



1^e session : Formation Allen-Bradley

www.locoplz.com

LOCOPLC

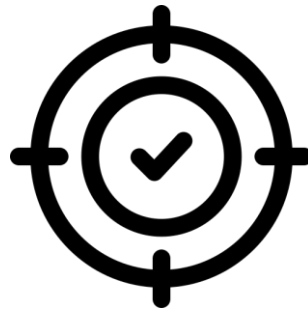
Les valeurs qui guident notre réussite

Discipline



La clé pour maintenir
la constance et
progresser chaque
jour

Rigueur



La précision et
l'exigence qui
garantissent la
qualité

Travail



L'effort quotidien qui
transforme les
objectifs en résultats

Table des matières

1

Introduction

2

Panorama des automates
Rockwell

3

Le PAC et ses spécificités

4

Environnement Rockwell

5

Simulation et mise en
œuvre

Introduction



Histoire & Positionnement d'Allen-Bradley



Allen-Bradley

- **1904** : Fondation par Lynde Bradley & Stanton Allen (Compression Rheostat Company → Allen-Bradley en 1910)
- Spécialisation : **composants électriques** (boutons-poussoirs, contacteurs, minuteriers)
- **1985** : Rachat par Rockwell International (1,65 Md\$) → renforcement en automatisation
- Aujourd'hui : **marque phare Rockwell Automation** (MicroLogix, CompactLogix, ControlLogix, PowerFlex, PanelView...)
- **Position dominante en Amérique du Nord** : forte implantation historique + standardisation précoce



Le PLC-2 : premières générations (1980s)



- Modèle **robuste et modulaire**, représentatif des premiers automates.
- Programmation en **Ladder**, compatible Windows 95 → Vista (mode 16 bits jusqu'à Windows 8).
- Fonctions principales :
 - Supervision et transfert de programmes
 - Commandes : **Examine ON/OFF, Latch/Unlatch, TON, TOF**
- Limites : **architecture séquentielle**, faible connectivité réseau, puissance réduite.
- Évolution vers les gammes modernes (**CompactLogix, ControlLogix**) intégrant :
 - **EtherNet/IP, OPC UA**
 - **Architecture orientée objet** et traitement plus puissant → transition vers le **PAC**

Panorama des automates Rockwell



Micro800

Description :

Série d'automates compacts destinés aux petites machines ou systèmes autonomes. Idéals pour les applications simples nécessitant peu d'entrées/sorties et une logique séquentielle de base.

Caractéristiques :

- Programmation avec Connected Components Workbench (CCW)
- Modules compacts et modulaires pour ajouter des E/S ou des communications
- Compatibles avec Ethernet/IP sur certains modèles
- Bon rapport qualité/prix pour petites applications

Limitation :

Capacité mémoire et puissance de calcul limitées, peu adaptées aux systèmes complexes ou aux environnements nécessitant de nombreuses communications et fonctionnalités avancées.



CompactLogix

Description :

Automates de taille moyenne, adaptés aux lignes de production ou aux systèmes nécessitant un contrôle plus avancé que les Micro800, tout en restant compacts et modulaires.

Caractéristiques :

- Programmation avec Studio 5000 Logix Designer
- Compatibles Ethernet/IP natif
- Support du contrôle de mouvement (motion control) pour plusieurs axes
- Mémoire et puissance de calcul adaptées à des applications industrielles de taille moyenne
- Modules E/S locaux et distants

Limitation :

Moins extensibles et moins puissants que la gamme ControlLogix, donc pas idéaux pour les systèmes très complexes ou critiques.



ControlLogix

Description :

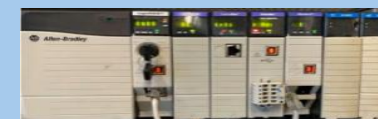
Gamme haute de gamme (PAC) destinée aux applications complexes et critiques, intégrant un haut niveau de puissance, de modularité et de communication.

Caractéristiques :

- Architecture modulaire orientée objet
- Programmation avec Studio 5000 Logix Designer
- Multi-protocoles : Ethernet/IP, ControlNet, DeviceNet, et possibilité d'ajouter d'autres modules de communication
- Contrôle de mouvement avancé, synchronisation multi-axes en temps réel
- Grande capacité mémoire et traitement rapide pour process continus et systèmes complexes
- Compatible avec les solutions logicielles Rockwell Automation (FactoryTalk, MES, SCADA)

Limitation :

Coût plus élevé à l'achat et à la maintenance, nécessitant une équipe formée à la programmation et au diagnostic des PAC Rockwell.





Différences CompactLogix vs ControlLogix

Point de comparaison	CompactLogix	ControlLogix
Positionnement	Gamme moyenne , adaptée aux petites et moyennes machines (200 à 300 points d'entrée sortie)	Gamme haut de gamme , destinée aux grandes applications industrielles
Taille du système	Plus compacte , pour machines individuelles ou petits process industriels	Très modulaire , idéale pour grandes usines, réseaux complexes et grand nombre de points d'E/S
Montage	Boîtier unique (compact) ou petit rack pour les modèles étendus	Châssis avec modules enfichables (rack)
CPU / Processeur	Moins puissant , une seule CPU par contrôleur	Très puissant , possibilité de plusieurs CPU dans un même châssis
Mémoire programme	Mémoire plus réduite	Grande capacité mémoire
Communication réseau	Principalement EtherNet/IP , plus quelques réseaux basiques	Supporte de nombreux réseaux industriels : EtherNet/IP, ControlNet, DeviceNet , etc. (pour communiquer avec des variateurs ou des robots)
Redondance / Haute disponibilité	Pas de redondance native	Possible (CPU redondantes, communication redondante)
Extensions	Nombre limité de modules d'extension	Large choix de modules d'extension : spécifiques, sécurité, motion control, etc.
Utilisation typique	Machines isolées , petites lignes de production, automatisation locale	Usines complètes , process continus, infrastructures critiques
Logiciel de programmation	Studio 5000 Logix Designer (même environnement que ControlLogix)	Studio 5000 Logix Designer
Prix indicatif	≈ 2 500 €	≈ 7 900 €

Le PAC et ses spécificités



Définition et caractéristiques

- **Origine :**
 - Evolution des PLC (2000s) face aux limites des relais câblés et de la logique séquentielle simple
- **Nouveaux besoins industriels :**
 - **Puissance de calcul** pour procédés complexes
 - **Intégration réseau** (machines, bases de données, supervision)
 - **Flexibilité logicielle** pour s'adapter rapidement
- **Rockwell Automation** popularise le concept avec **CompactLogix** et **ControlLogix**
- **Caractéristiques clés :**
 - Fiabilité du PLC + capacités d'un PC industriel
 - Gestion de la logique séquentielle et de processus complexes
 - Intégration réseau avancée et traitement de données → adapté à l'Industrie 4.0



Atouts techniques du PAC

Caractéristique	Éléments clés
Architecture avancée	▪ Modulaire et orientée objet ▪ Basée sur OS temps réel ▪ Évolution et personnalisation facilitées
Polyvalence de programmation	▪ Langages IEC 61131-3 (Ladder, FBD, ST, SFC...) ▪ Support de langages avancés (C/C++, scripts...)
Capacités réseau	▪ Multi-protocoles : EtherNet/IP, OPC UA, Modbus... ▪ Intégration IT/OT, bases SQL, Web Services
Puissance de calcul	▪ Adaptée aux systèmes complexes ▪ Calculs intensifs et synchronisation temps réel
Applications typiques	▪ Lignes de production intégrées ▪ Motion control multi-axes ▪ Robotique, vision industrielle, procédés continus



Différences PLC vs PAC

Critère	PLC (Automate Programmable Industriel)	PAC (Contrôleur d'Automatisation Programmable)
Origine	Développé pour remplacer les relais et systèmes câblés	Évolution du PLC , avec une architecture orientée PC et objets
Architecture	Architecture traditionnelle, séquentielle	Architecture modulaire, orientée objet , souvent sur OS temps réel
Langages de programmation	Langages IEC 61131-3 (LD, FBD, ST, SFC...)	Les mêmes langages IEC, avec en plus C/C++ , scripts, fonctions avancées
Capacités réseau	Limitée : Modbus, Profibus, Ethernet/IP	Étendue : Ethernet/IP, OPC UA, SQL, Web Services, intégration IT/OT
Puissance de traitement	Adaptée aux applications simples à moyennes	Plus élevée , adaptée aux systèmes complexes et au temps réel
Applications typiques	Machines industrielles, petites lignes de production	Systèmes complexes , robotique, vision, contrôle multi-axes
Maintenance	Simplicité, robustesse , maintenance facile	Plus complexe , mais diagnostics et analyse intégrés
Exemples de constructeurs	Siemens S7-1200/1500, Schneider M221/M241, Omron CP1H	Rockwell CompactLogix/ControlLogix, Siemens S7-1500T, Schneider M580, Beckhoff



Cas d'usage typiques

PLC

Cas d'usage typique	Description	Exemple concret
Machine d'assemblage simple	Commande de moteurs, vérins, capteurs sur une petite machine autonome	Une sertisseuse dans une ligne de production
Transporteurs	Gestion marche/arrêt, détection pièces , séquences simples	Convoyeur de palettes dans un entrepôt
Station de pompage	Gestion des pompes, niveaux, alarmes, cycles de démarrage	Poste de relevage dans un réseau d'eau
Machine d'emballage	Séquences répétitives avec peu de calculs complexes	Ensacheuse pour produits alimentaires

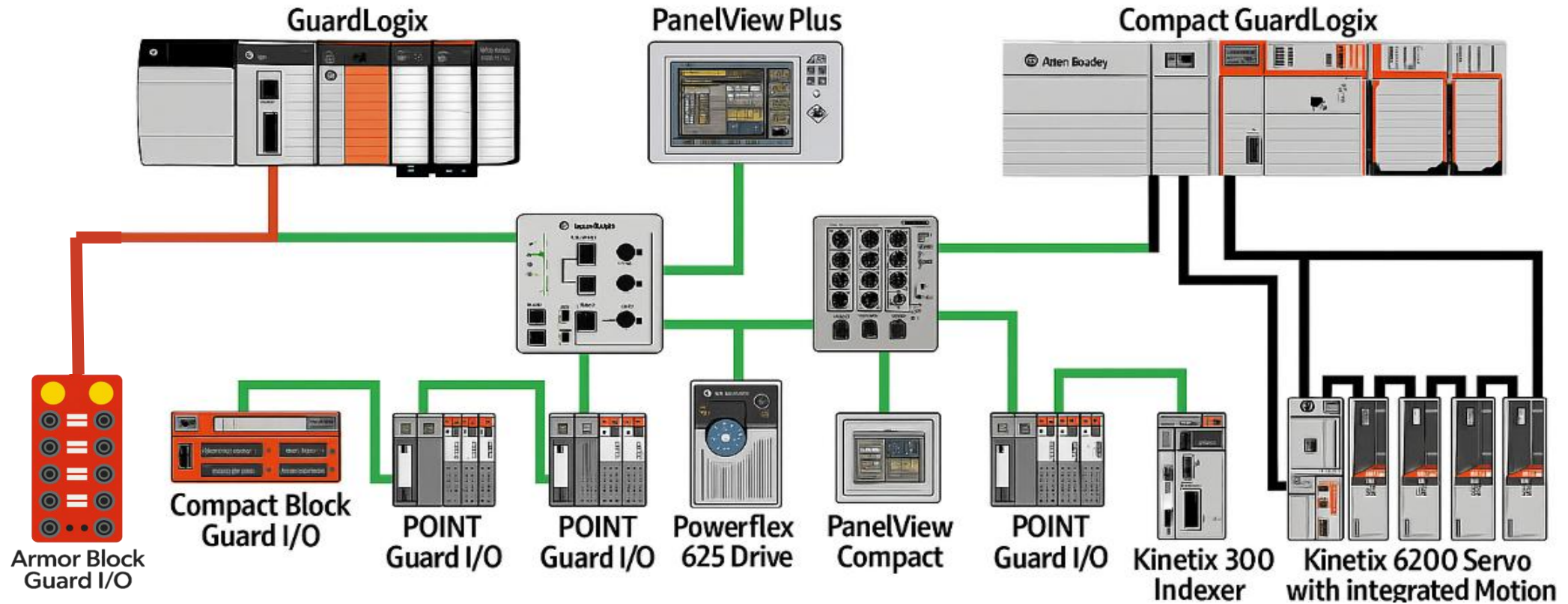
PAC

Cas d'usage typique	Description	Exemple concret
Ligne de production intégrée	Coordination de plusieurs machines, échanges de données avec ERP/MES	Ligne complète de fabrication de bouteilles
Motion Control multi-axes	Synchronisation précise de plusieurs moteurs en temps réel	Imprimante industrielle haute vitesse
Robotique et vision industrielle	Contrôle des robots + traitement d'images	Cellule robotisée pour soudage avec inspection caméra
Procédé continu complexe	Régulation multi-boucles , sécurité, analyse en ligne	Raffinerie, usine chimique, production pharmaceutique
Intégration IT/OT	Collecte et envoi de données vers le cloud, analyse big data	Usine connectée Industrie 4.0 avec prédiction de maintenance

Environnement Rockwell



Écosystème matériel



Rouge : Sécurité machine

Vert : Process

Noir : Motion

LOCOPLC



Écosystème matériel

Catégorie	Éléments clés
Contrôleurs	▪ GuardLogix : modulaire (rack), haute performance ▪ Compact GuardLogix : format compact pour applications moyennes ▪ Gestion logique standard + sécurité (SIL 3 / PL e)
I/O	▪ Armor / Compact Block Guard I/O : IP67, montage déporté proche machine ▪ POINT Guard I/O : modules en coffret ▪ Connexion via EtherNet/IP + CIP Safety
Interfaces opérateur (HMI)	▪ PanelView Plus : affichage, supervision, pilotage ▪ Reliés à l'automate en EtherNet/IP ▪ Gestion des alarmes & infos sécurité
Variateurs / Servo	▪ PowerFlex 525 : moteurs asynchrones (vitesse, couple), efficacité énergétique ▪ Kinetix 300 / 6200 : contrôle précis, motion multi-axes (Sécurité intégrée (ex. STO – Safe Torque Off))
Communication	▪ Réseau principal : EtherNet/IP (standard + sécurité CIP Safety) ▪ Extensions possibles : ControlNet (temps réel) / DeviceNet (capteurs intelligents)

Concepts logiciels

Fonctionnalités propres à Rockwell Automation (Studio 5000)

Routines

- Blocs de programme (Ladder, ST, FBD...) exécutés dans une Task.
- Chaque Program possède une Main Routine (point d'entrée) et peut appeler d'autres routines secondaires.

UDT (User Defined Data Type)

- Types de données personnalisés, véritables gabarits pour standardiser les variables.

AOI (Add-On Instruction)

- Blocs fonctionnels réutilisables qui encapsulent la logique et facilitent la standardisation.

Controller Fault Handler

- Routine centrale dédiée à la gestion des fautes majeures pour sécuriser et diagnostiquer.

Power-Up Handler

- Routine exécutée au redémarrage pour réinitialiser et remettre le système dans un état sûr.

Tasks (ordonnancement)

- Définissent quand et comment le programme s'exécute : Continuous, Periodic, Event.



Qu'est-ce qu'une Routine ?

- Une **routine** est un **bloc de programme** dans Studio 5000 (Ladder, ST, FBD, etc.)
- Elle contient **la logique qui s'exécute** (ex. commandes moteur, gestion d'alarmes, séquences)

Chaque **Program** est composé :

- **Main Routine** (obligatoire, point d'entrée),
- **Routines secondaires** appelées depuis la Main Routine ou via des conditions

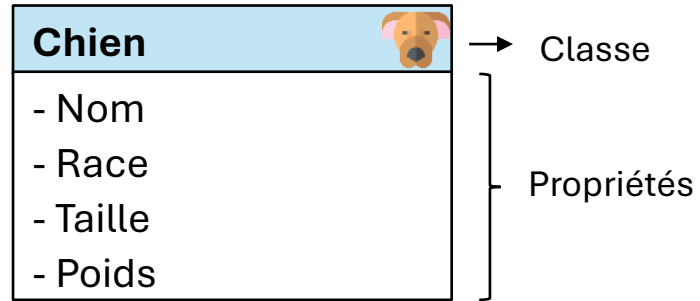
Les routines sont associées à une **Task**, qui détermine **quand et à quelle fréquence** elles s'exécutent (Continuous, Periodic, Event)



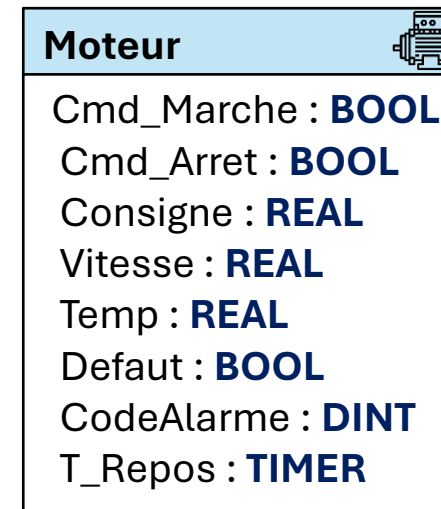
Qu'est-ce qu'un UDT ?

- Un **UDT (User Defined Data Type)** est un **type de données structuré** défini par l'utilisateur, qui regroupe plusieurs variables hétérogènes (BOOL, INT, REAL, TIMER...) sous un **même modèle**
- **Principe clé** : un UDT **ne contient que des données**. La **logique** est programmée séparément (ex. dans une AOI). Chaque instance est un **tag typé**, avec sa **propre mémoire** et des **valeurs indépendantes**
- Les UDT s'inscrivent dans une **logique de programmation orientée objet** : on définit un **gabarit réutilisable**, puis on crée des **objets** concrets (instances) à partir de ce gabarit
- Cela permet une **standardisation**, une **réutilisation** accélérée et une meilleure intégration avec les **IHM** (accès direct aux sous-membres)

Parallèle entre Programmation Orientée Objet et UDT



*Exemple de Classe
(programmation orientée objet)*



*Exemple d'UDT (User Defined
Data Type) en automatisation*



Qu'est-ce qu'un AOI ?

- Un **bloc fonctionnel** (instruction personnalisée) **réutilisable** dans Studio 5000, avec **paramètres** (In/Out/InOut) et **variables interne**
- **Encapsulation et révision** : publié/versionné comme une instruction native, traçable et partageable en bibliothèque
- S'inscrit dans une démarche **orientée objet** : on définit un comportement standard (moteur, vanne, axe) instanciable partout
- Travaille souvent avec un **UDT** passé en paramètres (ex. Cmd, Sts) : l'AOI exécute la logique et met à jour les données



UDT vs AOI — rôles et complémentarité

Aspect	UDT	AOI (bloc fonctionnel)
Nature	Type de données structuré (aucune logique)	Bloc fonctionnel encapsulant la logique
Rôle	Modéliser les données d'un objet (Cmd, Sts, Mes, Alm...)	Implémenter le comportement (séquences, protections, calculs)
Orienté objet	Gabarit de données → instances de tags	Comportement réutilisable → instances d'instruction
Réutilisation	Instancier des tags typés (mémoire indépendante)	Appeler le même code avec des paramètres (révisionnée)
HMI / MES	Accès aux sous-membres (Equip.Sts.Vitesse)	Expose des paramètres, lit/écrit dans les UDT
Quand l'utiliser ?	Standardiser les structures de variables	Standardiser la logique d'équipements/fonctions
Règle simple	UDT = données	AOI = logique (bloc fonctionnel)

Qu'est-ce que c'est le « Controller Fault Handler »

- **Routine système dédiée** chez Allen-Bradley, exécutée lors d'une faute majeure détectée par la CPU (ex. division par 0, perte d'I/O critique, dépassement watchdog)
- **Point centralisé** pour gérer les réactions d'urgence logicielles : couper, enregistrer, informer, préparer le redémarrage
- **Différence constructeur** : chez Siemens/Autres, logique éclatée dans des OB de diagnostic (OB80/121/122...), pas de routine universelle unique
- **Bénéfices** : diagnostic **plus rapide**, expérience HMI cohérente, standard d'entreprise reproductible d'un projet à l'autre



Ce que le Controller Fault Handler n'est pas

- **Ce n'est pas** une fonction Safety certifiée (EN ISO 13849 / IEC 61508) : c'est du logiciel, non sûr par conception.
- En **cas de panne matérielle grave** (CPU HS, coupure générale), la routine ne s'exécute pas.
- **Rôle complémentaire** à la Safety matérielle :
 - Safety (STO, arrêts d'urgence, modules Guard) **coupe l'énergie**.
 - CFH **documente la faute**, place le process en configuration sûre, guide l'opérateur pour un redémarrage sûr.



Qu'est-ce que c'est le « Power-Up Handler »

- **Routine spéciale exécutée une seule fois au démarrage** du contrôleur : après **mise sous tension / redémarrage / retour en RUN** suivant la configuration
- Elle s'exécute avant **l'exécution normale** des tasks et permet de **préparer l'état du système**

Objectif : replacer l'installation dans un état sûr et connu avant de démarrer la logique standard



À quoi sert-il concrètement ?

- **Initialiser** des compteurs, mémoires, pas de séquence, états internes.
- **Forcer/relâcher** des sorties vers un état sûr (frein, vannes fermées, vitesses = 0).
- **Recalculer des paramètres** (consignes par défaut, filtres, offsets capteurs).
- **Purger / remettre à zéro** des buffers de com., files d'alarmes, journaux temporaires.
- Lever des drapeaux HMI (ex. “Démarrage à froid”, “Reprise autorisée”).



Qu'est-ce qu'une Task ?

- Une **Task** définit quand et comment le contrôleur exécute ton programme (ordonnancement)
- Elle contient des **Programs/Routines** et est gérée par un **scheduler** du CPU
- **Types : Continuous** (boucle continue), **Periodic** (à période fixe), **Event** (sur événement)
- **Priorité** : nombre bas = plus prioritaire ; une Task prioritaire peut **préempter** les autres

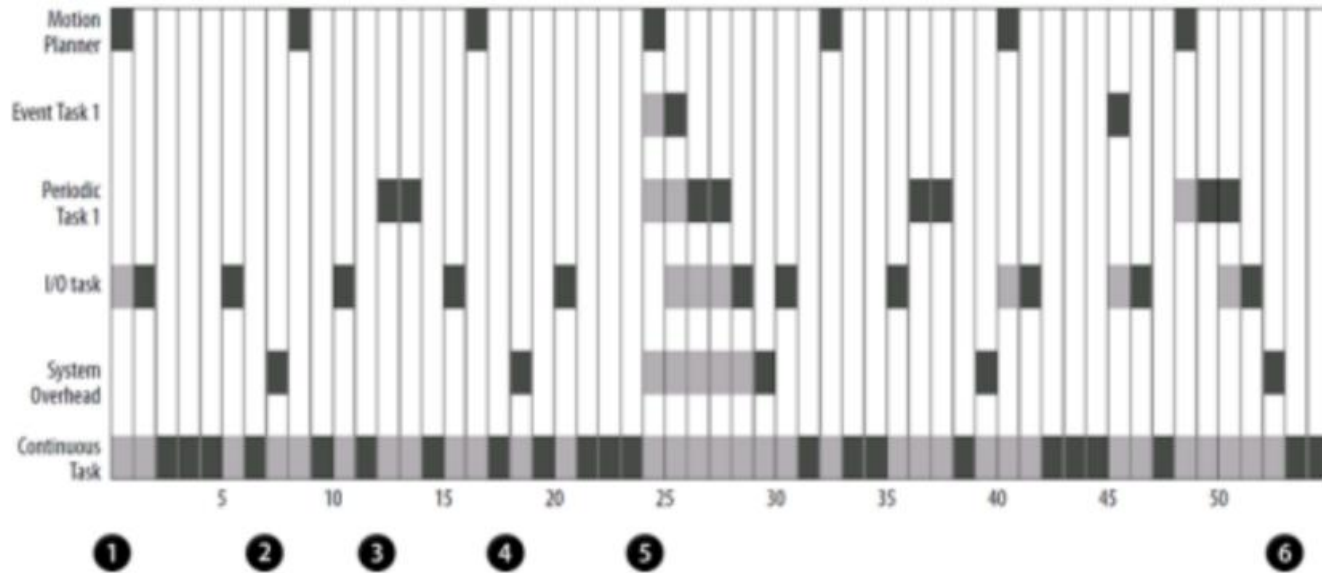


Les 3 types de Tasks (vue d'ensemble)

Type	Principe	Avantages	Limites	Usages typiques
Continuous	Boucle continue tant qu'il reste des rungs à exécuter.	Simple à mettre en œuvre.	Pas de période garantie (temps variable) ; peut ralentir la CPU si long.	Programmes simples , sans contraintes temps réel.
Periodic	Exécution à intervalle fixe (ex. 10 ms). Peut interrompre la Continuous.	Cycle stable et prédictible ; indispensable pour PID/asservissements .	Intervalle trop court ⇒ surcharge CPU (overlap).	Régulation, motion cyclique, acquisition régulière.
Event	Déclenchée par un événement (interruption module, changement tag, top codeur).	Réaction très rapide et prioritaire ; n'exécute que quand nécessaire.	Configuration plus poussée (sélection des déclencheurs).	Arrêt d'urgence, fronts rapides, interruptions com/I-O.



Bonnes pratiques & dimensionnement



Exemple visuel d'ordonnancement

Architecture conseillée :

1. **Periodic rapide** (ex. 5–10 ms) pour boucles critiques (PID, motion)
2. **Periodic moyenne** (20–50 ms) pour logique machine, séquences
3. **Continuous** pour supervision/HMI non critiques
4. **Event** pour événements matériels (E-Stop, top codeur, perte com)

Simulation et mise en œuvre

Studio 5000 Logix Emulate

Rôle :

- Simuler le fonctionnement d'un **automate Allen-Bradley (ControlLogix ou CompactLogix)** directement sur PC
- Permet de tester et déboguer les programmes **sans matériel physique**

Fonctionnement :

- Exécute la logique Ladder, ST, FBD... comme un automate réel
- Interface présentée sous forme de **rack virtuel avec slots numérotés**

Avantages :

- Tester et valider la logique avant mise en service
- Déboguer rapidement sans matériel

Limites :

- Ne simule pas le **process physique** (moteurs, capteurs, actionneurs)



RSLinX Classic

Rôle :

- Assurer la **communication** entre le PC et les automates Allen-Bradley (**réels ou simulés**).
- Sert d'**interface de liaison** entre Studio 5000, Logix Emulate, et les automates.

Nature :

- **Serveur de communication** développé par Rockwell.
- Fournit des **drivers** (OPC, Virtual Backplane, Ethernet/IP, USB, etc.).

The logo for RSLinx Classic, with "RSLinx" in red and "Classic" in grey, both in a stylized font.

Fonctionnement :

- Traduit les instructions de Studio 5000 en protocoles compréhensibles par le CPU.
- En simulation : utilise le **Virtual Backplane Driver** pour relier Studio 5000 au CPU émulé (Logix Emulate).
- En réel : permet la communication avec un automate physique via **EtherNet/IP, USB, RS-232, ControlNet, DeviceNet**.

En simulation :

- Indispensable même si tout est installé sur le **même PC**.
- Studio 5000 “voit” le CPU virtuel comme s’il s’agissait d’un **automate physique**.

Procédure de lancement d'une simulation

1. Préparation

- Ouvrir **Logix Emulate 5000** et insérer un CPU virtuel
- Vérifier que le slot CPU est bien défini (ex. slot 2)

2. Configuration RSLinx Classic

- Ouvrir **RSLinx Classic** → *Configure Drivers*
- Choisir **Virtual Backplane (AB_VBP)**
- Associer le driver au **slot CPU** de l'émulateur

3. Création du projet dans Studio 5000

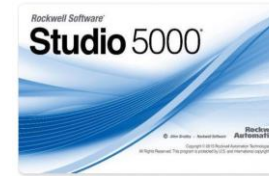
- Nouveau projet → choisir **Studio 5000 Logix Emulate Controller**
- Sélectionner le **firmware** identique à l'émulateur
- Associer le **slot configuré dans RSLinx**

4. Lancement de la communication

- Cliquer sur **Who Active** (Studio 5000)
- Sélectionner le driver RSLinx et le CPU virtuel
- Télécharger le programme vers l'émulateur

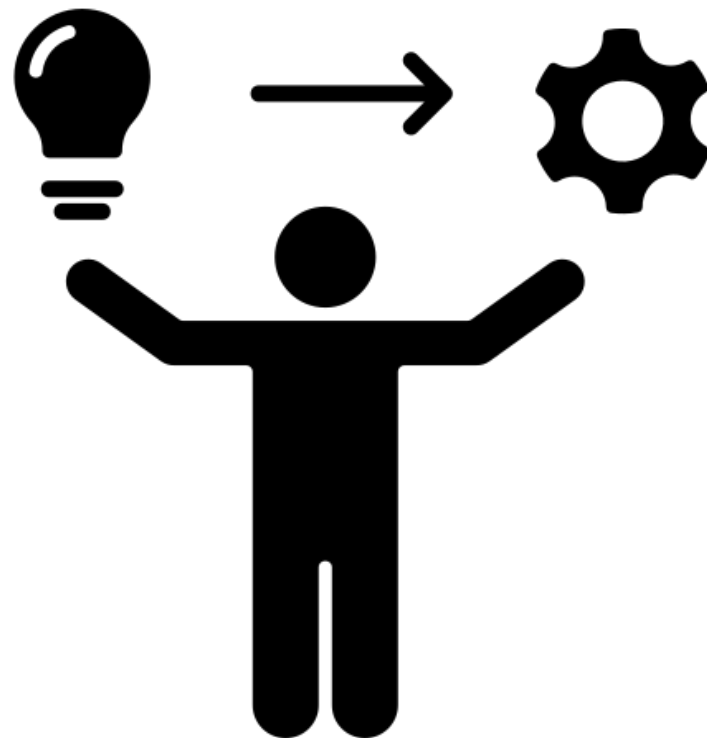
5. Exécution

- Passer le CPU virtuel en **Run Mode**
- Vérifier le comportement du programme dans l'émulateur (ou via HMI/Factory I/O)



➡ **RSLinx® Classic** ➡





Pratique