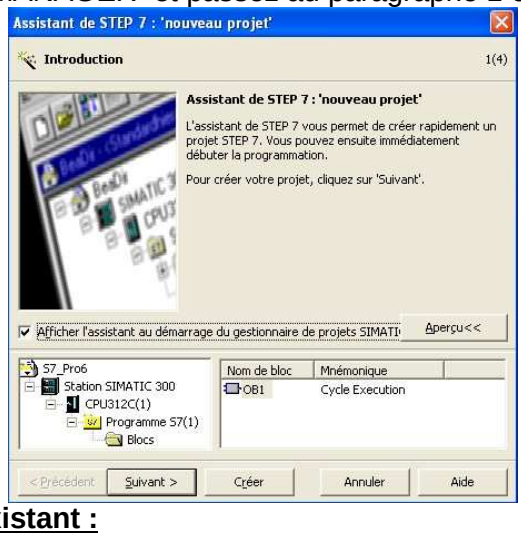
Fiche de programmation S7

Automate SIEMENS CPU 3xx 

Lancez le logiciel SIMATIC MANAGER et passez au paragraphe 1 ou 2.



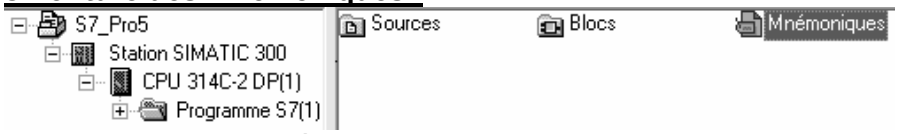
1.Ouverture d'un fichier existant :

Cliquez sur «Annuler », fermez les projets en cours. Puis cliquez sur « Fichier » et « Ouvrir ». Sélectionnez votre projet dans son répertoire et cliquez sur « OK ». Passez au paragraphe 3.

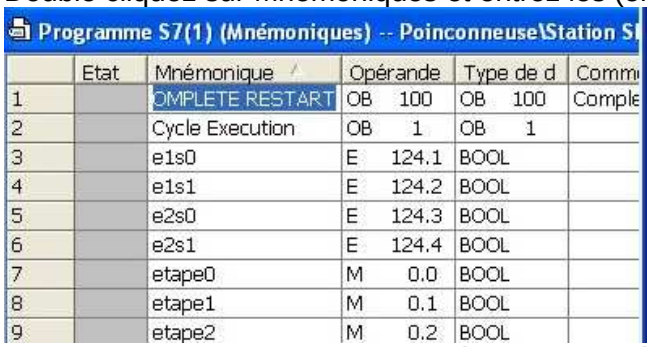
2.Création d'une programmation en LADDER :

Cliquez sur suivant et choisissez CPU314C 2DP.
Cliquez sur suivant et choisissez OB1 et CONT.
Cliquez sur suivant, donnez un nom à votre projet et cliquez sur Créer.
La fenêtre suivante s'ouvre :

3.Ecriture des mnémoniques :



Double cliquez sur Mnémoniques et entrez les (en voici un exemple).



Cliquez sur Enregistrer et fermez l'éditeur de mnémoniques.

4.Ecriture du programme :

On écrit le programme en utilisant différents blocs :

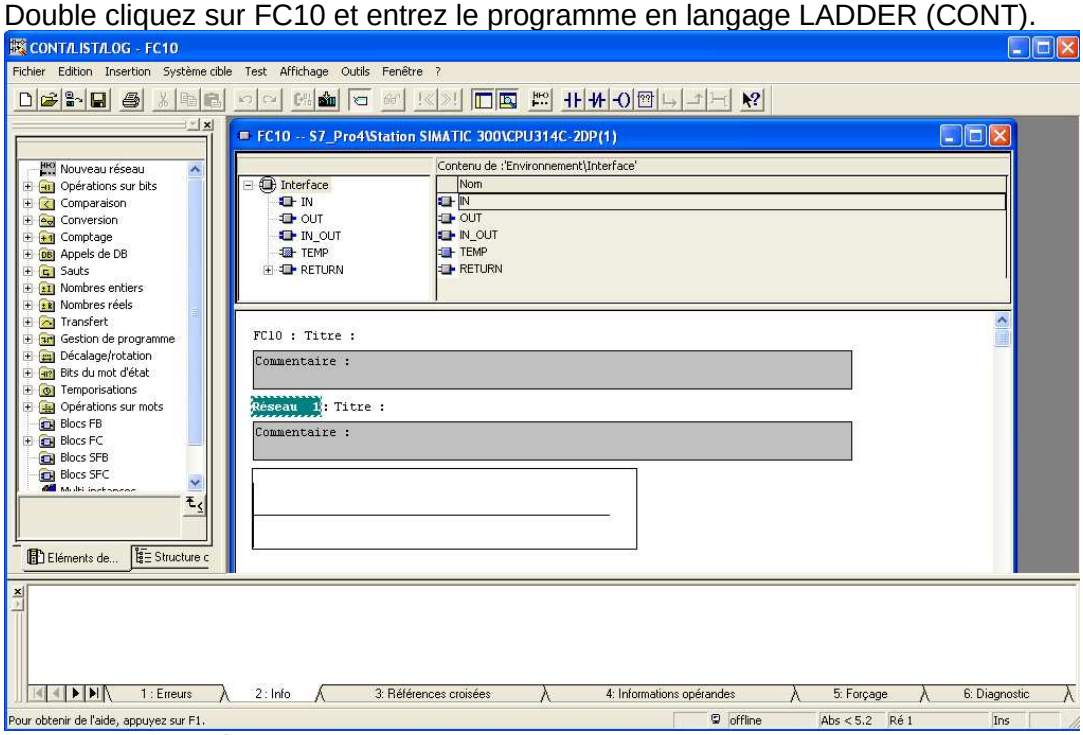
FC10 : étapes,
FC30 : actions internes,
FC40 : actions externes,
OB1 : appel des fonctions.

OB100 : initialisation.

Cliquez droit sur Blocs et choisissez "Insérer un nouvel objet", puis "Fonction" pour FC10 à FC40.

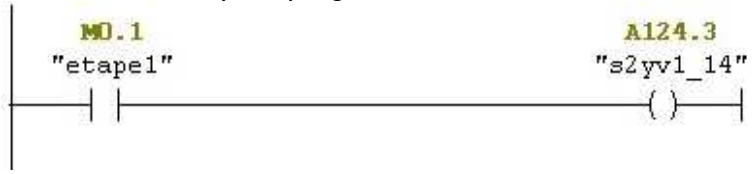
Cliquez droit sur Blocs et choisissez "Insérer un nouvel objet", puis "Bloc d'organisation" pour OB100.

Double cliquez sur FC10 et entrez le programme en langage LADDER (CONT).



Attention : un réseau et un seul par sortie.

Utilisez les icones suivants pour programmer 



Exemple :

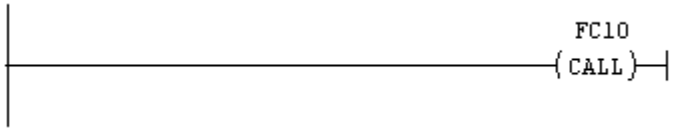
Cliquez sur « nouveau réseau »  pour passer au réseau suivant.

Une fois le programme entré, cliquez sur Enregistrer et fermer l'éditeur CONT.

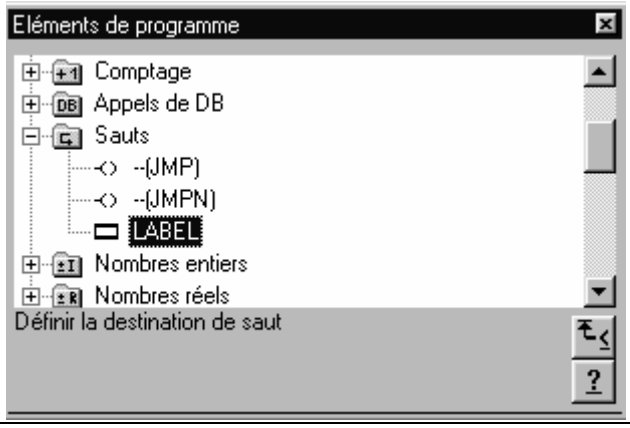
Recommencez pour FC20, FC30, FC 40, OB1 et OB100.

Le programme de OB1 sera un appel aux différentes fonctions (bloc CALL).

Exemple de l'appel de FC10 dans OB1 :

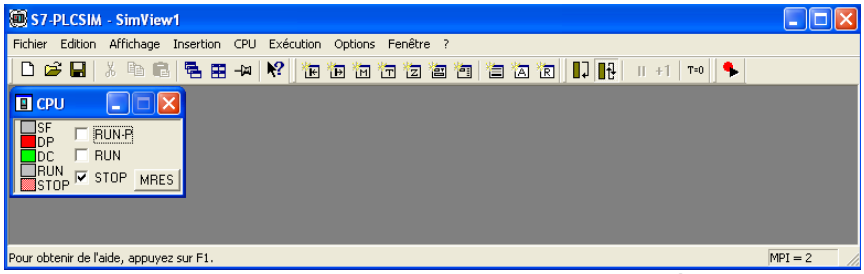


Remarque :
Pour insérer un label (opérations de saut), il faut insérer un élément de programme (commande Insertion et Eléments de programme). Choisir Label dans saut et le déplacer en début de réseau.



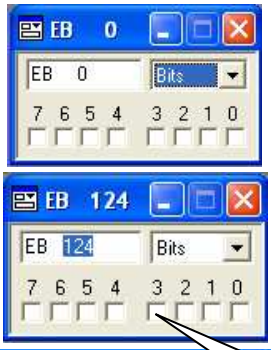
5.Test du programme avec l'automate de simulation :

Pour utiliser l'automate de simulation, cliquez sur l'icône 
La fenêtre du simulateur s'ouvre :

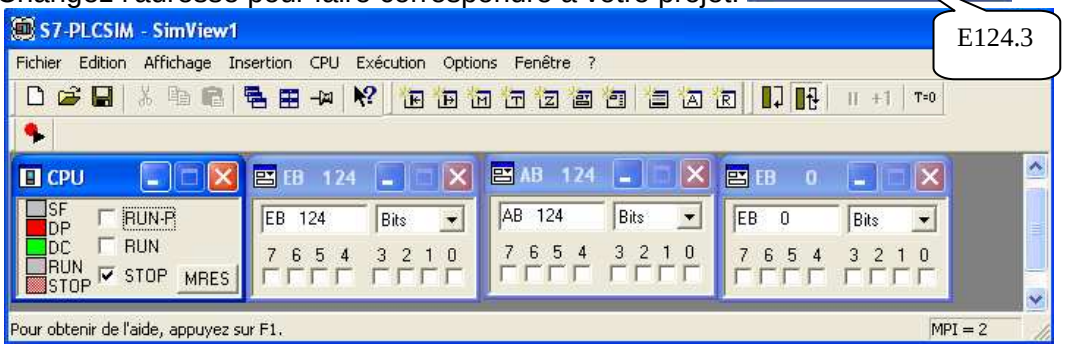


Vous devez configurer votre API avec les cartes et éventuellement des zones mémoires.

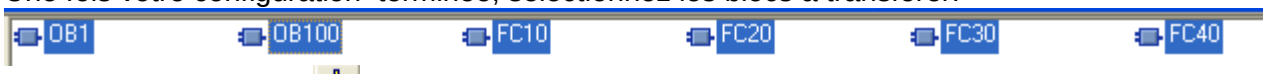
Pour ajouter une carte d'entrées, cliquez sur l'icône  .



Changez l'adresse pour faire correspondre à votre projet.

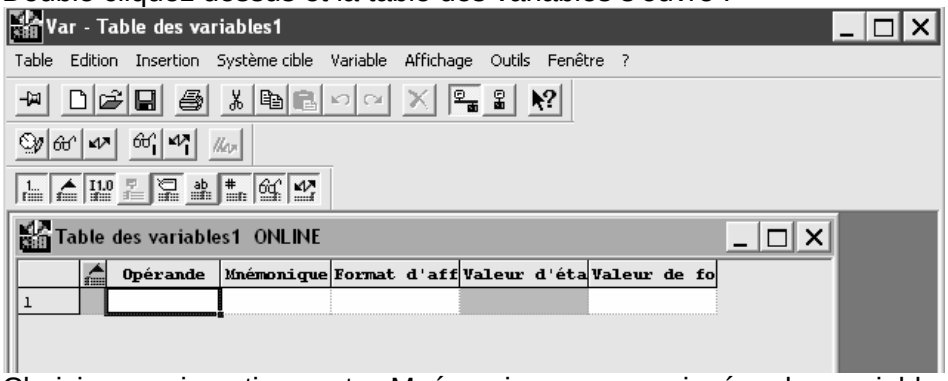


Une fois votre configuration terminée, sélectionnez les blocs à transférer.



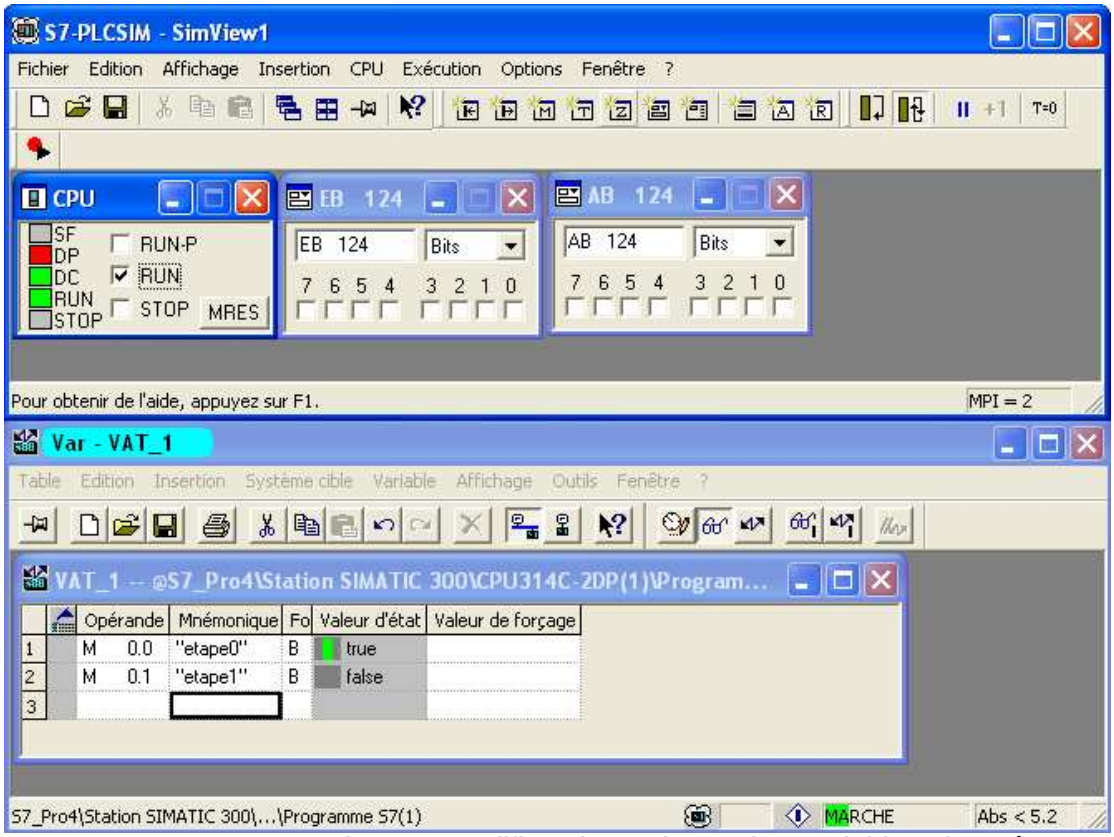
Cliquez sur « Charger » pour transférer votre programme dans la mémoire de l'API.

Vous devez créer une table d'animation en faisant un clic droit sur Blocs et choisissez "Insérer un nouvel objet", puis "Table des variables".
Double cliquez dessus et la table des variables s'ouvre :



Choisissez « insertion » et « Mnémoniques » pour insérer les variables à visualiser (toutes les étapes). Cliquez sur pour voir l'état de ces variables en lignes.
Enregistrez votre table.

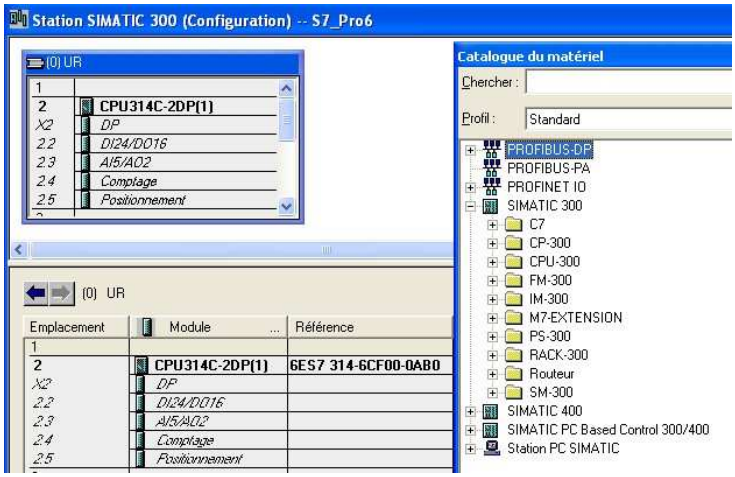
Placez l'automate en RUN.



Testez votre programmation en modifiant les valeurs des variables d'entrées. Corrigez là, si besoin.

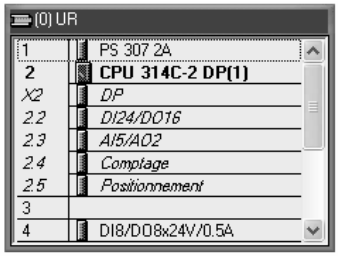
6.Configuration de l'automate :

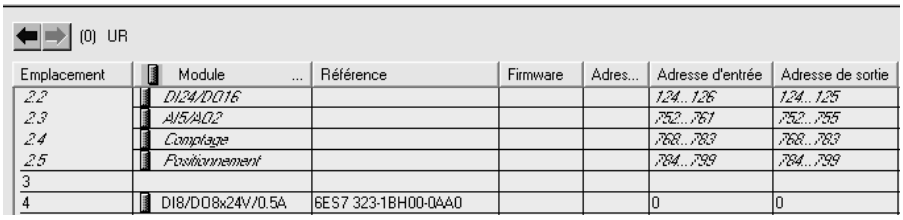
Cliquez sur « Station SIMATIC 300 » et double cliquez sur matériel.
La fenêtre suivante s'ouvre :



Si la fenêtre du catalogue n'est pas ouverte, ouvrez la par affichage et catalogue.
Vous devez ensuite choisir le matériel et le faire glisser à l'emplacement désiré.
Faites glisser l'alimentation PS 307 2A en position 1 (à choisir dans le dossier PS-300 (de SIMATIC 300) du catalogue).

Vous devez obtenir la configuration suivante (il faut ajouter la carte de simulation (dans DI/DO-300 de SM-300) :





Configurez l'UC si besoin avec un clic droit sur l'unité centrale et "propriétés de l'objet".
Cliquez sur « enregistrer et compiler »  .
Cliquez sur « Charger »  pour transférer votre configuration dans la mémoire de l'API.
Fermez HW config.

7.Test réel

Fermez l'automate de simulation. Sélectionnez les blocs à transférer.

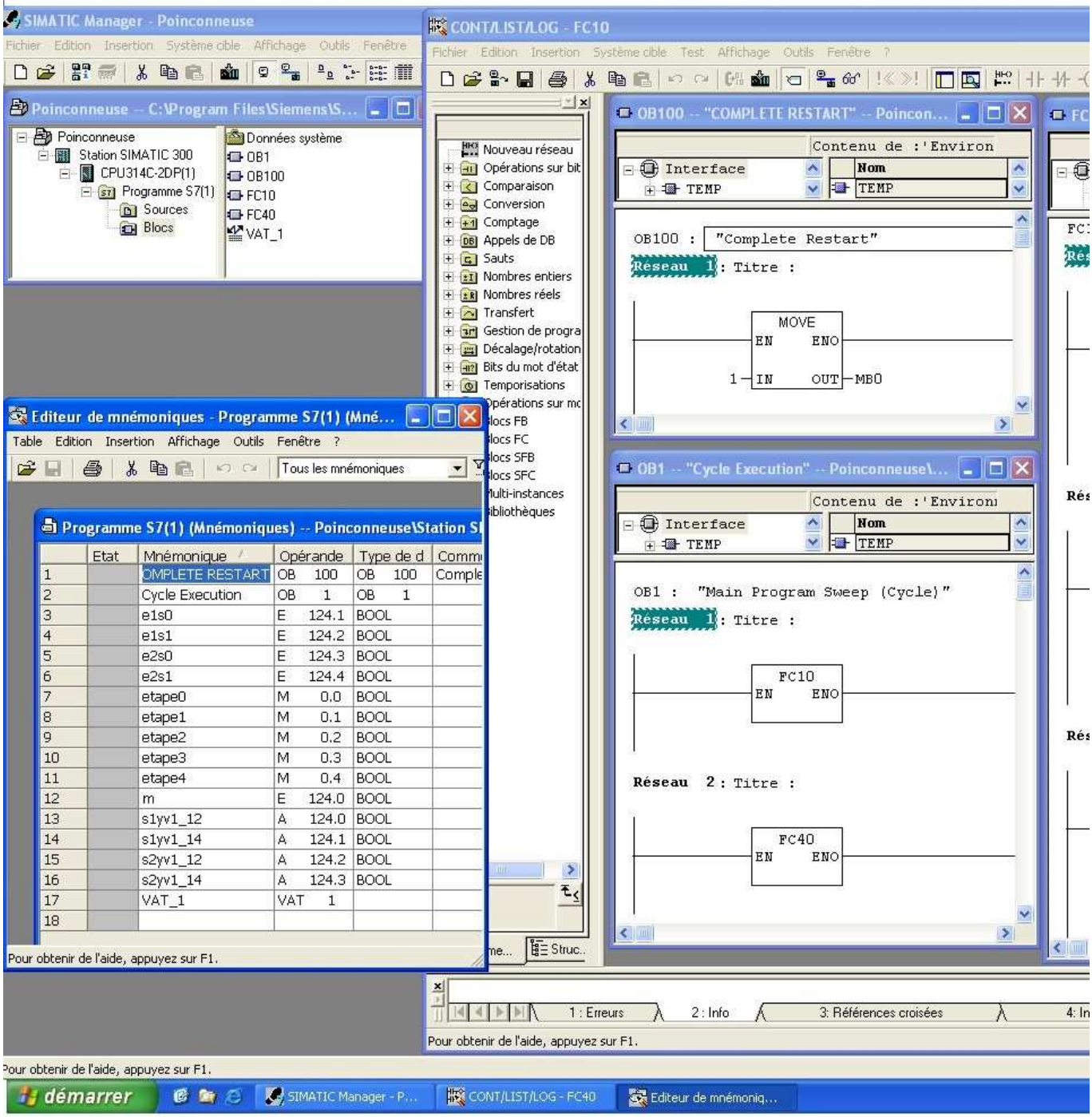
Cliquez sur « Charger »  pour transférer votre programme dans la mémoire de l'API.
Placez l'automate en RUN, avec le bouton situé sur l'unité centrale.
Testez votre programmation.

8.Impression

Sauvegardez votre projet.

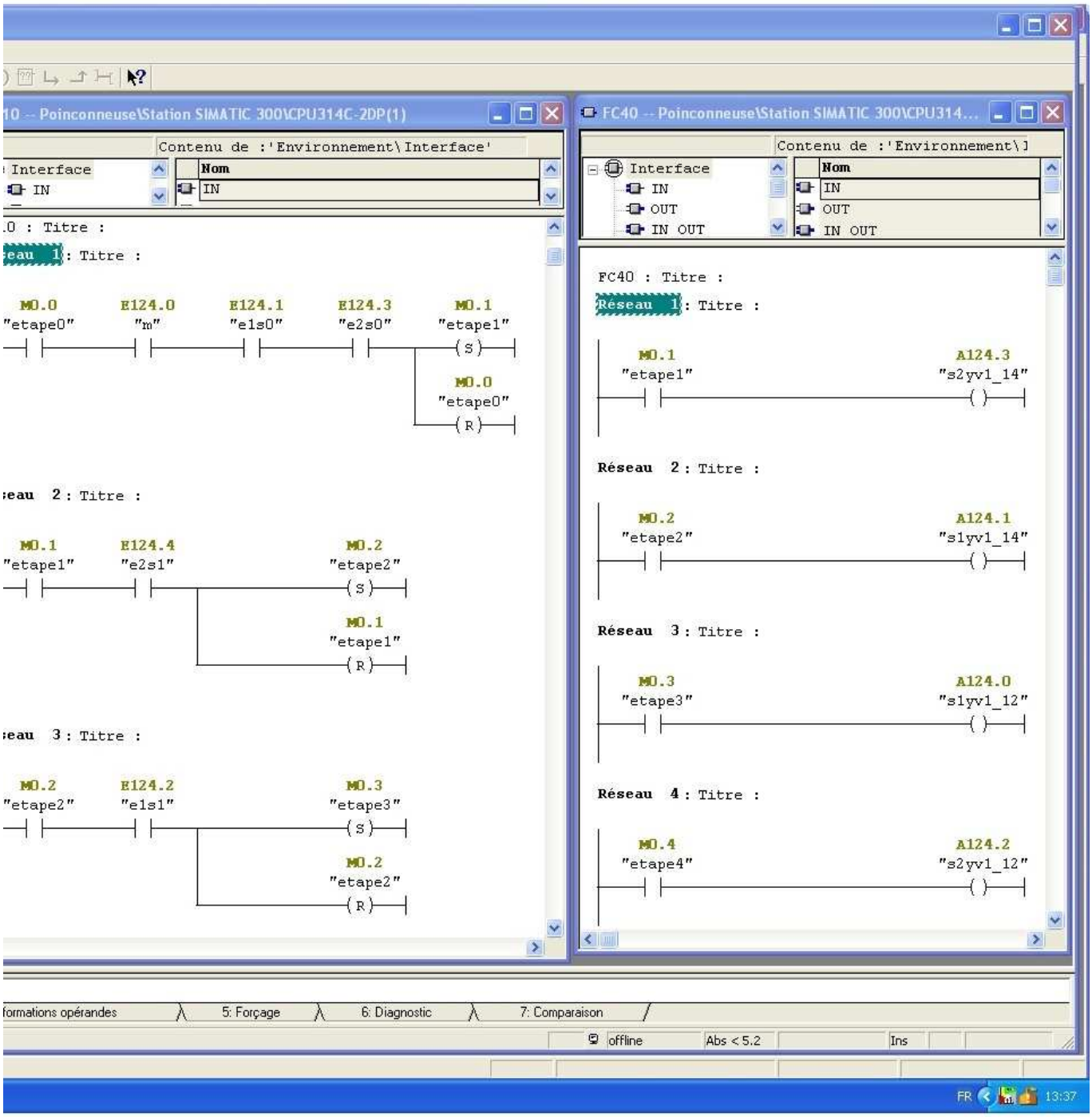
Pour imprimer, sélectionnez les blocs à imprimer, faites un clic droit et sélectionnez "Imprimer" puis "Objet".

Programmation SIEME



exemple : poinçonneuse

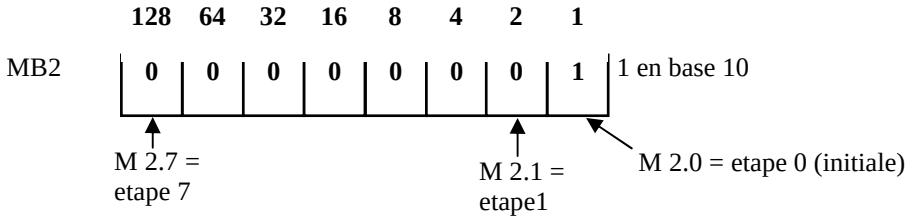
NS sous STEP 7



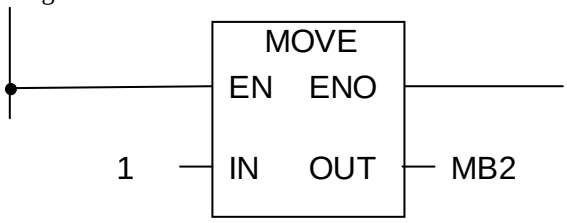
Annexes :

Programmation de OB100 :

OB100 : OB de démarrage à chaud. Il ne sera exécuté qu'une seule fois à la mise en RUN de l'API. On l'utilise pour initialiser le ou les grafcets.
Exemple : Soit un grafcet comportant 8 étapes de 0 à 7 avec l'étape 0 comme étape initiale, ce grafcet est mémorisé dans l'octet 2 (MB2).

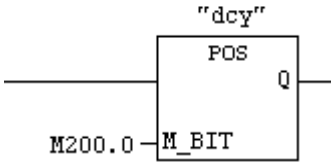


Initialiser le grafcet consiste à donner la valeur décimale « 1 » à l'octet MB2 à la mise en RUN de l'automate.
Programme de OB100 :



Utilisation des fronts montants :

Il faut utiliser le bloc POS, ce bloc utilise un mémento de front et l'entrée sur laquelle on veut détecter le front.
Exemple : Front montant de DCY



La sortie Q est à 1 au front montant de DCY.
M200.0 : mémento de front (on peut choisir n'importe quel mémento).

Le mémento de cadence (clignotement) :

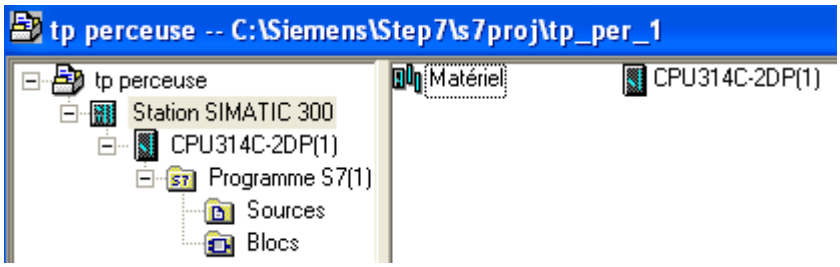
Le mémento de cadence est un octet. Chacun des bits de cet octet change d'état suivant une horloge interne.

Une durée de période et la fréquence correspondante sont affectées à chaque bit de l'octet de mémento de cadence :

Bit	7	6	5	4	3	2	1	0
Durée de période (s)	2	1,6	1	0,8	0,5	0,4	0,2	0,1
Fréquence (Hz) :	0,5	0,625	1	1,25	2	2,5	5	10

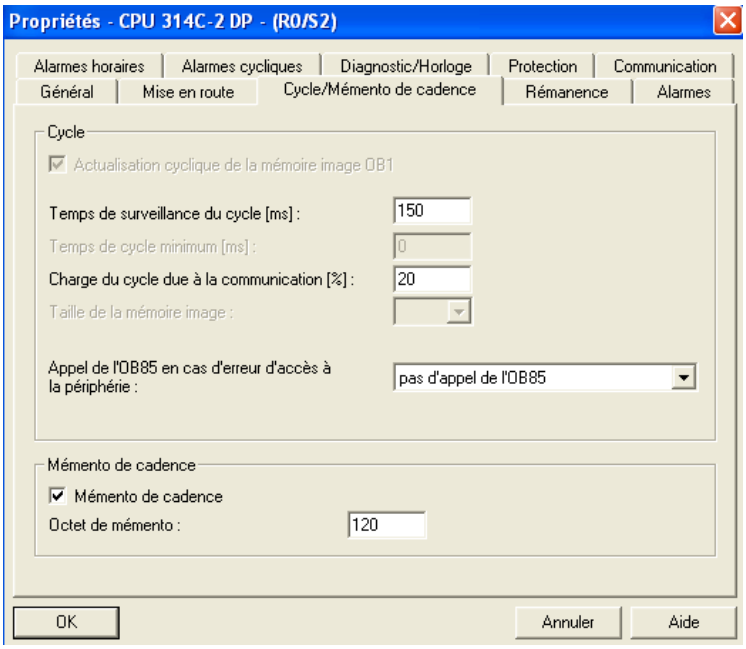
Exemple : On veut faire clignoter un voyant à la fréquence de 1Hz.

On choisit un octet de cadence en double cliquant sur Matériel

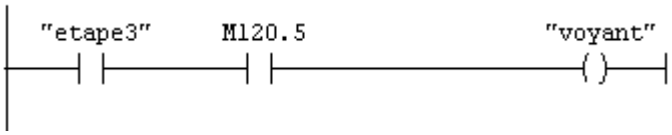


Double cliquez sur l'unité centrale et allez dans l'onglet « Cycle/Mémento de cadence »

Cochez « Mémento de cadence » et choisissez un octet (120 par exemple).



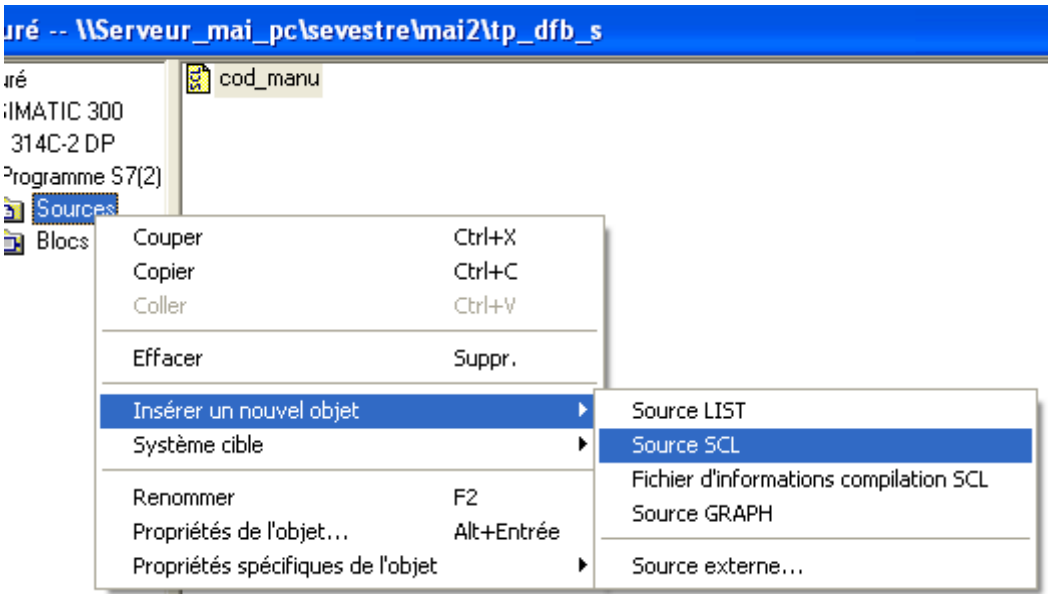
Le bit 5 de MB120 change d'état toutes les secondes



Programmation en langage structuré (SCL) :

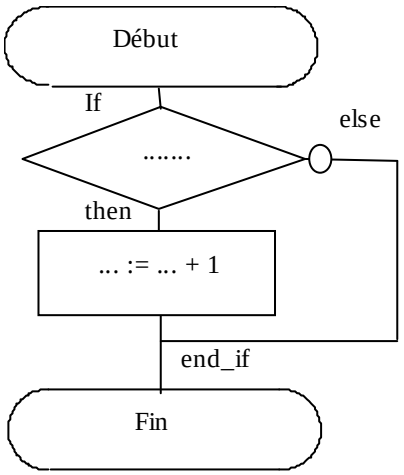
Créez un projet classique.

Dans « source », insérez une nouvelle source SCL

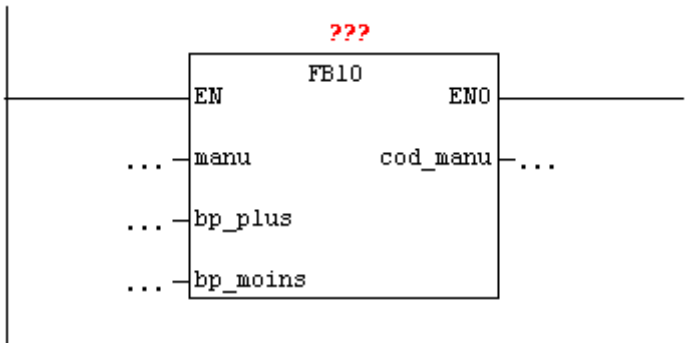


Exemple pour le bloc fonctionnel FB10 :

```
FUNCTION_BLOCK FB10

VAR_OUTPUT
  cod_manu : INT;
END_VAR

VAR_INPUT
   manu, bp_plus, bp_moins : BOOL;
end_var

IF manu AND bp_plus THEN
cod_manu:=cod_manu+1;
end_if;

END_FUNCTION_BLOCK
```



Enregistrez et compilez (Fichier, compiler). Le bloc FB10 est généré dans les blocs et peut être utilisé dans OB1.



Annexes

Le langage à contacts

Les éléments graphiques de base :

- ---| |--- Contact à fermeture
- ---| / |--- Contact à ouverture
- ---(SAVE) Sauvegarder RLG dans RB
- XOR Combinaison OU exclusif
- ---( ) Bobine de sortie
- ---(#)--- Connecteur (utilisé pour stocker des résultats intermédiaires)
- ---|NOT|--- Inverser RLG

Les opérations suivantes réagissent à un RLG égal à 1 :

- ---(S) Mettre à 1
- ---(R) Mettre à 0
- SR Bascule mise à 1, mise à 0
- RS Bascule mise à 0, mise à 1

D'autres opérations exécutent les fonctions suivantes en cas de front montant ou descendant :

- ---(N)--- Détecter front descendant
- ---(P)--- Détecter front montant
- NEG Détecter front descendant de signal
- POS Détecter front montant de signal

Les instructions numériques

Opérations de comparaison :

- CMP ? I Comparer entiers de 16 bits (16 Bit)
- CMP ? D Comparer entiers de 32 bits (32 Bit)
- CMP ? R Comparer réels

== IN1 égal à IN2
<> IN1 différent de IN2
> IN1 supérieur à IN2
< IN1 inférieur à IN2
>= IN1 supérieur ou égal à IN2
<= IN1 inférieur ou égal à IN2

Opérations d'affectation :

MOVE

Instructions arithmétiques sur entiers :

- ADD_I Additionner entiers de 16 bits
- SUB_I Soustraire entiers de 16 bits
- MUL_I Multiplier entiers de 16 bits
- DIV_I Diviser entiers de 16 bits
- ADD_DI Additionner entiers de 32 bits
- SUB_DI Soustraire entiers de 32 bits
- MUL_DI Multiplier entiers de 32 bits
- DIV_DI Diviser entiers de 32 bits
- MOD_DI Reste de division (32 bits)

sur réels :

- ADD_R Addition
- SUB_R Soustraction
- MUL_R Multiplication
- DIV_R Division

Instructions logiques :

- WAND_W ET mot
- WOR_W OU mot
- WXOR_W OU exclusif mot
- WAND_DW ET double mot
- WOR_DW OU double mot
- WXOR_DW OU exclusif double mot

Instructions de décalages :

- SHR_I Décalage vers la droite d'un entier de 16 bits
- SHR_DI Décalage vers la droite d'un entier de 32 bits
- SHL_W Décalage vers la gauche d'un mot
- SHR_W Décalage vers la droite d'un mot
- SHL_DW Décalage vers la gauche d'un double mot
- SHR_DW Décalage vers la droite d'un double mot

Instructions de conversion :

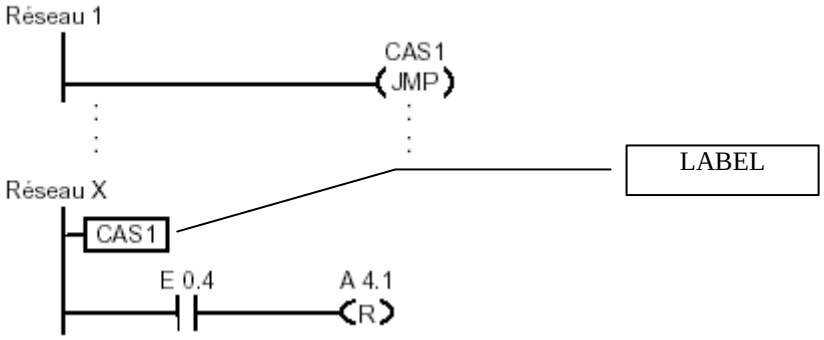
- **BCD_I Convertir nombre DCB en entier de 16 bits**
- **I_BCD Convertir entier de 16 bits en nombre DCB**
- **BCD_DI Convertir nombre DCB en entier de 32 bits**
- **I_DI Convertir entier de 16 bits en entier de 32 bits**
- **DI_BCD Convertir entier de 32 bits en nombre DCB**
- **DI_R Convertir entier de 32 bits en réel**
- **INV_I Complément à 1 d'entier de 16 bits**
- **INV_DI Complément à 1 d'entier de 32 bits**
- **NEG_I Complément à 2 d'entier de 16 bits**
- **NEG_DI Complément à 2 d'entier de 32 bits**
- **NEG_R Inverser le signe d'un nombre réel**
- **ROUND Arrondir**
- **TRUNC Tronquer à la partie entière**
- **CEIL Convertir réel en entier supérieur le plus proche**
- **FLOOR Convertir réel en entier inférieur le plus proche**

Les autres instructions

Opérations de saut :

- ---(JMP)--- Saut inconditionnel
- ---(JMP)--- Saut à l'intérieur d'un bloc si 1 (conditionnel)
- ---(JMPN)--- Saut à l'intérieur d'un bloc si 0 (conditionnel)

Exemple



Opérations sur blocs de données :

Exemple