

LA PROGRAMMATION D'UN AUTOMATE INDUSTRIEL

Table des matières

1	Le langage LADDER	2
1.1	Réseaux LADDER simples	2
1.1.1	Les contacts	2
1.1.2	Les bobines	2
1.1.3	Réseaux de base	3
2	Le langage GRAFCET	4
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	6
2.5	Règles de conception	7
2.6	Deux niveaux de GRAFCET	7
2.6.1	Niveau fonctionnel	8
2.6.2	Niveau opérationnel	8

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
Unity Pro EcoStructure	Tia Portal	CodeSys eCockpit	Niagara

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.

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

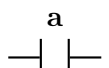
1.1.1 Les contacts

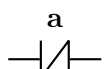
Les contacts représentent des entrées logiques (TOR).



À retenir

Il existe deux types de contacts :

 **Contact normalement ouvert** : actif lorsque la variable associée est à l'état 1.
 \Rightarrow Il représente donc la variable a

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



FIGURE 1: Réseaux LADDER de base

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 2



FIGURE 2: 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 3

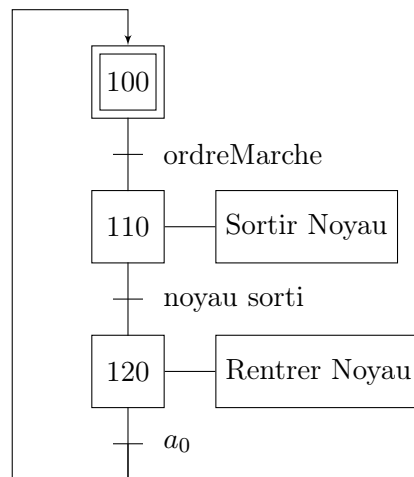


FIGURE 3: GRAFCET simple composé de trois étapes

2.1 Étapes et actions

Une étape est représentée par un carré comportant un numéro (Figure 4a). Elle est généralement associée à une (Figure 4b) ou plusieurs (Figure 4c) actions. Il est également possible d'ajouter une condition à une action (Figure 4d).



À retenir

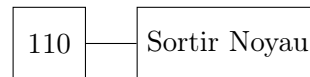
- Une action associée à une étape n'est effectuée que lorsque l'action associée est active.
- Une action conditionnelle est effectuée lorsque l'action est active et que la condition associée est vérifiée.

Il est également possible d'effectuer une action qu'au moment de l'activation d'une étape (Figure 4e). 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 toutes les 10 ms approximativement.

On peut représenter l'étape actuellement active dans un grafcet à l'aide d'une étoile comme illustré sur la Figure 4f.



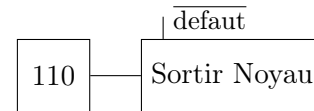
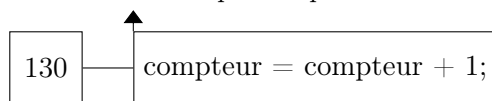
(a) Représentation d'une étape en GRAFCET



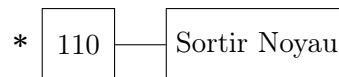
(b) Une étape et l'action associée



(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 courramment active

FIGURE 4: Etapes et action(s) associée(s)

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 5).

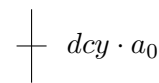


À retenir

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



(a) Une transition



(b) Une transition et sa réceptivité associée

FIGURE 5: Transitions et Réceptivité

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.

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. Comme la divergence en OU on rencontre aussi la 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 6.



À retenir

- Une convergence en **OU** commence **toujours** par une transition sur chacune de ses branches.
- Une convergence en **OU** se termine **toujours** par une transition sur chacune de ses branches.
- Les conditions d'une divergence doivent être exclusives.

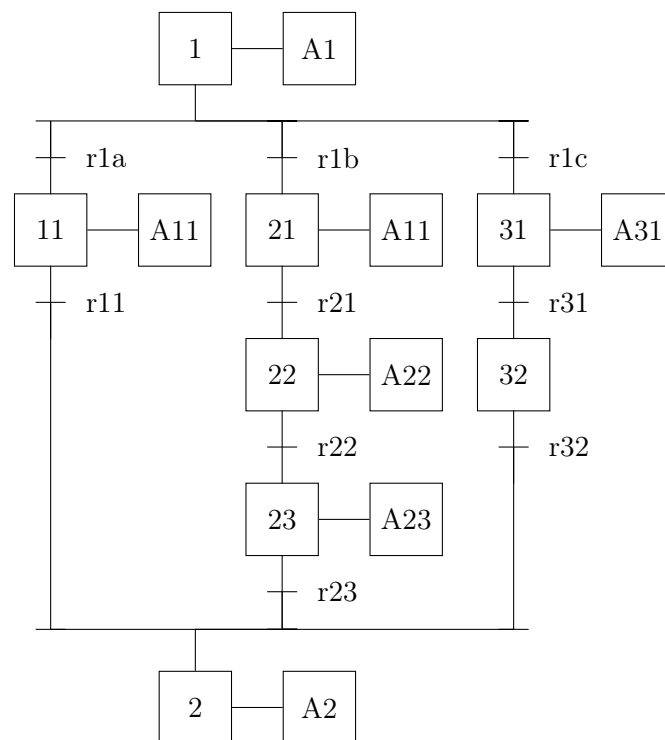


FIGURE 6: Exemple de divergence en **OU**. Une seule des branches est parcourue.

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, si plusieurs activités indépendantes pouvant se dérouler en parallèle. Le début d'une divergence en ET et la fin d'une convergence en ET d'un parallélisme structurel sont représentés par deux traits parallèles. Un exemple est donnée en Figure 7.

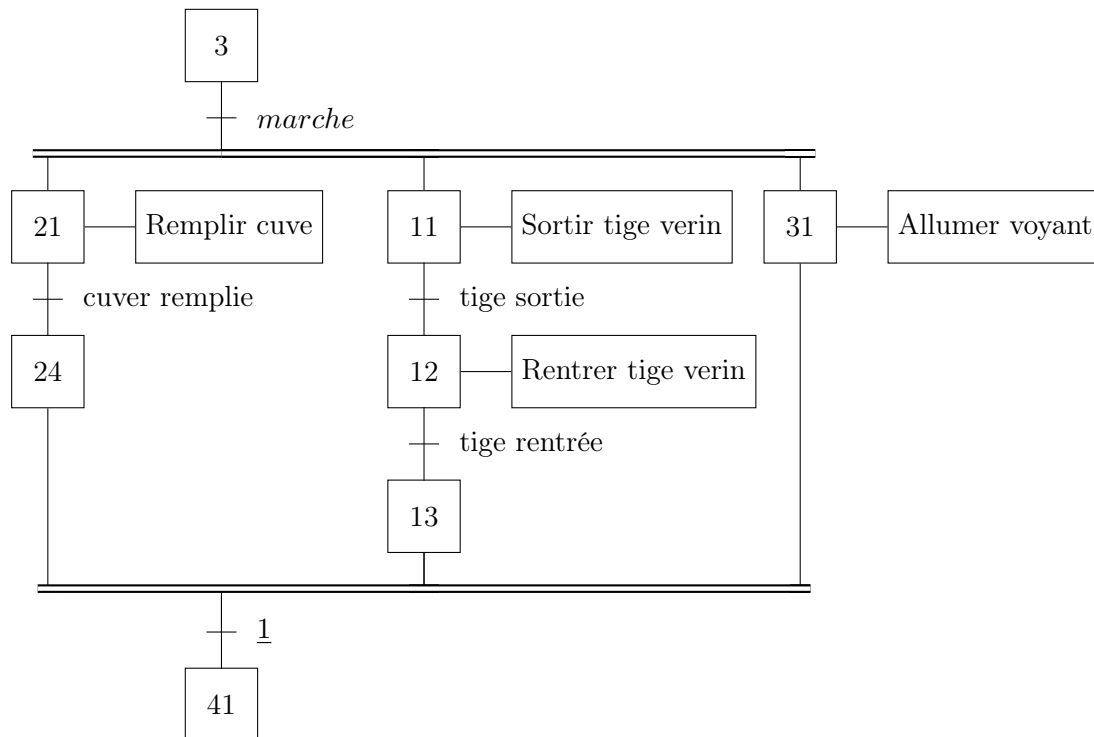


FIGURE 7: Exemple de divergence en **ET**. Les trois branches sont parcourue simultanément.



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.

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 vues sont décrits ci-dessous et illustrés en Figure ??.

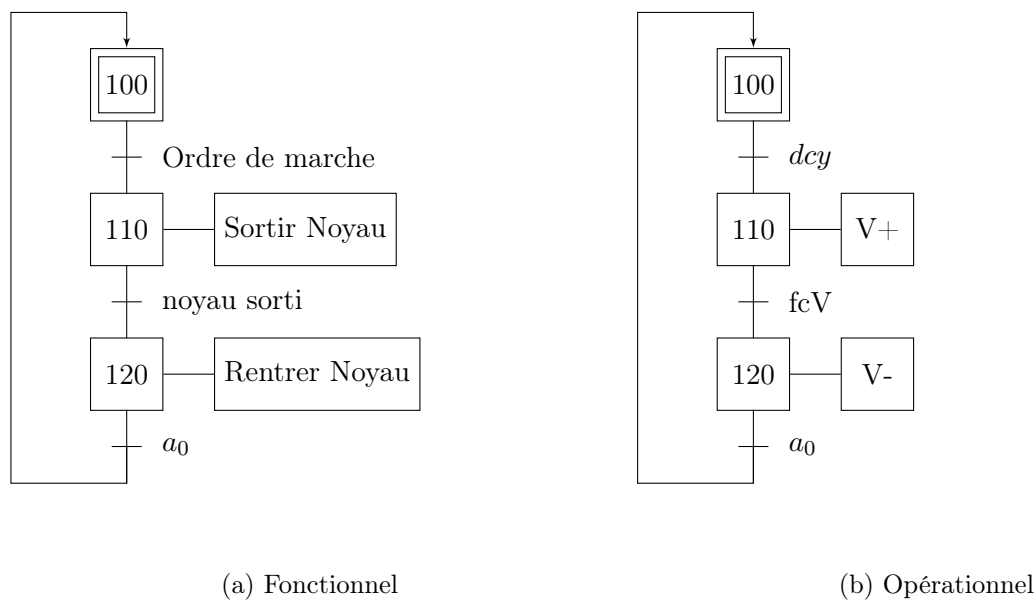


FIGURE 8: Grafcets fonctionnel et opérationnel

2.6.1 Niveau fonctionnel

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

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 contient le nom des variables associées aux entrées (pour les réceptivités) et aux sorties (pour les actions) de l'automate.