



4^e session : Formation Allen-Bradley

www.locoplz.com

- **Grafset appliqué à l'automatisme industriel**

Road Map

	30/08/2024	06/09/2024	12/09/2024	20/09/2024	27/09/2024	04/10/2024
Séance d'accueil						
Introduction						
Ladder						
SFC						
Maintenance diagnostique						
Intégration Factory talk view						

Table des matières

1

Introduction

2

Bases du Grafcet

3

Règles de fonctionnement

4

Structures de Grafcet

5

Niveaux de description

6

Logiques de systèmes

7

Programmation

8

Fonctionnalités Rockwell

Introduction

Introduction au Grafcet / SFC

Définition :

Le **Grafcet** (Graphe Fonctionnel de Commande Étape/Transition), appelé **SFC (Sequential Function Chart)** en anglais, est un outil graphique qui décrit le **fonctionnement séquentiel** d'un système automatisé.

Points clés :

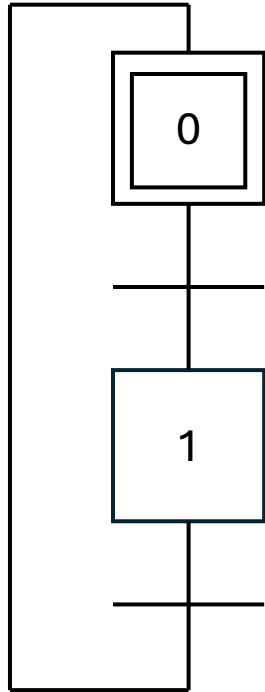
- Modélise un procédé étape après étape.
- S'applique à tout type de commande logique : électrique, pneumatique ou hydraulique.
- Utilisable aussi bien avec un système câblé (relais, contacteurs) qu'avec un système programmé (API).

En résumé :

Le Grafcet décrit **quoi faire** et **dans quel ordre**, sans préciser la technologie utilisée pour réaliser ces actions.

Bases du Grafcet

Éléments constitutifs du Grafcet



Étape initiale

- Symbole : double carré
- Active au démarrage
- Généralement sans action (repos)

Étape initiale

- Symbole : carré numéroté (numéro unique)
- Associées à une ou plusieurs actions (rectangle relié)
- Quand active, l'action s'exécute

Transitions

- Symbole : trait vertical coupé par un trait horizontal
- Associées à une réceptivité (condition logique)
- Si condition vraie → passage à l'étape suivante

Liaisons orientées

- Relient les étapes et définissent le sens de progression
- Une liaison de retour permet de boucler le cycle

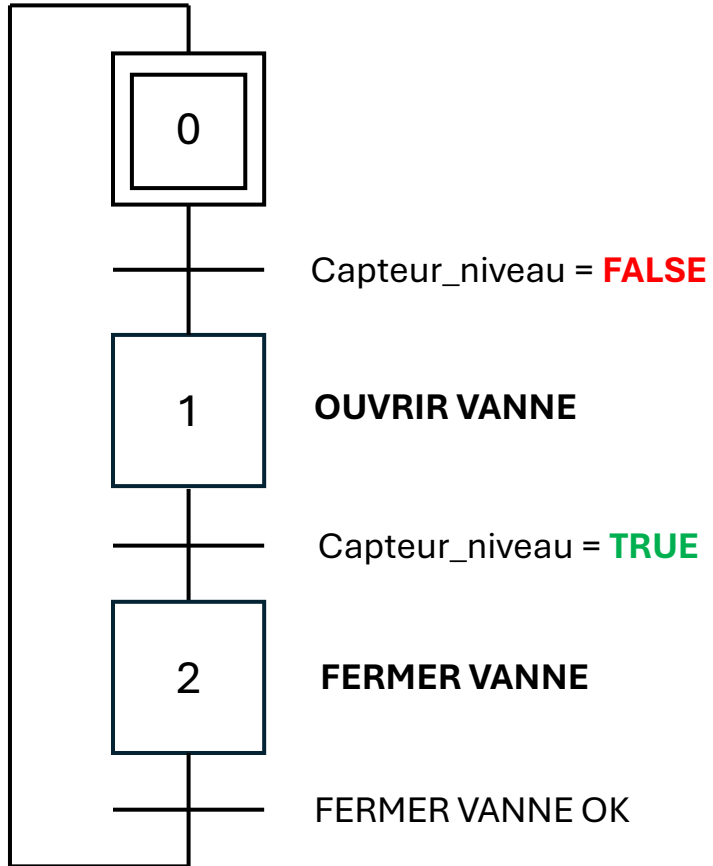
Fonctionnement

- Démarrage : seule l'étape initiale est active
- Une transition est franchie si :
 1. L'étape précédente est active
 2. La condition est vraie
- L'étape suivante s'active, l'ancienne se désactive
- Retour final à l'étape initiale pour un nouveau cycle

À retenir

- Langue commune entre ingénieurs et techniciens
- Étape = état + action associée
- Transition = condition de passage
- Une séquence est exclusive (1 étape active à la fois)
- Retour initial = répétition du cycle

Bonnes pratiques et avantages du Grafcet



Nommer les actions

- Toujours utiliser des **verbes d'action en majuscules** (OUVRIR, FERMER, ATTENDRE).
- Permet une **lecture immédiate** par un technicien, même en maintenance.

Avantage en conception

- Le Grafcet fait le lien entre le **cahier des charges** et le **code automate**.
- Gain de temps : inutile de redéfinir la logique à chaque étape du projet.

Synthèse du Grafcet

Définition

- **Grafcet = SFC** → langage graphique pour décrire un système séquentiel.
- Utilisé en **automatisme industriel** : électrique, pneumatique, hydraulique, câblé ou programmé.

Éléments clés

- Étape initiale (double carré, active au départ)
- Étapes (carrés numérotés avec actions)
- Transitions (traits avec conditions logiques)
- Liaisons orientées (enchaînement + retour au départ)

Cycle

- Initialisation → franchissement des transitions → activation/désactivation → retour à l'étape initiale

Valeur ajoutée

- Lisible par tous
- Normé
- Structurant pour la conception et la maintenance

Règles de fonctionnement

Règles d'évolution du Grafcet

Règle n°1 – Initialisation

- Au démarrage, une seule étape est définie comme **étape initiale**.
- Elle est **active dès la mise sous tension**.
- Généralement un **état de repos**.

Règle n°2 – Franchissement d'une transition

- Conditions nécessaires :
 - L'étape précédente est active.
 - La réceptivité associée est vraie.

Règle n°3 – Évolution des étapes actives

- Quand une transition est franchie :
 - L'étape suivante devient active.
 - L'étape précédente est désactivée.
- Assure une progression séquentielle claire.

Importance de ces règles

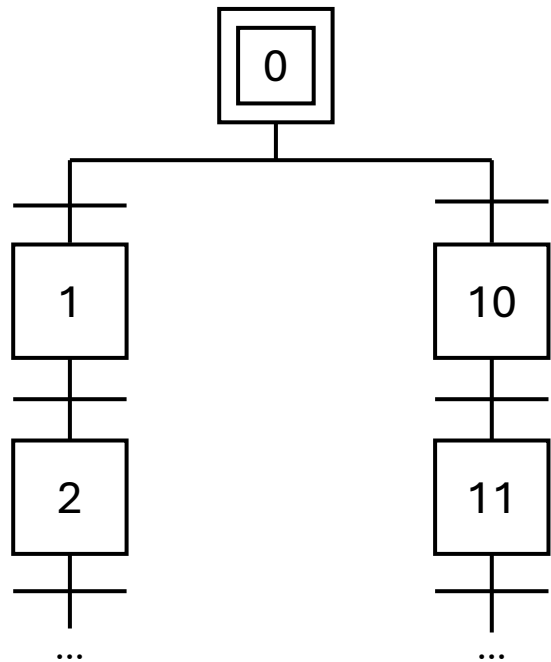
- Séquence déterministe → pas de confusion.
- Robustesse du cycle → pas d'activation incohérente.
- Fondement de la programmation structurée Grafcet/SFC.
- Déjà incluses dans le langage → pas besoin de les recoder.

Structures de Grafcet

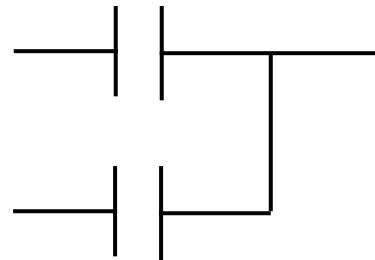
Divergences dans un Grafcet

Divergence OU (choix exclusif)

- Un seul chemin est activé parmi plusieurs possibles.
- Comparable à une **bifurcation de route** : on prend à gauche **ou** à droite.
- Exemple : une machine qui peut fonctionner soit en **mode automatique**, soit en **mode manuel**.

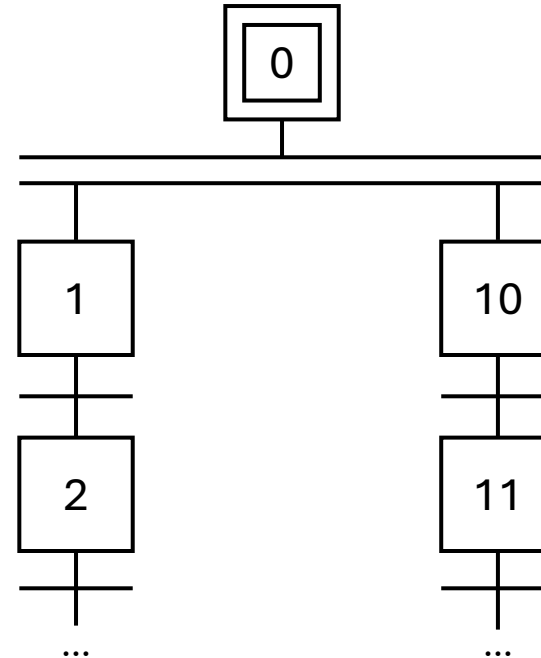


En Ladder :

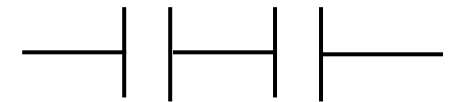


Divergence ET (chemins parallèles)

- Plusieurs étapes sont activées en même temps.
- Comparable à **faire deux actions en parallèle** : par ex. **faire tourner la machine et ventiler en même temps**.
- Chaque branche évolue selon ses propres transitions, puis peut converger plus tard.



En Ladder :



Structures de Grafcet

Grafcet linéaire

- Succession simple d'étapes et transitions.
- Une seule trajectoire possible → adapté aux systèmes simples (ex. convoyeur monotâche).

Grafcet avec reprise de séquence

- Répétition d'un même ensemble d'actions sans dupliquer le Grafcet.
- Exemple : palan qui déplace plusieurs pièces successivement.

Grafcet avec sélection de séquence (OU)

- Permet un choix exclusif entre plusieurs branches.
- Une seule branche activée en fonction des conditions.
- Exemple : machine qui perce uniquement si nécessaire.

Grafcet avec séquences simultanées (ET)

- Plusieurs branches actives en parallèle.
- Exemple : palan qui monte et se déplace en même temps.

Structure	Caractéristique	Exemple typique	Avantage	Limite
Linéaire	Séquence unique	Convoyeur simple	Simplicité	Pas de choix, pas de simultanéité
OU (sélection)	Choix exclusif entre branches	Percètris (percer ou non)	Flexibilité, adaptabilité	Risque d'ambiguïté si conditions mal définies
Reprise	Répétition d'actions	Palan manipulant plusieurs pièces	Évite duplication, lisibilité	Risque de boucle infinie
ET (simultanéité)	Branches parallèles indépendantes	Palan retour rapide	Optimisation temps de cycle	Risques de conflit, besoin de synchronisation

Niveaux de description

Niveaux de description du Grafcet – Théorie (1/3)

Niveau 1 – Partie opérative (fonctionnelle)

- Décrit uniquement **les fonctions à réaliser**, sans préciser la technologie utilisée.
- Utilise un langage simple, sous forme de phrases du type « *Autoriser* », « *Déplacer* », « *Attendre* ».
- C'est une vision **abstraite et indépendante** de la solution technique.

Niveau 2 – Partie commande (technologique)

- Décrit le système à travers ses **composants physiques réels** : moteurs, capteurs, actionneurs, lecteurs, etc.
- On établit le lien entre la **logique fonctionnelle** et les **éléments matériels concrets**.
- C'est la traduction des fonctions en termes d'équipements.

Niveau 3 – Partie automate (programmation API)

- Décrit le système en utilisant les **adresses d'entrées/sorties automate**.
- Chaque fonction ou composant est associé à une adresse (ex. %I pour les entrées, %Q pour les sorties, %M pour les mémoires internes).
- Sert de **plan direct pour la programmation** dans l'automate.

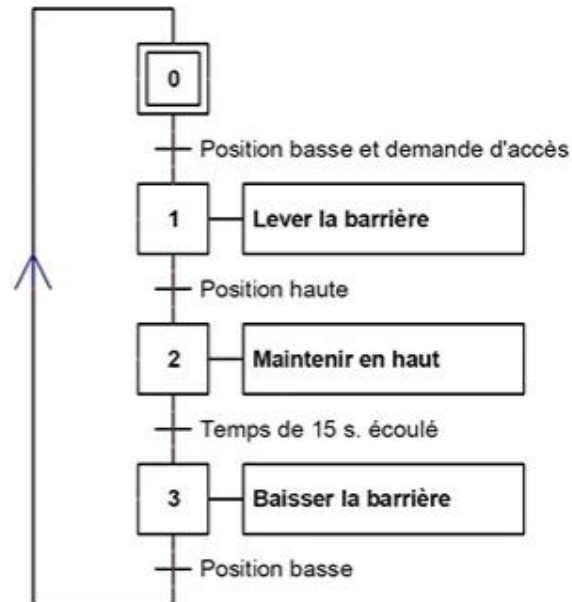
Niveau 1 → Quoi faire ? (fonctionnel, abstrait)

Niveau 2 → Avec quoi le faire ? (technologique, composants)

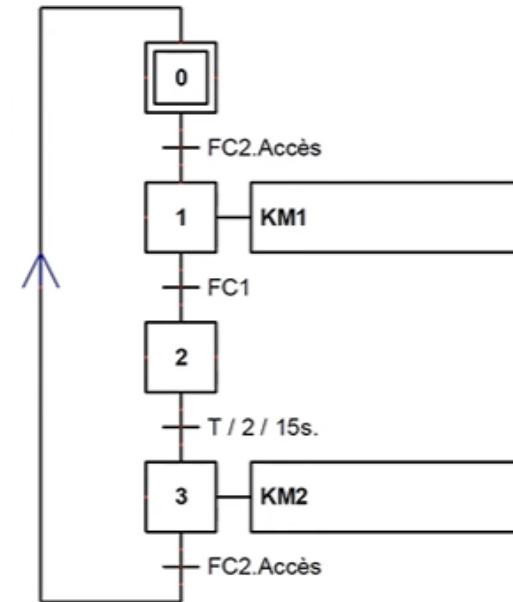
Niveau 3 → Comment le coder ? (adresses automate, API)

Niveaux de description du Grafcet – Théorie (2/3)

Niveau 1 – Partie opérative

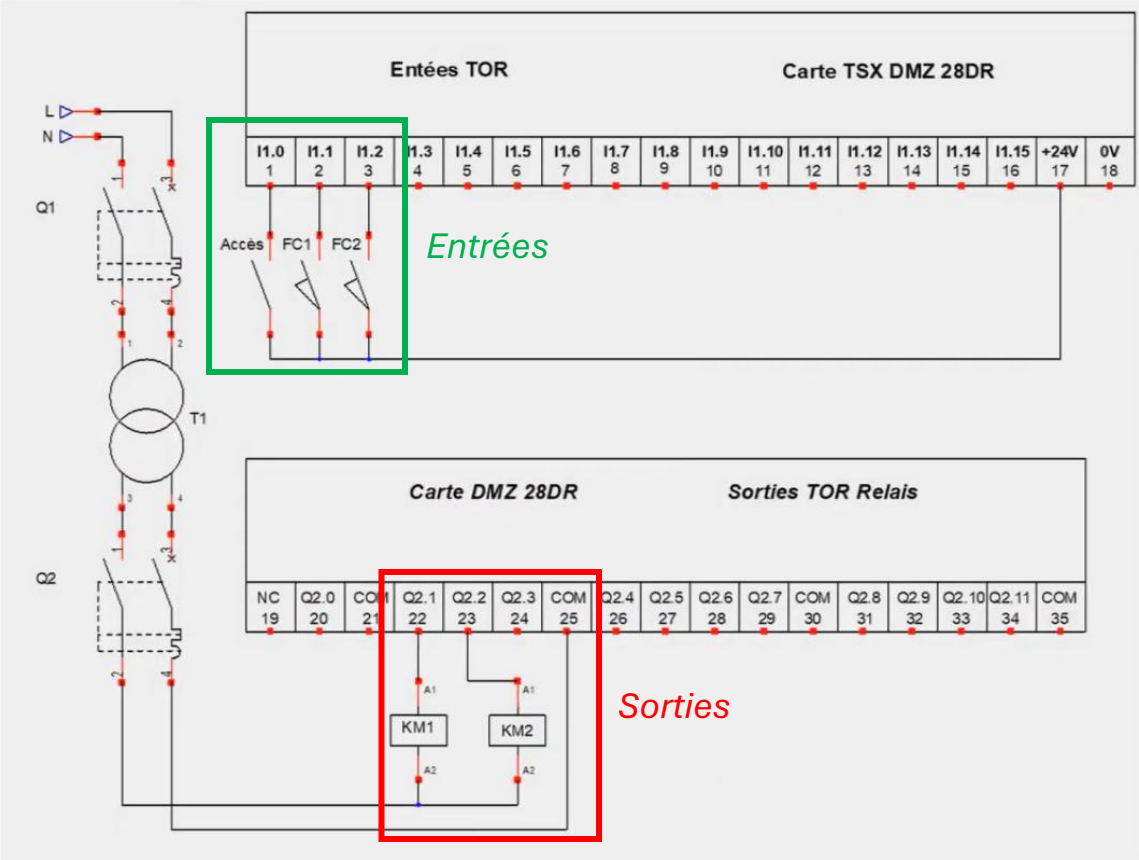


Niveau 2 – Partie commande

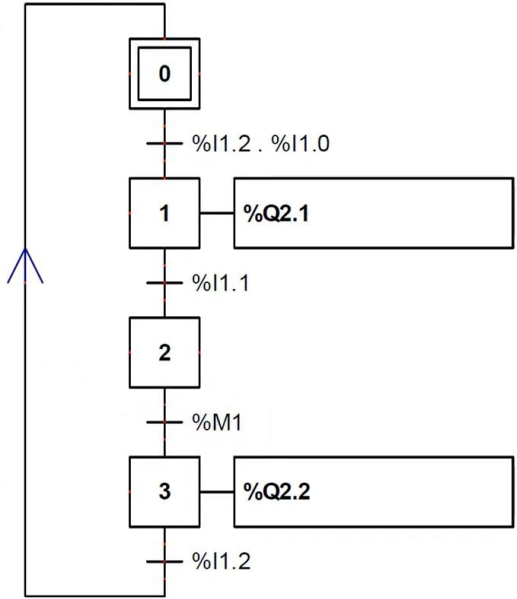


- Un moteur commandé par deux contacteurs : KM1 pour la montée et KM2 pour la descente,
- Un lecteur de carte magnétique (Accès),
- Un détecteur de position haute (FC1),
- Un détecteur de position basse (FC2).

Niveaux de description du Grafcet – Théorie (3/3)



Niveau 3 – Partie automate



Élément du système	Raccordement API
Accès	%I1.0
FC1	%I1.1
FC2	%I1.2
KM1	%Q2.1
KM2	%Q2.2

Logiques de systèmes

Systèmes combinatoires vs séquentiels

Système combinatoire

- La sortie dépend uniquement **des entrées instantanées**.
- Pas de mémoire → pas d'historique.
- Représentable par une **table de vérité complète**.

Exemple concret :

- **Bouton-poussoir → lampe** : si bouton appuyé, lampe ON.
- **Ventilateur** : ON si température > 30 °C.

Système séquentiel

- La sortie dépend **des entrées actuelles ET de l'état précédent**.
- Mémoire indispensable (basculer, étape active).
- Impossible à représenter par une table de vérité seule.

Exemple concret :

- **Barrière de parking** : carte validée + barrière fermée avant → alors ouvrir.
- **Cuve** : remplir → attendre niveau haut → fermer la vanne.

Lien avec le Grafcet

- Le Grafcet est un **système séquentiel**.
- Chaque étape = une **mémoire d'état**.
- Chaque transition dépend à la fois de la **réceptivité (entrée)** et de l'**étape active**.

- **Combinatoire** = réponse immédiate (appuie → allume).
- **Séquentiel** = réponse dépend aussi du passé (séquence d'étapes).

Programmation

Programmation d'un système séquentiel

Méthode empirique (Technique n°1)

On code directement en blocs (contacts, mémoires, temporisations).

Inconvénients :

- Tests multipliés à l'infini.
- Chaque correction peut introduire un nouveau bug.
- Impossible de vérifier toutes les situations.
- Résultat → perte de temps et manque de fiabilité.

Exemple : programmer une machine directement en Ladder sans Grafcet → obligé de tester des centaines de cas imprévus.

Méthode structurée avec Grafcet (Technique n°2)

Principe : utiliser le Grafcet comme **plan directeur**.

Étapes :

- Dessiner le Grafcet à partir du cahier des charges.
- Traduire chaque étape et chaque transition dans le langage automate (SFC, Ladder, ST, etc.).
- Tester la séquence → validation rapide.

Avantages :

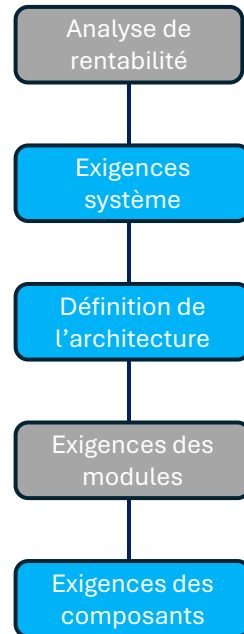
- Programme **structuré et lisible**.
- Réduction drastique du temps de mise au point.
- Sécurité → aucune étape oubliée, transitions claires.

Exemple simple : cuisson de pâtes

- Étape 1 : PRÉPARER
- Étape 2 : CUIRE
- Étape 3 : ÉGOUTTER

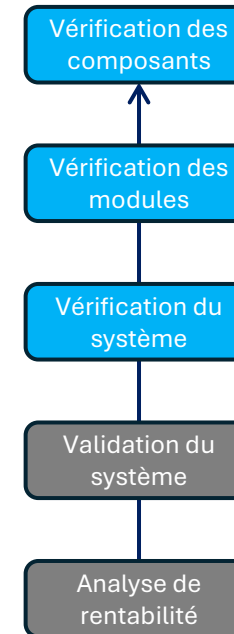
Lien entre Grafcet et le cycle en V

Conception



Conception des composants

Validation



Phase de conception

- **Exigences système → Grafcet opératif** : on décrit les fonctions attendues en langage simple, lisible par le client.
- **Définition de l'architecture / modules → Grafcet commande** : on intègre moteurs, capteurs, actionneurs.
- **Exigences composants → Grafcet automate** : on traduit en entrées/sorties API (%I, %Q) et en programme (SFC, ST, LD).

Phase de validation

- **Vérification des composants** : tests I/O (capteurs, moteurs).
- **Vérification des modules** : validation des séquences locales (Grafcet module par module).
- **Vérification/validation système** : cohérence globale, conformité avec le Grafcet opératif initial.

Fonctionnalités Rockwell

Qualifiers d'action dans un SFC (1/2)

Rappel :

- Dans un Grafcet/SFC, une étape contient une ou plusieurs **actions** (ex. démarrer un moteur, allumer une lampe).
- Mais il ne suffit pas de définir **quoi faire** → il faut aussi préciser **quand** et **combien de temps** l'action doit durer.
- C'est le rôle des **qualifiers**.

Un qualifier définit la durée de vie de l'action :

- Est-ce que l'action s'arrête dès que l'étape s'arrête ?
- Est-ce qu'elle continue même après la fin de l'étape ?
- Est-ce qu'elle démarre avec un délai ?
- Est-ce qu'elle envoie seulement une impulsion unique ?

Qualifiers d'action dans un SFC (2/2)

Qualifier	Nom complet	Fonctionnement	Exemple industriel
N	Non-Stored	Action active uniquement tant que l'étape est active.	Lampe de signalisation allumée uniquement pendant l'étape « Cycle en cours ».
S	Stored	Action reste active même après désactivation de l'étape, nécessite un Reset.	Convoyeur qui continue à tourner après la phase de chargement, jusqu'à reset.
L	Time Limited	Action démarre avec l'étape mais s'arrête automatiquement après un temps défini.	Pompe de lubrification qui s'arrête automatiquement après 15 s, même si l'étape continue.
SL	Stored and Time Limited	Action démarre avec l'étape, dure un temps défini, puis reste active tant qu'on ne fait pas de reset.	Ventilateur de refroidissement qui tourne 30 s puis reste en marche jusqu'à reset manuel.
D	Time Delayed	Action démarre après un délai si l'étape est toujours active.	Chauffage d'un four qui démarre 10 s après l'ouverture de la porte si celle-ci reste ouverte.
DS	Delayed and Stored	Action démarre après un délai et reste active même si l'étape s'arrête. Nécessite un reset.	Convoyeur tampon qui démarre 5 s après la demande et continue jusqu'à reset.
SD	Stored and Time Delayed	Action démarre après un délai et reste active même si l'étape est désactivée. Nécessite un reset.	Extracteur de poussières qui démarre 5 s après activation et continue jusqu'à reset, même si l'étape est terminée.
P	Pulse	Action exécutée une seule fois lors de l'activation de l'étape.	Électrovanne qui envoie une impulsion d'air pour éjecter une pièce.
P1	Pulse (Rising Edge)	Action exécutée une seule fois au front montant de l'étape.	Sirène qui donne un bip sonore au lancement d'un cycle.
PO	Pulse (Falling Edge)	Action exécutée une seule fois au front descendant (désactivation de l'étape).	Éclairage d'atelier qui s'éteint automatiquement avec un bip sonore en fin de cycle.
R	Reset	Sert à désactiver une action de type S, DS ou SD.	Étape « Arrêt » envoie un reset pour couper convoyeur et moteurs stockés.

Boolean Action

Fonctionnement :

- Quand l'étape devient active → le bit passe à 1.
- Quand l'étape s'arrête → le bit passe à 0.
- C'est au **programme Ladder ou ST externe** de lire ce bit et d'exécuter la logique associée (commande moteur, vanne, alarme...).

Définition :

- Une **Boolean Action** dans un SFC est une action simplifiée.
- Elle ne contient **aucune logique interne** :
 - Elle se contente de mettre **ON ou OFF** un bit lié à l'étape.
 - Ce bit est stocké dans une structure de tag (SFC_ACTION).

Points importants :

- Pas de lien direct entre le SFC et la logique → le **reset automatique n'affecte pas** les Boolean Actions.
- Les arrêts et réinitialisations doivent être gérés **manuellement** dans le Ladder/ST.

Une **Boolean Action** = simple interrupteur ON/OFF piloté par l'étape.
La vraie logique de commande se fait dans le Ladder ou le ST.

Action vs Boolean Action – Synthèse

Aspect	Action classique	Boolean Action
Contenu	Contient de la logique (ST, Ladder, appel de routine) → l'action « pense » toute seule.	Ne contient aucune logique → juste un bit ON/OFF.
Exécution	Le code est exécuté directement dans l'action.	Le code s'exécute ailleurs (dans le Ladder/ST) en surveillant le bit.
Reset automatique	Oui → géré par les qualifieurs (N, S, D, ...).	Non → doit être géré manuellement dans le Ladder/ST.
Complexité	Plus lourd, mais puissant et autonome.	Plus léger, utile pour des signaux simples.
Usage typique	Contrôler directement un moteur, une vanne, ou une séquence.	Activer un flag binaire pour informer une autre partie du programme.

Étape “STOP” dans un Grafcet



Étape “STOP” dans un Grafcet

1. Définition

- C’est une **étape terminale** : le cycle s’arrête et le Grafcet n’exécute plus d’actions tant qu’il n’est pas réinitialisé.
- Opposée à l’étape initiale (qui démarre automatiquement).
- Représentation : comme une étape classique, parfois marquée par un symbole ou un nom explicite (*Stop_000*).

2. Intérêt de l’utiliser

- **Fin d’un cycle unique** → la machine s’arrête après un cycle complet.
- **Mode maintenance/sécurité** → arrête les moteurs, coupe les actionneurs.
- **Éviter les boucles involontaires** → empêche un redémarrage immédiat.
- **Point de reprise contrôlé** → utile en fabrication batch ou en labo, permet plusieurs points de redémarrage définis.

3. Gestion dans le programme automate

- À l’arrivée sur STOP → plus de transitions actives.
- Pour relancer :
 - Reset logiciel (réactiver étape initiale via programme).
 - Reset matériel (commande HMI, bouton de réinitialisation).
- Norme IEC 60848 : l’étape STOP est optionnelle mais autorisée.

4. Exemple industriel

- Poste d’assemblage :
 - Prise de pièce
 - Vissage automatique
 - Contrôle qualité
 - STOP → fin du cycle
- Redémarrage → opérateur appuie sur **Cycle Start**.
- Avantage : évite que la machine boucle automatiquement sans action humaine.

Conclusion

- **Définition** : Outil graphique normalisé (IEC 60848) décrivant le fonctionnement séquentiel d'un système.
- **Éléments clés** : Etapes, transitions (conditions), liaisons logiques.
- **Règles** : Initialisation → franchissement conditionnel → activation/désactivation.
- **Structures** : Linéaire, OU (choix), reprise, ET (simultanéité).
- **Niveaux** : Opératif (quoi faire), commande (avec quoi), automate (comment coder).
- **Combinatoire vs séquentiel** : Immédiat vs dépendant du passé (mémoire).
- **Programmation** : Méthode structurée avec Grafcet → plus clair, plus fiable.