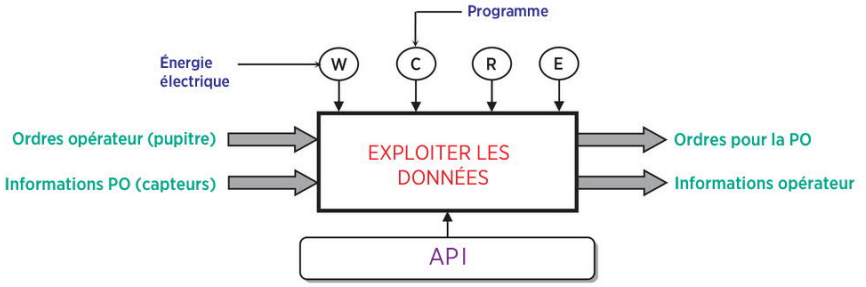
AU

7. L'Automate Pr ogr ammable Industriel

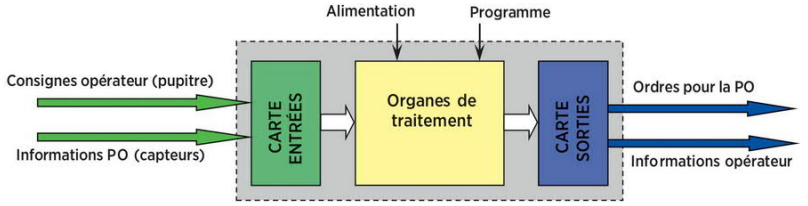


Fonction globale



L'Automate Programmable Industriel (API) permet d'automatiser un système industriel. Il doit être programmé en utilisant un langage de programmation adapté (*voir Fiche AU-8*).

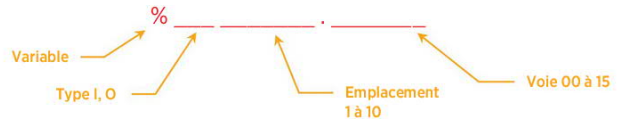
Structur e automate



Les organes de traitement permettent de traiter le programme contenu dans l'EEPROM (mémoire interne) grâce au microprocesseur et ainsi activer ou désactiver les sorties en fonction de l'état des entrées.

Adr essage entr ées/ sorties

Entrées (I pour Input) / Sorties (O pour Output) exemple : %I1.11 ; %O2.03



Remarque : pour certains automates (par exemple le TSX 37), la sortie est notée Q.

Type d' entr ées/ sorties

Les entrées/sorties d'un automate peuvent être de 2 types :

- logique, aussi appelée Tout Ou Rien (TOR), c'est-à-dire qu'elles n'ont que 2 états : active ou inactive ;
- analogique, qui permet de convertir une grandeur analogique en valeur numérique en entrée et inversement en sortie.

Exemples



Millénium 3 **Crouzet**
(voir Fiche AU-15)



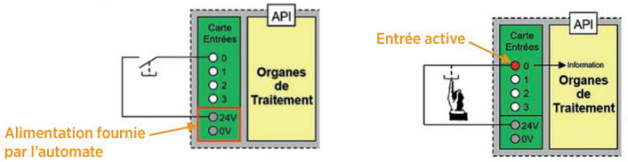
Simatic S7-1500 **Siemens**



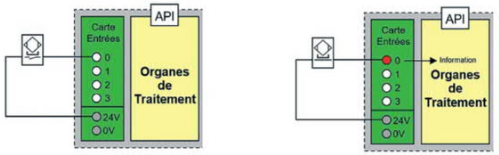
TSX Micro **Schneider**
(voir Fiche AU-9)

Câblage des entrées TOR

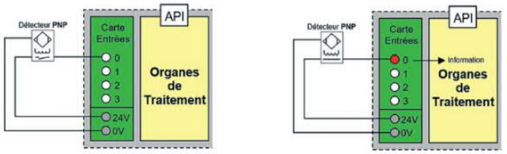
Raccordement d'un bouton poussoir NO



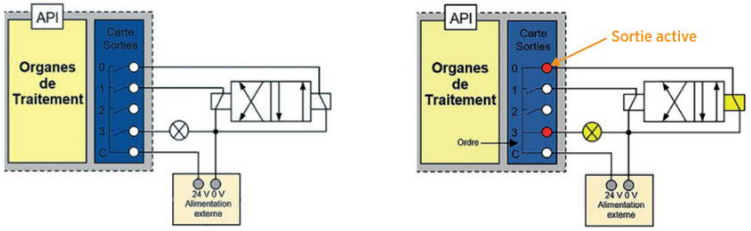
Raccordement d'un capteur 2 fils (capteur mécanique, ILS, ...)



Raccordement d'un capteur 3 fils (détecteur optique, inductif, capacitif...)



Câblage des sorties TOR



AU

8. Les langages de programmation



Introduction

L'**automate programmable industriel** (API *voir Fiche AU-7*), coordonne le fonctionnement du système automatisé auquel il appartient, en recevant les **informations** de ce dernier et en transmettant les **ordres** à la partie opérative.

L'enchaînement des actions du cycle machine, la réponse du système aux aléas, la gestion des sécurités et la synchronisation avec d'autres systèmes sont autant de contenus figurant dans le **programme automate**.

La programmation peut s'effectuer :

- directement sur l'API à l'aide de touches de programmation ;
- à l'aide d'une console de programmation reliée par un câble spécifique à l'API ;
- à l'aide d'un PC sur lequel est installé le logiciel de programmation associé à l'automate.

Bien que chaque automate dispose de son propre **langage de programmation**, il existe des langages universels à tous les types d'automates, parmi lesquels on distinguera deux familles : les **langages graphiques** et les **langages littéraux** (ou textuels).

Les langages normalisés

La norme internationale CEI 61131 (partie 3) définit **cinq langages de programmation standardisés** qui peuvent être utilisés pour les automates programmables industriels.

On retrouve 2 langages textuels :

- **IL** : « *Instruction List* » ou liste d'instructions ;
- **ST** : « *Structured Text* » ou texte structuré ;

et 3 langages graphiques :

- **LD** : « *Ladder Diagram* » ou schéma à contacts ;
- **FBD** : « *Function Block Diagram* » ou blocs fonctionnels ;
- **SFC** : « *Sequential Function Chart* » ou grafcet.

Le langage « Ladder »

LD : « *Ladder Diagram* » ou schéma à contacts.

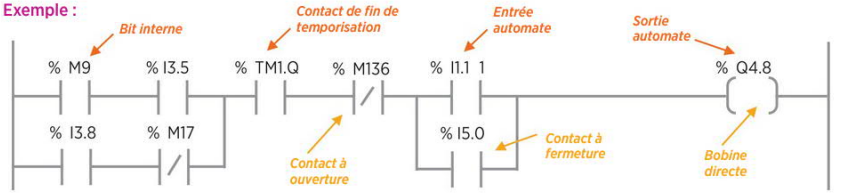
Langage le plus couramment utilisé dans les applications industrielles du fait de sa simplicité de lecture, le **Ladder** s'applique particulièrement à la programmation d'équations booléennes (vrai/faux), par représentation graphique d'entrées (contacts) et de sorties (relais ou bobines).

Les symboles graphiques représentant les variables sont organisés dans un réseau, appelé diagramme en « échelle » (« *ladder* » en anglais), placé entre deux barres d'alimentation à la façon d'éléments d'un schéma électrique à contacts.

Dans une programmation en Ladder, où chaque contact ou bobine représente une variable logique, les équations combinatoires régissant le fonctionnement du système sont lues de haut en bas et de gauche à droite.

De nombreux objets de programmation sont également disponibles pour compléter ce langage (bit, mot, temporisation, compteur, opérateur, set, reset, etc.).

Exemple :



Dans cet exemple, pour que la sortie automate soit activée, les variables d'entrée qui la conditionnent doivent avoir une valeur permettant de valider l'équation booléenne (voir Fiche AU-6) suivante :

$$\%Q4.8 = ((\%M9 \cdot \%I3.5) + (\%I3.8 \cdot \overline{\%M17})) \cdot \%TM1.Q \cdot \overline{\%M136} \cdot (\%I1.11 + \%I5.0)$$

Le langage « liste d'instructions »

IL : « *Instruction List* » ou liste d'instructions.

Semblable au langage assembleur, le langage par **liste d'instructions** est un langage de programmation textuel dit « de bas niveau » (proche du langage machine), pour les applications de petite taille à actions simples.

Quelques règles de base de la programmation IL :

- une seule instruction par ligne ;
- une ligne d'instruction peut comporter 4 champs : *Étiquette / Opérateur / Opérande / Commentaire ;*
- seuls les champs « Opérateur » et « Opérande » sont obligatoires ;
- les instructions opèrent sur un résultat courant ;
- un opérateur (d'affectation, logique, arithmétique ou de comparaison) indique le type d'opération à effectuer entre le résultat courant et l'opérande ;
- le résultat de l'opération est stocké dans le résultat courant ;
- dans une instruction, on peut trouver des modificateurs et un ou plusieurs opérandes séparés par des virgules.

```
Transfert :          LD      %M14
                     OR      %I3.8
                     ANDN    %TM2.Q
                     ST      %Q2.11

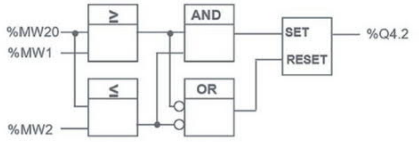
Défaut transfert :   OR (    %X28
                     AND     %I3.15
                     )
                     OR (    %X29
                     ANDN    %I3.15
                     )
                     JMP  Mode défaut
```

Le langage « blocs f onctions »

FBD : « *Function Block Diagram* » ou blocs fonctionnels.

Le **langage FBD** permet de programmer graphiquement des équations complexes, à l'aide de blocs fonctionnels, d'opérateurs ou de fonctions.

Les blocs sont pré-programmés dans l'automate ou programmés par l'utilisateur.



Les blocs sont représentés par des rectangles, interconnectés par des lignes de flux avec, à gauche de chaque bloc ses variables d'entrée, et à droite sa variable de sortie.

Les équations combinatoires régissant le fonctionnement du système sont lues de haut en bas et de gauche à droite.

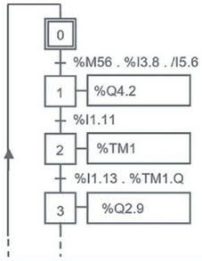
La langage « Gr afcet »

SFC : « *Sequential Function Chart* ».

Basé sur le Grafcet (*voir Fiche AU-5*), le **langage SFC** permet la programmation simple de tous les procédés séquentiels.

Le programme est défini comme une suite connue d'états stables appelés « étapes », reliés entre eux par des transitions franchissables par validation d'une condition booléenne (*voir Fiche AU-6*).

Les actions réalisées dans les différentes étapes sont écrites en utilisant d'autres langages de programmation (ST, IL, LD ou FBD).



La langage « te xte structur é »

ST : « *Structured Text* » ou texte structuré

Ce langage est un langage textuel dit « de haut niveau », proche des langages informatiques. Il permet la programmation de tout type d'algorithme plus ou moins complexe, difficilement modélisable avec les langages graphiques.

Quelques règles de base de la programmation ST :

- un programme ST est une suite d'énoncés (assignation, appel de fonctions, énoncé de sélection, d'itération, etc.) ;
- la fin d'un énoncé est marquée par un point-virgule ;
- les noms utilisés dans le code source sont délimités par des séparateurs (actifs ou passifs) qui ont un rôle d'opérateur ;
- des commentaires peuvent être librement insérés.

```
IF %M0 THEN
    FOR %MW17:=0 TO 31 DO
        IF %MW18[%MW17]<>0 THEN
            %MW3:=%MW18[%MW17];
            %MW 4:=%MW 17;
            %M1  0:=TRUE;
            EXIT; (*Sortie de la boucle*)
        ELSE
            %M1  0:=FALSE;
        END_IF;
    END_FOR;
ELSE
    %M10:=FALSE;
END_IF;
```


AU

9. L'automate TSX 37



Intr oduction

L'**automate programmable industriel** (API *voir Fiche AU-7*), coordonne le fonctionnement du système automatisé auquel il appartient, en recevant les informations de ce dernier et en transmettant les ordres à la partie opérative, en fonction des contenus figurant dans sa mémoire : le **programme automate**.

Il existe des **langages de programmation** (*voir Fiche AU-8*), normalisés et universels à tous les types d'automates, parmi lesquels les langages textuels (**IL** : « *instruction list* » ou liste d'instructions et **ST** : « *structured text* » ou texte structuré), et les langages graphiques (**LD** : « *ladder diagram* » ou schéma à contacts, **FBD** : « *function block diagram* » ou blocs fonctionnels et **SFC** : « *sequential function chart* » ou grafcet).

On trouve sur le marché de nombreux fournisseurs d'API : Schneider Télémécanique, April, Siemens, Omron, Crouzet, Allen-Bradley, etc. Chaque marque dispose de plusieurs gammes sur le marché, en fonction des besoins de l'installation. Parmi les automates les plus répandus sur les systèmes industriels, on trouve le **TSX 37**, aussi appelé TSX micro, de la marque **Schneider Elec** (anciennement Télémécanique).

Exemples de TSX 37



TSX 37-05 avec 1 carte d'E/S TOR et 1 slot vide



TSX 37 avec 3 slots



Module d'extension

Car actéristiques

- Alimentation : 24 V CC, 100-240 V~.
- Entrées automate disponibles :
  - entrées Tout-Ou-Rien (TOR) ;
  - entrées analogiques : 0-10 V pour capteurs ou potentiomètres.
- Sorties automate disponibles :
  - sorties relais standard ;
  - sorties analogiques.
- Possibilité de rajouter des bacs d'extensions, nombreuses configurations possibles et évolutives.
- Cartes de communication, cartes pour compteur.
- Adaptable sur rail DIN, coffret, tableau et armoire modulaire.



Cartes E/S TOR



Cartes E/S analogiques



Carte de communication



Carte pour compteur

Car actéristiques

