SIEMENS

Support d’apprentissage/ de formation

Siemens Automation Cooperates with Education (SCE) | A partir de la version V14 SP1

Module 031-200 TIA Portal
Principes de base de la programmation de FB avec SIMATIC S7-1200

siemens.com/sce

SIEMENS
Global Industry Partner of WorldSkills International

worldskills

Packages SCE pour formateurs adaptés à ces supports d’apprentissage/de formation

- **SIMATIC S7-1200 AC/DC/RELAIS (paquet de 6) "TIA Portal"**
  N° d'article : 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (paquet de 6) "TIA Portal"**
  N° d'article : 6ES7214-1AE30-4AB3
- **SIMATIC STEP 7 BASIC V14 SP1 Upgrade (for S7-1200) (paquet de 6) "TIA Portal"**
  N° d'article : 6ES7822-0AA04-4YE5

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

Le support d’apprentissage/de formation SCE pour une solution d'automatisation cohérente Totally Integrated Automation (TIA) ont été créés spécialement 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 SA n’assume aucune responsabilité quant au contenu.

Cette documentation ne peut être utilisée que pour une première formation aux produits/systèmes Siemens. Ce qui veut dire qu’elle peut être copiée, en partie ou dans son 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 copie de cette documentation, son exploitation et la communication de son contenu sont autorisés dans le cadre d’instituts publics de formation et de formation continue.

Toute exception requiert au préalable l’autorisation écrite de la part des interlocuteurs de Siemens SA : 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 la TU de Dresde, notamment le professeur Leon Urbas et l’entreprise Michael Dziallas Engineering ainsi que toutes les personnes ayant contribué à la réalisation de ce support d’apprentissage/de formation.

Sommaire

Principes de base de la programmation de FB

1 Objectif

Dans le présent chapitre, vous allez vous familiariser avec les éléments de base d'un programme : les ***blocs d'organisation (OB),*** les ***fonctions (FC)***, les **blocs fonctionnels (FB)** et les **blocs de données (DB).** Par ailleurs, nous vous montrerons comment programmer les fonctions et blocs fonctionnels compatibles avec la ***bibliothèque***. Vous allez vous familiariser avec le langage de programmation ***logigramme (LOG)*** et utiliser celui-ci pour programmer un bloc fonctionnel FB1 et un bloc d'organisation OB1.

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

2 Condition

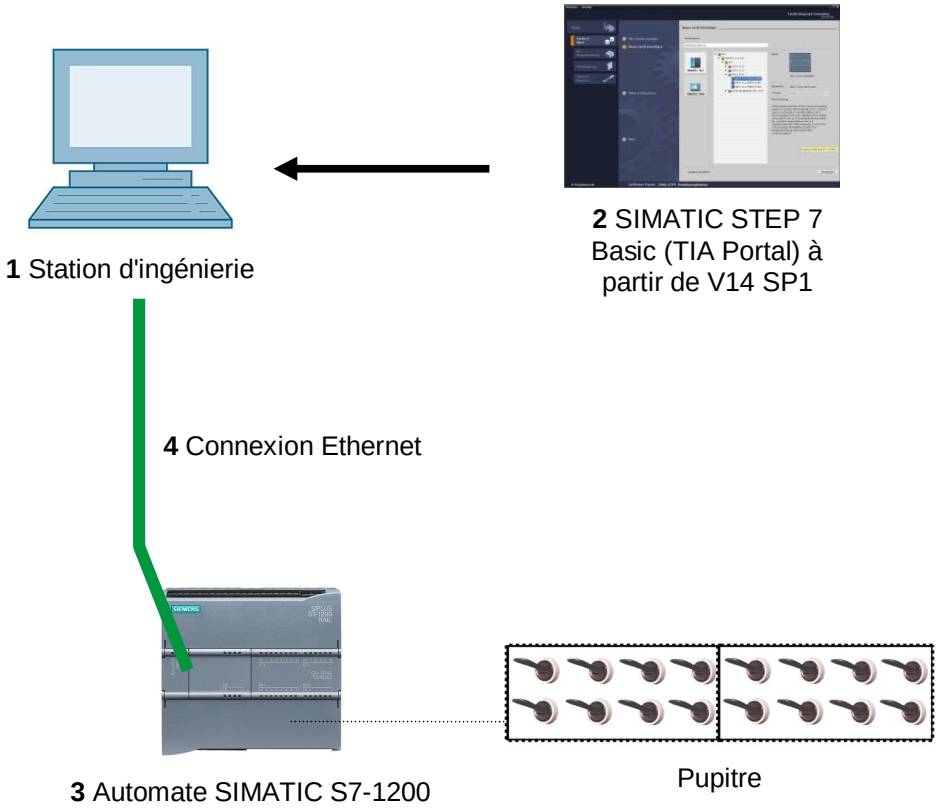
Ce chapitre s'appuie sur la configuration matérielle de la CPU1214C SIMATIC S7. Toutefois, il peut également être réalisé avec toute autre configuration matérielle munie d'une carte d'entrée/sortie TOR. Pour l'étude de ce chapitre, vous pouvez par ex. recourir au projet suivant :

SCE_FR_011_101_Configuration matérielle_CPU1214C.zap14

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 Basic dans TIA Portal – à partir de V14
3. Automate SIMATIC S7-1200, par exemple CPU 1214C DC/DC/DC avec Signal Board ANALOG OUTPUT SB1232, 1 AO – à partir du firmware V4.2.1

   Remarque : les entrées TOR doivent être mises en évidence sur un pupitre.
4. Connexion Ethernet entre la station d'ingénierie et l'automate



4 Partie théorique

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 du processus, c'est-à-dire commande des signaux de sortie en fonction des états des signaux d'entrée

– Réaction aux alarmes et entrées d'alarmes

– 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 gère les opérations suivantes :

– Traitement cyclique du programme (par ex. OB1)

– Comportement au démarrage de l'automate

– Traitement du programme déclenché par alarme

– Traitement des erreurs

Un projet doit contenir au moins ***un bloc d'organisation pour le traitement cyclique du programme***. Un OB est appelé par un ***évènement déclencheur*** comme représenté dans la Figure 1. Des priorités sont définies pour les différents OB afin que l'OB1 cyclique puisse par exemple être interrompu par un OB82 pour le traitement des erreurs.

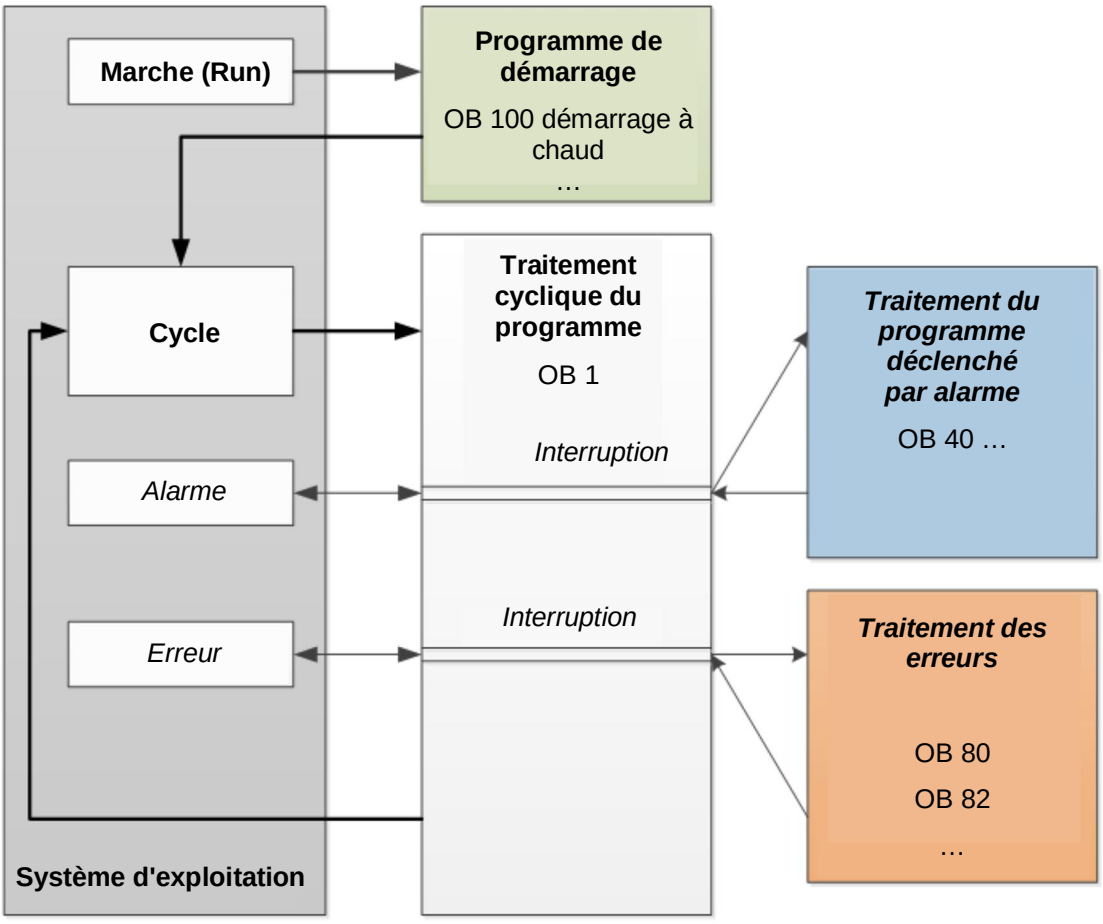


Figure 1 : Évènements déclencheurs dans le système d'exploitation et appels de OB

Les réactions suivantes sont possibles après qu'un évènement déclencheur s'est produit :

– Si vous avez affecté un OB à l'événement, il déclenchera l'exécution de l'OB affecté. Si la priorité de l'OB affecté est plus élevée que celle de l'OB en cours d'exécution, celui-ci est immédiatement exécuté (interruption). Si ce n'est pas le cas, le système attend d'abord jusqu'à ce que l'exécution de l'OB avec la priorité plus élevée soit terminée.
– Si l'événement n'est affecté à aucun OB, la réaction système par défaut est exécutée.

Le tableau 1 ci-dessous montre différents exemples d'évènements déclencheurs pour un SIMATIC S7-1200. Il contient aussi des numéros d'OB possibles et les réactions système prédéfinies qui sont exécutées lorsque le bloc d'organisation (OB) correspondant n'est pas présent dans l'automate.

Évènement déclencheur	**Numéros d'OB possibles**	**Réaction système prédéfinie**
Mise en route	100, ³ 123	Ignorer
Programme cyclique	1, ³ 123	Ignorer
Alarme horaire	10 à 11	-
Alarme de mise à jour	56	Ignorer
Temps de cycle imparti dépassé une fois	80	Ignorer
Temps de cycle imparti dépassé deux fois	80	STOP
Alarme de diagnostic	82	Ignorer

Tableau 1 : Numéros d'OB pour différents évènements déclencheurs

4.3 Mémoire image et traitement cyclique du programme

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

Le traitement cyclique du programme se déroule comme suit :

1. Au début du programme cyclique, le système vérifie si chacune des entrées est sous tension ou non. 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 alors 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 transférée comme état logique aux modules de sorties et celles-ci sont activées ou désactivées. La procédure reprend ensuite à partir du point 1.

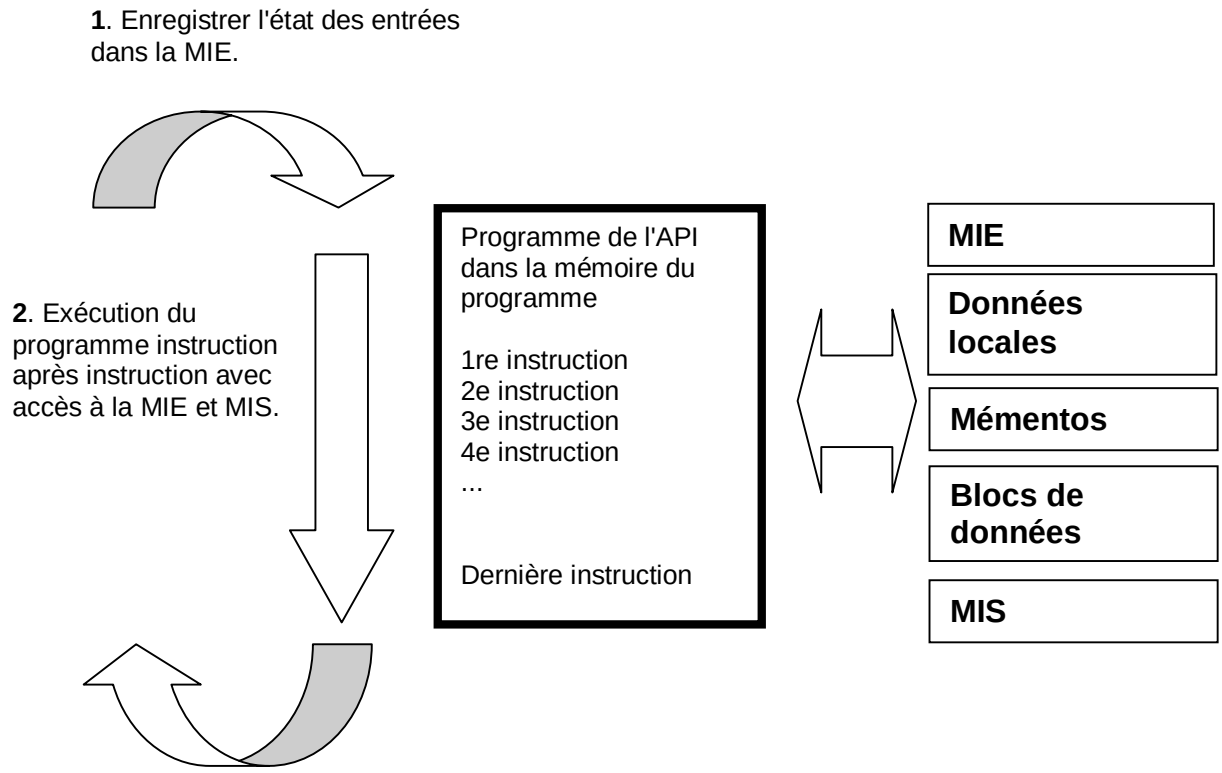


Figure 2 : Traitement cyclique du programme

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 interconnectés lors de l'appel d'une fonction. Des blocs de données globaux doivent être crées pour stocker durablement les données.

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 :

– Retourner un résultat dépendant des valeurs d'entrée pour les fonctions mathématiques.

– Exécuter des fonctions technologiques comme des commandes uniques avec combinaisons binaires.

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

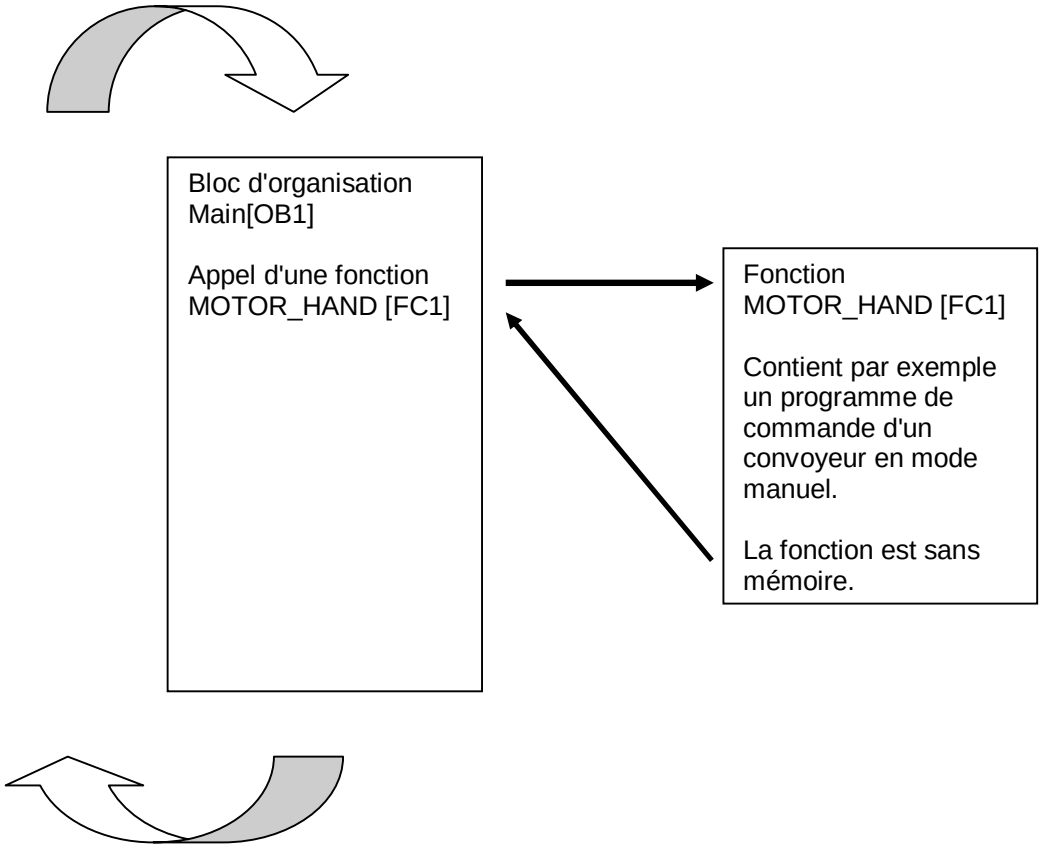


Figure 3 : Fonction avec appel d'un bloc d'organisation Main [OB1]

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 et d'entrée/sortie ainsi que leurs 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 le DB d'instance mais disponibles uniquement tout 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 ou

– toujours quand une information doit être enregistrée dans le programme. Par ex. un indicatif de mode de fonctionnement avec un bouton.

Les FB sont toujours exécutés quand un bloc fonctionnel est appelé par un autre bloc de code. Un FB 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 dans un bloc de données sous forme de **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.

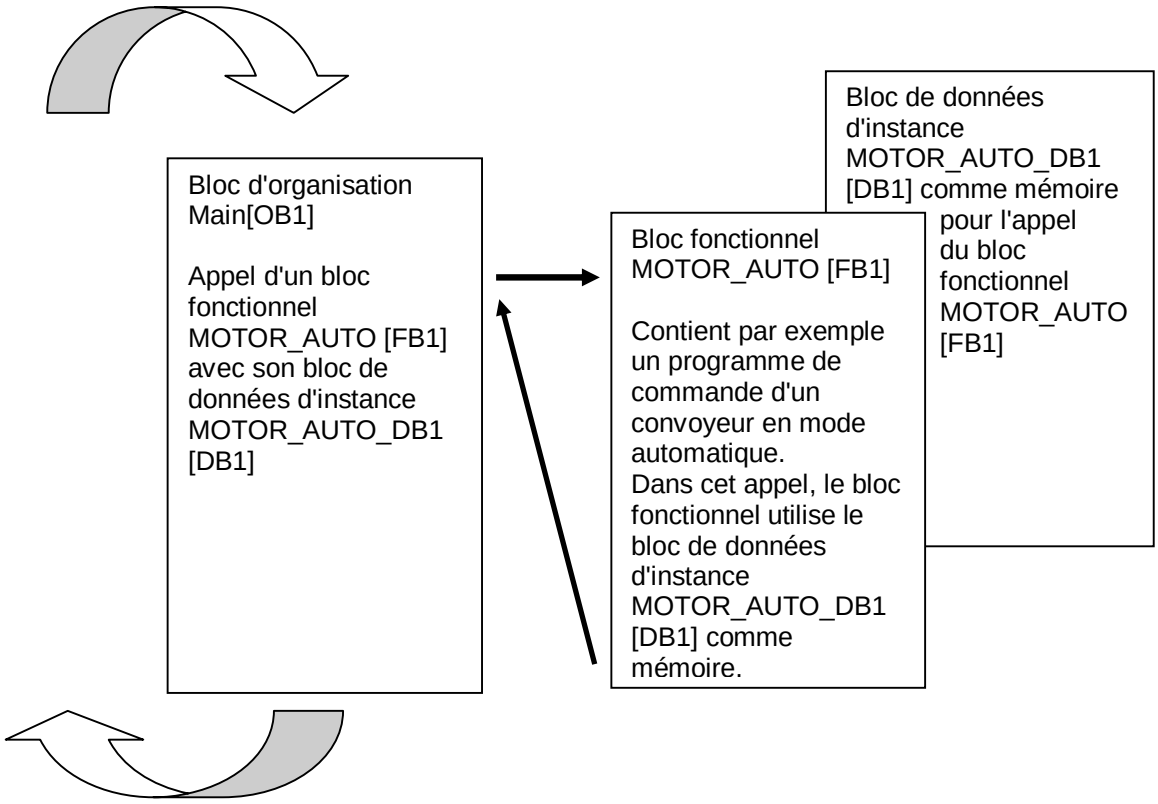


Figure 4 : Bloc fonctionnel et instance avec appel d'un bloc d'organisation Main [OB1]

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 stockent des données qui peuvent être utilisés ***par tous les autres blocs*** (voir figure 5). L'accès aux blocs de données d'instance doit être réservé au bloc fonctionnel correspondant. La taille maximale des blocs de données varie selon la CPU.

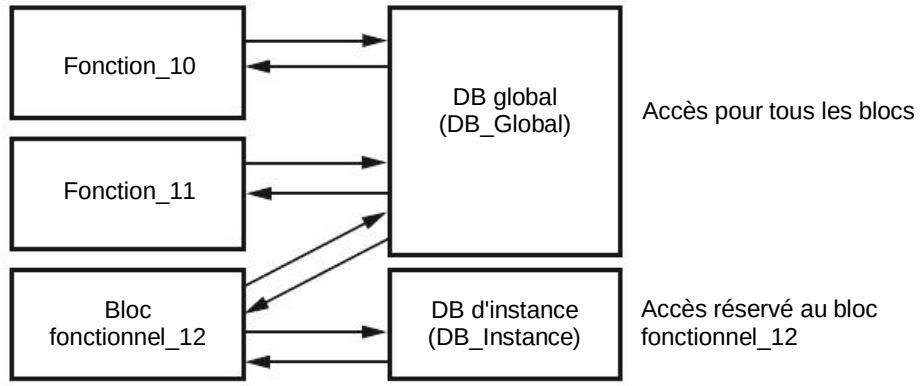


Figure 5 : Différence entre bloc de données global et bloc de données d'instance.

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 compatibles avec la bibliothèque

Un programme utilisateur peut être créé de façon linéaire ou structurée. La ***programmation linéaire*** consiste à écrire le programme utilisateur complet dans l'OB de cycle. Cela n'est toutefois recommandé que pour des programmes simples pour lesquels on utilise désormais d'autres systèmes de commande plus économique telle que LOGO!

Une ***programmation structurée*** est recommandée pour des programmes plus complexes. Vous pouvez subdiviser la tâche d'automatisation complexe en plusieurs petites tâches partielles à réaliser par des fonctions et blocs fonctionnels.

Il convient de créer des blocs de code compatibles avec la bibliothèque pour cela. Autrement dit, les paramètres d'entrée et les paramètres de sortie d'une fonction ou d'un bloc fonctionnel sont définis de manière générale et les variables globales actuelles (entrées/sorties) ne leurs sont attribuées que lors de l'utilisation du bloc.

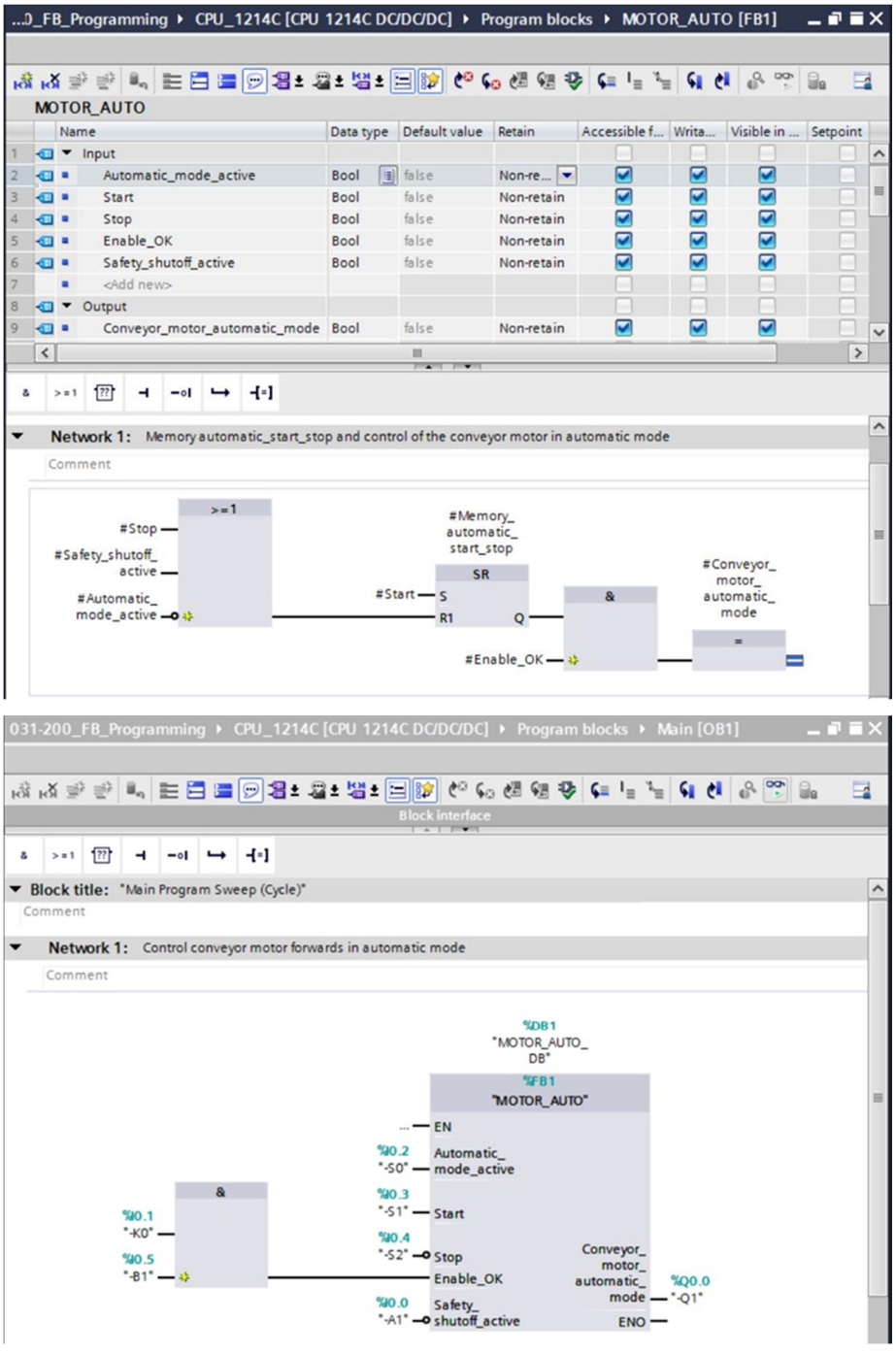


Figure 6 : Bloc fonctionnel compatible avec la bibliothèque avec appel dans OB1

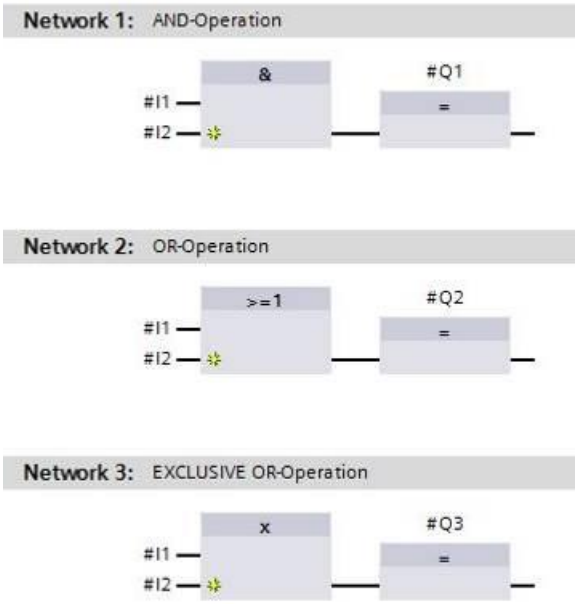
4.8 Langages de programmation

Les langages de programmation logigramme (LOG), schéma à contacts (CONT) et Structured Control Language (SCL) sont disponibles pour la programmation de fonctions et blocs fonctionnels pour l'automate SIMATIC S7-1200.

Le langage de programmation ***logigramme (LOG)*** est expliqué ci-après.

LOG est un langage de programmation graphique. La représentation est inspirée des systèmes de circuits électroniques. Le programme est représenté dans divers 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, 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 comme représenté dans la Figure 7 ci-dessous.



#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 dans LOG et table logique correspondante

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

Des éléments de programme comme des temporisations CEI et des compteurs CEI mettent à disposition des instructions plus complexes.

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

Mécanisme d'entrée de validation EN (enable)/sortie de validation ENO (ENable Output) :

– Une instruction sans mécanisme EN/ENO est exécutée indépendamment de l'état logique au niveau des entrées de la boîte.

– Des 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". Si des erreurs se produisent en cours de traitement, la sortie de validation ENO est remise à zéro. Si l'entrée de validation EN n'est pas imbriquée, la boite est toujours exécutée.

5 Application à réaliser

Dans le présent chapitre, nous voulons planifier, programmer et tester les fonctions de description du processus Installation de tri suivantes :

– Mode automatique – Moteur du convoyeur

6 Planification

Il n'est pas conseillé de programmer toutes les fonctions dans l'OB1 par souci de clarté et pour ne pas restreindre les possibilités de réutilisation. C'est pourquoi le code du programme est principalement contenu dans des fonctions (FC) et des blocs fonctionnels (FB). La décision visant à déterminer les fonctions à affecter au FB et celles à exécuter dans l'OB1 sera planifié ci-après.

6.1 ARRÊT D'URGENCE

L'ARRÊT D'URGENCE ne requiert pas une fonction propre. Tout comme le mode de fonctionnement, l'état actuel du relais ARRÊT D'URGENCE peut être utilisé directement sur les blocs.

6.2 Mode automatique – Moteur du convoyeur

Nous voulons emboîter le mode automatique du moteur du convoyeur dans un bloc fonctionnel (FB) „MOTOR_AUTO“. D'une part, cela permet d'assurer la clarté de l'OB1 et, d'autre part, de conserver les possibilités de réutilisation en cas d'extension de l'installation avec un convoyeur supplémentaire. Le tableau 2 ci-après contient les paramètres planifiés.

Input	Type de données	Commentaire
Automatique_activé	BOOL	Mode de fonctionnement automatique activé
Start	BOOL	Commande de démarrage du mode automatique
Stop	BOOL	Commande d'arrêt du mode automatique
Validation_OK	BOOL	Toutes les conditions de validation sont remplies
Arrêt automatique de sécurité_activé	BOOL	Dispositif d'arrêt automatique de sécurité, par ex. arrêt d'urgence actionné
Output		
Convoyeur_moteur_automatique	BOOL	Commande du moteur du convoyeur en mode automatique
Static		
Mémoire_automatique_démarrage_arrêt	BOOL	Mémoire pour la fonction de démarrage/arrêt dans le mode automatique

Tableau 2 : Paramètre du FB "MOTOR_AUTO"

Mémoire_automatique_démarrage_arrêt est déclenché et mémorisé par la commande Start, mais seulement à condition qu'aucun signal de réinitialisation ne soit présent.

Mémoire_automatique_démarrage_arrêt est réinitialisé lorsque le signal Stop est présent, le dispositif d'arrêt automatique de sécurité est activé ou que le mode automatique n'est pas activé (mode manuel).

La sortie Convoyeur_moteur_automatique est commandée lorsque le signal Mémoire_ automatique_démarrage_arrêt est mis à 1 et les conditions de validation sont remplies

6.3 Schéma technologique

La figure ci-dessous montre le schéma technologique pour l'application à réaliser.

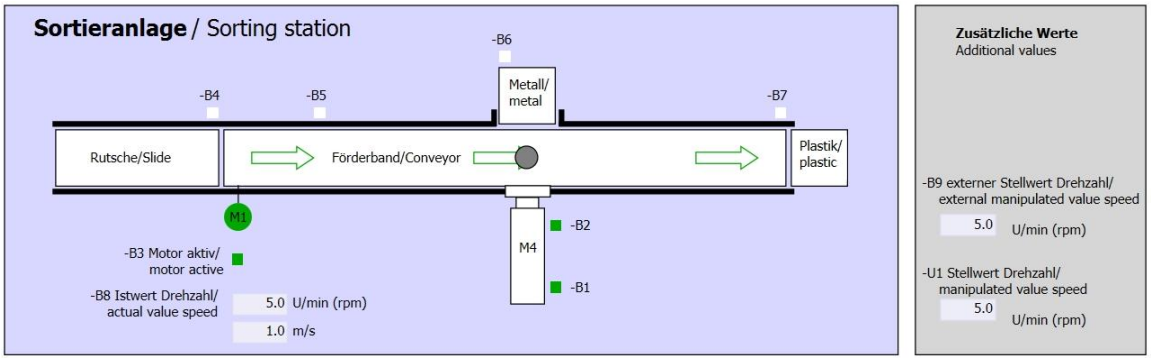


Figure 8 : Schéma technologique

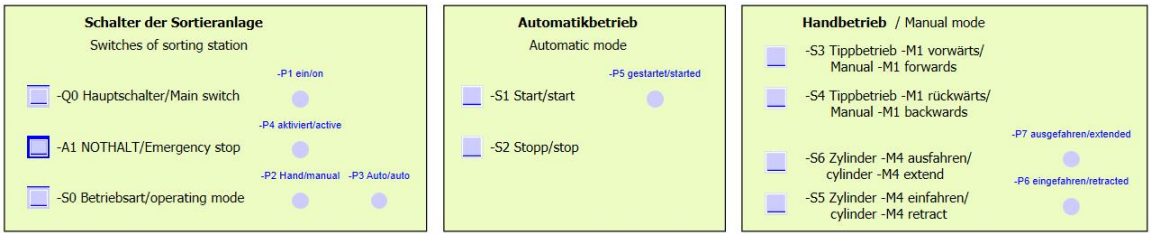


Figure 9 : Pupitre de commande

6.4 Tableau d'affectations

Cette application requiert les signaux suivants comme opérande.

DE	Type	Code	Fonction	NC/NO
E 0.0	BOOL	-A1	Message ARRET D'URGENCE ok :	NC
E 0.1	BOOL	-K0	Installation "Marche"	NO
E 0.2	BOOL	-S0	Commutateur mode Manuel (0)/ Automatique (1)	Manuel = 0 Auto=1
E 0.3	BOOL	-S1	Bouton démarrage automatique	NO
E 0.4	BOOL	-S2	Bouton arrêt automatique	NC
E 0.5	BOOL	-B1	Capteur tige du vérin -M4 rentrée	NO

DA	Type	Code	Fonction	
A 0.0	BOOL	-Q1	Moteur du convoyeur -M1 avance à vitesse fixe	

Legende zur Belegungsliste

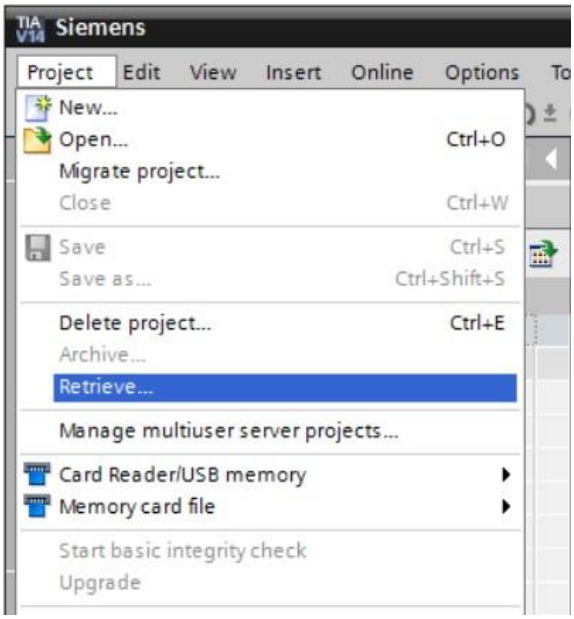
DE	Entrée TOR	DA	Sortie TOR
AE	Entrée analogique	AA	Sortie analogique
E	Entrée	A	Sortie
NC	Normally Closed (contact à ouverture)		
NO	Normally Open (contact à fermeture)		

7 Marche à suivre détaillée

Vous trouverez ci-après une description étape par étape de la marche à suivre pour la planification. Si vous vous en sortez déjà bien, vous pouvez vous contenter des numéros correspondant aux étapes pour réaliser l'application. Sinon, il vous suffit de suivre la procédure détaillée décrite ci-dessous.

7.1 Désarchiver un projet existant

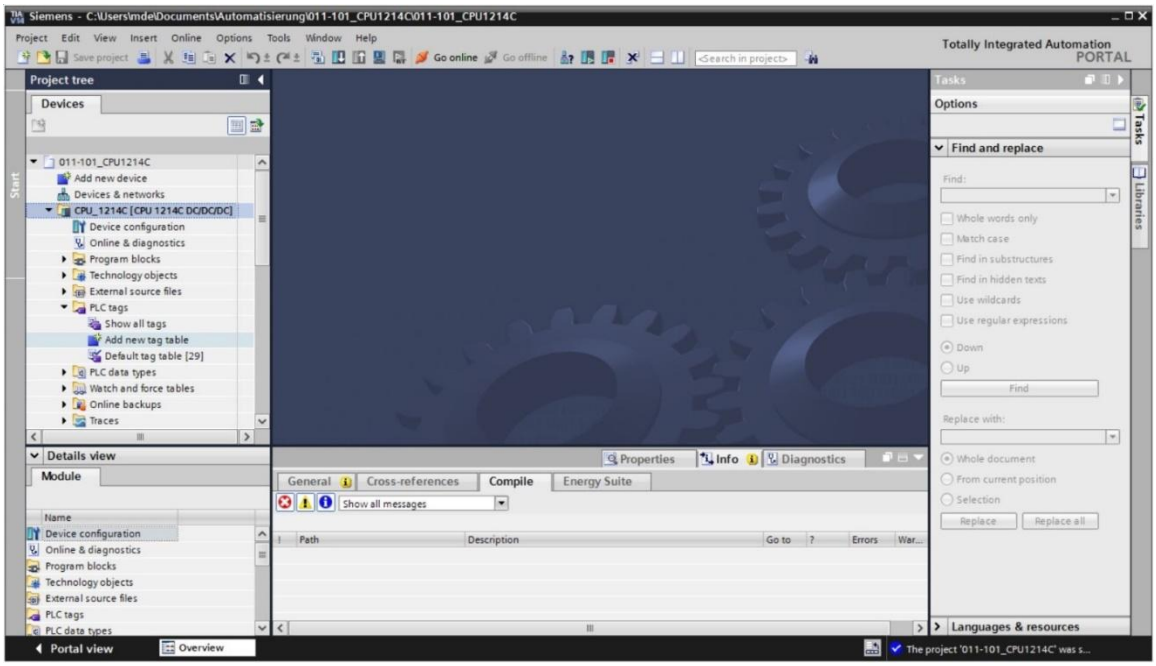
® Avant de commencer la programmation du bloc fonctionnel (FB) „MOTOR_AUTO“, nous avons besoin d'un projet avec une configuration matérielle (par ex. SCE_FR_011-101_Configuration matérielle_S7-1214C….zap). Pour désarchiver un projet existant, vous devez sélectionner l'archive correspondant sous ® Project (Projet) ® (Retrieve) Extraire dans la vue du projet. Confirmez votre sélection avec Ouvrir. (® Project (Projet) ® (Retrieve) Extraire ® Sélection d'une archive .zap ® Open (Ouvrir))



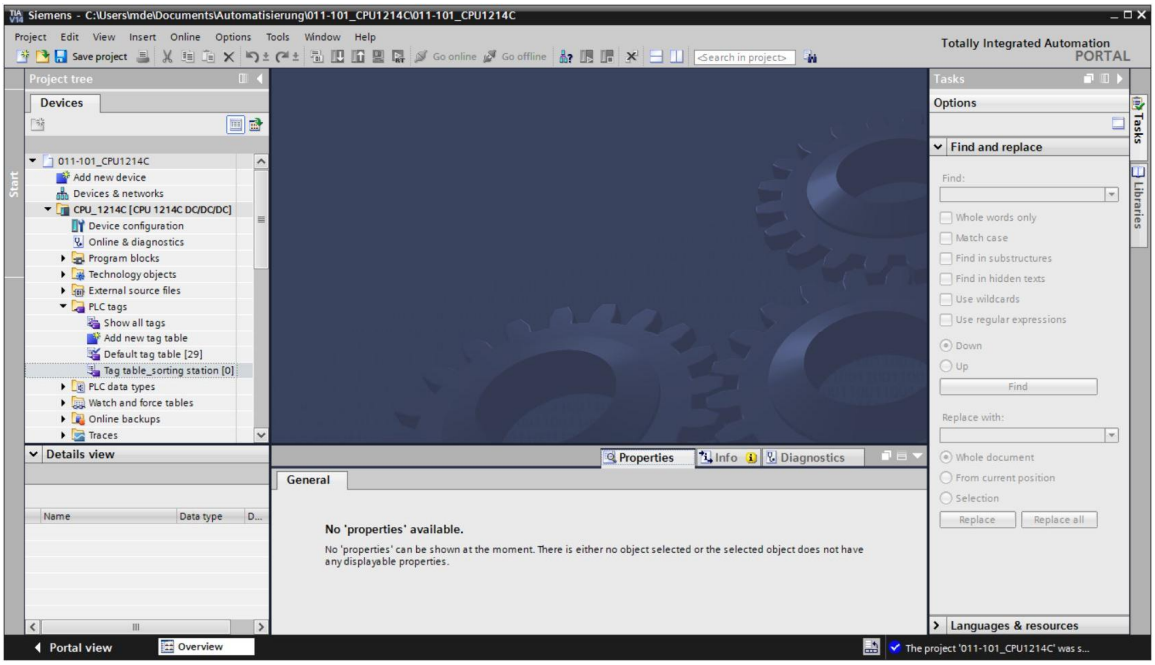
® Ensuite, vous pouvez sélectionner le répertoire cible dans lequel vous souhaitez enregistrer le projet désarchivé. Confirmez votre sélection avec "OK". (® Répertoire cible ® OK)

7.2 Création d'une nouvelle table des variables

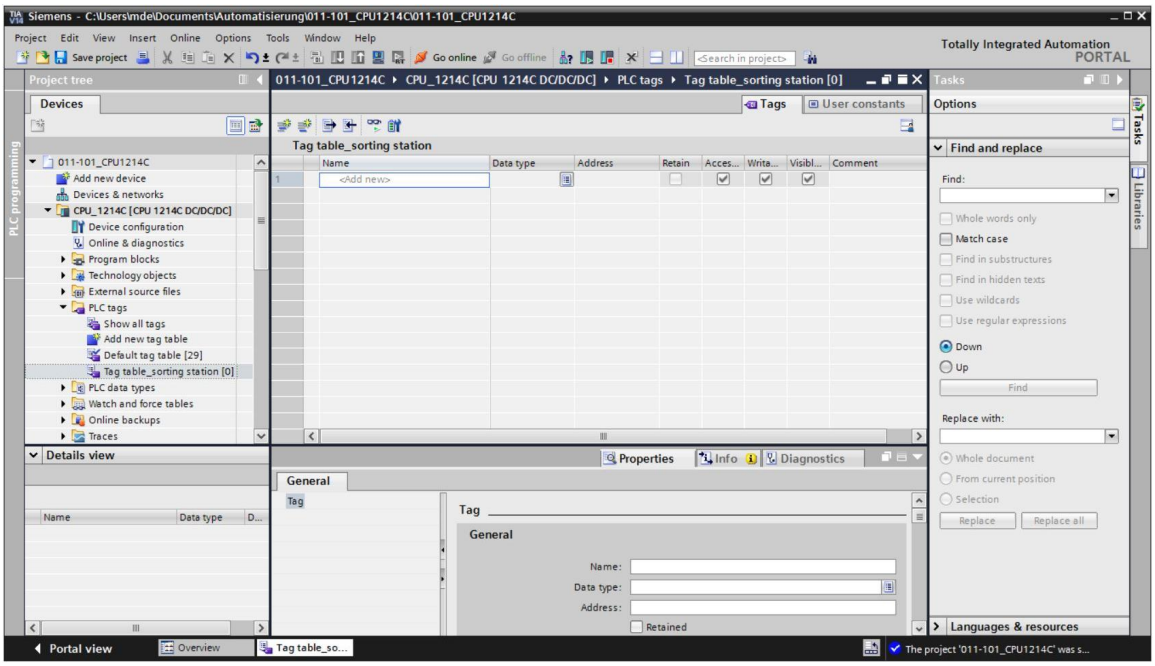
® Dans la vue du projet, naviguez jusqu'aux ® PLC tags (Variables API) de votre automate et créez une nouvelle table de variables en double-cliquant sur ® Add new tag table (Ajouter nouvelle table des variables).



® Renommez la table de variables que vous venez de créer en „Tag_table_sorting_station" (Table_de_variables_installation_de_tri). (® Clic droit sur „Tag_table_1" (Table_des_variables_1) ® „Renommer" ® Tag_table_sorting_-station (Table de variables_installation de tri)

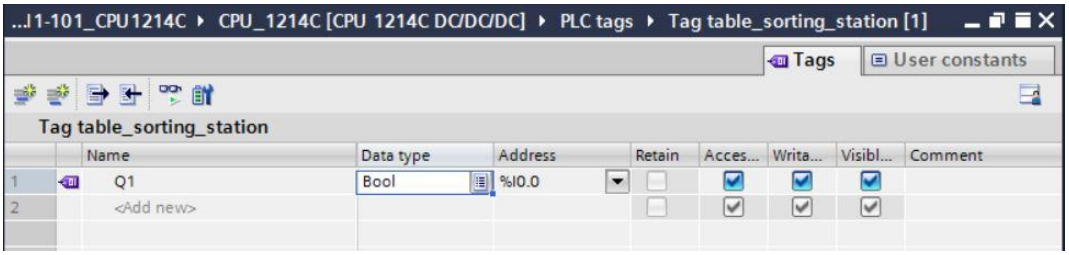


® Ensuite, ouvrez-la en double-cliquant dessus. (® Tag table_sorting station (Table des variables_installation de tri)

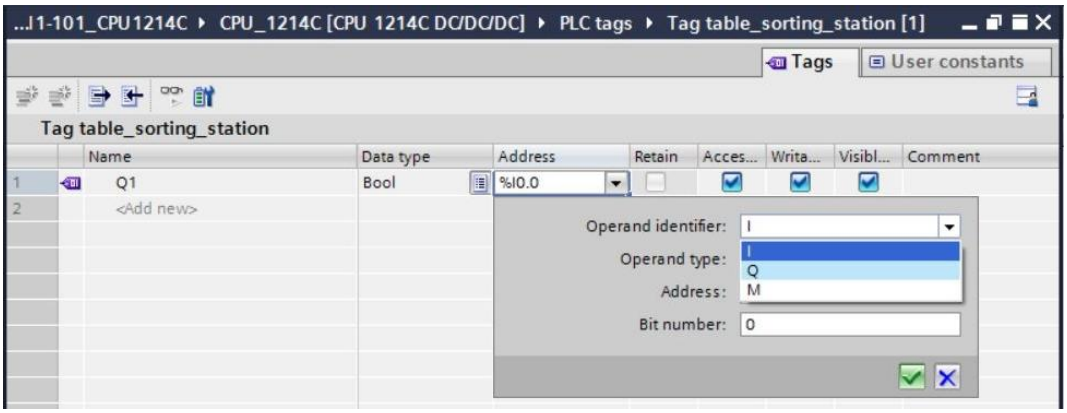


7.3 Création de nouvelles variables dans une table des variables

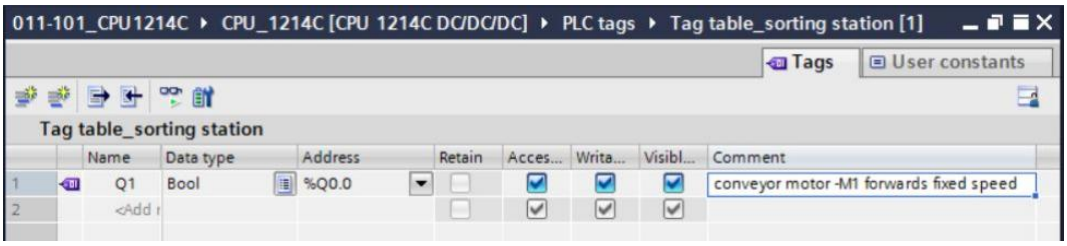
® Ajoutez le nom Q1 et confirmez la saisie avec la touche Entrée. Si vous n'avez pas encore créé de nouvelles variables, TIA Portal attribue automatiquement „Bool" comme type de données et l'adresse %E0.0 (I 0.0). (® <Add new> (Ajouter) ® Q1 ® Enter)



® Modifiez l'adresse de %A0.0 (Q0.0) en saisissant directement cette valeur ou en cliquant sur la flèche de déroulement pour ouvrir le menu d'adressage. Modifiez le type d'opérande à A et confirmez avec Enter ou en cliquant sur la coche. (® %E0.0 ® Operand identifier (Type d'opérande) ® A ®)



® Entrez le commentaire "conveyor motor -M1 forwards fixed speed" (moteur du convoyeur M1 avance à vitesse fixe) pour la variable.



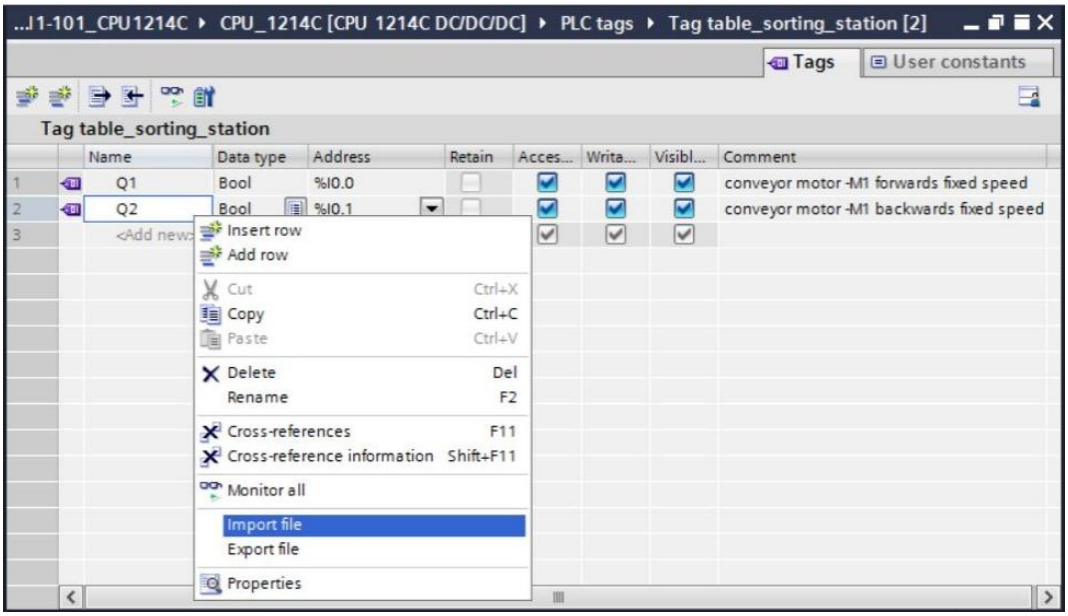
® Ajoutez une nouvelle variable Q2 dans la ligne 2. TIA Portal a automatiquement attribué le même type de données que dans la ligne 1 et augmenté l'adresse de 1 incrément de %A0.1 (Q0.1). Entrez le commentaire “conveyor motor M1 backwards fixed speed“ (moteur du convoyeur M1 en sens inverse à vitesse fixe).

(® <Add new> (Ajouter) ® Q2 ® Enter ® Comment (Commentaire) ® conveyor motor M1 backwards fixed speed (moteur du convoyeur M1 en sens inverse à vitesse fixe))

7.4 Importation de la "Table des variables_installation de tri"

® Pour insérer une table des mnémoniques déjà existante, cliquez avec le bouton droit de la souris sur un champ vide de la "Tag table_sorting station" (table des variables_installation de tri) créée. Sélectionnez "Import file" (Fichier d'importation) dans le menu contextuel.

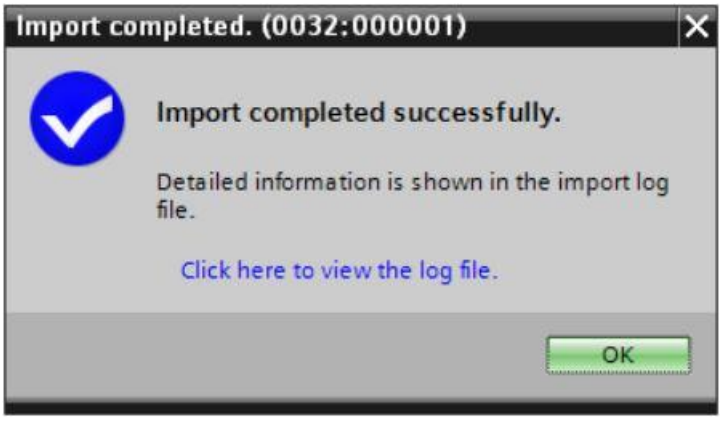
(® Clic droit dans un champ vide de la table de variables ® Import file (Fichier d'importation))



® Choisissez la table des mnémoniques voulue (par ex. au format Xlsx), puis confirmez votre choix à l'aide du bouton "Ouvrir".

(® SCE_FR_020-100_Table de variables Installation de tri… ® Open (Ouvrir))

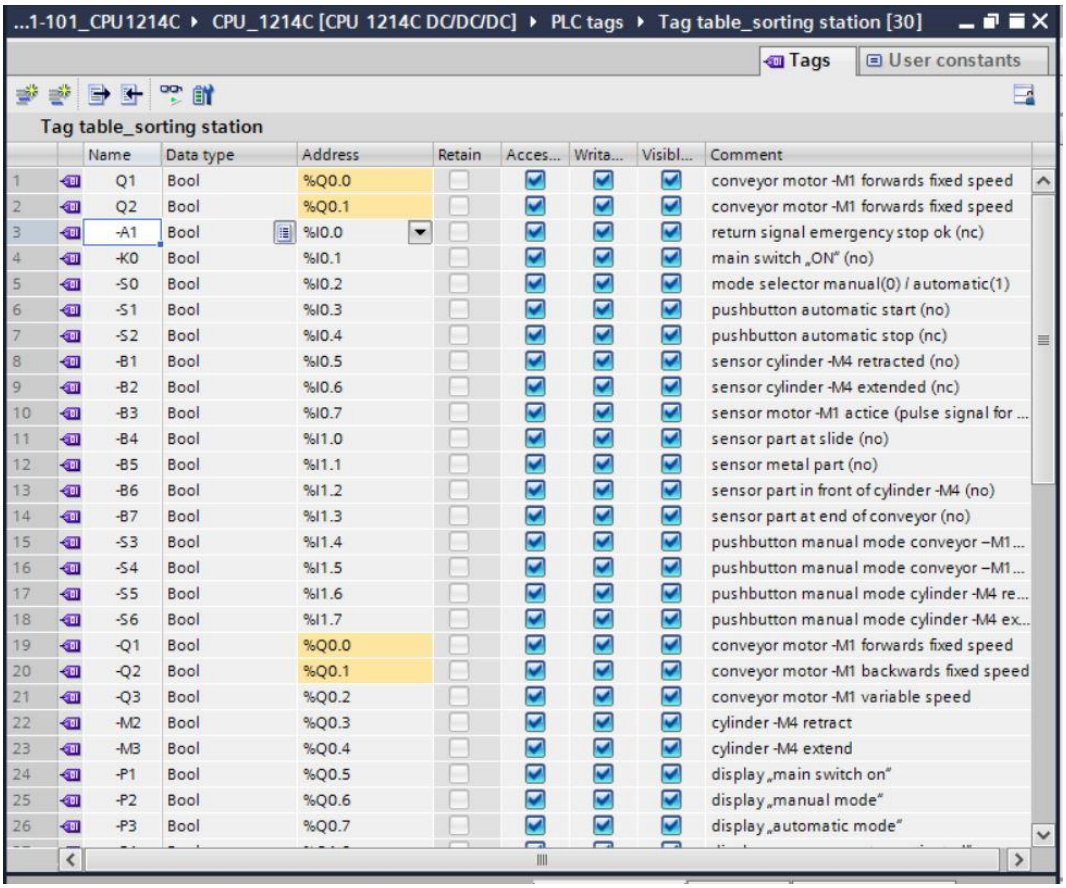
® Une fois l'importation terminée, une fenêtre de confirmation s'ouvre pour vous donner la possibilité de consulter le fichier journal de l'importation. Cliquez sur ® OK.



Vous constaterez que certaines adresses sont affichées en orange. Celles-ci existent en double et les noms des variables associées ont été numérotés automatiquement afin d'en garantir l'univocité.

® Pour supprimer les doublons, sélectionnez les lignes et appuyez sur la touche Suppr de votre clavier ou sélectionnez "Delete" (Supprimer) dans le menu contextuel.

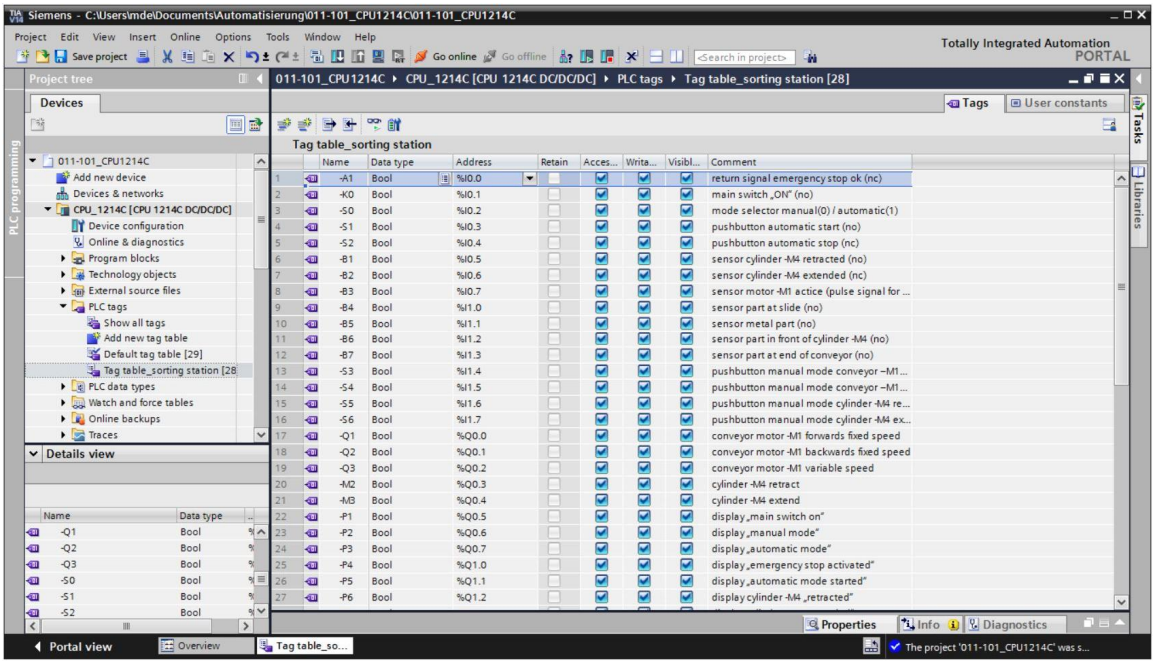
(® Clic droit sur variables sélectionnées ® Delete (Supprimer))



	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	Q1	Bool	%Q0.0		☑	☑	☑	conveyor motor -M1 forwards fixed speed
2	Q2	Bool	%Q0.1		☑	☑	☑	conveyor motor -M1 forwards fixed speed
3	-A1	Bool	%I0.0		☑	☑	☑	return signal emergency stop ok (nc)
4	-K0	Bool	%I0.1		☑	☑	☑	main switch „ON" (no)
5	-S0	Bool	%I0.2		☑	☑	☑	mode selector manual(0) / automatic(1)
6	-S1	Bool	%I0.3		☑	☑	☑	pushbutton automatic start (no)
7	-S2	Bool	%I0.4		☑	☑	☑	pushbutton automatic stop (nc)
8	-B1	Bool	%I0.5		☑	☑	☑	sensor cylinder -M4 retracted (no)
9	-B2	Bool	%I0.6		☑	☑	☑	sensor cylinder -M4 extended (nc)
10	-B3	Bool	%I0.7		☑	☑	☑	sensor motor -M1 actice (pulse signal for ...
11	-B4	Bool	%I1.0		☑	☑	☑	sensor part at slide (no)
12	-B5	Bool	%I1.1		☑	☑	☑	sensor metal part (no)
13	-B6	Bool	%I1.2		☑	☑	☑	sensor part in front of cylinder -M4 (no)
14	-B7	Bool	%I1.3		☑	☑	☑	sensor part at end of conveyor (no)
15	-S3	Bool	%I1.4		☑	☑	☑	pushbutton manual mode conveyor –M1...
16	-S4	Bool	%I1.5		☑	☑	☑	pushbutton manual mode conveyor –M1...
17	-S5	Bool	%I1.6		☑	☑	☑	pushbutton manual mode cylinder -M4 re...
18	-S6	Bool	%I1.7		☑	☑	☑	pushbutton manual mode cylinder -M4 ex...
19	-Q1	Bool	%Q0.0		☑	☑	☑	conveyor motor -M1 forwards fixed speed
20	-Q2	Bool	%Q0.1		☑	☑	☑	conveyor motor -M1 backwards fixed speed
21	-Q3	Bool	%Q0.2		☑	☑	☑	conveyor motor -M1 variable speed
22	-M2	Bool	%Q0.3		☑	☑	☑	cylinder -M4 retract
23	-M3	Bool	%Q0.4		☑	☑	☑	cylinder -M4 extend
24	-P1	Bool	%Q0.5		☑	☑	☑	display „main switch on"
25	-P2	Bool	%Q0.6		☑	☑	☑	display „manual mode"
26	-P3	Bool	%Q0.7		☑	☑	☑	display „automatic mode"

® Maintenant, vous avez devant vous une table des mnémoniques complète des entrées et sorties numériques. Enregistrez votre projet sous 031-200_Programmation de FB.

(® Project (Projet) ® Save as (Enregistrer sous)® 031-200_Programmation de FB ® Save (Enregistrer))

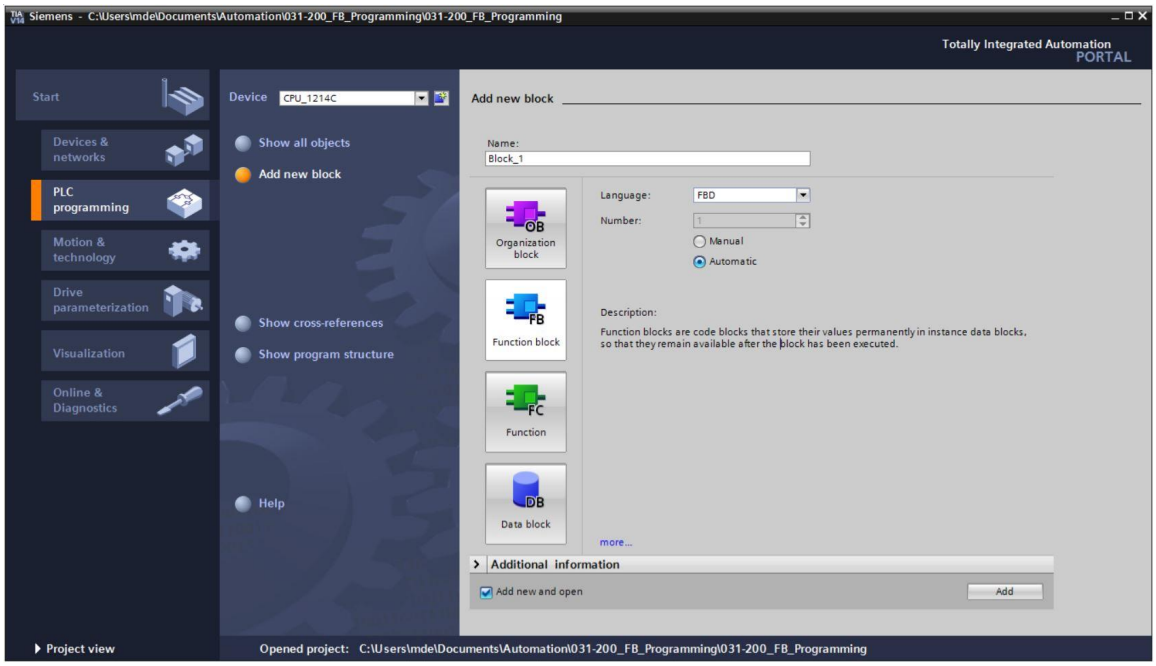


7.5 Création du bloc fonctionnel FB1 „MOTOR_AUTO“ pour le moteur du convoyeur en mode automatique

® Pour créer un nouveau bloc fonctionnel, cliquez dans la vue du portail dans la section PLC programming (Programmation API) sur „Add new block“ (Ajouter nouveau bloc).

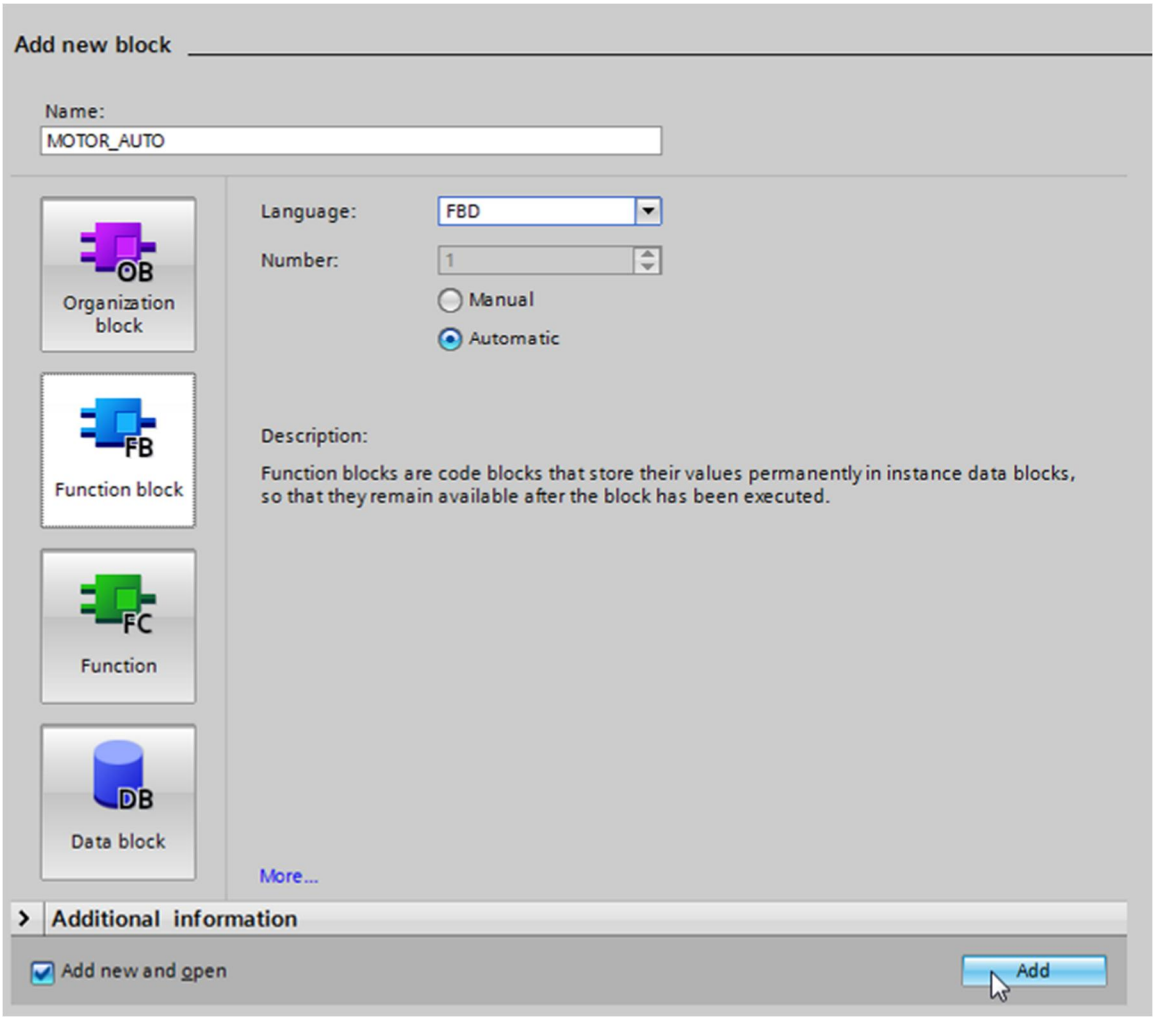
(® PLC programming (Programmation API) ® Add new block (Ajouter nouveau bloc) ® 

)



® Renommez votre nouveau bloc en : „MOTOR_AUTO“, vérifiez que LOG est choisi comme langage et activez la numérotation automatique. Cochez „Add new and open“ (Ajouter nouveau et ouvrir) pour que le bloc fonctionnel que vous avez créé s'ouvre automatiquement dans la vue du projet.Cliquez sur „Add“ (Ajouter).

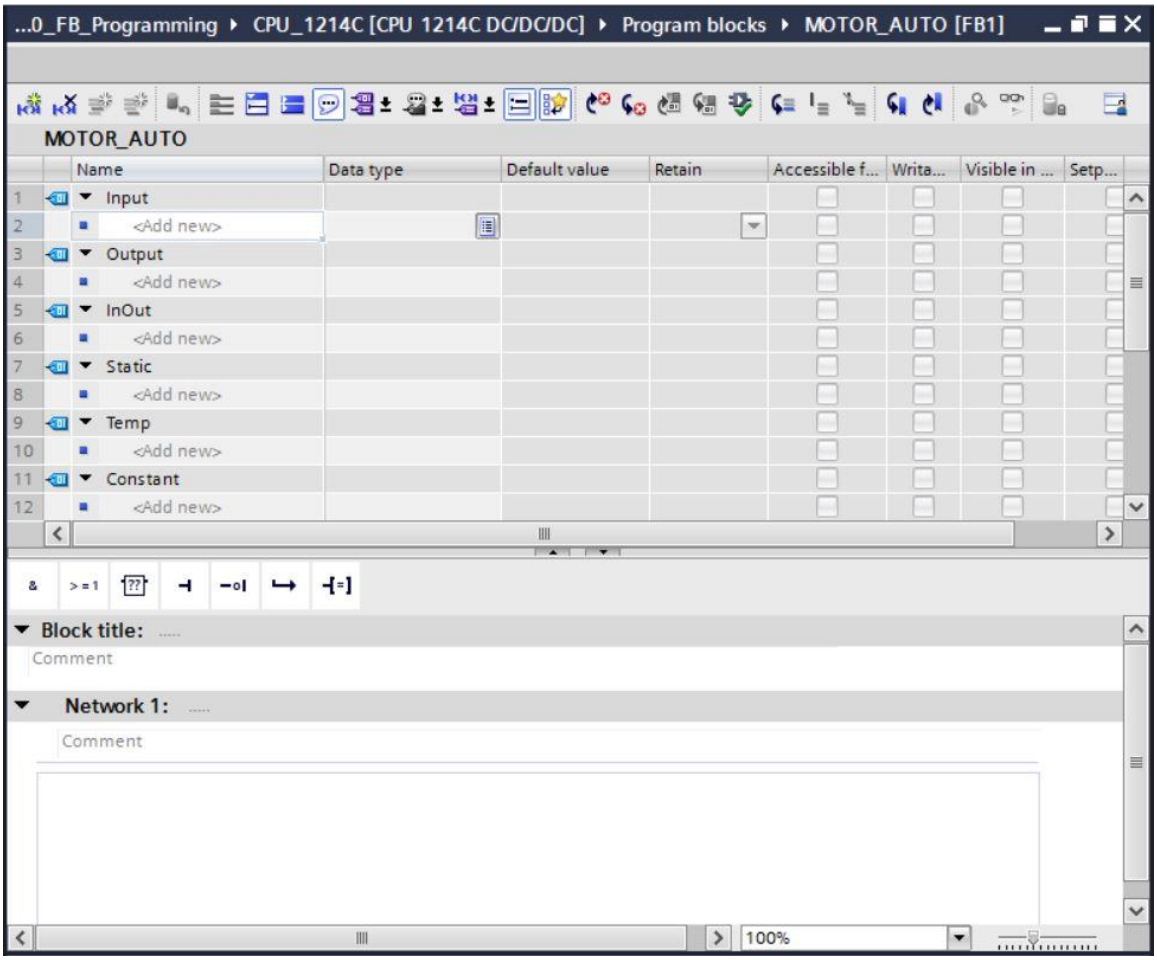
(® Name (Nom) : MOTOR_AUTO® Language (Langage) : LOG ® Number (Numéro) : automatique ® ☑ Add new and open (Ajouter nouveau et ouvrir) ® Add (Ajouter))



7.6 Définition de l'interface du FB1 „MOTOR_AUTO“

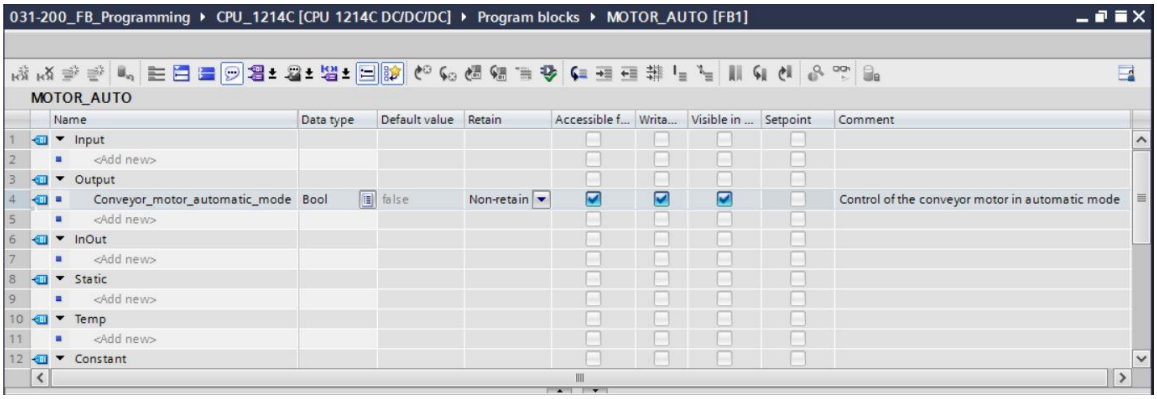
® Lorsque vous avez cliquez sur „Add new and open“ (Ajouter nouveau et ouvrir), la vue du projet s'ouvre avec une fenêtre pour vous permettre de créer le bloc que vous venez de générer.

® La déclaration de l'interface de votre bloc fonctionnel se trouve dans la partie supérieure de votre vue de programmation.



® La commande du moteur du convoyeur requiert un signal de sortie binaire. C'est pourquoi nous créons d'abord la variable de sortie #Moteur convoyeur_automatique de type „Bool". Nous faisons accompagner ce paramètre du commentaire „Control of the conveyor motor in automatic mode" (Commande du moteur du convoyeur en mode automatique).

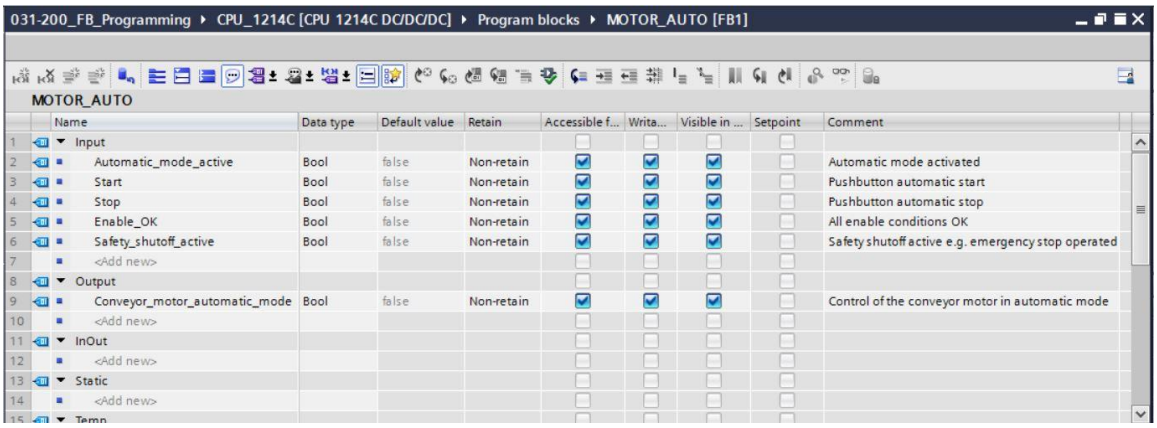
(® Output (Sortie)® Conveyor_motor_automatic_mode (Convoyeur_moteur_automatique) ® Bool ® Control of the conveyor motor in automatic mode (Commande du moteur du convoyeur en mode automatique))



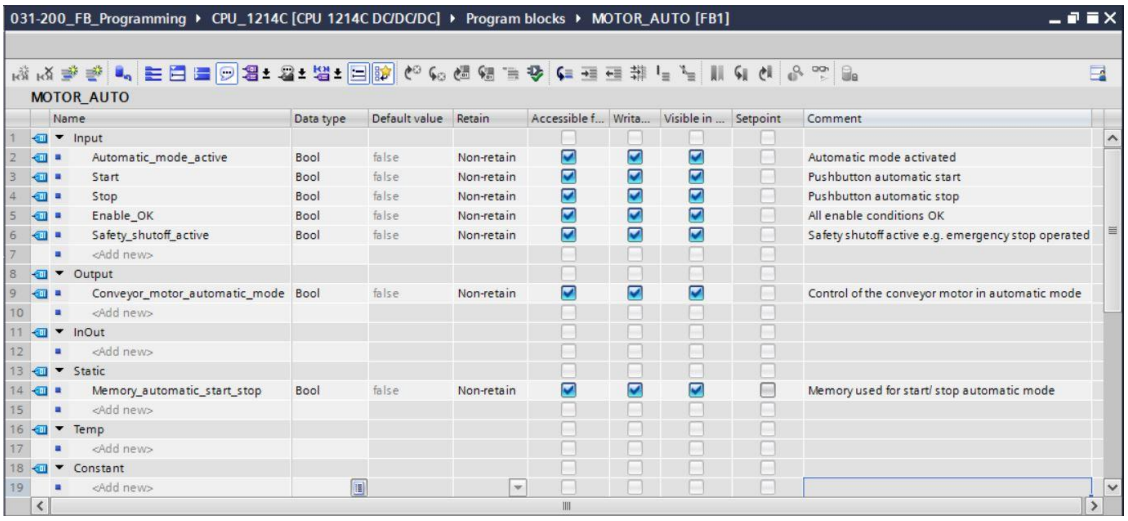
® Sous Input ajoutez d'abord le paramètre #Mode automatique_activé comme interface d'entrée et confirmez la saisie avec la touche Entrée ou quittez le champ de saisie. Le type de données „Bool" est attribué automatiquement. Il est conservé. Saisissez ensuite le commentaire „Automatic mode activated" (mode de fonctionnement automatique activé).

(® Input ® Automatic_mode_active (Mode de fonctionnement automatique activé)® Bool ® Automatic mode activated (mode de fonctionnement automatique activé)

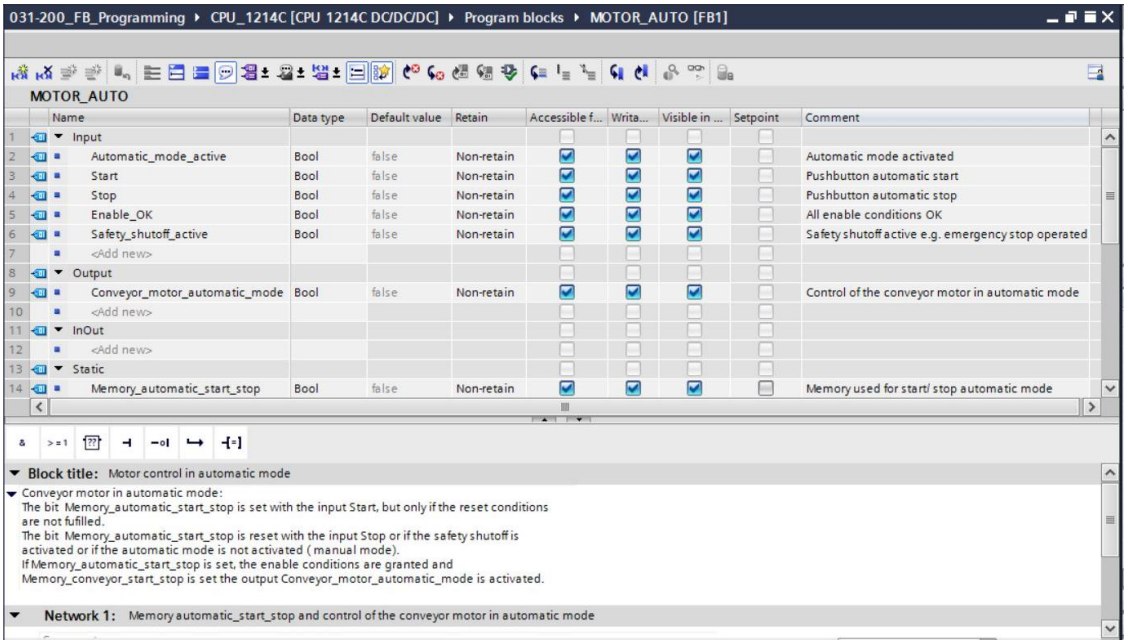
® Sous Input ajoutez ensuite comme autres paramètres d'entrée binaires #Start (Démarrage), #Stop (Arrêt), #Enable_OK (Validation) et #Safety_shutoff_active (arrêt automatique de sécurité_activé) et vérifiez leurs types de données. Complétez avec des commentaires utiles.



® Le convoyeur est démarré et arrêté à l'aide de boutons-poussoirs. Nous avons donc besoin d'une variable „Static" comme mémoire. Sous Static ajoutez la variable #Memory_automatic_start_stop (mémoire_automatique_démarrage_arrêt) et confirmez la saisie avec la touche Entrée ou quittez le champ de saisie. Le type de données „Bool" est attribué automatiquement. Il est conservé. Saisissez ensuite le commentaire „Memory used for start/stop automatic mode" (mémoire utilisée pour démarrage/arrêt du mode automatique). (® Static ® Memory_automatic_start_stop (mémoire_automatique_ démarrage_arrêt) ® Bool ® Memory used for start/stop automatic mode (mémoire utilisée pour démarrage/arrêt du mode automatique))

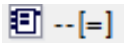


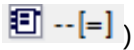
® Pour la documentation du programme, saisissez le titre du bloc, un commentaire de bloc et pour le réseau 1, un titre du réseau évocateur. (® Block title (Titre du bloc) : Motor control in automatic mode (commande du moteur en mode automatique) ® Network 1 (Réseau 1) : Memory automatic_start_stop and control of the conveyor motor in automatic mode (mémoire utilisée pour démarrage/arrêt du mode automatique et commande du moteur du convoyeur en mode automatique))

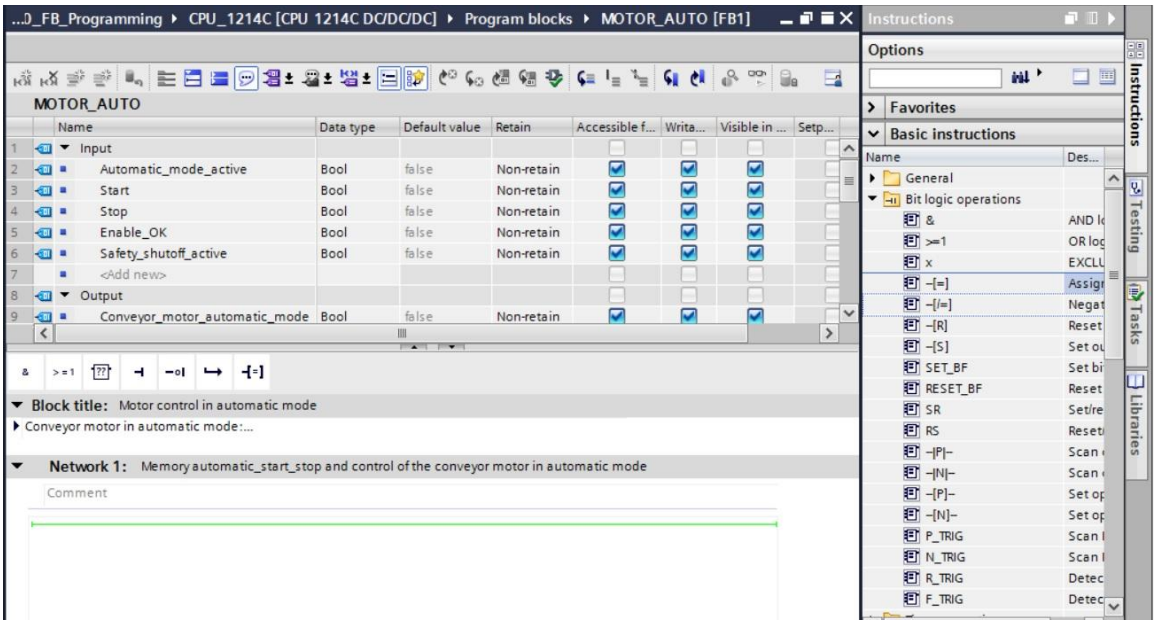


7.7 Programmation du FB1 : MOTOR_AUTO

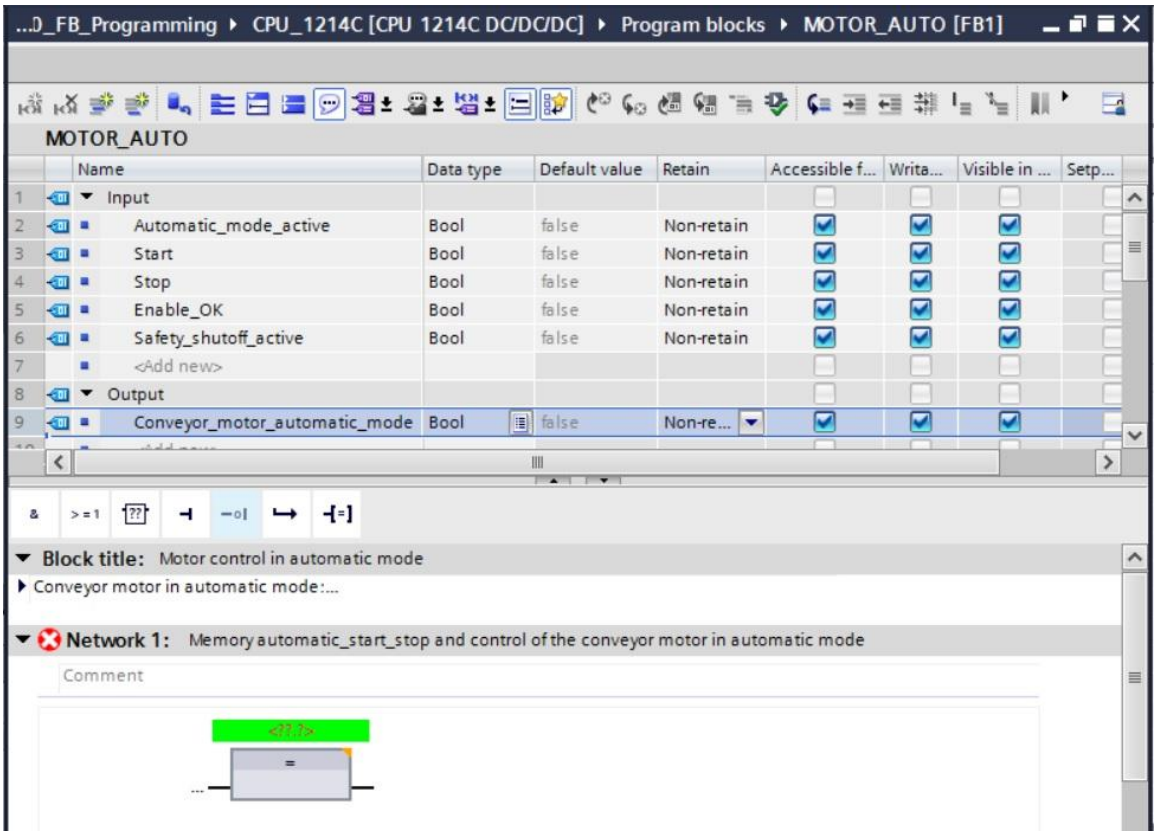
® Sous la déclaration de l'interface, vous verrez dans la fenêtre de programmation une barre d'outils contenant différentes fonctions logiques et, en dessous, une zone avec des réseaux. Nous y avons déjà défini le titre du bloc et un titre pour le premier réseau. Dans les réseaux, la programmation s'effectue en utilisant différents blocs logiques. Pour des raisons de lisibilité, le programme est subdivisé en plusieurs réseaux. Vous allez maintenant vous familiariser avec les différentes méthodes qu'il existe pour insérer des blocs logiques.

® Dans votre fenêtre de programmation, à droite, se trouve une liste des instructions que vous pouvez utiliser dans le programme. Recherchez sous ® Instructions de base ® Fonctions logiques, la fonction  (Affectation) et placez-la dans votre Réseau 1 grâce à un glisser-déposer (la ligne verte apparaît, pointeur de la souris avec le +).

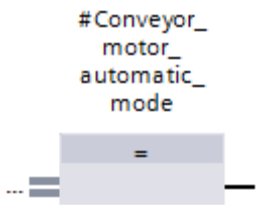
(® Instructions (Instructions) ® Basic instructions (Instructions de base) ® Bit logic operations (Fonctions logiques) ® )



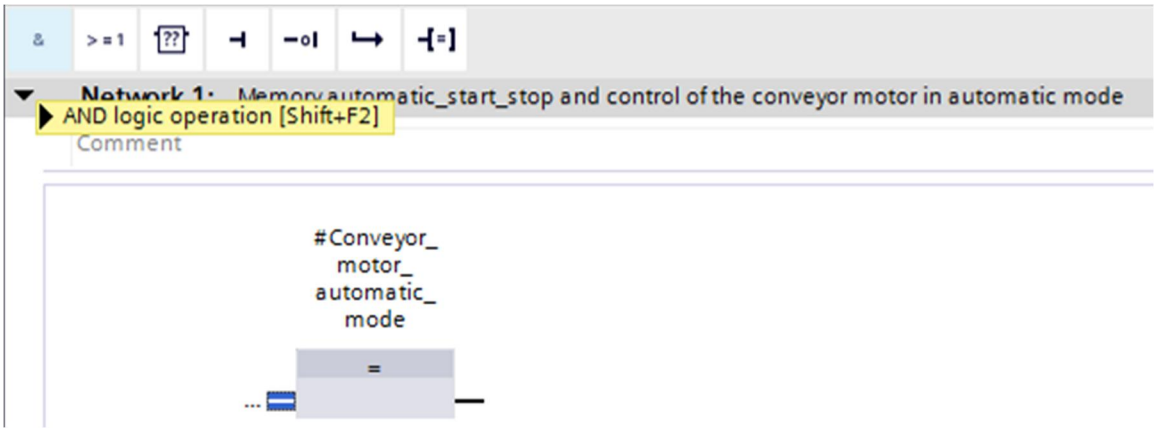
® Maintenant, faites glisser le paramètre Output #Convoyeur_moteur_automatique et déposez-le sur <??.?> au-dessus du bloc que vous venez de créer. La meilleure méthode pour sélectionner un paramètre dans la description de l'interface consiste à le prendre par l'icône bleue . (® Conveyor_motor_automatic_mode)



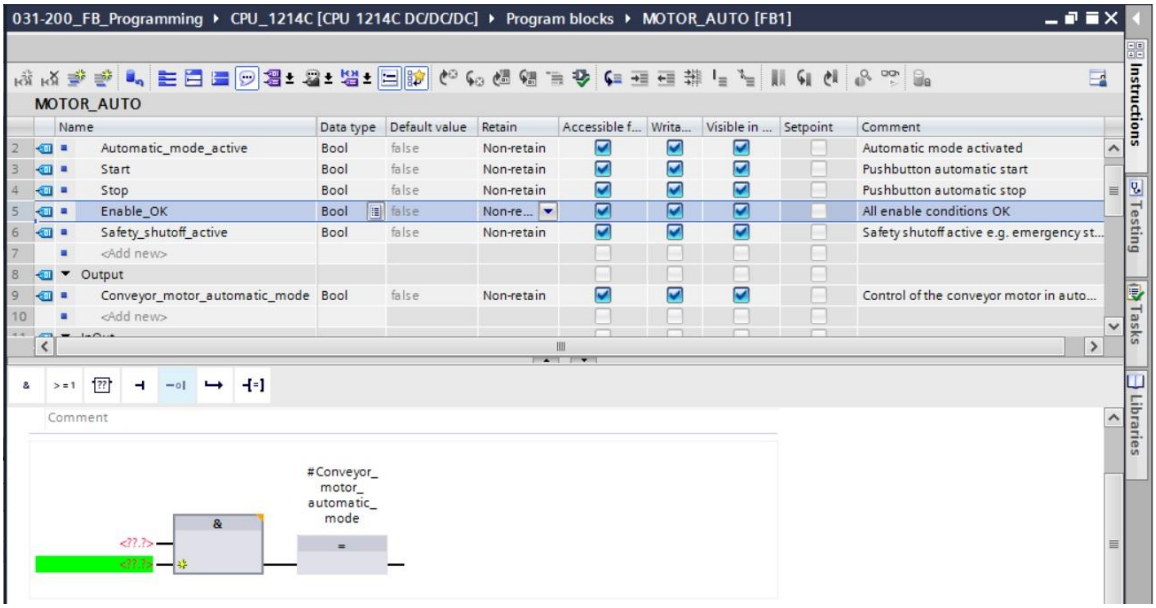
® Cela permet de définir que c'est ce bloc qui écrit le paramètre #Convoyeur_moteur_automatique. Toutefois, il manque encore les conditions d'entrée nécessaires pour le réaliser effectivement. Sur l'entrée du bloc d'affectation, nous voulons combiner une bascule SR et le paramètre #Enable OK (Validation_OK) par une opération ET. Pour cela, cliquez d'abord sur l'entrée du bloc pour que la ligne d'entrée soit représentée sur fond bleu.



® Cliquez sur l'icône dans la barre d'outils pour insérer une opération ET devant votre bloc d'affectation.

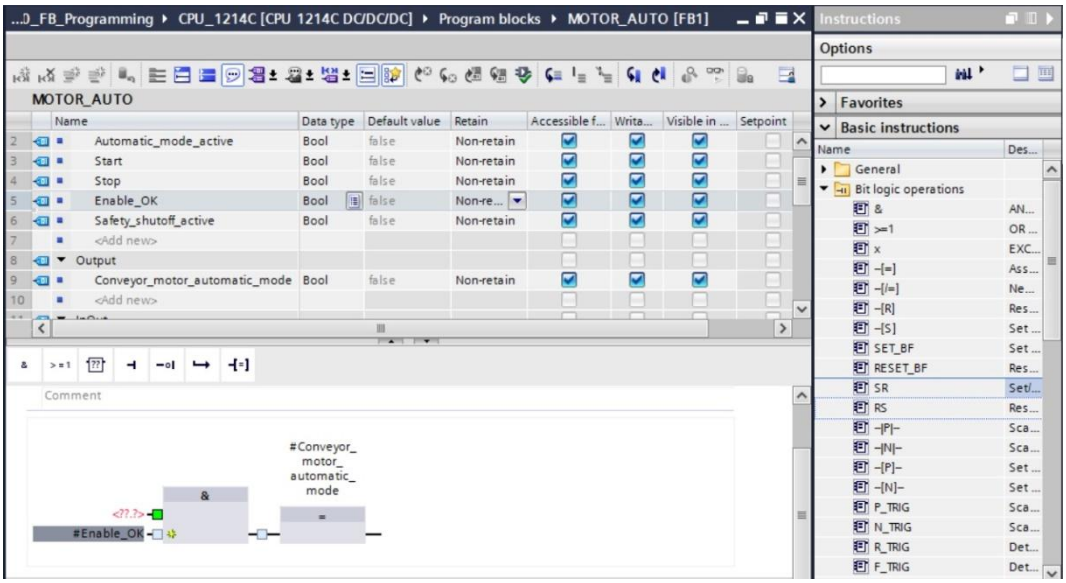


® Faites glisser le paramètre Input #Enable OK (Validation_OK) sur la deuxième entrée de l'opération logique & <??.?>. (® Enable OK (Validation_OK))

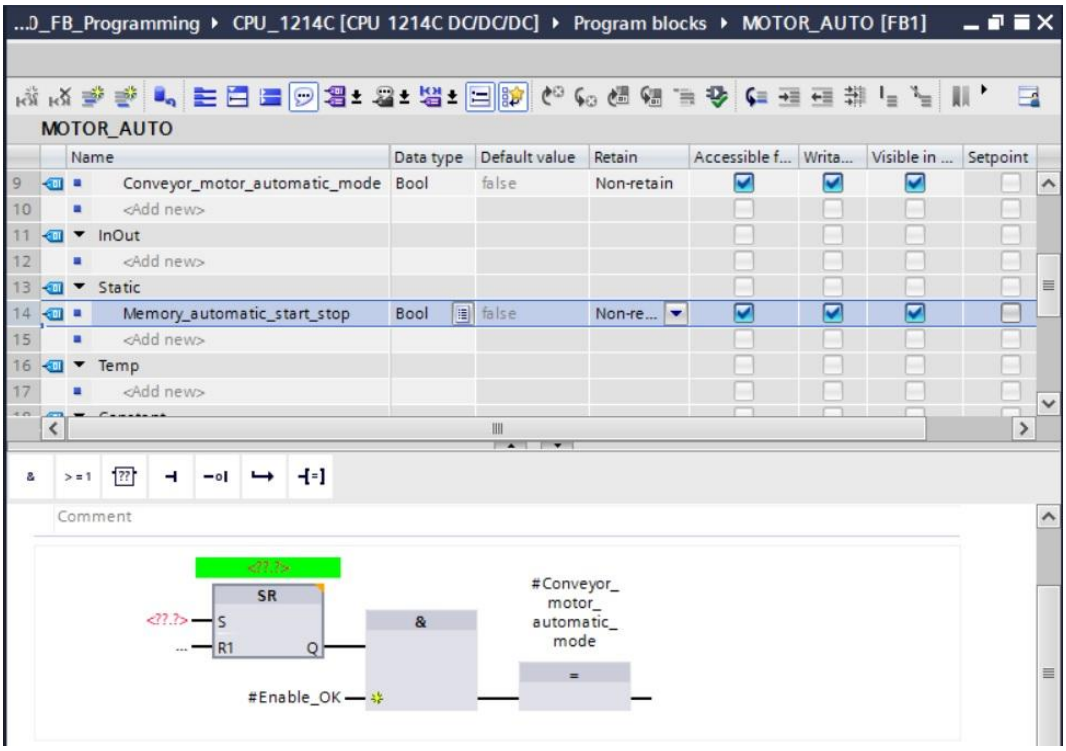


® Faites glisser de la liste des instructions sous ® Instructions de base ® Opérations logiques sur bits la fonction bascule bistable SR sur la première entrée de l'opération logique &.

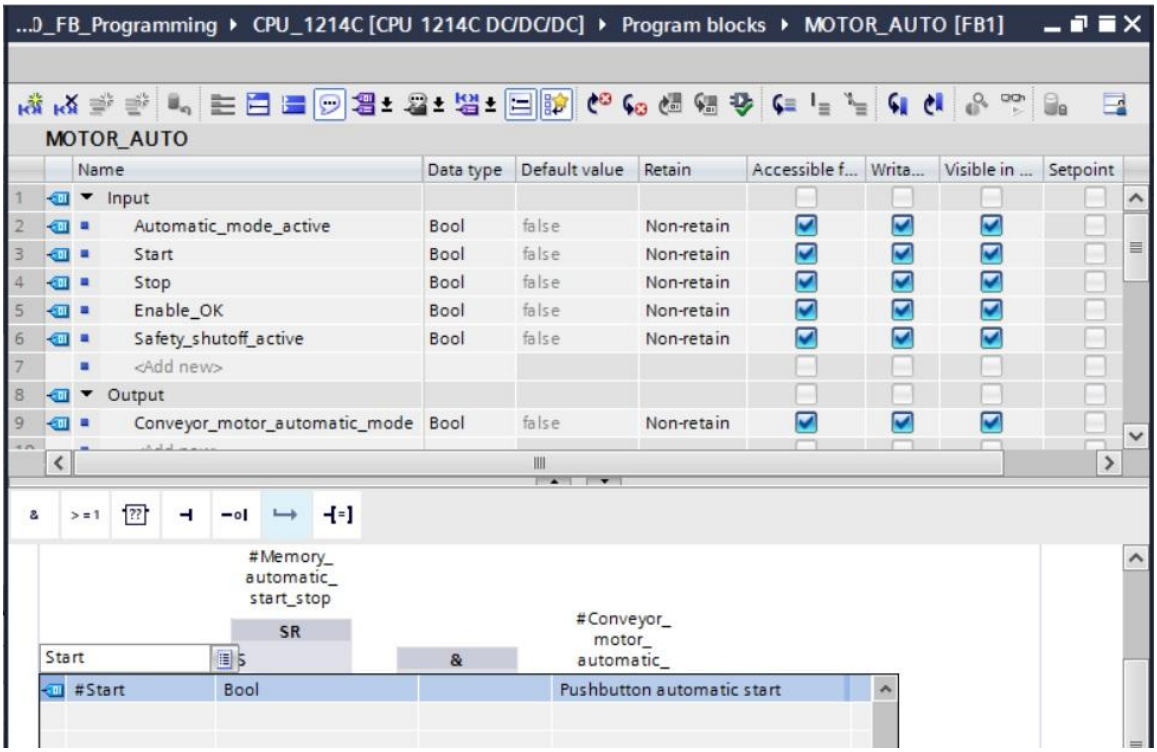
(® Instructions (Instructions) ® Basic instructions (Instructions de base) ® Bit logic operations (Fonctions logiques) ® SR ®)



® La bascule SR requiert une variable de mémoire. Pour cela, faites glisser le paramètre Static #Mémoire_automatique_démarrage_arrêt sur <??.?> au-dessus de la bascule SR. (® Memory_automatic_start_stopp (Mémoire_automatique_démarrage_arrêt))

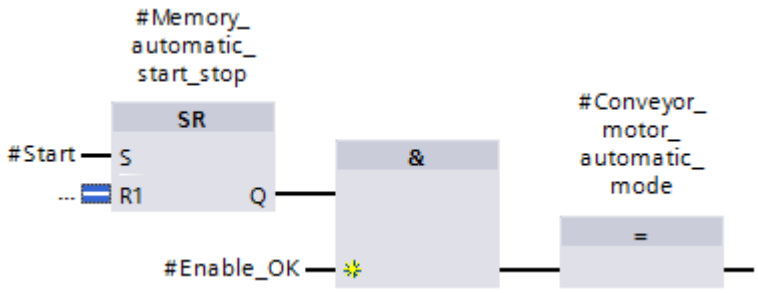


® Nous voulons mettre à 1 la #Mémoire_automatique_démarrage_arrêt avec la variable d'entrée #Démarrage. Pour cela, double-cliquez sur l'entrée S de la bascule SR <??.?>, un champ s'ouvre, saisissez „Start" dans ce champ pour voir la liste des variables qui commencent par „Start". Cliquez sur la variable #Start et validez avec ® Enter (touche Entrée). (® bascule SR ® <??.?> ® Start ® #Start ® Enter (Entgrée))

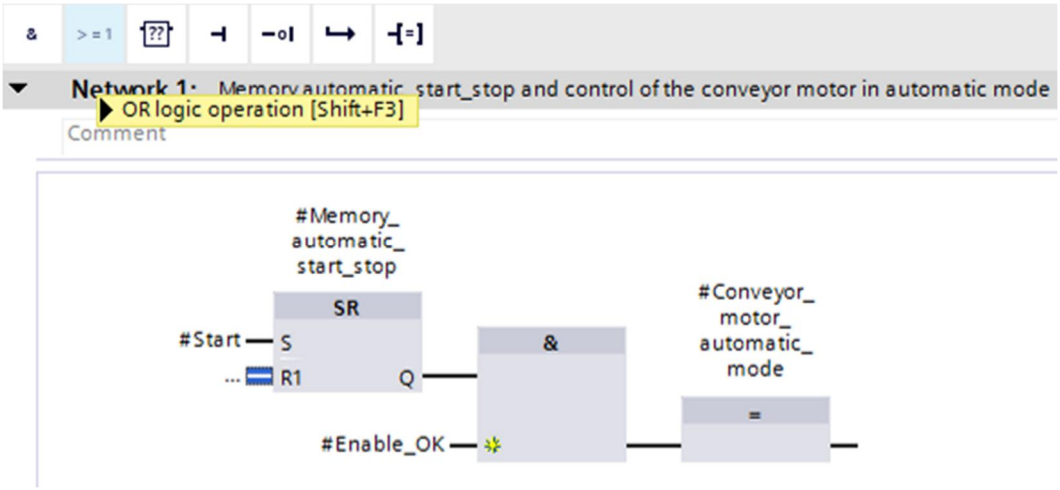


Remarque : *cette méthode d'affectation des variables comporte un risque de confusion avec les variables globales de la table des variables. C'est pourquoi il convient de privilégier la méthode d'affectation par glisser-déposer de la déclaration de l'interface décrite ci-dessus.*

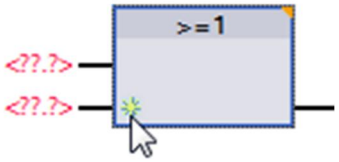
® Nous voulons arrêter le convoyeur sous plusieurs conditions. Il nous faut donc un bloc OU sur l'entrée R1 de la bascule SR. Cliquez d'abord sur l'entrée R1 de la bascule SR pour que la ligne d'entrée soit représentée sur fond bleu.



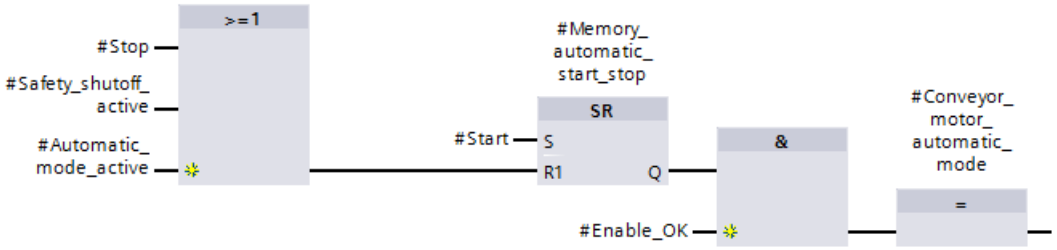
® Cliquez ensuite sur l'icône >=1 dans la barre d'outils pour insérer une opération OU.



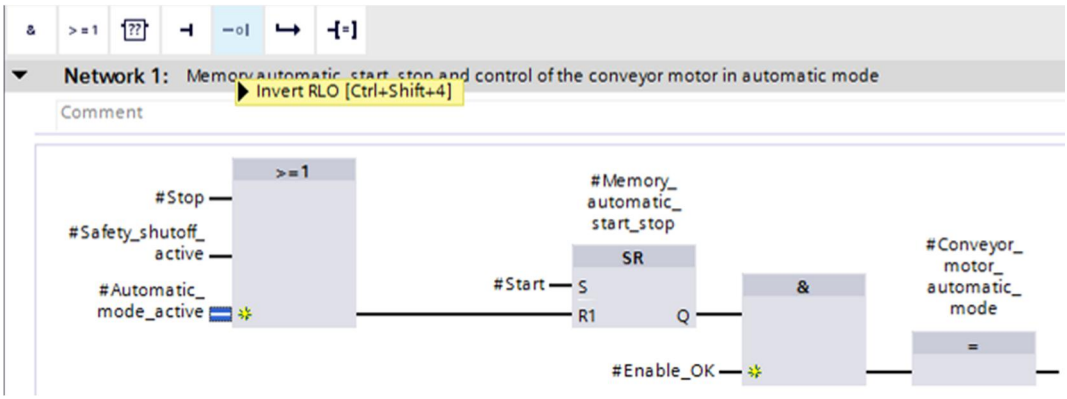
® Le bloc OU n'a que 2 entrées pour l'instant. Pour relier une variable d'entrée supplémentaire, cliquez sur l'étoile jaune de votre élément logique OU.

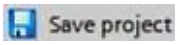


® Ajoutez sur les 3 entrées de l'élément logique OU les variables d'entrée #Stop (Arrêt), #Safety_shutoff_active (Arrêt automatique de sécurité_activé) et #Automatic_mode_active (Automatique_mode_activé).

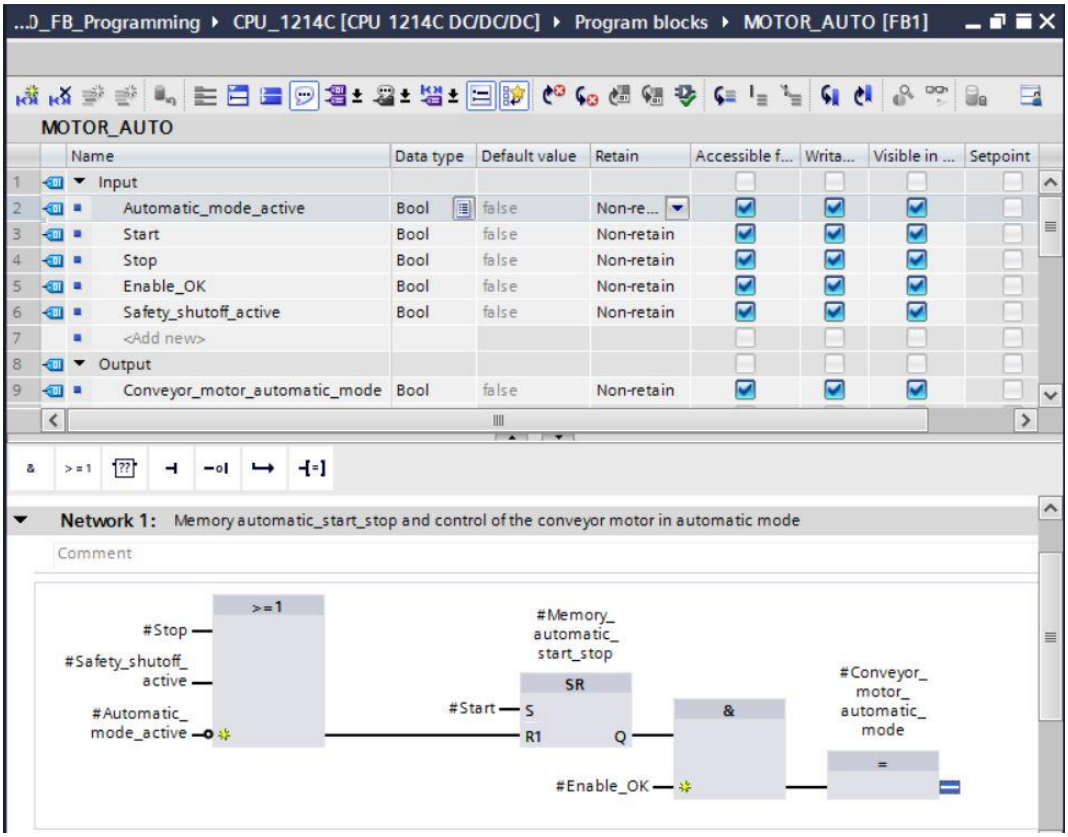


® Pour programmer une négation de l'entrée connectée au paramètre #Automatic_mode_active (Automatique_mode_activé), sélectionnez-la et cliquez sur  .

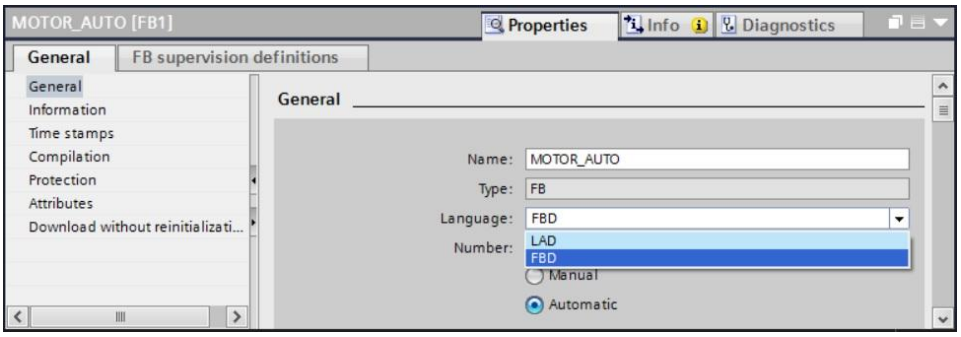


® N'oubliez pas de cliquer sur 

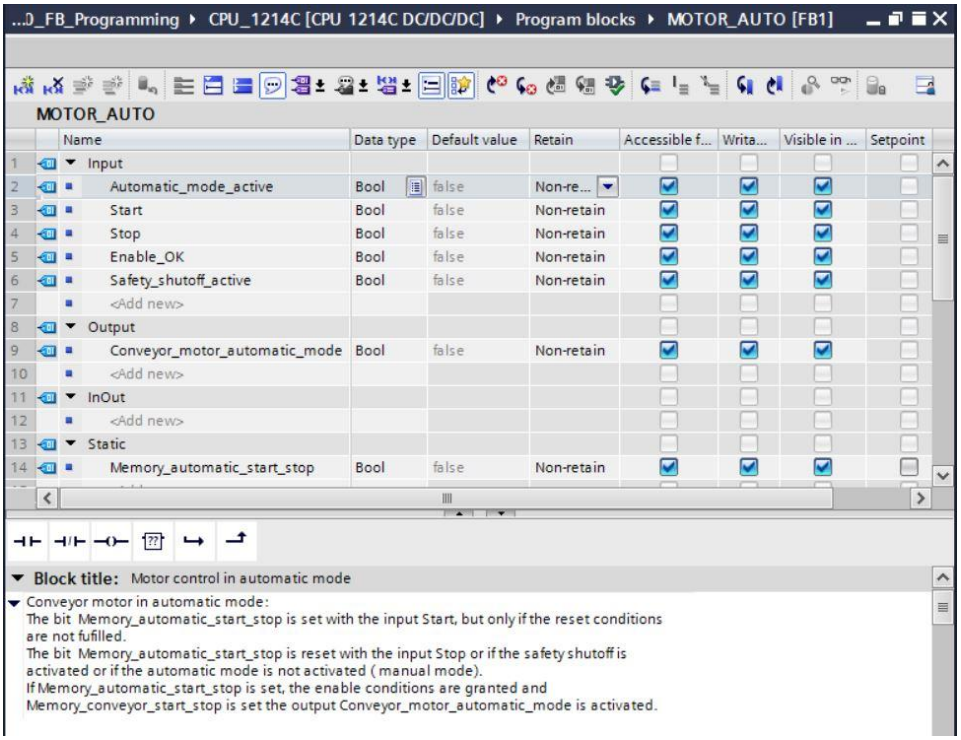
à la fin de chaque étape. Ci-après, le bloc de fonction „MOTOR_AUTO [FB1] terminé représenté dans le logigramme (LOG).

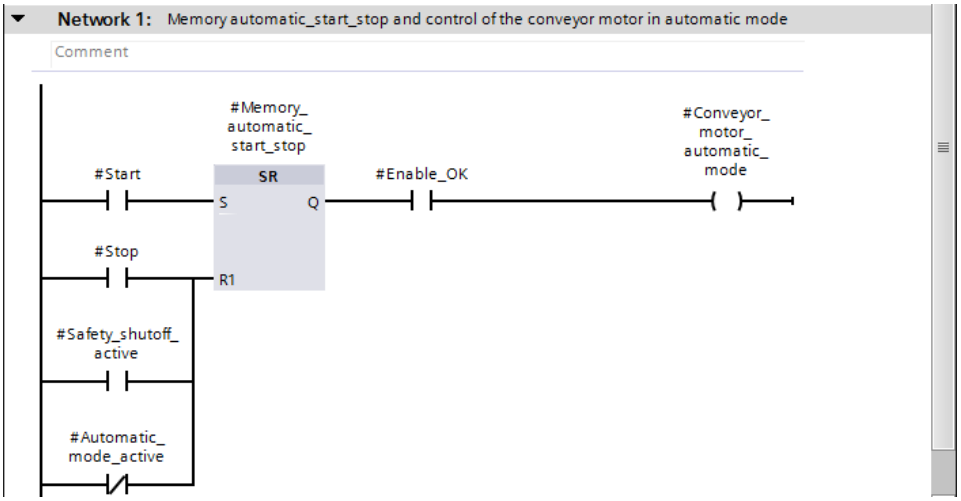


® Vous pouvez régler le „Langage“ sur CONT (schéma à contacts) dans l'onglet „General“ (Général) des propriétés du bloc. (® Properties (Propriétés) ® General (Général) ® Language (Langage) : CONT (LAD)



® Ci-dessous, le programme représenté dans CONT.

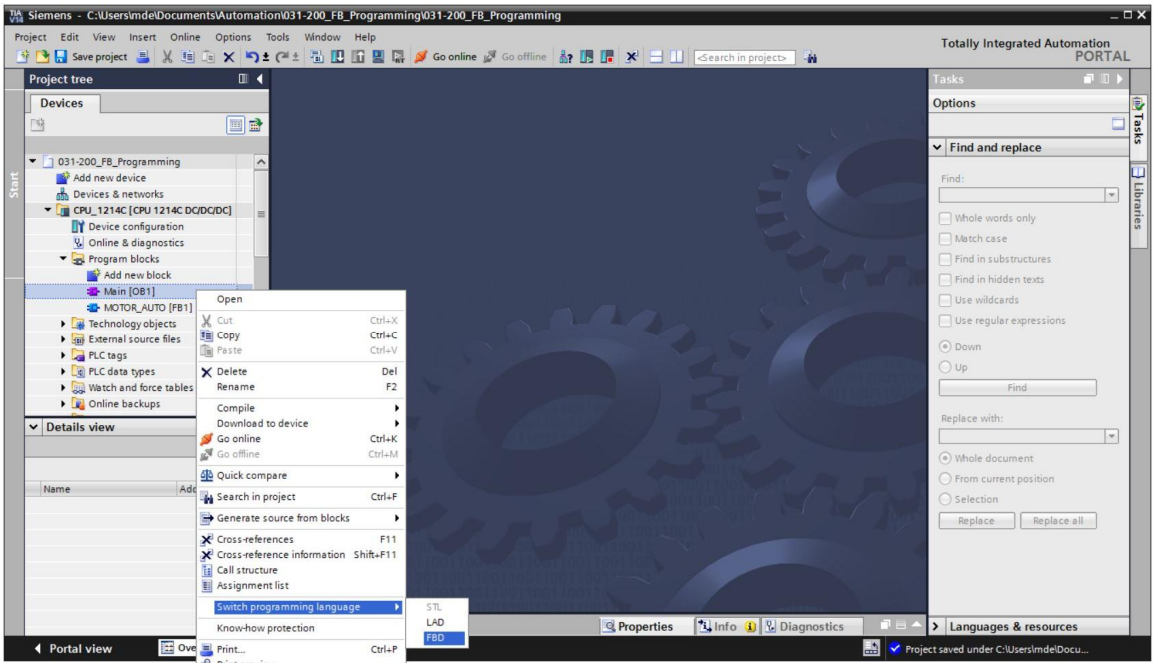




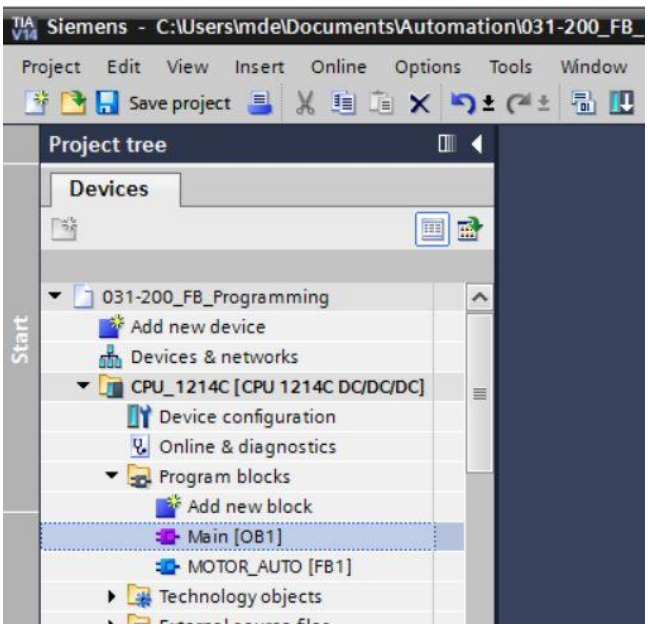
7.8 Programmation du bloc d'organisation OB1 – commande de l'avance du convoyeur en mode automatique

® Avant la programmation des bloc d'organisation „Main [OB1]“, nous commutons le langage de programmation sur LOG (logigramme). Pour cela, cliquez d'abord avec le bouton gauche de la souris sur „Main [OB1)“ dans le dossier „Blocs de programme“.

(® CPU_1214C [CPU 1214C DC/DC/DC] ® Program blocks (Blocs de programme) ® Main [OB1] ® Switch programming language (Commuter le langage de programmation) ® FBD (LOG))

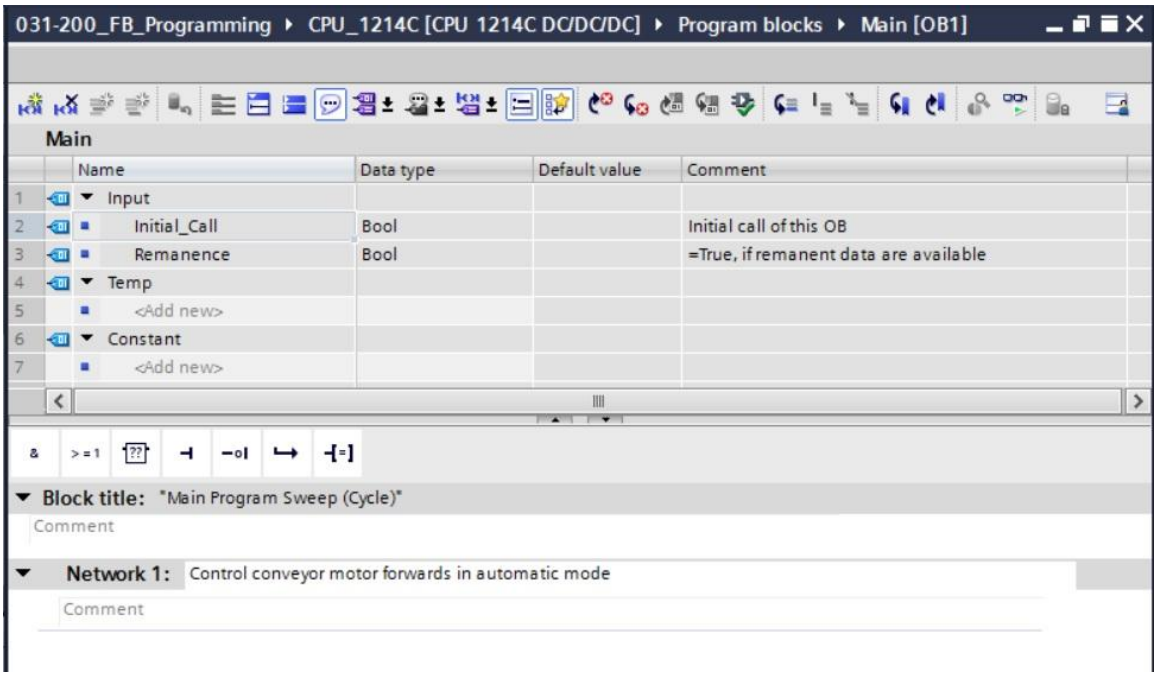


® Double-cliquez maintenant sur le bloc d'organisation "Main [OB1]" pour l'ouvrir.

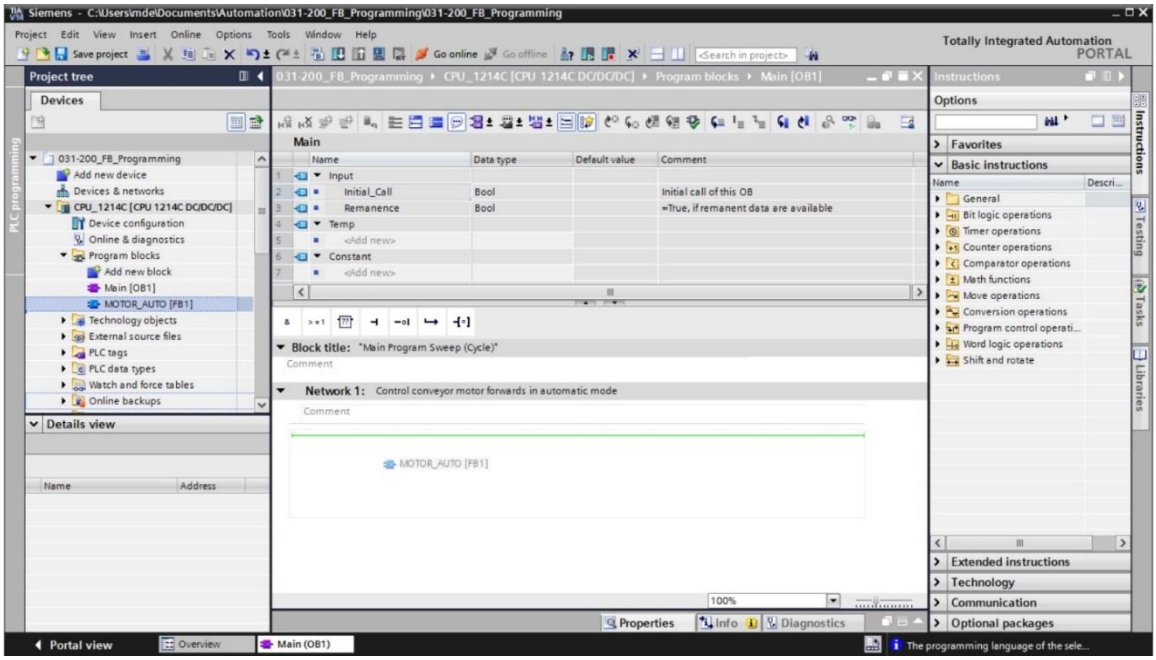


® Attribuez le nom "Commande de l'avance du convoyeur en mode automatique" au réseau 1.

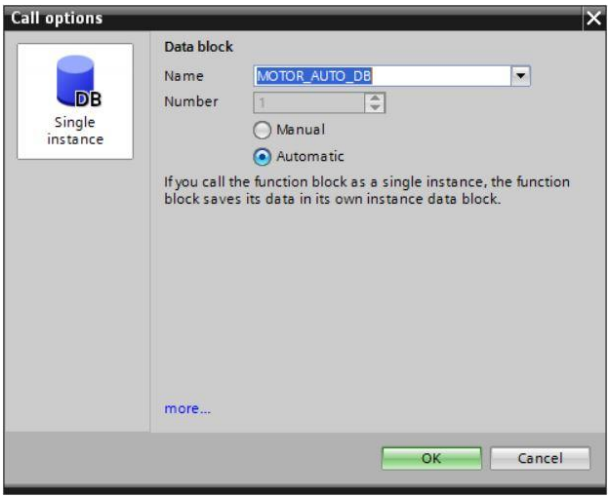
(® Network 1 (Réseau 1):… ® Control conveyor motor forwards in automatic mode (Commande de l'avance du convoyeur en mode automatique))



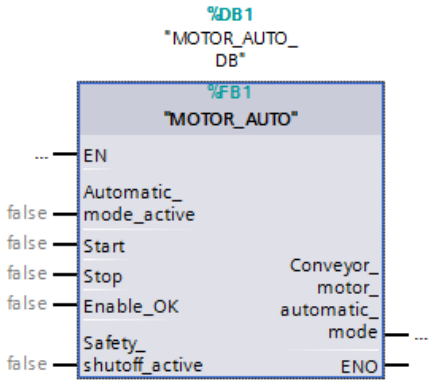
® Placez maintenant votre bloc fonctionnel „MOTOR_AUTO [FB1]“ dans le réseau 1 sur la ligne verte grâce à un glisser-déposer.



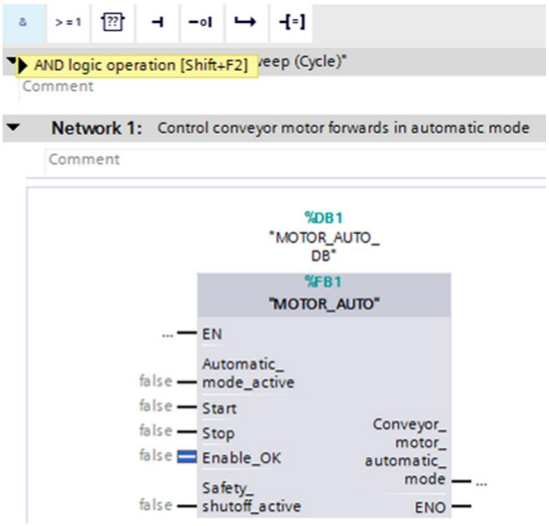
® Le bloc de données d'instance correspondant à cet appel du FB1 est généré automatiquement. Attribuez un nom et validez-le avec OK. (® MOTOR_AUTO_DB1 ® OK)



® Un bloc avec l'interface que vous avez définie, le bloc de données d'instance et les connexions EN et ENO est inséré dans le réseau 1.

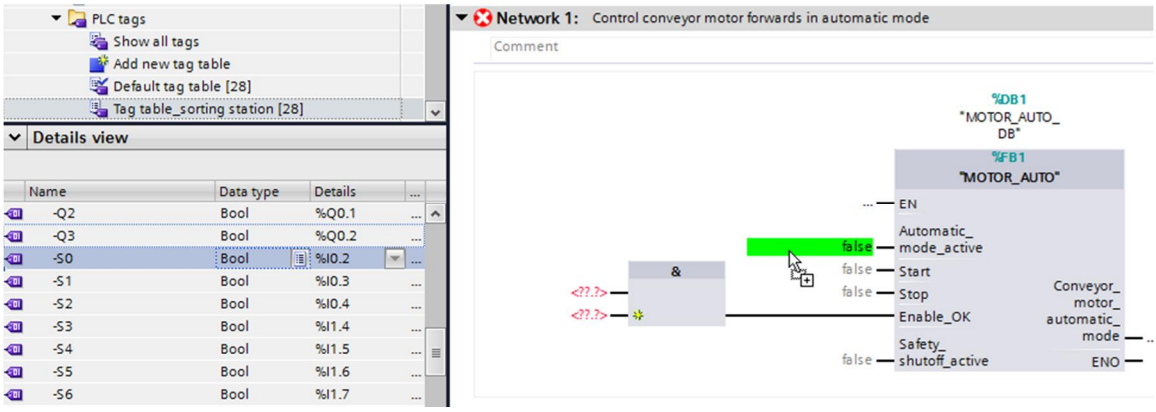


® Pour insérer un ET devant le paramètre d'entrée „Enable OK“ (Validation_OK), sélectionnez cette entrée et ajoutez l'opération ET en cliquant sur l'icône de la barre d'outils. (®)

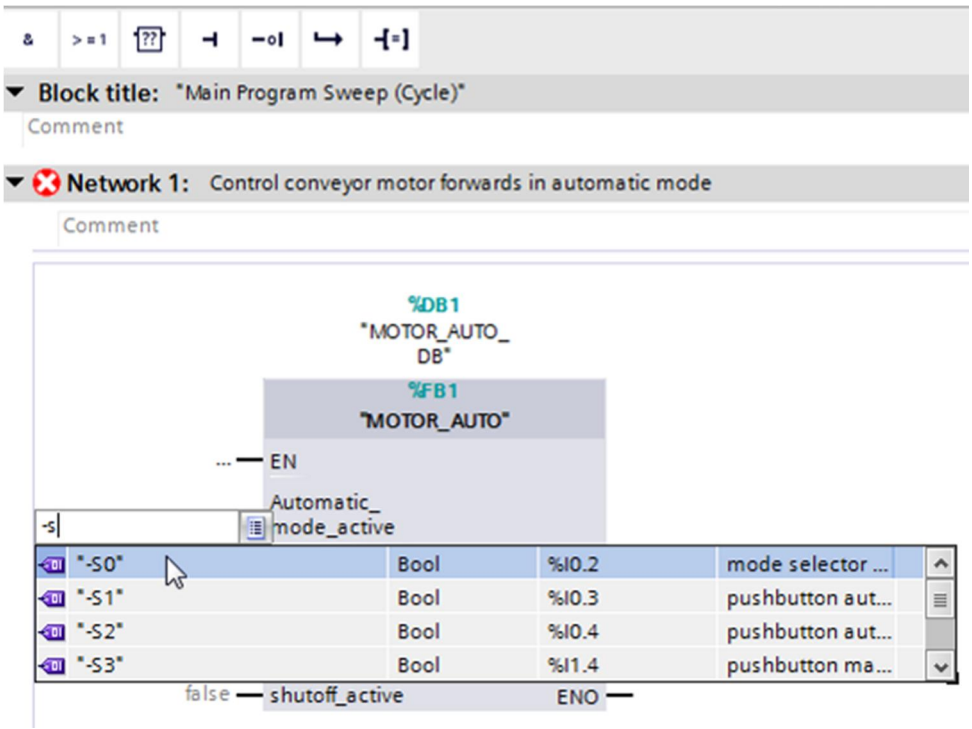


® Pour connecter le bloc avec les variables globales de la „table des variables_installation de tri", il existe 2 méthodes :

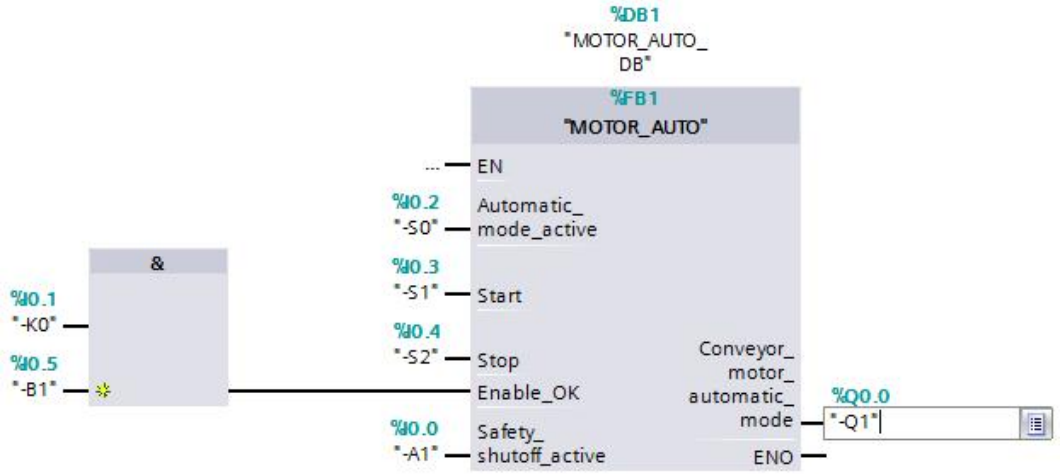
Vous pouvez sélectionner dans le navigateur du projet la „table des variables_installation de tri" et faire glisser la variable globale souhaitée de la vue détaillée sur l'interface de la FC1. (® Tag table_sorting station (Table des variables_installation de tri ® Vue détaillée ® -S0 ® Automatic_mode_active (Mode automatique activé))



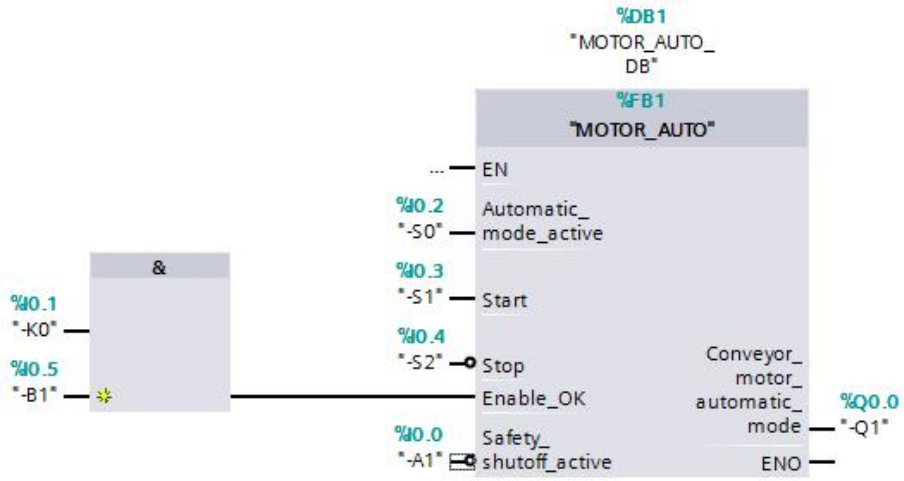
® Ou saisir pour <??.?> les caractères de début (par ex. : „-S") de la variable globale souhaitée et sélectionner dans la liste qui apparaît la variable d'entrée globale „-S0" (%E0.2). (® Automatic_mode_active (Mode automatique activé) ® -S ® -S0)



® Insérez les autres variables d'entrée „-S1“, „-S2“, „-K0“, „-B1“ et „-A1“, puis également la variable de sortie „-Q1“ (%A0.0) sur la sortie „Convoyeur_moteur_automatique“.

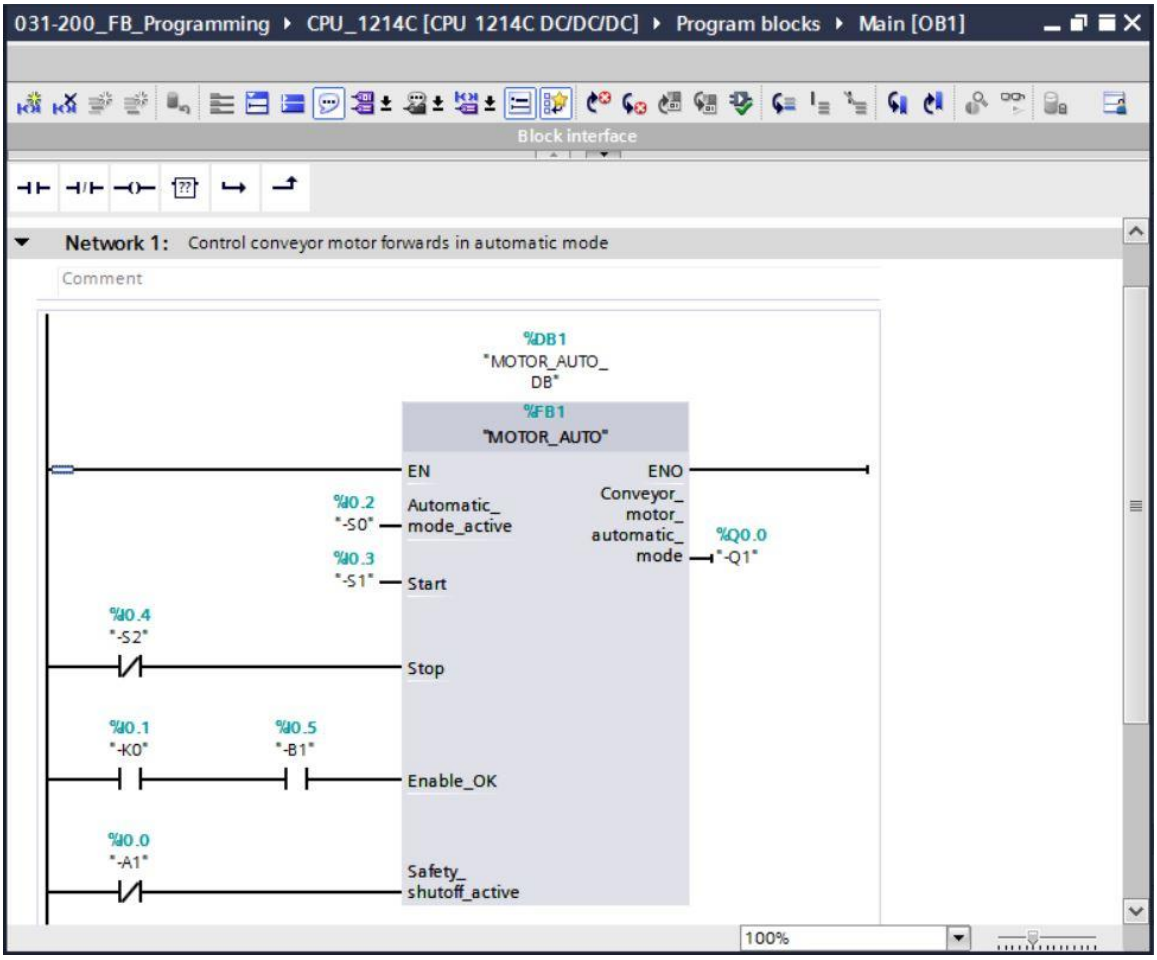


® Pour programmer une négation des requêtes des variables d'entrée „-S2“ et „-A1“, sélectionnez celles-ci et cliquez sur . (® -S2 ® ® -A1 ®)



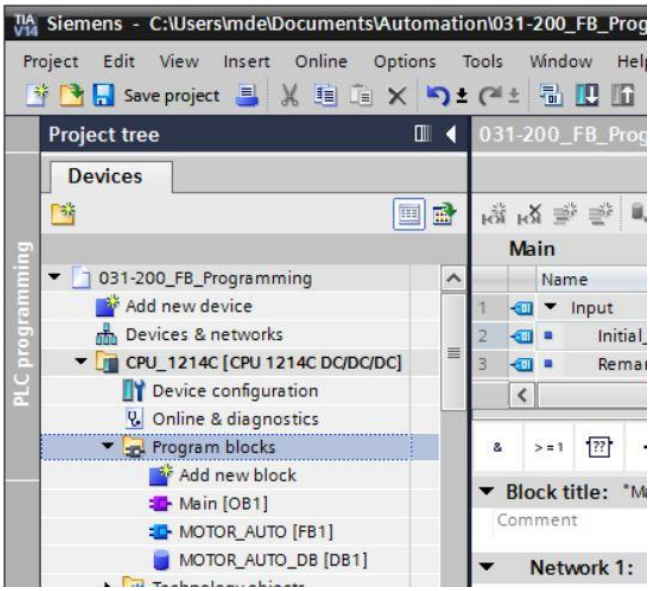
7.9 Résultat dans le schéma à contacts (CONT)

Voici à quoi ressemble le programme dans le schéma à contacts (CONT).

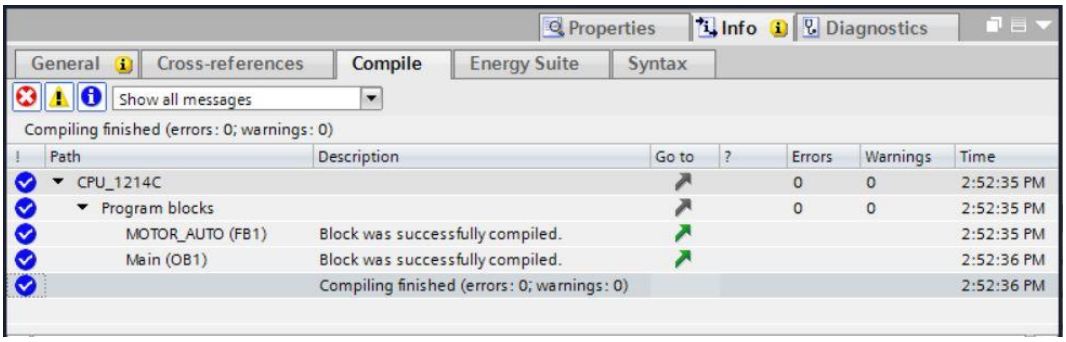


7.10 Enregistrer et compiler le programme

® Pour enregistrer votre projet, sélectionnez le bouton Save project dans le menu. Pour compiler tous les blocs, cliquez sur le dossier "Programm blocks" (Blocs de programme) et sélectionnez l'icône Compile (Compiler) dans le menu. (® Save project ® Programm blocks (Blocs de programme) ®)

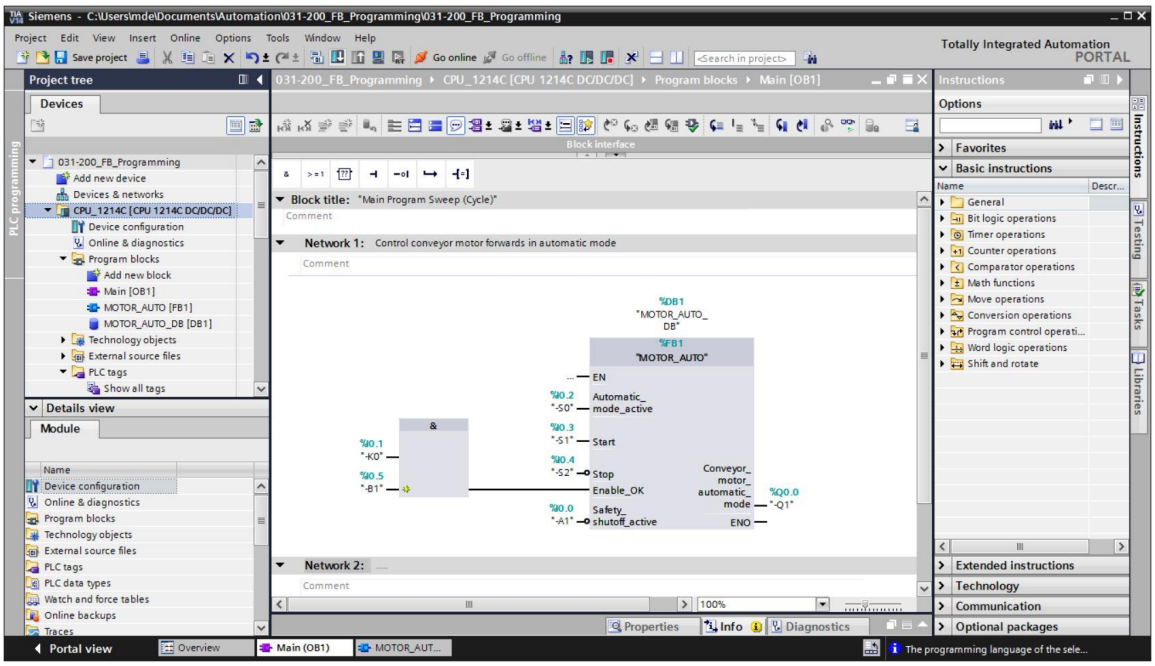


® Les blocs correctement compilés sont ensuite représentés dans la zone "Compile" de l'onglet "Info".



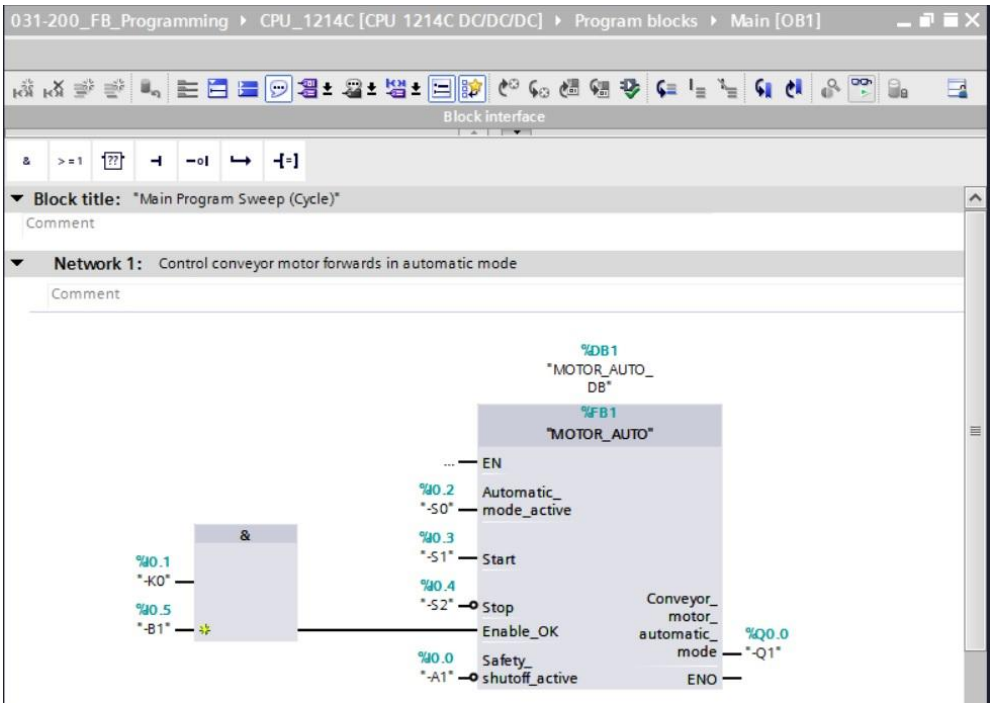
7.11 Charger le programme

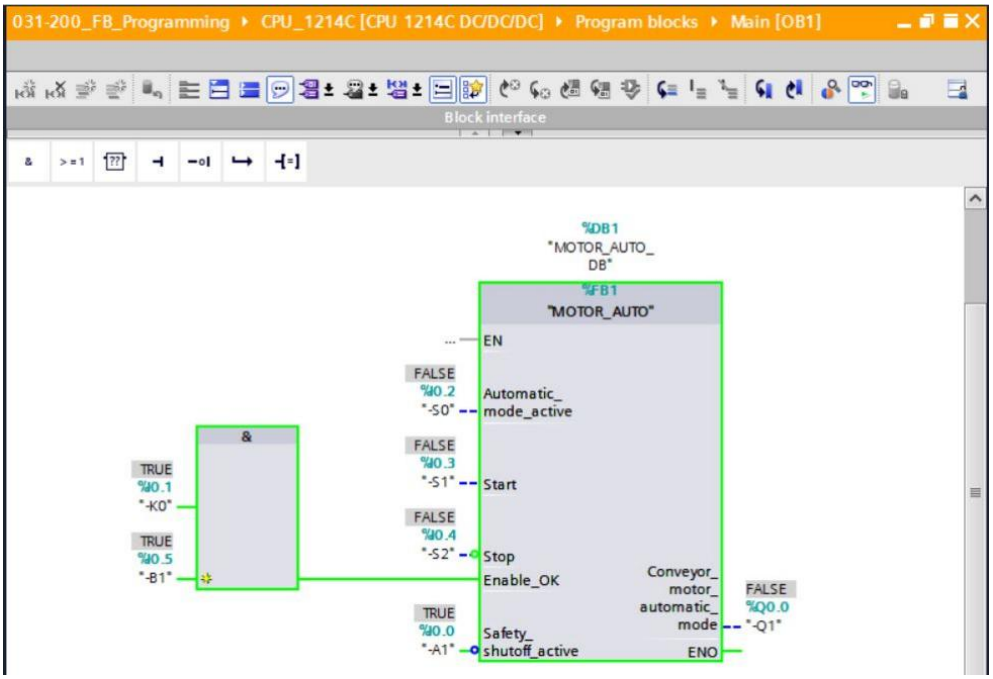
® Une fois que la compilation s'est correctement déroulée, vous pouvez charger toute la commande avec le programme créé, comme cela a été décrit dans les modules sur la configuration matérielle. (® )



7.12 Visualiser des blocs de programme

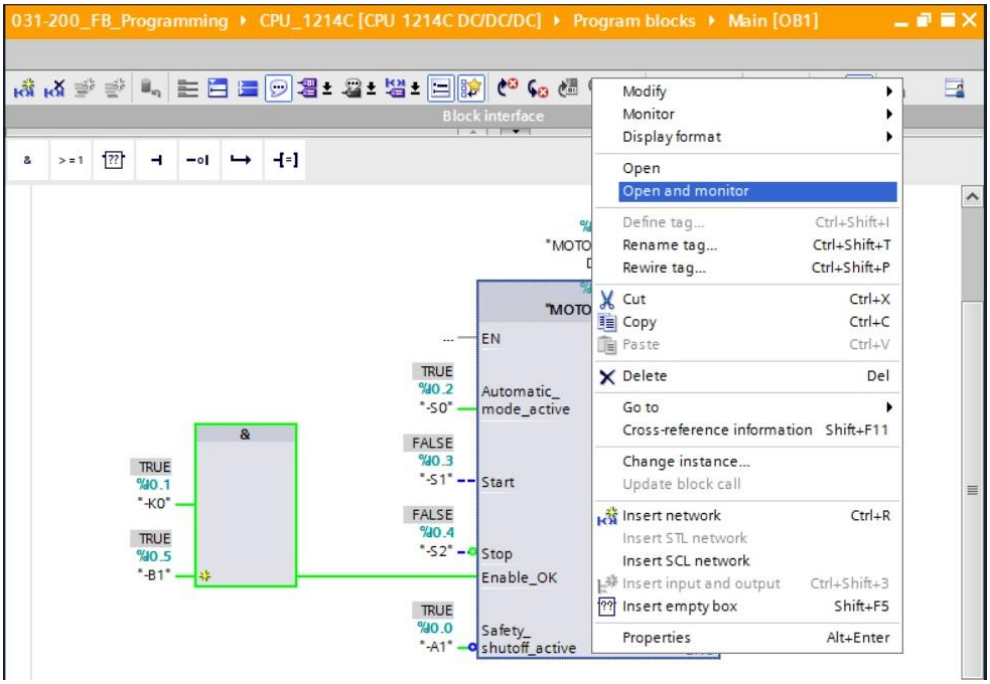
® Pour visualiser le programme chargé, le bloc souhaité doit être ouvert. Vous pouvez ensuite désactiver/activer la visualisation en cliquant sur l'icône . (® Main [OB1] ®)

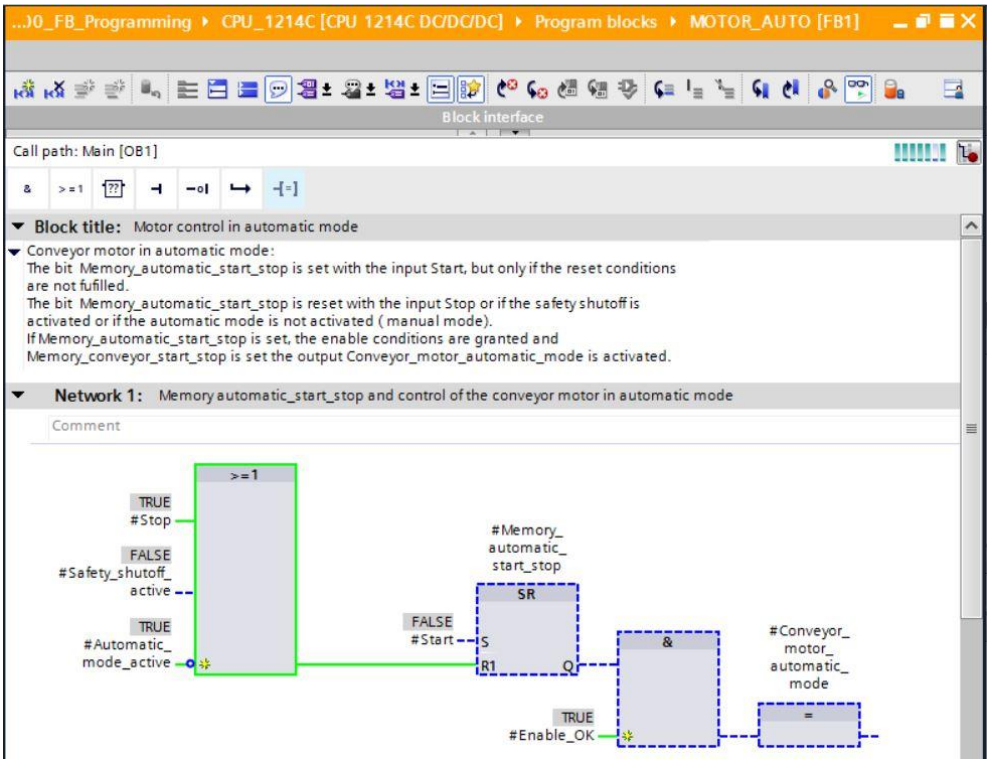




Remarque : *ici, la visualisation s'effectue sur la base des signaux et elle dépend de la commande. Les états logiques des opérandes sont représentés par TRUE ou FALSE.*

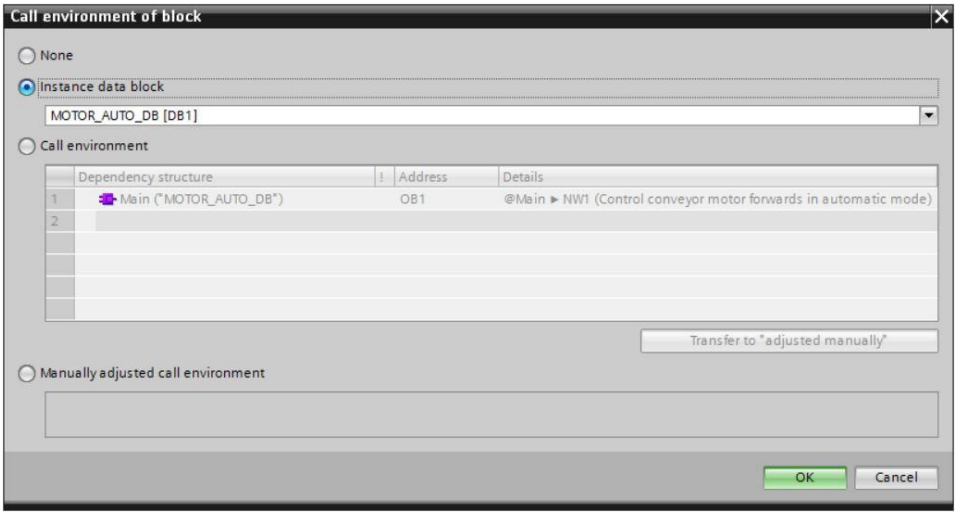
® Après avoir fait un clic droit de la souris, il est possible de sélectionner directement le bloc de fonction "MOTOR_AUTO" [FB1] appelé dans le bloc d'organisation "Main [OB1]" à "Open and monitor" (Ouvrir et surveiller). (® "MOTOR_AUTO" [FB1] ® Open and monitor (Ouvrir et surveiller))

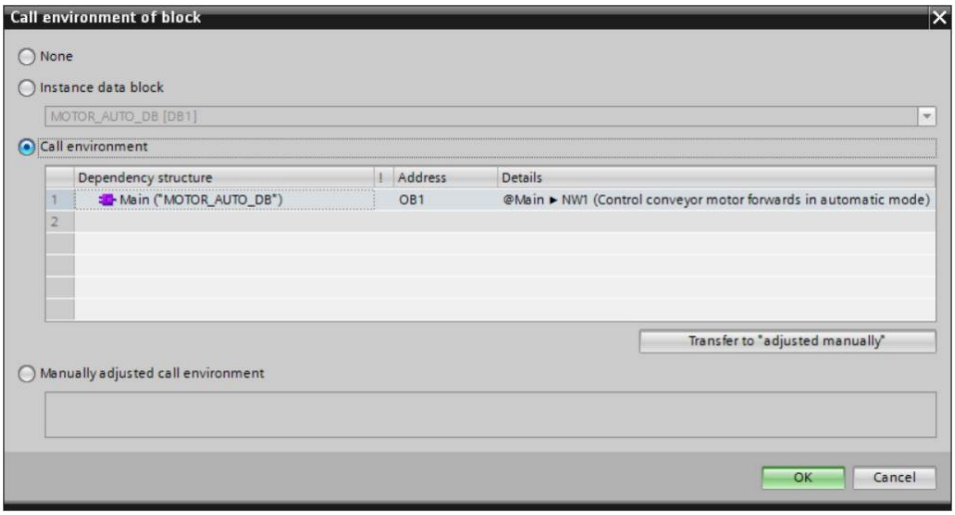




Remarque : *ici, la visualisation s'effectue selon les fonctions et elle dépend de la commande. L'actionnement des capteurs ou l'état de l'installation est représenté par TRUE ou FALSE.*

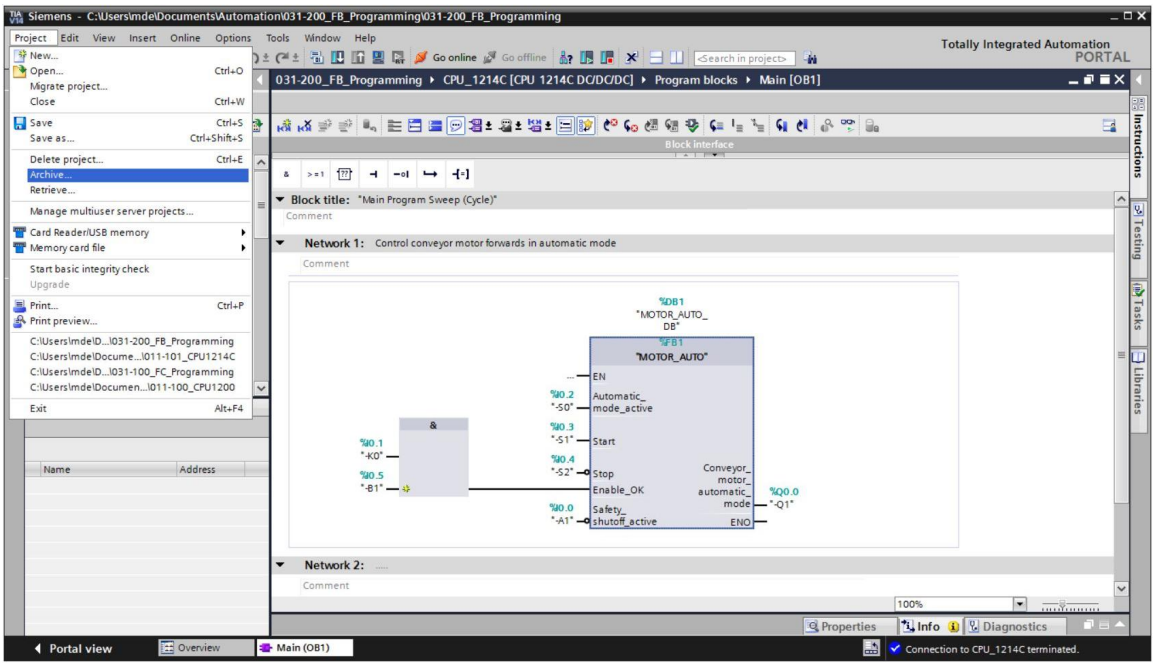
® Pour visualiser une occurrence d'un bloc fonctionnel „MOTOR_AUTO“ [FB1] appelé plusieurs fois, vous pouvez utiliser l'icône  . Vous avez la possibilité via l'environnement d'appel ou de définir l'environnement d'appel à travers le bloc de données d'instance. (®  ® Instance data bloc (Bloc de données d'instance) ® MOTOR_AUTO_DB1 [DB1] ® Call environment (Environnement d'appel) ® Address (Adresse) : OB1 ® Details (Détails) : Main NW1 ® OK)





7.13 Archiver le projet

® Pour terminer, nous voulons archiver le projet complet. Sélectionnez dans le menu ® "Project" (Projet) la commande ® "Archive..." (Archiver). Choisissez un dossier dans lequel vous souhaitez archiver votre projet et enregistrez-le sous "Archives projets TIA Portal" comme type de fichier. (® Project (Projet) ® Archive... (Archiver) ® Archives projets TIA Portal ® 031-200_Programmation de FB…. ® Save (Enregistrer)



7.14 Check-list

Nº	Description	Contrôlé
1	La compilation s'est déroulée correctement et sans message de d'erreur	
2	Le chargement s'est déroulé correctement et sans message de d'erreur	
3	Mise en circuit de l'installation (-K0 = 1) Tige du vérin rentrée / signalisation en retour activée (-B1 = 1) ARRET D'URGENCE (-A1 = 1) pas activé Mode AUTOMATIQUE (-S0 = 1) Bouton d'arrêt automatique pas actionné (-S2 = 1) Appui bref sur le bouton de démarrage automatique (-S1 = 1) Le moteur du convoyeur déclenche l'avance à vitesse fixe (-Q1 = 1) et reste en marche.	
4	Appui bref sur le bouton d'arrêt automatique (-S2 = 0) ® -Q1 = 0	
5	Activation ARRET D'URGENCE (-A1 = 0) ® -Q1 = 0	
6	Mode Manuel (-S0 = 0) ® -Q1 = 0	
7	Mise hors circuit de l'installation (-K0 = 0) ® -Q1 = 0	
8	Tige du vérin pas rentrée (-B1 = 0) ® -Q1 = 0	
9	Projet archivé correctement	

8 Exercice

8.1 Énoncé du problème – Exercice

Dans cet exercice, nous voulons compléter le bloc fonctionnel MOTOR_AUTO [FB1] avec une fonction d'économie d'énergie. Le bloc fonctionnel ainsi complété sera ensuite planifié, programmé et testé :

Pour des raisons d'économie d'énergie, le convoyeur ne doit marcher que lorsqu'une pièce est présente.

Par conséquent, la sortie Automatique_Moteur n'est commandée que lorsque le signal Mémoire_Automatique_Démarrage_Arrêt est mis à 1, les conditions de validation sont remplies et le signal Mémoire_Convoyeur_Démarrage_Arrêt est mis à 1.

Mémoire_Convoyeur_Démarrage_Arrêt est mis à 1 lorsque Capteur_Glissière_occupée signale la présence d'une pièce et réinitialisé lorsque Capteur_fin de convoyeur génère un front descendant ou le dispositif d'arrêt automatique de sécurité est activé ou que le mode automatique n'est pas activé (mode manuel).

8.2 Schéma technologique

La figure ci-dessous montre le schéma technologique pour l'application à réaliser.

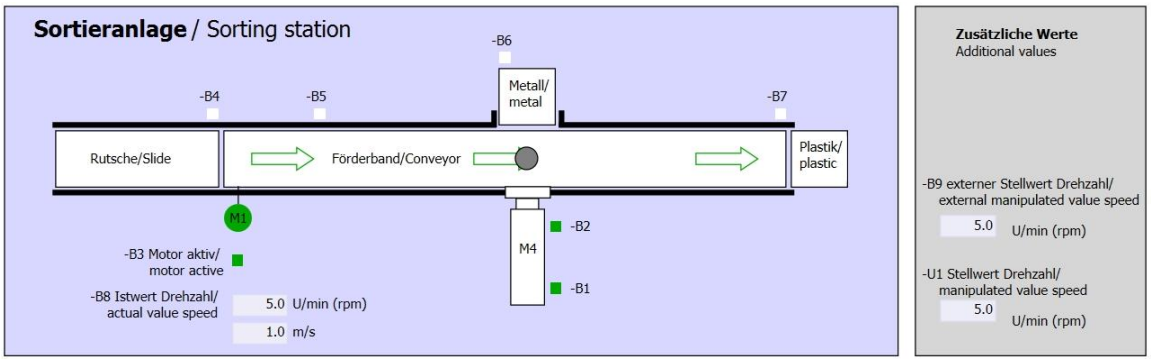


Figure 10 : Schéma technologique

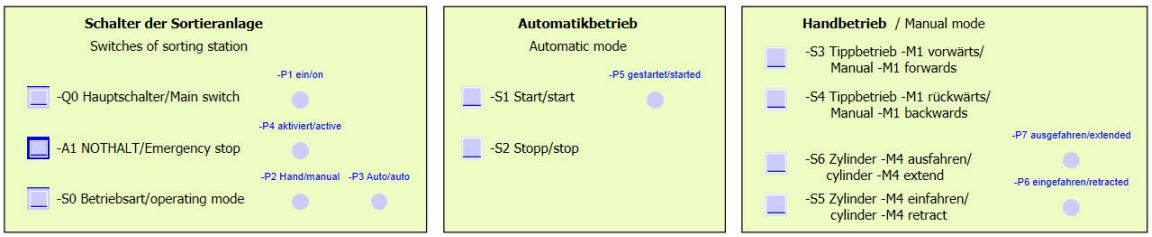


Figure 11 : Pupitre de commande

8.3 Tableau d'affectations

Cette application requiert les signaux suivants comme opérande global.

DE	Type	Code	Fonction	NC/NO
E 0.0	BOOL	-A1	Message ARRET D'URGENCE ok :	NC
E 0.1	BOOL	-K0	Installation "Marche"	NO
E 0.2	BOOL	-S0	Commutateur mode Manuel (0)/ Automatique (1)	Manuel = 0 Auto=1
E 0.3	BOOL	-S1	Bouton démarrage automatique	NO
E 0.4	BOOL	-S2	Bouton arrêt automatique	NC
E 0.5	BOOL	-B1	Capteur tige du vérin -M4 rentrée	NO
E 1.0	BOOL	-B4	Capteur glissière occupée	NO
E 1.3	BOOL	-B7	Capteur pièce à la fin du convoyeur	NO

DA	Type	Code	Fonction	
A 0.0	BOOL	-Q1	Moteur du convoyeur M1 avance à vitesse fixe	

Legende zur Belegungsliste

DE	Entrée TOR	DA	Sortie TOR
AE	Entrée analogique	AA	Sortie analogique
E	Entrée	A	Sortie
NC	Normally Closed (contact à ouverture)		
NO	Normally Open (contact à fermeture)		

8.4 Planification

Et maintenant, planifiez vous-même la mise en œuvre de l'application à réaliser.

Remarque : *vous trouverez des informations sur l'utilisation du front descendant dans SIMATIC S7-1200 dans l'aide en ligne.*

8.5 Check-list – Exercice

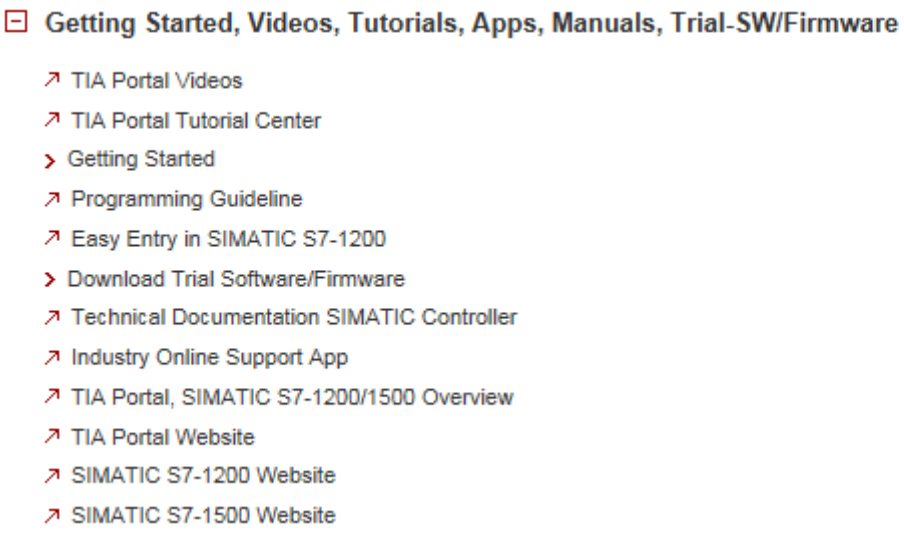
Nº	Description	Contrôlé
1	La compilation s'est déroulée correctement et sans message de d'erreur	
2	Le chargement s'est déroulé correctement et sans message de d'erreur	
3	Mise en circuit de l'installation (-K0 = 1) Tige du vérin rentrée / signalisation en retour activée (-B1 = 1) ARRET D'URGENCE (-A1 = 1) pas activé Mode AUTOMATIQUE (-S0 = 1) Bouton d'arrêt automatique pas actionné (-S2 = 1) Appui bref sur le bouton de démarrage automatique (-S1 = 1) Capteur glissière occupée activé (-B4 = 1) Le moteur du convoyeur déclenche l'avance à vitesse fixe (-Q1 = 1) et reste en marche.	
4	Capteur de fin de bande activé (-B7 = 1) ® -Q1 = 0	
5	Appui bref sur le bouton d'arrêt automatique (-S2 = 0) ® -Q1 = 0	
6	Activation ARRET D'URGENCE (-A1 = 0) ® -Q1 = 0	
7	Mode Manuel (-S0 = 0) ® -Q1 = 0	
8	Mise hors circuit de l'installation (-K0 = 0) ® -Q1 = 0	
9	Tige du vérin pas rentrée (-B1 = 0) ® -Q1 = 0	
10	Projet archivé correctement	

9 Informations complémentaires

Pour vous aider à vous familiariser ou à approfondir vos connaissances, des informations complémentaires tels que mise en route, vidéos, didacticiels, applis, manuels, guide de programmation et logiciel/firmware de démonstration sont disponibles sous le lien suivant :

www.siemens.com/sce/s7-1200

Vue d'ensemble des "Informations complémentaires"



Plus d'**informations**

Siemens Automation Cooperates with Education
siemens.com/sce

Supports d'apprentissage/de formation
siemens.com/sce/documents

Packages SCE pour formateurs
siemens.com/sce/tp

Partenaires SCE
siemens.com/sce/contact

L'entreprise numérique
siemens.com/digital-enterprise

Industrie 4.0
siemens.com/future-of-manufacturing

Totally Integrated Automation (TIA)
siemens.com/tia

TIA Portal
siemens.com/tia-portal

Automates SIMATIC
siemens.com/controller

Documentation technique SIMATIC
siemens.com/simatic-docu

Industry Online Support
support.industry.siemens.com

Catalogue de produits et système de commande en ligne Industry Mall
mall.industry.siemens.com

Siemens AG
Digital Factory
P.O. Box 4848
90026 Nuremberg
Allemagne