TP AUTOMATISME

Automate Programmable Industriel(API TSX 27)

Encadré par:
M. KHATORY
Réalisé par:
Saoussane MAATI / Youssef LIMY

Avant propos :

Les automatismes nécessitent un grand nombre d'entrées sorties le système de relayage devenant trop complexe, on le remplace par un système de logique programmée appelée automate programmable.

I : Automatisation

L'automatisation de la production consiste à transférer tout ou partie des tâches de coordination, auparavant exécutées par des opérateurs humains, dans un ensemble d'objets techniques appelé PARTIE COMMANDE.

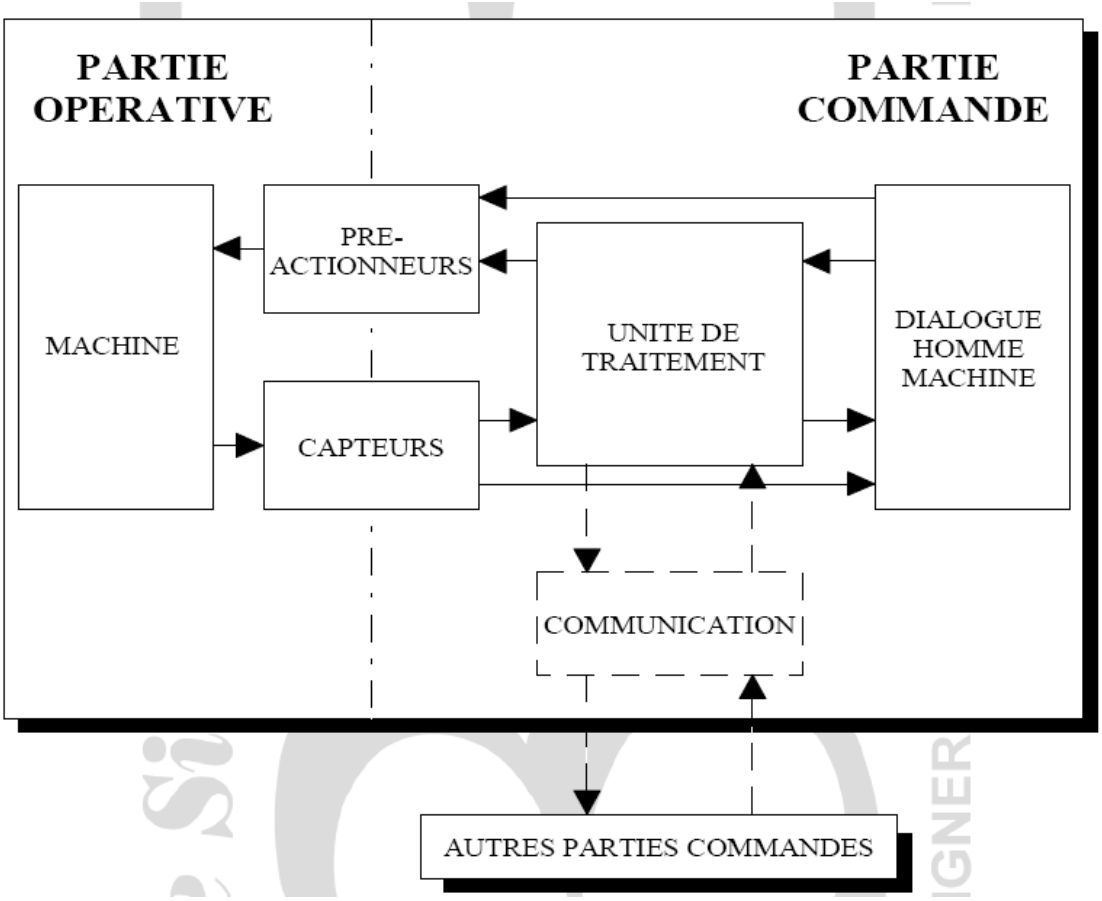
La Partie Commande mémorise le SAVOIR FAIRE des opérateurs pour obtenir la suite des actions à effectuer sur les matières d'œuvre afin d'élaborer la valeur ajoutée.

Elle exploite un ensemble d'informations prélevées sur la Partie Opérative pour élaborer la succession des ordres nécessaires pour obtenir les actions souhaitées.

I-1 : Structure d'un système automatisé

Tout système automatisé COMPORTE :

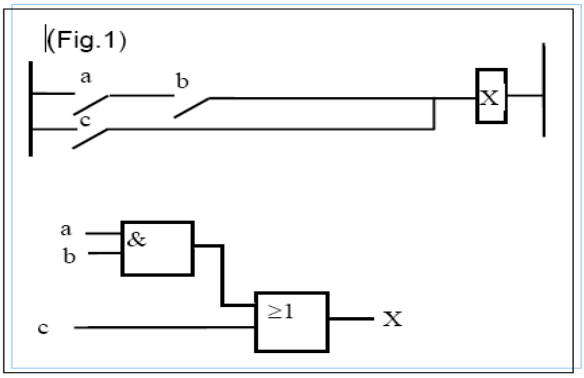
- Une **PARTIE OPERATIVE** (P.O.) : procédant au traitement des matières d'œuvre afin d'élaborer la valeur ajoutée,
- Une **PARTIE COMMANDE** (P.C.) : coordonnant la succession des actions sur la Partie Opérative avec la finalité d'obtenir cette valeur ajoutée.



Remarque

Pour la partie commande d'un automatisme le concepteur à le choix entre deux familles de solutions : la logique câblée et la logique programmé.

A : La logique câblée :

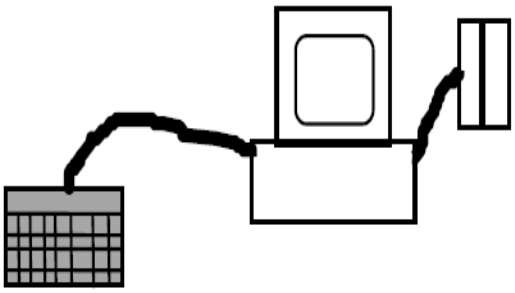


Exemple: X = (a.b)+c

L'automatisme est obtenu en reliant entre eux les différents constituants de base ou fonctions logiques par câblage.

La logique câblée correspond à un traitement parallèle de l'information. Plusieurs constituants peuvent être sollicités simultanément.

B : La logique programmée.



Elle correspond à une démarche séquentielle, seule une opération élémentaire est exécuté à la fois, c'est un traitement série.

Le schéma électrique est transcrit en une suite d'instruction qui constitue le programme.

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.

II : automate programmable industriel

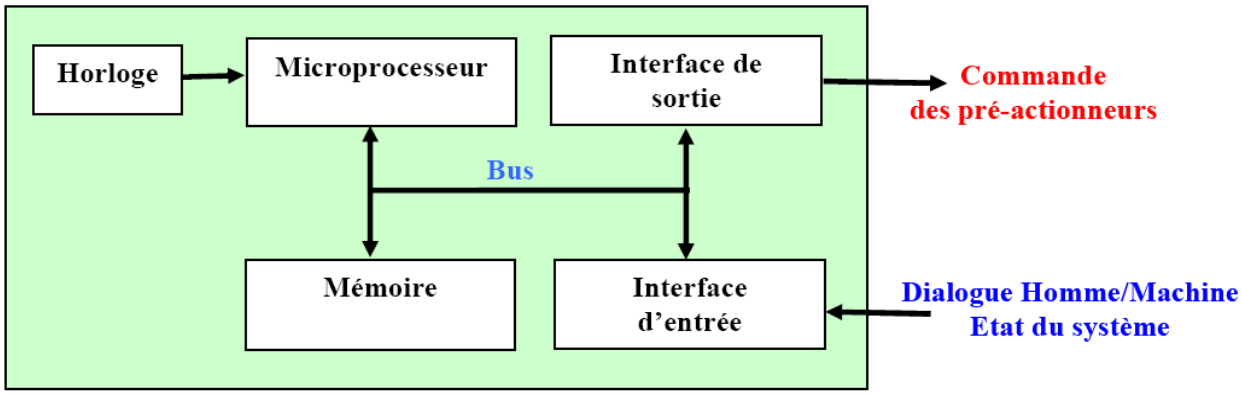
1 : Définition

Appareil électronique qui comporte une mémoire programmable par un utilisateur automaticien (et non informaticien) à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme par exemple :

- Logique séquentielle et combinatoire
- Temporisation, comptage, décomptage et comparaison
- Calcul arithmétique
- Réglage, asservissement, régulation, etc, pour commander, mesurer et contrôler au moyen d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de machines ou de processus, en environnement industriel.

2 : Architecture d'un API

La structure interne d'un API peut se représenter comme suit :



L'automate programmable **reçoit** les informations relatives à l'état du système et puis **commande** les pré-actionneurs suivant le programme inscrit dans sa mémoire.
Un API se compose donc de trois grandes parties :
- Le processeur ;
- La zone mémoire ;
- Les interfaces Entrées/Sorties.

Le microprocesseur :

Le **microprocesseur** réalise toutes les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul... à partir d'un programme contenu dans sa **mémoire**.

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

La zone mémoires :

a)- La Zone mémoire va permettre :

- De recevoir les informations issues des capteurs d'entrées.
- De recevoir les informations générées par le processeur et destinées à la commande des sorties (valeur des compteurs, des temporisations, …).
- De recevoir et conserver le programme du processus.

b)-Action possible sur une mémoire :

- ***ECRIRE*** pour modifier le contenu d'un programme
- ***EFFACER*** pour faire disparaître les informations qui ne sont plus nécessaires
- ***LIRE*** pour en lire le contenu d'un programme sans le modifier

c)- Technologie des mémoires :

- ***RAM*** (Random Acces Memory): mémoire vive dans laquelle on peut lire, écrire et effacer (contient le programme)
- ***ROM*** (Read Only Memory): mémoire morte accessible uniquement en lecture.
- ***EPROM*** : mémoires mortes reprogrammables effacement aux rayons ultra-violets.
- ***EEPROM*** : mémoires mortes reprogrammables effacement électrique

Remarque :

La capacité mémoire se donne en mots de 8 BITS (Binary Digits) ou octets.

Les interfaces d'entrées/sorties :

Les **entrées** reçoivent des informations en provenance des **éléments de détection (capteurs)** et du **pupitre opérateur (BP).**

Les **sorties** transmettent des informations aux **pré-actionneurs (relais, électrovannes …)** et aux **éléments de signalisation (voyants)** du pupitre.

a)- Interfaces d'entrées :

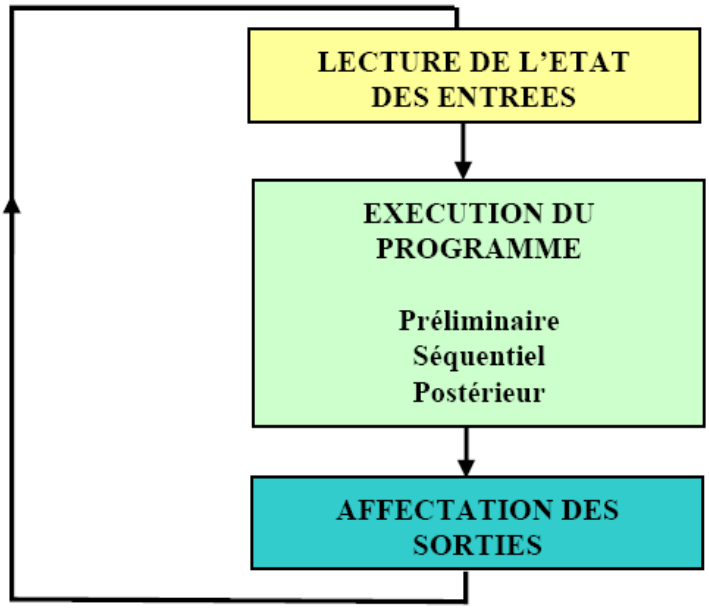
Elles sont destinées à :
- Recevoir l'information en provenance des capteurs.
- Traiter le signal en le mettant en forme, en éliminant les parasites et en isolant électriquement l'unité de commande de la partie opérative.

Donc pour commander une sortie automate l'unité de commande doit envoyer :
- Un **1** logique pour actionner une sortie API
- Un **0** logique pour stopper la commande d'une sortie API

3 : Fonctionnement api

L'automate programmable **reçoit** les informations relatives au système, **il traite** ces informations en fonction **du jeu d'instruction et modifie** l'état de ses sorties qui **commandent** les pré-actionneurs.

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

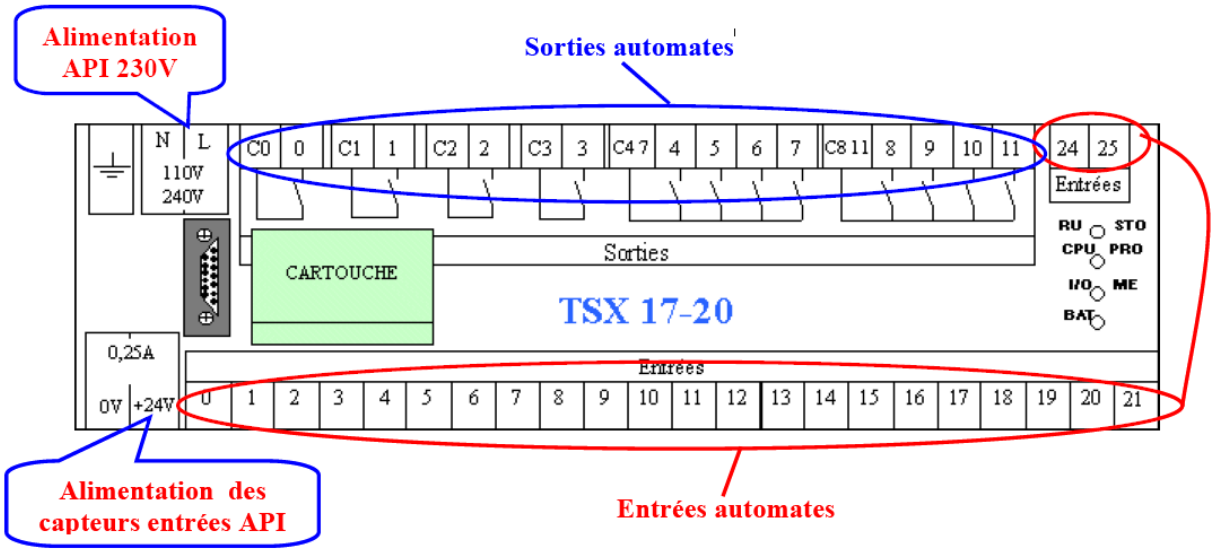


4 : Alimentation de l'apis

L'alimentation intégrée dans l'API, fournit à partir des tensions usuelles des réseaux **(230 V, 24 V=)** les tensions continues nécessaires au fonctionnement des circuits électroniques.

5 : Raccordement automate

Présentation



6 : Les fonctions de base d'un automate

Un automate programmable permet de remplacer une réalisation câblée comportant des composants combinatoires (portes) et séquentiels (bascules, séquenceurs,...) par un programme. Un programme est une suite d'instructions, qui sont exécutées l'une après l'autre. Si une entrée change alors qu'on ne se trouve pas sur l'instruction qui la traite et que l'on ne repasse plus sur ces instructions, la sortie n'est pas modifiée. C'est la raison de la nécessité de bouclage permanent sur l'ensemble du programme.

Par rapport à un câblage, on a donc deux désavantages : temps de réponse (un changement des entrées sera pris en compte au maximum après le temps d'un passage sur l'ensemble du programme, c'est ce qu'on appelle le temps de scrutation, qui sera souvent de l'ordre de la milliseconde) et non simultanéité (on n'effectue qu'un instruction à la fois). Mais ces temps étant en général très inférieurs aux temps de réaction des capteurs et actionneurs (inertie d'un moteur par exemple), ceci n'est que rarement gênant. L'avantage est que c'est programmable, donc facilement modifiable.

Tout automate programmable possède :

 Des entrées, des sorties, des mémoires internes : toutes sont binaires (0 ou 1), on peut les lire (c.a.d connaître leur état) (même les sorties), mais on ne peut écrire (modifier l'état) que sur les sorties et les mémoires internes. Les mémoires internes servent pour stocker des résultats temporaires, et s'en resservir plus tard.

 Des fonctions combinatoires : ET, OU, NON (mais aussi quelquefois XOR, NAND,...).

 Des fonctions séquentielles : bascules RS (ou du moins Set et Reset des bascules), temporisations, compteurs/décompteurs mais aussi quelquefois registres à décalage, etc...

 Des fonctions algorithmiques : sauts (vers l'avant mais aussi quelquefois saut généralisés), boucles, instructions conditionnelles...

 De plus il permet de créer, essayer, modifier, sauver un programme, quelquefois par l'intermédiaire d'une console séparable et utilisable pour plusieurs automates. Désormais cette fonctionnalité est également possible sur PC, permettant une plus grande souplesse, une assistance automatique, des simulations graphiques,... mais pour un prix supérieur.

Ce qui différencie les automates, c'est la capacité (entrées, sorties, mémoires internes, taille de programme, nombre de compteurs, nombre de temporisations), la vitesse mais surtout son adaptabilité (possibilité d'augmenter les capacités, de prendre en compte de l'analogique et numérique, de converser via un réseau...).

7 : Description des menus (utiles) sur la console T407

A - Menu principal [TSX 17-20]

 ADJ (adjust) : permet de visualiser ou modifier toute variable.

 DBG (debug) : mise au point : permet de visualiser le programme et voir l'état des capteurs, sorties, étapes actives... (Trait plein dans le programme si actif, interrompu si 0) et mettre des points d'arrêt dans le programme.

 PRG : créer ou modifier le programme.

 TRF (transfert) : pour mémorisation sur EEPROM et impression sur imprimante (RS232).

B - Menu PRG (dans tous les cas)

 CLM (clear memory) : efface le programme actuel, permet de définir si le nouveau programme sera en langage à contacts (LAD) ou Grafcet (SEQ).

 CNF (config) : configuration de l'automate, de la liaison RS232 pour l'imprimante (LINE), des bobines à sauvegarder même en cas de coupure de courant (SAV)...

 NAME : permet de donner un nom au programme.

 LK : vérifie si le programme en mémoire ne comporte pas d'erreur. .

 FREE : retasse le programme (à faire après de nombreuses modifications).

C - Menu PRG en mode ladder (LAD)

 TOP : aller au premier réseau.

 BOT : (bottom) aller après le dernier réseau (on passe ensuite au dernier par la flèche vers le haut).

 LAB : donner un numéro de réseau (label) puis [ENT] pour le visualiser.

 INS : insère un nouveau réseau (vide) devant le réseau actuel.

 DEL (delete) : supprime le réseau actuel.

 SCH (search) : permet de rechercher tous les réseaux comportant une bobine ou contact donné.

 [ZM] (zoom) : visualise l'adresse d'un contact ou bobine (exemple I1,2), on peut se déplacer dans le réseau par les flèches.

 [CLR] (clear) : retour au niveau supérieur (ZM->LAD->PRG->principal)

 [Quit] : retour direct au menu principal.

D - en mode ZOOM (sous PRG en mode LADDER)

 LAB : donner au réseau actuel un numéro de label (0 à 999).

 " " : donner un commentaire au réseau actuel (15 caractères maxi, sera affiché au dessus du réseau).

 MOD : permet de modifier l'élément pointé (l'effacer par [DEL] par exemple), on valide le réseau modifié par [ENT].

E - Menu PRG en mode GRAFCET

On dispose de 8 pages (0 à 7) de 14 lignes de 8 colonnes. On peut au maximum prendre en compte 96 étapes, les divergences et convergences sont limitées à 4 voies. L'écran ne montre qu'une petite partie de la page, mais le numéro de page (P), de ligne (L) et de colonne (C) sont toujours affichés. On se déplace par les flèches, ou en tapant P, L, C ou X (étape) suivi du numéro désiré. Les fonctions sont approximativement les mêmes qu'en mode ladder, hormis :

- **DLP** : effacement d'une page complète.
- **[ZM]** : face à une transition, la définit (si réseau vide, réceptivité toujours fausse).
- **[ZM]** : face à une étape, définit son action (étape d'attente si réseau vide).
- **MOVE** : déplace l'élément actuel (par les flèches) puis valider par [ENT].

F - Menu DBG

- **R/S** : passe de RUN à STOP et inversement (on peut aussi utiliser le contacteur sur la platine).
- **PRG** : visualiser le programme et l'état des variables (trait plein=1, pointillé=0), insertion de points d'arrêt.
- **CY/** : exécution cycle par cycle.
- **STP** : liste des étapes actives.
- **/L** : point d'arrêt sur un label, /o sur une étape.
- **S/L** et **S/o** : blocage sur un label ou une étape.

8 : Les grandes marques de l'api

9 : Quelques applications

Assemblage carrosserie Peugoet 206 (PSA)

- 13 robots de soudure/assemblage
- 30 automates Premium/Micro

Peinture carrosserie Citroën Picasso

- 5 étapes (bains décapage, anti-corrosion, …), 12 teintes
- 400 transporteurs à rouleaux
- 60 automates Premium + moniteurs

Tunnel de Cointe (Liège liaison E25/E40)

- signalisation/statistique/ éclairage...
- > 20 automates

Convoyeur Laboratoire

- 1 robot «pick & place»
- … 20 actionneurs
- … 12 capteurs
- 1 automate Micro

Iii : Langage Ladder

1 : Définition

Le « Ladder » **(LD)** est un langage graphique de programmation. Proche dans sa représentation graphique des schémas électriques, c'est un langage visuel très simple d'utilisation. Associé au « Function Block Diagram » **(FBD)** le ladder devient un langage complet de programmation.

Un schéma « Ladder » est constitué de plusieurs réseaux. Chaque réseau possède une ligne d'alimentation gauche, une ligne d'alimentation droite et des branches reliant les entrées situées à gauche et les sorties situées à droite.

L'évaluation de chaque réseau se fait de la gauche vers la droite.

L'évaluation de l'ensemble des réseaux se fait du haut vers le bas.

2 : Origine

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.

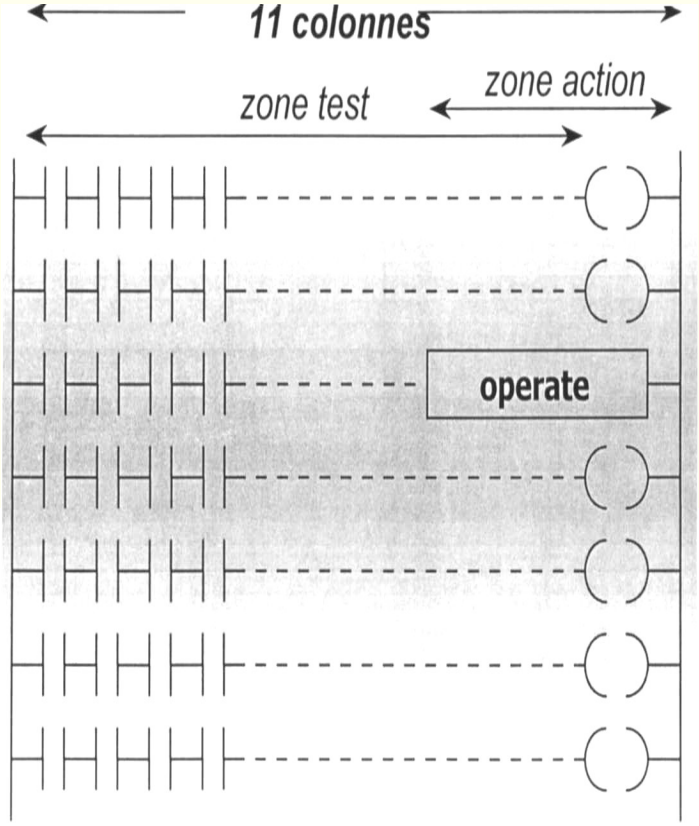
Partant de ces principes, le LADDER a été créé et normalisé dans la norme **CEI 61131-3**. Il est, depuis, très utilisé dans la programmation des Automates Programmable Industriel.

3 : Principe

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, d'où le nom 'LADDER' (échelle).

4 : Structure d'un réseau ladder



Capteur …

… bobine

Scrutation ligne par ligne

5 : 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.

A - Eléments du langage

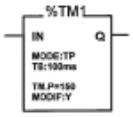
Elements de test (zone « test »)

-| |- : Détection état 1 du bit entrée (« contact passant » si état 1)

-|/|- : Détection état 0 du bit entrée (« contact passant » si état 0)

-|P|- : Détection front montant (« contact passant » sur un cycle)

-|N|- : Détection front descendant (« contact passant » sur un cycle)



: blocs fonctions standards (ex. timers, compteurs, drums, …)

Elements de liaison

——— : Connexion horizontale

| : Connexion verticale

B - Eléments de base du langage

Elements d'action

-( )-	: Ecrire l'état du test (0 ou 1) dans le bit
-(/)-	: Ecrire l'état inverse du test (0 ou 1) dans le bit
-(S)-	: Ecrire et mémoriser l'état 1 dans le bit si l'état du test vaut 1 (set)
-(R)-	: Ecrire et mémoriser l'état 0 dans le bit si l'état du test vaut 1 (reset)
-(#)-	: propre au Grafcet: provoque le passage à l'étape suivante
-(C)-	: branchement à un sous-programme si test = 1
<return>	: retour de sous-programme si test = 1
<halt>	: arrêt du programme si test = 1

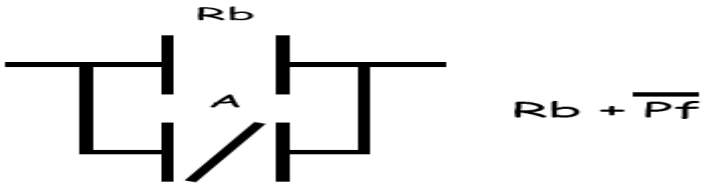
6 : Type de contacts

A - Contacts en série

L'association de contacts en série permet de réaliser des « ET » logiques.

B - Contacts en parallèle

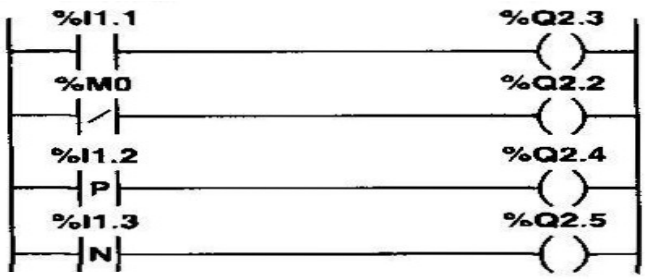
L'association de contacts en parallèle permet de réaliser des « OU » logiques.

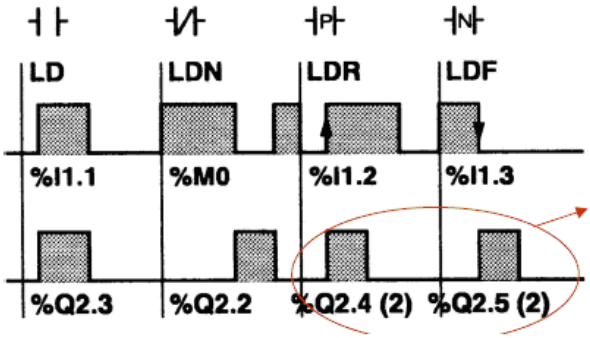


7 : Fonctions de base (sur bits)

A - Instruction de chargement

Instruction

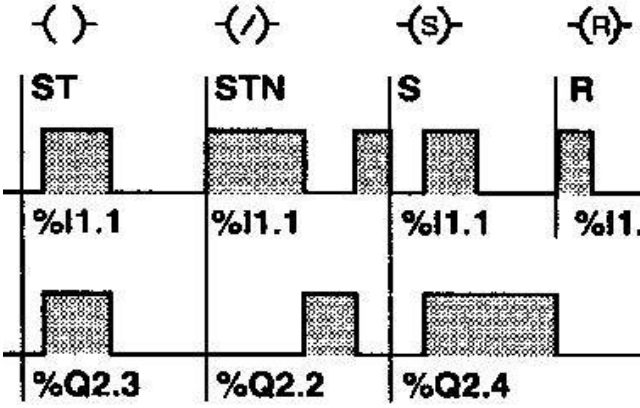


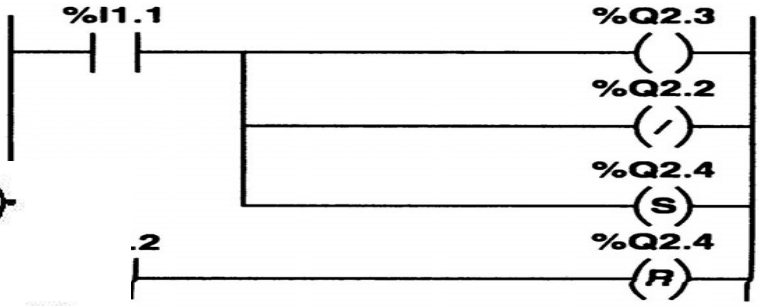


ogramme

B - Instruction d'affectation

Instruction

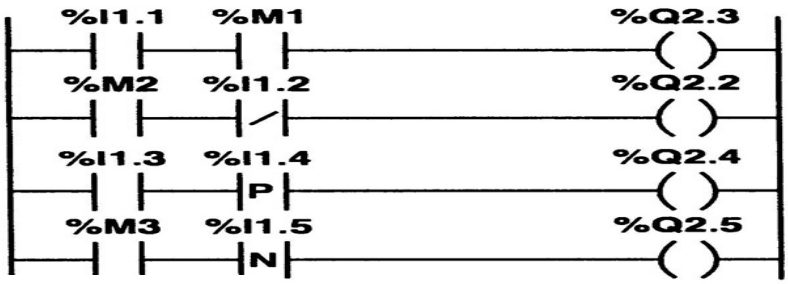


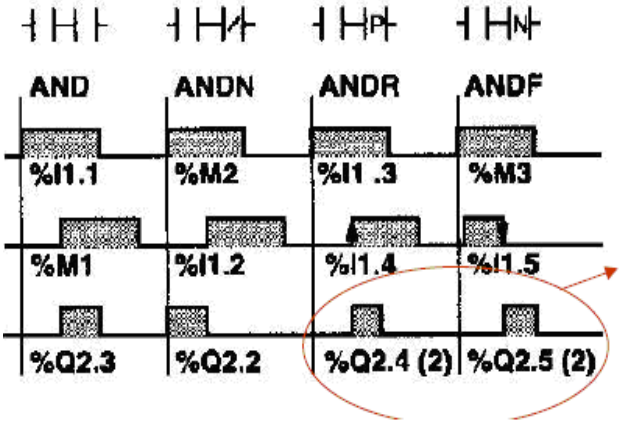


Chronogramme

C - Instruction « ET »

Instruction

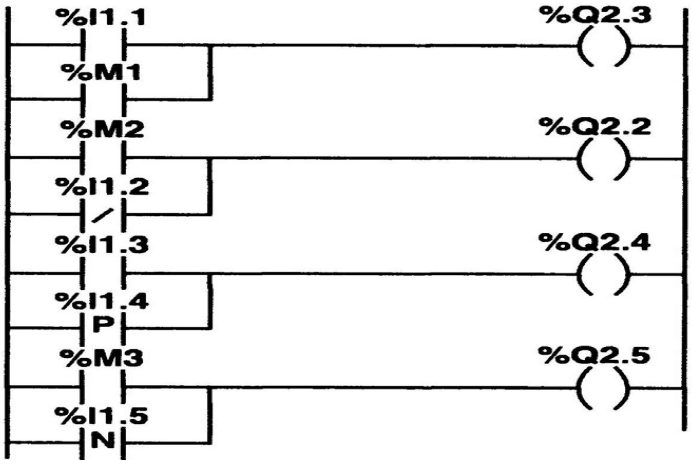


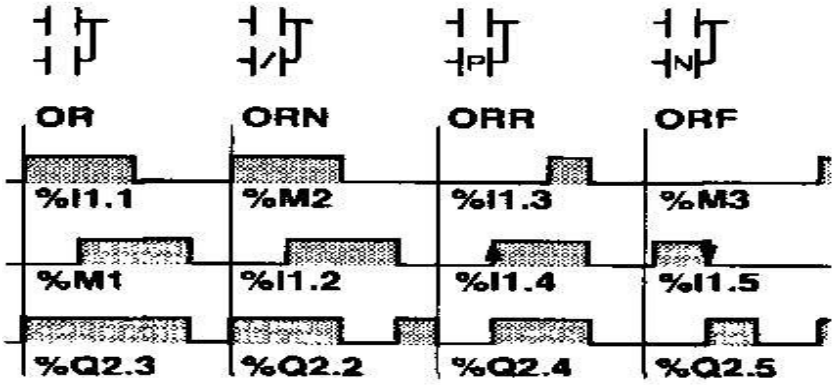


Chronogramme

D - Instruction « OU »

Instruction

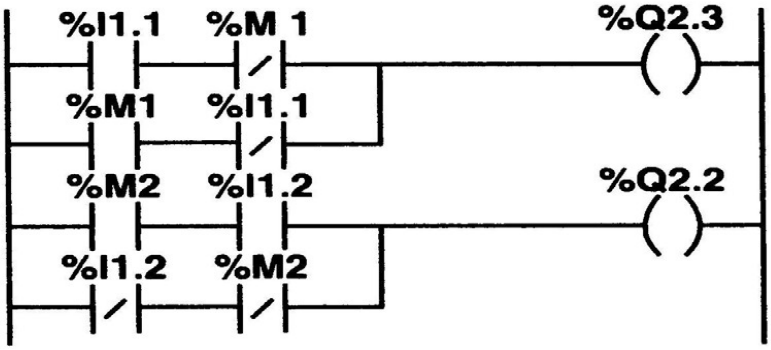


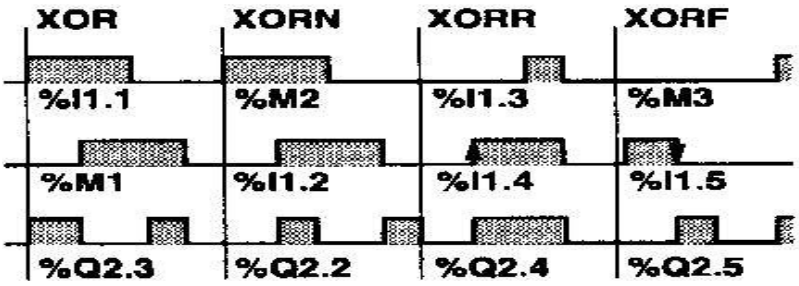


Chronogramme

E - Instruction « OU exclusif » (XOR)

Instruction





Chronogramme

II] PRESENTATION RICAPITULATIVE DE L'A.P.I

- ***Définition :***

L'automate programmable est un système de traitement logique d'informations dont le programme de fonctionnement est effectué à partir d'instructions établies en fonction de processus à réaliser.

- ***Saisie des informations :***

Capteurs mécaniques (contacts), pression, température, déplacement, comptage impulsion, etc…

- ***Interface d'entrée :***

Isoler électriquement (découplage) le circuit puissance et le traitement.
Mise en forme du signal.
Système antiparasite.

- ***Traitement logique :***

Effectuer les opérations logique ET, OU, Mémoire etc. Par utilisation d'un système microprocesseur.

- ***Interface de sortie :***

Elles permettent de commander des relais, des électrovannes, des contacteurs, des moteurs PAS à PAS.

- ***Structure d'un automate programmable*** :

L'automate programmable est construit par autour d'un microprocesseur ou d'un processeur câblé.
Les entrées sont nombreuses et acceptent des signaux venant des capteurs en milieu industriel.
Les sorties sont traitées pour actionner des contacteurs, relais, etc.…
Le langage de programmation est simple, accessible aux non initiés.

- ***Type de programmations :***

  - Programmation par schéma.
  - Programmation par Logigramme.

Programmation par GRAFCET

Manipulation

1 : L'équation va et vient

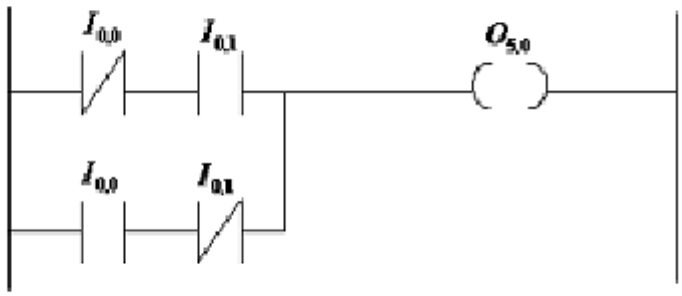
$S = A \oplus B$:

$S = a\bar{b} + \bar{a}b$

Entrées : $I_{0,0}$ et $I_{0,1}$

Sorties : $O_{5,0}$

Schéma de contact de S



Commentaire du tableau

Puisqu'on a travaillé avec API TSX 47 les entrés sont désignées par la lettre I (input) d'indice avec les sorties sont désignées par la lettre O d'indice avec

. Donc d'après ces données on arrive à désigner nos entrées : **'a'** par et **'b'** par

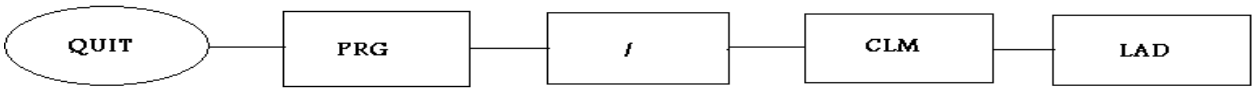
et la sortie **'S'** par .

Pour établir le schéma à contact de S on va se baser sur les composants de langage Ladder.

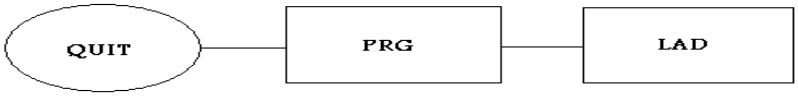
Programmation de S sur TSX 47

Pour programmer S sur TSX 47 on doit suivre les démarches suivantes :

- Avant tout il faut effacer la mémoire (LAD) par la méthode suivante :



- Ouvrir une page de programmation LAD (Ladder) par la méthode :



- Après avoir ouvrir la page de programmation LAD (une ladder) et avant de commencer la programmation il faut saisir une étiquette du réseau, on appuyant sur la touche LAB et par suite on écrit à l'aide de la console ''L1'' puis on appuie sur entrer.
- On appuie sur la touche ''/'' pour afficher les composants de langage Ladder.
- On établie le programme selon le schéma indiqué dans le tableau ci-dessus.
- Après avoir terminé l'écriture du programme on doit l'exécuter suivant les démarches suivantes :
  - On appuie sur la touche entrer puis sur quitter et après on suivre les démarches de l'exécution comme il est indiqué dans le schéma ci-dessous :

QUIT ⟶ DBG ⟶ R/S ⟶ YES

Si l'automate était en mode RUN, il deviendra an mode STOP. Après la réussite de l'exécution la lampe verte de RUN s'allume.

Fonctionnement du programme

L'entrer	La sortie
a = 0 et b=0	S=0
a = 0 et b=1	S=1
a = 1 et b=0	S=1
a = 1 et b=1	S=0

Remarques :

Parmi les avantages de ce mode c'est qu'on peut allumer une lampe et l'éteindre de puis deux endroits différents, comme les escaliers ou les chambres à coucher …

2) programmation ladder en monostable

Soit l'équation de marche arrêt d'un moteur électrique

$x = (m + X)\bar{a}$

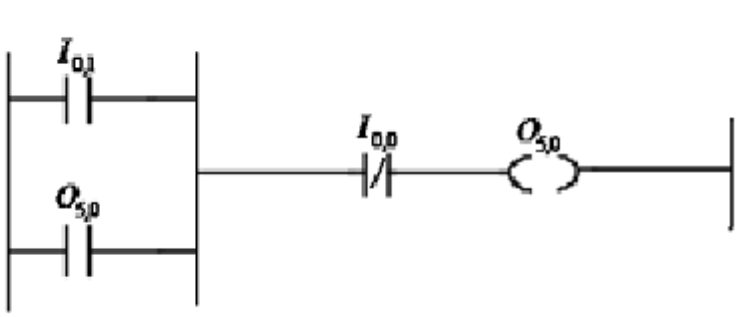
m : marche (bouton poussoir BP)

a : arrêt ; **X** : variable secondaire

Entrées : $I_{0,0}$ et $I_{0,1}$

Sorties : $O_{5,0}$

Schéma de contact de S



Commentaire du tableau :

D'après l'équation on constate qu'on a 3 entrées **a**, **m** et **X** et une seule sortie **x** par contre au niveau de désignation des entées et des sorties on va considérer seulement **a** et **m** comme des entrées par contre **X** et **x** comme une seule sortie, tout ça dans le but d'utiliser

X comme maintien. Donc les entrées **a** et **m** sont désignés successivement par et et

la sortie **X** et **x** sont désignés par .

Programmation de S sur TSX 47

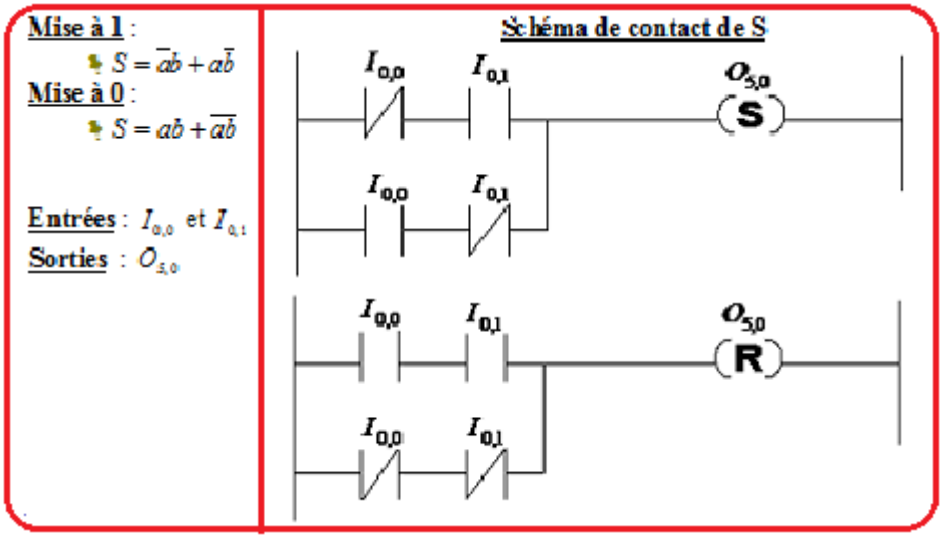
Pour la programmation de x sur TSX 47 on va suivre les mêmes démarches indiquées dans la question 1 pour la programmation de S.

Fonctionnement du programme

L'entrer	La sortie
a = 0 et m=0	x=1 et X=1
a = 0 et m=1	x=1 et X=1
a = 1 et m=0	x=0 et X=0
a = 1 et m=1	x=0 et X=0

On remarque que dans les deux cas **''a = 0 et m=0''** ou **''a = 0 et m=1''** le moteur marche. Donc pour éviter cette contradiction et pour que le bouton **'m'** soit seulement un bouton de marche il doit être un bouton poussoir. Et on remarque aussi la sortie **'x'** s'éteint (égale à 0) lorsque **'a'** est aussi égale à 0. Donc le bouton **'a'** doit être toujours un bouton d'arrêt.

3) programmation Ladder en bistable



Commentaire du tableau :

Par la même méthode suivie dans le (va et vient) et (la programmation ladder monostable)

on arrive à designer nos entrées **'a'** et **'b'** par et et la sortie par

Programmation de S sur TSX 47

On programme S sur TSX 47 par la même méthode appliquée pour la programmation S dans la question 1 'va et vient'.

4) temporisateur

a) réglage d'un temporisateur à 10s avec une base de temps de 1s

Programmation sur TSX 47

Pour la programmation sur TSX 47 les démarches suivies dans la première question 1 sont toujours valables sauf pour la programmation du temporisateur on doit suivre ces étapes :

On appuie sur la touche ZM (zoom), puis sur TB pour régler la base de temps et après sur PRE pour entrer la valeur de présélection.

Après avoir régler le temporisateur on continue l'établissement du programme.

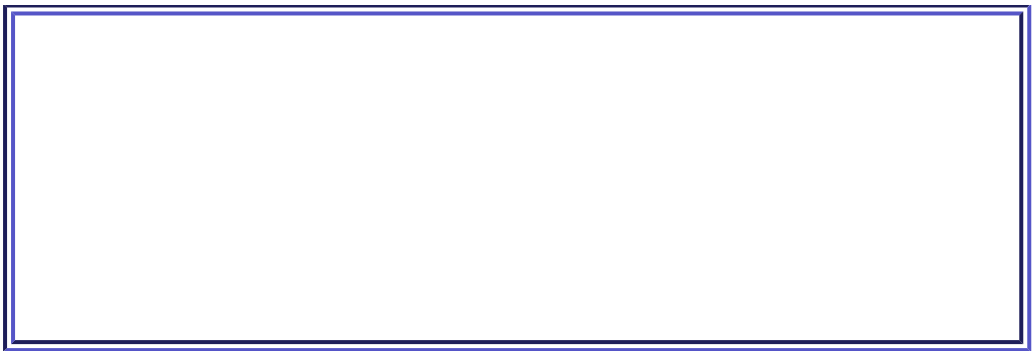
Fonctionnement du programme

Une fois qu'on met le programme en marche (=1) le temporisateur fonctionne et il se comporte comme un décompteur. En remarque que à ce temps la sortie s'allume la première (=1) et que la sortie reste éteinte. De plus la valeur de la présélection 10s commence à décroître d'une unité à chaque impulsion de TB. Lorsque la valeur de présélection es t égale à 0 la sortie s'éteint (=0) et la sortie s'allume (=1)
Au niveau de fonctionnement du temporisateur on distingue deux cas :

Cas 1 : allumer l'entrer (=1) après que les 10s sont écoulées, on constate que le temporisateur revient à 0 et commence de nouveau.

Cas 2 : éteindre l'entrer (=0) avant que les 10s s'écoulent, on constate que dans ce cas le temporisateur recommence de compter à nouveau.

b) réalisation du schéma :



Programmation sur TSX 47

Même démarches utilisées dans le cas 'a'.

Fonctionnement du programme

Dans ce cas le temporisateur fonctionne lorsque les deux entrées E et C sont à l'état 1.Il se comporte alors comme un décompteur.
Pour le mode de fonctionnement de ce temporisateur on constate que :

Lorsque les deux entrées E=C=0 ou l'une des deux égale à 1 le temporisateur ne fonctionne pas et par conséquence aucune de ces deux sortie ne s'allument.

lorsque les deux entrées E=C=1 le temporisateur fonctionne et commence à compter, de plus on constate que la sortie qui s'allume la première et après que les 15s sont écoulées elle s'éteint et la sortie s'allume.

Alors on peut distinguer des cas de fonctionnements exceptionnels de ce temporisateur :

Cas 1 : on met les deux entrées E=C=1 on constate que la sortie s'allume par contre la sortie reste éteinte, mais avant que les 15s seront écoulées on éteint les deux entrées pour quelque secondes (E=C=0) puis on les remet à 1 (E=C=1) on remarque que le temporisateur recommence à compter de nouveau sans prendre en considération les secondes qui sont déjà écoulées.

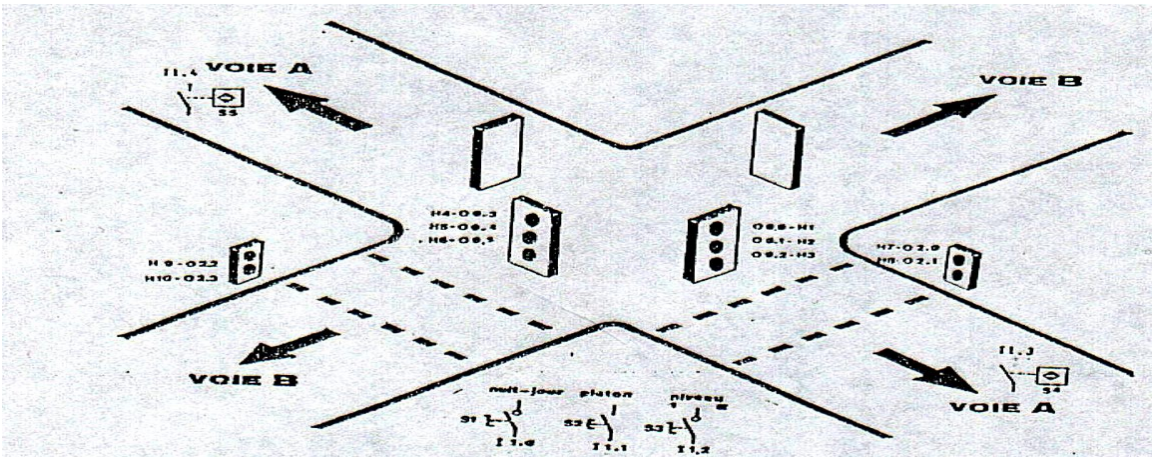
Cas 2 : après quelque secondes (< à15s) d'allumer les deux entrées (E=C=1), on éteint l'entrer E tout en gardant l'entrer C allumé puis on allume E on remarque que le temporisateur recommence à compter de 0.

Cas 3 : dans ce cas on va répéter la même expérience faite dans le cas 2 sauf qu'on va éteindre cette fois l'entée C à la place de E puis on va l'allumée. On constate que le temporisateur ne commence pas à compter de 0 mais il continue.

Cas 4 : dans ce cas on va allumer les deux entrer et après que les 15s seront écoulées on éteint seulement l'entrée E tout en gardant C allumée puis l'allume. On constate que le temporisateur commence à compter de 0.

Cas 5 : même expérience faite dans le cas 4 sauf on va éteindre l'entée C à la place de E puis on l'allume mais dans ce cas on constate que le temporisateur continue à compter.

III] MANIPULATION 2 : FEUX DE CARREFOUR :



● Cahier des charges :

-L'ensemble est l'arrêt.

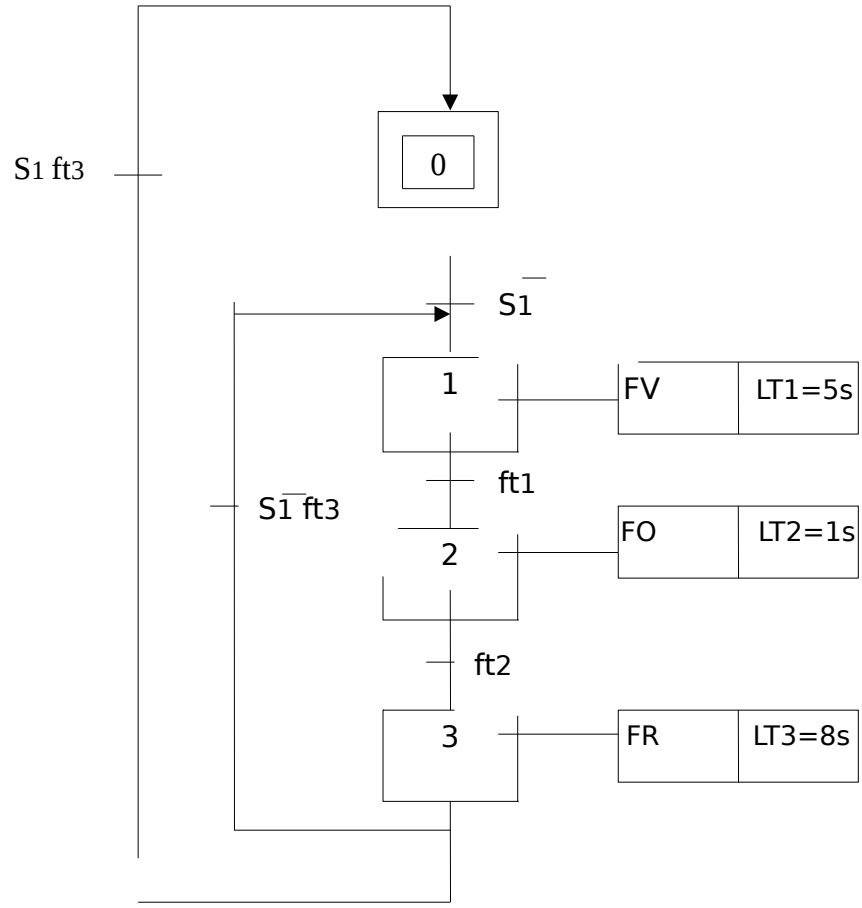
-une action sur le bouton tournant S1 en position jour provoque :

- Le feu vert de la voie A s'allume (FV).

- Au bout d’un temps préréglé de 5 secondes. le feu vert s’éteint. Le feu orange s’allume (FO).
- Au bout d’un temps préréglé de 8 secondes. Le feu orange s’éteint,le feu rouge s’allume(FR).

-Si le bouton S1 est toujours en position « jour ». Le feu passe au vert.
-Si le bouton S1est en position « nuit »l’ensemble est à l’arrêt.

● Le GRAFECET niveau 2 relatifs à la commande de feux de carrefour :

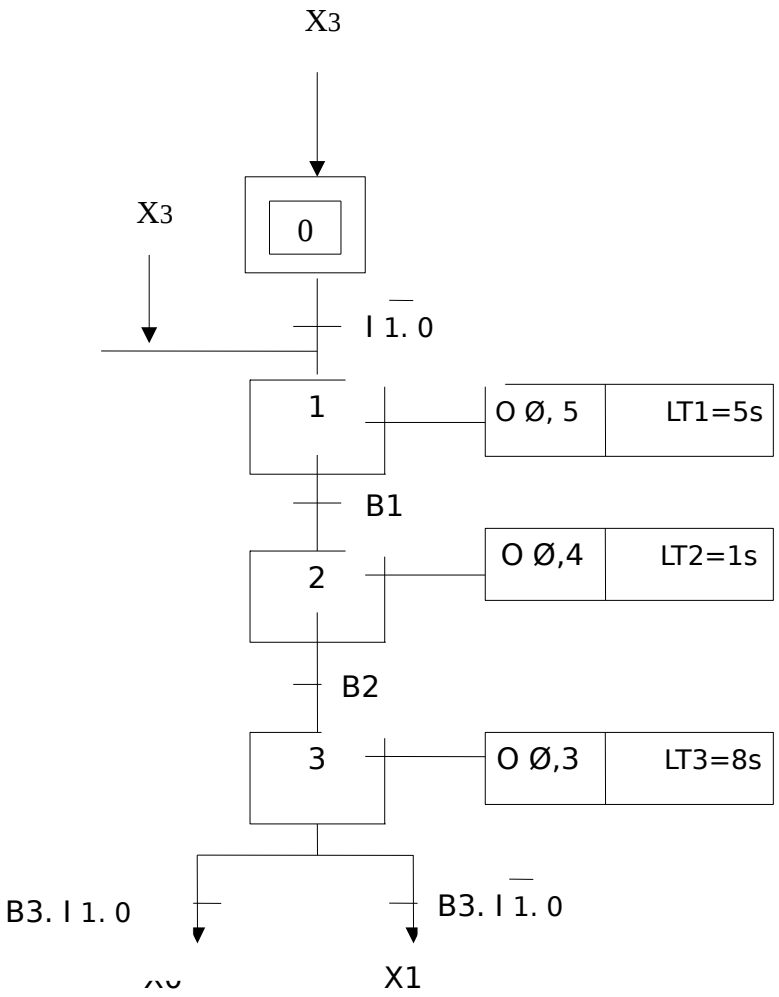


● Tableau des références choisies pour les différents variables :

Variables	Références	

S_1	I 1. 0		
ft_1	B1		
ft_2	B2		
ft_3	B3		
FV	O Ø,5		
FO	O Ø,4		
FR	O Ø,3		
LT1	T1 (5s)		
LT2	T2 (1s)		
LT3	T3 (8s)		

● Le GRAFCET niveau 3 relatifs à la commande de feux de carrefour :



@ PROGRAMMATION DE GRAFCET SUR L A.P.I :

On suit les mêmes étapes citées à l'exercice 1. et on trace en ladder les équations de sorties suivantes :

- **Mise en équation du grafcet niveau 3 :**

Equations d' activations: **Equations d'activations:** **Equations de sorties:**

$X_0 = EI + ft3X_3S_1$ $X_{0eff} = X_1$ T1 (5s) = O Ø.5 =X_1

$X_1 = \overline{S_1}X_0 + ft3\ \overline{S_1}X_3$ $X_{1eff} = X_2$ T2(1s) = O Ø.4 = X_2

$X_2 =$ ft1 X_1 $X_{2eff} = X_3$ T(3s) = O Ø.3 = X_3

X3 = ft2 X_2 $X_{3eff} = X_2 + X_1$

● **Language ladder**:

LAB2