


# AUTOMATISMES INDUSTRIELS

## Programmation des automates

Cours 2

3h30 -  
v1.0 

*IUT de Cachan - 9 Avenue de la division Leclerc - 94230 Cachan*

## LA PROGRAMMATION D'UN AUTOMATE INDUSTRIEL

### Table des matières

<b>1</b>	<b>Le langage LADDER</b>	<b>2</b>
1.1	Réseaux LADDER simples . . . . .	2
1.1.1	Les contacts . . . . .	2
1.1.2	Les bobines . . . . .	3
1.1.3	Réseaux de base . . . . .	3
<b>2</b>	<b>Le langage GRAFCET</b>	<b>4</b>
2.1	Étapes et actions . . . . .	4
2.2	Transitions . . . . .	5
2.3	Saut d'étapes et reprise de séquence . . . . .	5
2.4	Divergences et Convergences . . . . .	6
2.4.1	Divergence/Convergence en OU . . . . .	6
2.4.2	Divergence en ET . . . . .	7
2.5	Règles de conception . . . . .	7
2.6	Deux niveaux de GRAFCET . . . . .	8
2.6.1	Niveau fonctionnel (Fig 7a) . . . . .	8
2.6.2	Niveau opérationnel 7b . . . . .	8
<b>3</b>	<b>Le texte structuré</b>	<b>9</b>
3.1	syntaxe du langage . . . . .	9
3.1.1	opérateurs . . . . .	9
3.1.2	structure IF . . . . .	9
3.2	Description d'un grafcet en texte structuré . . . . .	10
<b>4</b>	<b>Bonnes pratiques</b>	<b>10</b>

## Introduction

Le logiciel de développement d'un API (Automate Programmable Industriel) est un logiciel permettant de configurer et de programmer l'API associé. Chaque constructeur d'API impose son propre logiciel de développement.

Schneider	Siemens	Wago	Tend	Omron
Unity Pro EcoStructure	Tia Portal	CodeSys eCockpit	Niagara	CX-Programmer

TABLE 1: Logiciel de développement pour chaque constructeur.

## 1 Le langage LADDER

Le langage LADDER fut conçu dans les années 1970 pour faire passer les électrotechniciens de la saisie de schémas électriques de systèmes à relais à la programmation. Il est donc simple et proche de la description de schémas électriques. On le réserve à la description de fonctions combinatoires ou à des calculs simples.



### À retenir

- Une ligne d'un programme LADDER est appelée réseau.
- Un réseau est composé de contacts, de bobine et/ou de blocs.

### 1.1 Réseaux LADDER simples

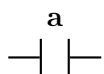
#### 1.1.1 Les contacts

Les contacts représentent des entrées logiques (TOR).



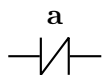
### À retenir

Il existe deux types de contacts :



**Contact normalement ouvert** : actif lorsque la variable associée est à l'état 1.

⇒ Il représente donc la variable  $a$



**Contact normalement fermé** : actif lorsque la variable associée est à l'état 0.

⇒ Il représente donc la variable  $\bar{a}$

### 1.1.2 Les bobines

Les bobines représentent les sorties logiques (TOR) de l'API.



#### À retenir

Une sortie  $S$  de l'automate est active lorsque la bobine associée  $\text{---} \overset{S}{( )} \text{---}$  est active.

### 1.1.3 Réseaux de base

La traduction en réseau LADDER de l'équation  $S = a$  est donc représentée sur la figure 1a. L'équation en réseau LADDER de l'équation  $S = \bar{a}$  est représentée sur la figure 1b

Comme dans un circuit électrique, il est possible de programmer des équation combinatoires en langage LADDER. Les fonctions **ET** et **OU** sont représentée sur la figure 1

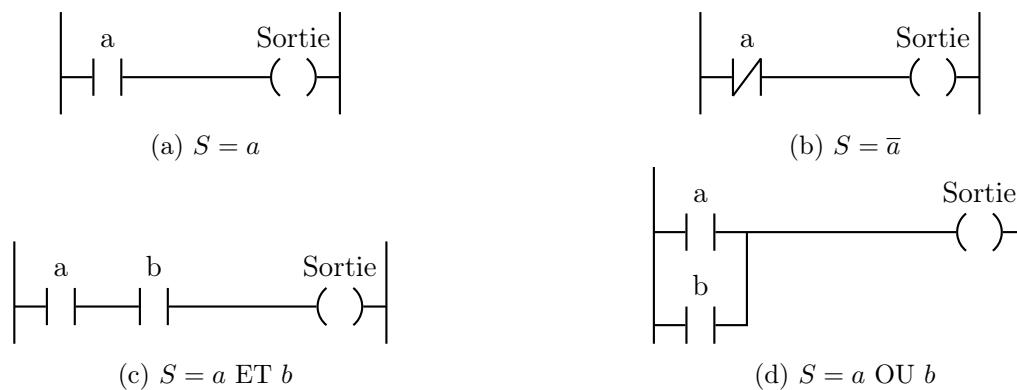
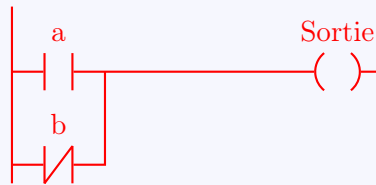


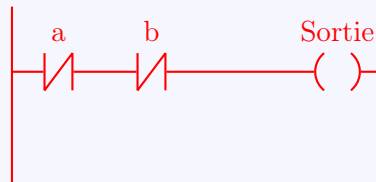
FIGURE 1: Equations logiques simples

**Activité 1**

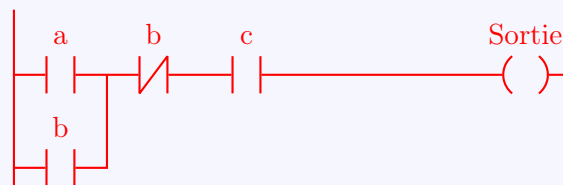
**Question 1** Dessinez le réseau LADDER de l'équation  $S = a + \bar{b}$



**Question 2** Dessinez le réseau LADDER de l'équation  $S = \bar{a} \cdot \bar{b}$



**Question 3** Dessinez le réseau LADDER de l'équation  $S = (a + b) \cdot \bar{b} \cdot c$



## 2 Le langage GRAFCET

Le langage GRAFCET, aussi appelé SFC (Sequentiel Function Chart) est, comme son nom l'indique, dédié à la description et à la programmation de **problèmes séquentiels**. Il décrit donc une suite d'étapes séparées par des transitions.

Un GRAFCET est très proche conceptuellement d'une machine à état. Il est représenté tel que sur la Figure 2

### 2.1 Étapes et actions

Une étape est représentée par un carré comportant un numéro (Figure 3a). Elle est généralement associée à une (Figure 3b) ou plusieurs (Figure 3c) actions. Il est également possible d'ajouter une condition à une action (Figure 3d).

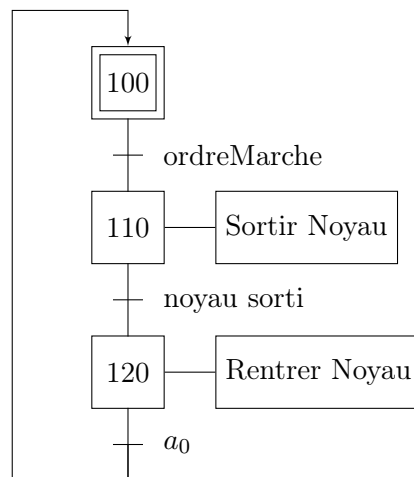


FIGURE 2: GRAFCET simple composé de trois étapes

**À retenir**

- Une action associée à une étape n'est effectuée que lorsque cette étape est active.
- Une action conditionnelle est effectuée lorsque **l'étape est active et que la condition associée est vérifiée**.

Il est également possible d'effectuer une action qu'au moment de l'activation d'une étape (Figure 3e). Cela est particulièrement utile pour incrémenter un compteur. En effet, sans cela l'action *compteur = compteur + 1* serait effectuée durant toute la durée de l'action associée. Cela incrémenterait le compteur à chaque cycle automate donc approximativement toutes les 10 ms.

Pour un grafcet en cours d'exécution, on peut représenter l'étape active dans un grafcet à l'aide d'une étoile comme illustré sur la Figure 3f.

## 2.2 Transitions

Une transition est **toujours** placée entre deux actions. Ce sont les transitions qui décrivent la façon de passer d'une action à une autre. Une transition est associée à une **réceptivité** (Voir Figure 4a).

**À retenir**

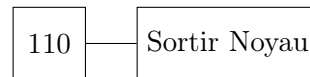
Une transition sera franchie si et seulement si l'étape précédente est active **ET** la réceptivité associée est vérifiée.

## 2.3 Saut d'étapes et reprise de séquence

Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées sont inutiles à réaliser. La reprise de séquence (ou boucle) permet de reprendre, une ou plusieurs fois, une séquence tant qu'une condition n'est pas obtenue.



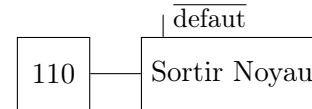
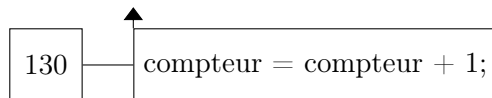
(a) Représentation d'une étape en GRAFCET



(b) Une étape et l'action associée



(c) Une étape et deux actions simultanées. Les deux actions sont actives tant que l'étape 110 est active.

(d) Une étape avec action conditionnelle : L'action *Sortir Noyau* ne sera exécutée que si *default* est à 0.

(e) Une étape et une action sur activation. L'action n'est effectuée qu'au moment de l'activation de l'étape.

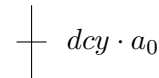


(f) Etape couramment active

FIGURE 3: Etapes et action(s) associée(s)



(a) Une transition



(b) Une transition et sa réceptivité associée

FIGURE 4: Transitions et Réceptivité

## 2.4 Divergences et Convergences

### 2.4.1 Divergence/Convergence en OU

On dit qu'il y a Aiguillage ou **divergence en OU** lorsque le grafcet se décompose en deux ou plusieurs séquences selon un choix conditionnel. Après une divergence en OU on rencontre souvent une **convergence en OU**. On dit qu'il y a convergence en OU, lorsque deux ou plusieurs séquences du grafcet converge vers une seule séquence.

Un exemple est donnée en Figure 5.



#### À retenir

- Une divergence en **OU** est **toujours** suivie d'une transition sur chacune de ses branches.
- Une convergence en **OU** est **toujours** précédée d'une transition sur chacune de ses branches.
- Les conditions d'une divergence doivent être exclusives.

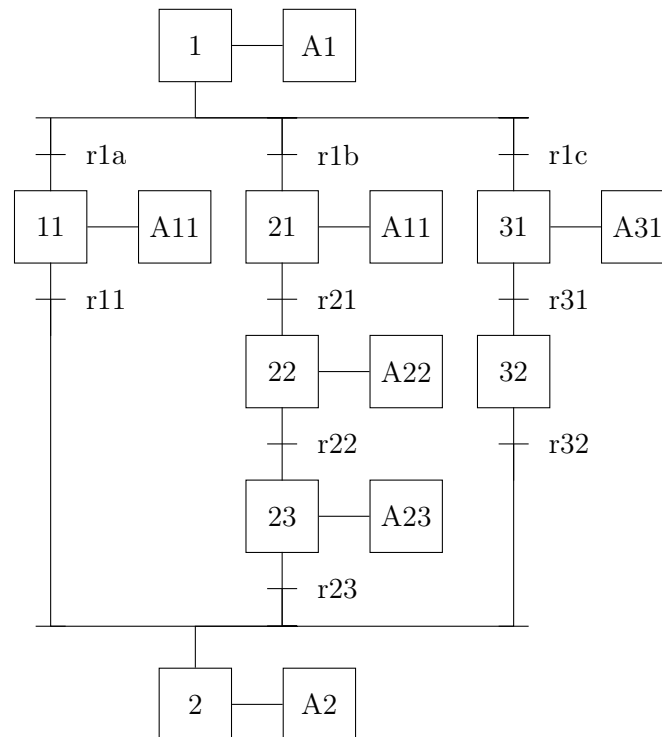


FIGURE 5: Exemple de divergence en **OU**. Une seule des branches est parcourue.

### 2.4.2 Divergence en ET

Au contraire de l'aiguillage, où ne peut se dérouler qu'une seule activité à la fois, on dit qu'on se trouve en présence d'un parallélisme structurel (**divergence en ET**), si plusieurs activités indépendantes peuvent se dérouler en parallèle. Un parallélisme structurel commence par une divergence en ET et termine par une convergence en ET. Celles-ci sont représentées par deux traits parallèles. Un exemple est donnée en Figure 6.



#### Remarque

La dernière étape d'une divergence en ET est souvent vide pour attendre les autres branches.

## 2.5 Règles de conception



#### À retenir

- Les actions ne concernent que les **actionneurs** ou des variables internes.
- Les réceptivités ne concernent que les capteurs ou des variables internes
- Une étape est **toujours** suivie d'une transition
- Une transition est **toujours** suivie d'une étape.

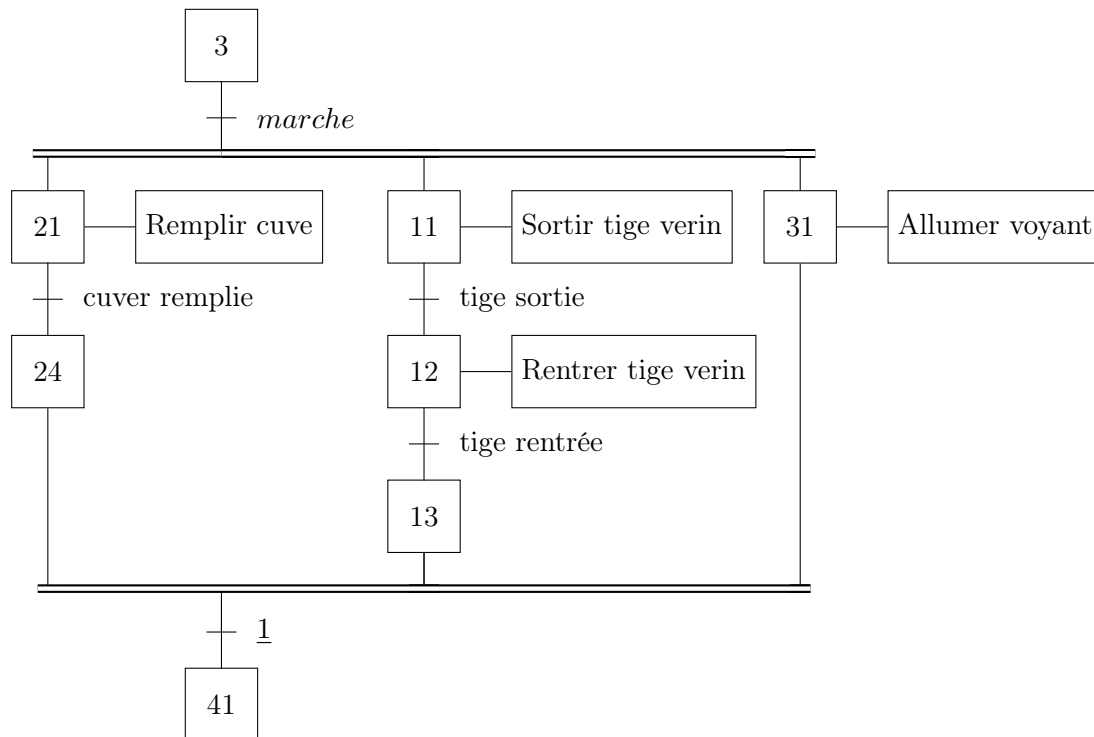


FIGURE 6: Exemple de divergence en **ET**. Les trois branches sont parcourue simultanément.

## 2.6 Deux niveaux de GRAFCET

On peut écrire les grafcet d'un point de vue fonctionnel ou d'un point de vue opérationnel. Ces deux points de vue sont décrits ci-dessous et illustrés en Figure 7.

### 2.6.1 Niveau fonctionnel (Fig 7a)

Ce grafcet utilise des mots du cahier des charges. C'est la première étape d'élaboration d'un programme. Il est destiné à être compris par tous les collaborateurs d'un projet. Ce sont donc des verbes qui sont utilisés pour décrire les actions d'une étape.

### 2.6.2 Niveau opérationnel 7b

Ce niveau de grafcet sert à la programmation de l'automate. C'est celui qui sera retranscrit dans le logiciel de programmation et envoyé à l'automate. Il est souvent moins lisible que le grafcet fonctionnel et est destiné aux intervenants sur la partie opérative.

Ce grafcet est décrit à l'aide des noms de variables associées aux entrées (pour les réceptivités) et aux sorties (pour les actions) de l'automate.



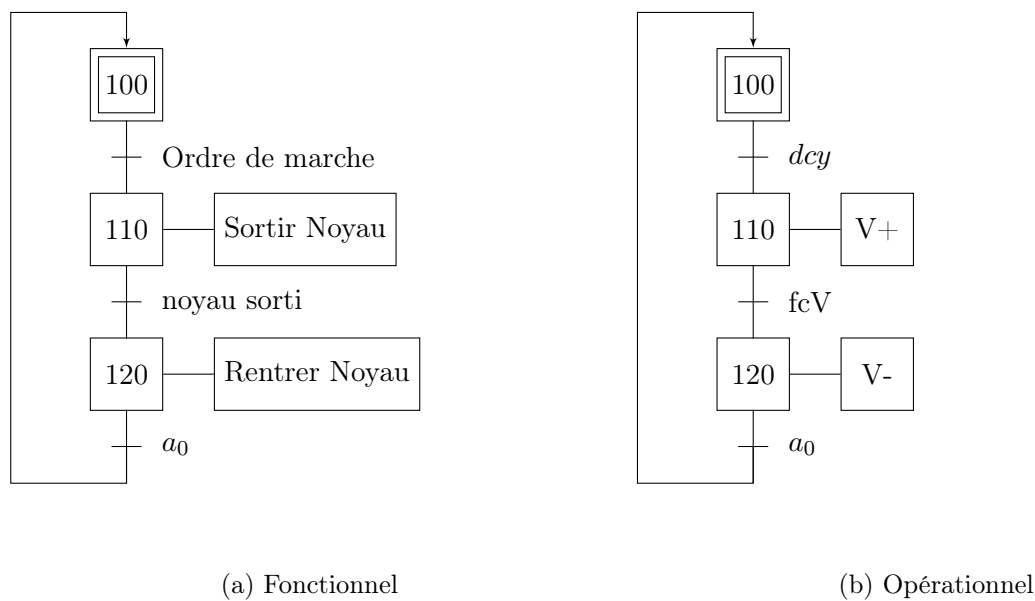


FIGURE 7: Grafcets fonctionnel et opérationnel

### 3 Le texte structuré

Le texte structuré est un langage de programmation proche du langage C.

Il consiste en une suite d'instructions séparées par un `;`. Ces instructions sont exécutées séquentiellement à chaque cycle automate.

#### 3.1 syntaxe du langage

##### 3.1.1 opérateurs

**Affectation :** On affecte une variable à l'aide de l'opérateur `:=`

- exemple : `compteur := 10;`

**Test :** Les opérateurs test (pour les structure IF par exemple) sont les suivants : `=` (égal) `>` (supérieur), `<` (inférieur), `<>` (différent).

##### 3.1.2 structure IF

La structure IF permet de faire un test et de n'exécuter une portion de code que si une condition est vérifiée.

```
IF Capteur3 = TRUE THEN
  N:=3;
ELSE
  N:=4;
END_IF
```

FIGURE 8: Exemple d'une structure IF en texte structuré.

### 3.2 Description d'un grafcet en texte structuré

De plus en plus, les constructeurs préconisent l'utilisation du texte structuré pour décrire les graf-cets.

La structure (Etape et transitions) d'un grafcet est décrit en texte structuré d'une manière analogue à la description d'une machine à état en langage C : A l'aide de la structure *switch - case*. Un exemple est donné Figure 9.

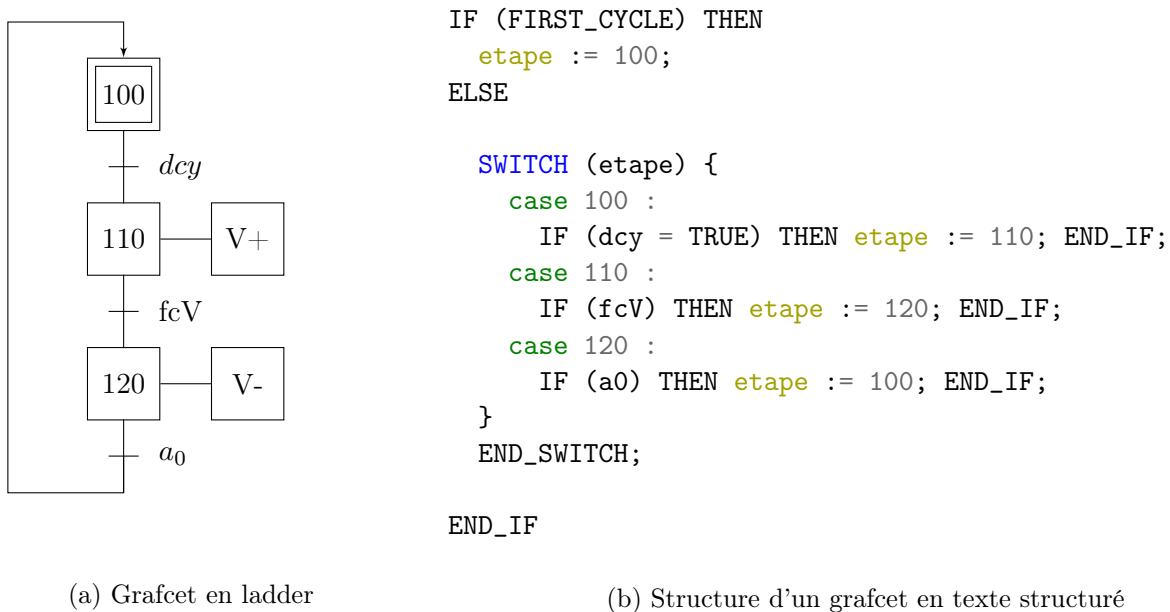


FIGURE 9: Structure d'un grafcet en texte structuré

## 4 Bonnes pratiques

Il est fortement conseillé de séparer la structure du programme de l'état des actionneurs.

Cela signifie que vous devrez créer un grafcet sans les actions associées (en SFC ou en ST selon les TPs). Une tâche à part gèrera les actions.

Un exemple du fichier action associé au grafcet de la figure 9 est donné ci-dessous (Figure 10). Dans ce code, on trouve deux écritures possible pour un même comportement.

L'accès à l'état (actif ou non) d'un grafcet codé en SFC dépend des logiciels. La programmation de grafcet sera vue dans les différents TPs sur les parties opératives.

```

IF (etape = 110) THEN Vout := 1 ; END_IF;
Vin := (etape = 120);

```

FIGURE 10: Fichier actions associé au grafcet de la Figure 9