SIEMENS

Dossier de formation SCE

Siemens Automation Cooperates with Education | 05/2017

Module 032-200 TIA Portal

Principes de base de la programmation de FB avec SIMATIC S7-1500

Cooperates with Education

Automation

SIEMENS

Packages SCE pour formateurs adaptés à ces dossiers de formation

Automates SIMATIC

- **SIMATIC ET 200SP Open Controller CPU 1515SP PC F et HMI RT SW**
  N° d'article: 6ES7677-2FA41-4AB1
- **SIMATIC ET 200SP Distributed Controller CPU 1512SP F-1 PN Safety**
  N° d'article: 6ES7512-1SK00-4AB2
- **SIMATIC CPU 1516F PN/DP Safety**
  N° d'article : 6ES7516-3FN00-4AB2
- **SIMATIC S7 CPU 1516-3 PN/DP**
  N° d'article: 6ES7516-3AN00-4AB3
- **SIMATIC CPU 1512C PN avec logiciel et PM 1507**
  N° d'article : 6ES7512-1CK00-4AB1
- **SIMATIC CPU 1512C PN avec logiciel, PM 1507 et CP 1542-5 (PROFIBUS)**
  N° d'article : 6ES7512-1CK00-4AB2
- **SIMATIC CPU 1512C PN avec logiciel**
  N° d'article : 6ES7512-1CK00-4AB6
- **SIMATIC CPU 1512C PN avec logiciel et CP 1542-5 (PROFIBUS)**
  N° d'article : 6ES7512-1CK00-4AB7

SIMATIC STEP 7 Software for Training

- **SIMATIC STEP 7 Professional V14 SP1- Licence monoposte**
  N° d'article : 6ES7822-1AA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1 - Licence salle de classe 6 postes**
  N° d'article : 6ES7822-1BA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1- Licence de mise à niveau 6 postes**
  N° d'article : 6ES7822-1AA04-4YE5
- **SIMATIC STEP 7 Professional V14 SP1 - Licence salle de classe 20 postes**
  N° d'article : 6ES7822-1AC04-4YA5

Veuillez noter que les packages pour formateurs ont parfois été remplacés par de nouveaux packages.
Vous pouvez consulter les packages SCE actuellement disponibles sous : siemens.com/sce/tp

Formations

Pour les formations Siemens SCE régionales, contactez votre interlocuteur SCE régional siemens.com/sce/contact

Plus d'informations sur le programme SCE

siemens.com/sce

Remarque d’utilisation

Les dossiers de formation SCE pour la solution d'automatisation cohérente Totally Integrated Automation (TIA) ont été spécialement créés pour le programme "Siemens Automation Cooperates with Education (SCE)" à des fins de formation pour les instituts publics de formation et de R&D. Siemens AG n’assume aucune responsabilité quant au contenu.

Cette documentation ne peut être utilisée que pour une première formation aux produits/systèmes Siemens. Autrement dit elle peut être copiée, en partie ou en intégralité, pour être distribuée aux participants à la formation afin qu'ils puissent l'utiliser dans le cadre de leur formation. La diffusion et la duplication de cette documentation, l'exploitation et la communication de son contenu sont autorisées au sein d’instituts publics de formation et de formation continue.

Toute exception requiert au préalable l’autorisation écrite de la part des interlocuteurs Siemens AG : Monsieur Roland Scheuerer roland.scheuerer@siemens.com.

Toute violation de cette règle expose son auteur au versement de dommages et intérêts. Tous droits réservés, en particulier en cas de délivrance de brevet ou d'enregistrement d'un modèle déposé.

Il est expressément interdit d’utiliser cette documentation pour des cours dispensés à des clients industriels. Tout usage de cette documentation à des fins commerciales est interdit.

Nous remercions l'Université technique de Dresde, en particulier Prof. Dr.-Ing. Leon Urbas et l’entreprise Michael Dziallas Engineering ainsi que toutes les personnes ayant contribué à la réalisation des dossiers de formation.

Sommaire

PRINCIPES DE BASE DE LA PROGRAMMATION DE FB

1 Objectif

Ce chapitre vous présente les éléments de base d'un programme API – les ***blocs d'organisation (OB),*** les ***fonctions (FC)***, **blocs fonctionnels (FB)** et **blocs de données (DB)**. Il présente également la programmation des fonctions et des blocs fonctionnels ***réutilisables***. Vous découvrez le langage de programmation ***logigramme (LOG)*** et l'utilisez pour programmer bloc fonctionnel fonction FB1 et un bloc d'organisation OB1.

Les automates SIMATIC S7 énumérés au chapitre 3 peuvent être utilisés.

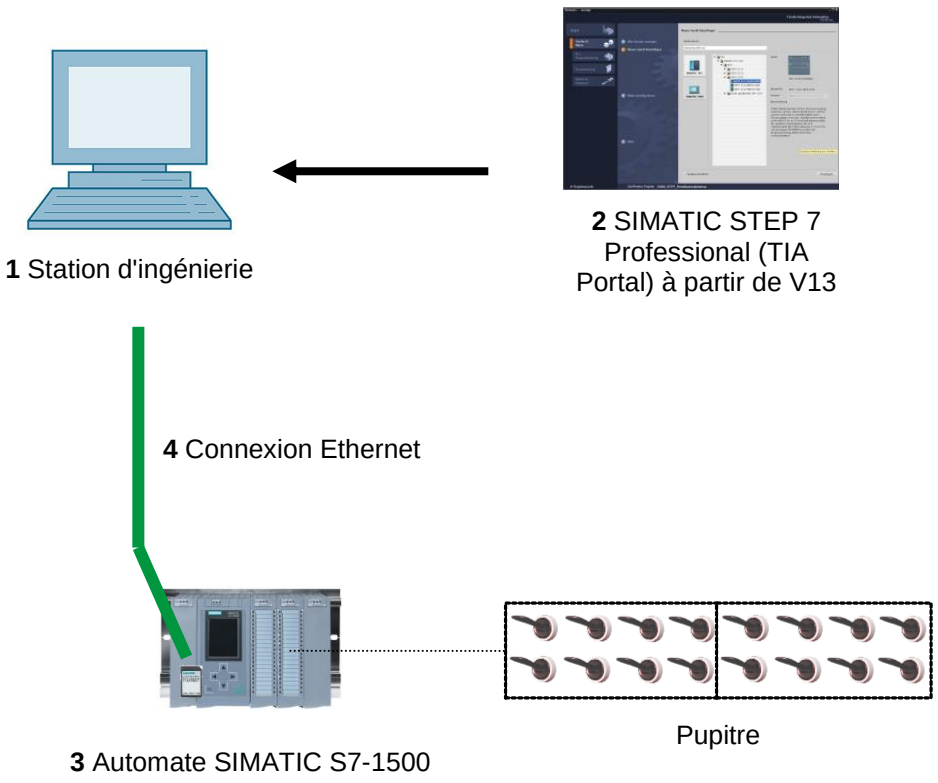
2 Conditions requises

Ce chapitre s'appuie sur la configuration matérielle de SIMATIC S7 CPU1516F-3 PN/DP, mais il peut aussi s'appliquer à d'autres configurations matérielles possédant des entrées et sorties TOR. Pour ce chapitre, vous pouvez par ex. utiliser le projet suivant :

SCE_FR_012_101__Configuration matérielle_CPU1516F.zap13

3 Configurations matérielles et logicielles requises

1. Station d'ingénierie : Le matériel et le système d'exploitation sont la condition de base (pour plus d'informations, voir le fichier Lisezmoi sur les DVD d'installation de TIA Portal)
2. Logiciel SIMATIC STEP 7 Professional dans TIA Portal – à partir de V13
3. Automate SIMATIC S7-1500/S7-1200/S7-300, par exemple CPU 1516F-3 PN/DP – à partir du firmware V1.6 avec carte mémoire et 16DI/16DO ainsi que 2AI/1AO
   Remarque : les entrées TOR doivent être mises en évidence sur un pupitre.
4. Connexion Ethernet entre la station d'ingénierie et l'automate



4 Théorie

4.1 Système d'exploitation et programme utilisateur

Chaque automate (CPU) contient un ***système d'exploitation*** qui organise toutes les fonctions et processus de la CPU n'étant pas liés à une tâche d'automatisation spécifique. Font partie des tâches du système d'exploitation :

- Déroulement du démarrage (à chaud)
- Actualisation de la mémoire image des entrées et de la mémoire image des sorties
- Appel cyclique du programme utilisateur
- Acquisition des alarmes et appels des OB d'alarme
- Détection et traitement des erreurs
- Gestion des zones de mémoire

Le système d'exploitation est un composant de la CPU et est déjà installé dans la CPU à la livraison.

Le ***programme utilisateur*** contient toutes les fonctions requises pour le traitement de tâches d'automatisation spécifiques. Font partie des fonctions du programme utilisateur :

- Vérification des conditions préalables au démarrage (à chaud) à l'aide d'OB de démarrage
- Traitement des données de process, c'est-à-dire pilotage des signaux de sortie en fonction de l'état des signaux d'entrée
- Réaction aux alarmes et aux entrées d'alarme
- Traitement des perturbations dans l'exécution normale du programme

4.2 Blocs d'organisation

Les blocs d'organisation (OB) constituent l'interface entre le système d'exploitation de l'automate (CPU) et le programme utilisateur. Ils sont appelés par le système d'exploitation et gèrent les opérations suivantes :

- Traitement cyclique du programme (p.ex. OB1)
- Comportement au démarrage de l'automate
- Traitement du programme déclenché par alarme
- Traitement des erreurs

Un projet doit contenir au minimum un ***bloc d'organisation pour le traitement cyclique du programme***. Un OB est appelé par un ***événement de démarrage***, comme indiqué à la Figure 1. Les OB ont des priorités définies, afin que par ex. un OB82 puisse interrompre l'OB1 cyclique pour traiter les erreurs.

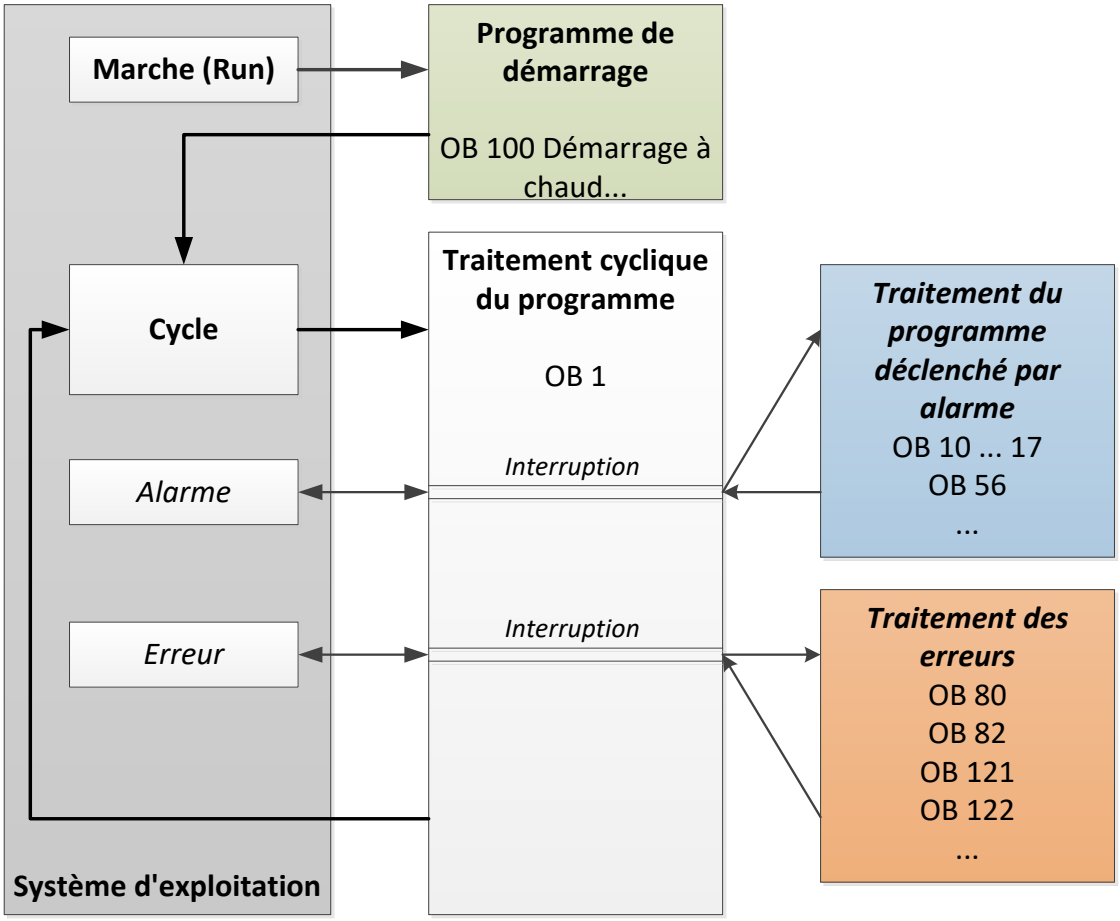


Figure 1 : Événements de démarrage du système d'exploitation et appels d'OB

Une fois que l'évènement de démarrage est intervenu, les réactions suivantes sont possibles :

- Si l'événement est affecté à un OB, cet événement lance l'exécution de l'OB assigné. Si la priorité de l'OB affecté est supérieure à celle de l'OB qui vient d'être exécuté, cet OB est exécuté immédiatement (Interrupt). Si ce n'est pas le cas, l'opération est différée jusqu'à ce que l'OB de priorité supérieure soit exécutée.
- Si l'événement n'est affecté à aucun OB, la réaction système par défaut est exécutée.

Le Tableau 1 donne pour une SIMATIC S7-1500 quelques exemples d'événements de démarrage dont les numéros d'OB possibles et la réaction système par défaut devraient ne pas se trouver dans l'automate.

Événement de démarrage	**Numéros d'OB possibles**	**Réaction système par défaut**
Mise en route	100, ≥ 123	Ignorer
Programme cyclique	1, ≥ 123	Ignorer
Alarme horaire	10 à 17, ≥ 123	-
Alarme de mise à jour	56	Ignorer
Temps de surveillance du cycle dépassé une fois	80	STOP
Alarme de diagnostic	82	Ignorer
Erreur de programmation	121	STOP
Erreur d'accès à la périphérie	122	Ignorer

Tableau 1 : Numéros de l'OB pour différents événements de démarrage

4.3 Mémoire image et traitement cyclique du programme

Si les entrées (E) et les sorties (A) sont adressées dans le programme utilisateur cyclique, les états des signaux ne sont pas interrogés directement par les modules d'entrées/sorties, mais la zone de mémoire de la CPU est accédée. Cette zone de mémoire contient une image des états des signaux et est appelée **mémoire image**.

Le traitement cyclique du programme s'effectue comme suit :

1. Au début du programme cyclique, le système demande si les entrées doivent ou non être sous tension. L'état de ces entrées est enregistré dans la **mémoire image des entrées (MIE)**. Si l'entrée est sous tension, l'information 1 ou "High" sera enregistrée. Si l'entrée n'est pas sous tension, l'information 0 ou "Low" sera enregistrée.
2. Le processeur exécute le programme stocké dans le bloc d'organisation cyclique. L'information d'entrée requise à cet effet est prélevée dans la **mémoire image des entrées (MIE)** lue auparavant et les résultats logiques sont écrits dans une **mémoire image des sorties (MIS)**.
3. A la fin du cycle, la **mémoire image des sorties (MIS)** est transmise sous forme d'état de signal aux modules de sortie et activée ou désactivée. La procédure reprend ensuite à partir du point 1.

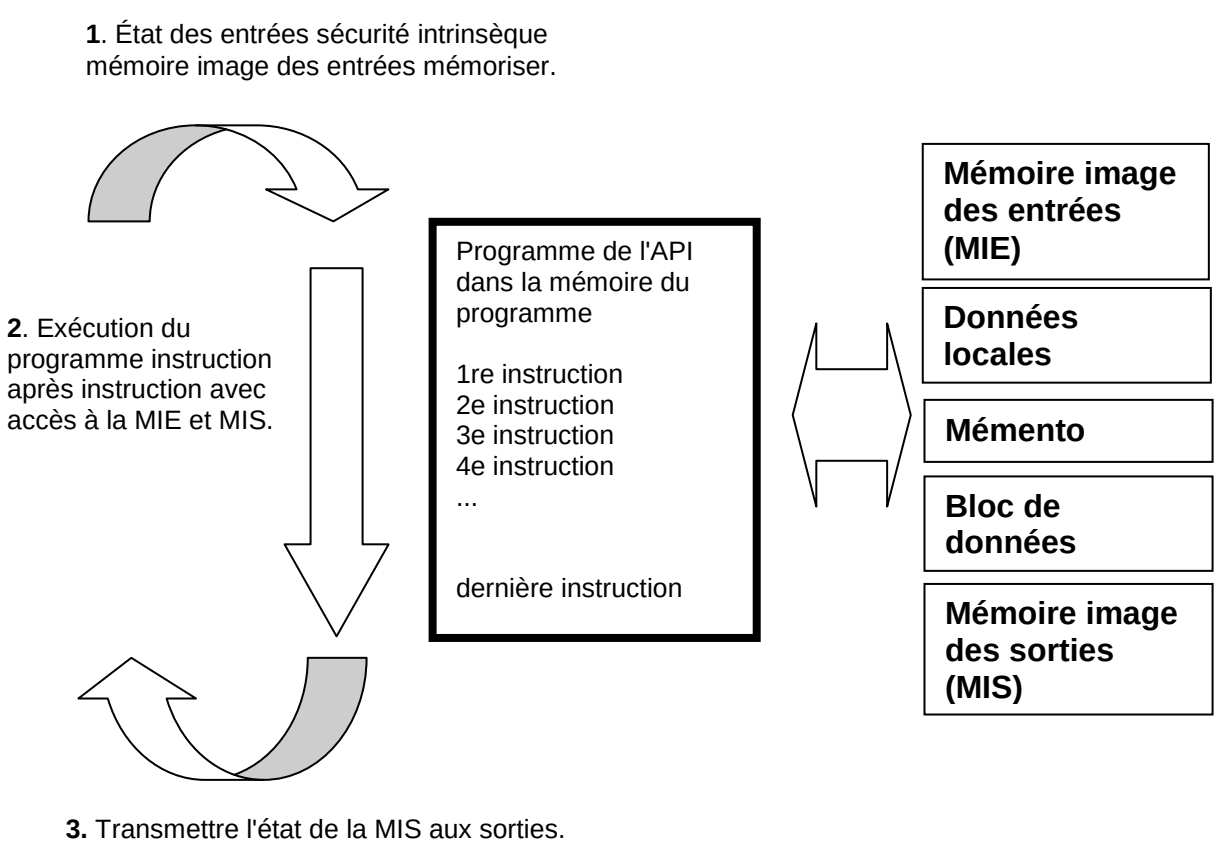


Figure 2 : Traitement cyclique du programme

Remarque : *le temps requis par le processeur pour l'exécution du programme s'appelle le temps de cycle. Ce dernier dépend entre autres du nombre et du type d'instructions, ainsi que de la puissance du processeur de l'automate.*

4.4 Fonctions

Les fonctions (FC) sont des blocs de code sans mémoire. Elles n'ont ***pas de mémoire de données*** dans laquelle il est possible d'enregistrer les valeurs de paramètres de bloc. C'est pourquoi tous les paramètres d'interface doivent être connectés lors de l'appel d'une fonction. Pour enregistrer les données durablement, il convient de créer auparavant des blocs de données globaux.

Une fonction contient un programme qui est toujours exécuté quand un autre bloc de code appelle cette fonction.

Les fonctions peuvent par exemple servir dans les cas suivants :

- fonctions mathématiques qui fournissent un résultat en fonction des valeurs d'entrée
- fonctions technologiques comme les commandes uniques avec combinaisons binaires

Une fonction peut également être appelée plusieurs fois à divers endroits du programme.

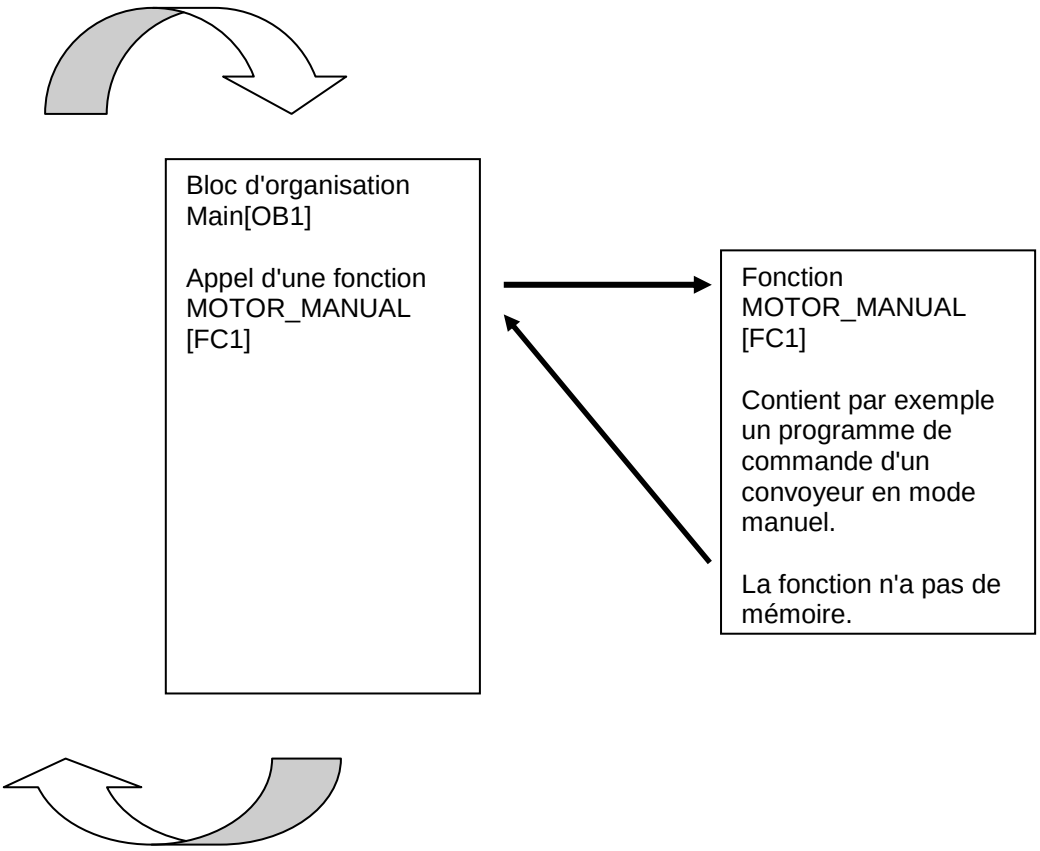


Figure 3 : Fonction avec appel provenant du bloc d'organisation Main[OB1]

4.5 Blocs fonctionnels et blocs de données d'instance

Les blocs fonctionnels sont des blocs de code qui mémorisent durablement leurs variables d'entrée, de sortie, d'entrée/sortie et les variables statiques dans des blocs de données d'instance, afin qu'il soit ***possible d'y accéder même après le traitement de blocs***. Pour cette raison, ils sont aussi appelés blocs avec mémoire.

Les blocs fonctionnels peuvent aussi travailler avec des variables temporaires. Cependant, les variables temporaires ne sont pas enregistrées dans la DB d'instance mais disponibles uniquement le temps d'un cycle.

Les FB sont utilisés pour des tâches qui ne peuvent être mises en œuvre avec des fonctions :

- Toujours quand les temporisations et les compteurs sont nécessaires dans un bloc.
- Toujours quand une information doit être enregistrée dans le programme. Par ex. un indicatif de mode de fonctionnement avec un bouton.

Les blocs fonctionnels sont toujours exécutés quand un bloc fonctionnel est appelé par un autre bloc de code. Un bloc de fonction peut aussi être appelé plusieurs fois à divers endroits du programme. Ceci facilite la programmation de fonctions complexes et répétitives.

Un appel d'un bloc fonctionnel est désigné par le terme "instance". Pour chaque instance d'un FB, une zone mémoire lui est affectée, contenant les données utiles au traitement du bloc. Cette mémoire est fournie par des blocs de données que le logiciel génère automatiquement.

Il est également possible de fournir de la mémoire pour plusieurs instances grâce un bloc de données en **multi-instance**. La taille maximale des DB d'instance varie selon la CPU. Les variables déclarées dans le bloc fonctionnel déterminent la structure du bloc de données d'instance.

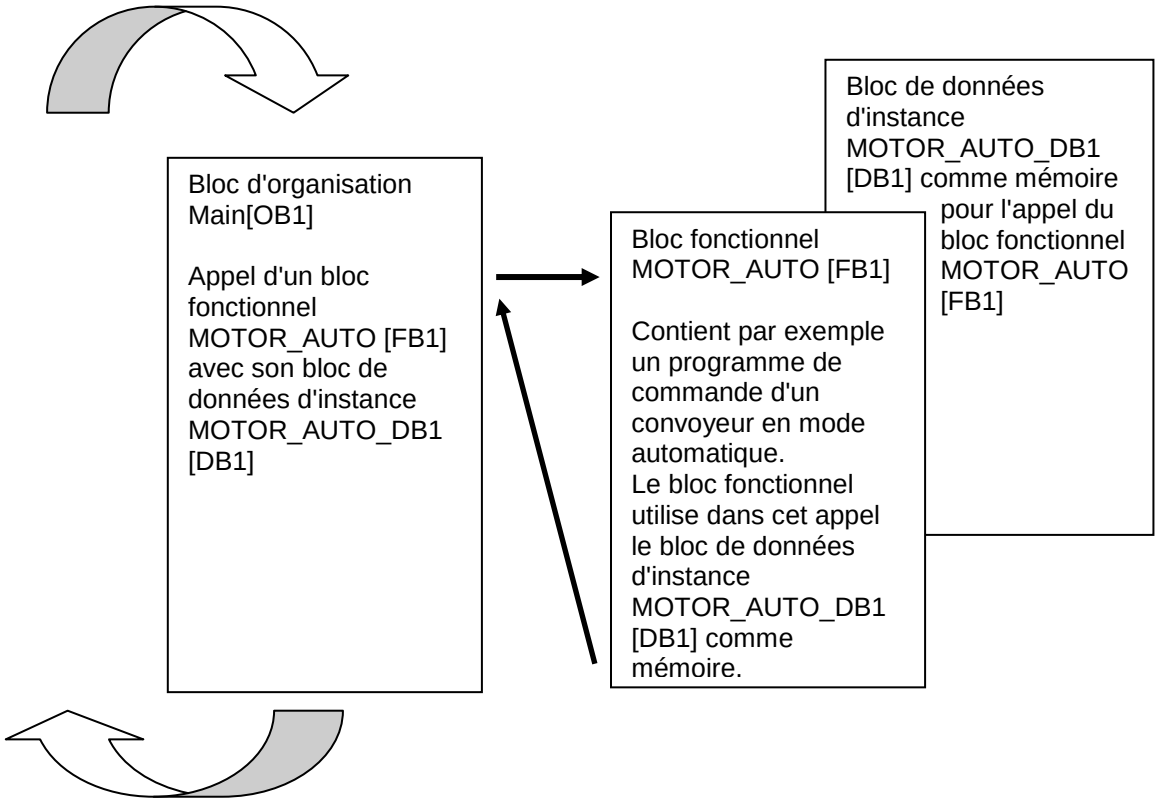


Figure 4 : Bloc fonctionnel et instance avec appel provenant du bloc d'organisation Main[OB1]

4.6 Blocs de données globaux

Contrairement aux blocs de code, les blocs de données ne contiennent pas d'instructions, mais ils sont utilisés pour enregistrer les données utilisateur.

Les blocs de données contiennent donc des données variables qui sont utilisées dans le programme utilisateur. La structure des blocs de données globaux peut être définie au choix.

Les blocs de données globaux enregistrent des données qui peuvent être utilisées par ***tous les autres blocs*** (voir figure 5). Seul le bloc fonctionnel correspondant doit accéder aux blocs de données d'instance. La taille maximale des blocs de données varie selon la CPU.

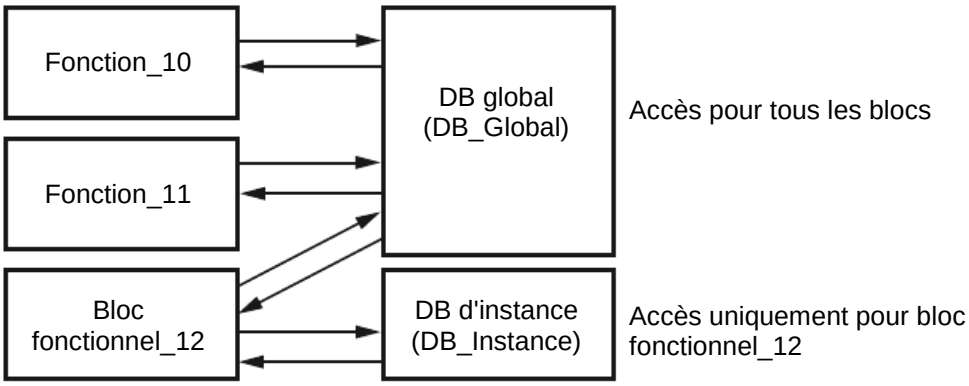


Figure 5 : Différence entre DB global et DB d'instance.

Exemples d'application pour les **blocs de données globaux :**

- Enregistrement des informations pour la gestion d'un magasin. "Où se trouve quel produit ?"
- Enregistrement des recettes de produits donnés.

4.7 Blocs de code réutilisables

Un programme utilisateur peut être écrit de façon linéaire ou structurée. La ***programmation linéaire*** écrit l'ensemble du programme utilisateur dans l'OB cyclique. Elle est uniquement recommandée pour des programmes très simples dans lequel on utilise entretemps des systèmes de commande moins onéreux comme LOGO!

La ***programmation structurée*** est toujours recommandée pour écrire des programmes complexes. La tâche d'automatisation peut être divisée en petites unités qui peuvent être résolues avec des fonctions et des blocs fonctionnels.

Il est recommandé d'utiliser de préférence des blocs de code réutilisables. Cela signifie que les paramètres d'entrée et de sortie d'une fonction ou d'un bloc fonctionnel sont définis globalement et qu'ils se voient attribuer des variables globales réelles (entrées/sorties) au moment de l'utilisation du bloc.

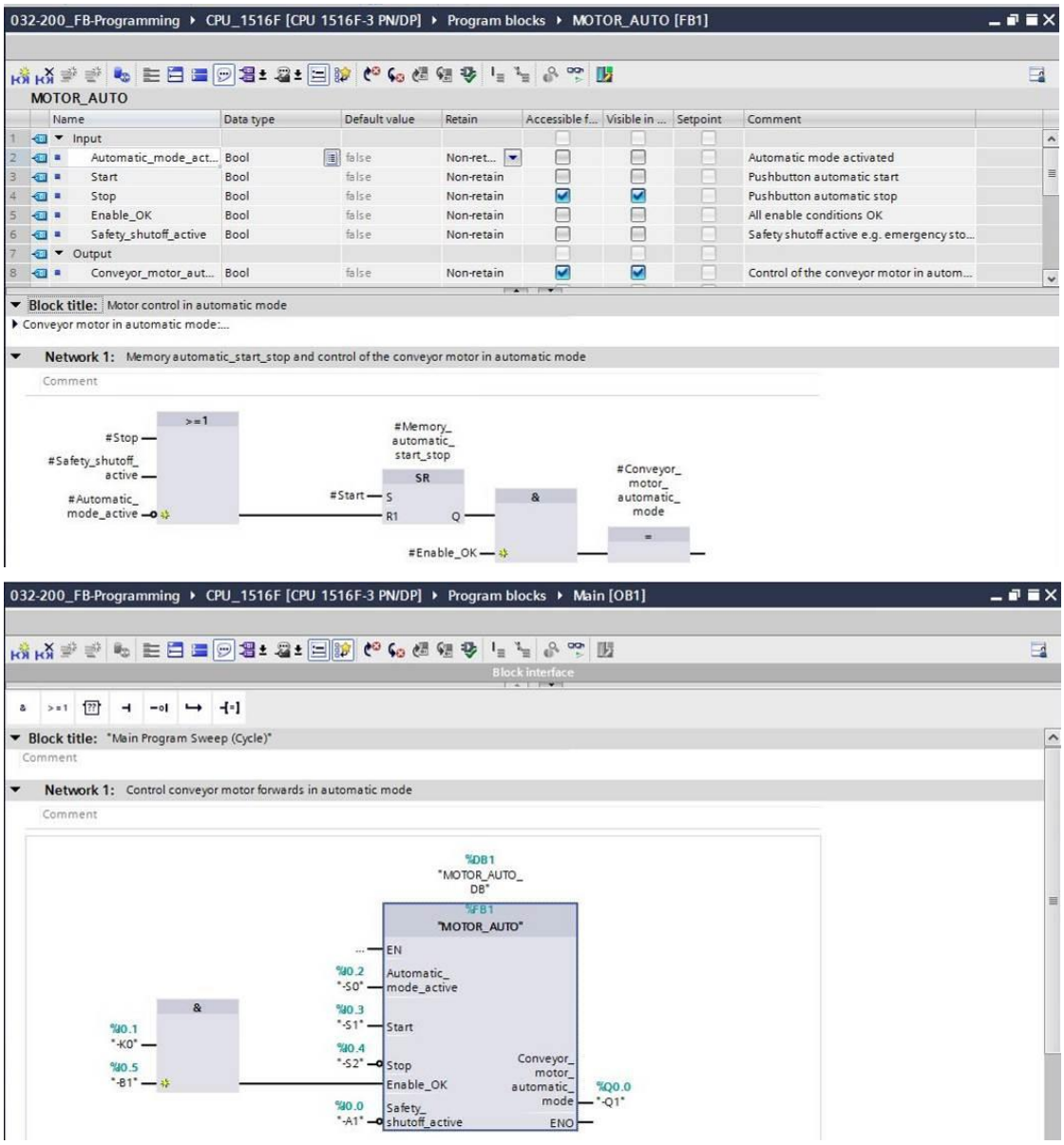


Figure 6 : Bloc fonctionnel réutilisable avec appel dans OB1

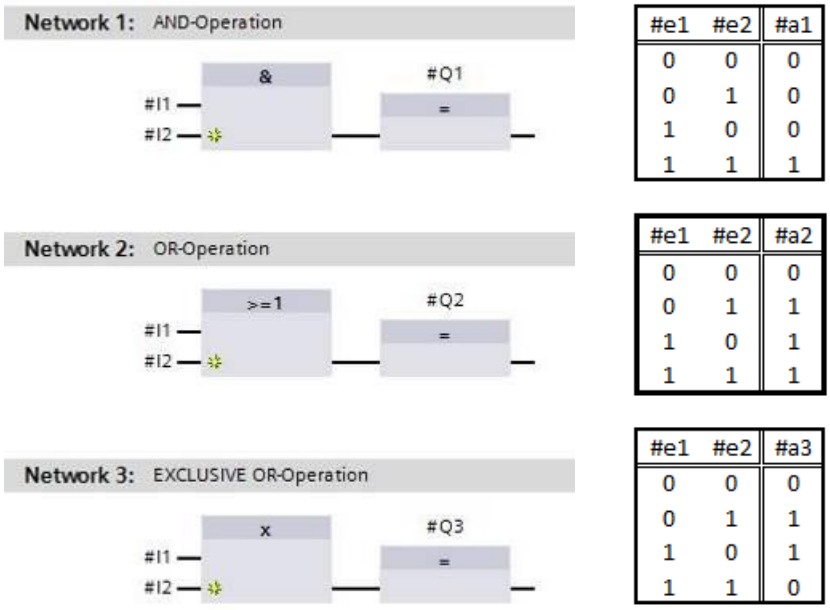
4.8 Langages de programmation

La programmation des fonctions peut être réalisée dans les langages de programmation suivants : logigramme (LOG), schéma à contacts (CONT), liste d'instructions (LIST) et Structured Control Language (SCL). Pour les blocs fonctionnels, le langage GRAPH permet de programmer des graphes séquentiels.

Les paragraphes suivants présentent le langage de programmation ***logigramme (LOG)***.

LOG est un langage de programmation graphique. La représentation est inspirée des systèmes de circuits électroniques. Le programme est représenté sous forme de réseaux. Un réseau contient un ou plusieurs chemins logiques. Les signaux binaires et analogiques sont combinés entre eux par des boîtes. Pour représenter la logique binaire, on utilise les symboles logiques graphiques connus de l'algèbre booléenne.

Avec les fonctions binaires, vous pouvez interroger les opérandes binaires et combiner leurs états logiques. Les instructions "Opération logique ET", "Opération logique OU" et "Opération logique OU EXCLUSIF" sont des exemples de fonctions binaires (voir Figure 7).



#e1	#e2	#a1
0	0	0
0	1	0
1	0	0
1	1	1

#e1	#e2	#a2
0	0	0
0	1	1
1	0	1
1	1	1

#e1	#e2	#a3
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7 : Fonctions binaires en LOG et table logique correspondante.

Les instructions simples permettent de forcer des sorties binaires, d'évaluer les fronts ou d'exécuter des fonctions de saut dans le programme.

Les instructions complexes proposent des éléments de programme comme les temporisations et compteurs CEI.

La boîte vide est un emplacement réservé dans lequel vous pouvez sélectionner l'instruction voulue.

Entrée de validation EN (ENable) / sortie de validation ENO (ENable Output) -Mécanisme :

- Une instruction sans mécanisme EN/ENO est exécutée indépendamment de l'état logique au niveau des entrées de la boîte.
- Les instructions avec mécanisme EN/ENO ne sont exécutées que si l'état logique de l'entrée de validation EN est "1". Si le traitement de la boîte est correct, la sortie de validation ENO est à l'état logique "1". Dès qu'une erreur survient en cours de traitement, la sortie de validation ENO est remise à zéro. Si l'entrée de validation EN n'est pas connectée, la boite est toujours exécutée.

5 Énoncé du problème

Ce chapitre a pour but de planifier, programmer et tester les fonctions suivantes de la description du process Installations de tri.

- Mode automatique - Moteur du convoyeur

6 Planification

Afin de conserver une certaine lisibilité et de s'assurer qu'elles sont réutilisables, il est recommandé de ne pas programmer toutes les fonctions dans OB1. Le code du programme est donc principalement réparti dans les fonctions (FC) et les blocs fonctionnels (FB). Le choix des fonctions réparties dans les FB et de celles qui doivent exécuter dans OB1 est planifié comme suit.

6.1 ARRET D'URGENCE

L'arrêt d'urgence n'a pas besoin de fonction propre. Tout comme le mode de fonctionnement, l'état du relais ARRET D'URGENCE peut être utilisé directement sur les blocs.

6.2 Mode automatique - Moteur du convoyeur

Le mode automatique du convoyeur doit être encapsulé dans un bloc fonctionnel (FB) "MOTOR_MANUAL". De cette manière, l'OB1 reste lisible et, si l'installation est équipée d'un convoyeur supplémentaire, la réutilisation est possible. Les paramètres prévus sont présentés au tableau 2.

Input	Type de données	Commentaire
Mode automatique	BOOL	Mode de fonctionnement mode automatique activé
Commande_démarrage	BOOL	Commande de démarrage pour le mode automatique
Commande_arrêt	BOOL	Commande d'arrêt pour le mode automatique
Validation_OK	BOOL	Toutes les conditions de validation sont remplies
Disjoncteur_actif	BOOL	Disjoncteur actif ou arrêt d'urgence déclenché
Output		
Moteur du convoyeur_automatique	BOOL	Pilotage du moteur du convoyeur en mode automatique
Static		
mémoire_automatique_marche_arrêt	BOOL	Mémoire pour fonction de démarrage et d'arrêt en mode automatique

Tableau 2 : Paramètres pour FB "MOTOR_AUTO"

La fonction mémoire_automatique_marche_arrêt est activée avec mémorisation par la commande de démarrage, mais seulement si les conditions de réinitialisation ne sont pas remplies.

Mémoire_automatique_marche_arrêt est mis à 0 si la commande d'arrêt est active, si le disjoncteur est actif ou si le mode automatique n'est pas activé (mode manuel).

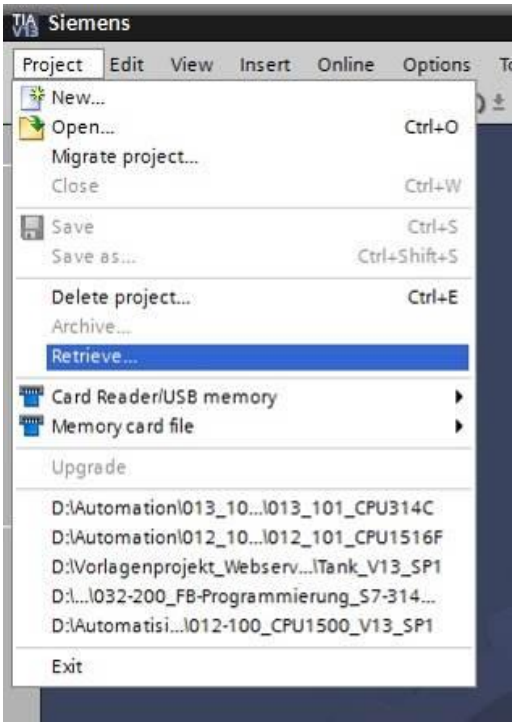
La sortie Moteur du convoyeur_automatique est activée si mémoire_automatique_marche_arrêt est mis à 1 et les conditions de validation sont remplies.

7 Instructions structurées par étapes

Vous trouverez ci-après des instructions pour réaliser la planification. Si vous êtes déjà expérimenté, les étapes numérotées vous suffisent. Sinon, suivez les étapes détaillées des instructions.

7.1 Désarchiver un projet existant

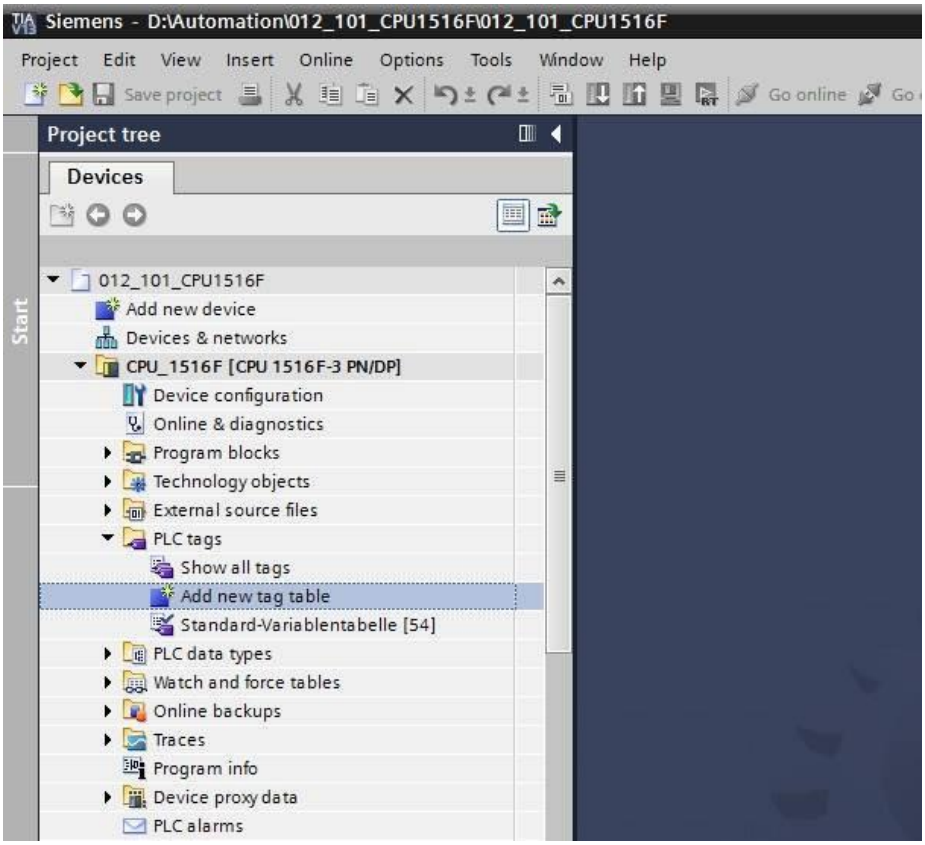
→ Avant de commencer la programmation du bloc fonctionnel (FB) "MOTOR_AUTO", il nous faut un projet avec une configuration matérielle (p.ex. SCE_FR_012-101_configuration matérielle_S7-1516F_R1502.zap). Pour désarchiver un projet existant, vous devez rechercher l'archive à partir de la vue de projet sous →Project (Projet)→Retrieve (Désarchiver). Confirmez votre choix avec "Open (Ouvrir)". (→Project (Projet)→Retrieve (Désarchiver)→ Sélectionner une archive zap → Open (Ouvrir))



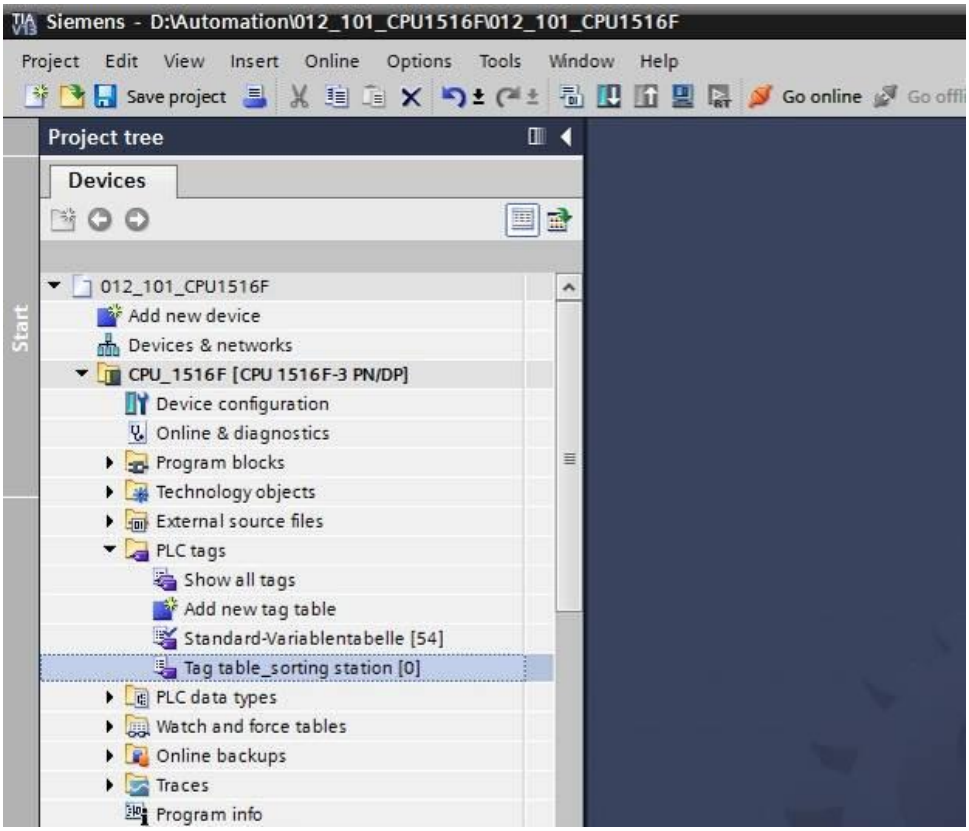
→ Sélectionner ensuite le répertoire cible pour enregistrer le projet désarchivé. Confirmez votre sélection par "OK". (→ Répertoire cible → "bouton ""OK")

7.2 Création d'une nouvelle table des variables

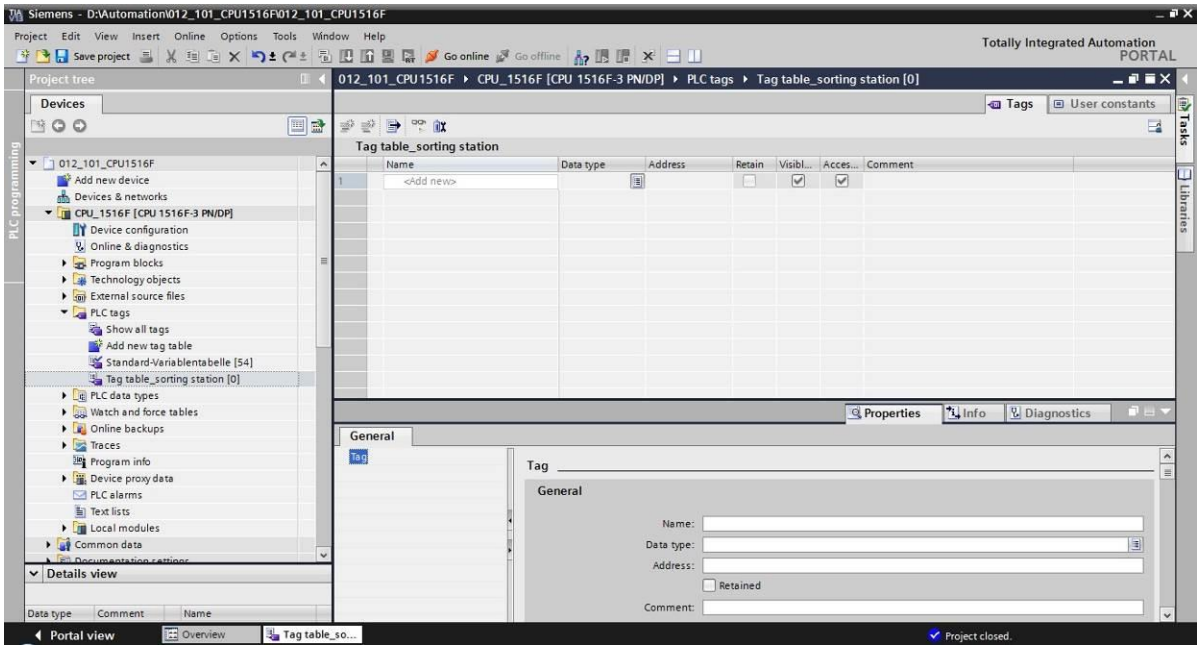
→ Dans la vue du projet (Project tree), afficher les → PLC tags (Variables API) de l'automate et créer une nouvelle table des variables en double-cliquant sur → "Add new tag table (Ajouter table des variables)".



→ Renommer cette nouvelle table "tag table_sorting station (table des variables_installation de tri)". (→ Clic droit sur "tag table_1" → "Rename (Renommer)" → tag table_sorting station (table des variables_installation de tri))

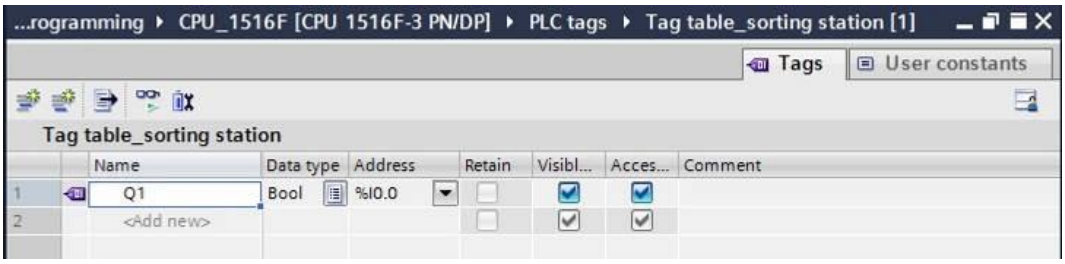


→ L'ouvrir ensuite par double clic. (→ table des variables_installation de tri)

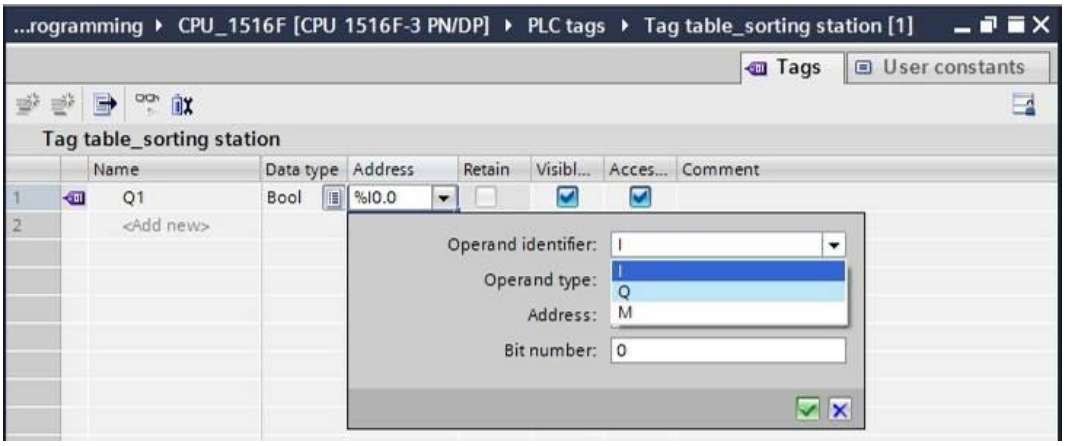


7.3 Création de nouvelles variables dans une table des variables

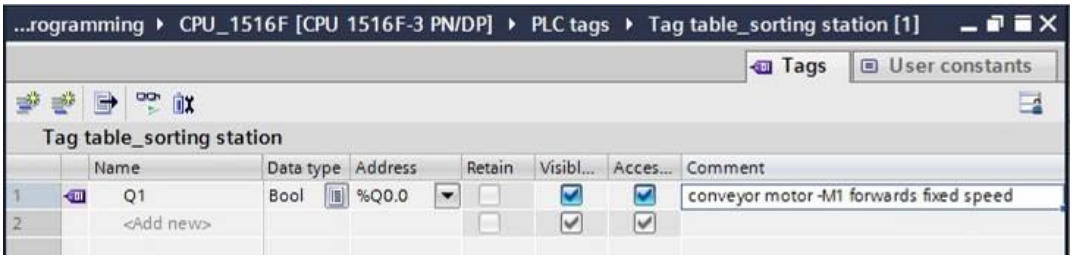
→ Ajouter le nom Q1, puis confirmez votre saisie en appuyant sur la touche Entrée. Si vous n'avez pas encore créé d'autres variables, TIA Portal a attribué automatiquement le type de données "Bool" et l'adresse %E0.0 (I 0.0). (→ <Add new> (créer)→ Q1 → Entrée)



→ Modifier l'adresse en %A0.0 (Q0.0), soit par saisie directe, soit en cliquant sur la flèche de la liste déroulante pour ouvrir le menu d'adressage ; changer l'identifiant d'opérande en Q et confirmer avec Entrée ou en cliquant sur la coche. (→ %E0.0 → Operand identifier (identifiant d'opérande) → Q →)

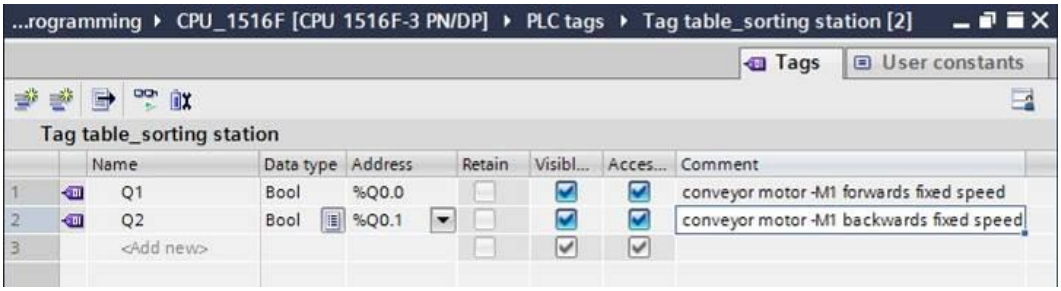


→ Inscrire le commentaire suivant "conveyor motor -M1 forwards fixed speed (moteur du convoyeur -M1 avant vitesse de rotation fixe".



→ Ajouter une nouvelle variable Q2 dans la ligne 2. TIA Portal a automatiquement attribué le même type de données qu'à la ligne 1 et incrémenté l'adresse de 1 : %A0.1 (Q0.1). Saisir le commentaire "conveyor motor -M1 backwards fixed speed (moteur du convoyeur -M1 arrière vitesse de rotation fixe)".

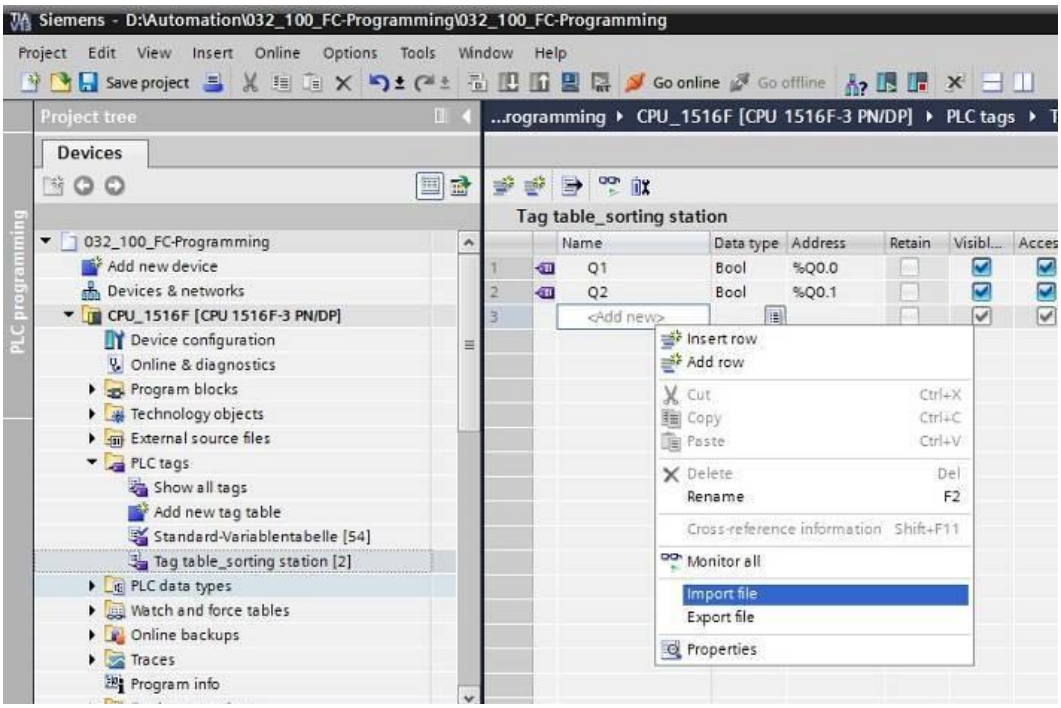
(→ <Add new (créer> → Q2 → Entrée → Comment (commentaire) → conveyor motor -M1 backwards fixed speed (moteur du convoyeur -M1 arrière vitesse de rotation fixe))



7.4 Importation de la "table des variables_installation de tri"

→ Pour ajouter une table des mnémoniques existante, cliquer avec le bouton droit de la souris sur un emplacement vide de la "table des variables_installation de tri". Dans le menu contextuel, choisissez "Import file (Importer fichier)".

(→ clic droit sur un emplacement vide de la table des variables → Import file (Importer fichier))



→ Choisissez la table des mnémoniques voulue (p.ex. au format .xlsx) et confirmer la sélection avec "Open (ouvrir)".

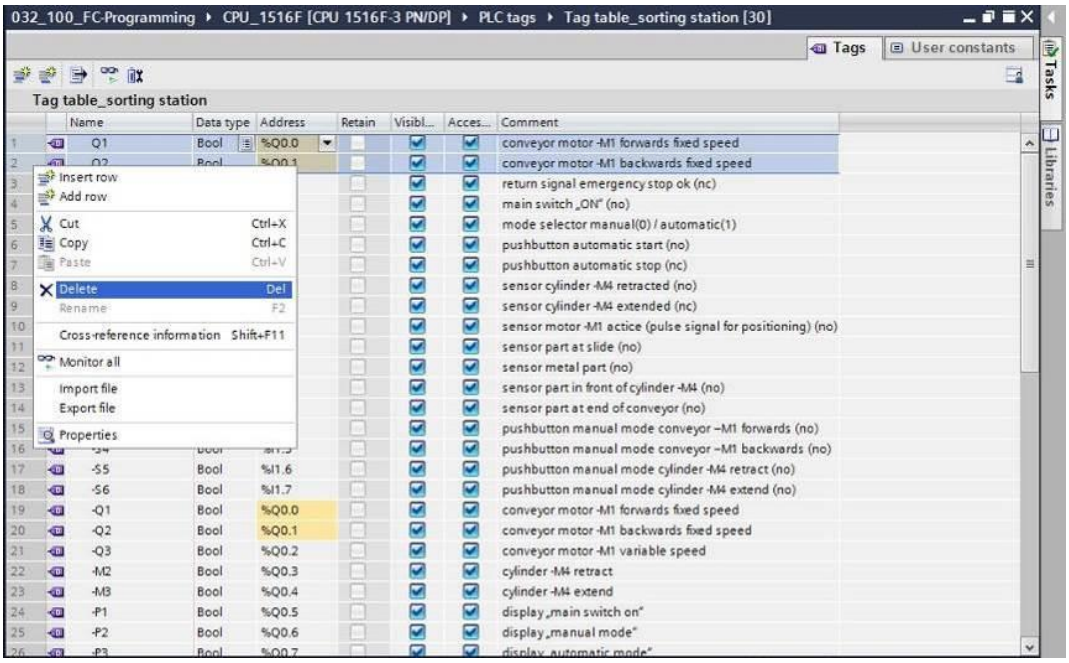
(→ SCE_FR_020-100_table des variables installation de tri… → Open (ouvrir))

→ Une fois l'importation terminée, une fenêtre de confirmation s'affiche et vous pouvez consulter le fichier journal de l'importation. Cliquer sur → OK.

→ Certaines adresse sont surlignées en orange. Il s'agit d'adresses en double et les noms des variables ont été numérotées automatiquement pour éviter toute équivoque.

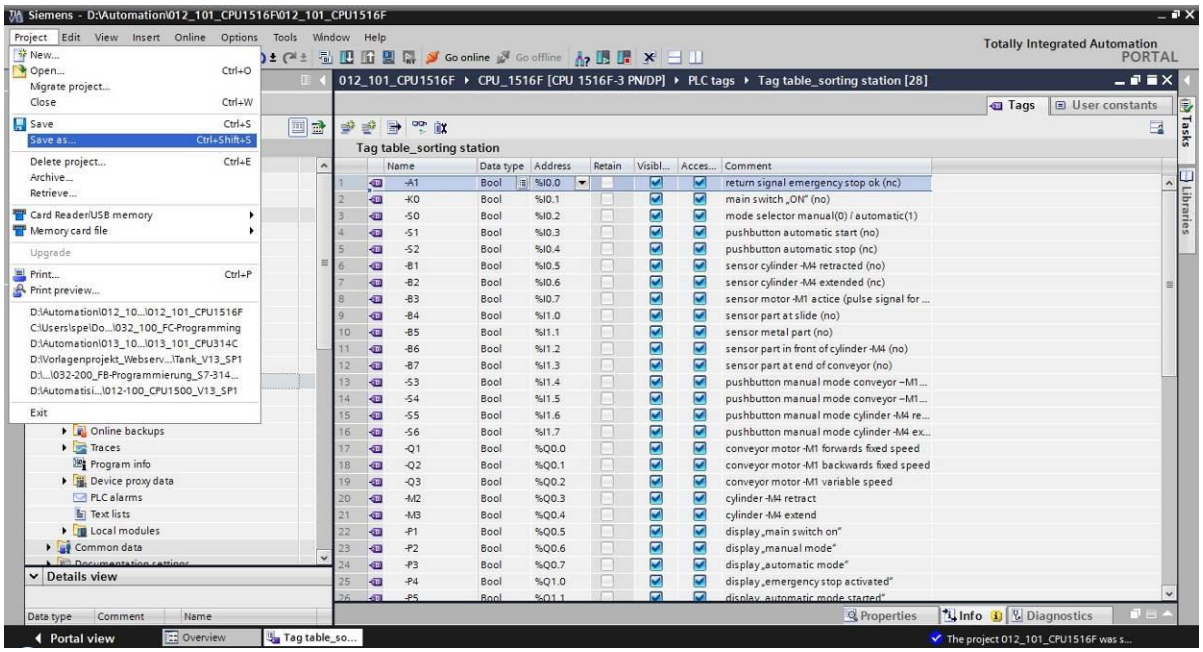
→ Supprimer les variables en double en sélectionnant la ligne et en appuyant sur la touche Suppr. du clavier ou dans le menu contextuel en choisissant la commande Delete (Supprimer).

(→ Clic droit sur la variable sélectionnée → Delete (Supprimer))



→ Vous avez devant vous une table des mnémoniques complète des entrées et sorties TOR. Enregistrer le projet sous le nom 032-100_Programmation de FC.

(→ Project (Projet) → Save as (Enregistrer sous) … → 032-200_Programmation de FB → Save (Enregistrer))

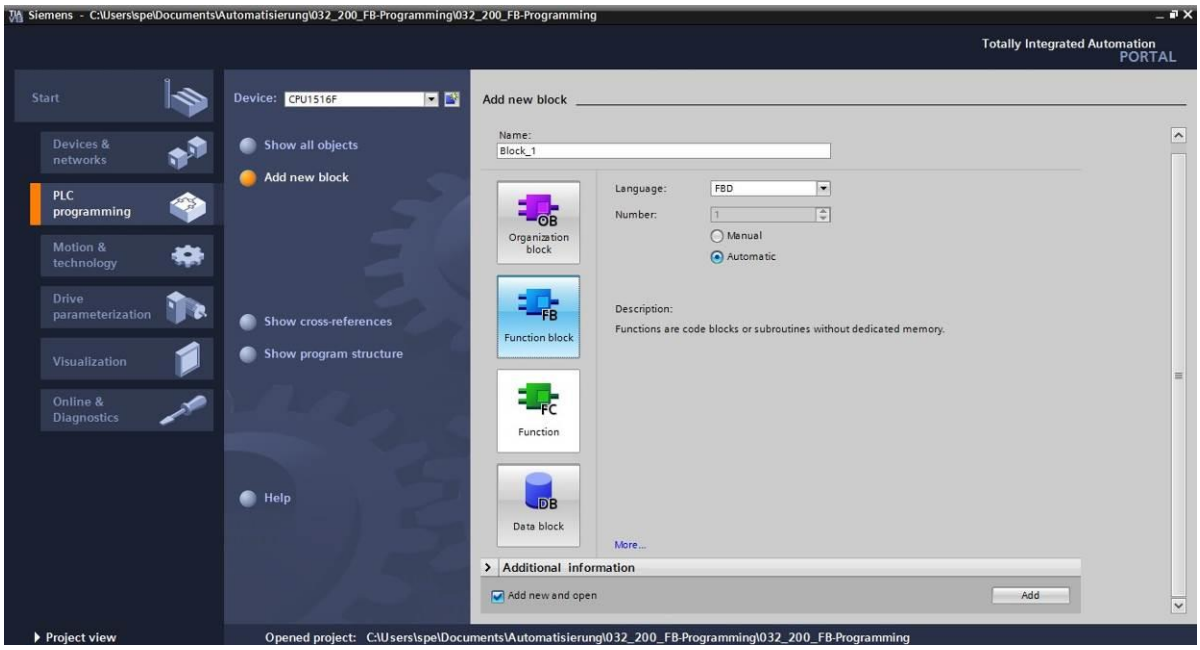


7.5 Création du bloc fonctionnel FB1 "MOTOR_AUTO" pour le moteur du convoyeur en mode automatique

→ Dans la vue du portail, sous PLC programming (Programmation de l'API), cliquer sur "Add new block (Créer un bloc)" pour créer un nouveau bloc fonctionnel.

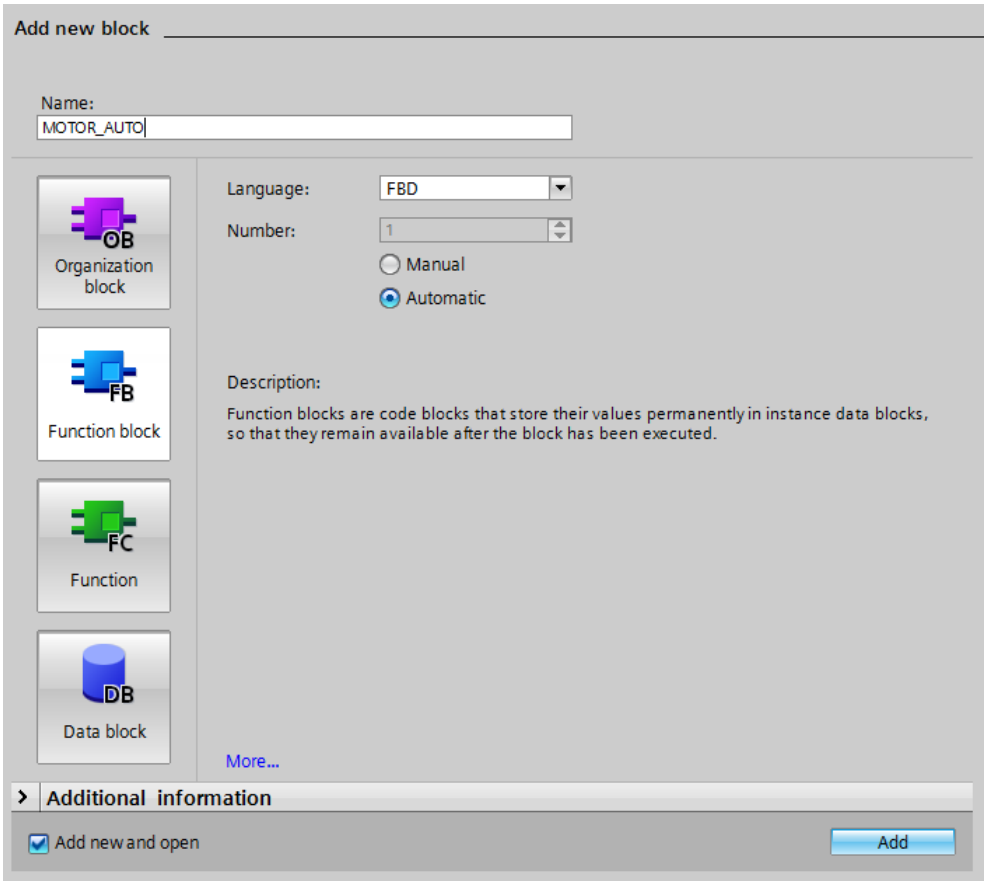
(→ PLC programming (Programmation de l'API) → Add new block (Créer un bloc)→

)



→ Nommer ce bloc : "MOTOR_AUTO", indiquer comme langage FBD (LOG) et accepter l'attribution automatique des numéros. Activer la case à cocher "Add new and open (créer et ouvrir)" pour atteindre automatiquement la vue du projet du bloc fonctionnel que vous venez de créer. Cliquer sur "Add (Ajouter)".

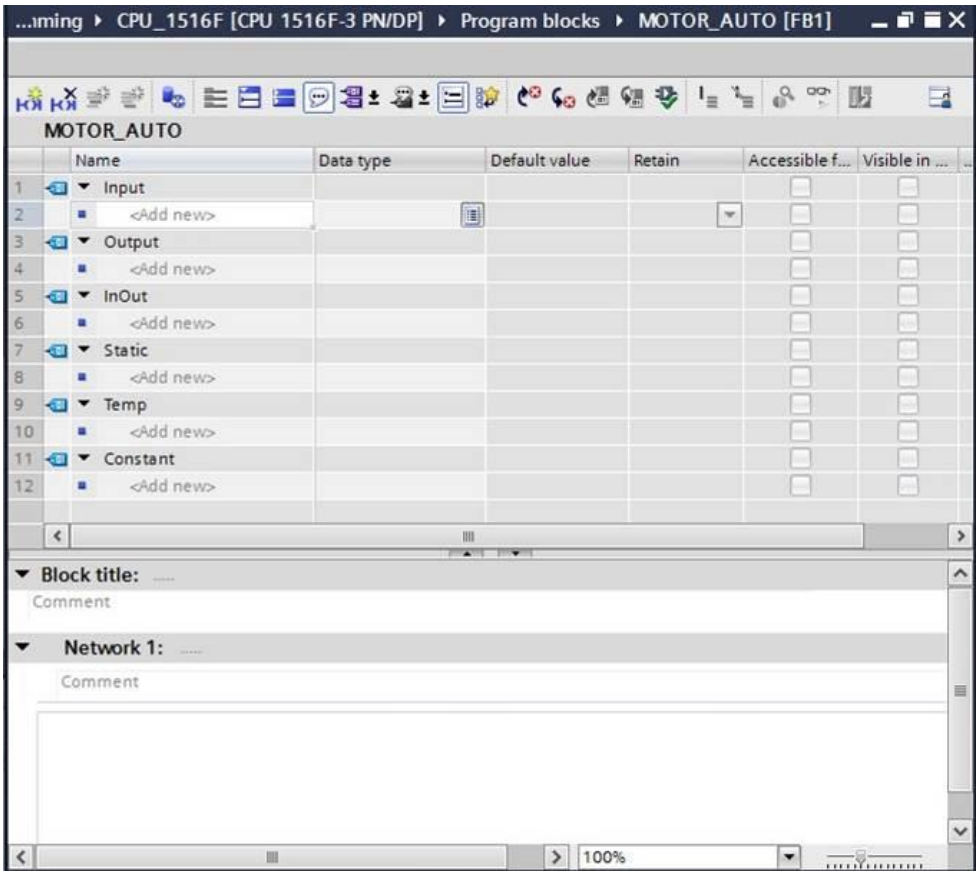
(→ Name (nom) : moteur→ Langage : FBD (LOG)→ Number (numéro) : Automatic (automatique) → ☑ Add next and open (créer et ouvrir) → Add (ajouter))



7.6 Définir l'interface du FB1 "MOTOR_AUTO"

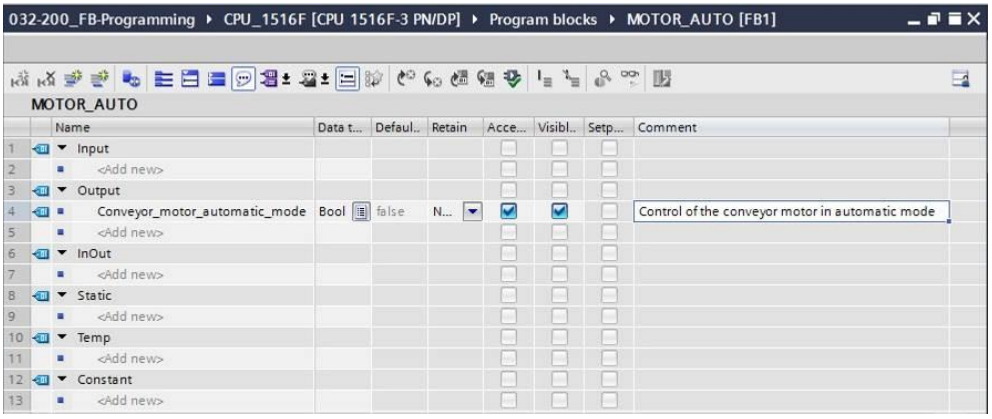
→ Si vous avez cliqué sur "Add new and open (créer et ouvrir), la vue du projet s'affiche avec une fenêtre de création du bloc qui vient d'être généré.

→ Dans la partie supérieure de la vue de programmation, vous trouvez la description de l'interface du bloc fonctionnel.



→ Un signal de sortie binaire est nécessaire pour piloter le moteur du convoyeur. Nous allons donc créer en premier la variable de sortie locale #Conveyor_motor_automatic_mode de type "Bool". Ajouter en commentaire "control of the conveyor motor in automatic mode (commande du moteur du convoyeur en mode automatique)".

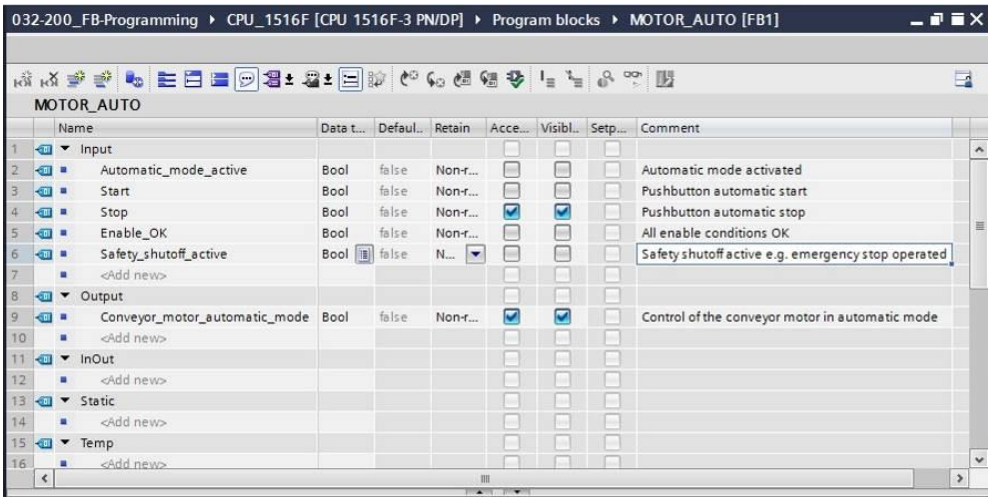
(→ Output : Conveyor_motor_automatic_mode (Moteur du convoyeur_mode_automatique)→Bool→ control of the conveyor motor in automatic mode (commande du moteur du convoyeur en mode automatique)



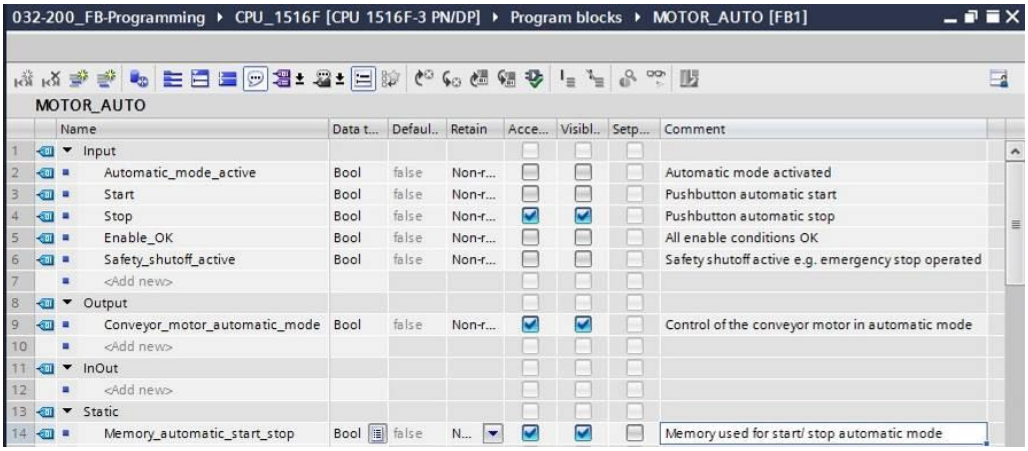
→ Sous Input, ajouter comme interface d'entrée le paramètre #automatic_mode_active (mode_automatique_actif), puis confirmez votre saisie en appuyant sur la touche Entrée ou en quittant la zone de texte. Le type de données "BOOL" est attribué automatiquement. Il est conservé. Saisir ensuite le commentaire "automatic mode activated (mode automatique activé)".

(→ automatic_mode_active (mode_automatique_actif) → BOOL → automatic mode activated (mode automatique activé))

→ Sous Input, ajouter d'autres paramètres d'entrée binaires #Start (démarrage), #Stop (arrêt), #Enable_OK (validation_ok) et #Safety_shutoff_active (disjoncteur_actif) et vérifier les types de données. Compléter par des commentaires parlants.

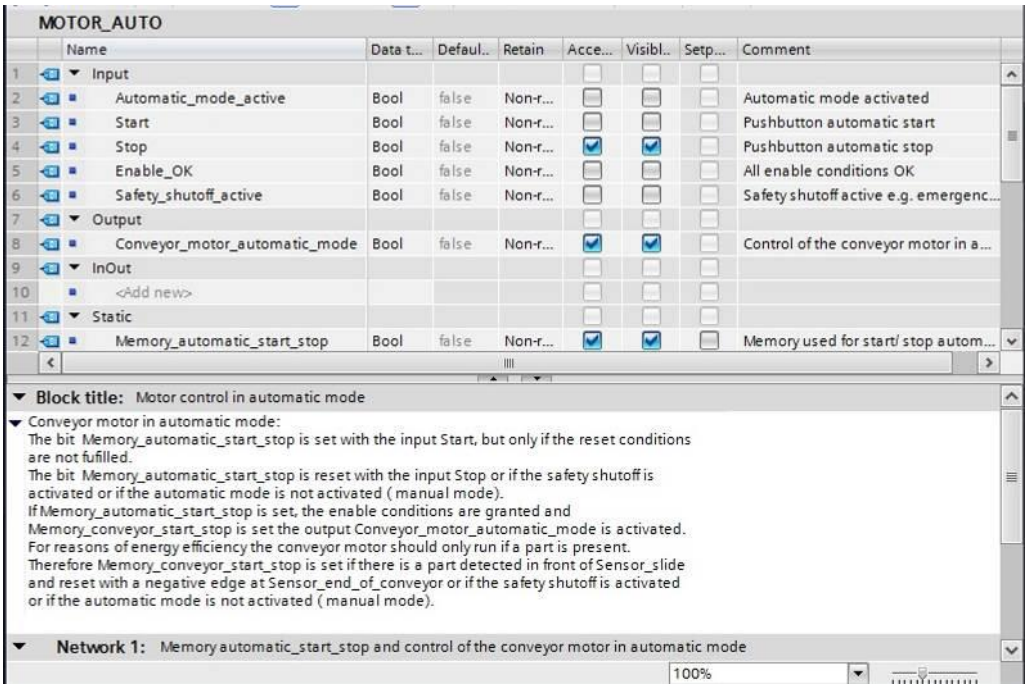


Le démarrage et l'arrêt du convoyeur s'effectue à partir des boutons. C'est pourquoi il nous faut une variable "static" comme mémoire. Sous Static, ajouter la variable #Memory_automatic_start_stop (mémoire_automatique_marche_arrêt), puis confirmez votre saisie en appuyant sur la touche Entrée ou en quittant la zone de texte. Le type de données "BOOL" est attribué automatiquement. Il est conservé. Saisir ensuite le commentaire "Memory used for start/stop automatic mode (mémoire utilisée pour démarage/arrêt du mode automatique)". (→Memory_automatic_start_stop (mémoire_automatique_marche_arrêt) → Bool → Memory used for start/stop automatic mode (mémoire utilisée pour démarage/arrêt du mode automatique))



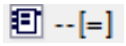
→ Pour documenter la programmation du bloc, saisir un commentaire de bloc et donner au réseau 1 un titre parlant.

(→ Bloc title (titre du bloc : Conveyor motor in automatic mode (moteur du convoyeur en mode automatique) → Network (réseau) 1 : Memory automatic_start_stop and control of the conveyor motor in automatic mode (mémoire démarrage_arrêt_automatique et pilotage du moteur du convoyeur en mode automatique))

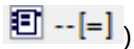


7.7 Programmation du FB1 : MOTOR_AUTO

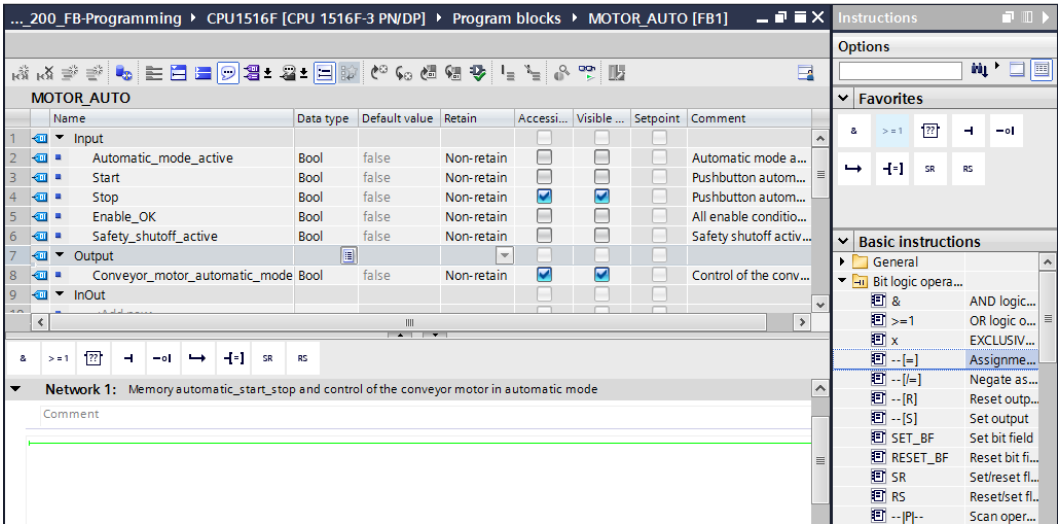
→ Sous la description de l'interface, vous voyez dans la fenêtre de programmation une barre d'outils avec différentes fonctions logiques et en dessous une zone avec des réseaux. Nous avons déjà défini le titre du bloc et le titre du premier réseau. La programmation s'effectue dans le réseau en utilisant des blocs logiques. La répartition sur plusieurs réseaux sert à maintenir la lisibilité. Les paragraphes suivants expliquent comme ajouter des blocs logiques.

→ Dans la partie droite de la fenêtre de programmation se trouve la liste des instructions qui peuvent être utilisées dans le programme. Sous → Basic instructions (instructions de base) → Bit logic operations (opérations logiques sur bits), rechercher la fonction 

(affectation) et la faire glisser sur le réseau 1 (une ligne verte apparait, pointeur avec symbole +).

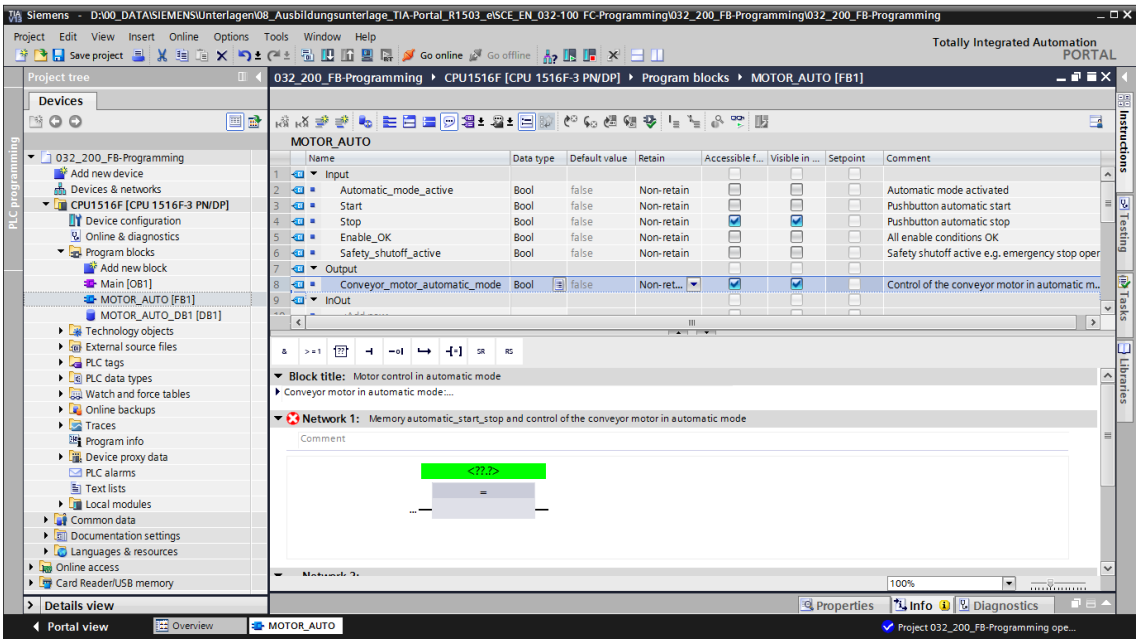
(→ instruction → Instructions simples → fonction logique combinatoire sur bits → 

)

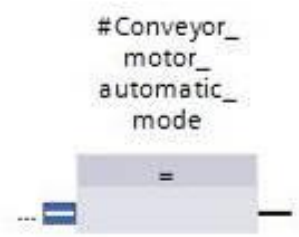


→ Faire glisser le paramètre de sortie #Conveyor_motor_automatic_mode sur <??.?> au-dessus du bloc qui vient d'être ajouté. Pour sélectionner plus facilement un paramètre dans la description de l'interface, le saisir sur le symbole bleu.

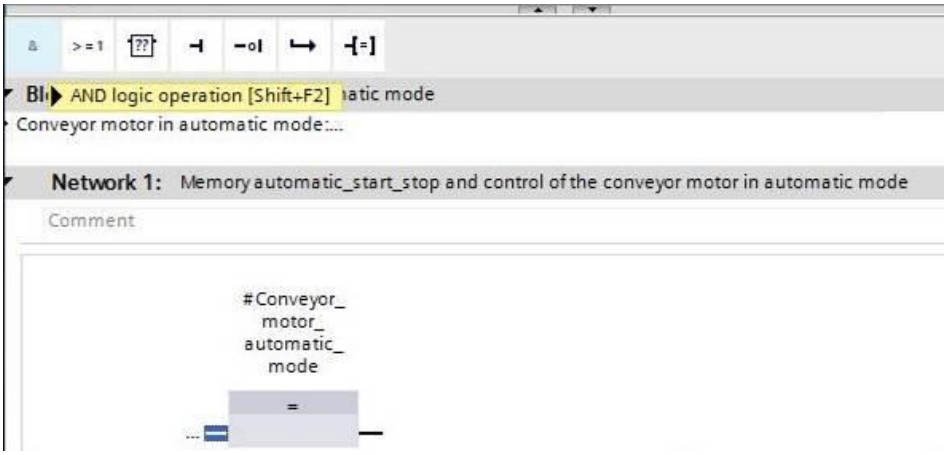
(→ Conveyor_motor_automatic_mode (Moteur du convoyeur_automatique))



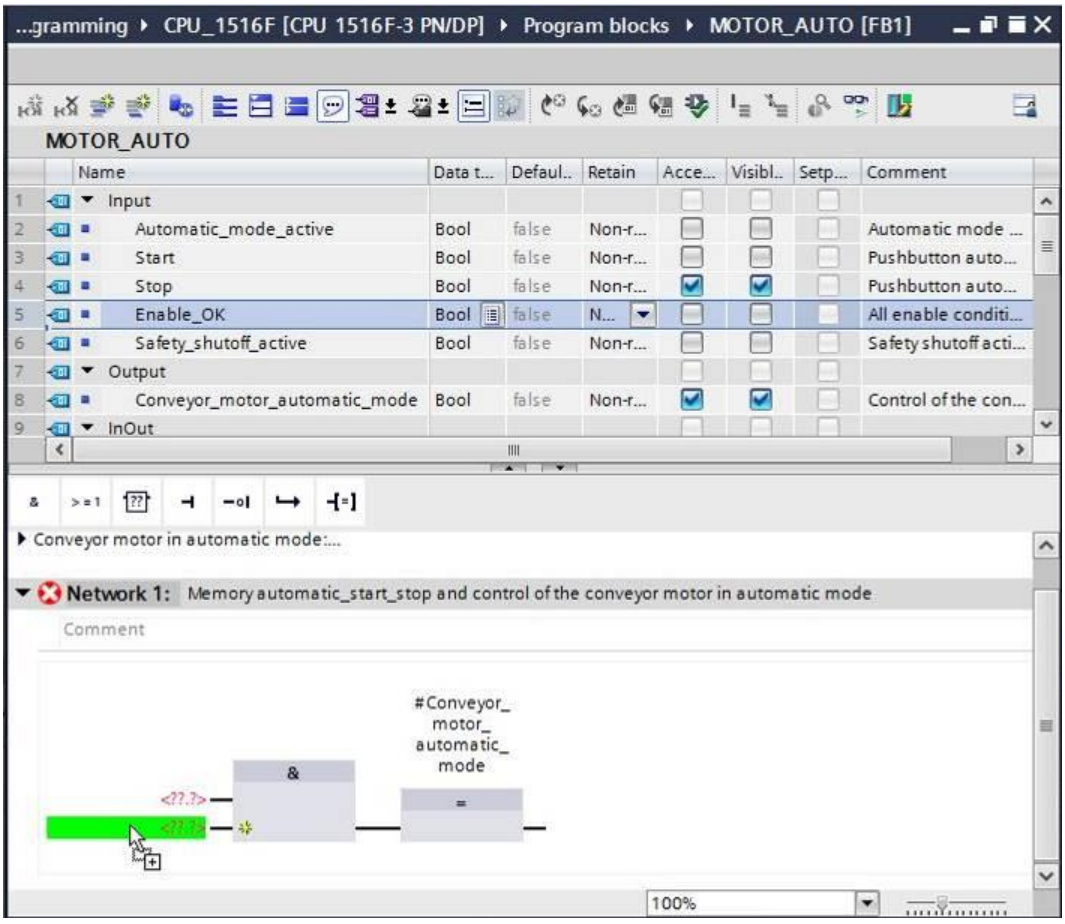
→ Ceci définit que le paramètre #Conveyor_motor_automatic_mode est écrit par ce bloc. Il manque encore les conditions d'entrée pour que cette opération soit effectivement possible. L'entrée du bloc d'affectation doit aussi être connectée à une bascule SR et le paramètre #Enable_OK à un ET. Cliquer pour cela sur l'entrée du bloc, afin que le trait d'entrée s'affiche sur fond bleu.



→ Cliquer sur & dans la barre d'outils logique pour ajouter une liaison Et avant le bloc d'affectation.

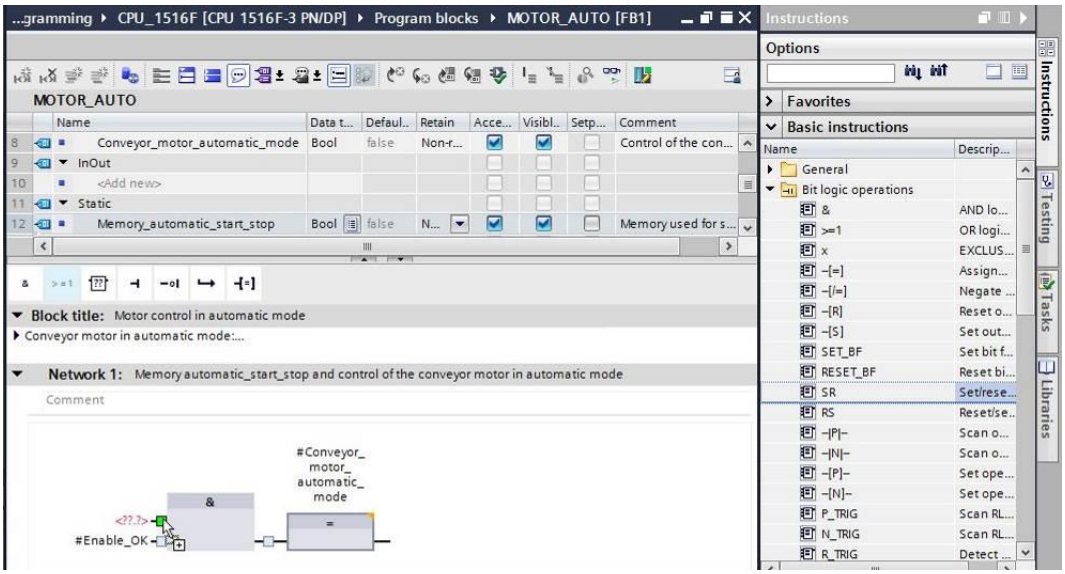


→ Faire glisser le paramètre d'entrée #Enable_OK sur la deuxième entrée de la liaison & <??.?>. (→ Enable_ok (Validation_ok)

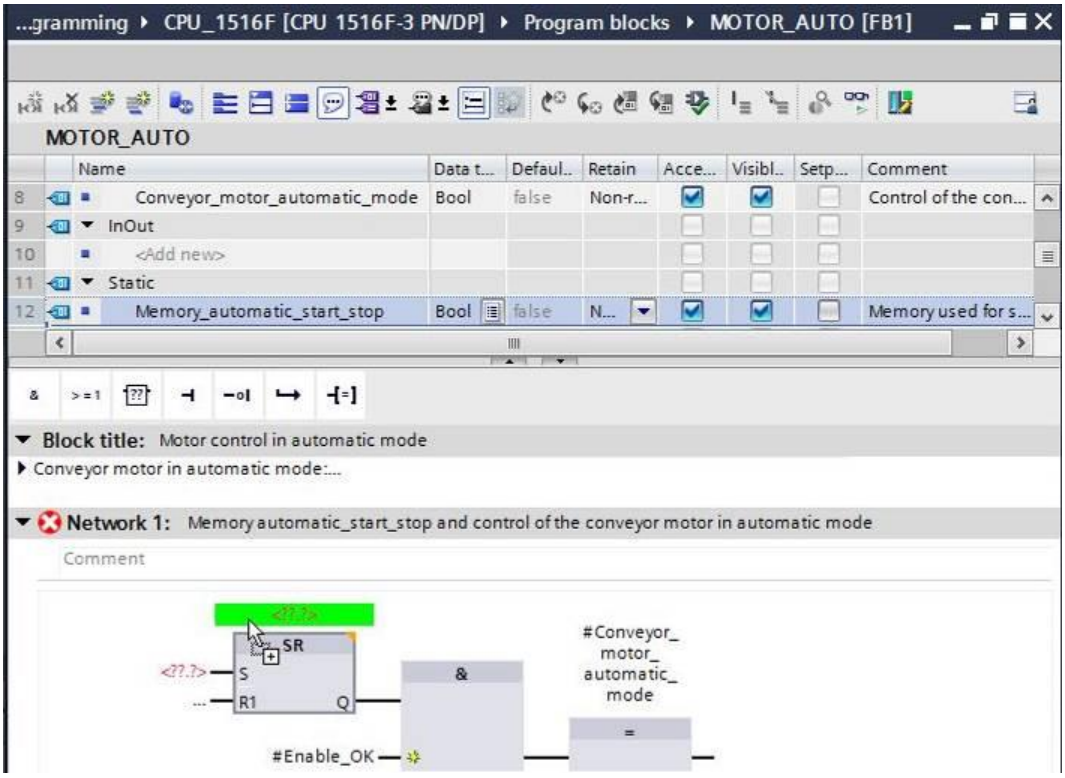


→ Faire glisser de la liste des instructions sous →Basic instructions (instructions de base) → Bit logic operations (opérations logiques sur bits) la fonction Set/Reset Flipflop (Bascule SR) SR sur la première entrée de la liaison &.

(→ Instructions → Basic instructions (Instructions de base) → Bit logic operations (opérations logiques sur bits) → SR →)

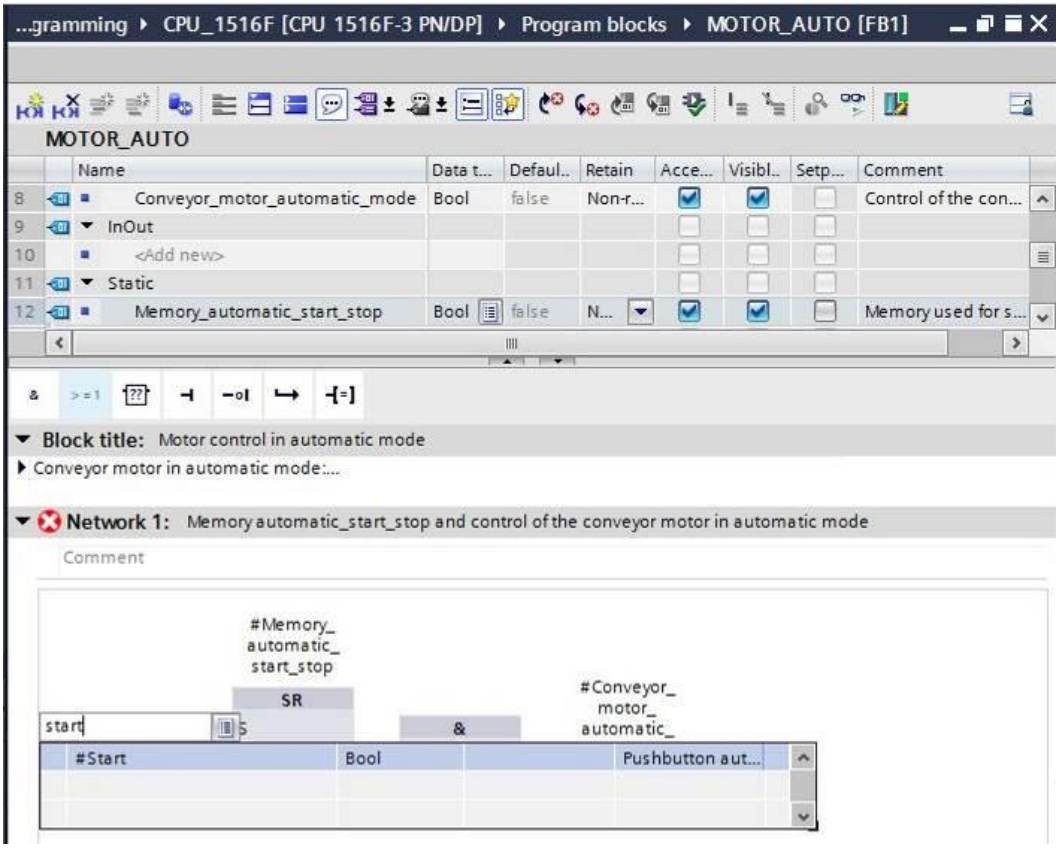


→ La bascule SR a besoin d'une variable mémoire. Faire glisser le paramètre statique #Memory_automatic_start_stop (mémoire_automatique_marche_arrêt) sur "…" sur <??.?> au-dessus de la bascule SR. (→ Memory_automatic_start_stop (mémoire_automatique_marche_arrêt))



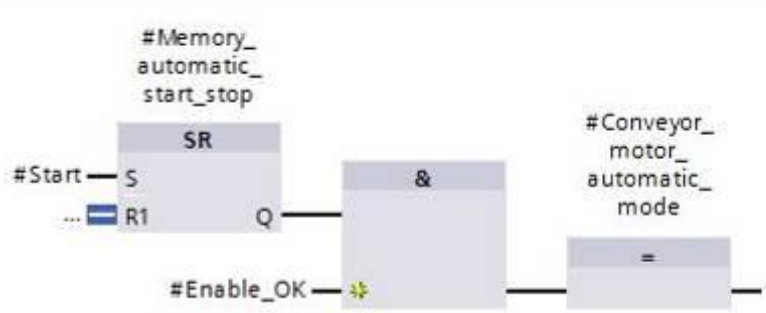
→ Memory_automatic_start_stop (mémoire_automatique_marche_arrêt) doit être affectée de la variable d'entrée #Start (démarrage). Double cliquer sur l'entrée S de la bascule SR <??.?> et dans la zone de texte qui s'affiche, saisir "Start (démarrage)" pour voir la liste des variables commençant par "Start (démarrage) (si vous faites la recherche en anglais). Cliquer sur la variable #Start (démarrage) et valider par → Entrée.

(→ SR-Flipflop (bascule SR) → <??.?> → Start (démarrage) → #Start (démarrage)→ Entrée)

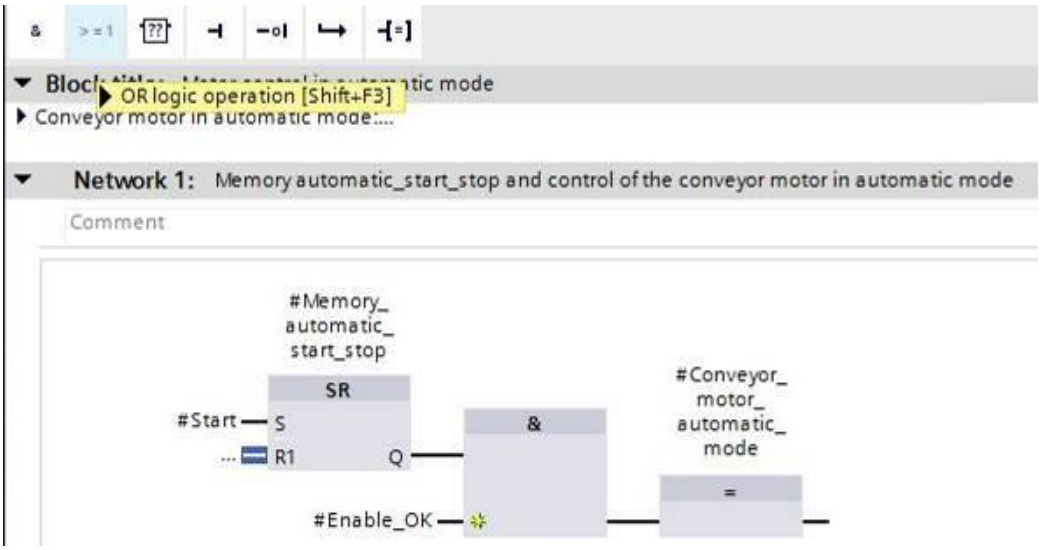


Remarque : *il existe un risque de confusion entre cette variante de l'affectation de variables et les variables globales de la table des variables. C'est pourquoi il faut privilégier l'option consistant à faire glisser la variable de la description de l'interface comme montré auparavant.*

→ Plusieurs conditions doivent pouvoir arrêter le convoyeur. A l'entrée R1 de la bascule SR, il faut donc un bloc OU. Cliquer pour cela sur l'entrée R1 de la bascule SR, afin que le trait d'entrée s'affiche sur fond bleu.

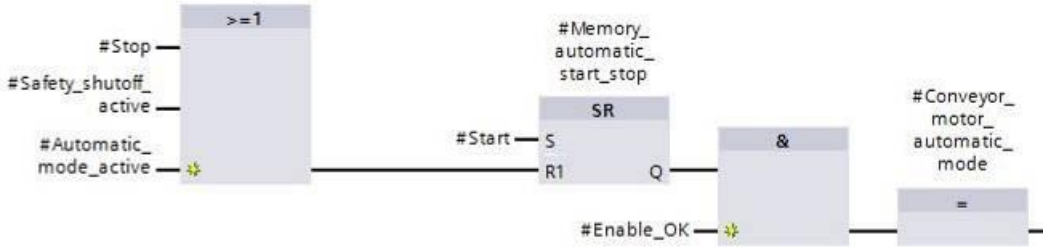


→ Cliquer ensuite sur >=1 dans la barre d'outils logique pour ajouter une liaison OU.

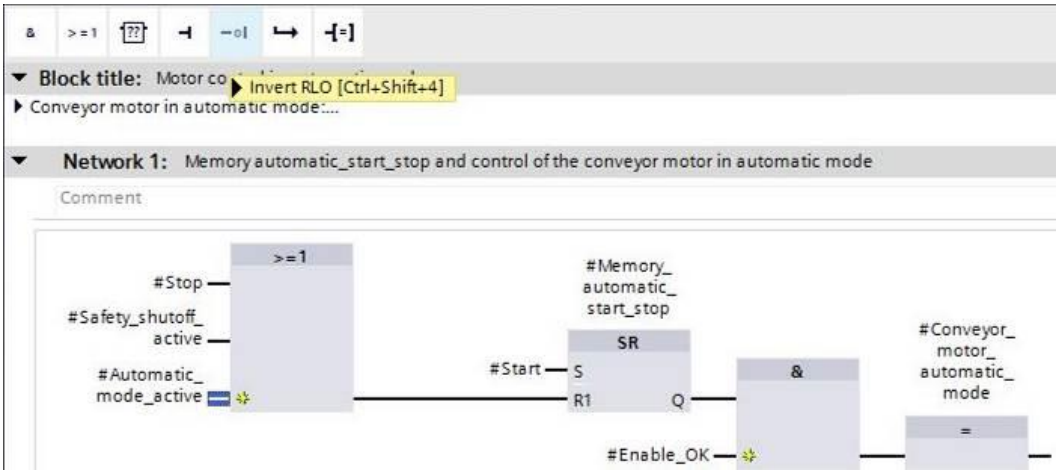


→ Le bloc OU n'a que deux entrées. Cliquer sur l'étoile jaune du circuit OU pour ajouter une variable d'entrée supplémentaire.

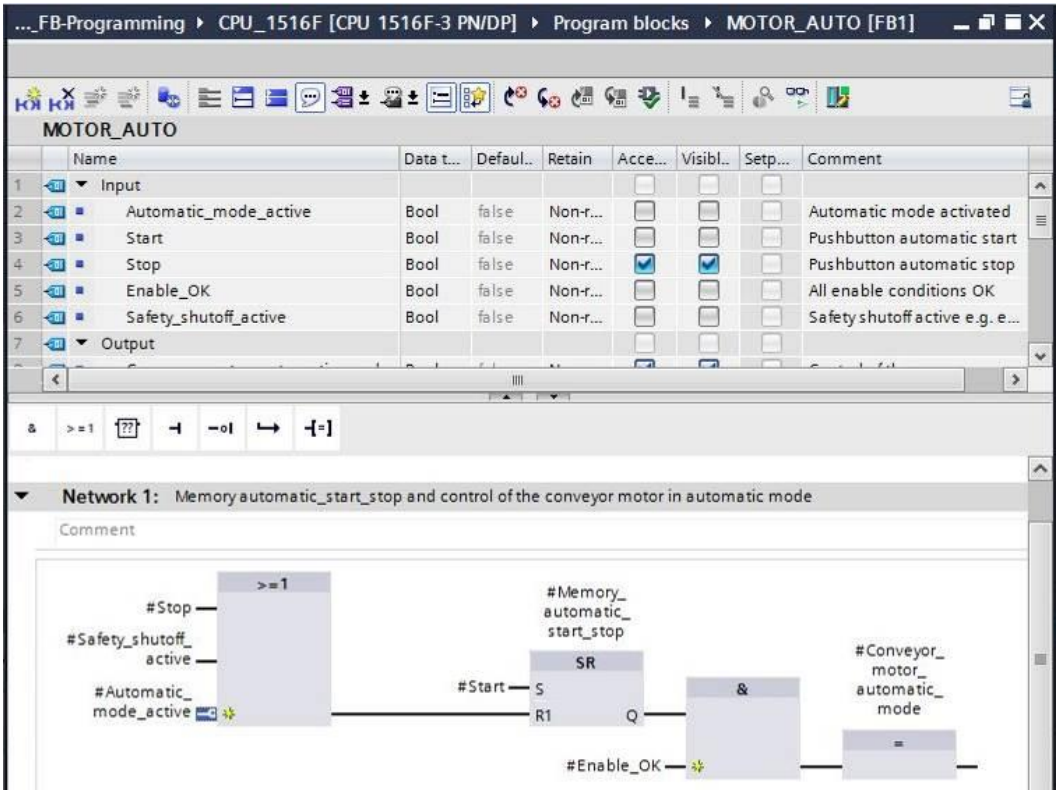
→ Sur les trois entrées du circuit OU, ajouter les variables d'entrée #Stop (arrêt), #Safety_shutoff_active (disjoncteur_actif) et #Automatic_mode_active (mode_automatique_actif).



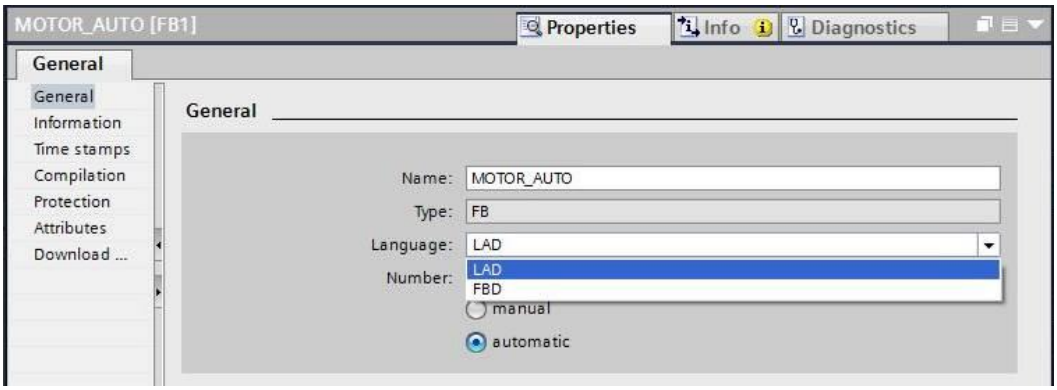
→ Ajouter une négation (Invert RLO) à l'entrée connectée au paramètre Automatic_mode_active (mode_automatique_actif) en le sélectionnant, puis en cliquant sur .



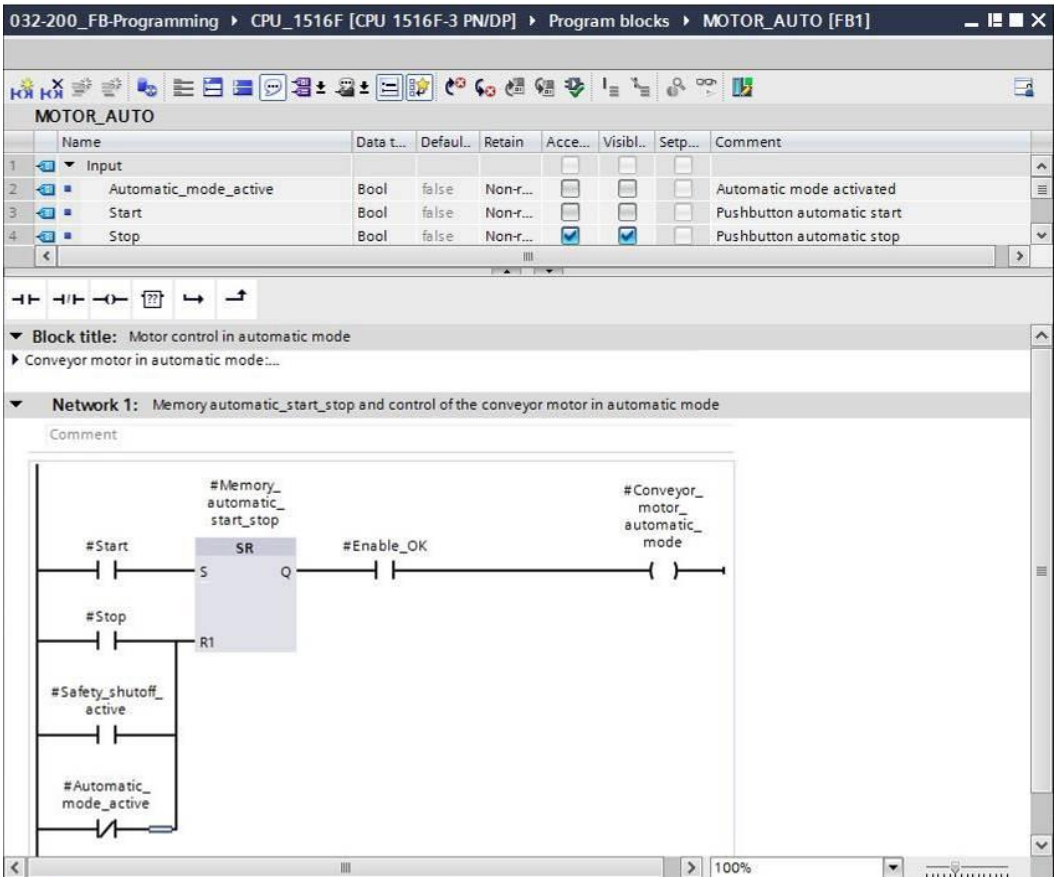
→ Ne pas oublier de cliquer sur Save project. Le bloc fonctionnel "MOTOR_AUTO [FB1]" terminé est représenté ci-dessous en FBD (LOG).



→ Dans les propriétés du bloc, sous "General (Général)", "Language", vous pouvez choisir LAD (CONT - schéma à contacts). (→ Properties (Propriétés) → General (Général) → Language (langage) : LAD (CONT))



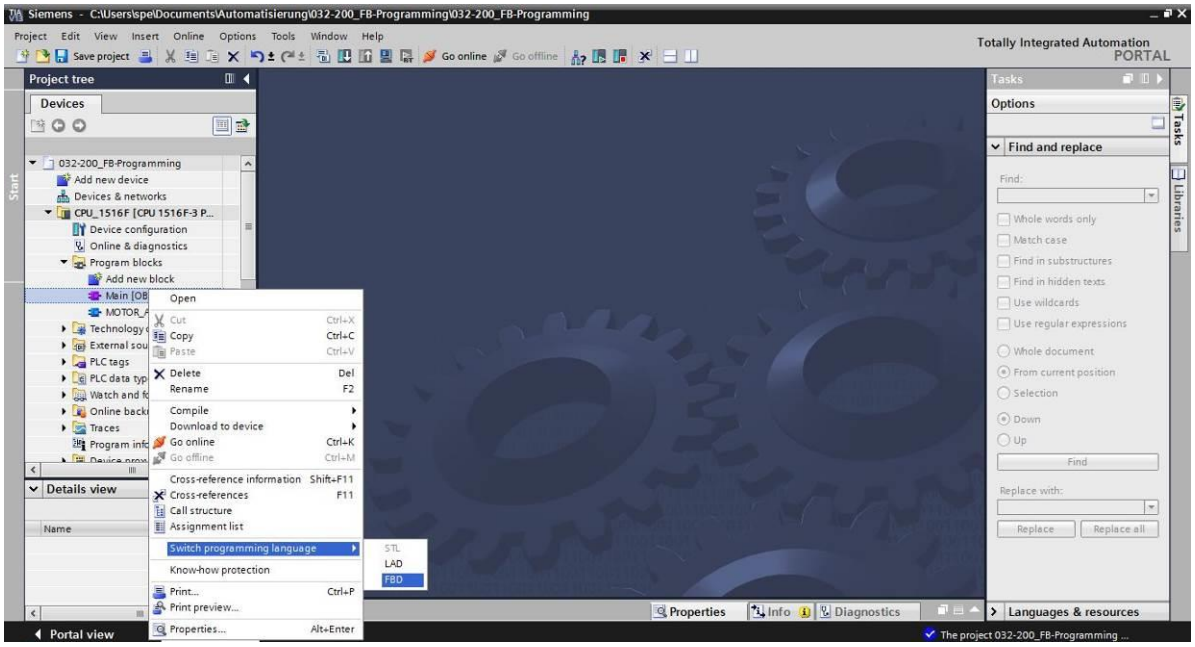
→ En LAD (CONT), le programme se présente comme suit :



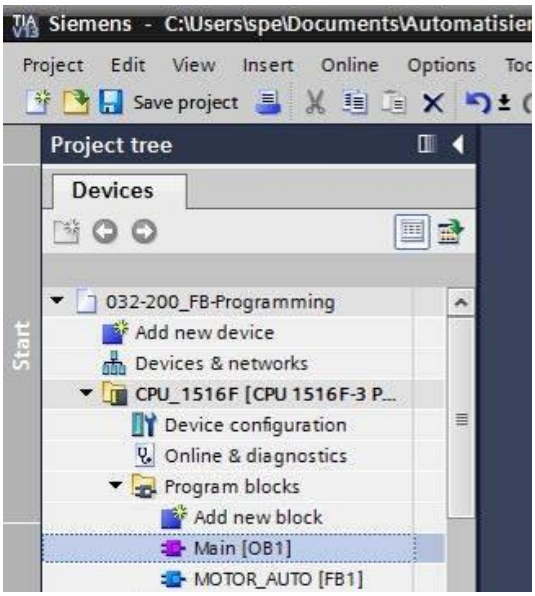
7.8 Programmation du bloc d'organisation OB1 – Commande du convoyeur vers l'avant en mode automatique

→ Avant de programmer le bloc d'organisation "Main[OB1]", nous allons changer le langage de programmation et choisir FBD (LOG) (logigramme). Faire un clic gauche dans le dossier "Program blocks (Blocs de programme)" sur "Main[OB1)".

(→ CPU_1516F[CPU 1516F-3 PN/DP → Program blocks (Blocs de programme) → Main [OB1] → changer le langage de programmation → FBD (LOG))

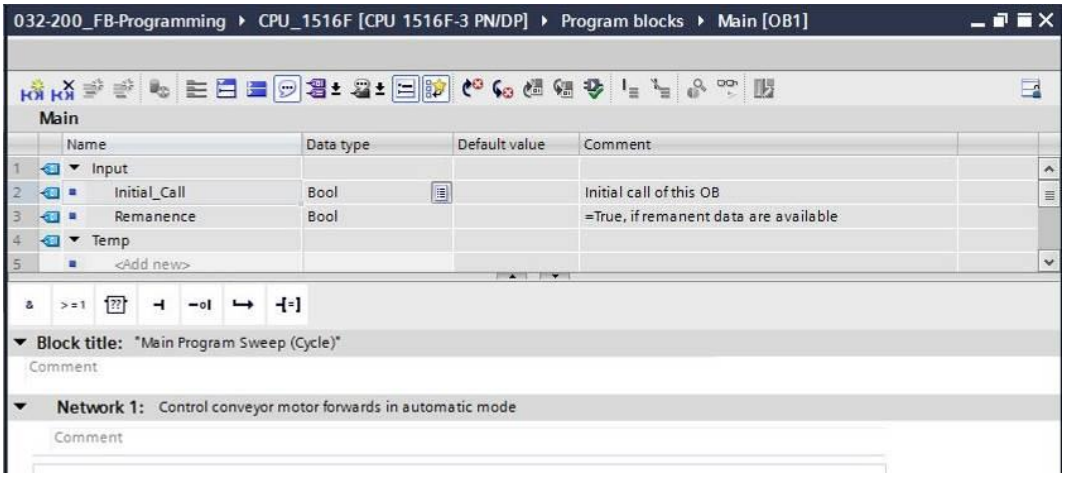


→ Ouvrir le bloc d'organisation "Main [OB1]" par double clic.

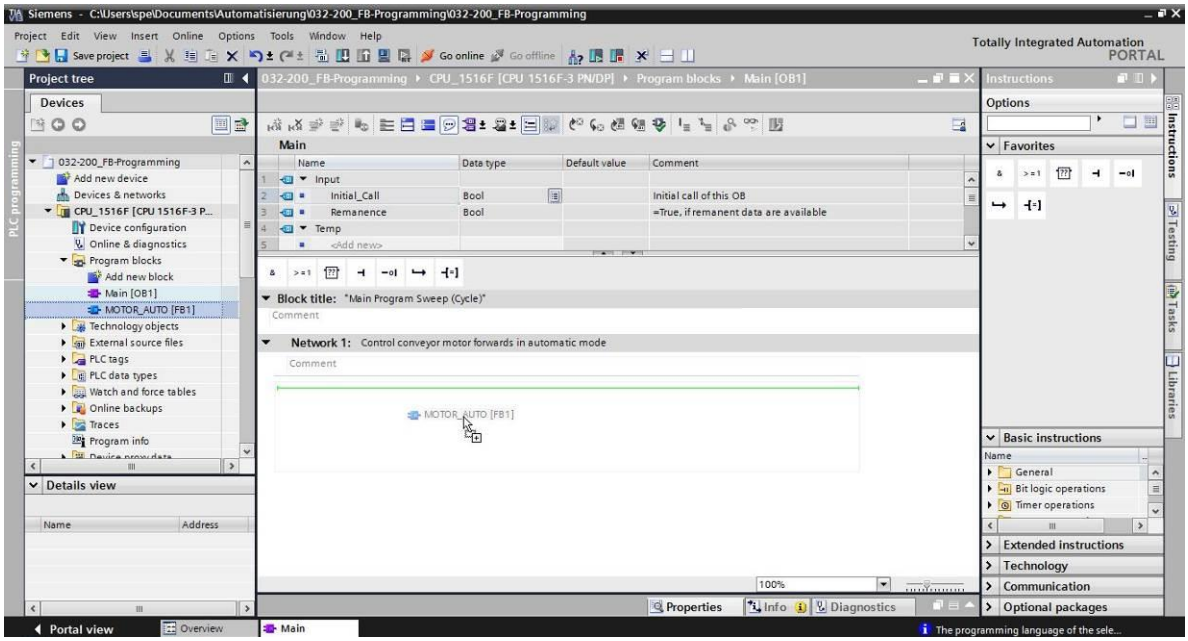


→ Attribuer au réseau 1 le nom "Commande du convoyeur vers l'avant en mode automatique"

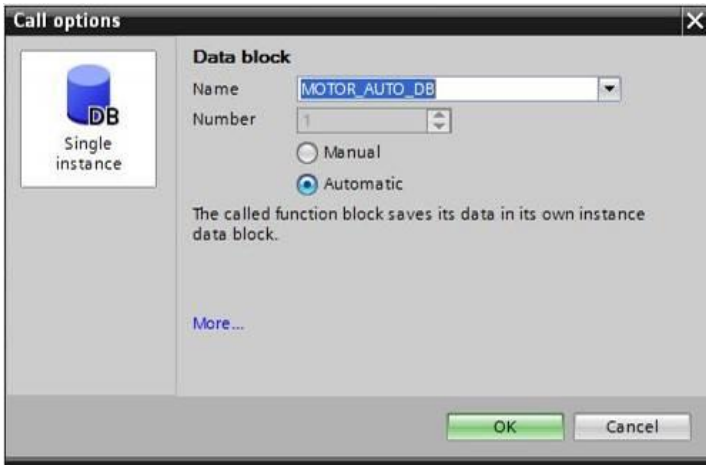
(→Network 1 : ... → Control conveyor motor forwards in automatic mode (Commande du convoyeur vers l'avant en mode automatique))



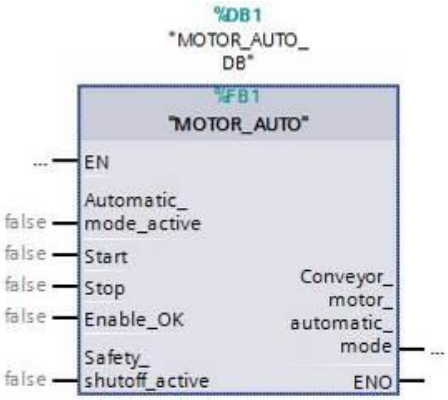
→ Faire glisser le bloc fonctionnel "MOTOR_AUTO [FB1]" dans le réseau 1 sur la ligne verte.



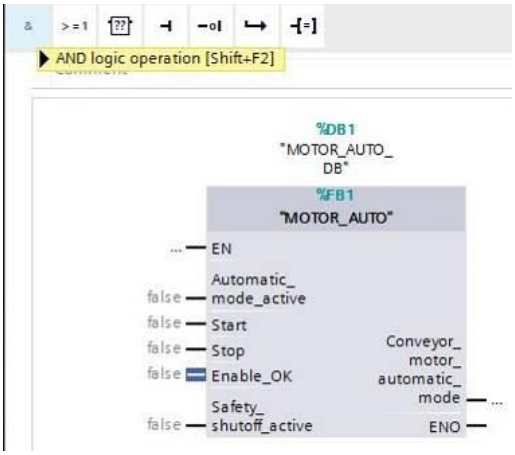
→ Le DB d'instance pour cet appel du FB1 est créé automatiquement. Attribuez un nom et confirmez par "OK". (→ MOTOR_AUTO_DB1 → OK)



→ Un bloc est ajouté au réseau 1. Il contient l'interface que vous avez définie, le bloc de données d'instance, ainsi que les connecteurs EN et ENO.

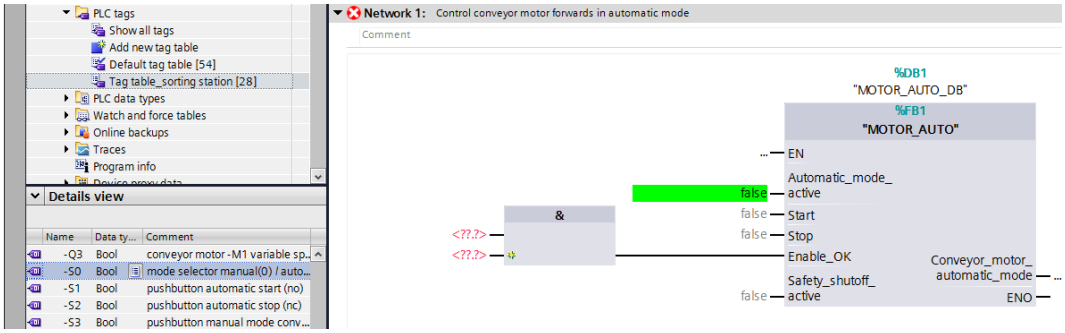


→ Pour ajouter une liaison ET devant le paramètre d'entrée "Enable_OK (validation_ok)“, sélectionner l'entrée et ajouter le ET en cliquant sur & dans la barre d'outils logique. (→ &)

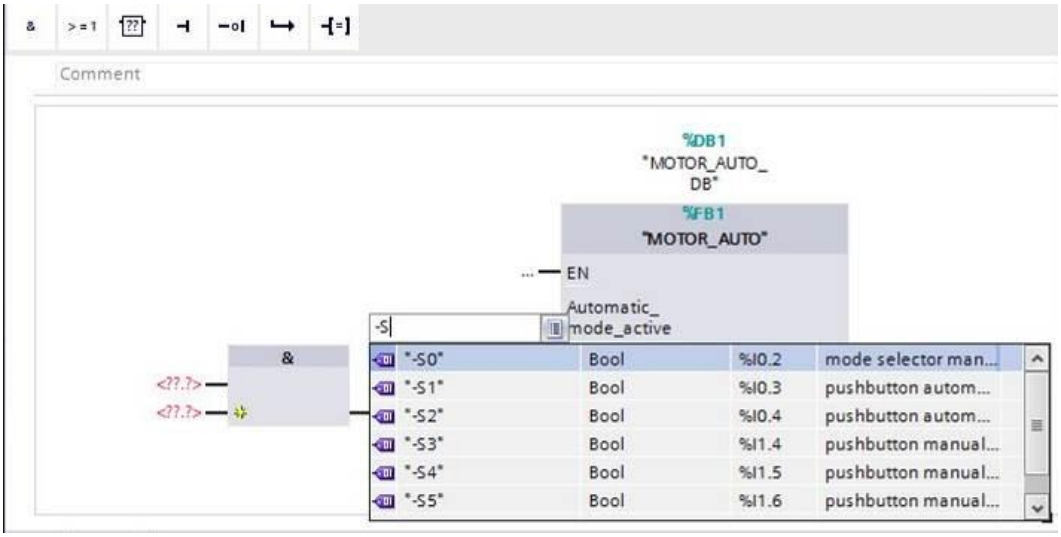


→ Pour connecter le bloc avec les variables globales de la table des variables de l'installation de tri "Tag table_sorting station", deux options sont offertes :

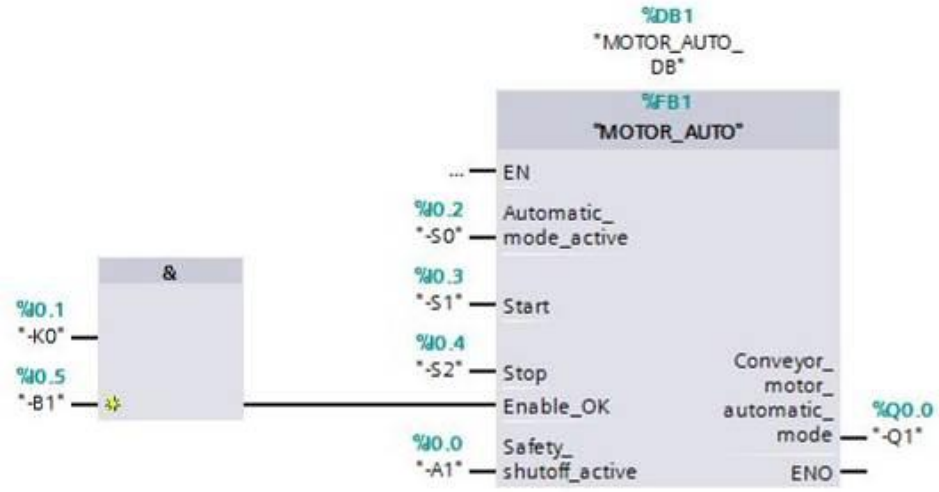
Soit sélectionner dans le navigateur du projet "Tag table_sorting station (table des variables Installation de tri)" et faire glisser la variable globale voulue de la vue de détail sur l'interface du FC1 (→ Tag table_sorting station (table des variables Installation de tri → Details view (vue de détail) → -S0 → automatic_mode_active)



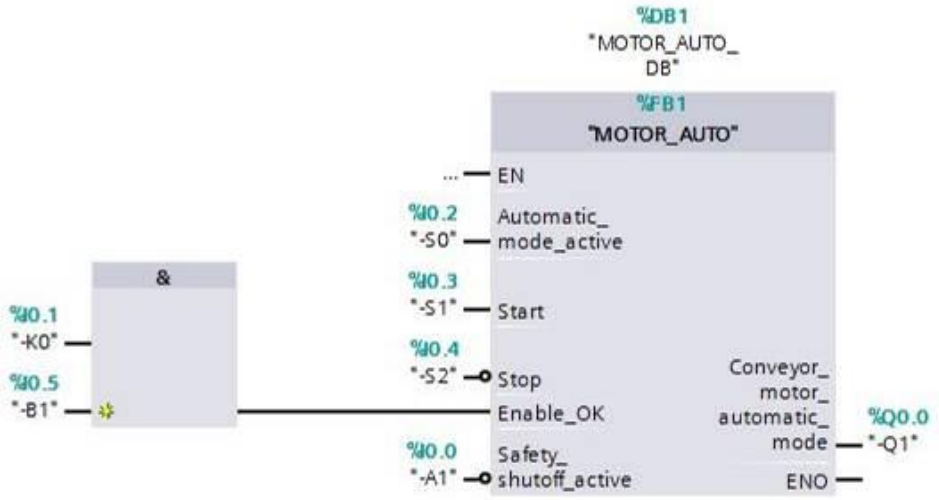
→ Soit saisir sous <??.?> la première lettre de la variable globale voulue et sélectionner sur la liste la variable d'entrée globale "-S0" (%E0.2). (→ automatic_mode_active → -S → -S0)



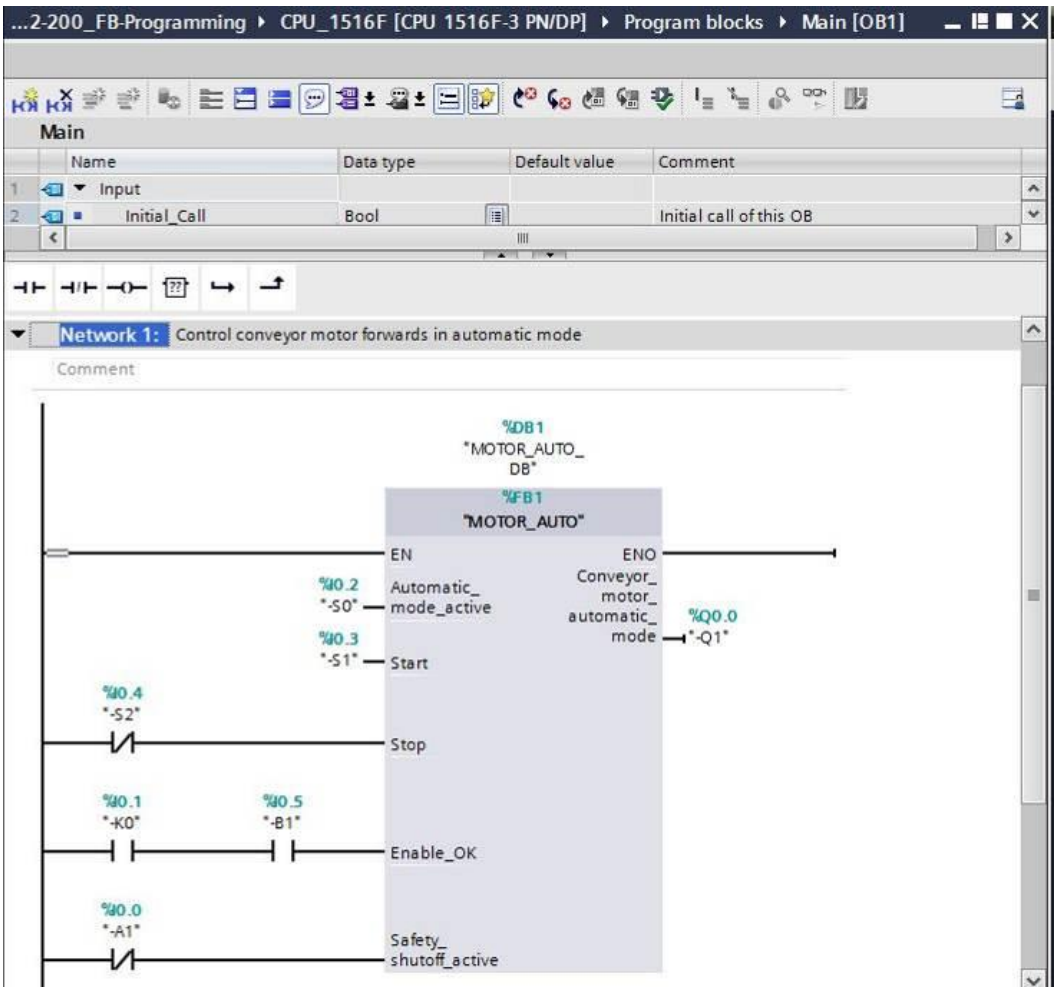
→ Ajouter les autres variables d'entrée "-S1", "-S2", "-K0", "-B1" et "-A1" et sur la sortie "Conveyor_motor_automatic_mode" la variable de sortie "-Q1" (%A0.0).



→ Ajouter une négation aux requêtes des variables d'entrée "-S2" et "-A1" en les sélectionnant, puis en cliquant sur  . (→ -S2 →  → -A1 → )

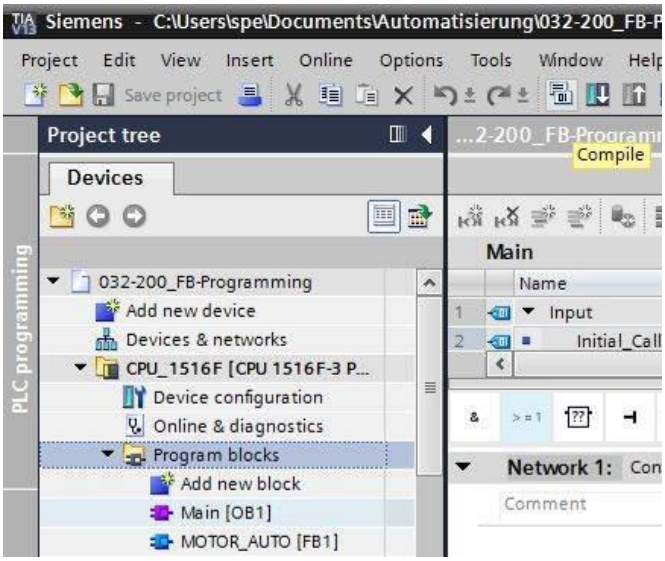


7.9 En langage de programmation LAD (CONT) (schéma à contacts), le résultat est le suivant.

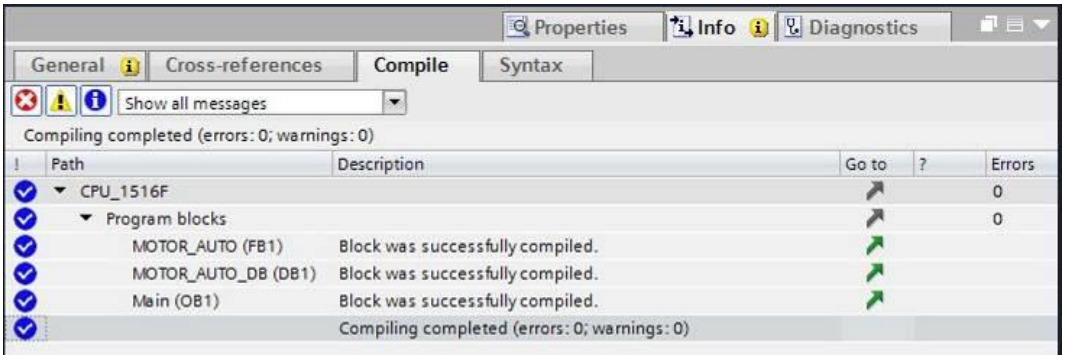


7.10 Enregistrer et compiler le projet

→ Pour enregistrer le projet, sélectionner "Save project" dans le menu. Pour compiler tous les blocs, cliquer sur le dossier "Programm blocks (Blocs de programme)" et dans le menu sur Compile. (→ Save project → Programm blocks (Blocs de programme) →)



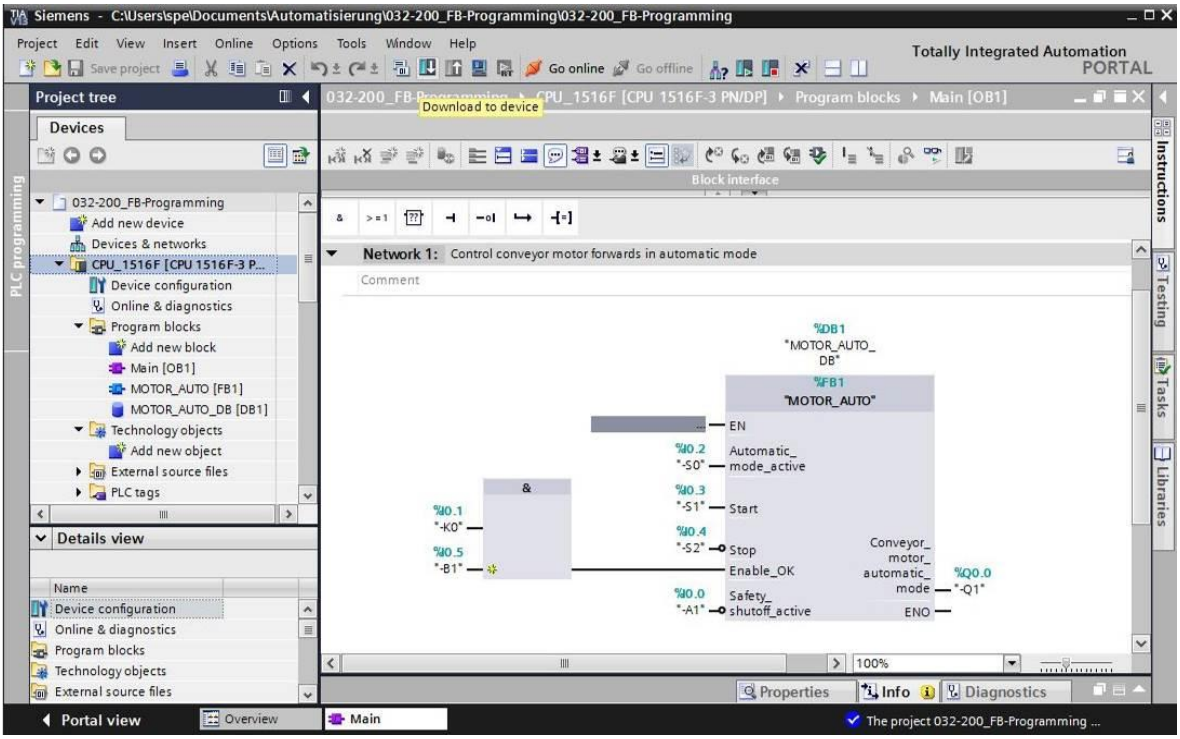
→ Dans la zone "Info" "Compile" les blocs compilés avec succès sont affichés.



7.11 **Charger le programme**

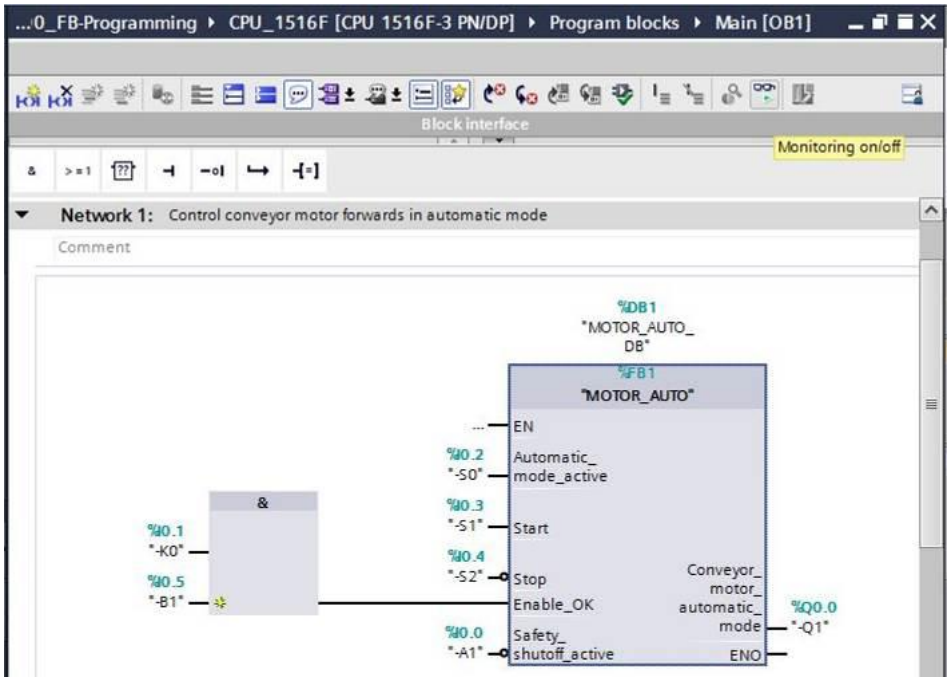
→ Une fois la compilation terminée avec succès, le programme créé peut être chargé dans l'automate comme décrit auparavant dans les modules sur la configuration matérielle.

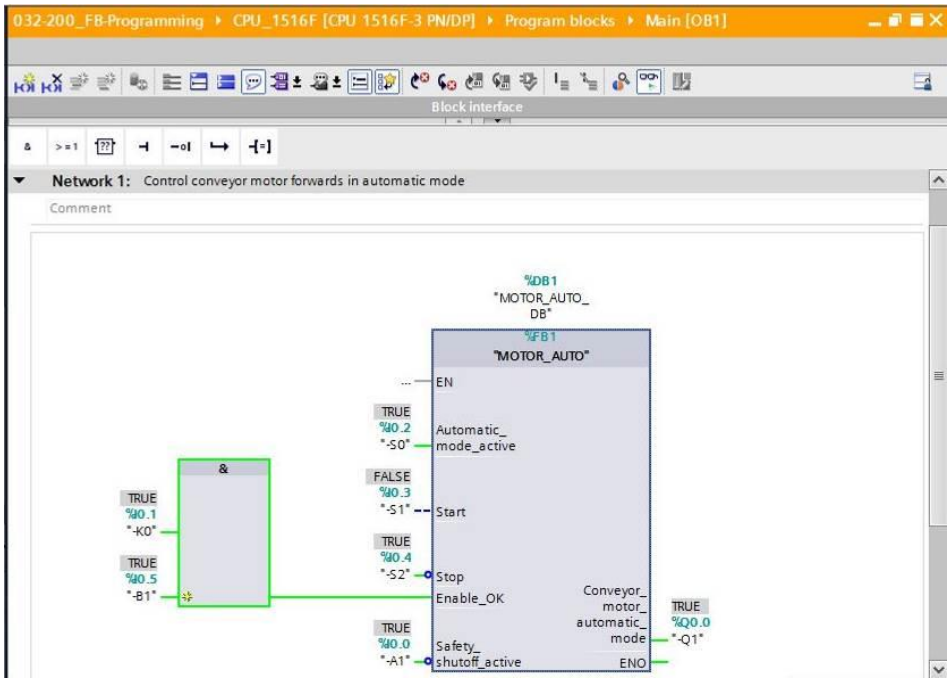
(→)



7.12 Visualiser les blocs de programme

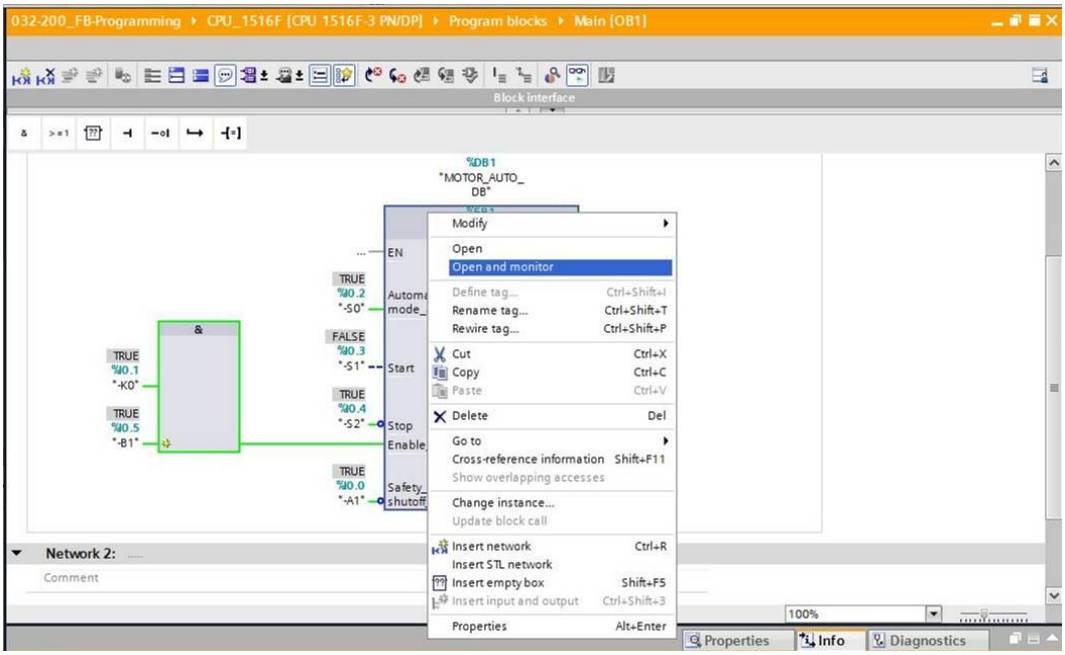
→ Pour visualiser le programme chargé, le bloc voulu doit être ouvert. Ensuite, un clic sur  permet d'afficher ou de masquer la visualisation. (→ Main [OB1] → )

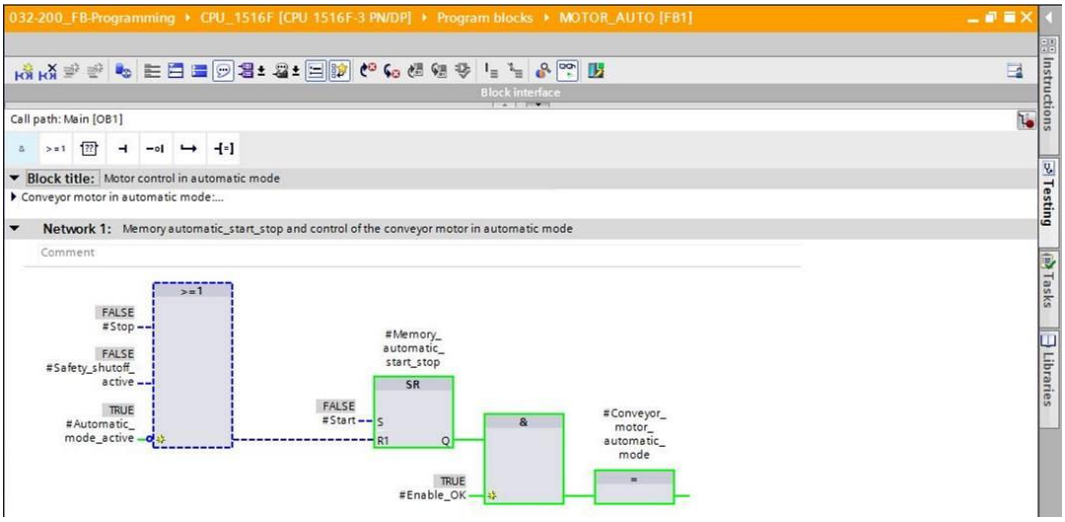




Remarque : *la visualisation s'effectue par signal et par automate. L'état des signaux sur la borne sont signalés par TRUE ou FALSE.*

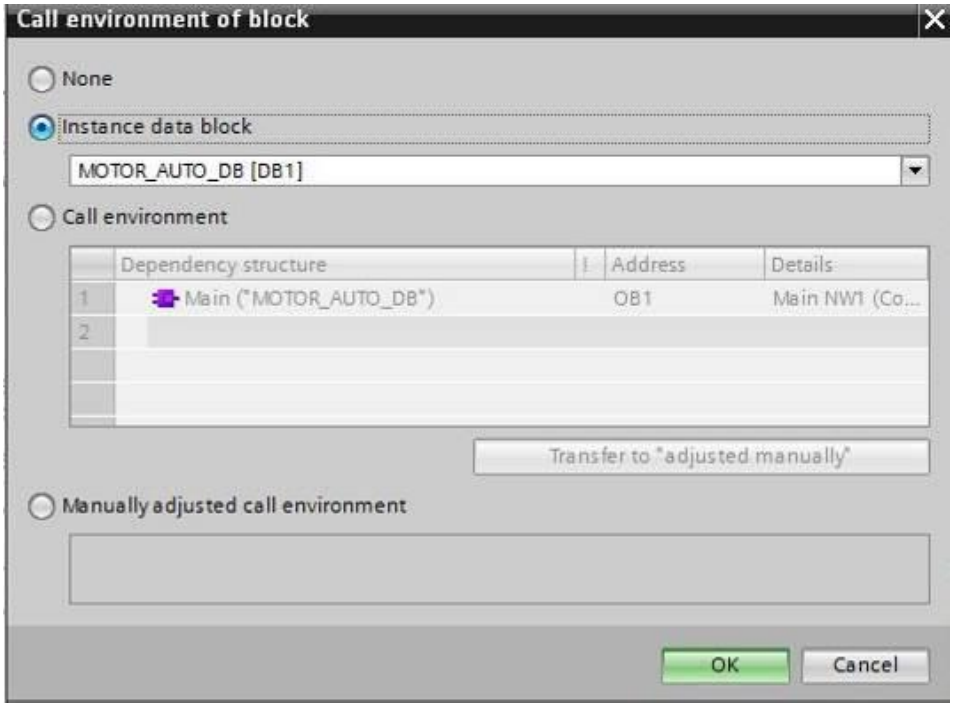
→ Le bloc d'organisation appelé "MOTOR_AUTO" [FB1] dans le bloc d'organisation "Main [OB1]" peut être ouvert et visualisé par clic droit ("Open and monitor").
(→ "MOTOR_AUTO" [FB1] → Open and monitor (ouvrir et visualiser)

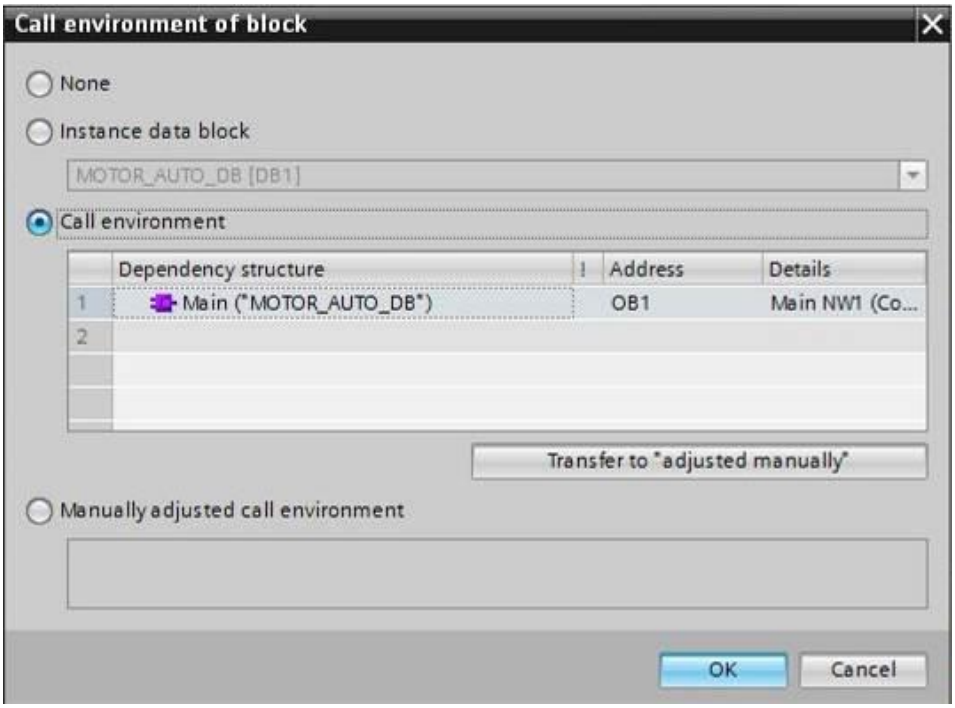




Remarque : *la visualisation s'effectue par fonction et par automate. L'actionnement des capteurs et l'état de l'installation sont signalés par TRUE ou FALSE.*

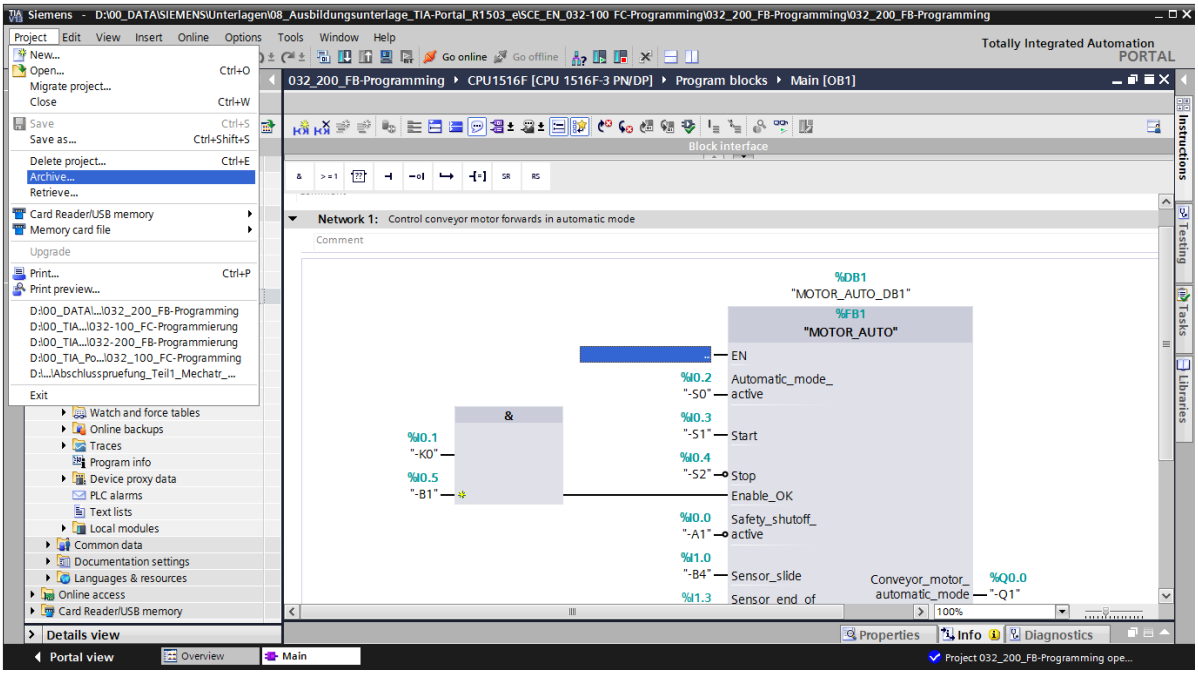
→ Si une occurrence donnée d'un bloc fonctionnel "MOTOR_AUTO" [FB1] appelé plusieurs fois doit être visualisée, utiliser  . Il est possible de définir l'environnement d'appel via l'environnement d'appel ou via le bloc de données d'instance. (→  → Instance data block (bloc de données d'instance) → MOTOR_AUTO_DB1 [DB1] → Call environment (Environnement d'appel) → Address (adresse) : OB1 → Details : Main NW1 → OK)





7.13 Archivage du projet

→ Pour finir, nous voulons archiver le projet complet. Sous la commande de menu → "Project (Projet)" sélectionner → "Archive…". Choisir le dossier d'archivage du projet et l'enregistrer au format "Archive de projet TIA Portal". (→ Project (Projet) → "Archive" → Archive de projet TIA Portal → 032-200_Programmation de FB…. → Save (Enregistrer))



8 Liste de contrôle

Nº	Description	Vérifié
1	Compilation réussie et sans message d'erreur	
2	Chargement réussi et sans message d'erreur	
3	Mettre en marche l'installation (-K0 = 1) Vérin rentré / Réponse activée (-B1 = 1) Arrêt d'urgence (-A1 = 1) non activé Mode AUTOMATIQUE (-S0 = 1) Bouton Arrêt Automatique non actionné (-S2 = 1) Actionner brièvement le bouton Démarrage automatique (-S1 = 1) puis moteur du convoyeur avant vitesse fixe (-Q1 = 1) s'enclenche et reste en marche.	
4	Actionner brièvement le bouton arrêt automatique (-S2 = 0) → -Q1 = 0	
5	Activer l'arrêt d'urgence (-A1 = 0) → -Q1 = 0	
6	Mode manuel (-S0 = 0) → -Q1 = 0	
7	Éteindre l'installation (-K0 = 0) → -Q1 = 0	
8	Vérin non rentré (-B1 = 0) → -Q1 = 0	
9	Le projet a été archivé avec succès	

9 Exercice

9.1 Énoncé du problème - exercice

Dans cet exercice, vous allez compléter le bloc fonctionnel MOTOR_AUTO [FB1] en lui ajoutant une fonction d'économie d'énergie. Le bloc fonctionnel ainsi complété doit être planifié, programmé et testé :

Par économie d'énergie, le convoyeur ne doit fonctionner que si une pièce se trouve dessus.

La sortie moteur_automatique est activée si Memory_automatic_start_stop (mémoire_automatique_marche_arrêt) est mis à 1, les conditions de validation sont remplies et Memory_conveyor_start_stop (mémoire_convoyeur_marche_arrêt) mis à 1.

De ce fait, Memory_conveyor_start_stop (mémoire_convoyeur_marche_arrêt) est mis à 1 si capteur_toboggan_occupé signale une pièce et mis à 0 si Sensor_end_of_conveyor (capteur_fin_de_convoyeur) crée un front descendant, si le disjoncteur est actif ou si le mode automatique n'est pas activé (mode manuel).

9.2 Planification

Planifiez seul la réalisation de l'énoncé.

Remarque : *voir l'aide en ligne pour l'utilisation des fronts négatifs dans SIMATIC S7-1500.*

9.3 Liste de contrôle - Exercice

Nº	Description	Vérifié
1	Compilation réussie et sans message d'erreur	
2	Chargement réussi et sans message d'erreur	
3	Mettre en marche l'installation (-K0 = 1) Vérin rentré / Réponse activée (-B1 = 1) Arrêt d'urgence (-A1 = 1) non activé Mode AUTOMATIQUE (-S0 = 1) Bouton Arrêt Automatique non actionné (-S2 = 1) Actionner brièvement le bouton Démarrage automatique (-S1 = 1) Capteur toboggan affecté activé (-B4 = 1) puis moteur du convoyeur avant vitesse fixe (-Q1 = 1) s'enclenche et reste en marche.	
4	Capteur convoyeur fin activé (-B7 = 1) → -Q1 = 0	
5	Actionner brièvement le bouton arrêt automatique (-S2 = 0) → -Q1 = 0	
6	Activer l'arrêt d'urgence (-A1 = 0) → -Q1 = 0	
7	Mode manuel (-S0 = 0) → -Q1 = 0	
8	Éteindre l'installation (-K0 = 0) → -Q1 = 0	
9	Vérin non rentré (-B1 = 0) → -Q1 = 0	
10	Le projet a été archivé avec succès	

10 Informations complémentaires

Des informations complémentaires vous sont proposées afin de vous aider à vous exercer ou à titre d'approfondissement, par ex. : mises en route, vidéos, didacticiels, applis, manuels, guides de programmation et logiciel/firmware d'évaluation sous le lien suivant :

www.siemens.com/sce/s7-1500