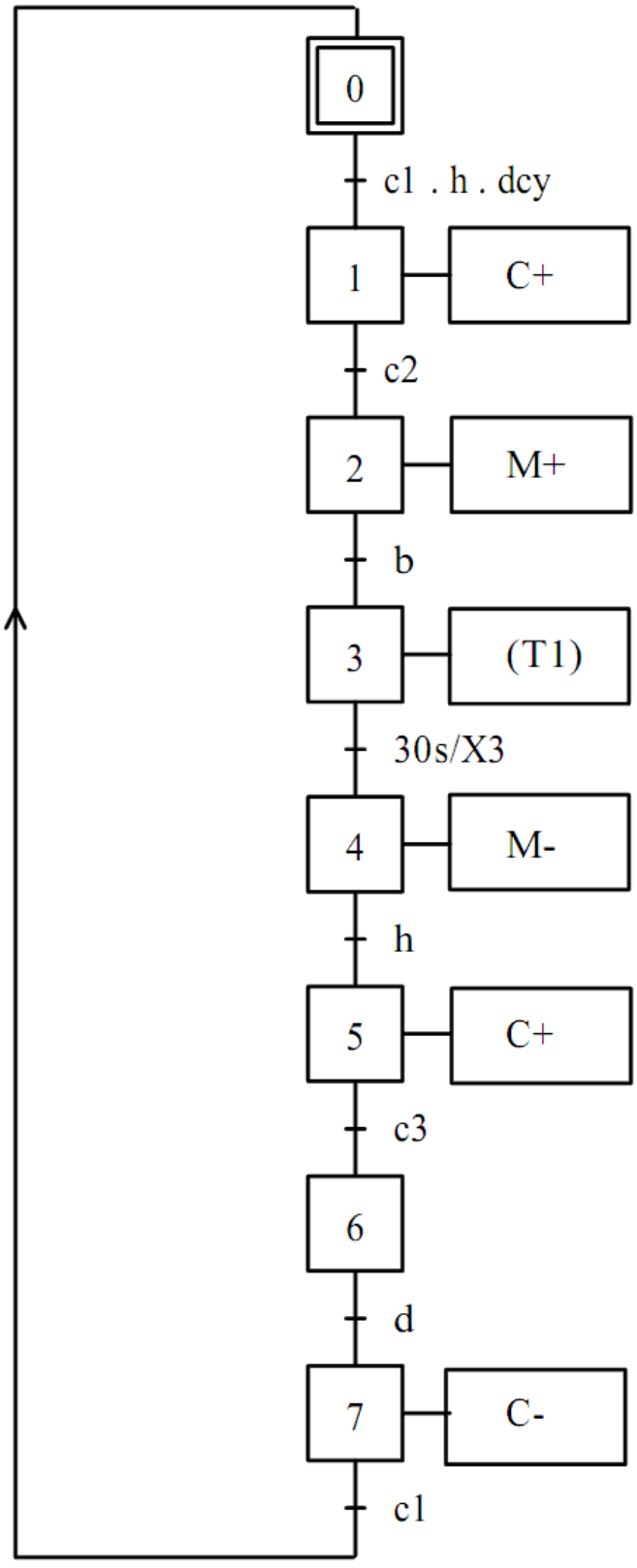
III.2.3.a. Le grafcet point de vue système



III.2.3.b. Le grafcet point de vue Partie Opérative



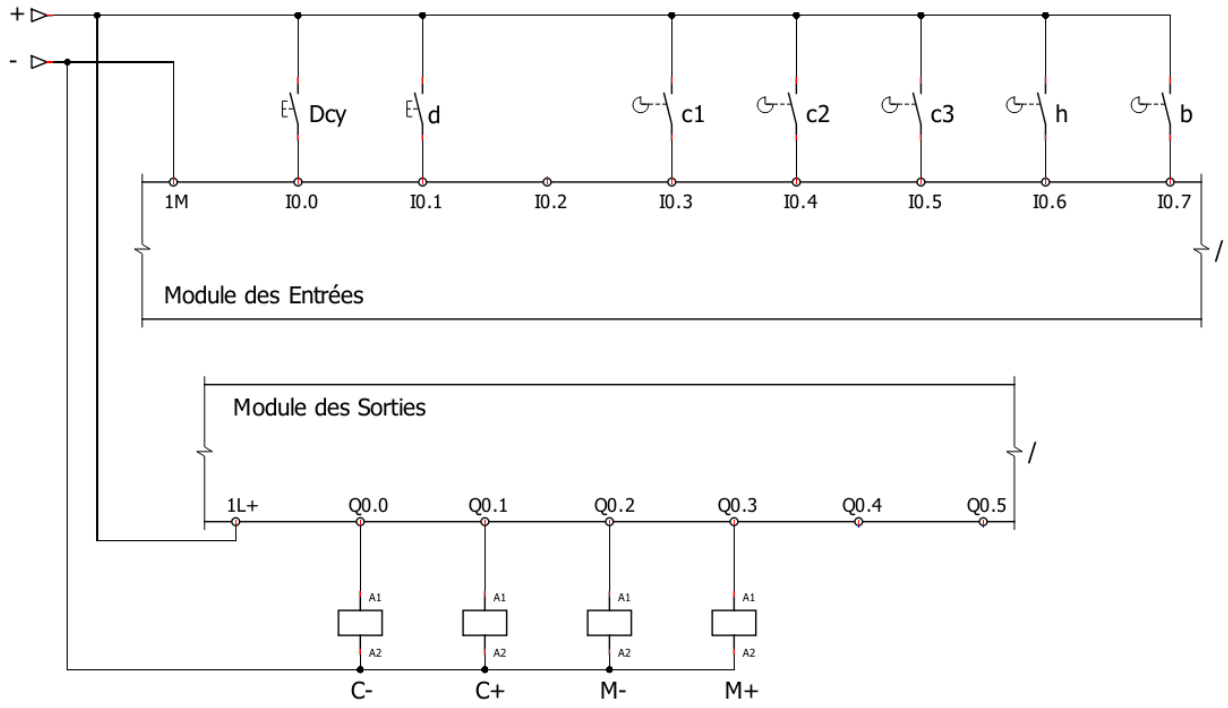
III.2.3.c.Le grafcet point de Partie commande



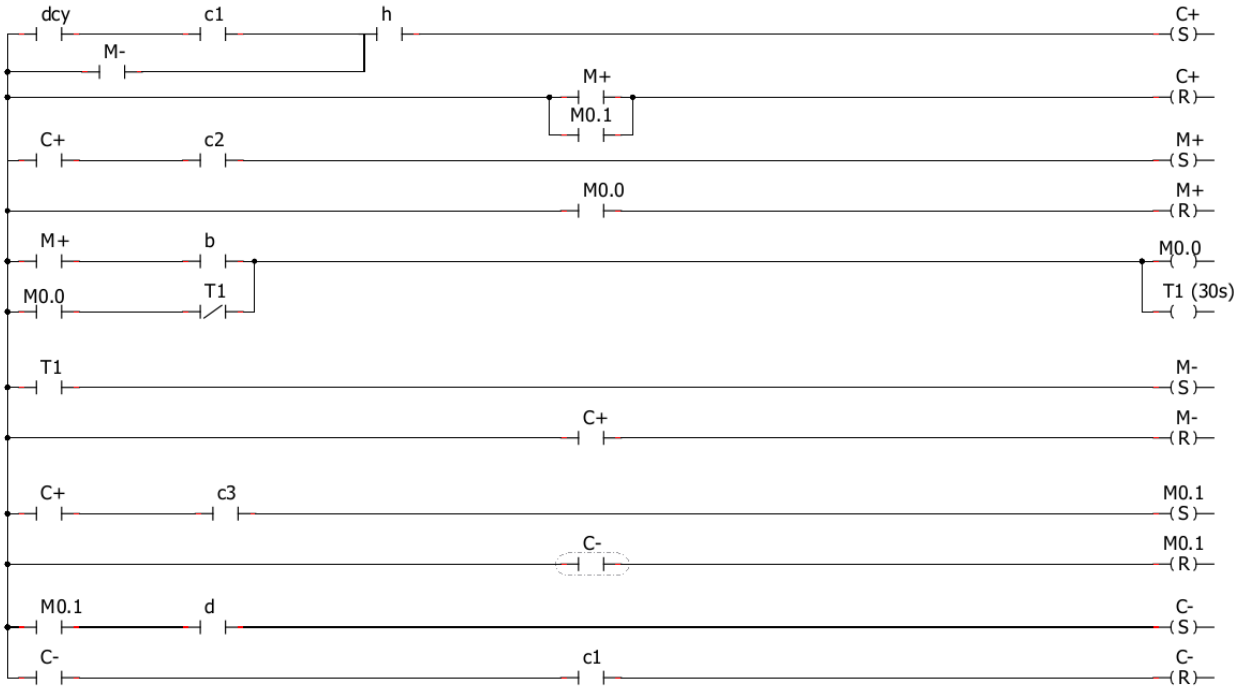
III.2.3.d. Tableau des adresses des entrées et des sorties

Entrées	Sorties
Dcy ----------> I 0.0	**C^- ----------> Q 0.0**
d ----------> I 0.1	**C^+ ----------> Q 0.1**
c1 ----------> I 0.3	**M^- ----------> Q 0.2**
c2 ----------> I 0.4	**M^+ ----------> Q 0.3**
c3 ----------> I 0.5	
h ----------> I 0.6	
b ----------> I 0.7	

III.2.3.e.Câblage des entrées et des sorties



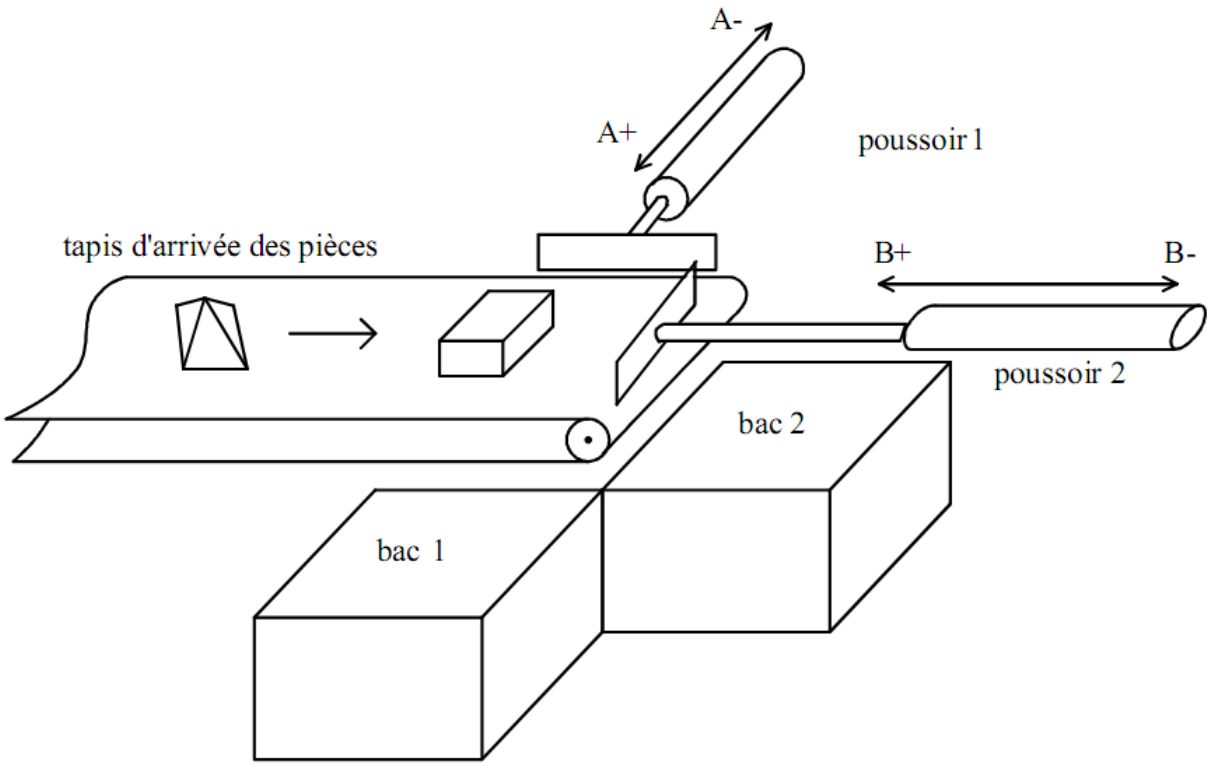
III.2.3.f. Programme Ladder



(S) = mise à 1 , (R) = mise à 0
M0.0 : mémoire pour l'étape du temporisateur
M0.1 : etape d'attente pour le déchargement

III.3. Exercice 03 : TRI DE PIÈCES

III.3.1. Cahier des charges



III.3.2. Cycle de fonctionnement :

- Quand le système est en fonctionnement (bouton bistable m à 1) le tapis apporte une pièce.
- Quand la pièce est contre le poussoir 2, on a 2 possibilités :
- Si la pièce est pyramidale, le poussoir 1 la pousse dans le bac 1.
- Si la pièce est prismatique, le poussoir 2 se recule et le tapis fait tomber la pièce dans le bac 2.

Remarque : On ne tiendra pas compte du fonctionnement du tapis pour les GRAFCET point de vue PO et PC.

Les capteurs utilisés sont les suivants :

- Poussoir 4 et 2 rentrés : a0 et b0
- Poussoir 1 et 2 sortis : a1 et b1

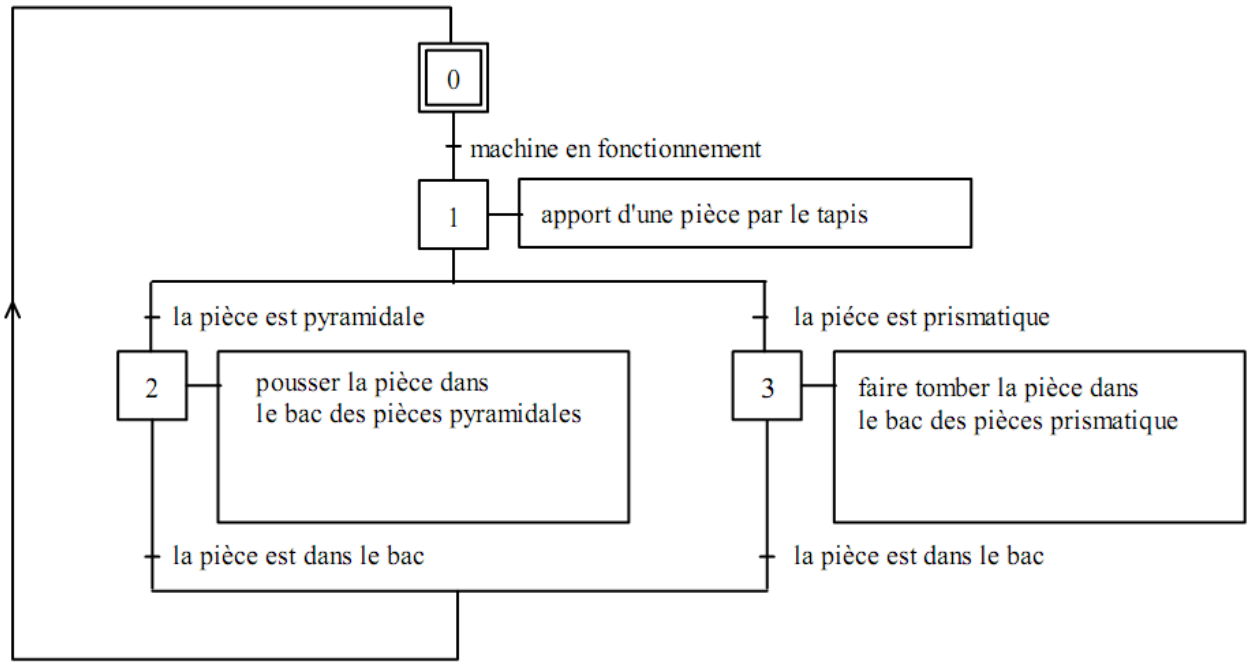
- Pièce pyramidale contre le poussoir 2 : t
- Pièce prismatique contre le poussoir 2 : p
- Pièce tombée dans le bac 2 : b2

Donner :

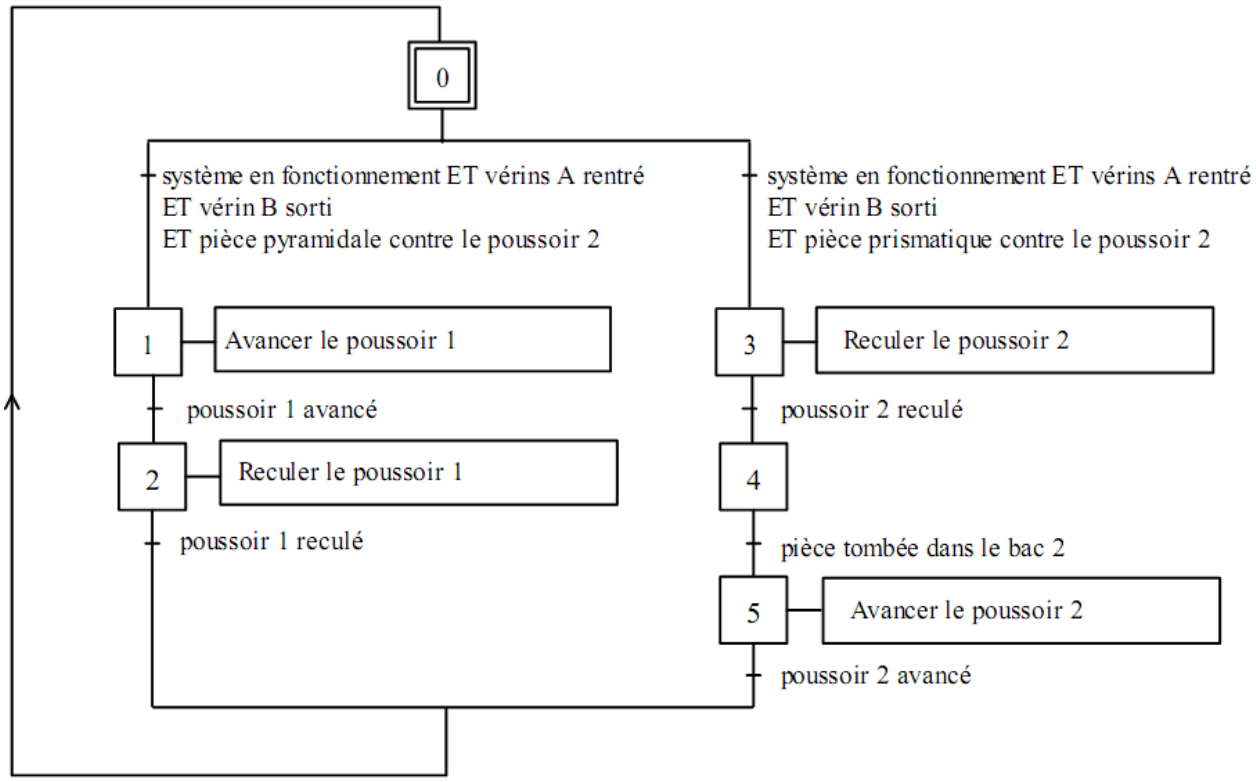
1. Le grafcet point de vue système
2. Le grafcet point de vue Partie Opérative
3. Le grafcet point de vue Partie Commande

III.3.3. Solution :

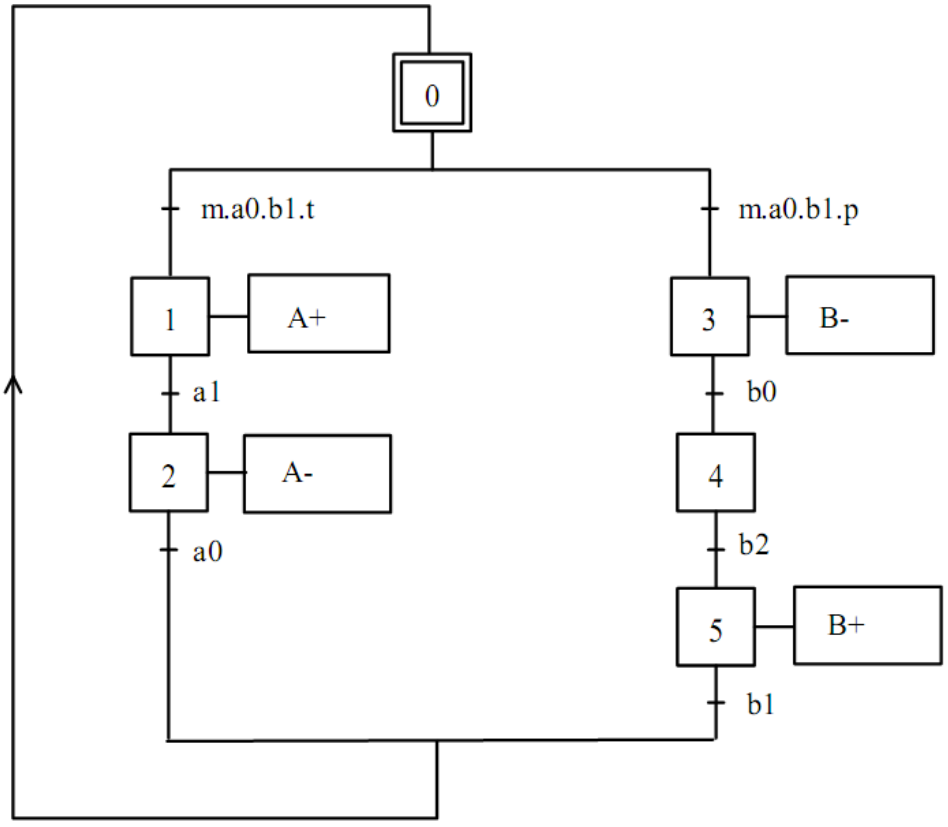
III.3.3.a. Grafcet point de vue système :



III.3.3.b. Grafcet point de vue Partie Opérative :



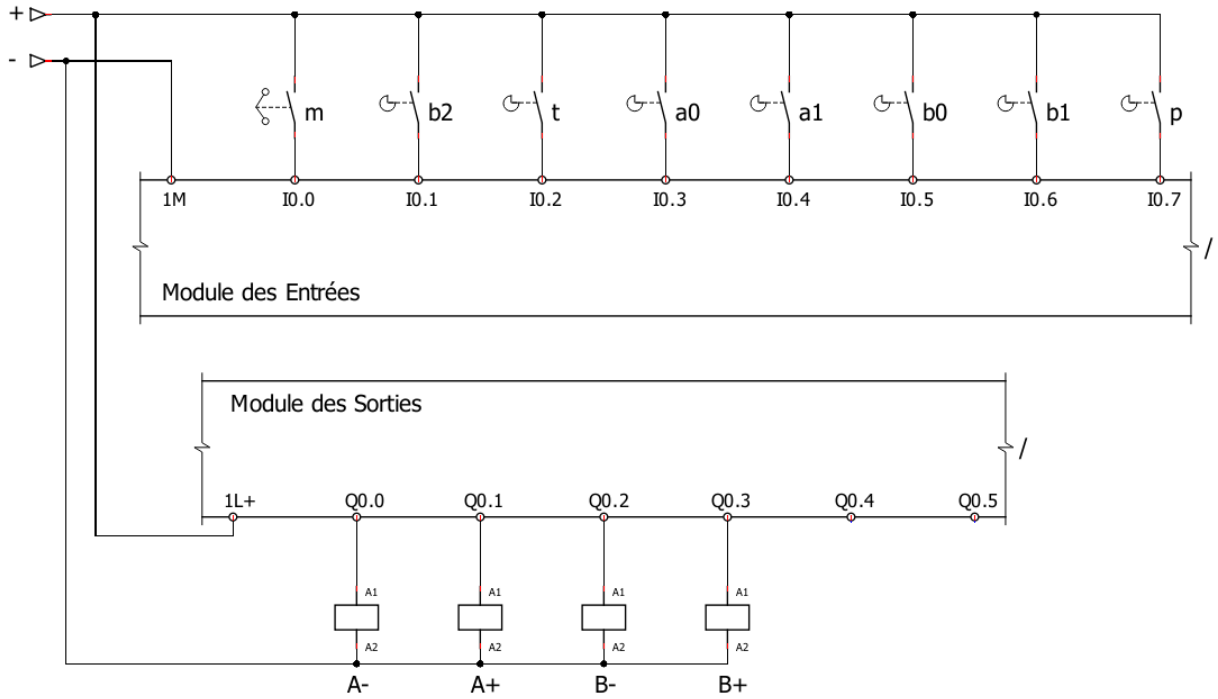
III.3.3.c.Grafcet point de vue Partie Commande :



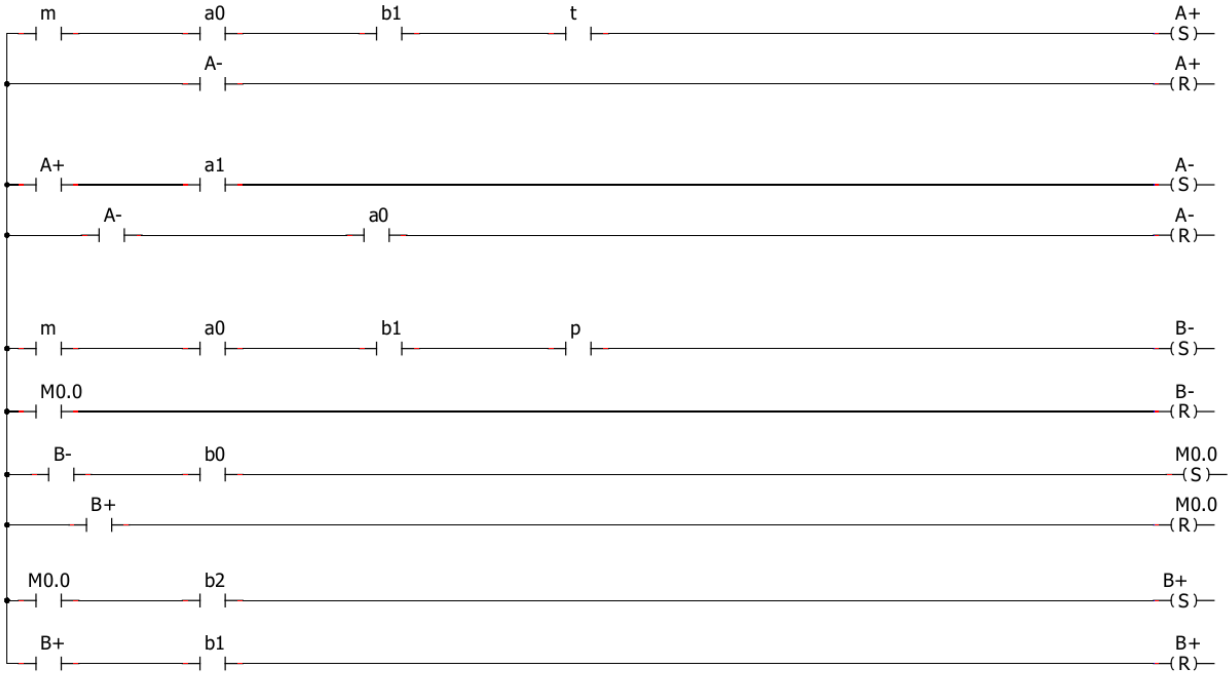
III.3.3.d. Tableau des adresses des entrées et des sorties

Entrées	Sorties
m ----------> I 0.0	**A^- ----------> Q 0.0**
b2----------> I 0.1	**A^+ ----------> Q 0.1**
t ----------> I 0.2	**B^- ----------> Q 0.2**
a0 ----------> I 0.3	**B^+ ----------> Q 0.3**
a1 ----------> I 0.4	
b0 ----------> I 0.5	
b1 ----------> I 0.6	
p ----------> I 0.7	

III.3.3.e.Câblage des entrées et des sorties



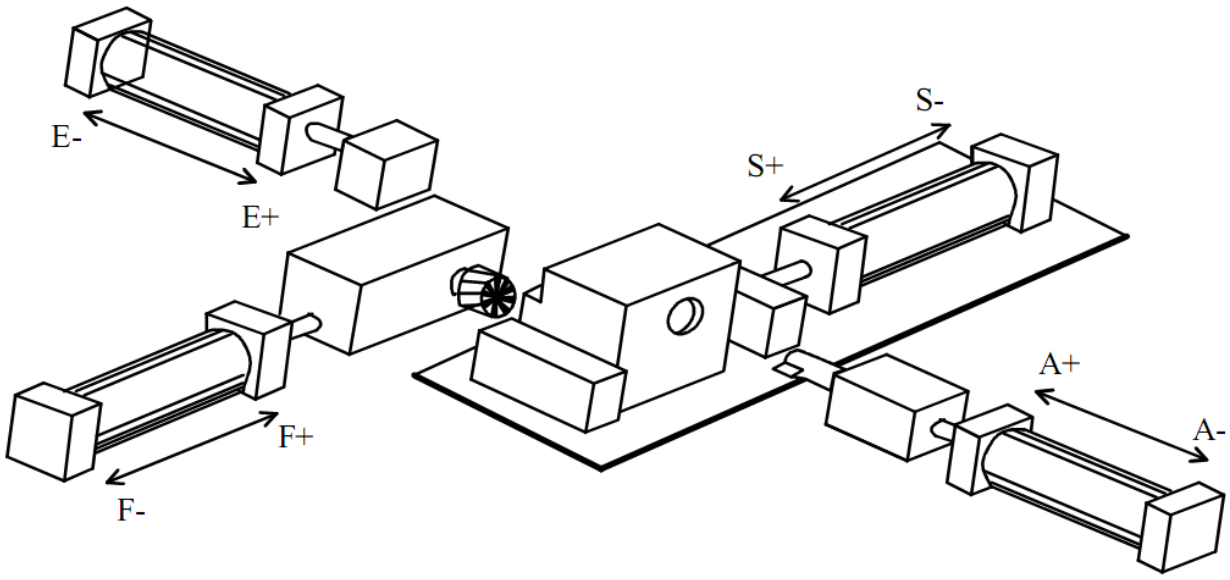
III.3.3.f. Programme Ladder



(S) = mise à 1 , (R) = mise à 0
M0.0 : étape 4

III.4. Exercice 04 : MACHINE SPÉCIALE D'USINAGE

III.4.1. Cahier des charges



III.4.2. Cycle de fonctionnement :

Si on appuie sur le bouton de départ cycle (dcy) quand les têtes d'usinages sont en position arrière, que les vérins d'éjection et de serrage sont reculés et qu'une pièce est présente, le système serre la pièce. On effectue alors simultanément les deux usinages.

- Le fraisage : la fraise avance en vitesse lente puis recule en vitesse rapide.
- Le lamage :
  - Le grain d'alésage avance en vitesse lente.
  - Une fois en fin de lamage on attend 1 seconde pour avoir un fond plat.
  - Le retour s'effectue alors en vitesse rapide.

  Après cela la pièce est desserrée puis éjectée par le vérin E.

  Remarques :

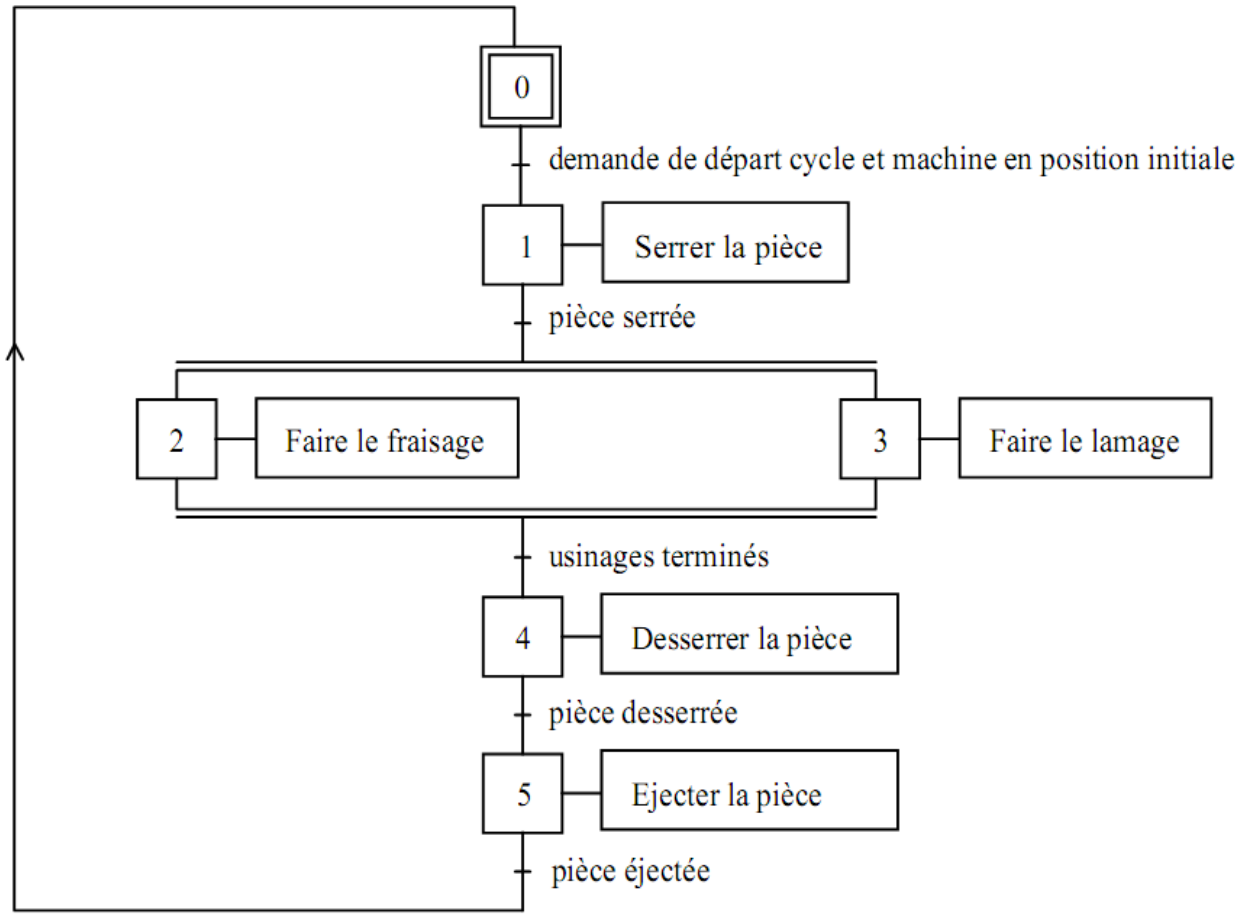
- Pour des raisons de simplicité, on ne tiendra pas compte du fonctionnement des moteurs de broches d'usinages.
- Les vérins A, F et S sont des vérins double effet commandés par des distributeurs bistables.
- Le vérin E est un vérin double effet commandé par un distributeur monostable.
- Les capteurs de contrôle des mouvements sont :
  - a0 et a1 pour le vérin d'alésage.
  - e0 et e1 pour le vérin d'éjection.
  - f0 et f1 pour le vérin de fraisage.
  - s0 et s1 pour le vérin de serrage.
- Le capteur de présence pièce fonctionne comme suit :
  - p = 1 : il y a une pièce dans le montage.
  - p = 0 : il n'y a pas de pièce dans le montage.

Donner :

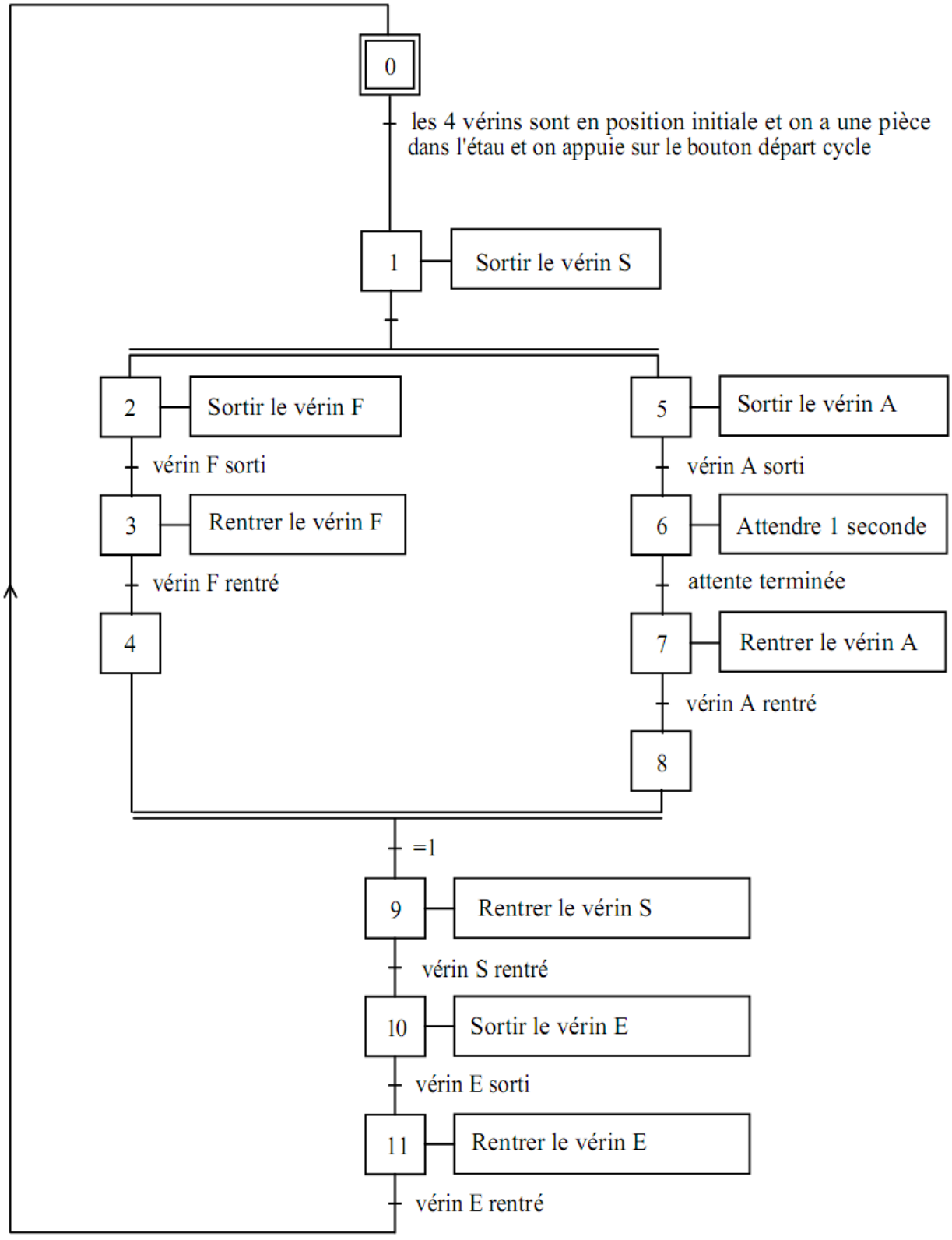
1. Le grafcet point de vue système.
2. Le grafcet point de vue Partie Opérative.
3. Le grafcet point de vue Partie Commande.

III.4.3. Solution :

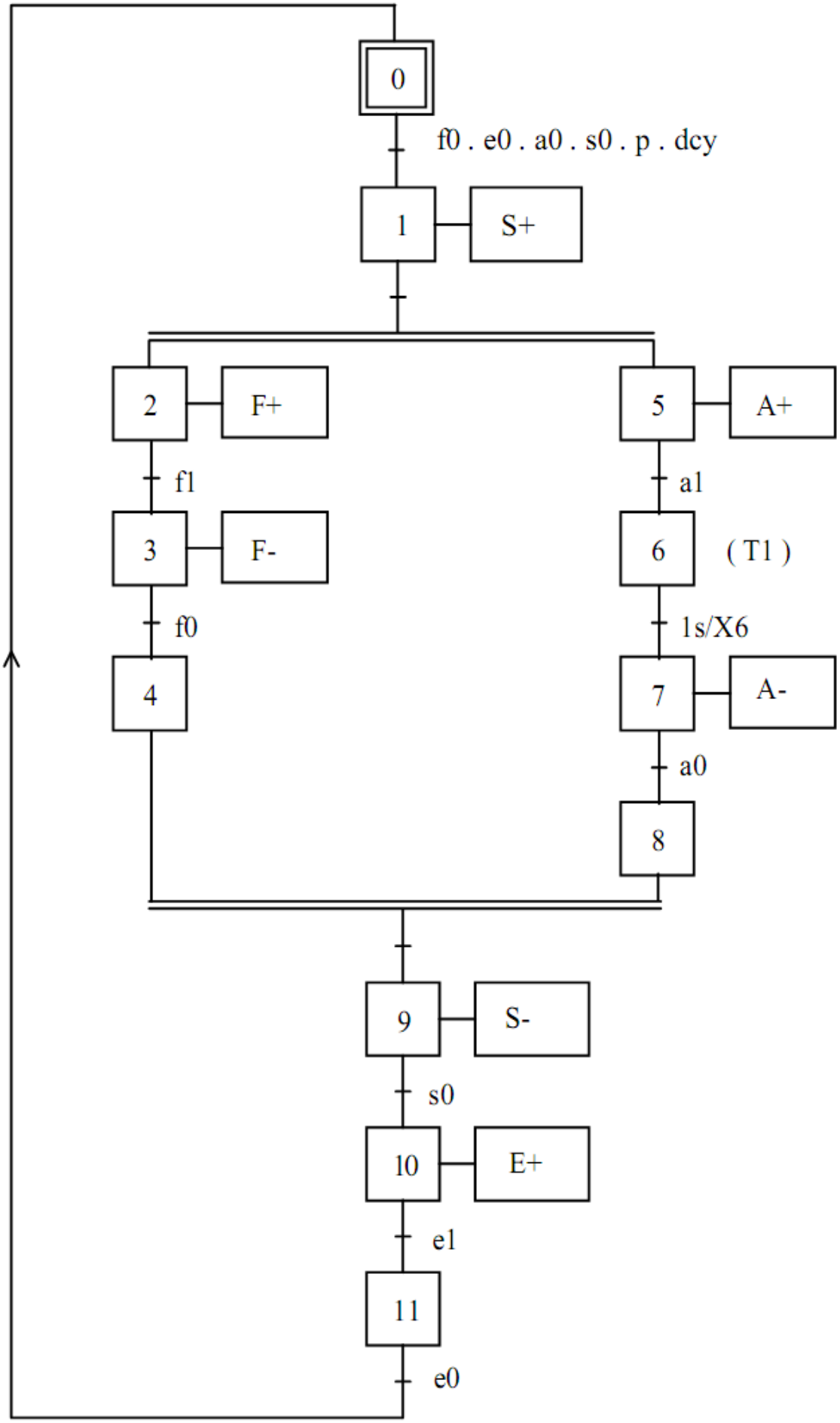
III.4.3.a. Grafcet point de vue système



III.4.3.b. Grafcet point de vue Partie Opérative

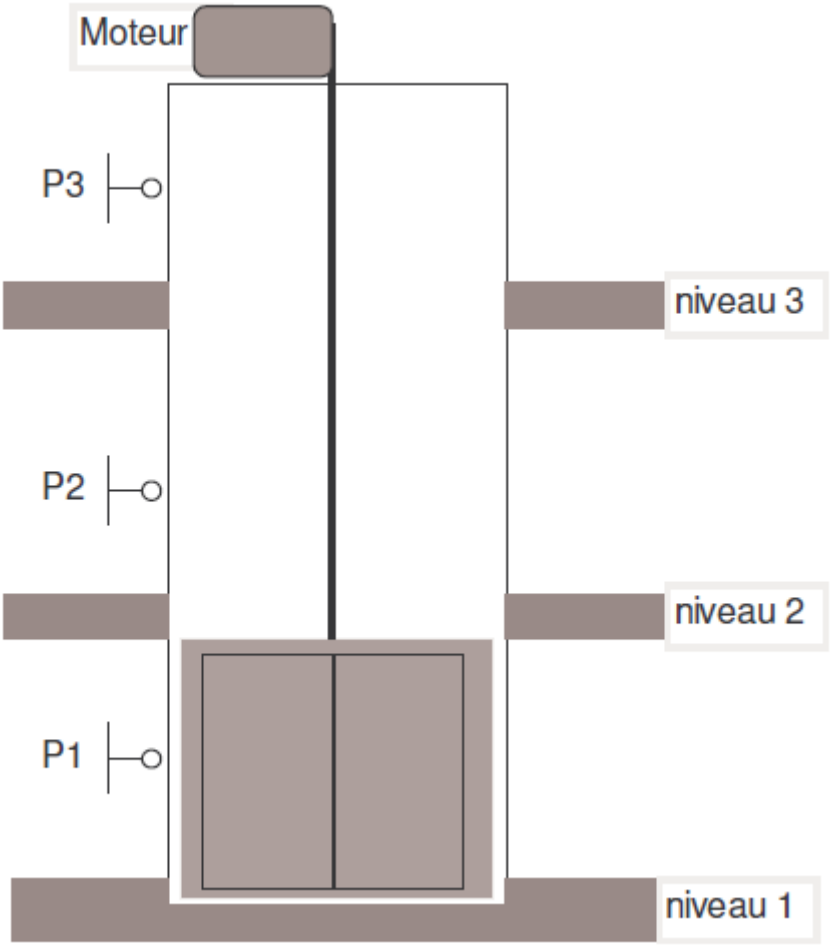


Grafcet point de vue Partie Commande


0
f0 . e0 . a0 . s0 . p . dcy
1
S+
2
F+
5
A+
f1
a1
3
F-
6
(T1)
f0
1s/X6
4
7
A-
a0
8
9
S-
s0
10
E+
e1
11
e0

III.5. Exercice 05 : Un monte-charge

III.5.1. Cahier des charges



III.5.2. Cycle de fonctionnement

Un monte-charge, programmé pour desservir régulièrement les trois niveaux d'une société, se trouve à la mise sous tension au niveau 1, les portes ouvertes. L'opérateur lance le cycle en appuyant sur un bouton de départ cycle Dcy. Il y a alors, au bout d'un temps T0 de 5s, la fermeture des portes, la montée de la cabine jusqu'au niveau 2 puis l'ouverture des portes. Il y séjourne pendant un temps T1 de 5mn. Enfin il monte au niveau 3, y reste pendant un temps T2 de 5mn avant de redescendre au niveau 1 en position initiale.

III.5.3. Désignation des préactionneurs :

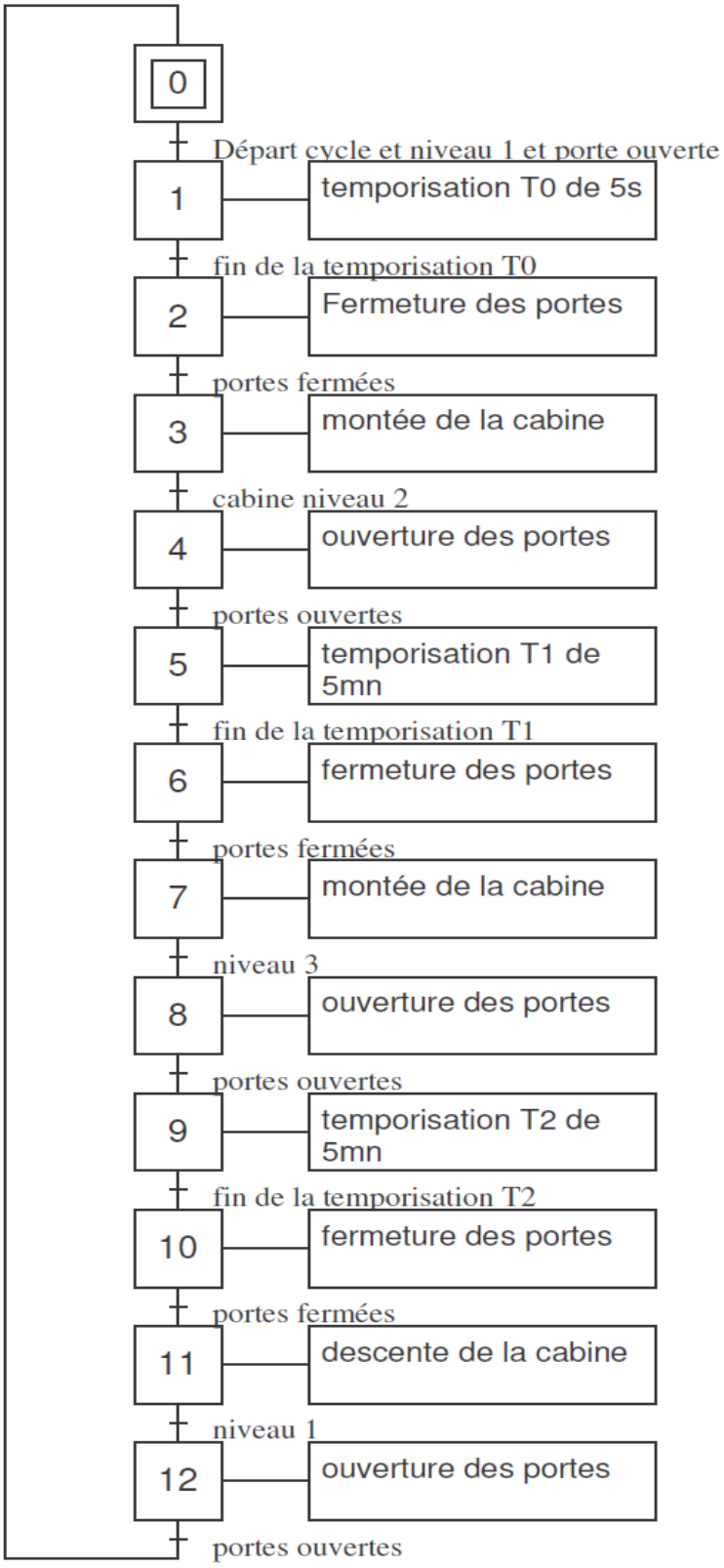
- OU : ouverture des portes
- FER : fermeture des portes
- KMH : contacteur moteur déplacement vers le haut
- KMB : contacteur moteur déplacement vers le bas
- P1 : niveau 1
- P2 : niveau 2
- P3 : niveau 3
- Dcy : départ du cycle
- PO : portes ouvertes
- PF : portes fermées.

Donner :

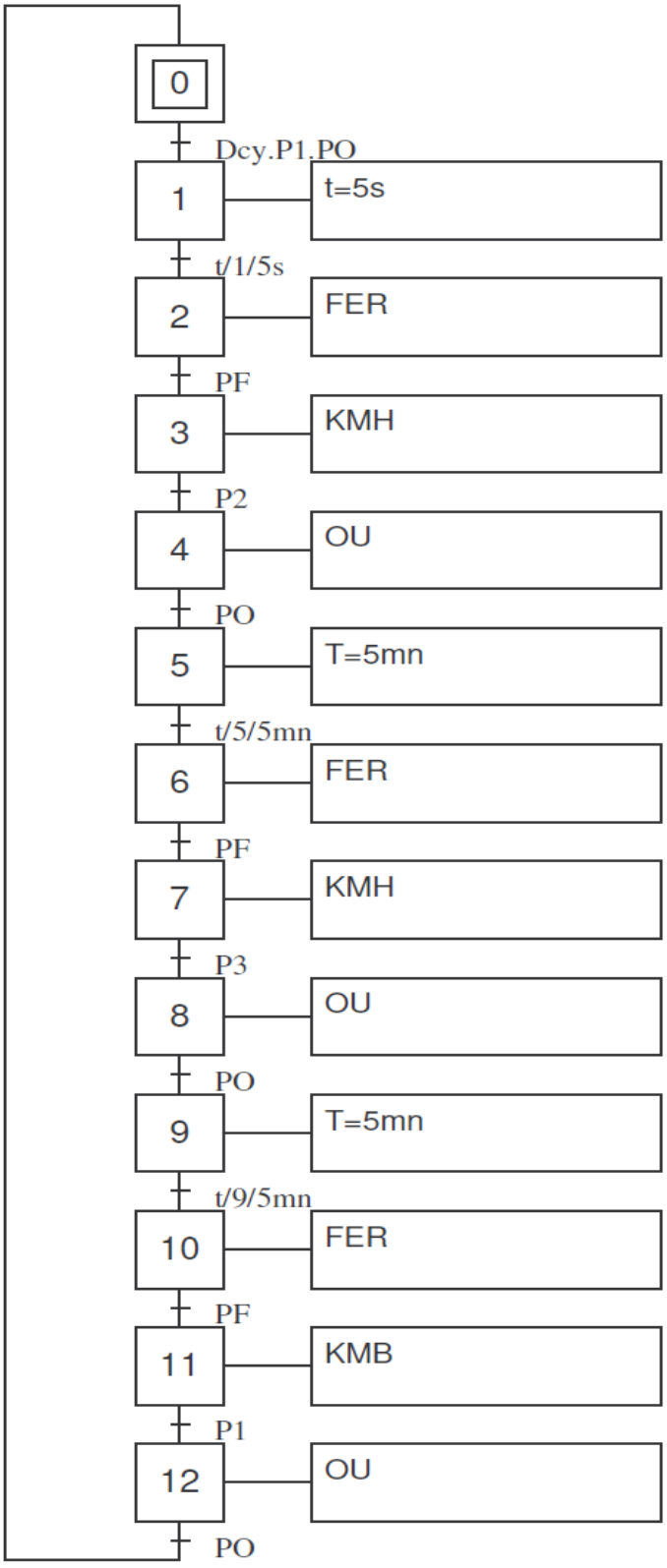
a) Chercher le GRAFCET de spécifications fonctionnelles correspondant à ce fonctionnement en prenant soin de respecter les règles d'évolutions.
b) Chercher le GRAFCET de spécifications technologiques.

III.5.4. Solution :

III.5.4.a. GRAFCET FONCTIONNEL :

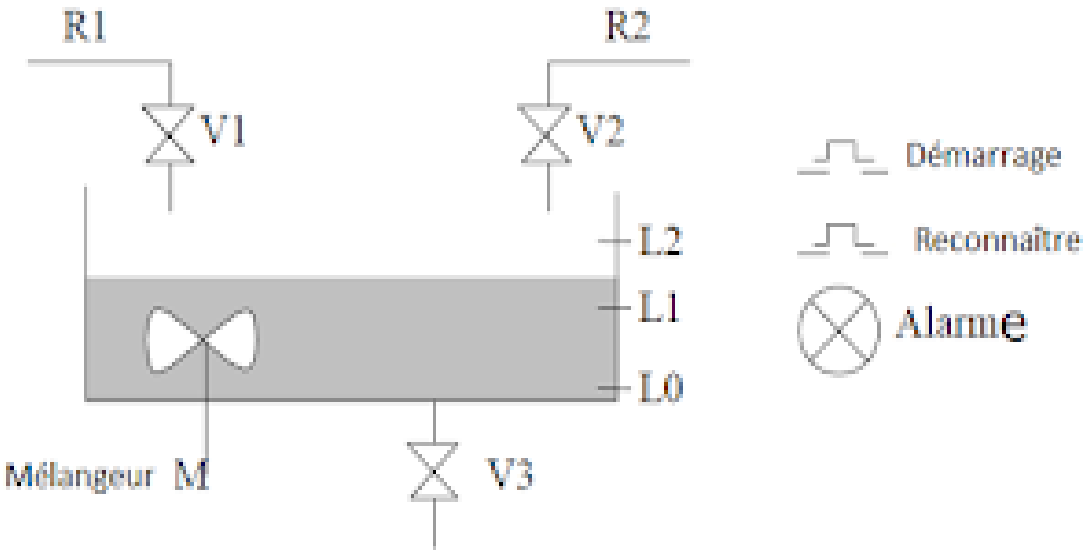


III.5.4.b. GRAFCET TECHNOLOGIQUE



III.6. Exercice 06 : Un processus chimique

III.6.1. Cahier des charges



Considérons un processus chimique simple qui combine deux réactifs pour produire le résultat final. Il pourrait fonctionner en versant d'abord suffisamment d'un réactif dans un récipient pour atteindre un niveau particulier, puis en versant suffisamment de deuxième réactif jusqu'à ce qu'un second niveau soit atteint (tout en mélangeant les deux réactifs), puis en versant le produit.

Abréviations

Niveaux : L0, L1, L2

Vannes : V1, V2, V3

Réactifs : R1, R2

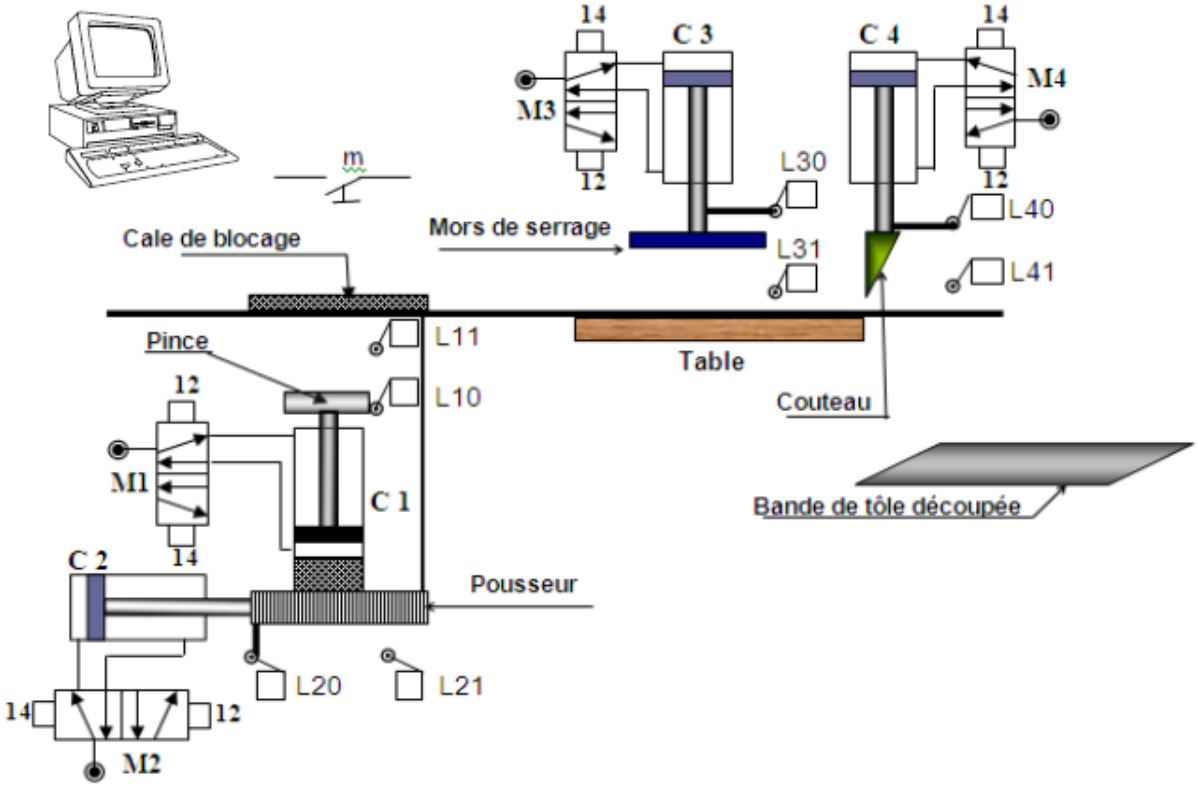
III.6.2. La séquence d'opérations souhaitée est la suivante :

a) Lorsque vous appuyez sur le bouton de démarrage, V1 doit être ouvert jusqu'à ce que le niveau L1 soit atteint.
b) Lorsque L1 est atteint, le mélangeur doit commencer à mélanger et simultanément V2 doit être ouvert.
c) Lorsque L2 est atteint, le mélangeur doit s'arrêter, V3 devrait être ouvert jusqu'à ce que le niveau du réservoir passe sous L0.

d) Si après 10 minutes le niveau du réservoir n'est pas sous L0, une alarme est déclenchée. Le bouton "acquitter" arrête l'alarme et permet de redémarrer le processus de contrôle.

III.7. Exercice 07 :

III.7.1. Poste automatique de découpage de la tôle



III.7.2. Fonction du système

Le système sert à découper automatiquement la tôle avec des longueurs prédéterminées.

III.7.2.a. Description de fonctionnement

L'action sur le bouton de mise en marche « m » enclenche le cycle de fonctionnement suivant :

- Bloquer la tôle par un dispositif formé par (vérin C1+Pince + cale) appelé unité de blocage.
- pousser la tôle par un dispositif formé par (vérin C2+ Pousseur) appelé unité de poussage.
- serrer la tôle par un dispositif formé par (vérin C3+ Mors de serrage) appelé unité de serrage.
- découper la tôle par un dispositif formé par (vérin C4+ Mors de serrage) appelé unité de découpage.

L'opération de découpage prend fin dés que le capteur L41 est actionné ensuite le cycle recommence.

4- Le Système est commandé par un automate S7-224 Siemens :

Le dialogue Homme /Système est assuré à l'aide d'un Pupitre d'exploitation Comportant (écran + clavier +boutons +voyants +Souris).

Le système est constitué par :

- Quatre vérins pneumatiques C1, C2, C3 et C4.
- Quatre Distributeurs pneumatiques Bistables M1, M2, M3 et M4

Huit capteurs de position pneumatiques L10, L11, L20, L21, L30, L31, L40, L41

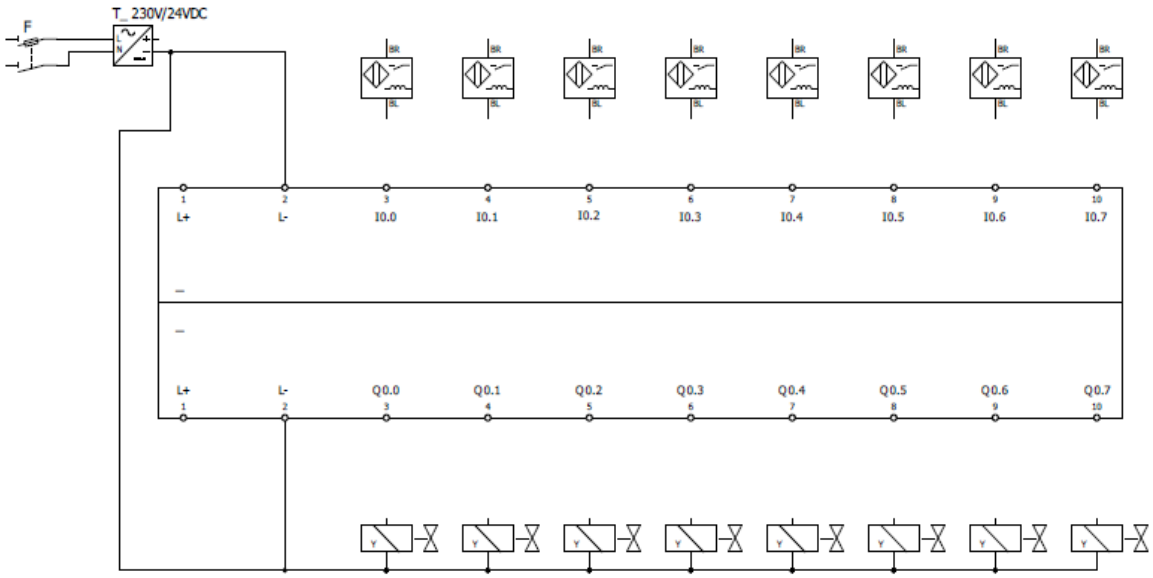
Remarque : Bistable : le distributeur garde sa position en l'absence de signal de pilotage (fonction mémoire).

III.7.2.b. Travail demandé :

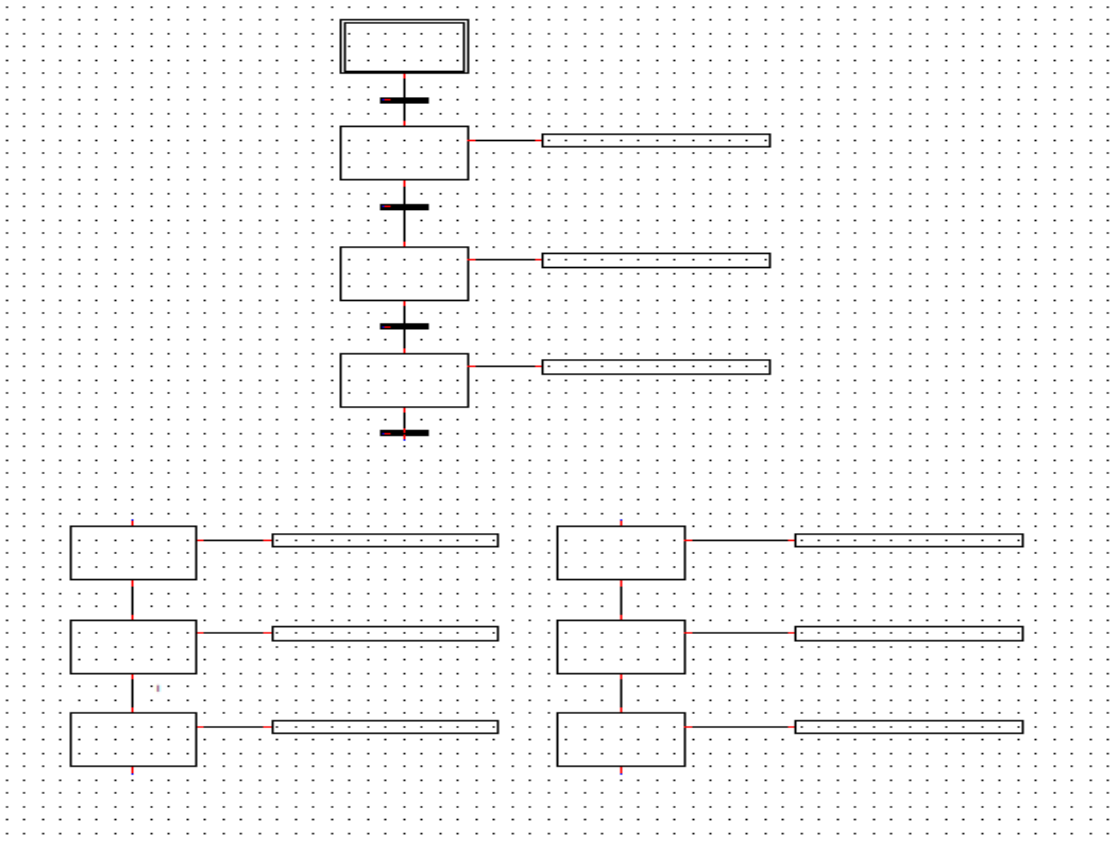
1- Compléter ce tableau avec les adresses d'E/S et les bits internes que vous utilisez.

Entrées	Adresses	Sorties	Adresses
m	I1.0		

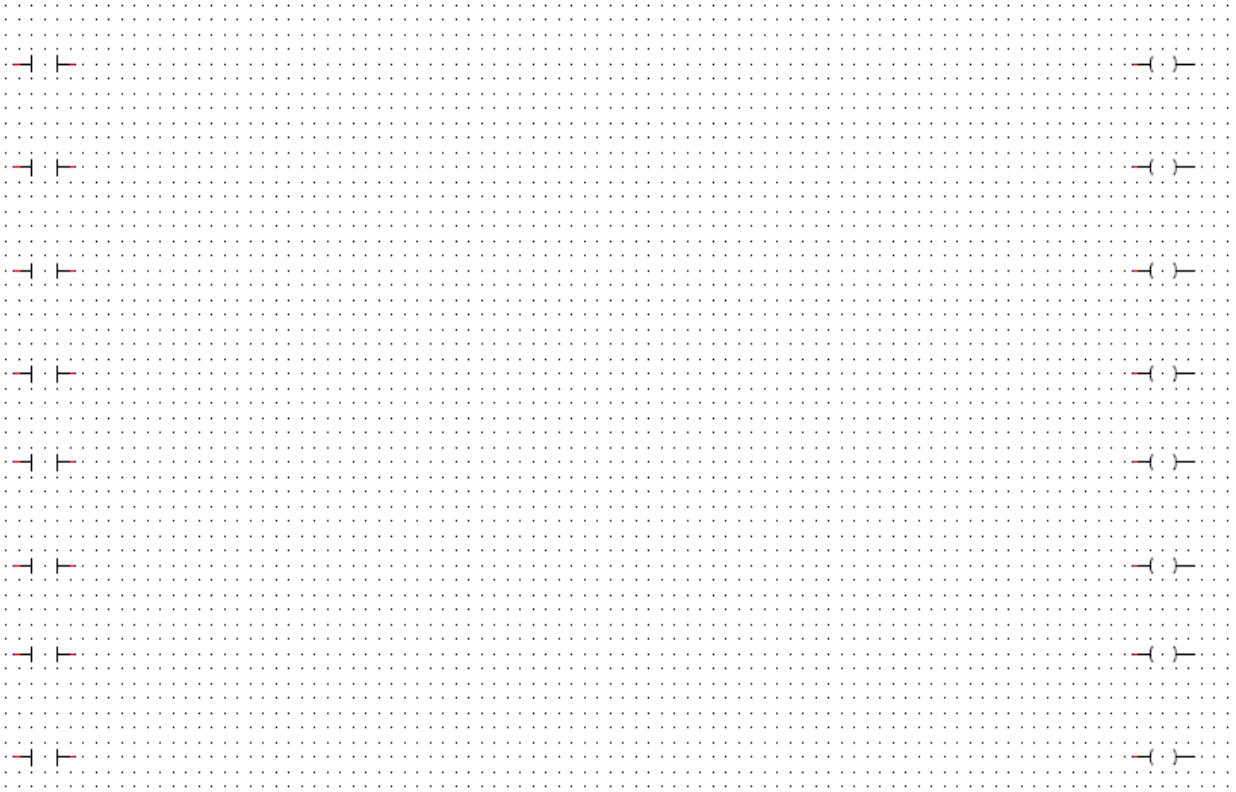
2- Faire un schéma de raccordement des entrées et sorties sur l'automate.



3- Compléter le GRAFCET



4- Construire votre programme selon votre câblage.



IV. Références Bibliographiques

[1] Progressez avec les microcontrôleurs PIC, Gérard Samblancat, Ed. Dunod, 2006

[2] Programmation en C des PIC, Christian Tavernier, Ed. Dunod, 2006

[3] Microcontrôleurs AVR : Description et mise en oeuvre, Christian Tavernier, Ed. Dunod, 2009

[4] Advanced PIC microcontroller projects in C, Dogan Ibrahim, Ed. Elsevier, 2008

[5] Microcontrollers in C, T. V. Sickle, Ed. LLH Publishing, 2001.

[6] Programmable Logic Controllers: Programming Methods and Applications, John R. Hackworth and Frederick D. Hackworth, Jr.

[7] Automates programmables industriels, traduction de Hervé Soulard, 2e édition, 2015