



2<sup>e</sup> session : Formation Allen-Bradley

[www.locoplcl.com](http://www.locoplcl.com)

- **Ladder Logic** : bases, fonctions avancées et bonnes pratiques

**LOCOPLC**



# Synthèse de la séance précédente

## Automates Rockwell

- **Micro800** → applications simples, peu d'E/S.
- **CompactLogix** → taille moyenne, motion multiaxes, Ethernet/IP.
- **ControlLogix (PAC)** → haut de gamme, modulaire, multi-protocoles, processus complexes.

## PLC vs PAC

- **PLC** : logique séquentielle simple, applications locales
- **PAC** : architecture orientée objet, puissance de calcul élevée, intégration IT/OT → adapté Industrie 4.0

## Écosystème Rockwell

- **Contrôleurs** (GuardLogix, CompactGuard)
- **I/O** (standard & sécurité, CIP Safety)
- **Interfaces HMI** (PanelView Plus)
- **Variateurs** (PowerFlex, Kinetix – motion control)
- **Communication** : EtherNet/IP, ControlNet, DeviceNet

## Outils logiciels (Studio 5000)

- **Routines** : blocs de logique (Ladder, ST, FBD)
- **UDT (User Data Type)** : structuration des données
- **AOI (Add-On Instruction)** : blocs logiques réutilisables
- **Fault Handler** : gestion centralisée des erreurs critiques
- **Power-Up Handler** : réinitialisation au démarrage
- **Tasks** : Continuous, Periodic, Event (ordonnancement)
- **Simulation** : Logix Emulate + RSLink pour tester sans matériel



# Road Map

|                                  | 30/08/2024 | 06/09/2024 | 13/09/2024 | 20/09/2024 | 27/09/2024 | 04/10/2024 |
|----------------------------------|------------|------------|------------|------------|------------|------------|
| 1. Séance d'accueil              |            |            |            |            |            |            |
| 2. Introduction                  |            |            |            |            |            |            |
| 3. Ladder                        |            |            |            |            |            |            |
| 4. SFC / ST                      |            |            |            |            |            |            |
| 5. Maintenance diagnostique      |            |            |            |            |            |            |
| 6. Intégration factory talk view |            |            |            |            |            |            |

# Table des matières

**1**

Introduction & historique

**2**

Définition & usages

**3**

Popularité du Ladder

**4**

Composants de base

**5**

Cycle d'exécution

**6**

Fonctions logiques de base

**7**

Fonctions avancées

**8**

Exercice

# Introduction & historique



# De la logique câblée au Ladder



Avant 1968

1968

1970-1980

Aujourd'hui

- Automatisation avec **armoires de relais électromécaniques**.
- Schémas ressemblant à une **échelle** (rails verticaux + barreaux horizontaux).
- Complexité → difficile et coûteux à modifier.

- General Motors cherche une alternative.
- Richard Morley développe le **premier PLC (Modicon 084)**.
- Inspiration des schémas électriques → adoption facile par les électriciens.

- Intégration par **Modicon, Siemens, Allen-Bradley, Omron**.
- Langage simple, visuel, familier.
- **1993 : normalisation IEC 61131-3**
- Reconnaissance officielle de 5 langages standards :
- Ladder (LD), FBD, ST, IL (abandonné), SFC.

- Toujours dominant dans l'industrie.
- Intégré dans les grands logiciels : **TIA Portal, Studio 5000, Codesys**.
- Évolutions : temporisations, compteurs, communications (Ethernet, Profibus...).

# Définition & usages



# Qu'est-ce que le Ladder ?

Langage de programmation **graphique** pour automates programmables (API).

Inspiré des **schémas de câblage à relais**.

Forme d'**échelle** :

- 2 rails verticaux (alimentation)
- Barreaux horizontaux (*rungs*)

Adapté à tout processus **industriel combinatoire ou séquentiel**.



# Popularité du Ladder



# Les atouts du Ladder

1

## Langage graphique proche des schémas électriques

- Ressemble à un **circuit à relais**.
- Symboles (contacts, bobines) → familiers pour les électriciens.
- On « programme en dessinant » → pas besoin de coder.

2

## Visualisation et débogage aisés

- Lecture **de gauche à droite** → flux logique clair.
- Surbrillance des contacts activés en temps réel.
- Dépannage rapide → suivi visuel du courant logique.

3

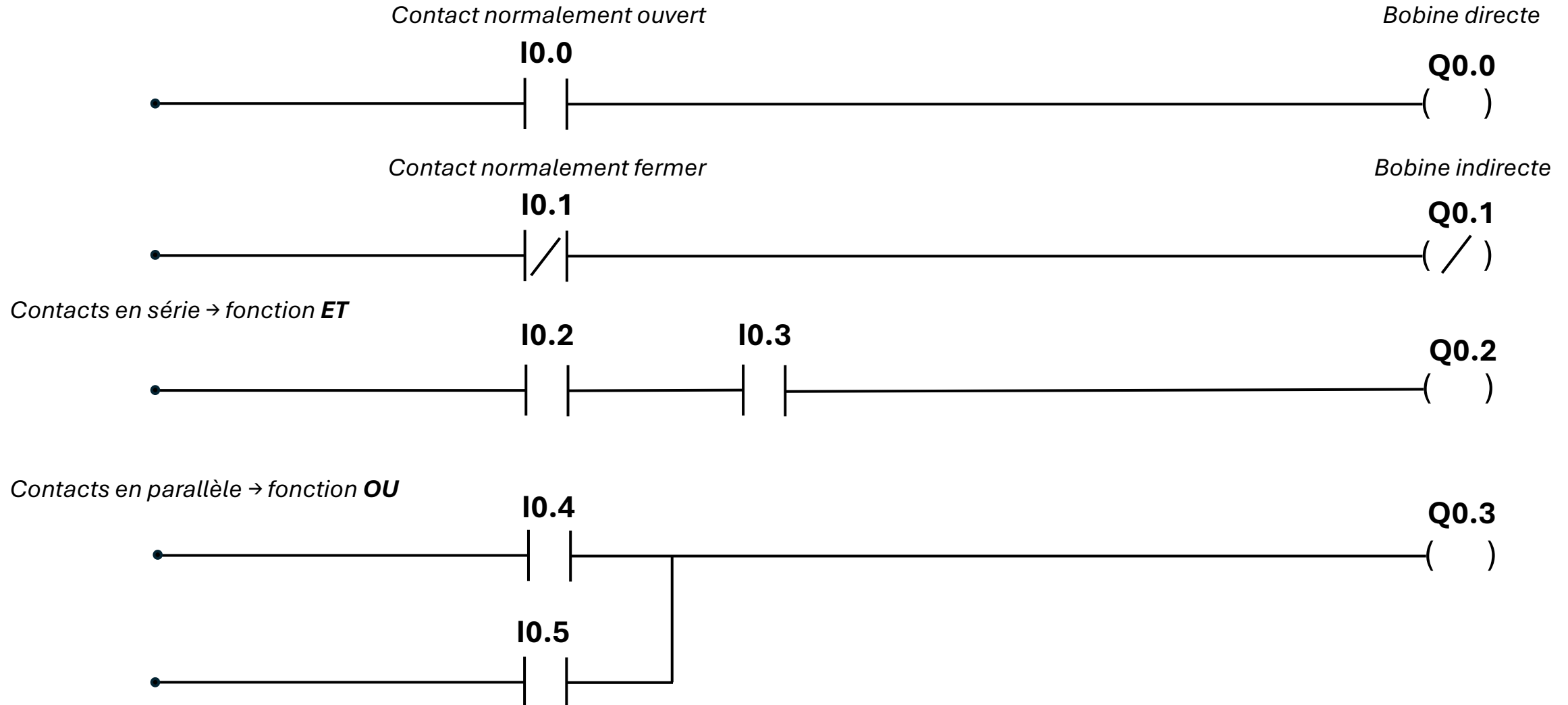
## Simplicité des concepts de base

- Basé sur les **états binaires (0/1, Vrai/Faux)**.
- Repose sur les opérations **ET, OU, NON**.
- Apprentissage rapide → idéal pour débutants.

# Composants de base



# Construire une logique en Ladder



# Cycle d'exécution



# Évaluation logique et ordre d'exécution

## Cycle de fonctionnement

L'automate lit et exécute le programme en boucle (scan PLC).

## Évaluation logique

Chaque contact agit comme un interrupteur :

- Fermé (1) → laisse passer le courant logique.
- Ouvert (0) → bloque le courant.

Si le courant atteint la bobine → sortie activée.

## Ordre d'exécution :

- Rung 1 (de gauche à droite)
- Rung 2
- ... jusqu'au dernier
- Retour au début

## Branches parallèles

- Contacts en parallèle = chemins alternatifs.
- Si au moins une branche est valide → sortie activée.
- Si toutes sont ouvertes → sortie inactive.



# Lecture – Exécution – Sorties

## Cycle en 3 étapes

1

### Lecture des entrées

- L'automate lit l'état des capteurs, boutons, fins de course...
- Stockage en mémoire interne.

2

### Mise à jour des sorties

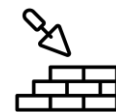
- Application des résultats aux actionneurs : moteurs, électrovannes, voyants...

3

### Mise à jour des sorties

- Application des résultats aux actionneurs : moteurs, électrovannes, voyants...

# Fonctions logiques de base



# Les fondamentaux de la logique booléenne

## 1. Fonction ET ( $Y = a \cdot b$ )

- Vraie si **toutes** les entrées sont vraies.
  - Contacts en **série**.
- Exemple : A ET B doivent être actifs pour activer Y.

|   |   |   |
|---|---|---|
| a | b | s |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 2. Fonction OU ( $Y = a + b$ )

- Vraie si **au moins une** entrée est vraie.
  - Contacts en **parallèle**.
- Exemple : A OU B suffit pour activer Y.

|   |   |   |
|---|---|---|
| a | b | s |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## 3. Fonction NON ( $Y = \overline{a}$ )

- Inverse l'état logique.
  - Contact **NC (normalement fermé)**.
- Exemple : Si A = 0 → Y activée ; Si A = 1 → Y désactivée.

|   |   |
|---|---|
| a | s |
| 0 | 1 |
| 1 | 0 |

## 4. Fonction NAND ( $Y = \overline{a \cdot b}$ )

- Sortie vraie sauf si toutes les entrées sont vraies.

|   |   |   |
|---|---|---|
| a | b | s |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## 5. Fonction NOR ( $Y = \overline{a + b}$ )

- Sortie vraie uniquement si toutes les entrées sont fausses.

|   |   |   |
|---|---|---|
| a | b | s |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## 6. Fonction XOR ( $Y = a \oplus b$ )

- Sortie vraie si une seule entrée est vraie, mais pas les deux.

|   |   |   |
|---|---|---|
| a | b | s |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

# Fonctions avancées



# Fonctions avancées du Ladder

## Temporisateurs (Timers)

Introduisent la notion de **temps** dans la logique.

- **TON (On Delay)** → active la sortie après un délai.
- **TOF (Off Delay)** → maintient la sortie active après coupure.
- **TP (Pulse)** → génère une impulsion de durée fixe.

## Compteurs (Counters)

Permettent de **compter des événements**.

- **CTU (Count Up)** → incrémente.
- **CTD (Count Down)** → décrémente.
- **CTUD (Up/Down)** → bidirectionnel.

## Compareurs & opérations mathématiques

Comparer des valeurs ( $>$ ,  $<$ ,  $=$ )

Réaliser des calculs (addition, soustraction, etc.).

## Mémoires & bascules

**SET / RESET** → mémorisation d'états.

**Détection de fronts** (montant/descendant).

Permettent la création de **séquences logiques**.



# Mémoires et séquences en Ladder

## Mémorisation d'états

**Bobine SET (S)** → verrouille une sortie à 1 jusqu'à réinitialisation.

**Bobine RESET (R)** → force une sortie à 0.

- Permet de garder un moteur ou un voyant actif **même après relâchement du bouton**.

## Détection de fronts

- **Front montant** → détection du passage de 0 à 1.
- **Front descendant** → détection du passage de 1 à 0.
- Utilisés pour déclencher une action **une seule fois** à un instant précis.

## Séquences

Construction de cycles pas à pas (ex. démarrage, fonctionnement, arrêt).

Exemple :

- Appui bouton START → moteur ON.
- Capteur de position → lancer le convoyeur.
- Capteur fin de course → arrêt cycle.

## Comparateurs

Permettent de comparer des valeurs numériques :

- > Supérieur
- < Inférieur
- = Egalité

Exemple : activer une alarme si la **température > seuil**.

## Opérations mathématiques

Addition, soustraction, multiplication, division.

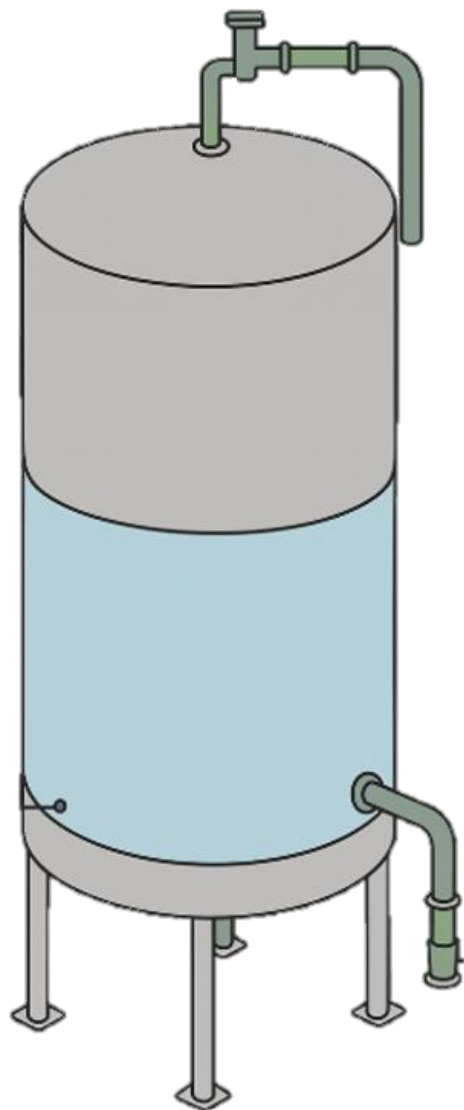
Utilisées pour :

- Calculer une consigne.
- Réaliser un dosage (ex. mélange de liquides).
- Surveiller une variable analogique (pression, vitesse, niveau...).

# Exercise



# Gestion d'une cuve avec électrovannes



## Contraintes sur les seuils

- Le **seuil de fin de remplissage** doit être **> seuil de remplissage**
- Le **seuil de remplissage** ne doit pas être **< 10 %**

## Fonctionnement opérateur

- L'opérateur peut **forcer le remplissage** en appuyant sur le BP, même si le seuil de fin de remplissage est atteint/dépassé
- Si le **niveau très haut (LSHH)** est atteint → **EV1 se coupe immédiatement**

## Gestion des électrovannes (EV1 et EV2)

- Chaque vanne possède des capteurs **ouvert/fermé**
- Si une vanne est **pilotée** : capteur *ouvert* doit s'activer en **< 25 s** → sinon **défaut ouverture**
- Si une vanne n'est **pas pilotée** : capteur *fermé* doit s'activer en **< 25 s** → sinon **défaut fermeture**
- Un défaut sur une vanne **interdit son pilotage**
- **Voyant défaut** allumé si défaut présent
- **BP Reset** (au relâchement) efface les défauts

## Simulation / logique de niveau

- La simulation doit gérer la **valeur du niveau cuve** (visible sur émulateur)
- Si possible, coder la simulation dans une **task séparée** pour plus de clarté
- **EV1 ouverte** → niveau augmente (remplissage rapide)
- **EV2 ouverte** → niveau baisse (vidange plus lente)

| Entrées                                    | Sorties                    |
|--------------------------------------------|----------------------------|
| <b>DI0</b> : BP Forçage remplissage        | <b>DO0</b> : Pilotage EV1  |
| <b>DI1</b> : BP Evacuation                 | <b>DO1</b> : Pilotage EV2  |
| <b>DI2</b> : BP Arrêt évacuation           | <b>DO2</b> : Voyant défaut |
| <b>DI3</b> : BP Reset défaut               |                            |
| <b>DI8</b> : LSLL = 1 quand niveau atteint |                            |
| <b>DI9</b> : LSHH = 0 quand niveau atteint |                            |
| <b>DI16</b> : EV1 Ouverture                |                            |
| <b>DI17</b> : EV1 Fermeture                |                            |
| <b>DI19</b> : EV2 Ouverture                |                            |
| <b>DI20</b> : EV2 Fermeture                |                            |