1. Introduction

La matérialisation d'un **GRAFCET** peut être réalisée de deux façons :

- **Logique câblée** à base de **séquenceur** : elle est simple et adaptée à des petits systèmes figés.
- **Logique programmée** à base d'ordinateur, de microcontrôleur ou **d'automate programmable industriel** : cette solution a l'avantage d'être flexible et évolutive puisqu'elle s'adapte facilement à tout changement du système par un simple changement de programme.

2. Séquenceur électronique

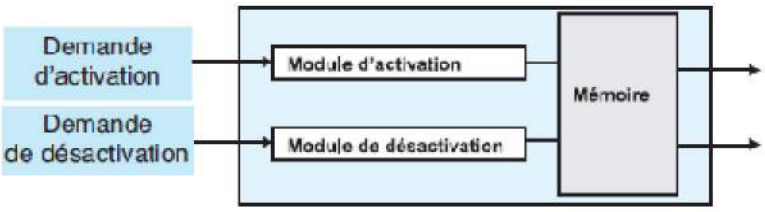
21. Définition

Un séquenceur est une mise en cascade d'un ensemble de **modules d'étapes** qui commande une suite d'événements structurés par un **GRAFCET**. Chaque module matérialise une étape.
Les séquenceurs sont à technologie **pneumatique**, **électromagnétique** ou **électrique** et **électronique**.
Dans ce qui suit, on s'intéresse aux séquenceurs électroniques.

21. Module d'étape

Un **module d'étape** est constitué d'un :

- Module d'activation.
- Module de désactivation.
- Module de mémorisation ou mémoire.



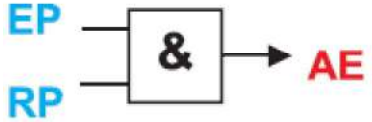
211. Module d'activation

Pour qu'une étape soit active il faut que :

- L'étape (ou les étapes) immédiatement précédente (s) soit (soient) active(s)

ET

- La (les) réceptivité (s) immédiatement précédente (s) soit (soient) vraie (s).



AE : Activation étape
EP : Etapes précédentes
RP : Réceptivité précédente

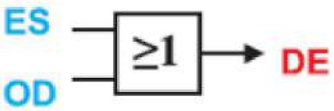
212. Module de désactivation

Pour désactiver une étape il faut que :

- L'étape (ou les étapes) immédiatement suivante (s) soit (soient) active(s).

OU

- L'ordre de désactivation (remise à zéro **RAZ**) soit demandé.



DE : Désactivation étape
ES : Etapes suivante
OD : Ordre de désactivation

213. Mémoire

La fonction mémoire est matérialisée par :

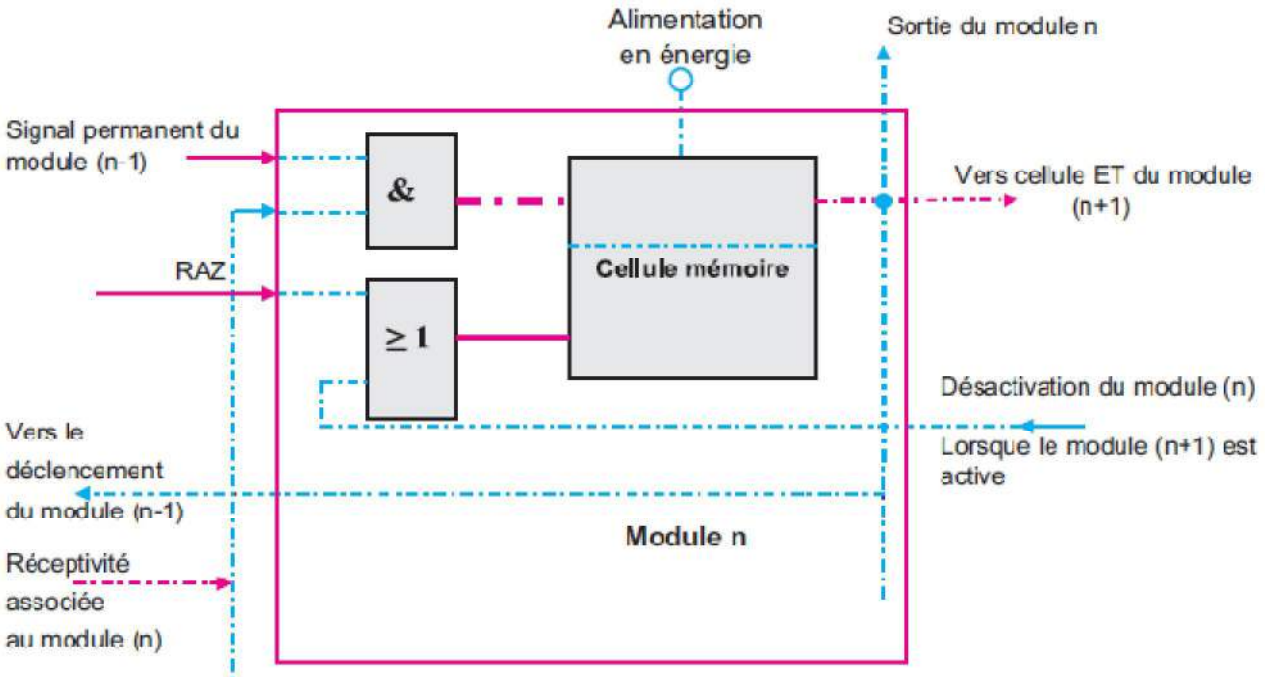
- Deux entrées **A** (pour l'activation) et **D** (pour la désactivation).
- Deux sorties complémentaires **S** et $\overline{S}$.

C'est une bascule **RS** dont l'entrée **Set** sert pour l'activation et l'entrée Reset sert pour la désactivation.

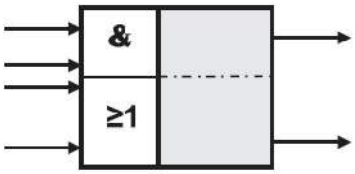
214. Module d'étape

Structure

L'association des trois fonctions précédentes forme le module d'étape suivant :

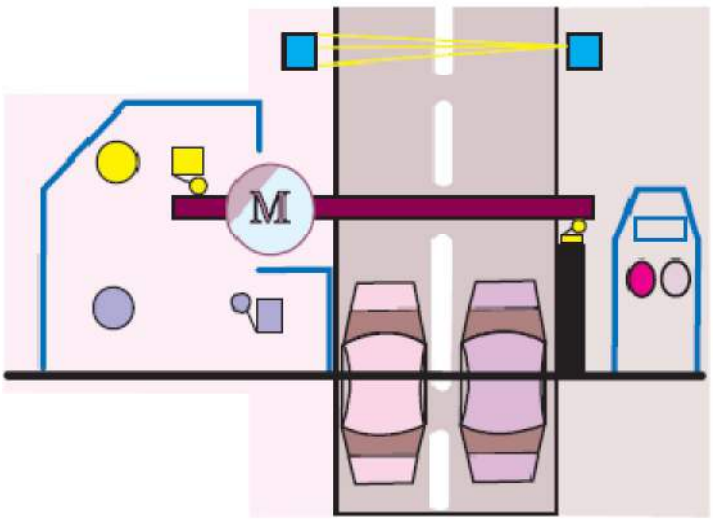


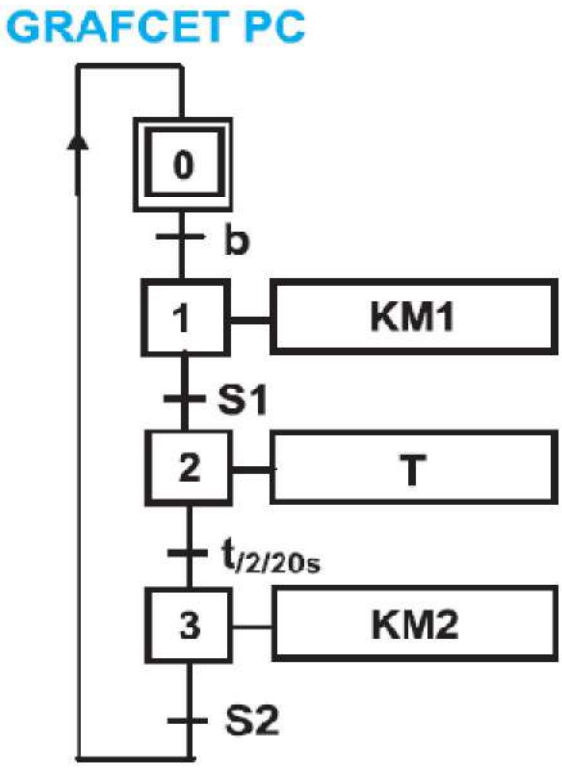
Symbole



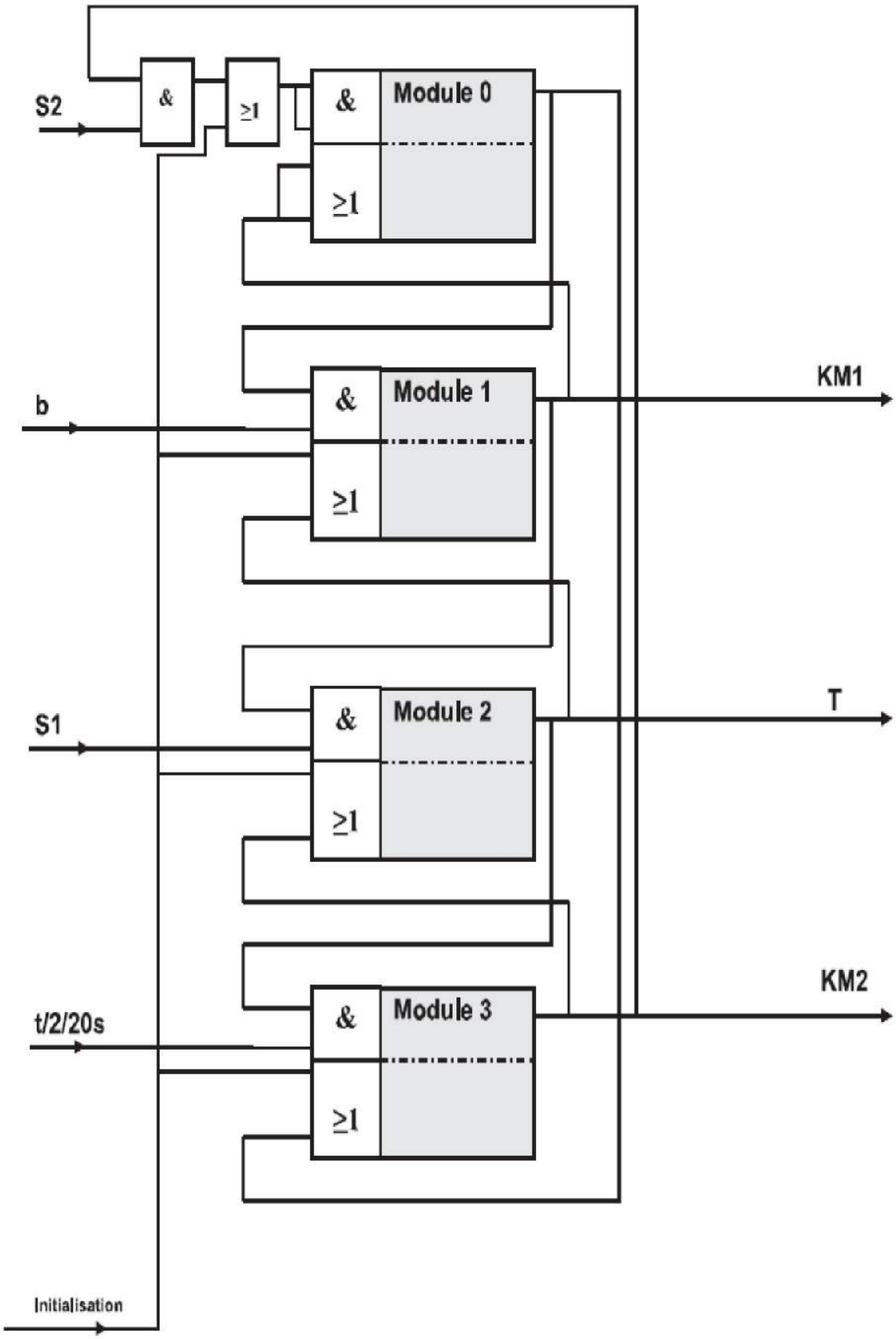
23. Exemple : Barrière automatique

Le fonctionnement de la barrière est décrit par un **GRAFCET** d'un point de vue de la partie commande.





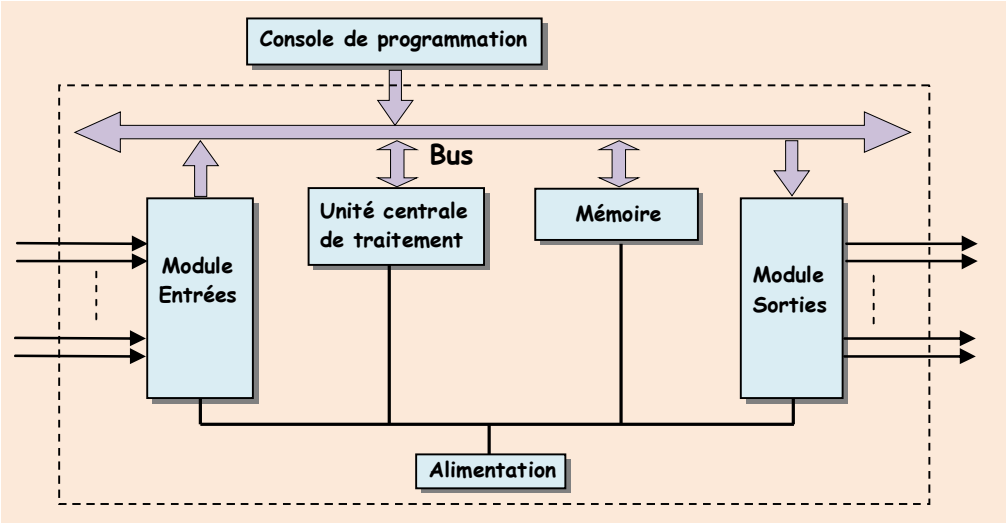
Le schéma du câblage sur un séquenceur électronique qui matérialise le **GRAFCET** ci-contre est le suivant :


S2
&
≥1
&
Module 0
≥1
b
&
Module 1
≥1
KM1
S1
&
Module 2
≥1
T
t/2/20s
&
Module 3
≥1
KM2
Initialisation

3. Automate programmable industriel : API

31. Structure

Un Automate Programmable Industriel est une machine électronique programmable destinée à piloter en ambiance industrielle et en temps réel des systèmes automatisés.
Il existe de nombreuses marques d'automates programmables. Parmi les plus courantes on cite :
Siemens, **Allen Bradley**, **Modicon**, **Schneider Electric** (ex. Télémécanique), **Omron**, **Cegelec**, etc.
La structure interne d'un **API** est représentée par la figure suivante :



311. Mémoire

Elle permet De :

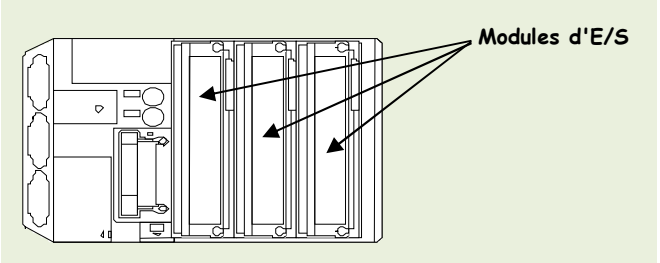
- Recevoir les informations issues des entrées.
- Recevoir les informations générées par l'unité centrale de traitement (processeur) et destinées à la commande des sorties (valeurs des sorties, des temporisations, etc).
- Recevoir et conserver le programme d'automatisation du processus.

312. Unité centrale de traitement

Elle réalise toutes les fonctions logiques et arithmétiques à partir d'un programme contenu dans sa mémoire. Elle lit et écrit dans la mémoire et actualise les sorties. Elle est connectée aux autres éléments (mémoire et interfaces d'entrées/sorties) par un **Bus** parallèle qui véhicule les informations entre ces éléments.

313. Interfaces d'entrées/sorties

Les entrées reçoivent des informations en provenance des capteurs et du pupitre opérateur.
Les sorties transmettent des informations aux préactionneurs et aux éléments de signalisation du pupitre.
Ces interfaces d'entrées/sorties (**E/S**) se présentent généralement sous forme d'interfaces modulaires qu'on ajoute selon le besoin.



Fonction Traiter

L'interface d'entrée a pour fonction de :

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

Généralement les entrées sont désignées par le symbole **%Ii.j** où **i** est le numéro du module et **j** le numéro de l'entrée dans ce module, le signe **%** est spécifique au constructeur (ici Télémécanique).
Exemple : **%I0.3** représente l'entrée **3** du module **0**.

L'interface de sortie a pour fonction de :

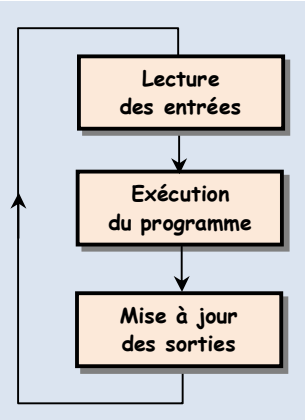
- Commander les préactionneurs et les éléments de signalisation du système.
- Adapter les niveaux de tension de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières.

Généralement les sorties sont désignées par le symbole **%Qi.j** où **i** est le numéro du module et **j** le numéro de la sortie dans ce module.
Exemple : **% Q1.5** représente la sortie **5** du module **1**.

314. Console de programmation

C'est généralement un ordinateur où est installé le logiciel de programmation spécifique à l'**API**.
Ce logiciel permet d'éditer le programme, de le compiler et de le transférer à l'automate. L'ordinateur peut également servir de poste opérateur pour assurer la conduite de l'unité. Un autre logiciel est alors nécessaire pour établir le dialogue avec l'automate.

32. Cycle d'exécution d'un automate



Durant son fonctionnement, un **API** exécute le même cycle de fonctionnement qu'on appelle **cycle automate**.
La durée de ce cycle est typiquement de 1 à 50 ms :

- Avant chaque traitement, l'**API** lit les entrées et les mémorise durant le cycle automate.
- Il calcule les équations logiques de fonctionnement du système en fonction des entrées et d'autres variables internes puis il les mémorise.
- Les résultats sont recopiés dans les sorties.

33. Programmation de l'automate

La programmation d'un **API** consiste à traduire dans le langage spécifique de l'automate, les équations de fonctionnement du système à automatiser. Parmi les langages normalisés, on cite :

331. IL : Instruction List ou liste d'instructions

Ce langage textuel de bas niveau est un langage à une instruction par ligne. Il ressemble, dans certains aspects, au langage assembleur employé pour la programmation des microprocesseurs.

332. ST : Structured Text ou texte structuré

Ce langage textuel de haut niveau est un langage évolué. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

333. LD : Ladder Diagram ou schéma à contacts

Ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (true ou false).

334. SFC : Sequential Function Chart

Issu du langage **GRAFCET**, ce langage de haut niveau permet la programmation aisée de tous les procédés séquentiels.

335. FBD : Function Block Diagram

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.

NB : Généralement, les constructeurs d'**API** proposent des environnements logiciels graphiques pour la programmation.

4. Module logique Zelio

41. Présentation

C'est un **API** commercialisé par le constructeur **Schneider Electric**. IL est programmable à l'aide du logiciel **Zelio Soft** soit en langage **FBD** ou en langage à contacts (**Ladder**). Cette programmation nécessite la connexion de l'**API** à un ordinateur via le port série.
Les modules logiques **Zelio** sont variés. On optera dans un exemple ultérieur pour le module **SR2 B121BD**.IL est destiné à la réalisation de petits équipements d'automatisme (petites machines de finition, de confection, d'assemblage ou d'emballage, etc).
Le modèle utilisé a pour caractéristiques principales :

- **4 entrées TOR I1** à **I4**.
- **4 entrées mixtes** (**TOR/Analogique**) **IB** à **IE**.
- **4 sorties** à relais **Q1** à **Q4**.
- **Interface Homme/machine** avec boutons et affichage LCD.
- Langages de programmation **Ladder** et **FBD**.

42. Programmation en langage Ladder

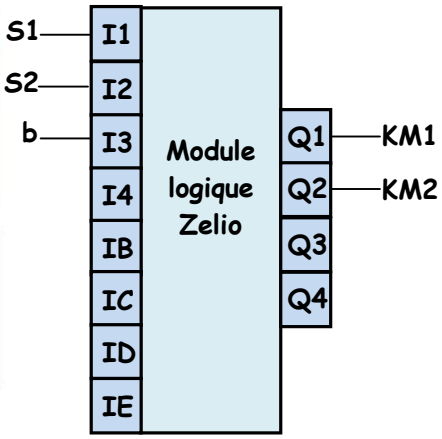
On reprend l'exemple précédant de la **barrière automatique**.

421. Affectation des Entrées/Sorties

Entrées		
Information	Capteur	Entrée API
Barrière levée	Fin de course S1	I1
Barrière baissée	Fin de course S2	I2
Billet tiré du distributeur	Détecteur b	I3

Sorties		
Action	Actionneur/Préactionneur	Sortie API
Lever la barrière	Moteur M/ Relais KM1	Q1
Baisser la barrière	Moteur M/ Relais KM2	Q2

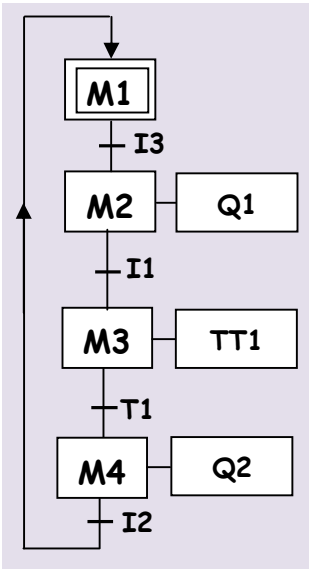
Configuration des entrées/sorties au niveau de l'API



422 Réalisation du GRAFCET en langage Ladder

Il faut d'abord commencer par transformer le **GRAFCET PC** en **GRAFCET** codé Automate en respectant le tableau des affectations des entrées/sorties ci-dessus.

GRAFCET codé de l'API



La programmation du **GRAFCET** en langage **Ladder** de l'**API** consiste à associer à chaque étape **i** du **GRAFCET** une variable interne **Mi** de l'API.
Le **GRAFCET** du point de vue partie commande **PC** étudié contient **4** étapes.
On utilisera alors **4** variables internes **M1**, **M2**, **M3** et **M4**.
Le programme est alors constitué de **2** phases de traitement :

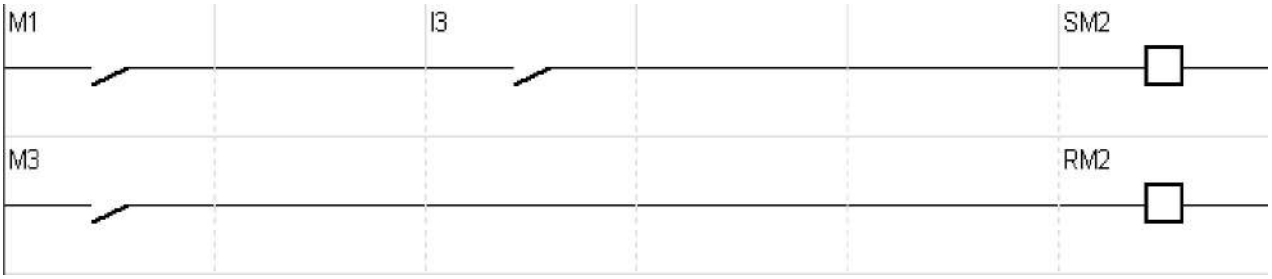
Phase de traitement séquentiel :

Cette partie du programme décrit l'évolution séquentielle des étapes en calculant l'état des variables internes **Mi** représentant les étapes, autrement dit, établir les équations logiques d'activation et de désactivation des étapes.
Les étapes sont programmées comme des bobines ou des relais auxiliaires avec une mise à **1** et une remise à **0** :

- La mise à 1 (**Set**) est déduite de l'équation d'activation de l'étape.
- La remise à 0 (**Reset**) est déduite de l'équation de désactivation de l'étape.

Exemple : Set(M2)=M1.I3 et Reset(M2)=M3

La traduction des ces deux équations logiques en langage **Ladder** est donc :



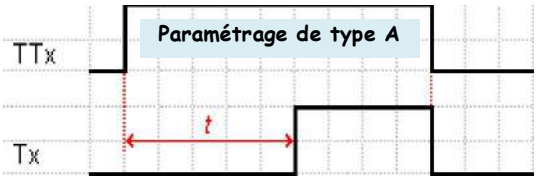
Phase de traitement postérieur :

Cette partie détermine l'état des sorties.
Exemple : La sortie **Q1** est active si on est dans l'étape **1** associée à la variable interne **M2**.

Génération d'une temporisation

L'étape **2** du **GRAFCET PC** est associée à une temporisation de **20** secondes. Cette étape correspond à la variable interne **M3**. Pour réaliser cette temporisation, on utilise un temporisateur **T** paramétré en type **A**.

- **TTx** présente la commande du temporisateur **x**.
- **Tx** symbolise le contact du temporisateur **x**.
- **t** est le retard ou la durée de la temporisation.
- **x** est l'indice du temporisateur choisi (1, 2, 3, ...).

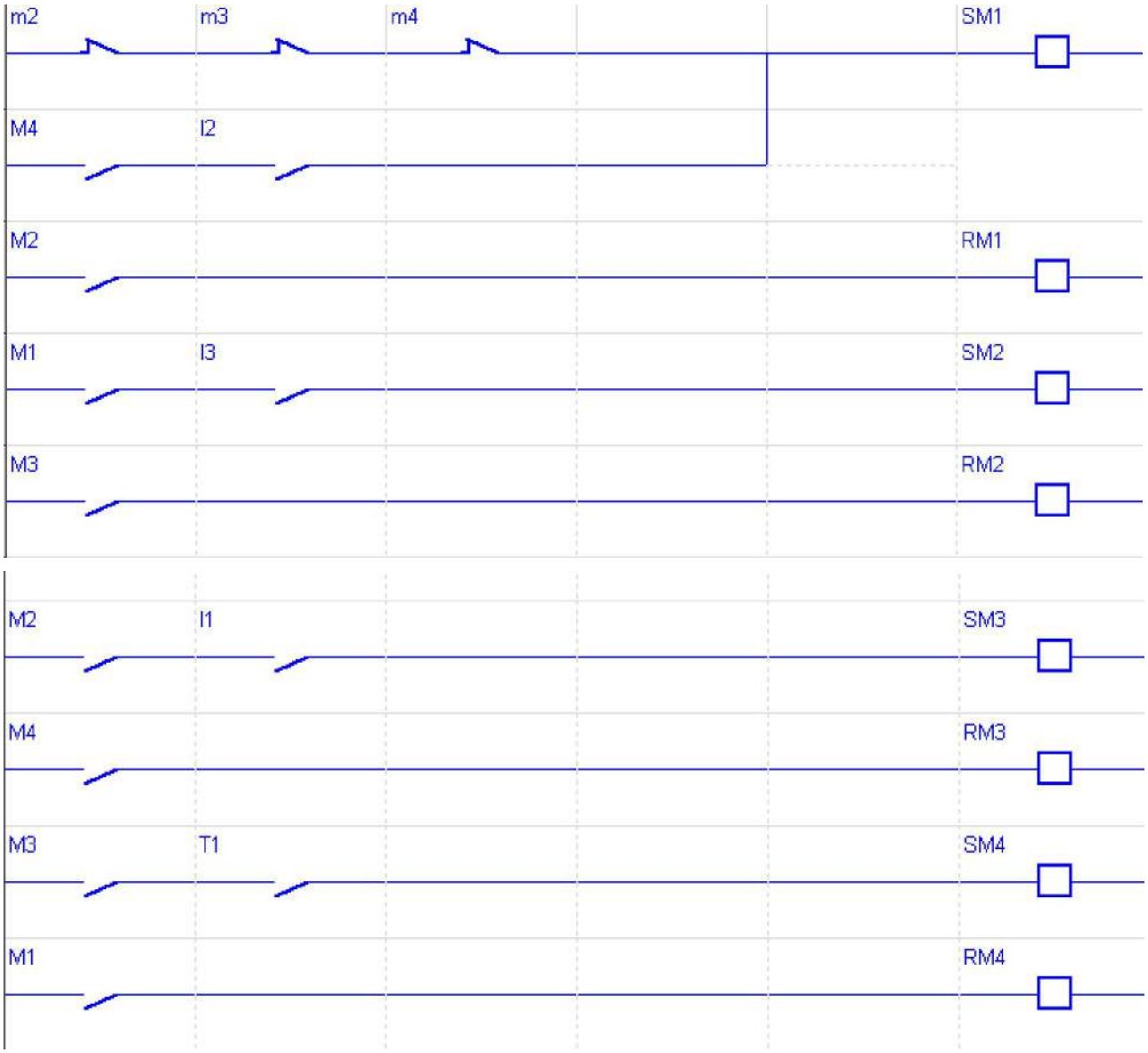


Lorsque le temporisateur est activé **TTx=1**, le contact correspondant **Tx** ne sera à l'état haut qu'après un retard **t**.
Une fois le temporisateur sera désactivé **TTx=0**, le contact correspondant **Tx** passe aussi à 0.

Le programme complet en langage **Ladder** qui permet la gestion de la barrière par l'automate est le suivant :
On note que l'étape initiale **M1** doit être active au démarrage quand toutes les autres étapes ne sont pas actives. En plus, elle sera aussi activée par l'étape précédente **M4**.

Fonction Traiter

Phase de traitement séquentiel



Phase de traitement postérieur

Fonction Traiter

423 Câblage de l'automate