

# Chap V - API et PL7

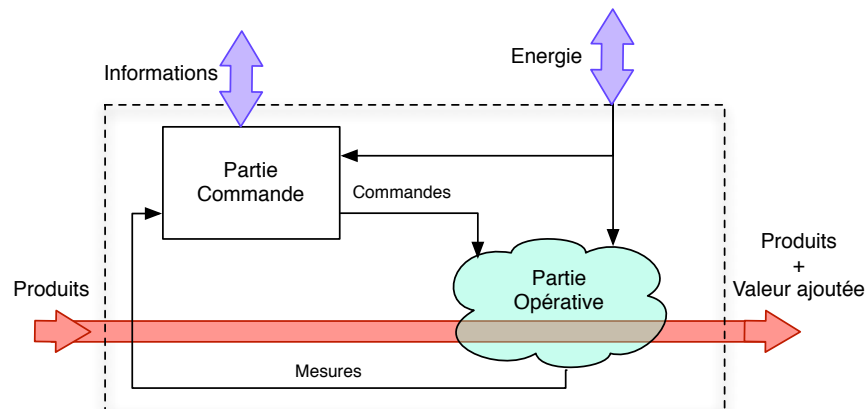
## 1. Généralités

### 1.1. Organisation d'un système automatisé

Un système de production a pour but d'apporter une valeur ajoutée à de la matière d'œuvre dans un contexte donné. Quand ce système est automatisé, on peut généralement le décomposer en deux parties :

- Une partie opérative dont les actionneurs agissent sur le processus automatisé.
- Une partie commande qui coordonne les différentes actions de la partie opérative et qui communique avec le ou les opérateurs.

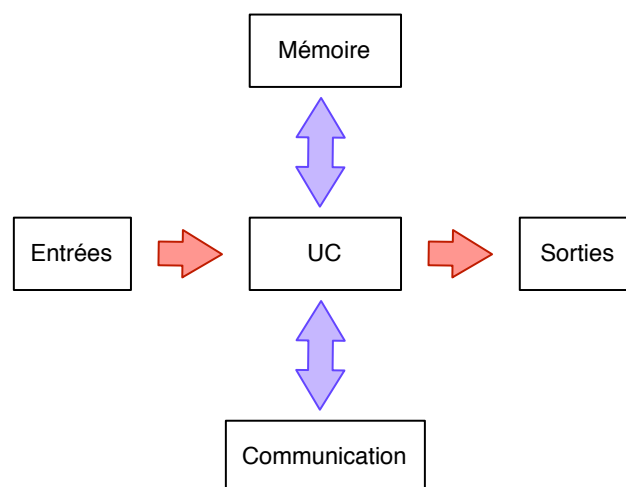
C'est dans la partie commande que l'on retrouvera les Automates Programmables Industriels.



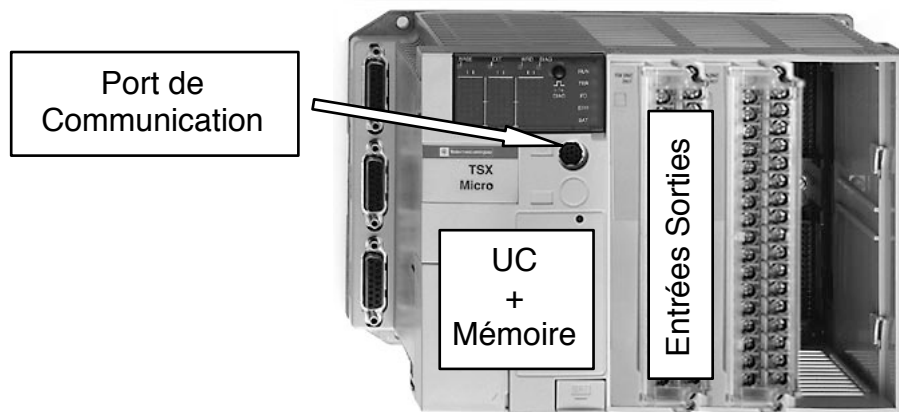
### 1.2. Structure générale d'un A.P.I.

L'unité centrale de l'automate programmable est entourée de différents éléments ;

- D'entrées qui lui permettent d'être informé de ce qui se passe sur le procédé ;
- De sorties qui lui permettent d'agir sur le procédé ;
- De mémoire où sont stockées les instructions du programme utilisateur et les éléments nécessaires à son fonctionnement ;
- D'un ou plusieurs modules de communication, qui lui permette de communiquer avec l'utilisateur.



On retrouve ces différents éléments sur le TSX Micro :



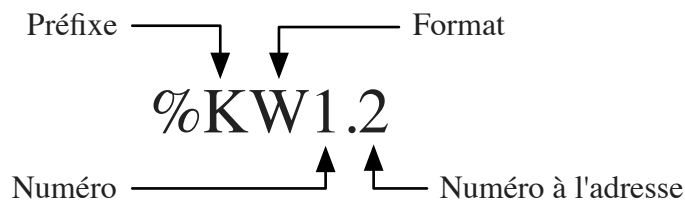
### 1.3. PL7-PRO

PL7-PRO est le langage que nous allons utiliser pour programmer nos automates. Il permet l'accès à tous les éléments des TSX. Le programme sera écrit sur ordinateur puis transféré sur l'automate. Il ne pourra être validé qu'en présence de celui-ci.

## 2. Les objets disponibles sur PL7-PRO

### 2.1. Repère

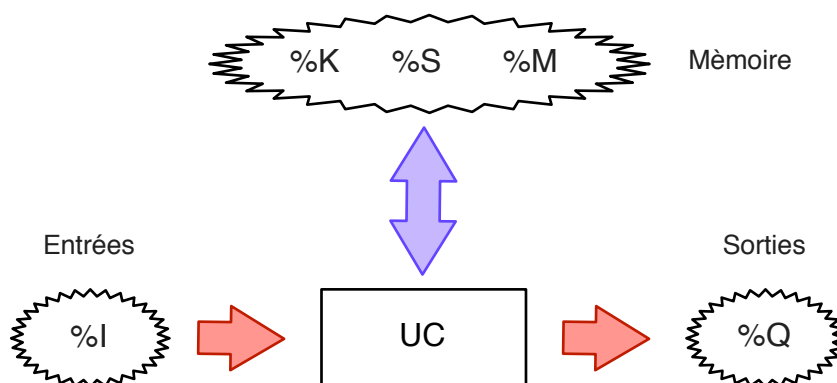
Tous les objets manipulés ont un repère qui commence par le caractère % suivi de lettres ou de nombre qui nous informe sur son identité.



### 2.2. Préfixe

Mémoire	M	Système	S	Constantes	K
Entrée	I	Sortie	Q		

En fonctionnement, les entrées ne sont modifiables que par la partie opérative. C'est l'automate qui fixe les valeurs des sorties. L'automate peut lire et écrire sur des mémoires, mais uniquement lire les objets système et les constantes.

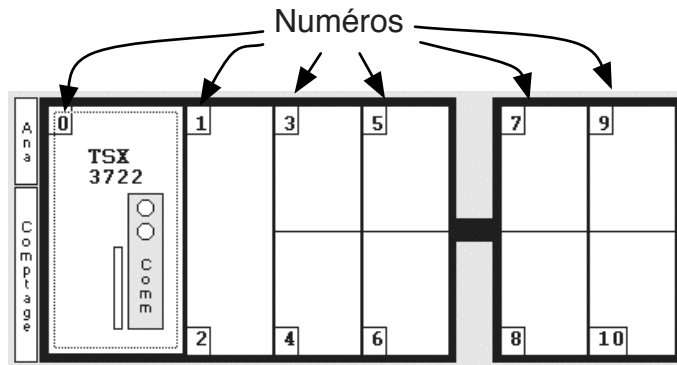


### 2.3. Format

Booléen (1 bit)		Byte (8 bits)	B	Word (16 bits)	W
Dword (32 bits)	D	Réel	F		

### 2.4. Numéro

Pour les entrées-sorties, le numéro nous donne une indication sur la situation géographique de l'élément. Pour les autres éléments, il donne juste une indication sur le rang de l'élément.



### 2.5. Front

Pour les entrées et sorties uniquement (%Ii.j et %Qi.j), on peut détecter leur front montant, en utilisant les contacts suivants :

Front Montant		Front descendant	
---------------	--	------------------	--

### 2.6. Bascule RS

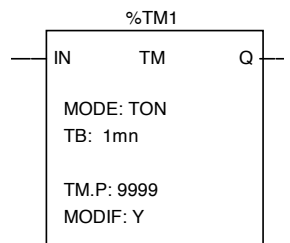
Une bascule RS est accessible pour les sorties booléens (%Ii.j) et les mémoires booléennes (%Mi) en utilisant les contacts suivants :

SET		RESET	
-----	--	-------	--

### 2.7. Temporisation

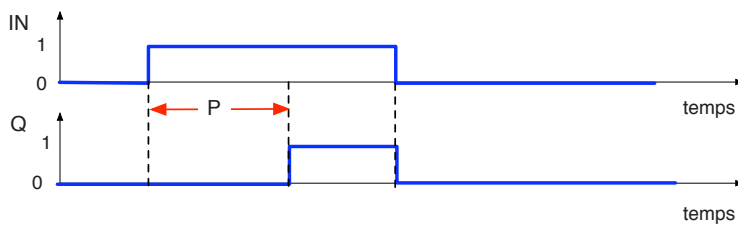
La temporisation (%Tmi) s'insère dans le réseau comme un objet avec une entrée et une sortie. La sortie fournira une valeur logique en retard par rapport aux entrées. La variable %Tmi.V fournit la valeur du temps.

IN	Entrée		
Q	Sortie	%Tmi.Q	
MODE	Mode de fonctionnement		TON, TOFF, TP
TB	Base de temps		10ms, 100ms, 1s, 1mn
P	Présélection	%Tmi.P	de 0 à 9999

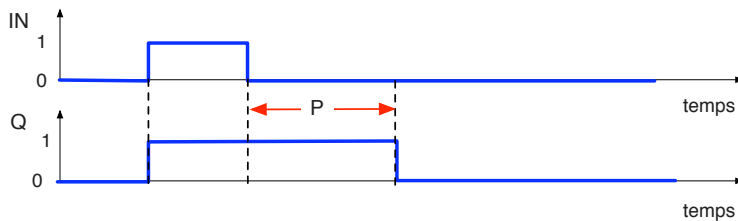


**Les modes de fonctionnements :**

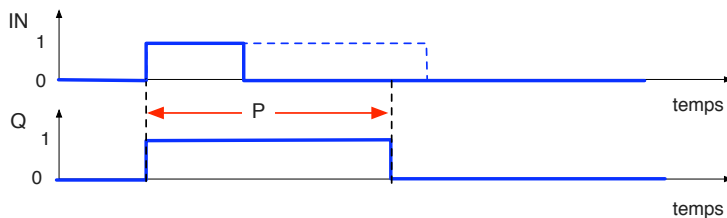
TON, temporisation travail :



TOFF, temporisation repos :



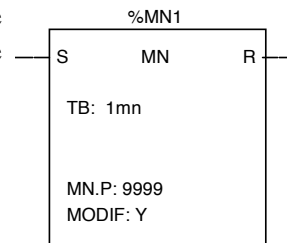
TP, monostable :



**2.8. Monostable**

Le monostable (%MNi) s'insère dans le réseau comme un objet avec une entrée et une sortie. La sortie fournira une valeur logique d'une durée paramétrable. La variable %MNi.V fournit la valeur du temps restant avant retour à zéro.

S	Mise à 1 sur front montant		
R	Sortie	%MNi.R	= 0 si MNi.V = 0
TB	Base de temps		1mn, 1s, 100ms, 10ms
MN.P	Valeur de présélection	%MNi.P	de 0 à 9999

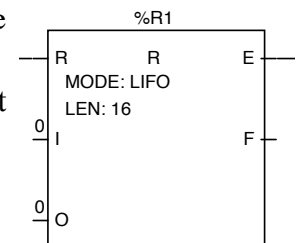


**2.9. Registre**

Un registre (%Ri) est un bloc mémoire permettant de stocker jusqu'à 255 mots de 16 bits de deux manières différentes :

- file d'attente (premier entré, premier sorti) appelée pile FIFO (First In, First Out),
- pile (dernier entré, premier sorti) appelée pile LIFO (Last In, First Out).

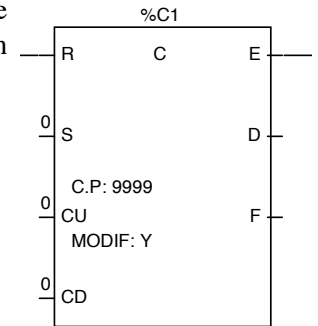
R	Remise à Zéro	
I	In, provoque l'entrée du mot %Ri.I dans le registre	
O	Out, provoque la sortie d'un mot du registre dans %Ri.O	
TYP	FIFO ou LILO	
LEN	Longueur	1 à 255
E	Registre vide	%Ri.E



### 2.10. Compteur

Le compteur (%Ci) s'insère comme un objet avec quatre entrées et trois sorties dans le schéma à contacts. Les sorties fourniront des valeurs logiques calculées à partir de son état. La variable %Ci.V fournit la valeur du compteur.

R	Remise à Zéro		%Ci.V := 0
S	Remise à Présélection		%Ci.V := %Ci.P
C.P.	Présélection	%Ci.P	
CU	Comptage (le front montant incrémente la valeur courante)		%Ci.V := %Ci.V+1
CD	Décomptage (le front montant décrémente la valeur courante)		%Ci.V := %Ci.V-1
E	Débordement décomptage	%Ci.E	
D	Présélection atteinte	%Ci.D	
F	Débordement comptage	%Ci.F	



#### Exemple d'utilisation :

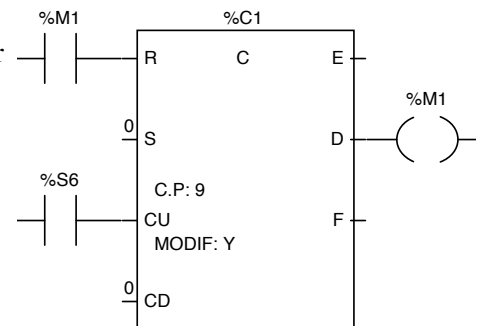
Le bit interne %S6 incrémente le compteur toutes les secondes.

Lorsque le compteur atteint la valeur présélectionnée (9), le compteur repasse à 0.

Le compteur peut avoir 9 états différents :

$$\%C1.V = 0, 1, 2, 3, 4, 5, 6, 7, 8.$$

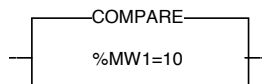
Le cycle dure 9 secondes.



### 3. Autres objets

#### 3.1. Bloc COMPARE

Le bloc compare permet de faire des comparaisons entre des nombres de mêmes types, entiers ou réels. Le circuit est fermé quand la comparaison est vraie, ouvert sinon.



#### 3.2. Bloc OPERATE

A l'aide de l'opérateur 'OPERATE', l'utilisateur est à même de faire des calculs (produit, division, somme et soustraction) entre des entiers ou des réels. Attention, pas de mélange des genres, les éléments présents dans une formule doivent être de mêmes types. Comme il s'agit d'une affectation et non d'une comparaison, on utilisera le := pour affecter le résultat du calcul de droite à l'élément de gauche.

