



HEPL

Haute Ecole de la Province de Liège



Automation

Automation

Introduction :

Ce cours s'adresse aux étudiants de deuxième année du bachelier en informatique et système finalité industrielle.

L'informaticien industriel doit être capable de mettre en œuvre une application de **contrôle** et de **supervision d'automatisme**. Les systèmes informatiques ainsi que les automatismes à contrôler peuvent être variés et donc de nombreuses possibilités doivent être vues dans les différents cours de ce bachelier.

Les automatismes peuvent se retrouver dans différents domaines d'activités tels que l'aérospatiale, la domotique, l'automobile, le monde industriel, le contrôle d'accès, l'imagerie médicale, ...

De même les systèmes informatiques utilisés afin de contrôler ces systèmes peuvent également être différents. Nous pouvons trouver des **PC**, des **systèmes embarqués**, des **automates programmables**, ...

Un **automate** programmable industriel ou API est un **système informatique adapté aux contraintes du monde industriel**. Il est généralement utilisé dans des chaînes de productions qui demandent d'avoir une précision et une fiabilité maximale avec une longévité importante du matériel. Cette longévité concerne également la disponibilité des pièces de rechange.

Automation

Introduction :

L'objectif de ce cours est donc de **s'initier** à l'utilisation et à la **programmation** d'automate programmable industriel. L'étudiant sera capable d'analyser un système à **automatiser** et de concevoir un programme réalisant son automatisation sur un automate programmable (souvent noter API dans la littérature).

Nous apprendrons également à **analyser** et **comprendre** un **programme existant** dans les différents langages standards utilisés dans l'automation. Suivant le modèle d'automate, la **méthode d'écriture** du programme sera différente.

On peut comparer à la programmation informatique en choisissant par exemple le langage C ou le langage Python pour réaliser son programme . En automation, vous allez trouver des méthodes de programmation basée sur une symbolique particulière qui définit les grands types de programmation utilisés et avec en plus, un langage propre à la marque d'automate (Schneider, Siemens, Omron, ...).

la plate-forme **M340** de **Schneider Electric** est l'automate programmable industriel utilisé afin de réaliser nos applications de contrôles et de supervisions en laboratoire.

Automation

IT et OT :

L'IT : Les technologies de l'information (ou **IT** pour **Information Technology**) représentent le domaine du traitement de l'information par l'utilisation de matériel informatique (PC, Serveur, Switch, Routeur, ...) et d'outils logiciels (ERP, GED, logiciels de développements, gestion de fichiers, sauvegarde, ...)

Les données reprises dans le domaine de l'IT concerne l'ensemble des éléments de l'entreprise permettant de travailler entre les membres de l'entreprise et les clients ou fournisseurs. Elle reprend les outils nécessaires aux services commerciaux, administratifs et les finances.

Cette technologie pilote les **SIE**, qui sont les **Systèmes d'Information de l'Entreprise**.

Le service « **IT** » a en charge la **gestion des équipements** permettant de faire fonctionner le réseau, la bureautique, le réseau Wifi, la téléphonie VOIP, ...

C'est lui **qui organise** et assure le bon **fonctionnement** de tous les **outils de communication et d'information de la société**.

Automation

IT et OT :

L'OT : OT (ou **Operational Technology**) – est moins connu que l'IT, même dans le milieu industriel. Il existe pourtant depuis toujours, depuis que chaînes industrielles et outils de production se sont digitalisés. L'OT désigne la technologie informatique qui s'occupe de la partie process. Elle a en charge les systèmes d'information industriels, que l'on appelle **SII** pour **Système de l'Information Industrielle**.

l'OT reprend l'ensemble des compétences possibles dédiées au fonctionnement de l'outil de production : la robotique, les machines-outils, les tapis de transport de matériel, les étiqueteuses, les palettiseurs, ... C'est l'ensemble des technologies, des langages informatiques, des matériels, réseaux et logiciels utiles à la fabrication de produits, machines, pièces détachées, ...

on parle le plus souvent d'**automaticien**, de **responsable de la maintenance industrielle**, **d'équipe en charge de l'informatique industrielle**, de **responsable de production**.

L'objectif de **l'OT** est d'assurer **sécurité** et **pérennité** du **fonctionnement** de l'outil de production.

Automation

Les deux métiers de l'automation :

Partie Automate :

Une première partie du rôle du programmeur est de **rédiger le programme** selon l'analyse fonctionnelle et le cahier des charges. Ce programme permettra de piloter la ligne de production.

Ce dernier est injecté dans l'automate et va être constamment en dialogue avec l'environnement de la ligne de production par l'intermédiaire des différents **actionneurs et capteurs**.

Partie Supervision :

Sur un système de pilotage de ligne de production, on trouve également des éléments propre au matériel faisant partie de l'automation autre que l'automate et ces entrées et sorties. Un deuxième métier va consister en la **réalisation d'écrans graphiques** permettant une interaction entre l'utilisateur de la ligne et le programme présent dans l'automate. Cette interaction étant bidirectionnelle.

Automation

Les deux métiers de l'automation :

Le matériel de supervision :

Il est constitué de PC industriel ou d'écran dédié équipé de mémoire et module de communication, ces derniers sont souvent tactile.



Automation

Les deux métiers de l'automation :

Dans la pratique :

Les **programmeurs « automate »** vont travailler en étroite collaboration avec les personnes qui réalisent les « **écrans de supervision** » dans le cadre de grands projets. Si le process à réaliser est petit, il faut être capable de mener à bien les deux métiers. Il existe toute une série de chose à prendre en compte pour avoir un outil efficace pour le personnel qui pilote la ligne de production et également pour le personnel de maintenance.

En plus de la programmation des mouvements, il va être nécessaire de définir une bonne gestion des défauts de l'installation. Ces derniers seront remonté vers la supervision.

La supervision permettra de visualiser la ligne de production avec animation des mouvements et elle devra prévoir une partie dédiée aux défauts détectés par l'automate.

Nous discuterons plus tard de la gestion des défauts, cette dernière doit être programmée de façon très rigoureuse avec une méthode bien précise.

Automation

Les deux métiers de l'automation :

Dans la pratique :

En plus de ces deux grands groupes qui sont « l'automate » et ça « supervision », il apparait de plus en plus nécessaire d'avoir une **liaison** entre le monde de **l'industrie** et le monde de **l'informatique**.

En effet, il est nécessaire de connaître les données de production, de suivre la capacité de la ligne, de mesurer son rendement, ...

Il va donc y avoir une étroite relation avec l'informatique de l'entreprise afin de mettre à disposition l'ensemble de ces données.

Nous aurons une ou plusieurs personnes qui auront pour responsabilité de s'occuper du traitement de ces données. Il faudra programmer la collecte de ces dernières soit côté automate soit sur la supervision et parfois dans les deux environnements.

L'informaticien en charge de ce travail devra avoir une **bonne connaissance** du milieu de **l'automatisme** (automate, supervision), des **bases de données** et des **réseaux** industriels et classiques.

L'évolution de l'industrie

- Il est évident que le monde de l'industrie et ces grands acteurs ont pris les devants et mettent en place toute une série d'outils à la disposition des ingénieurs, informaticiens, opérateurs de ligne, ... L'ensemble des décideurs de l'environnement de production est touché par ces nouvelles technologies.
- Nous retrouvons le concept abordé dans les cours de logiciel de contrôle concernant l'industrie 4.0 et les objets connectés.
- L'Industrie 4.0 est toujours au début de son développement : l'Industrie 4.0, c'est l'usine ultra-connectée du futur qui s'annonce comme la prochaine « révolution industrielle ». Le **concept** de **connecter** les **machines** d'une usine ou de plusieurs usines par un intranet possède un grand potentiel pour l'optimisation des process manufacturiers, pour la réduction des matériaux utilisés et pour une plus grande flexibilité à réagir aux besoins individuels de chaque client. C'est l'arrivée des objets connectés (**IOT**) et des programmes machine dédiés (MindSphere).

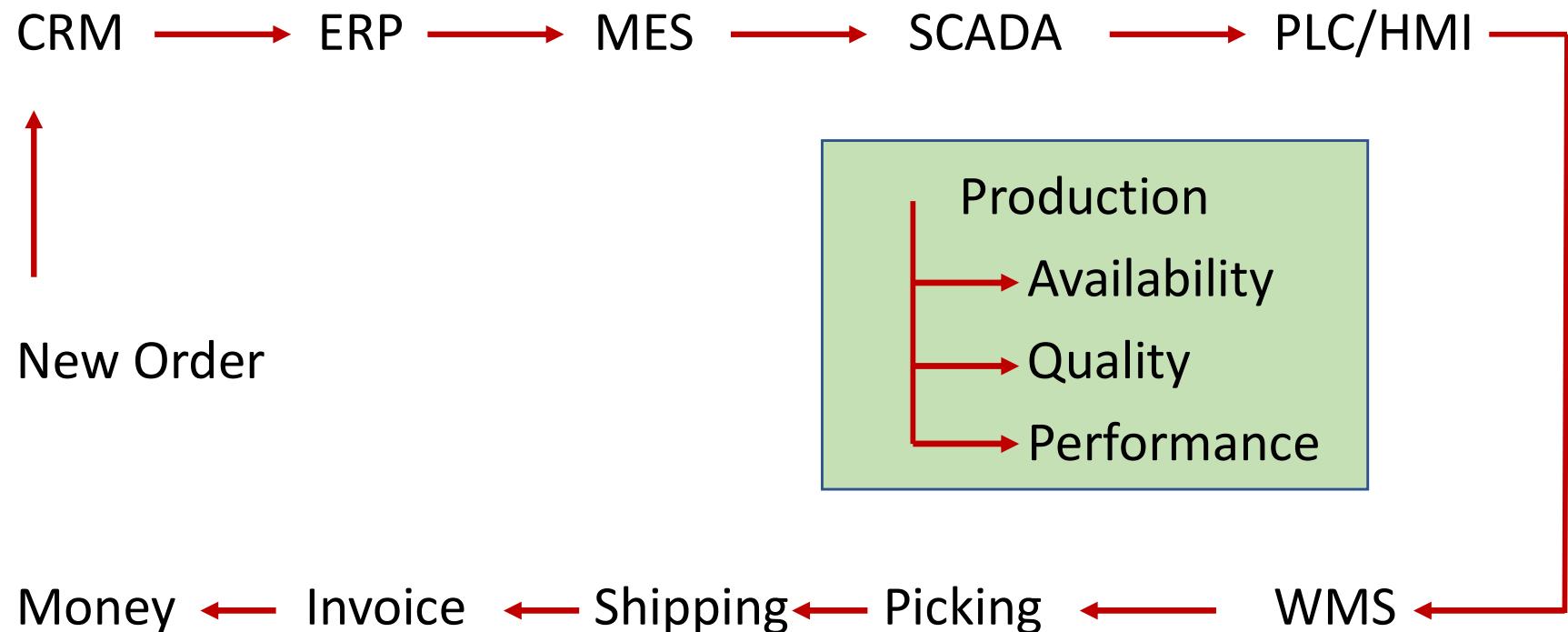
Extrait de : gimelec-industrie-4.0-lusine-connectee-septembre-2013

L'évolution de l'industrie

- Loin d'être de simples vitrines technologiques, ces initiatives sont associées au minimum à des objectifs très précis de gains d'efficacité. Dans certaines entreprises, elles **visent** plus fondamentalement à **repenser la production**, pour la rendre plus flexible, voire le produit lui-même, en lui associant des services numériques.
- l'Industrie 4.0 vise à faire **converger le monde virtuel**, de la conception aux systèmes de gestion (logistique, finance, etc.), avec les machines de production et les produits de fabrication.
- La réflexion tourne autour de la **méthode de production** et de **comment l'améliorer**, d'un point de vue technique, mais pas seulement, l'enjeu majeur consiste à optimiser les processus de fabrications et de **prévenir** d'éventuels problèmes sur les lignes de production.

L'évolution de l'industrie

- L'état de l'art actuel dans les entreprises :



L'évolution de l'industrie

- **SCADA** : Est un système de contrôle et d'acquisition de données en temps réel (**SCADA**) (anglais : Supervisory Control And Data Acquisition) est un système de gestion à grande échelle permettant de traiter en temps réel un grand nombre de mesures et de contrôler à distance des installations techniques.
- **CRM** : est l'acronyme de **Customer Relationship Management** qui signifie en français Gestion de la Relation Client (GRC). Le **CRM** a pour objectif de s'occuper du traitement des données clients et prospects afin d'en optimiser le flux.
- **ERP** : (Enterprise Resource Planning) ou également appelé PGI (Progiciel de Gestion Intégré) est un système d'information qui permet de gérer et suivre au quotidien, l'ensemble des informations et des services opérationnels d'une entreprise.
- **MES** : Est un logiciel de pilotage de la production (en anglais américain **manufacturing execution system** ou MES) est un logiciel collectant en temps réel les données de production d'une usine ou d'un atelier, données qui sont analysées quant à la traçabilité, le contrôle de la qualité, le suivi de production.
- **PLC / HMI** : Ce sont les systèmes de pilotage de la ligne de production (PLC = Programmable Logic Device et HMI = Human Machine Interface), ils sont à la base du pilotage de la production, on parle de contrôle commande. Nous nous situons au niveau Automate (PLC) et supervision (HMI).

L'évolution de l'industrie

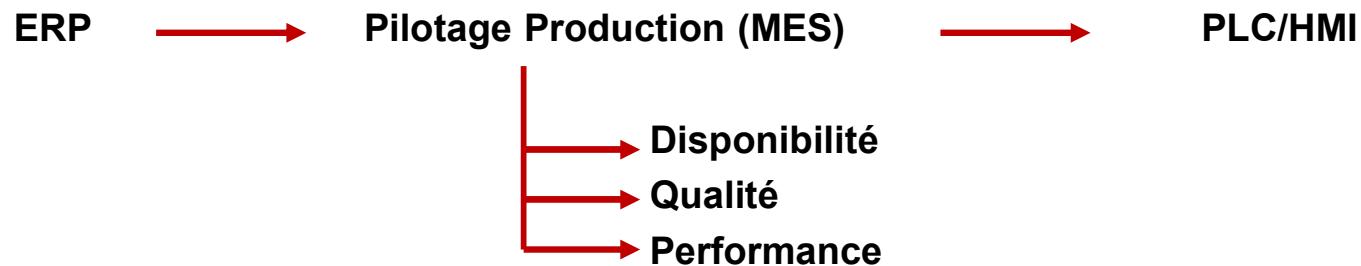
- **WMS** : Est un système qui optimise et gère la gestion des stocks au sein d'un entrepôt. L'acronyme WMS signifie en anglais Warehouse Management System, que l'on peut traduire par Système de gestion d'entrepôt. Il permet de savoir à tout moment ce qui est ou non disponible en stock dans un ou des entrepôts.
- **Shipping** : est la partie qui concerne la livraison du produit vers le client suivant la commande que l'on retrouve dans l'ERP. Il est compléter de documents fourni par l'ERP et préparer via le **Picking** dans l'entrepôt.
- **Invoice** : Correspond à la facturation de la commande client qui, en général est établie également dans l'ERP si le flux est bien respecté.
- **Money** : Constitue la dernière étape, elle représente le paiement de la commande par le client, elle est également suivie dans l'ERP. Mais elle est souvent suivie, d'un point de vue comptable, hors ERP dans les plus petites structures.

La partie industrie 4.0 interviendra au niveau de la production, le but, comme expliqué précédemment, étant d'améliorer les process, le rendement, la maintenance,... , de cette dernière. Aujourd'hui, la vision de cette industrie 4.0 va plus loin, et il existe des projets visant directement par exemple, la gestion de l'approvisionnement de matière première pour la production, la collecte de données permet d'affiner le suivi du process.

Mais il est intéressant de noté que depuis le 3.0, beaucoup de choses sont à disposition.

L'évolution de l'industrie

- Etude de l'industrie 3.0 et de ces possibilités :



On va introduire la notion d'indicateur :

OEE : est un indicateur de performance synthétique qui permet de mesurer l'efficacité d'une ligne de production. Cet indicateur tient compte de la disponibilité de l'équipement, de la performance de la machine ainsi que de la qualité des matières produites. Il permet de quantifier la production en indiquant la proportion de produits qui respectent le cahier des charges (produits qui peuvent être vendus).

Traduction : Overall Equipment Effectiveness ou Efficiency Globale des Équipements.

Le But étant de savoir comment cet indicateur est calculé, d'où proviennent les données qui permettent ça mise en œuvre dans l'industrie 3.0 et comment nous allons pouvoir intervenir sur la ligne de production pour modifier ce paramètre afin d'améliorer la quantité de produits bons. Ensuite nous passerons au 4.0 pour voir la différence.

Automation

L'automation :

Définition : La définition de l'automation est la capacité de piloter les machines de façon automatique.

Il est nécessaire de rédiger **un cahier des charges**, de faire **une analyse fonctionnelle** du process à réaliser.

Ces éléments permettent de prévoir le matériel et les besoins en mesure, commande. Ces derniers vont définir la programmation de l'automate pour le pilotage optimal des machines qui composent la ligne de production.

- Les besoins des secteurs sont très variés mais les principales problématiques auxquelles les industriels sont confrontés, qu'ils soient issus de grands groupes ou de PME / PMI, sont de trois types :
- **La conception** : on part de zéro.
- **La modification ou l'ajout d'équipement** : On parle alors d'amélioration continue. Le client a par exemple une machine sur laquelle il veut ajouter un tapis roulant.
- **Le revamping** : Il est nécessaire lorsque les automates sont devenus obsolètes et qu'ils doivent être remplacés : il faut alors redévelopper les applications du client sur du matériel plus récent. C'est le même cas lors de matériel devenu trop vieux.

Automation

Le cahier des charges et l'analyse fonctionnelle :

Par quoi commencer ? On aura une préférence à décrire ce que le **processus doit réaliser**, par exemple un tapis qui conduit de la matière provenant d'un silo à un élément qui permet de mélanger cette dernière.

Il apparaît que pour réaliser cet automatisme, il est nécessaire de définir les différentes opérations, ensuite de descendre plus bas dans le raisonnement et de lister l'ensemble des mouvements, commandes, prise de mesure, ...

Ces éléments vont faire partie de l'analyse du développement à faire et constituent **l'analyse fonctionnelle** de notre processus.

Pour le **cahier des charges**, on va devoir, tout comme pour l'analyse fonctionnelle, rédiger l'ensemble des besoins nécessaires à la mise en œuvre de notre process. La différence se situe dans la description précise des éléments nécessaires pour y parvenir.

On entend par là, les divers composants comme les moteurs, tapis, contact de fin de course, type d'automate en fonction du nombre d'entrée / sortie, ...

Automation

Le cahier des charges et l'analyse fonctionnelle :

En général le **cahier des charges englobe l'ensemble** des **besoins** nécessaire au projet. Ce qui **inclus l'analyse fonctionnelle** et le **matériel**.

La description des différentes parties devra être la plus précise possible. Ce cahier des charges va être utilisé pour **soumissionner** le projet aux entreprises désireuses de réaliser le dossier. Il est parfois exécuter en interne dans un département spécialisé d'une société ou d'un groupe.

Cette méthode est valable pour les trois éléments cité précédemment :

- **Conception** « from scratch »
- **Ajout** d'un nouvel équipement **ou modification** d'un existant
- **Revamping** d'une ligne complète. On entend par Revamping : modernisation d'une ligne de production, ce qui passe en général par le remplacement de la partie automates, mesures et d'autres éléments afin d'améliorer la vitesse de production mais aussi le rendement de la ligne.

Automation

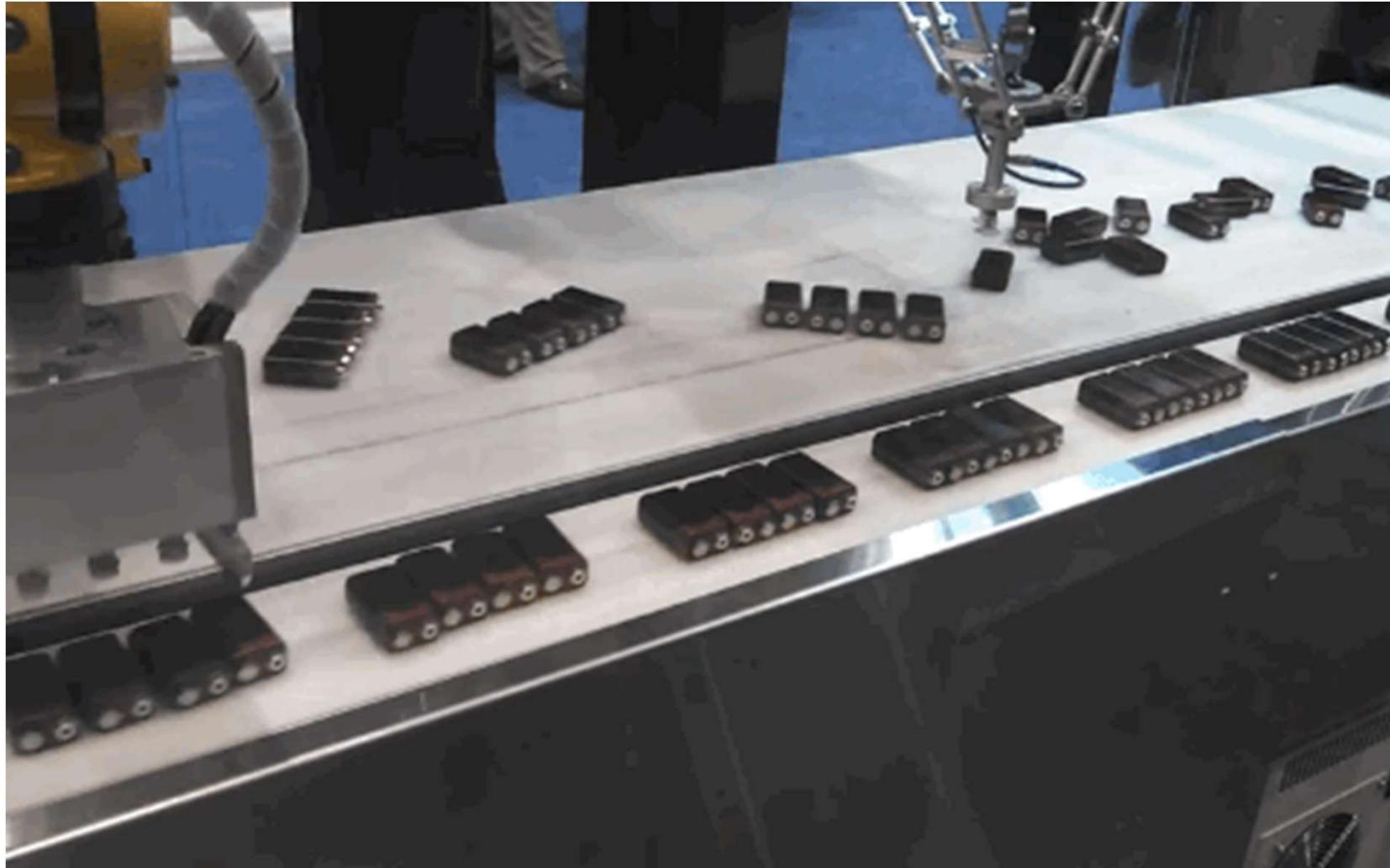
Le cahier des charges et l'analyse fonctionnelle :

A votre avis, quels sont les points de ce cahier des charges qui seront des éléments dont il faudra prendre en compte pour la partie développement informatique?

- L'analyse fonctionnelle car elle va décrire les différentes étapes qui constituent le processus à réaliser.
- Les éléments de puissances (moteurs, bras articulé, robot, vanne, piston, ...). Il font partie de la liste des éléments à commander (sorties),
- Les mesures de position, vitesse, poids, ... (entrées),
- Le type d'automate à choisir en fonction de la liste des entrées / sorties, mais également en fonction du type d'environnement de travail (Humidité, température, poussières, ...).
- La méthode de conduite de la ligne de production, elle va définir d'autres éléments à prévoir en plus de l'automate qui a pour rôle principal, le pilotage de la partie répétitive du cycle de la ligne.

Automation

Exemple :



Automation

Classement des systèmes :

On classe les problèmes à étudier en deux groupes :

- Les systèmes à **logique combinatoire**, à une combinaison des entrées correspond un et un seul état d'une même sortie. Il s'agit de systèmes répondant à une table de vérité ou une combinaison des entrées donne un état des sorties.
- Les systèmes à **logique séquentielle**, à plusieurs combinaison des entrées, correspond plusieurs état d'une même sortie.

La logique séquentielle se distingue de la **logique combinatoire** par le fait que dans cette dernière, **les sorties ne réagissent qu'aux entrées**, sans que le système ne soit **sensible à l'histoire** de ces **entrées**, Par contre en **logique séquentielle**. Il faut alors **toujours prendre en compte** les **séquences** d'entrées **et** de sorties du système que l'on veut contrôler.

Automation

Classement des systèmes :

En logique combinatoire :

L'état de la (ou des) sortie(s) à un instant donné ne dépend que du circuit et de la valeur des entrées à cet instant.

En logique séquentielle :

L'état de sortie du circuit à un instant donné dépend de la valeur des entrées à cet instant et de la valeur de la (ou des) sortie(s) aux instants antérieurs.

La **logique séquentielle** fait donc intervenir la **notion de mémoire** contrairement à la logique combinatoire. Là est la différence.

Pour passer d'une logique à l'autre, pas besoin d'éléments nouveaux : il suffit, par exemple, que la sortie d'une porte logique soit bouclée sur son entrée. On passe alors d'un système combinatoire à un système séquentiel.

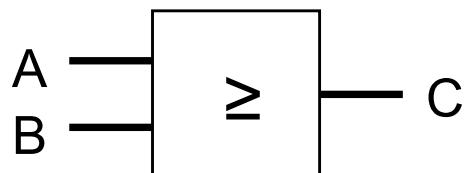
Automation

Classement des systèmes :

Exemple :

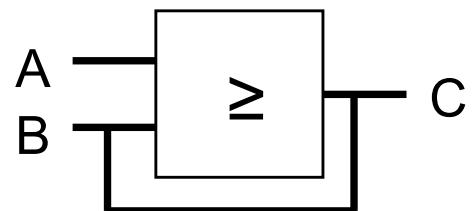
Utilisation d'une porte ou dans le cas d'un système combinatoire et transformation de cette dernière pour obtenir un système séquentiel.

Système Combinatoire



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Système Séquentiel



A	B	C
0	0	0,1
0	1	1
1	0	1
1	1	1

Si de base, A et C sont à 0 et que A passe à 1 alors C = 1. Si maintenant A repasse à 0, C reste à 1. La porte logique a gardé en mémoire l'état antérieur de la sortie.

Dans cette porte OU, il suffit que l'une ou l'autre de ces entrées soit à 1 pour que la sortie le soit aussi. La sortie étant reliée à l'entrée, il suffit que S soit une fois à 1 pour qu'elle le soit continuellement. Donc dans la table de vérité, le fait d'avoir A et B à 0 peut avoir comme résultat C = 1.

Automation

Classement des systèmes :

Système Séquentiel :

Dans la plupart des problèmes que l'on devra mettre en équation pour mettre en place un automatisme, il faudra utiliser des systèmes séquentiels. Nous allons nous intéresser à l'analyse et aux méthodes de programmation de ces derniers.

Exemple de système séquentiel : Pilotage d'un moteur avec deux boutons poussoir, nous allons regarder plusieurs cas différents pour le fonctionnement du moteur en fonction des éléments de commande et de la façon de connecter ces derniers.

- Utilisation d'un moteur avec un bouton marche, un bouton arrêt et une sortie pour piloter la partie puissance du moteur. Marche commande le démarrage (un appui simple) et arrêt commande le stop du moteur (un appui simple). Il faudra un relai de puissance pour faire fonctionner le moteur, car l'automate est la partie contrôle / commande de l'installation et fonctionne en courant faible comme pour l'Arduino ou le Raspberry vu précédemment.

Automation

Système séquentiel :

Schéma de l'installation :

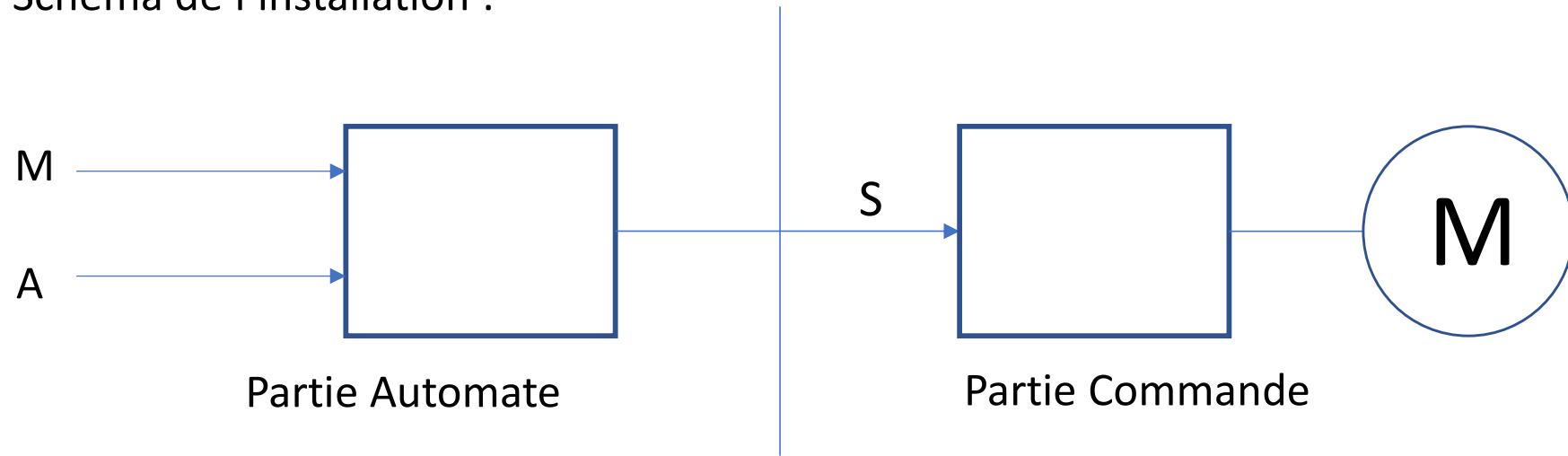


Table de vérité :

A	M	S
0	0	0
0	1	0
0	0	1
1	0	0
1	1	??

arrêt

Mise en marche

marche

Mise à l'arrêt

Arrêt prioritaire 0 ou Marche prioritaire 1 ?

Nous pouvons observer que la ligne 1 et 3 présentent une même combinaison des entrées et une combinaison différente des sorties donc on a un Système Séquentiel.

Automation

Système séquentiel :

Schéma de l'installation :

A	M	S	
0	0	0	arrêt
0	1	0	Mise en marche
0	0	1	marche
1	0	0	Mise à l'arrêt
1	1	??	Arrêt prioritaire 0 <u>ou</u> Marche prioritaire 1 ?

Nous remarquons que la valeur de sortie S est fonction des deux entrées M et A mais également de l'état précédent de la sortie S.

Pour résoudre un problème de ce type, il est nécessaire **d'utiliser une variable interne**, que l'on appelle < mémoire > ou **mémento**. Cette mémoire interne permet de mémoriser l'état de la sortie dans le cas qui nous occupe.

Pour mettre en évidence cette technique, on va ajouter une colonne dans la table de vérité, cette dernière va nous donner la valeur de la variable mémoire de la sortie (mémento) afin de suivre l'évolution de la sortie en fonction des appuis sur Marche et Arrêt.

Automation

Système séquentiel :

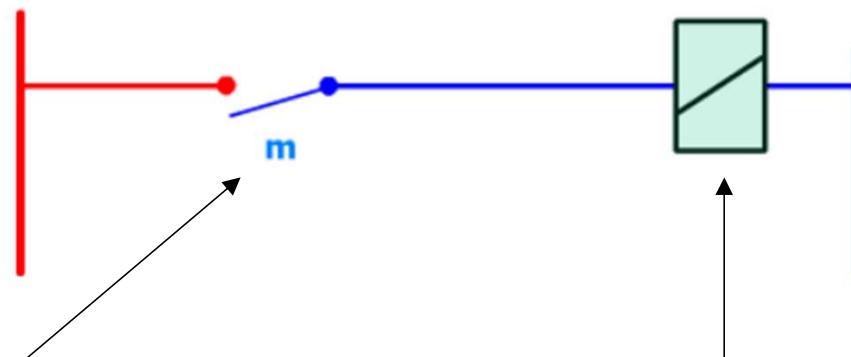
Nous allons décomposer le problème et voir comment mettre une méthode pour résoudre le problème, nous allons envisager les deux cas :

- Marche prioritaire,
- Arrêt prioritaire.

Le plus facile est d'utiliser une représentation électrique de notre système, en décomposant en étapes :

Marche du moteur :

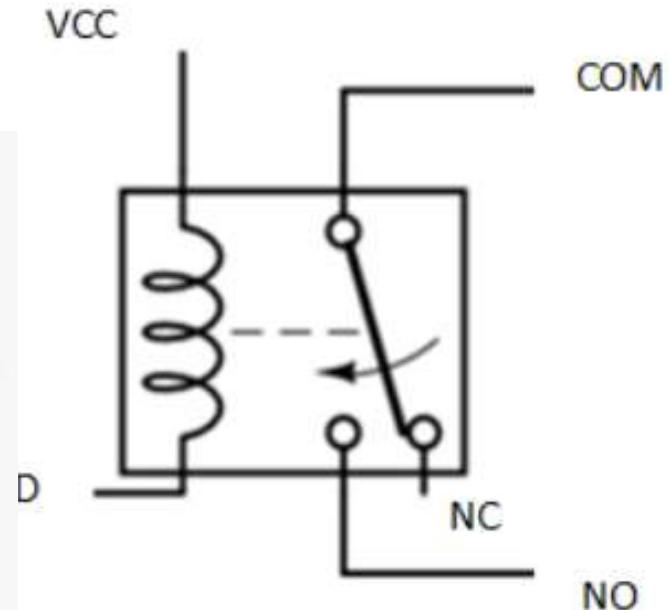
Enclenchement du moteur étape 1/4



La base : un **bouton marche** contrôle la **bobine du relais ou contacteur**

Automation

Système séquentiel : A quoi sert le contacteur ?



C'est l'élément qui va permettre de passer la puissance nécessaire au moteur afin de le faire fonctionner. Tout comme l'Arduino ou le Raspberry, l'automate est un élément de pilotage des installations utilisant des entrées / sorties à courant faible (de l'ordre du milliampère). Il est donc nécessaire de transmettre l'état de la sortie bas niveau vers un élément connecté au réseau de distribution (220V / 380V).

Ce rôle est réalisé par le contacteur de puissance.

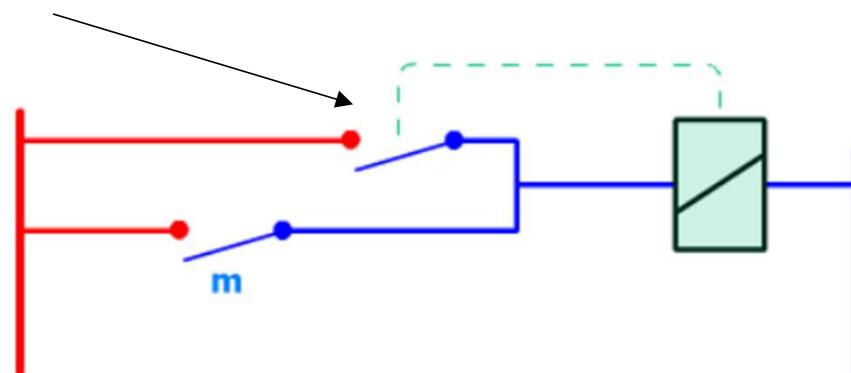
Automation

Système séquentiel :

Marche du moteur :

Enclenchement du moteur étape 2/4

Un **contact NO du relais ou contacteur** est installé en **parallèle** sur le bouton **marche**.



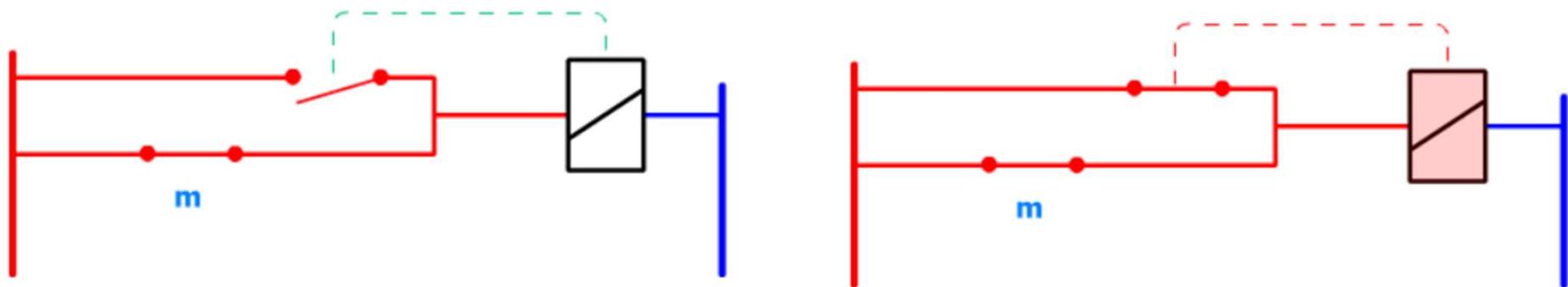
Lors de l'appuis sur le bouton marche, le contact NO va s'enclencher et « maintenir » l'alimentation du moteur.

Automation

Système séquentiel :

Marche du moteur :

Enclenchement du moteur étape 3/4



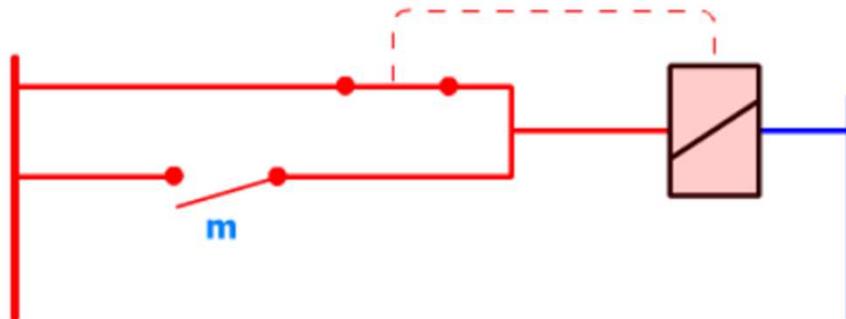
Lors de l'appuis sur le bouton marche, **le contact NO** est maintenu par l'alimentation du contacteur. Le moteur est donc maintenu en marche grâce au **contact NO**
Si on relâche le bouton marche, le moteur continue à tourner car il est alimenté par le réseau parallèle constitué par le contact NO.

Automation

Système séquentiel :

Marche du moteur :

Enclenchement du moteur étape 4/4



Lors du relâchement de l'appuis sur le bouton marche, **le contact NO** est maintenu par l'alimentation du contacteur.

Il reste à mettre en place une solution pour pouvoir mettre le moteur dans son état de repos (arrêt du moteur).

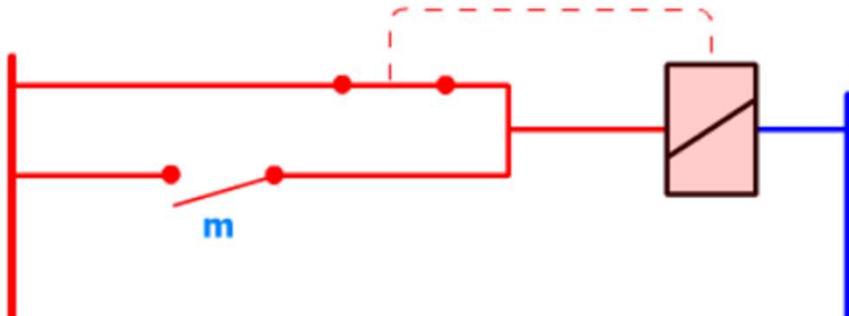
Nous pouvons maintenant réfléchir à la notion **d'arrêt prioritaire ou marche prioritaire**.

Automation

Système séquentiel :

Marche du moteur :

étape 4/4



Il nous faut raisonner sur ce schéma et ajouter dans chacun des cas, un élément qui permettra de couper notre contacteur qui alimente le moteur.

- Cas de l'arrêt prioritaire :

Il faut que l'élément que l'on ajoute coupe l'alimentation du contacteur, quelque soit l'appuis sur le bouton de marche.

Si nous rappelons de l'algèbre de Boole, et en particulier les portes logiques, le cas d'une porte « ET » donne systématiquement un état bas si une de ces entrées est à l'état bas (0). Il faut donc ajouter au circuit un bouton supplémentaire, en série avec ce circuit. Ce dernier devra vérifier la condition de l'appui simultané sur marche / arrêt en donnant priorité à l'arrêt.

Automation

Système séquentiel :

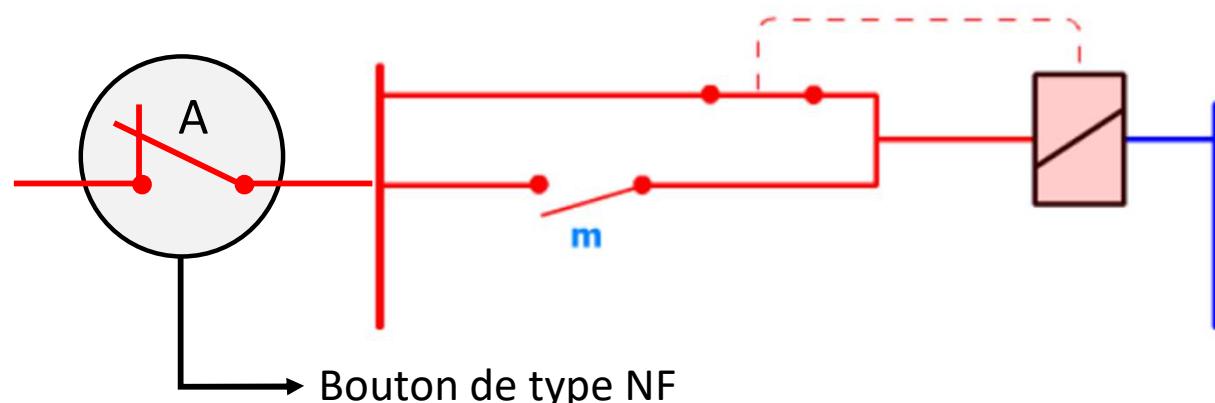
Arrêt prioritaire :

Il nous faut raisonner sur ce schéma et ajouter dans chacun des cas, un élément qui permettra de couper notre contacteur qui alimente le moteur.

Il faut que l'élément que l'on ajoute coupe l'alimentation du contacteur, quelque soit l'appui sur le bouton de marche (m).

Si nous nous rappelons de l'algèbre de Boole, et en particulier les portes logiques, le cas d'une **porte « ET »** donne systématiquement un état bas si une de ces entrées est à l'état bas (0). Il faut donc ajouter au circuit un bouton supplémentaire, **en série** avec ce circuit. Ce dernier devra vérifier la condition de l'appui simultané sur marche / arrêt en donnant priorité à l'arrêt.

Schéma :



Automation

Système séquentiel :

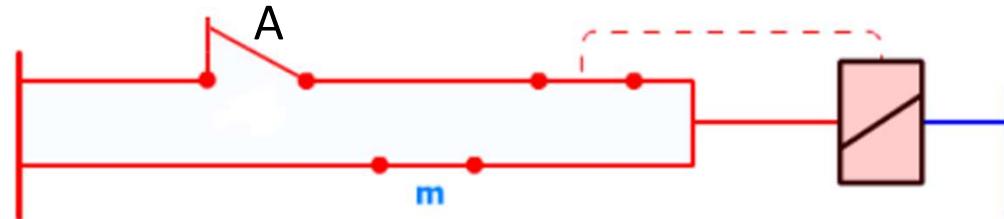
Marche prioritaire :

Il nous faut raisonner sur ce schéma et ajouter dans chacun des cas, un élément qui permettra de démarrer notre contacteur qui alimente le moteur.

Il faut que l'élément que l'on ajoute donne l'alimentation au contacteur lorsque l'on effectue un appui simultané sur le bouton marche et le bouton arrêt.

Ici le cas d'une **porte « OU »** donne systématiquement un état haut si une de ces entrées est à l'état haut (1). Il faut donc ajouter au circuit un bouton supplémentaire, **en parallèle** avec ce circuit. Ce dernier devra vérifier la condition de l'appui simultané sur marche / arrêt en donnant priorité à la marche du moteur.

Schéma :



Automation

Système séquentiel :

Cycle d'un automate :

Un automate est une machine séquentielle, elle effectue un cycle composé des opérations suivantes :

- **Acquisition des valeurs d'entrées** (Digitales / Analogiques)
- Ecriture des valeurs dans la **MIE** (Mémoire Image des Entrées)
- **Exécution du cycle de l'automate** (c'est la partie où l'automate exécute son programme en utilisant la MIE et en écrivant le résultat des sorties dans la MIS : Mémoire Image des Sorties)
- **Mise à jour de la table des sorties MIS.**

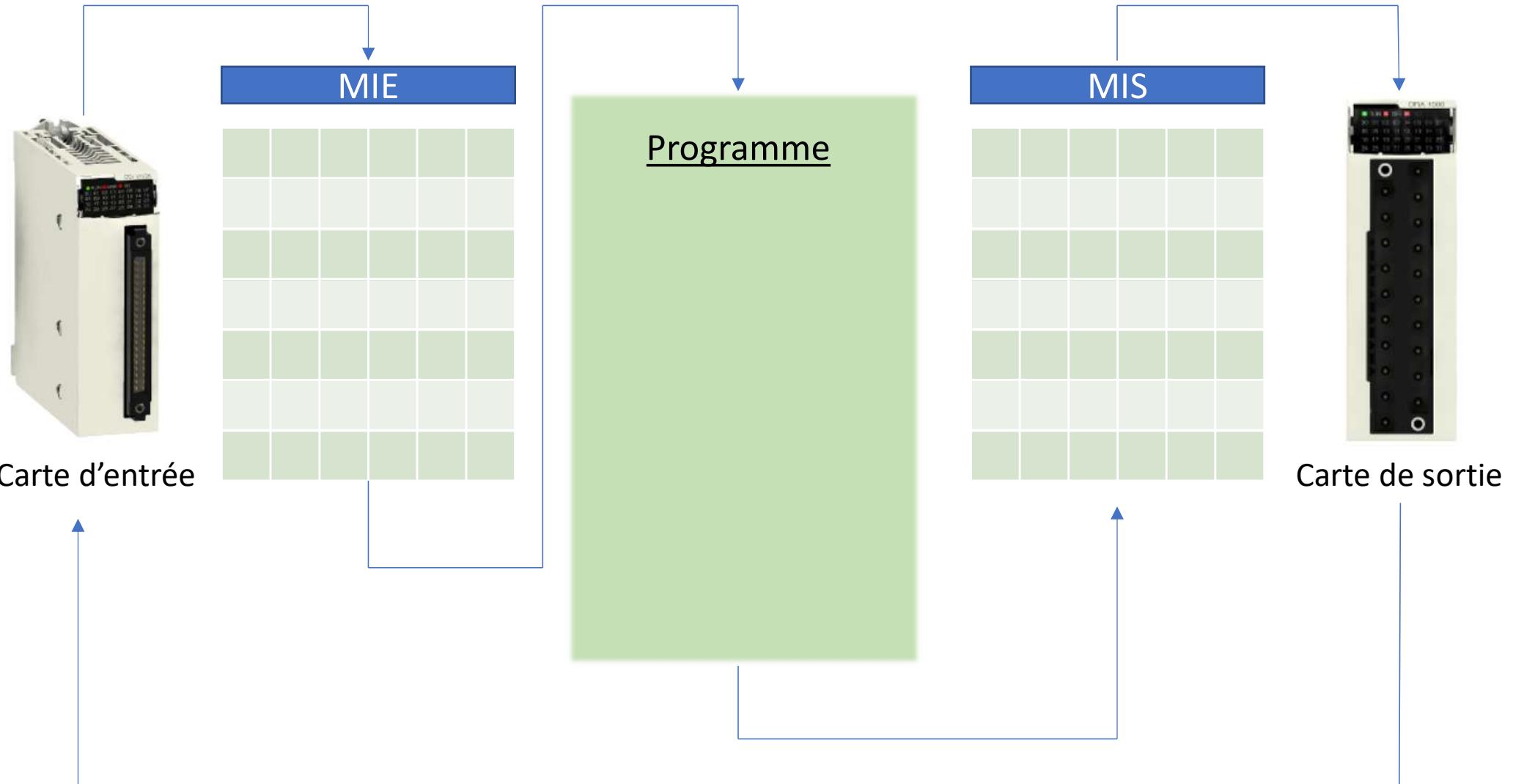
Une fois le cycle terminé, l'automate recommence en boucle la séquence des opérations décrites précédemment.

Il va également contrôler le **temps d'exécution** de son **cycle**, et le surveiller en « durée ». L'utilisateur programme un temps de surveillance (**Watch Dog**). Si le cycle dépasse ce temps, l'automate passe en sécurité et est à l'état STOP.

Le programme n'est plus exécuté. Il est donc essentiel d'avoir un programme qui tourne dans le délais fixé par le chien de garde.

Automation

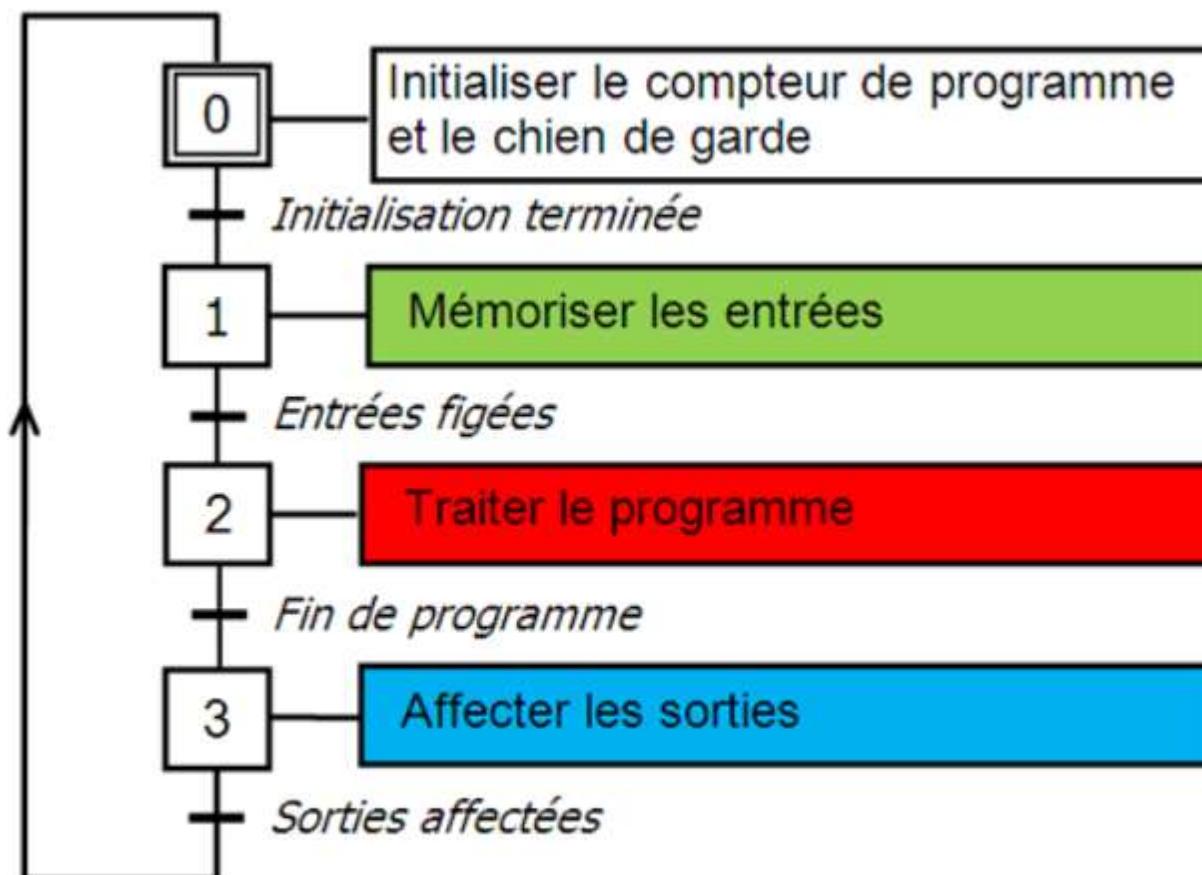
Système séquentiel : Cycle d'un automate : schéma



Automation

Système séquentiel :

Cycle d'un automate : schéma



Fonctionnement cyclique d'un API

Automation

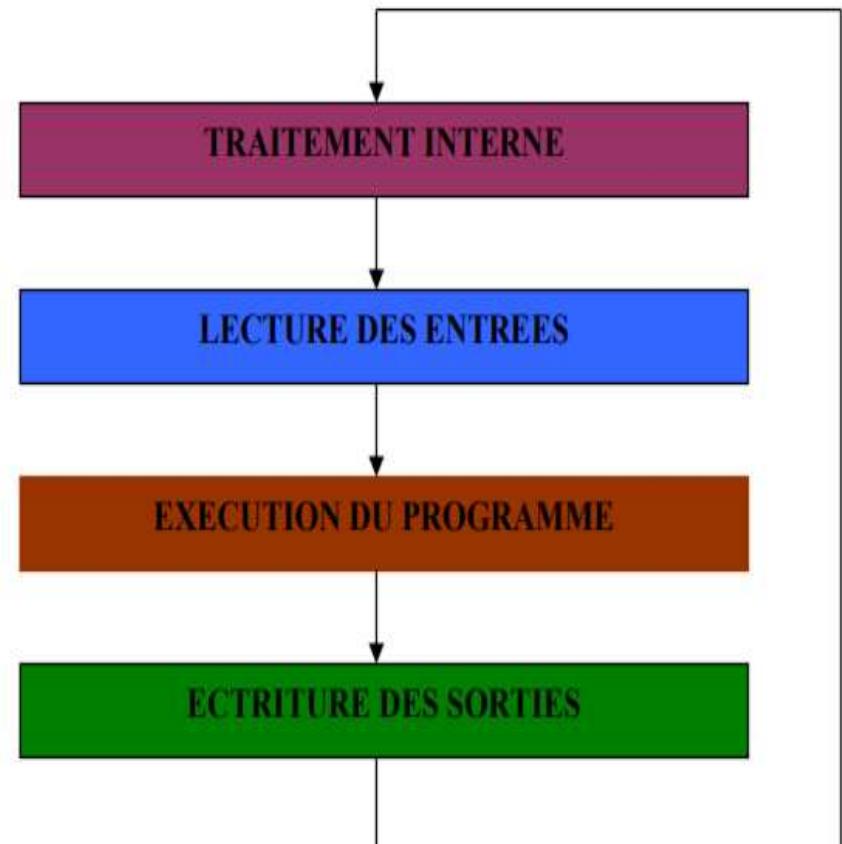
Le cycle d'un point de vue temporel:

Traitement interne : L'automate effectue des opérations de contrôle et de mise à jour de paramètres systèmes (bits systèmes). Ce traitement permet de gérer tous les défauts de fonctionnements de l'automate et de réagir en conséquence.

Lecture des entrées : ($\pm 100 \mu s$ par module d'entrées) L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées en RAM.

Exécution du programme : L'automate exécute séquentiellement (instruction par instruction) et écrit les sorties dans la mémoire image de sorties en RAM.

Ecriture des sorties : ($\pm 100 \mu s$ par module de sorties) L'automate recopie (de façon synchrone) la mémoire image de sorties vers les sorties réelles. Ce mode opératoire est effectué de façon cyclique par l'automate.



Automation

Système séquentiel :

Arrêt prioritaire :

Schéma :

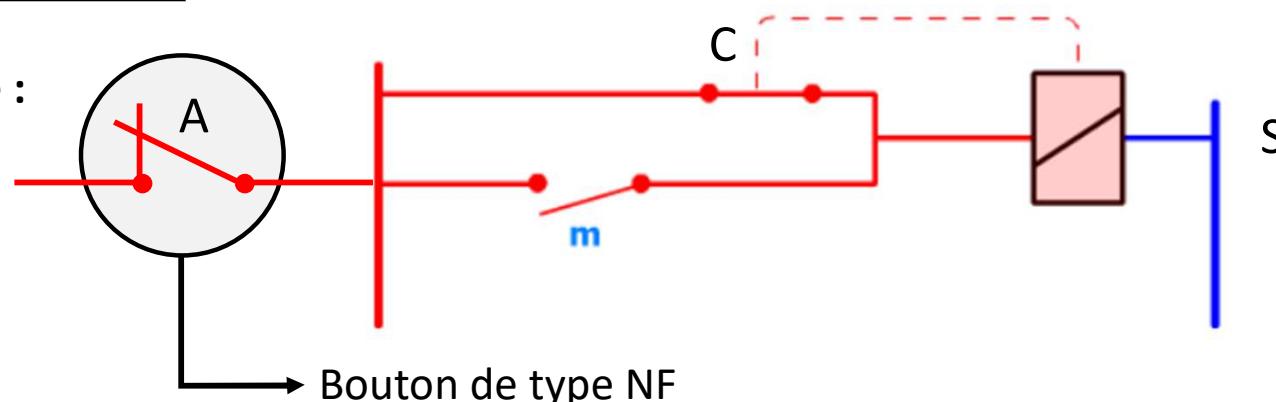
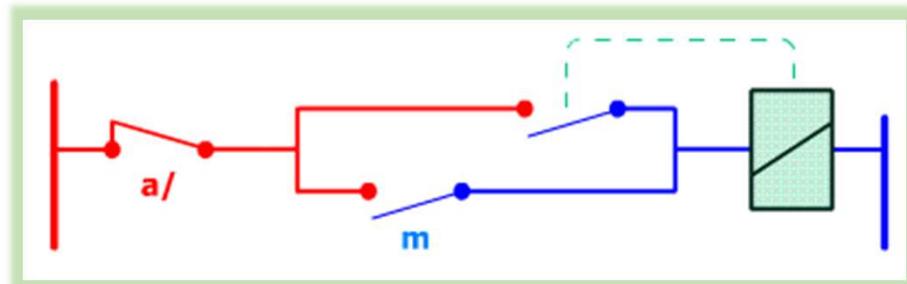


Table de vérité :

m	A	C	S
0	0	0	0
1	0	0	1
1	0	1	1
0	0	1	1
0	1	1	0
0	1	0	0
0	0	0	0
1	1	0	0

Équivalent : porte ET



Mémorisation Marche

Mémorisation Arrêt
Arrêt prioritaire

Automation

Système séquentiel :

Marche prioritaire :

Schéma :

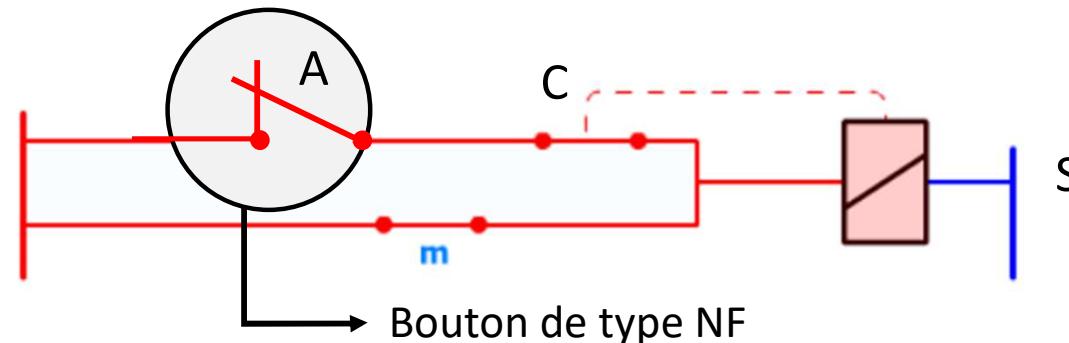
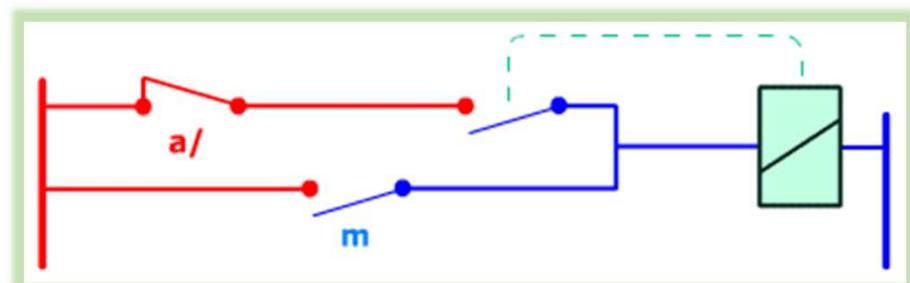


Table de vérité :

m	A	C	S
0	0	0	0
1	0	0	1
1	0	1	1
0	0	1	1
0	1	1	0
0	1	0	0
0	0	0	0
1	1	1	1

Équivalent : porte OU



Mémorisation Marche

Mémorisation Arrêt

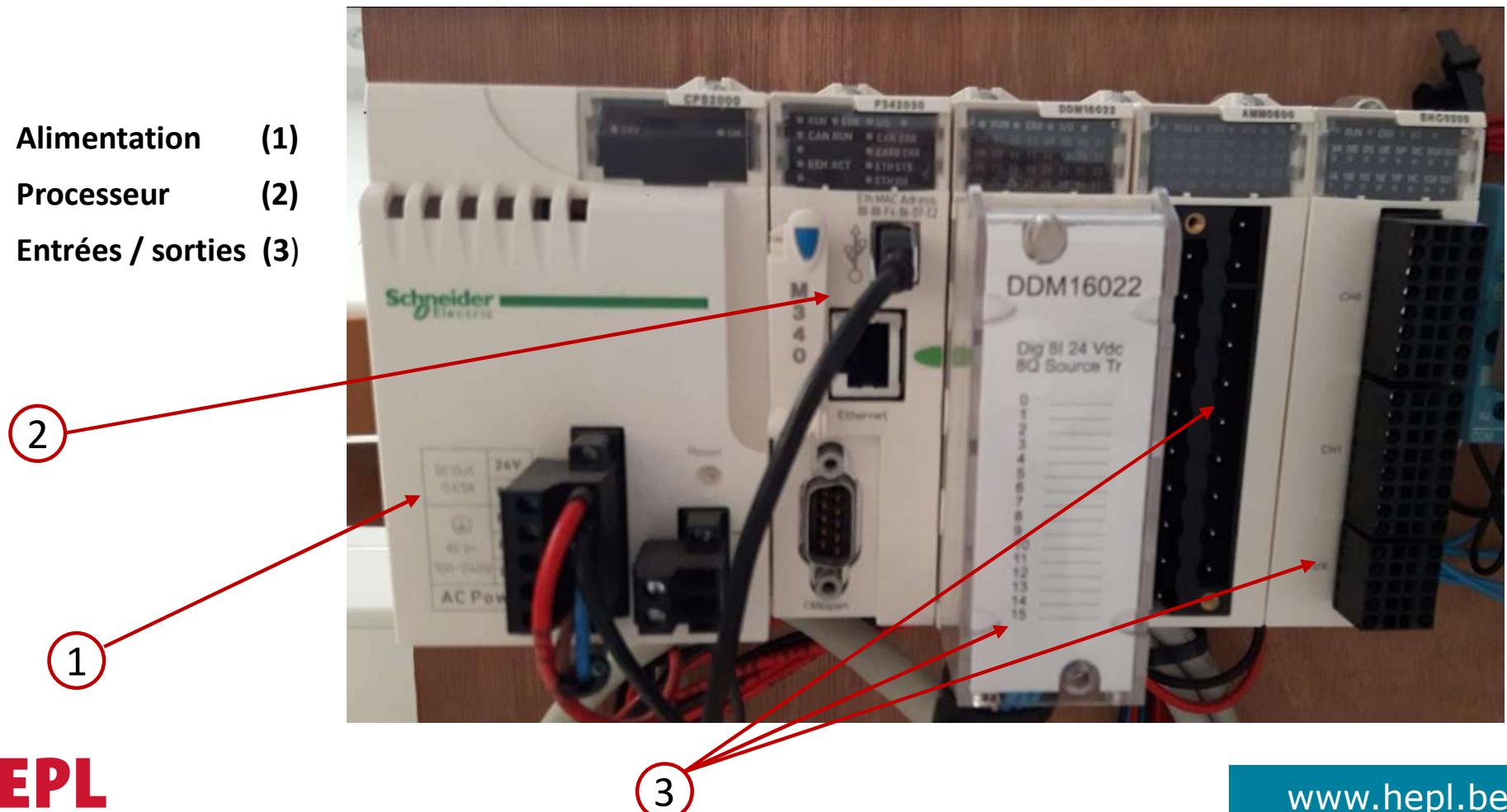
Marche prioritaire

Automation

L'automate du laboratoire M340 :

Il est équipé d'une alimentation, d'un processeur (unité centrale) et de cartes d'extensions (entrées / sorties, de type digitale ou analogique)

- Alimentation (1)
- Processeur (2)
- Entrées / sorties (3)

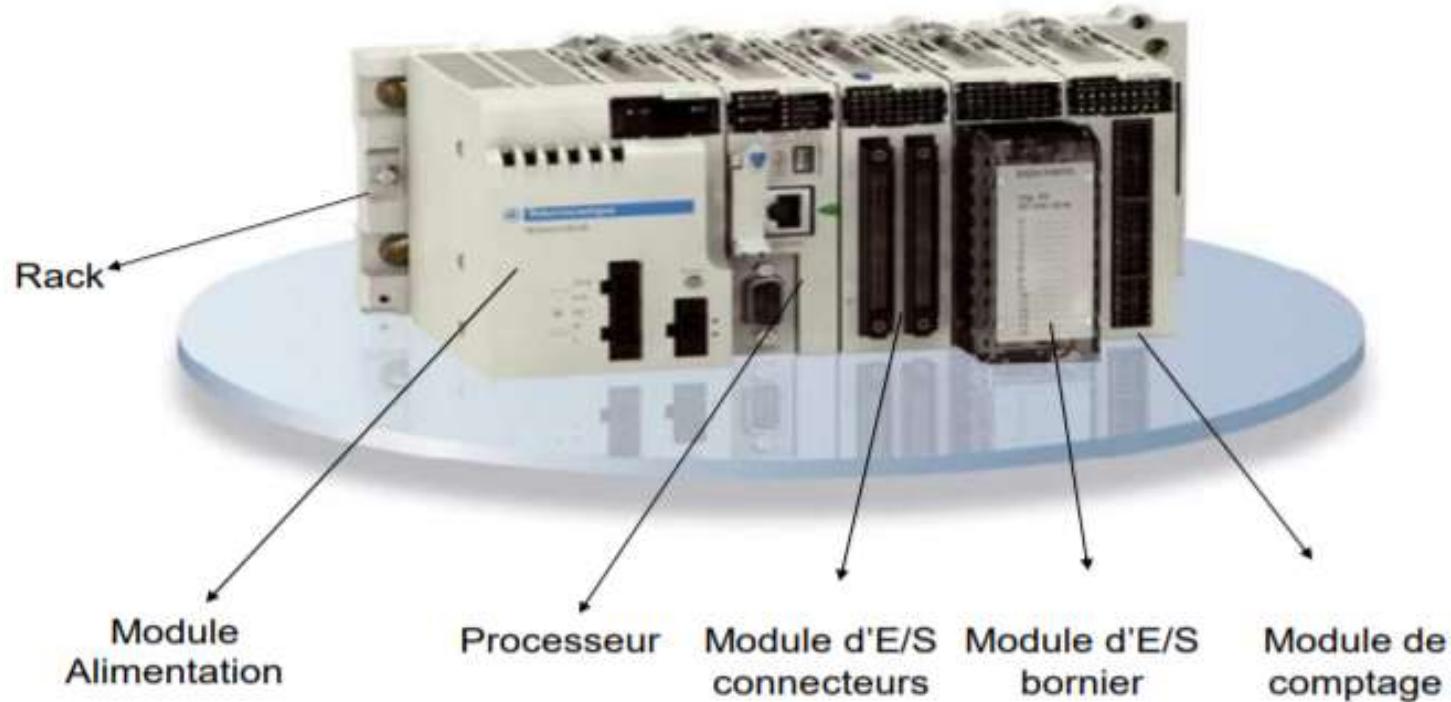


Automation

L'automate du laboratoire M430 :

Il est équipé d'une alimentation, d'un processeur (unité centrale) et de cartes d'extensions (entrées / sorties, de type digitale ou analogique)

La plate-forme M340



Automation

L'automate du laboratoire M430 :

Affichage sur le processeur :



- **RUN :**

- Allumé : Automate en marche.
- Clignotant : Automate en mode stop ou erreur logiciel.
- Eteint : Automate non configuré.

- **I/O allumé :**

- Erreur de configuration ou erreur sur le module I/O.

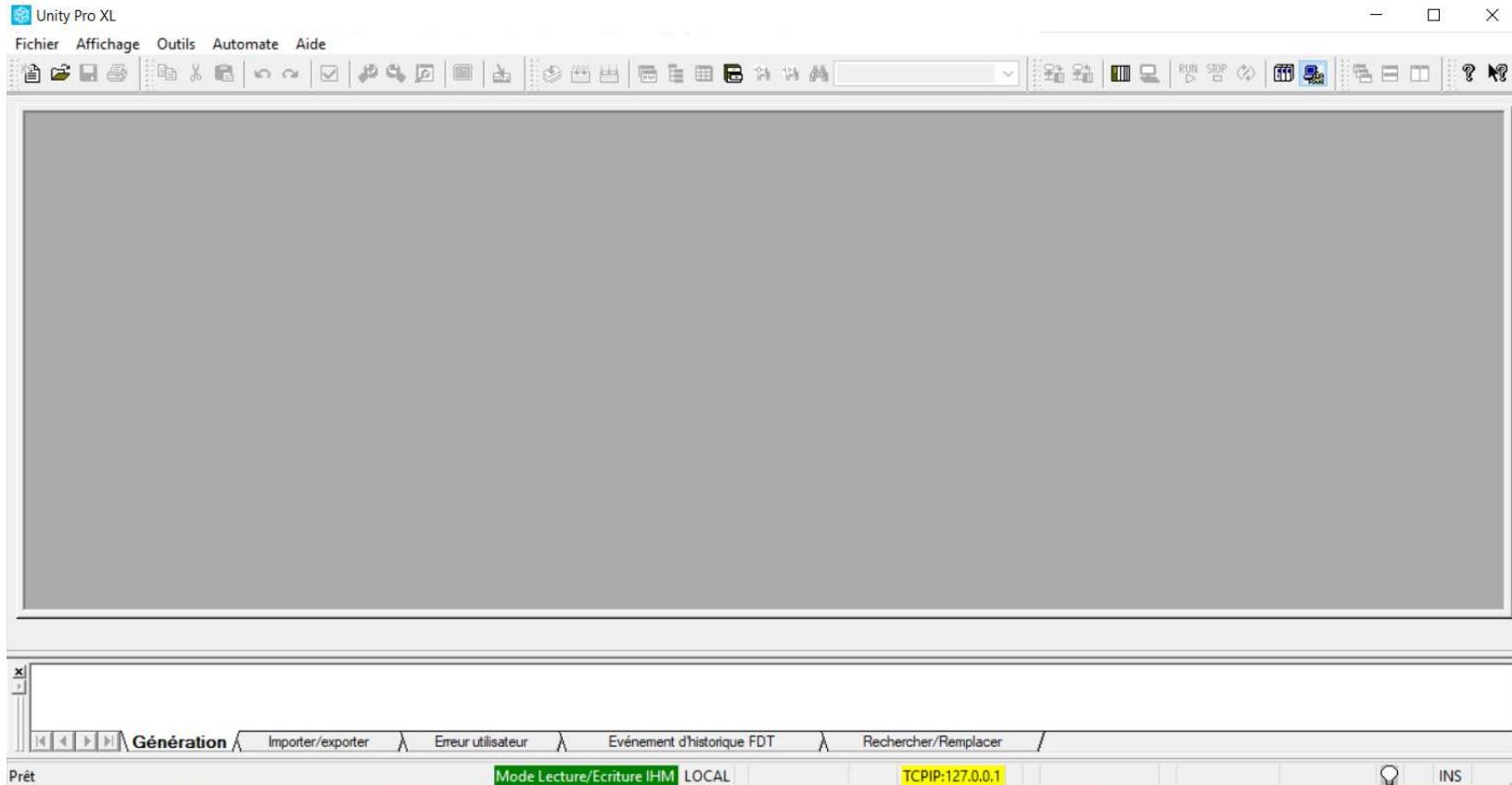
- **ERR allumé :**

- Erreur processeur ou système.

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL



L'interface permet la création et la simulation d'un programme pour notre automate avec partie contrôle / commande et supervision d'installation.

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL

L'interface permet la création et la simulation d'un programme pour notre automate avec partie contrôle / commande et supervision d'installation. Il arrive parfois qu'après installation, il ne soit pas possible de faire une connexion vers l'automate en USB.

Dans ce cas il est nécessaire d'installer la mise à jour du driver pour Unity Pro XL concernant la partie « Driver USB ».

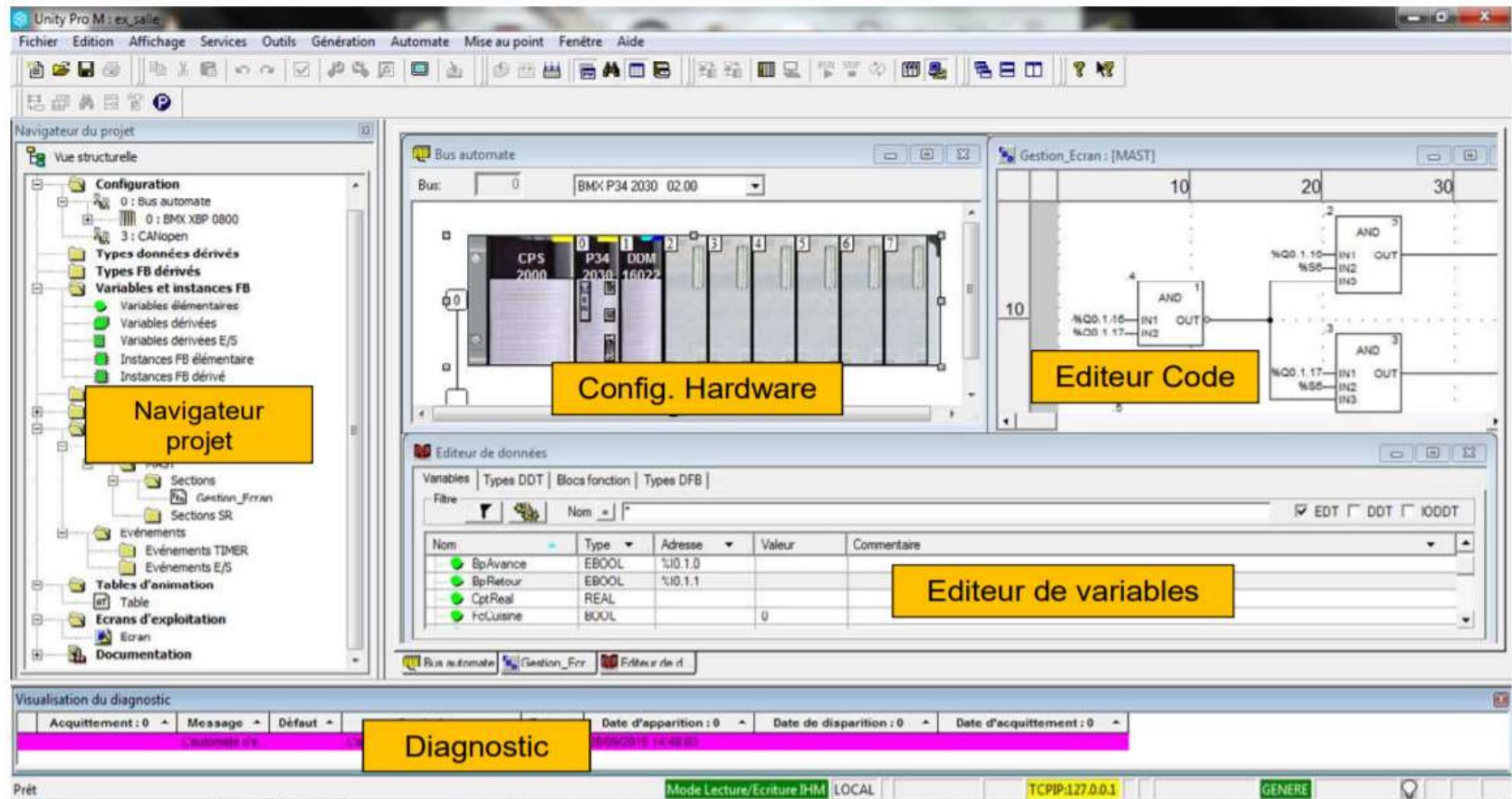
Ce driver peut être trouver sur le site du groupe Schneider Electric, en faisant une recherche sur :

<https://www.se.com/id/en/download/document/Schneider+PLCUSBDriverSuite/>

The screenshot shows a product page for Schneider Electric's PLCUSBDriverSuite. At the top, there's a navigation bar with links for Support, Product Documentation and Software downloads, and the specific product page for Schneider PLCUSBDriverSuite. The main title is "Schneider PLCUSBDriverSuite". Below it is a large thumbnail image of a document icon. To the right, there are several product details listed: Reference (Schneider PLCUSBDriverSuite), Type (Software - Hotfix & Patch), Languages (English/ French), Date (26/02/2015), and Version (V1.0). Further down, under "Product Ranges:", there's a link to "Easy Harmony GXU". Under "Files for Download", there's a file entry for "SchneiderPLCUSBDriverSuite (.exe)" with a download icon.

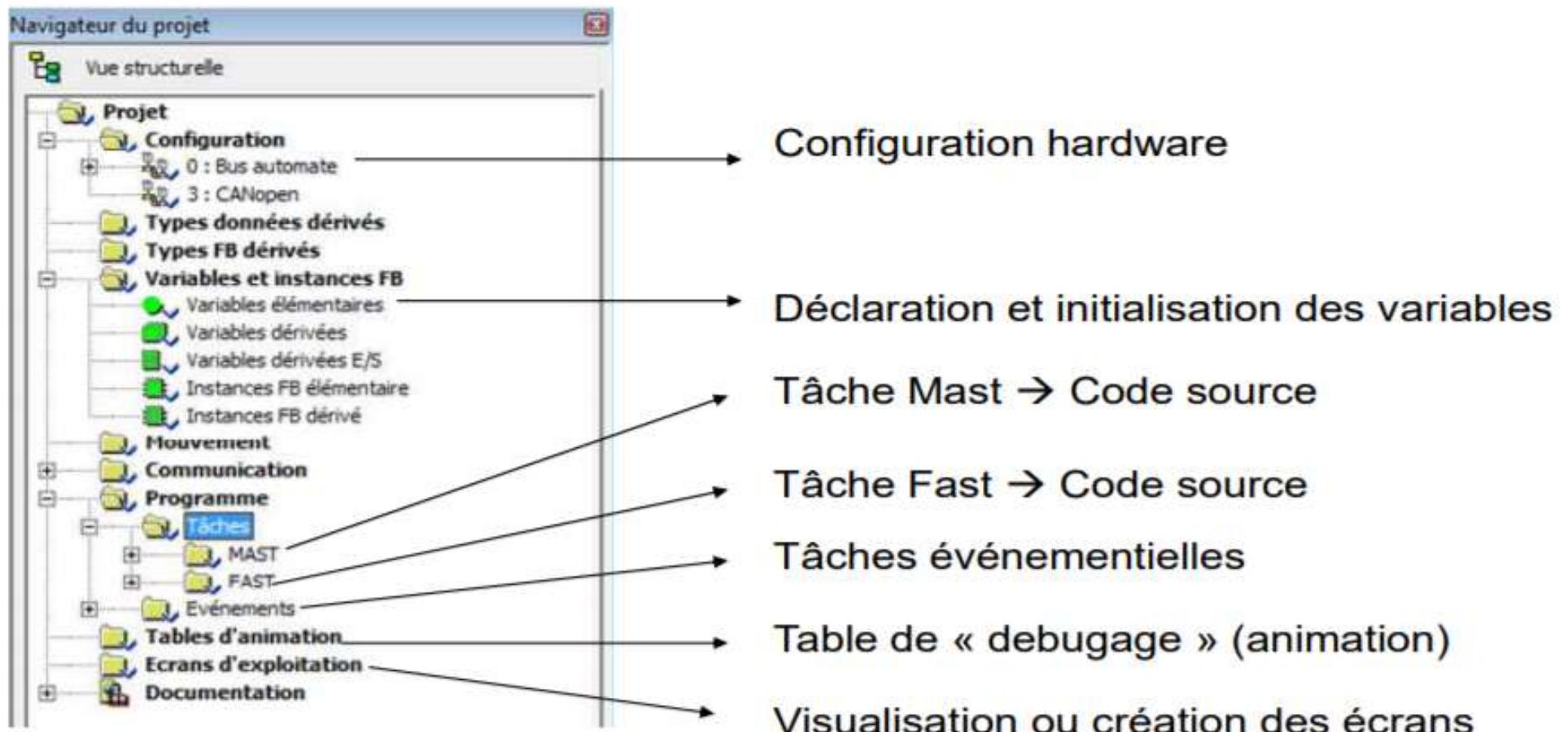
Automation

L'interface Utilisateur : Unity Pro XL



Automation

L'interface Utilisateur : Le navigateur de projet



Automation

L'interface Utilisateur : Réaliser une configuration matérielle :

Pour la **configuration matérielle**, il faudra utiliser les menus de configurations adéquats. Cette configuration va dépendre de la partie rack, alimentation, type de processeur et également cartes d'entrée / sortie. Si le matériel dépasse un rack, il faudra également ajouter un coupleur et un deuxième rack avec le reste du matériel.

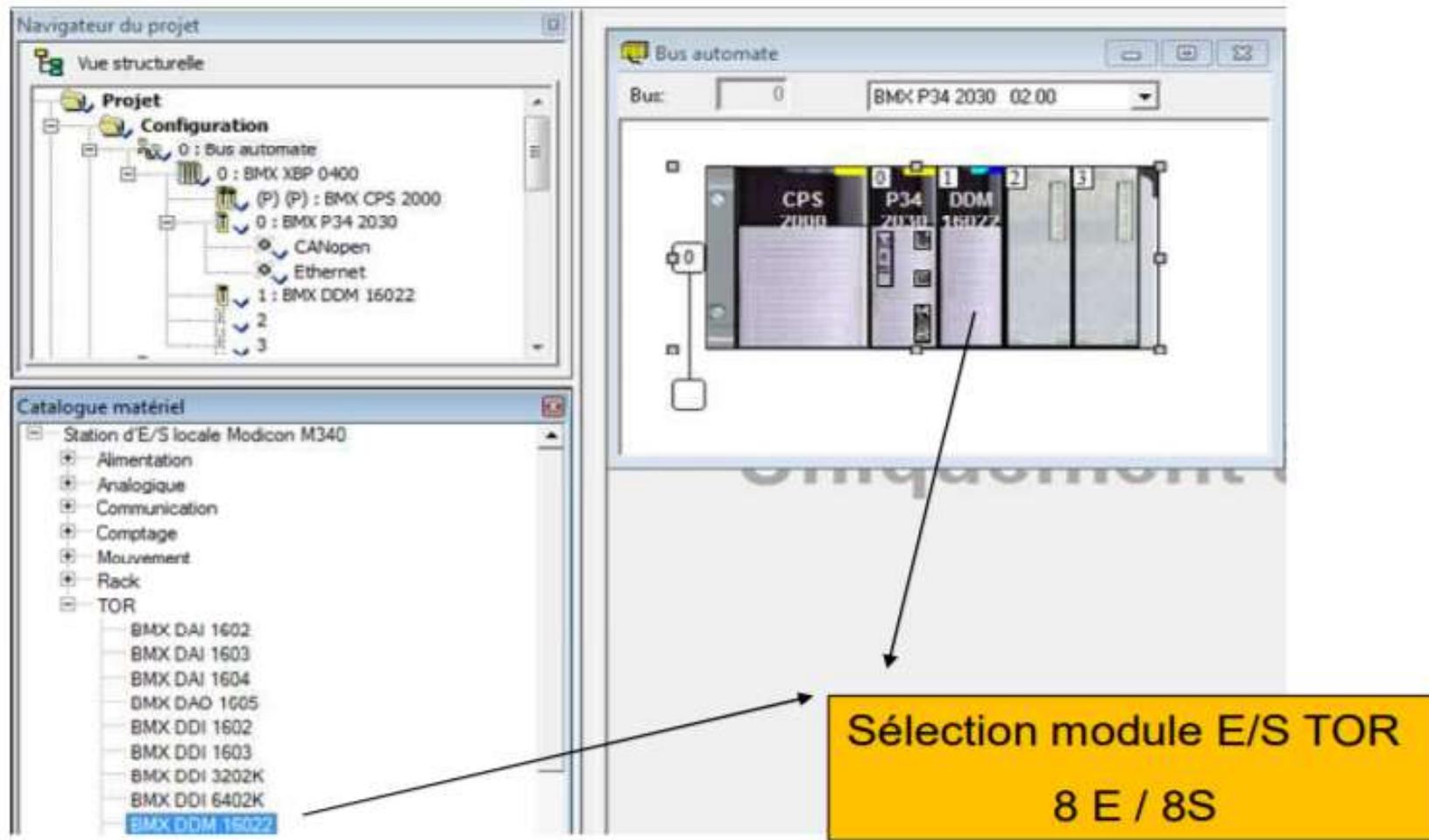
On aura :

- Configuration du rack local.
- Configuration de l'alimentation.
- Configuration de l'automate (unité centrale, par exemple M340).
- Configuration des modules E/S, des modules « métiers ».
- Configuration coupleur. (cartes coupleur pour l'extension de rack).
- Configuration de carte de communication (pour réseau autre que TCP-IP, comme par exemple une communication Profibus, ASI ou autre).
- ...

Une fois cette partie terminée, on va pouvoir s'occuper des variables programme.

Automation

L'interface Utilisateur : Exemple de configuration matérielle :



Automation

L'interface Utilisateur : Configuration des variables du processus :

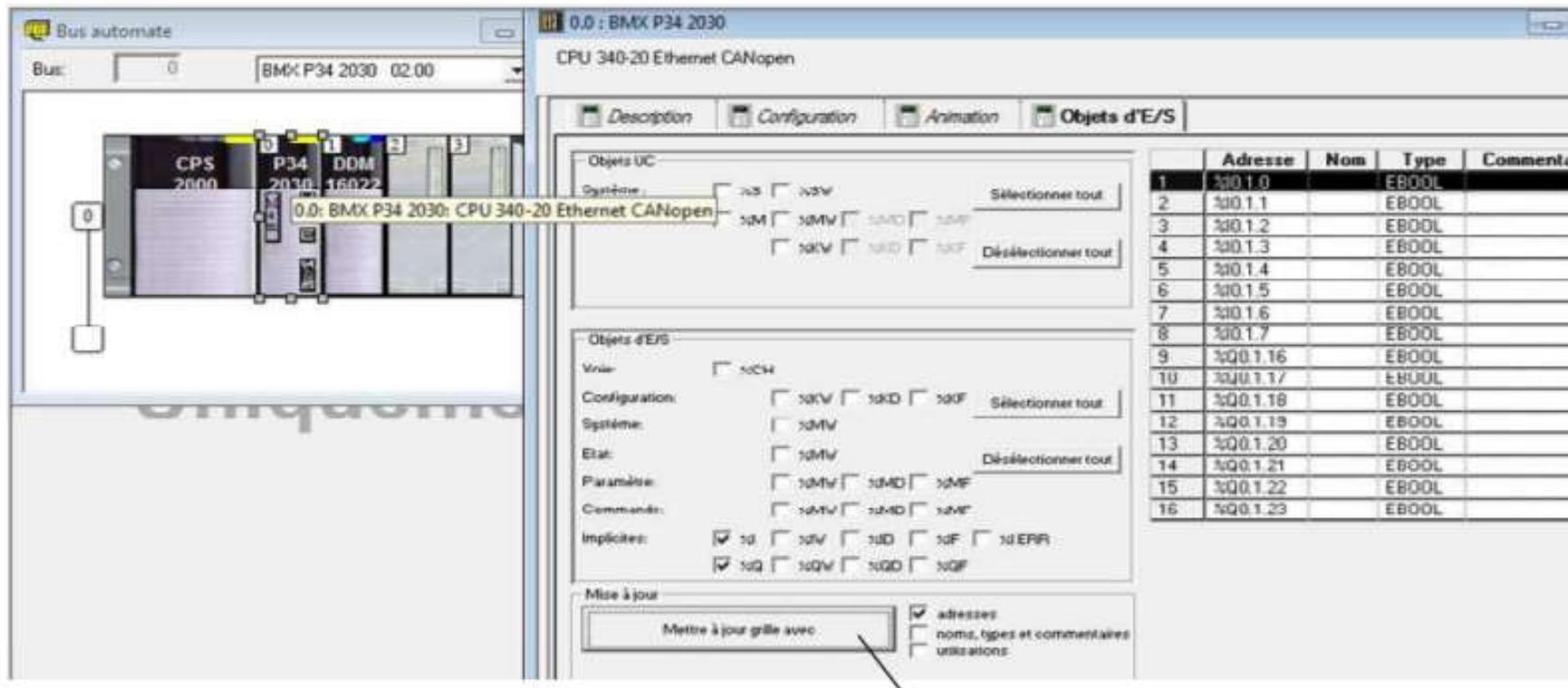
Pour pouvoir choisir la configuration matérielle, il va être nécessaire de faire l'analyse du cahier des charges et analyse fonctionnelle.

Ceci aura deux buts précis :

- Dresser la configuration des variables nécessaires au process, cette liste comprend la liste des entrées / sorties, et également des différentes variables internes.
- Définir la liste des cartes d'entrée / sortie en fonction de l'analyse du point précédent.
- Choisir, si l'automate fait partie d'un réseau, les cartes de communication.
- Voir si il est nécessaire de prévoir un coupleur (extension du rack).
- Prévoir, si nécessaire, la partie carte lié à la sécurité (ASI Bus et cartes dédiées).
- Réfléchir aux conditions d'utilisation du matériel (situation : milieu humide, température basse ou haute, poussières, ...).
- Tenir compte de la vitesse du processus, cela va impacter le choix de l'unité centrale.
- Choix d'une carte dédiée comme par exemple une carte de pesage.
- ...

Automation

L'interface Utilisateur : Configuration des variables I/O :

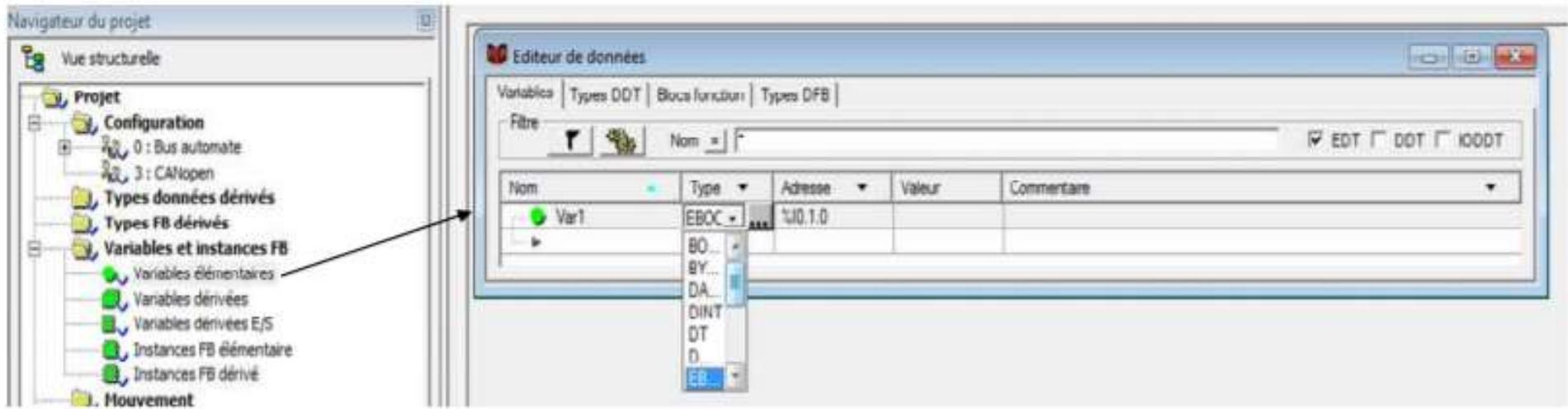


Click affichage E/S

On va définir dans cette partie, la liste des variables automate qui est directement reliée en entrée ou en sortie à une des cartes de la configuration matérielle sur la configuration matérielle de notre automate.

Automation

L'interface Utilisateur : Configuration des variables internes :

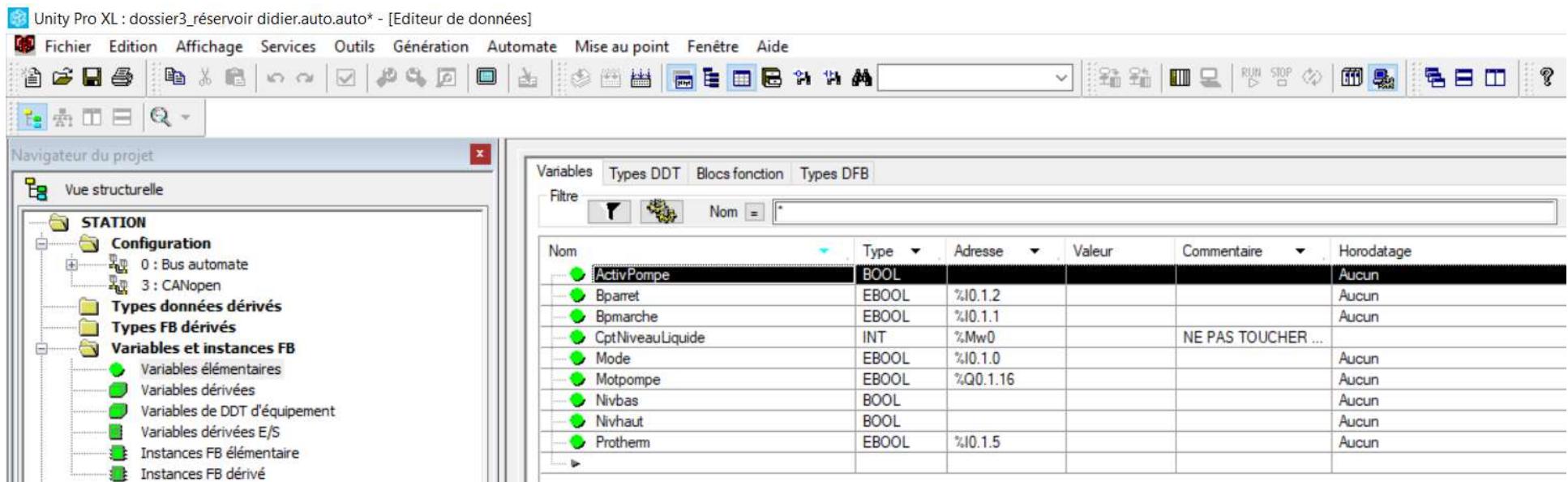


Dans cette partie, nous allons choisir le nom de variable que l'on va utiliser dans la rédaction du programme automate, en plus du nom de variable, c'est dans cette partie que le type de la variable sera choisi (comme on choisirait INT, REAL ou autre dans un langage de haut niveau). Nous trouverons différents types :

- EBOOL
- BOOL
- INT
- Byte
- Word
- dWord
- structure personnelle, ...

Automation

L'interface Utilisateur : Configuration des variables internes : exemple.



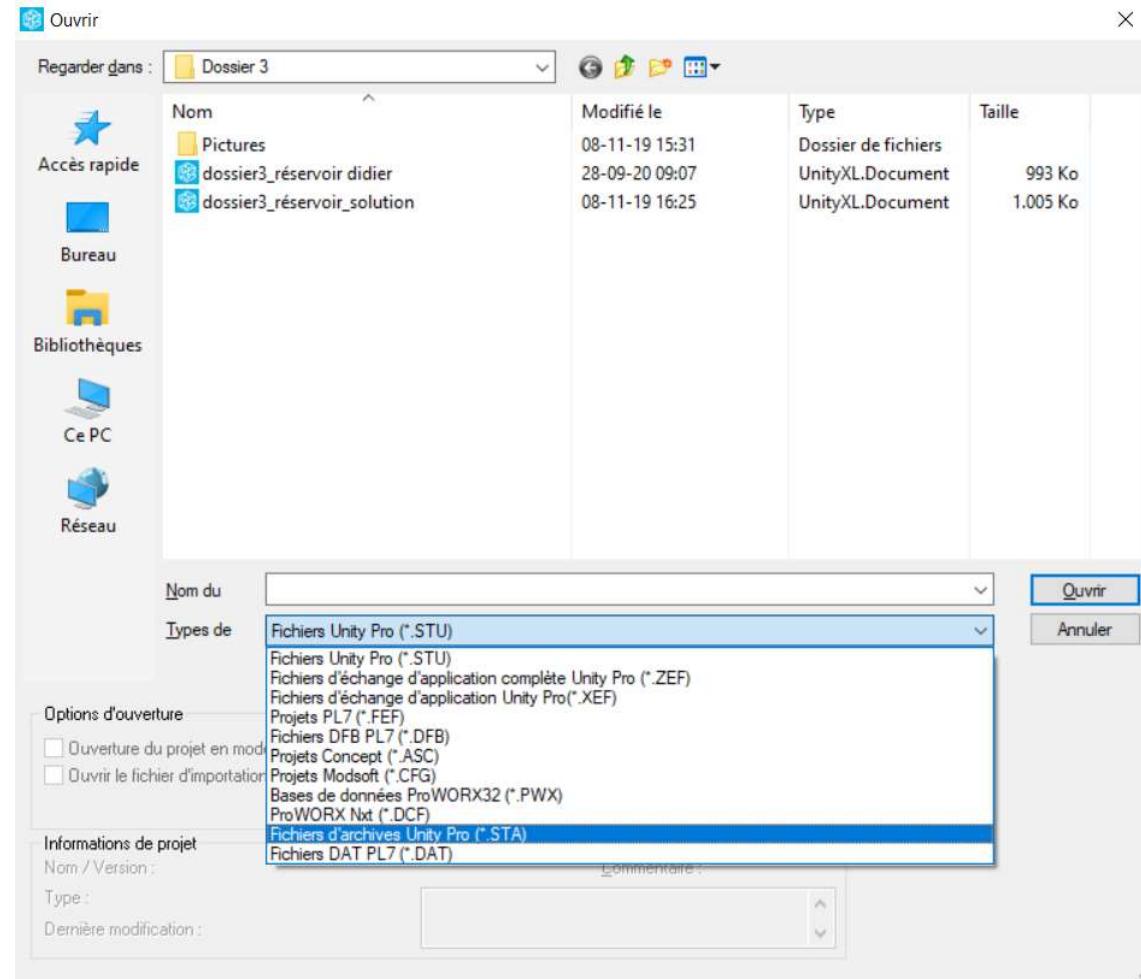
Il s'agit des variables définies pour un processus de pompage, on va utiliser :

- deux niveaux (bas, haut)
- une protection thermique sur la pompe
- un mode marche manuel ou automatique.
- Un bouton marche
- Un bouton arrêt
- Une variable interne pour le niveau de liquide

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL, ouverture d'un projet



Utiliser le menu Fichier, « Ouvrir » et choisir le fichier avec l'extension « .STA ».

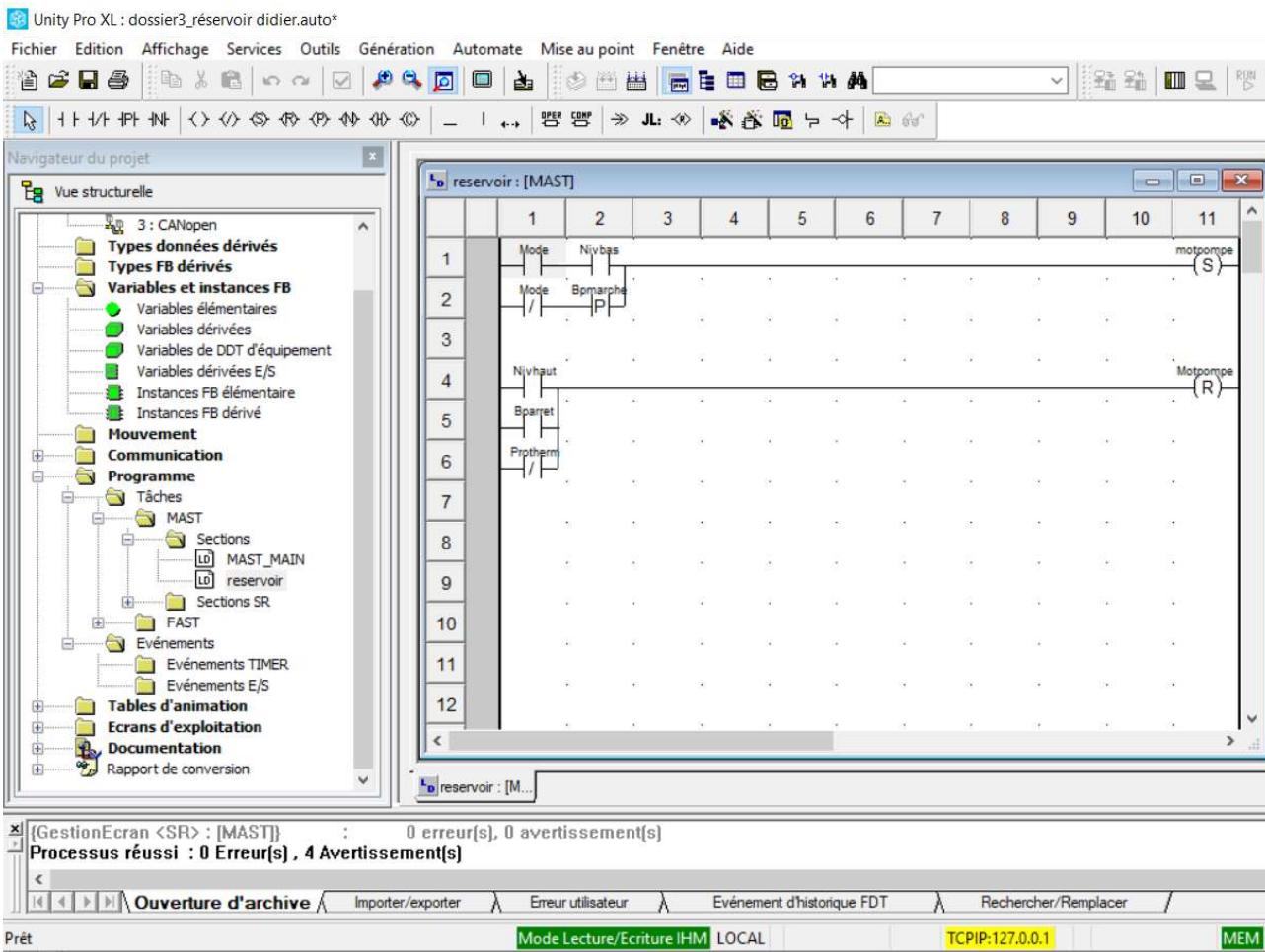
Cette opération ouvre la dernière sauvegarde de votre projet et va vous permettre de travailler dans l'interface.

- Ecriture de programme
- Création de table de variables
- Utilisation de table d'animation
- Ouverture d'écran de supervision
- Forçage de variables
- Utilisation du mode « simulation »
- ...

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL : Exemple Tache MAST



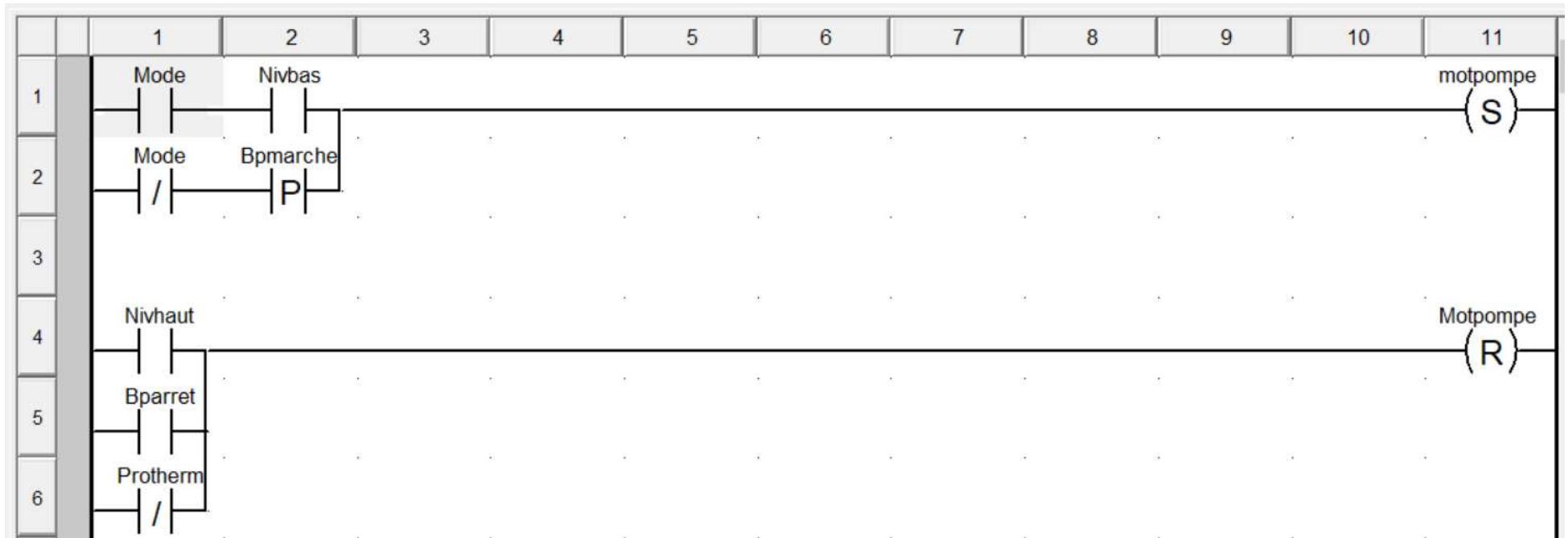
On trouve sur l'exemple, une programmation d'un remplissage d'une cuve avec contrôle de niveau minimum et remplissage automatique ou manuel.

Il est possible d'associer à ce programme ça partie supervision. Afin de voir le fonctionnement de l'installation et les différents défauts.

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL : Exemple Tache MAST



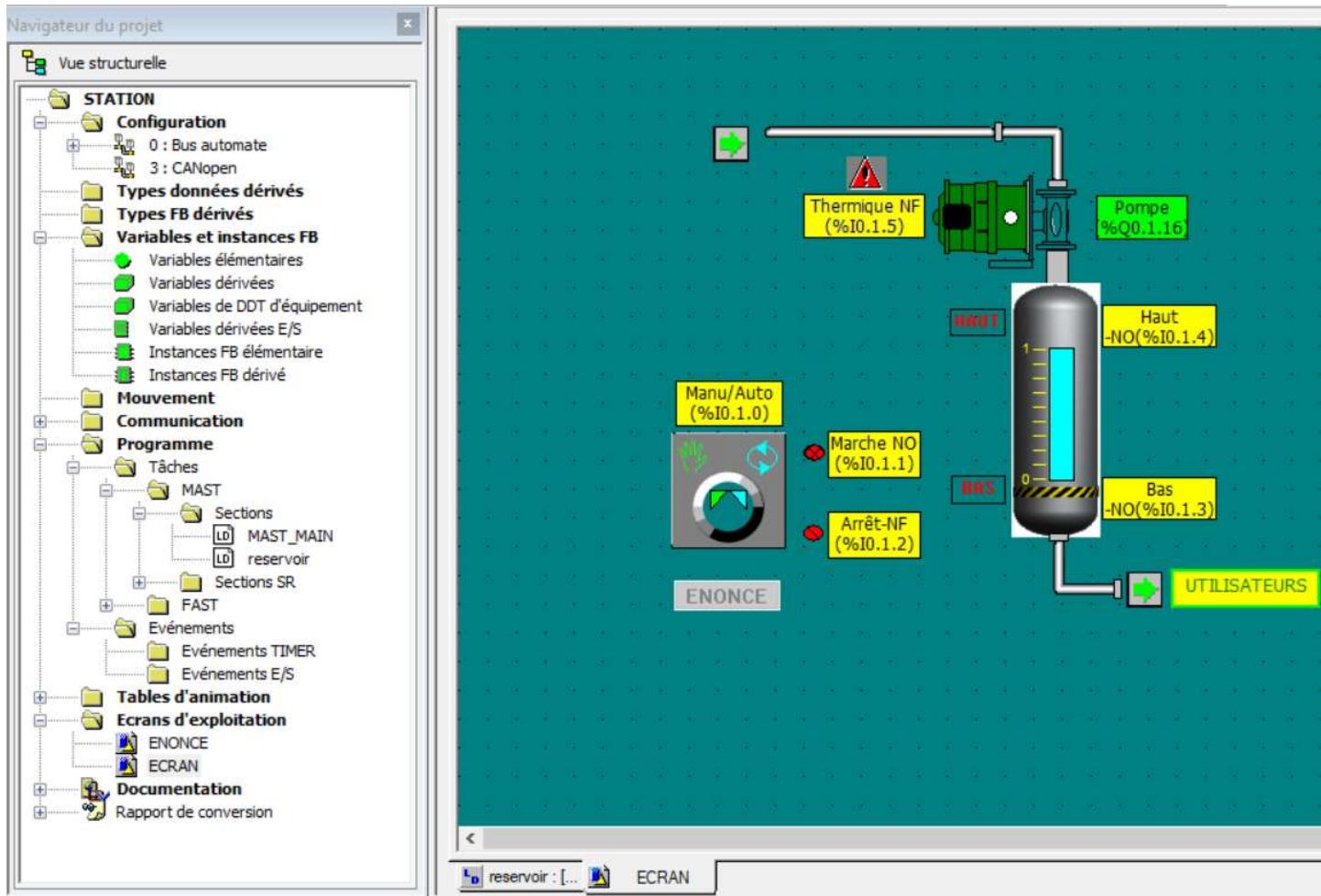
Ici la programmation est faite en langage « Contact » ou Ladder (Siemens). On utilise la liste des symboles pour mettre en équation le système à piloter.

Automation

Les outils pour la programmation du M340 :



L'interface Utilisateur : Unity Pro XL : Exemple de Supervision



Cette partie représente la simulation de la partie contrôle / commande sous forme d'écran de supervision. Cette vue est animée en temps réel par l'automate et est le reflet

Automation

Les outils pour la programmation du M340 :

L'interface Utilisateur : Unity Pro XL : Table d'animation

A screenshot of the Unity Pro XL software interface. On the left, the "Navigateur du projet" (Project Navigator) shows a tree view of project components under "Vue structurelle": STATION (Configuration, Types données dérivés, Types FB dérivés, Variables et instances FB, Mouvement, Communication, Programme, Tables d'animation), and MAST, FAST, Evénements, and Evénements E/S under Tâches. On the right, a "Table" window is open, showing a grid with columns "Nom", "Valeur", "Type", and "Commentaire". The table contains four entries: Bpmarche (EBOOL), Bparet (EBOOL), Mode (EBOOL), and prothem (EBOOL).

Nom	Valeur	Type	Commentaire
Bpmarche		EBOOL	
Bparet		EBOOL	
Mode		EBOOL	
prothem		EBOOL	

Cette partie du programme permet de forcer l'état d'une variable pour simuler le processus que l'on veut mettre en place. Une fois l'automate en « RUN » il est possible de forcer à l'état 1 ou à l'état 0 une ou plusieurs variables.

Automation

Mise en mode simulation du programme :

Menu Automate :

Dans ce menu se trouve les sous-menu suivants :

- Connexion
- Définir l'adresse
- Mode standard
- Mode Simulation
- Passage en mode run lors de la demande d'envoi du programme
- ...

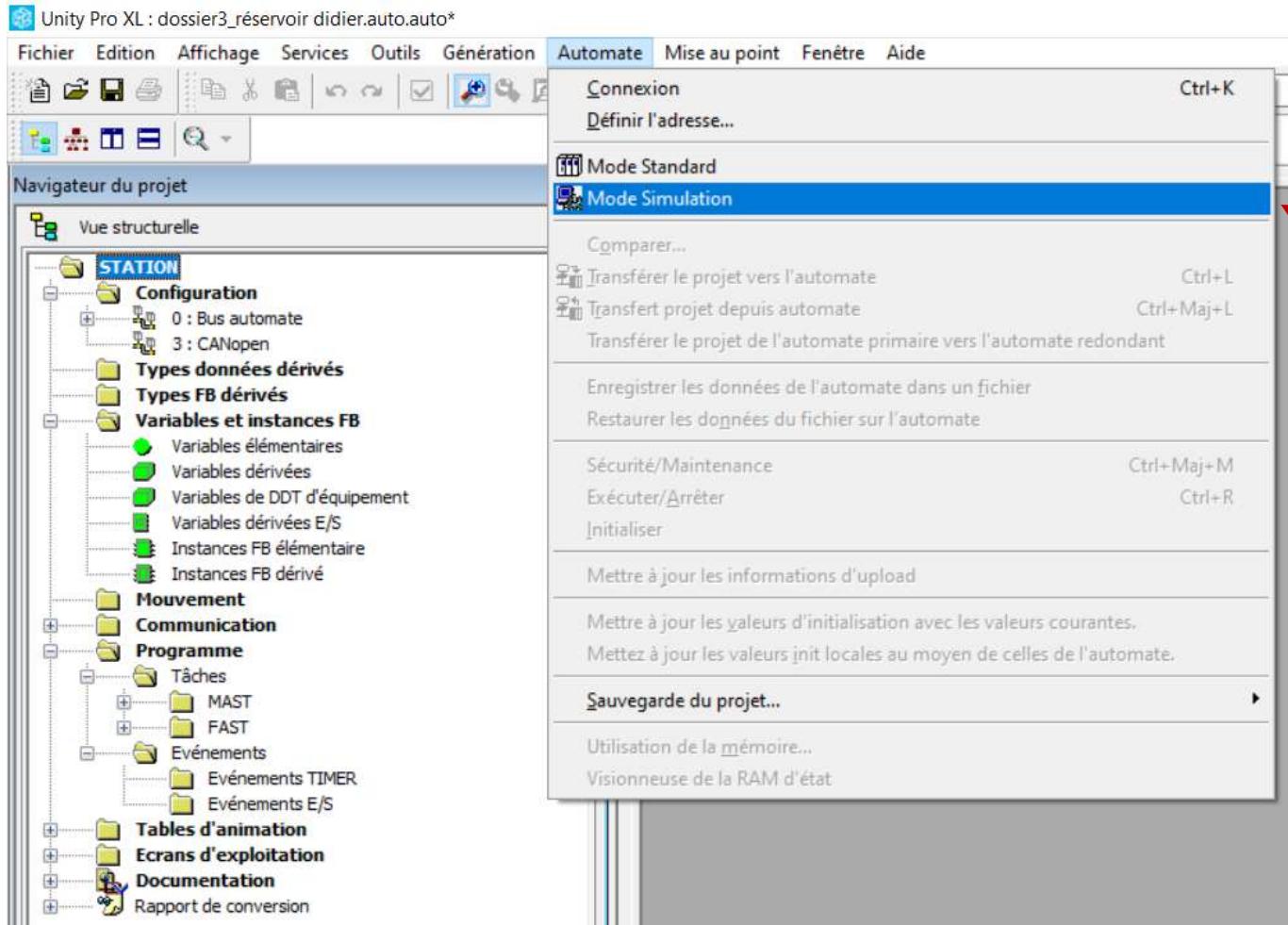
On va utiliser le mode simulation, il permet de vérifier la programmation sans disposer du matériel automate. Une fois ce mode activé, il faut télécharger le programme vers l'automate. Ensuite on va passer l'automate en mode « RUN » et il sera possible de voir les différentes parties de programme en fonctionnement (Taches, table de variable, supervision, ...).

Avant de pouvoir simuler un programme, il sera nécessaire de définir la configuration matérielle de l'automate : Unité centrale, alimentation, carte d'entrée / sortie, coupleurs, ...

Automation

Mise en mode simulation du programme :

Menu Automate : Description :

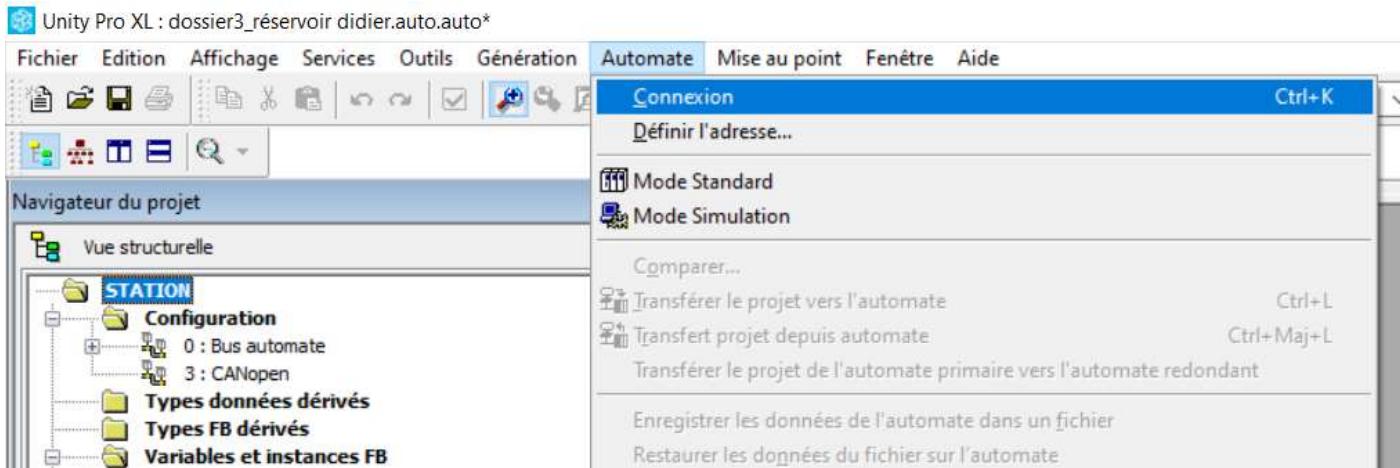


Mode simulation, il va permettre à l'utilisateur de vérifier la programmation qui sera envoyée vers l'automate

Automation

Mise en mode simulation du programme :

Menu Automate : Connexion :



Mode **Connexion**, il va permettre à l'utilisateur de faire la connexion avec l'automate en mode simulation

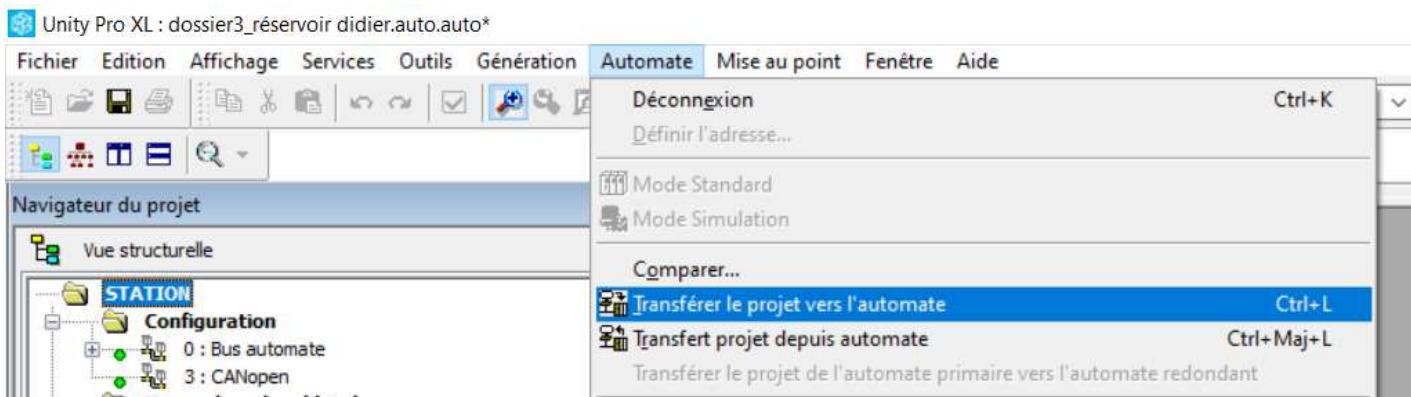
Une fois que l'on a validé le mode connexion il est nécessaire de transférer le programme (configuration matérielle, variables, les différentes tâches, les écrans d'animation, ...) vers l'automate.

Il suffit de choisir à nouveau le menu automate et de sélectionner la fonction transfert vers l'automate.

Automation

Mise en mode simulation du programme :

Menu Automate : Transfert :



Mode **Transfert** pour l'envoi de l'ensemble du projet dans l'automate, toujours en mode de simulation.

Ensute on va passer l'automate en mode RUN, il est possible de le faire directement en cochant la case « passer en mode RUN après transfert », ou vous utilisez le bouton « RUN » qui apparait dans la barre de menu une fois que le programme est transféré dans l'automate.

Automation

Mise en mode simulation du programme :

Menu Automate : Mode RUN :



Mode Exécuter pour le passage en mode « run » de l'automate, le projet est prêt à être simulé avec le logiciel.

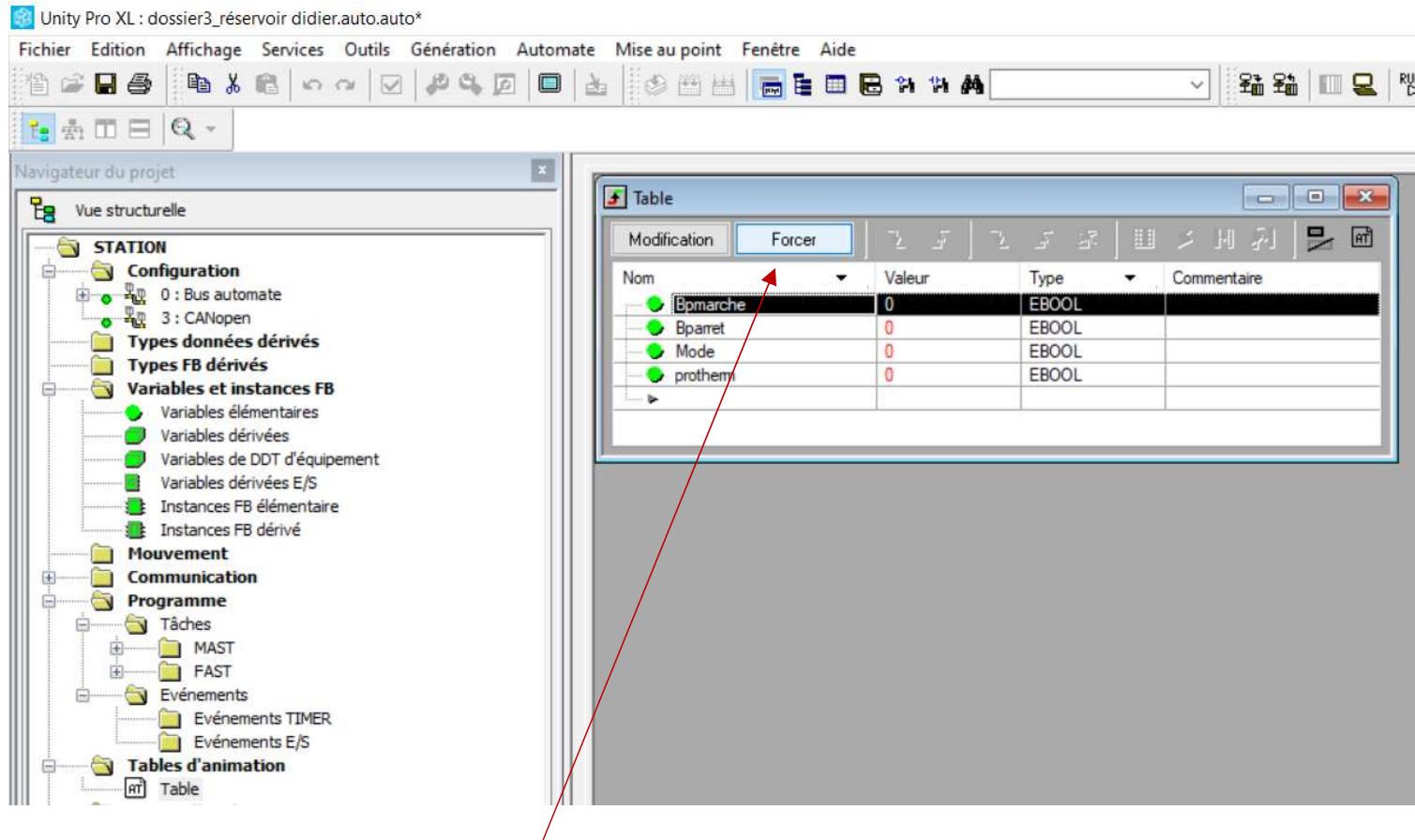
Pour pouvoir faire de la simulation, il faut utiliser les tables d'animations. C'est dans ces tables que l'on va définir les variables utiles à la simulation du programme. Il faudra penser aux différentes entrées nécessaires et également aux différentes variables internes que l'on a défini afin de faire la simulation.

Par exemple, pour un moteur avec un bouton « marche » et un « arrêt », nous devrons définir ces deux variables dans la table.

Automation

Mise en mode simulation du programme :

Menu Automate : Table d'animation :



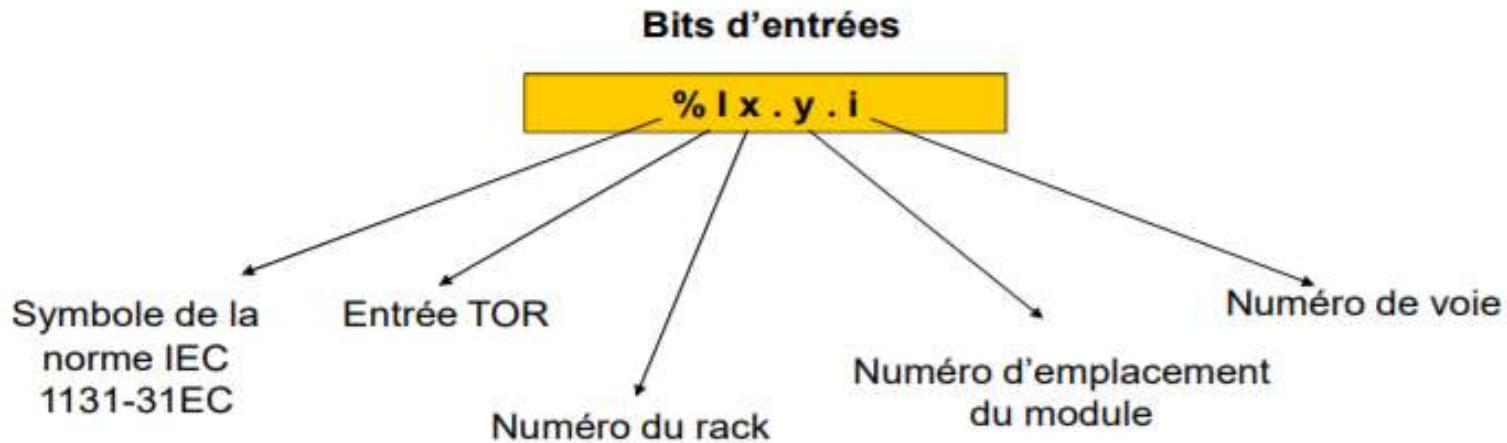
On utilise la commande « Forcer » pour mettre une variable à « 1 » ou à « 0 »

Automation

Les variables Automate:

Automate : Types de variables : Bit d'entrée

Adressage des bits sur M340



- Exemples :

%I0.5.15 = Entrée 15 du module situé à l'emplacement 5 du rack 0.

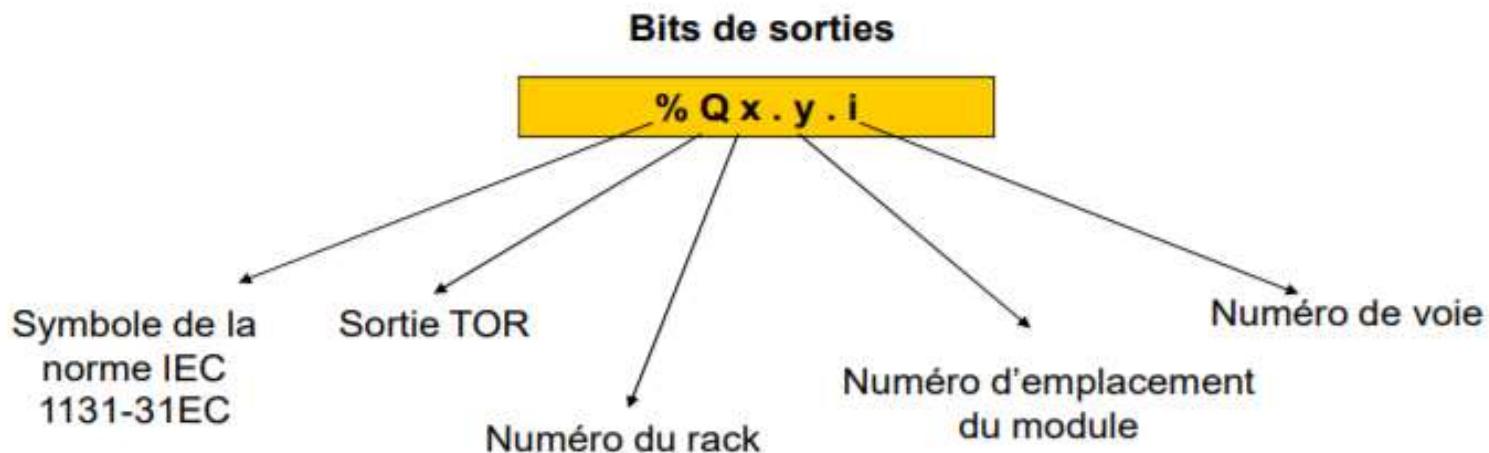
%I0.3.0 = Entrée 0 du module situé à l'emplacement 3 du rack 0.

Automation

Les variables Automate:

Automate : Types de variables : Bit de sortie

Adressage des bits sur M340



- Exemples :

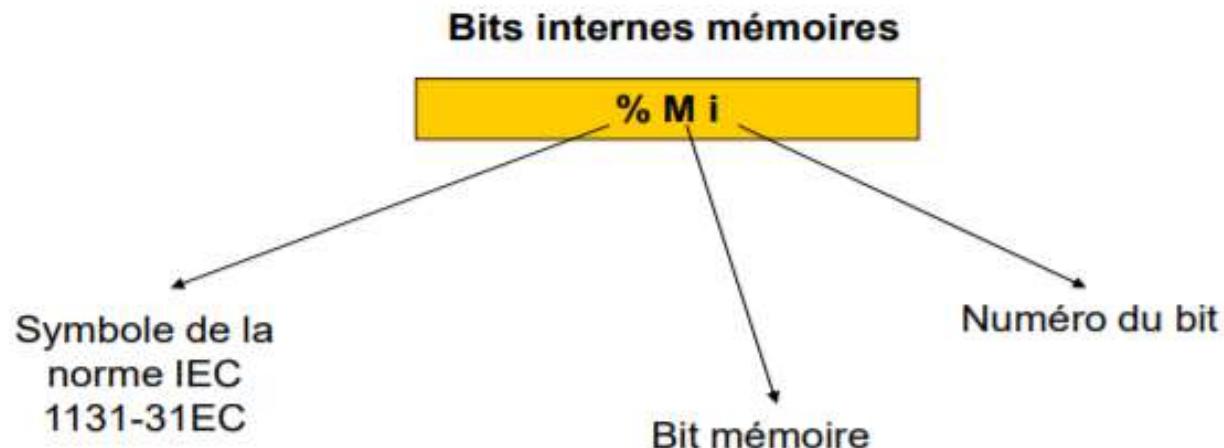
%Q0.6.11 = Sortie 11 du module situé à l'emplacement 6 du rack 0.
%Q0.3.5 = Sortie 5 du module situé à l'emplacement 3 du rack 0.

Automation

Les variables Automate:

Automate : Types de variables : Bit interne

Adressage des bits sur M340

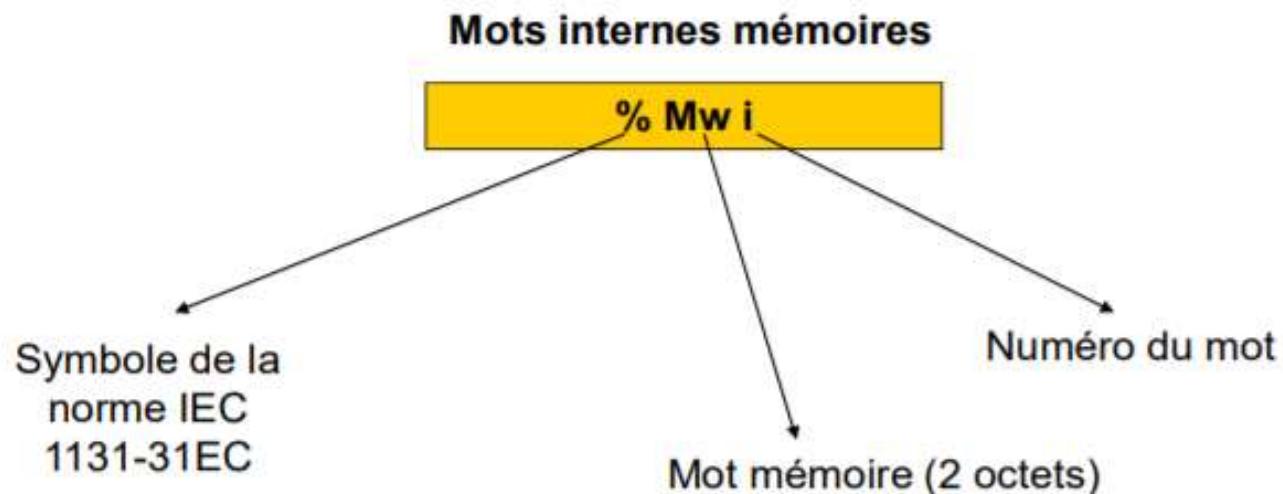


Automation

Les variables Automate:

Automate : Types de variables : Mot interne

Adressage des bits sur M340

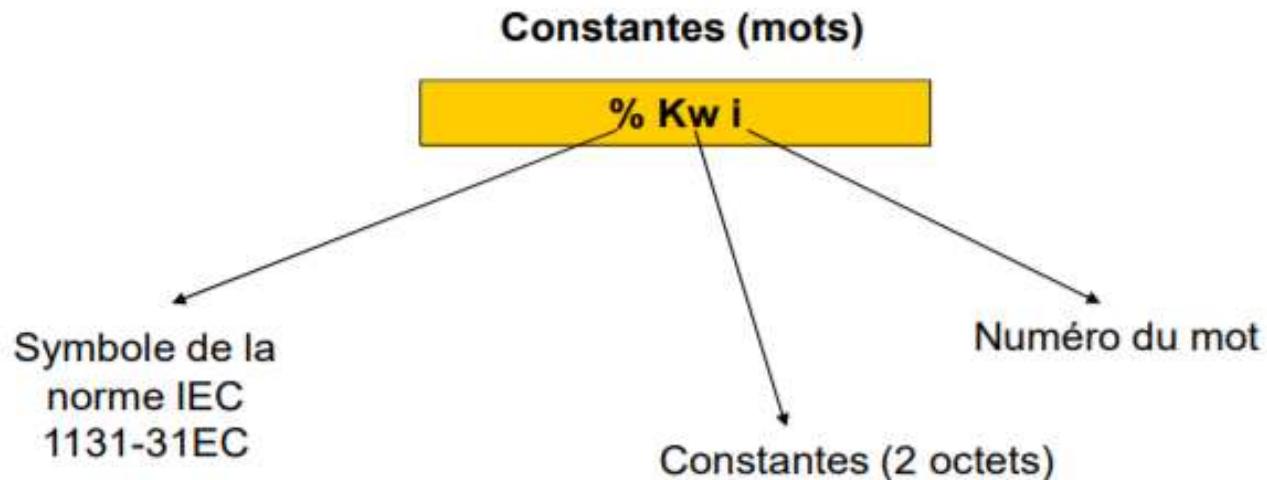


Automation

Les variables Automate:

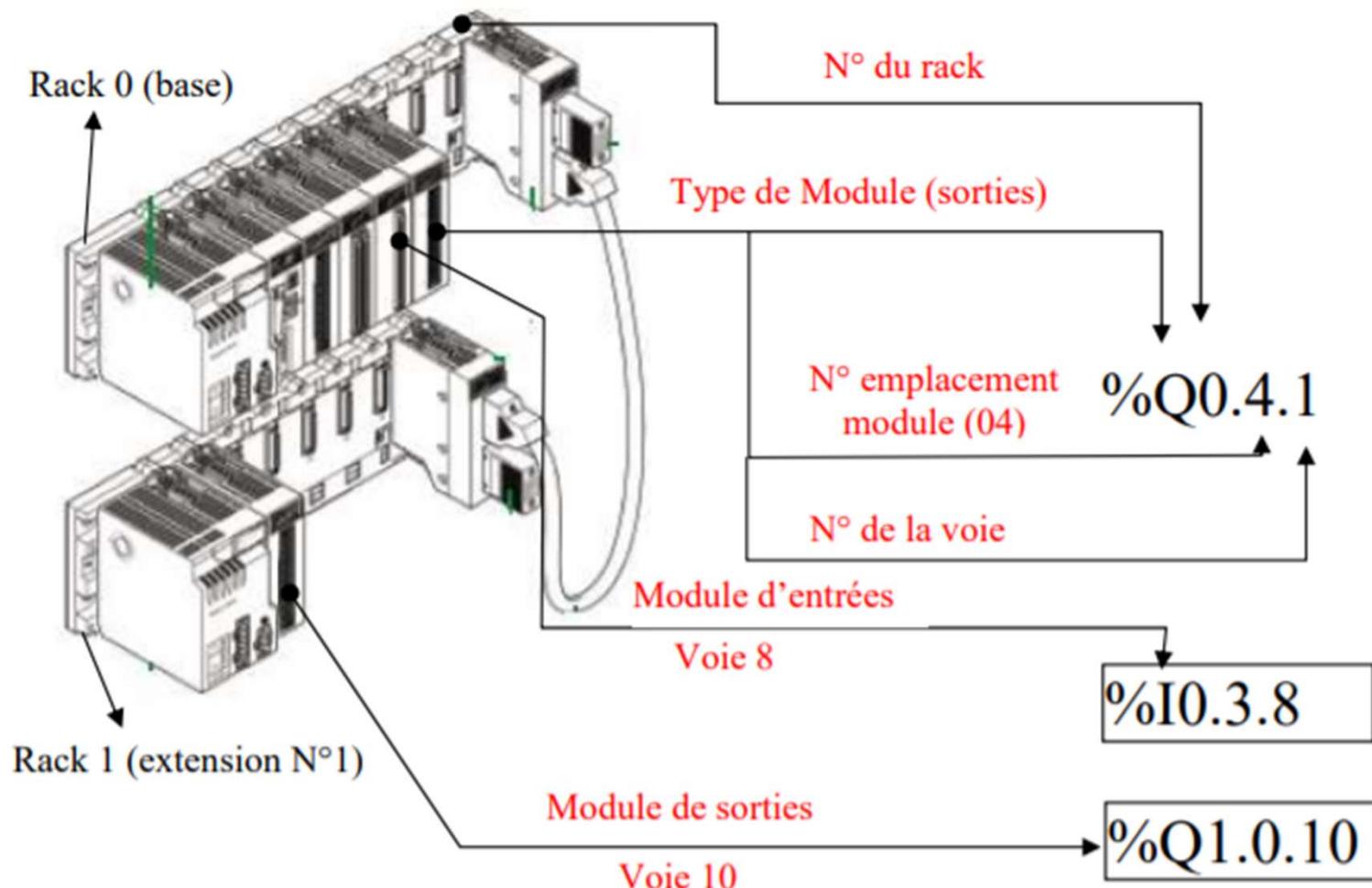
Automate : Types de variables : Constante interne

Adressage des bits sur M340



Automation

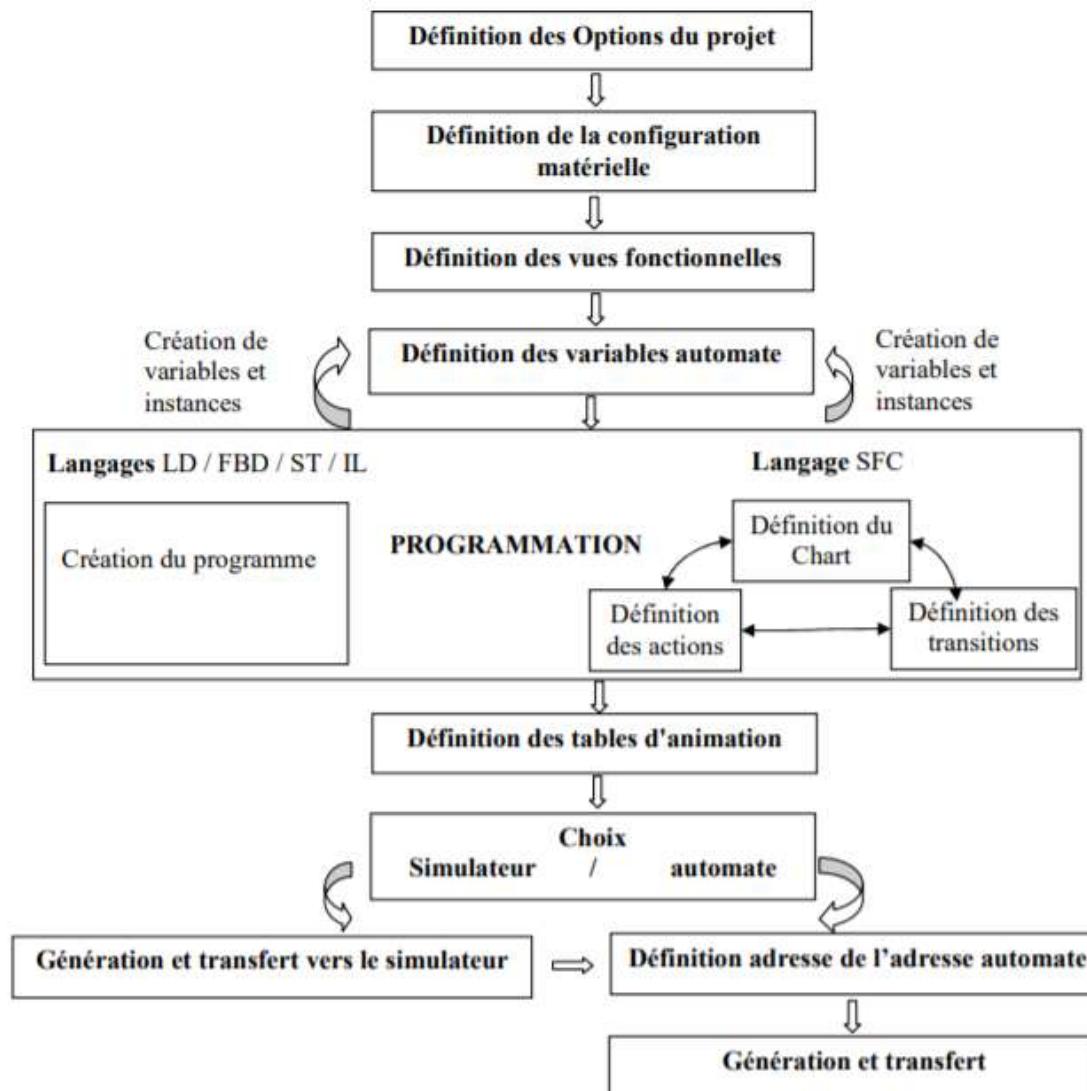
Adressage :



Automation

Méthodologie :

Méthodologie de création d'une Application Unity Pro



Automation

Les blocs de fonctions :

Dans la programmation des automates, tout comme dans des langages informatiques, il est possible d'utiliser des bibliothèques d'objets destinés à la réalisation des différents programmes à mettre en œuvre.

On appelle ces fonctions dans le jargon des automates « **Bloc Fonction** » ou « **FBD** ». Vous pouvez également créer vos propres blocs de fonctions « **DFB** ».

Le logiciel Unity Pro vous permet de créer des blocs fonction utilisateur **DFB**, en utilisant les langages d'automatismes. Un DFB est un bloc de programme que vous avez écrit, afin de répondre aux spécificités de votre application.

Il comprend : une ou plusieurs sections écrites en **langage à contacts (LD)**, en **liste d'instructions (IL)**, en **littéral structuré (ST)** ou en **langage à blocs fonctionnels (FBD)**, des paramètres d'entrée / de sortie, des variables internes publiques ou privées.

Les blocs fonction vous permettent de **structurer** et d'optimiser votre application. Vous pouvez les utiliser dès qu'une séquence de programme est répétée plusieurs fois dans votre application ou pour figer une programmation standard (par exemple, l'algorithme de commande d'un moteur incluant la prise en compte des sécurités locales). **L'export puis l'import** de ces blocs fonction permet leur utilisation par un groupe de programmeurs travaillant sur une même application ou dans des applications différentes.

Automation

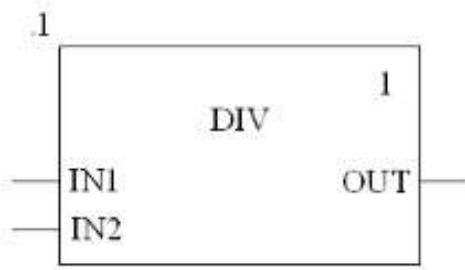
Les blocs de fonctions :

Fonction élémentaire :

Les fonctions élémentaires (EF) n'ont pas d'état interne. Lorsque les valeurs d'entrée sont identiques, la valeur de sortie est la même à chaque exécution de la fonction. Par exemple, l'addition de deux valeurs donne toujours le même résultat. Une fonction élémentaire est **représentée** graphiquement comme un **cadre** avec des **entrées** et une **sortie**. Les entrées sont toujours représentées sur la gauche et la sortie toujours sur la droite du cadre.

Le nom de la fonction, c'est-à-dire le type de fonction, est affiché au **centre** du cadre

Fonction élémentaire



Remarque : pour certaines fonctions élémentaires, il est possible d'augmenter le nombre d'entrées.

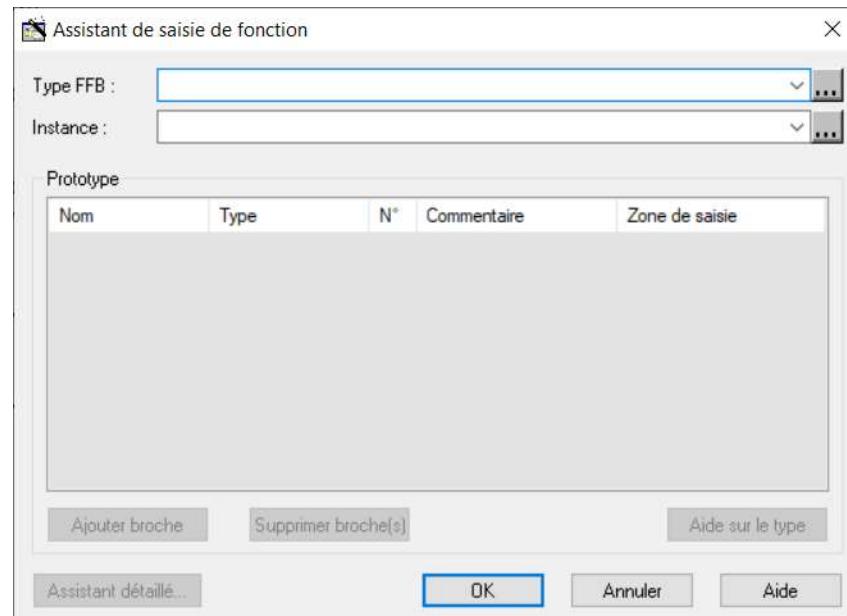
Automation

Les blocs de fonctions :

Fonction élémentaire : Accès depuis l'interface Unity



On utilise l'assistant de saisie (FFB), ensuite on va pouvoir choisir la fonction à utiliser.

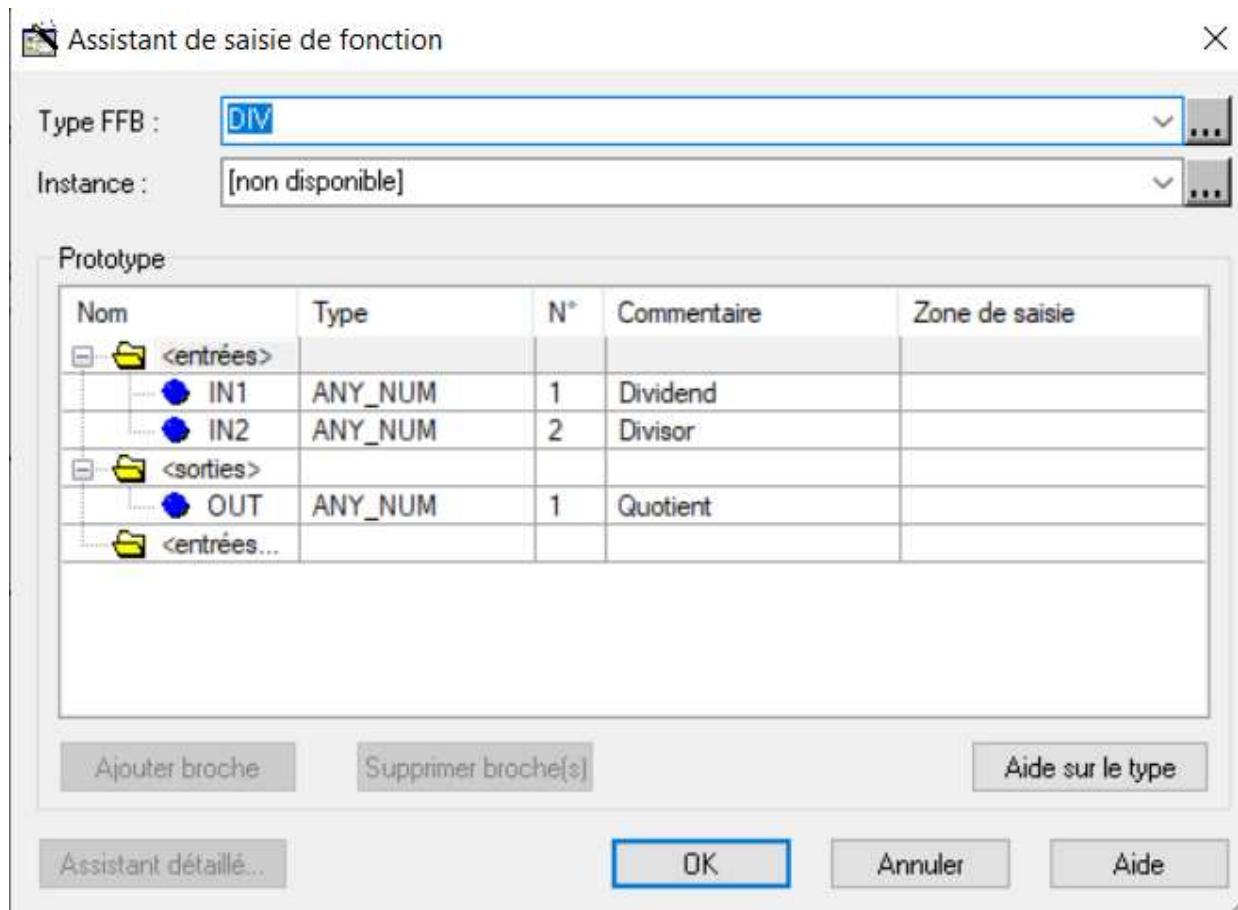


Automation

Les blocs de fonctions :

Fonction élémentaire : Exemple DIV

On utilise l'assistant de saisie (FFB), ensuite choisir DIV.



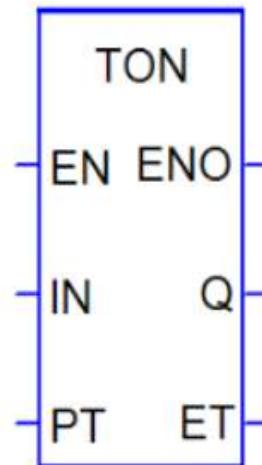
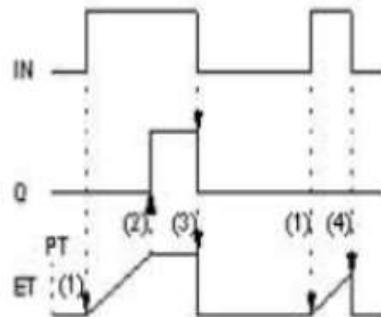
Automation

Les blocs de fonctions :

Bloc TON (Retard au Démarrage) : (EN : Enable Bloc)

Chronogramme

Représentation de la temporisation à l'enclenchement TON :



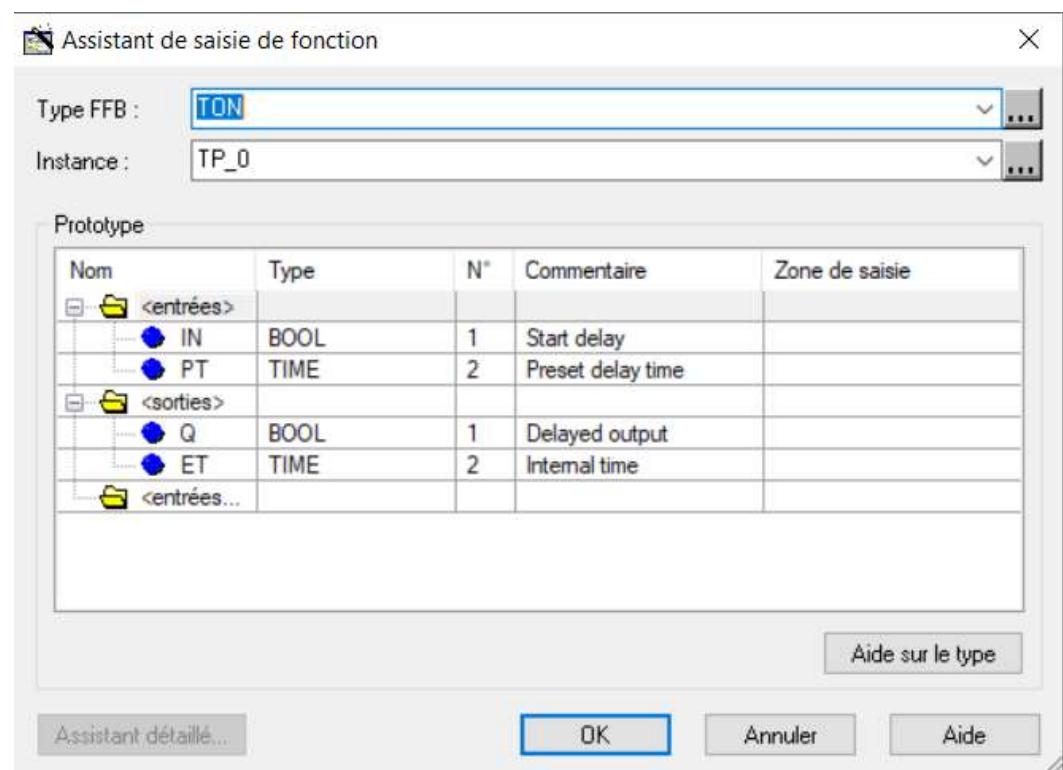
- Le bloc fonction est utilisé pour le retard de mise en marche.
- Les paramètres supplémentaires EN et ENO peuvent être configurés
- (1) Si IN passe à "1", l'horloge interne (ET) se déclenche.
- (2) Si l'horloge interne atteint la valeur de PT, Q passe à "1".
- (3) Si IN passe à "0", Q passe à "0" et l'horloge interne s'arrête/est remise à zéro.
- (4) Si IN passe à "0" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne passe à "1".

Automation

Les blocs de fonctions :

Bloc TON (Retard au Démarrage):

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	déclenchement temporisation
PT	TIME	Temps de retard
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne



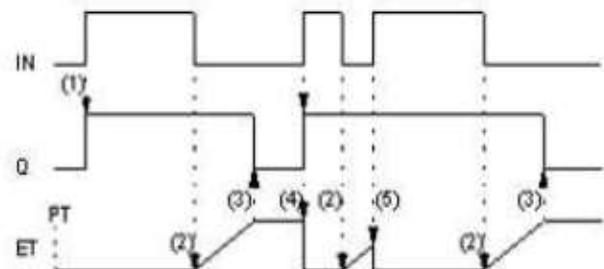
Automation

Les blocs de fonctions :

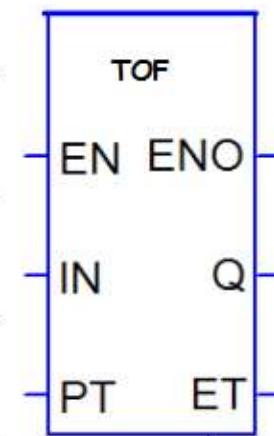
Bloc TOF (Retard au Déclenchement) :

Chronogramme

Représentation de la temporisation au déclenchement TOF :



- (1) Si IN passe à "1", Q passe à "1".
- (2) Si IN passe à "0", l'horloge interne (ET) se déclenche.
- (3) Si l'horloge interne atteint la valeur de PT, Q passe sur "0".
- (4) Si IN passe à "1", Q passe à "1" et l'horloge interne s'arrête/est remise à zéro.
- (5) Si IN passe à "1" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne soit remis à "0".



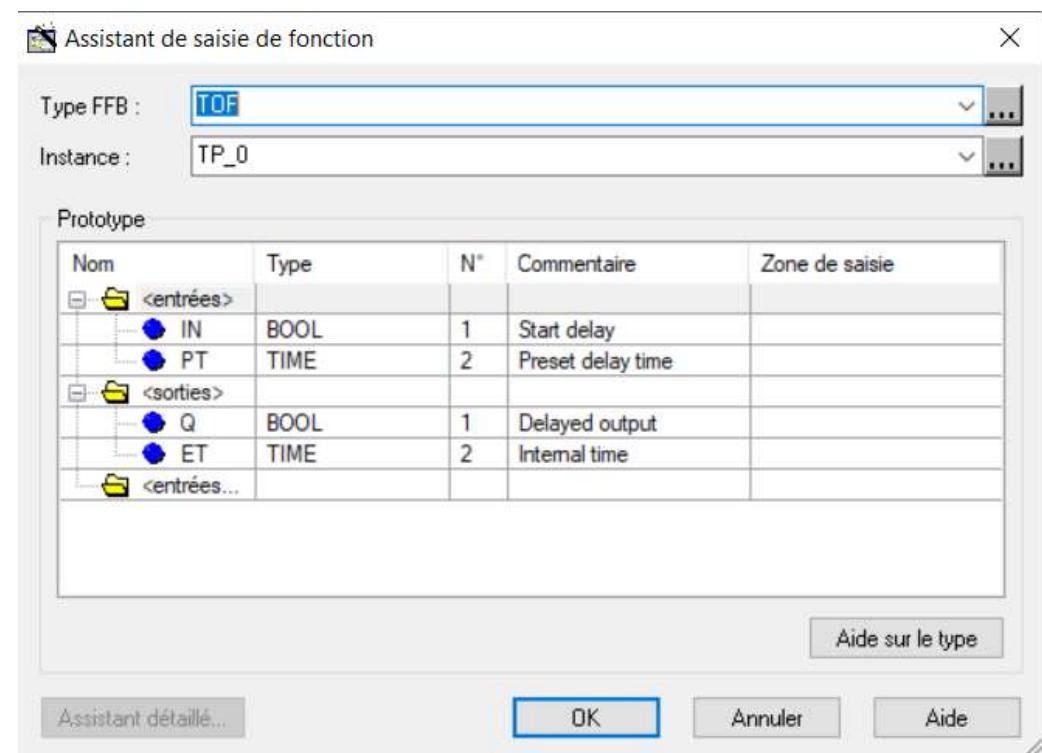
- Le bloc fonction est utilisé pour effectuer un retard au déclenchement
- Les paramètres supplémentaires EN et ENO peuvent être configurés

Automation

Les blocs de fonctions :

Bloc TOF (Retard au Déclenchement) :

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	Déclenchement temporisation
PT	TIME	Temps de retard
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne



Automation

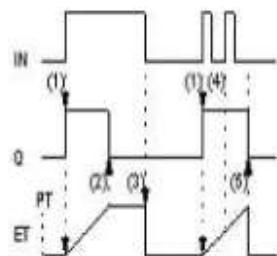
Les blocs de fonctions :

Bloc TP (Impulsion):

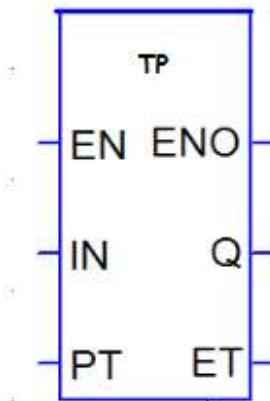
TP :

Chronogramme

Représentation de l'impulsion TP :



- (1) Si IN passe à "1", Q passe à "1" et l'horloge interne (ET) se déclenche.
- (2) Si l'horloge interne atteint la valeur de PT, Q est remis à "0" (indépendamment de IN).
- (3) L'horloge interne s'arrête/est remise à zéro, si IN est remis à "0".
- (4) Si l'horloge interne n'a pas encore atteint la valeur de PT, la présence d'un cycle d'impulsion au niveau de IN n'a aucune influence sur l'horloge interne.
- (5) Si l'horloge interne a atteint la valeur de PT et que IN est à l'état "0", l'horloge interne s'arrête/est remise à zéro et Q est remis à "0".



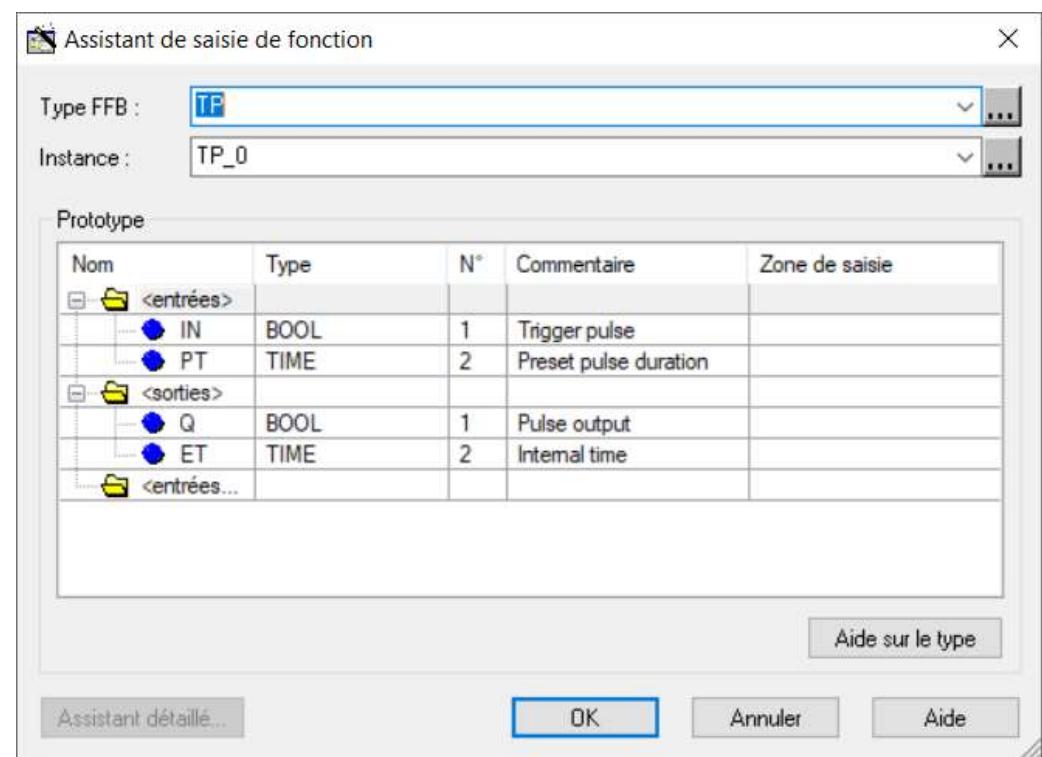
- Le bloc fonction est utilisé pour générer une impulsion de durée définie
- Les paramètres supplémentaires EN et ENO peuvent être configurés

Automation

Les blocs de fonctions :

Bloc TP (Impulsion) :

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	Déclenchement d'impulsion
PT	TIME	durée de l'impulsion
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne



Automation

Les blocs de fonctions :

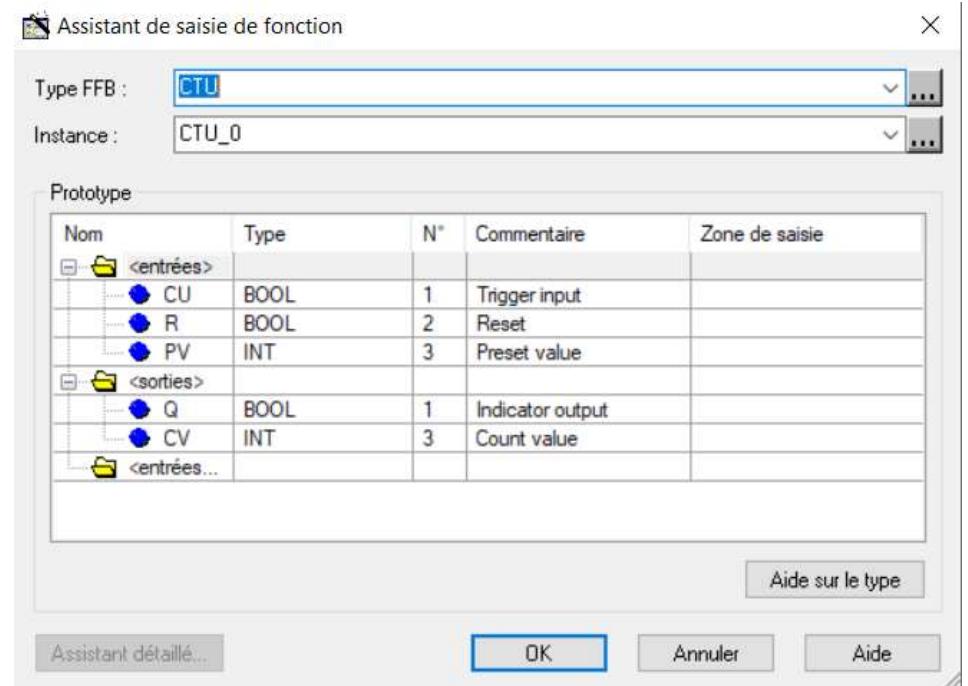
Bloc CTU (Comptage) :

Ces blocs fonction servent au comptage.

En présence d'un signal "1" à l'entrée R, la valeur "0" est attribuée à la sortie CV.

Chaque fois que la valeur, à l'entrée CU, passe de "0" à "1", la valeur de CV est incrémentée de 1. Si CV est supérieur ou égal à PV, la sortie Q passe à "1".

Le compteur ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu.



Automation

Les blocs de fonctions :

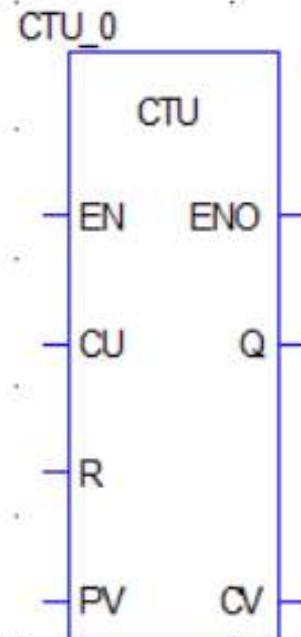
Bloc CTU (Comptage):

Paramètres d'entrées

Paramètres	données	signification
CU	BOOL	Comptage
R	BOOL	Remise à 0
PV	INT	Présélection

Paramètres de sorties

Paramètres	données	signification
Q	BOOL	Sortie
CV	INT	Valeur de comptage



Les paramètres EN et ENO peuvent être configurés.

Il existe d'autre type de compteur, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT. Les types se nomment CTU_DINT, CTU_UINT, CTU_UDINT

Automation

Les blocs de fonctions :

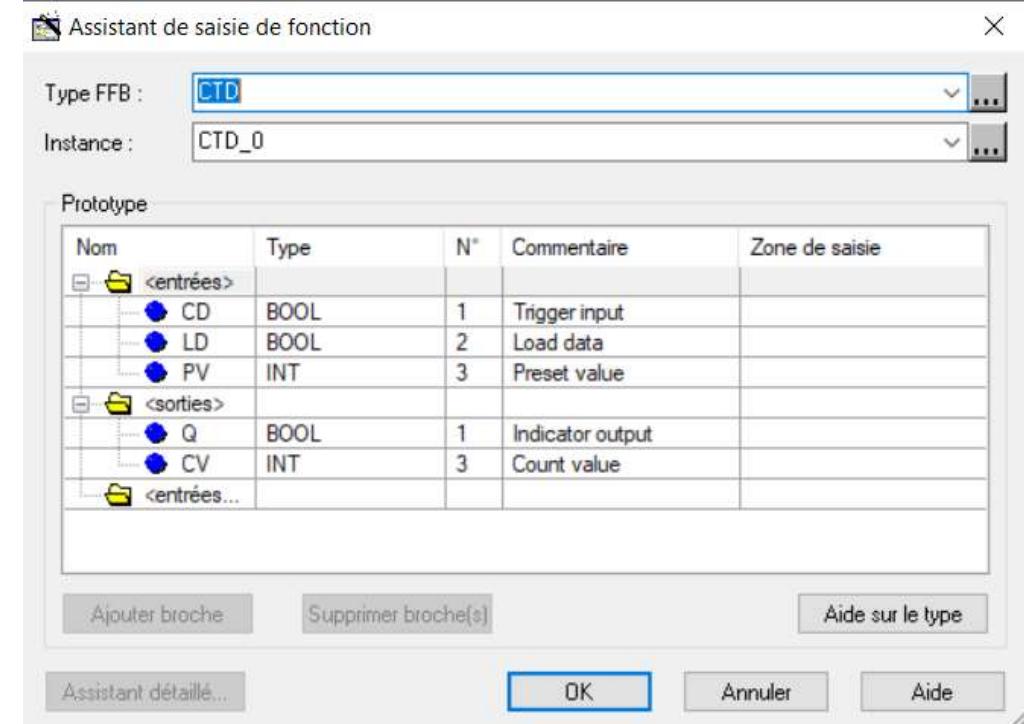
Bloc CTD (décomptage):

Ces blocs fonction servent au décomptage.

En présence d'un signal "1" à l'entrée LD, la valeur de l'entrée PV est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.

Si CV est inférieur ou égal à 0, la sortie Q est à 1.

Le bloc ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu.



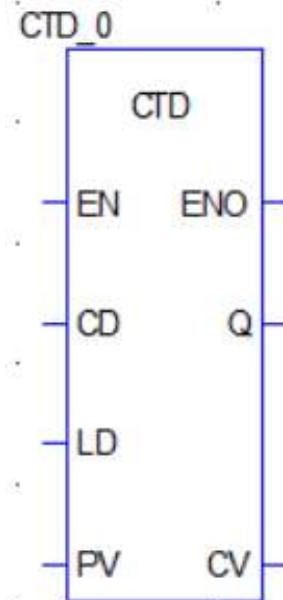
Automation

Les blocs de fonctions :

Bloc CTD (Décomptage):

Paramètres d'entrées		
Paramètres	données	signification
CD	BOOL	Décomptage
LD	BOOL	Chargement des données
PV	INT	Présélection

Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
CV	INT	Valeur de comptage



Les paramètres EN et ENO peuvent être configurés.

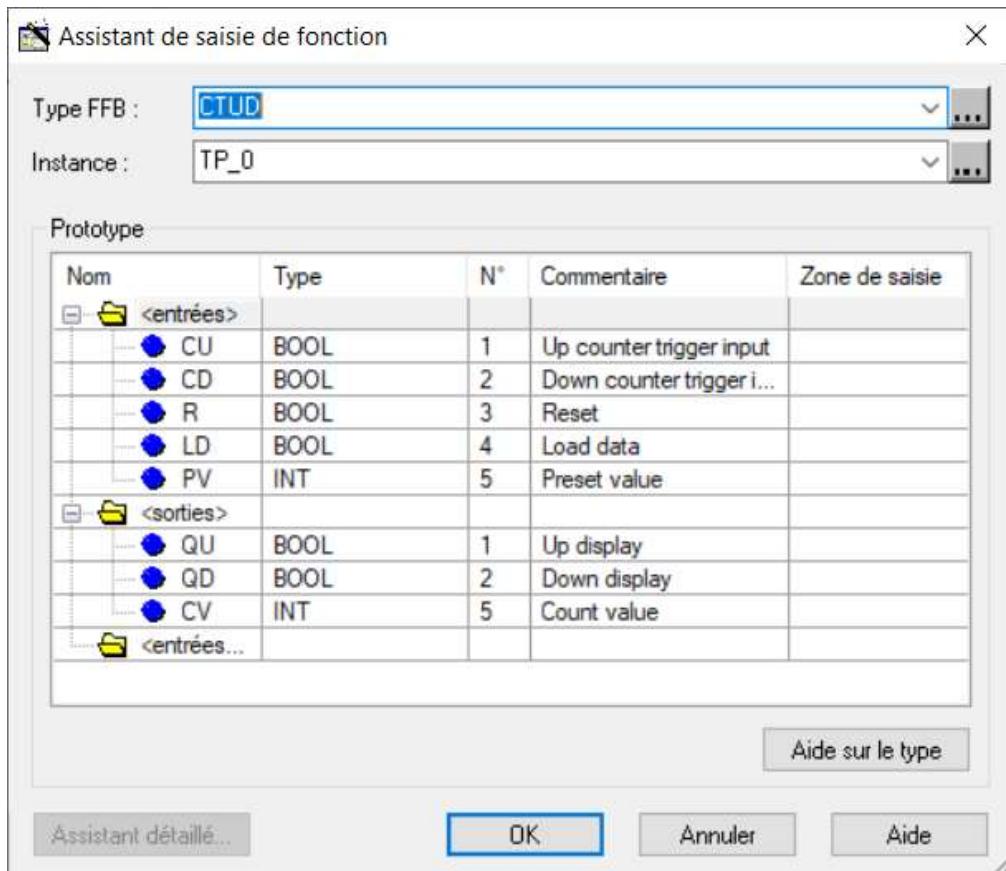
Il existe d'autre type de décompteur, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT.

Les types se nomment **CTD_DINT**, **CTD_UINT**, **CTD_UDINT**

Automation

Les blocs de fonctions :

Bloc CTUD (Décomptage / Comptage):



Ces blocs fonction servent au décomptage et au comptage.

En présence d'un signal "1" à l'entrée R, la valeur "0" est attribuée à la sortie CV. En présence d'un signal "1" à l'entrée LD, la valeur de l'entrée PV est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CU, passe de "0" à "1", la valeur de CV est incrémentée de 1. Chaque fois que la valeur, à l'entrée CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.

En cas de présence simultanée du signal "1" aux entrées R et LD, l'entrée R est prédominante.

Si CV est supérieur ou égal à PV, la sortie QU passe à "1"

Si CV est inférieur ou égal 0, la sortie QD passe à "1".

Le bloc ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu.

Automation

Les blocs de fonctions :

Bloc CTUD (Décomptage / Comptage):

Les paramètres EN et ENO peuvent être configurés.
Il existe d'autre type de bloc, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT. Les types se nomment **CTUD_DINT**, **CTUD_UINT**, **CTUD_UDINT**

Paramètres d'entrées

Paramètres	données	signification
CU	BOOL	Comptage
CD	BOOL	Décomptage
R	BOOL	Remise à 0
LD	BOOL	Chargement des données
PV	INT	Présélection

Paramètres de sorties

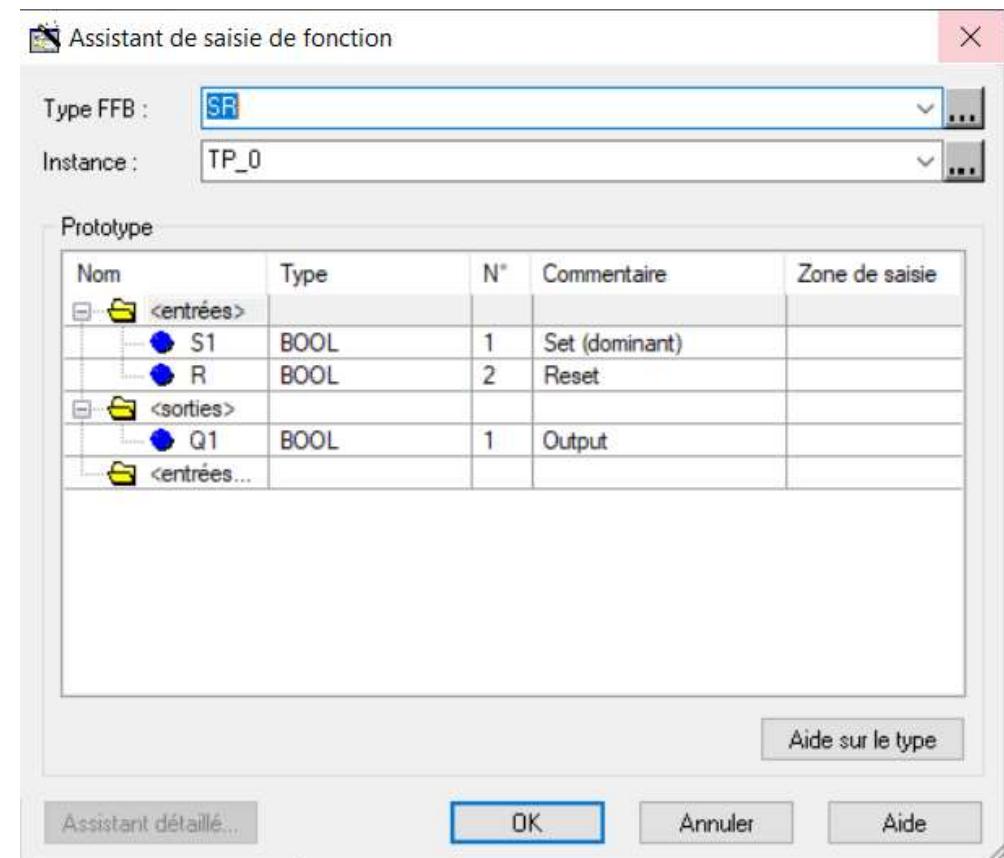
Paramètres	données	signification
QU	BOOL	Sortie de comptage
QD	BOOL	Sortie de décomptage
CV	INT	Valeur de comptage

Automation

Les blocs de fonctions :

Bloc RS (Bascule Set / Resset):

Une bascule **SR** est une **fonction mémoire**. Cette fonction mémoire est réalisée par un opérateur logique qui peut stocker une information jusqu'à ce que cette information soit effacée par une autre information. L'opération de stockage d'information s'appelle "SET" (Mise à un) l'opération d'effacement s'appelle "RESET" (Mise à zéro). Ces opérateurs peuvent être électriques, électroniques, pneumatiques ou constitué sur un automate par un bloc de fonction SR.

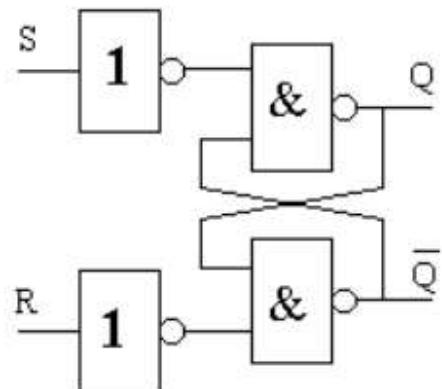


Automation

Les blocs de fonctions :

Bloc RS (Bascule Set / Resset):

A la mise sous tension la bascule est déjà positionnée c'est pour cela que l'état [0, 0] est fonction de l'état précédent. Si on peut décrire le comportement de la bascule pour les états [0,1] et [1, 0], on est par contre incapable de décrire son fonctionnement pour l'état [1,1], on peut dire qu'il est **instable**. Il existe des **mémoires à marche prioritaire** on les distingue par un petit astérisque. Pour cette mémoire l'indétermination (état [1, 1]) est levée.



R	S	Q	\bar{Q}
0	0	Etat précédent	
0	1	1	0
1	0	0	1
1	1	Etat interdit	

Automation

Les blocs de fonctions :

Bascule RS dans le cas du télérupteur :

Qu'entend - t'on par télérupteur, ce nom est utilisé dans le monde de l'électricité et permet de faire un allumage depuis un point avec comme fonctionnement : Un appui sur un bouton allume une lampe, l'appui suivant éteint la lampe, et ainsi de suite. Il n'y a qu'un seul interrupteur pour la commande de la lampe faisant passer la sortie de l'état 0 à l'état 1 et ensuite à l'état 0.

Cette façon de faire demande d'avoir un câblage en électricité ou l'ensemble des boutons, si vous en aviez plusieurs, sont en parallèle sur la commande. Vous trouverez sur une installation domestique, comme discuté précédemment, une partie commande et une partie puissance.

Nous allons regarder la représentation électrique et comment réfléchir à la transformation à apporter au circuit électrique pour le remplacer par un automate programmable.

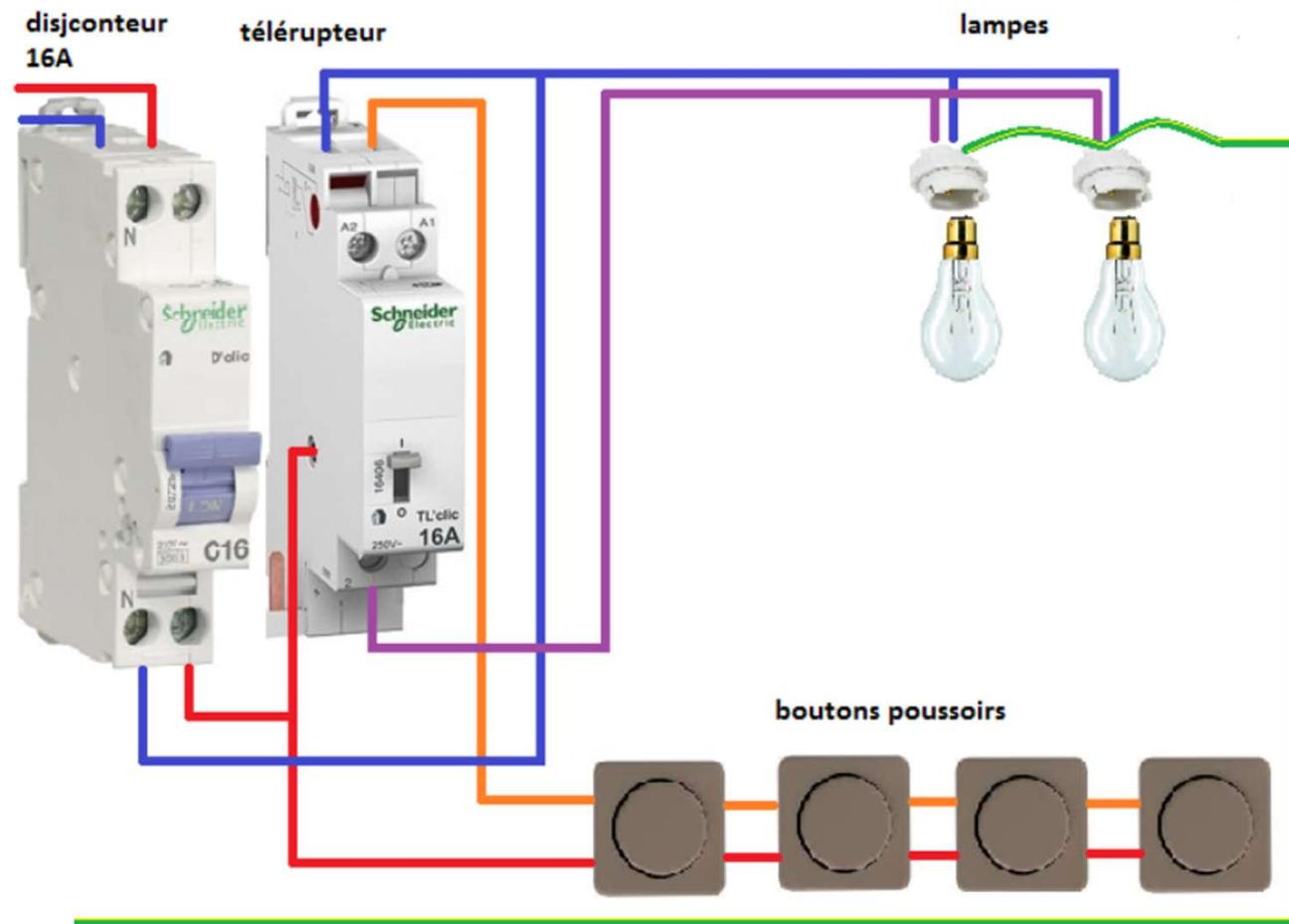
Cet exercice sera à réaliser au laboratoire en incluant la réflexion un ou plusieurs boutons de commandes.

Automation

Exercice de laboratoire :

Bascule RS dans le cas du télérupteur :

Schéma électrique :



Automation

Exercice de laboratoire :

Bascule RS dans le cas du télérupteur :

Sur base du schéma précédent, on remarque que nous avons un pilotage de la lampe avec plusieurs boutons, et un élément qui permet de faire la commande, c'est le « télérupteur ». Il y a également un disjoncteur qui permet de faire passer la puissance. Trois couleurs de câble électrique qui sont :

- Le neutre, (**bleu**)
- une phase, (**Mauve**)
- et la terre, (**Vert** et **jaune**).

Pour réaliser la programmation, nous aurons besoin d'un élément qui va permettre d'allumer et éteindre la lampe. Le plus simple sera de passer par un **bloc de fonction** de type « **SR** ».

Il est **important** de ne pas oublier le fonctionnement du **cycle de l'automate** et surtout de ça vitesse d'exécution.

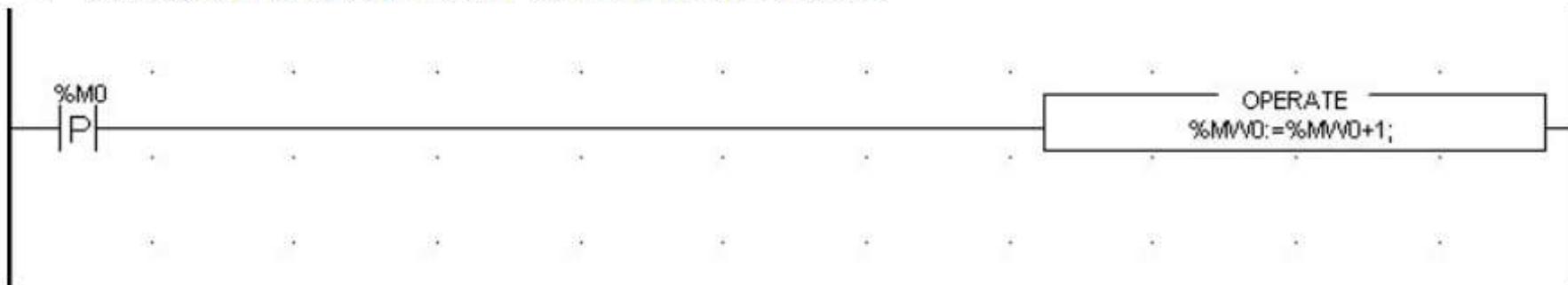
La programmation devra être du type « contact », il faut également réfléchir à quelle information sera nécessaire pour piloter la bascule. Le système sera de type séquentiel car on devra tenir compte de l'état précédent de la lampe pour la commande suivante.

Automation

Astuce : Détection de front :

Contexte :

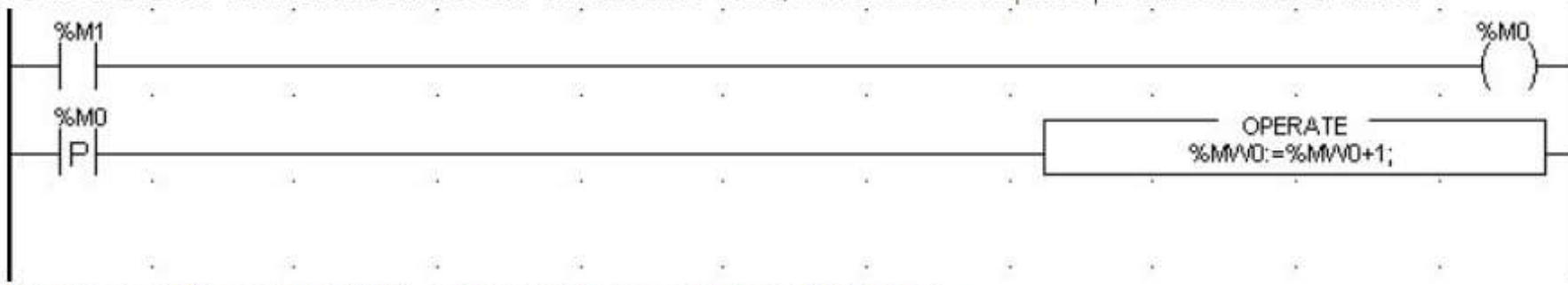
- le programme est testé avec le simulateur de Unity Pro,
- un front est utilisé sur une variable de type EBOOL (dans l'exemple ci-dessous front positif sur le bit %M0)
- la variable EBOOL est mise à 1 via une table d'animation.



Dans ces conditions, **le simulateur ne sait pas gérer le front** et voit à chaque tour de cycle le front à 1.

Le mot %MW0 n'est pas incrémenté à chaque tour de cycle automate.

Pour détecter le front montant sur la variable %M0, il faut utiliser le principe du schéma suivant :



Dans ce cas la variable %M1 est mise à 1 via une table d'animation.

Le mot %MW0 est incrémenté une seule fois.

Le front montant de %M0 est bien détecté.

Automation

Les blocs de fonctions :

Bascule RS dans le cas du télérupteur :

Exercice complémentaire : réaliser le cas du télérupteur sur Arduino en utilisant un bouton et une diode. Calculer la valeur des résistances avec justification et rédiger le programme pour obtenir le même résultat que celui obtenu avec l'automate.

Rappel : pensez à la boucle « Loop » de l'Arduino et son temps d'exécution car vous allez rencontrer les mêmes problèmes dans la gestion de la programmation.

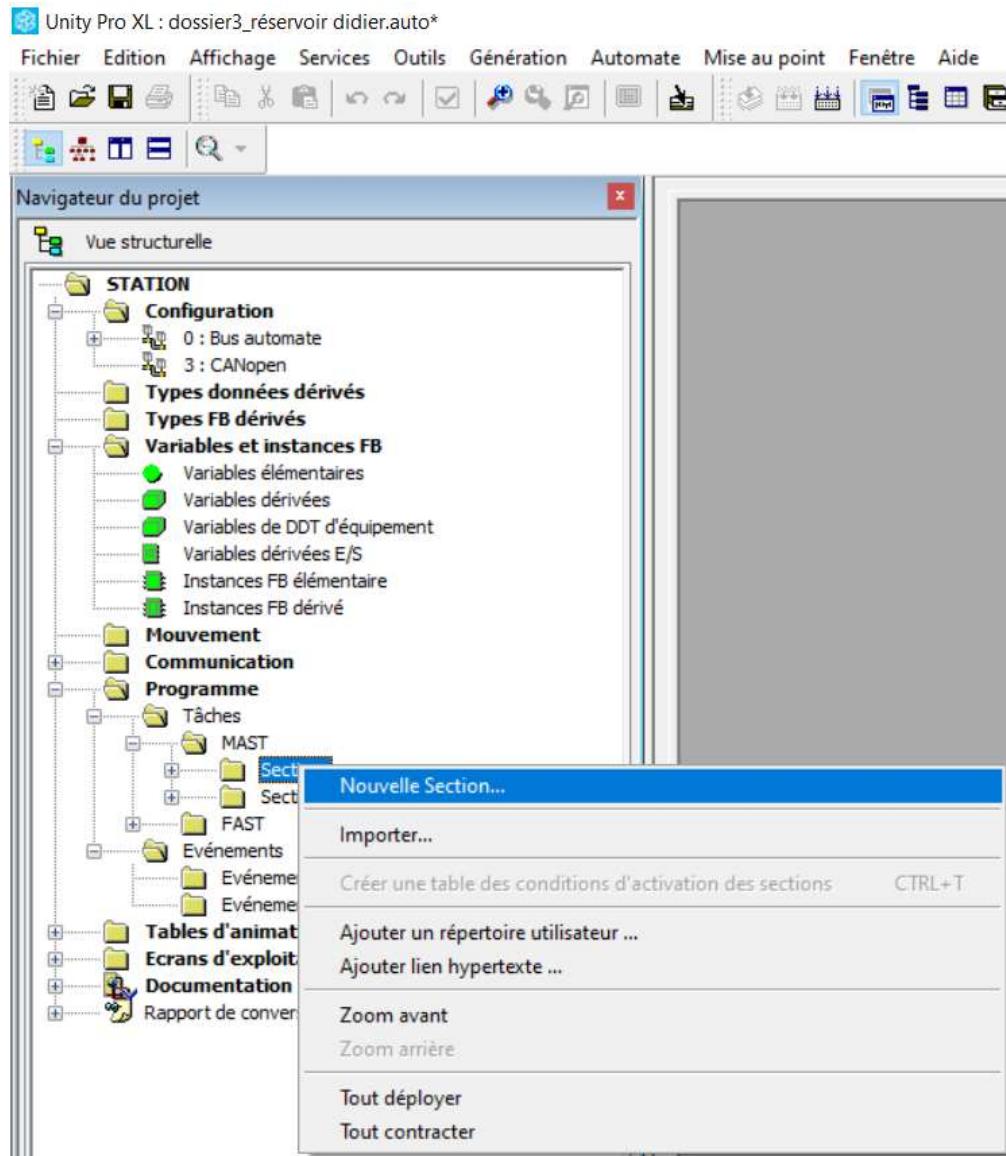
Automation

Les différents langages de programmation :

On peut Sélectionner le langage :

- **ST** : littéral structure
- **IL** : list instruction
- **FBD** : fonctionnal bloc diagram
- **SFC** : grafcet
- **LD** : ladder

Pour effectuer le choix du langage, il faut le déterminer au moment de la création de la tache « Mast », « Fast » ou « événementielle ».



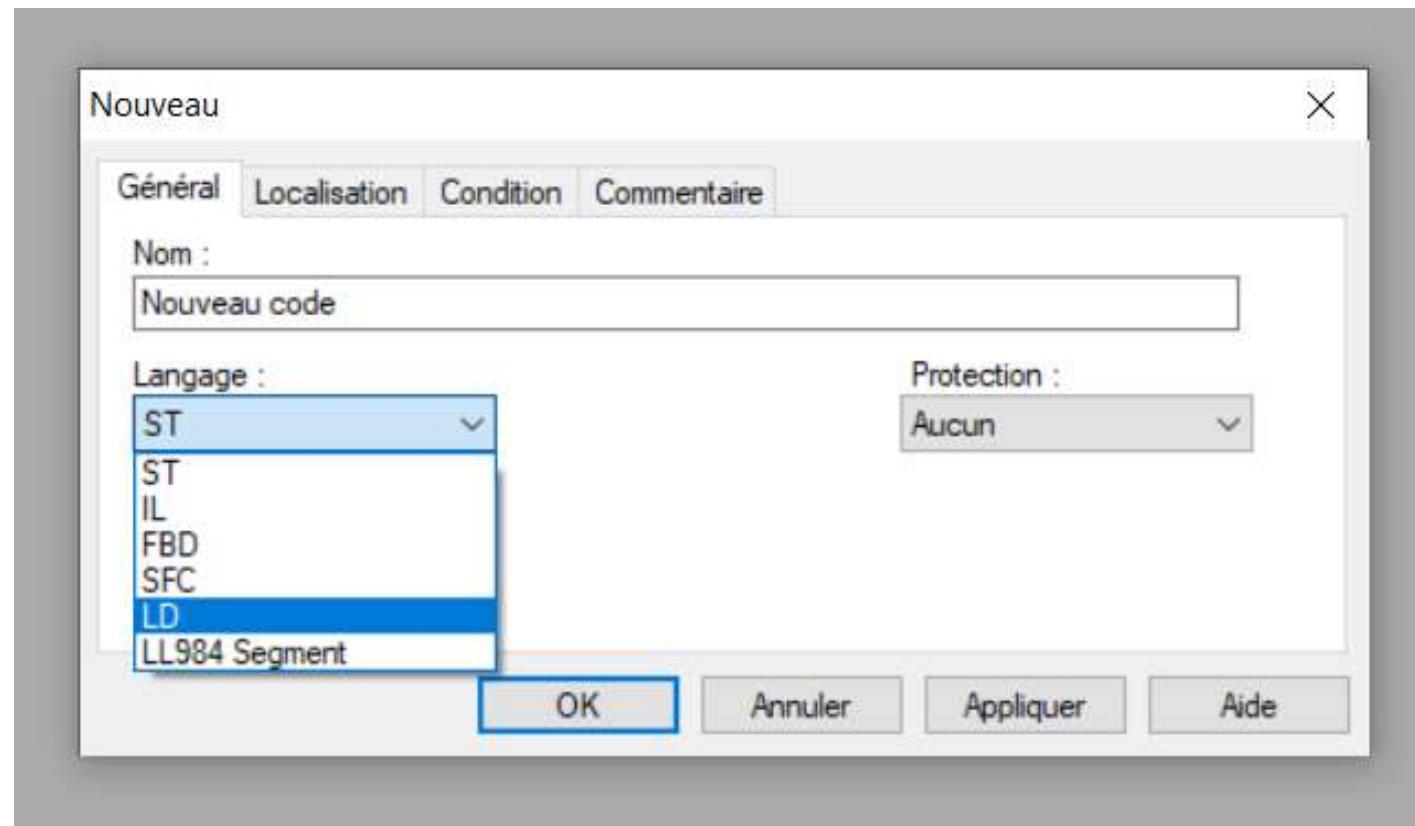
Automation

Les différents langages de programmation :

On peut Sélectionner le langage :

- ST : littéral structure
- IL : list instruction
- FBD : fonctionnal bloc diagram
- SFC : grafcet
- LD : ladder

Pour effectuer le choix du langage, il faut le déterminer au moment de la création de la tache « Mast », « Fast » ou « événementielle ».



Automation

Les différents langages de programmation :

ST : littéral structure

ST est un langage de programmation de niveau supérieur, comme PASCAL ou C. Le code des programmes est constitué d'expressions et d'instructions. Contrairement à IL (Instruction List), il permet d'utiliser plusieurs constructions pour programmer des boucles, ce qui permet le développement d'algorithmes complexes.

Exemple :

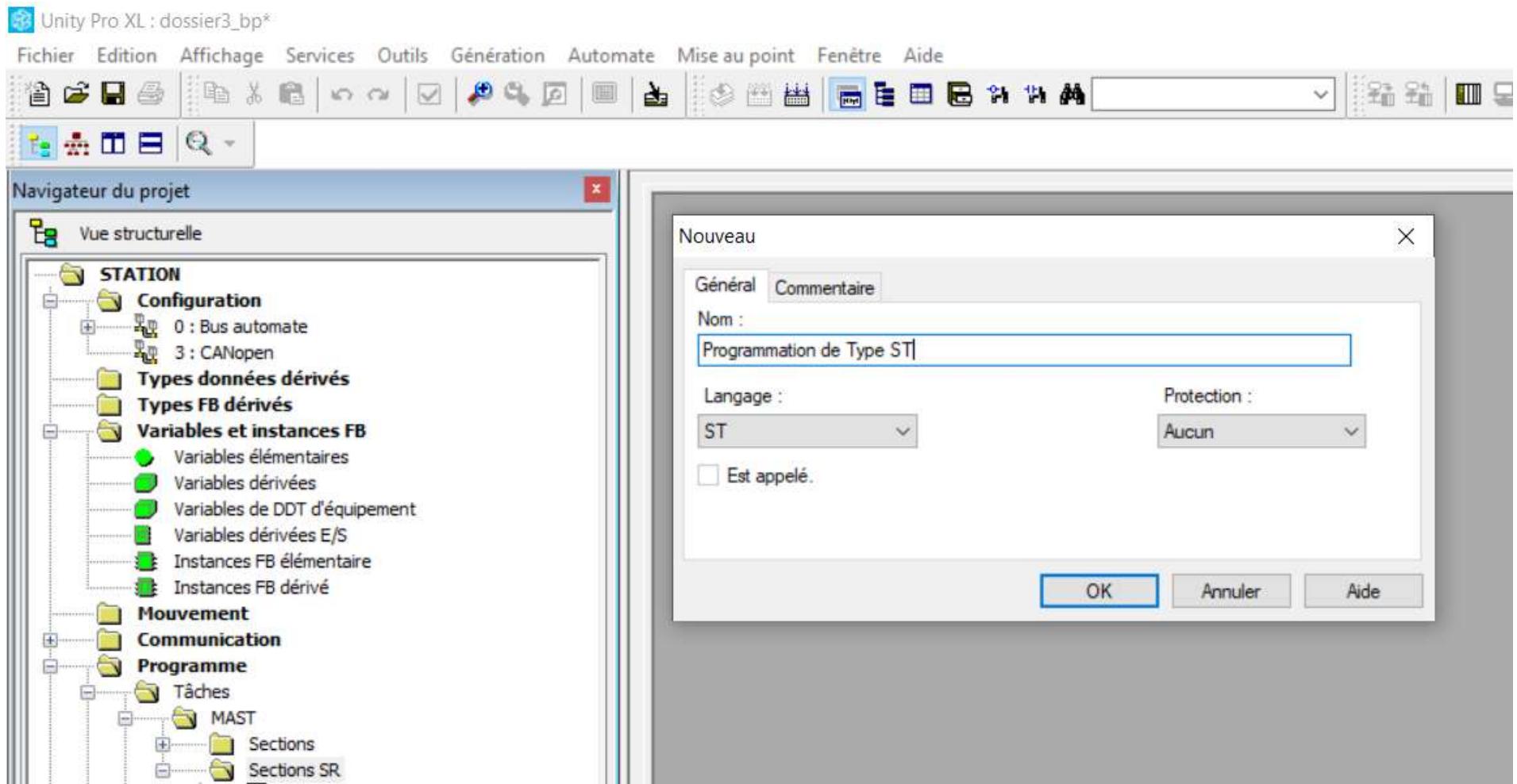
```
IF value < 7 THEN
    WHILE value < 8 DO
        value:=value+1;
    END_WHILE;
END_IF;
```

Pour pouvoir utiliser ce type de programmation, comme expliqué précédemment, il faut choisir le mode ST au moment de la création d'une « SR » par exemple

Automation

Les différents langages de programmation :

ST : littéral structure



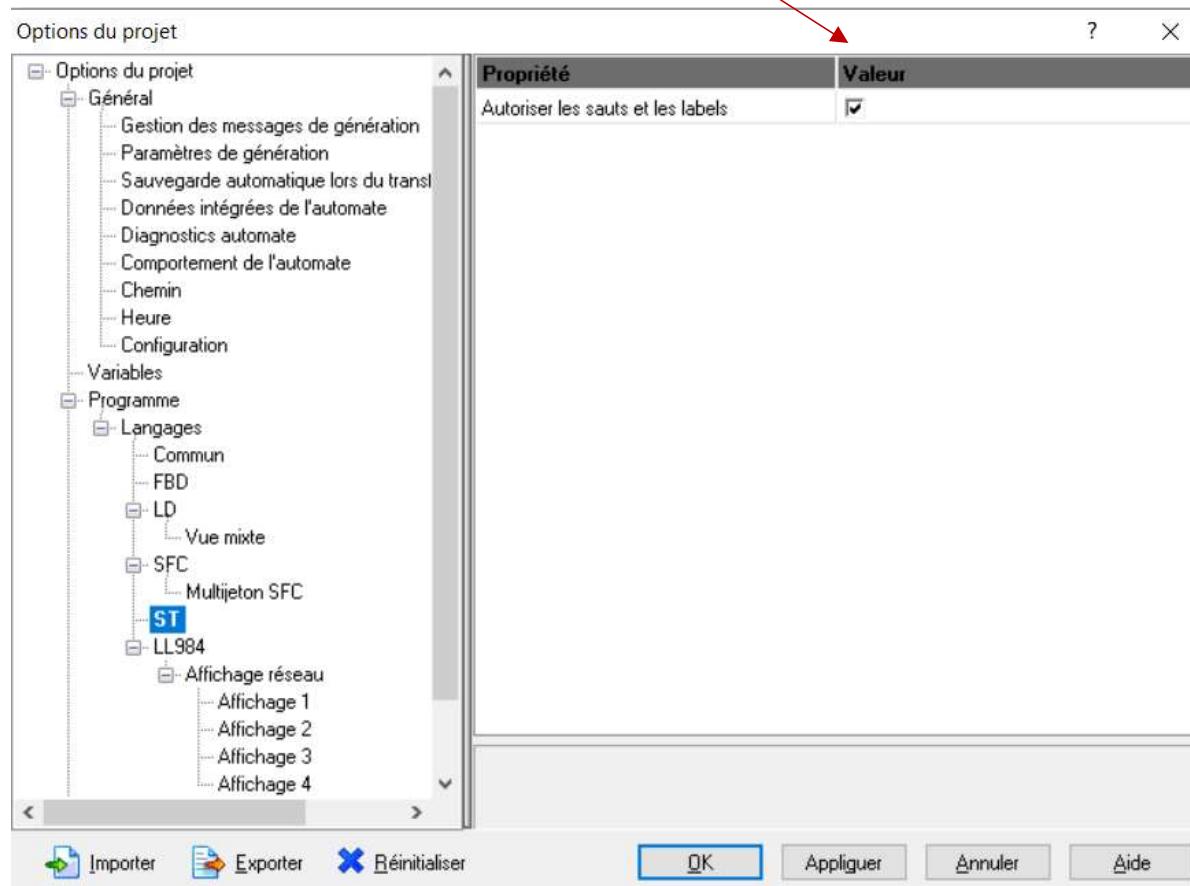
Automation

Les différents langages de programmation :

ST : littéral structure

Dans ce type de langage, on peut utiliser des « Labels » et des « sauts » de projet, comme définis dans la norme IEC-61131-3.

Dans outil, options du projet, il faut vérifier que l'option est activée :



Automation

Les différents langages de programmation :

ST : littéral structure

Les **instructions** sont les **commandes** du langage de programmation ST, une instruction doit toujours être terminée par un « ; » comme en C. Une ligne peut contenir plusieurs instructions séparées par un « ; ».

Les variables :

Nom	Notation	Taille (bit)	Etendue
Booléen	BOOL	1	[0(false),1(true)]
Entier court	SINT	8	[-128 ; 127]
Entier	INT	16	[-32 768 ; 32 767]
Entier double	DINT	32	[-2 147 483 648 ; 2 147 483 647]
Entier court non signé	USINT	8	[0 ; 255]
Entier non signé	UINT	16	[0 ; 65 535]
Entier double non signé	UDINT	32	[0 ; 4 294 967 295]
Nombre flottant	REAL	32	[-2 ¹²⁸ ; 2 ¹²⁸]
Nombre flottant long	LREAL	64	[-2 ¹⁰²⁴ ; 2 ¹⁰²⁴]
Entier sur 8 bits (en hexa)	BYTE	8	[16#00 ; 16#FF]
Entier sur 16 bits (en hexa)	WORD	16	[16#0000 ; 16#FFFF]
Entier sur 32 bits (en hexa)	DWORD	32	[16#0000 ; 16#FFFFFFFF]
Temps	TIME(*)		T#0,00s à T#21 474 836,47s
TIME_OF_DAY, DATE_AND_TIME, DATE,			DATE_AND_TIME#2001-04-02- 10:15:29.00 DATE#2001-04-02
Chaine de caractères	STRING	8 bits / ASCII	1 à 255 caractères ASCII

Automation

Les différents langages de programmation :

ST : littéral structure

Les opérateurs :

Opérateurs	Description	Opérateur	Description
Fonctions logiques			
NOT	NON	OR	OU
AND	ET	XOR	OU Exclusif
Opérateurs arithmétiques			
+	Addition	-	Soustraction
mod	Reste de la division entière	/	Division
		*	Multiplication
Opérateurs de comparaison			
=	Égal à	<>	Different de
>	Supérieur à	>=	Supérieur ou égal à
<	Inférieur à	<=	Inférieur ou égal à

Automation

Les différents langages de programmation :

ST : littéral structure

Les commentaires :

Pour ajouter des commentaires à vos programmes, il faut utiliser les parenthèses avec une étoile :
(* Commentaires *).

Exemple de définition de variable : Pour définir les variables on utilise les mots clefs « VAR » et « END_VAR ».

VAR

```
X ARRAY [1..5] OF INT := 1,2,3,4,5; (* Tableau de 5 entiers *)
END_VAR
```

Le Langage met à disposition les boucles FOR, les structures de type IF THEN ELSE, le WHILE, REPEAT et également CASE.

Exemple :

```
k:=0;                      (* Initialisation de la variable k à 0)
for i:=0 to 10 do          (* Début de boucle *)
    k:=k-i;                (* Affectation *)
end_for;                  (* Fin de boucle *)
```

Automation

Les différents langages de programmation :

ST : littéral structure

Les Expressions :

Une expression est une construction qui renvoie une valeur après son évaluation. Cette valeur est utilisée dans les instructions.

Les expressions sont constituées d'opérateurs, d'opérandes et/ou d'affectations. Un opérande peut être une constante, une variable, un appel de fonction ou une autre expression.

33	(* Constante *)
ivar	(* Variable *)
fct(a,b,c)	(* Appel de fonction *)
a AND b	(* Expression *)
(x*y) / z	(* Expression *)
real_var2 := int_var;	(* Affectation, voir ci-dessous *)

Ordre des opérations

L'évaluation d'une expression passe par le traitement des opérateurs en fonction de certaines règles. L'opérateur ayant le niveau d'exécution le plus élevé est traité en premier, suivi de l'opérateur de niveau suivant, et ainsi de suite jusqu'à ce que tous les opérateurs aient été traités.

Automation

Les instructions :

- Les instructions définissent le traitement à appliquer aux expressions.

Instruction	Exemple
<u>affectation</u>	A:=B; CV := CV + 1; C:=SIN(X);
Appel d'un <u>bloc fonction</u> et utilisation de la sortie correspondante	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
<u>RETURN</u>	RETURN;
<u>IF</u>	D:=B*B; IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN C:=B; ELSE C:=D; END_IF;
<u>CASE</u>	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

Automation

Les instructions :

- Les instructions définissent le traitement à appliquer aux expressions.

<u>FOR</u>	J:=101; FOR I:=1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J:=I; 'EXIT; END_IF; END_FOR;
<u>WHILE</u>	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END WHILE;
<u>REPEAT</u>	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
<u>EXIT</u>	EXIT;
<u>CONTINUE</u>	CONTINUE;
<u>JMP</u>	label1: i:=i+1; IF i=10 THEN JMP label2; END_IF JMP label1; label2:

Automation

Les différents langages de programmation :

ST : littéral structure

Les Opérandes :

Les différents points de menu renvoient vers la documentation du constructeur sur les différents types d'opérandes disponibles, le but étant de voir comment associer des éléments de la périphérie d'entrée / sortie à ce type de programmation :

- [Constantes](#)
- [Variables](#)
- [Adresses](#)
- [Fonctions](#)

⚠ AVERTISSEMENT

BOUCLE INFINIE ENTRAÎNANT UN COMPORTEMENT IMPRÉVU DE L'ÉQUIPEMENT

Assurez-vous que la limite supérieure applicable au type de variable utilisé dans les instructions `FOR` est suffisamment élevée pour prendre en charge l'opération `<VALEUR_FIN> + 1`.

Le non-respect de ces instructions peut provoquer la mort, des blessures graves ou des dommages matériels.

Automation

Les différents langages de programmation :

ST : littéral structure

L'adressage en mode ST:

Comment doit-on procéder pour utiliser une variable de la périphérie de l'automate ?

- Adresses

Adresses directes

Présentation

Une adresse directe définie contient les informations suivantes :

- Emplacement dans la mémoire
- Format de la mémoire (taille)
- Décalage dans la mémoire. Le décalage est indiqué par un entier. Pour une adresse de bit, il est suivi d'un point, puis d'un numéro indiquant la position du bit.
- Syntaxe
- %<préfixe de zone mémoire><préfixe de taille><numéro|.numéro|.numéro...>
- Les préfixes de zone mémoire suivants sont pris en charge :

Automation

Les différents langages de programmation :

ST : littéral structure

L'adressage en mode ST:

Les préfixes de zone mémoire sont les suivants :

I	entrée (entrées physiques via le pilote d'entrée, capteurs)
Q	sortie (sorties physiques via le pilote de sortie, acteurs)
M	emplacement dans la mémoire

Les préfixes de taille suivants sont pris en charge :

X	bit unique
Aucun	bit unique
B	octet (8 bits)
W	mot (16 bits)
D	mot double (32 bits)

Automation

Les différents langages de programmation :

ST : littéral structure

L'adressage en mode ST Exemple :

Exemple d'adresse	Description
%QX7.5	bit de sortie 7.5
%Q7.5	
%IW215	mot d'entrée 215
%QB7	octet de sortie 7
%MD48	mot double à l'emplacement 48 dans la mémoire
ivar AT %IW0: WORD;	déclaration de variable contenant une attribution d'adresse Pour plus d'informations, reportez-vous au chapitre Déclaration d'une adresse AT .

Modes d'adressage par octets et par mots

Les équipements utilisent soit le mode d'adressage par octets, soit le mode d'adressage par mots.

Exemples :

Mode	Exemple
Adressage par octets	$ADR(%IW1) = ADR(%IB1)$
Adressage par mots	$ADR(%IW1) = ADR(%IB2)$

Automation

Les différents langages de programmation :

ST : littéral structure

L'adressage en mode ST :

Le deuxième élément de l'adresse de bit (le numéro suivant le point) est compris dans la plage suivante :

Adressage par octets : 0...7

Adressage par mots : 0...15

Vous pouvez configurer différemment la manière dont sont gérées les adresses de bit sur vos équipements. Le compilateur « EcoStruxure Machine Expert » saura les interpréter comme il se doit.

Exemple : sur un équipement avec adressage par octets, l'octet 2 (IB2) est adressé via %IX2.5. Sur un équipement avec adressage par mots, cette adresse correspond au mot 2, qui désigne un emplacement différent dans la mémoire.

NOTE : en l'absence d'adresse de bit unique explicite, les valeurs booléennes sont attribuées bit à bit.

Par exemple, un changement de valeur varbool1 AT %QB7 affecte la plage QX0.0 à QX0.7.

Automation

Les différents langages de programmation :

ST : littéral structure

L'adressage en mode ST :

L'illustration suivante présente un programme en langage structuré PL-7

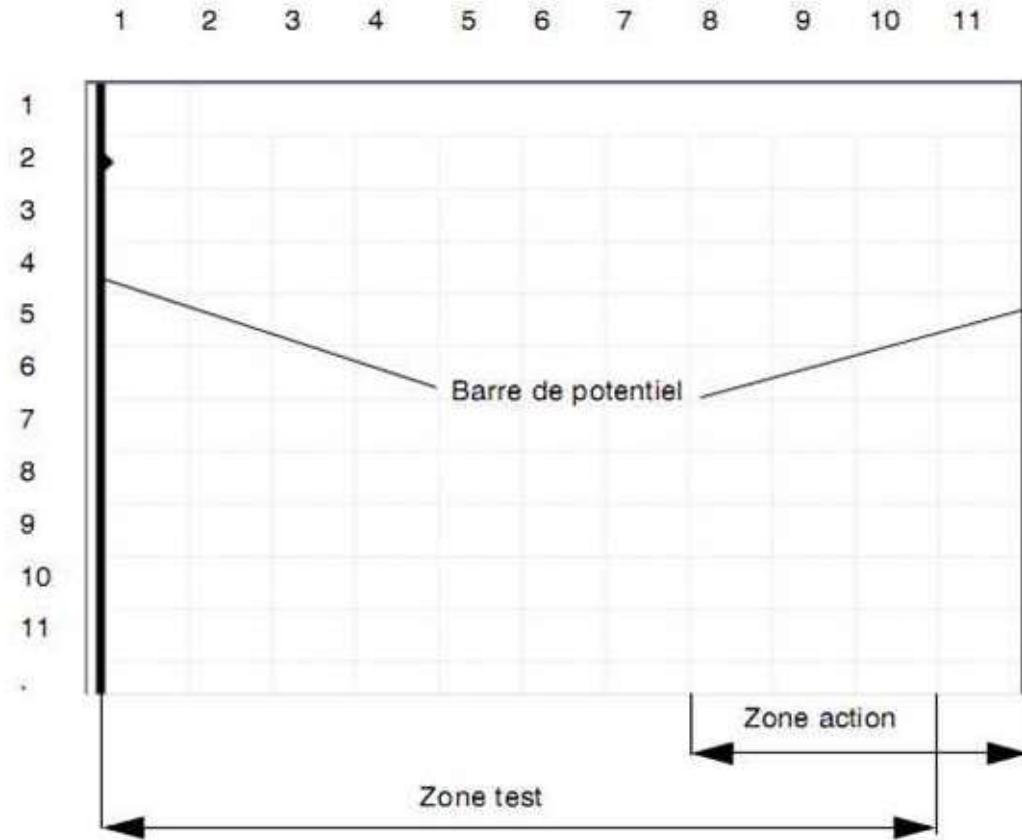
(* Recherche du premier élément non nul dans un tableau de 32 mots, détermination de sa valeur(%MW10), de son rang (%MW11). Cette recherche s'effectue si %M0 est à 1, %M1 est mis à 1 si un élément non nul existe, sinon il est mis à 0*)

```
IF %M0 THEN
    FOR %MW99:=0 TO 31 DO
        IF %MW100[%MW99]<>0 THEN
            %MW10:=%MW100[%MW99];
            %MW11:=%MW99;
            %M1:=TRUE; EXIT; (*Sortie de la boucle*)
        ELSE %M1:=FALSE;
        END_IF;
    END_FOR;
ELSE
    %M1:=FALSE;
END_IF;
```

Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

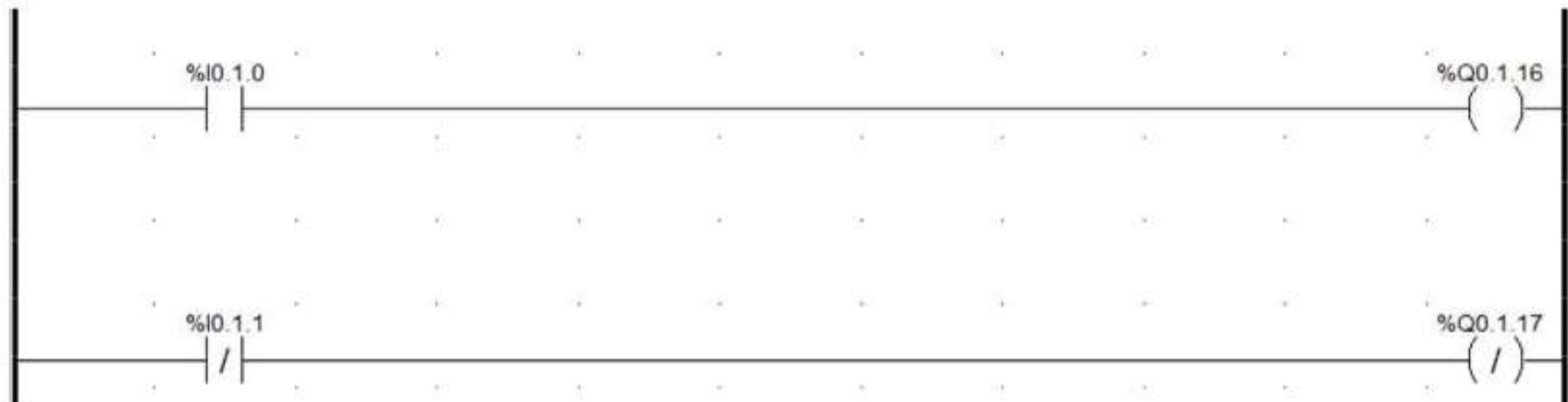


Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

2000 lignes maximum et 64 colonnes



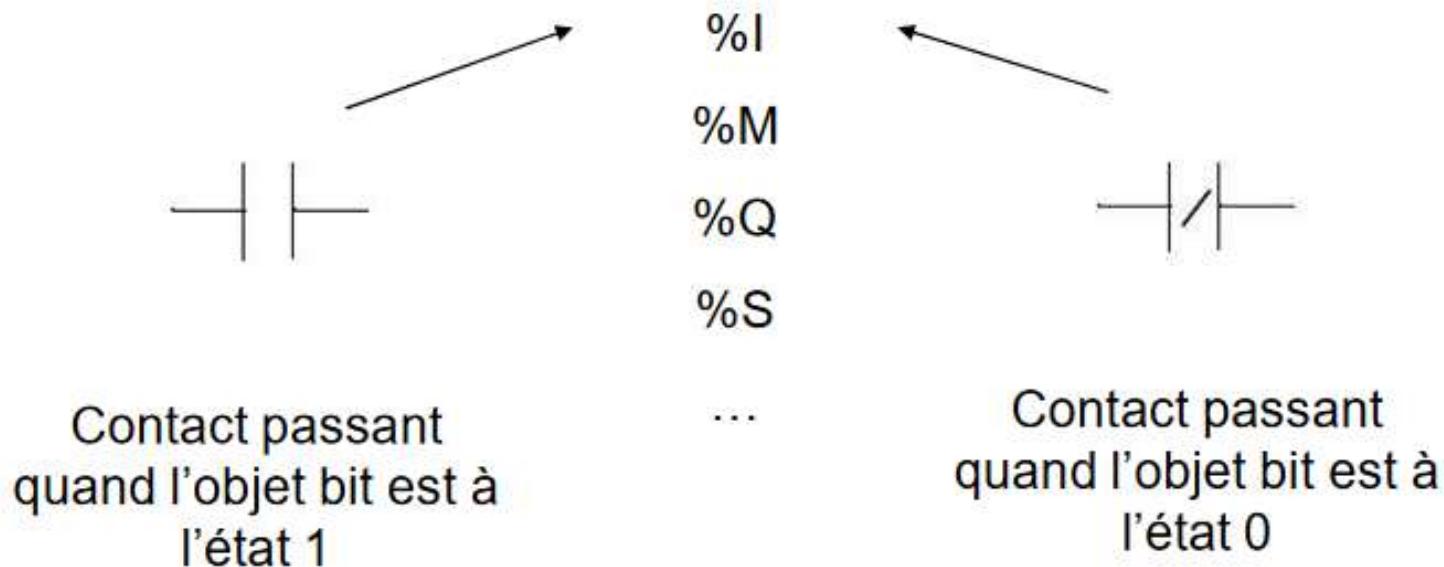
Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

Les éléments graphiques de test

Objet bit :



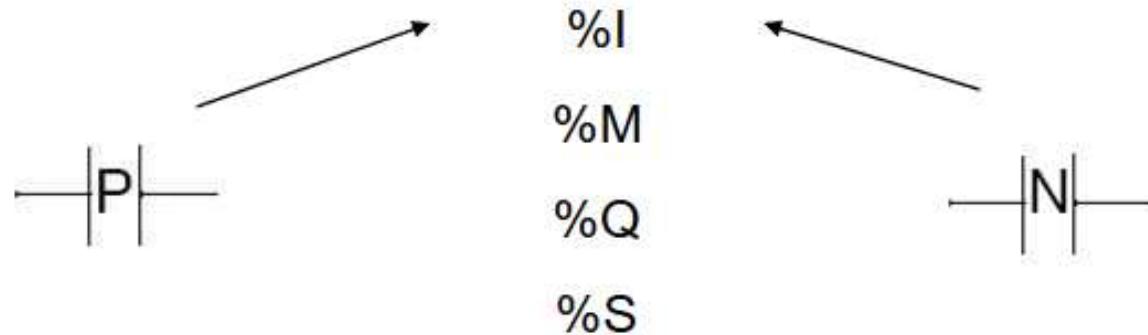
Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

Les éléments graphiques de test

Objet bit :



Front montant :
contact passant
quand l'objet bit
passe de 0 à 1.

...

Front descendant :
contact passant
quand l'objet bit
passe de 1 à 0.

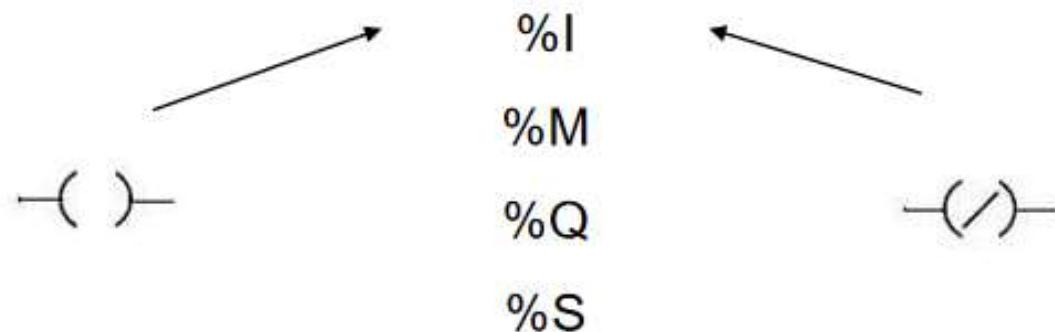
Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

Les éléments graphiques d'action

Objet bit :



L'objet bit associé prend la valeur du résultat de la zone test

...

L'objet bit associé prend la valeur inverse du résultat de la zone test

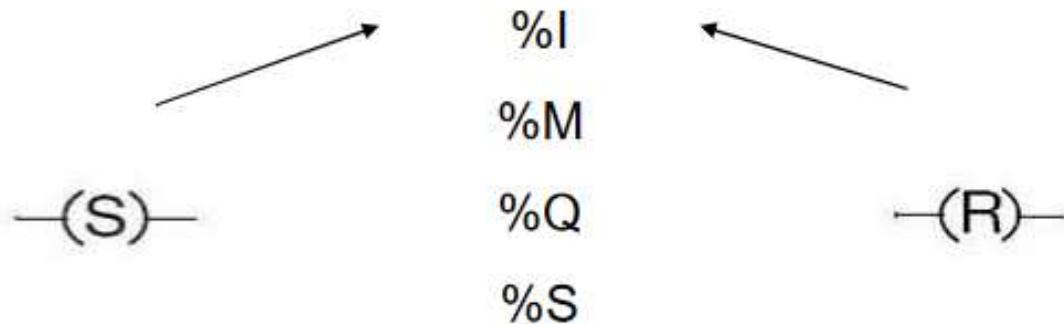
Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact

Les éléments graphiques d'action

Objet bit :



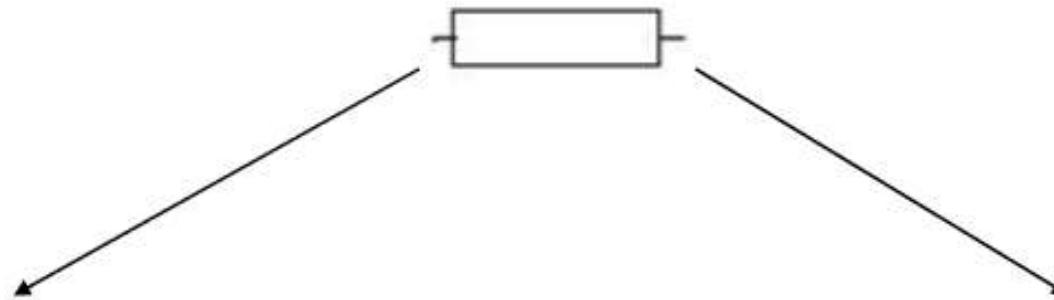
L'objet bit associé est mis et maintenu à 1 lorsque le résultat de la zone test est à 1

... L'objet bit associé est mis et maintenu à 0 lorsque le résultat de la zone test est à 1

Automation

Les différents langages de programmation :

- LD : Lader Diagram, langage à contact



- Comparaison de 2 opérandes
- Sortie passe à 1 quand le résultat est vérifié.

Réalise l'opération :
-Logique
-Arithmétique

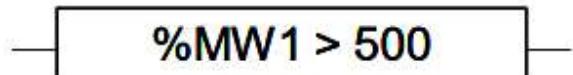
Automation

Les différents langages de programmation :

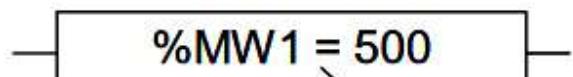
- LD : Lader Diagram, langage à contact

Exemples de blocs

COMPARAISON



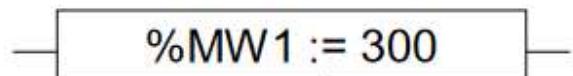
Vrai si la variable mémoire MW1 > 500



Vrai si la variable mémoire MW1 = 500

Ceci n'est pas une affectation

OPERATION



Affectation de la variable MW1 à 300

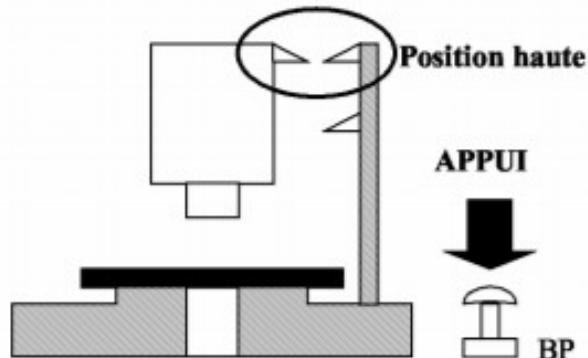
Ceci est une affectation

Automation

Les différents langages de programmation :

- SFC : grafcet

Graphe Fonctionnel de Commande Etape et Transition



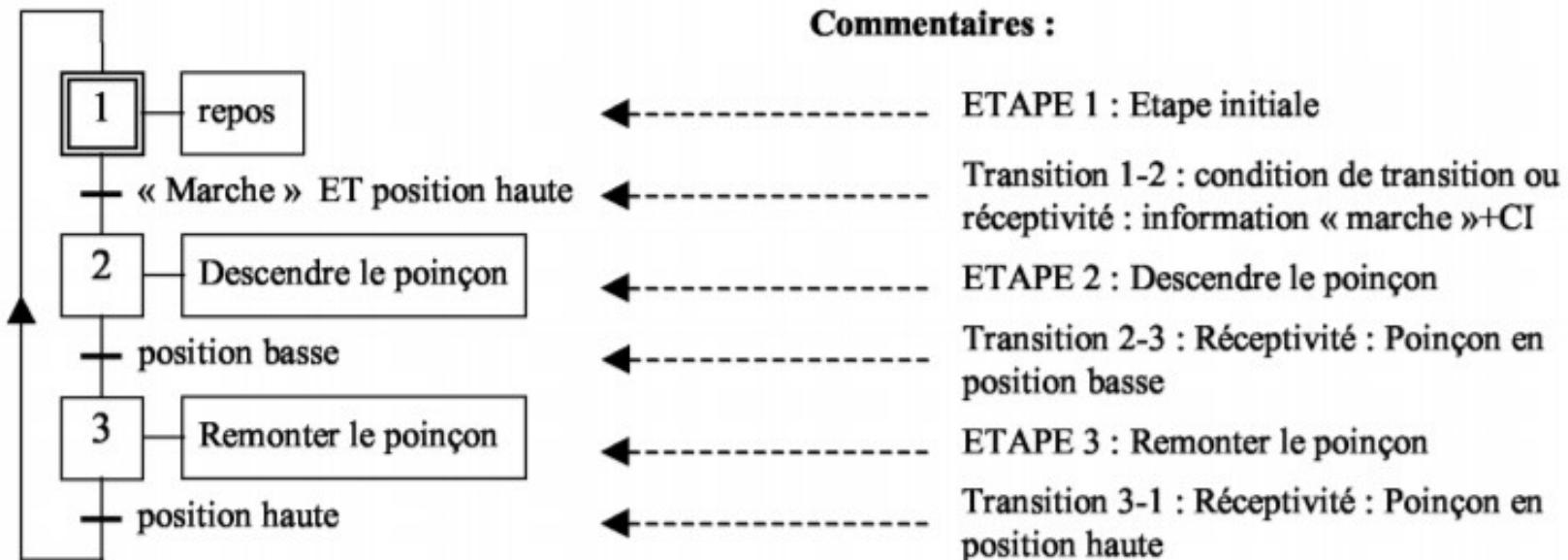
Le passage de l'état de repos à la descente du poinçon s'effectue si : **L'opérateur** fournit l'information « **marche** » par appui sur le BP Et le poinçon est en **position haute**.

Ces deux informations constituent la condition de transition de l'étape 1 à l'étape 2 : Cette condition est appelée **réceptivité** associée à la **transition T1-2**. Il est donc possible de représenter le comportement automatique de cette poinçonneuse par un **outil graphique**, à savoir **le Grafcet**. Celui-ci est basé sur la notion d'étapes auxquelles sont associées des actions, et de transitions auxquelles sont associées des **réceptivités**. Cette première représentation prend en compte uniquement la partie fonctionnelle des spécifications et donc fait abstraction de toute réalisation technologique. => Ce Grafcet est appelé «Grafcet fonctionnel » ou «Grafcet de niveau I »

Automation

Les différents langages de programmation :

- SFC : grafcet



On considère le comportement du Grafcet par un ensemble **d'actions** et de **transitions**. Le Grafcet correspond à une succession alternée d'étapes et de transitions, chaque étape est associée au comportement ou à l'action à obtenir, chaque transition est associée aux informations permettant le franchissement sous forme d'une condition logique appelée **réceptivité**.

Automation

Les différents langages de programmation :

- SFC : grafcet

Les différents Grafcet :

Il y a deux types de représentation :

- La **représentation fonctionnelle** ou de **niveau 1** donne une interprétation de la solution retenue pour un problème posé, en précisant la coordination des tâches opératives. Elle permet une compréhension globale du système.
- La **représentation technologique** ou de **niveau 2** donne une interprétation en tenant compte des choix technologique relatifs à la partie de commande de l'automatisme ; le type et la désignation des appareillages

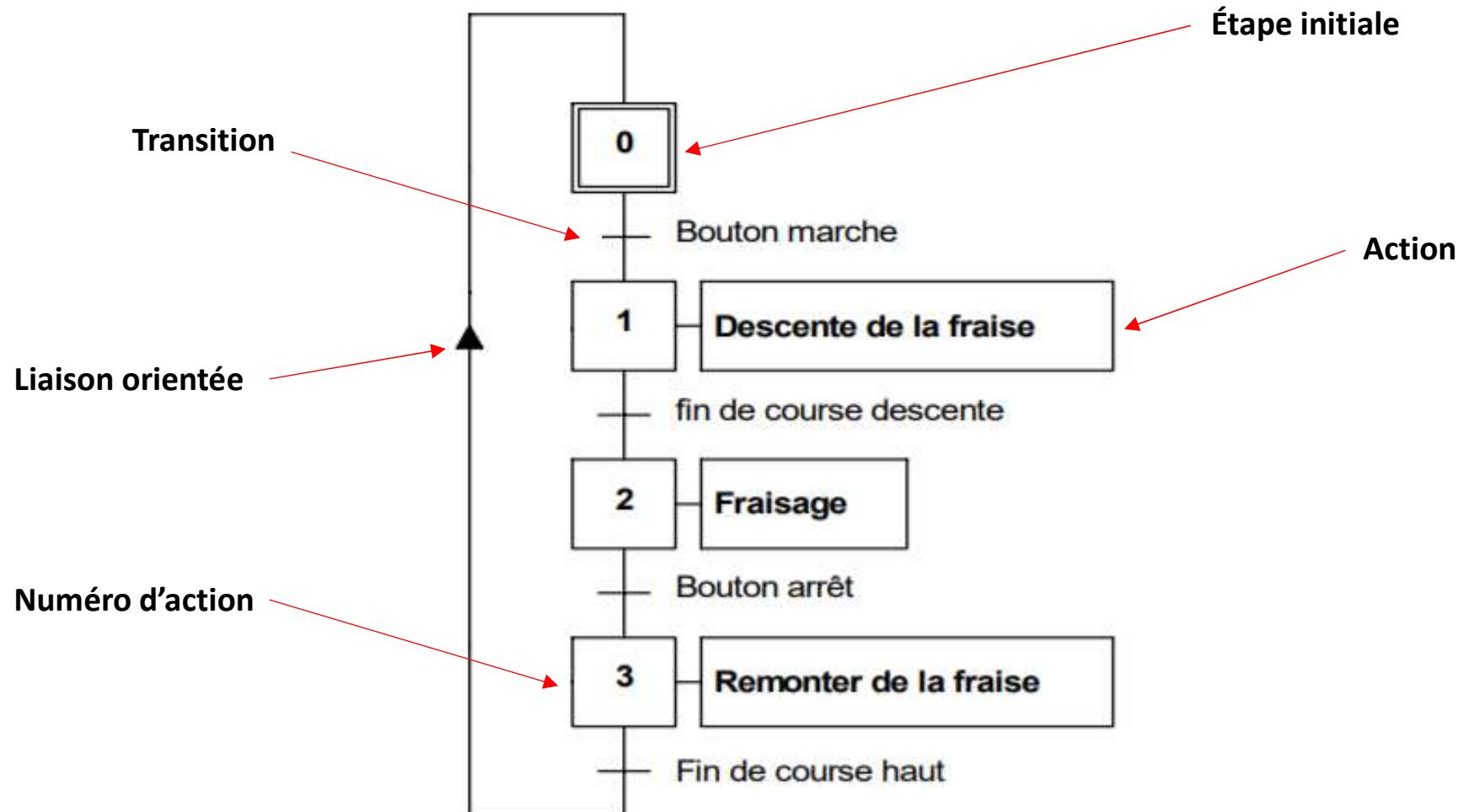
Exemple :

- Une Fraiseuse : Description du fonctionnement :
- On appuie sur le bouton marche de la fraiseuse,
- La fraise descend,
- Une fois la position basse atteinte le fraisage s'effectue,
- On appuie sur le bouton arrêt,
- Le fraisage s'arrête et la fraise remonte,
- Une fois le fin de course haut atteint la fraiseuse est en position initiale.

Automation

Les différents langages de programmation :

Graphique :

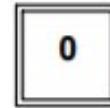


Automation

Les différents langages de programmation :

Description des étapes graphiques :

Etape initiale : L'étape initiale caractérise l'état du système au **début du fonctionnement**.



Etape : Une étape correspond à un comportement stable du système.

Les étapes sont numérotées dans l'ordre croissant.

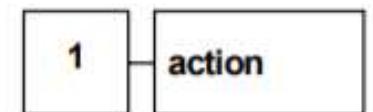
A chaque étapes on peut associer une ou plusieurs actions.



Transition : Les transitions indiquent les possibilités d'évolutions du cycle, à chaque transition est associée à une **réceptivité**.

Réceptivité : La **réceptivité** est la condition logique pour l'évolution du grafcet. Si la réceptivité est **vrai (=1)** le cycle peut évoluer. Les réceptivités proviennent du système de commandes, des fins de courses ou d'informations provenant de la partie opérative de l'installation.

Action : L'action est associée à une étape, elle est active lorsque le cycle est arrivé sur l'étape. Il est possible de définir les actions conditionnelles, temporisé (electrovanne, contacteur, marche d'un tapis, enclenchement d'une sortie, ...)



Liaisons orientées : Le Grafcet se lit de haut en bas, autrement il est nécessaire d'indiquer son évolution avec des liaisons orientées constituées de flèche indiquant le sens.



Automation

Les différents langages de programmation :

Quelques règles :

Situation initiale

- Un grafcet commence par une étape initiale qui représente la situation initiale avant évolution du cycle.

Franchissement d'une transition

- Une transition est soit validée ou non validée ; elle est valide lorsque toutes les étapes immédiatement précédentes sont actives. Lorsque la transition est valide et que la réceptivité associée est vraie elle est alors obligatoirement franchie.

Évolution des étapes actives

- Le franchissement d'une transition entraîne l'activation des étapes immédiatement suivante et la désactivation des étapes immédiatement précédentes.

Transitions simultanées

- Plusieurs transitions simultanément franchissables sont simultanément franchies.

Activation et désactivation simultanées

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

La durée de franchissement d'une transition ne peut jamais être rigoureusement nulle, même si elle peut être rendue aussi petite que l'on veut. Il en est de même pour la durée d'activation d'une étape.

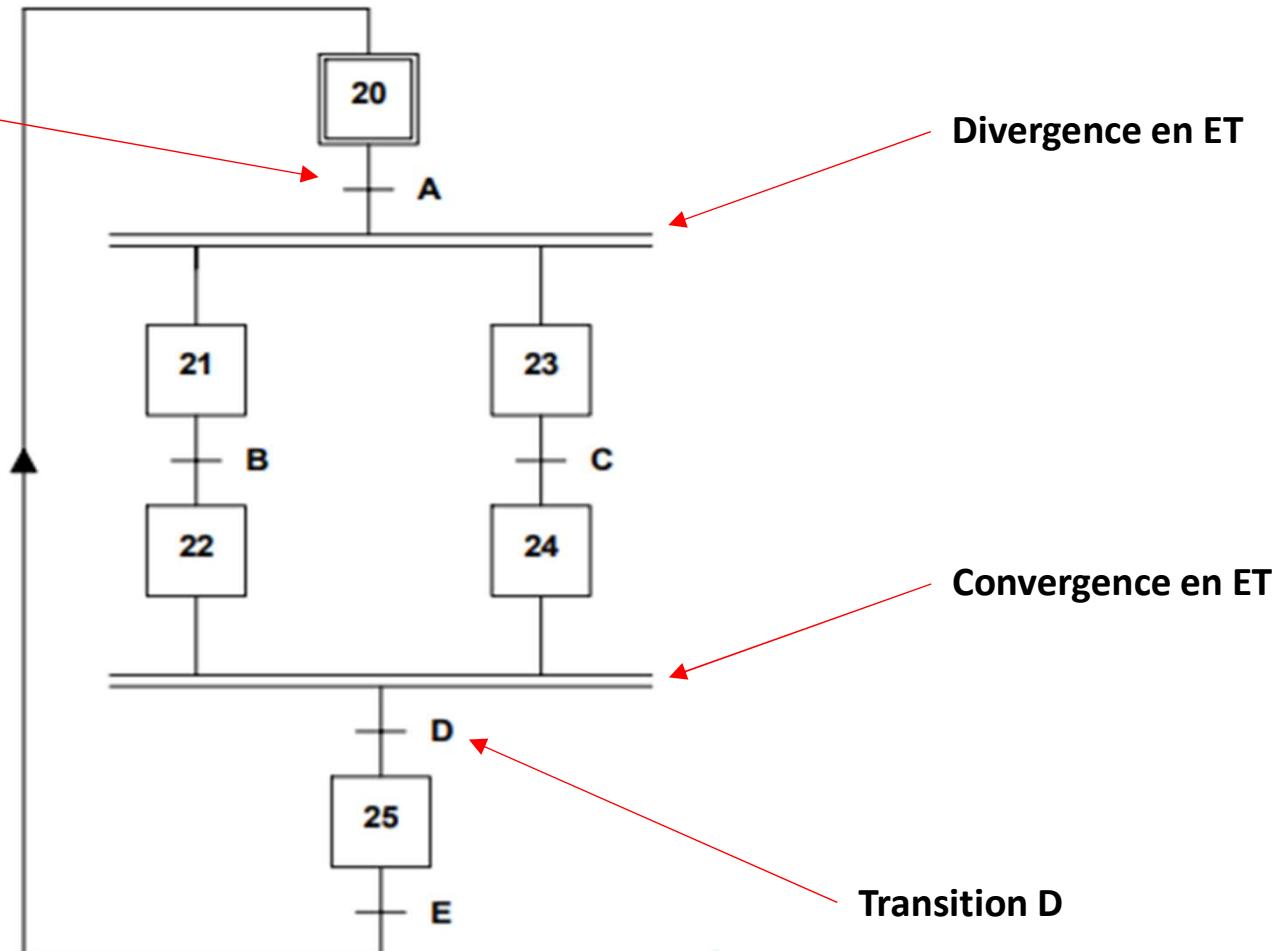
Automation

Les différents langages de programmation :

Divergence et convergence en ET :

Transition A

Divergence en ET : représentation par 2 trait identique et parallèle ; lorsque la transition A est franchie les étapes 21 et 23 sont actives.
Convergence en ET : La transition D sera active lorsque les étapes 22 et 24 seront actives, si la réceptivité associé à la transition D est vraie alors elle est franchie et l'étape 25 devient active et désactive les étapes 22 et 24. Le nombre de branche peut être supérieur à 2, **après une divergence en ET on trouve une convergence en ET.**



Automation

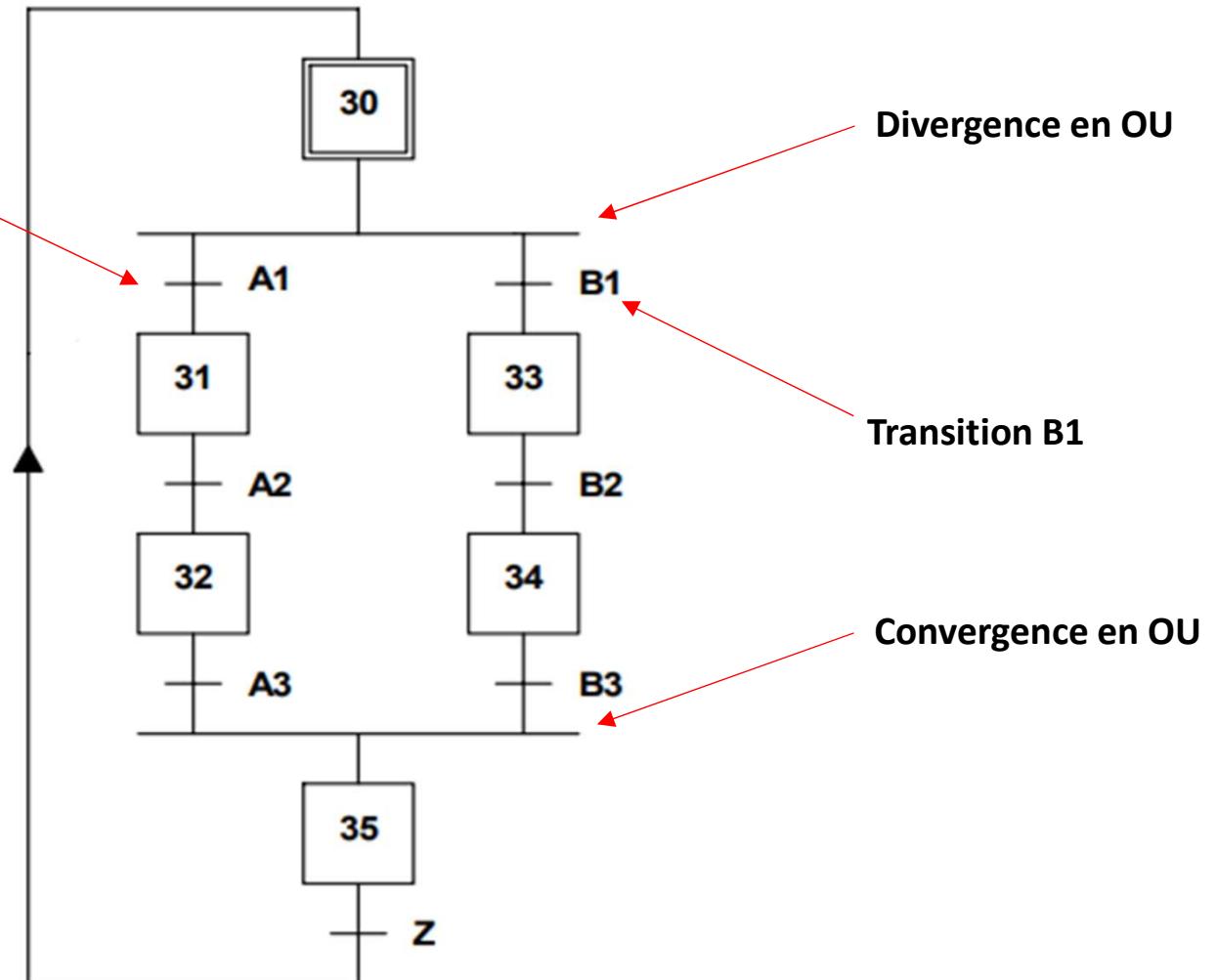
Les différents langages de programmation :

Divergence et convergence en OU :

Transition A1

Divergence en OU : l'évolution du système se dirige vers une des branches en fonction des réceptivités A1, B1 et de leurs transitions associées.

Convergence en OU : Après une divergence en OU on trouve une convergence en OU vers une étape commune dans l'exemple l'étape 35. Le nombre de branche peut être supérieur à 2, A1 et B1 peuvent être vrais simultanément.



Automation

Les différents langages de programmation :

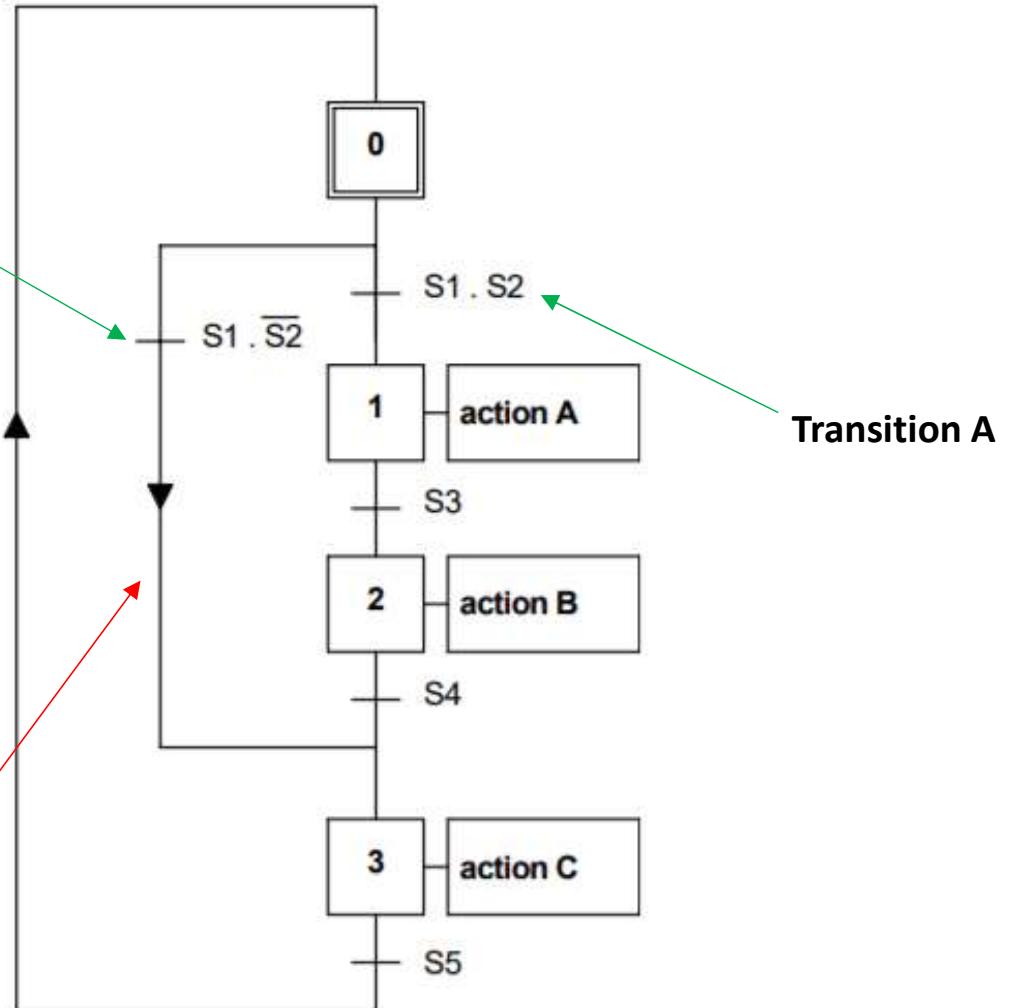
Saut d'étape :

Transition B

Le **saut d'étape** permet de sauter une ou plusieurs étapes en fonction de la progression d'un cycle.

Sur le grafct ci-dessus après l'étape initiale 0 un choix entre 2 transitions A et B s'effectue ; La transition A associé à sa réceptivité nous permet de continuer le cycle sur l'étape 1, La transition B associé à sa réceptivité nous permet de passer à l'étape 3, les étapes 1 et 2 sont ignorées lors du cycle.

Saut d'étape



Automation

Les différents langages de programmation :

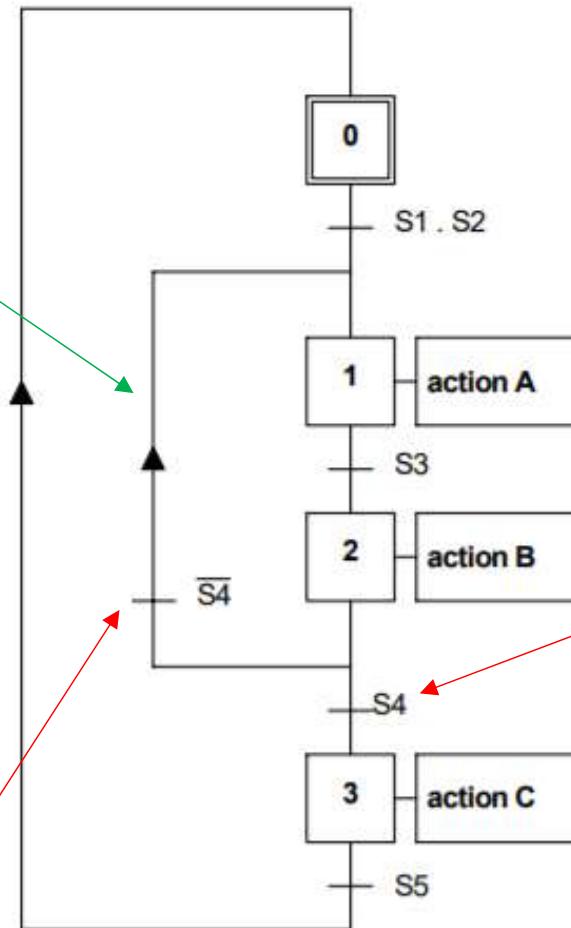
Reprise d'étape :

Reprise d'étape

La **reprise d'étape** permet de ne pas continuer le cycle mais de reprendre une séquence précédente lorsque les actions à réaliser sont répétitives.

Sur le grafcet ci-dessus après l'étape 2 un choix entre 2 transitions A et B s'effectue ; La transition A associé à sa réceptivité nous permet de reprendre le cycle sur l'étape 1, La transition B associé à sa réceptivité nous permet de passer à l'étape 3..

Transition A



Automation

Les réseaux en automation :

Les différents types de réseaux :

On retrouve le réseau **TCPIP classique** avec, pour certains automates, une modification sur le protocole IP classique qui permet de garantir à l'envoyeur que le destinataire a bien reçu l'information.

Il existe également **les réseaux de terrain** pour permettre l'envoi de données vers les actuateurs et la collecte des données venant des différentes cartes d'acquisition. Par exemple on trouve souvent le **réseaux Profibus, Bus CANopen**

Pour la sécurité des personnes, on utilise **un réseau de couleur jaune** qui est entièrement déconnecté de la partie automate et des autres réseaux. Ce dernier peut donner une commande immédiate d'arrêt à une machine et également envoyer des informations vers les automates pour passer l'installation en mode sécurité ou tout simplement mettre une installation à l'arrêt. Ce type de réseau est dénommé **ASI BUS**.

Ces différents réseaux vont être choisis en fonction de la **topologie de terrain**, des **distances** entre les différents éléments, de la complexité de l'installation et d'autres particularités comme les **conditions d'utilisations** liées à l'environnement de travail de l'installation (température, humidité, poussière, danger d'explosion, ...).

Nous allons regarder dans un premier temps les différentes topologies, réseau en **étoile**, en **maille**, **point à point**, **arbre**, **en bus**, **en anneau**.

Automation

Les réseaux en automation :

Les différents types de réseaux :

Les éléments qui constituent le réseau :

Gateway :

Unité fonctionnelle qui permet l'interconnexion de deux réseaux d'architecture différente.

Routeur :

Il crée une segmentation logique de réseaux. Il assure le passage de l'information entre deux sous-réseaux logiques distincts en choisissant le meilleur chemin. C'est la couche réseau qui assure ce routage. Il n'est pas transparent, il faut donc l'adresser pour le traverser.

Switch :

Il transmet les données reçues sur un port, seulement vers le port sur lequel la station destinatrice est connectée. Il assure la prolongation du support au delà des limites en distance du standard (segment) en réalisant une remise en forme des signaux. Il supprime les collisions et les paquets non valides et réduit la charge moyenne sur le réseau entier. Synonyme : Bridge.

Hub :

Les données reçues sur un port sont envoyées à tous les autres ports. Le hub ne possède pas de mémoire interne et diffuse les collisions ; plus il y a d'équipements, plus il y a de collisions et plus la charge est importante.

Automation

Les réseaux en automation :

Les différents types de réseaux :

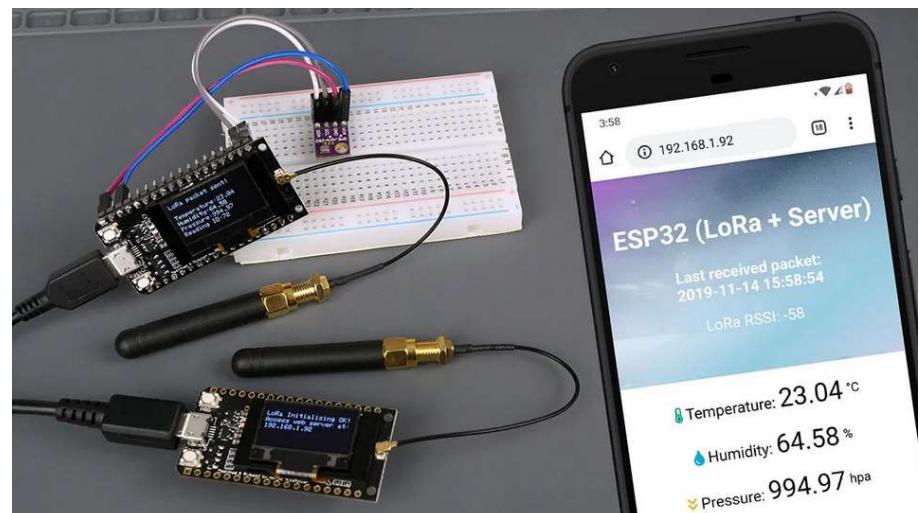
Réseau **point à point**



Il n'est plus utilisé aujourd'hui que dans des cas très particuliers, lorsqu'il est nécessaire de relier deux machines entre elles. On peut utiliser le protocole RS232 ou 485, mais également d'autres. Ce type de communication refait surface dans le cadre des transmissions de données pour des projets isolés où il n'y a que l'émetteur et le récepteur. On peut avoir une communication unidirectionnelle ou bidirectionnelle. On trouve aujourd'hui en IoT ou industrie 4.0, toute une série de nouveaux protocoles comme : Lora, SegBee, Bluetooth, Wifi, SigFox, ... et bien d'autres.

Un exemple du monde de l'IOT :

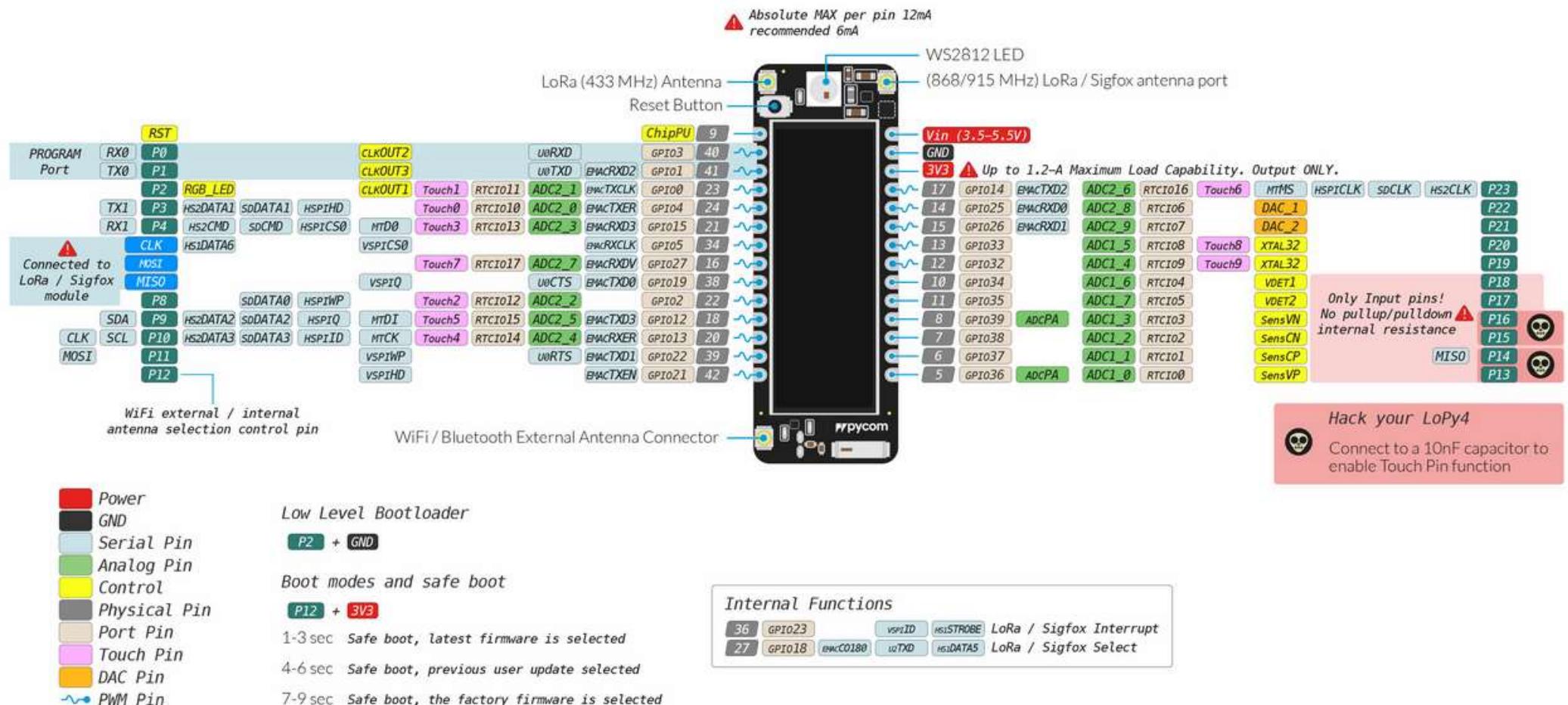
LOPY 4



Automation

Les réseaux en automation :

Un exemple du monde de l'IOT : **Lopy 4**



Automation

Les réseaux en automation :

Les différents types de réseaux :

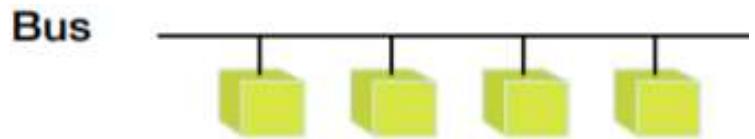
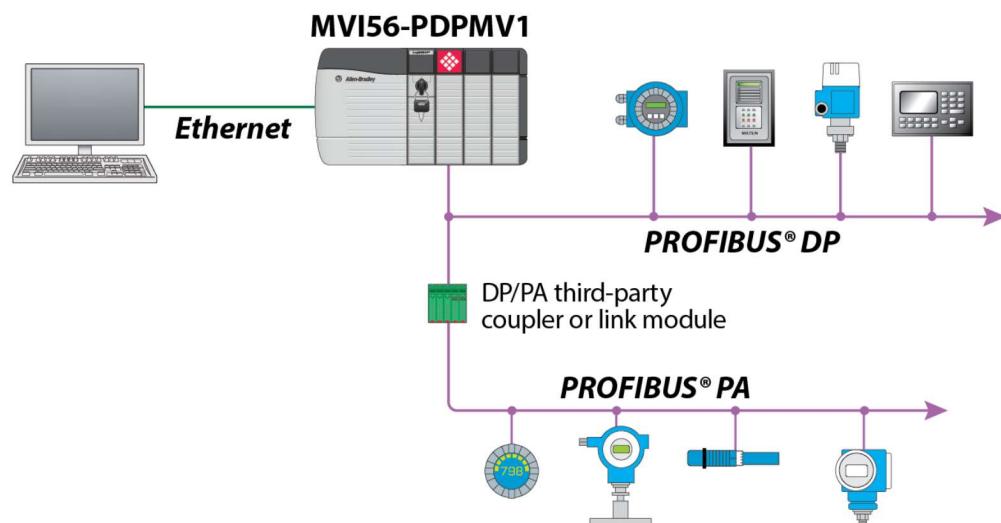
Réseau Bus

C'est la topologie la plus commune aux LAN (Local Access Network) car la plus économique. Chaque nœud est raccordé au bus par l'intermédiaire de modules de dérivation actifs ou passifs suivant le protocole de réseau.

C'est également souvent employé dans les bus dédié contrôle / commande des installations. On y trouve des particularités au niveau des câbles, d'un point de vue environnement.

Comme par exemple des bus ayant des niveaux d'étanchéité élevé, mais également un nombre de partenaires important avec des technologies de mise en œuvre aisée pour l'installateur ou le programmeur.

Exemple : Profibus



PA → Process Automation.

PROFIBUS PA utilise le même protocole que **PROFIBUS DP**, il s'agit simplement d'un profil qui permet de standardiser des caractéristiques et comportement d'appareils standard. **DP** est utilisé pour Périphérie Déportée.

Automation

Les réseaux en automation :

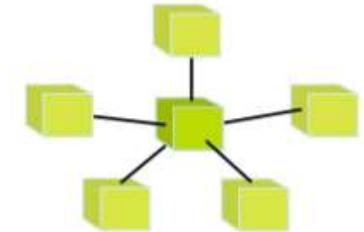
Les différents types de réseaux :

Réseau en étoile

Une **approche ancienne**, non normalisée, qui correspondait à une organisation de machines esclaves **reliées par liaisons point à point à une machine maître**. Aujourd’hui cette topologie correspond soit au câblage par étoile passive comme **un Hub** (équipement statique qui simule un réseau local auquel sont raccordés des abonnés par dérivation, certains hubs pouvant être raccordés à un réseau principal).

Le nœud central n'est pas un abonné du réseau mais une unité de distribution dont le bon fonctionnement est indispensable à la communication. Cette dernière correspond à une partie de réseau (exemple des **PC familiaux rattachés à un serveur Internet** ou à un réseau local de stations connectées à un serveur), dans la mesure où les équipements sont peu nombreux, car le **coût de câblage** devient **vite prohibitif** et les performances ne sont pas élevées. De plus le hub ne possède aucune intelligence, il envoie les informations à l'ensemble des partenaires en même temps.

Etoile



Automation

Les réseaux en automation :

Les différents types de réseaux :

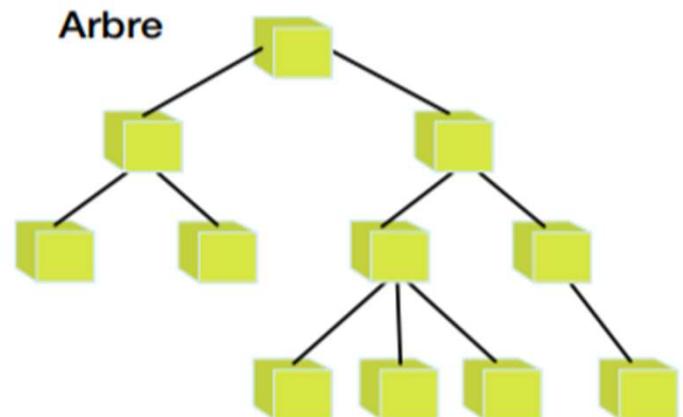
Réseau en arbre

Chaque nœud peut être un abonné ou un switch. C'est une variante de la topologie en étoile avec les mêmes faiblesses.

Cette architecture revient en force avec les techniques du type Ethernet en fibre optique **10 base F**.

Le réseau est découpé en tronçons reliés par des ponts qui filtrent les trames en fonction du destinataire diminuant ainsi les collisions.

On peut associer des éléments qui vont permettre de filtrer les communications à chaque coupleur, afin d'éliminer au maximum les collisions lors des transmissions.



Automation

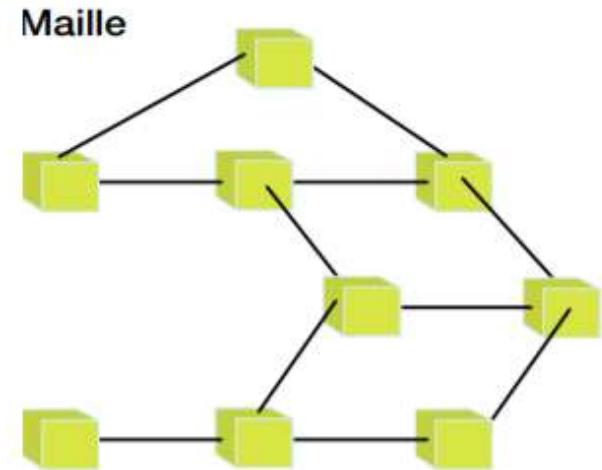
Les réseaux en automation :

Les différents types de réseaux :

Réseau en maille

C'est la forme classique d'organisation **WAN** (World Access Network). Les équipements sont reliés entre eux pour former une toile (Web) d'araignée (comme Internet). Chaque **nœud** a un **rôle de routeur**. Pour atteindre un nœud, les chemins sont multiples et choisis en fonction de critères comme la **disponibilité** d'un nœud ou d'un WAN, la **qualité** de transmission ou la **charge** ponctuelle sur un tronçon WAN donné.

C'est un réseau qui permet beaucoup plus de souplesse à l'utilisation et avec des performances supérieures. On retrouve souvent un mélange dans le type de réseau et de câble utilisé.



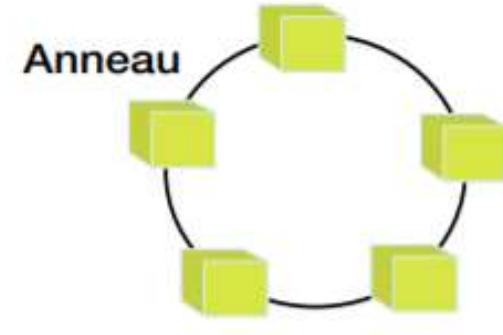
Automation

Les réseaux en automation :

Les différents types de réseaux :

Réseau **en anneau** :

Réseaux Locaux Industriels **CAN - TELWAY - TOKEN RING** (IBM)



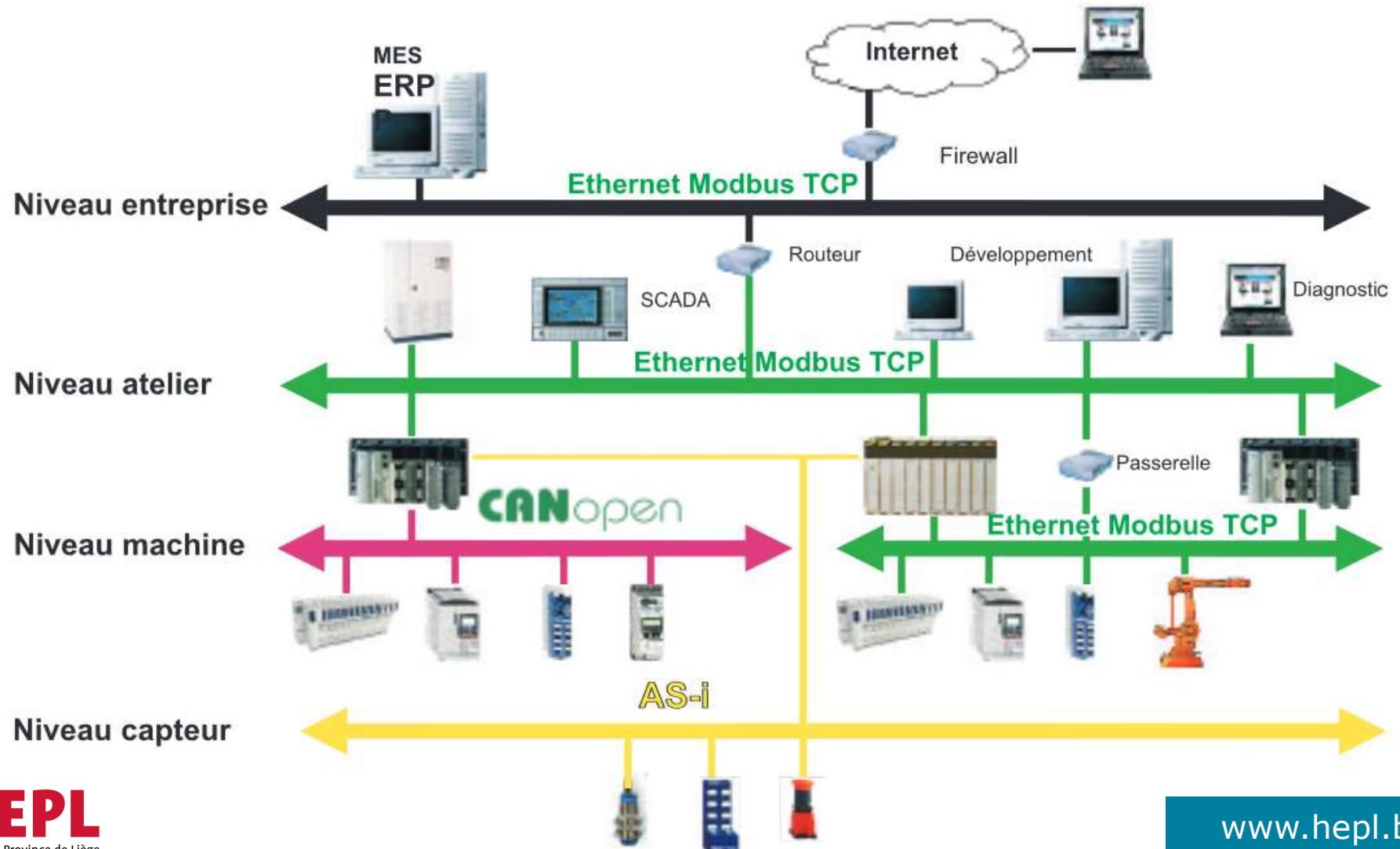
Chaque nœud est relié à ses voisins pour former une boucle fermée et a **un rôle actif** dans la propagation des échanges. Cette structure est bien adaptée aux LAN, en particulier si l'on recherche la disponibilité. Une rupture de liens entre deux nœuds peut être gérée pour garantir la communication. Chaque nœud a la possibilité de régénérer le signal.

La structure se prête facilement à l'utilisation de la **fibre optique**, les distances de couverture pouvant être grandes.

Automation

Les réseaux en automation :

Les différents types de réseaux : Exemple



Automation

L'automate : Structure de programmation

Rappel : Le **temps de scrutation** est défini comme étant le temps nécessaire à l'automate pour réaliser l'ensemble des quatre opérations. Ce temps est de l'ordre de la **dizaine de ms** pour les applications standards.

Deux types de structure logicielle sont possibles :

Une structure mono tâche : Le traitement se fait de façon normale. Le programme n'est composé que de la tâche maître ou « Mast ».

Une structure multi tâche : On ajoute deux autres tâches à la tâche maître : la **tâche rapide** et la **tâche événementielle**. La tâche rapide est périodique (paramétrable) pour laisser le temps à la tâche maître de s'exécuter et la tâche événementielle est prioritaire sur les autres tâches.

PRIORITE CROISSANTE

Tâche maître

Tâche rapide

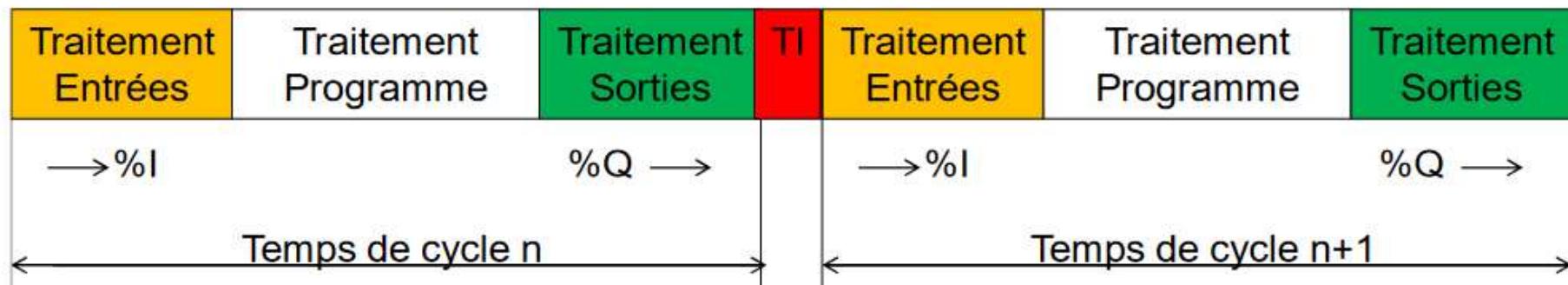
Tâche événementielle

Automation

L'automate : Structure mono tâche

Une seule tâche MAST c'est le Cyclique (par défaut). Il est périodique (inférieur à 255 ms). Surveillance du temps de cycle par « Watch dog » , ce dernier est configurable par l'utilisateur.

Dans ce mode, vous avez plusieurs façons de rédiger votre programme. Nous allons considérer l'utilisation du langage à contact (appeler « Ladder » chez Siemens). Il est important de remarquer que l'on ne trouve dans la tache Mast, que l'appel aux différentes taches secondaires de type « SR » pour l'automate M340.



TI, correspond au temps de traitement interne.

Automation

L'automate : Structure mono tâche

Dans ce type de programmation, on concentre l'ensemble du programme dans la tâche « **MAST** » et on découpe la programmation en sous routines « **SR** » par exemple si le choix de programmation est le « **CONTACT** ».

Il est important de ne pas oublier qu'il ne faut pas faire de double affectation des sorties, tenant compte que c'est la dernière affectation qui sera prise en compte.

Nous trouverons également différentes parties de code répartie dans les « **SR** », comme en programmation classique. On peut comparer cette façon de faire à la création d'un programme structuré utilisant un code principal et différentes fonctions.

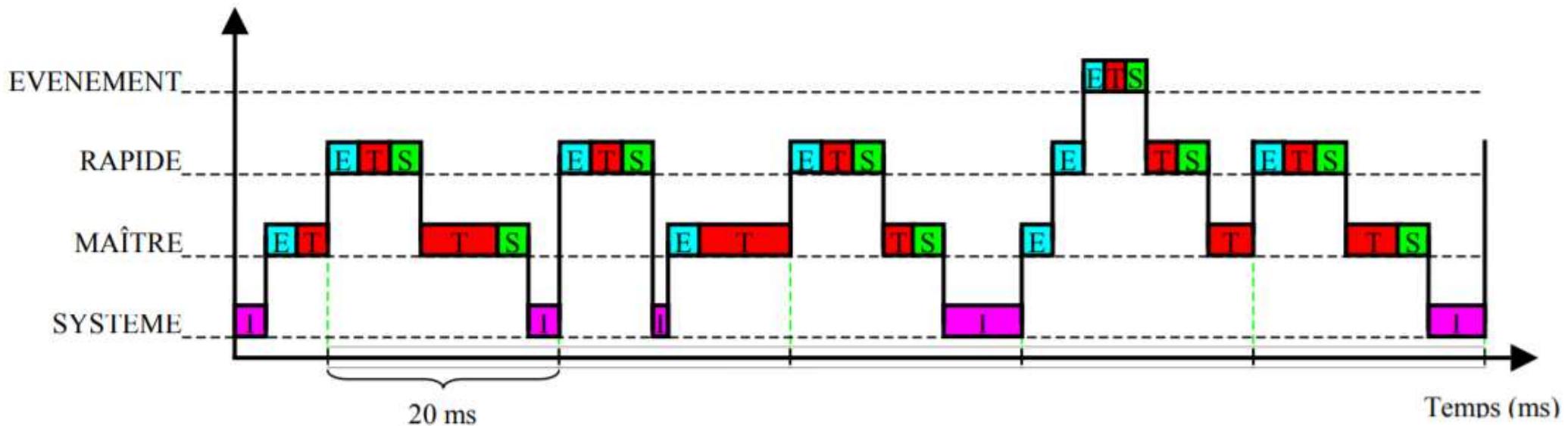
Suivant les besoins du process, il sera nécessaire d'utiliser d'autres particularités de la programmation, il sera alors possible de choisir des parties de programmation utilisant la tâche « **FAST** » par exemple.

Cette façon de faire est décrite comme étant une programmation « multi tâche », nous allons voir comment elle est représentée en diagramme « temporel ».

Automation

L'automate : Structure multi tâche

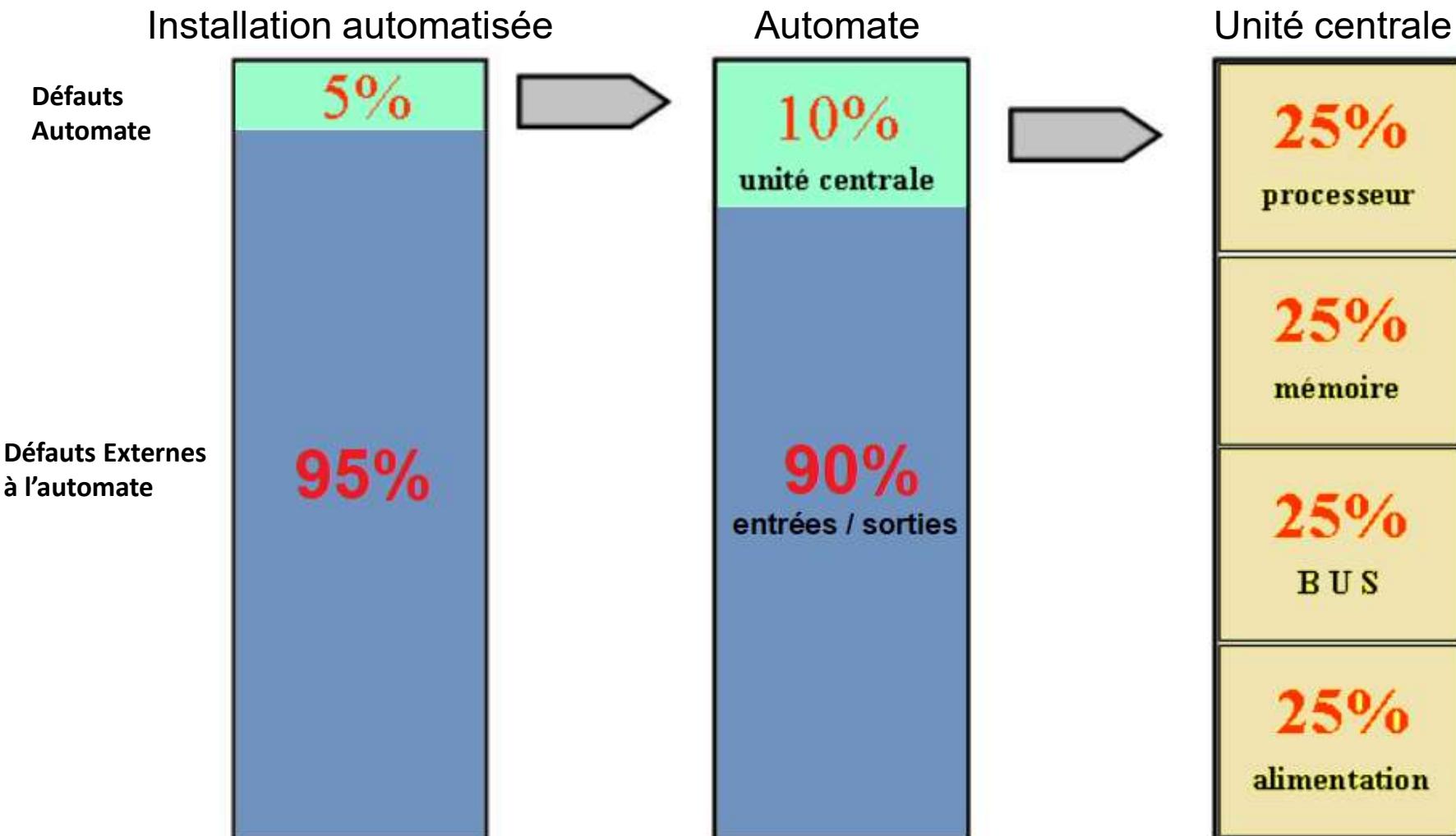
Pour la programmation multi tâche, on parle de « MAST » et « FAST », c'est-à-dire tâche « maître » et « tâche rapide », sur le diagramme temporel, on peut observer que les tâches sont sérialisées, donc cette notion « multi » ne veut pas dire qu'il y a exécution en même temps.



La périodicité de la tâche rapide est ici fixée à 20ms. Il faudra veiller aux temps de cycle de la tâche maître. Il ne faut pas dépasser le temps global fixé dans le « Watch Dog ».

Automation

L'automate : FIABILITE, SECURITE, DISPONIBILITE



La fiabilité d'un API se mesure à partir du taux de défaillance de chacun de ses constituants.

Automation

L'automate : FIABILITE, SECURITE, DISPONIBILITE

D'après une étude des constructeurs, **95% des défauts** survenant dans une installation automatisée **sont d'origine externe à l'automate**. Parmi les **5% restants**, **90%** concernent les éléments **d'entrées/sorties**. Les **10% restants** sont affectés à l'**unité centrale** et se répartissent de façon à peu près uniforme entre le **microprocesseur**, la **mémoire**, le **Bus**, **l'alimentation**. Cette analyse montre que :

- l'unité centrale de l'API est fiable et présente un haut niveau de disponibilité.
- que pour augmenter la disponibilité de l'installation dans son intégralité, il faut chercher, par des fonctions de diagnostics, aux **95% des pannes occasionnées par les défauts extérieurs**.

Ce constat montre que dans le cadre de la programmation sur automate, il faut avoir une réflexion différente de la programmation habituelle. Si, en mode simulation, l'automate ne passe plus en « STOP », on peut considérer que la programmation est correcte. Une fois que le programme est transféré dans l'unité centrale, si des défauts apparaissent, il faut orienter les recherches sur le matériel :

- Vérifier les connexions des entrées / sorties
- Vérifier l'état des éléments connectés sur les entrées (capteurs)

Automation

L'automate : FIABILITE, SECURITE, DISPONIBILITE

Disponibilité

La disponibilité est définie comme la rapport entre la somme des temps de production et du temps total pouvant être selon les cas considérée comme:

temps de mise à disposition

temps d'engagement

temps normal de production La disponibilité correspond à l'aptitude du système à réaliser sa fonction en permanence.

L'Automate est fiable et présente un **haut niveau de disponibilité**

On va mesurer la disponibilité du matériel « automate » en fonction de valeurs, ces dernières sont les suivantes :

« **MTBF** » ou Mean Time Between Failure = **temps moyen entre deux pannes**.

« **TBF** » ou Time Between Failure = **temps entre deux pannes**.

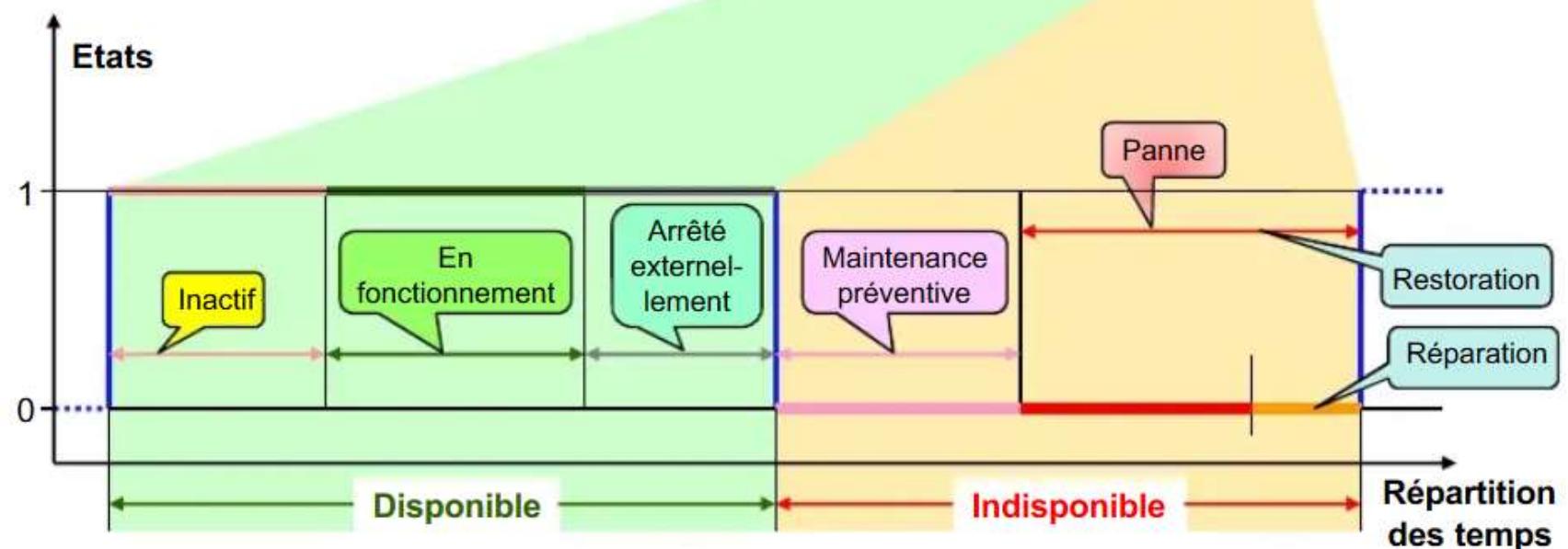
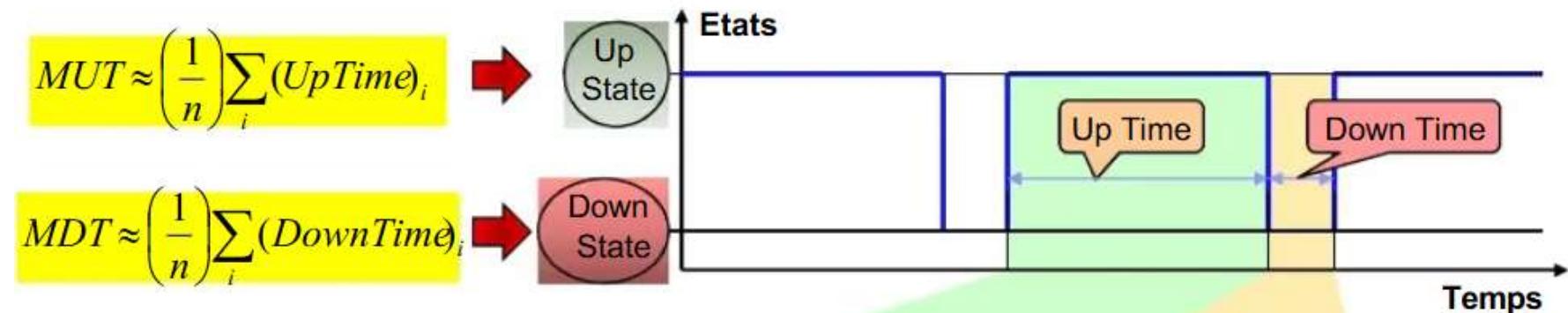
« **MTTF** » ou Mean Time To Fail = **Temps moyen avant une panne**.

« **MUT** » ou Mean Up Time = **Temps moyen de disponibilité**.

« **MDT** » ou Mean Down Time = **Temps moyen d'indisponibilité**.

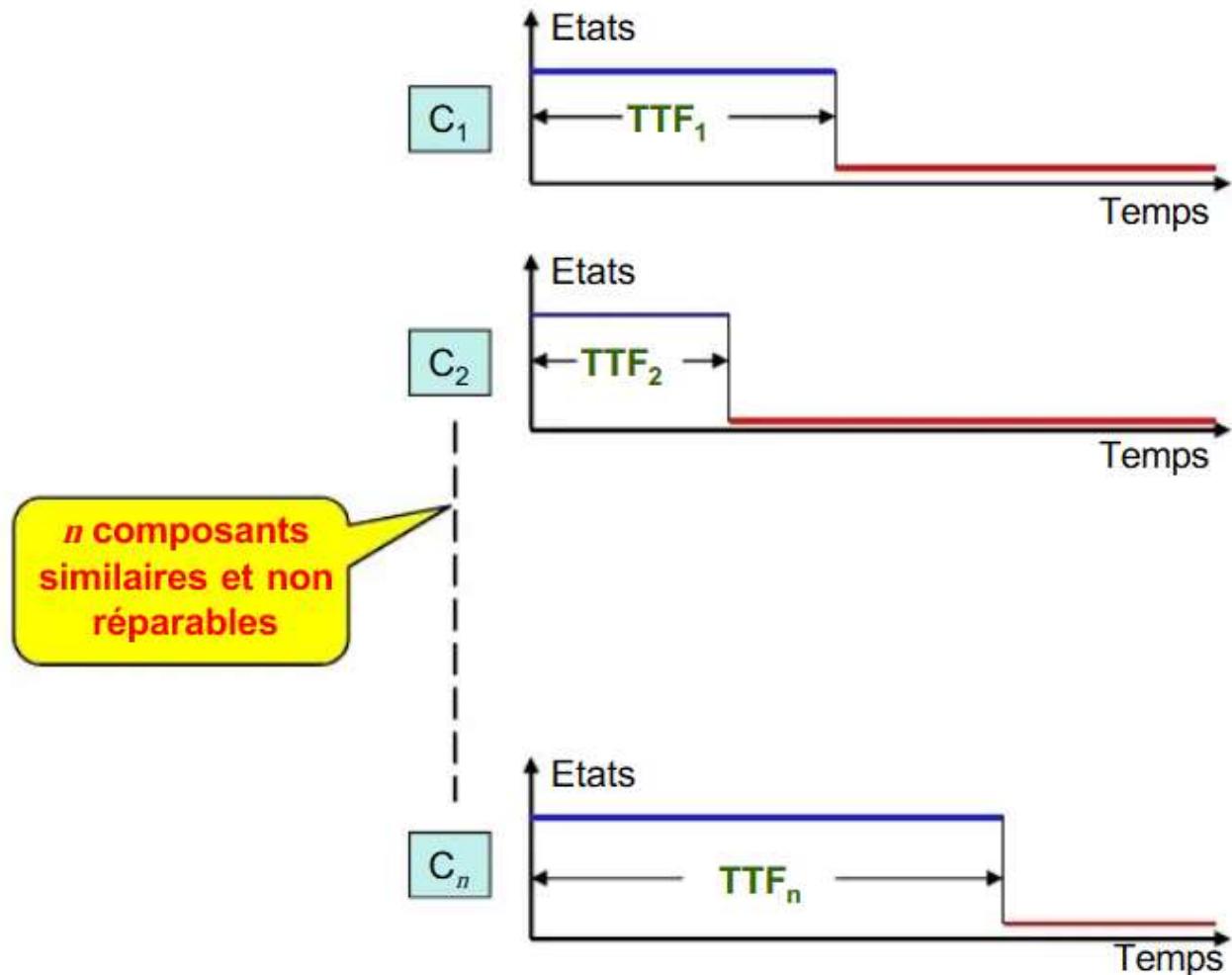
Automation

L'automate : MUT, MDT



Automation

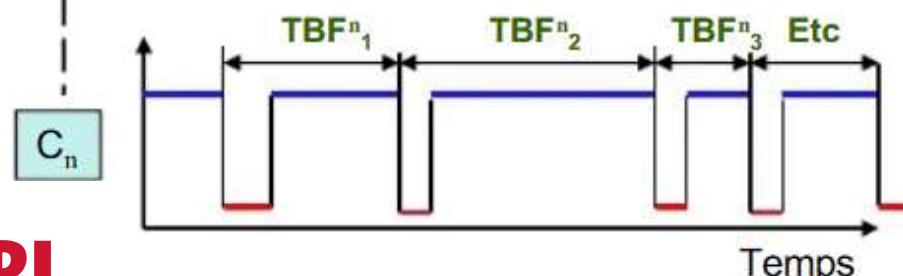
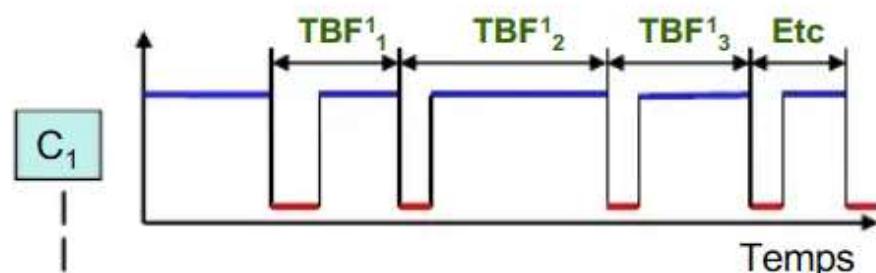
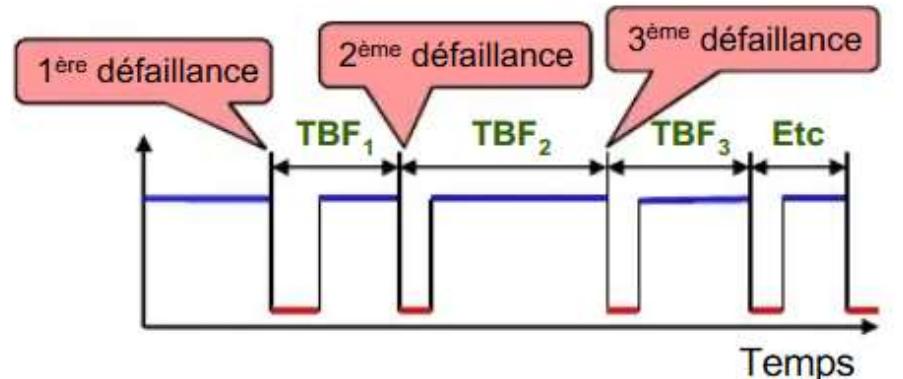
L'automate : MTTF



$$MTTF \approx \left(\frac{1}{n} \right) \sum_i (TTF)_i$$

Automation

L'automate : TBF, MTBF



n TBF_i relevés sur une entité réparable

$$MTBF \approx \left(\frac{1}{n} \right) \sum_i (TBF)_i$$

n entités similaires et réparables : relevés sur l'entité (i)

$$MTBF \approx \frac{1}{\sum n_i} \sum_i \sum_j TBF_j^i$$

Automation

L'automate : modes reprise à chaud, reprise à froid.

Nous avons parlé de cycle automate, il comprenait l'acquisition des entrées dans la table MIE (mémoire image des entrées), l'exécution du cycle de l'automate avec mise à jour de la table MIS (mémoire image des sorties) et en fin de cycle, l'ensemble des données est transférée vers les sorties en une fois.

Ne pas oublier que le fait d'avoir un cycle n'autorise pas d'affecter plusieurs fois une même sortie, car uniquement la dernière affectation sera prise en compte!

Si, un problème matériel, logiciel (programmation) ou tout simplement une perte du réseau électrique vous avez un arrêt de l'automate, il faudra tenir compte de différents éléments pour pouvoir re démarrer l'installation en toute sécurité pour les opérateurs de ligne.

Afin de faciliter ce démarrage, l'automate met à disposition des blocs de programmes qui ne sont exécutés que dans le cas particulier d'un arrêt.

Il faudra distinguer la notion de **reprise à chaud** et de **reprise à froid**.

Automation

L'automate : Mode reprise à chaud

On considère par reprise à chaud, que l'automate est **sous tension** lors de son arrêt. On peut avoir ce cas par un arrêt provoqué par l'utilisateur qui passe volontairement l'automate en STOP, mais également dans le cas d'une erreur de programmation ou d'un retour de mesure défectueux.

- Les conséquences d'une reprise à chaud sur un automate PREMIUM sont les suivantes :
- Reprise scrutation programme à partir de la ligne où la coupure a eu lieu,
- Sur cette fin d'exécution programme, il n'y a pas de phase "action sur les sorties",
- **Mise à 1 de %S1,**
- Initialisation des files de messages,
- Envoi des paramètres de configuration à tous les coupleurs métiers selon les valeurs de réglage : analogique, comptage, communication, etc...,
- Désactivation des tâches EVT et FAST jusqu'à la fin du premier cycle de MAST,
- RAZ des files événements,
- Relance la tâche MAST

Automation

L'automate : Mode reprise à froid

On considère par reprise à froid, que l'automate est **hors tension**. C'est le cas de figure que l'on pourrait considérer comme un premier démarrage de l'installation ou celui d'une coupure électrique qui arrête l'installation en l'état.

- Les conséquences d'une reprise à froid sur un automate Premium sont les suivantes :
- Mise à leur valeur d'initialisation des bits et mots d'E/S et internes,
- Initialisation des séquences SFC (étapes initiales),
- Initialisation des blocs fonctions à partir des données de configurations (EFB et DFB),
- RAZ des mots internes si sauvegarde non demandée en configuration CPU,
- Annulation des forçages des bits et blocage d'étape,
- Envoi des paramètres de configuration à tous les coupleurs métiers : analogique, comptage, commande axes, communication, etc.,
- RUN ou STOP après reprise selon configuration CPU et/ou état entrée RUN/STOP si elle existe (entrée prioritaire sur configuration)
- **Mise à 1 de %S0**

Automation

L'automate : Mode reprise

Il est nécessaire de prévoir dans ça programmation, de définir la reprise, donc le fonctionnement de la ligne de production dans les deux cas. Il est évidant que ces cas sont différents.

Il faudra, pour cette marque d'automate, utiliser deux bits système qui permettrons à l'utilisateur de définir la marche à suivre en programmation pour éviter la casse matérielle et surtout de veiller à la sécurité des personnes.

Comme expliqué dans les deux slides précédents, il faut tenir compte de ce que l'automate fait dans un cas ou l'autre. Ici vous avez la possibilité d'ajouter un appel à une SR par exemple qui définira une série de variables qui vont mettre l'installation dans un mode qui correspond à la reprise désirée pour un fonctionnement correct de l'installation.

Nous remarquons que pour les automates de la gamme Schneider M340, il n'existe pas de bloc prédéfini suivant le type de démarrage, mais bien des bits système qui permettent de définir si l'arrêt est à chaud ou à froid. => **bits %S0 (Froid) et %S1 (Chaud).**

Automation

Le matériel Petra :

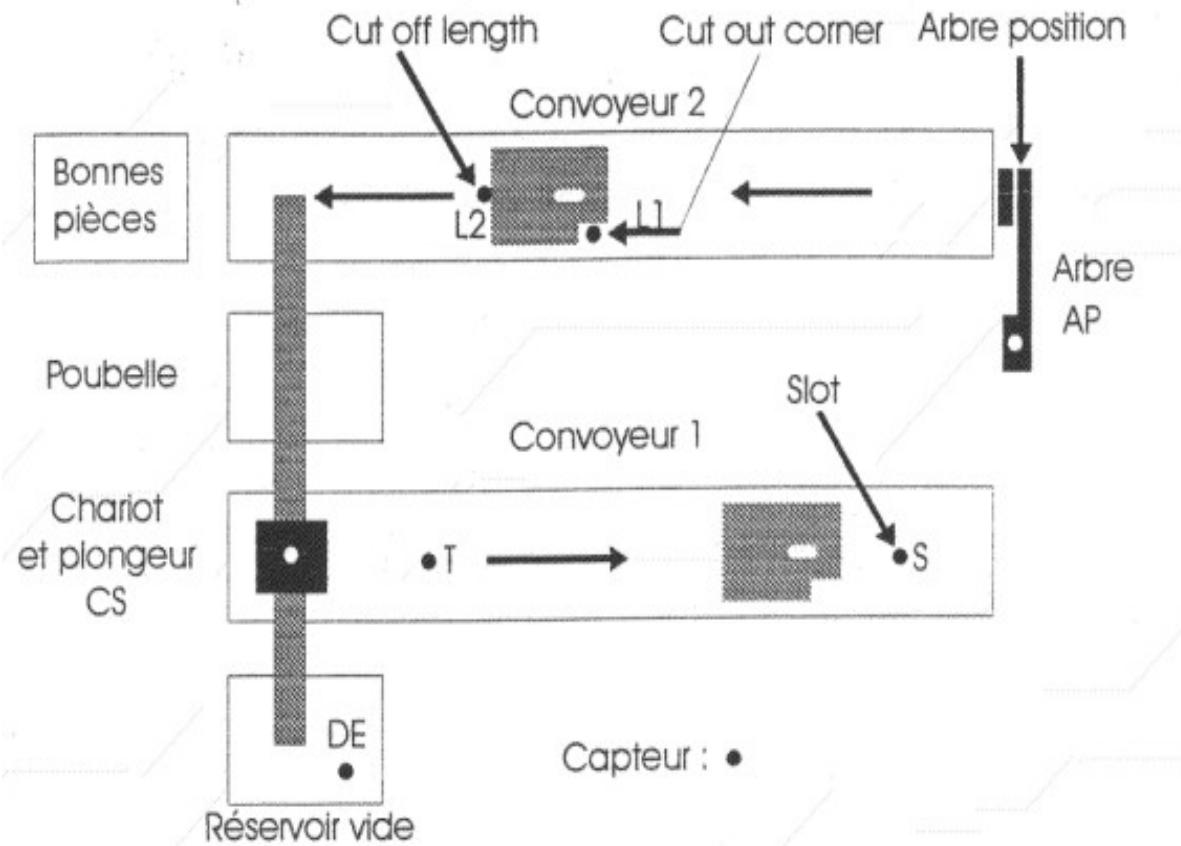


Il serait utile de réfléchir à ces deux cas pour la programmation du Petra dans le cadre du laboratoire. Le but étant de définir les deux cas et de simuler les arrêts afin de voir comment l'installation ce comporte.

Il faudra faire une analyse de la situation dans les deux cas et définir la marche à suivre pour la reprise du fonctionnement dans des conditions optimales.

Automation

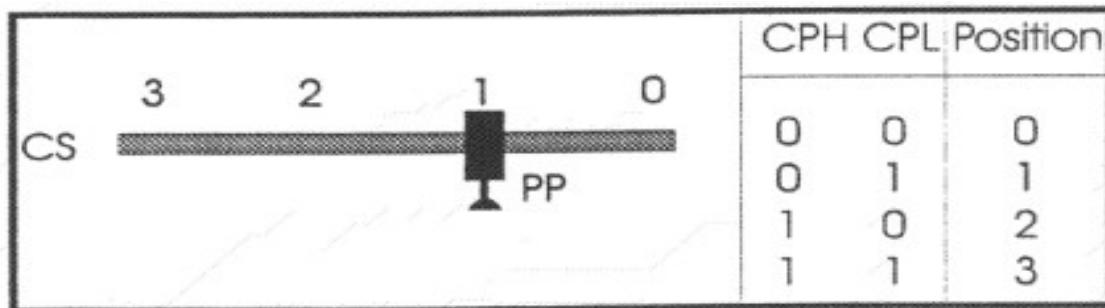
Le matériel Petra :



Pour effectuer la programmation du Petra, il est nécessaire de bien décomposer les différentes séquences qui composent le fonctionnement de cette mini ligne de tri de pièces. En observant l'installation, on remarque la présence de différents capteurs qui permettent de déterminer des éléments de forme sur les pièces utilisées par le Petra.

Automation

Le matériel Petra :



Le mouvement du bras sur son axe qui permet de déplacer les pièces est un bel exemple de réflexion.

Il est faut réfléchir à la séquence des opérations afin de pouvoir programmer son mouvement. De plus il est combiner à un deuxième élément qui vous permet de saisir une pièce (piloté par air comprimé), il s'agit d'un piston qui peut monter ou descendre sur lequel on a placé une valve que l'on peut mettre en dépression (aspiration) afin de saisir la pièce.

Pour libérer la pièce, on coupe la dépression. La pièce est ainsi déposée sur une partie de l'installation en fonction de la position du bras.

Automation

Le matériel Petra Exercice :

- Définir la liste des éléments (entrées / sorties),
- Rédiger les différentes séquences,
- Penser aux types de pièces à traiter,
- Définir les variables internes nécessaires à la programmation du process,
- Ajouter la reprise à chaud et à froid,
- Imaginer un mode de gestion des défauts.
- Dessiner une vue de supervision avec gestion des défauts

Automation

Sources :

- <https://blog.fr-techteam.com/quest-ce-que-lautomation>
- <https://prendreletempsdecomprendre.wordpress.com/2015/08/26/logique-combinatoire-et-logique-sequentiellequelle-difference/>
- <https://www.tecnipass.com/cours-electronique.cem-numerique-logique.sequentielle>
- <https://lab4sys.com/les-automates-programmables-industriels-api/>
- <http://sti-monge.fr/maintenancesystemes/wp-content/uploads/2013/02/manuel-de-r%C3%A9ference.pdf>
- [Les Expressions du langage ST](#)
- [Les Instructions du langage ST](#)

Questions / Réponses