

## Support de Formation : GRAFCET

Source : WEB / Université de Poitiers – Institut Pprime

### Sommaire

<b>1 Synthèse des systèmes séquentiels : méthodologie .....</b>	<b>2</b>
<b>2 Présentation du GRAFCET .....</b>	<b>2</b>
<b>3 Fonction du GRAFCET .....</b>	<b>2</b>
<b>4 Définitions : GRACET .....</b>	<b>2</b>
<b>5 Les règles d'évolution .....</b>	<b>3</b>
<b>6 Les structures de base du GRAFCET .....</b>	<b>4</b>
6.1 Séquence unique.....	4
6.2 Séquences simultanées et alternatives .....	4
6.3 : Saut d'étapes.....	4
<b>7 Les actions associées .....</b>	<b>5</b>
7.1 Actions à niveaux : .....	5
7.2 Actions mémorisées : .....	5
7.3 Actions conditionnelles : .....	5
7.4 Actions temporisées .....	6
7.5 Actions simultanées.....	6
7.6 Grafcets hiérarchisés : .....	6
<b>8 Compter en langage grafcet .....</b>	<b>7</b>
<b>9 Chaîne fonctionnelle et points de vue d'un grafcet.....</b>	<b>8</b>
9.1 Etude d'un système automatisé : .....	8
9.2 Exemple : chaîne d'embouteillage .....	9
9.2.1 Grafcet d'un point de vue partie opérative.....	10
9.2.2 Grafcet d'un point de vue partie commande.....	11
9.2.3 Grafcet d'un point de vue partie commande pour l'automate industriel programmable.....	11
<b>10 Exemple d'application .....</b>	<b>12</b>
10.1 Grafcet d'étude .....	12
10.2 Programmation indirecte des règles d'évolution du GRAFCET – Machine d'état .....	13
10.2.1 Méthodologie.....	13
10.2.2 Séquence ou cycle d'une seule séquence .....	14
10.2.3 Sélection de séquences exclusive, regroupement de séquences .....	14
10.2.4 Activation de séquences parallèles, synchronisation de séquences .....	15
10.2.5 Programmation des réceptivités particulières.....	16
10.2.6 Programmation des actions.....	17
10.2.7 Exemple .....	20

## 1 Synthèse des systèmes séquentiels : méthodologie

Ce document présente une méthode de synthèse des systèmes séquentiels qui permet de passer rapidement d'une spécification GRAFCET ou diagramme d'états à la programmation de l'application. Cette méthode est basée sur la recherche des graphes d'états (situations accessibles) du système et n'est utilisable qu'avec des langages de programmation structurés (Structured Text ou C).



## 2 Présentation du GRAFCET

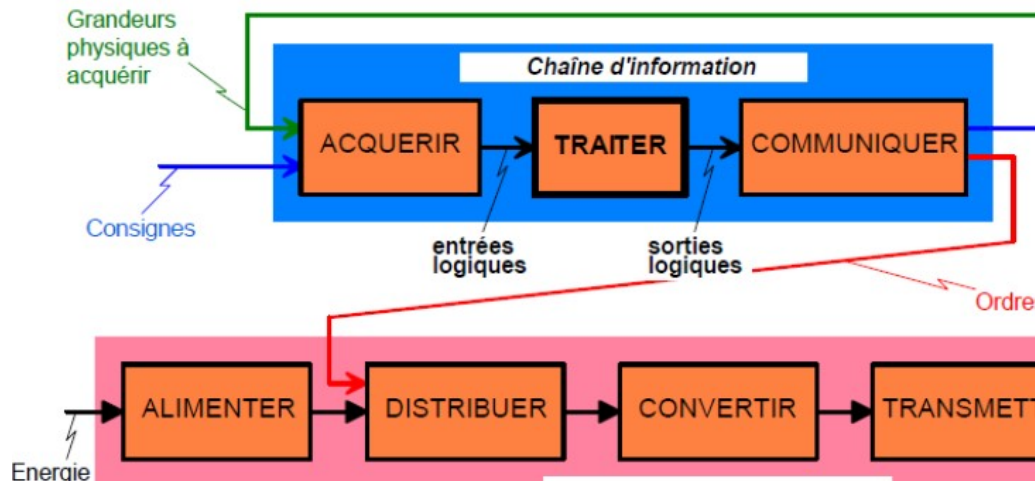
Le GRAFCET (GRAphe Fonctionnel de Commande des Etapes et Transitions) est l'outil de représentation graphique de tout système automatisé dont les évolutions peuvent s'exprimer séquentiellement. Il a été conçu par l'ADEPA (Agence pour le Développement de la Productique Appliquée à l'industrie).

C'est un langage clair, strict, permettant de traduire un fonctionnement sans ambiguïté.

Le GRAFCET est devenu à l'heure actuelle plus qu'un outil de description, c'est un langage de programmation graphique.

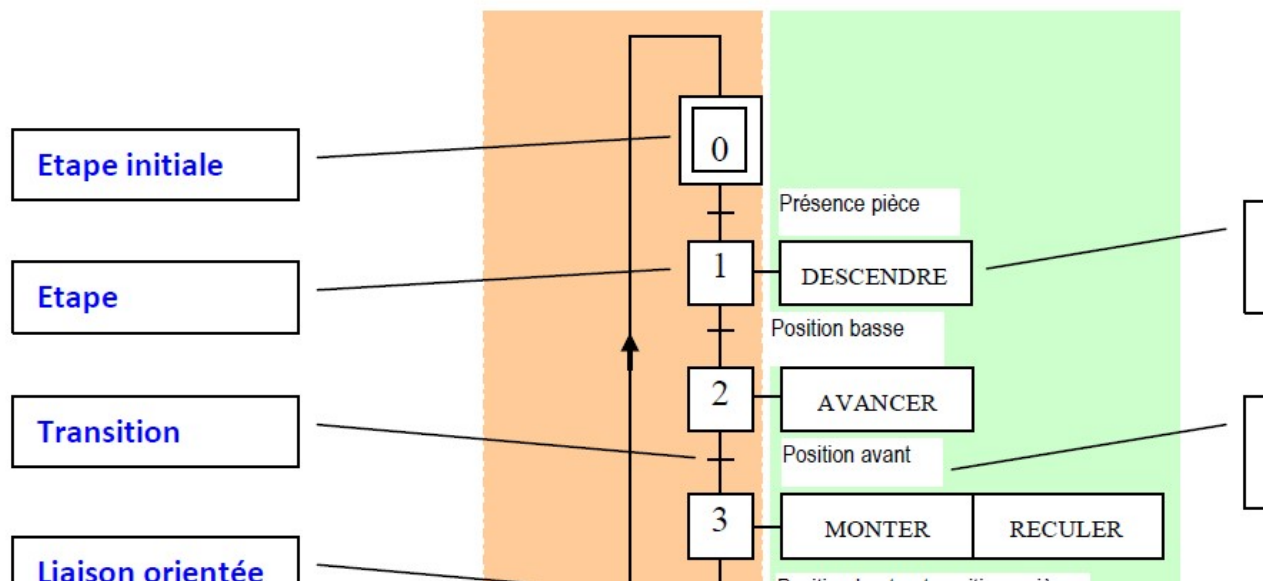
## 3 Fonction du GRAFCET

Les automates programmables industriels réalisent la fonction TRAITER de la chaîne d'information :



## 4 Définitions : GRAFCET

Le GRAFCET est un outil graphique de description des comportements d'un système logique. Il est composé d'étapes, de transitions et de liaisons :



Une **LIAISON** est un arc orienté (ne peut être parcouru que dans un sens). A une extrémité d'une liaison il y a une (et une seule) étape, à l'autre une transition. On la représente par un trait plein rectiligne, vertical ou horizontal.

Une **ETAPE** correspond à une phase durant laquelle on effectue une **ACTION** pendant une certaine durée. On numérote chaque étape par un entier positif, mais pas nécessairement croissant par pas de 1, il faut simplement que jamais deux étapes différentes n'aient le même numéro. Une étape est dite active lorsqu'elle correspond à une phase "en fonctionnement", c'est à dire qu'elle effectue l'action qui lui est associée. On représente quelquefois une étape active à un instant donné en dessinant un point à l'intérieur.

Une **TRANSITION** est une condition de passage d'une étape à une autre. Elle n'est que logique (dans son sens Vrai ou Faux), sans notion de durée. La condition est définie par une **RECEPTIVITE** qui est généralement une expression booléenne (c.à.d avec des ET et des OU) de l'état des capteurs.

## 5 Les règles d'évolution

### Règle 1 : Situation initiale

L'étape initiale caractérise le comportement de la partie commande d'un système en début de cycle. Elle correspond généralement à une position d'attente. L'étape initiale est activée sans condition en début de cycle. Il peut y avoir plusieurs étapes initiales dans un même grafcet.

### Règle 2 : Franchissement d'une transition

Une transition est validée si toutes les étapes immédiatement précédentes sont actives. L'évolution du grafcet correspond au franchissement d'une transition qui se produit sous deux conditions :

- si cette transition est validée
- si la réceptivité associée à cette transition est vraie

Si ces deux conditions sont réunies, la transition devient franchissable et est obligatoirement franchie.

### Règle 3 : Evolution des étapes actives

Le franchissement d'une transition entraîne simultanément l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes celles immédiatement précédentes.

### Règle 4 : Evolutions simultanées

Plusieurs transitions simultanément franchissables sont simultanément franchies.

### Règle 5 : Activations et désactivations simultanées

Si, au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste active.

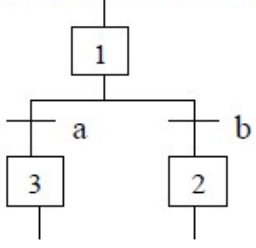
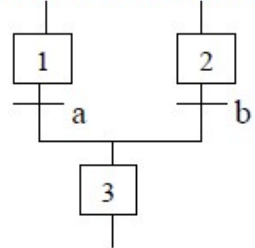
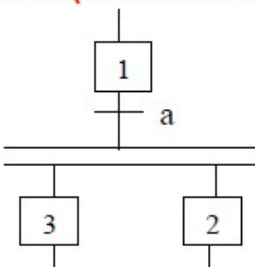
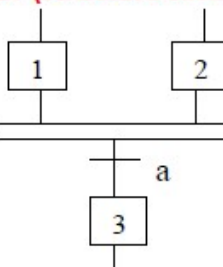
## 6 Les structures de base du GRAFCET

### 6.1 Séquence unique

C'est une suite d'étapes pouvant être activées les unes après les autres

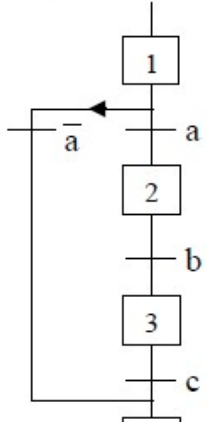
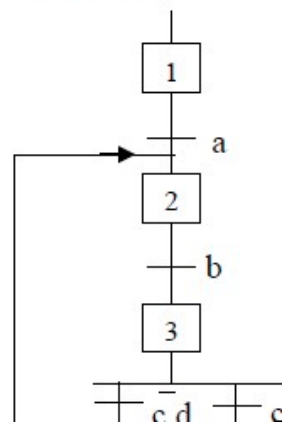
### 6.2 Séquences simultanées et alternatives

Plusieurs séquences sont actives en même temps, après le franchissement d'une transition.

<p><b>Divergence en OU (structure alternative) :</b></p>  <p>Si 1 active et si a seul, alors désactivation de 1 et activation de 3, 2 inchangé. Si a et b puis 1 active alors désactivation 1, activation 2 et 3 quel que soit leur état précédent. (règle 4)</p>	<p><b>Convergence en OU (structure alternative) :</b></p>  <p>Si 1 active et a sans b, alors désactivation de 1, 2 reste inchangé. Si 1 et 2 et a et b alors 3 seule active</p>
<p><b>Divergence en ET (structure simultanée) :</b></p> 	<p><b>Convergence en ET (structure simultanée) :</b></p> 

### 6.3 : Saut d'étapes

Il permet de sauter une ou plusieurs étapes :

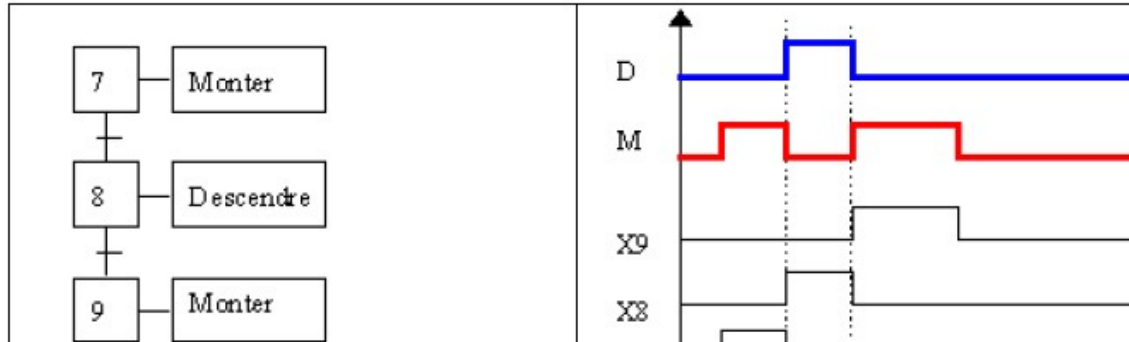
<p><b>Boucle Si Alors</b></p> 	<p><b>Boucle Répéter Tant que</b></p> 
---	---

## 7 Les actions associées

Les actions sont précisées dans un cadre lié à l'étape, de manière générale, l'action n'est vraie que si l'étape est active. La norme européenne CEI précise la nature de l'action par une lettre précisant la nature de l'action.

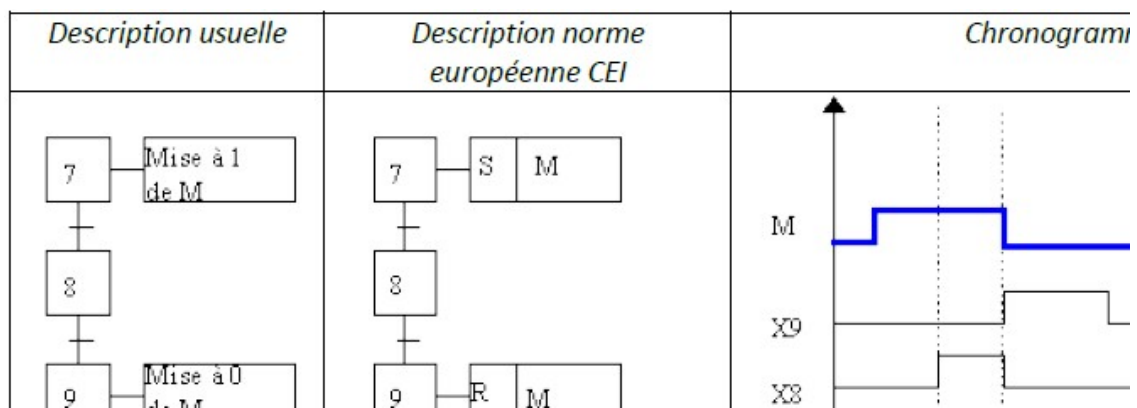
### 7.1 Actions à niveaux :

La sortie n'est vraie que si l'étape est active



### 7.2 Actions mémorisées :

On distingue la mise à 1 et la mise à 0 de l'action

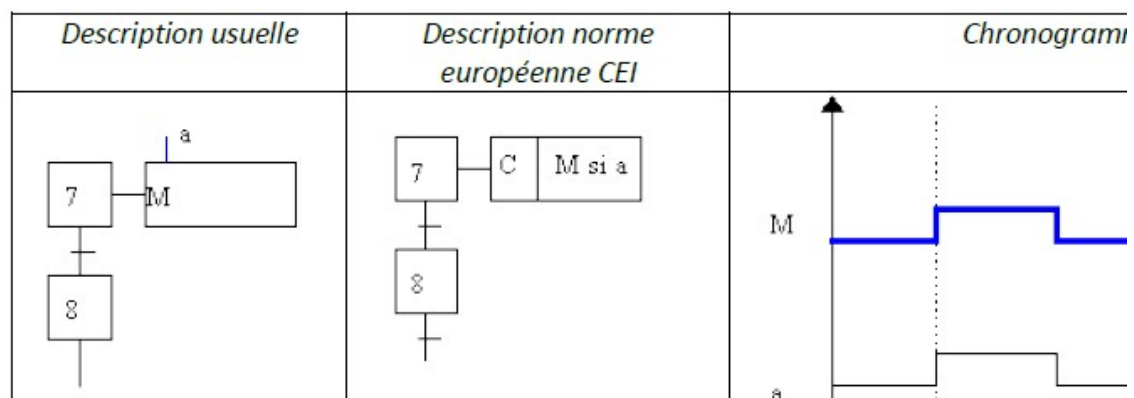


La norme CEI précise la mise à 1 et la mise à 0 par les lettres S (set) et R (reset).

**A** Attention il est interdit d'utiliser une action mémorisée avec des sorties automate (risque de danger à la remise sous tension suite à un arrêt imprévu du système exemple arrêt d'urgence...).

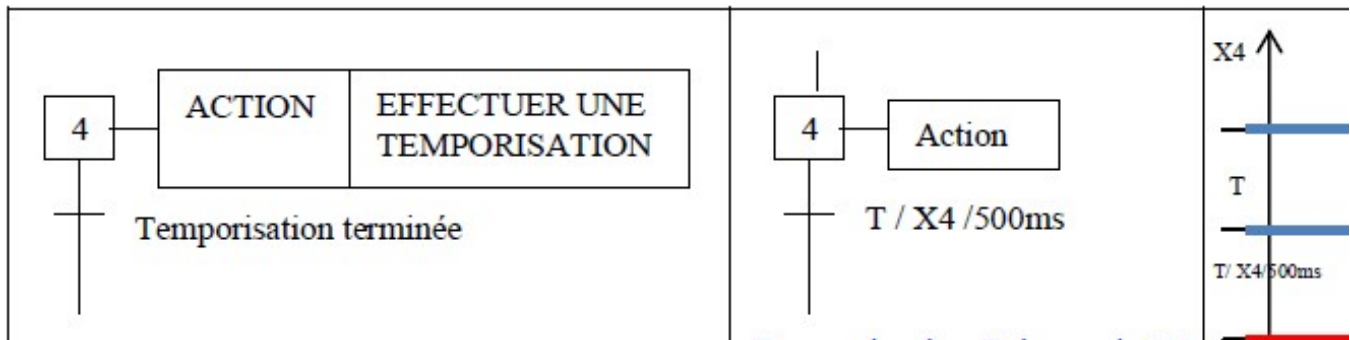
### 7.3 Actions conditionnelles :

Une action Conditionnelle n'est vraie que si l'étape est active ET la condition est vraie.

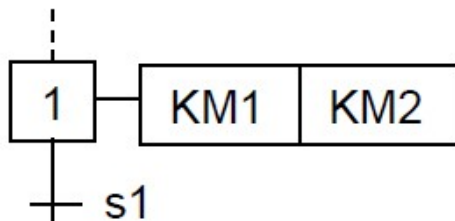


La norme CEI précise les actions conditionnelles par un C.

## 7.4 Actions temporisées



## 7.5 Actions simultanées

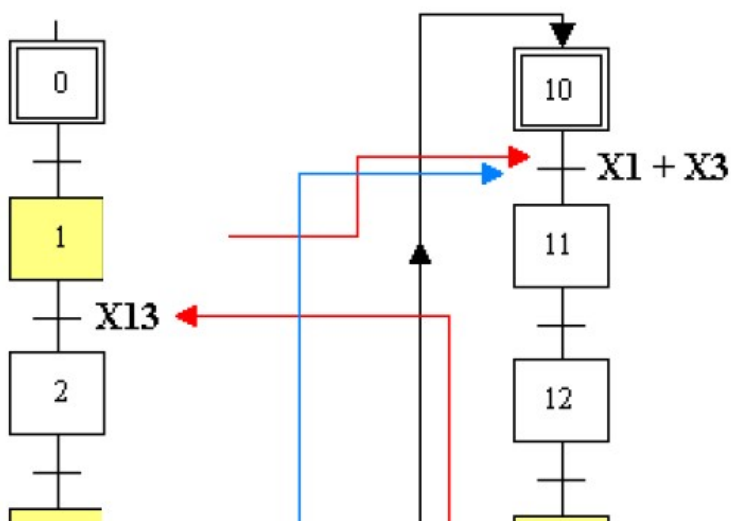


Plusieurs actions sont commandées  
l'étape 1 et le reste durant sont activ

## 7.6 Grafcets hiérarchisés :

Grafcet Principal

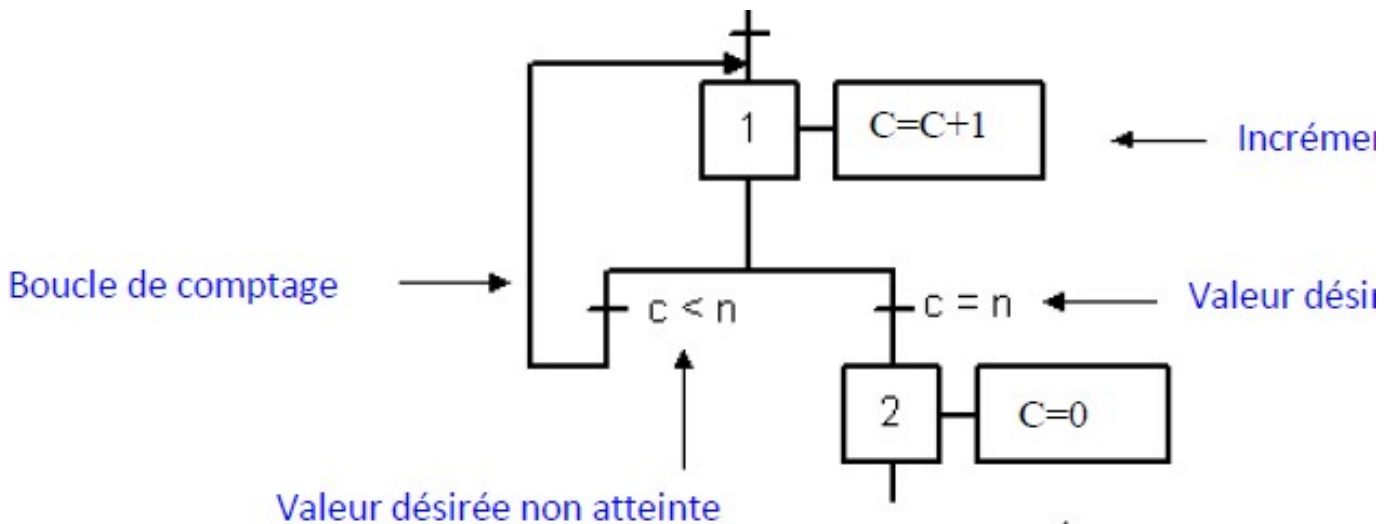
Grafcet T1



Un Grafcet principal commande  
Grafcets de tâches. Très utilisée  
simplifiée des Grafcets comp

## 8 Compter en langage grafcet

Les grafcets sont lus par les automates de façon cyclique. Le compteur étant, dans notre cas, une information interne à l'automate, il faudra veiller à l'incrémenter ou le décrémenter au travers d'étapes conditionnées sous peine de le voir évoluer de manière aléatoire.



Grafcet partie opérative	Grafcet partie commande
<div>1</div> <div>Incrémenter le compteur</div>	<div>1</div> <div>C1 = C1 + 1</div>
<div>1</div> <div>Décrémenter le compteur</div>	<div>1</div> <div>C1 = C1 - 1</div>

Les différents types de tests réalisables dans les réceptivités sont les suivants :

Type de test	Syntaxe	Type de test	
égal	=	supérieur	
différent	<>	inférieur ou égal	

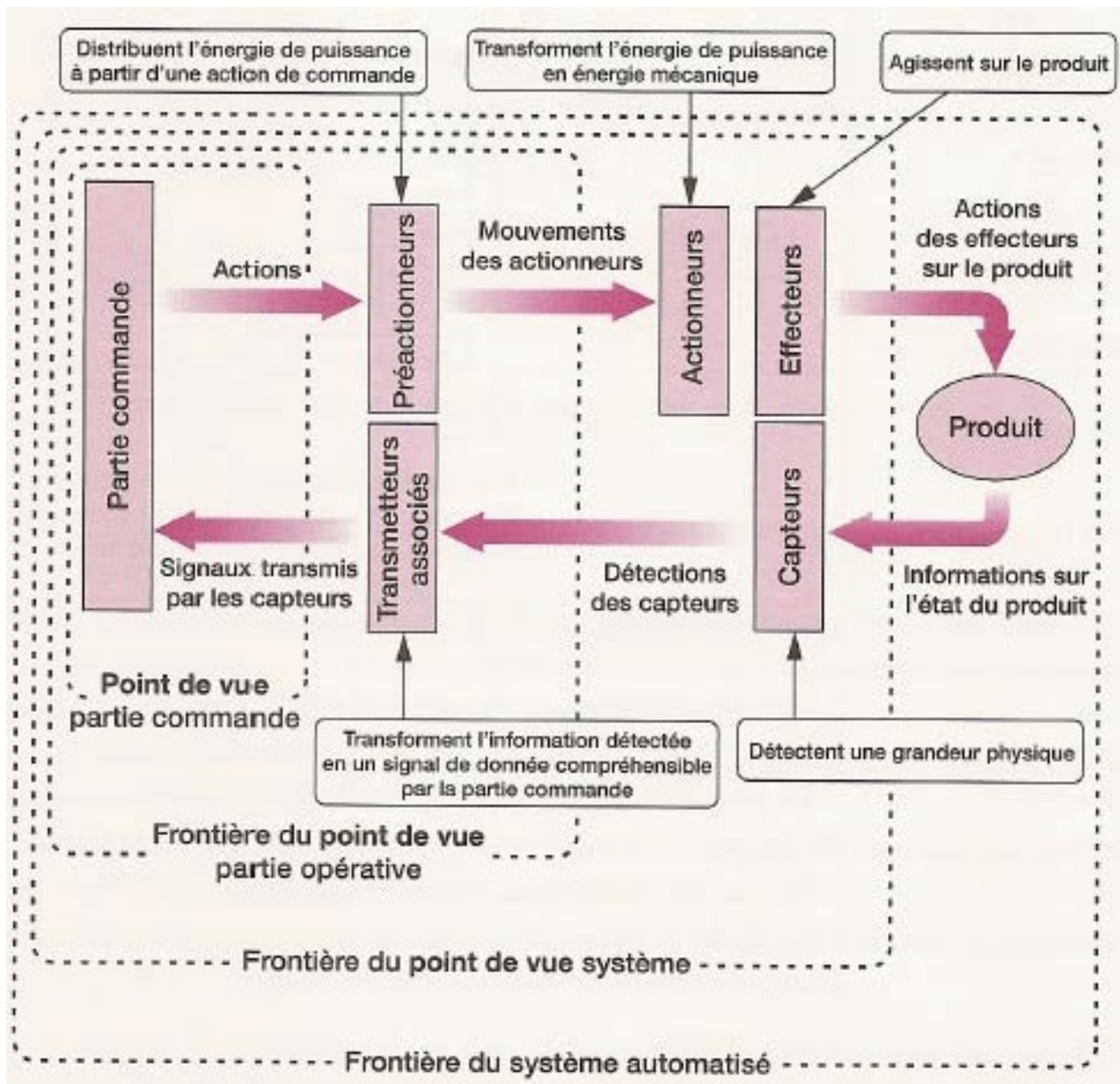


## 9 Chaîne fonctionnelle et points de vue d'un grafcet

### 9.1 Etude d'un système automatisé :

On distingue 3 phases dans l'étude d'un système automatisé :

- ↳ le point de vue système,
- ↳ le point de vue partie opérative,
- ↳ le point de vue partie commande.





### Le procédé :

Le procédé est l'ensemble des fonctions successives exécutées sur un même produit au cours de sa fabrication

### Le processus :

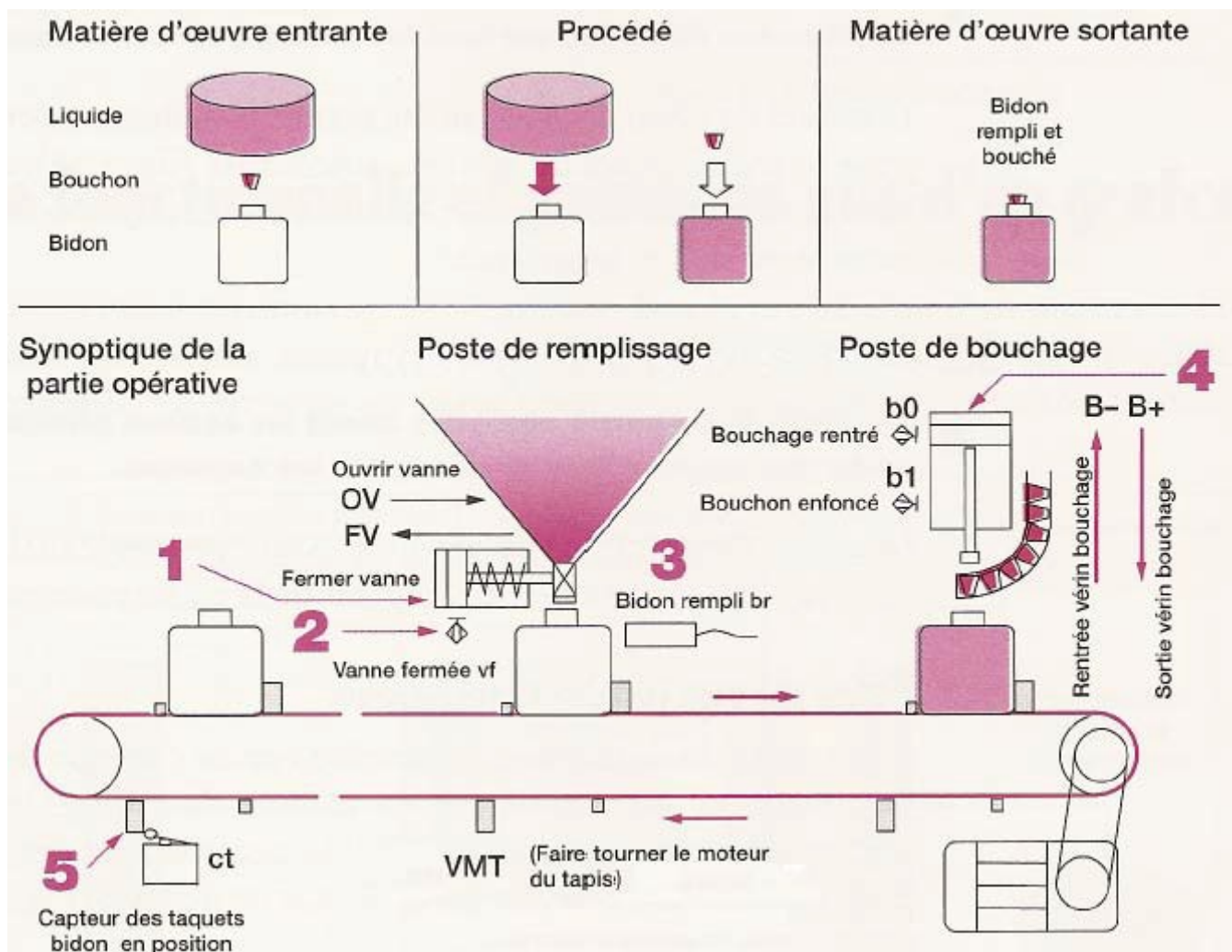
Le processus est l'organisation du procédé. C'est la succession des fonctions simultanées réalisées sur tous les produits présents dans le système automatisé.

✓ **Le point de vue système** décrit le comportement du système vis-à-vis du produit. Il montre l'enchaînement des actions sur le produit.

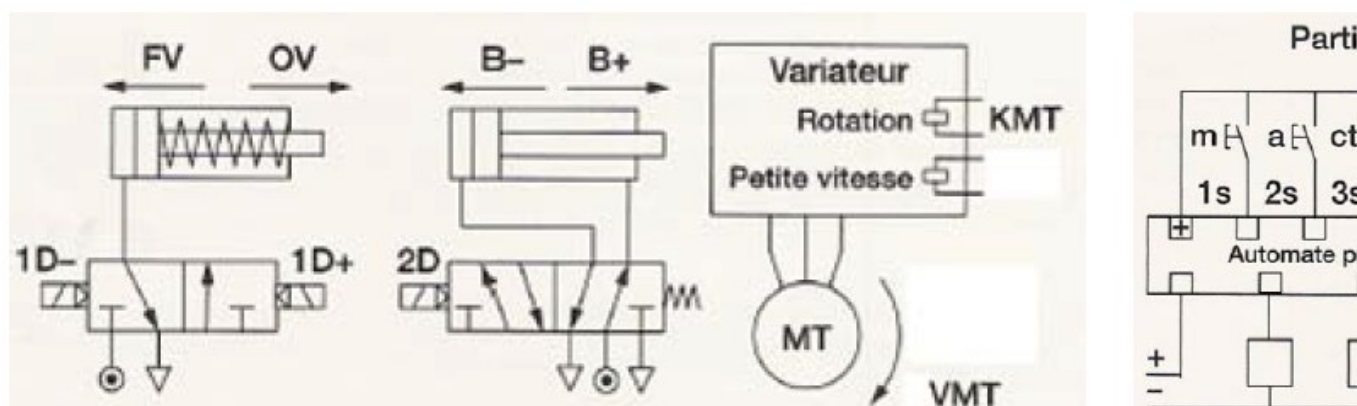
✓ **Le point de vue partie opérative** décrit les actions produites par les actionneurs à partir des informations acquises par les capteurs.

✓ **Le point de vue partie commande** décrit le comportement de la partie commande par rapport à la partie opérative en tenant compte du choix de la technologie employée. Un schéma de câblage (électrique et pneumatique) décrit le raccordement des transmetteurs et des préactionneurs à la partie commande.

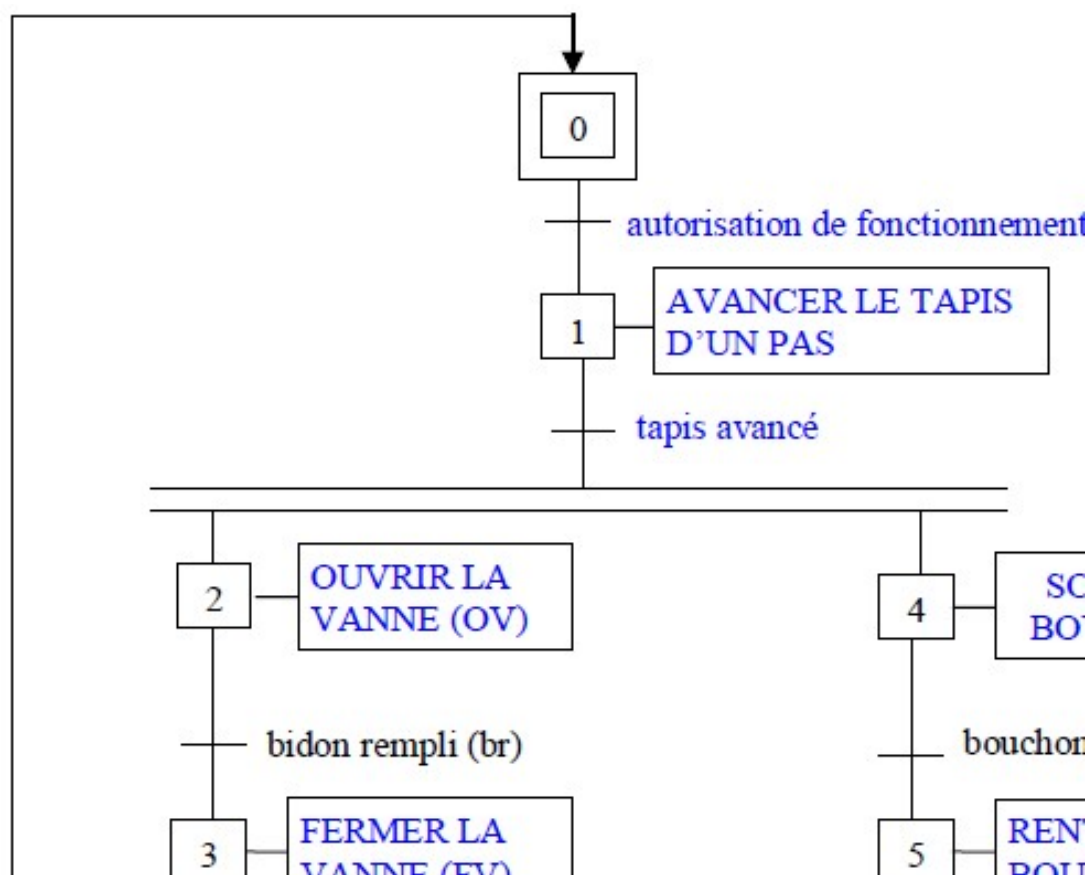
## 9.2 Exemple : chaîne d'embouteillage



- 1) Le remplissage se fait par soutirage. L'ouverture et la fermeture de la vanne sont effectuées par un vérin pneumatique.
- 2) Un capteur « vanne\_fermée » indique la position complètement fermée de la vanne.
- 3) Un capteur « bidon\_rempli » permet de contrôler le niveau de remplissage de façon satisfaisante.
- 4) Le bouchage est assuré par un vérin presseur muni de deux capteurs fin de course b0 et b1
- 5) Le transfert des bidons est assuré par un convoyeur à taquets permettant un positionnement correct des bidons. Le capteur « bidon\_en\_position » informe la partie commande de l'arrêt du tapis.
- 6) Pupitre opérateur : « m » bouton marche et « a » bouton arrêt.



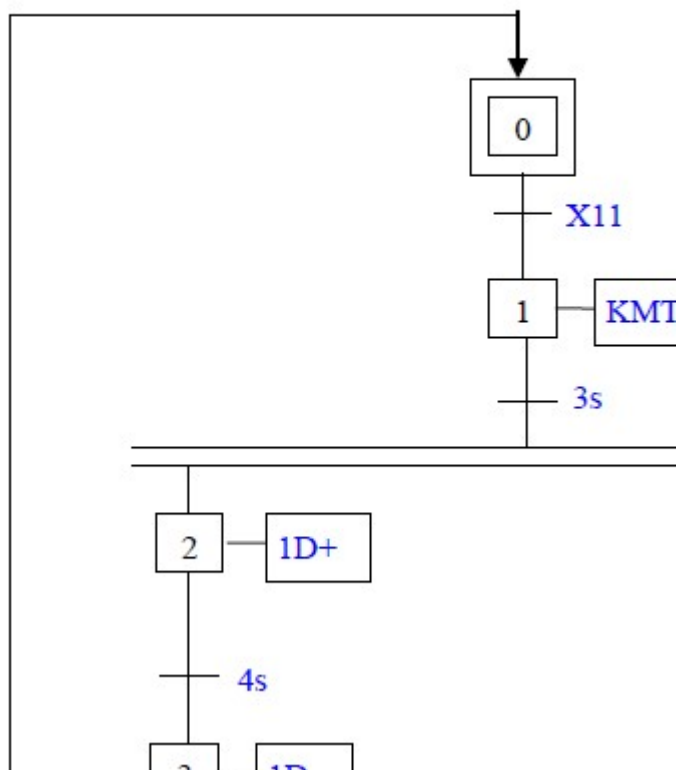
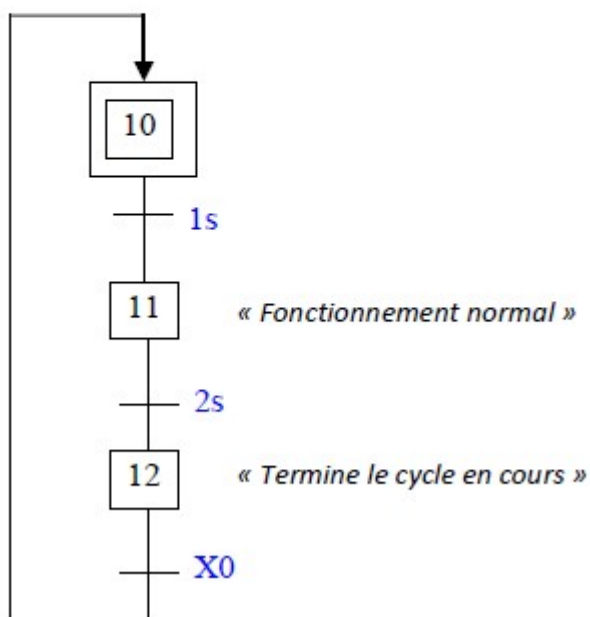
### 9.2.1 Grafset d'un point de vue partie opérative



### 9.2.2 Grafset d'un point de vue partie commande

Grafset de fonctionnement normal

Grafset de conduite : GC



### 9.2.3 Grafset d'un point de vue partie commande pour l'automate industriel programmable

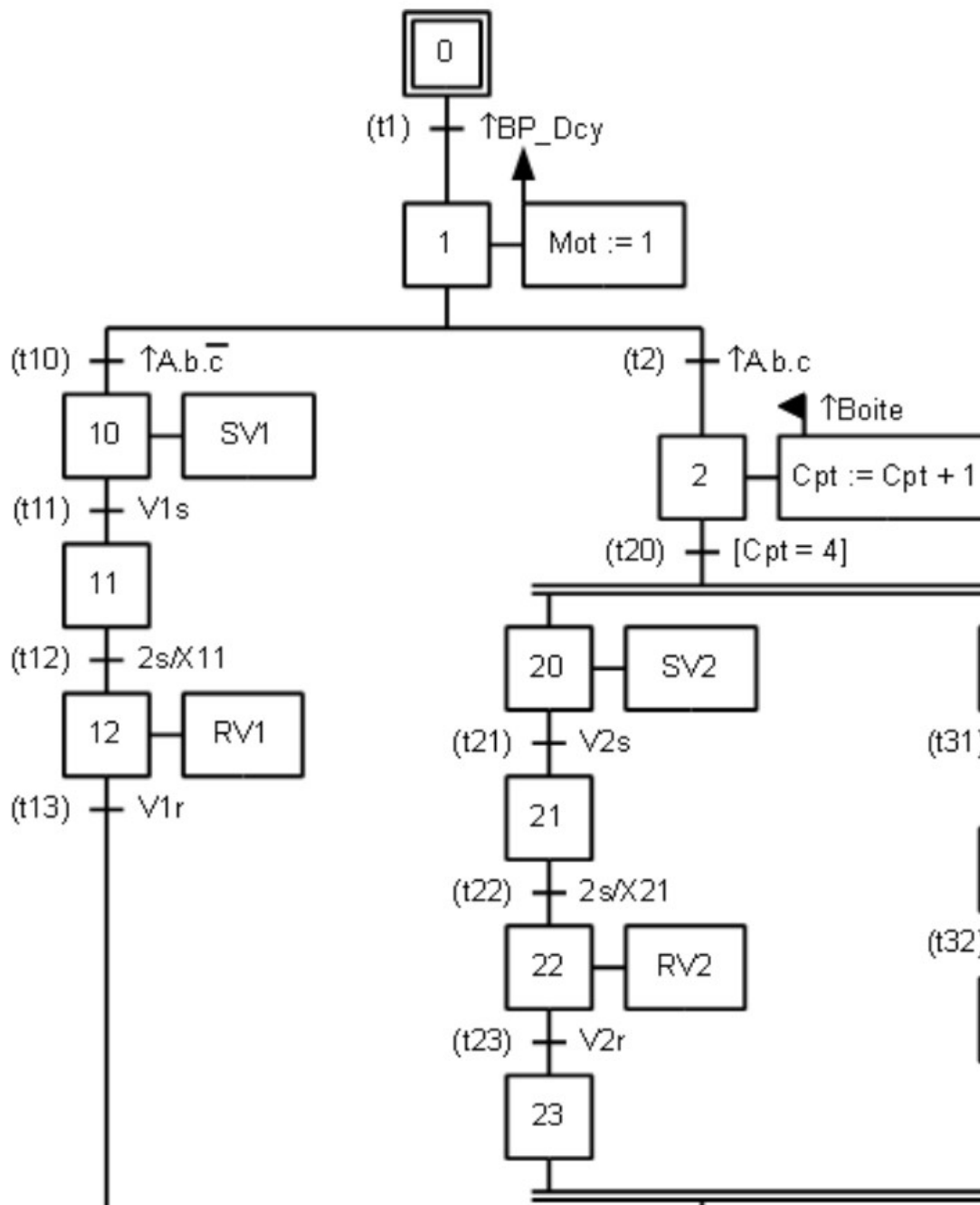
Pour pouvoir implanter le grafset dans l'automate programmable, il faut affecter les mnémoniques utilisés pour les capteurs et les préactionneurs aux entrées et sortie de l'automate.

Désignation (capteurs, préactionneurs)		Mnémonique
Entrées (Capteurs)	Marche (m)	1s
	Arrêt (a)	2s
	capteur taquet (ct)	3s
	bidon rempli (br)	4s
	vanne fermée (vf)	5s
	Bouchage rentré (b0)	6s0
	Bouchon enfoncé (b1)	6s1
	Rotation du moteur	KMT
	Formation vidéo (fvp)	1D

## 10 Exemple d'application

### 10.1 Grafset d'étude

La spécification ci-dessous servira de base à la réalisation du programme détaillé.



## 10.2 Programmation indirecte des règles d'évolution du GRAFCET – Machine d'état

Cette méthode de programmation nécessite une transformation de la spécification GRAFCET en un ensemble de graphe d'états (situations accessibles) ou de GRAFCETs à étape active unique. L'avantage de cette méthode est d'exclure tous les aléas de fonctionnement. L'inconvénient de cette méthode est que le travail de transformation, s'il est mal effectué, peut induire des modifications du cycle de fonctionnement.

### 10.2.1 Méthodologie

Dans cette méthode nous transformerons la spécification GRAFCET en un ensemble de graphes et de sous-graphes d'état. Pour cela, nous rechercherons l'ensemble des situations accessibles à partir de la situation initiale.

Les différents états de chacun des graphes seront représentés par une variable entière. Chaque état correspondra à une valeur unique de la variable d'état. Chaque graphe d'état sera programmé en utilisant une structure de choix (case). Nous obtenons le pseudo code suivant :

```
// Gestion de l'évolution du graphe d'état
DECIDER SUR Etat ENTRE
    « valeur de l'état » :
        SI Rec1 ALORS
            // Programmation des actions mémorisées ici ou à
            Décider sur
            Actions mémorisées à la désactivation de l'état c
            Actions mémorisé à l'activation de l'état suivant
            Etat = « valeur de l'état suivant »
        SINON SI Rec2 ALORS
            // [...]
        FIN DE SI
    « valeur de l'état » :
        // [...]
FIN DE DECIDER SUR
... ..
```

**Attention :** Cette méthode respecte parfaitement les règles d'évolution du GRAFCET à la condition impérative de respecter la hiérarchie entre les différents graphes d'état. Remarques :

Le temps de franchissement des transitions est égal au temps de cycle ou la période de la tâche. Ce temps représente le temps de réaction (de réponse) de votre système. ]

L'état de chaque état (étape dans certain cas) est représenté par :

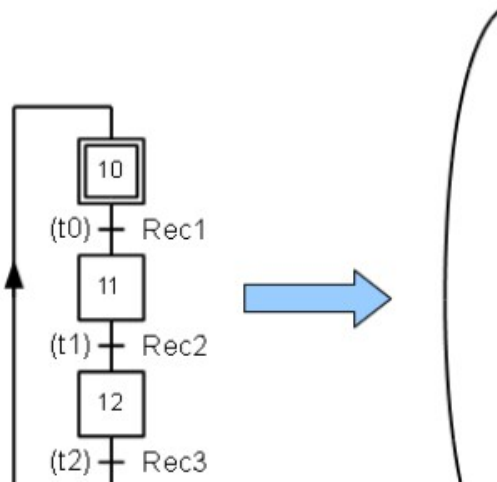
- Etat  $\neq$  valeur représentant l'état « i »  $\Rightarrow$  Etat « i » non actif,
- Etat = valeur représentant l'état « i »  $\Rightarrow$  Etat « i » actif,
- A l'extérieur du « DECIDER SUR » :
  - o utiliser le bloc fonctionnel « R\_TRIG » pour déterminer l'instant d'activation de l'état « i » (CLK := (Etat = valeur représentant l'état « i »)),
  - o utiliser le bloc fonctionnel « F\_TRIG » pour déterminer l'instant d'activation de l'état « i » (CLK := (Etat = valeur représentant l'état « i »)),
- A l'intérieur du « DECIDER SUR », voir pseudo code ci-dessus.

Nous allons, dans les paragraphes suivants, montrer comment programmer les différentes structures de GRAFCET (cf. norme IEC 60848).



### 10.2.2 Séquence ou cycle d'une seule séquence

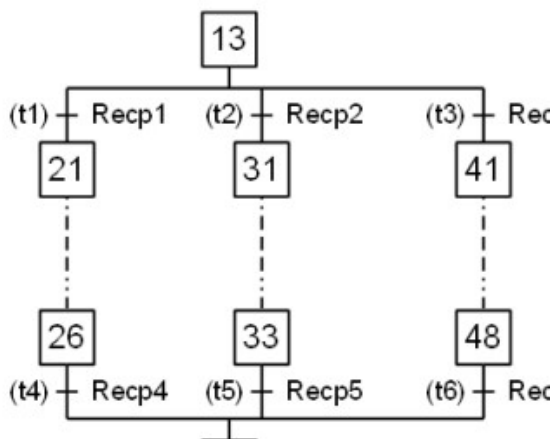
Dans le cas d'une séquence ou d'un cycle d'une seule séquence chaque étape représente un état (une situation) du GRAFCET. Aucune transformation de la spécification GRAFCET n'est nécessaire. On utilisera une seule variable d'état et chaque étape sera représentée par une valeur unique. Pour l'exemple générique ci-dessous nous avons :



Pseudo code	Programmation en Stru
<p>Soit « Etat » notre variable d'état.  Etat = 10 → étape 10 active,  Etat = 11 → étape 11 active,  Etat = 12 → étape 12 active.</p> <pre>// Initialisation Etat ← 10 // Exécution cyclique // Gestion du graphe DECIDER SUR Etat ENTRE 10 :     SI Rec1 ALORS         Etat ← 11     FIN DE SI 11 :     SI Rec2 ALORS         Etat ← 12     FIN DE SI 12 :     SI Rec3 ALORS         Etat ← 10     FIN DE SI</pre>	<p>Soit « Etat » notre variable d'état.  Etat = 10 → étape 10 active,  Etat = 11 → étape 11 active,  Etat = 12 → étape 12 active.</p> <pre>PROGRAM INIT     Etat := 10 ; END_PROGRAM  PROGRAM CYCLIC     // Gestion du graphe     CASE Etat OF         10 :             IF Rec1 THEN                 Etat := 11             END_IF         11 :             IF Rec2 THEN                 Etat := 12             END_IF         12 :             IF Rec3 THEN                 Etat := 10             END_IF     END_CASE</pre>

### 10.2.3 Sélection de séquences exclusive, regroupement de séquences

Dans le cas d'une sélection exclusive de séquences, chaque étape représente un état (une situation) du GRAFCET. Aucune transformation de la spécification GRAFCET n'est nécessaire. On utilisera une seule variable d'état et chaque étape sera représentée par une valeur unique. Pour l'exemple générique ci-dessous nous avons :



Pseudo code	Programmation en Stru
<p>Les réceptivités « Recp1 », « Recp2 » et « Recp3 » sont exclusives.</p> <pre>// Exécution cyclique // Gestion du graphe DECIDER SUR Etat ENTRE // [...] 13 :     SI Recp1 ALORS         Etat ← 21     SINON SI Recp2 ALORS         Etat ← 31     SINON SI Recp3 ALORS         Etat ← 41     FIN DE SI // [...] 26 :     SI Recp4 ALORS         Etat ← 14     FIN DE SI 33 :     SI Recp5 ALORS         Etat ← 14     FIN DE SI 48 :     SI Recp6 ALORS</pre>	<p>Les réceptivités « Recp1 », « Recp2 » et « Recp3 » sont exclusives.</p> <pre>PROGRAM CYCLIC     // Gestion du graphe     CASE Etat OF         // [...]         13 :             IF Recp1 THEN                 Etat := 21             ELSEIF Recp2 THEN                 Etat := 31             ELSEIF Recp3 THEN                 Etat := 41             END_IF         // [...]         26 :             IF Recp4 THEN                 Etat := 14             END_IF         33 :             IF Recp5 THEN                 Etat := 14             END_IF         48 :             IF Recp6 THEN</pre>



### 10.2.4 Activation de séquences parallèles, synchronisation de séquences

Chaque séquence de l'activation de séquences parallèles donnera naissance à un sous graphe d'état autonome. L'ensemble des séquences sera représenté par un état (état des séquences parallèles) du graphe principal.

**SI** condition d'activation **ALORS**

Activation du premier état de chaque sous graphe

Activation de l'état des séquences parallèles du graphe principal

**FIN DE SI**

La synchronisation de séquences aura lieu en faisant un « et logique » entre la condition de synchronisation et les états de fin de chaque séquence.

Etat des séquences parallèles :

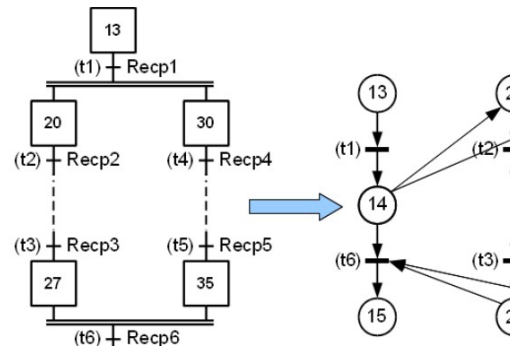
**SI** Dernier état de chaque séquence parallèle **ET** condition de synchronisation **ALORS**

Désactivation des états des différents sous graphes

Activation de l'état suivant

**FIN DE SI**

Exemple :



Pseudo code	Programmation en Str
<pre>// Exécution cyclique // Gestion du graphe DECIDER SUR Etat ENTRE // [...] 13 :     SI Recp1 ALORS         Etat ← 14         Br1 ← 20         Br2 ← 30     FIN DE SI 14 : // Synchronisation de séquences SI (Br1=27) ET (Br2=35) ET Recp6 ALORS     Etat ← 15     Br1 ← 0     Br2 ← 0 FIN DE SI // Sous graphe d'état 20-27 DECIDER SUR Br1 ENTRE 20 :     SI Recp2 ALORS         Br1 ← 21     FIN DE SI 21 :     // [...] 26 :     SI Recp3 ALORS         Br1 ← 27     FIN DE SI FIN DE DECIDER SUR // Sous graphe d'état 30-35 DECIDER SUR Br2 ENTRE 30 :     SI Recp4 ALORS         Br2 ← 31     FIN DE SI 31 :     // [...] 34 :</pre>	<pre>PROGRAM CYCLIC // Gestion du graphe CASE Etat OF // [...] 13 :     IF Recp1 THEN         Etat := 14         Br1 := 20 ;         Br2 := 30 ;     END_IF 14 : // Synchronisation de séquences IF (Br1=27) AND Recp6 THEN     Etat := 15     Br1 := 0 ;     Br2 := 0 ; END_IF // Sous graphe d'état 20-27 CASE Br1 OF 20 :     IF Recp         Br1     END_IF 21 :     // [...] 26 :     IF Recp         Br1     END_IF END_CASE // Sous graphe d'état 30-35 CASE Br2 OF 30 :     IF Recp         Br2     END_IF 31 :     // [...] 34 :     IF Recp</pre>

### 10.2.5 Programmation des réceptivités particulières

Le tableau ci-dessous présente la façon de programmer les différents types de réceptivités définies dans la norme GRAFCET. Les tests ci-dessous sont exécutés à l'intérieur du « DECIDER SUR », ce sont les conditions de passage d'un état à un autre. L'exécution des blocs fonctionnels doit se faire à l'extérieur de la structure « DECIDER SUR ».

Réceptivité	Programmation (en ST)
$(tj) \vdash [Cpt \leq 4]$	<pre>IF (Cpt &lt;= 4) THEN   // [...]</pre>
$(tj) \vdash \uparrow a$	<pre>(* R_TRIG_0 est une instance du FB standard R_TRIG_0(CLK := a) ; IF R_TRIG_0.Q THEN   // [...]</pre>
$(tj) \vdash \downarrow b$	<pre>(* F_TRIG_0 est une instance du FB standard F_TRIG_0(CLK := b) ; IF F_TRIG_0.Q THEN   // [...]</pre>
$(tj) \vdash 4s/a$	<pre>(* TON_0 est une instance du FB standard TON_0(IN := a, PT := T#4s) ; IF TON_0.Q THEN   // [...]</pre>
$(tj) \vdash 4s/a/1s$	<pre>(* TON_0 est une instance du FB standard (* TOF_0 est une instance du FB standard TON_0(IN := a, PT := T#4s) ; TOF_0(IN := TON_0.Q, PT := T#4s) ; IF TOF_0.Q THEN   // [...]</pre>

## 10.2.6 Programmation des actions

### 10.2.6.1 Actions continues

La norme IEC 60848 dit :

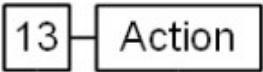
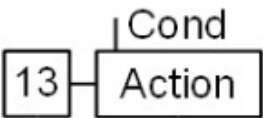

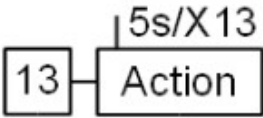
- En mode continu, c'est la situation du GRAFCET qui permet d'indiquer qu'une variable de sortie est vraie ou fausse. Pour une situation **stable** du GRAFCET, les valeurs des sorties relatives aux actions continues sont assignées :
  - ✓ à la valeur vraie, pour toutes les sorties relatives aux actions associées aux étapes actives et pour lesquelles les conditions éventuelles d'assignation sont vérifiées,
  - ✓ à la valeur fausse, pour toutes les autres sorties.

La méthode présentée ici nous oblige de faire ce travail dans la transformation de la spécification GRAFCET en graphes et sous-graphe d'états.

- En mode continu il y a continuité d'action. Dans le cas d'une même action associée à plusieurs étapes successives, le GRAFCET impose « la continuité d'action » c'est-à-dire qu'il n'existe pas un instant (franchissement d'une transition) même aussi court que l'on veut pendant lequel l'action n'a plus lieu.

La méthode présentée assure la continuité d'action.



Le tableau ci-dessous présente la façon de programmer les différents types d'actions continues définies dans la norme GRAFCET.

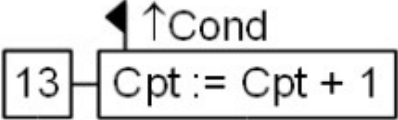
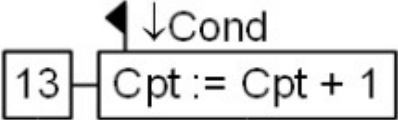
<i>Action</i>	<i>Programmation (en ST)</i>
	<b>Action continue</b> Action := (Etat = 13) <b>OR</b> ... autres ét l'action ... ;
	<b>Action conditionnelle</b> Action := ((Etat = 13) <b>AND</b> Cond) <b>OR</b> ...
	<b>Action retardée</b> (* TON_0 est une instance du FB standa TON_0(IN := (Etat = 13), PT := T#5s) ; Action := TON_0.Q <b>OR</b> ... .. ;
	<b>Action limitée dans le temps</b> (* TP_0 est une instance du FB standar TP_0(IN := (Etat = 13), PT := T#5s) ; Action := TP_0.Q <b>OR</b> ... .. ;
	<b>Action conditionnelle liée au temps</b> (* TON_0 est une instance du FB standa

### 10.2.6.2 Actions mémorisées – Assignment sur événement

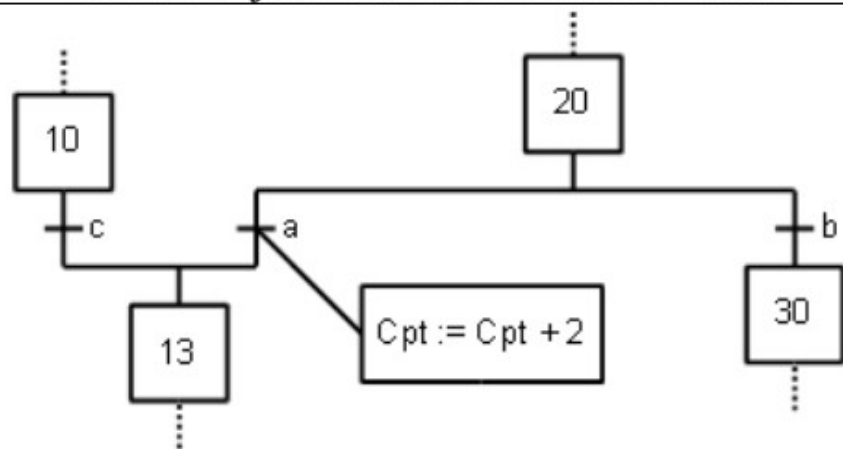
En mode mémorisé, c'est l'association d'une action à des événements (activation / désactivation d'étapes, franchissement d'une transition, ...) qui permet d'indiquer qu'une variable de sortie prend et garde la valeur imposée si l'un de ces événements se produit. On appelle affectation le fait de mémoriser, à un instant donné, la mise à une valeur déterminée d'une variable interne ou de sortie.

Le tableau ci-dessous présente la façon de programmer les différents types d'actions mémorisées définies dans la norme GRAFCET

<i>Action</i>	<i>Programmation (en ST)</i>
	<p><b>Action à l'activation de l'étape</b></p> <ul style="list-style-type: none"> <li>- <i>A l'intérieur du « DECIDER SUR »</i></li> </ul> <pre> 12 :     IF Condition d'évolution THEN         Etat := 13 ;         Act := val ;     END_IF 13 :     // [...] </pre> <ul style="list-style-type: none"> <li>- <i>Ou à l'extérieur du « DECIDER SUR »</i></li> </ul> <p>(* R_TRIG_0 est une instance de R_TRIG *)</p> <pre> R_TRIG_0(CLK := (Etat = 13)) ; IF R_TRIG_0.Q THEN     Act := val ; END_IF </pre>
	<p><b>Action à la désactivation de l'étape</b></p> <ul style="list-style-type: none"> <li>- <i>A l'intérieur du « DECIDER SUR »</i></li> </ul> <pre> 13 :     IF Condition d'évolution THEN         Etat := 14 ;         Act := val ;     END_IF 14 :     // [...] </pre>

Action	Programmation (en ST)
	<p><b>Action sur événement</b>  <b>Obligatoirement à l'extérieur du « DECIDE »</b></p> <pre>(* R_TRIG_0 est une instance d R_TRIG *) R_TRIG_0(CLK := Cond) ; <b>IF</b> (Etat = 13) <b>AND</b> R_TRIG_0.Q <b>TH</b>     Cpt := Cpt + 1 ; <b>END_IF</b></pre>
	<p><b>Action sur événement</b>  <b>Obligatoirement à l'extérieur du « DECIDE »</b></p> <pre>(* F_TRIG_0 est une instance d F_TRIG *) F_TRIG_0(CLK := Cond) ; <b>IF</b> (Etat = 13) <b>AND</b> F_TRIG_0.Q <b>TH</b>     Cpt := Cpt + 1 ; <b>END_IF</b></pre>

**Action au franchissement d'une transition**



Le franchissement de la transition de réceptivité « a » correspond à la l'étape 20 et à l'activation de l'étape. Nous avons donc la programmation

- **A l'intérieur du « DECIDER SUR »**

20 :

**IF** a **THEN**

Etat := 13 ;

Cpt := Cpt + 1 ;

**ELSIF** b **THEN**

Etat := 30 ;

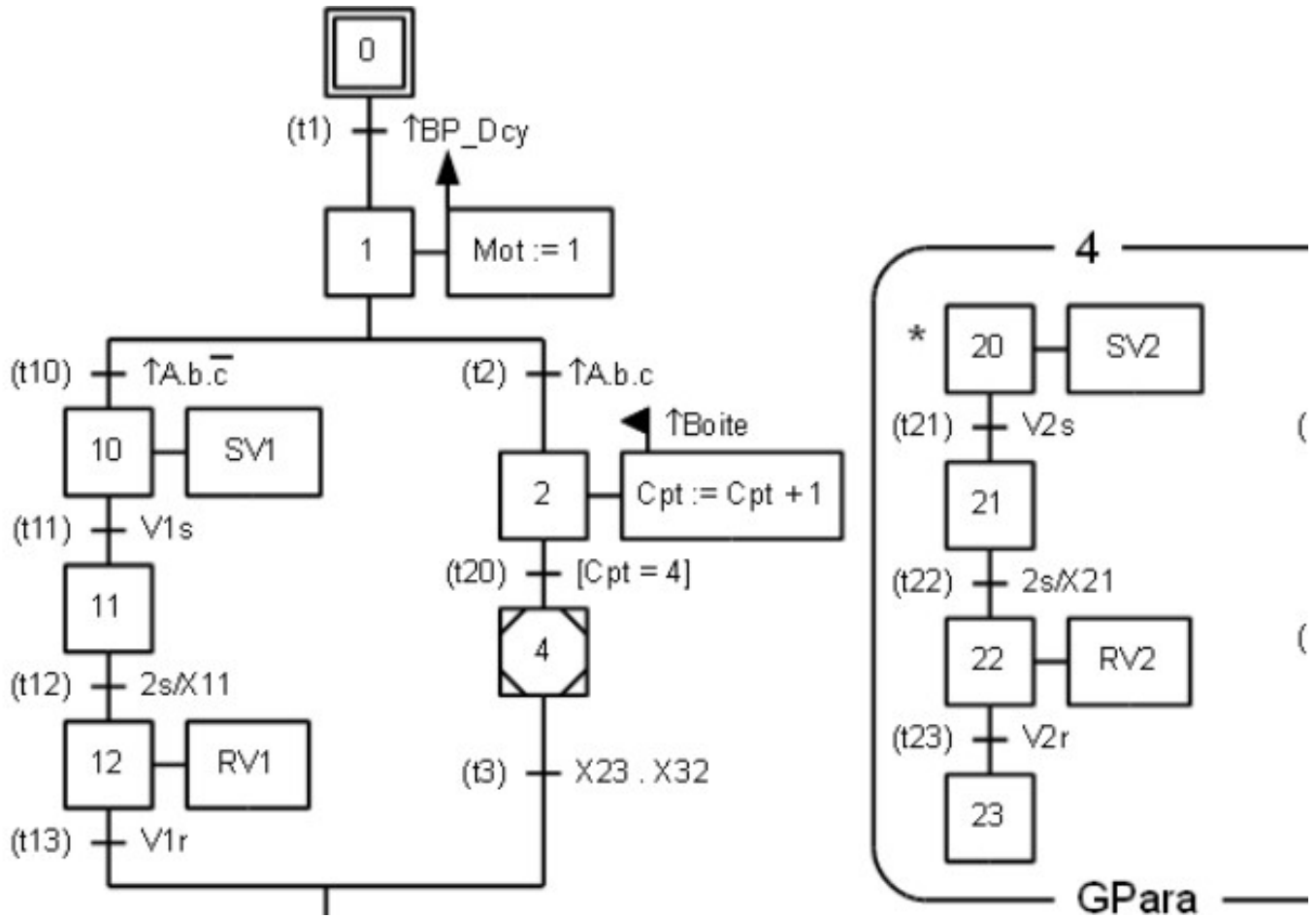
**END\_IF**



### 10.2.7 Exemple

Dans un premier temps, nous pouvons transformer la spécification en un ensemble de graphes pour lesquels nous n'aurons qu'une seule étape active à un instant « t ».

Spécification équivalente avec 3 graphes :



L'état du graphe principal sera représenté par la variable « Etat », la séquence {20, 21, 22, 23} sera représentée par la variable « Br1 » et la séquence {30, 31, 32} sera représentée par la variable « Br2 ». Les actions mémorisées seront programmées à l'intérieur des structures « DECIDER SUR »



Programmation de l'exemple d'application :

<pre> PROGRAM _INIT      // Initialisation de la situation initiale     Etat := 0;     Br1 := 0;     Br2 := 0;  END_PROGRAM  PROGRAM _CYCLIC      // Détection de tous les fronts     R_TRIG_BP_Dcy(CLK := BP_Dcy);     R_TRIG_A(CLK := A);     R_TRIG_Boite(CLK := Boite);     F_TRIG_B(CLK := B);     // Toutes les tempos     TON_X11(IN := (Etat = 11), PT := T#2s);         </pre>	<pre> // Gestion des graphes CASE Etat OF     0 :         IF R_TRIG_BP_Dcy.Q             Etat := 1 ;         // Action à l'activation de         Mot := TRUE ;         END_IF     // Sélection de séquences     1 :         IF R_TRIG_A.Q AND t             Etat := 2 ;         ELSIF R_TRIG_A.Q AN             Etat := 10 ;         END_IF     // Activation de séquences     2 :         IF (Cnt = 4) THEN         </pre>
<pre>         4 :         // Synchronisation de séquences         IF (Br1 = 23) AND (Br2 = 32) THEN             Etat := 3 ;             Mot := TRUE ;         END_IF         // Gestion de la séquence {20, 21,         22, 23}         CASE Br1 OF             20 :                 IF V2s THEN                     Br1 := 21 ;                 END_IF             21 :                 IF TON_X21.Q THEN                     Br1 := 22 ;                 END_IF             22 :                 IF V2r THEN                     Br1 := 23 ;                 END_IF         END_CASE         // Gestion de la séquence {30, 31,         32}         CASE Br2 OF             30 :                 IF V1s THEN                     Br2 := 31 ;                 END_IF             31 :                 IF V1r THEN                     Br2 := 32 ;                 END_IF         END_CASE     .         </pre>	<pre>         // Gestion de la séquence {         10 :             IF V1s THEN                 Etat := 11 ;             END_IF         11 :             IF TON_X11.Q THEN                 Etat := 12 ;             END_IF         12 :             IF V1r THEN                 Etat := 3 ;             END_IF         END_CASE          // Programmation des action         // Actions continues         SV1 := (Etat = 10) OR (f         RV1 := (Etat = 12) OR T(         SV2 := (Br1 = 20);         RV2 := (Br1 = 22);          // Compteur         IF (Etat = 2) AND R_TRI(             Cpt := Cpt + 1;         END_IF      END_PROGRAM         </pre>