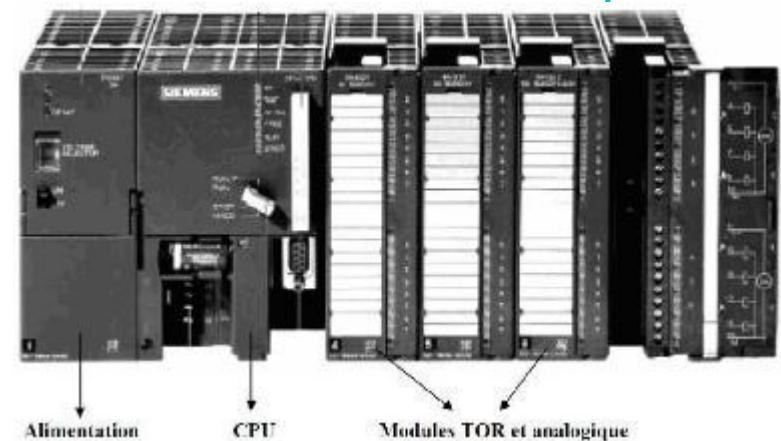
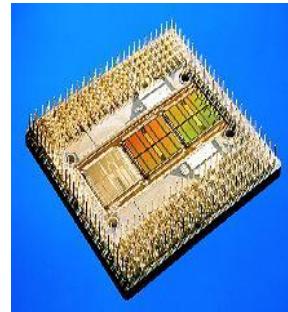


Informatique industrielle

Structure d'un système Automatisé



Année universitaire 2016/2017

Pr. Khalid BENJELLOUN

bkhalid@emi.ac.ma

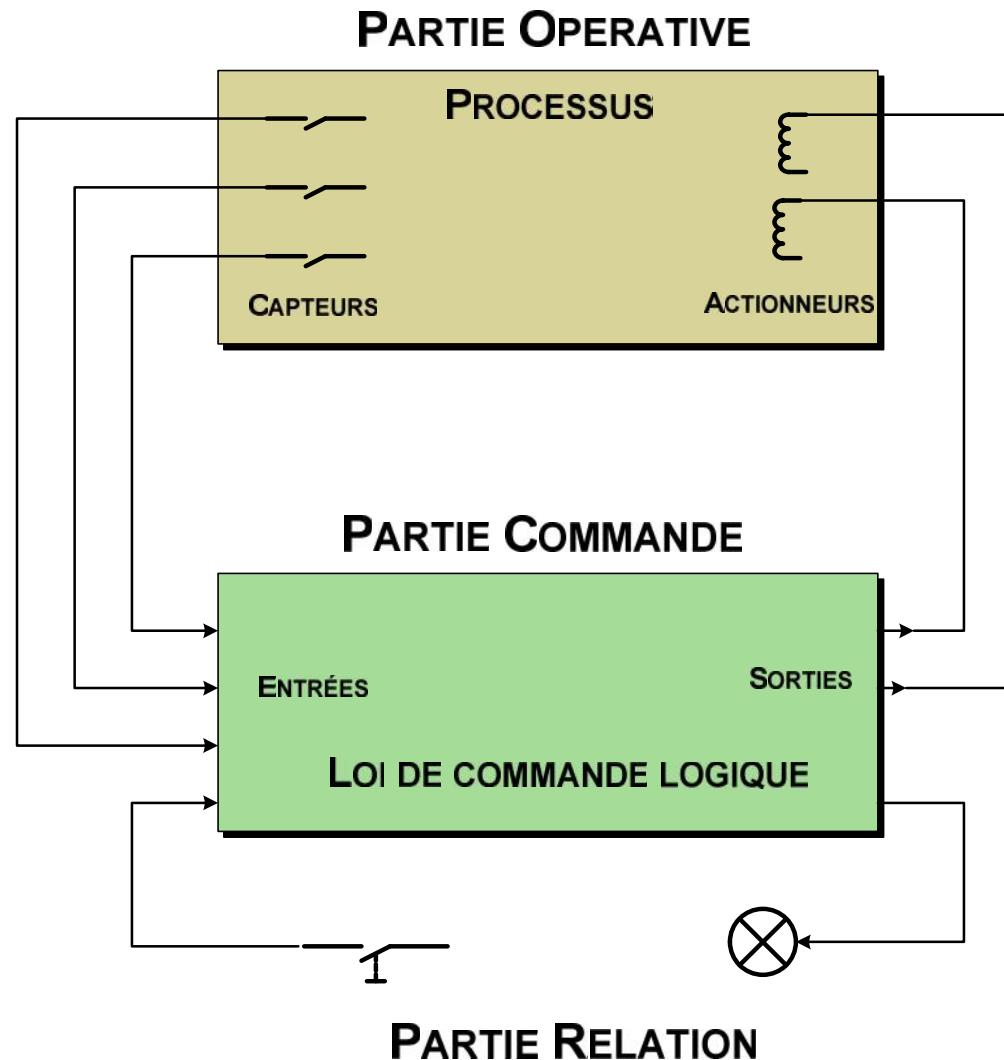
Département Electrique

Section Automatique et Informatique Industrielle



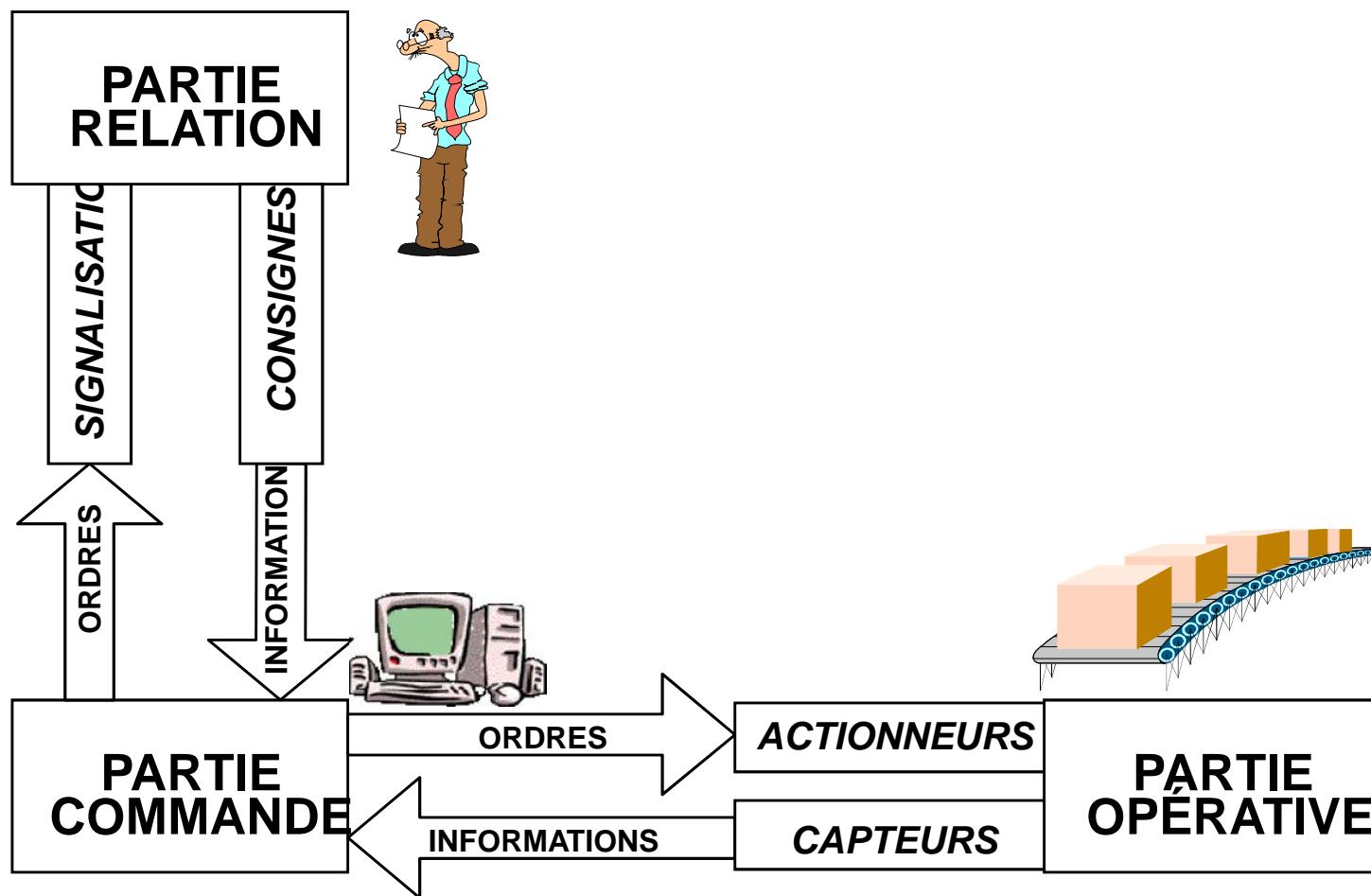


STRUCTURE GENERALE D'UN SYSTÈME AUTOMATISE



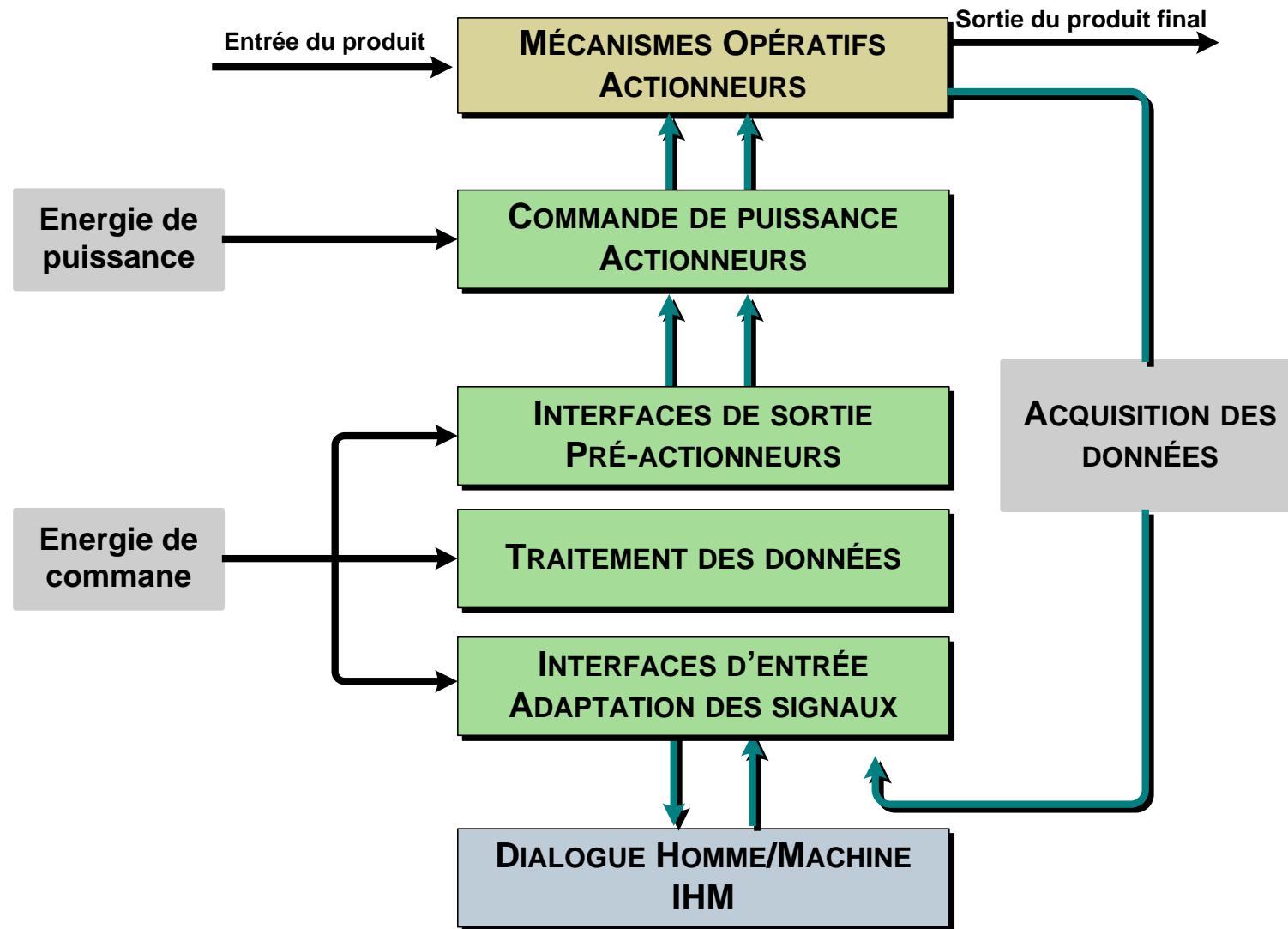


LES SYSTÈMES AUTOMATISÉS





STRUCTURE GENERALE D'UN SYSTÈME AUTOMATISE





STRUCTURE GENERALE D'UN SYSTÈME AUTOMATISE

ROLE DE CHAQUE PARTIE

La *partie opérative* est le processus physique à automatiser pour obtenir le travail désiré. Elle est matérialisée par ce que l'on appelle plus généralement la machine.

Son rôle est d'exécuter le travail et d'assurer la production prévue. Cette partie ne travaille que sur ordre. Ce n'est qu'un exécutant.

La *partie commande* élabore les ordres destinés à la machine, en fonction des comptes rendus venant de cette même machine, des consignes qu'elle reçoit de l'opérateur et de l'automatisme câblé ou programmé. La *partie commande* élabore également les signaux de visualisation. Elle est concrétisée par l'armoire de commande. Sa fonction première est d'élaborer les ordres de commande à destination de la machine. La *partie commande* va donc traiter l'ensemble des informations afin de prendre les décisions qui s'imposent.



STRUCTURE GENERALE D'UN SYSTÈME AUTOMATISE

ROLE DE CHAQUE PARTIE

Pour exploiter, régler et dépanner la machine, l'opérateur émet des consignes et reçoit des informations en retour. Ces différentes fonctions sont assurées par l'intermédiaire **du pupitre opérateur.**

La *partie relation ou dialogue* permet l'ensemble des communications entre l'homme et la machine.

Elle a pour principal but :

- ✓ de permettre la gestion de production,
- ✓ de suivre la production en quantité et qualité,
- ✓ de faciliter le travail de la maintenance en apportant une aide au diagnostic



DE LA LOGIQUE CABLEE A LA LOGIQUE PROGRAMMEE

La partie commande peut être en :

■ **Logique câblée,**

Ou

■ **Logique programmée.**

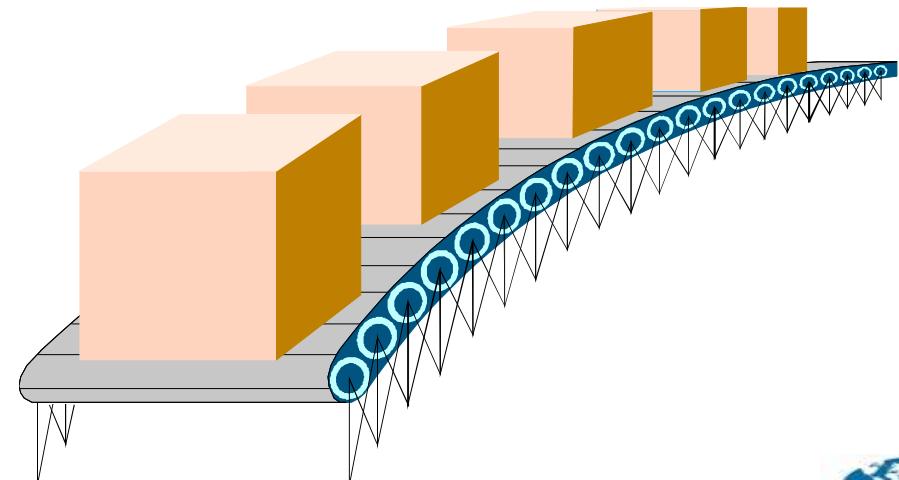
De la logique câblée à l'automate programmable

- Le programme de commande des automates à logique câblée, couramment utilisés dans le passé, était déterminé par le câblage des contacteurs et des relais, spécifique à la tâche à exécuter.
- Aujourd'hui, on utilise des automates programmables pour résoudre les tâches d'automatisation. La logique stockée dans la mémoire programme du système d'automatisation est indépendante de la configuration matérielle et du câblage et peut donc être modifiée à tout moment à l'aide d'une console de programmation.



LES SYSTÈMES AUTOMATISÉS LA PARTIE OPERATIVE

- Moteurs électriques (C.A. ou C.C.)
- Vérins pneumatiques ou hydrauliques
- Vannes (électriques ou pneumatiques)
- Éléments chauffants
-

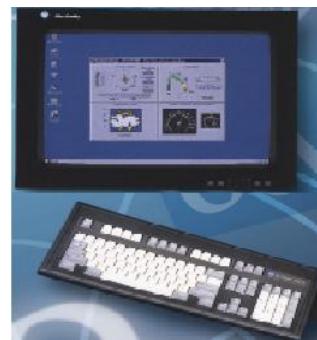
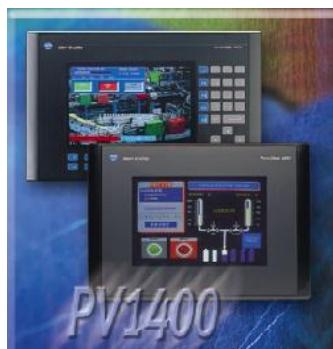




LES SYSTÈMES AUTOMATISÉS

PARTIE RELATION OU DIALOGUE HOMME/MACHINE

- Panneaux de commande
 - Voyants, indicateurs
 - Poussoirs, sélecteurs
- Interfaces Homme-Machine
- Alarmes





SAP

Informatique industrielle-10

LES SYSTÈMES AUTOMATISÉS

PARTIE RELATION OU DIALOGUE HOMME/MACHINE





Eléments de la P.O. et de la P.C.

Diagramme fonctionnel		Exemples	Commentaires
EFFECTEUR	<pre> graph LR W1[Matière d'œuvre] --> AGIR[AGIR Sur la M.O.] AGIR --> W2[Matière d'œuvre] style AGIR fill:#00AEEF,color:#FFF </pre>	Fraise, Foret, Mors d'étau, pince de robot.... 	Les effecteurs sont multiples et variés et sont souvent conçus spécialement pour s'adapter à l'opération qu'ils ont à réaliser sur la Matière d'œuvre. Ils reçoivent leur énergie des actionneurs.
ACTIONNEUR	<pre> graph LR W1[Energie d'entrée (W1)] --> CONVERTIR[CONVERTIR L'énergie] CONVERTIR --> W2[Energie utile (W2)] style CONVERTIR fill:#00AEEF,color:#FFF </pre>		Convertissent l'énergie qu'ils reçoivent des pré-actionneurs en une autre énergie utilisée par les effecteurs. Ils peuvent être Pneumatiques, Hydrauliques ou Electriques
PRE-ACTIONNEUR	<pre> graph LR W1[Energie du réseau (W1)] --> DISTRIBUER[DISTRIBUER L'énergie] DISTRIBUER --> Energie[Energie distribuée à l'actionneur] Energie --> Pilotage[Pilotage] style DISTRIBUER fill:#00AEEF,color:#FFF </pre>		Distribuent l'énergie aux actionneurs à partir des ordres émis par la PC.
CAPTEUR	<pre> graph LR W1[Information source] --> DETECTEUR[DETECTOR MESURER Une grandeur] DETECTEUR --> W2[Information image] W2 --> Pilotage[Pilotage] style DETECTEUR fill:#00AEEF,color:#FFF </pre>		Renseignent la PC sur l'état de la PO, Ils peuvent détecter des positions, des pressions, des températures, des débits... Peuvent être électriques ou pneumatiques. Signaux du type TOR, Analogique ou Numérique.
TRAITEMENT	<pre> graph LR W1[Signal d'entrée (capteurs, consignes...)] --> TRAITER[TRAITER L'information] TRAITER --> W2[Signal de sortie] W2 --> Pilotage[Pilotage des préactionneurs, signalisation...] style TRAITER fill:#00AEEF,color:#FFF </pre>		Dans les systèmes modernes, l'API assure de plus en plus cette fonction. Certains systèmes purement pneumatiques peuvent être contrôlés par des séquenceurs ou des fonctions logiques.
DIALOGUE	<pre> graph LR W1[Consignes de l'opérateur, Infos de la PC] --> FAIRE[FAIRE COMMUNIQUER Homme/machine] FAIRE --> W2[Infos vers opérateur] W2 --> Pilotage[Pilotage] style FAIRE fill:#00AEEF,color:#FFF </pre>		L'unité de dialogue permet à l'opérateur d'envoyer des consignes à l'unité de traitement et de recevoir de celle-ci des informations sur le déroulement du processus.



Définition d'un Automate Programmable

- **Un automate programmable est un appareil dédié au contrôle d'une machine ou d'un processus industriel, constitué de composants électroniques, comportant une mémoire programmable par un utilisateur non informaticien, à l'aide d'un langage adapté. En d'autres termes, un automate programmable est un calculateur logique, ou ordinateur, au jeu d'instructions volontairement réduit, destiné à la conduite et la surveillance en temps réel de processus industriels.**



Définition d'un Automate Programmable

Trois caractéristiques fondamentales distinguent totalement l'Automate Programmable Industriel (API) des outils informatiques tels que les ordinateurs (PC industriel ou autres):

- il peut être directement connecté aux capteurs et pré-actionneurs grâce à ses entrées/sorties industrielles,
- il est conçu pour fonctionner dans des ambiances industrielles sévères (température, vibrations, micro-coupures de la tension d'alimentation, parasites, etc.),
- et enfin, sa programmation à partir de langages spécialement développés pour le traitement de fonctions d'automatisme fait en sorte que sa mise en oeuvre et son exploitation ne nécessitent aucune connaissance en informatique.



AUTOMATE PROGRAMMABLE

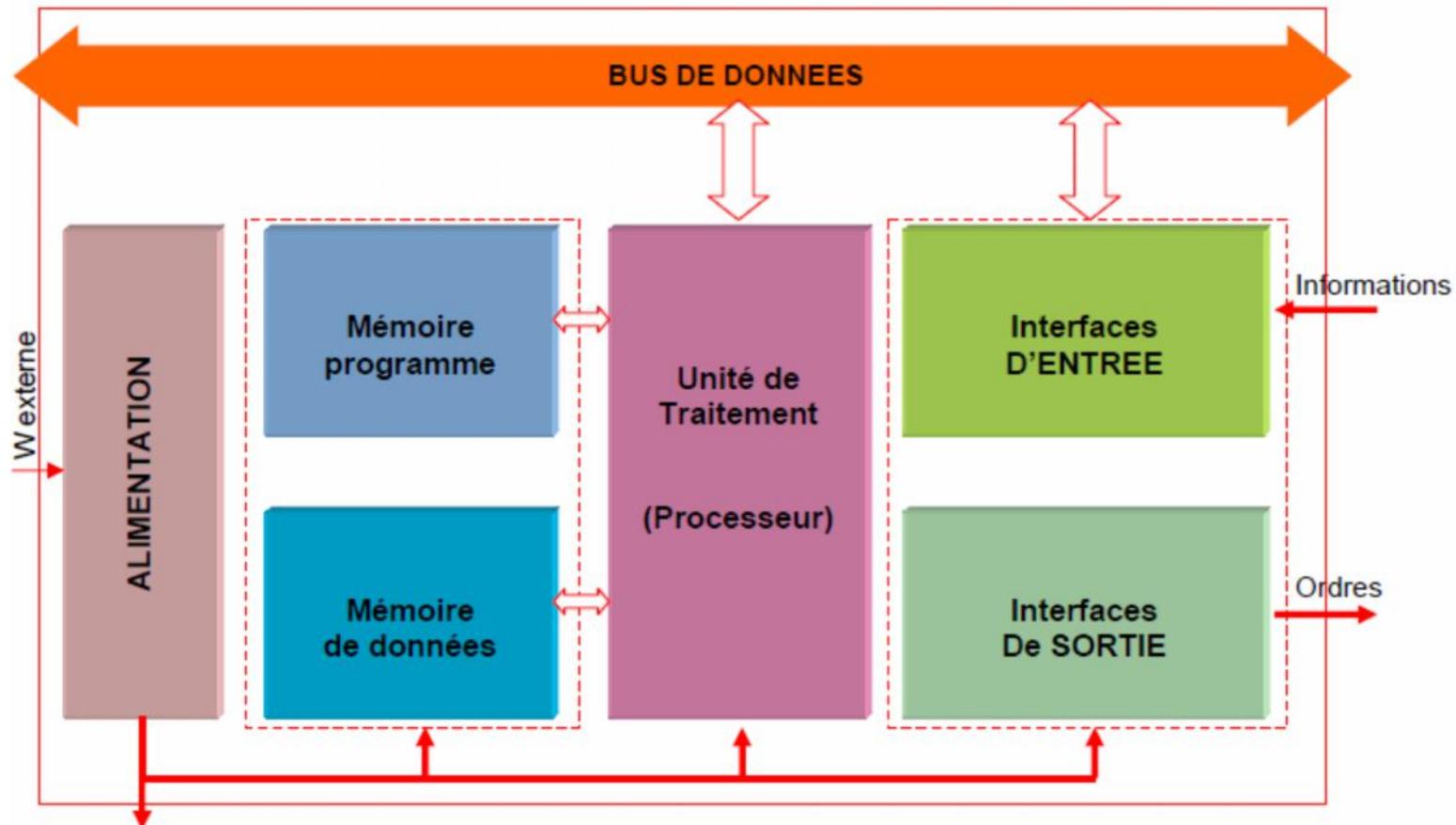
Un API, c'est un ensemble Hardware et Software

■ Au niveau Hardware, il se compose:

- d'une alimentation
- d'une unité centrale ou CPU
- de modules I/O digitales ou analogiques
- de cartes intelligentes métier ou coupleurs de communication pour dialoguer avec le monde extérieur.
- d'un ou de plusieurs bus de communication pour le dialogue entre la CPU et tous ces modules



Structure interne d'un Automate programmable





Structure interne d'un Automate programmable

Un automate programmable est un appareil électronique qui comporte une mémoire programmable par un utilisateur automaticien à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatismes, par exemple :

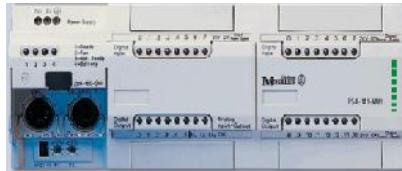
- Logique séquentielle et combinatoire
- Temporisation
- Comptage, décomptage, comparaison
- Calcul arithmétique
- Réglage, asservissement, régulation, Etc...



Base

- La conception d'un automate peut être :

- Monobloc : souvent pour les "petits modèles"



Nano Schneider, S7-200
Siemens, C20 Omron...

- Modulaires : Rack + cartes



Premium Schneider,
S7-300/400 Siemens,
CQM1 Omron, ...

- Dans tous les cas, les éléments composants ces API sont du même type.
- Alimentation : transforme la tension secteur (24, 48V =, 110, 230 V~) en tension continue (4, 12, ... V=) pour alimenter l'électronique des cartes.
- Bus : Ensemble des fils permettant la circulation des informations entre les différents constituants de l'API. (Energie, signaux, Adressage, ...)



classification

nombre
d'équipement

Cartes dédiées



?

automate compact



Logique
cablée



Relais
programmable

automate
modulaire
(réseaux et
métiers)



SOFT PLC : Pc industriel
et logiciel de contrôle
commande



Volume & niveau
d'automatisme



AUTOMATE PROGRAMMABLE

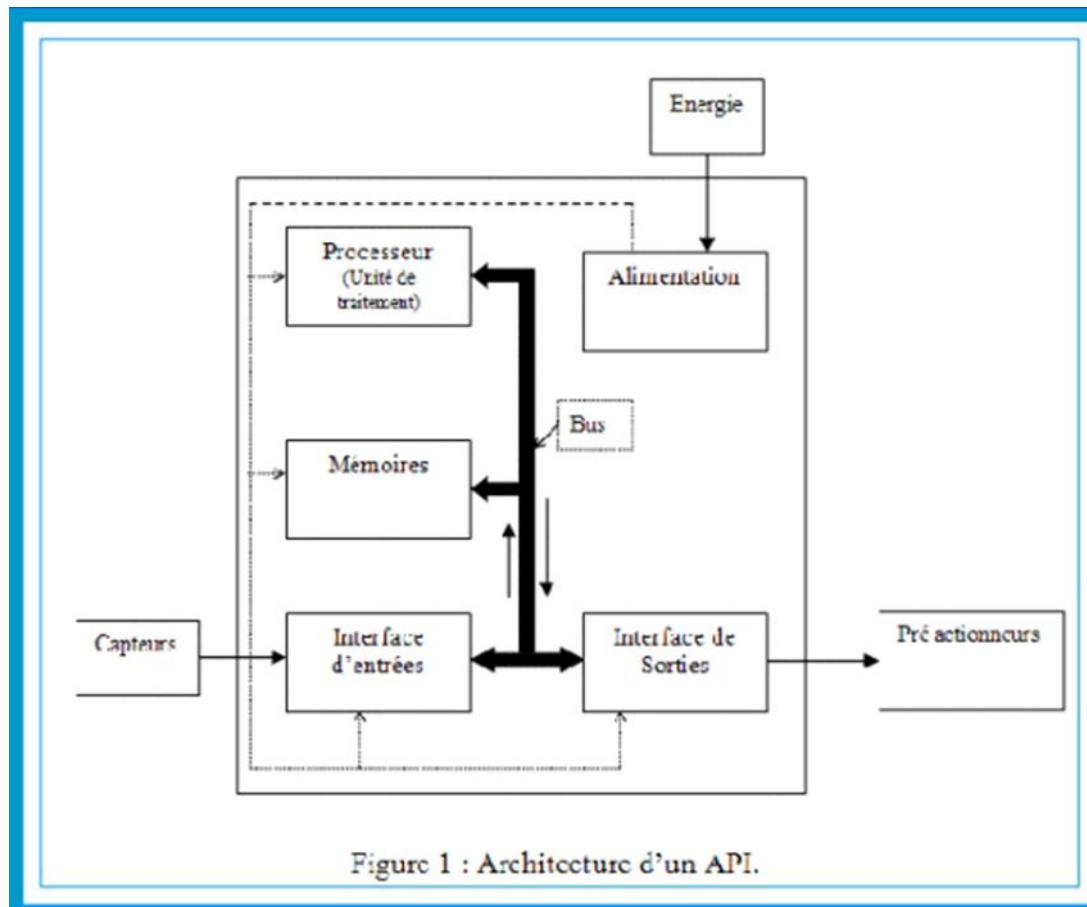
■ Au niveau Software

- Un système d'exploitation dans la CPU
- De la mémoire dans la CPU répartie en différentes zones, MIE, MIS, bits internes, tempos, compteurs, données, programme
- Pour créer le programme, un atelier logiciel



Architecture d'un API

- La structure interne d'un API peut se représenter comme suit :



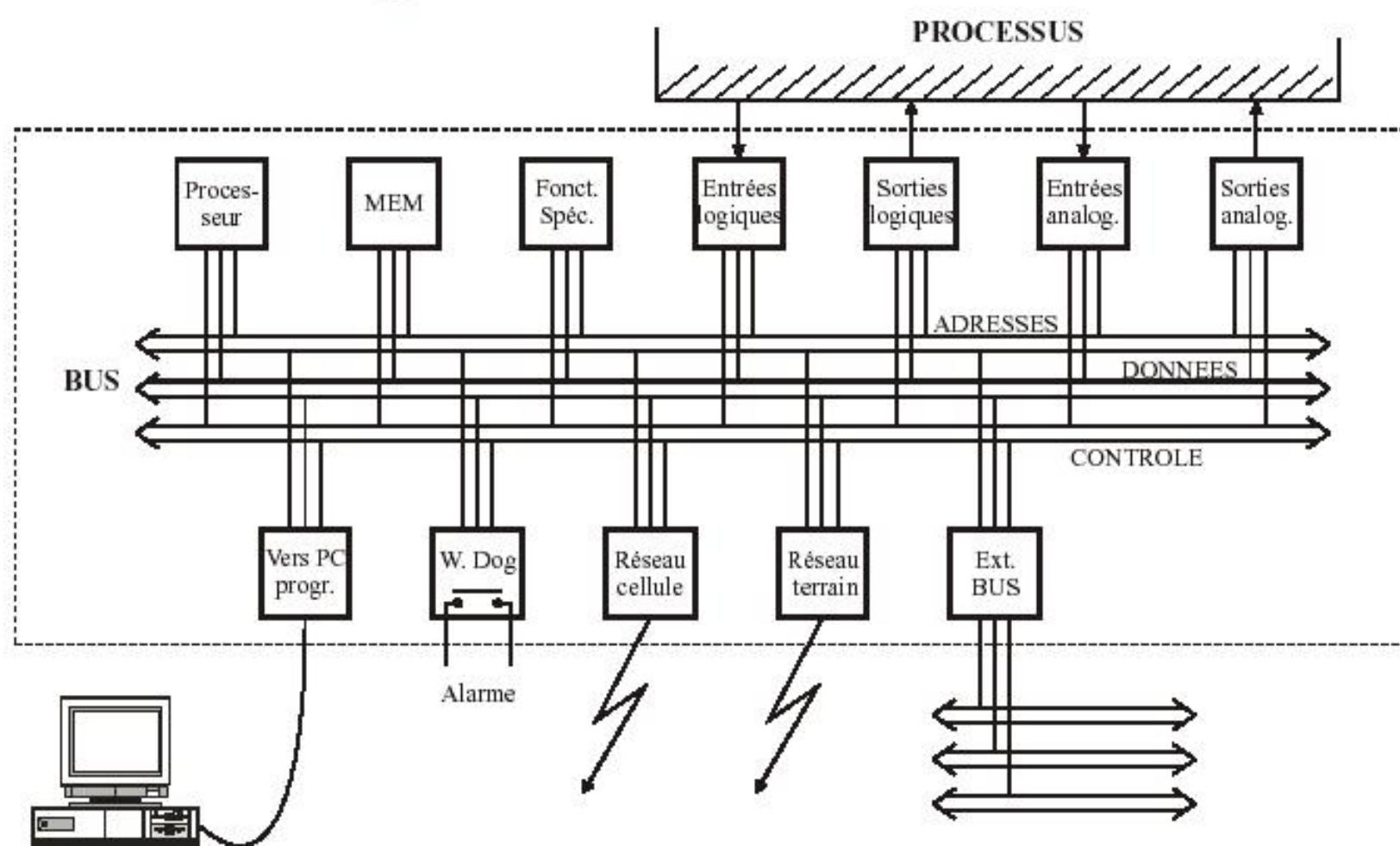


Architecture d'un API

- L'automate programmable reçoit les informations relatives à l'état du système et puis commande les pré actionneurs suivant le programme inscrit dans sa mémoire.
- Un API se compose donc de trois grandes parties:
 - Le processeur ;
 - La mémoire ;
 - Les interfaces Entrées/sorties

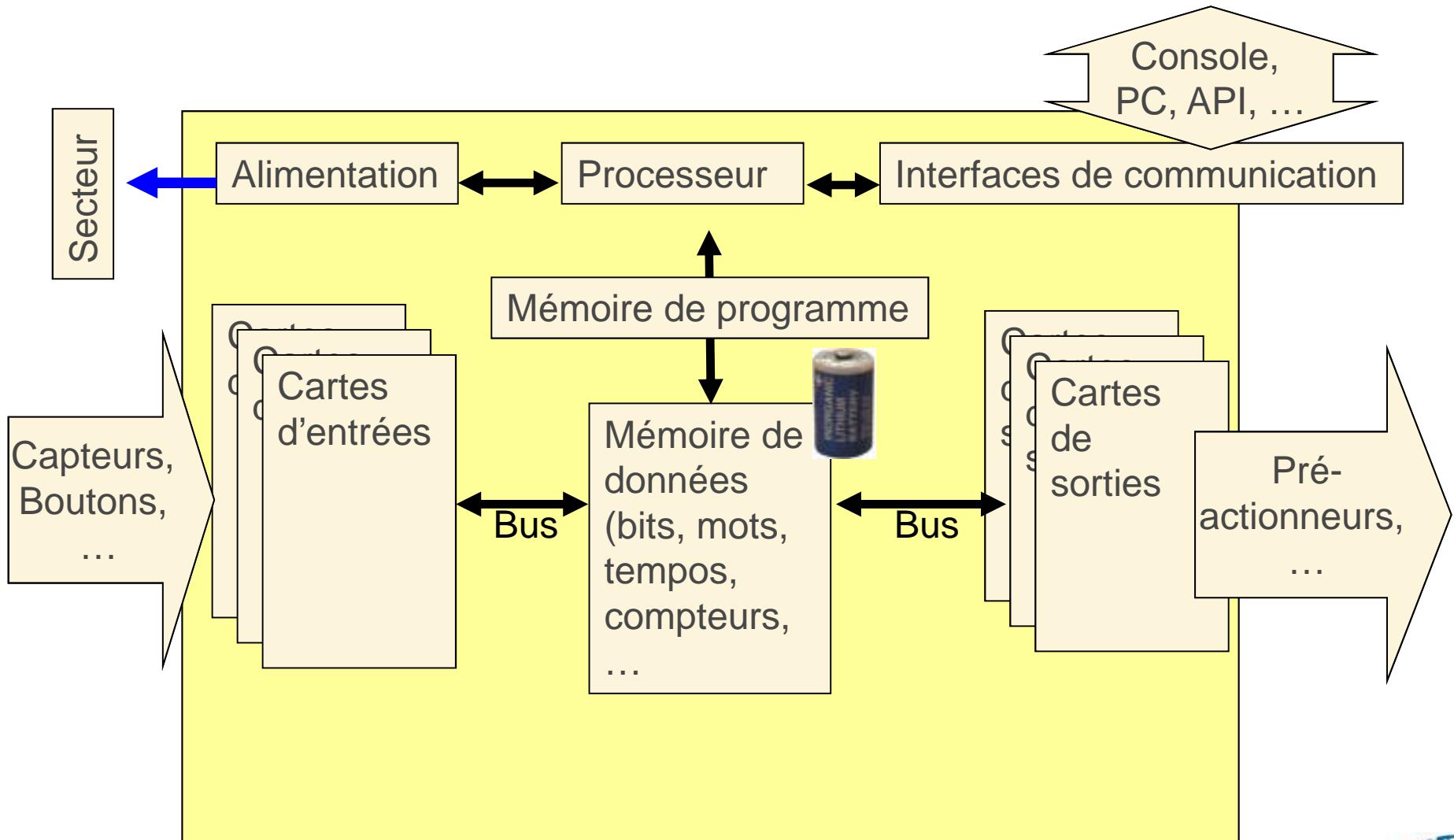


TECHNOLOGIE DE REALISATION D'UN AUTOMATE



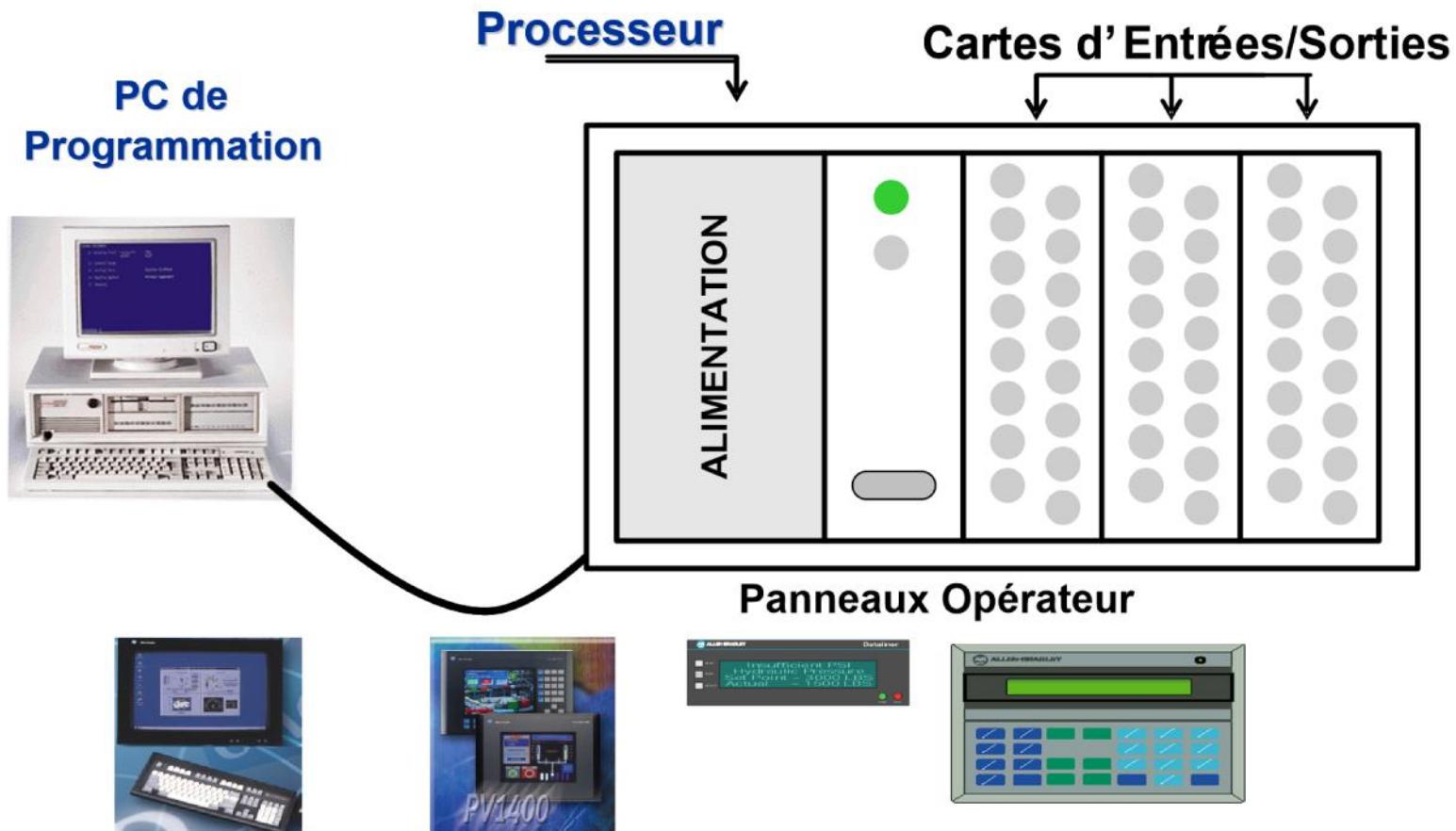


STRUCTURE D'UN AUTOMATE





STRUCTURE D'UN AUTOMATE PROGRAMMABLE





STRUCTURE D'UN AUTOMATE PROGRAMMABLE

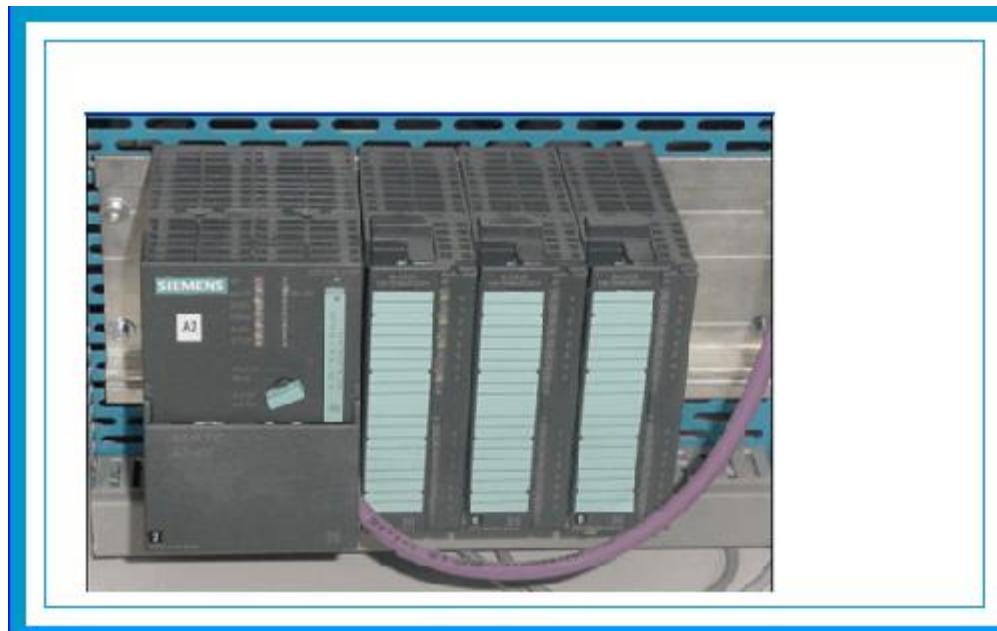


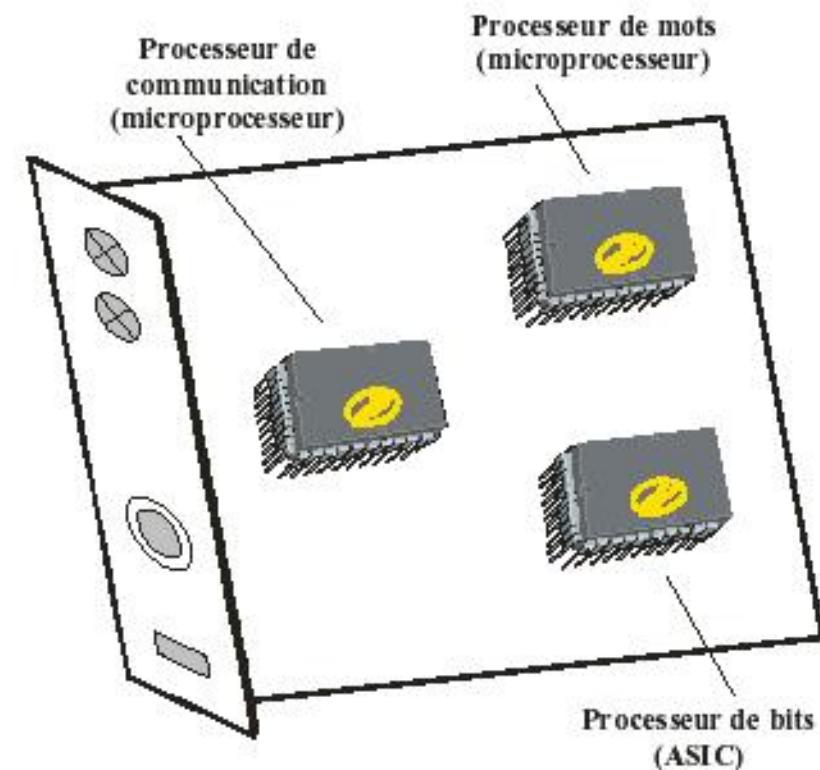
Figure: Exemple d'une architecture réelle d'un API S7-300 (marque de Siemens AG).



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE PROCESSEUR

■ Structure d'une carte processeur

- Un micro-processeur standard utilisé pour les fonctionnalités additionnelles (arithmétiques)**
- Un circuit ASIC (Application Specific Integrated Circuit) pour les traitements purement logiques**
- Un processeur pour la prise en charge des communications vers la console et les réseaux informatiques**





PROCESSEUR

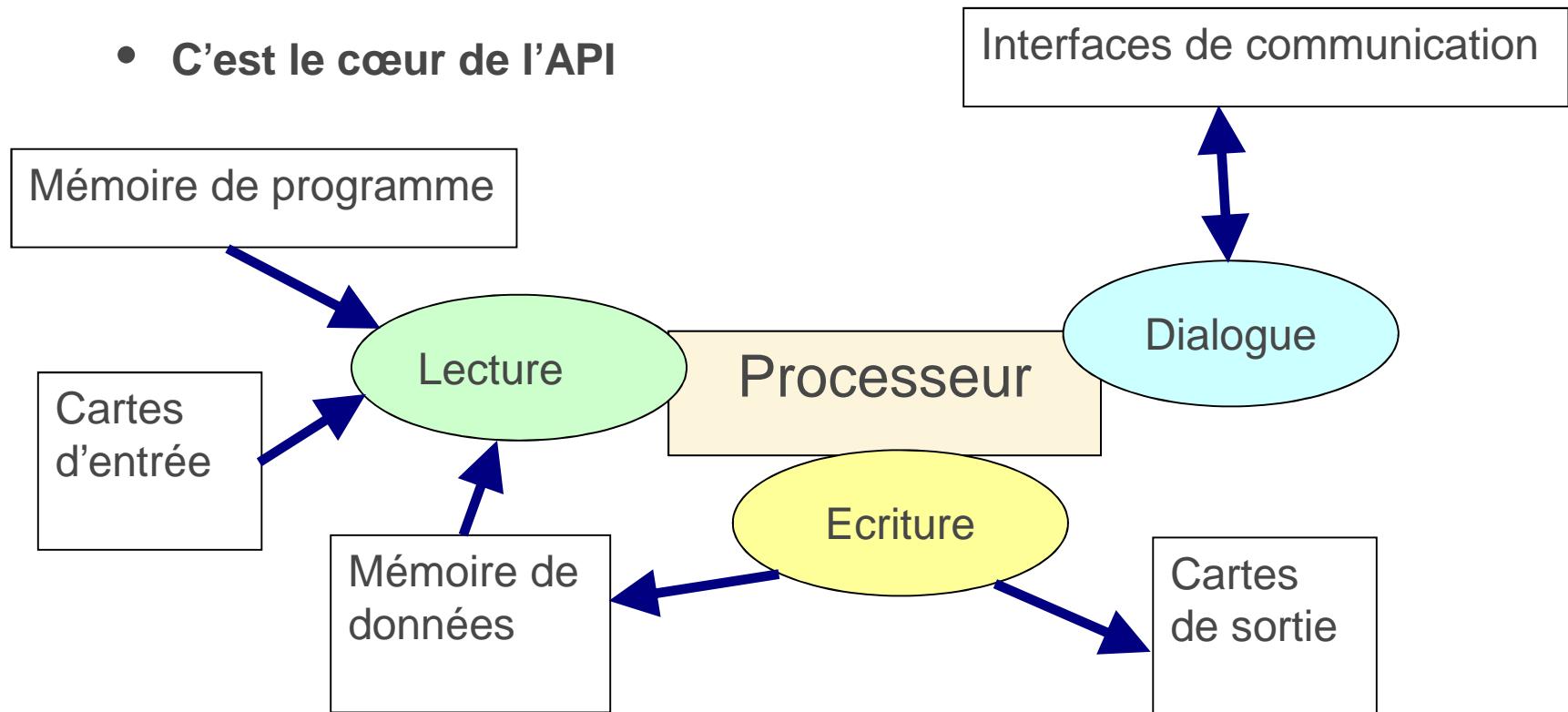
Le processeur, ou unité centrale (UC), a pour rôle principal le traitement des instructions qui constituent le programme de fonctionnement de l'application (les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul PID, etc..). Mais en dehors de cette tâche de base, il réalise également d'autres fonctions :

- Gestion des entrées/sorties.
- Surveillance et diagnostic de l'automate par une série de tests lancés à la mise sous tension ou cycliquement en cours de fonctionnement.
- Dialogue avec le terminal de programmation, aussi bien pour l'écriture et la mise au point du programme qu'en cours d'exploitation pour des réglages ou des vérifications des données.
- Un ou plusieurs processeurs exécutent ces fonctions grâce à un micro logiciel préprogrammé dans une mémoire de commande, ou mémoire système. Cette mémoire morte définit les fonctionnalités de l'automate. Elle n'est pas accessible à l'utilisateur.



PROCESSEUR

- Encore appelé :
 - Unité de traitement (UT),
 - Central Process Unit (CPU),
 - Unité Centrale (UC).
- C'est le cœur de l'API





PROCESSEUR

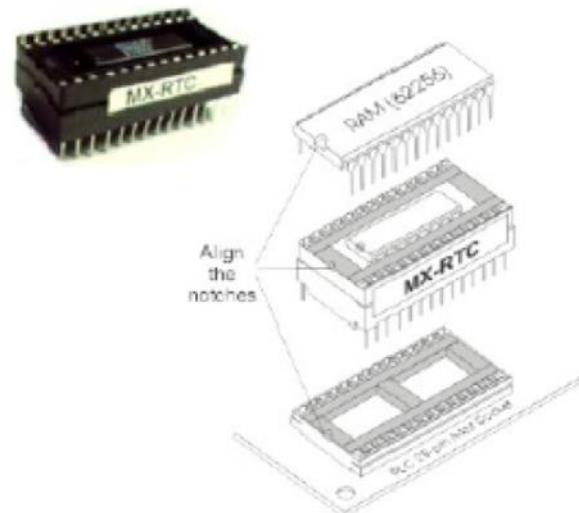
Le processeur gère l'ensemble des échanges informationnels en assurant :

- La lecture des informations d'entrée
- L'exécution des instructions du programme mis en mémoire
- La commande ou l'écriture des sorties



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE MEMOIRES

- Utilisation de mémoire à semi-conducteurs
- Données: RAM (Random Acces Memory) mémoire vive
 - Lecture / écriture
 - Perte d'information en cas de coupure d'alimentation
 - Technique CMOS (faible consommation)





STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE MEMOIRES

■ Programme:

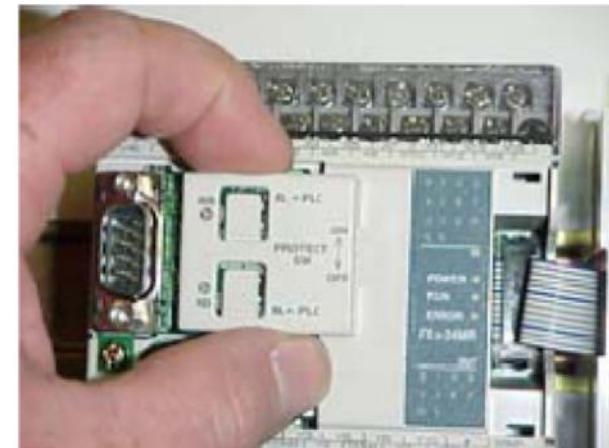
- La mémoire doit pouvoir subir sans dommage les coupures de courant
- EPROM (Erasable Programmable Read Only Memory) : Effacé après exposition aux UV
- EEPROM (Electrically...)

Effacement par voie électrique in situ

- Ram gardées

■ Systèmes

- EPROM ou PROM





STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE MEMOIRES

Une mémoire programme

La mémoire programme de type RAM contient les instructions à exécuter par le processeur afin de déterminer les ordres à envoyer aux préactionneurs reliés à l'interface de sortie en fonction des informations recueillies par les capteurs reliés à l'interface d'entrée.

Une mémoire de données

La mémoire de donnée permet le stockage de :

- l'image des entrées reliées à l'interface d'entrée
- L'état des sorties élaborées par le processeur
- Les valeurs internes utilisées par le programme (résultats de calculs, états intermédiaires ,...)
- Les états Forcés ou non des E/S



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE MEMOIRES

- **Mémoire de programme** : Contient le programme à exécuter. De type RAM, EPROM, EEPROM, Flash, ..., (*suivant la marque, le modèle, le choix de l'utilisateur, ...*).
- Quelque fois composé d'une zone de **RAM** sauvegardée par pile et d'une autre zone d'**EEPROM**, amovible ou non, qui sert de sauvegarde sans pile.
- **Mémoire de données** : Contient les informations dont l'API à besoin pour exécuter le programme (Valeurs des entrées, consignes) ou que l'API pilote (valeur des sorties, temporisations, compteurs, ...).
- Quelques fois il existe une zone de bits et une zone de mots indépendantes l'une de l'autre.
- Une partie de cette mémoire de type RAM (mots, valeur des temporisations et compteurs) est sauvegardée en cas de coupure d'alimentation.
- Ces 2 zones mémoire peuvent cohabiter sur la même carte électronique ou être situées sur des cartes différentes



La mémoire de l'API est l'élément fonctionnel qui peut recevoir, conserver et restituer des informations.

L'espace mémoire peut être divisé en deux parties :

- ▶ La mémoire Programme qui permet le stockage des instructions à exécuter par l'API.
- ▶ La mémoire de données qui permet le stockage de l'état des E/S et des variables internes.

Les mémoires utilisées dans un API peuvent être des types suivants :

- ▶ **R.A.M. (Random Access Memory)** mémoire à accès aléatoire. Cette mémoire doit être alimentée électriquement pour pouvoir conserver les informations. On l'appelle également la **mémoire vive**. Avant son exécution, le programme est transféré dans cette mémoire qui permet d'atteindre des vitesses en lecture et écriture très rapides.
- ▶ **R.O.M. (Read Only Memory)** mémoire à lecture uniquement. Appelée également **mémoire morte**, elle permet de stocker des informations indéfiniment sans aucune alimentation électrique.
- ▶ **P.R.O.M. (Programmable Read Only Memory)** mémoire de type ROM mais Programmable. C'est une ROM que l'on peut programmer une seule fois.
- ▶ **E.P.R.O.M. (Erasable Programmable Read Only Memory)** mémoire de type PROM que l'on peut effacer par exposition du circuit aux rayons ultra-violets.
- ▶ **E.E.P.R.O.M. (Electrical Erasable Programmable Read Only Memory)** mémoire de type PROM que l'on peut effacer électriquement en écrivant à nouveau sur le contenu de la mémoire. Ce type de mémoire par sa simplicité de mise en œuvre tend à remplacer de plus en plus la mémoire EPROM.



Exemple de carte d'extension mémoire pour un API TSX 37.

Format PCMCIA type 1
- RAM 32K / 64K
- EEPROM 32K / 64K



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE INTERFACES ENTREES/SORTIES

- Modules réalisant l'interface entre les signaux du processus (vers actionneurs ou venant des capteurs) et les signaux du bus interne de l'automate
- Fonctions
 - Découplage mécanique (bornier)
 - Découplage électrique entre les signaux du processus et l'électronique PLC
 - Synchronisation des transferts conformément aux procédures d'échange du BUS du PLC
- Entrées / sorties
 - Logiques
 - Analogiques



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE INTERFACES ENTREES/SORTIES

- Généralités :
 - On distingue les cartes TOR (tout ou rien) et les cartes ANA (Analogique)
 - Existent pour différentes tension d'utilisation : courant continu (24, 48 V), courant alternatif '10/220 V),
 - Une carte est en général constituée de plusieurs voies, par multiples de 2 (2, 4 , 8, 16, 32, 64),
 - Les différentes voies ont souvent des communs d'alimentation (groupe de 8, de 16 , ...)



Les interfaces entrées/sorties:

- Les entrées/sorties TOR (Tout ou Rien) assurent l'intégration directe de l'automate dans son environnement industriel en réalisant la liaison entre le processeur et le processus. Elles ont toutes, de base, une double fonction :
- Une fonction d'interface pour la réception et la mise en forme de signaux provenant de l'extérieur (capteurs, boutons poussoirs, etc.) et pour l'émission de signaux vers l'extérieur (commande de pré-actionneurs, de voyants de signalisation, etc.). La conception de ces interfaces avec un isolement galvanique ou un découplage opto-électronique assure la protection de l'automate contre les signaux parasites.



Les interfaces entrées/sorties:

Une interface d'ENTREE

L'interface d'entrée permet la connexion à l'API d'une multitude de capteurs pouvant être :

- TOR (logiques ou Tout Ou Rien)
- Numériques
- Analogiques

Ces différentes entrées sont mises en forme par l'interface d'entrée avant d'être stockées dans la mémoire de données.

Une interface de SORTIE

L'interface de sortie permet la connexion à l'API d'une multitude de préactionneurs pouvant être :

- TOR (logiques ou Tout Ou Rien)
- Numériques
- Analogiques



Les interfaces entrées/sorties:

- Une fonction de communication pour l'échange des signaux avec l'unité centrale par l'intermédiaire du bus d'entrées/sorties.

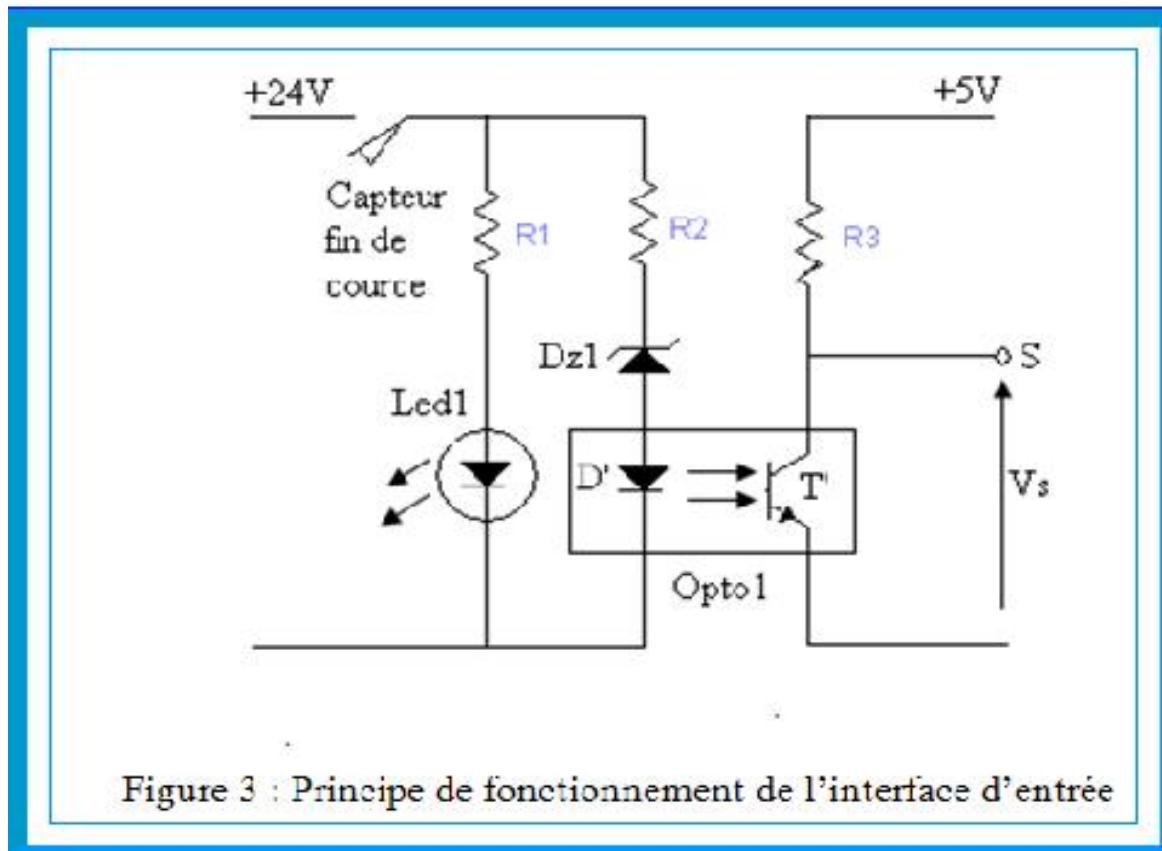
Le fonctionnement de l'interface d'entrée (figure suivant) peut être résumé comme suit :

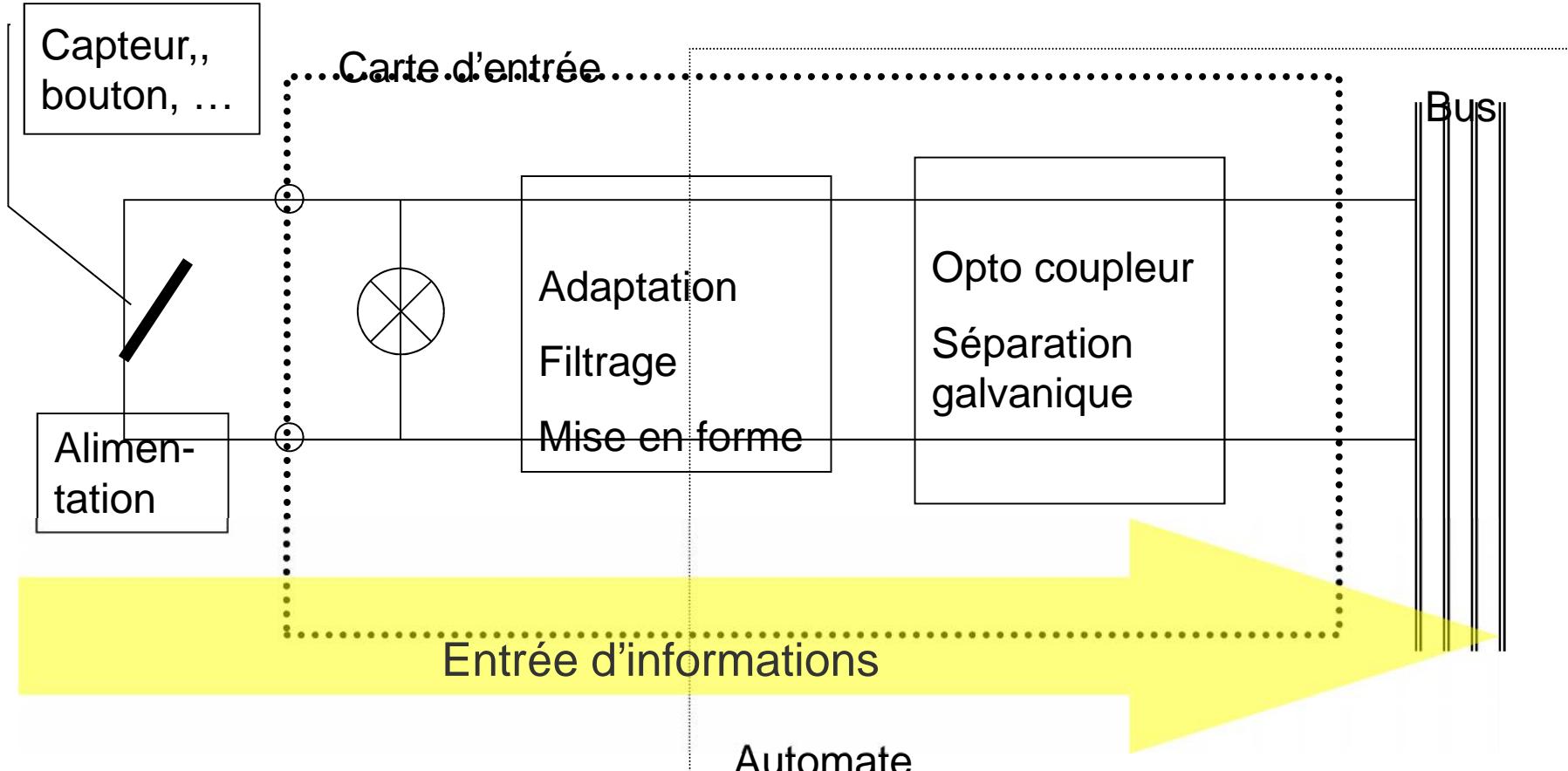
Lors de la fermeture du capteur ;

- La « Led 1 » signale que l'entrée de l'API est actionnée.
- La « Led D' » de l'optocoupleur « Opto 1 » s'éclaire.
- Le phototransistor « T' » de l'optocoupleur « Opto 1 » devient passant.
- La tension Vs=0V.
- Donc lors de l'activation d'une entrée de l'automate, l'interface d'entrée envoie un « 0 » logique à l'unité de traitement et un « 1 » logique lors de l'ouverture du contact du capteur (entrée non actionnée).



Les interfaces entrées/sorties:



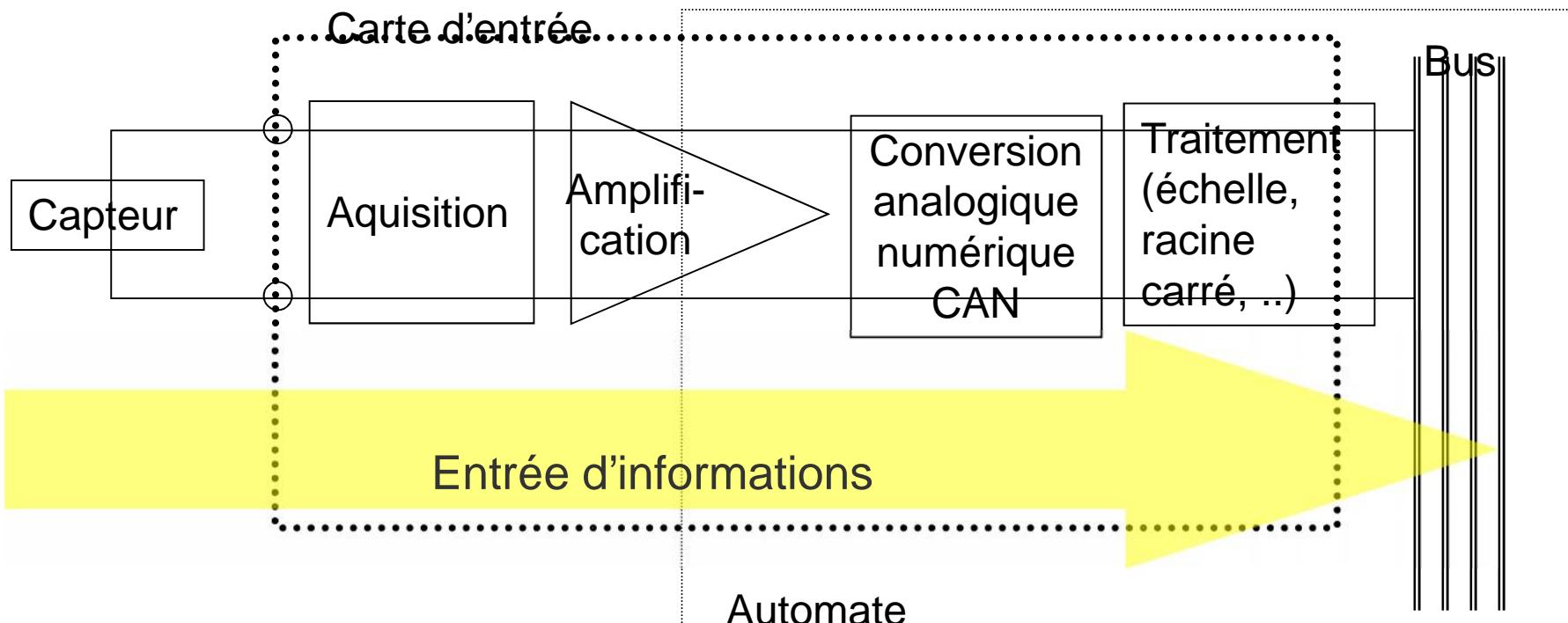


Schémas théorique d'une voie d'entrée d'une carte d'entrée TOR



Cartes d'entrées ANA

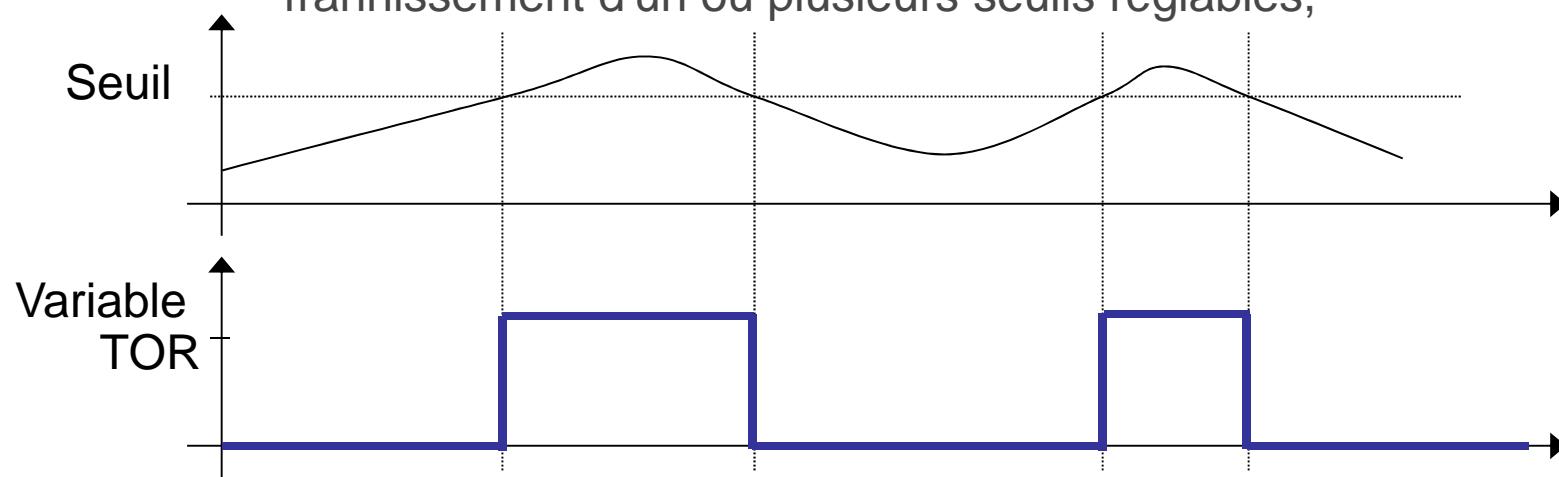
- A conversion complète : transforme la grandeur analogique en valeur numérique, généralement sur X bits + signe Exemple pour une tension évoluant de -10 V à + 10 V, la carte fournit une valeur évoluant de 0 à 8192.





Cartes d'entrées ANA

- Utilisées pour surveiller des valeurs de tension ou de courant fournies par des capteurs qui mesurent des grandeurs physiques telles que : température, pression, niveau, ...
- Elles sont caractérisées par l'amplitude et la nature du signal :
 - Tension : 0 à 10 V, -10 à +10 V, -5 à + 5 V, ...
 - Courant : 0 à 20 mA, 4 à 20 mA
- Il existe 2 types principaux de carte :
 - A détection de seuil : active un relais suivant le franchissement d'un ou plusieurs seuils réglables,





Cartes de sorties TOR

- Permettent d'envoyer des informations, des ordres à l'extérieur de l'API.
- Plusieurs fonctions :
 - Connexion : bornier à vis, bornier à ressort, cosses, prises à souder,
 - Mémorisation du résultat : par une bascule,
 - Adaptation en puissance : afin d'adapter la puissance interne à la puissance désirée en sortie, et en fonction de la technologie utilisée, il existe les circuits suivants :
 - Amplificateur à transistor ou à triac,
 - Transistor de puissance, triacs, relais
 - Séparation galvanique : un coupleur optoélectronique sert à isoler électriquement l'intérieur de l'extérieur de l'API, dans le cas des sorties à transistor ou à triacs. Dans le cas des sorties à relais, le relais assure directement cette fonction.
 - Protection : pour protéger la carte et la charge, on peut trouver :
 - Fusibles, disjoncteurs, diodes
 - Visualisation de l'état logique de la sortie grâce à une diode électroluminescente.



Les interfaces entrées/sorties:

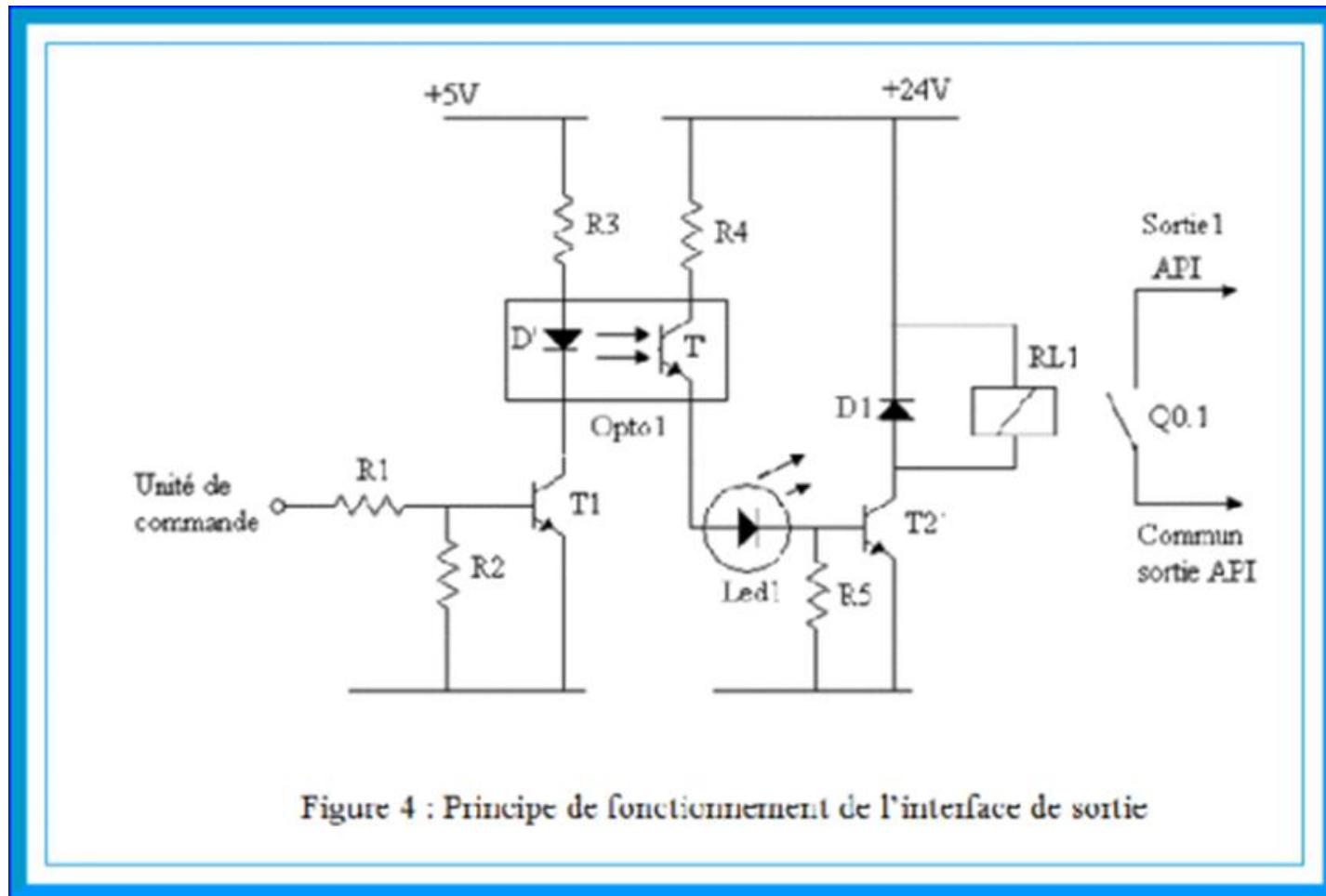
Le fonctionnement de l'interface de sortie peut être résumé comme suit :

Lors de commande d'une sortie automate ;

- L'unité de commande envoi un « 1 » logique (5V).
- « T1 » devient passant, donc la « Led D' » s'éclaire
- Le photo-transistor « T' » de l'optocoupleur « Opto1 » devient passant.
- La « Led1 » s'éclaire.
- « T2 » devient passant.
- La bobine « RL1 » devient sous tension et commande la fermeture du contact de la sortie « Q0.1 ».
- Donc pour commander un API, l'unité de commande doit envoyer :
 - Un « 1 » logique pour actionner une sortie API
 - Un « 0 » logique pour stopper la commande d'une sortie API

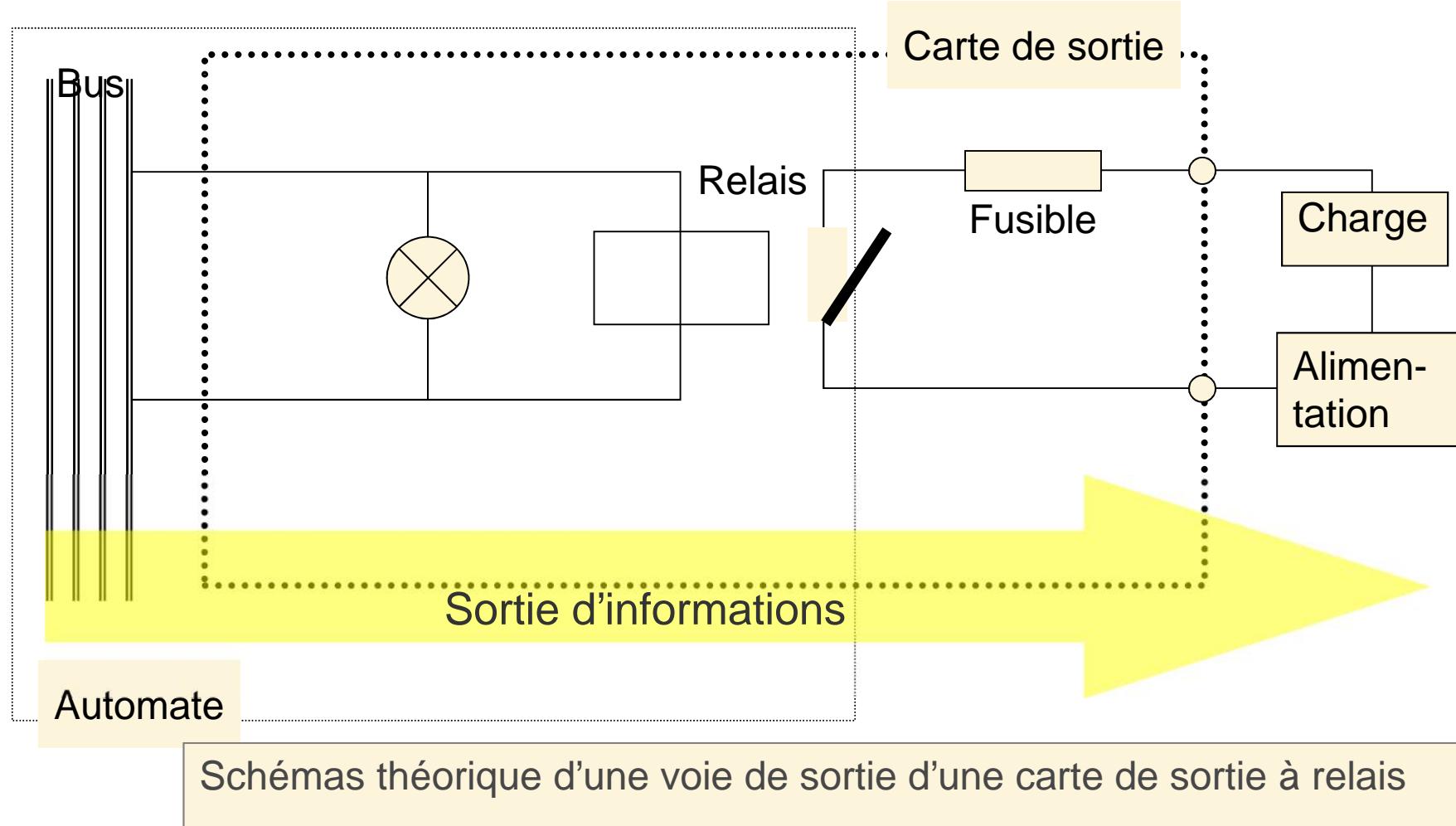


Les interfaces entrées/sorties:





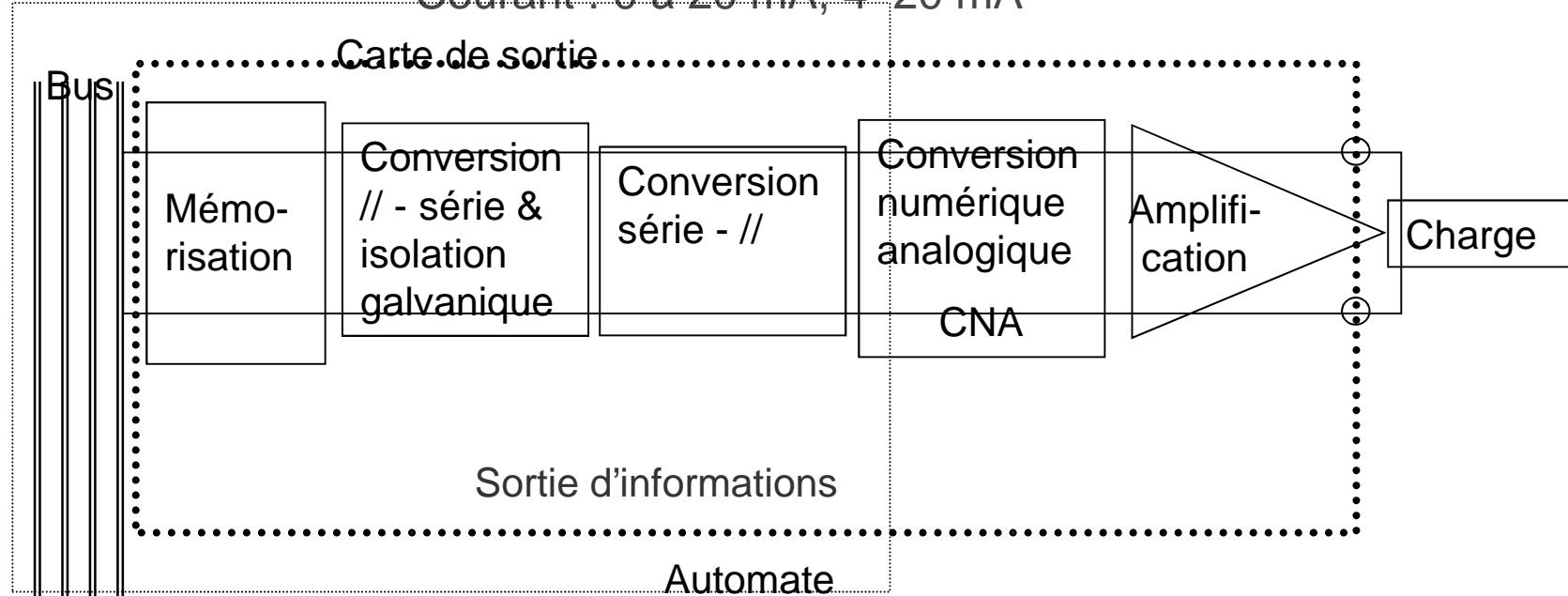
Cartes de sorties TOR





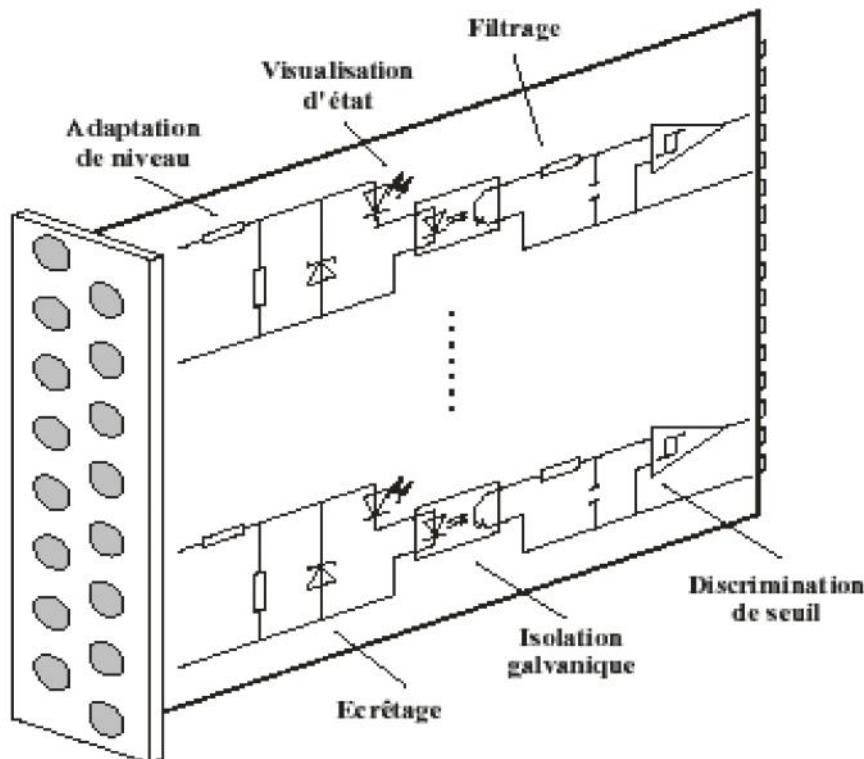
Cartes de sorties NA

- Utilisées pour piloter en tension ou en courant des actionneurs de type variateurs de vitesse, électrovannes à commande proportionnelle, ...
- Elles donnent l'image analogique d'une valeur numérique codée sur un mot (en général 8 à 16 bits) définie par le programme de l'API :
- Elles sont caractérisées par l'amplitude et la nature du signal :
 - Tension : 0 à 10 V, -10 à +10 V, ...
 - Courant : 0 à 20 mA, 4 à 20 mA





STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE D'ENTREES/SORTIES



Signaux de processus:

24, 48 V (CC ou AC)
110, 220 V (AC)

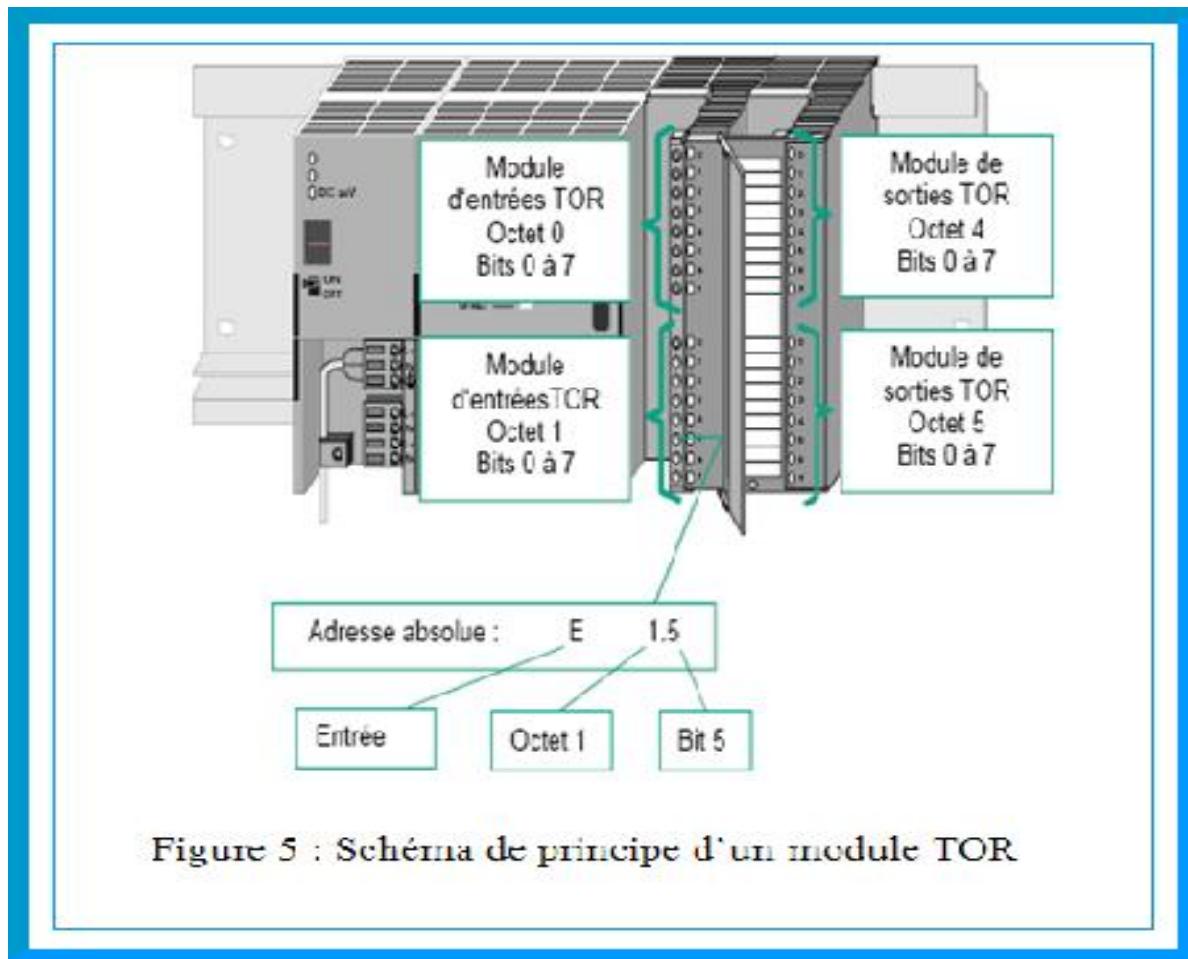
Signaux électroniques:

5 V (CC)



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE D'ENTREES/SORTIES

La figure suivant donne une idée concrète sur un module TOR industriel.





Interfaces de communication

- "Prises" pour connecter :
 - La console de programmation, (programmation , mise au point, maintenance)
 - Une console de test, (mise au point, maintenance)
 - D'autres API (Réseau Local Industriel)
 - Un PC (Supervision, Réseau Local Industriel)
- Cette "prise" peut être :
 - Une liaison série simple,
 - Un Port USB,
 - Un Port Ethernet
 - Une liaison série avec un protocole de communication spécifique : Réseau local industriel (Modbus, Profibus, ...), Bus de terrain (CAN, FIP, Profibus, ...)
- Ces interfaces peuvent être :
 - Intégrées à l'UC,
 - Des cartes d'extension



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE LE BUS

C'est un ensemble de conducteurs qui réalisent la liaison entre les différents éléments de l'automate. Dans un automate modulaire, il se présente sous forme d'un circuit imprimé situé au fond du bac et supporte des connecteurs sur lesquels viennent s'enficher les différents modules : processeur, extension mémoire, interfaces et coupleurs.

Le bus est organisé en plusieurs sous ensembles destinés chacun à véhiculer un type bien défini d'informations :

- Bus de données.
- Bus d'adresses.
- Bus de contrôle pour les signaux de service tels que tops de synchronisation, sens des échanges, contrôle de validité des échanges, etc..
- Bus de distribution des tensions issues du bloc d'alimentation.



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE

Alimentation

Elle élabore à partir d'un réseau 220V en courant alternatif, ou d'une source 24V en courant continu, les tensions internes distribuées aux modules de l'automate.

A fin d'assurer le niveau de sûreté requis, elle comporte des dispositifs de détection de baisse ou de coupure de la tension réseau, et de surveillance des tensions internes. En cas de défaut, ces dispositifs peuvent lancer une procédure prioritaire de sauvegarde.

Les schémas de principe et raccordement sont, respectivement, illustrés par les figures suivantes:



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE

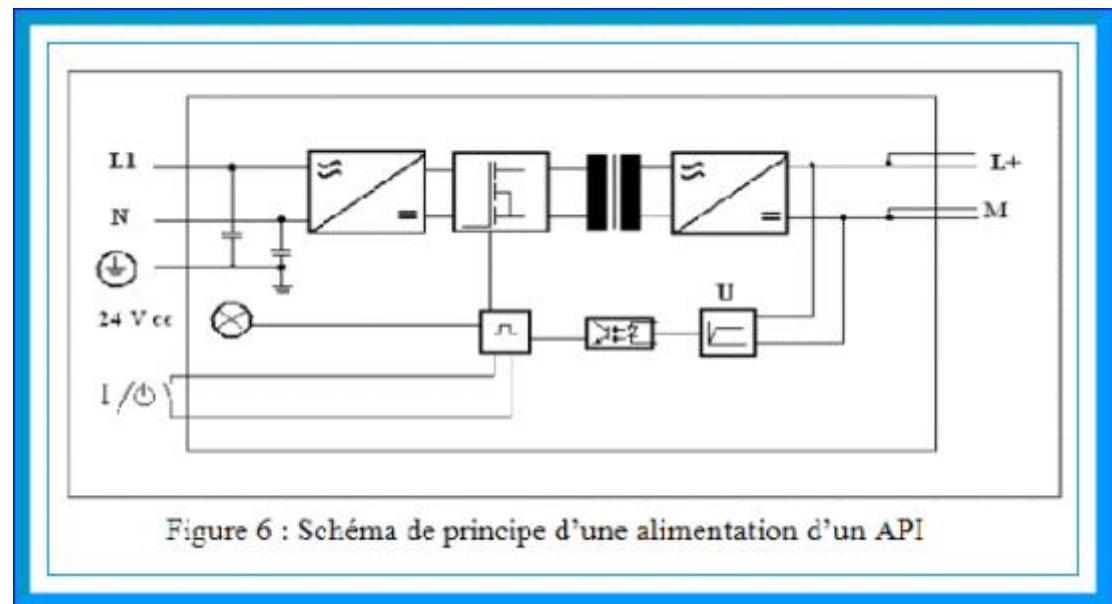
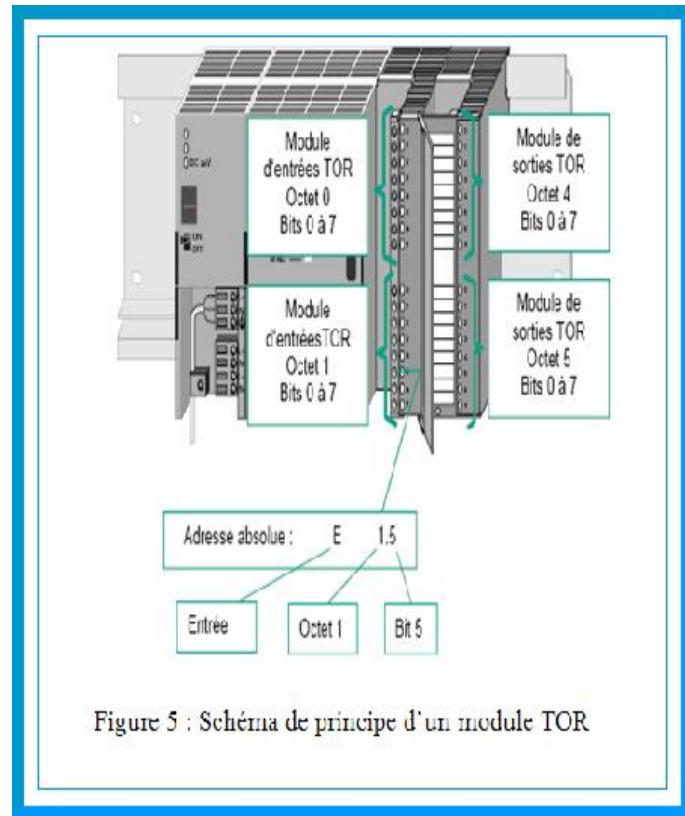
Alimentation

Le module d'alimentation transforme l'énergie externe provenant du réseau en la mettant en forme afin de fournir aux différents modules de l'API les niveaux de tension nécessaires à leur bon fonctionnement. Plusieurs niveaux de tension peuvent être utilisés par les circuits internes (3v, 5v, 12v, 24v...) il sera dimensionné en fonction des consommations des différentes parties.



STRUCTURE INTERNE D'UN AUTOMATE PROGRAMMABLE

Alimentation





Choix d'un API

Le choix d'un API est fonction de la partie commande à programmer. On doit tenir compte de plusieurs critères.

- ✓ Nombres d'entrées/sorties intégrés.
- ✓ Temps de traitement (scrutation).
- ✓ Capacité de la mémoire.
- ✓ Nombre de compteurs.
- ✓ Nombre de temporiseurs



Avantages des API

évolutivité

très favorable au évolution. très utilisé en reconstruction d'armoire.

fonctions

assure les fonctions Conduites, Dialogue, Communication et Sûreté.

taille des applications

gamme importante d'automate

vitesse

temps de cycle de quelque ms

modularité

haute modularité. présentation en rack



développement
d'une application
et documentation

très facile avec des outils de
programmation de plus en plus puissant

architecture de
commande

centralisée ou décentralisée avec
l'apparition d'une offre importante en
choix de réseaux , bus de terrain, blocs
E/S déportées.

mise en oeuvre

mise au point rendu plus facile avec
l'apparition des outils de simulation de
PO

maintenance

échange standards et aide au
diagnostique intégrée

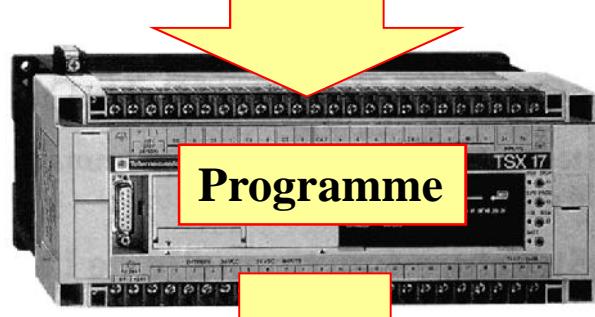
portabilité d'une
application

norme IEC 1131



Fonction d'un API

Informations
(capteurs, dialogue)
Entrées



Traiter les informations entrantes pour émettre des ordres de sorties en fonction d'un programme.

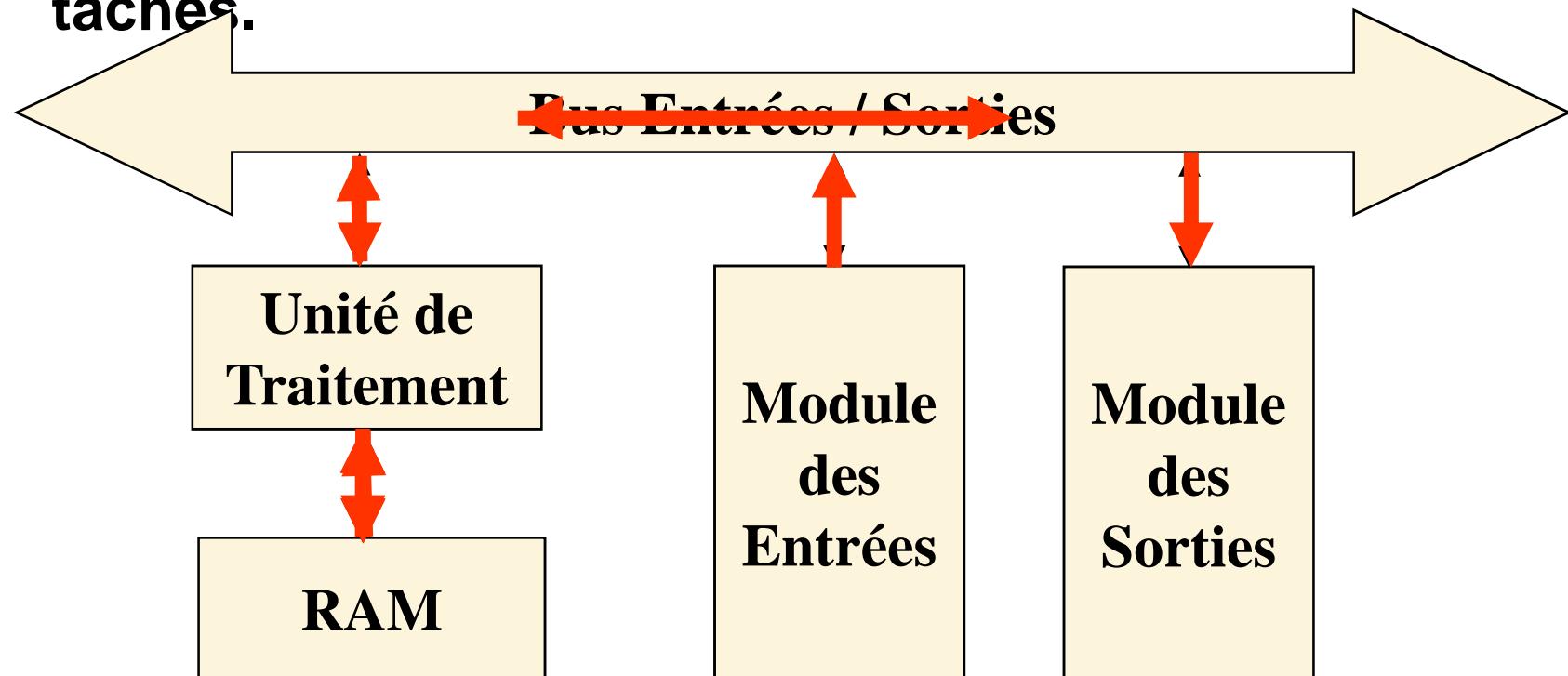
Sorties
Ordres
(préactionneurs, dialogue)

Automates
Programmables



Fonctionnement API

Le cycle de fonctionnement est organisé en 4 tâches.

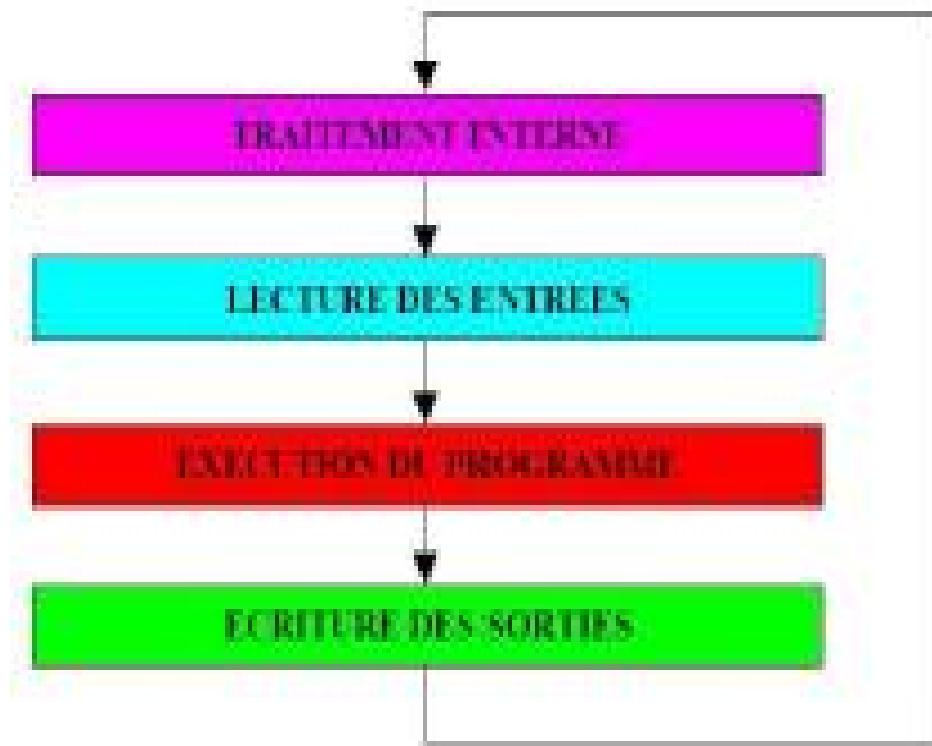


Tâche 4 : Affectation des sorties (émission des ordres)



FONCTIONNEMENT DE L'API

Le cycle de fonctionnement de l'API est décrit ci-dessous.



Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique).

Traitemet interne : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).

Lecture des entrées : L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées.

Exécution du programme : L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.

Ecriture des sorties : L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties.



FONCTIONNEMENT DE L'API

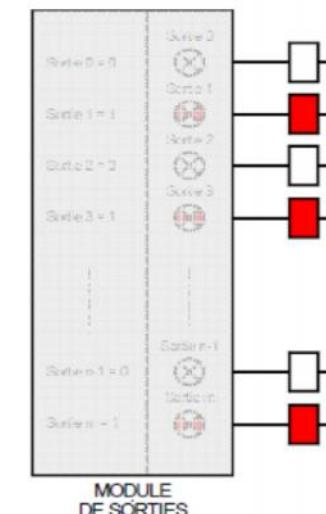
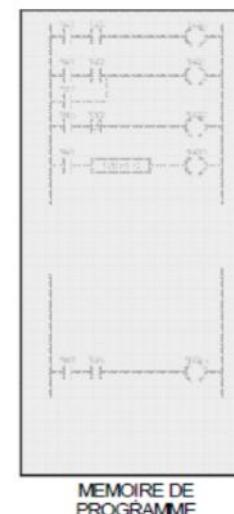
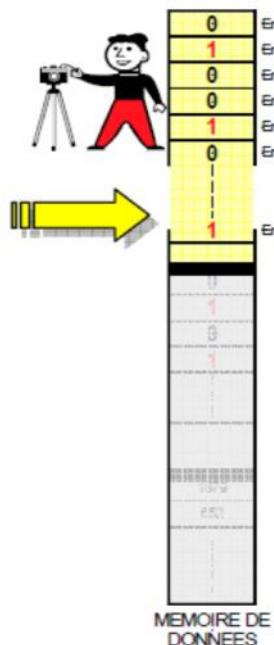
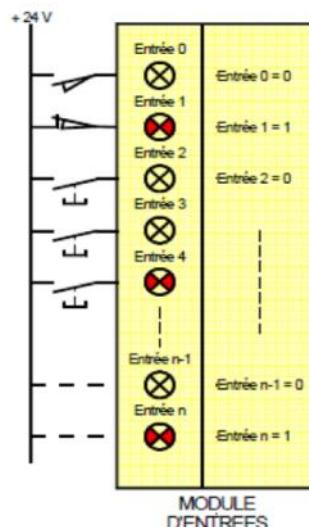
PHASE 1

PHOTOGRAPHIE DES ENTREES

Lorsque l'API est en fonctionnement, trois phases se succèdent :

Durant cette phase qui dure quelques micro-secondes :

- les entrées sont « photographiées » et leurs états logiques sont stockés dans une zone spécifique de la mémoire de donnée.
- Le programme n'est pas scruté.
- Les sorties ne sont pas mises à jour.



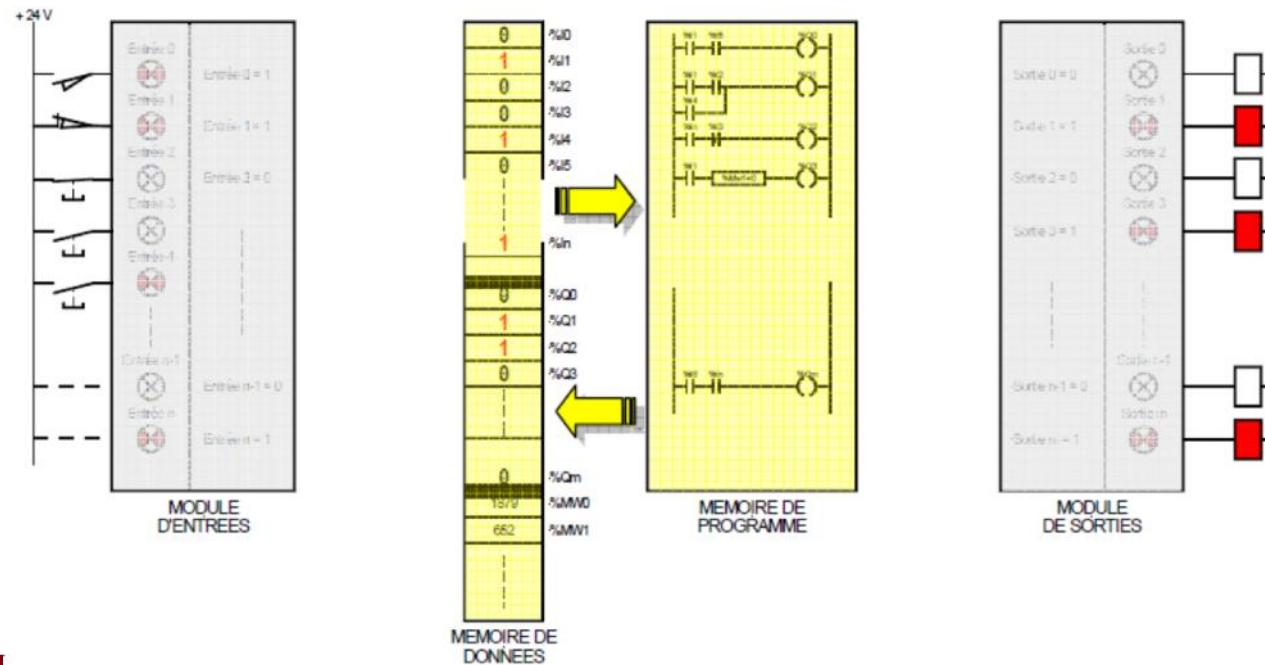


FONCTIONNEMENT DE L'API

PHASE 2**EXECUTION DU PROGRAMME**

Durant cette phase qui dure quelques milli-secondes :

- Les instructions de programme sont exécutées une à une. Si l'état d'une entrée doit être lu par le programme, c'est la valeur stockée dans la mémoire de données qui est utilisée.
- Le programme Détermine l'état des sorties et stocke ces valeurs dans une zone de la mémoire de données réservée aux sorties.
- Les entrées ne sont pas scrutées.
- Les sorties ne sont pas mises à jour.





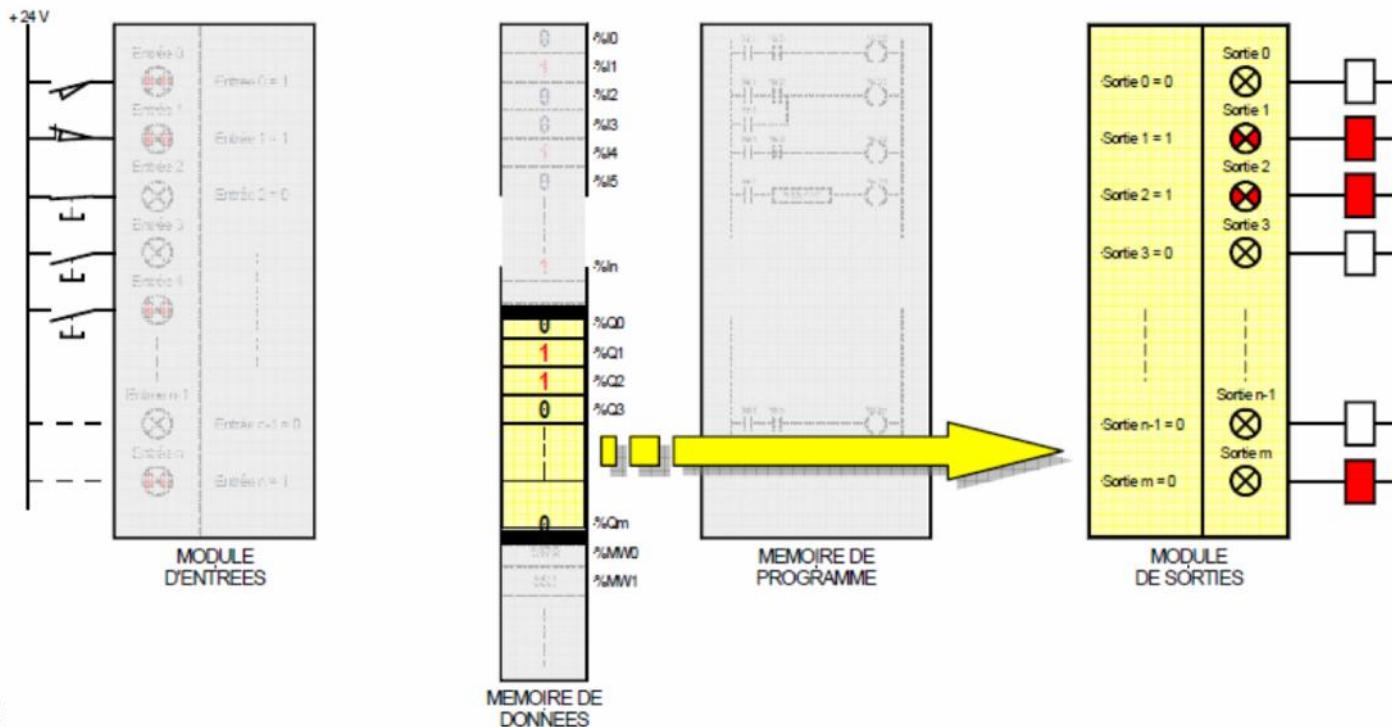
FONCTIONNEMENT DE L'API

PHASE 3

MISE À JOUR DES SORTIES

Durant cette phase qui dure quelques micro-secondes :

- Les états des sorties mémorisés précédemment dans la mémoire de données sont reportés sur le module de sorties.
- Les entrées ne sont pas scrutées.
- Le programme n'est pas exécuté.





FONCTIONNEMENT D'UN API MODES DE FONCTIONNEMENT

Généralement, un automate possède **2 modes** de fonctionnement principaux:

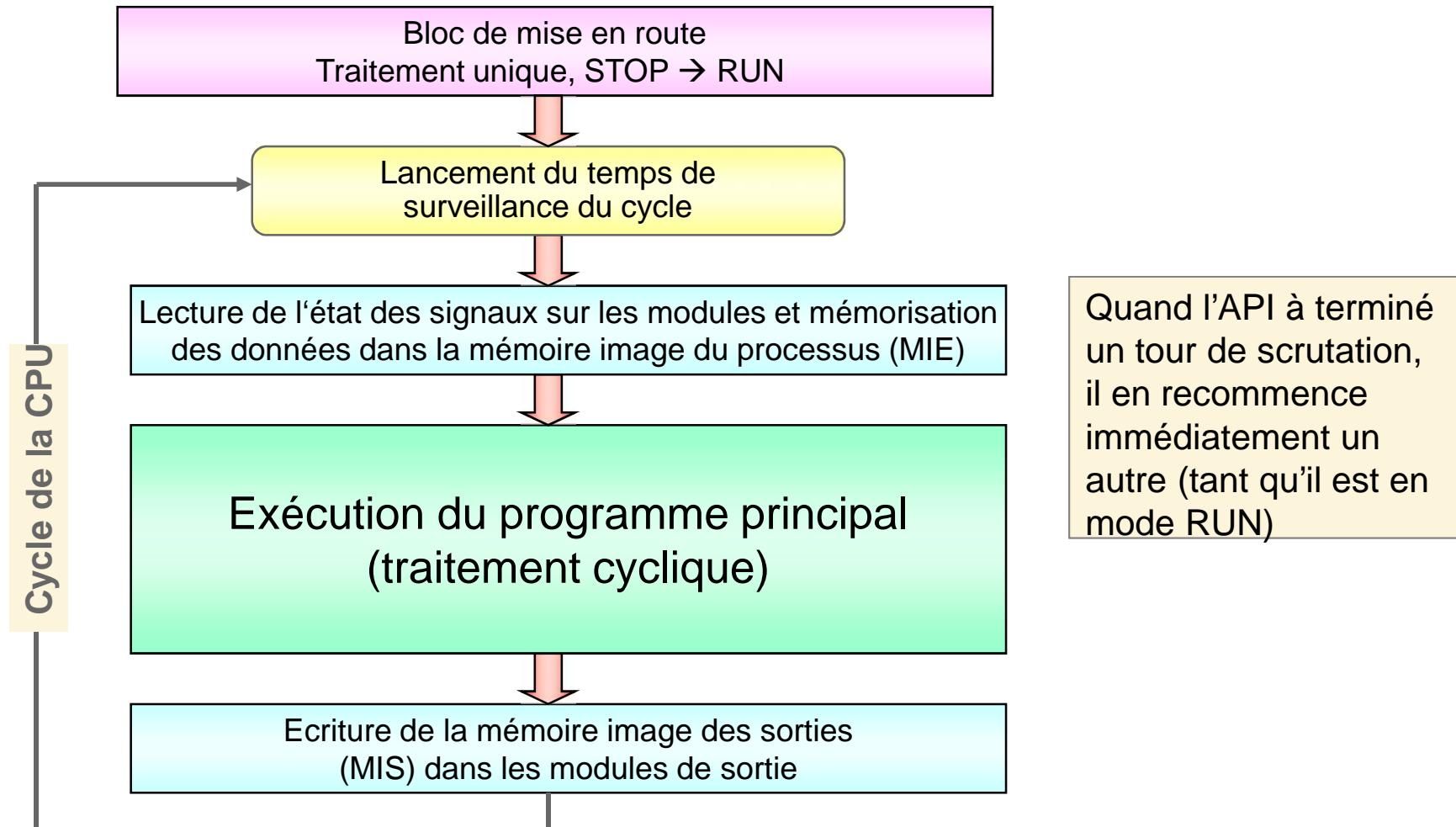
- Mode **STOP** ou **MANUEL** : L'automate n'exécute pas le programme.
- Mode **RUN** ou **START** ou **AUTOMATIQUE**: L'automate exécute le programme en exécutant des cycles de scrutination.

Le changement de mode de fonctionnement peut être effectué par une ou plusieurs des options suivantes:

- Un commutateur ou une clé disposé sur la CPU de l'API,
- Une commande sur la console de programmation,



FONCTIONNEMENT D'UN API CYCLE DE SCRUTATION STANDARD





FONCTIONNEMENT D'UN API SURVEILLANCE DE CYCLE- WATCH DOG – CHIEN DE GARDE

C'est une minuterie de valeur (fixe ou réglable) légèrement supérieure au temps de cycle maximum autorisé pour un automate donné.

Cette minuterie est enclenchée à **chaque début de cycle**.

Si le cycle de scrutination se termine avant la fin de la minuterie, l'automate continue sa scrutination.

Si la minuterie se termine avant la fin du cycle de scrutination, une alarme se déclenche, se traduisant, suivant l'automate, par :

L'allumage ou le clignotement d'un voyant,

Le positionnement d'un indicateur dans la mémoire de l'automate (bit système),

La retombée d'un contact électrique (souvent câblé dans la chaîne d'arrêt d'urgence du système à automatiser),

Le passage de l'API en mode arrêt.

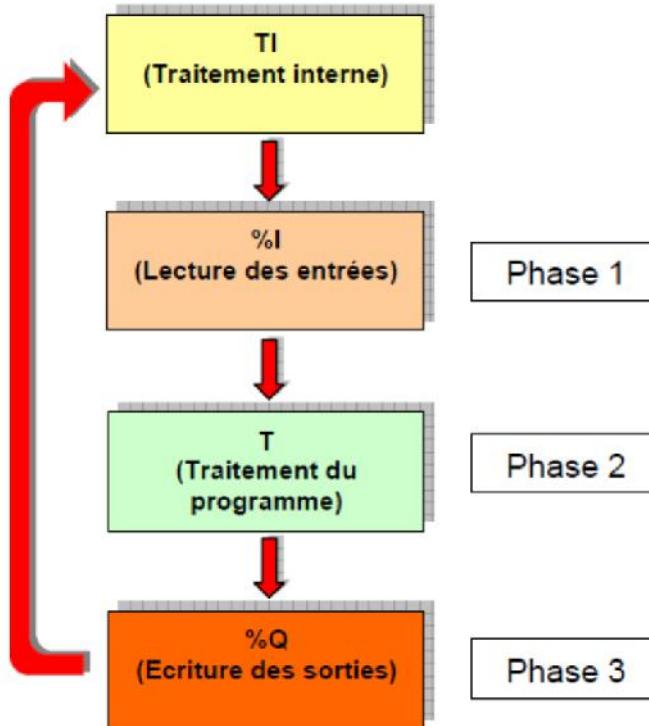
Ce défaut peut être provoqué par plusieurs causes :

Problème d'accès sur une carte d'entrées/sorties,

Boucle sans fin dans le programme, consécutive à un saut en arrière ou une instruction répétitive dont la condition de fin de test est mal définie,



Fonctionnement d'un A.P.I.



Si pour une raison quelconque le temps de cycle mesuré par le chien de garde est supérieur au temps de cycle maxi configuré, l'API signale le défaut et arrête le traitement.

L'enchaînement des trois phases se répète sans cesse de façon cyclique lorsque l'API est en fonctionnement :

1. Lecture des entrées (%I)
2. Traitement du programme (T)
3. Ecriture des sorties (%Q)

Avant chaque cycle l'API effectue des traitements internes afin de vérifier ses circuits et les sollicitations extérieures.

Le temps de cycle de l'ordre de quelques milli-secondes est surveillé par un circuit électronique appelé «Chien de garde».



FONCTIONNEMENT D'UN API SURVEILLANCE DE CYCLE- WATCH DOG – CHIEN DE GARDE

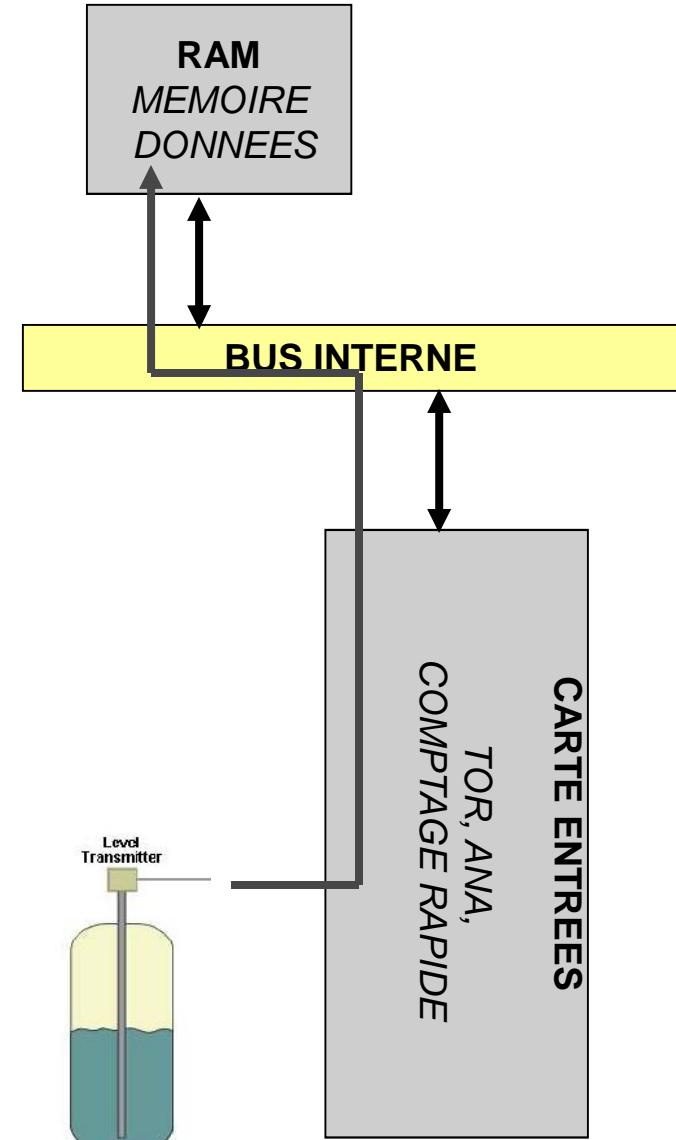
- Système chargé de surveiller et de contrôler le bon fonctionnement du matériel et du logiciel
- Fonction de « chien de garde » (watch dog) qui contrôle le cycle de l'automate
- Principe:
 - Obliger l'automate à envoyer à chaque cycle une impulsion au système de surveillance
 - Vérifier que le temps entre 2 impulsions ne dépasse pas une limite
 - Déetecter ainsi panne de processeur, bouclage intempestif...

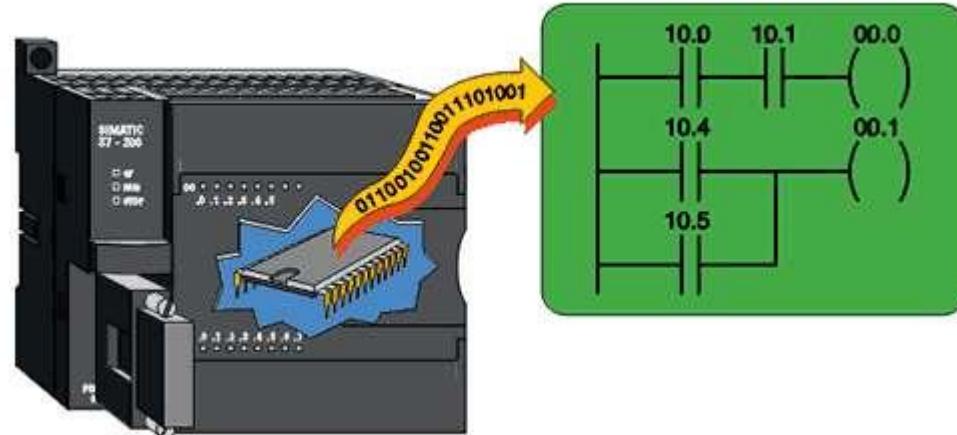


Traitement : définitions



écriture en mémoire de l'état des informations présentes sur les entrées (réalise une image du monde extérieur)

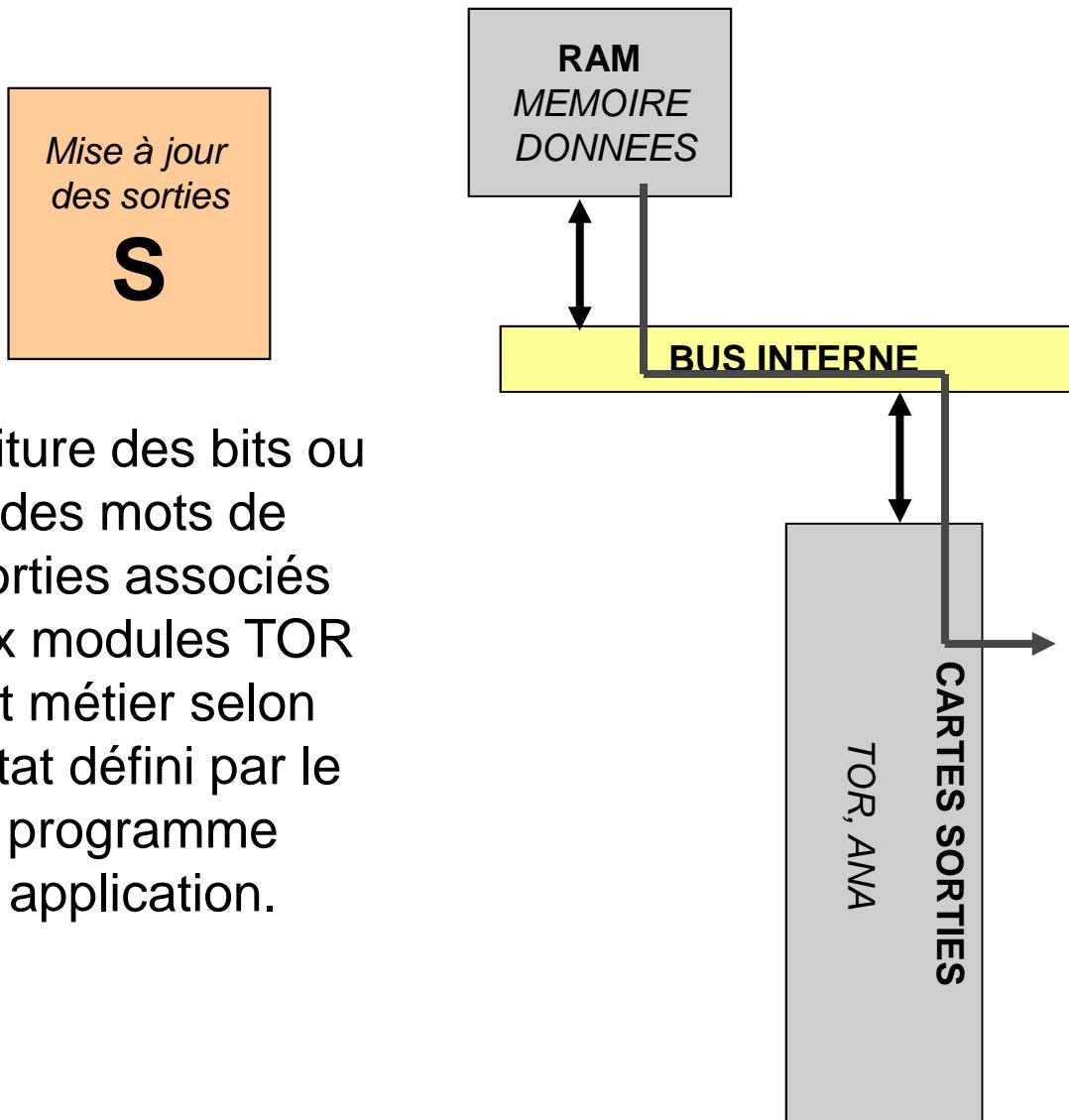




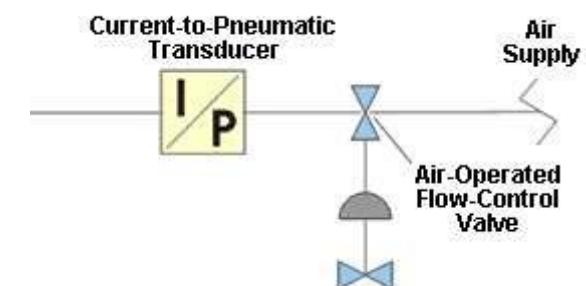
*Traitemen
t du programme*

T

exécution du
programme
application, écrit
par l'utilisateur.



écriture des bits ou
des mots de
sorties associés
aux modules TOR
et métier selon
l'état défini par le
programme
application.





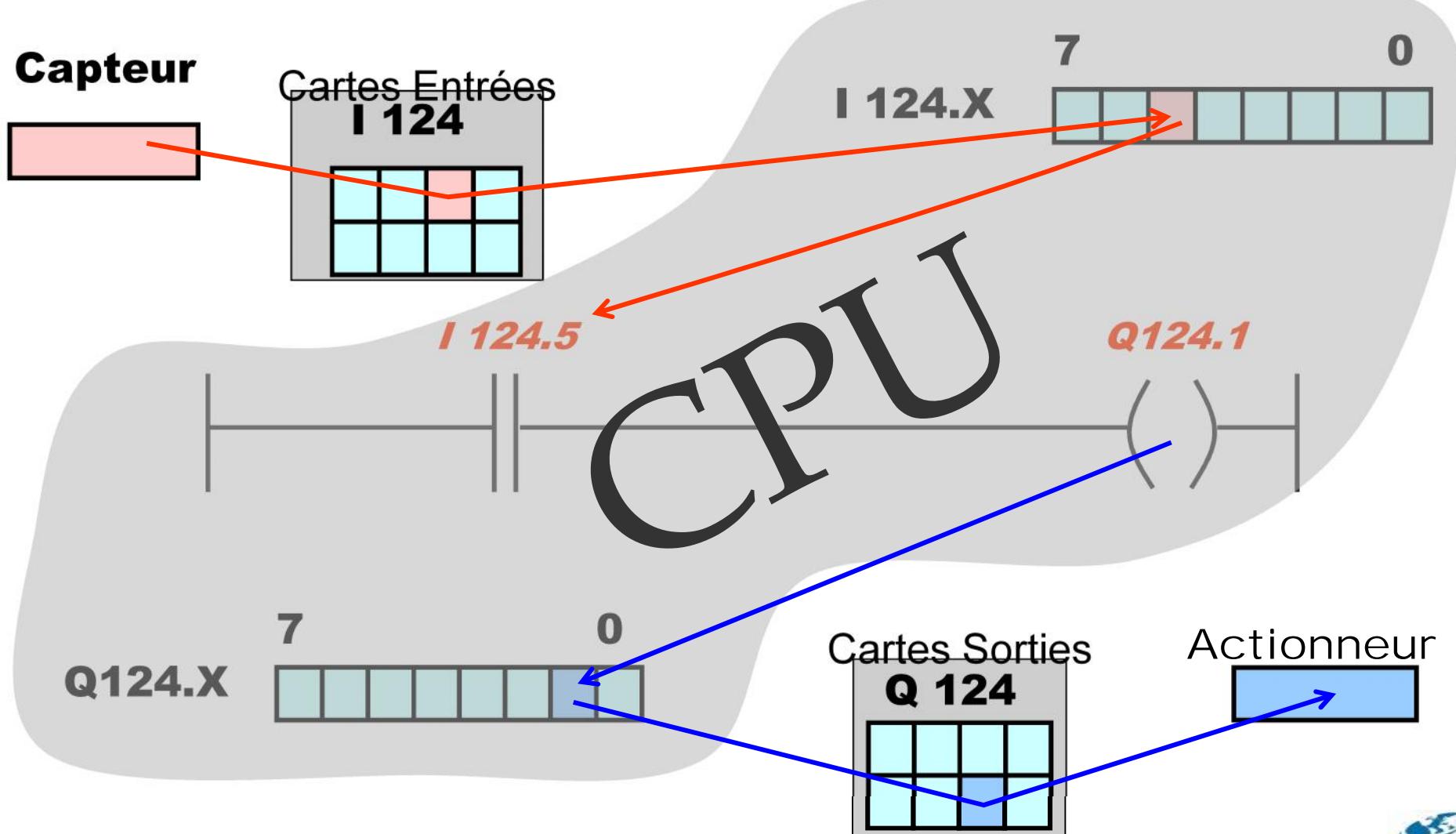
L'ensemble constitue une tâche



**Temps de cycle
 $T_C = T_E + T_T + T_S$**



Exemple

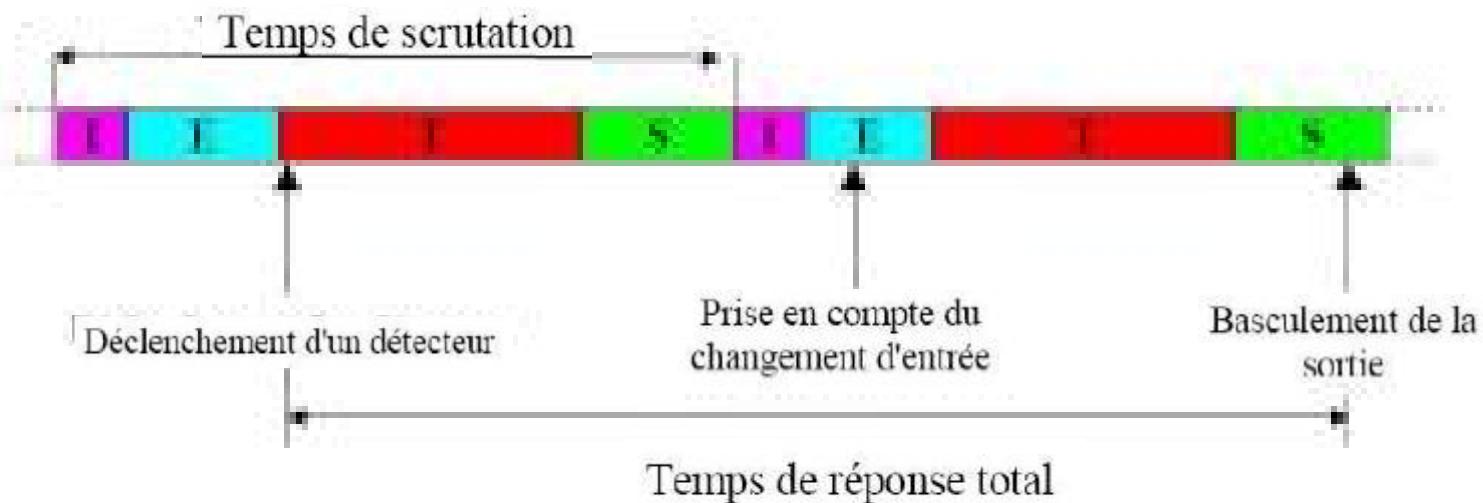




FONCTIONNEMENT D'UN API

Le temps de scrutation est le temps mis par l'automate pour traiter l'ensemble des quatre opérations. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards.

Le temps de réponse total (TRT) est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante.





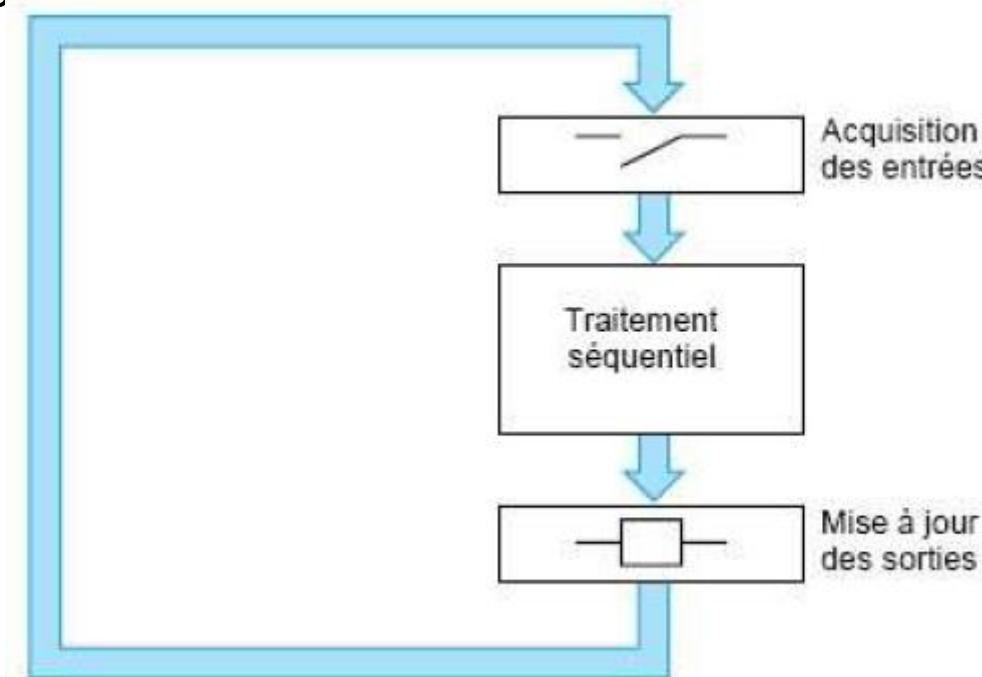
FONCTIONNEMENT D'UN API

Le temps de réponse total est au plus égal à deux fois le temps de scrutation (sans traitement particulier). Le temps de scrutation est directement lié au programme implanté. Ce temps peut être fixé à une valeur précise (fonctionnement périodique), le système indiquera alors tout dépassement de période. Dans certains cas, on ne peut admettre un temps de réponse aussi long pour certaines entrées : ces entrées pourront alors être traitées par l'automate comme des événements (traitement événementiel) et prises en compte en priorité (exemples : problème de sécurité, coupure d'alimentation ...). Certains automates sont également pourvus d'entrées rapides qui sont prises en compte avant le traitement séquentiel mais le traitement événementiel reste prioritaire.



FONCTIONNEMENT D'UN API

Une structure **mono tâche** : L'automate exécute une seule tâche : la tâche « maître ». Cette tâche (séquentielle) non périodique, est surveillée par un chien de garde logiciel fixé à 150 ms. Elle se programme en Grafcet et/ou en langage à contacts.



L'exécution de la tâche maître peut être :

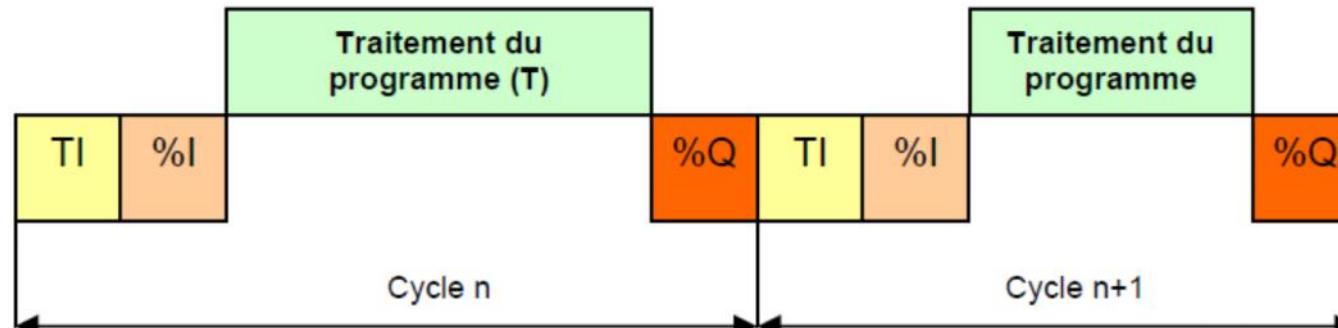
- **Cyclique**
- ou **périodique**

Le programme d'une application monotâche est associé à une seule tâche utilisateur la tâche maître MAST.



FONCTIONNEMENT D'UN API

Exécution CYCLIQUE

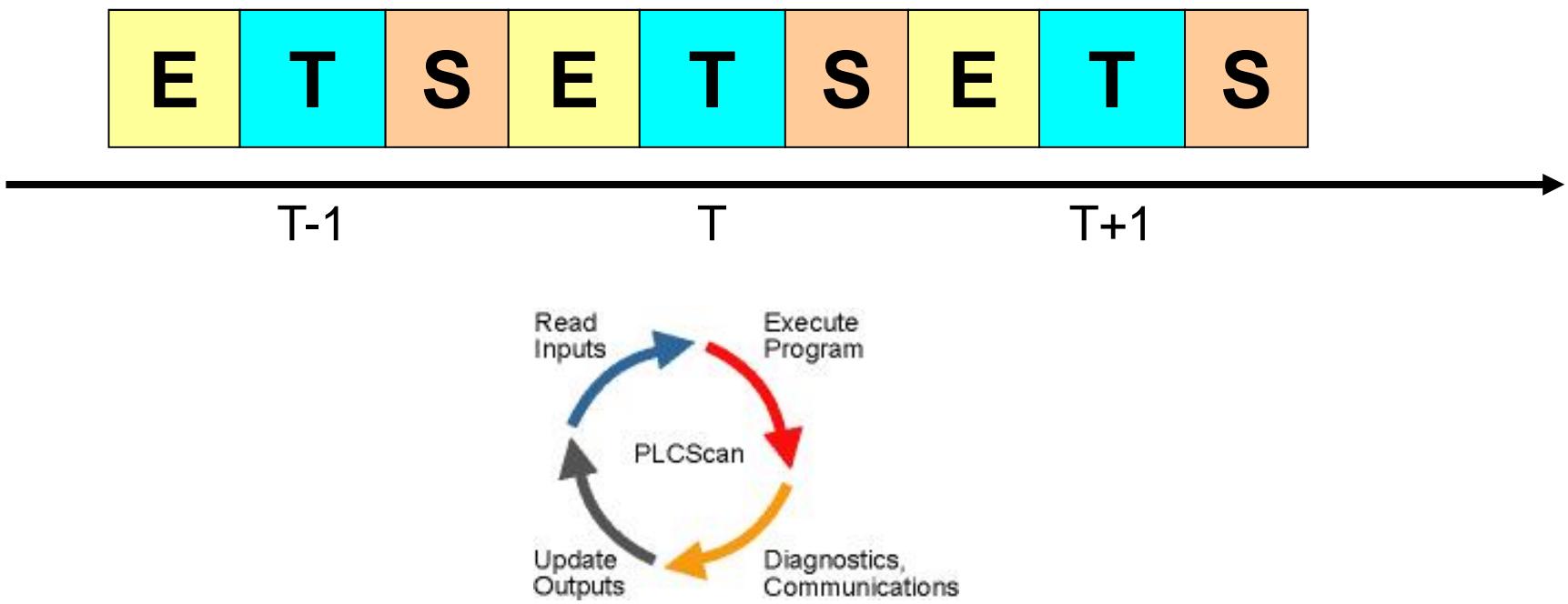


Repère	Phase	Description
TI	Traitement Interne	Le système réalise implicitement la surveillance de l'automate (gestion des bits et mots système, mise à jour des valeurs courantes de l'horodateur, mise à jour des voyants d'état, détection des passages RUN/STOP, ...) et le traitement des requêtes en provenance du terminal (modifications et animation).
%I	Acquisition des Entrées	Ecriture en mémoire de l'état des informations présentes sur les entrées des modules TOR et métier associées à la tâche.
T	Traitement du programme	Exécution du programme application, écrit par l'utilisateur.
%Q	Mise à jour des Sorties	Ecriture des bits ou des mots de sorties associés aux modules TOR et métier associés à la tâche selon l'état défini par le programme application.

La durée d'un cycle ne doit pas dépasser le temps réglé pour le chien de garde.



Ce type de fonctionnement consiste à enchaîner les cycles les uns après les autres.

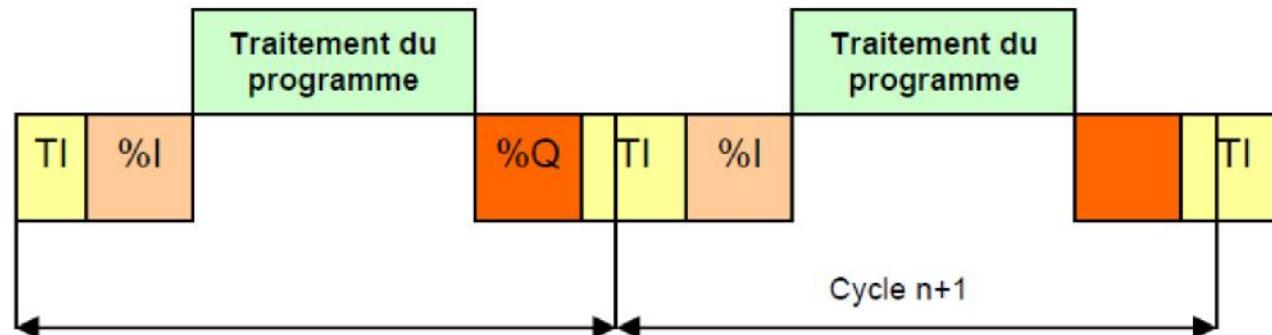


fonctionnement mono-tâche asynchrone (ou cyclique)



FONCTIONNEMENT D'UN API

Exécution PERIODIQUE



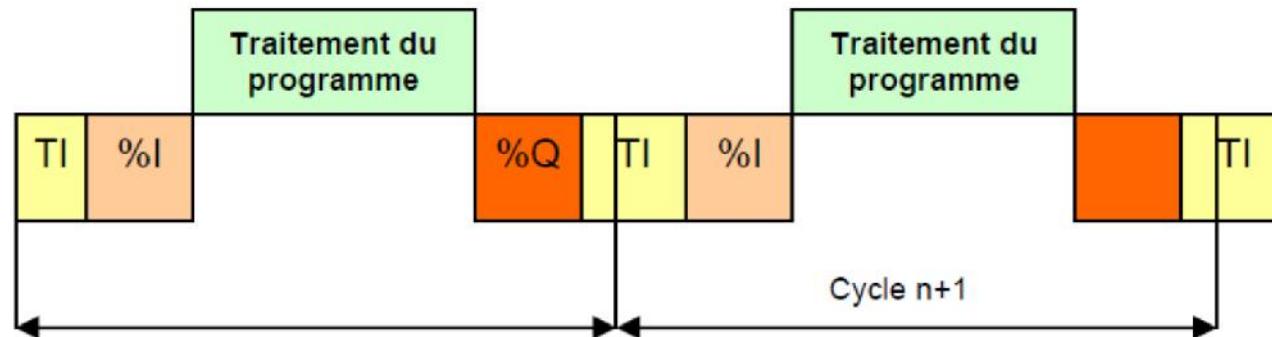
Dans ce mode de fonctionnement, l'acquisition des entrées, le traitement du programme application et la mise à jour des sorties s'effectuent de façon périodique selon un temps défini en configuration (de 1 à 255 ms). En début de cycle automatique, un temporisateur dont la valeur courante est initialisée à la période définie en configuration, commence à décompter. Le cycle automatique doit se terminer avant l'expiration de ce temporisateur qui relance un nouveau cycle.

Le processeur effectue dans l'ordre le traitement interne, l'acquisition des entrées, le traitement du programme application et la mise à jour des sorties.



FONCTIONNEMENT D'UN API

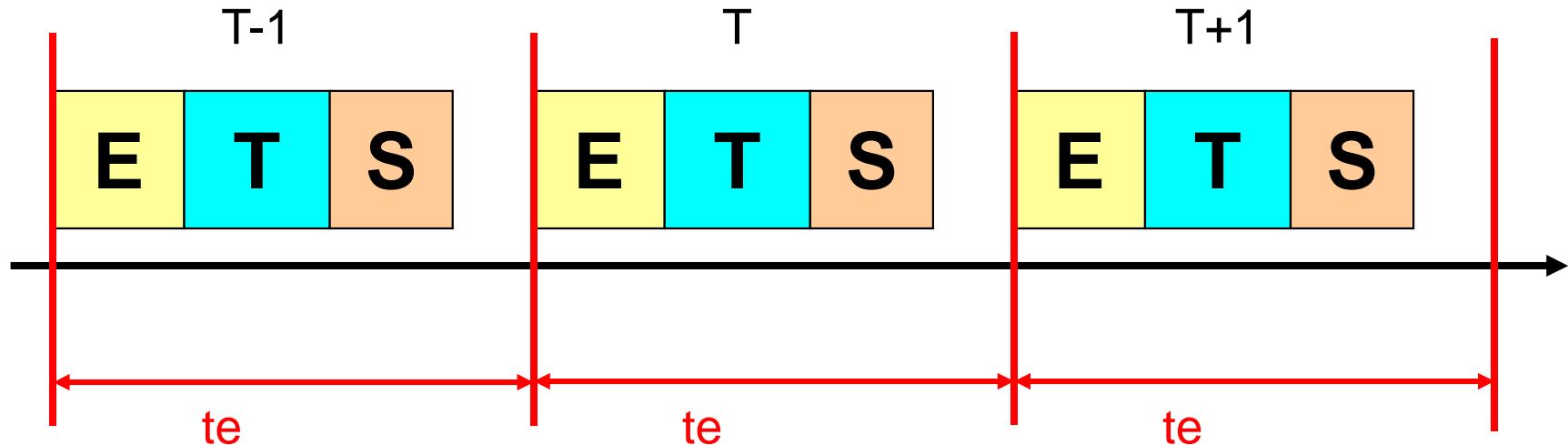
Exécution PERIODIQUE (suite)



- Si la période n'est pas encore terminée, le processeur complète son cycle de fonctionnement jusqu'à la fin de la période par du traitement interne.
- Si le temps de fonctionnement devient supérieur à celui affecté à la période, l'automate signale un débordement de période par la mise à l'état 1 du bit système %S19 de la tâche, le traitement se poursuit et est exécuté dans sa totalité (il ne doit pas dépasser néanmoins le temps limite du chien de garde). Le cycle suivant est enchaîné après l'écriture implicite des sorties du cycle en cours.



Dans ce mode de fonctionnement, l'acquisition des entrées, le traitement du programme et la mise à jour des sorties s'effectue de façon périodique **te ms** selon un temps défini par configuration API .

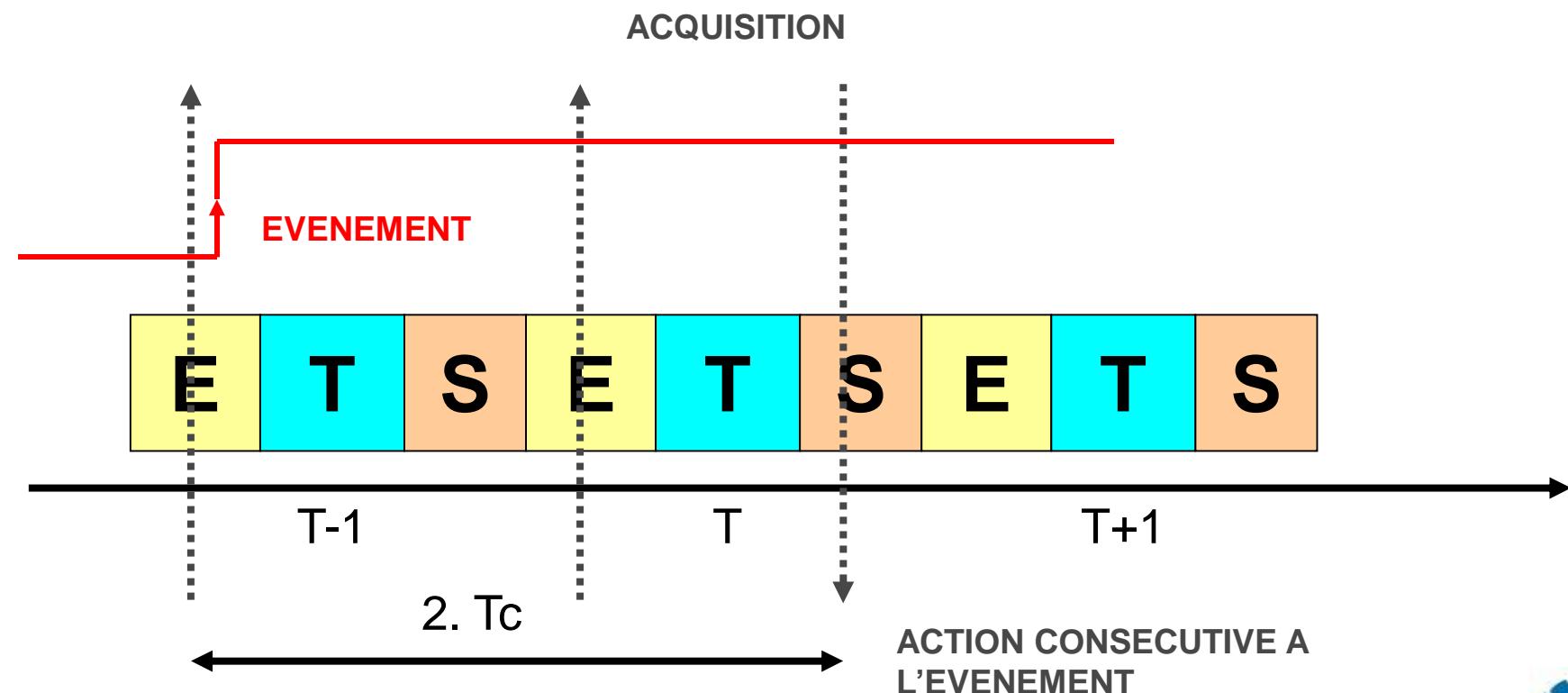


fonctionnement mono-tâche **synchrone (périodique)**



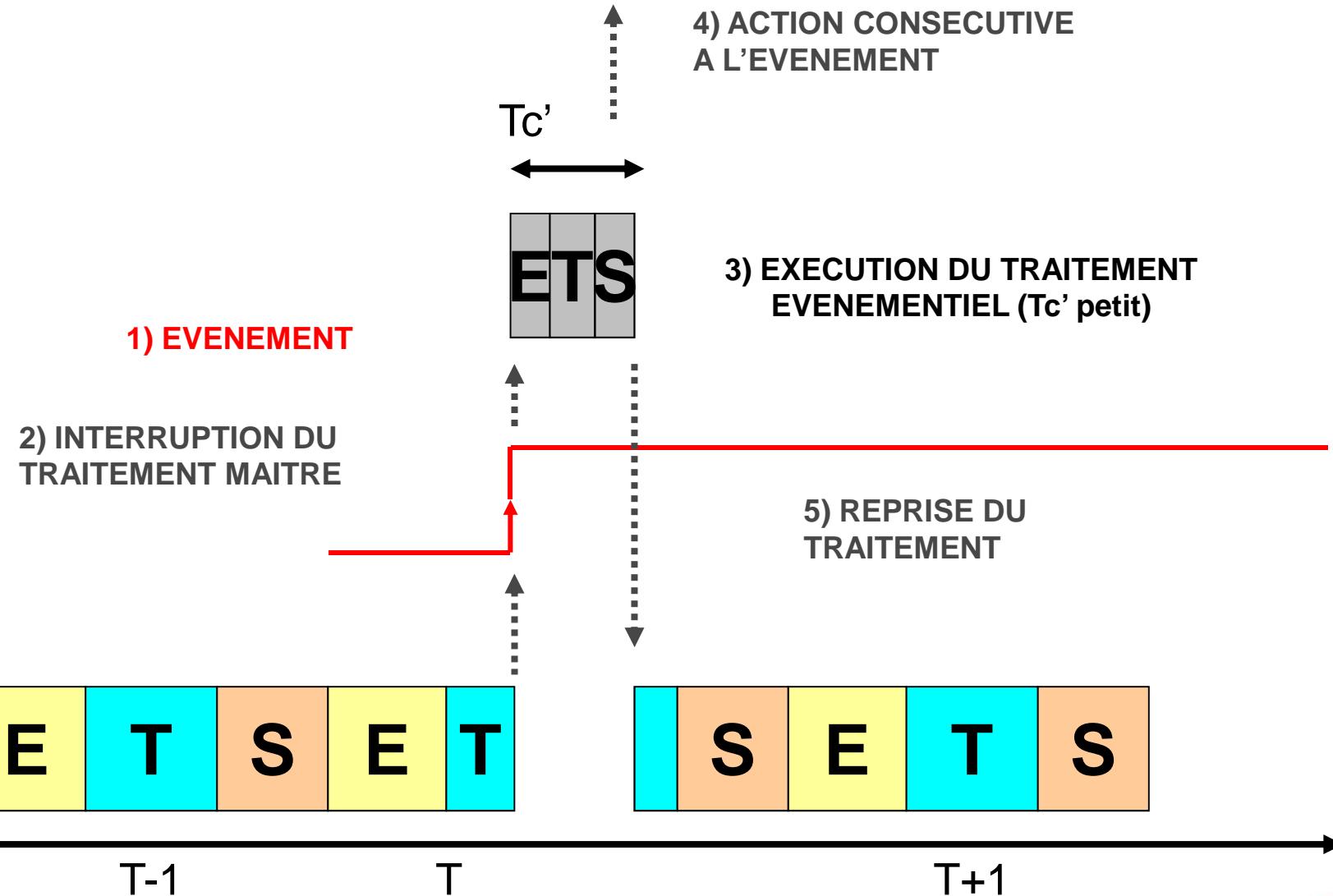
Retard dans le traitement de l'événement

Les deux modes de traitements (cyclique ou périodique) sont appelé différé. Dans le pire des cas, il peut s'écouler à peu près 2 fois le temps de cycle moyen avant que l'UT réagisse à l'apparition d'un événement





Les interruptions





Les interruptions

Les traitements événementiels permettent donc de prendre en compte des événements de commande et de les traiter le plus rapidement possible.

Des instructions du langage utilisées dans le programme application, permettent de masquer ou démasquer ces traitements événementiels.

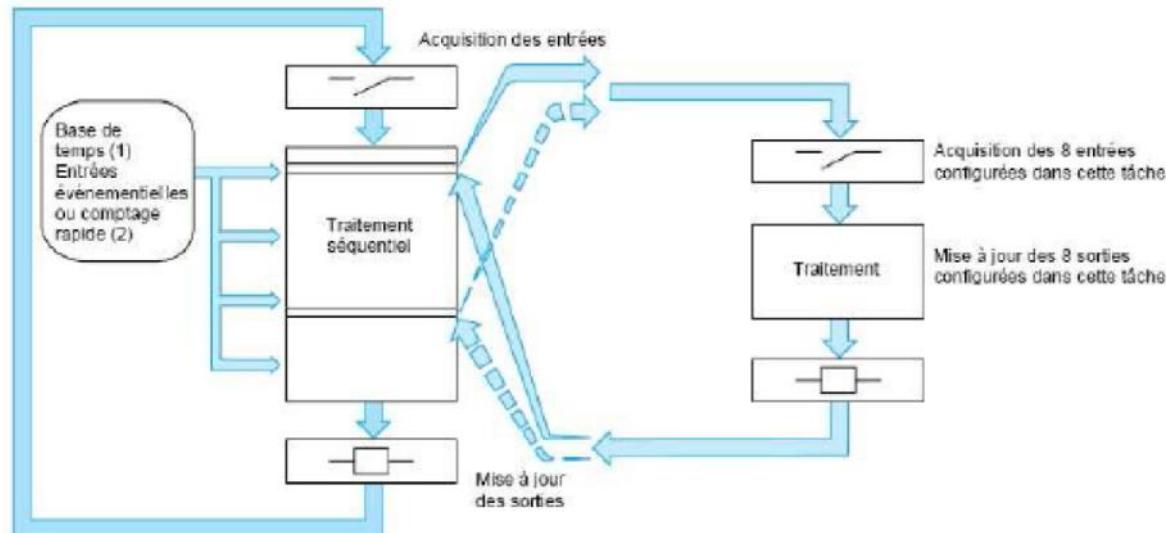
Ces tâches sont prioritaires. Le traitement, qu'elles doivent assurer, est nécessairement court afin de ne pas perturber l'exécution des autres tâches.

La configuration permet de choisir les voies des modules d'entrées qui sont rafraîchies en début de traitement et les voies de sorties qui sont mises à jour en fin de traitement. Les données associées à la voie qui a déclenché l'interruption sont rafraîchies automatiquement.



FONCTIONNEMENT D'UN API

Une structure **multitâche** : A la tâche précédente peut être rajouté deux autres tâches : la tâche rapide et la tâche événementielle.

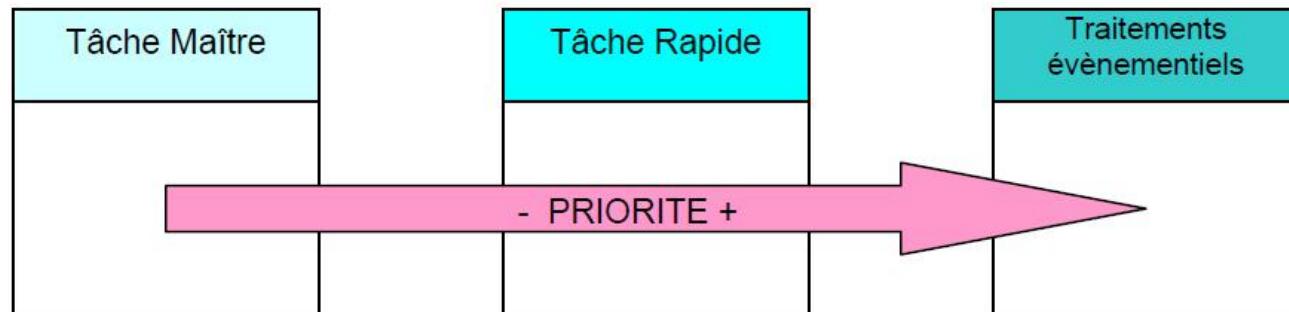


Tâche maître et **Tâche rapide** (cette tâche autorise le traitement d'informations de courte durée avec une fréquence d'exécution élevée. Elle se programme uniquement en langage à contacts).



FONCTIONNEMENT D'UN API

Structure logicielle MULTI-TACHES



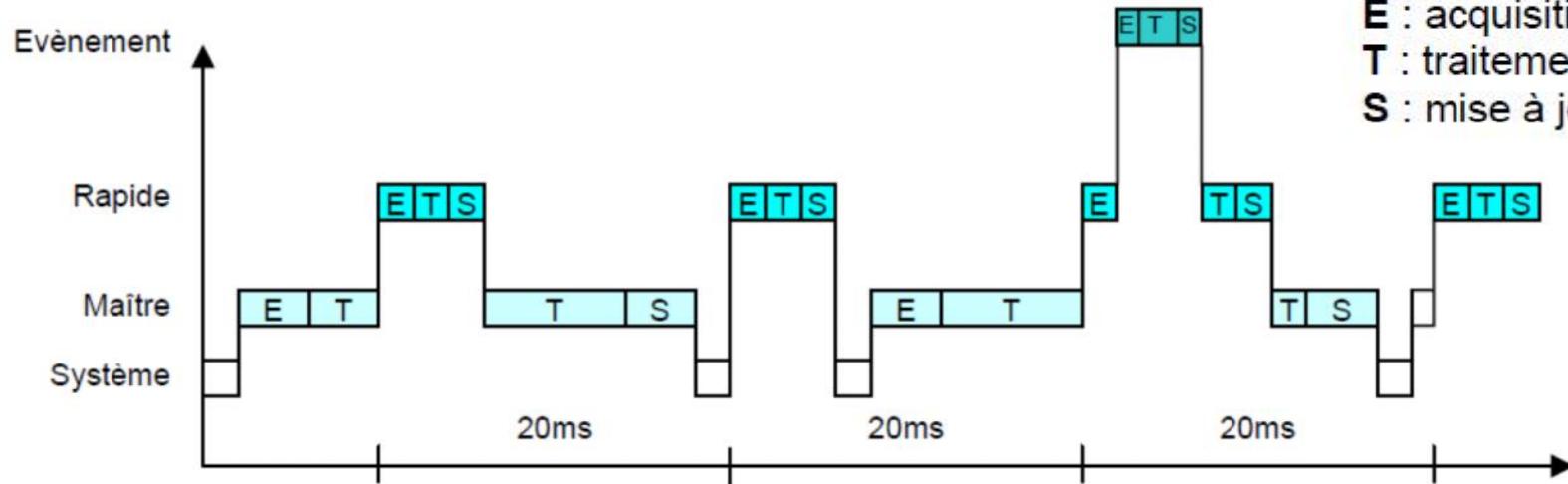
La tâche maître est par défaut active.

La tâche rapide est par défaut active si elle est programmée.

Le traitement événementiel est activé lors d'apparition de l'événement qui lui a été associé.

Légende :

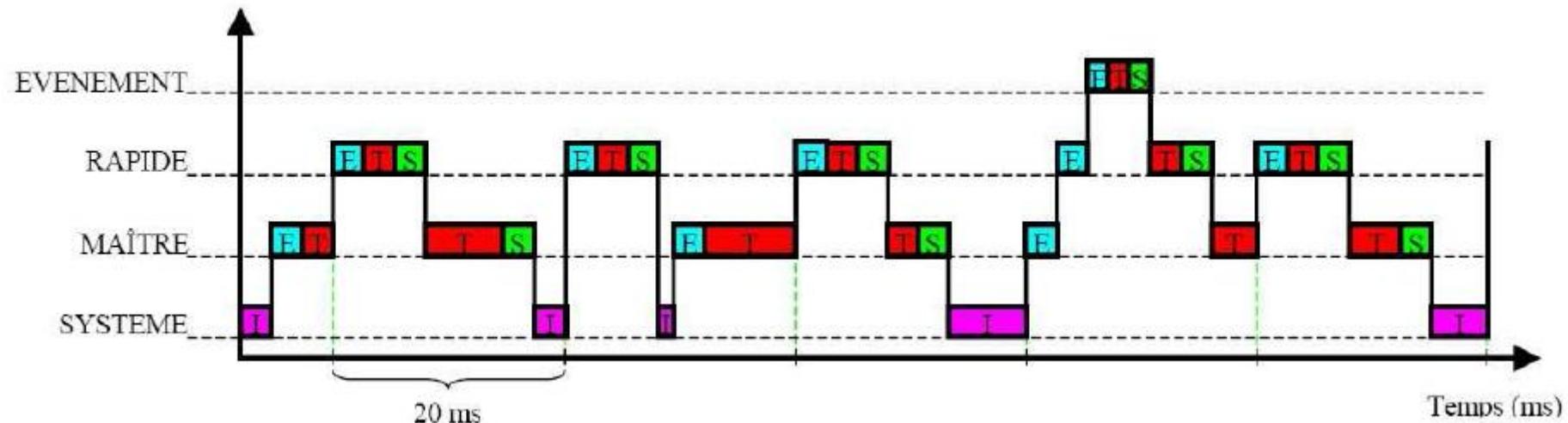
E : acquisition des entrées
T : traitement du programme
S : mise à jour des sorties





FONCTIONNEMENT D'UN API

Structure logicielle MULTI-TACHES

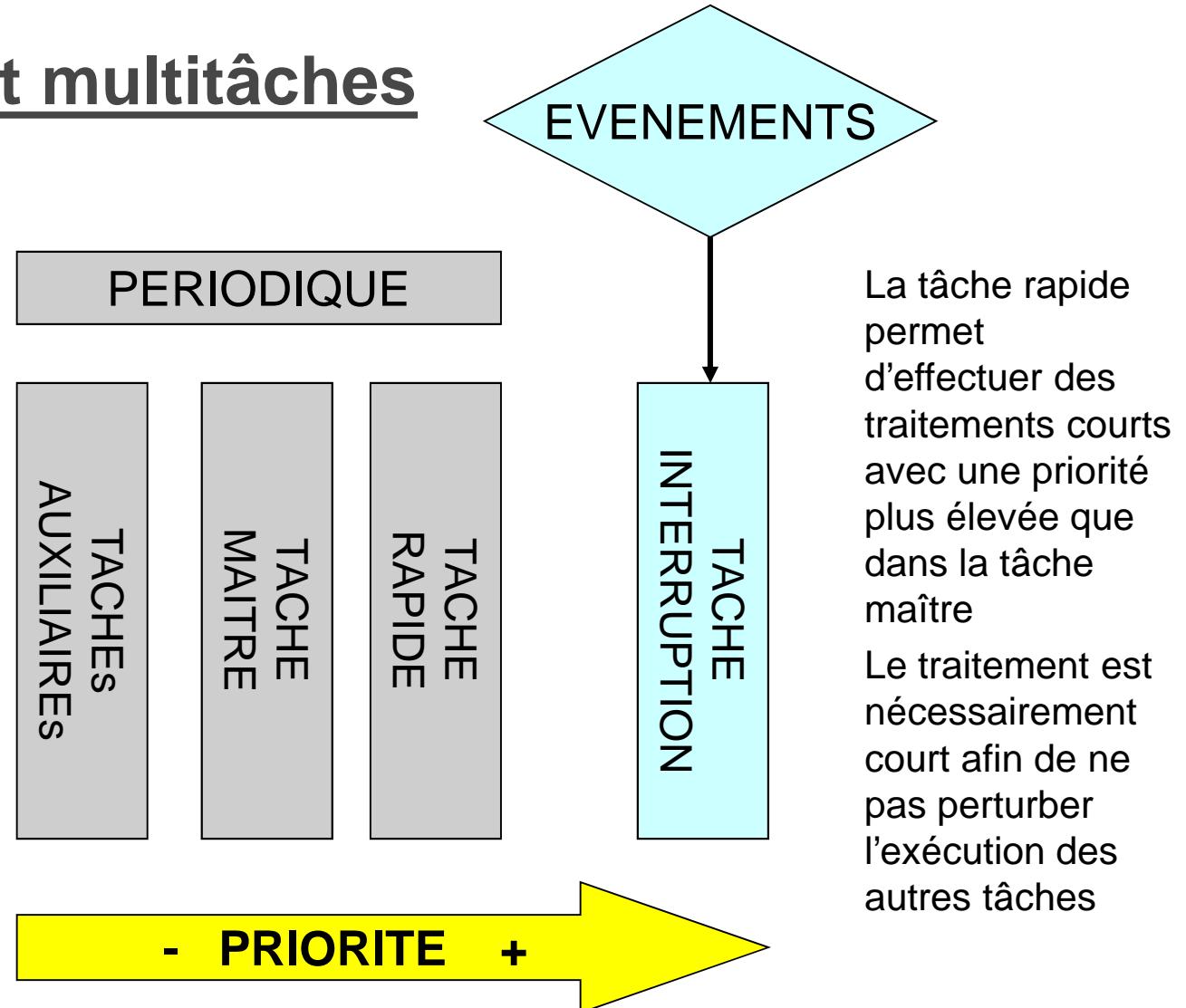


Exemple de traitement multitâche: La périodicité de la tâche rapide est ici fixée à 20ms. Il faudra veiller aux temps de cycle de la tâche maître.

La tâche rapide est alors périodique pour laisser le temps à la tâche «maître» de s'exécuter (la moins prioritaire). La tâche événementielle est prioritaire sur les autres tâches.



Traitement multitâches



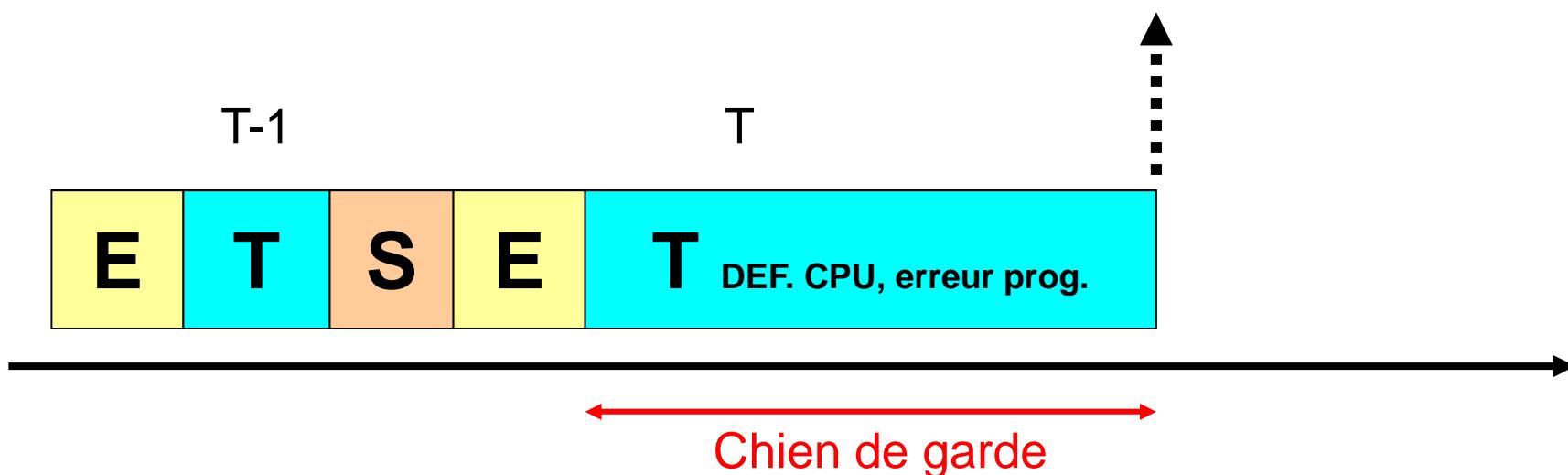


La durée d'exécution de la tâche maître, en fonctionnement cyclique ou périodique, est contrôlée par l'automate (chien de garde) et ne doit pas dépasser la valeur définie en configuration

Dans le cas de débordement, l'application est déclarée en défaut, ce qui provoque l'arrêt immédiat de l'automate

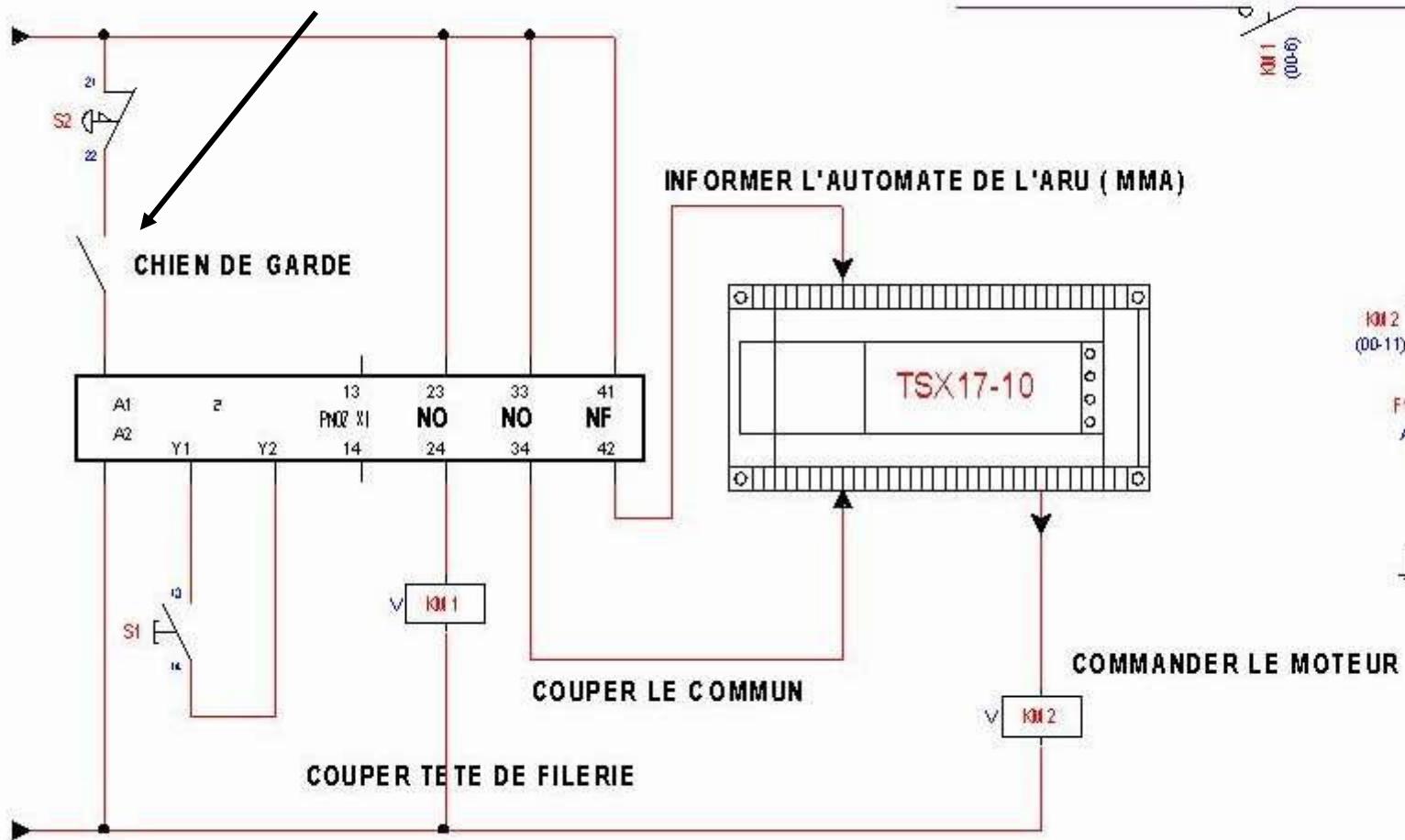
Chien de garde

! STOP CPU





Un contact est associé au watchdog
(chien de garde)



-+ { 00-13)
-+ { 00-13)
-+ { 00-13)

-+ { 00-14)
-+ { 00-14)
-+ { 00-14)



PROGRAMMATION DES API

Langages de programmation :

Il existe 4 langages de programmation des automates qui sont normalisés au plan mondial par la norme **CEI 61131-3**.

Chaque automate se programmant via une console de programmation propriétaire ou par un ordinateur équipé du logiciel constructeur spécifique.

- **GRAFCET ou SFC** : ce langage de programmation de haut niveau permet la programmation aisée de tous les procédés séquentiels.
- **Schéma par blocs ou FBD (Function Bloc Diagram)** : ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.
- **Schéma à relais ou LD (Ladder Diagram)** : ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (true/false).
- **Texte structuré ou ST (Structured Text)** : ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.



PROGRAMMATION DES API

Langages de programmation (suite):

Il existe un 5^{ème} langage

– Liste d'instructions ou IL : ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.

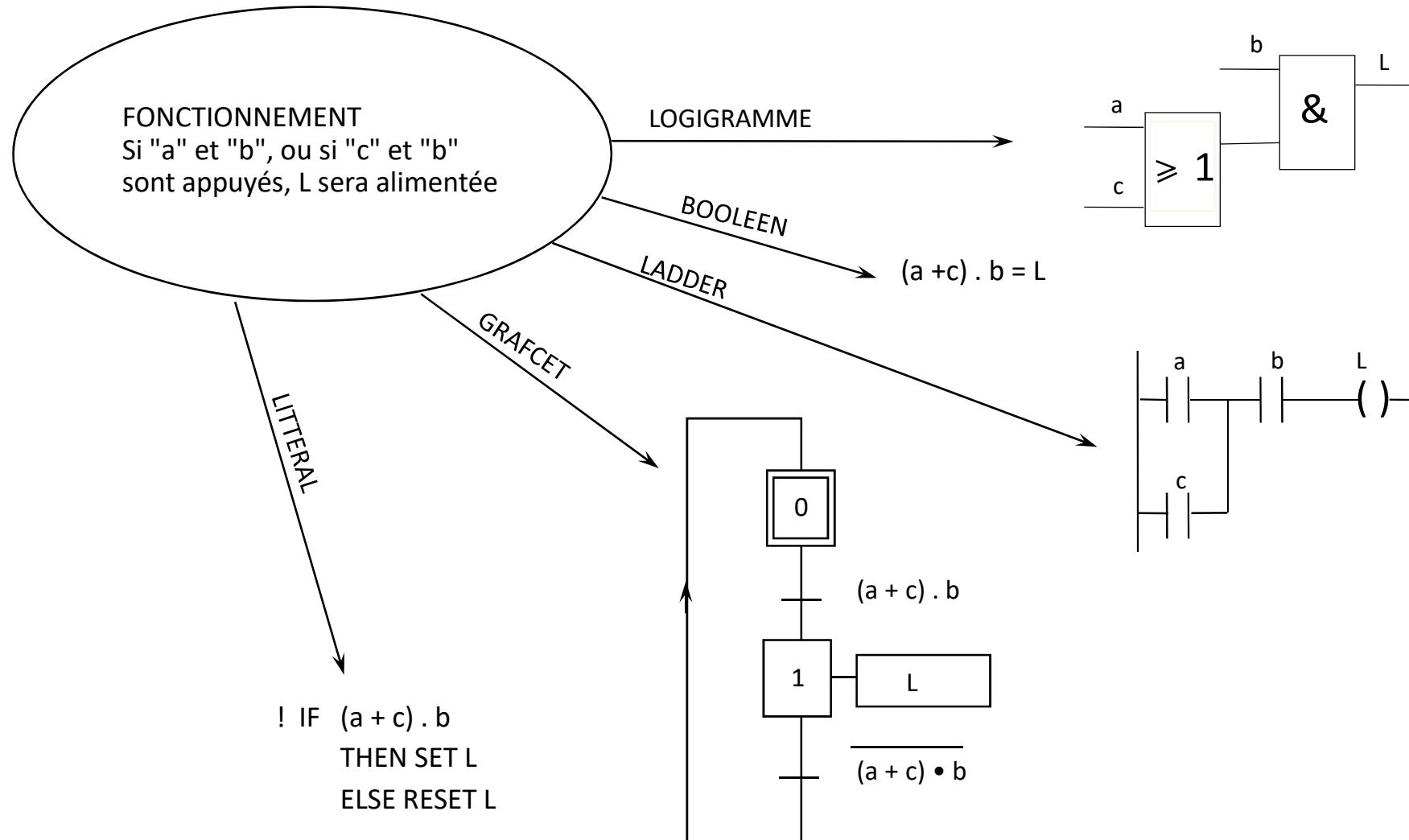
Pour programmer l'automate, l'automaticien peut utiliser :

- une console de programmation ayant pour avantage la portabilité.
- un PC avec lequel la programmation est plus conviviale, communiquant avec l'automate par le biais d'une liaison série RS232 ou RS485 ou d'un réseau de terrain.



Le langage de programmation

Notions de langage





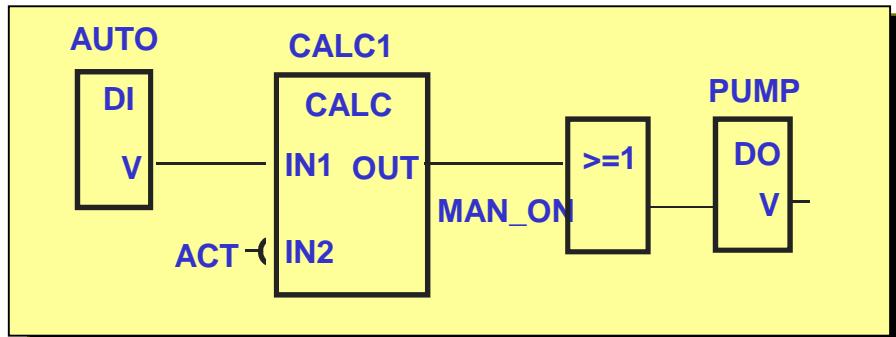
Le langage de programmation

langage liste d'instructions	langages graphiques	langage littéral
LD %I1.1 ORN %I1.2 AND %I1.3 ST %Q2.1 LD %Q2.2 STN %I1.4	<p>schéma à contacts</p> <p>Le schéma à contacts montre une logique AND entre les entrées %I1.1, %I1.2 et %I1.3. Les sorties sont %Q2.1 et %Q2.2.</p> <pre>graph LR; I1.1 --- --- C1[]; I1.2 --- --- C1; I1.3 --- --- C1; C1 --- --- Q2.1["%Q2.1"]; C1 --- --- Q2.2["%Q2.2"];</pre>	<pre>! IF (%I1.1 OR NOT %I1.3) AND %I1.3 THEN SET %Q2.1 ; ELSE RESET %Q2.1; END-IF ;</pre>
LD %M12 (#) 04 LD %I1.7 (#) 05 LD %I1.10	<p>GRAFCET</p> <p>Le schéma GRAFCET montre un état initial avec une transition vers l'état 03. L'état 03 a une transition vers l'état 04. L'état 04 a une transition vers l'état 05. L'état 05 a une transition vers l'état initial.</p> <pre>graph TD; Start(()) --> 03[03]; 03 --> 04[04]; 04 --> 05[05]; 05 --> Start;</pre>	<pre>! IF %I1.4 THEN RESET %Q2.2 ; ELSE SET %Q2.2 ; END-IF ;</pre>

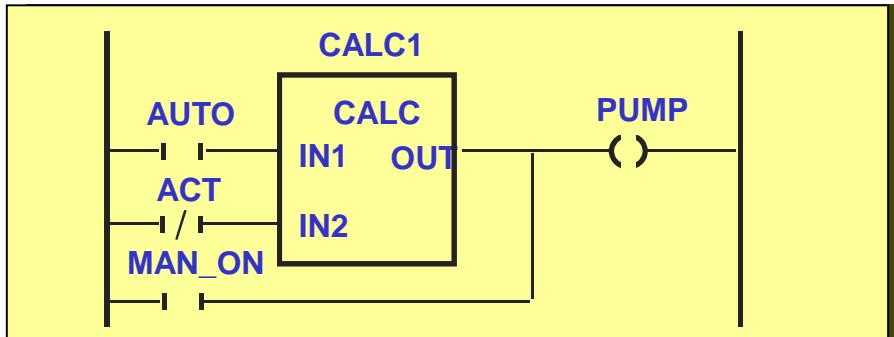


Les langages IEC1131

Function Block Diagram (FBD)



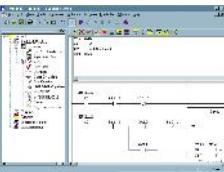
Ladder Diagram (LD)



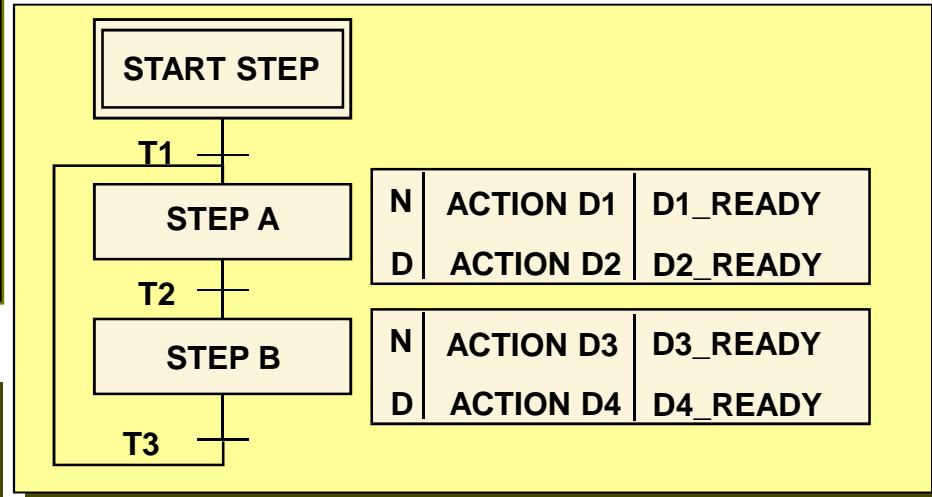
Instruction List (IL)

```

A: LD    %IX1 (* PUSH BUTTON *)
ANDN %MX5 (* NOT INHIBITED *)
ST     %QX2 (* FAN ON *)
    
```



Sequential Flow Chart (SFC)



Structured Text (ST)

```

VAR CONSTANT X : REAL := 53.8 ;
Z : REAL; END_VAR
VAR aFB, bFB : FB_type; END_VAR

bFB(A:=1, B:='OK');
Z := X - INT_TO_REAL(bFB.OUT1);
IF Z>57.0 THEN aFB(A:=0, B:="ERR");
ELSE aFB(A:=1, B:="Z is OK");
END_IF
    
```



Comparaison des langages

LANGAGE	AVANTAGES	INCONVENIENTS
LD	facile à lire et à comprendre par la majorité des électriciens langage de base de tout PLC	suppose une programmation bien structurée
FBD	Très visuel et facile à lire	Peut devenir très lourd lorsque les équations se compliquent
ST	Langage de haut niveau (langage pascal) Pour faire de l'algorithmique	Pas toujours disponible dans les ateliers logiciels
IL	langage de base de tout PLC type assembleur	très lourd et difficile à suivre si le programme est complexe Pas visuel.
SFC	Description du fonctionnement (séquentiel) de l'automatisme. Gestion des modes de marches Pas toujours accepté dans l'industrie...	Peu flexible



SAP

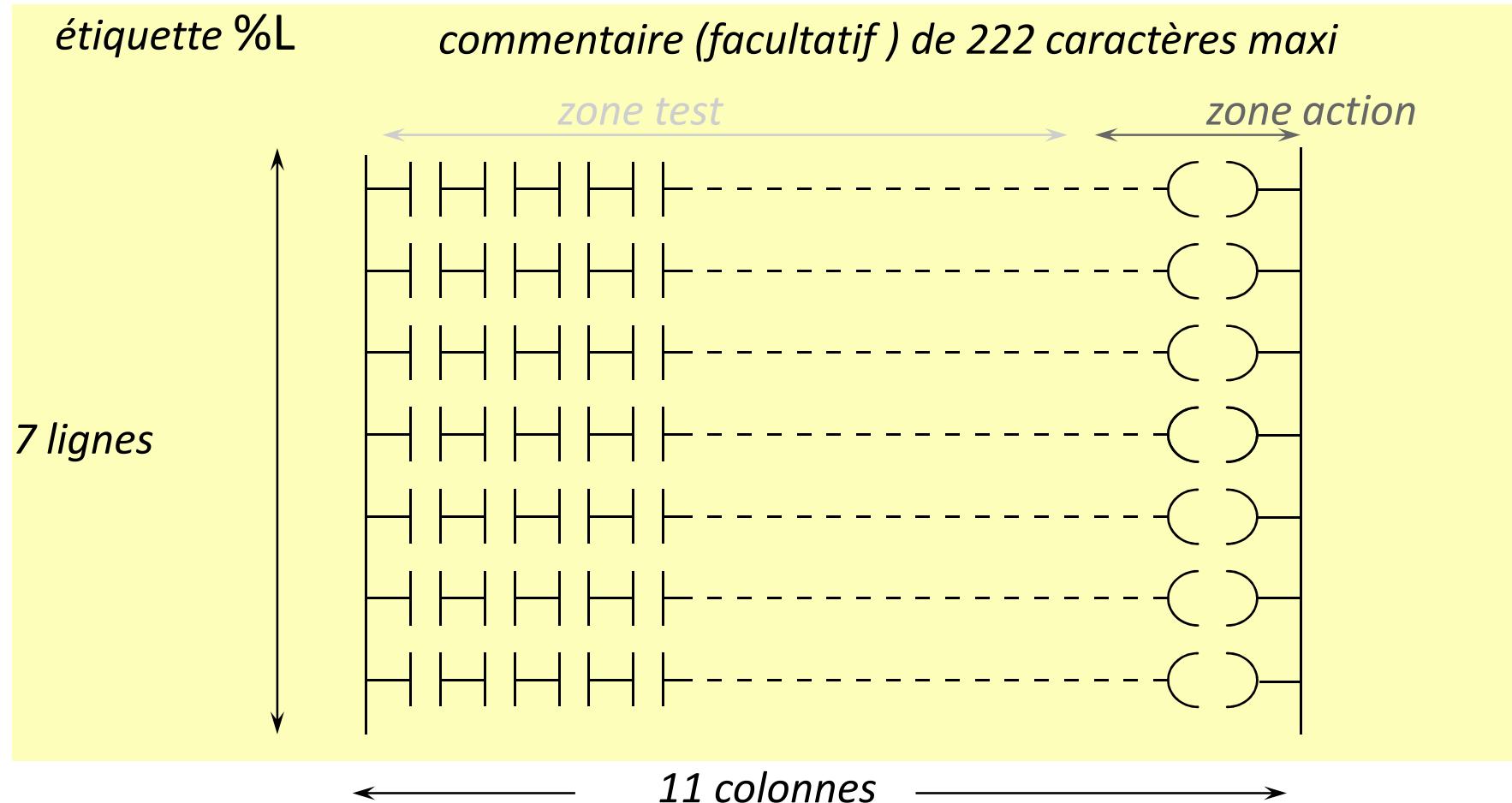
Informatique industrielle-98

Réseau de contacts et instructions sur bits



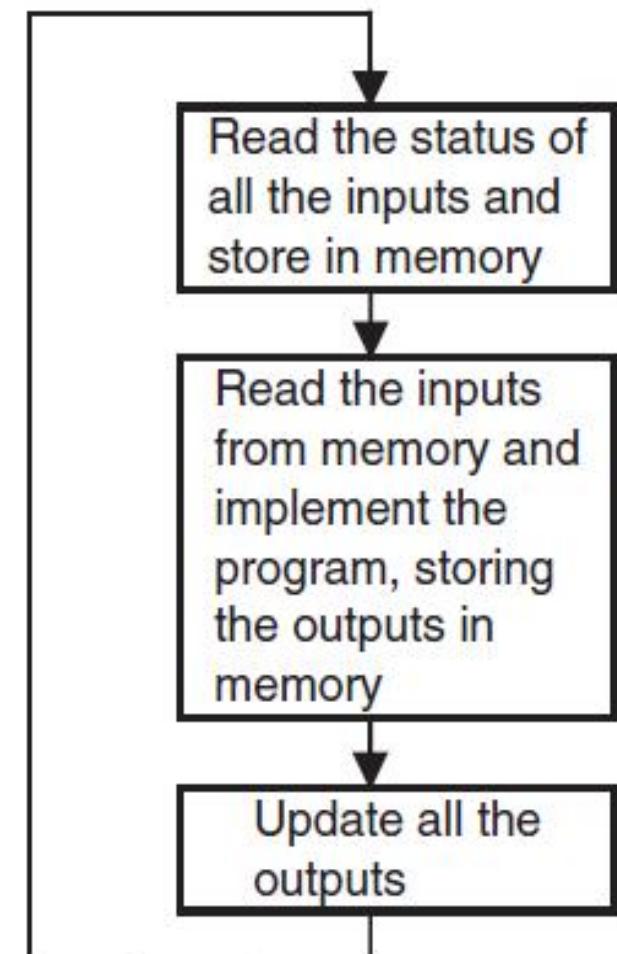
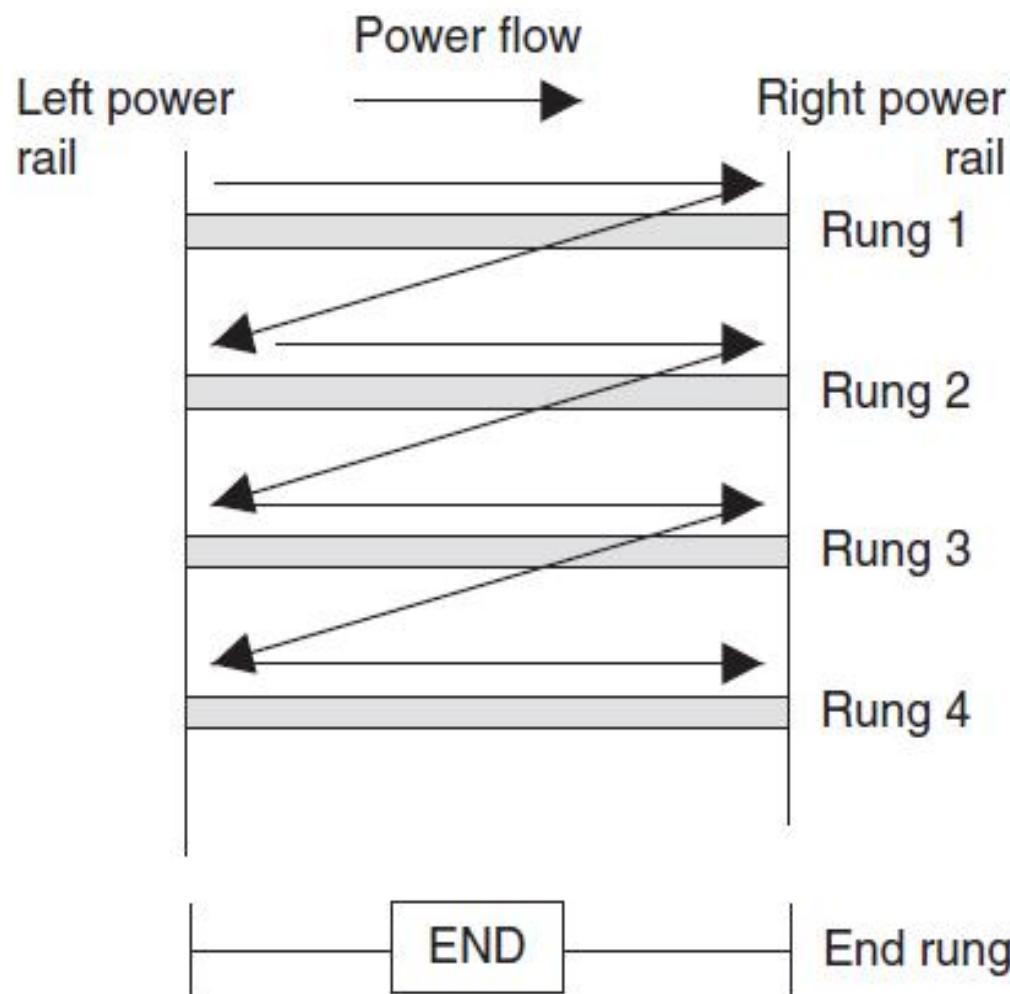
Réseau de contacts et instructions sur bits

Structure d'un réseau de contacts





L'exécution du programme





Réseau de contacts et instructions sur bits

Structure d'un réseau de contacts

- Etiquette (ou label) %Li (*i compris entre 0 et 999*)
Permet de repérer un réseau (ou rung) mais n'est pas obligatoire.
Elle est indispensable pour permettre un branchement après un saut de programme (JUMP).
L'ordre des étiquettes est quelconque : c'est l'ordre de saisie des réseaux qui est pris en compte par le système lors de la scrutation.
- Commentaires



Ils sont mémorisés dans l'automate,
ils consomment donc de la mémoire programme !

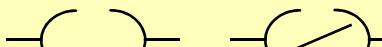


Réseau de contacts et instructions sur bits Eléments graphiques

tests directs / inverses



détection front montant / descendant



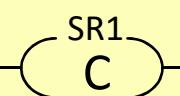
bobine directe / inverse



bobine d'enclenchement



bobine de déclenchement



%L10
>>>

bobine CALL

< RETURN >

JUMP à un autre réseau

instruction de retour de sous programme

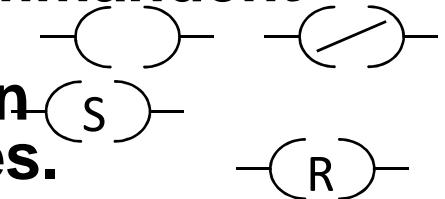


Réseau de contacts et instructions sur bits Eléments graphiques

- **Tests**
- Ils testent l'état des bits d'entrées/sorties de l'automate et des variables internes au programme.



- **Bobines**
- Associées à des objets bits, elles commandent les sorties de l'automate raccordées aux organes de commande ou de visualisation (relais, voyants...) et les variables internes.

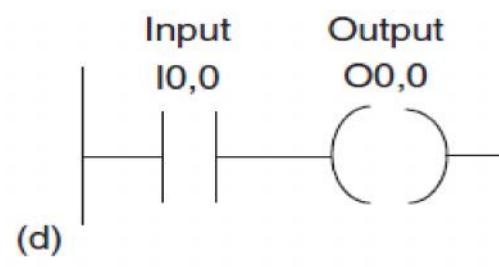
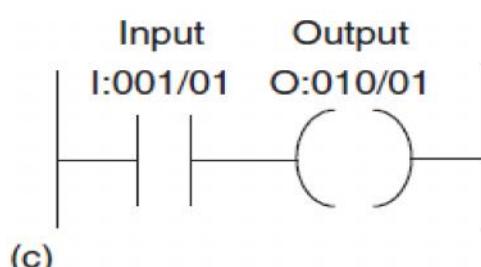
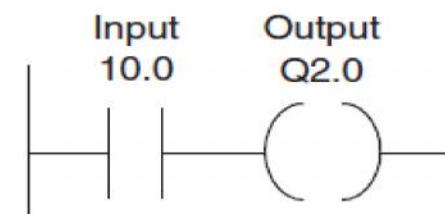
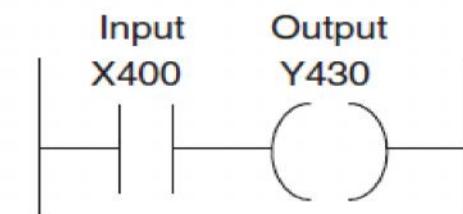
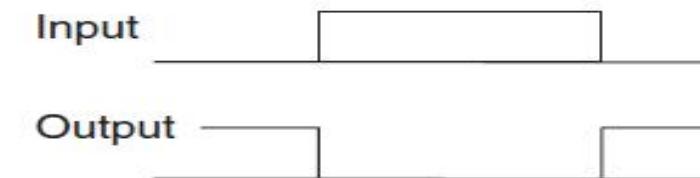
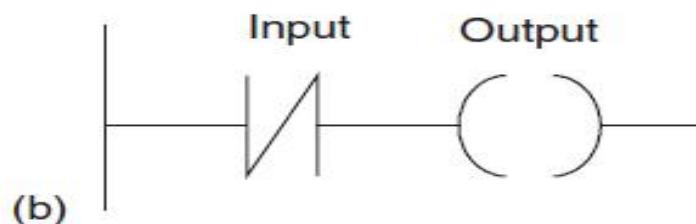
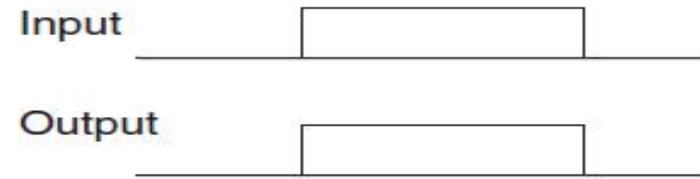
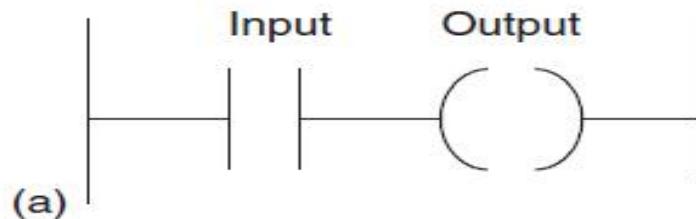


- Cas particulier : - la bobine d'appel à un sous-programme





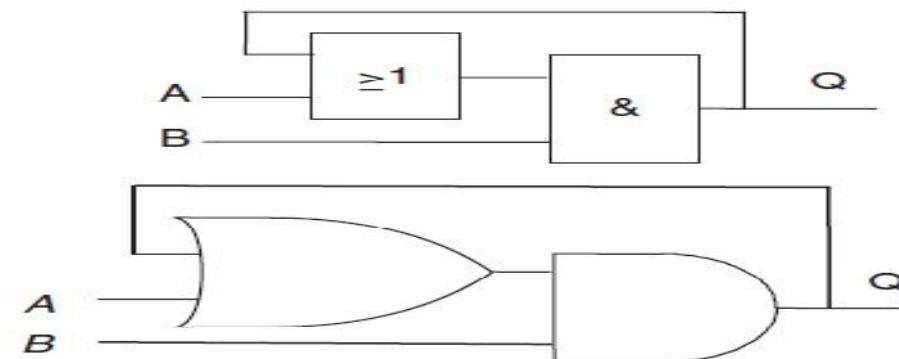
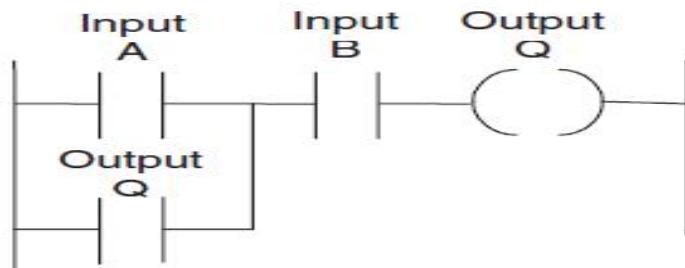
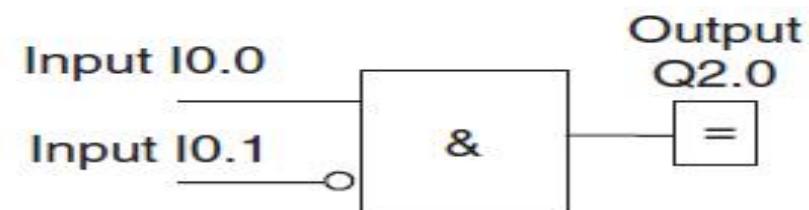
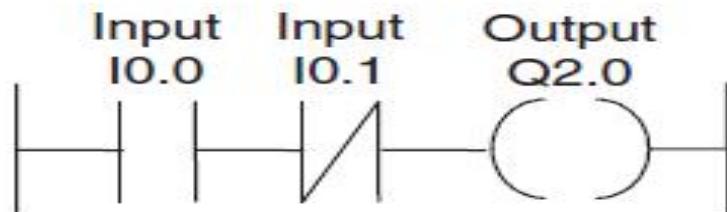
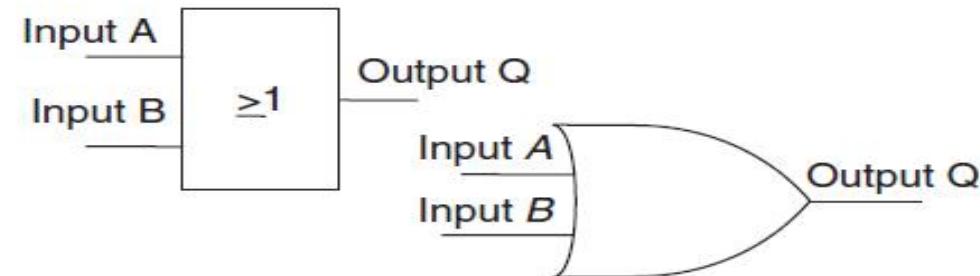
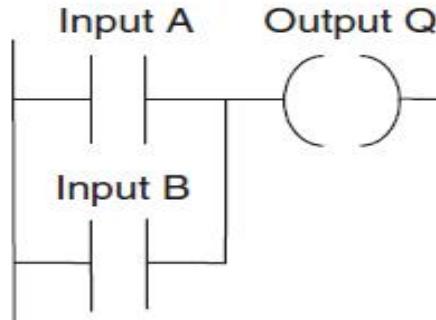
Barreau d'un diagramme Ladder



Notation: (a) Mitsubishi,
(b) Siemens, (c) Allen-Bradley,
(d) Telemecanique.

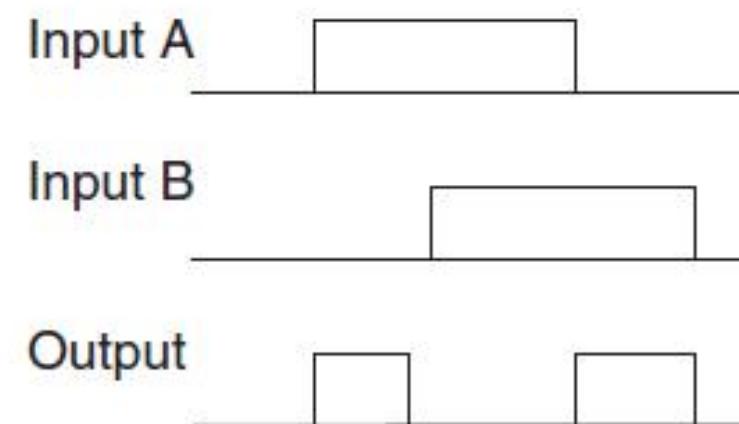
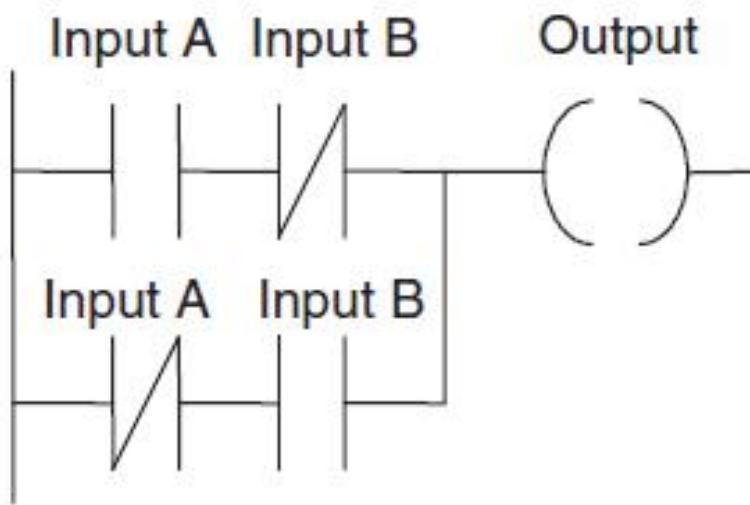
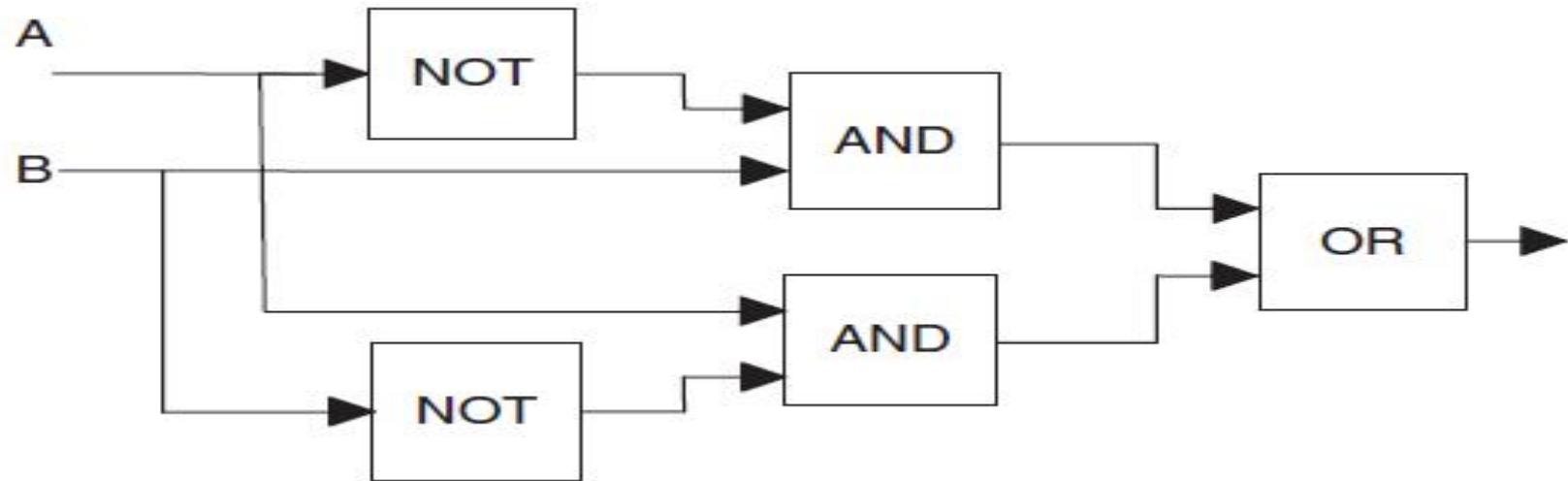


Réalisation de quelques fonctions logique



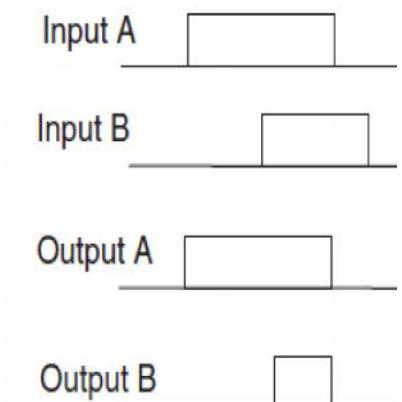
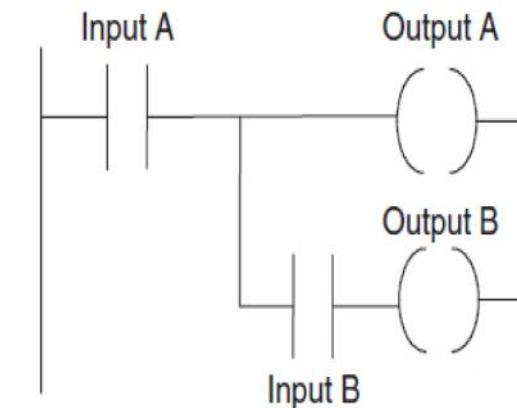
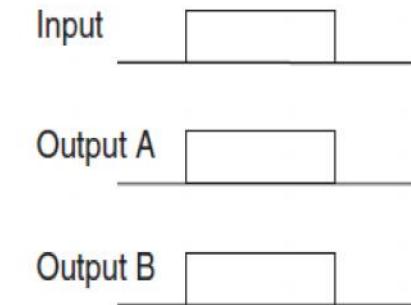
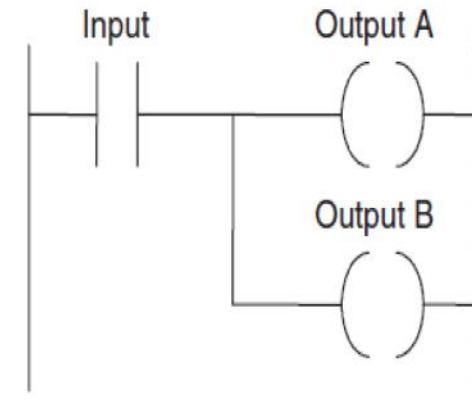
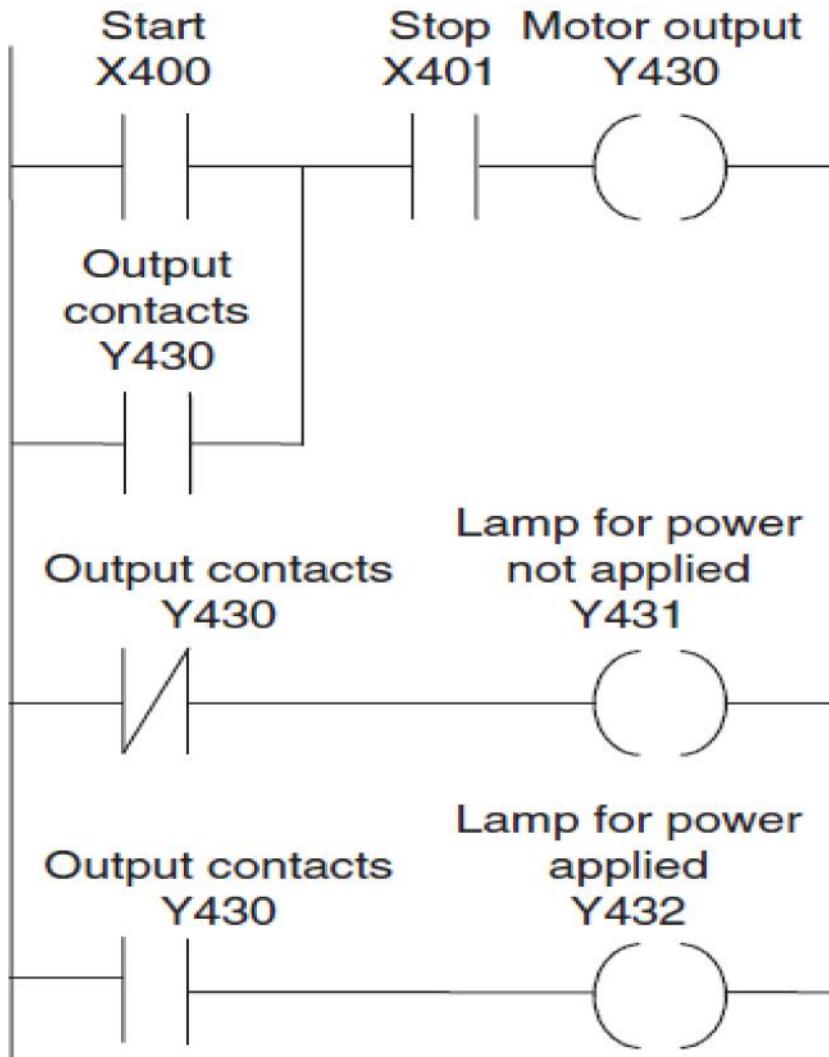


- XOR logique



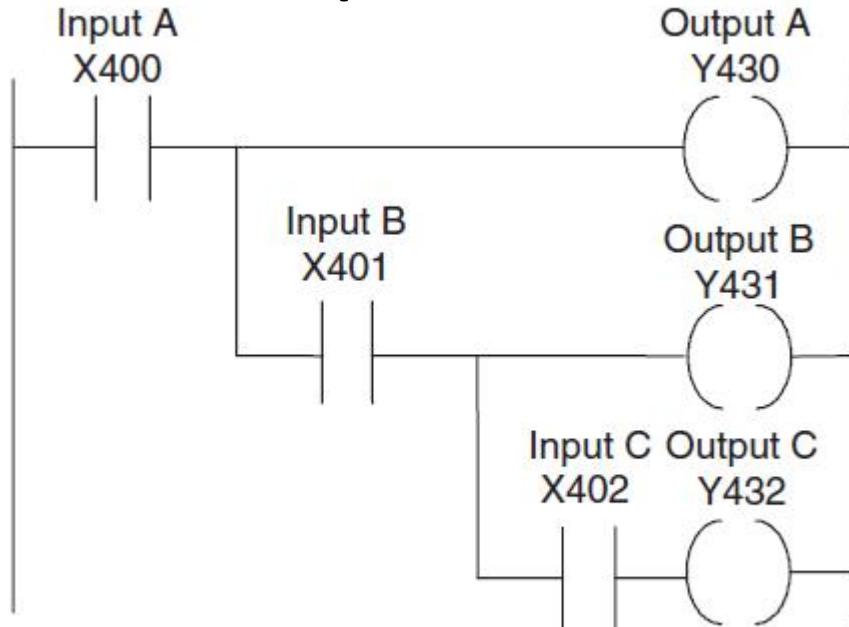


• Sorties multiple

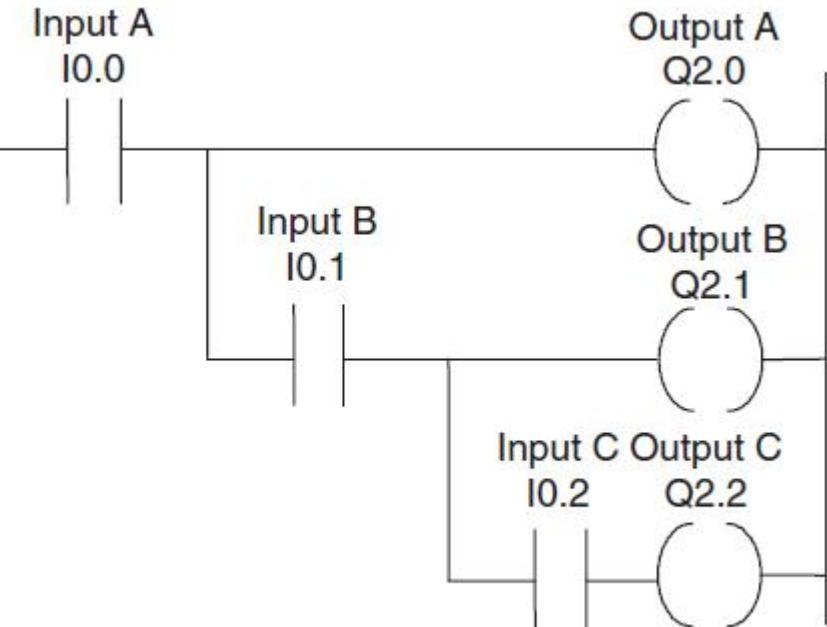
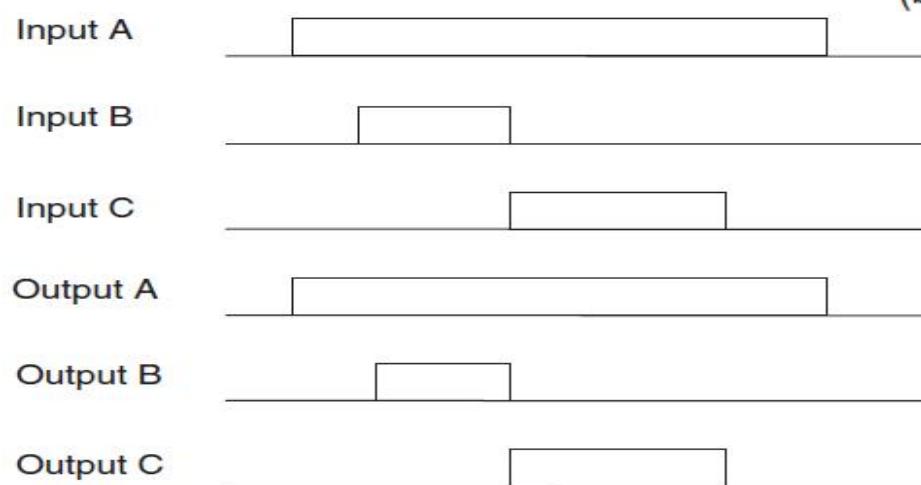




- Sorties séquencés



(a)



(b)



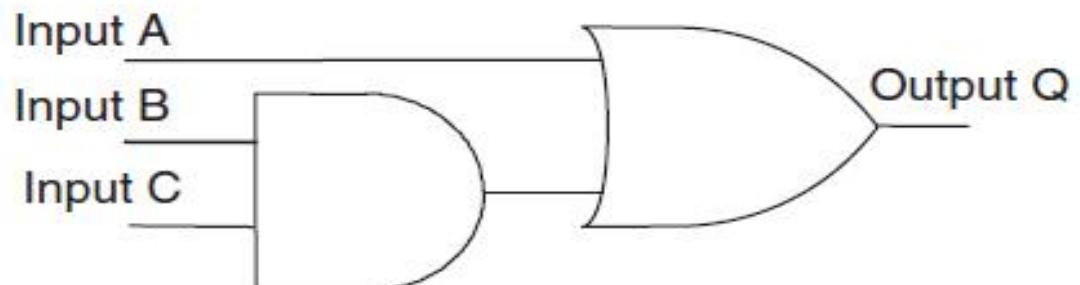
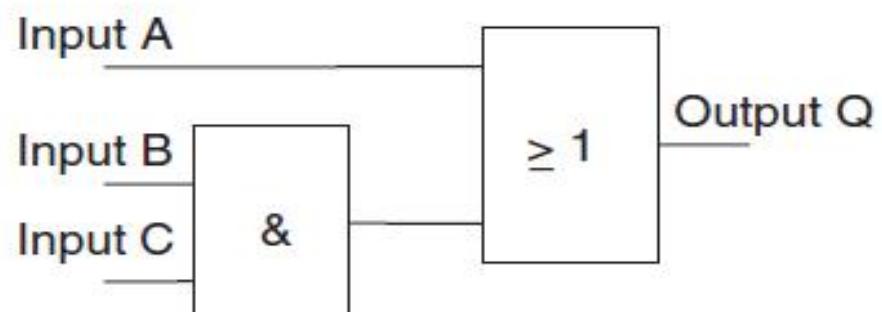
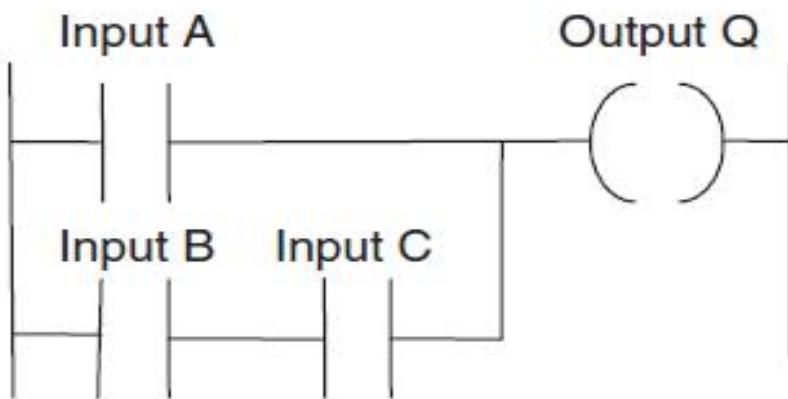
$$A + B \cdot C = Q$$

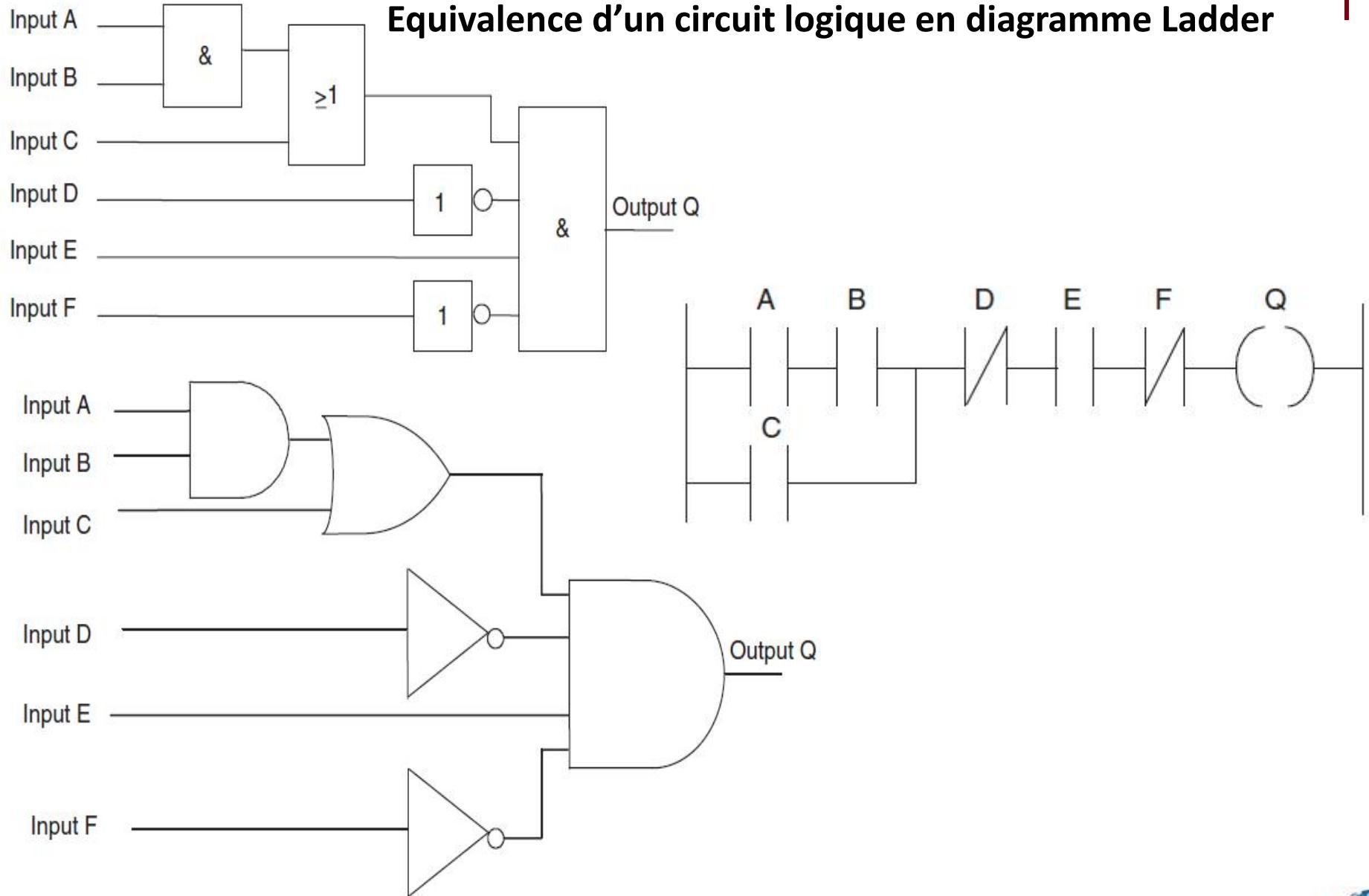
1. La réalisation avec la notation Mitsubishi

$$X400 + X401 \cdot X402 = Y430$$

2. La réalisation avec la notation Siemens

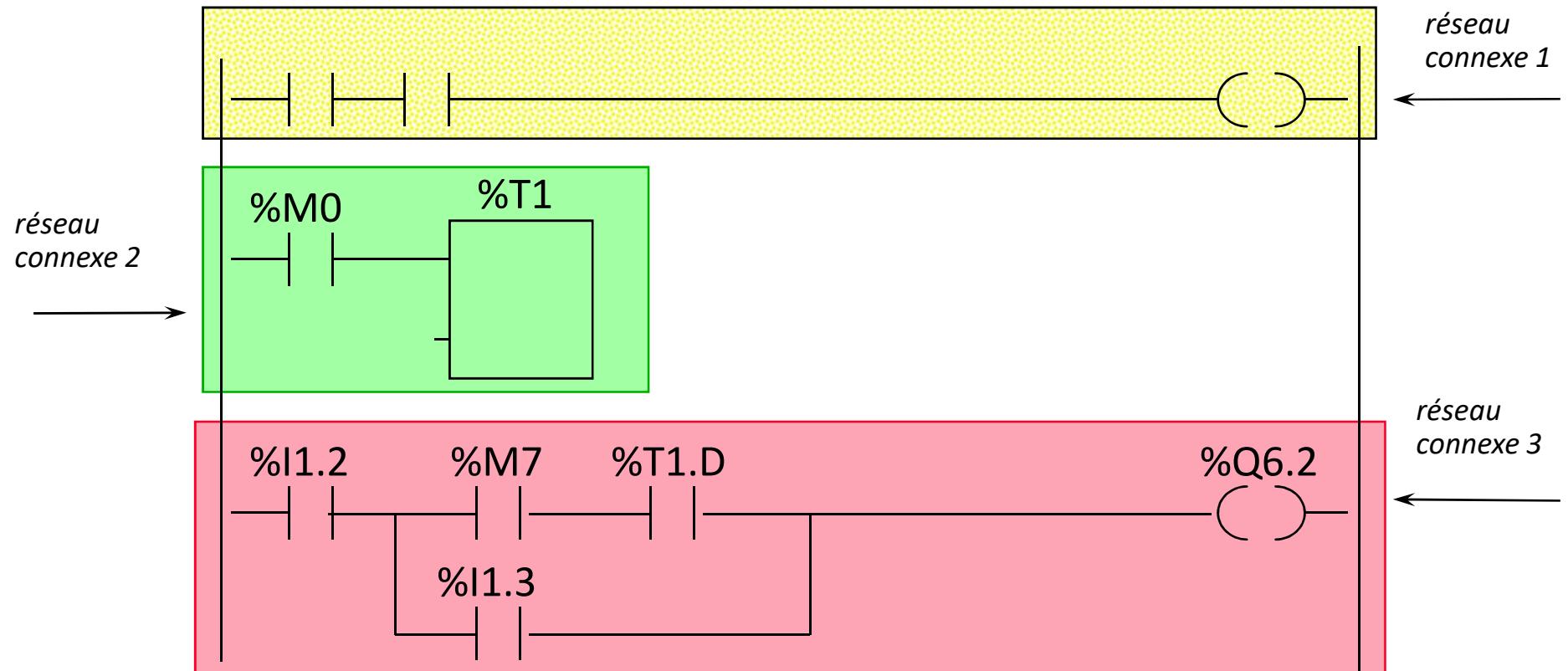
$$I0.0 + I0.1 \cdot I0.2 = Q2.0$$



**Equivalence d'un circuit logique en diagramme Ladder**



Réseau de contacts et instructions sur bits Exécution d'un réseau

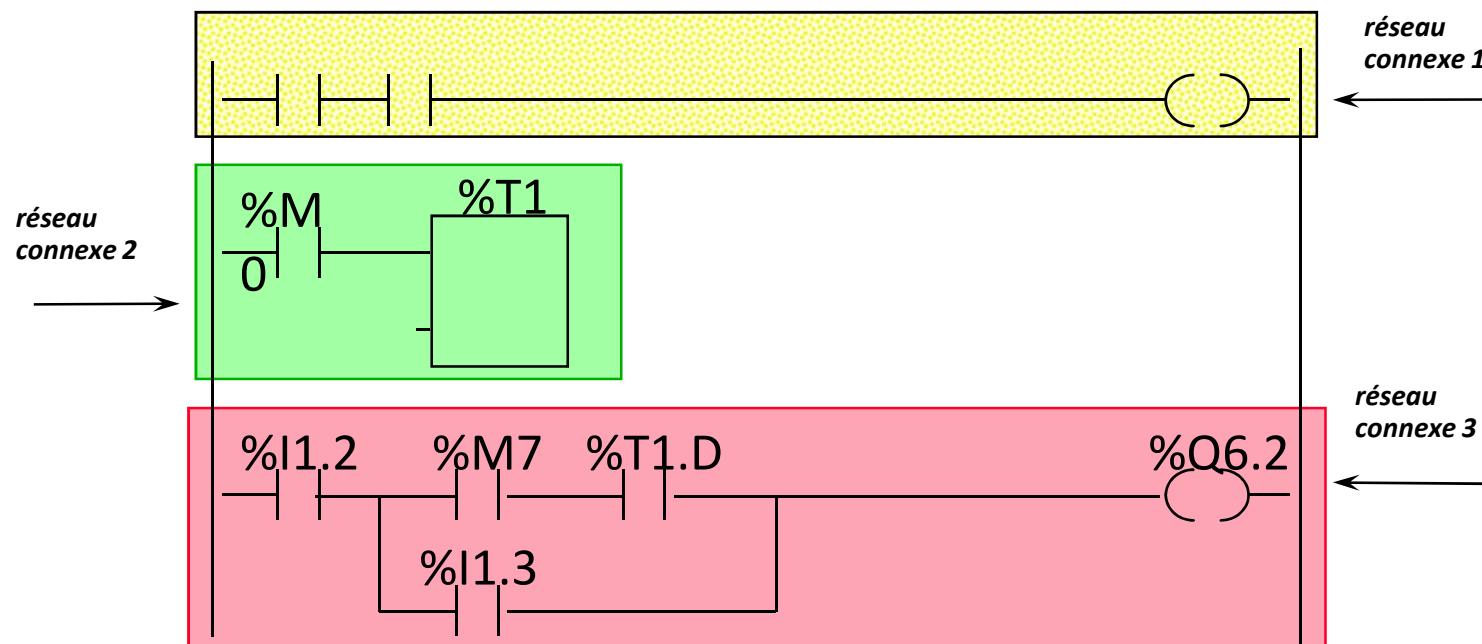


Réseau connexe = éléments graphiques reliés entre eux mais indépendants des autres éléments du réseau (pas de liaisons verticales en limite de réseau connexe)



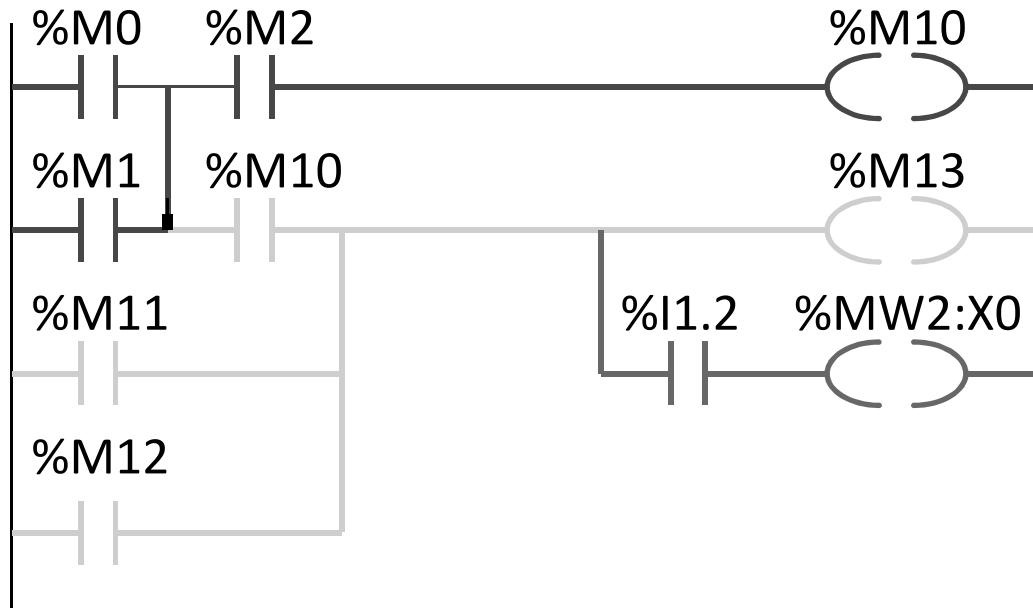
Réseau de contacts et instructions sur bits Exécution d'un réseau

- La scrutation d'un réseau s'effectue réseau connexe par réseau connexe puis, pour un réseau connexe, de haut en bas et pour chaque ligne, de gauche à droite





Réseau de contacts et instructions sur bits Exemple



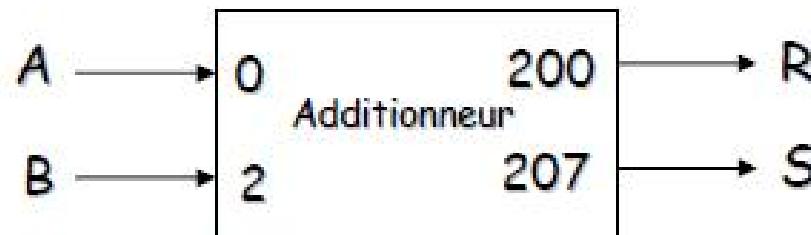
Ordre d'évaluation:

- 1 : %M0, %M1, %M2, %M10
- 2 : %M10, %M11, %M12, %M13
- 3 : %I1.2, %MW2:X0

évaluation première bobine (%M10)
évaluation deuxième bobine (%M13)
évaluation troisième bobine (%MW2:X0)



Exemple 1: L'additionneur binaire

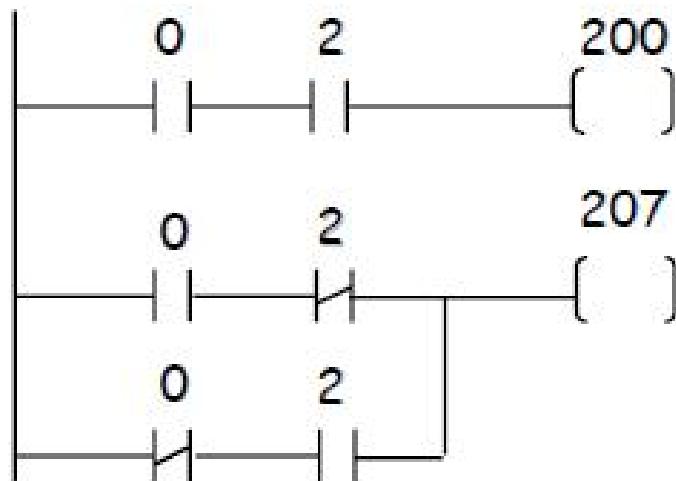


Equations:

$$R = A \cdot B$$

$$S = A \cdot \overline{B} + \overline{A} \cdot B$$

Programme:

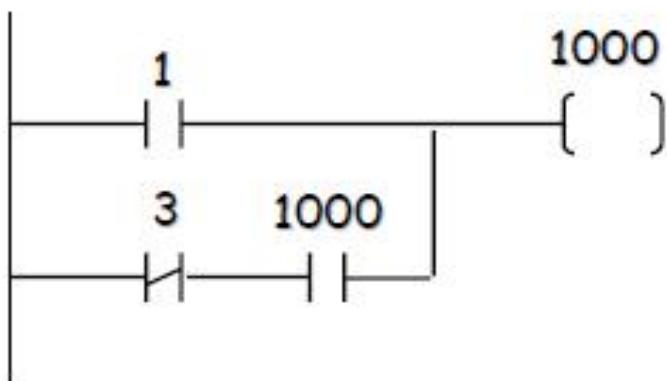
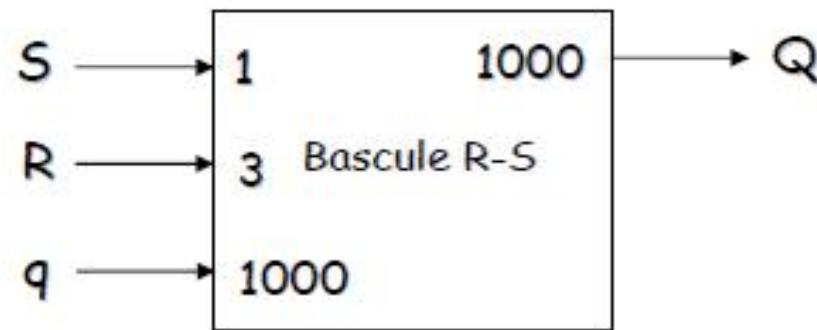




Exemple 2: Programmation d'une cellule R-S

utilisation de variable interne

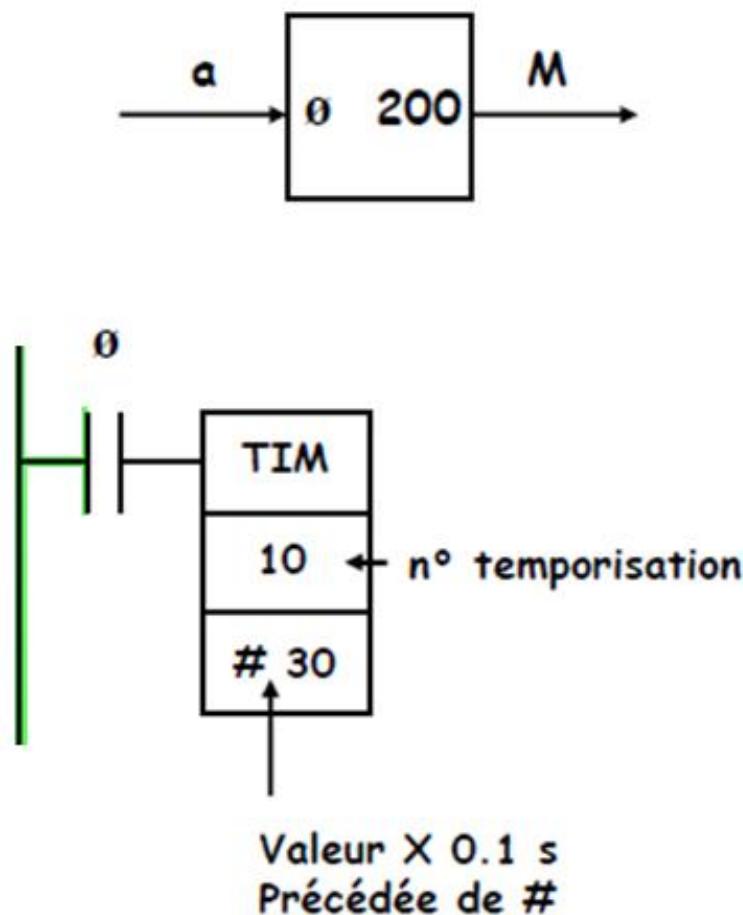
$$Q = S + \overline{R} \cdot q$$



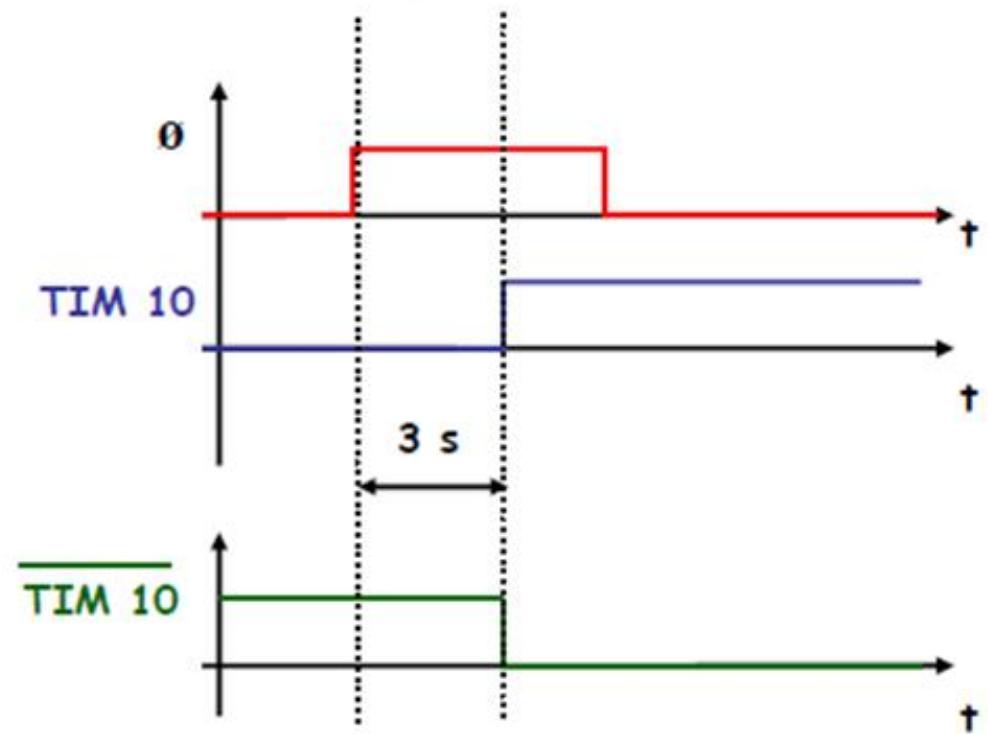


Temporisation en langage Ladder

- Schéma général

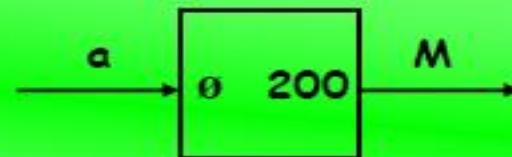
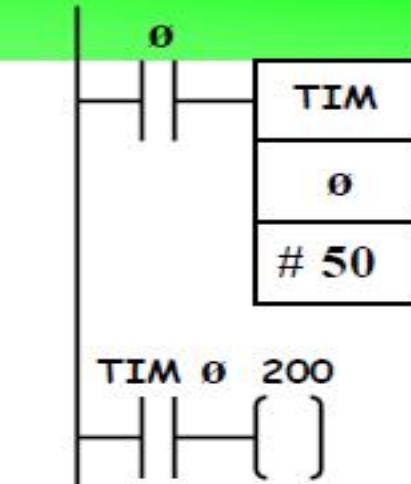


Chronogramme

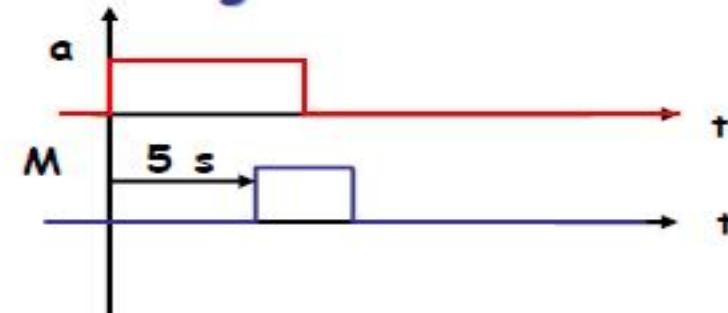




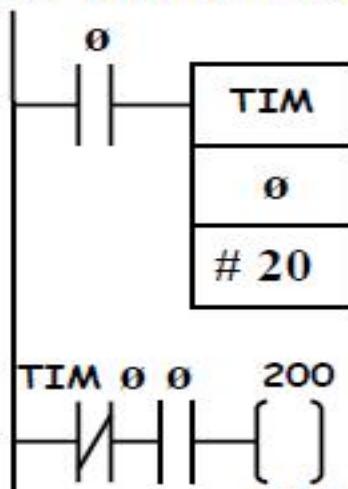
- Action retardée



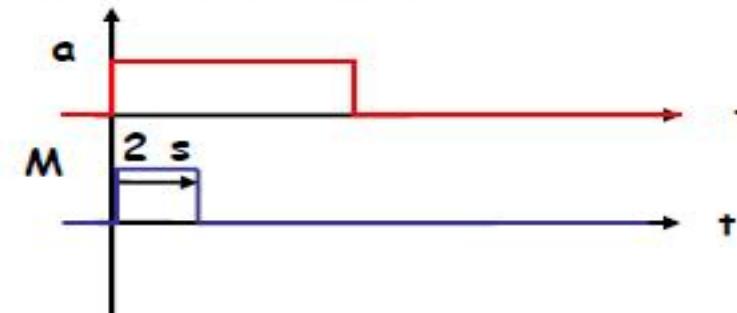
Chronogramme



- Action limitée dans le temps

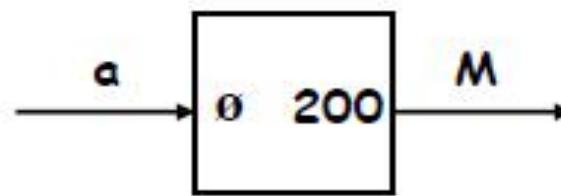
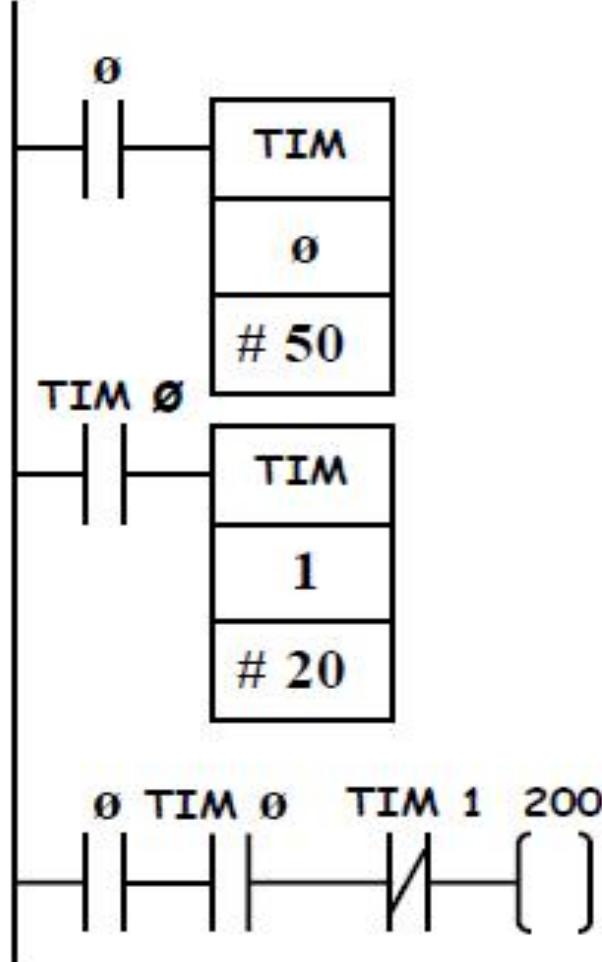


Chronogramme

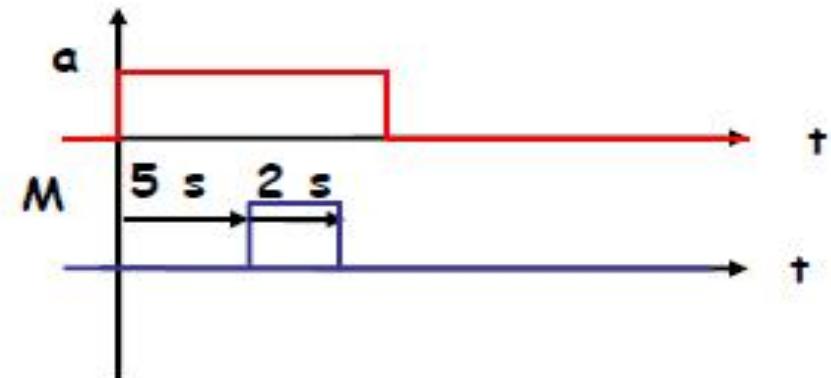




- Action retardée et limitée dans le temps



Chronogramme





Le langage Ladder

Les blocs fonctions



Les blocs fonctions Blocs fonctions

- Les blocs fonctions sont préprogrammés dans l'automate
 - Chaque bloc possède:
 - des entrées pour le commander
 - une représentation graphique
 - un numéro
 - des sorties qui indiquent son état
 - des paramètres internes qui permettent de l'adapter à l'application
- The diagram shows a rectangular function block labeled "% TM1". On the left side, there is an input terminal labeled "IN" and an output terminal labeled "Q". On the right side, there is a parameter setting labeled "MODE:TP" with "TB=1s" below it. At the bottom, there are other parameters: "TM.P:200" and "MODIF:Y". Arrows point from the text descriptions on the left to the corresponding parts of the block: one arrow points to the "IN" terminal, another to the "Q" terminal, and a third to the parameter area.

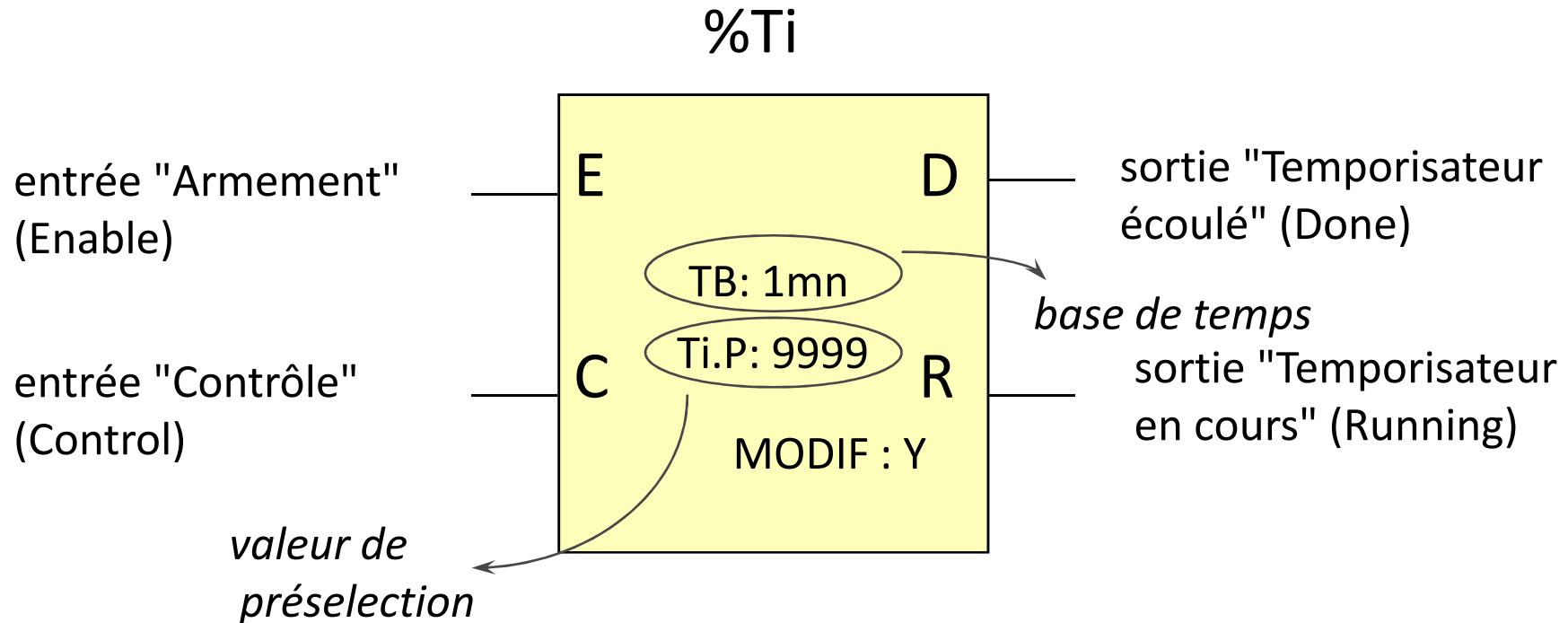


Les blocs fonctions Blocs fonctions

	Nombre maxi
• % TMi : Temporisateur / Monostable % Ti <i>(= bloc à la norme)</i>	64 (<i>si aucun configuré</i>)
• % MNi : Monostable	8
• % Ci : Compteur / Décompteur	32
• % Ti : Timer <i>(= bloc série 7)</i>	64 (<i>si aucun %TMi configuré</i>)
• % Ri : Registre	4
• % DRI : Programmateur Cyclique (Drum)	8



Les blocs fonctions Le temporisateur série 7

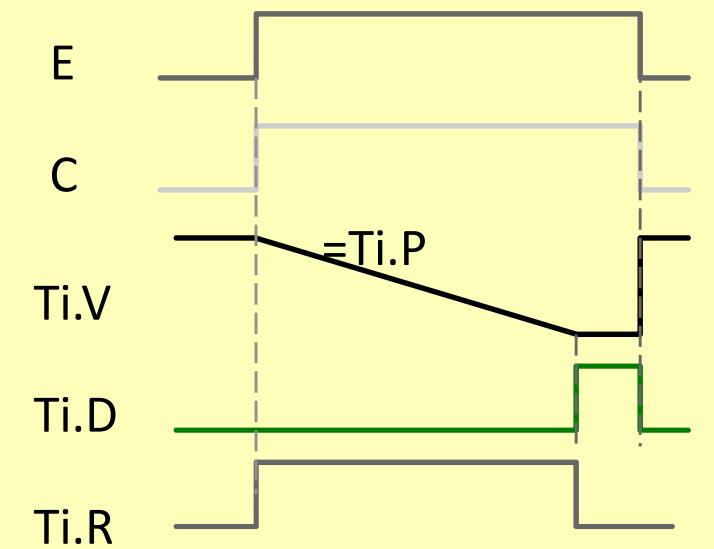
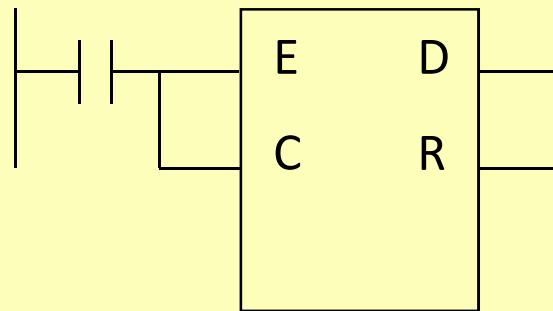


Le temporisateur évolue lorsque ses deux entrées E et C sont à 1



Les blocs fonctions Exemples

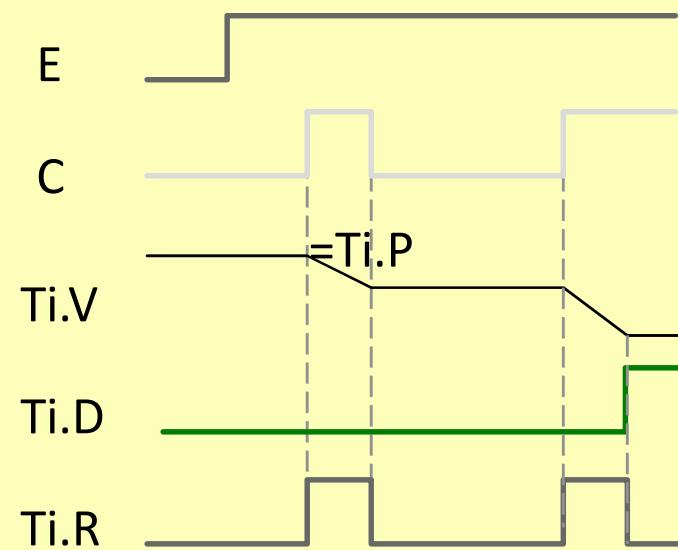
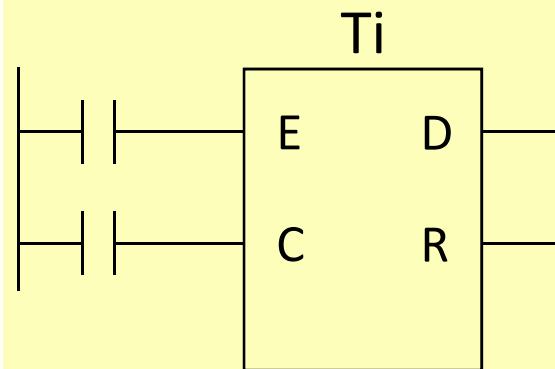
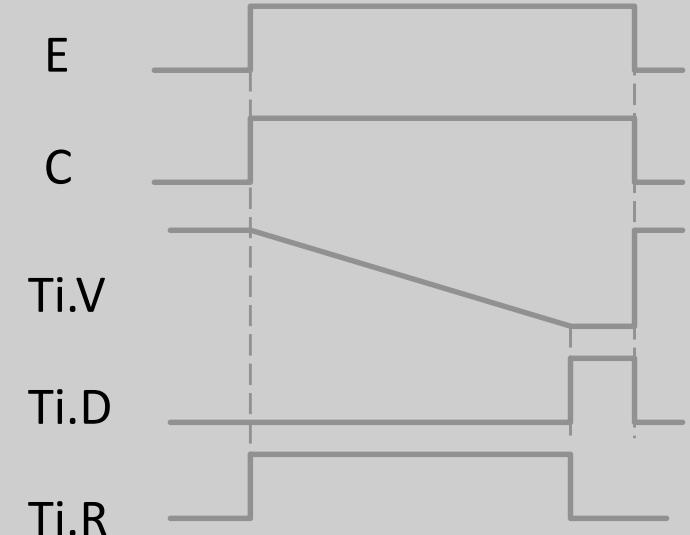
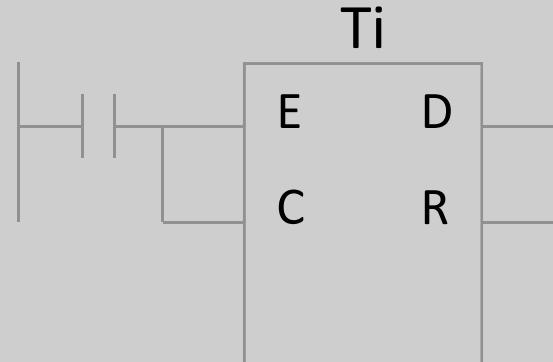
Ecoulement
continu





Les blocs fonctions: Exemples

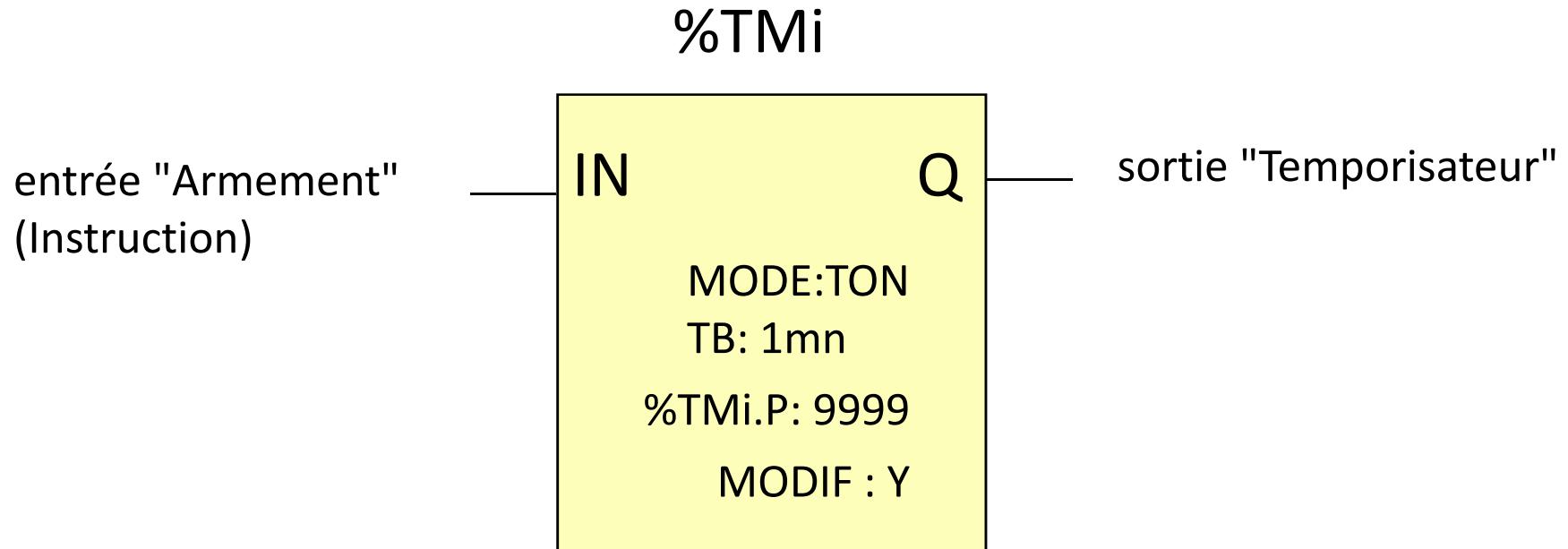
Ecoulement continu



Ecoulement discontinu



Les blocs fonctions Le temporisateur %TMI



3 modes de fonctionnement :

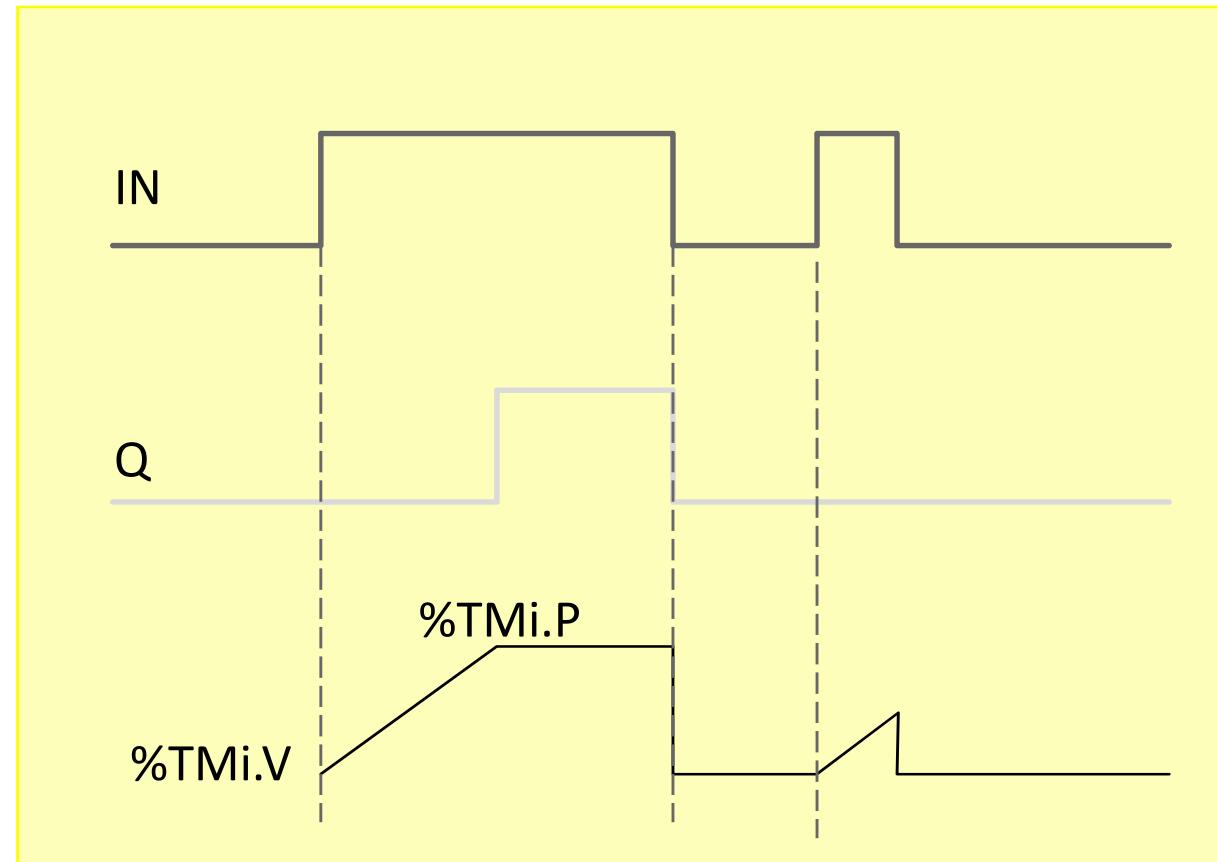
TON = retard à l'enclenchement
TOF = retard au déclenchement
TP = impulsion de durée précise



Les blocs fonctions
Le temporisateur %TMi

MODE TON :

retard à
l'enclenchement

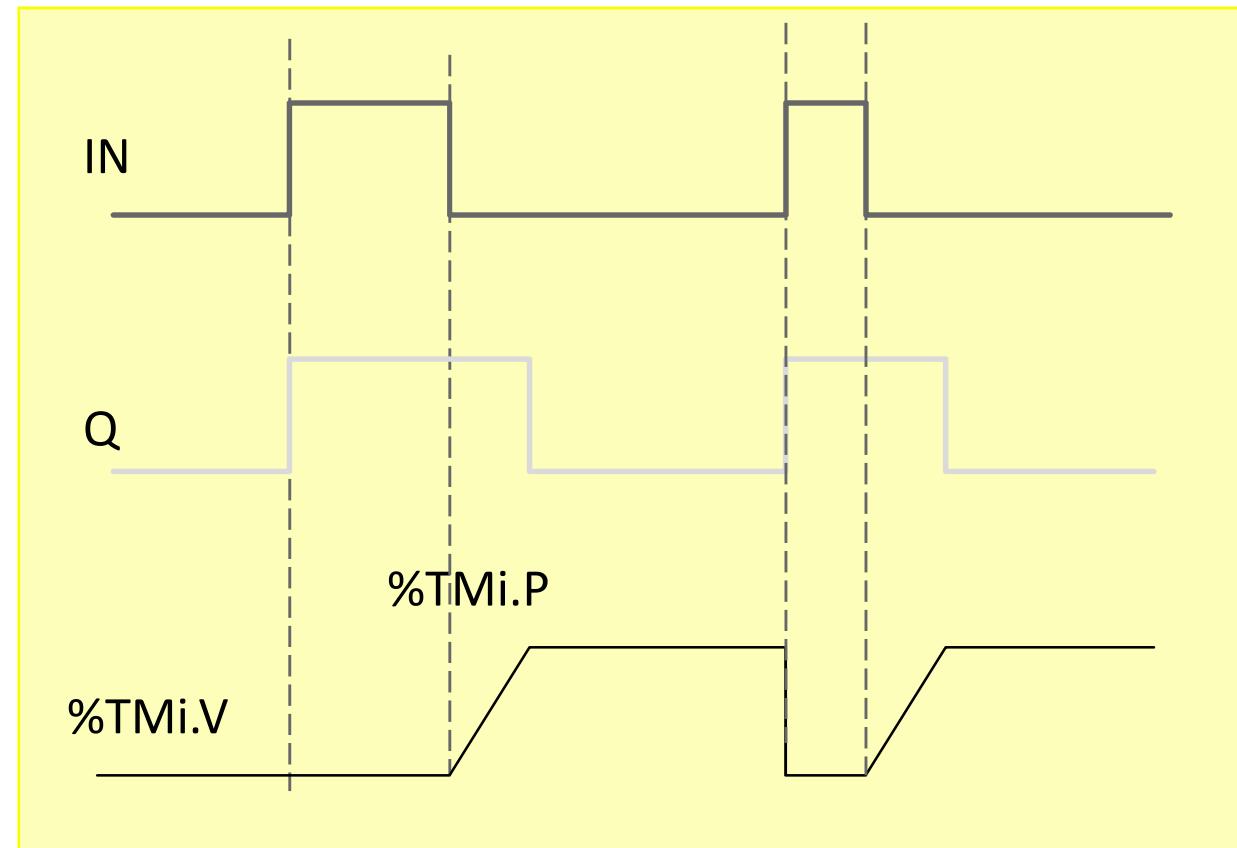




Les blocs fonctions Le temporisateur %TMi

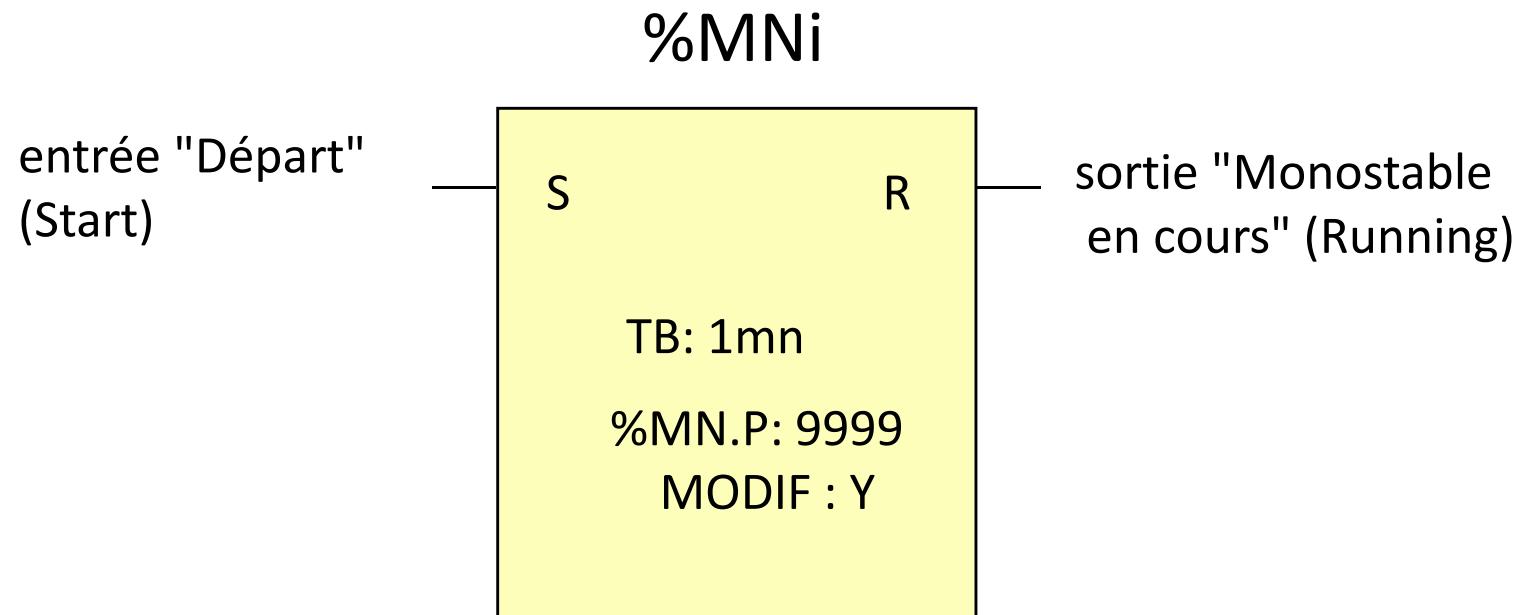
MODE TOF :

retard au
déclenchement





Les blocs fonctions Le monostable %MNI

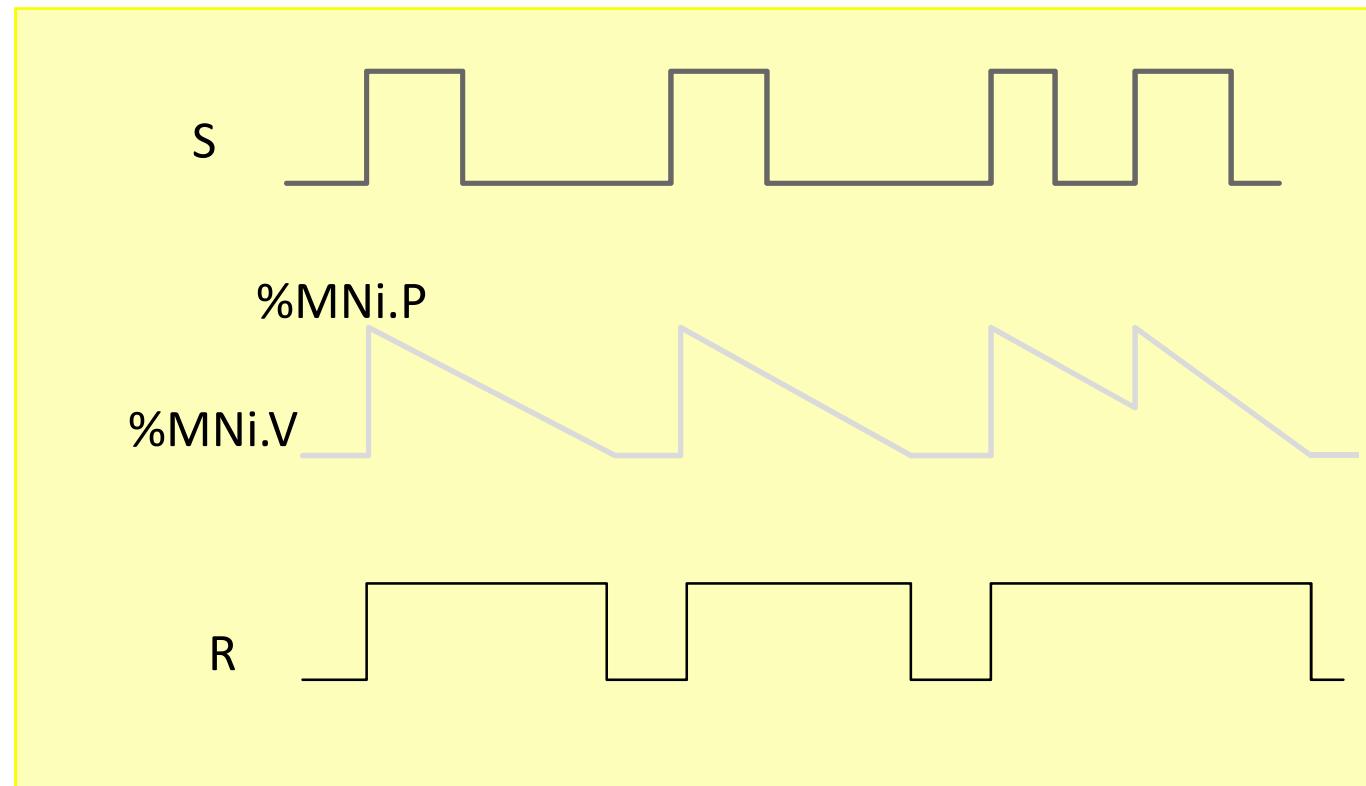


Le monostable permet d'élaborer une impulsion de durée précise



Les blocs fonctions Le monostable %MNi

Exemple

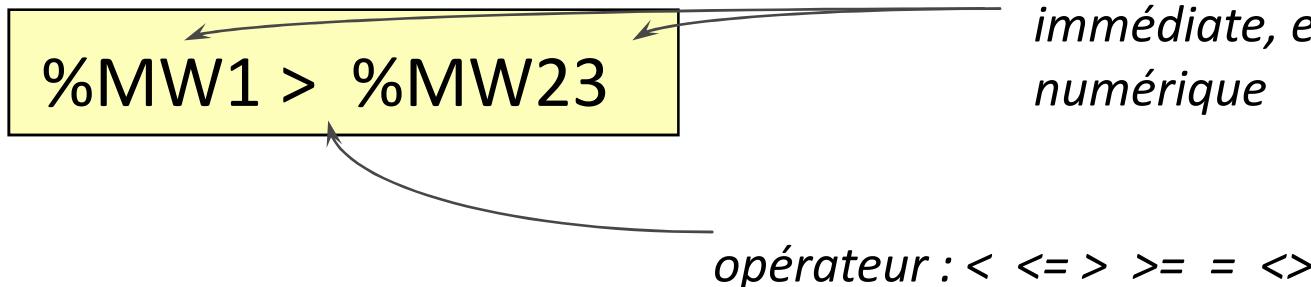




Les blocs fonctions

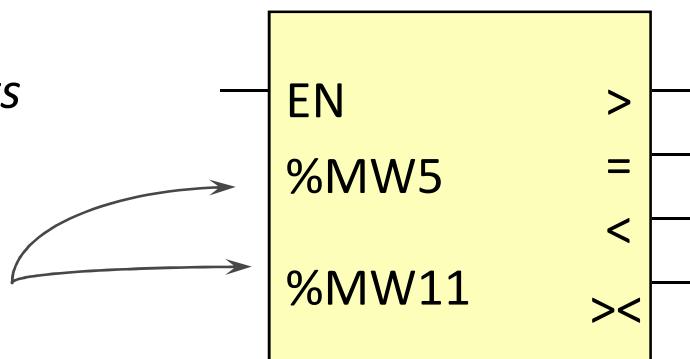
Les blocs comparaisons

■ Comparateur horizontal



■ Comparateur vertical

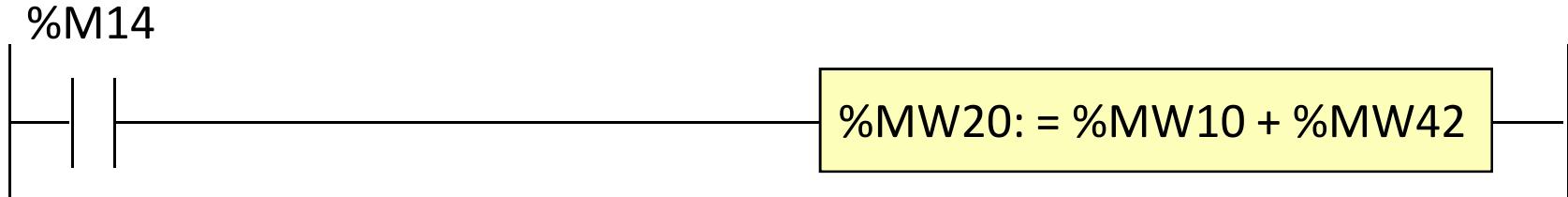
opérande : mot de 16 bits (indexé ou non), valeur immédiate





Les blocs fonctions

Le bloc "OPERATE"



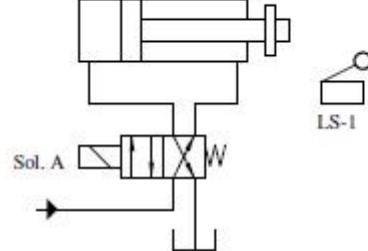
- Permet la saisie:
 - d'opérations de forme simple ou complexe
ex : $%MW30 := (%MW1 * %MW44) + %MW2 - %MW4$
 - des instructions avancées (sur chaîne de caractères, horodateur , tableaux)
 - des fonctions liées aux métiers (régulation, communication...)
- Le symbole $:=$ exprime le transfert



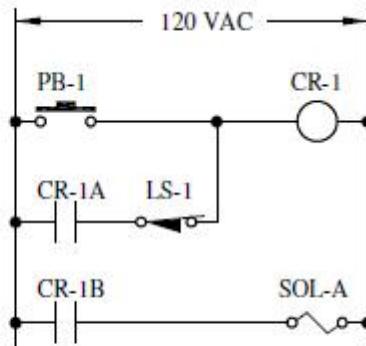
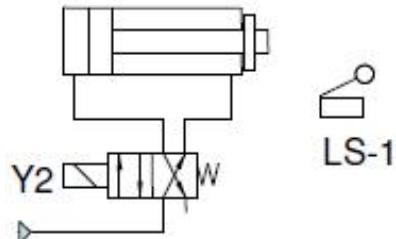
le résultat de l'opération est toujours placé à gauche de l'opérateur



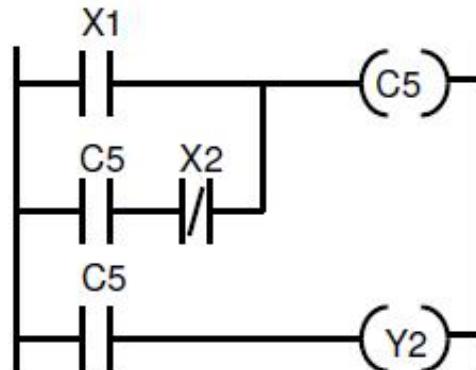
Exemple d'un simple système hydraulique :



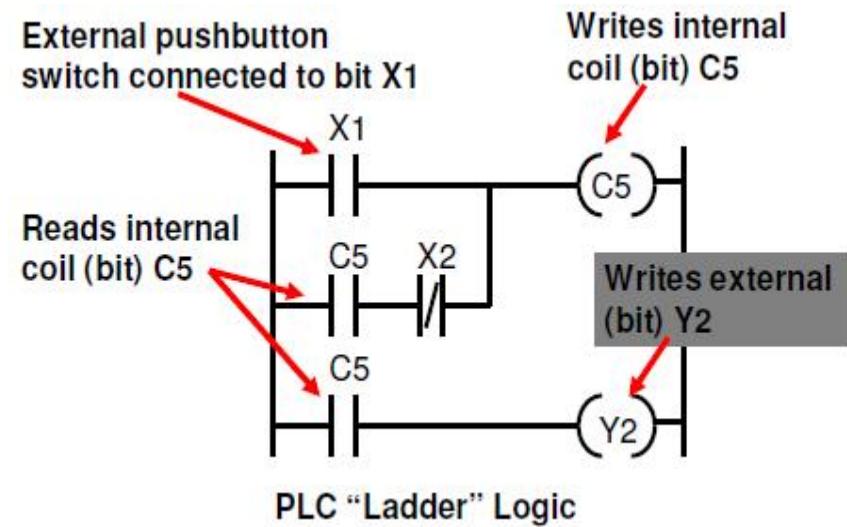
Hydraulic System

Relay Ladder Logic (RLL)
Control System

Pneumatic System

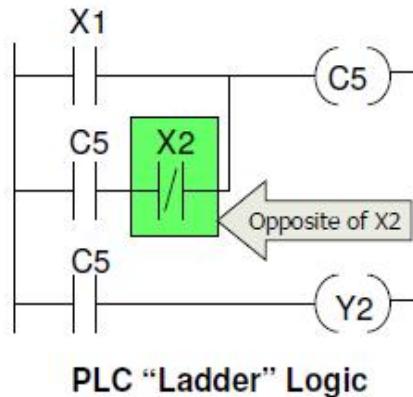
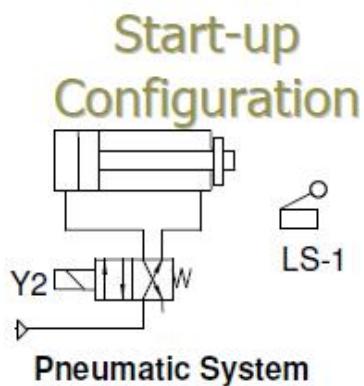


PLC "Ladder" Logic

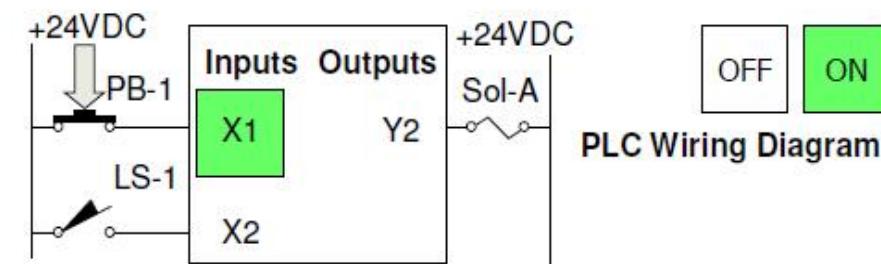
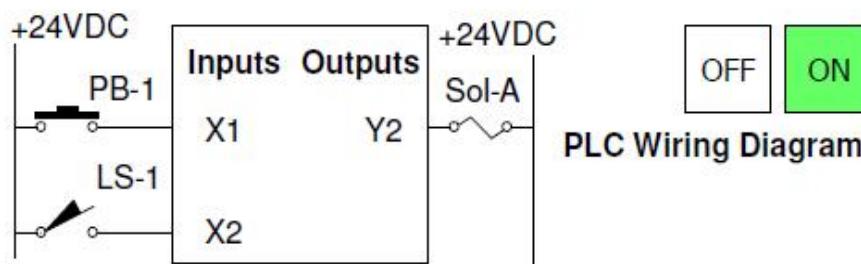
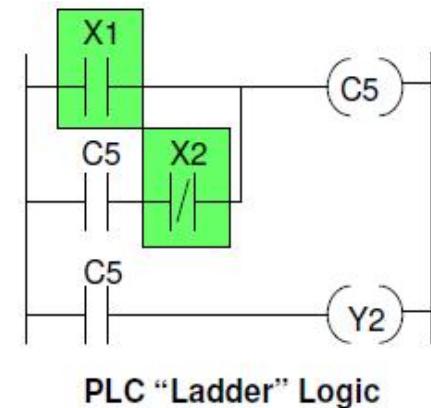
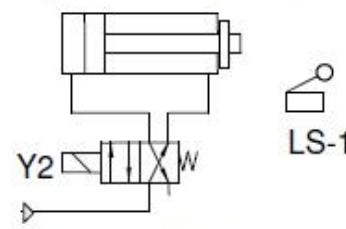




Suite de l'exemple



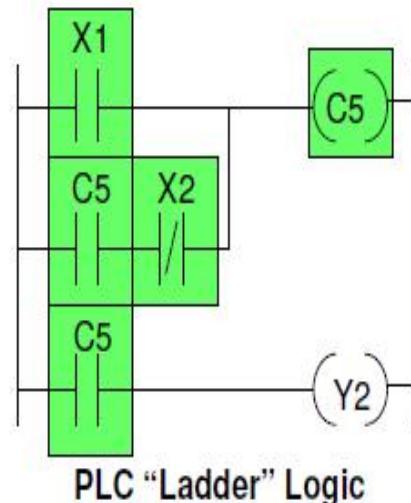
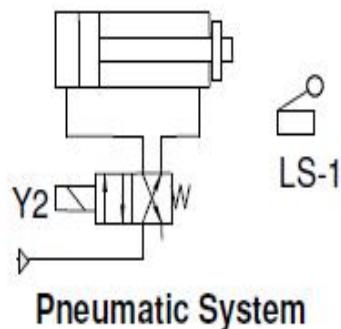
- Press the PB-1 pushbutton
- Contacts X1 turn ON



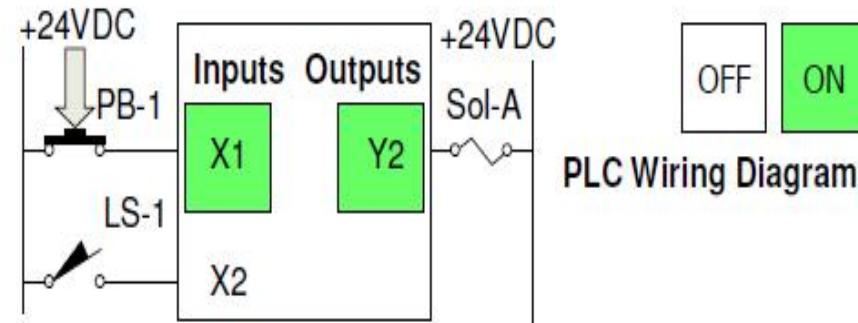
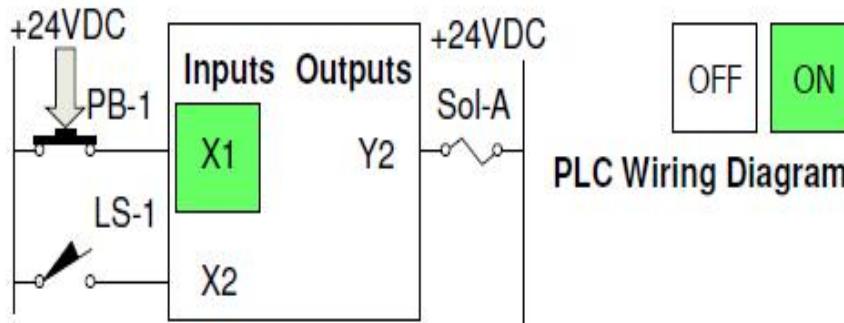
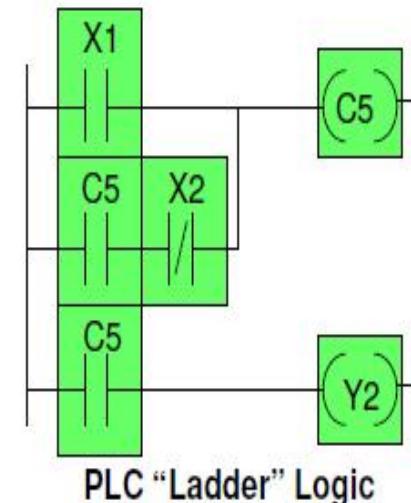
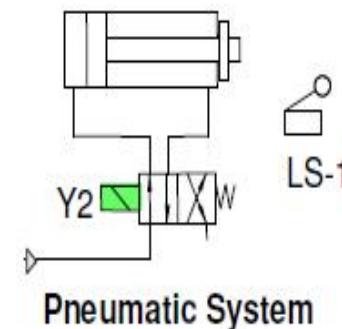


Suite de l'exemple

- Output relay C5 turned ON
- Inputs C5 read as ON



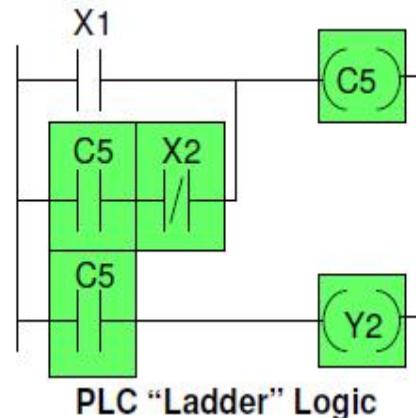
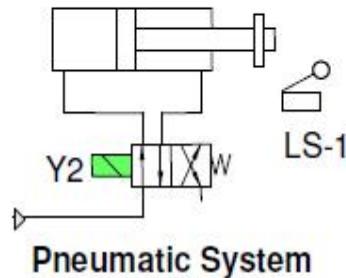
- Output relay Y2 turned ON
- Valve solenoid shifts spool



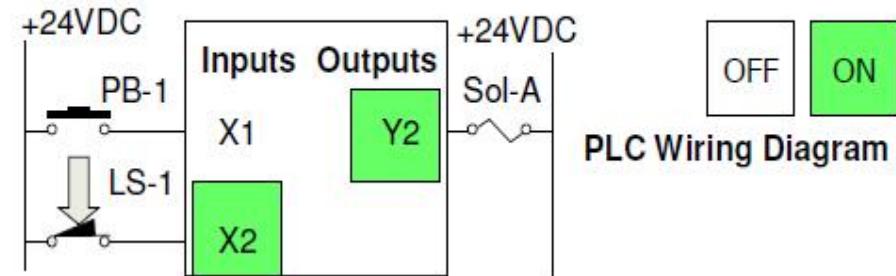
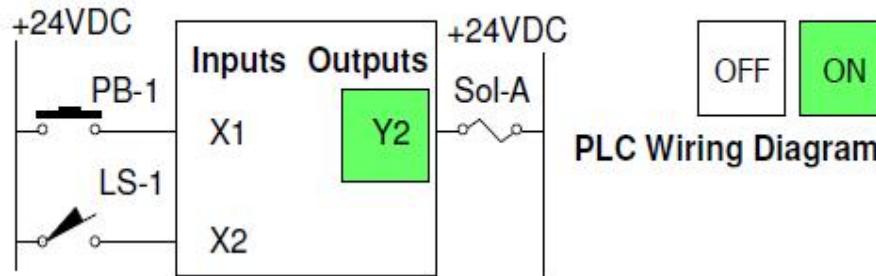
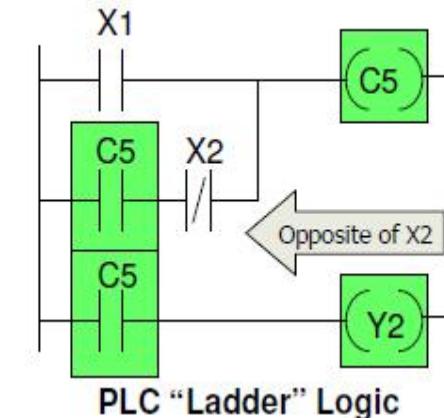
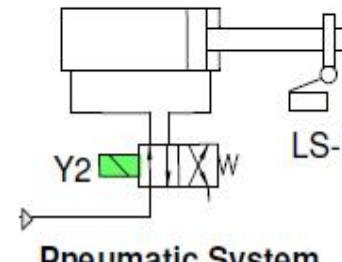


Suite de l'exemple

- Pushbutton released
- Contact X1 turned OFF



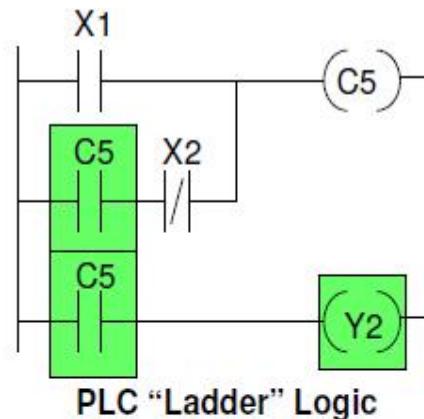
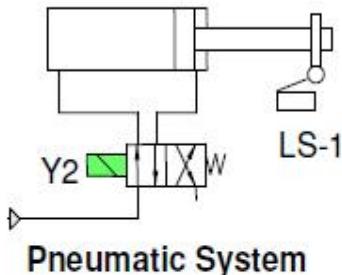
- LS-1 is activated
- Contact X2 turned ON



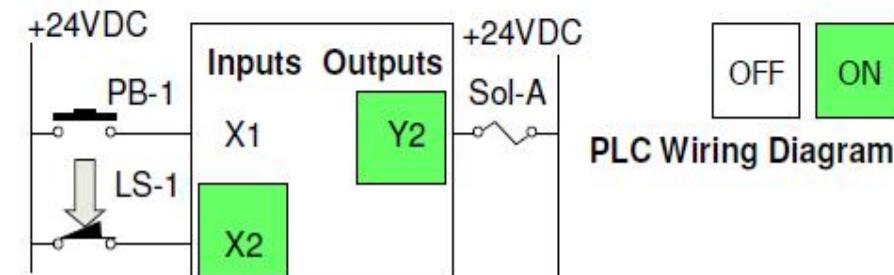
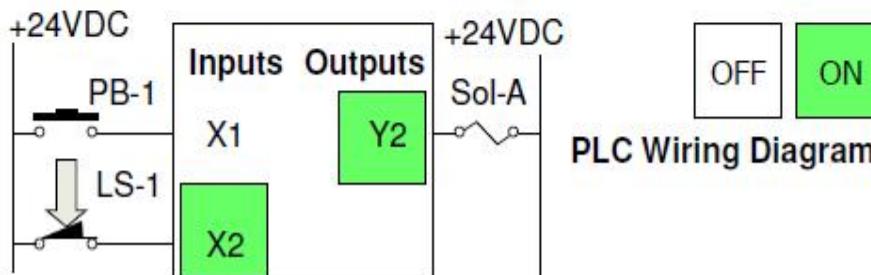
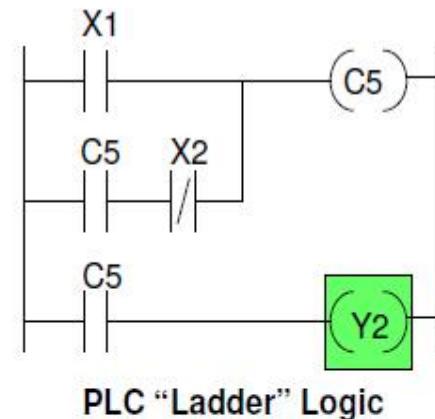
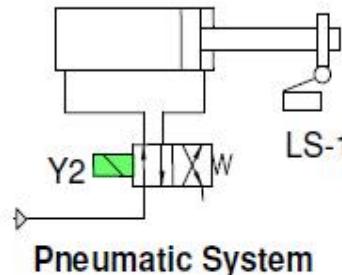


Suite de l'exemple

- Hold is "broken"
- Output relay C5 turned OFF



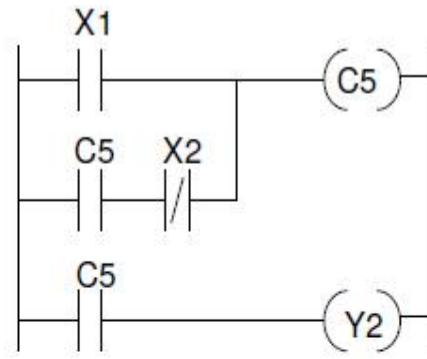
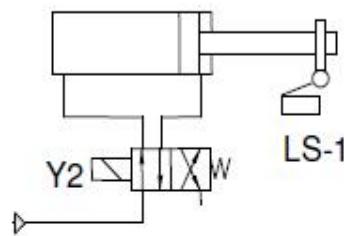
- Contacts C5 turned OFF



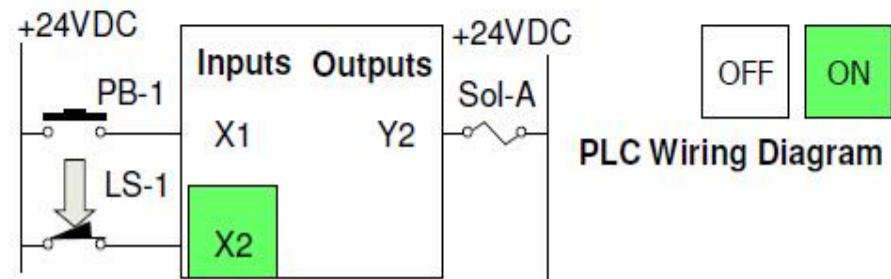
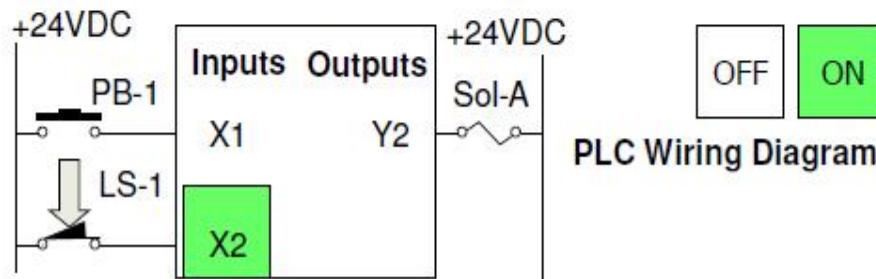
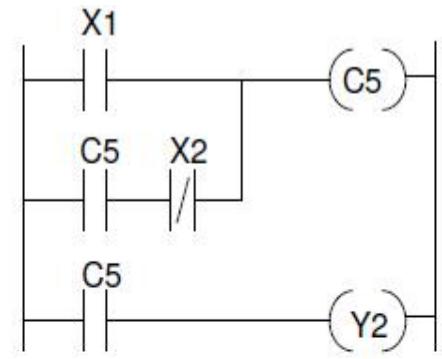
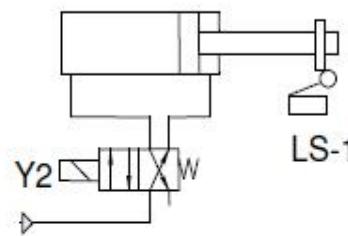


Suite de l'exemple

- Output Y2 turned OFF
- Solenoid Y2 turned OFF



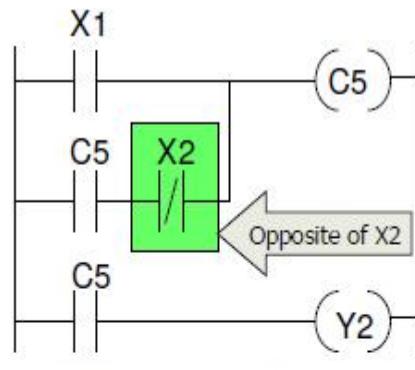
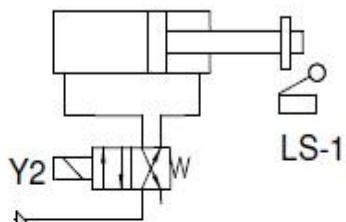
- Valve spool shifts to left
- Cylinder begins to retract



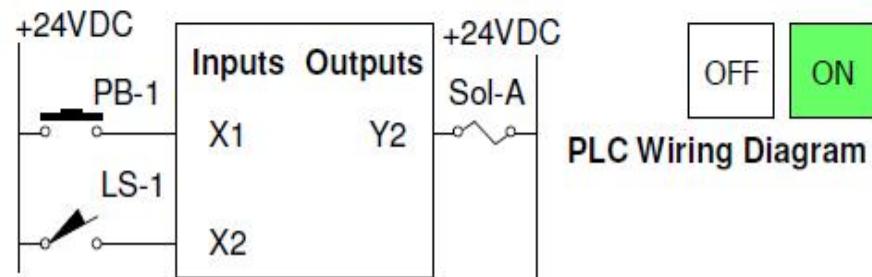
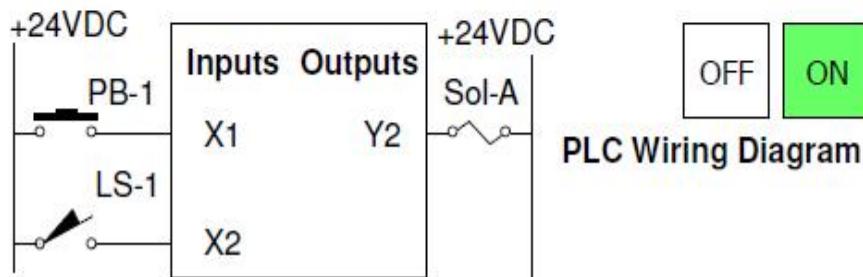
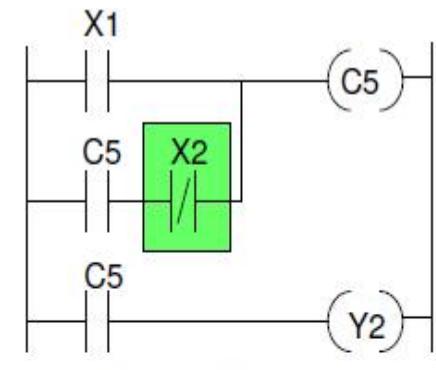
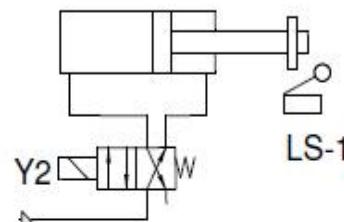


Suite de l'exemple

- Limit switch LS-1 released
- Input X2 goes OFF

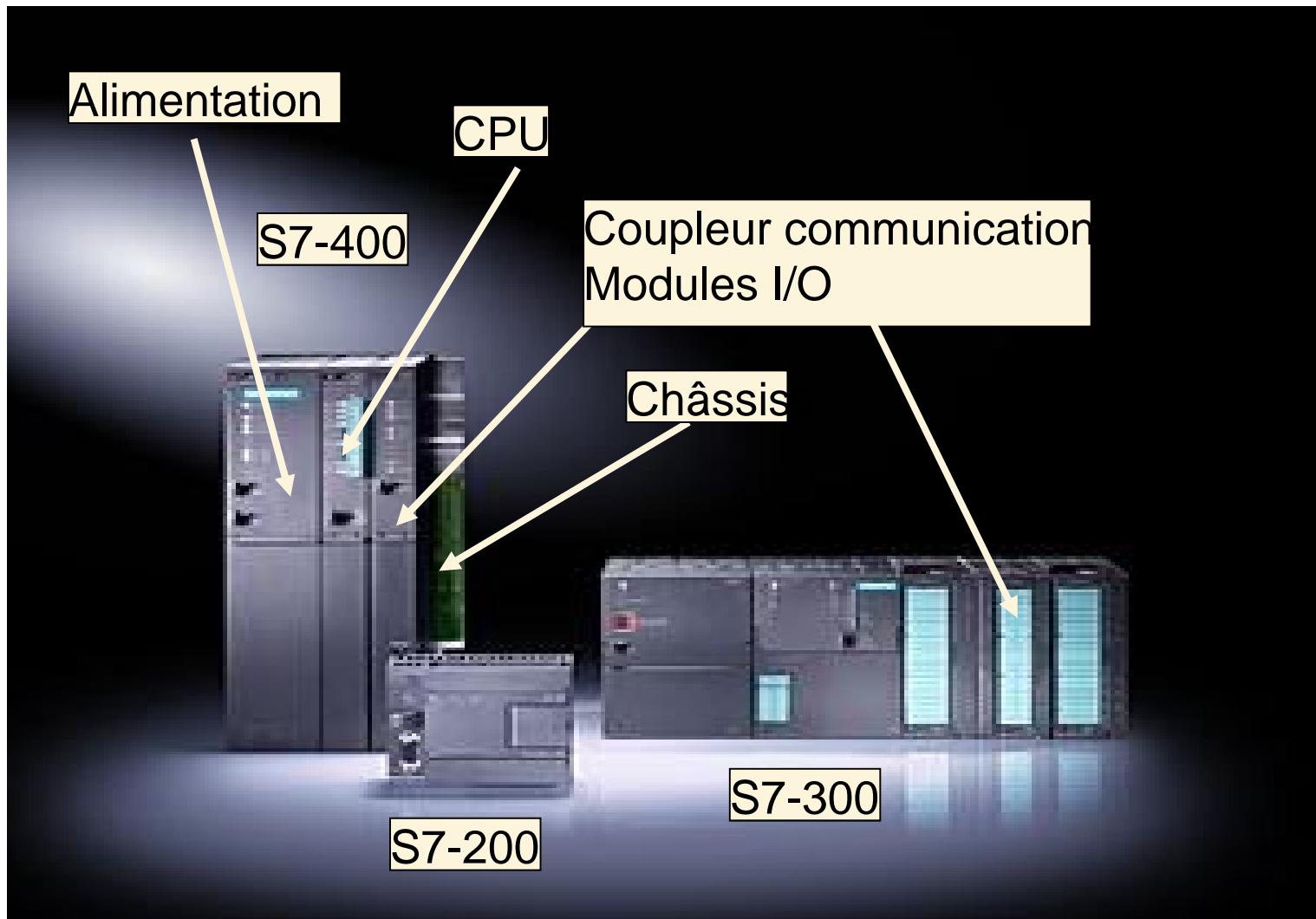


- Cylinder fully retracts to initial start configuration





LA GAMME S7-200, S7-300, S7-400





L'AUTOMATE S7-200

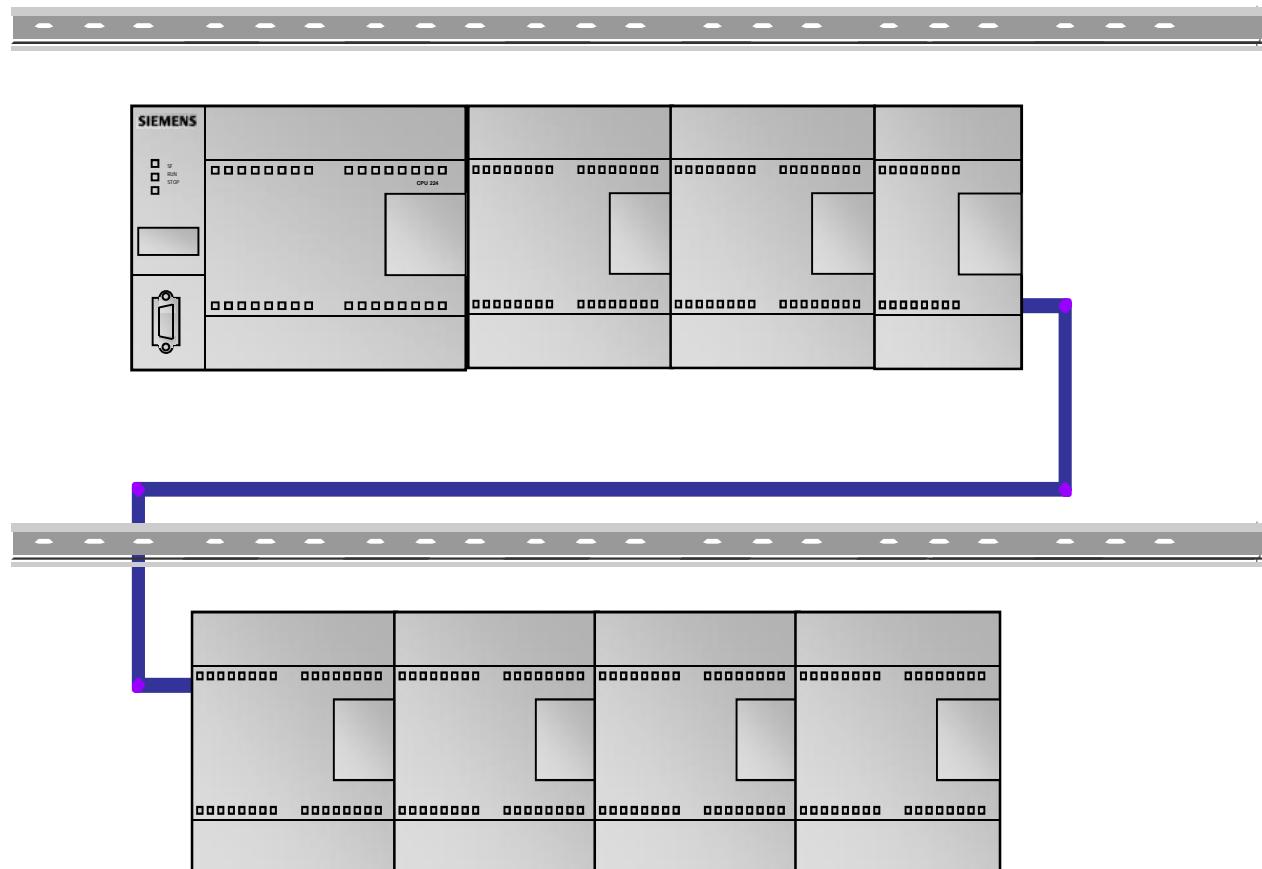




SAP

Informatique industrielle-141

L'AUTOMATE S7-200





SAP

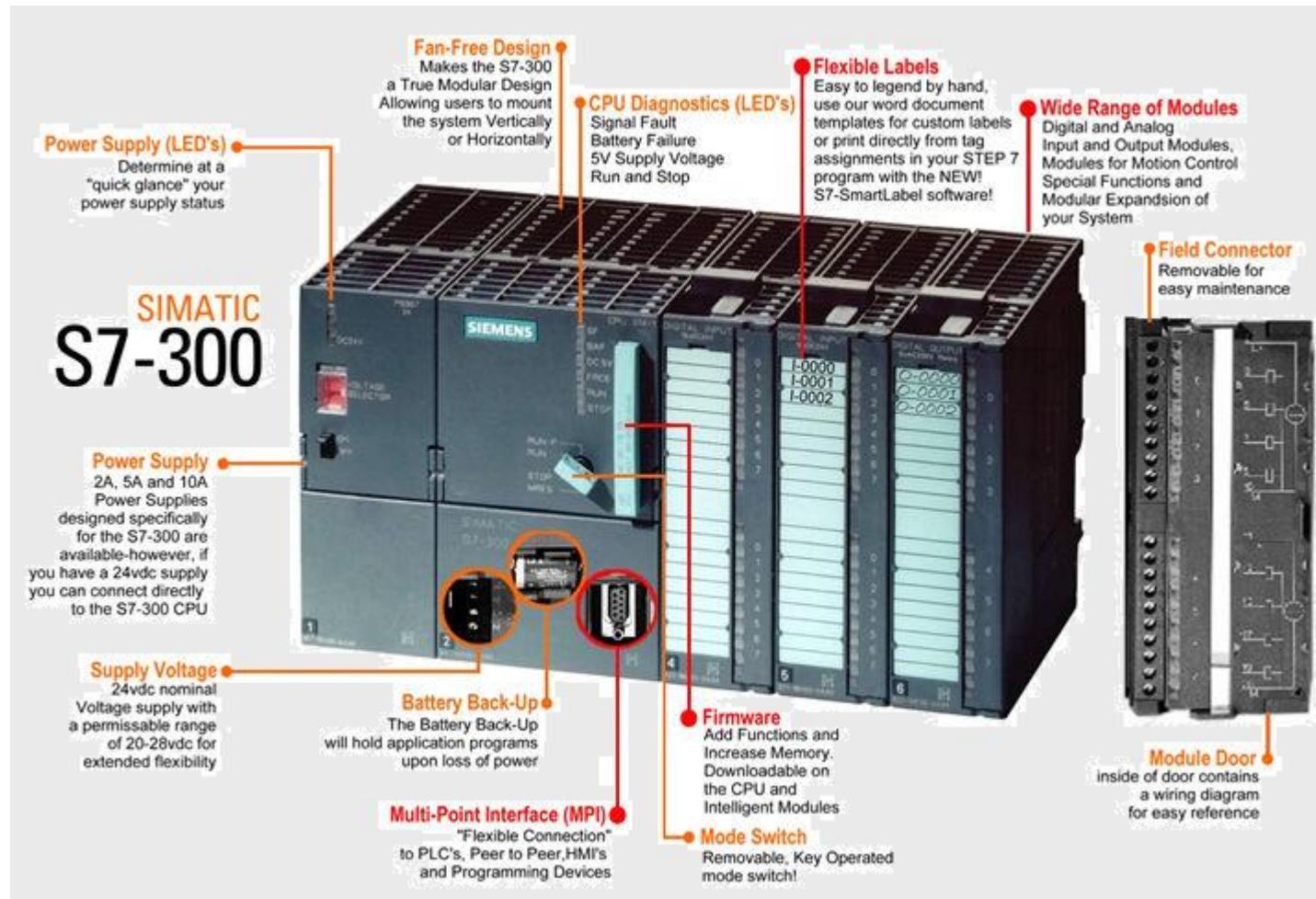
Informatique industrielle-142

L'AUTOMATE S7-200



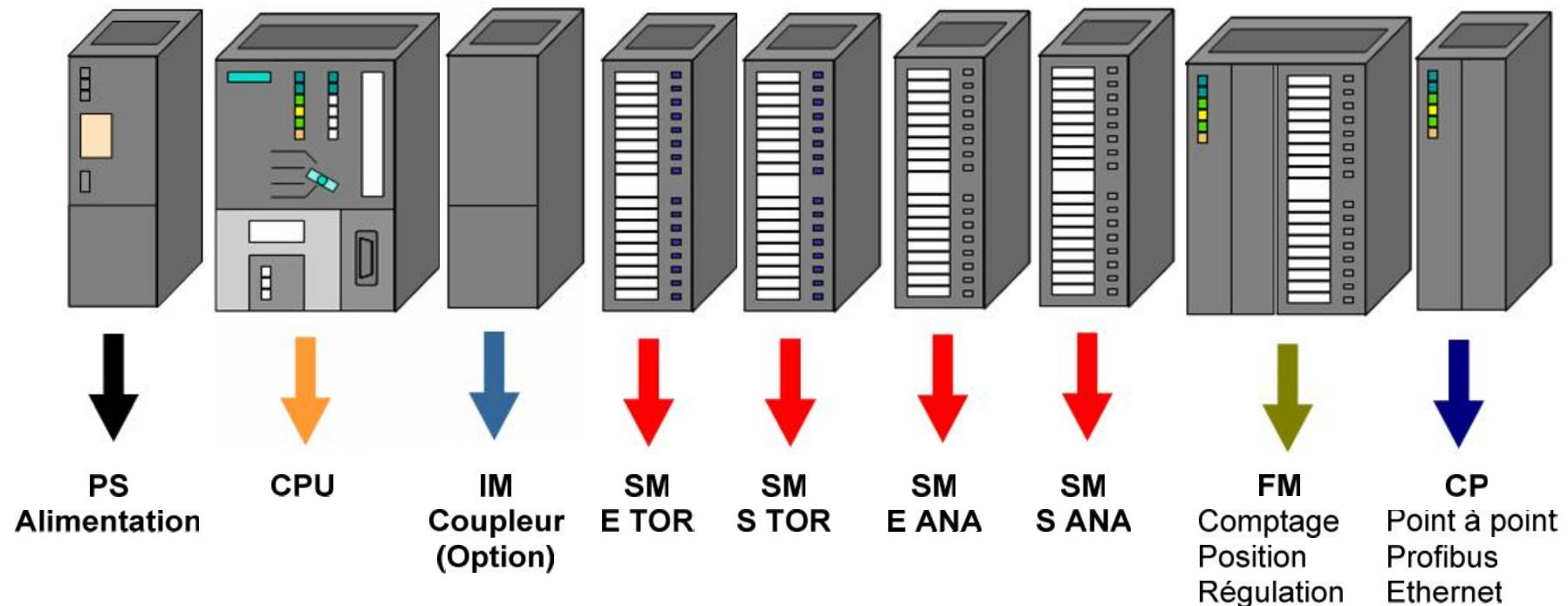


L'AUTOMATE S7-300



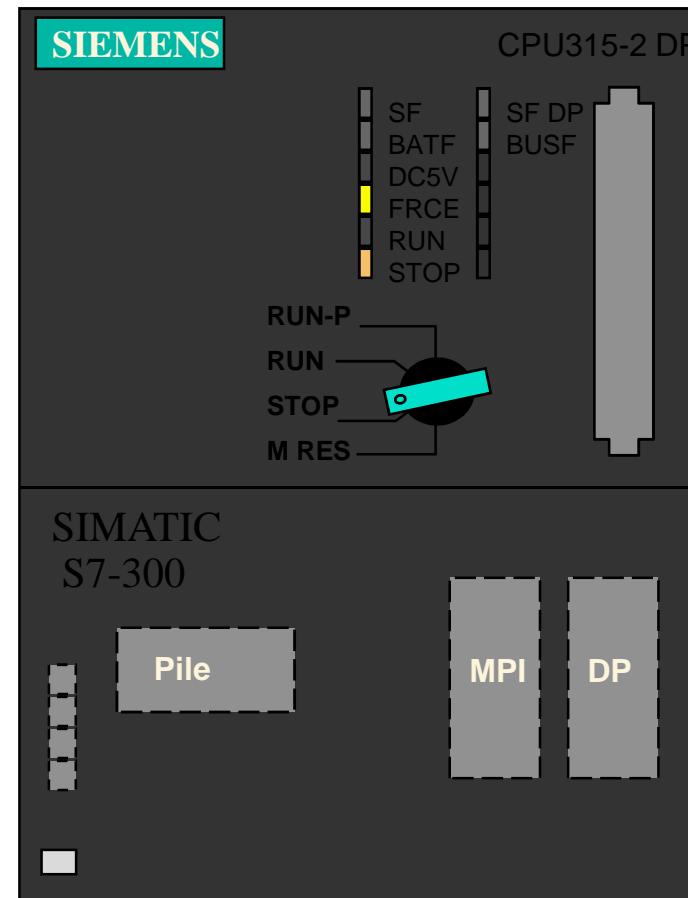
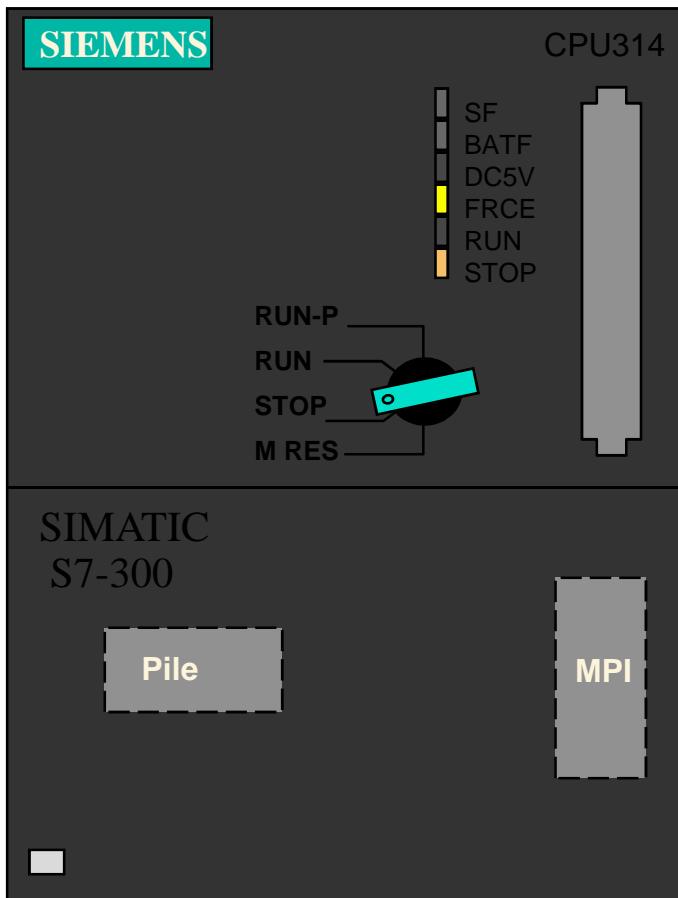


L'AUTOMATE S7-300





CPU S7-300





SAP

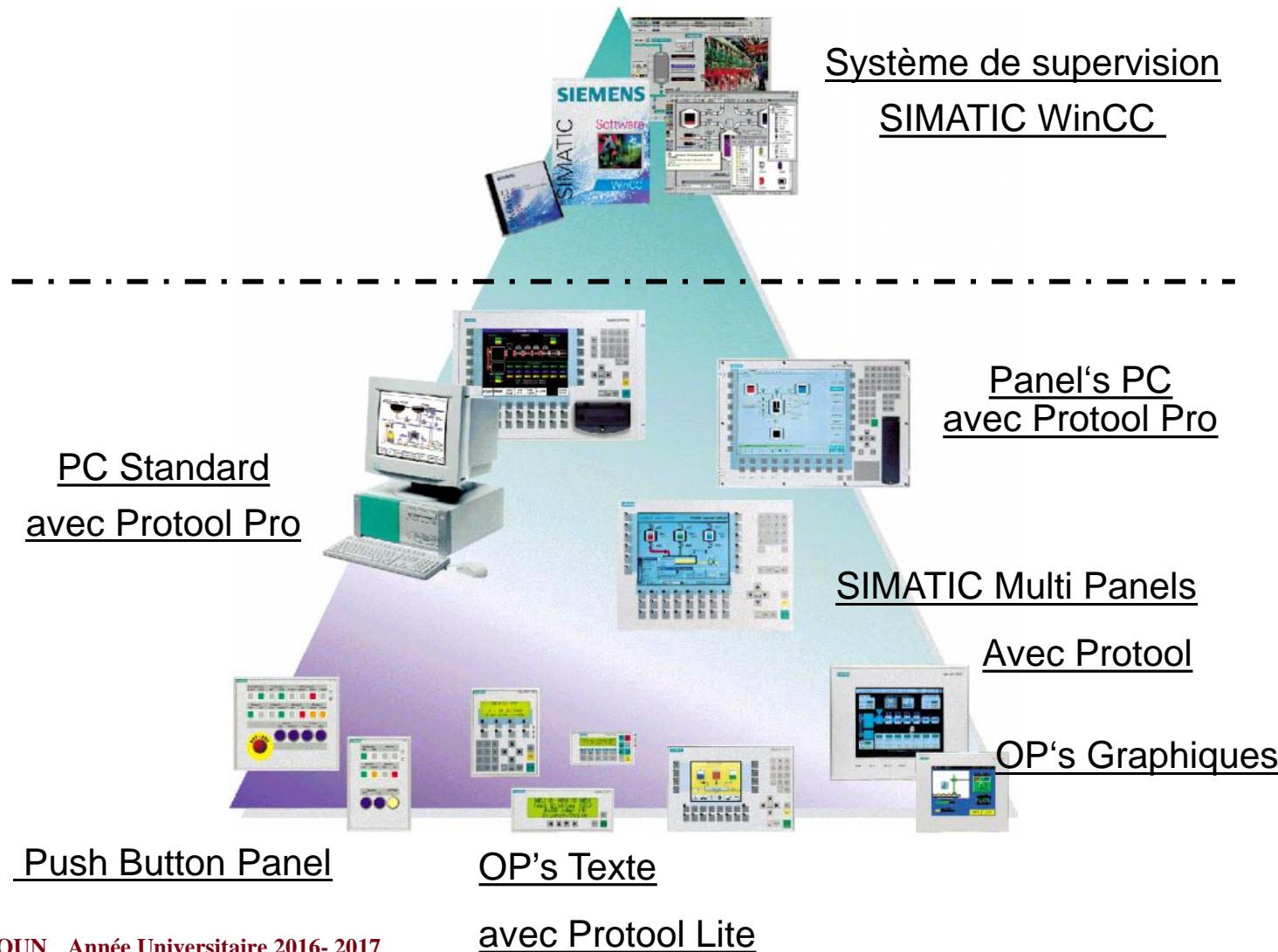
Informatique industrielle-146

L'AUTOMATE CPU S7-400





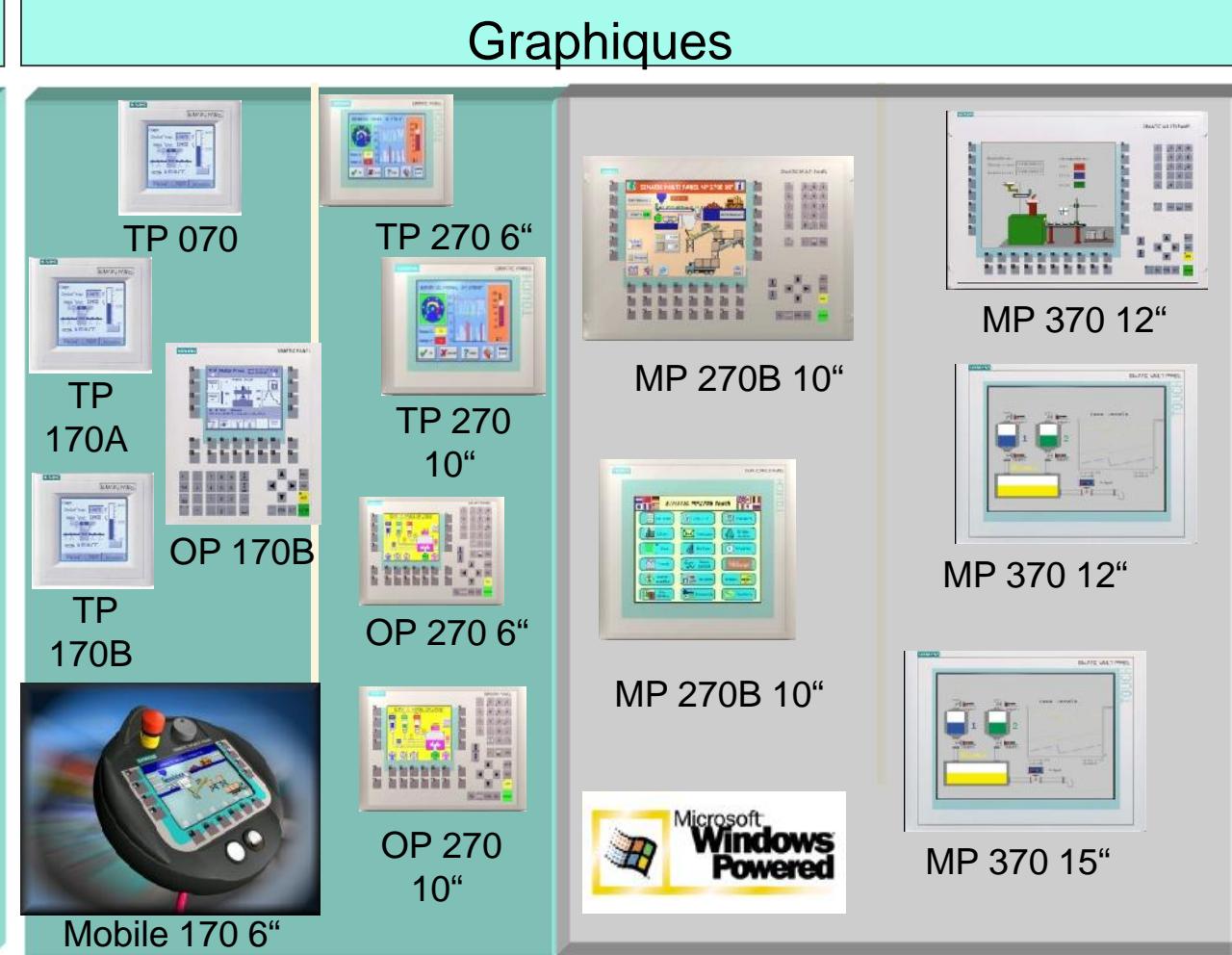
INTERFACES HOMME/MACHINE





INTERFACES HOMME/MACHINE

Textes





SAP

Informatique industrielle-149

CONSOLE DE PROGRAMMATION



PG 720



PG 740

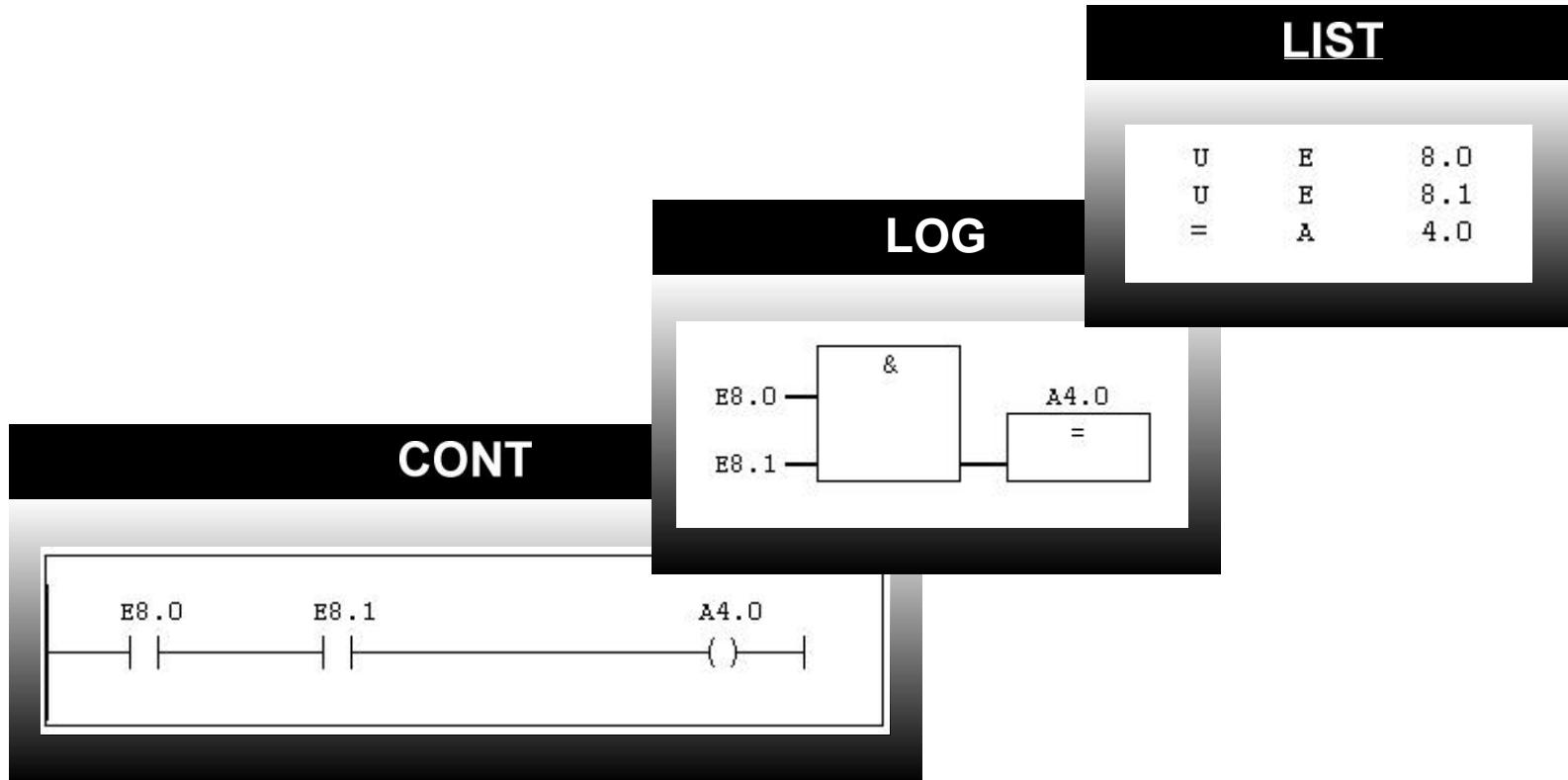




SAP

Informatique industrielle-150

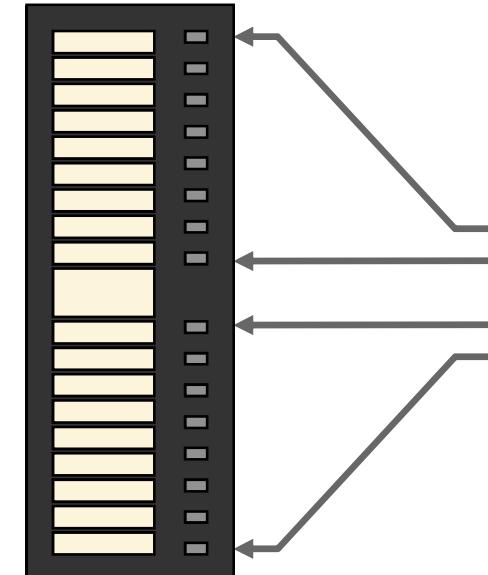
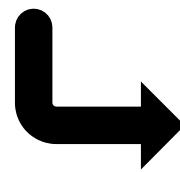
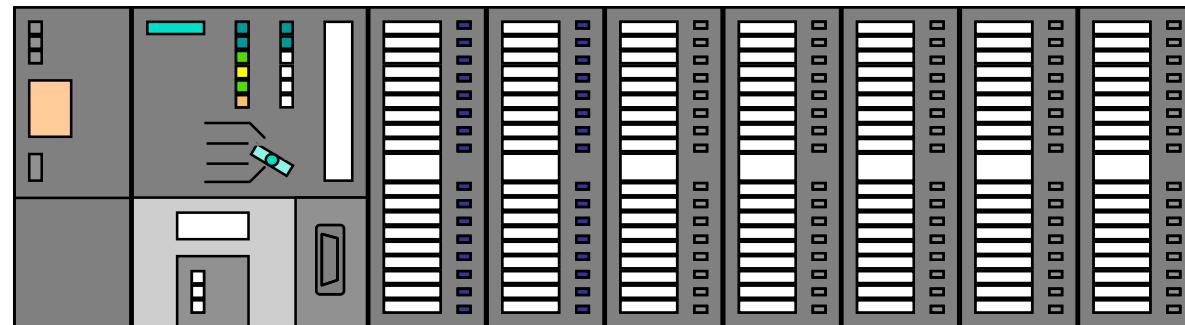
LE LANGAGE STEP7





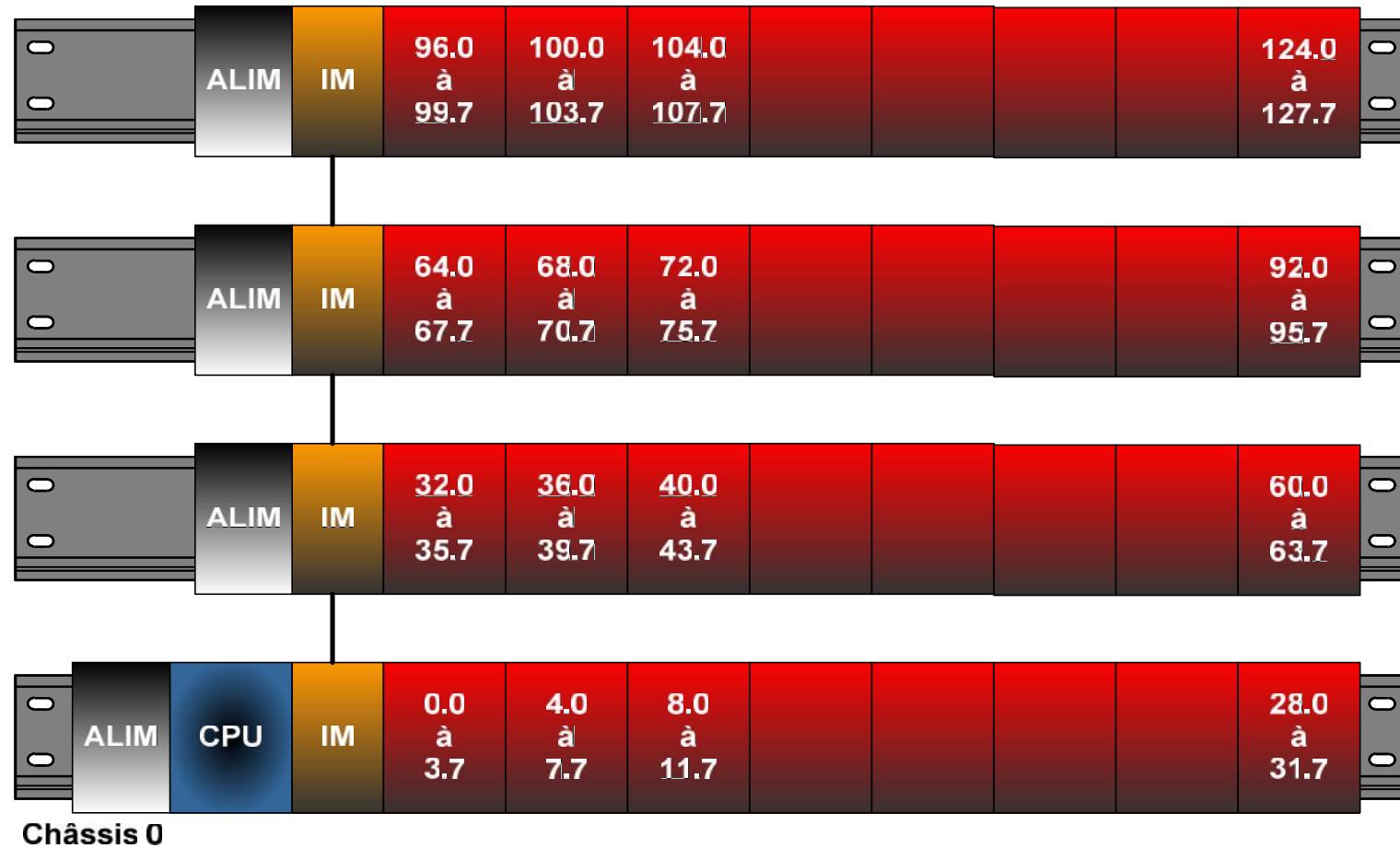
ADRESSAGE DES MODULES TOR S7 300

N° empl → 1 2 3 4 5 6 7





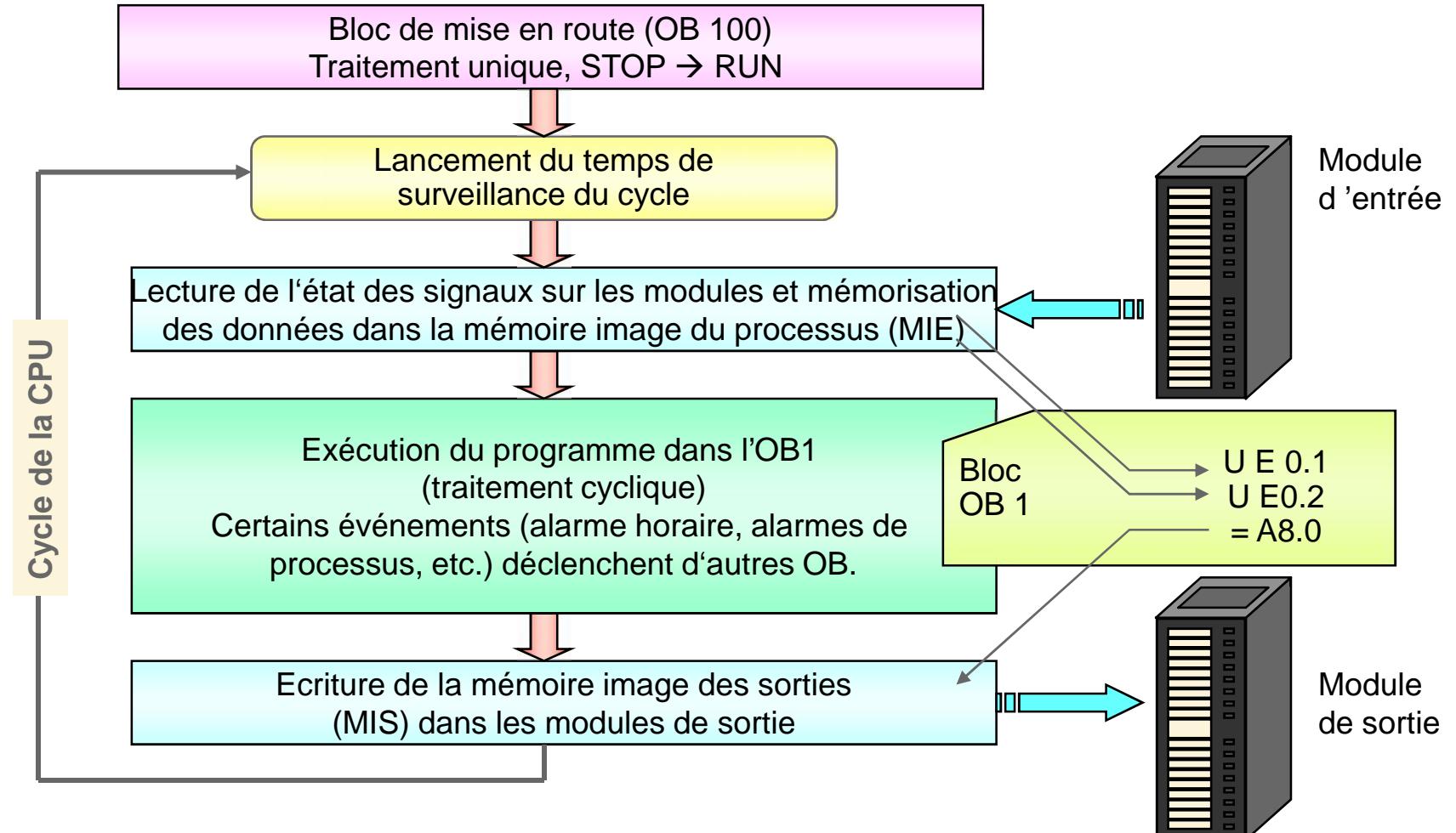
ADRESSAGE DES MODULES TOR S7 300



N° empl → 1 2 3 4 5 6 7 8 9 10 11

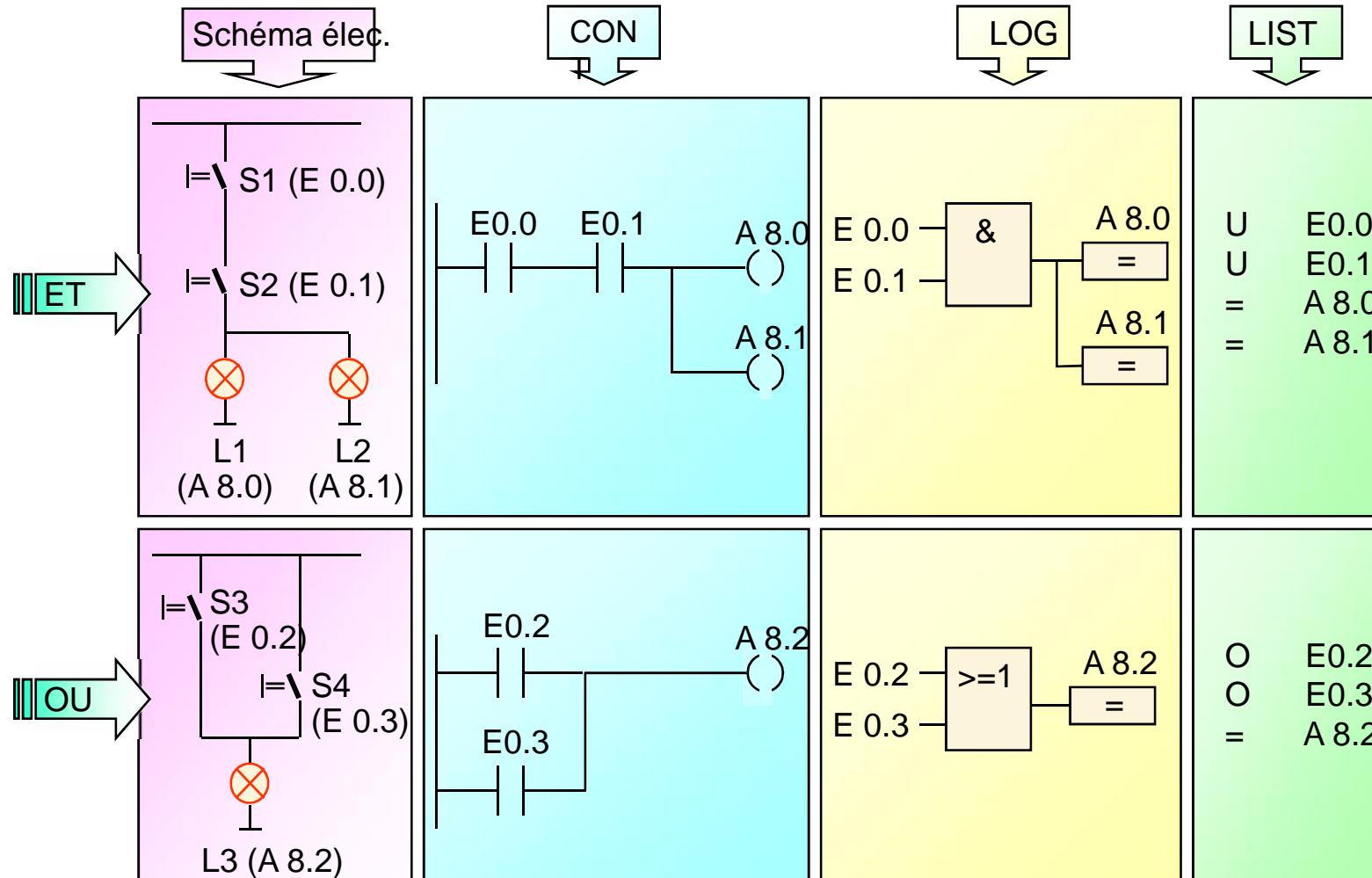


TRAITEMENT CYCLIQUE DU PROGRAMME



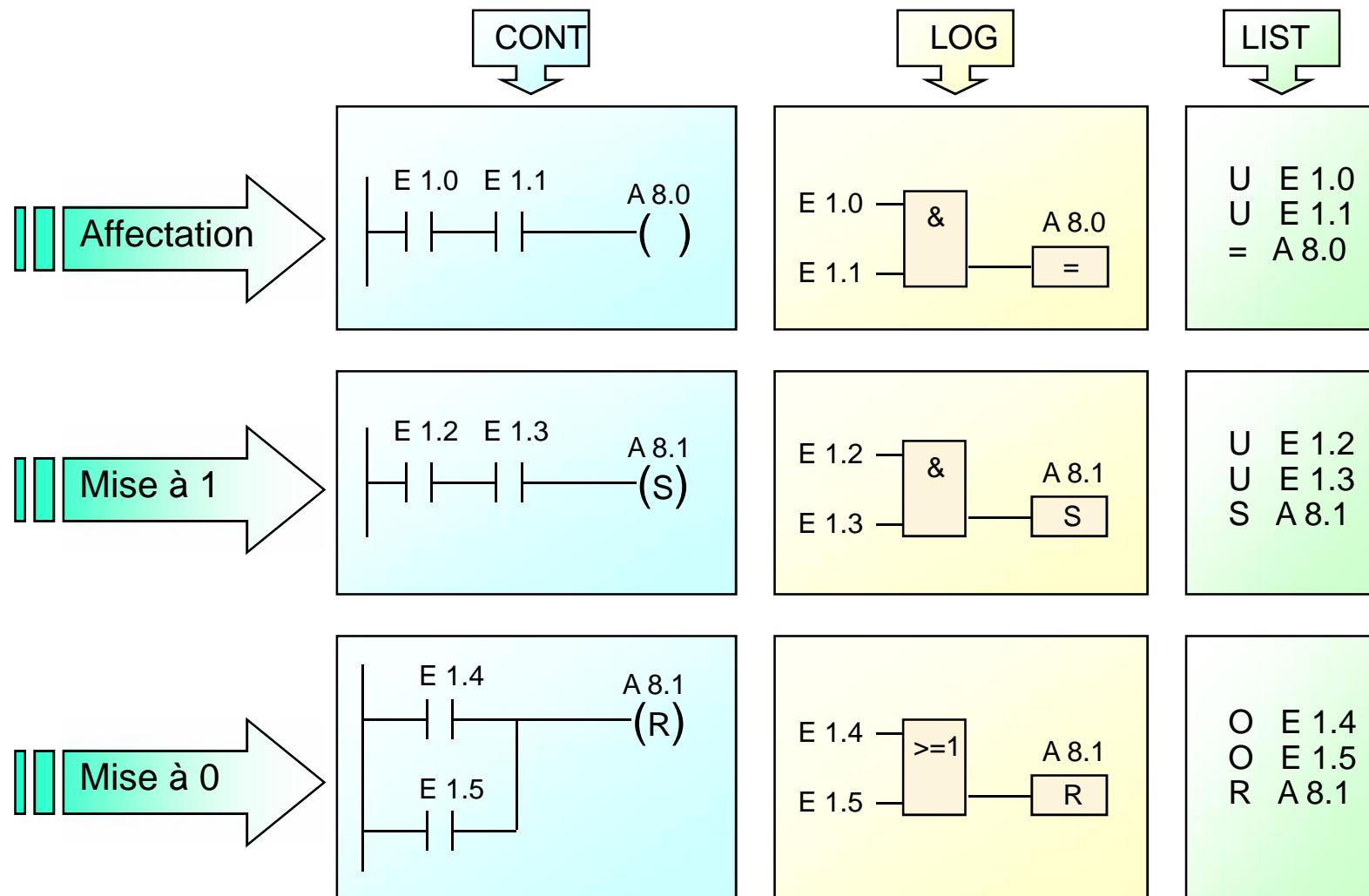


OPERATIONS COMBINATOIRE BINAIRES: ET OU



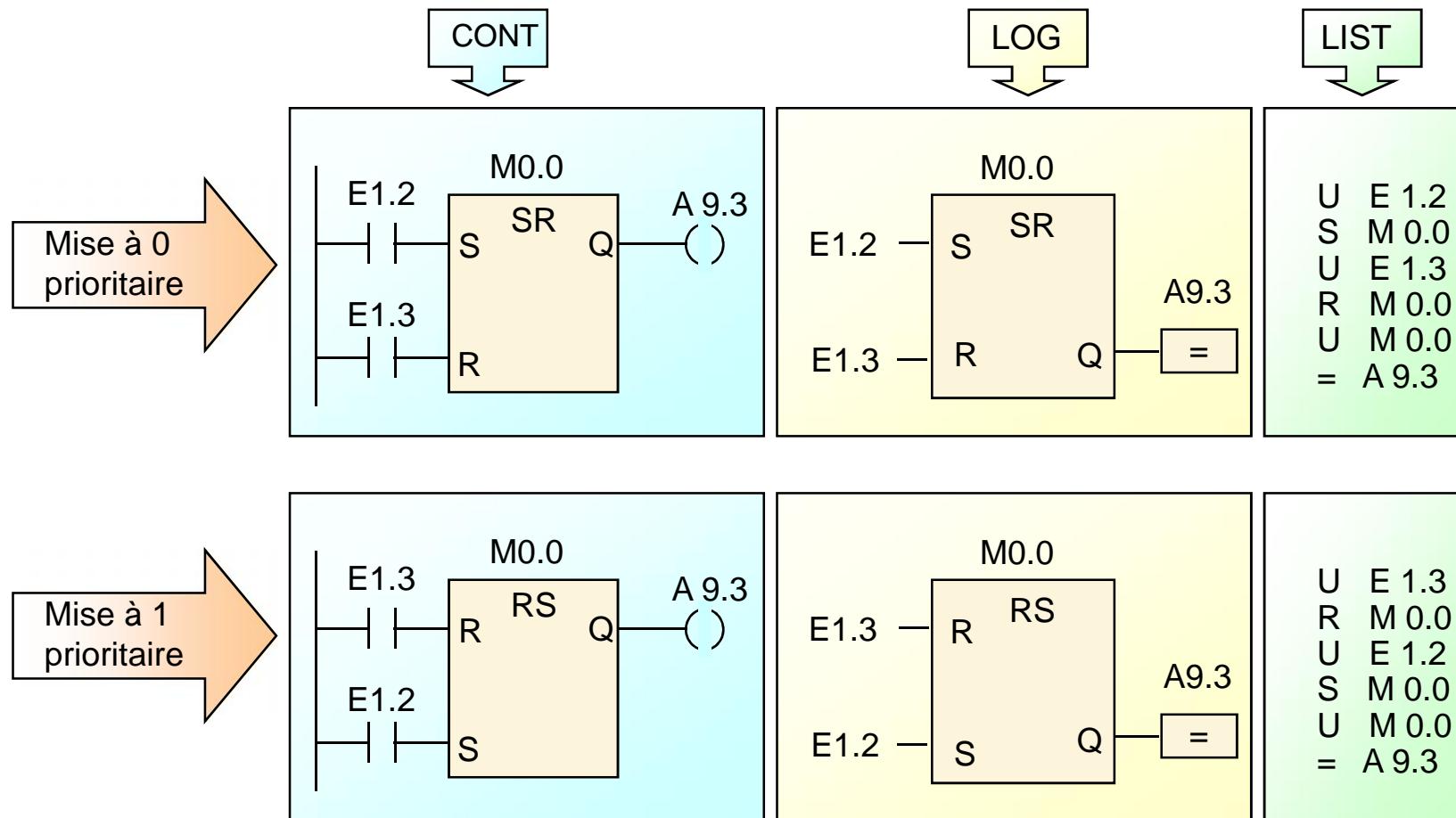


AFFECTATION, MISE A 1, MISE A 0



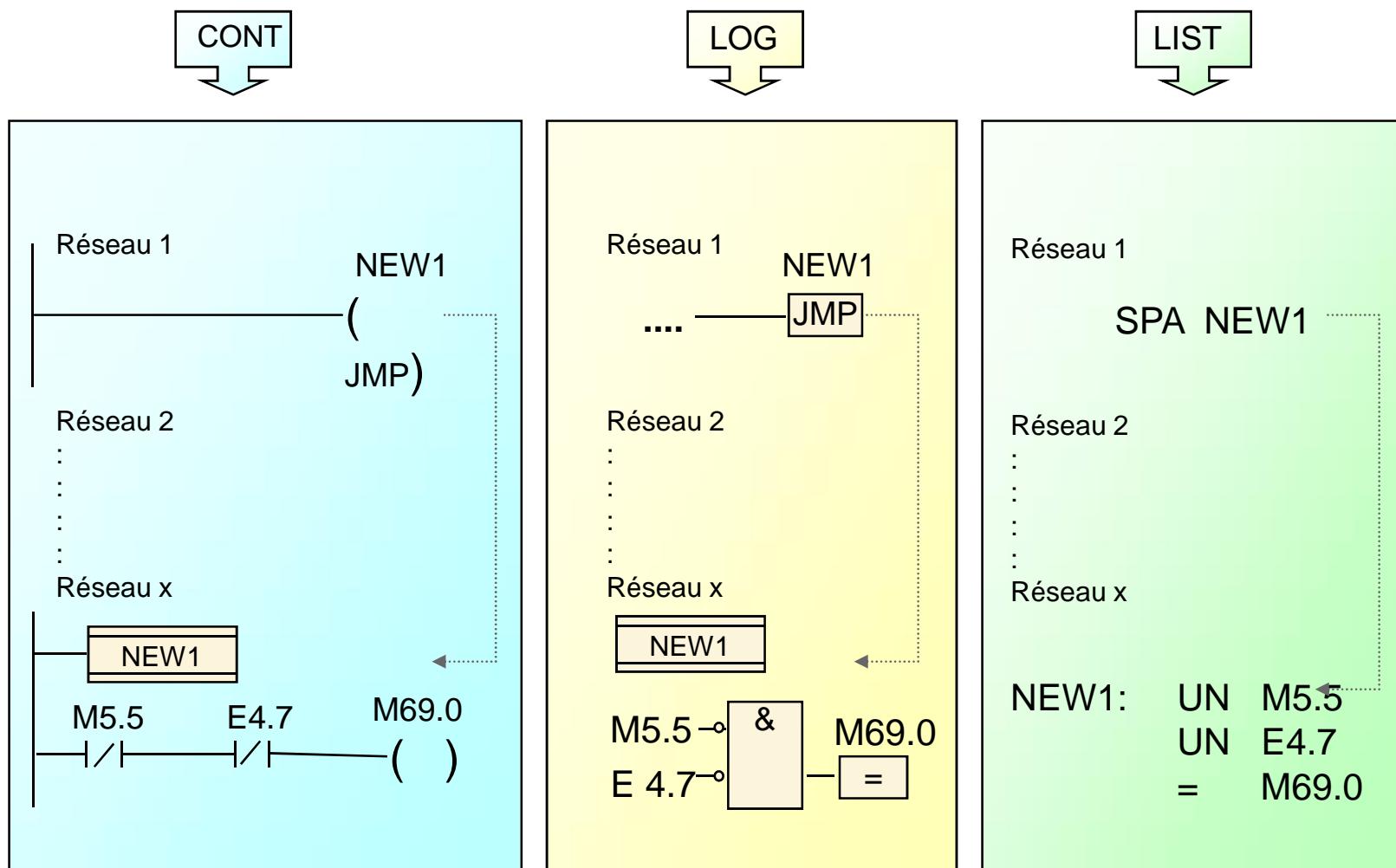


BASCULE MISE A 1/MISE A 0



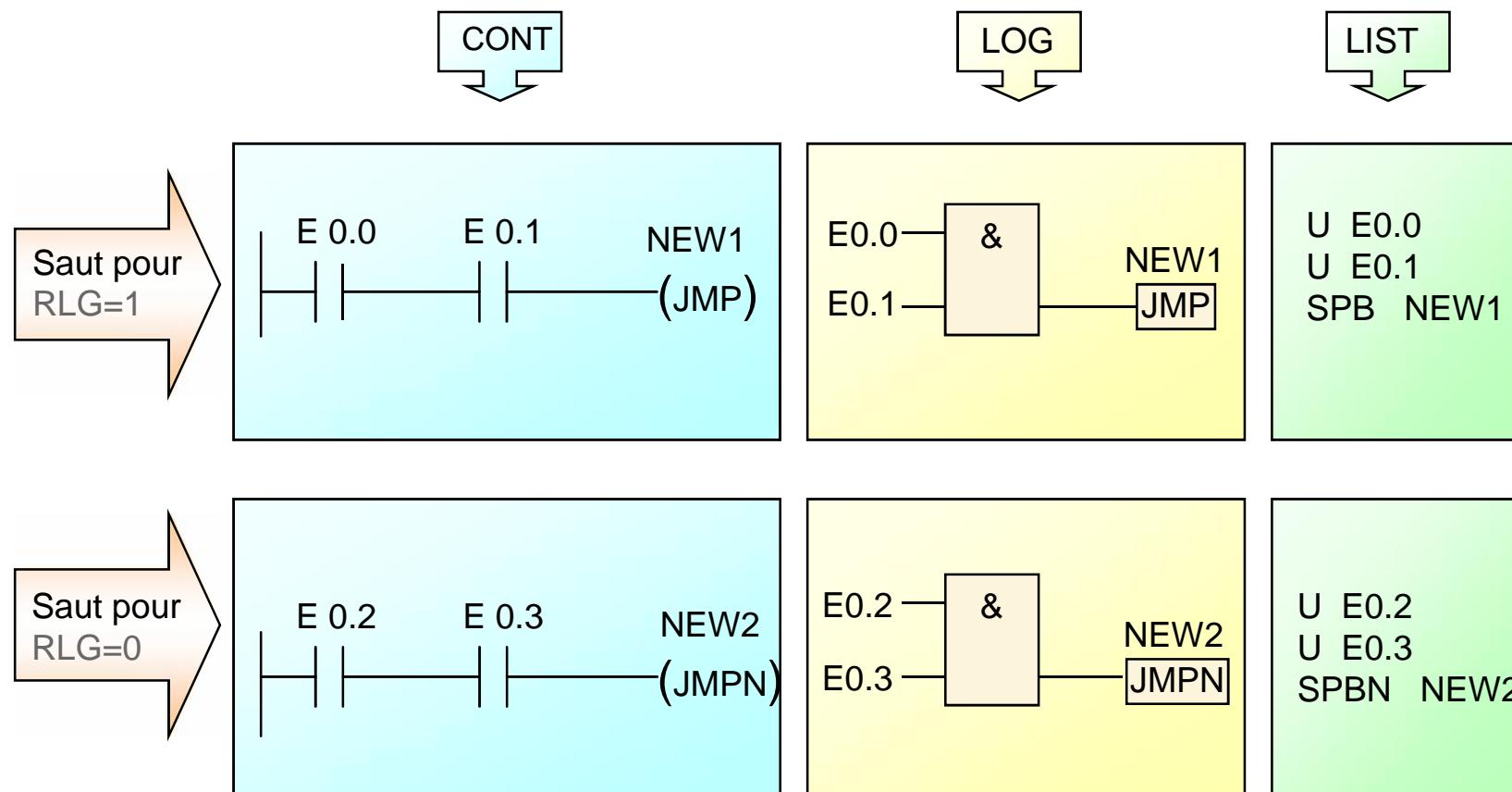


SAUT INCONDITIONNEL



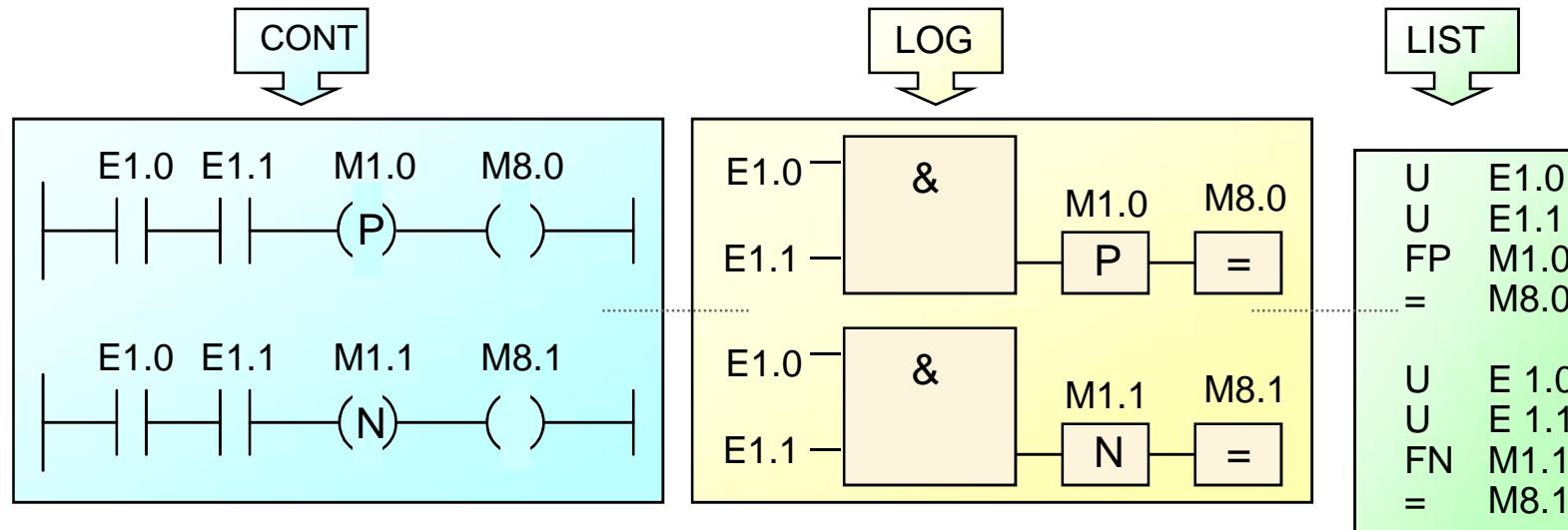


SAUT INCONDITIONNEL

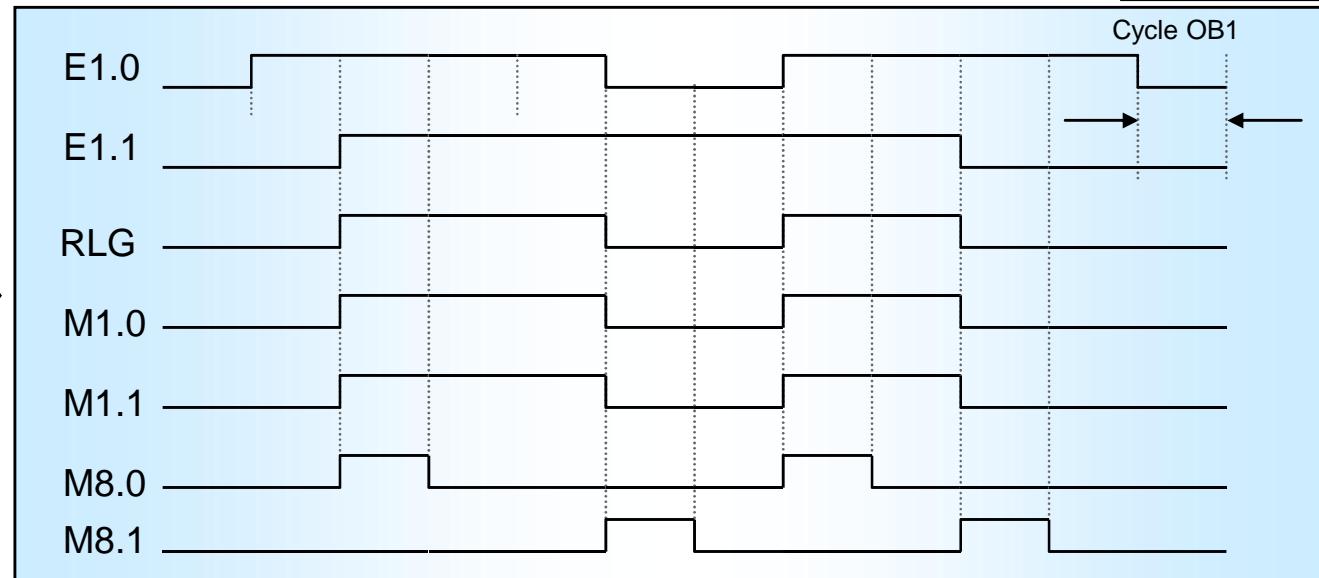




REPONSE AUX FRONTS

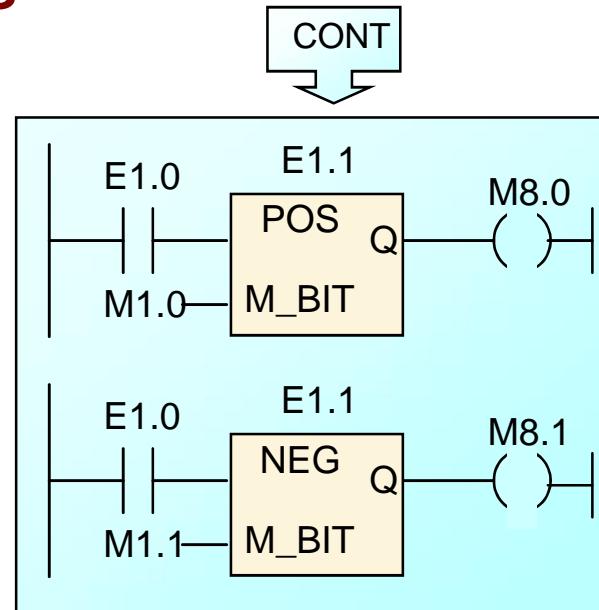


Exemple

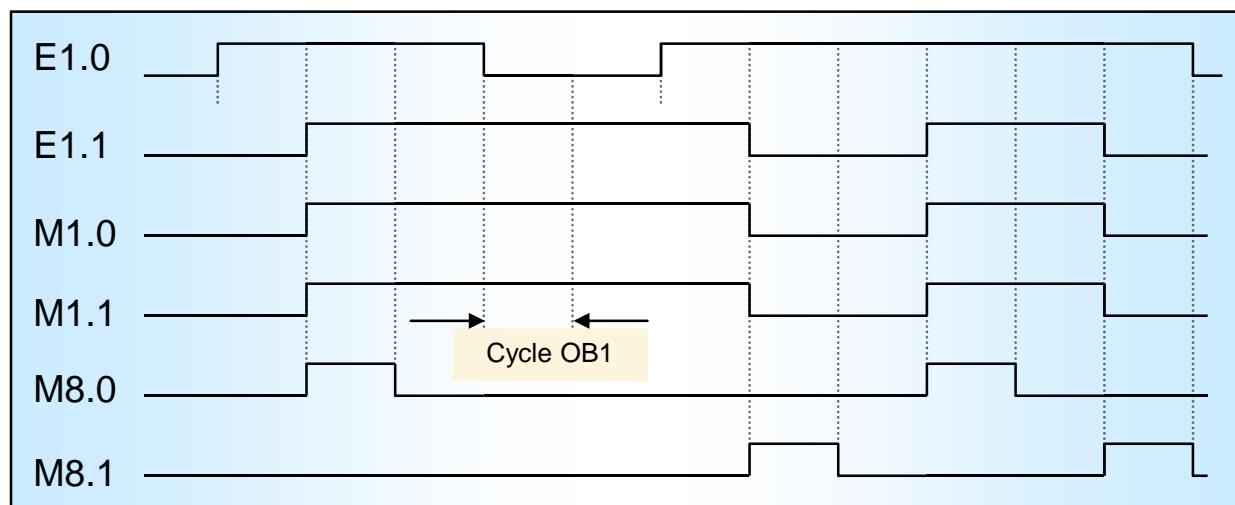




REPONSE AUX FRONTS

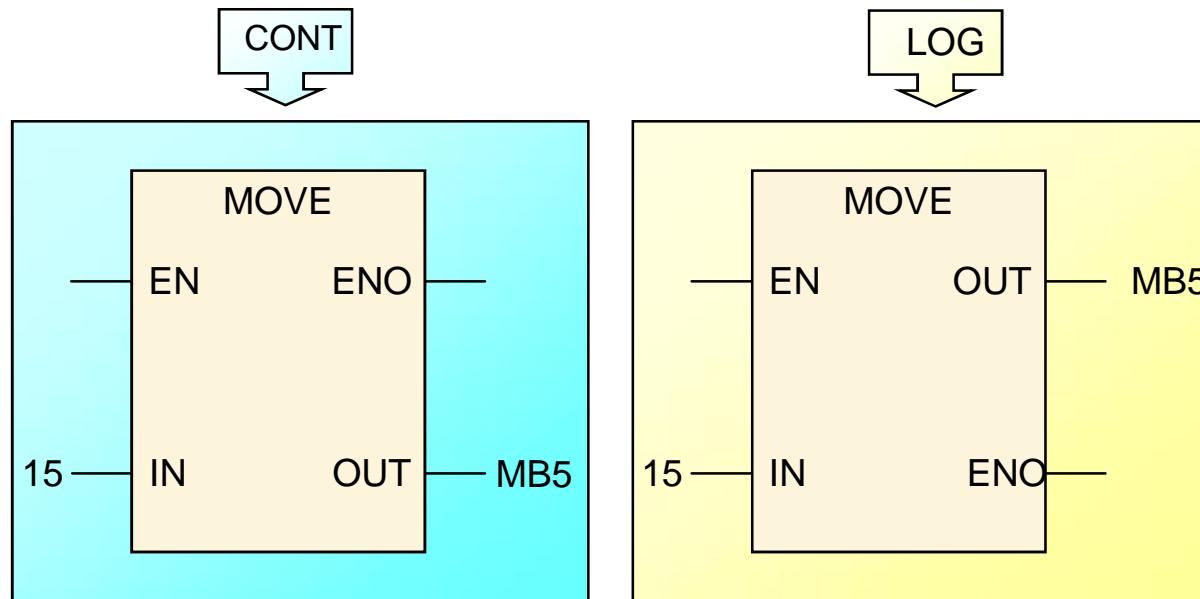


➤ Exemple





CHARGEMENT ET TRANSFERT DE DONNEES

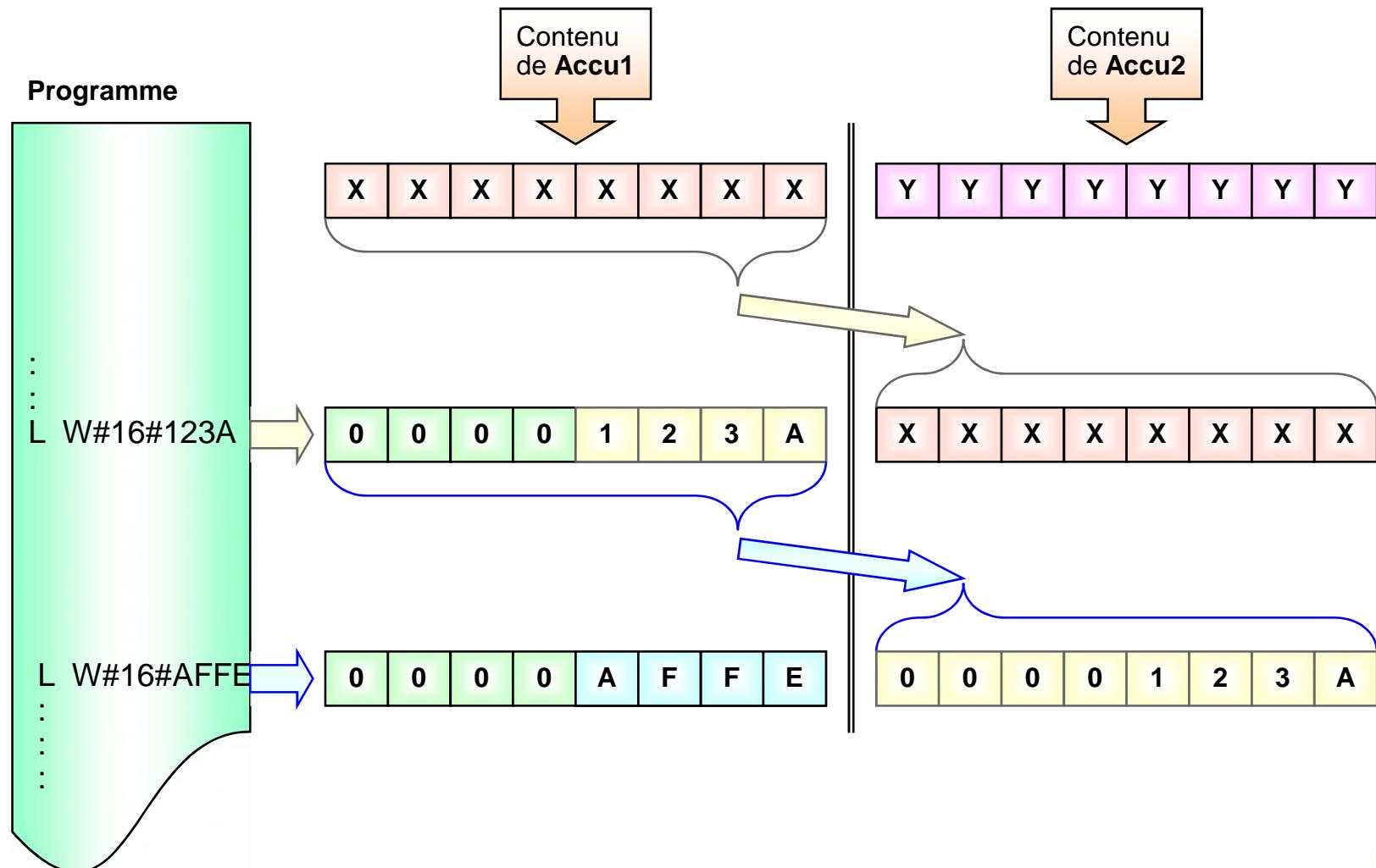


Exemples de
chargement

L +5	// Constante 16 bits (Integer)
L L#523123	// Constante 32 bits (Long integer)
L W#16#EF	// Octet hexadécimal
L 2#0010 0110 1110 0011	// Valeur binaire 16 bits
L 3.14	// Constante 32 bits (Real)

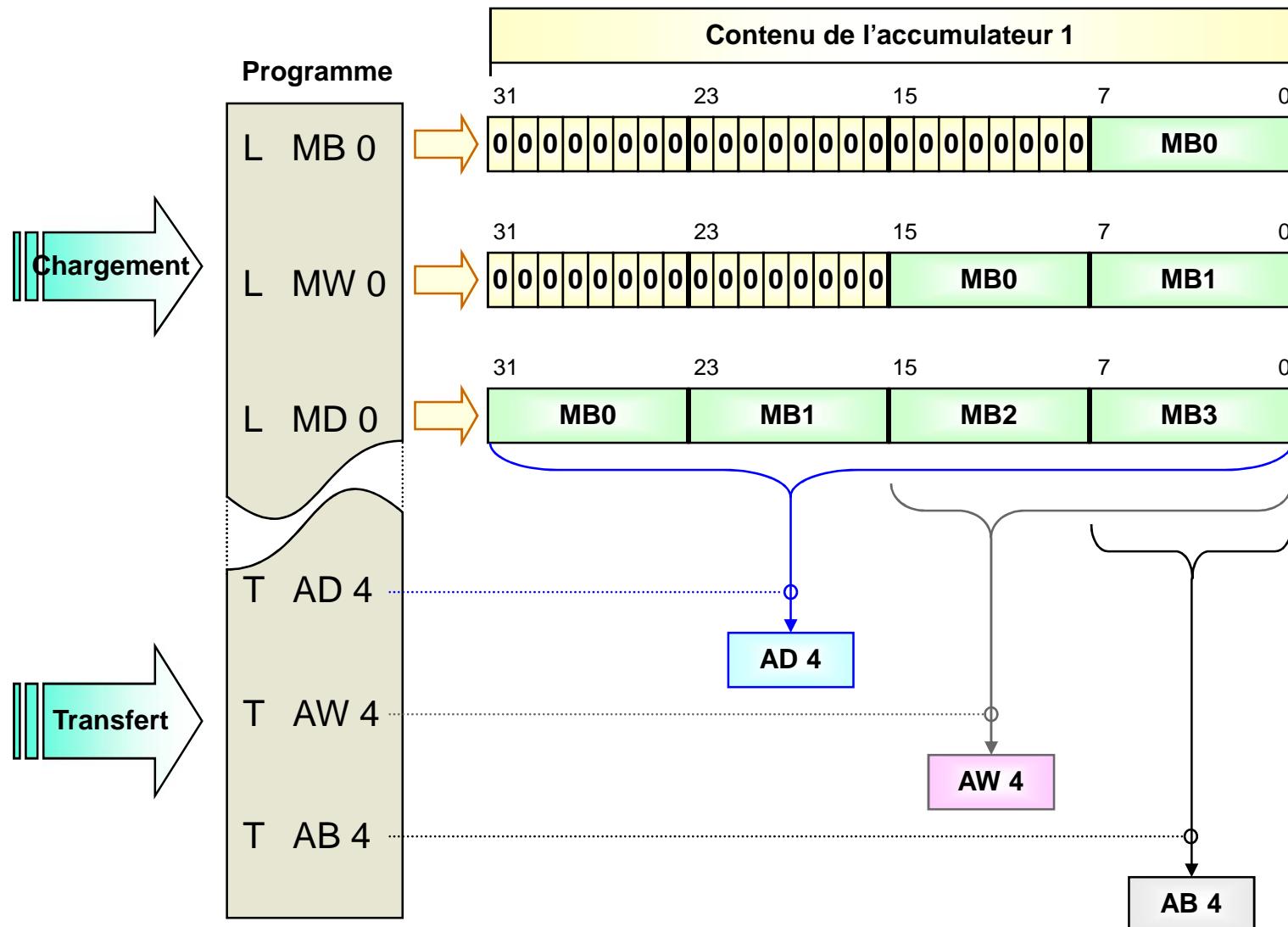


CHARGEMENT ET TRANSFERT DE DONNEES





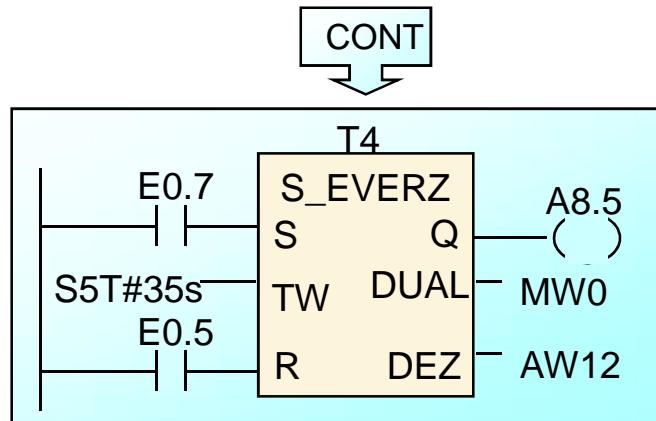
CHARGEMENT ET TRANSFERT DE DONNEES



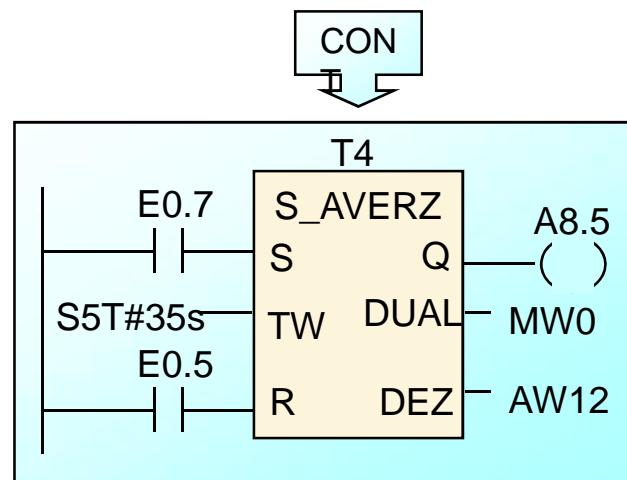


TEMPORISATION

TEMPORISATION SOUS FORME DE RETARD A LA MONTEE (SE)

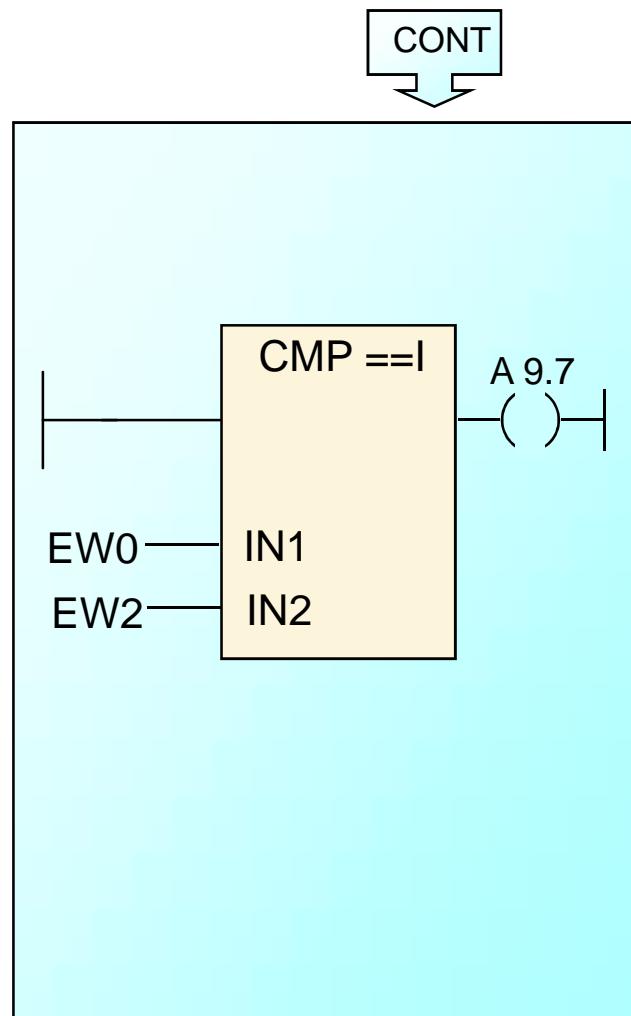


TEMPORISATION SOUS FORME DE RETARD A LA RETOMBEE (SE)





OPERATION DE COMPARAISON





OPERATION ARITHMETIQUE DE BASE

