IV. Configuration :

Utilisé pour les API modulaires, la configuration permet de préciser le nombre, le type et l'emplacement des cartes et modules utilisés.

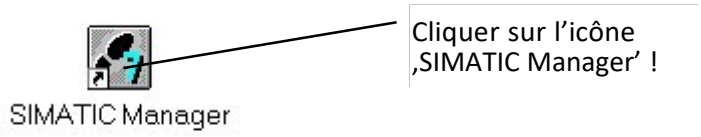
Pendant le montage de l'API S7-300, la CPU produit une configuration pratique et stocke celle-ci dans les données système (SDB).

Avec l'outil 'Configuration HW' il est possible de créer une configuration théorique dérivant de cette dernière et ainsi de configurer une nouvelle conception. De plus, on peut aussi charger une configuration existante depuis une CPU. En plus des modules comme la CPU, d'autres paramètres peuvent être prédéfinis (par ex. comportement de démarrage et de cycle d'une CPU, choix d'un octet de cadence …).

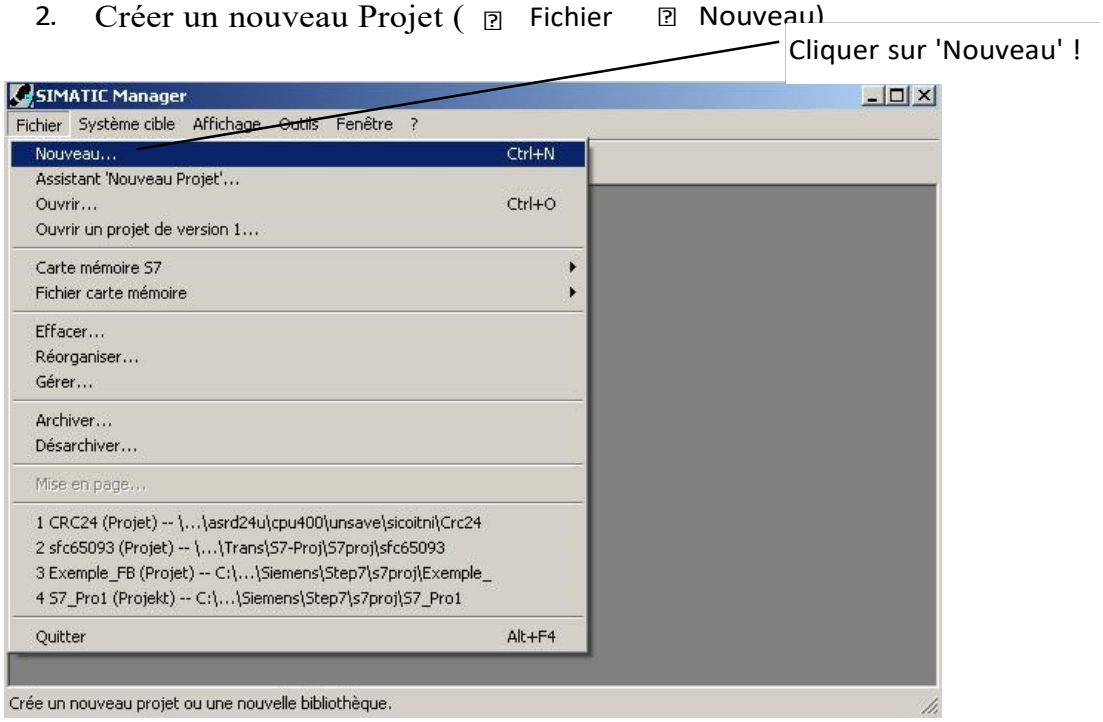
IV.1 Exemple de configuration :

L'utilisateur doit suivre les étapes suivantes pour la réalisation de la configuration matérielle de l'automate, l'écriture du Programme S7, le réglage du régulateur PID ainsi que son chargement et son exécution dans la CPU :

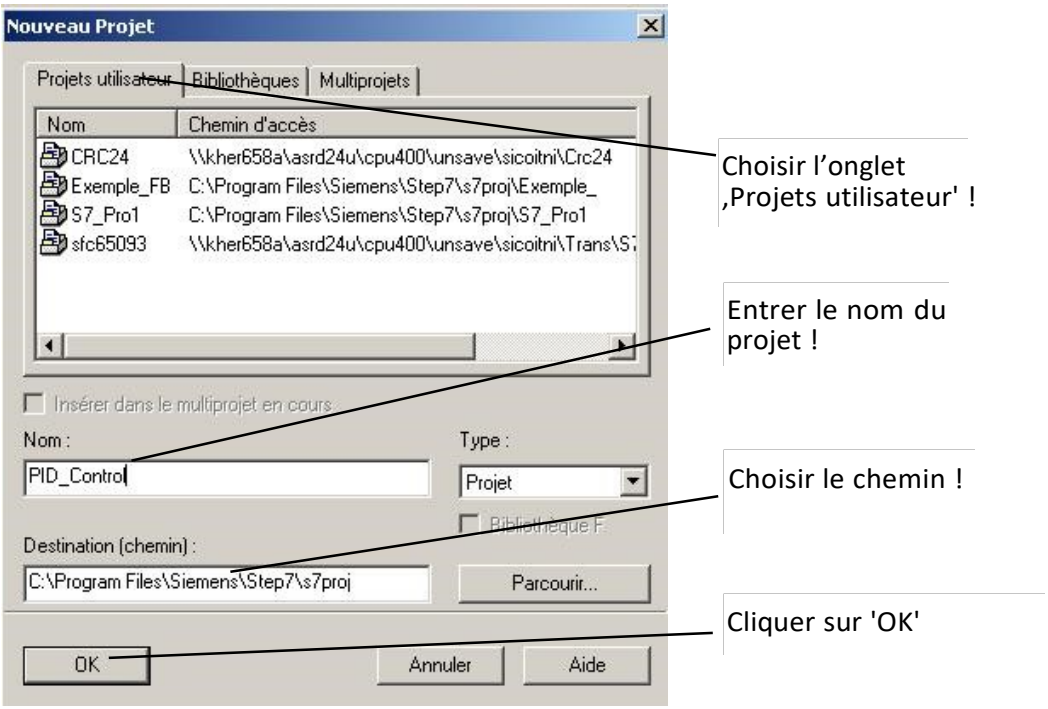
1. Ouvrir SIMATIC Manager



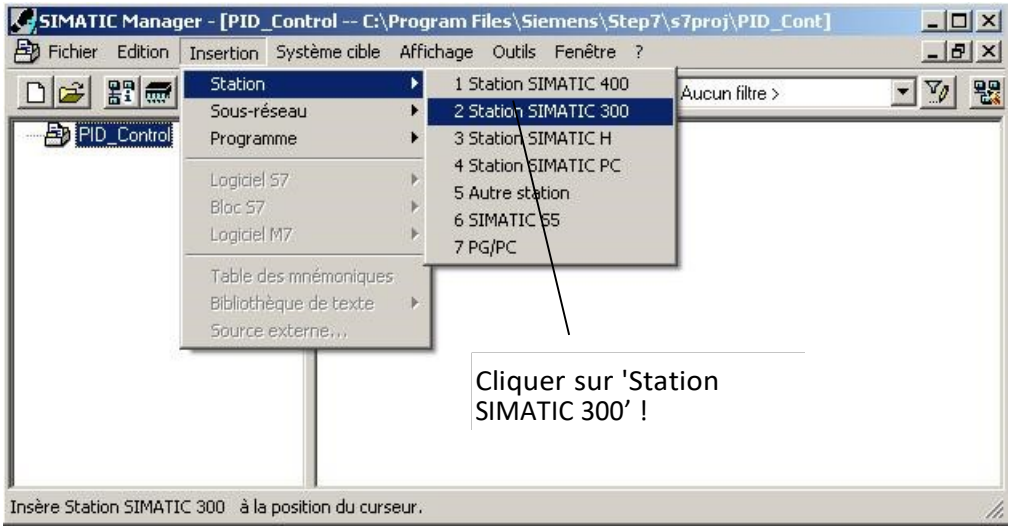
2. Créer un nouveau Projet (Fichier Nouveau)



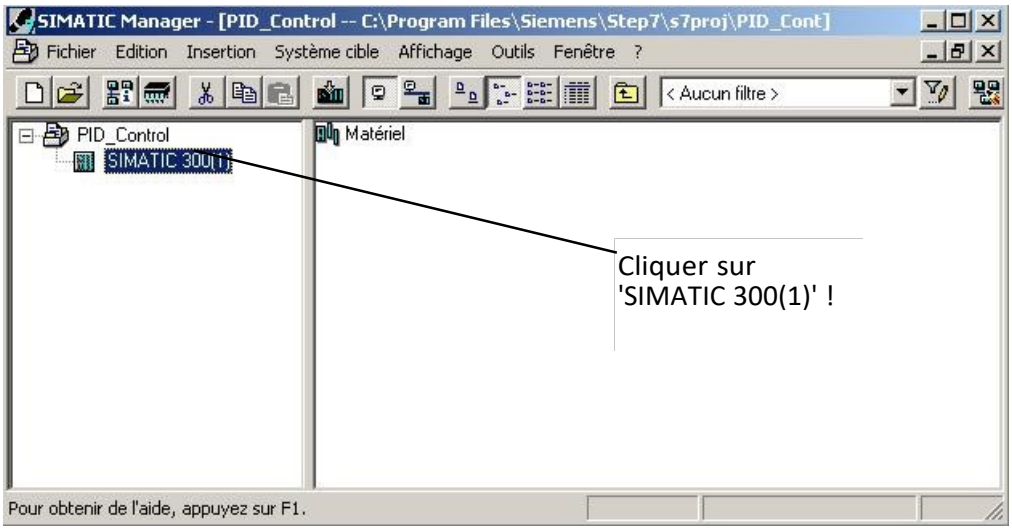
3. Créer un nouveau projet, donner le chemin de sauvegarde ainsi que le nom du projet (Projets utilisateur PID_Control OK)



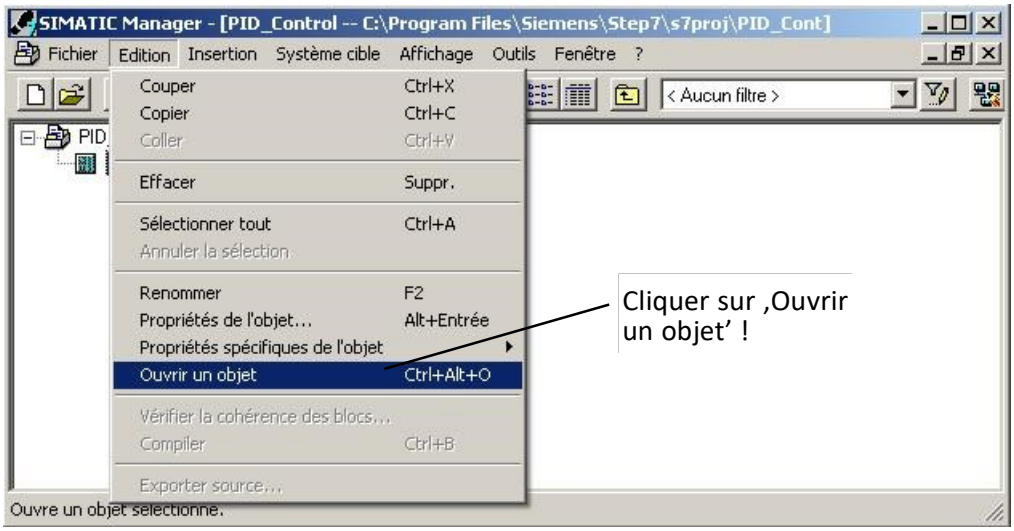
3. Insérer un objet de type ‘Station SIMATIC 300’ (Insertion Station Station SIMATIC 300)



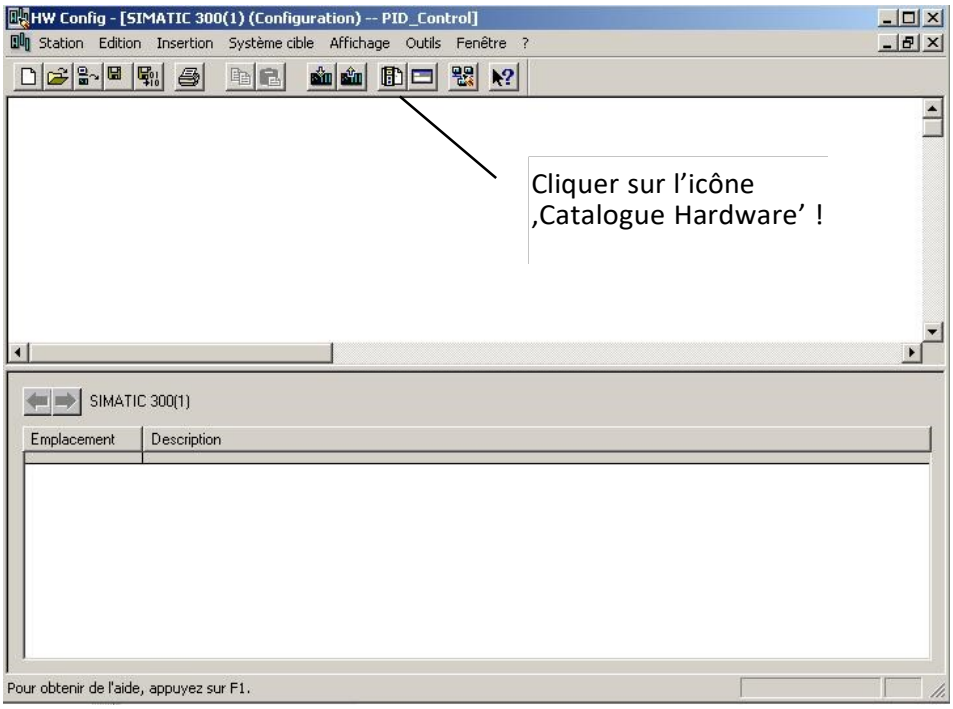
4. Sélectionner ‚SIMATIC 300 (1)’



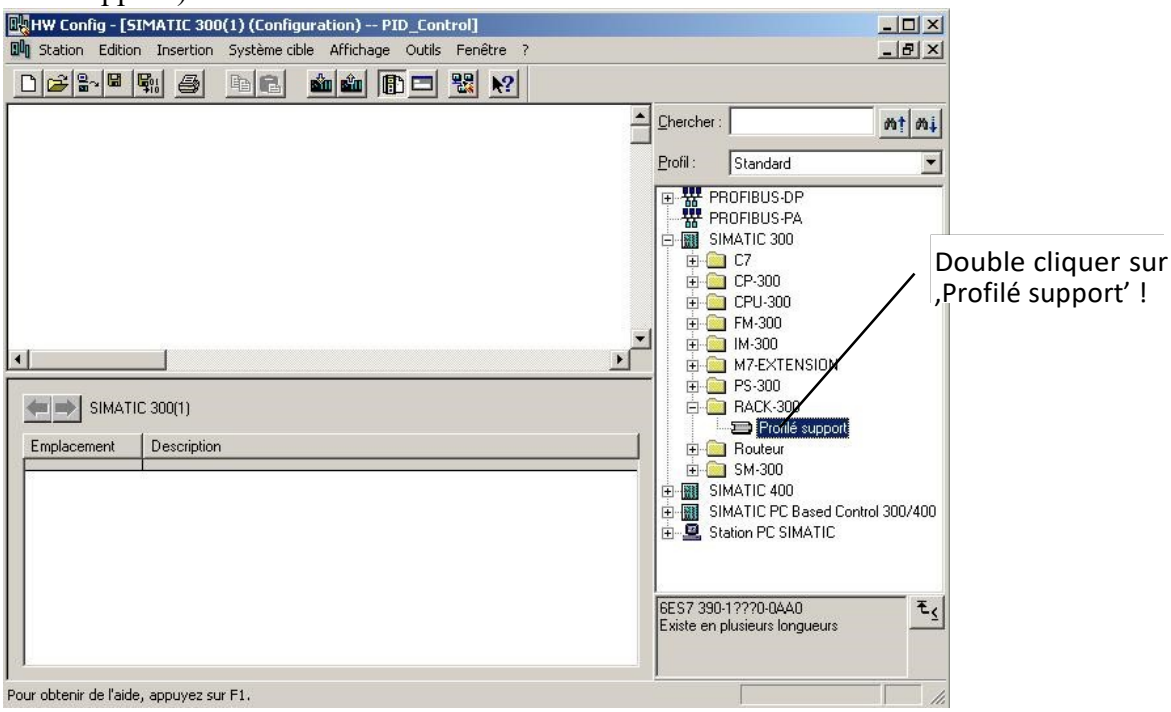
5. Ouvrir le programme de configuration matérielle (Edition Ouvrir un objet)



6. Ouvrir le catalogue Hardware. Il se compose des répertoires suivants :
- PROFIBUS-DP, SIMATIC 300, SIMATIC 400 et SIMATIC PC Based Control, ainsi que tous les supports et composants modules d'interface nécessaires à la configuration de vos installations.

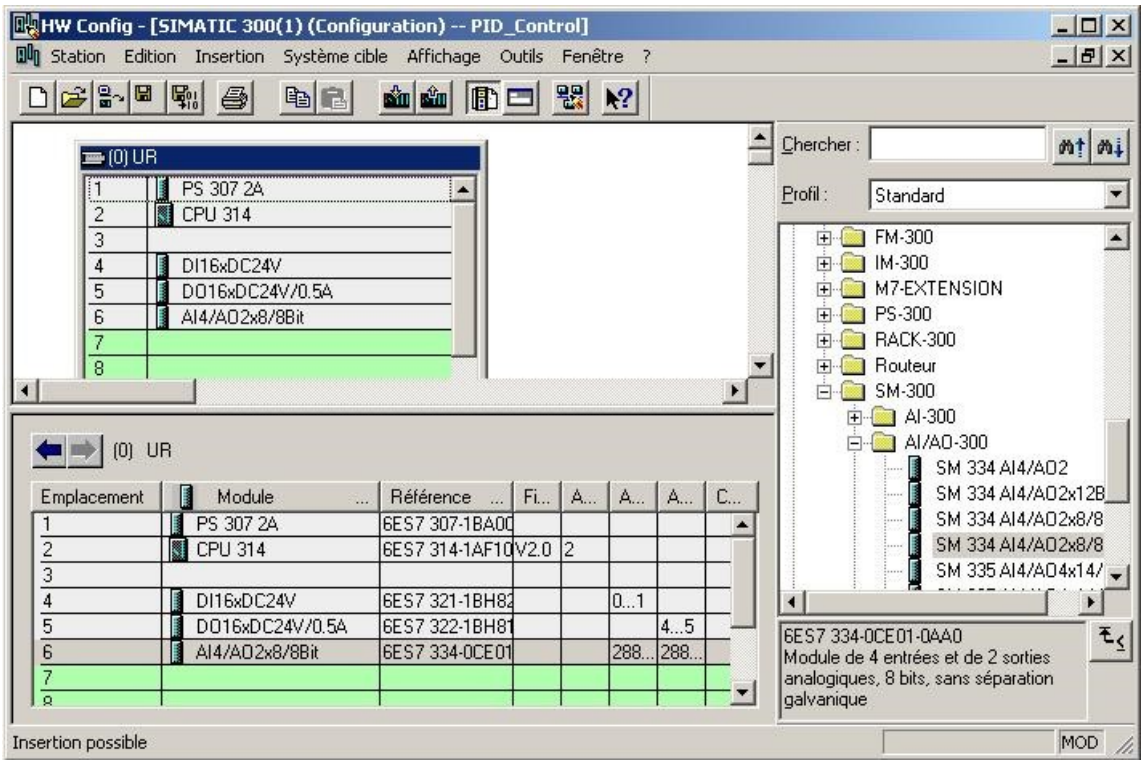


7. Insérer un support profilé (□ SIMATIC 300 □ RACK-300 □ Profilé Support).



Il apparaît alors automatiquement un tableau pour la configuration du Rack 0.

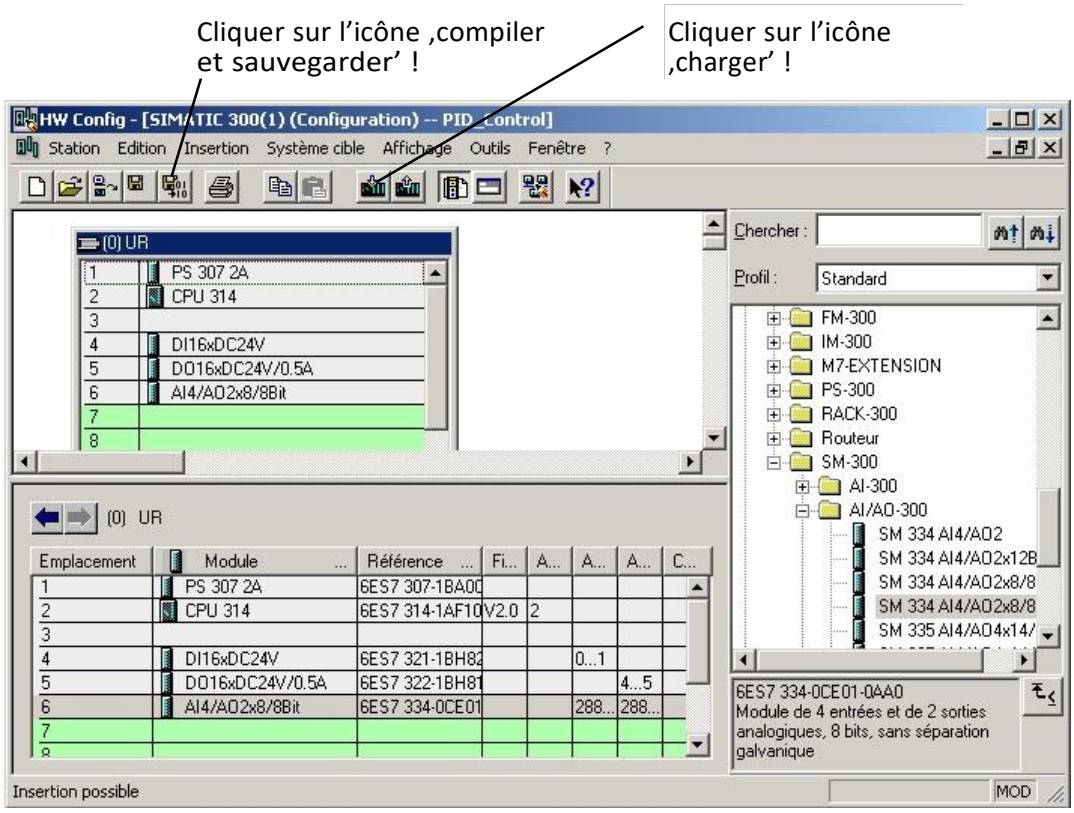
8. Choisir les composants présents sur votre rack dans le catalogue Hardware et les insérer dans le tableau de configuration. Pour se faire, il suffit de cliquer sur l'élément voulu et de le faire glisser à la place souhaitée dans le tableau.



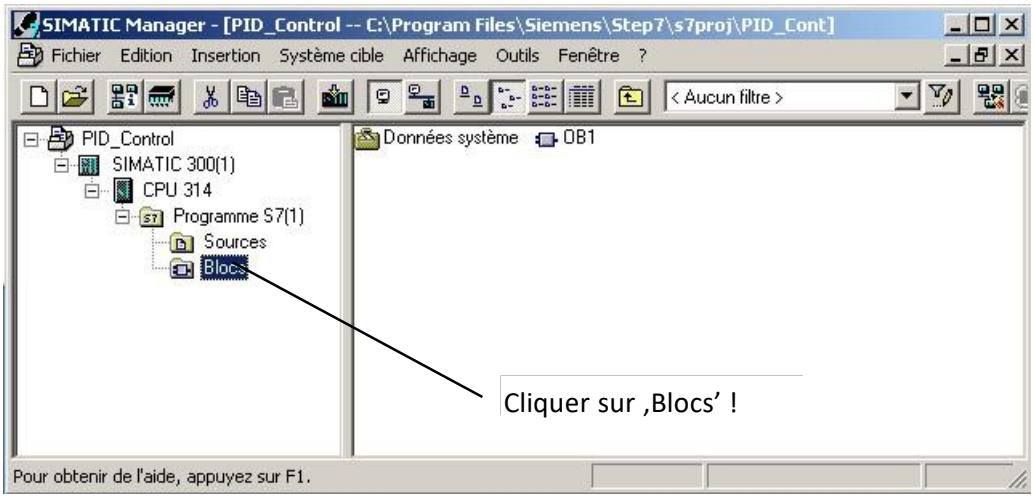
Remarque ! La place n°3 du rack est réservée pour certains composants et reste donc vide pour notre exemple.

9. Noter les adresses des éléments d'entrées / sorties. Celles-ci sont attribuées automatiquement en fonction de la place sur le rack. Il faut auparavant choisir une vue détaillée dans le menu 'Affichage' . (□ Affichage □ Détail).

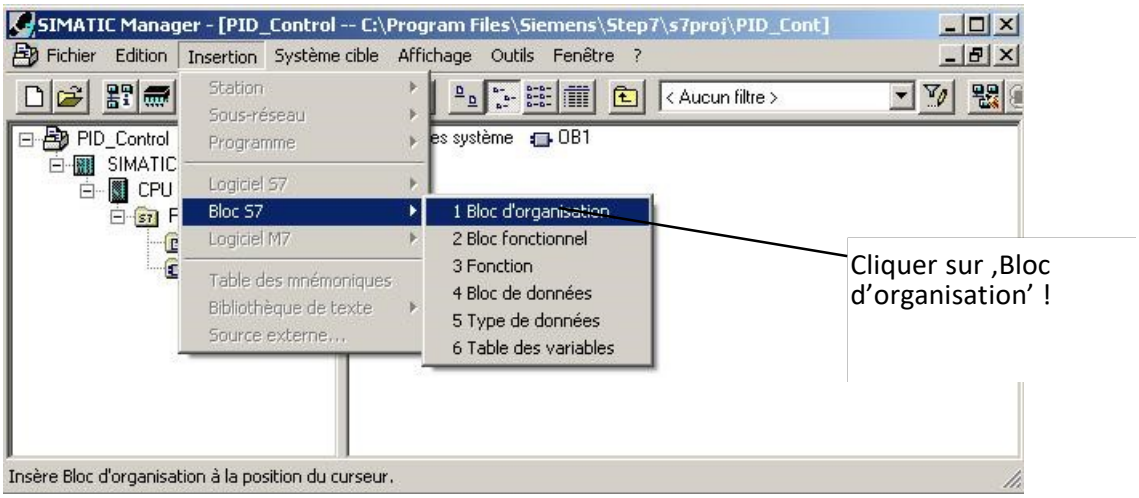
Sauvegarder et charger la configuration matérielle dans l'automate (le bouton de commande d'état doit être sur la position 'Stop' !)



10. Sélectionner le dossier Blocs dans SIMATIC Manager



11. Insérer un bloc d'organisation (□ Insertion □ Bloc S7 □ Bloc d'organisation)

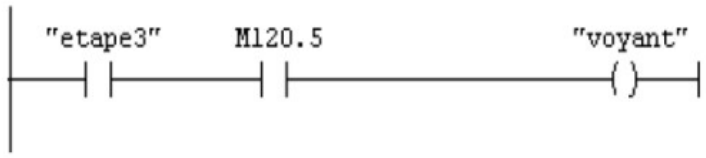


IV.2 Le mémento de cadence (clignotement) :

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 : Pour le mémento de cadence on choisit l'octet 120. Le bit 5 de MB120 change d'état toutes les secondes :



V. Structure d'un programme STEP 7

Le STEP 7 utilise des Blocs d'organisation (OB) et des Fonctions et Blocs fonctionnels (FB et FC) permettant l'écriture d'un programme dans différents modules, chaque module traitera une fonction de l'automatisme qui seront appelles par le programme principale (OB 1).

V.1 Les Blocs d'organisation OB (OB 1 à OB 122)

Les OB sont appelés par le système d'exploitation, on distingue plusieurs types :

- Ceux qui gèrent le traitement de programmes cycliques
- Ceux qui sont déclenchés par un événement,
- Ceux qui gèrent le comportement à la mise en route de l'automate programmable
- Et en fin, ceux qui traitent les erreurs.

Il existe 7 blocs d'organisation différents :

- ❖ OB cyclique (Program cycle), il s'agit de blocs traités de manière cyclique. Ce sont des blocs de code de niveau supérieur dans le programme, dans lesquels vous pouvez programmer des instructions ou appeler d'autres blocs. Le bloc cyclique OB1 est déjà créé à la création du projet.
- ❖ OB de démarrage (Startup), le traitement de ces OB est réalisé qu'une fois, lorsque la CPU passe de STOP en RUN. Le traitement de l'OB de démarrage est suivi de celui de l'OB cyclique.
- ❖ OB d'alarme temporisée (Time delay interrupt), ils interrompent le traitement cyclique du programme après écoulement d'un temps défini. Vous indiquez le temps de retard dans le paramètre d'entrée de l'instruction étendue "SRT_DINT".
- ❖ OB d'alarme cyclique (Cyclic interrupt), ils interrompent le traitement cyclique du programme à intervalles de temps définis. Vous pouvez spécifier les intervalles de temps dans cette boîte de dialogue ou dans les propriétés de l'OB.
- ❖ OB d'alarme du processus (Hardware interrupt), ils interrompent le traitement cyclique du programme en réponse à un événement matériel. Vous définissez l'événement matériel dans les propriétés du matériel.
- ❖ OB d'erreur de temps (Time error interrupt), ils interrompent le traitement cyclique du programme lorsque le temps de cycle maximum est dépassé. Vous définissez le temps de cycle maximum dans les propriétés de la CPU.
- ❖ OB d'alarme de diagnostic (Diagnostic error interrupt), ils interrompent le traitement cyclique du programme lorsque le module pour lequel l'alarme de diagnostic a été activée détecte une erreur.

Vous retrouverez ces informations en ajoutant un OB à votre programme.

Ces blocs déterminent la structure du programme utilisateur. Les OB sont directement appelés par le système d'exploitation de la CPU en réaction à un événement (à condition toutefois de les avoir programmé et insérés dans l'automate).

- ➢ **Programme cyclique OB 1**

Lors d'une exécution normale de programme, les traitements se font de façon cyclique.

L'exécution du programme contenu dans l'OB 1 est démarrée une fois par cycle (quand il est fini, il recommence). On peut se servir de l'OB 1 pour appeler des blocs de type FC ou FB.

Le bloc OB1 est généré automatiquement lors de la création d'un projet. C'est le programme cyclique appelé par le système d'exploitation.

V.2 Les Fonctions et Blocs fonctionnels FC et FB

Le dossier bloc, cité auparavant, contient les blocs que l'on doit charger dans la CPU pour réaliser la tâche d'automatisation, il englobe :

- Les blocs de code (OB, FB, SFB, FC, SFC) qui contiennent les programmes,
- Les blocs de données DB d'instance et DB globaux qui contiennent les paramètres du programme.

a. Les blocs fonctionnels (FB), (SFB)

Le **FB** est un sous-programme écrit par l'utilisateur et exécuté par des blocs de code. On lui associe un bloc de données d'instance relatif à sa mémoire et contenant ses paramètres.

Les SFB système sont utilisés pour des fonctions spéciales intégrées dans la CPU.

b. Les fonction (FC), (SFC)

La **FC** contient des routines pour les fonctions fréquemment utilisées. Ces fonctions sont des blocs de code sans mémoire et sauvegarde ses variables temporaires dans la pile de données locales. Cependant elle peut faire appel à des blocs de données globaux pour la sauvegarde de ses données.

Les blocs fonctionnels **SFC** sont des blocs de code qui sauvegardent en permanence leurs valeurs dans des blocs de données d'instance afin qu'il soit possible d'y accéder même après le traitement du bloc.

Les SFC sont utilisées pour des fonctions spéciales, intégrées dans la CPU S7, elle est appelée à partir du programme.

V.3 Blocs de données (DB)

Les blocs de données sont des zones de données dans le programme utilisateur qui contiennent des données utilisateur.

Vous pouvez sélectionner 2 types de bloc :

- Un bloc de données global, qui est indépendant de tout autre bloc. (Par exemple nous programmons un DB Global pour toutes les données d’échange entre API et HMI).
- Un bloc de données d'instance, qui dépend d’un bloc fonctionnel, il s’agit de la mémoire des valeurs du bloc dont il dépend.