

## Chapitre 4 : Le LADDER (Réseaux à contacts)

### I. Le langage à contacts LADDER

#### 1. Généralités

C'est un langage à contact très proche des circuits électriques.

##### Les données

Les bits d'entrées sont représentés en ladder de la façon suivante : **%Ix,y**

x : n° du module d'interface,

y : n° de l'entrée utilisée sur la carte d'entrées.

Les bits de sorties sont représentés en ladder de la façon suivante : **%Qx,y**

x : n° du module d'interface,

y : n° de l'entrée utilisée sur la carte de sorties.

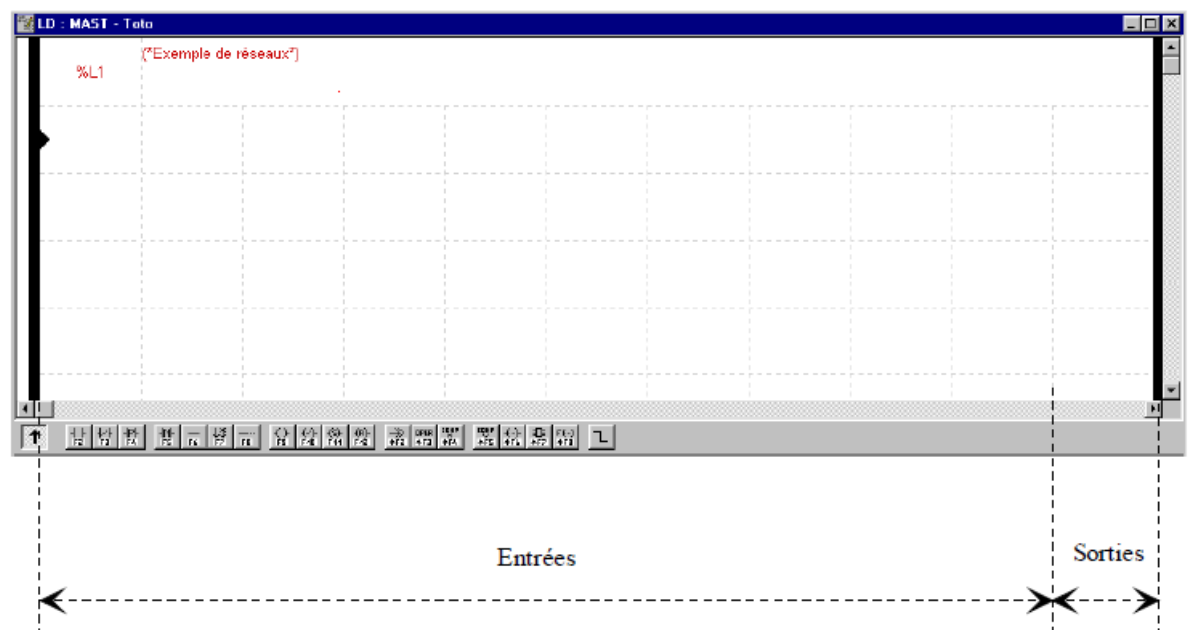
Un automate possède des bits internes pour permettre de faire des calculs logiques. Ces bits internes sont représentés de la façon suivante : **%Mx** avec x : [0..255].

Un automate possède des mots internes pour permettre de faire des calculs arithmétiques ou mémoriser des informations.

Il existe deux sortes de mots :

- Des mots internes variables qui sont représentés de la façon suivante : **%Mwx** avec x : [0..128].
- Des mots internes constants qui sont représentés de la façon suivante : **%Kwx** avec x : [0..128].

**Programme** : Un programme en langage LADDER se compose d'une suite ordonnée de "réseaux".



**Réseau** : Un réseau se compose :

- d'une **étiquette**, encore appelée LABEL, formée des lettres %L suivie d'un numéro compris entre 1 et 999. Deux réseaux différents ne peuvent porter la même étiquette.
- d'un **commentaire** (facultatif), composé de caractères alphanumériques.
- du **réseau proprement dit** : composé de deux barres équipotentiellees verticales, entre lesquelles nous comptons 11 cases (ce nombre de cases n'est pas fixe). Chaque case peut recevoir un "élément graphique" ; nous pouvons ainsi tracer un "schéma électrique".

**Éléments du LADDER** : Le langage LADDER se compose :

- du langage à contacts,
- des blocs comparaisons,
- des blocs opérations,
- des blocs fonctions.

2

## 2. Le langage à contacts

**Éléments graphiques** : Les éléments graphiques dont nous disposons pour programmer un réseau se classent en trois catégories.

- **Les contacts (entrées, test de l'état des sorties, temporisation...)** :

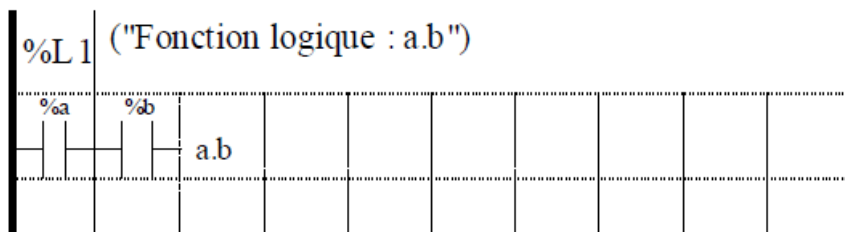
- les contacts "travail" (variable logique non inversée) :



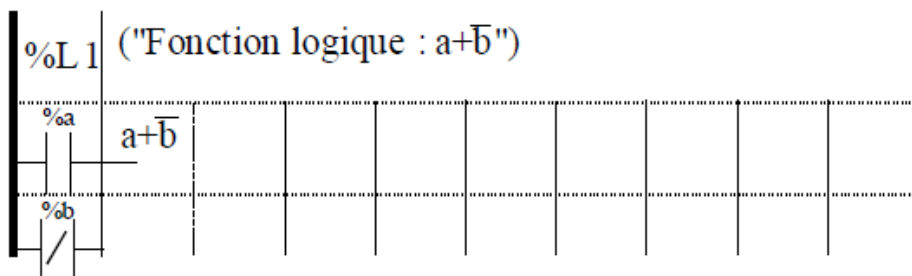
- les contacts "repos" (variable logique inversée) :



Exemple de fonction :  $a.b$



Exemple de fonction :  $a + \bar{b}$




- les contacts pour faire les fronts montants :





- les contacts pour faire les fronts descendants :




- **Les bobines (sorties) :**



- les bobines directes : 


- les bobines inverses : 

- les bobines d'enclenchement ou bobines SET : 

- les bobines de déclenchement ou bobines RESET : 

- **Les connexions entre chaque élément :**

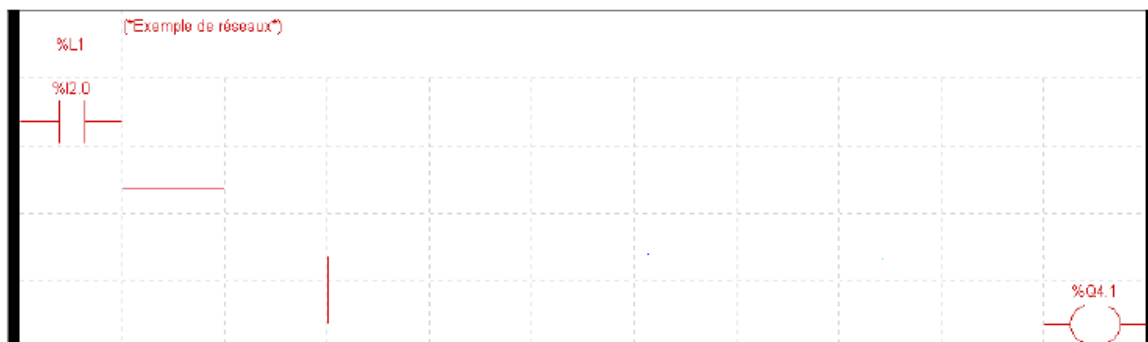
- les connexions horizontales :  

- Les connexions verticales : 

Les contacts et les liaisons horizontales se placent dans toutes les cases appartenant aux 10 premières colonnes.

Les liaisons verticales se placent entre deux colonnes, à cheval sur deux lignes.

Les bobines se placent dans les cases de la 11<sup>e</sup> colonne.



- **Fonctionnement des contacts :** chaque contact est associé à un bit, lu par le processeur. Le contact travail doit être considéré comme fermé si et seulement si son bit associé vaut 1.
- **Fonctionnement des bobines directes et inverses :** Chacune de ces bobines est associée à un bit, écrit par le processeur.

Pour une bobine directe, le processeur fixe ce bit à 1 si la bobine est alimentée et le fixe à 0 dans le cas contraire.

Pour une bobine inverse, le processeur fixe ce bit à 0 si la bobine est alimentée et le fixe à 1 dans le cas contraire.

- **Fonctionnement des bobines d'enclenchement (bobines SET) et de déclenchement (bobines RESET) :** Chacune de ces bobines est associée à un bit, écrit par le processeur.

Pour une bobine d'enclenchement, le processeur fixe ce bit à 1 si la bobine est alimentée, et ne le modifie pas dans le cas contraire.

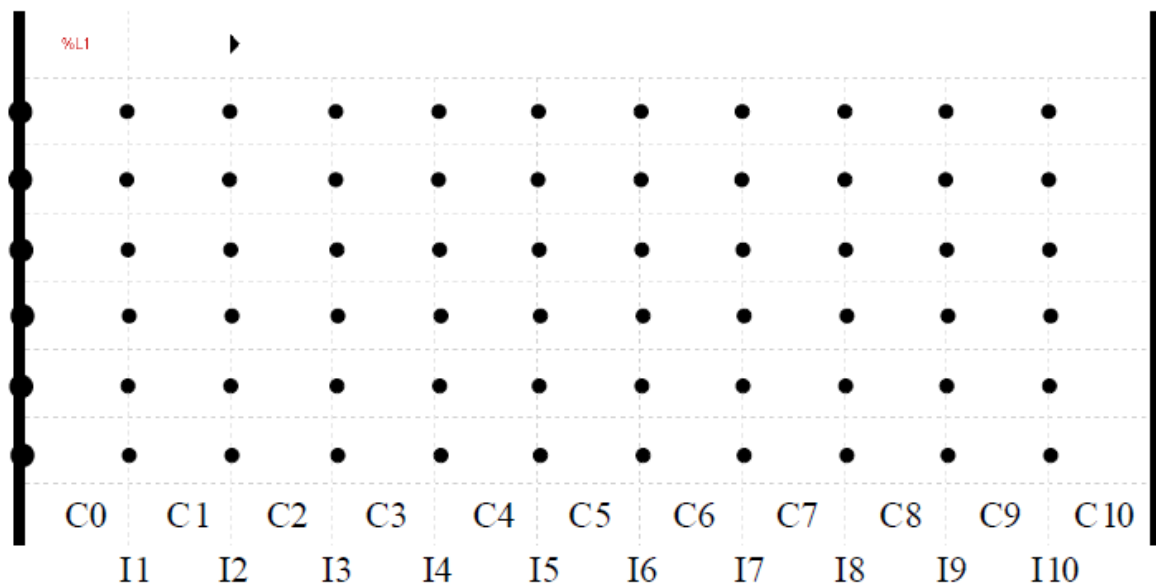
Pour une bobine de déclenchement, le processeur fixe ce bit à 0 si la bobine est alimentée, et ne le modifie pas dans le cas contraire.

### 3. Structuration d'un réseau

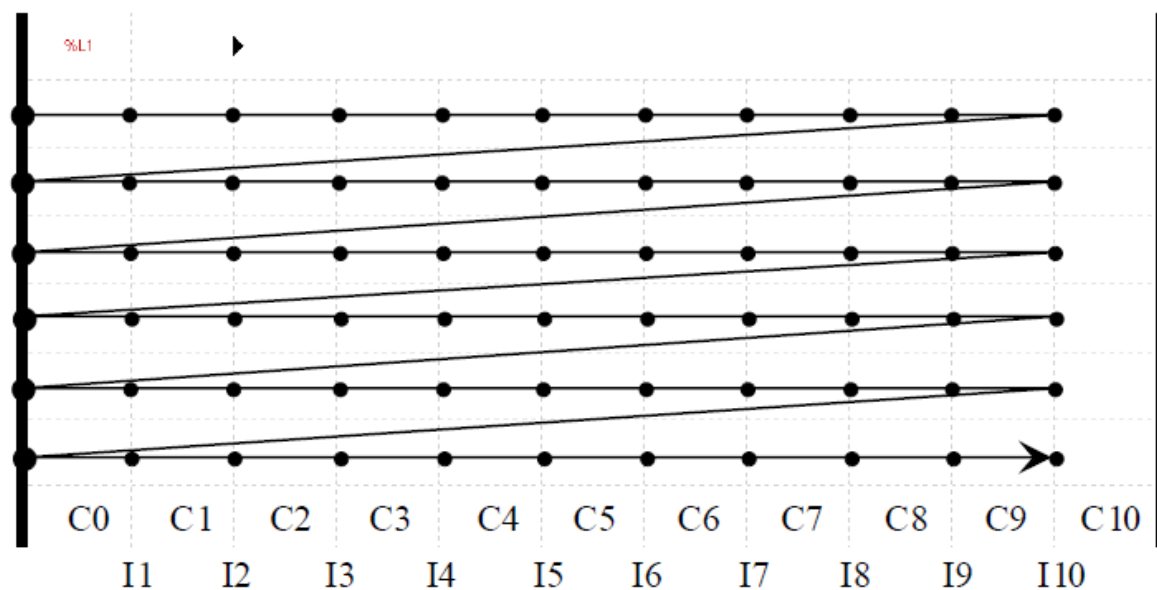
#### *Langage à contacts*

4

- **Noeuds d'un réseau :** Pour comprendre la scrutation d'un réseau, il faut se figurer les "noeuds" d'un réseau, tels qu'ils apparaissent sur la figure ci-dessous, et remarquer que tous les éléments graphiques aboutissent à des noeuds.



- **Règles de scrutation :** La scrutation d'un réseau exploité en langage à contacts obéit aux règles suivantes.
  - Les noeuds de la barre équipotentielle de gauche sont tous au "potentiel logique" 1,
  - La scrutation du réseau s'effectue dans l'ordre suivant :
    - Scrutation de la ligne 1,
    - Scrutation de la ligne 2,
    - Scrutation de la ligne 3,
    - Scrutation de la ligne 4,
    - .
    - .
    - Scrutation de la ligne 10 etc..

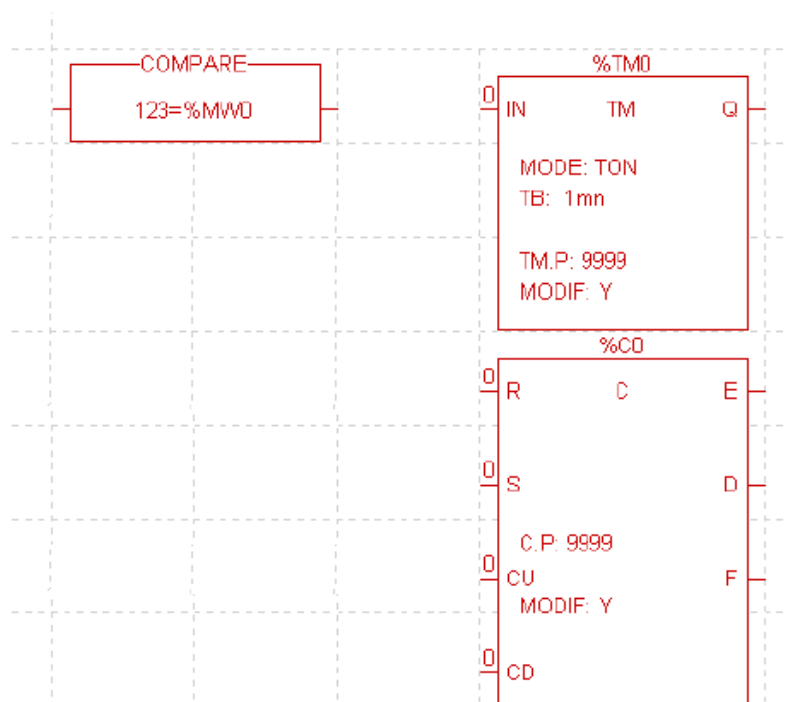


5

- lors de la scrutation d'une case appartenant à l'une des 10 premières colonnes, le processeur fixe le potentiel des noeuds de droite de cette case :
  - à la valeur 0 si cette case n'est pas conductrice
  - à la valeur du potentiel du noeud de gauche si cette case est conductrice
- lors de la scrutation d'une case de la colonne 11, le processeur déclare "alimentée" une bobine dont le noeud de gauche est au potentiel logique 1.

### Scrutation des blocs

- **Blocs comparaison - blocs fonctions** : Dans un réseau LADDER, la représentation d'un bloc comparaison ou d'un bloc fonction est un rectangle occupant deux colonnes (la colonne 11 étant exclue)



Un tel bloc présente donc un ou plusieurs noeuds d'entrée et un ou plusieurs noeuds de sortie.

Le principe de scrutation est le même que celui d'un réseau exploité en langage à contact. Toutefois :

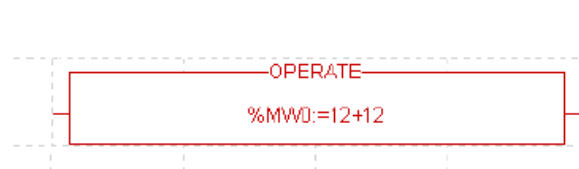
- les noeuds internes du bloc ne sont pas scrutés,
- seule la case en haut à gauche du bloc est scrutée,
- lors de la scrutation de cette case, l'exécution de la comparaison ou de la fonction est déclenchée, selon :

- les valeurs des potentiels logiques des noeuds d'entrée,
- la nature du bloc.

et les potentiels des noeuds de sortie sont fixées en conséquence.

6

- **Blocs opérations** : Dans un réseau LADDER, la représentation d'un bloc opération est un rectangle occupant quatre colonnes (dont la colonne C10)



Un tel bloc présente donc un noeud d'entrée.

#### 4. Les blocs de comparaison

- **Élément graphique** : Dans un réseau, l'élément graphique représentant un bloc comparaison occupe la place de deux contacts. Il peut se placer dans les colonnes C0 à C10 :



- **Fonctionnement** : Le bloc comparaison se comporte comme un contact dont l'état dépend d'une condition exprimée à l'intérieur de ce bloc. Ce contact est
  - fermé si la condition est vérifiée
  - ouvert dans le cas contraire.
- **Opérandes** : Un bloc comparaison fait appel à deux opérandes, notés OP1 et OP2. Il s'agit de mots de 16 bits.

L'opérande OP1 peut être : un mot interne, un mot constant ou une valeur immédiate.

L'opérande OP2 peut être : un mot interne, un mot constant ou une valeur immédiate.

- **Comparaisons** : La condition exprimée à l'intérieur d'un bloc comparaison est l'une des six comparaisons suivantes :

OP1 < OP2	OP1 <= OP2
OP1 > OP2	OP1 => OP2
OP1 = OP2	OP1 <> OP2

## 5. Les blocs opérations

### Généralités

- **Élément graphique** : dans un réseau, l'élément graphique représentant un bloc opération occupe la place de trois contacts et d'une bobine.



- **Fonctionnement** : Le bloc opération se comporte comme une bobine, dont l'alimentation provoque l'exécution d'une opération exprimée à l'intérieur de ce bloc.
- **Opérandes** : Un bloc opération fait appel à deux opérandes, notés OP1 et OP2. Il s'agit, selon, les cas, de mots internes, de mots constants, de chaînes de bits, ou de valeurs immédiates.

En cas d'opérateur à une opérande, OP1 n'est pas utilisé.

- **Opérateur** : Un bloc opérateur fait appel à un opérateur, noté OP. Le tableau ci-dessous donne la liste des ces opérateurs.

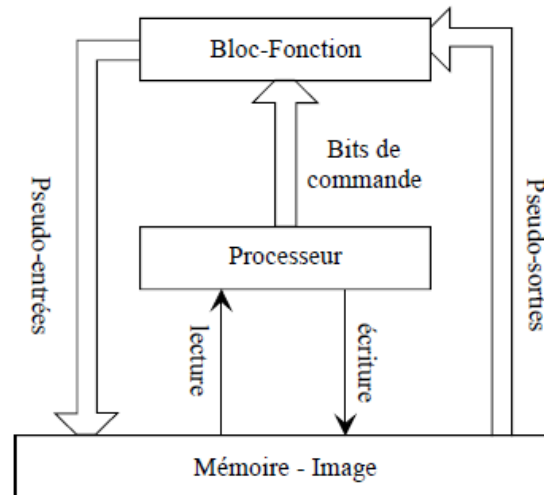
Opérateurs		OP
Arithmétiques à un opérande	Transfert	:=
	Conversion BCD → Binaire	INT TO BCD
	Conversion Binaire → BCD	BCD TO INT
Logiques à un opérande	Complément bit à bit	NOT
	Décalage circulaire à droite	ROR
	Décalage circulaire à gauche	ROL
Arithmétiques à deux opérandes	Addition	+
	Soustraction	-
	Multiplication	*
	Division	/
Logiques à deux opérandes	ET bit à bit	AND
	OU bit à bit	OR
	OU exclusif bit à bit	XOR
	Affectation	:=

## 6. Les blocs Fonctions

### Généralités

- **Principe** : Bien qu'incorporé au processeur, un bloc fonction doit être considéré comme une ressource physique séparée de celui-ci. Un tel bloc fonction :
  - reçoit des ordres du processeur par des "bits de commande" non adressés,
  - échange avec la mémoire image des pseudo - variables d'entrées - sorties" adressées.





- **Limitation du nombre** : puisqu'il s'agit de ressources physiques, chaque type de bloc fonction est numéroté, et limité en nombre.
- **Lecture et écriture** : Il s'agit de lecture et d'écriture par le processeur. Donc :
  - les pseudo variables d'entrées peuvent être lues, mais non écrites,
  - les pseudo variables de sorties peuvent être lues et écrites.

### *Les temporisations*

- **Élément graphique** : Dans un réseau, l'élément graphique représentant un temporisateur occupe trois lignes et deux colonnes. Il peut se placer dans les colonnes C0 à C10 :



- **Caractéristiques :**

#### Bits de commande :

- bit de lancement IN.

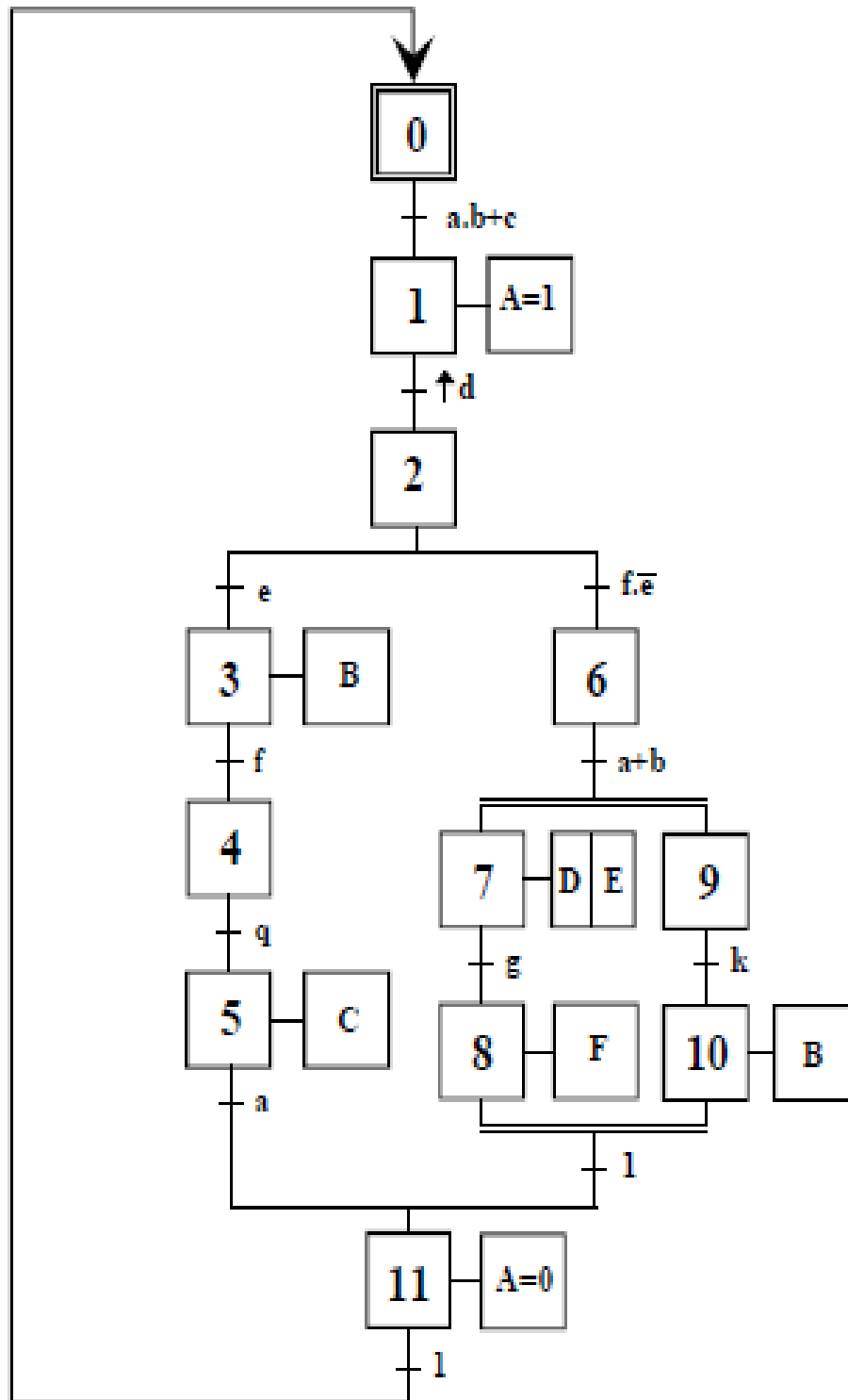
- **Bases de temps** : La valeur de la base de temps est fixée lors du paramétrage du bloc. Les valeurs possibles correspondent à des bits systèmes cadencés par le processeur aux périodes indiquées par le tableau ci-dessous. Nous avons 10 ms, 100 ms, 1 s et 1 mn.

En conséquence, il faut toujours adopter la base de temps la plus faible parmi celles qui sont compatibles, avec l'application.



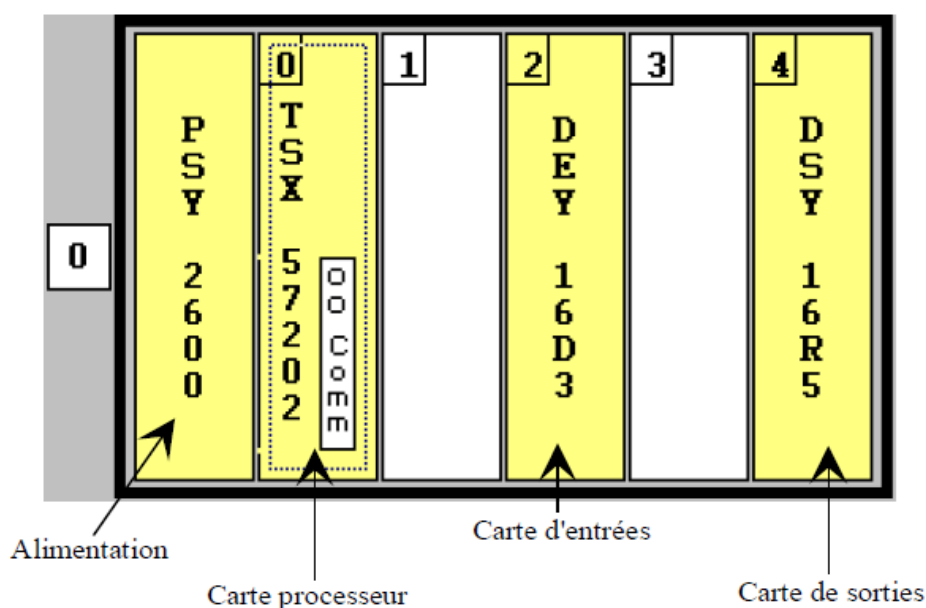
## II. Représentation d'un GRAFCET en LADDER

Pour représenter un Grafcet en langage LADDER il faut utiliser un mot qui contient le numéro de l'étape active. Le mot utilisé s'appellera par exemple %Mw1. Nous avons ici un Grafcet contenant un ET avec deux branches. A cause de ce ET nous allons avoir deux étapes actives en même temps. Donc pour mémoriser deux étapes il faut utiliser deux mots : %Mw1 et %Mw2.



Nous avons un automate avec une carte de 16 entrées et une carte de 16 sorties. La carte d'entrées est positionné en 2<sup>ème</sup> position. La carte de sorties est positionné en 4<sup>ème</sup> position.

**Configuration de l'automate :**



Donc les entrées seront numérotées de %I2.0 à %I2.15 et les sorties seront numérotées de %Q4.0 à %Q4.15.

La première chose à faire est un tableau de correspondance. Ce tableau permet de faire la liaison entre les entrées ou les sorties utilisées sur le Grafcet et les entrées/sorties utilisées dans l'automate.

**Tableau de correspondance :**

Entrées Grafcets	Entrées Automates	Sorties Grafcets	Sorties Automates
a	%I2.0	A	%Q4.0
b	%I2.1	B	%Q4.1
c	%I2.2	C	%Q4.2
d	%I2.3	D	%Q4.3
e	%I2.4	E	%Q4.4
f	%I2.5	F	%Q4.5
g	%I2.6		
k	%I2.7		
q	%I2.8		
i (Init)	%I2.9		

