







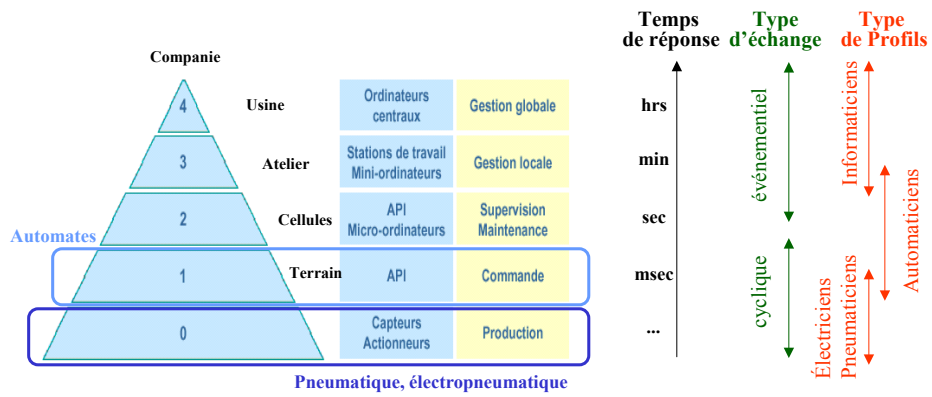
Automates programmables

1: Technologie et Langage Ladder

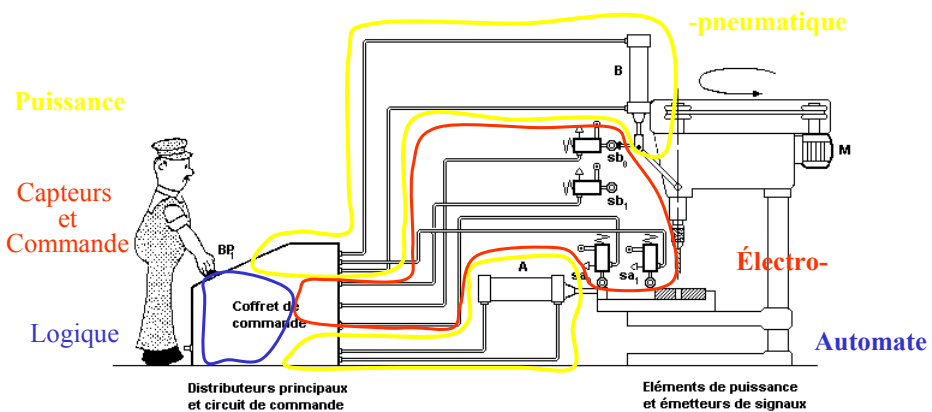
Contenu du cours

-  **Introduction / Applications**
-  **Technologie et Principes**
-  **Autour de l'automate**
-  **Objets adressables**
-  **Programmation en langage à contacts**
-  **Exemples**

Automates et Systèmes Automatisés

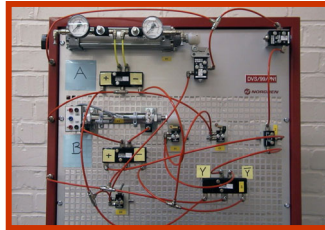


Forage et clamage automatisés

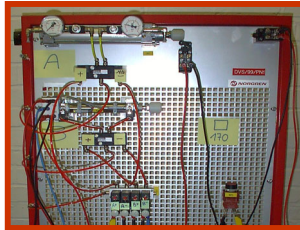


$$\text{Cycle : } \begin{pmatrix} BP_1 \\ sa_0 \end{pmatrix} A + sa_1 B + sb_1 B - sb_0 A -$$

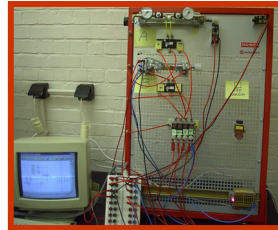
Logique câblée vers programmée



Puissance: pneumatique
Commande: pneumatique
Logique : pneumatique



Puissance: pneumatique
Commande: électrique
Logique : électrique



Puissance: pneumatique
Commande: électrique
Logique : programmée

max. 2 ... 3 cylindres
max. 2 mémoires (5/2 ou relais)

• PC
• PLC

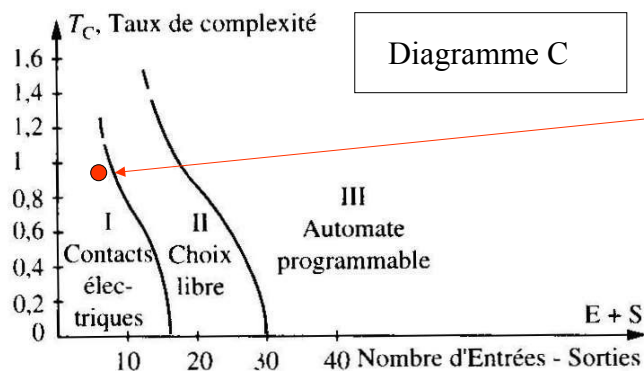
Logique Electropneumatique ou Automate

Machine à actionneurs
pneumatiques prioritaires

E : nombre d'entrées

S : nombre de sorties

$T_C = (\# \text{ étapes} + \# \text{ séquences}) / (E + S)$



« Hardware » disponible

1. Micro-contrôleur / DSP

- prix très attractif
- adapté aux applications de série (pas pour la production)
- temps donc coût de développement importants
- programmation (re-programmation) plus complexe

2. Automate programmable («PLC»)

- plus coûteux (> 750 Euros)
- programmation (re-) simple
- très vaste gamme d'interfaces (PLC/capteurs, PLC/actionneurs, PLC/opérateur)
- Très robuste : adapté au milieu industriel, gestion de bâtiments, voies de communications, ...

évolution
○○○

· unité centrale PC
· carte graphique
· carte réseau ...

3. Micro-ordinateur («PC»)

- Stockage et traitement de bcp d'informations (registres, mémoire, disque,...)
- Hardware peu robuste
- Software peu fiable

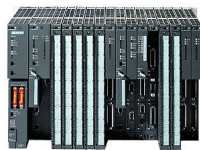
évolution
○○○

cartes i/o , cartes automatismes
châssis industriel

Introduction

PLC: Les grandes marques

Siemens



Mitsubishi



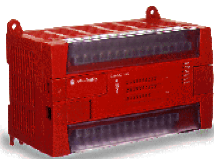
Télémécanique
(Schneider)



Jetter



Allen-Bradley
(Rockwell Automation)



Omron



Introduction

Quelques applications ...

Assemblage carrosserie Peugeot 206 (PSA)



- 13 robots de soudure/assemblage
- 30 automates Premium/Micro

Peinture carrosserie Citroën Picasso



- 5 étapes (bains décapage, anti-corrosion, ...), 12 teintes
- 400 transporteurs à rouleaux
- 60 automates Premium + moniteurs

Tunnel de Cointe (Liège liaison E25/E40)



- signalisation/statistique/éclairage...
- > 20 automates

Convoyeur Laboratoire

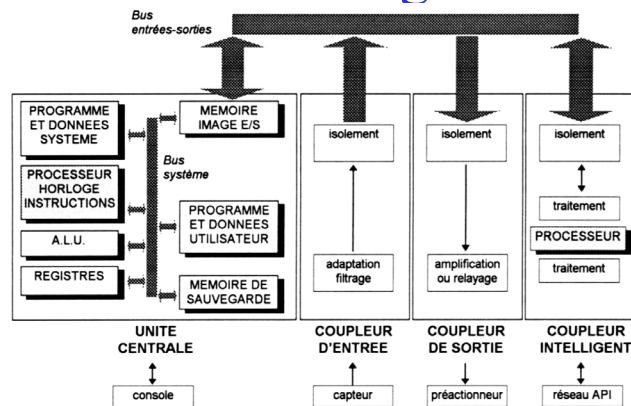


- 1 robot «pick & place»
- ... 20 actionneurs
- ... 12 capteurs
- 1 automate Micro

Contenu du cours

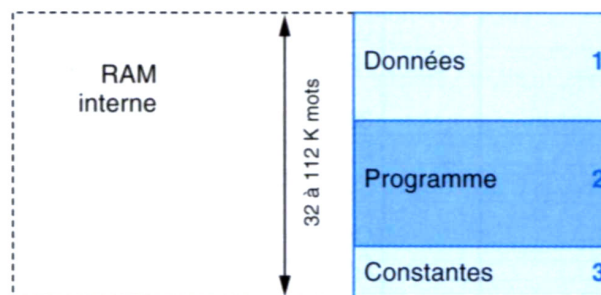
- ✍ **Introduction / Applications**
- ✍ **Technologie et Principes**
- ✍ **Autour de l'automate**
- ✍ **Objets adressables**
- ✍ **Programmation en langage à contacts**
- ✍ **Exemples**

Architecture générale



- Mémoire programme et données utilisateur (RAM) + mémoire E(E)PROM → 20 ... 200 Kbyte
- Mémoire programme système (ROM)
- Registres spécialisés paramétrables (tempos, compteurs, programmeurs, ...)
- Unité d'arithmétique et logique (ALU) → Performances d'un PC 486 !
- Mémoires image des entrées et des sorties
- Bus entrée-sortie / système
- Coupleurs entrée / sortie

Ex.: Mémoire d'un Tsx Premium



32 Kmots* répartis en:

- 7 Kmots de données applications
- 24 Kmots de programme, de constante et de données systèmes

=> RAM interne sauvegardée par pile (3 ans d'autonomie)

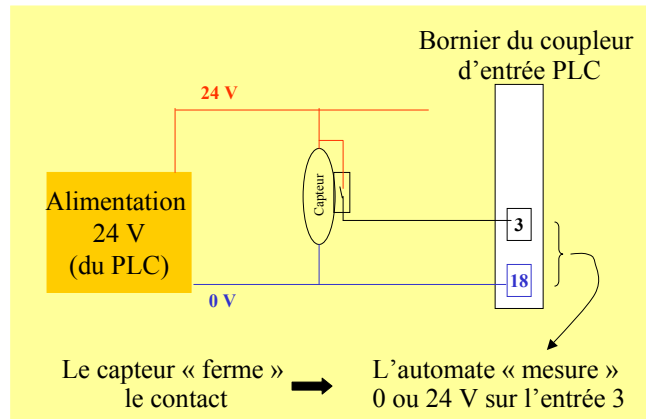
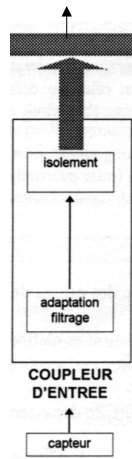
=> Extension mémoire possible via carte mémoire PCMCIA

* Mots = 2 octets (16 bits)

Entrées et Coupleur d'entrée

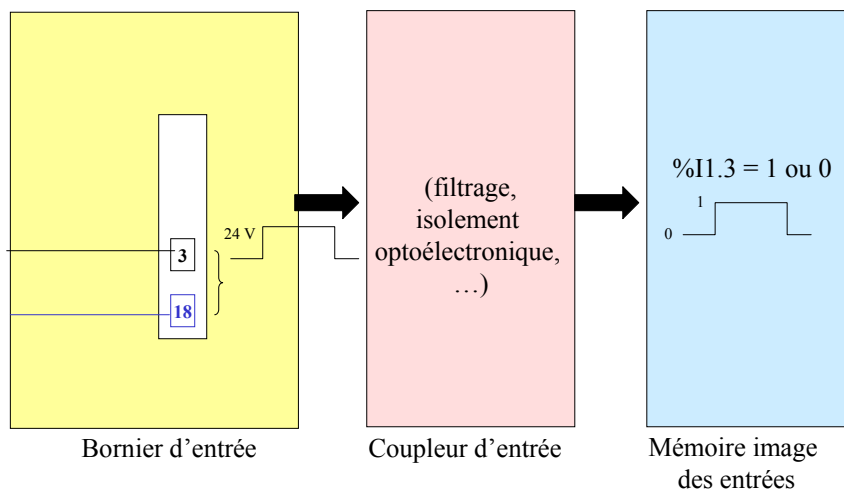
1. « Connectique »

vers mémoire image



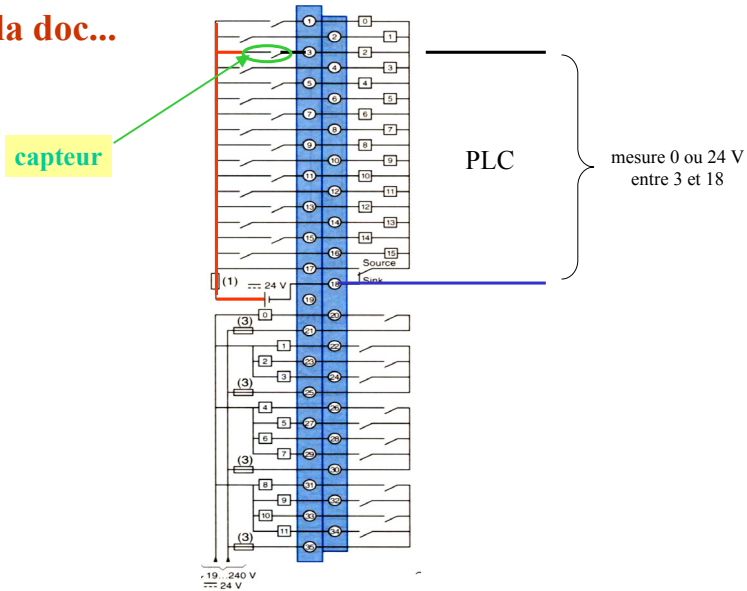
Entrées et Coupleur d'entrée

2. Mémoire image



Entrées et Coupleur d'entrée

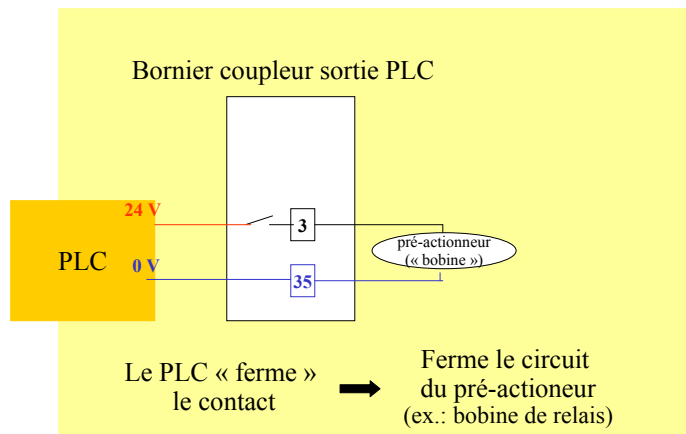
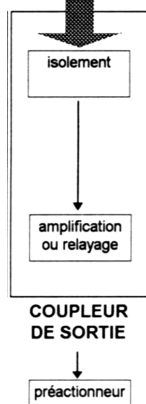
3. Dans la doc...



Sorties et Coupleur de sorties

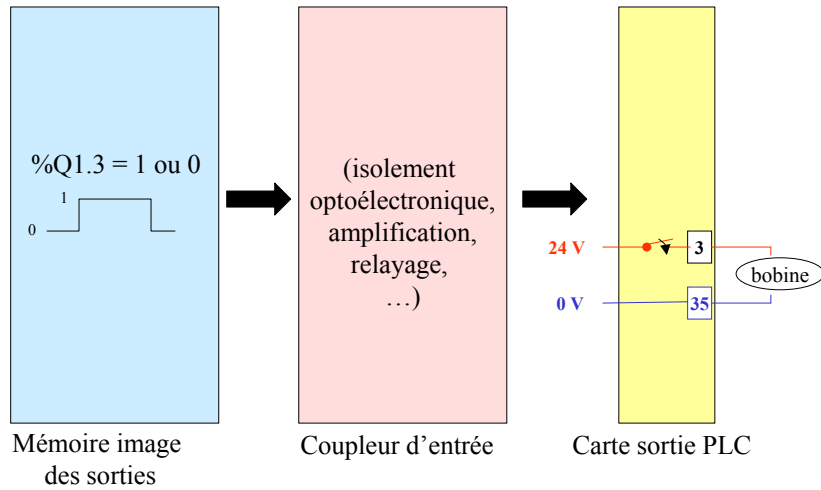
« from » mémoire image

1. « Connectique »



Sorties et Coupleur de sorties

2. Mémoire image

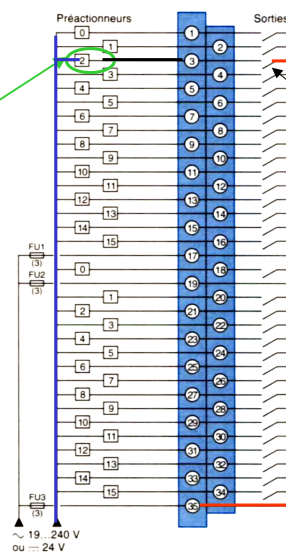


Sorties et Coupleur de sorties

3. Dans la doc...

préactionneur

TSX DSZ 32R5



PLC

« ferme » le contact 3

Principe

Principe de base

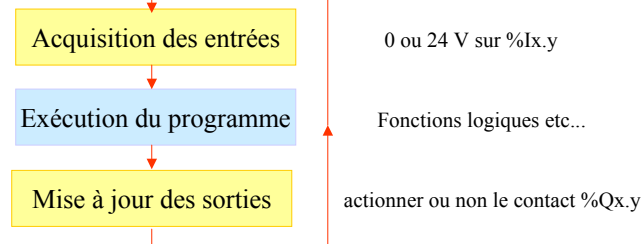
Rôle du programme

La RAM contient le programme utilisateur (liste d'instructions).
Le programme est relu cycliquement de façon ininterrompue*.

Deux modes de fonctionnement:

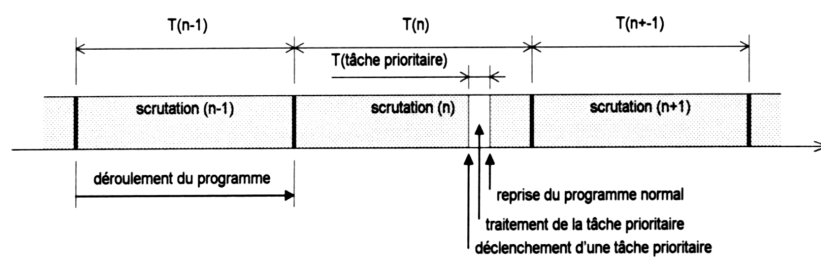
- **Scrutation cyclique** = mode par défaut de l'automate
- **Scrutation périodique** (période fixe) : sur certains modèles

... en gros ...:



* sauf bien entendu lors d'un arrêt automate (provoqué ou non)

Durée de cycles



Ordres de grandeur

Instruction : nano => microsecondes

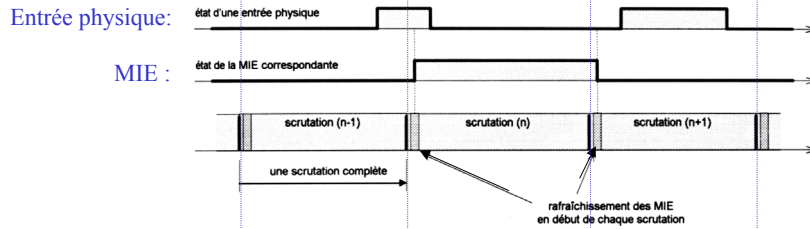
Scrutation : $x * 10$ millisecondes

Tâche prioritaire : $x * 10$ millisecondes

Scrutation et mémoire entrées

Problème:

L'apparition et la disparition des signaux délivrés par les capteurs ont lieu indépendamment de l'automate: les états des entrées peuvent « basculer » à tout instant.



Principe:

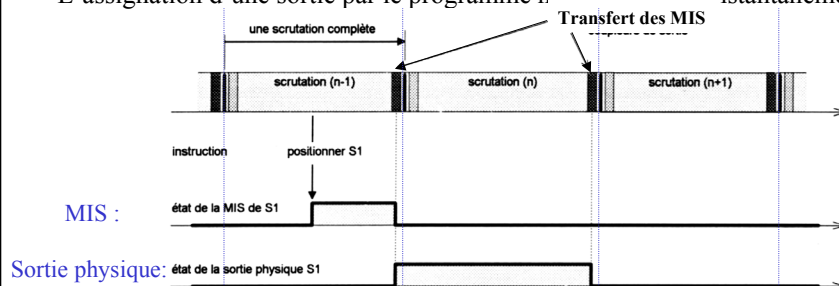
- En début de scrutation, les états des entrées physiques vues par les coupleurs d'entrée/sortie sont **mémorisés** dans la mémoire image des entrées (MIE).
- Pendant l'exécution du programme, toute instruction qui requiert l'état d'une entrée provoque en fait la lecture de l'**image « figée »** de cette entrée.
Image = bit interne « %Ix.y »
- Les MIE sont rafraîchies en **début** de chaque nouvelle scrutation.

Principe

Scrutation et mémoire sorties

Problème:

L'assignation d'une sortie par le programme ne peut se faire instantanément.



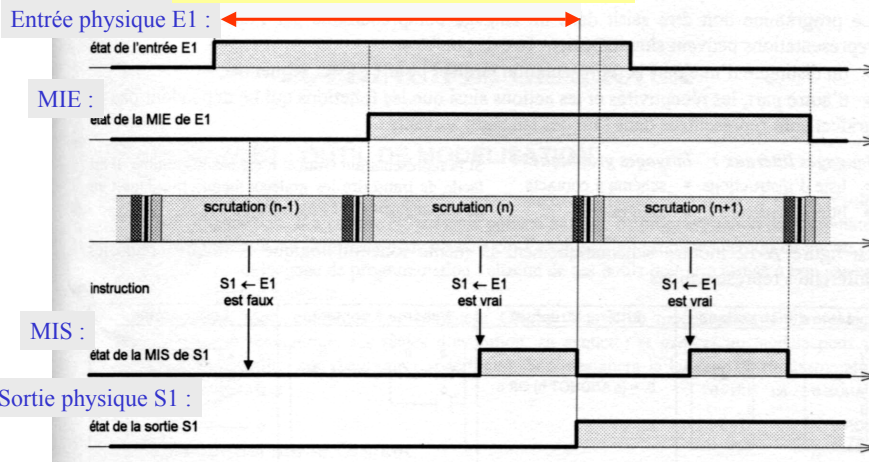
Principe:

- Chaque instruction du programme qui positionne une sortie, positionne en fait le bit interne (« Qx.y ») **image de la sortie** en question: sortie réelle non affectée.
- Les MIS sont rafraîchies au fur et à mesure de l'exécution du programme.
- A la **fin** de la scrutation, les états des MIS sont **transférés** vers les sorties via les coupleurs, qui maintiennent ces sorties jusqu'à la fin du cycle suivant.

Temps de réaction

Instruction $S1 := E1$

délai de réalisation de l'instruction ($x \cdot 10$ milliseec.)



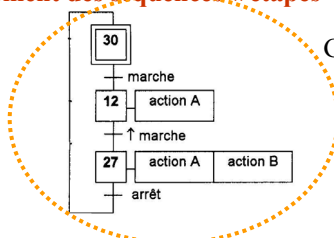
Langages de programmation

1. Encodage des étapes et des transitions (bas niveau)

80 % des applications actuelles (Télémécanique)

| Liste d'instructions | Littéral structuré | Schéma à contacts | |
|---|---|---|---|
| $\{$ LD a ANDN b) OR c ST S ou $\{$ U - a UN - b) O - c = - S | $S := (a \text{ ANDNOT } b) \text{ OR } c$ permet les structures habituelles des langages informatiques : IF, CASE, FOR, WHILE, REPEAT | très courant, facile à mettre en oeuvre grâce à sa ressemblance avec les schémas électriques. | permet de programmer rapidement des fonctions complexes par paramétrage de blocs fonctionnels |
| type « assembleur » | type « C » | type « électrique » | type « simulink » |
| Noms des langages équivalents dans la norme CEI 61131-3 (SFC, issu du Grafcet, est réservé aux fonctions séquentielles) | | | |
| IL | ST | LD | FBD |

2. Enchaînement des séquences « étapes - transitions » (haut niveau)



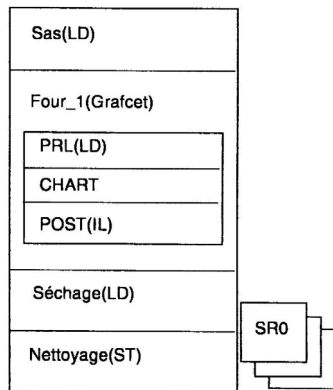
Grafcet (« SFC ») ou langage ci-dessus

Application monotâche

Tâche maître

- Cyclique ou périodique
- Seule à être programmable en Grafcet

MAST



Une tâche maître* comprend:

- 1 ou plusieurs sections (entité autonome)
- et des sous-programmes (appel dans sections)

Exemple:

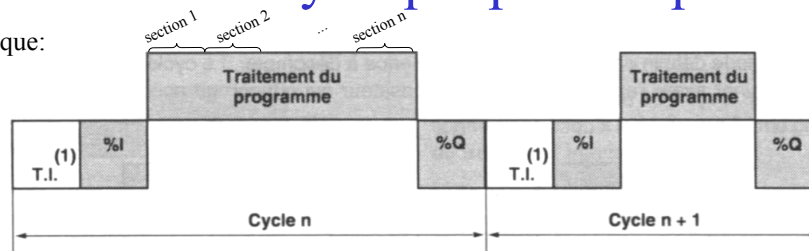
- une section « Sas » en Ladder
- une section « Four1 » en Grafcet
- une section séchage en ladder
- une section « Nettoyage » en littéral
- des sous-programmes SR0, SR1, ...

Les sections sont scrutées dans l'ordre d'apparition du navigateur

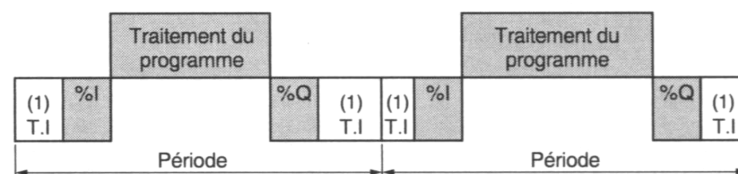
(*) Télémécanique

Exécution cyclique/périodique

Cyclique:



Périodique (de 1 à 255ms)



Bits et mots système « utiles »:

%S19: débordement de période
%SW33: temps exécution dernier cycle
%SW34: temps du cycle le plus long
%SW35: temps du cycle le plus court

Application multitâche

Tâche maître

- Toujours présente, cyclique ou périodique
- Seule à être programmable en Grafcet
- la moins prioritaire

Tâche rapide

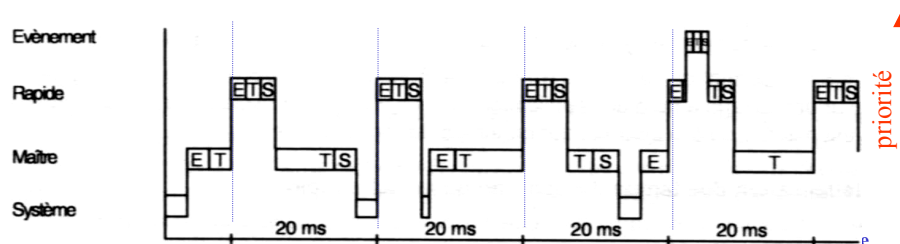
- Optionnelle, toujours périodique
- Traitements courts pour ne pas pénaliser la tâche maître (ex. 20 msec)
(tâches de surveillance; ex.: alarme température four, ...)

Tâche événementielle («action réflexe rapide»)

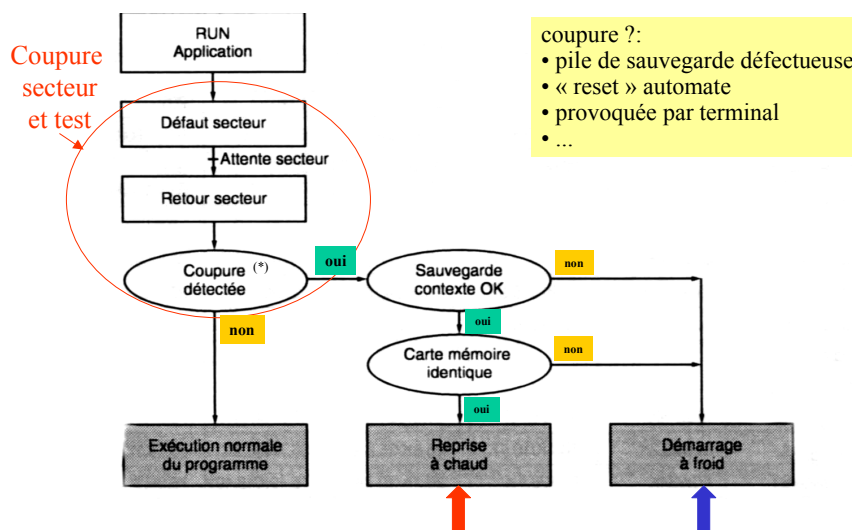
- action très rapide
- sur une entrée ou une sortie
- la plus prioritaire

Exemple:

Détection de seuils d'une entrée comptage =>
action = mise à 1 d'une sortie TOR.

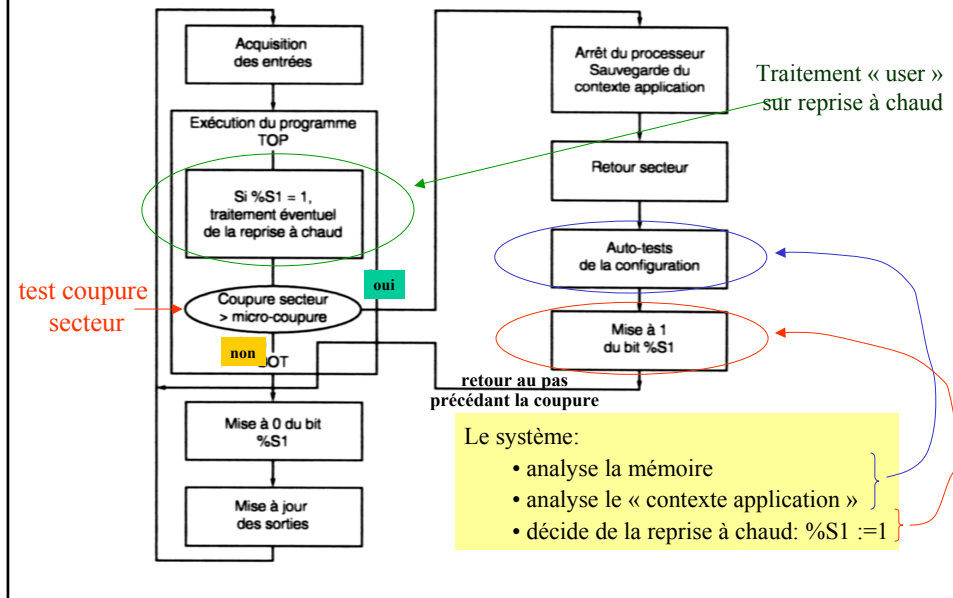


Traitement sur coupure secteur

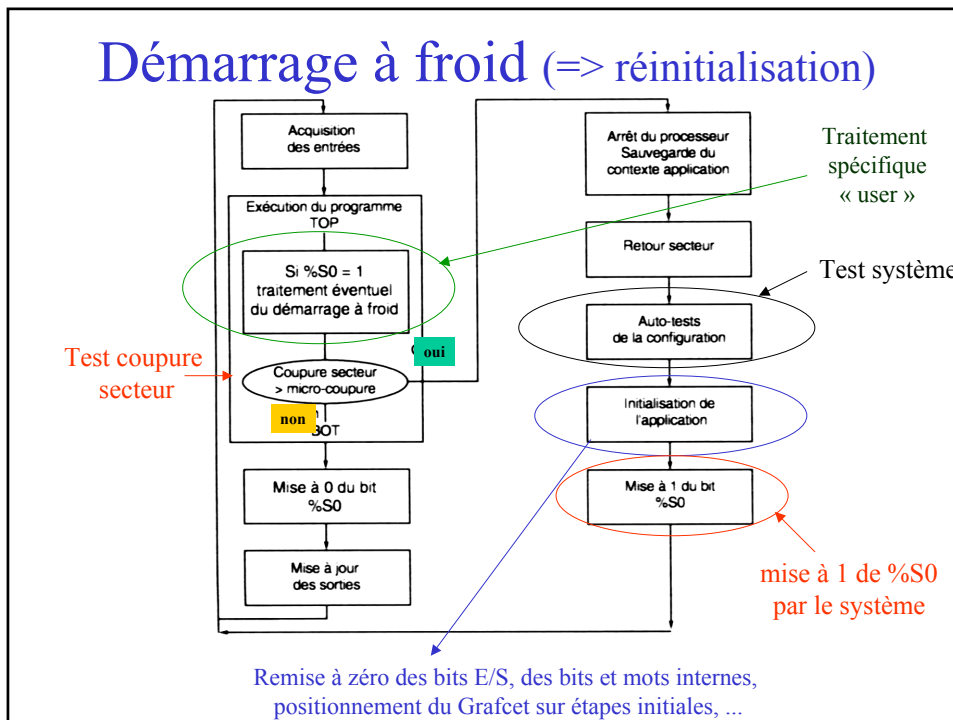


(*) si durée > temps filtrage de l'alim (1 msec en DC, 10 msec en AC)

Reprise à chaud (=> pas de réinitialisation)

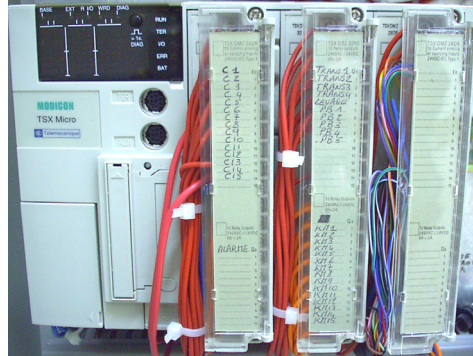


Démarrage à froid (=> réinitialisation)



Un automate «type »

Télémécanique Micro TSX3721
 (« pour automatismes de complexité faible/moyenne »)



- RAM : 20 K mots (programme, données, constantes)
- Nombre d 'E/S «TOR» : base : 192; base + extension : 256
- Temps d'exécution par K-instructions booléennes: 0.15 ms
- 2 extensions possibles PCMCIA (extension mémoire et communication)
- Fonctions intégrées: horodateurs, temporisateurs et compteurs

Un automate «type »

TSX Micro 3721

bac de 3 emplacements
pour modules e/s TOR

Bloc de visualisation

- run: run / stop
- ter: échange terminal
- i/o: défaut i/o
- err: erreur CPU
- bat: pile de sauvegarde

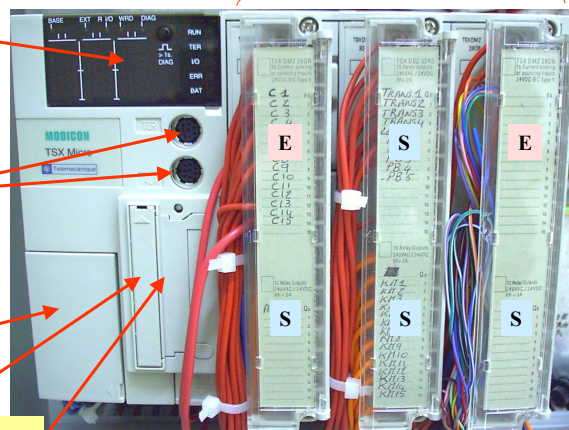
Prise TER et AUX pour:

- Terminal de programmation
- connection à un autre automate
- bus pour capteurs spécifiques
- imprimant ou écran de contrôle

Alimentation (24V)

Emplacements :

- carte extension mémoire (PCMCIA)
- coupleur de communication (PCMCIA)



TSX DMZ 28 DR

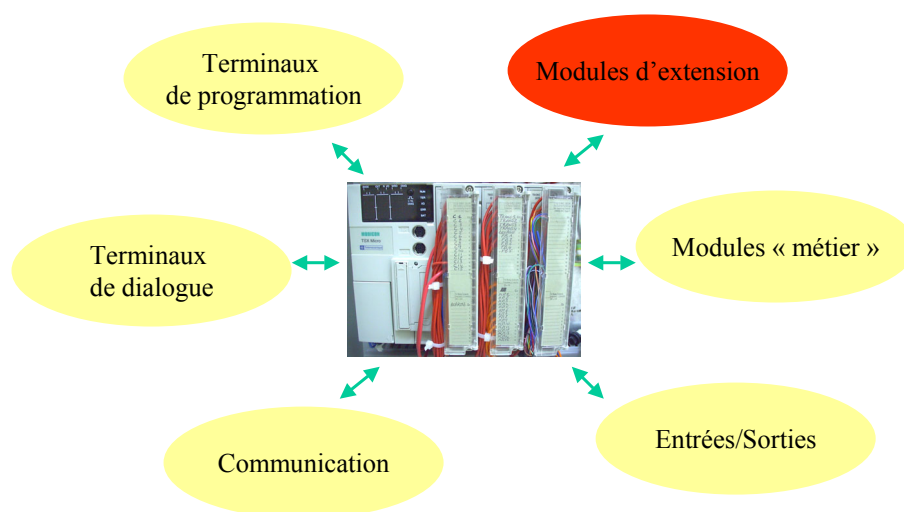
TSX DMZ 28 DR

TSX DSZ 32 R5

Contenu du cours

- ✍ Introduction / Applications
- ✍ Technologie et Principes
- ✍ **Autour de l'automate**
- ✍ Objets adressables
- ✍ Programmation en langage à contacts
- ✍ Exemples

Autour de l'automate



Modules d'extension

Modules d'entrées sortie TOR

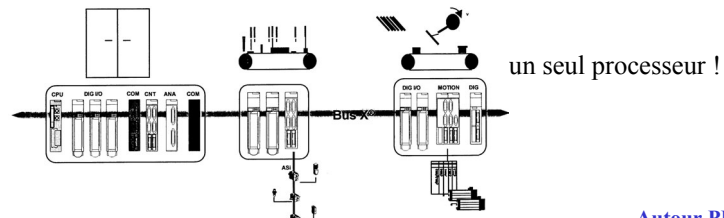


Variantes :

- E, S ou E/S
- 24V DC, 220V AC
- ampérage relais de sortie
- logique positive/négative
- niveau de protection E/S
- ...

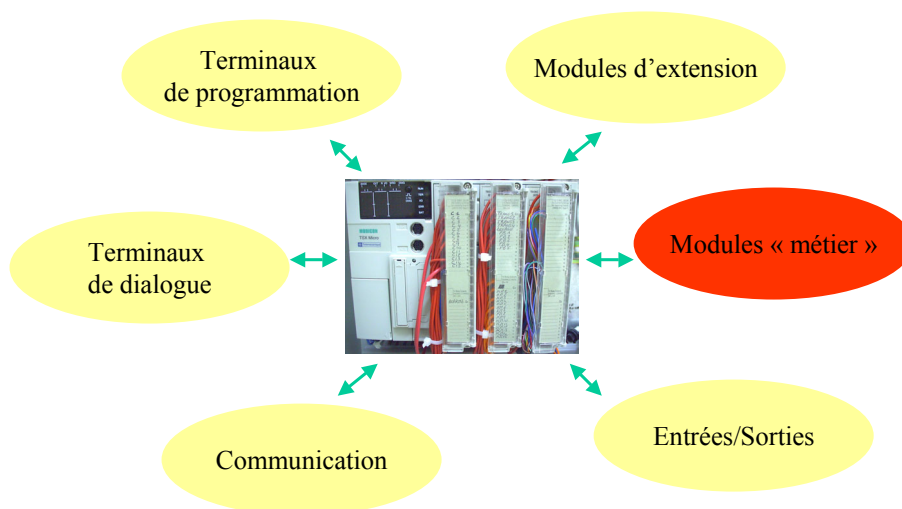
Modules d'entrées sortie TOR déportée

Permet d'étendre le rack de l'automate «autour de la machine»



Autour PLC

Autour de l'automate

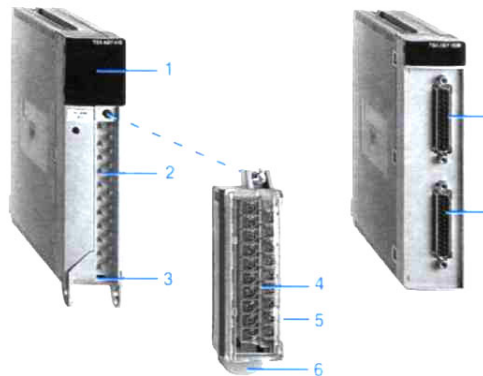


Modules « métiers »

Modules d'entrées sortie analogiques

Fonction : surveillance, mesure et régulation de process continus

- entrées : $\pm 0 \text{ V}$ / ADC (ex. : sur 12 ou 16 bits)
- sorties : $\pm 10 \text{ V}$; $0 \dots 20 \text{ mA}$ / DAC (ex. : sur 12 bits)



Modules « métiers »

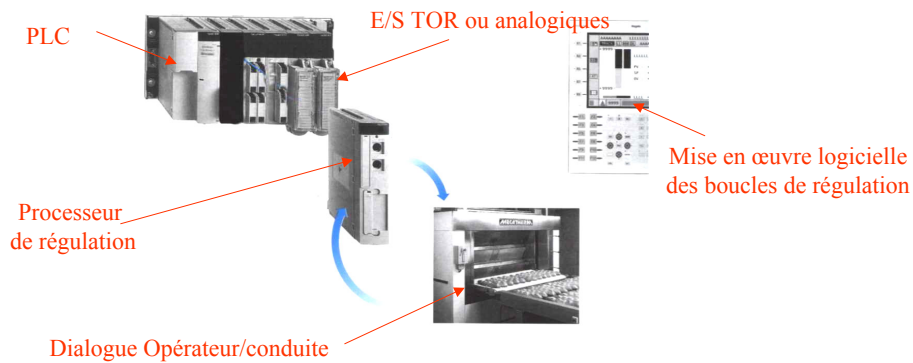
Modules de régulation de process

Fonction :

pilotage de process simples tels que fours, groupes frigorifiques
asservissement mécanique (couple, vitesse,...)

Intègrent :

- le calcul « floating point »
- des boucles de régulation prédéfinies et paramétrables
(Boucle cascade, PID, auto-adaptatifs, ...)

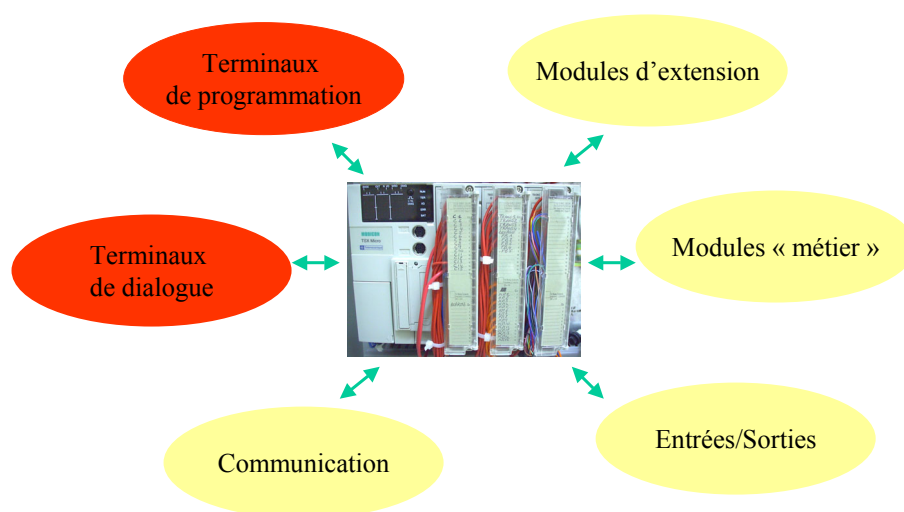


Modules « métiers »

"PLC Toolboxes"

- Module de pesage
- Module de commande d'axe
- Module d'alimentation process
- Module de comptage
- ...

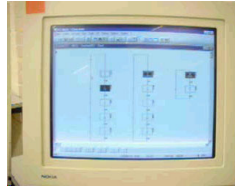
Autour de l'automate



Terminaux de programmation

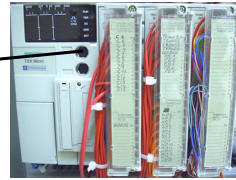
PC standard

Programme « PL7pro »
(Ladder, Grafset, ...)



chargement
application

récupération
application



Terminaux industriels



Terminal de programmation
(Atelier, Bureau d'études)

Console portable

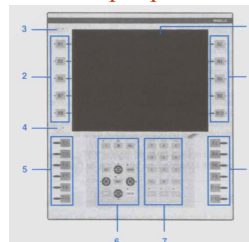


Réglage/Maintenance
(en Atelier)

Autour PLC

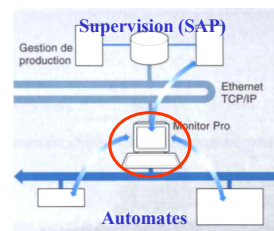
Terminaux de dialogue

Station Graphique



« Magelis »
(Télémécanique)

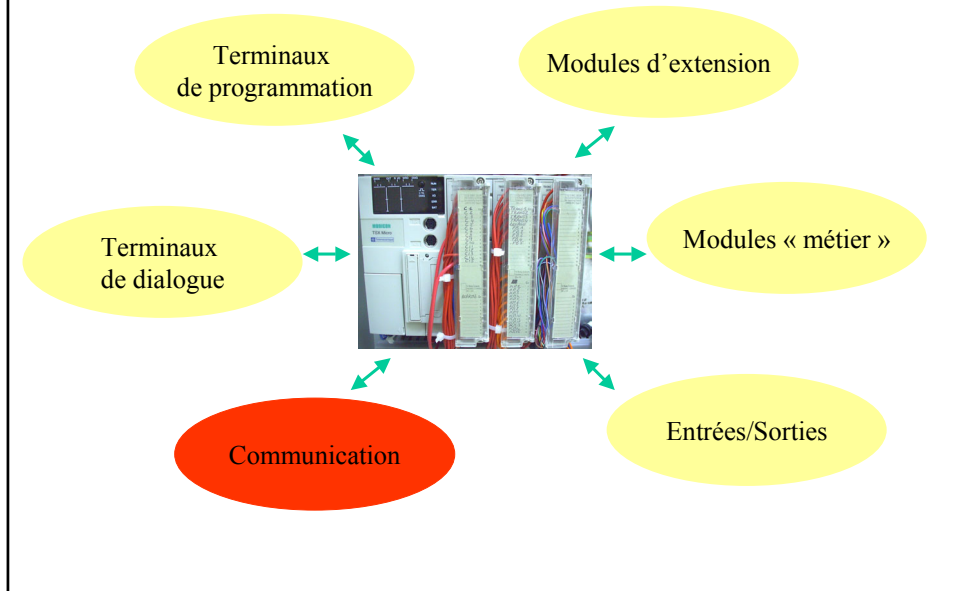
Logiciel de supervision



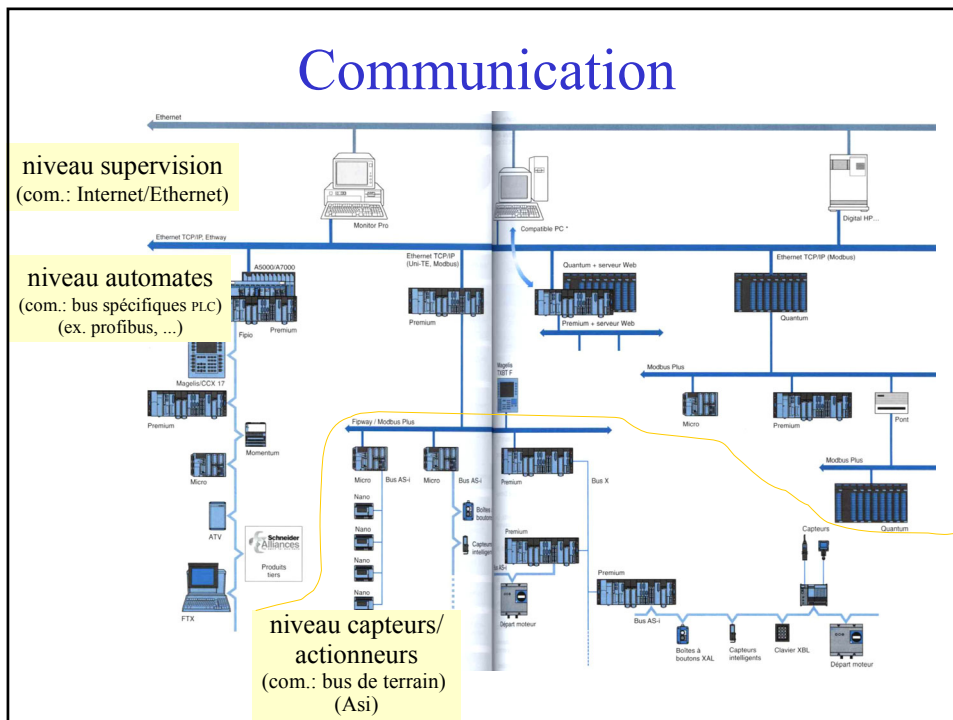
Fonctions:

- «Synopsis» de l'installation - visualisation temps réel des grandeurs
- Courbes de tendances « temps réel » (« Scope »)
- Chargement, déchargement, stockage et base de données applications
- Gestion des alarmes (transmission via réseau ...)
- Gestion du temps « absolu » (déclencher un processus les lundis à 8h45)
- Interface entre automates et bases de données et système SAP

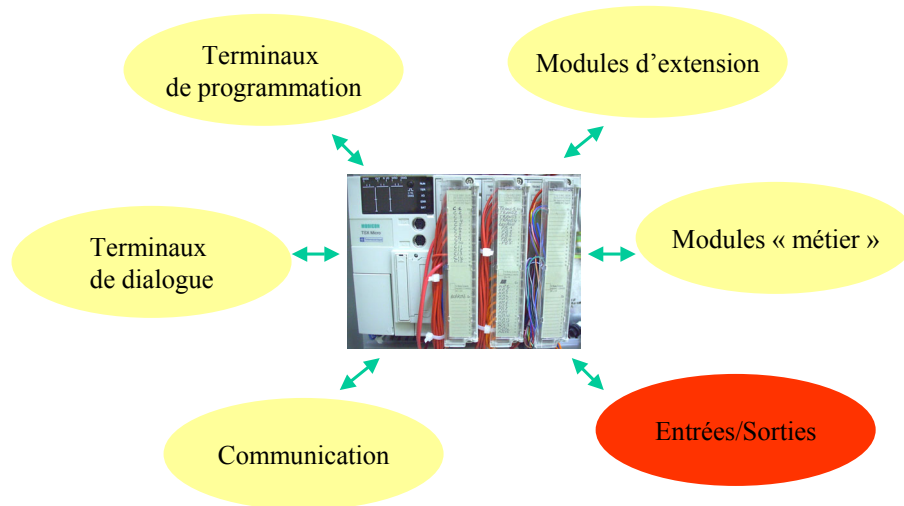
Autour de l'automate



Communication

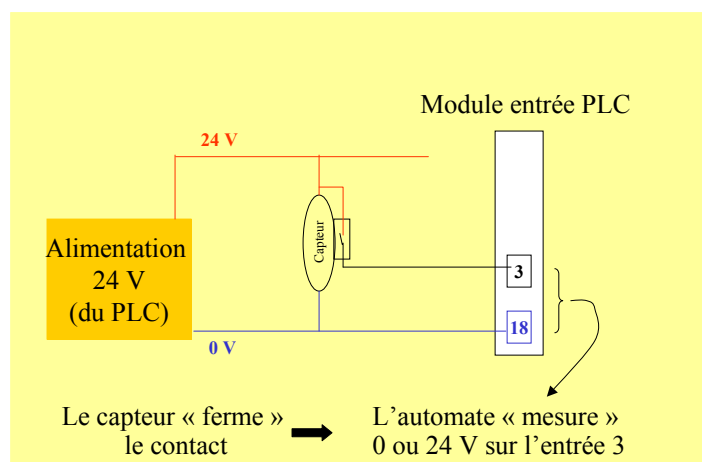


Autour de l'automate



Entrées (TOR)

Rappel



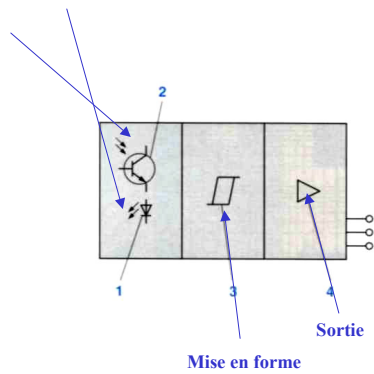
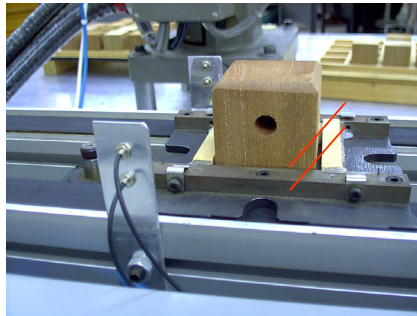
Détecteur photoélectrique (TOR)

Principe:

- émetteur de lumière (diode électroluminescente)
lumière rouge, verte (spectre visible) ou infrarouge (invisible)
- récepteur sensible (phototransistor)

Exemple

Détection défaut cube



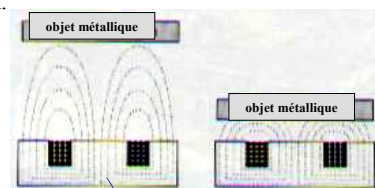
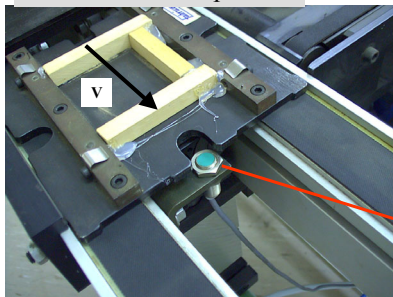
Détecteur de proximité inductif (TOR)

Principe:

- Un courant **oscillant** dans un bobinage induit un champ magnétique (voir figure) dans l'armature. L'inductance du circuit L se voit fortement modifiée lors de la présence de l'objet métallique, provoquant un amortissement des **oscillations**: cet amortissement qui est détecté (*) et mis en forme pour activer la sortie TOR.
- Distance: de 0 à ... 60 mm

Exemple au labo:

Détection arrivée palettes



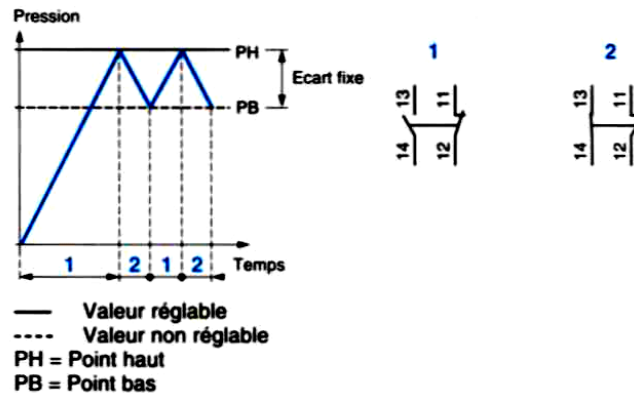
(*) seuils détectés à l'aide d'un Trigger de Schmidt

Pressostats, vacuostats (TOR)

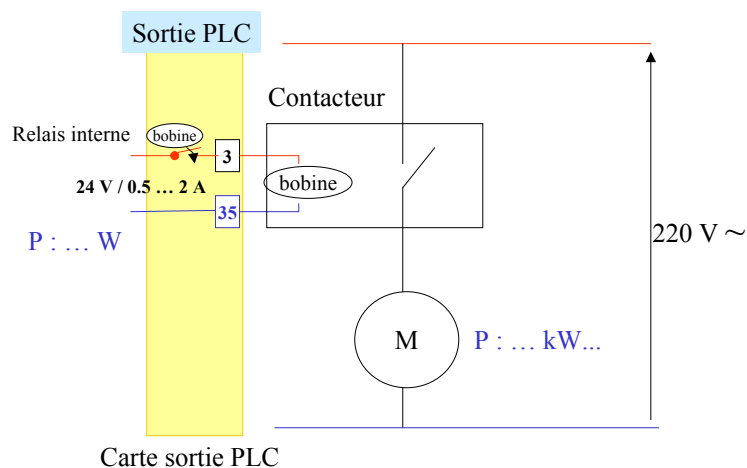
Principe:

- Transformation d'un signal de pression [ou dépression] en contact électrique Tout ou Rien (« TOR »)

Exemple: régulation entre deux seuils:

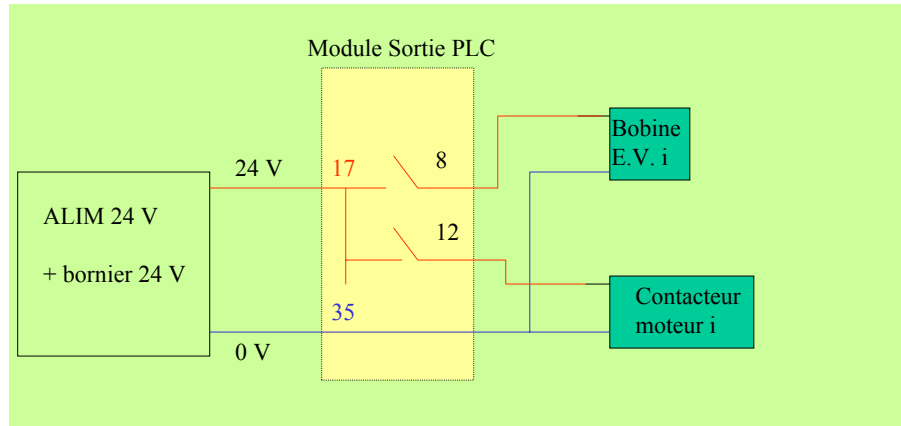


Sorties (TOR)

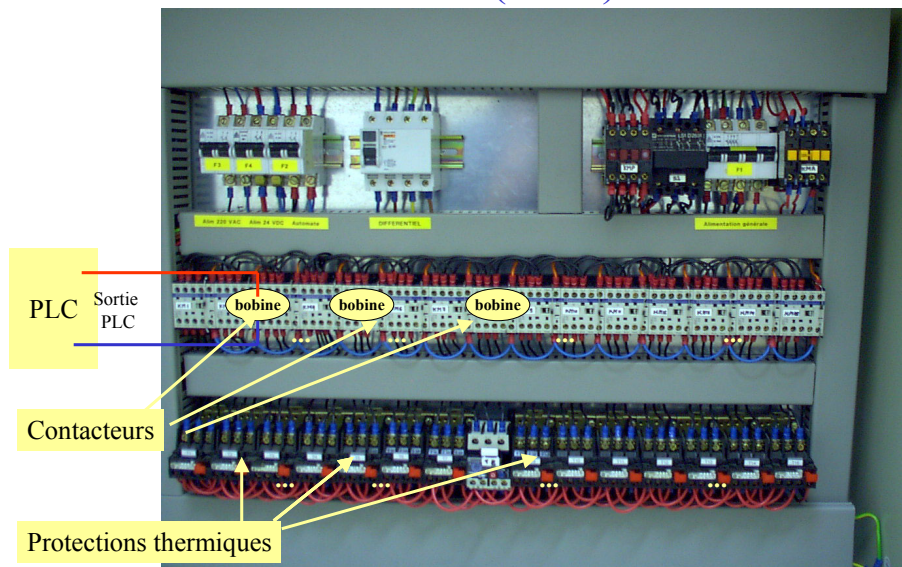


En principe, on ne connecte pas l'actionneur « (M) » directement sur le circuit de sortie ...
 Il est piloté par un relais, un contacteur de puissance.

Sorties (TOR)



Sorties (TOR)



Contenu du cours

- ✍ **Introduction / Applications**
- ✍ **Technologie et Principes**
- ✍ **Autour de l'automate**
- ✍ **Objets adressables**
- ✍ **Programmation Ladder**
- ✍ **Exemples**

Objets booléens : bits

Bits d'entrée/sortie : %I , %Q

- lecture (0 ou 1) de l'état d'une entrée % Ix.i
- écriture ou lecture de l'état d'une sortie % Qx.j

Bits internes : %M

- lecture/écriture (0 ou 1) d'un bit interne % Mi (mémoire)

Bits système : %S

- lecture/écriture (0 ou 1) d'un bit système (%Sj) (voir slide suivant)

Bits de blocs fonction

- lecture d'un bit d'un bloc
(ex. %Mni.R : état d'un bloc monostable)

Bits d'état des étapes Grafcet : %X

- lecture (0 ou 1) de l'état d'une étape Grafcet (%Xi pour l'étape i)
0: inactive ; 1 : active

Objets adress.

Bits système: « %Si »

| Bit | Fonction | Etat initial | Gestion |
|---------------------|--|--------------|-----------|
| %S0 | 1 = démarrage à froid (reprise secteur avec perte des données) | 0 | S ou U->S |
| %S1 | 1 = reprise à chaud (reprise secteur sans perte de données) | 0 | S ou U->S |
| %S4,%S5, %S6,%S7 | Base de temps 10 ms, 100 ms, 1 s, 1 mn | - | S |
| %S8 | Test du câblage(Utilisable sur automate non configuré) | 1 | U |
| %S9 | 1 =passage en repli des sorties | 0 | U |
| %S10 | 0 =défaut entrées/sorties | 1 | S |
| %S11 | 1 =débordement chien de garde | 0 | S |
| %S13 | 1 = premier cycle après mise en RUN | - | S |
| %S15 | 1 =défaut chaîne de caractères | 0 | S->U |
| %S16 | 0 =défaut E/S tâche | 1 | S->U |
| %S17 | état du bit sorti, lors d'une opération de décalage | 0 | S->U |
| %S18 | 1 =débordement ou erreur arithmétique | 0 | S->U |
| %S19 | 1 =débordement de période tâche | 0 | S->U |
| %S20 | 1 =débordement d'index | 0 | S->U |
| %S21 | 1=initialisation Grafcet | 0 | S |
| %S22 | 1= désactivation Grafcet | 0 | S |
| %S23 | 1= Grafcet figé | 0 | S |
| %S24 | 1 = remise à 0 de macro-étapes en fonction de %SW22 à %SW25 | 0 | U->S |
| ...etc..... | | | |

U : géré par l'utilisateur

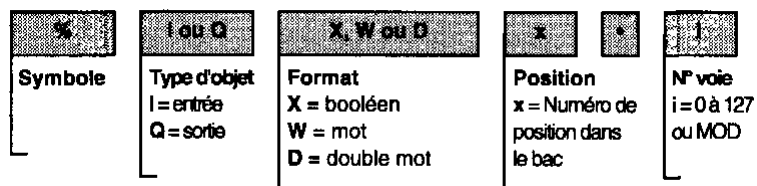
S : géré par le système

U->S : mise à 1 par U, mise à zéro par S

S->U : mise à 1 par S, mise à zéro par U

Adressage des E/S

Conventions d'adressage



.MOD : accès aux informations générale du module

.ERR : information de défaut de voie

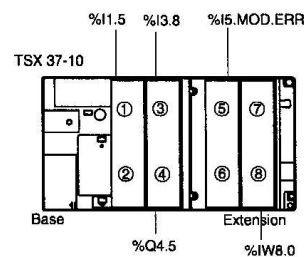
Exemple

%I1.5 : entrée 5 du module 1

%I3.8 : entrée 8 du module 3

%Q4.5 : sortie 5 du module 4

%I5.MOD.ERR : information défaut du module 5



Objets Mots : x octets

Conventions d'adressage

| % | M, K ou S | B, W, D ou F | I |
|---------|--|--|--------|
| Symbole | Type d'objet M = interne K = constant S = système | Format B = octet W = mot D = double mot F = flottant | Numéro |

Mots internes variables (%MW...)

pour mémoriser des valeurs en cours de programme

(W : 16 bits : -32768 à +32767 ; F : flottant 32 bits = simple précision)

Mots internes constants (%KW...)

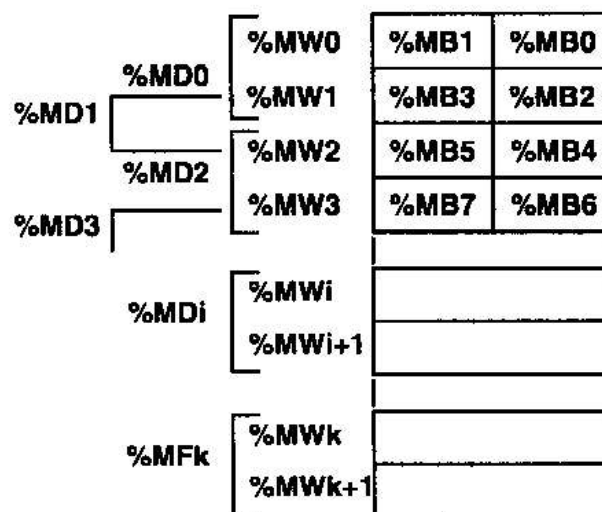
pour mémoriser des valeurs constantes : modifiables par terminal

Mots Systèmes (%SW...)

renseignent sur l'état du système (étapes actives, durée des tâches, ...)

fonctions utiles: horodateur, ...

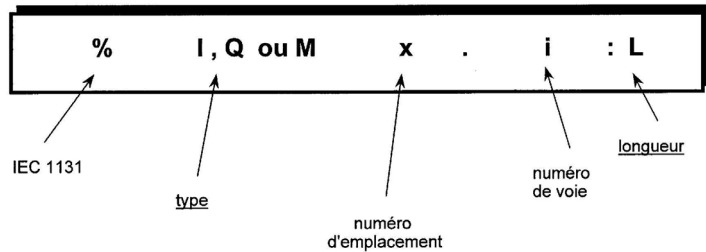
Objets Mots : recouvrement!



! no comment

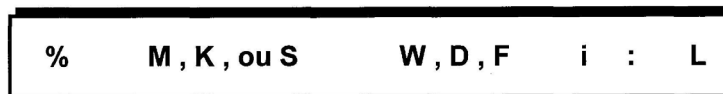
Tableaux

de bits :



exemple: %M10:6 désigne : [%M10 %M11 %M12 %M13 %M14 %M15]

de mots :



■ Exemples:

%KW10:5.....%MW20:3.....%MD14:6.....%SW50:4.....

Objets adress.

Adressage direct / indexé

Direct:

Adressage fixe et défini à l'intérieur du programme

exemple: %MW26 : mot interne d'adresse 26

Indexé:

INDEX = %MWi

Bits : %I1.0[%MW15] %Q2.0[%MW0] %M20[%MW3]

Mot interne simple ou double : %MW30[%MW2]

Mot constant simple ou double : %KW28[%MW25]

Mot flottant : %MF2 [%MW0]

Tableau de mots : %MW62[%MW2]:8

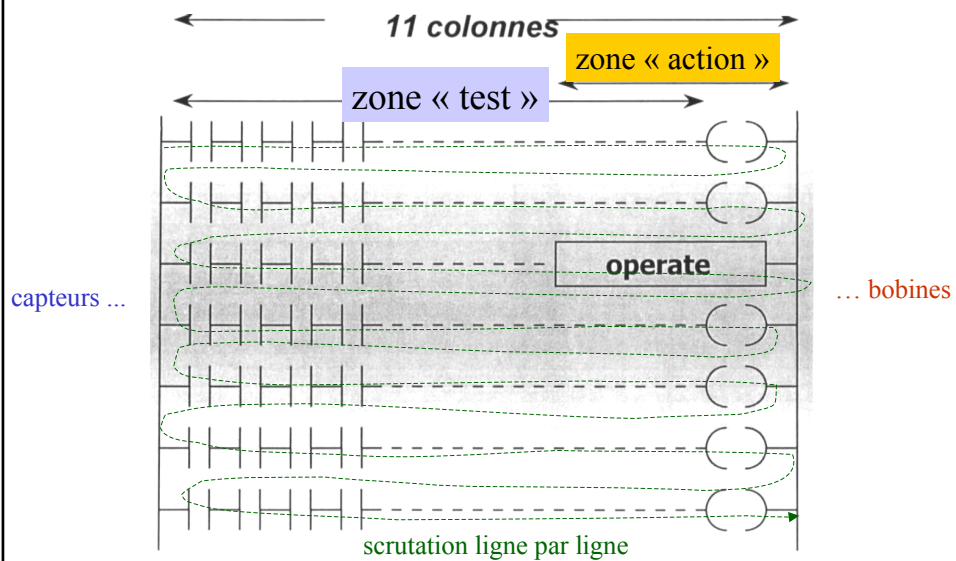
Attention : Indexation sur mot double : % MD6 [% MW0]

si % MW0=5 → l'adresse du mot sera : 6 + (2x5) = 16

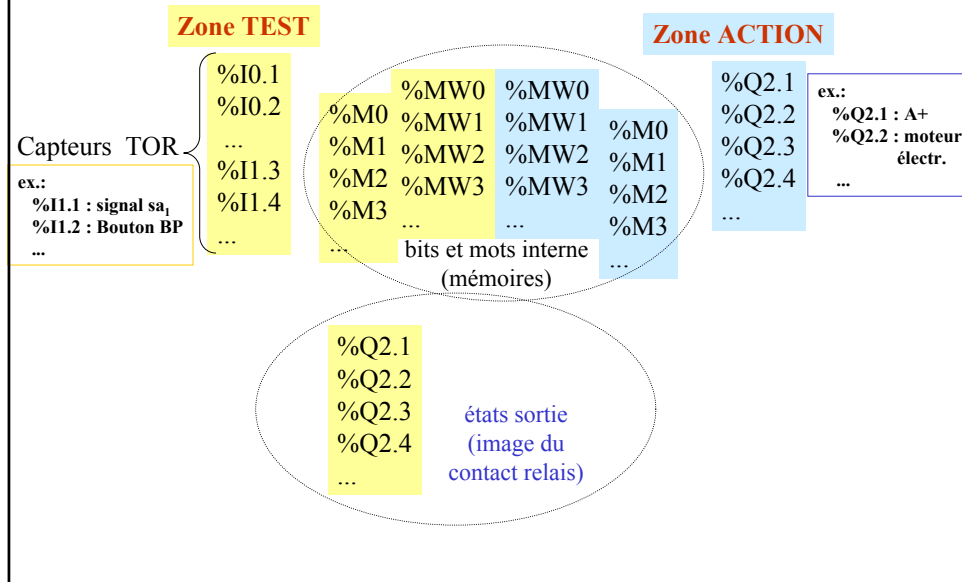
Contenu du cours

- ✍ Introduction / Applications
- ✍ Technologie et Principes
- ✍ Autour de l'automate
- ✍ Objets adressables
- ✍ **Programmation en langage à contacts**
- ✍ Exemples

Structure d'un réseau ladder



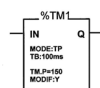
Elements pour les exercices



Elements du langage

Elements de test (zone « test »)

- | | - : Détection état 1 du bit entrée (« contact passant » si état 1)
- |/| - : Détection état 0 du bit entrée (« contact passant » si état 0)
- |P| - : Détection front montant (« contact passant » sur un cycle)
- |N| - : Détection front descendant (« contact passant » sur un cycle)



: blocs fonctions standards (ex. timers, compteurs, drums, ...)

Elements de liaison

- : Connexion horizontale
- | : Connexion verticale

Elements de base du langage (sur bits)

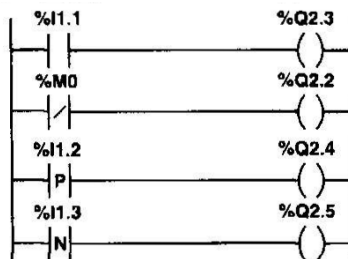
Elements d'action

- ()- : Ecrire l'état du test (0 ou 1) dans le bit
- (/)- : Ecrire l'état inverse du test (0 ou 1) dans le bit
- (s)- : Ecrire et mémoriser l'état 1 dans le bit si l'état du test vaut 1 (set)
- (R)- : Ecrire et mémoriser l'état 0 dans le bit si l'état du test vaut 1 (reset)
- (#)- : propre au Grafcet: provoque le passage à l'étape suivante
- (c)- : branchement à un sous-programme si test = 1
- <return> : retour de sous-programme si test = 1
- <halt> : arrêt du programme si test = 1

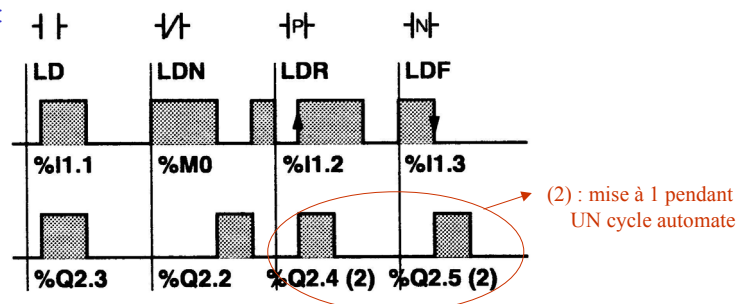
Fonctions de base (sur bits)

Instruction de chargement

Instructions:



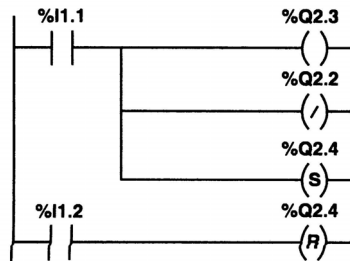
Chronogramme:



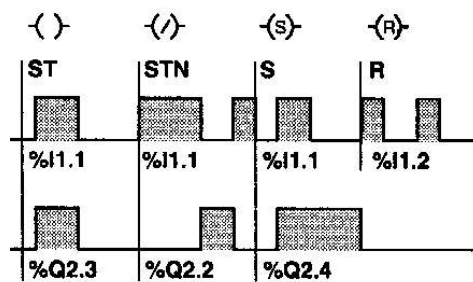
Fonctions de base (sur bits)

Instruction d'affectation

Instructions:



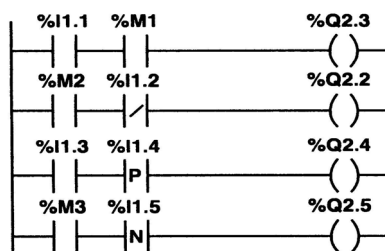
Chronogramme:



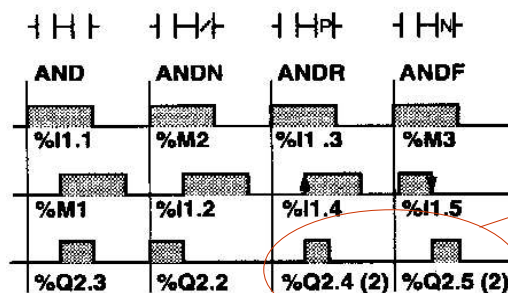
Fonctions de base (sur bits)

Instruction « ET »

Instructions:



Chronogramme:

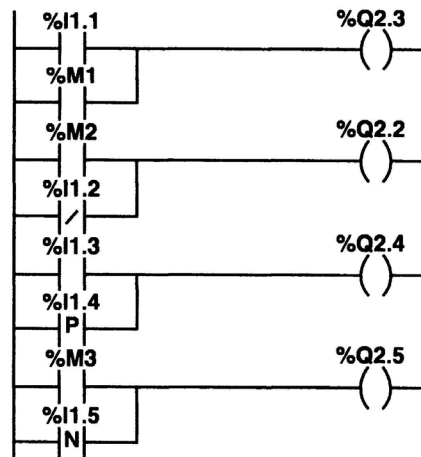


(2) : mise à 1 pendant
UN cycle automate

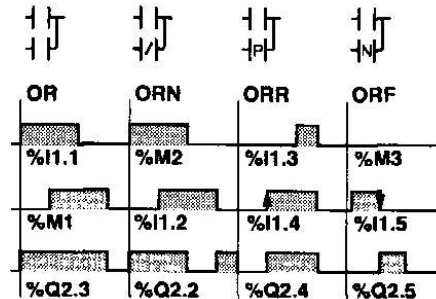
Fonctions de base (sur bits)

Instruction « OU »

Instructions:



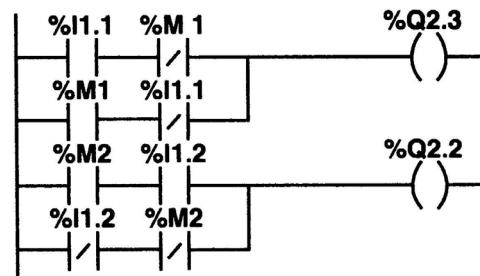
Chronogramme:



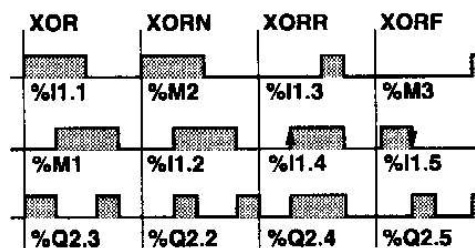
Fonctions de base (sur bits)

Instruction « OU exclusif » (XOR)

Instructions:



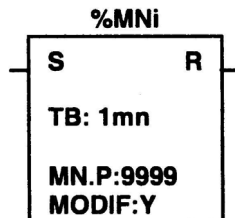
Chronogramme:



Blocs prédéfinis (sur bits)

Monostable

Instructions:



Valeurs :

%Mni.P : valeur de présélection (0 ... 9999)

%Mni.V : valeur courante: décroît de %Mni.P à 0

Y/N : %Mni.P modifiable ou non par terminal

Base de temps : TB

Sortie (« Running ») :

%Mni.R passe à 0 si %Mni.V=0

Armement (« Start ») :

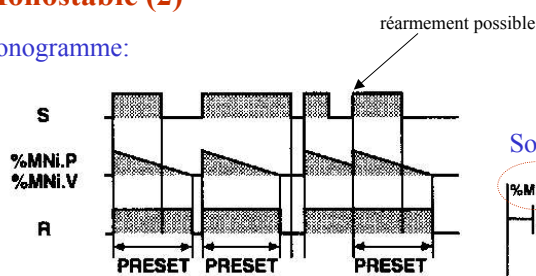
sur front montant de l'entrée S

Attention: S = détection interne front montant => OK sur instruction -| |-

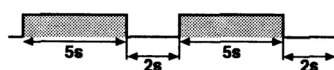
Blocs prédéfinis (sur bits)

Monostable (2)

Chronogramme:



Exemple:



Faire clignoter la lampe %Q3.0:

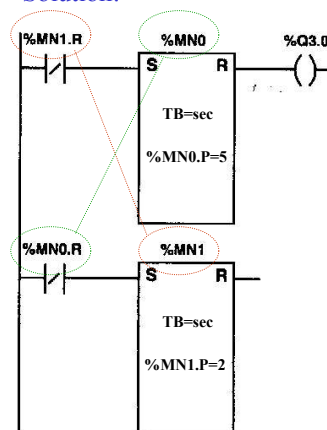
5 sec ON

2 sec OFF

5 sec ON

...

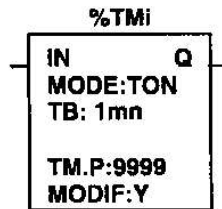
Solution:



Blocs prédéfinis (sur bits)

Temporisateur

Instructions:



Mode :

TON : tempo enclenchement
TOF : tempo déclenchement
TP : monostable

Valeurs :

%Tmi.V : valeur courante
%Tmi.P : valeur de présélection (0 ... 9999)
Y/N : Tmi.P modifiable ou non par terminal

Base de temps : TB

Armement (IN) :

sur front montant pour TON ou TP
sur front descendant pour TOF

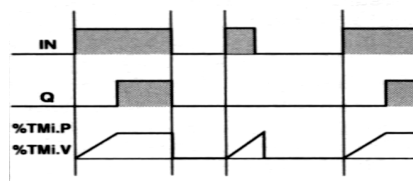
Détection interne d'un front en IN => instruction pilote « -| - » correcte

Blocs prédéfinis (sur bits)

Temporisateur (2)

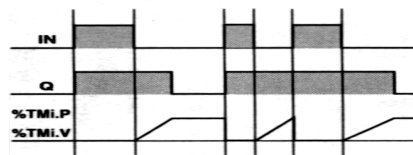
Mode TON

retard à l'enclenchement



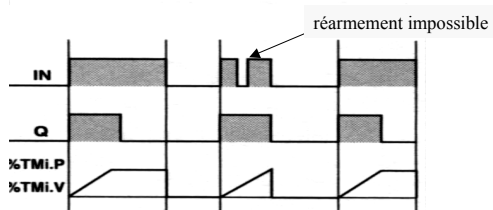
Mode TOF

retard au déclenchement



Mode TP

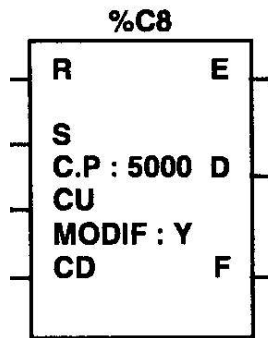
monostable de durée donnée



Blocs prédéfinis (sur bits)

Bloc compteur

Instructions:



Valeurs :

%Ci.V : valeur courante

%Ci.P : valeur de présélection
(0 ... 9999)

Armement (sur front montant) :

CU : entrée comptage : incrémente %Ci.V

CD : entrée décomptage : décrément %Ci.V

R : remise à zéro (si R=1, on force %Ci.V=0)

S : Présélection (si S=1, %Ci.V= %Ci.P)

Sorties :

D « done » passe à 1 qd %Ci.V= %Ci.P

F : débordement comptage (> 9999)

E : débordement décomptage (<0)

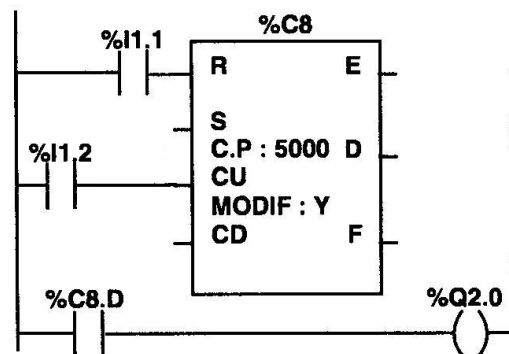
Blocs prédéfinis (sur bits)

Bloc compteur (2)

Exemple:

- Comptage d'un nombre de pièces=5000.
- Chaque impulsion sur %I1.2 provoque l'incrément du compteur %C8
- A la fin du comptage, le moteur %Q2.0 est mis en route.
- Le compteur est remis à zéro si %I1.1 = 1

Solution:

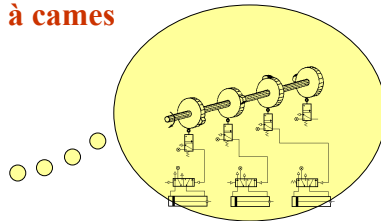
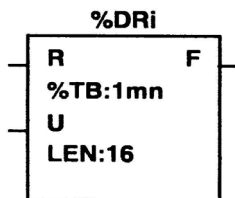


Détection interne front montant en CU et CD => instruction « -| - » correcte

Blocs prédéfinis (sur bits)

Bloc « Drum » ou programmeur à cames

Instructions:



Valeurs :

LEN : nombre de pas (1 à 16)

Base de temps : TB

TB : base de temps

%DRI.V : durée d'un pas (en test ou lecture)

%DRI.S : numéro du pas (test ou lecture)

Armement (U, R) et vérification (F)

U: « up » : sur front montant, provoque le passage au pas suivant

R « reset » : initialisation au pas 0

F: « full » : Mis à 1 lors du dernier pas en cours.

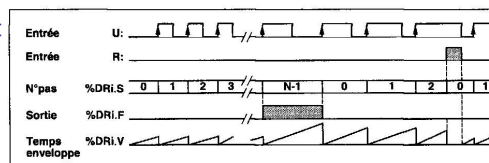
Blocs prédéfinis (sur bits)

Bloc « Drum » (2)

Définition des actions: à chaque pas, une action de sortie => Tableau à 2 entrées

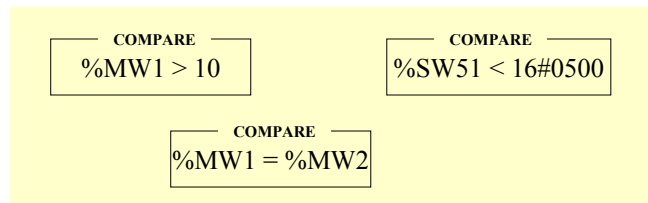
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | REPÈRES |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|---------|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | %Q2.1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | %Q2.3 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | %Q3.5 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | %M0 |
| 4 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | %M10 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | %Q2.6 |
| 6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | %Q2.7 |
| 7 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | %Q2.8 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | %M20 |
| 9 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | %M30 |
| A | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | %Q2.9 |
| B | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | %Q3.6 |
| C | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | %M5 |
| D | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | %M6 |
| E | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | %M7 |

Chronogramme :

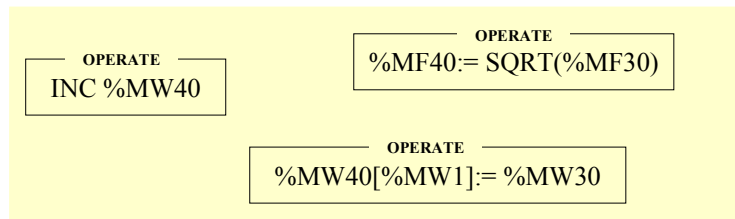


Tests et Opérations sur mots

Dans la zone « test » : Compare

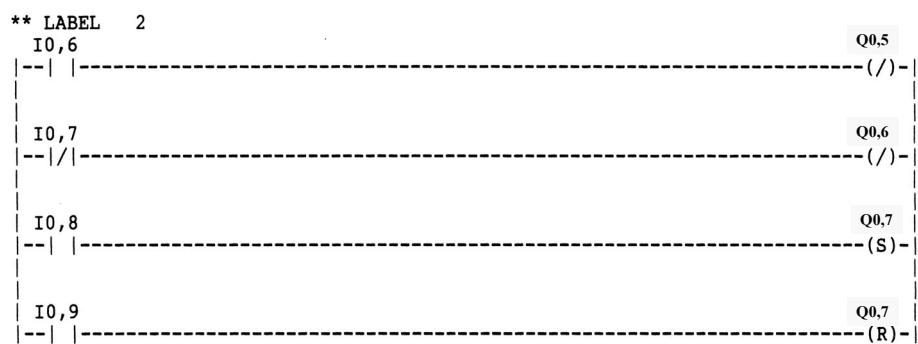


Dans la zone « action » : Operate



Exercice

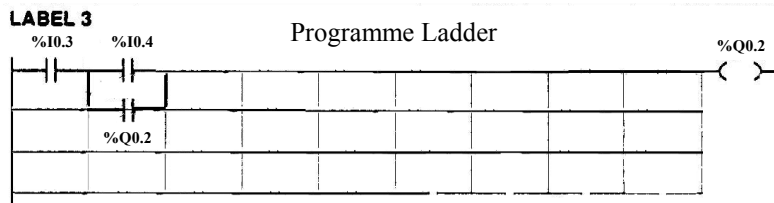
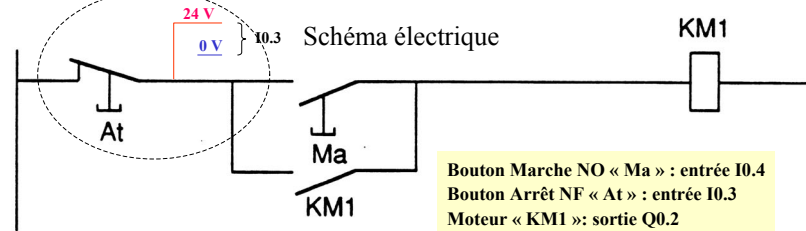
Bobines inverses, Set - Reset



Exemples

Exercice

Marche / arrêt moteur



Attention : ne pas « confondre » le type de contact physique avec le type de contact programmé : ex. : Bouton arrêt « At »

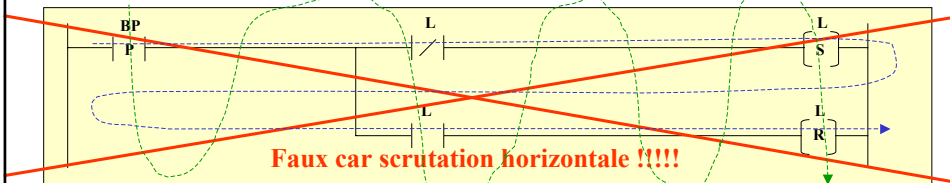
Exemples

Exercice

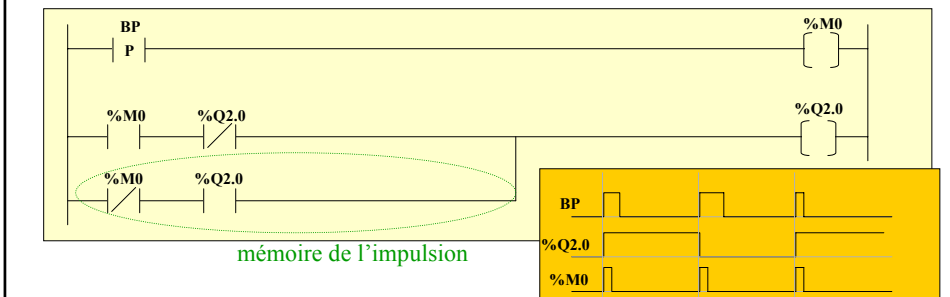
Le télérupteur (l'exercice « benchmark ») :

« en poussant successivement sur BP, la lampe L s'allume, s'éteint, s'allume, ... »

Version 1 :

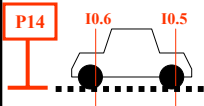


Version 2 :

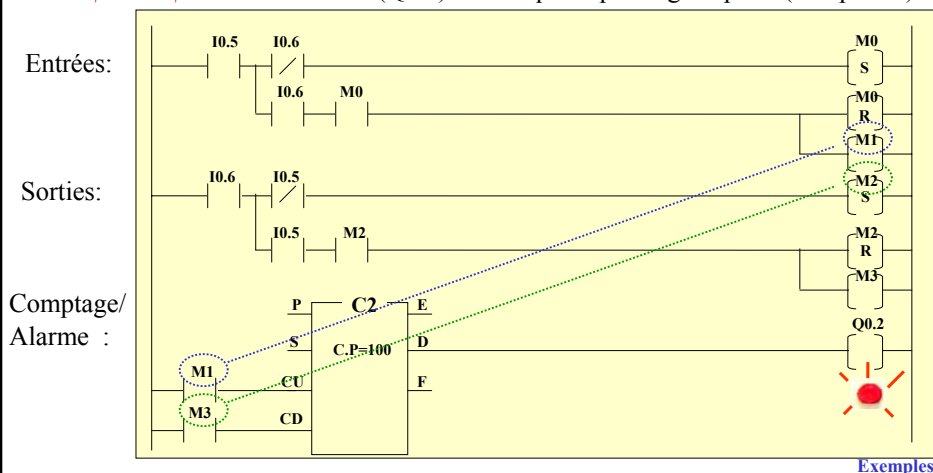


Exercice

Comptage de visiteurs dans un parking: énoncé/solution



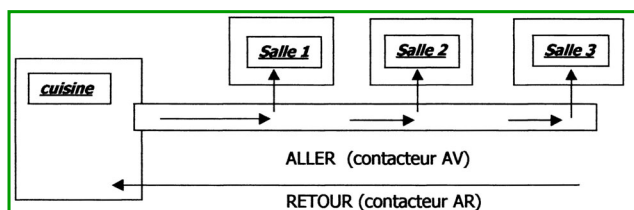
Un passage contrôle, grâce à 2 cellules (I0.5 et I0.6), les entrées et les sorties des véhicules.
L'ordre dans lequel elles sont occultées indique le sens.
Une lumière (Q0.2) avertit que le parking est plein (100 places)



Exemples

Exercice

Commande d'un passe-plat: énoncé



Le pupitre de commande situé dans la cuisine d'un restaurant comporte :

- Un interrupteur à trois positions permettant la sélection de la salle I1 (salle 1) – I2 (salle 2) – I3 (salle 3).
- Un bouton poussoir A (NO) qui démarre le transfert vers les salles.
- Un bouton poussoir R (NO) qui démarre le retour vers la cuisine.

NB : Le retour du passe-plat se fait toujours directement jusqu'à la cuisine sans transit. La commande 'aller' par contre, permet l'alimentation de la salle 1 puis de la salle 2 puis de la salle 3 sans demande de retour.

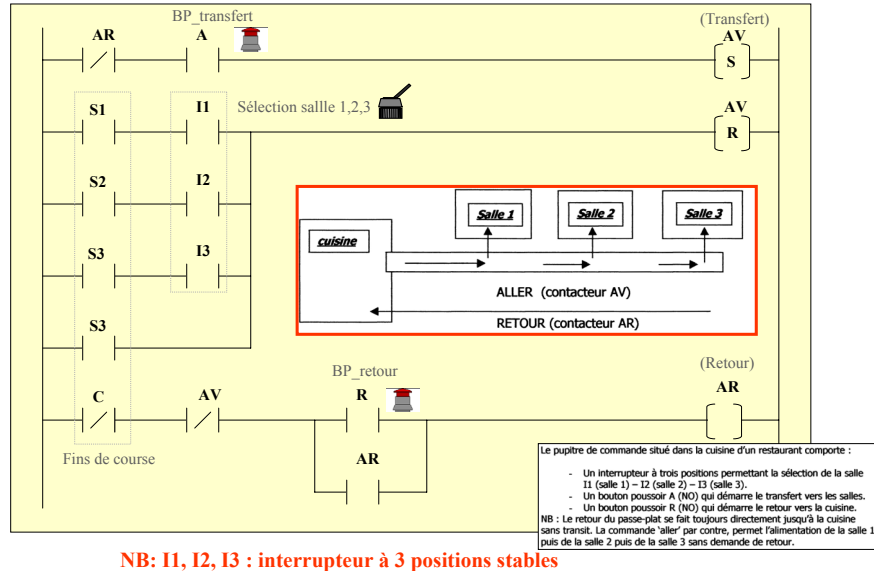
Quatre fins de course détectent la position du passe plat :

- C (NO) pour la cuisine.
- S1 (NO) pour la salle 1.
- S2 (NO) pour la salle 2.
- S3 (NO) pour la salle 3.

Exemples

Exercice

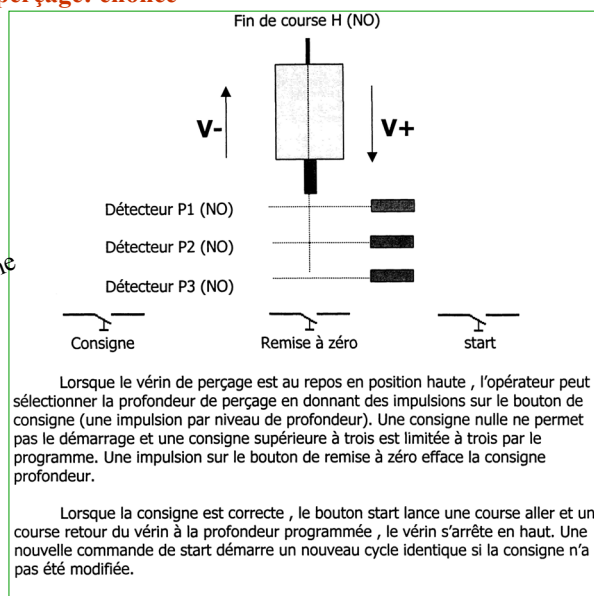
Commande d'un passe-plat: une solution



Exercice

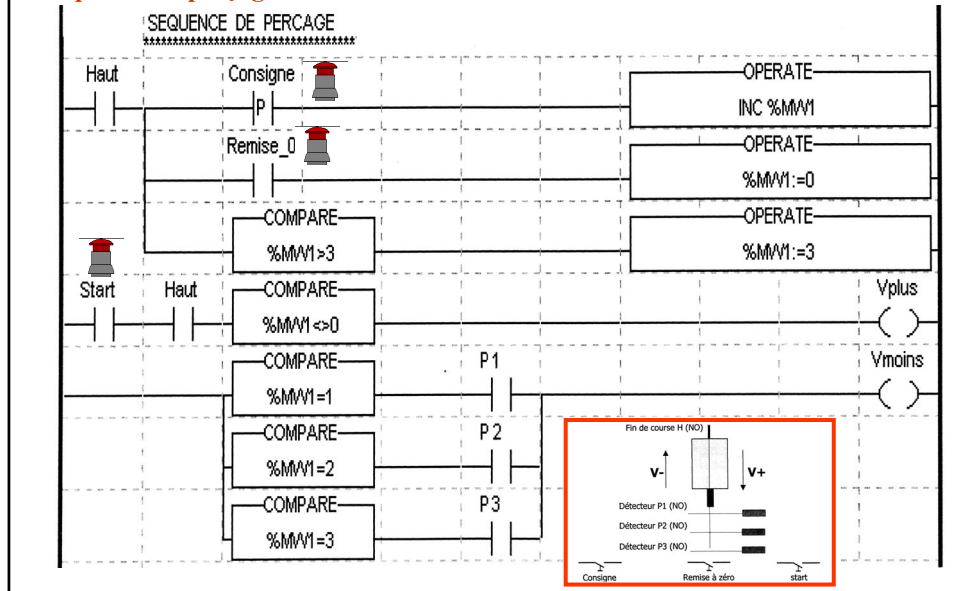
Séquence de perçage: énoncé

Lire attentivement ... et le programme est déjà quasi écrit !



Exercice

Séquence de perçage: une solution



Exercice

Alarme entretien énoncé:

Soit un moteur asynchrone (contacteur C) commandé par deux boutons poussoirs : marche M (NO) et arrêt A (NF).

Après 3000 heures de fonctionnement effectif, un voyant V clignote (1 Hz) afin de signaler la nécessité d'un entretien.

Un bouton poussoir ACQ (NO) remet l'horloge à zéro ce qui éteint le voyant (prise en charge de l'entretien par l'opérateur).

Une solution:

Commande moteur:

