Cette norme des langages de programmation des automates programmables permet de les classer suivant trois catégories :

|  |  |  |
| --- | --- | --- |
| **Langages littéraux** | **Langages graphiques** | **Structure de programme** |
| **Langage IL** (liste d’instructions)  **Langage ST** (littéral structuré) | **Langage LD** (à contacts)  **Langage FBD** (à blocs fonctionnels) | **Diagramme SFC** (suite de séquences) |

La structure d’un programme séquentiel utilise les éléments **du diagramme fonctionnel de séquence SFC à ne pas confondre avec la description du comportement d’un système** (connu sous le nom de GRAFCET en France, voir norme **CEI 60848**)

I – ELEMENTS COMMUNS AUX DIFFERENTS LANGAGES

11 – Principaux types de données élémentaires :

Le type de données, la description et le nombre de bits sont précisés dans le tableau suivant :

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de données** | **Description** | **Taille ou nombre de bits** | **Etendue** |
| **BOOL** | Booléen | 1 | 0 ou 1 |
| **BYTE** | Cordon de 8 bits | 8 | Pas d’étendue numérique pour ce type de données |
| **WORD** | Cordon de caractères de longueur 16 | 16 |
| **DWORD** | Cordon de caractères de longueur 32 | 32 |
| **LWORD** | Cordon de caractères de longueur 64 | 64 |
| **INT** | Entier signé | n=16 | -2 n-1 à + (2n+1 – 1) |
| **UINT** | Entier non signé | n=16 | 0 à (2n – 1) |
| **REAL** | Nombre réel | 32 | Virgule flottante |
| **STRING** | Cordon de caractères | Encadré par deux apostrophes | |
| **TIME** | Durée | Dépend de l’application concernée | |

12 – Représentation des variables :

Une variable permet d’identifier des objets de données dont le contenu peut varier (données associes aux entrées, aux sorties ou aux emplacements mémoire de l’API)

Le tableau suivant permet de représenter symboliquement une variable. La variable débute par le symbole %.

|  |  |
| --- | --- |
| **Préfixe** | **Signification** |
| **I** | Input :Emplacement d’une entrée automate |
| **Q** | Output : Emplacement d’une sortie automate |
| **M** | Emplacement de mémento ou mémoire interne |
| **X** | Taille d’un seul bit |
| **B** | Taille d’un byte ou octet |
| **W** | Taille d’un word : mot de 16 bits |
| **D** | Taille d’un double Word : mot double de 32 bits |
| **L** | Taille d’un mot long : mot de 64 bits |

Exemples :

* %IW125 : Emplacement du mot d’entrée 125
* %QB17 : Emplacement de l’octet de sortie 17
* %MD48 : Emplacement du mot mémoire double 48

13 – Unités d’organisation de programmes :

Fonction : Module ayant plusieurs entrées possibles, une seule variable de sortie et pas de mémoire interne

Bloc fonctionnel : Module ayant plusieurs variables d’entrée et de sortie possibles et une mémoire interne

Programme : Module construit à l’aide de fonctions et de blocs fonctionnels. L’organisation interne d’un programme peut faire intervenir un diagramme fonctionnel de séquence SFC.

II – INVENTAIRE DES FONCTIONS STANDARDS:

Description des principales fonctions communes à tous les langages de programmation d’API :

Fonctions de conversion de types

**Exemple** cette fonction permet la conversion d’une variable d’entrée du type réel en une variable de sortie du type entier.

**REAL TO INT**

REAL

INT

Fonctions numériques

**Exemple** : cette fonction permet de calculer la racine carrée d’une variable d’entrée

**SQRT**

REAL

REAL

Fonctions cordons de bits

**Exemple**: cette fonction permet le décalage à gauche de N bits d’une variable d’entrée et le remplissage de zéros à droite

**IN SHL OUT**

**N**

Cordon de bits

Cordon de bits

UINT

**Exemple**: cette fonction réalise le ET Booléen

**IN1 AND OUT**

**IN2**

BIT

BIT

BIT

Fonctions de sélection et de comparaison

**Exemple** : cette fonction permet de déterminer la valeur maximale entre trois variables d’entrée

**IN1 MAX OUT**

**IN2**

**IN3**

INT

INT

INT

INT

**Exemple** : cette fonction permet de sélectionner une des deux variables d’entrée suivant la variable G

**G SEL OUT**

**IN1**

**IN2**

BOOL

INT

INT

INT

**Exemple** : cette fonction réalise la comparaison de supériorité entre deux variables d’entrée (IN1>IN2)

**IN1 GT OUT**

**IN2**

REAL

BOOL

REAL

II – LANGAGES LITTERAUX :

21 – Langage IL (Instruction List ou langage à liste d’instructions)

Comme son nom l’indique, le programme est constitué d’une suite d’instructions respectant le format suivant :

|  |  |  |  |
| --- | --- | --- | --- |
| **Etiquette**  (non obligatoire) | **Opérateur** | **Opérande(s)** | **Commentaire**  (non obligatoire) |

Ce langage est proche du langage de programmation d’un microprocesseur : l’assembleur

**Exemple**: Soit à commander une électrovanne EV du schéma TI suivant :

LH

Type NO

Nmax

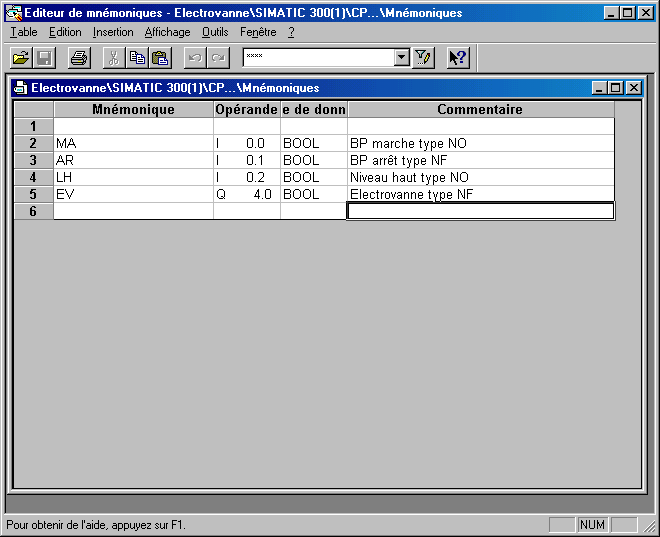
**MA type NO**

**AR type NF**

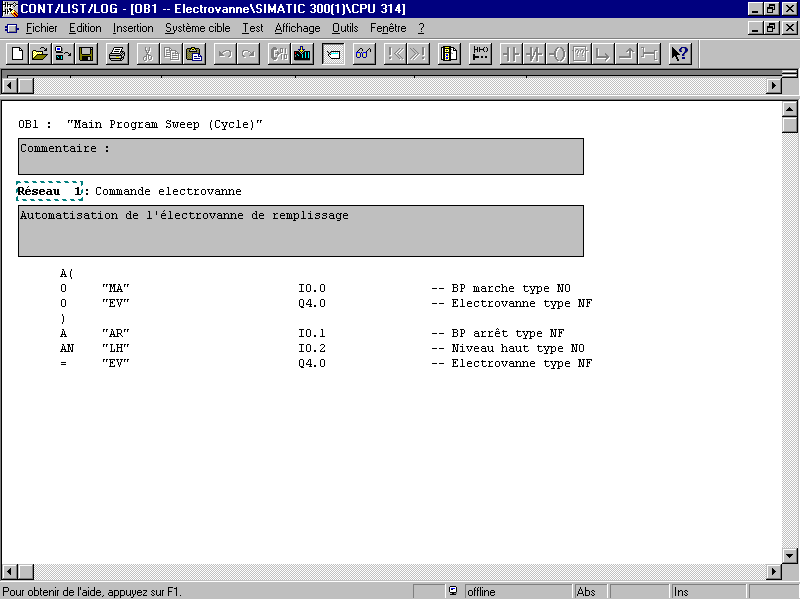
**EV**

|  |  |  |  |
| --- | --- | --- | --- |
| **Etiquette**  **(non obligatoire)** | **Opérateur** | **Opérande(s)** | **Commentaire**  **(non obligatoire)** |
| Début : | AND ( |  |  |
|  | OR | %I0.0 | BP Marche NO |
|  | OR | %Q4.0 | Electrovanne |
|  | ) |  |  |
|  | AND | %I0.1 | BP Arrêt NF |
|  | AND N | %I0.2 | Niveau haut NO |
|  | ST | %Q4.0 | Affectation électrovanne |

Exemple de programmation avec le logiciel STEP7 de Siemens :



Programme correspondant en IL ou LIST :



22 – Langage ST ( Structured Text) ou langage littéral structuré)

Ce langage est composé d’expressions littérales constituées d’opérateurs et d’opérandes et d’énoncés.

Ce langage est proche d’un langage informatique comme le PASCAL.

Exemple 1 de la commande de l’électrovanne :

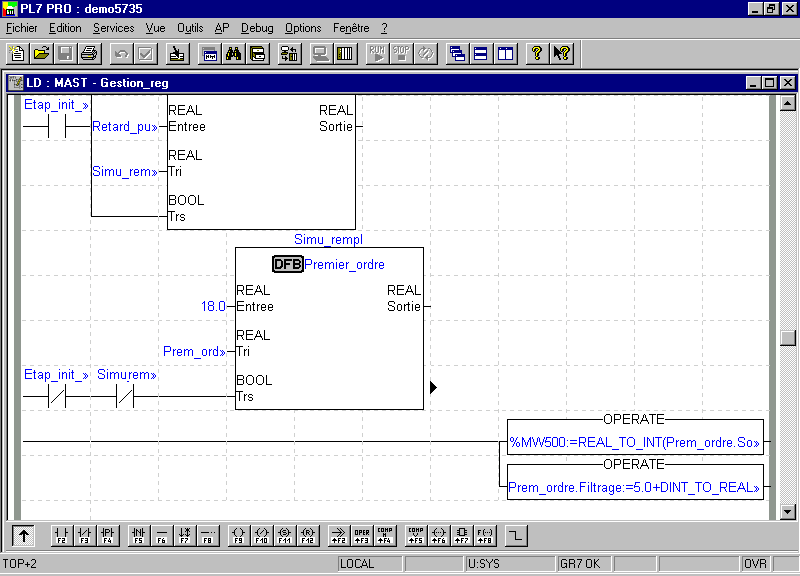
%L1 (« commande electrovanne »)

IF (%I0.0 OR %Q4.0) AND %I0.1 AND NOT %I0.2 THEN SET %Q4.0

END IF;

Exemple 2 issu d’un programme de démonstration « Machine de dosage et mélange produits » du logiciel PL7 de Schneider.

Soit la simulation d’un premier ordre sous forme d’un bloc fonctionnel DFB « Premier ordre » (voir langage FBD) :



Le contenu de ce bloc écrit en **langage ST** est le suivant :

(\*\*\* Routine de Simulation d'un procédé de 1er ordre pour Sortie PID \*\*\*)

(\* Récupération de la Période de la tâche MAST nécessaire à la formule \*)

IF Trs THEN

Sortie := Tri;

ELSE

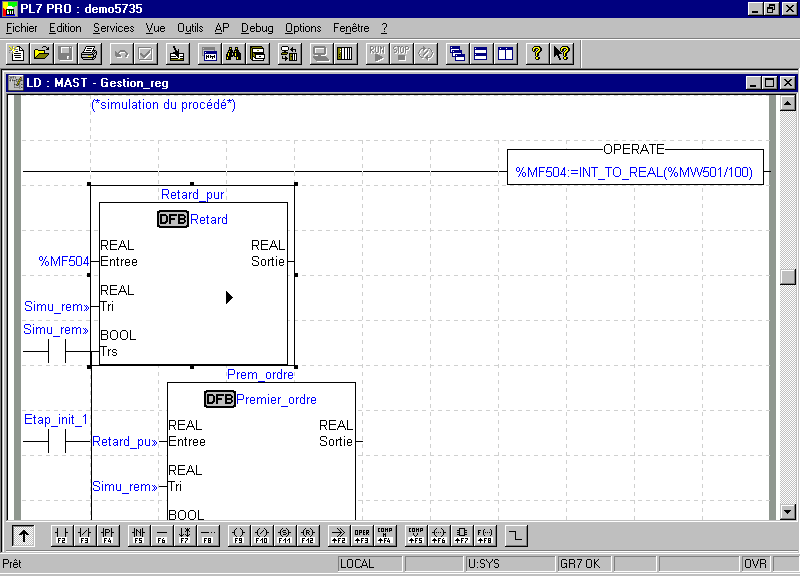
Te:=INT\_TO\_REAL(%SW0)/1000.0;

Sortie:=(Filtrage/(Filtrage+Te))\*Sortie+(Gain\*Te)/(Filtrage+Te)\*Entree;

END\_IF;

Avec : **%SW0** : Période de scrutation de la tâche maître. Permet de modifier la période de la tâche maître définie en configuration, par le programme utilisateur ou par le terminal. La période est exprimée en ms (1..255ms). %SW0=0 en fonctionnement cyclique.

Soit la simulation d’un retard pur sous forme d’un bloc fonctionnel DFB **« Retard »** :



Le contenu de ce bloc écrit en **langage ST** est le suivant :

Nb := REAL\_TO\_INT(1000.0\*Retard/INT\_TO\_REAL(%SW0));

IF (Nb>=999) THEN

Nb := 999;

END\_IF;

IF Trs THEN

Memoire := Tri;

Sortie := Tri;

Pos := 0;

ELSE

Sortie := Memoire[Pos];

Memoire[Pos]:= Entree;

Pos := Pos +1;

IF (Pos>= Nb) THEN

Pos := 0;

END\_IF;

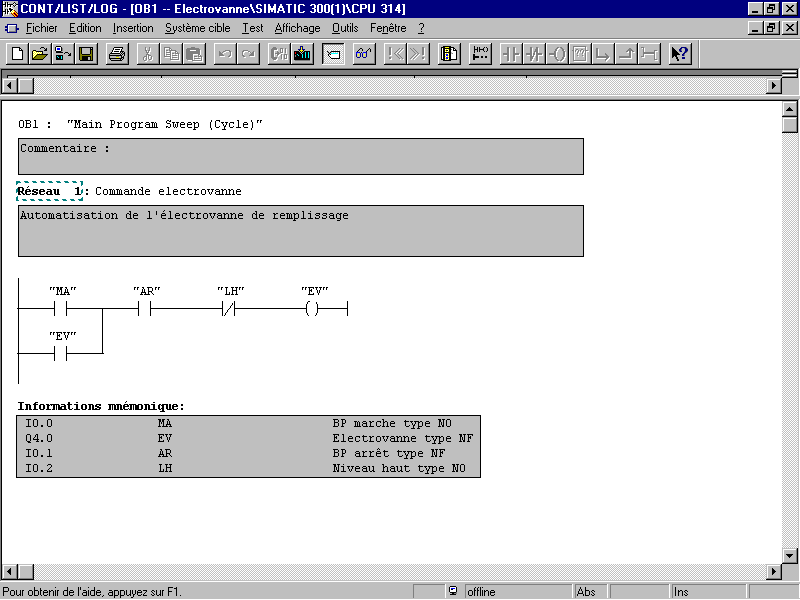
END\_IF;

III – LANGAGES GRAPHIQUES :

31 – Langage LD (Ladder Diagram) ou langage à contacts :

Ce langage est constitué de réseaux de contacts et de bobines entre deux barres d’alimentation. Ce langage est proche des schémas électriques.

En reprenant l’exemple de la page 3 et sur API Siemens en langage à contacts STEP7 :

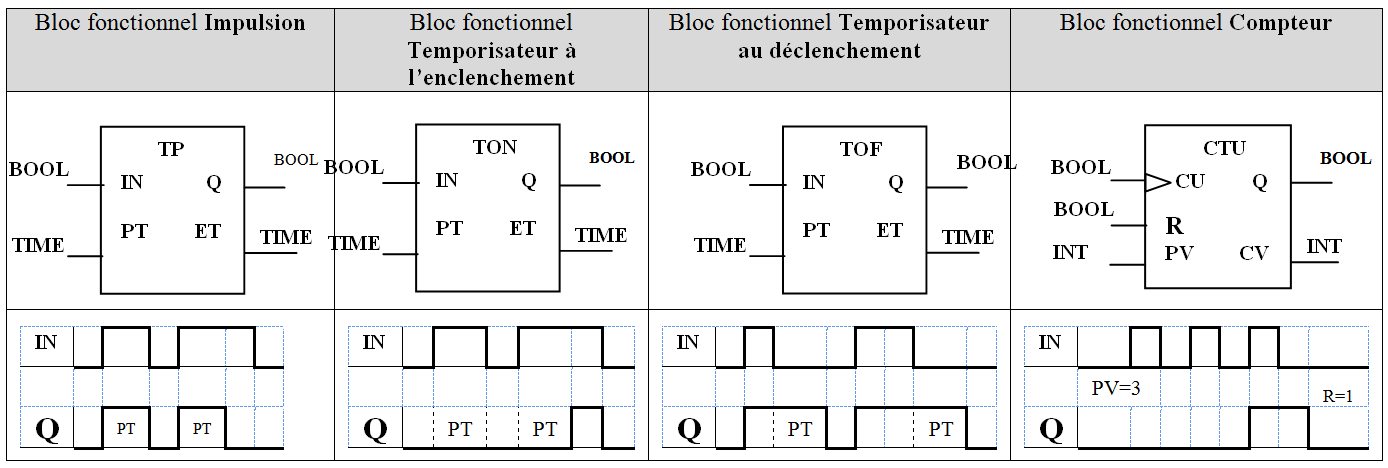


32 – Langage FBD (Function Bloc Diagram) ou langage en blocs fonctionnels :

Ce langage se compose de réseaux de fonctions préprogrammées ou non, représentées par des rectangles. Ces blocs fonctionnels sont connectés entre eux par des lignes, le flux des signaux se faisant de la sortie (à droite) d’une fonction vers l’entrée à gauche de la fonction raccordée.

*Exemples de blocs fonctionnels standards (fourni par le constructeur de logiciel) :*

* Bloc fonctionnel compteurs
* Bloc fonctionnel temporisateurs
* Bloc fonctionnel PID…



Exemples de blocs fonctionnels utilisateurs (développés par le programmateur et réutilisables)

***Exemple 1 de la page 3 :***

Arrêt

**Commande**

Marche **EV**

**Électrovanne**

Seuil haut

**BOOL**

**BOOL**

**BOOL**

**BOOL**

Exemple 2 : Alarmes niveaux d’une cuve

|  |  |
| --- | --- |
| Cahier des charges :   * Surveiller les seuils haut et bas d’une cuve * Convertir le niveau en % * Signaler un défaut du capteur   Soit à créer le bloc fonctionnel suivant : | **SEL type NO**  LT1  Nmax  **EV**  Nmini  **LL1**  **LH1**  **LD1** |

**SEUILS** **M**

**EN**

**MD**

**ML**

**Ma**

**MH**

**BOOL**

**BOOL**

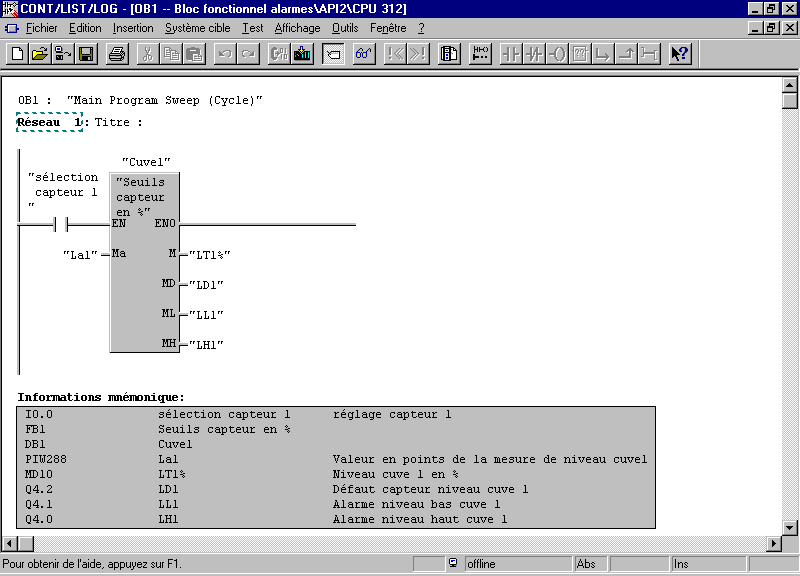
**INT**

**REAL**

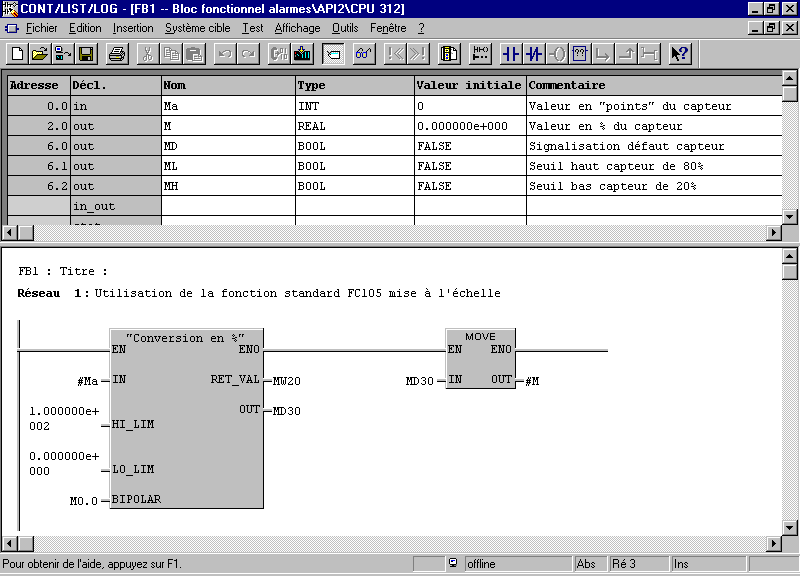
**BOOL**

**BOOL**

Programme en STEP 7 : À l’appel du bloc FB, on indique le N° du bloc de données d’instance DB associé (mémoire interne). Ce bloc sert à mémoriser l’état des différentes variables déclarées



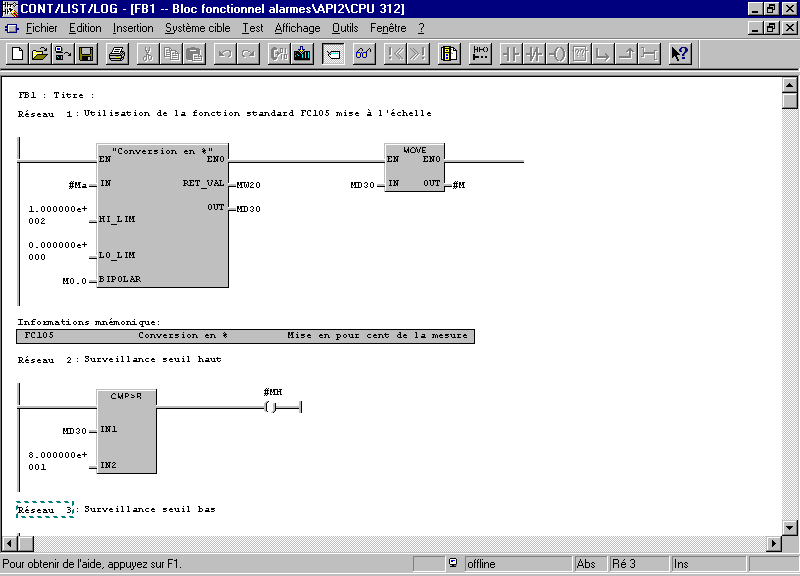
Liste des variables formels associées au bloc fonctionnel « SEUILS » :

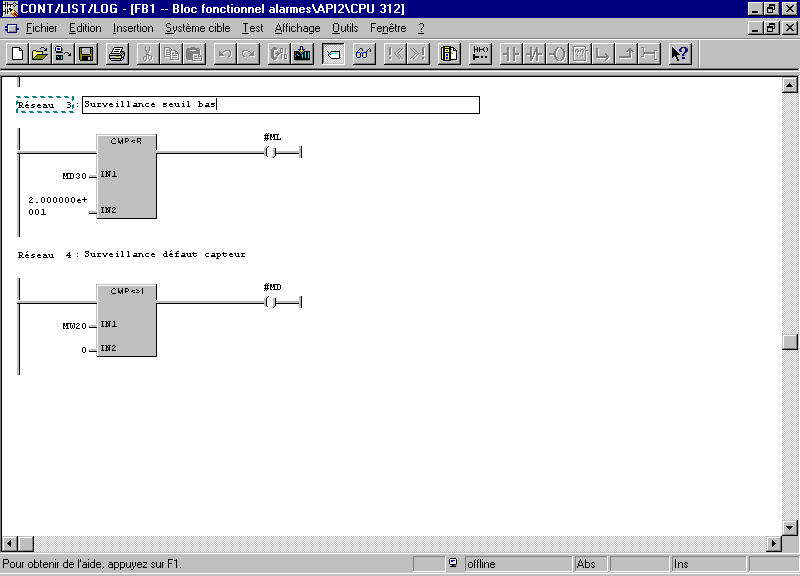


Le contenu du bloc fonctionnel est le suivant :

Il fait appel à une fonction standard : « «Mise à l’échelle (SCALE) » qui prend une valeur entière (IN) et la convertit selon l’équation ci-après en une valeur réelle exprimée en unités physiques, comprises entre une limite inférieure (LO\_LIM) et une limite supérieure (HI\_LIM) , dans notre cas entre 0 et 100% :

* OUT = [ ((FLOAT (IN) – K1)/(K2–K1)) \* (HI\_LIM–LO\_LIM)] + LO\_LIM
* Pour une entrée unipolaire :K1 = 0.0 et K2 = +27648.0





IV – STRUCTURATION D’UN PROGRAMME :

Le diagramme SFC (Sequentiel Function Chart) ou langage fonctionnel de séquences :

Le diagramme fonctionnel de séquence SFC à ne pas confondre avec la description du comportement d’un système (connu sous le nom de GRAFCET en France, voir norme CEI 60848).

Ce langage est destiné à la description de fonctions de commande séquentielles.

Le programme correspondant est constitué d’un ensemble d’étapes et de transitions reliés entre elles par des liaisons dirigées. Chaque étape est associée à un ensemble d’actions. Chaque transition est associée à une condition de transition.



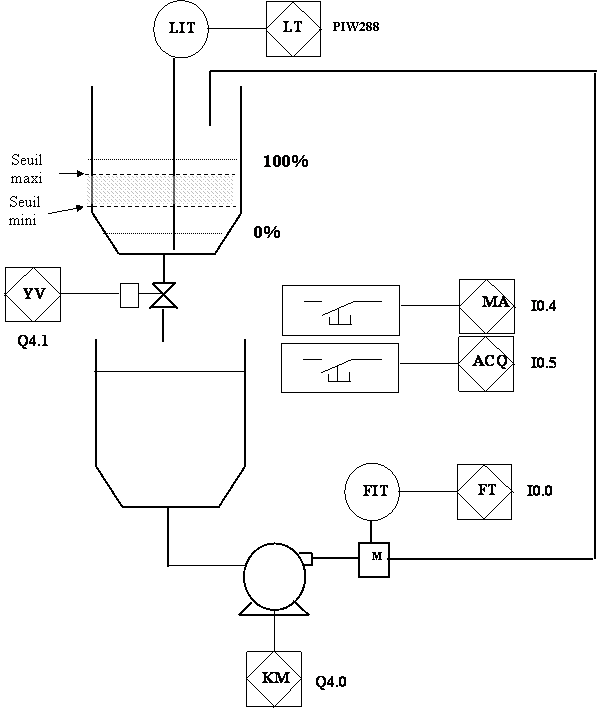
Le programme API traduisant cette représentation peut utiliser l’un des quatre langages de programmation.

Exemple de la même déclaration d’actions sous différents langages:

|  |  |
| --- | --- |
| Déclaration d’actions en langage FBD : | ACTION\_4  %IX1  %MX3  S8.X  %QX17  &  LT  C  D  S1  R  %MX10  SR  Q1  MEM\_1  **S5** |
| Déclaration d’actions en langage ST : | ACTION\_4  %QX17 := %IX1 & %MX3 & S8.X ;  MEM\_1 (S1 := (C<D)) ;  %MX10 := MEM1\_Q ;  **S5** |

Exemple de programme écrit en SFC : ici le logiciel STEP7-GRAPH de Siemens :

Schéma de l’installation :



Cahier des charges :

On dispose d’une cuve munie d’un capteur de niveau et d’un débitmètre à sortie à impulsions.

On demande de déterminer, de façon automatique et à l’aide d’un API, le volume d’eau compris entre les repères seuil mini et seuil maxi de la cuve

Table de mnémoniques :

****

|  |  |
| --- | --- |
| **Grafcet de l’application** | **Grafcet en langage SFC de STEP7** |
| **3**  **POMPE : = 0**  Acquittement « Mesure effectuée »  **2**  **Comptage impulsions**  Niveau liquide ≥ **seuil maxi**  **1**  **POMPE : = 1**    Niveau liquide ≥ **seuil mini**  **4**  **Vidange cuve et Raz compteur d’impulsions**  Niveau liquide < 0%  **0**  Mise en marche système. (Niveau liquide < 0%)  (🡩 Impulsion débitmètre) |  |