

Dossier de formation SCE

Siemens Automation Cooperates with Education | 05/2017

Module 032-200 TIA Portal

ooperates vith Education SIEMENS

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

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
 Nº diamina + CES3540 25N00 44P2
- 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 <u>roland.scheuerer@siemens.com</u>.

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

1	Ob	jectif	5
2	Co	nditions requises	5
3	Co	nfigurations matérielles et logicielles requises	6
4	Th	éorie	7
	4.1	Système d'exploitation et programme utilisateur	7
	4.2	Blocs d'organisation	8
	4.3	Mémoire image et traitement cyclique du programme	9
	4.4	Fonctions	11
	4.5	Blocs fonctionnels et blocs de données d'instance	12
	4.6	Blocs de données globaux	13
	4.7	Blocs de code réutilisables	14
	4.8	Langages de programmation	15
5	Én	oncé du problème	16
6	Pla	nification	16
	6.1	ARRET D'URGENCE	16
	6.2	Mode automatique - Moteur du convoyeur	16
7	Ins	tructions structurées par étapes	17
	7.1	Désarchiver un projet existant	17
	7.2	Création d'une nouvelle table des variables	18
	7.3	Création de nouvelles variables dans une table des variables	20
	7.4	Importation de la "table des variables_installation de tri"	21
	7.5	Création du bloc fonctionnel FB1 "MOTOR_AUTO" pour le moteur du convoyeur en mode automatique	24
	7.6	Définir l'interface du FB1 "MOTOR_AUTO"	26
	7.7	Programmation du FB1 : MOTOR_AUTO	29
	7.8	Programmation du bloc d'organisation OB1 – Commande du convoyeur vers l'avant en mode automatique	э 37
	7.9	En langage de programmation LAD (CONT) (schéma à contacts), le résultat est le suivant	42
	7.10	Enregistrer et compiler le projet	43
	7.11	Charger le programme	44
	7.12	Visualiser les blocs de programme	45
	7.13	Archivage du projet	48
8	Lis	te de contrôle	49
9	Ex	ercice	50
	9.1	Énoncé du problème - exercice	50
	9.2	Planification	50
	9.3	Liste de contrôle - Exercice	51
1() Info	ormations complémentaires	52

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
- 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	lgnorer
Programme cyclique	1, ≥ 123	Ignorer
Alarme horaire	10 à 17, ≥ 123	-
Alarme de mise à jour	56	lgnorer
Temps de surveillance du cycle dépassé une fois	80	STOP
Alarme de diagnostic	82	lgnorer
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 :

 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.

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.



1. État des entrées sécurité intrinsèque mémoire image des entrées mémoriser.

3. Transmettre l'état de la MIS aux sorties.

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).



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

 \rightarrow 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 \rightarrow Project

(Projet)→Retrieve (Désarchiver). Confirmez votre choix avec "Open (Ouvrir)". (→Project

(Projet)→Retrieve (Désarchiver)→ Sélectionner une archive zap → Open (Ouvrir))

Project	Edit	View	Insert	Online	Option			
👫 New.								
👌 Oper	ı				Ctrl+O			
Migra	ate proj	ect						
Close	2				Ctrl+W			
Save					Ctrl+S			
Save	85			Ctrl	+Shift+S			
Dele	te proje	ct			Ctrl+E			
Archi	ve							
Retri	eve							
👕 Card	Reader	/USB m	emory		•			
Mem	ory car	d file			•			
Upgr	ade							
D:IAu	tomati	on\013_	10\013	_101_CPU	314C			
Dilar	itomati	on\012_	10\012	_101_CPU	1516F			
DIMU	rlagen	projekt_	Webserv.	ITank_V1	3_SP1			
D:IVc	D:11032-200_FB-Programmierung_S7-314							
D:///c D:///c	032-20	0_10410						
D:/Vc D:/Vc D://	032-20 Itomati	si\012	-100_CPU	1500_V13	B_SP1			

→ 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)".

M Siemens - D:\Automation\012_101_CPU1516F\0	012_101_CPU1516F
Project Edit View Insert Online Options To	ools Window Help C ⁴ ± 🗟 🛄 🛄 🚆 🎇 💋 Goonline 💋 Go
Project tree	
Devices	
1 O O 1	
▼ □ 012_101_CPU1516F	•
🗧 📑 Add new device	
📅 🛗 Devices & networks	
CPU_1516F [CPU 1516F-3 PN/DP]	
Device configuration	
😼 Online & diagnostics	
🕨 🚘 Program blocks	
Technology objects	
External source files	
▼ 2 PLC tags	
how all tags	
📑 Add new tag table	
💥 Standard-Variablentabelle [54]	
PLC data types	
Watch and force tables	
Online backups	
🕨 🔄 Traces	and the second se
Program info	
Device proxy data	
PLC alarms	

→ 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))

VA	Siemens - D:\Automation\012_101_CPU1516	FV012_101_	CPU1516	F			
Pr	oject Edit View Insert Online Options	Tools Wind	low Hel	P	S 60.00	line 🔊	Go offi
-	Project tree			Ta ¹ ET	0001	ine y	Goom
	Devices						
	B 0 0						
	▼ [] 012_101_CPU1516F	^					
ť	Add new device						
Sta	🚡 Devices & networks						
	CPU_1516F [CPU 1516F-3 PN/DP]						
	Device configuration						
	🗓 Online & diagnostics						
	🕨 🔂 Program blocks	≡					
	Technology objects						
	External source files						
	🔻 🔚 PLC tags						
	a Show all tags						
	🂕 Add new tag table	and the second se					
	🎽 Standard-Variablentabelle [54]						1.0
	👆 Tag table_sorting station [0]						
	PLC data types						
	Watch and force tables						
	🕨 🙀 Online backups						
	🕨 📴 Traces						
	Program info						

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

M Siemens - D:\Automation\012_101_CPU1516F	012_101_CPU1	516F				_							a x
Project Edit View Insert Online Options 1	fools Window	Help							Т	tally Int	oursted Au	tomation	
📑 📑 🔒 Save project 🔳 🐰 🗐 🗐 🗙 🍤 生	@ * 집 🖪	🖸 🖳 📑 💋 Go online	🖉 Go offline 🔥 🖪	. × ∃						Any in	egrated Ad	PORTA	L
Project tree	□ < 012	_101_CPU1516F + CPI	J_1516F [CPU 1516F-	3 PN/DP] > PLC	tags ▶ Ta	ig table_	_sorting	station [0]				_ # = >	: (
Devices										🗉 Tags	User (constants	10
1900		🥑 🖻 😤 ûx											Tas
g		Tag table sorting station	n										Ś
▼ 1 012_101_CPU1516F	^	Name	Data type	Address	Retain	Visibl	Acces	Comment					
Add new device	1	<add new=""></add>	and the										
Devices & networks				620		(00)	0						5
▼ 1 CPU 1516F [CPU 1516F-3 PN/DP]													
Device configuration													00
V. Online & diagnostics													
Program blocks	=												1
Technology objects													11
External source files													1
PLC tags													1
Show all tags													1
Add new tap table													11
Standard-Variablentabelle [54]													11
Tag table sorting station [0]													11
PIC data types													1
Watch and force tables							0	Properties	*i Info	2 Di	annostics		Л
Online backups								stroperties		0.01	ignostics	and the second s	-
Traces	G	eneral											4
Program info		9	Tree										4
Device proxy data			1ag										4
PLC alarms			General										1
Text lists													48
Local modules			•	Name:									
Common data				Data type:									
Documentation rettinor	~			Addresse									
✓ Details view				Address:									
	1				Retained								
Data tuna Comment Name				Comment:									-
out type connent name		-											4

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)

ro	gran	nming ► CPU_1	516F [CPU	1516F-3 PN	DP] 🕨	PLC tag	s 🕨 Tag	g table_sorting st	ation [1]	_ = = ×
								🕣 Tags	🗉 User	constants
1	-	🖻 😤 🕅 🗶								
1	ag t	table_sorting st	ation							
-		Name	Data type	Address	Retain	Visibl	Acces	Comment		
1		Q1	Bool 🔳	%10.0						
2		<add new=""></add>				2	V			

→ 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 → ∞)

station [1] 🛛 🗖 🗮 🗙
User constants
-
×
~

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

ro	gran	nming ► CPU_1	516F [CPU	1516F-3 PN	'DP] →	PLC tag	s 🕨 Tag	g table_sorting st	ation [1] 🛛 💻 🖬 🚍	×
								🕣 Tags	User constants	
1	-	🖻 😤 🕅							E	4
	Tag t	able_sorting sta	ation							
		Name	Data type	Address	Retain	Visibl	Acces	Comment		
1	-	Q1	Bool 🔳	%Q0.0 💌				conveyor motor -M1	forwards fixed speed	
2		<add new=""></add>				1	V			

 → 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)".

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

								🕣 Tags	User constants
3h	* [🕈 😤 🕅 🗶							-
1	lag ta	ble_sorting sta	ation						
	P	lame	Data type	Address	Retain	Visibl	Acces	Comment	
								A 12 YO M A REAL AND A	
		Q1	Bool	%Q0.0				conveyor motor -M1	forwards fixed speed
		Q1 Q2	Bool Bool	%Q0.0 %Q0.1				conveyor motor -M1 conveyor motor -M1	forwards fixed speed backwards fixed speed

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)".

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



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

 $(\rightarrow$ SCE_FR_020-100_table des variables installation de tri... \rightarrow 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 \rightarrow OK.

Import co	mpleted with warnings (0032:000031) $ imes$
	Import completed with warnings.
	Detailed information is shown in the import log file.
	Click here to view the log file.
	ОК

- → 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).

								a Tags	User constants	
) 🖻 🤭 🕅								-	
Tag	g table_sorting	g station								
	Name	Data ty	pe Address	R	etain	Visibl	Acces	Comment		
4	Q1	Bool	≝ %Q0.0	-				conveyor motor -M1 forwards fixed speed		~
-14	ബ റാ	Rool	\$001					conveyor motor -M1 backwards fixed speed		
1	Insert row							return signal emergency stop ok (nc)		Ш
100	Add row							main switch "ON" (no)		
X	Cut		Ctrl+X					mode selector manual(0) / automatic(1)		
睡	Сору		Ctrl+C					pushbutton automatic start (no)		
1	j Paste		Ctrl+V					pushbutton automatic stop (nc)		-
×	Delete		Del					sensor cylinder -M4 retracted (no)		
1	Rename		F2					sensor cylinder -M4 extended (nc)		
-	Concernation and	information 6	Li6.211					sensor motor -M1 actice (pulse signal for positioning) (no)		
-	crossreterence	anomation 5	nna+r i i					sensor part at slide (no)		
*	Monitor all							sensor metal part (no)		
	Import file							sensor part in front of cylinder -M4 (no)		
	Export file							sensor part at end of conveyor (no)		
ø	Properties				<u></u>			pushbutton manual mode conveyor -M1 forwards (no)		
100	-24	0001	1011.2					pushbutton manual mode conveyor -M1 backwards (no)		
	-55	Bool	%11.6					pushbutton manual mode cylinder -M4 retract (no)		
-	-56	Bool	%11.7					pushbutton manual mode cylinder -M4 extend (no)		
-	-Q1	Bool	%Q0.0					conveyor motor -M1 forwards fixed speed		
	🖸 -Q2	Bool	%Q0.1					conveyor motor -M1 backwards fixed speed		
1	-Q3	Bool	%Q0.2		8			conveyor motor -M1 variable speed		
-	-M2	Bool	%Q0.3					cylinder -M4 retract		
-	-M3	Bool	%Q0.4					cylinder -M4 extend		
-	💷 -P1	Bool	%Q0.5					display "main switch on"		
-	-P2	Bool	%Q0.6					display_manual mode"		-
	en -P3	Bool	\$00.7		0			display automatic mode"		Y

→ 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.

 $(\rightarrow$ Project (Projet) \rightarrow Save as (Enregistrer sous) ... \rightarrow 032-200_Programmation de FB

 \rightarrow Save (Enregistrer))

Project Edit View Incert Online	Ontionr	Tools 1	Mindow	Hele									
New	opuons	+ CI+			1 🖪 🗹 c	o online all (o offline	. 18 18	v .			Totally Integrated	Automation
Open	Ctrl+O	T (100		a int 🏓 o	o onnine just o	o onnine o						TORT
Migrate project			01	2_101	_CPU1516	F • CPU_15	16F [CPU 1	516F-3	PN/DP]	PLC t	ags • Tag table_sorting station [28]		
Close	Ctrl+W											🕣 Tags 🔳 U	ser constants
Save	Ctrl+S	m.	ə 🚽	i =62	-								-
Save as Ctrl+	Shift+S			Tank		a station							
Delete project	Ctrl+E			Tagu	able_sorun	gstation		Les de la	Large 1				
Archive			<u>^</u>		Name	Data type	Address	Retain	VISIDI	Acces	Comment		
Retrieve			1		-A1	Bool	E %10.0				return signal emergency stop ok (nc)		-
			4	-10	-K0	BOOI	1610.1				main switch "ON" (no)		
Card ReaderIUSB memory			3		-50	Bool	%10.2				mode selector manual(0) / automatic(1)		
wemory card me			4	-	-51	Bool	%10.3				pushbutton automatic start (no)		
Upgrade			5	-0	-52	Bool	%10.4				pushbutton automatic stop (nc)		
Print	Ctrl+P		- 0	-	-61	Bool	%10.5				sensor cylinder M4 retracted (no)		
Print preview			7	-0	-B2	Bool	%10.6				sensor cylinder -M4 extended (nc)		
Diautomatical012 10 1012 101 CRUI	6165		8	-0	-83	Bool	%10.7				sensor motor -M1 actice (pulse signal for		
CilicersispelDo 1032 100 EC-Program	mina		9	-0	-84	Bool	%11.0				sensor part at slide (no)		
Diautomation 013 10 013 101 CPU3	140		10	-00	-85	Bool	%11.1				sensor metal part (no)		
DiVorlagenprojekt Websery Tank V13	S SP1		11	-0	-86	Bool	%11.2				sensor part in front of cylinder -M4 (no)		
D: 1032-200 EB-Programmierung S7-3	114		12	-00	-87	Bool	%11.3				sensor part at end of conveyor (no)		
D:Automatisi\012-100 CPU1500 V13	SP1		13	-00	-53	Bool	%11.4				pushbutton manual mode conveyor -M1		
12100 Contraction			14	-0	-54	Bool	%11.5				pushbutton manual mode conveyor -M1		
Exit			15	-0	-55	Bool	%11.6				pushbutton manual mode cylinder -M4 re		
Online backups			16	-03	-56	Bool	%11.7				pushbutton manual mode cylinder -M4 ex		
Traces			17	-0	-Q1	Bool	%Q0.0				conveyor motor -M1 forwards fixed speed		
Program info			18	-00	-Q2	Bool	%Q0.1				conveyor motor -M1 backwards fixed speed		
Device proxy data			19	-00	-Q3	Bool	%Q0.2				conveyor motor -M1 variable speed		
PLC alarms			20	-0	-M2	Bool	%Q0.3				cylinder -M4 retract		
Text lists			21	-0	-M3	Bool	%Q0.4				cylinder -M4 extend		
Local modules			22	-00	-P1	Bool	%Q0.5				display "main switch on"		
Common data			23	-03	-P2	Bool	%Q0.6				display "manual mode"		
Documentation rettinor			× 24	-03	-P3	Bool	%Q0.7				display "automatic mode"		
✓ Details view			25	-00	-P4	Bool	%Q1.0				display "emergency stop activated"		
			26	-67	-85	Bool	%011				display, automatic mode started"		
Data type Comment Name											Properties 1	Info 3 2 Diagnosti	s I-
Refine		20	_	-									

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.

 $(\rightarrow$ PLC programming (Programmation de l'API) \rightarrow Add new block (Créer un bloc) \rightarrow



nens - C:\Users\sp	e\Documents\	Automatisierung\032_200_FB-Programm	ng/032_200_F8-Programming	The Review of Automation
				Totally Integrated Automation PORTA
		Device: CPU1516F	Add new block	
Devices &	*	Show all objects	Name: Block_1	
PLC programming	*	🥚 Add new block	Language: F8D	
Motion & echnology	*		Organization block	
)rive barameterizatio	n î 🏊	Show cross-references	Description: Functions are code blocks or subroutines without dedicated n	nemory.
		Show program structure	Function block	
Inline &	10			
			Function	
		Help		
			Data block More	
			> Additional information	
			Add new and open	Add

→ 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)".

 $(\rightarrow \text{Name (nom)} : \text{moteur} \rightarrow \text{Langage} : \text{FBD (LOG)} \rightarrow \text{Number (numéro)} : \text{Automatic}$ (automatique) $\rightarrow \blacksquare$ Add next and open (créer et ouvrir) \rightarrow Add (ajouter))

Add new block				
Name:				
MOTOR_AUTO				
	Language:	FBD		
OB	Number:	1		
Organization		O Manual		
block		 Automatic 		
FB	Description:			
Function block	Function blocks are so that they remain	e code blocks that store the n available after the block h	ir values permanently in instan las been executed.	ce data blocks,
	,			
Function				
DB				
Data block				
	More			
> Additional inform	nation			
Add new and open				Add

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.

	Name					
		Data type	Default value	Retain	Accessible f.	Visible in .
	▼ Input			1		
2	Add new>			-	1 A	<u> </u>
-01	▼ Output	·			í ă	
4	Add new>				ā	
5 🕢	 InOut 					
5	Add new>					
7 -0	▼ Static					
в	Add new>					
	▼ Temp					
10	Add new>					
11 🕣	 Constant 					
12	Add new>					
<		III				
- Bloc	ck title:		10. Martin			
Comr	nent					
• N	letwork 1:					
C	omment					
C	omment					

 → 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) \rightarrow Bool \rightarrow control of the conveyor motor in automatic mode (commande du moteur du convoyeur en mode automatique)

03	2-2	00	_FB-Programming CPU_1516F	[CPU 15	16F-3 P	N/DP] 🕨	Progra	am bloc	ks ▶ I	MOTOR_AUTO [FB1]	_∎≡×
ið	i Id	K 3	🖗 🔮 💺 🚍 🚍 💬 📲 ± 🍇	2 ± 🖃 8	\$ 60	60 ta	📾 🕹	${}^{I}\equiv {}^{X}\equiv$	°. °	° 🔢	
	M	DTO	DR_AUTO								
		Na	ime	Data t	Defaul	Retain	Acce	Visibl	Setp	Comment	
1	-	-	Input								
2			<add new=""></add>								
3	-	-	Output								
4	-		Conveyor_motor_automatic_mode	Bool 🔳	false	N 💌				Control of the conveyor motor in	automatic mode
5			<add new=""></add>								
6		•	InOut								
7			<add new=""></add>								
8	-	•	Static								
9			<add new=""></add>								
10		•	Temp								
11			<add new=""></add>								
12	-	•	Constant								
13			<add new=""></add>								

→ 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)".

 $(\rightarrow automatic_mode_active (mode_automatique_actif) \rightarrow BOOL \rightarrow automatic mode activated (mode automatique activé))$

→ 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.

03	2-2	00	_FB-Programming CPU_1516F	[CPU 15	16F-3 P	N/DP] ▶	Progr	am bloc	.ks ▶	MOTOR_AUTO [FB1] 🛛 🗕 🖬 🖬	i×
ю́	i iq	X I	🖗 🔮 💺 🖿 🚍 💬 🕄 ± 🍣	2 ± 🖃 8	\$ 6 0	€ ∂ @≣	9 B	I _≡ " _≡	e ⁰ , 0	°. 📙	4
	M	DTO	DR_AUTO								
		Na	me	Data t	Defaul	Retain	Acce	Visibl	Setp	Comment	
1	-	•	Input								^
2	-		Automatic_mode_active	Bool	false	Non-r				Automatic mode activated	
3	-	=	Start	Bool	false	Non-r				Pushbutton automatic start	
4	-		Stop	Bool	false	Non-r				Pushbutton automatic stop	
5	-	=	Enable_OK	Bool	false	Non-r				All enable conditions OK	
6	-00		Safety_shutoff_active	Bool 🔳	false	N 💌				Safety shutoff active e.g. emergency stop operated	
7			<add new=""></add>								
8	-00	-	Output								
9	-		Conveyor_motor_automatic_mode	Bool	false	Non-r				Control of the conveyor motor in automatic mode	
10			<add new=""></add>								
11	-	-	InOut								
12			<add new=""></add>								
13	-01	-	Static								
14			<add new=""></add>								
15	-	•	Temp								
16			<add new=""></add>			1		0			~

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)". (\rightarrow Memory_automatic_start_stop (mémoire_automatique_marche_arrêt) \rightarrow Bool \rightarrow Memory used for start/stop automatic mode (mémoire utilisée pour démarage/arrêt du mode automatique))

03	2-2	00	_FB-Programming CPU_1516F	[CPU 15	16F-3 P	N/DP] ▶	Progr	am bloo	ks ▶ I	MOTOR_AUTO [FB1] _ 🖬 i	ΞX
Ň	Ŀв	5 3	0 🔄 💺 🖿 🚍 💬 🕾 ± 🎗	2 ± 😑	\$ CO	60 ell	€ ₽	1 ₂ 3 ₂	e [,] °	°⊳ 1 2	
	MC	T	DR_AUTO								
		Na	me	Data t	Defaul	Retain	Acce	Visibl	Setp	Comment	
1	-	-	Input								1
2	-0		Automatic_mode_active	Bool	false	Non-r				Automatic mode activated	
3	-		Start	Bool	false	Non-r				Pushbutton automatic start	
4	-		Stop	Bool	false	Non-r		~		Pushbutton automatic stop	
5	-		Enable_OK	Bool	false	Non-r				All enable conditions OK	
6	-		Safety_shutoff_active	Bool	false	Non-r				Safety shutoff active e.g. emergency stop operated	
7		=	<add new=""></add>								L
3	-	•	Output								
9	-		Conveyor_motor_automatic_mode	Bool	false	Non-r				Control of the conveyor motor in automatic mode	
0			<add new=""></add>								
ÍŤ	-	•	InOut								
12			<add new=""></add>								
13	-	•	Static								
14			Memory_automatic_start_stop	Bool 🔳	false	N 💌				Memory used for start/ stop automatic mode	

→ 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))

	MC	т	DR_AUTO									
		Na	me	Data t	Defaul	Retain	Acce	Visibl	Setp	Comment		
1	-0	٠	Input									ŝ
2	-		Automatic_mode_active	Bool	false	Non-r		(reco		Automatic mode activated		1
3	-		Start	Bool	false	Non-r				Pushbutton automatic start		J
4	-		Stop	Bool	false	Non-r				Pushbutton automatic stop	1	1
5	-0		Enable_OK	Bool	false	Non-r		and a		All enable conditions OK		l
6	-0		Safety_shutoff_active	Bool	false	Non-r				Safety shutoff active e.g. emerg	enc	i
7	-	•	Output									
8	-0		Conveyor_motor_automatic_mode	Bool	false	Non-r				Control of the conveyor motor i	n a	
9	-	•	InOut									
10			<add new=""></add>									
11	-	•	Static									
12			Memory_automatic_start_stop	Bool	false	Non-r		\sim		Memory used for start/ stop aut	om 🗸	•
	<					111					>	
-	RIO	ck	title: Motor control in automatic mode		(113)						/	
▼ C T a I I F	ionv he b re n he b ctiv Men ior re	eyo oit l oit l ate mol ory eas	r motor in automatic mode: Memory_automatic_start_stop is set with fuliled. Memory_automatic_start_stop is reset w d or if the automatic mode is not activat ry_automatic_start_stop is set, the enab _conveyor_start_stop is set the output C ons of energy efficiency the conveyor mode ons of energy efficiency the conveyor mo	h the inpu vith the in ted (man ile conditi onveyor_i otor shoul	ut Start, bu put Stop (ual mode ons are g motor_au d only rur	ut only if the set). ranted ar tomatic_t i if a part i	he reset afetyshu nd mode is is preser	conditio toff is activate nt.	ns d.		=	100 MM
•	nd i nd i rift	he Net	e wemory_convergences at Sensor_end_ automatic mode is not activated (manu work 1: Memory automatic_start_stop	pf_convey ual mode) p and con	or or if the	e safety sl	nutoffis nutoffis r motor i	nsor_sild activated in autom	atic moi	de		

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

 Image: --[=] (affectation) et la faire glisser sur le réseau 1 (une ligne verte apparait, pointeur avec symbole +).

(\rightarrow instruction \rightarrow Instructions simples \rightarrow fonction logique combinatoire sur bits \rightarrow \blacksquare --[=]

,	_20(0_FB-Programming ► CPU1516F [C	x	Instructions					Þ							
											Optio	ons				
ьŝ	i d	(🗇 🛎 👞 🖿 🎘 🚍 💬 🗐 ± 🖉	1 ± 🖃 🛍	e 6 6	Ç:: 12 I.	1_ 0	92 HJ		E	1	Ċ			iti j	•	
	MC				·				× Favorites							=
		Name	Data type	Default value	Retain	Accessi	Visible	Setpoint	Comment		_	uvonic				-
1		▼ Input								^	8	> = 1	??	-	-01	
2	-00	 Automatic_mode_active 	Bool	false	Non-retain				Automatic mode a							
3	-00	 Start 	Bool	false	Non-retain				Pushbutton autom	≡	→	-1-1	SR	RS		
4		 Stop 	Bool	false	Non-retain	~			Pushbutton autom							
5	-00	Enable_OK	Bool	false	Non-retain				All enable conditio							
6	-00	 Safety_shutoff_active 	Bool	false	Non-retain				Safety shutoff activ.	. 1	✓ B	asic in	structi	ons		_
7		 Output 									<u>ک</u>	Genera				~
8		 Conveyor_motor_automatic_mode 	Bool	false	Non-retain				Control of the conv	.	 Bit logic opera 					
9	-00	 InOut 								~		E &		AN) logic	
10	K	· · · · · · · · · · · · · · · · · · ·							>	> III >=1 OR logic					logic o	≡
	1		_		-					-		Ξx		EXC	LUSIV	
8		>=1 [??] → -ol → -[=] SR	RS									E[=	1	Ass	ianme	
												E [/=	=]	Neo	jate as	
•	r	<pre>letwork 1: Memory automatic_start_stop</pre>	and contro	of the convey	or motor in a	utomatic r	node			\sim		🗉 [R]	Res	et outp	
	C	omment										🗉 [S]	Set	output	
	_											E SET	BF	Set	bit field	
										=		E RES	ET_BF	Res	et bit fi	
											SR Set/re			reset fl		
												🗉 RS		Res	et/set fl.	
													E IPI Scan oper			

→ Faire glisser le paramètre de sortie #Conveyor_motor_automatic_mode sur <??.?> audessus du bloc qui vient d'être ajouté. Pour sélectionner plus facilement un paramètre dans la description de l'interface, le saisir sur le symbole bleu⁴.

W	Siemens - D:\00_DATA\SIEMENS\Unterlagen\08	Ausbildungsunterlage_TIA-Portal_R1503_e\SCE_EN_032-100 FC-Programming\032_200_FB-Programming\03	2_200_FB-Programming _ □ ×
P	roject Edit View Insert Online Options	xols Window Help	Totally Integrated Automation
	Project tree	032 200 FB-Programming + CPU1516F [CPU 1516F-3 PN/DP] + Program blocks + MOTOR AUT	
	Dovisor		
		Ky Ky 및 등, 42 등 등 등 등, 15 + 15 + 15 + 15 + 15 + 15 + 15 + 15	
		MOTOR_AUTO	l l l l l l l l l l l l l l l l l l l
	 032_200_FB-Programming 	Name Data type Default value Retain Accessible f Visible in	Setpoint Comment 9
	Add new device	1 📲 🔻 Input	^ "
	d Devices & networks	2 💶 Automatic_mode_active Bool false Non-retain	Automatic mode activated
Ē	CPU1516F [CPU 1516F-3 PN/DP]	3 🚾 = Start Bool false Non-retain	Pushbutton automatic start = 🙎
	Device configuration	4 📲 Stop Bool false Non-retain 🗹 🗹	Pushbutton automatic stop
	Online & diagnostics	5 💶 🗉 Enable_OK Bool false Non-retain	All enable conditions OK
	 Program blocks 	6 💶 🛚 Safety_shutoff_active Bool false Non-retain 📃 📃	Safety shutoff active e.g. emergency stop oper
	📑 Add new block	7 🚾 🕶 Output	
	📲 Main [OB1]	8 📲 🔹 Conveyor_motor_automatic_mode Bool 📑 false 🛛 Non-ret 💌 🗹 💽	Control of the conveyor motor in automatic m
	MOTOR_AUTO [FB1]	9 🚾 🔻 InOut	
	MOTOR_AUTO_DB1 [DB1]	e dellama	
	Technology objects		· · · · · · · · · · · · · · · · · · ·
	 External source files 	a >=1 1271 → →a1 → →1=1 sa as	
	PLC tags		
	PLC data types	 Block title: Motor control in automatic mode 	
	Watch and force tables	Conveyor motor in automatic mode:	<u> </u>
	Online backups	••••••••••••••••••••••••••••••••••••••	
	🕨 🔄 Traces	• Wetwork 1: Memory automatic_start_stop and control of the conveyor motor in automatic mode	
	Program info	Comment	=
	Device proxy data		
	PLC alarms	??.	
	Text lists		
	Local modules		
	Common data		
	Documentation settings		
	Languages & resources		
	Online access		×
	Card Reader/USB memory	♥ Natural D	100%
	> Details view	<u>a</u> .	Properties 🗓 Info 🔋 📱 Diagnostics 👘 🗏 📥
	Portal view Overview		Resident 032, 200, ER Reserver mine and

(→ ^{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 ^a 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 &

	PN/DP]	Progra	am bloci	ks ▶ N	NOTOR_	AUTO	(FB1) - 7	i X
MOTOR_AUTO		* *	v o ca	~			-	-
Name	Data t	Defaul	Retain	Acce	Visibl	Setp	Comment	
i 📶 🔻 Input								1
🛛 💶 🔹 Automatic_mode_active	Bool	false	Non-r				Automatic mode	
🛛 📹 🔹 Start	Bool	false	Non-r				Pushbutton auto	1
📶 = Stop	Bool	false	Non-r				Pushbutton auto	
🕣 = Enable_OK	Bool 🔳	false	N 🔻			-	All enable conditi	
all Safety_shutoff_active	Bool	false	Non-r				Safety shutoff acti	
🕣 🔻 Output								
al Conveyor_motor_automatic_mode	Bool	false	Non-r				Control of the con	
🛛 📶 🔻 İnOut					n	n		
<							3	
Conveyor motor in automatic mode: Onveyor Motor in automatic mode: Onveyor Motor in automatic_start_stop	p and con	trol of the	conveyo	r motor i	n autom	atic mod	de	
Conveyor motor in automatic mode: Network 1: Memory automatic_start_stop Comment # A A A A A A A A A A A A	p and con Conveyor motor_ utomatic mode	trol of the - -	e conveyo	r motor i	n autom	atic moi	ie .	
Conveyor motor in automatic mode: Network 1: Memory automatic_start_stop Comment # a <77.7> *********************************	p and con Conveyor motor_ utomatic mode =	trol of the - -	e conveyo	r motor i	n autom	atic mor	Je	-

<??.>. (\rightarrow Tenable_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) E SR sur la première entrée de la liaison & ■.

(→ Instructions → Basic instructions (Instructions de base) → Bit logic operations (opérations logiques sur bits) → $\boxed{ESR} \rightarrow \boxed{=}$)

| | | | | | | | | 0 | options | | |
|--|---------------------------------|-------------|---------|----------|----------|---------|--------------------|------|--|-------------------------------|---|
| a.x == = 🛼 🖿 🚍 📖 🖘 🤊 | 2 • 🗐 | 04 | 6- MB | Ga 4De | 1_ %_ | 0.0 | o 113 Ed | ſ | ML N | it 🗖 | 1 |
| | | e C | 40 Cm | | | 0. | | | Frankter | | - |
| MOTOR_AUTO | | n () | | | 10000 | | le como | Ľ | ravontes | | |
| Name | Data t | Defaul. | Retain | Acce | VISIDI | Setp | Comment | 1 | Basic instructions | | |
| Conveyor_motor_automatic_mode | BOOI | talse | Non-r | | | | Control of the con | ^ N | ame | Descrip | |
| InOut | | | | | | | | | 🛅 General | | ^ |
| o < <ad style="text-align: center;"></ad> | | | | | | | | - | Bit logic operations | | |
| Static | | 1 * * | | | - | | | | E & | AND Io | |
| 2 au Memory_automatic_start_stop | BOOI I | laise | N | | | | Memory used for s | ~ | E >=1 | OR logi | |
| < | | 1111 | | | | | > | | E × | EXCLUS | = |
| | | | | | | | | | | Assign | |
| | | | | | | | | | | Negate | |
| Block title: Motor control in automatic mode | | | | | | | | | [] -[R] | Reset o | |
| Conveyor motor in automatic mode: | | | | | | | | - 11 | 🗉 -[S] | Set out | |
| | | | | | | | | | E SET_BF | Set bit f | |
| Network 1: Memory automatic_start_stor | and con | trol of the | conveyo | or motor | in autom | atic mo | de | | RESET_BF | Reset bi | |
| Commant | | | | | | | | | 🗉 SR | Set/rese | |
| comment | | | | | | | | - | E RS | Reset/se | |
| | | | | | | | | | E - P - | Scan o | |
| | Conveyor | - | | | | | | | E - N - | Scan o | |
| = | | | | | | | | | E -[P]- | Set ope | |
| - | motor_ | | | | | | | | Contra Co | Contraction in the second | |
| | utomatic
mode | - | | | | | | | 囯 -[N] | Set ope | |
| 8 | utomatic
mode | - | | | | | | | □ -[N] □ P_TRIG | Set ope
Scan RL | |
| | motor_
utomatic
mode
= | - | | | | | | | -[N]- P_TRIG N_TRIG | Set ope
Scan RL
Scan RL | |

 → 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. (\rightarrow ^{\sim} 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.

 $(\rightarrow$ SR-Flipflop (bascule SR) \rightarrow <??.> \rightarrow Start (démarrage) \rightarrow #Start (démarrage) \rightarrow Entrée)

1	NON	OR	e_AU	то				Data	t Defaul.	Retain	Acce	Visibl	Setp	Comment	
4	•		Cor	iveyor	_moto	r_auto	matic_mo	de Bool	false	Non-r				Control of	the con
-	•	1	nOut												
)			<ad< td=""><td>ld nev</td><td><></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></ad<>	ld nev	<>										
	•	S	tatic		-			Deel	m eus	N					
		2	Mer	nory_	autom	atic_s	tart_stop	BOOI	iaise	N				imemory us	ed for s.
1	-	_	_	1		1									
B Co	>= llock	tit	tle: I	⊢ Motor in au	–∘I contro tomati	⊔ Iinau cmod	-[=] tomatic m e:	ode							
E Co	>= llock nvey Ne	tit or i	motor	⊣ Motor in au : Me	–∘I contro tomati	⊢ I in au c mod autom	-[=] tomatic m e: atic_start_	ode stop and o	control of th	e conveyo	r motor i	in autom	atic mo	de	
B Co	>= lock nvey Ne Cor	1 or i tw	tle: 1 motor ork 1	⊢ Motor in au : Me	–∘I contro tomati	➡ I in au c mod autom	-[=] tomatic m e: atic_start_	ode stop and c	control of th	e conveyo	r motor i	in autom	atic mo	de	
B Co	>= llock nvey Ne Cor	tii tii tw	motor work 1	⊢ Motor in au : Me	–ol contro tomati mory a au s	→ I in au c mod automa Memo utoma tart_st	-[=] tomatic m e: atic_start_ ry tic op	ode stop and d	control of th	e conveyo	r motor i	in autom	atic mo	de	
B Co	>= llock nvey Ne Cor	tin or n	Tie: 1 motor rork 1	⊢ In au : Me	-ol contro tomati mory a # au s	Lin au c mod automa Memo utoma tart_st SR	-[+] tomatic m e: atic_start_ ry tic op	ode stop and d	control of th	e conveyo #Convey motor_ automat	r motor i or_	in autom	atíc mo	de	

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.



 \rightarrow 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 ³ du circuit OU pour ajouter une variable d'entrée supplémentaire.

	>=1
?.?	
?.?	<u> </u>

→ 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



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

ю	ă I	3	1	9 <u>8</u> 9			3 🗃	<u>9</u> 2	1± 22 ± 🖃	😰 😥	60 (11)	₩ \$	1 <u>=</u> %	e ² °	20 D	E	4
	N	10	тс	R_AL	ло												
	T		Na	me					Data t	Defaul.	Retain	Acce	Visibl	Setp	Comment		
	-	1	•	Input													^
	-	0		AL	itomat	ic_mod	de_acti	ve	Bool	false	Non-r				Automatic mode ad	tivated	=
	-	0		St	art				Bool	false	Non-r				Pushbutton automa	atic start	-
	-	1		St	ор				Bool	false	Non-r				Pushbutton automa	atic stop	
	-	1		Er	able_0	ЭК			Bool	false	Non-r				All enable condition	ns OK	
2	-	1		Sa	fety_sl	hutoff_	active		Bool	false	Non-r				Safety shutoff activ	e e.g. e	
7	-		•	Outpu	Jt												-
	r	71	-	-			14			e 1					e · 1 / 1		
	10		-														(I)
•		N	let	work	1: Me	emory	automa	atic_sta	rt_stop and co	ntrol of th	e conveyo	r motor	in autom	natic mo	de		^
			#Sa	fety_s #Auto mode_	#Stop hutoff_ active matic_ active	-	>=1		ŝ	#Start —	#Memo automa start_st SR SR	ry_ tic_ op Q —	_	&	#Conveyor_ motor_ automatic_ mode		

→ 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))

MOTOR_AUTO [F81]	Properties	🗓 Info 🔋 🕑 Diagnostics	┛╘▼
General				
General	Comment			
Information	General			
Time stamps				
Compilation	Name:	MOTOR_AUTO		
Protection	Tune	EB		
Attributes	, ijpe.			
Download	Language:	LAD		<u> </u>
	Number:	LAD		
	• -	manual		
		automatic		
		0		

→ 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)".

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



 \rightarrow 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"

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

03	2-2	00_	FB-Pr	ogran	nminę	g ▶ (CPU_	1516F [CPU	1516F	-3 PN/DP] 🕨 I	Program blocks → Main [OB1]	_ # # X
н	я н Ма	oğ ∃ ain	9 ∉9	80	E	3 6	9	3±2±[= 😰	୯° ६ ₀ ৫≣ 9		3
	1	Nan	ne					Data type		Default value	Comment	
1		•	Input									^
2			Ini	tial_Ca				Bool			Initial call of this OB	=
З	-		Re	maner	nce			Bool			=True, if remanent data are available	
4	-	•	Temp									
5			<a< td=""><td>dd nev</td><td>V></td><td></td><td></td><td></td><td></td><td></td><td></td><td>~</td></a<>	dd nev	V>							~
8		>=1	177	-	-01	↦	-[=]	l				
•	Blo	ock t	itle:	"Main	Progra	m Swe	eep (C	Cycle)*				
- 2	Com	men	t									
•	1	Netv	work '	1: Co	ntrol c	onvey	or mo	tor forwards in	automa	atic mode		
	(Comn	nent									

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

Project tree	◀ 032-200_FB-Programming CP				_ # = ×	Instructions	
Devices						Options	
300	🗄 🖂 🥔 👻 🕾 🔚 🚍	🗩 🗶 ± 😹 ± 🔚 🕼	e 60 60 60 6	a •			
	Main					✓ Favorites	
032-200_FB-Programming	Name	Data type	Default value	Comment			0.024
Add new device	1 📲 👻 Input		_		^	· · · · · · · · · · · ·	-01
h Devices & networks	2 🔄 🔹 Initial_Call	Bool		Initial call of this OB	1	→ -[=]	
Device configuration	Remanence	8001		= irue, if remanent data are available			
V. Online & diagnostics	5 Add news				~		
🕶 🙀 Program blocks			interest interest	91	- Anno		
Add new block	▲ >=1 1??? -1 -01 -> -						
Hain [OB1]	▼ Block title: *Main Program Swee	p (Cycle)*					
MOTOR_AUTO [FB1]	Comment						
External source files	 Network 1: Control conveyor 	motor forwards in autom	atic mode				
PLC tags	Company		in the mode				
C PLC data types	Comment						
Watch and force tables	1				-		
Online backups	- MO	OR_AUTO [FB1]					
Image Traces						✓ Basic instructions	_
Device provideta	~					Name	-
						🕨 🛅 General	
Details view						Bit logic operations	
						G Timer operations	
Name Address						< m	>
						> Extended instruction	ons
						> Technology	
				100%	·	> Communication	
						and a second	

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

-	Data block		
Single nstance	Name Number The called fur data block.	MOTOR_AUTO_DB	s own instance

→ 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 (\rightarrow Tag table_sorting station (table des variables Installation de tri \rightarrow Details view (vue de détail) \rightarrow -S0 \rightarrow 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)

> = 1	??	٦	-ol	ч	-[=]					
Comn	nent									
							MOT	%DB1 OR_AUTO_ DB		
							TMOT	%FB1 "OR_AUTO"		
							0000000			
							— EN			
						-5				
				&		-si -si		%10.2	mode selector man	-
		<11.1>	_	&		-s -s= *-s0* -s= *-s1*	EN Automatic_ mode_active Bool Bool	%10.2	mode selector man pushbutton autom	-
		<11.7> <11.7>		&		-5 • *-50* • *-51*	EN Automatic_ mode_active Bool Bool Bool	%10.2 %10.3 %10.4	mode selector man pushbutton autom pushbutton autom	×
		<11.7> <11.7>		&	_	-s -50" -51" -60" -52" -60" -53"	Automatic_ Automatic_ mode_active Bool Bool Bool Bool Bool	%10.2 %10.3 %10.4 %11.4	mode selector man pushbutton autom pushbutton autom pushbutton manual	< III
		<11.7> <11.7>	42	&		-S -S -S -S -S -S -S -S -S -S	Automatic_ Automatic_ mode_active Bool Bool Bool Bool Bool Bool	%10.2 %10.3 %10.4 %11.4 %11.5	mode selector man pushbutton autom pushbutton autom pushbutton manual pushbutton manual	< III

→ 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



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

Name		Data type	Default value	Comment	
🗉 🔻 Input					
Initial_C	all	Bool		Initial call of this OB	
¢					
12					
Network 1: C	ontrol conveyor mo	tor forwards in autom	atic mode		
Comment					
		MOT	TOR_AUTO_ DB WFB1		
	90 	*MOT *MOT EN S0* Automatic_ S0* mode_active 0.3 S1* Start	AUBI OR_AUTO_ DB" "VFB1 COR_AUTO" EN: Conveyor moto automatic mod	0	
"40.4 150"	90 *4 *4	*MOT *MOT EN 0.2 Automatic_ S0* — mode_active 0.3 S1* — Start	SUB1 OR_AUTO_ DB* SFB1 OR_AUTO* Conveyor motor automatic mod	0 r%Q0.0 re*-Q1*	
	90 90 	*MOT MOT EN 0.2 Automatic_ s0* mode_active 0.3 S1* Start Stop	SUB1 OR_AUTO_ DB" SFB1 TOR_AUTO" EN Conveyor motor automatic mod	0 r =%Q0.0 le —↓*-Q1*	
*40.4 *52* 	94 94 	*MOT MOT EN 50° — Automatic_ 50° — mode_active 0.3 51° — Start Stop	SUB 1 OR_AUTO_ DB* SOR_AUTO* Conveyor motor automatic mod	0 	
	%4 ** ** **	*MOT *MOT *MOT *MOT *MOT * S0* — Automatic_ s0* — mode_active * * * * * * * * * * * * *	AUBI OR_AUTO_ DB" "VFB1 OR_AUTO" Conveyor motor automatic mod	0 r%Q0.0 le*-Q1*	

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.

		Q Properties	🗓 Info 🤢 🗓 Diagnosti	cs 🗍 🗆 🗸
0	eneral 🚺 Cross-references	Compile Syntax		
٢	🚹 📵 Show all messages			
Co	ompiling completed (errors: 0; warnir	igs: 0)		
1	Path	Description	Go to	? Errors
0	▼ CPU_1516F		~	0
0	 Program blocks 		7	0
0	MOTOR_AUTO (FB1)	Block was successfully compiled.	~	
0	MOTOR_AUTO_DB (DB1)	Block was successfully compiled.	~	
Ø	Main (OB1)	Block was successfully compiled.	7	
O	1	Compiling completed (errors: 0; warning	s:0)	
1000				

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. (\rightarrow Main [OB1] \rightarrow) ..0_FB-Programming > CPU_1516F [CPU 1516F-3 PN/DP] > Program blocks > Main [OB1] TX 👸 🖓 후 한 💺 🖽 🗮 💬 웹 ± 월 ± 달 🕼 🧐 행 행 🗣 🕍 🖌 🛞 🔢 -Monitoring on/off >=1 ??? -[=] 8 - - - 01 -> ~ Network 1: Control conveyor motor forwards in automatic mode Comment %DB1 "MOTOR_AUTO_ DB" FB "MOTOR_AUTO" EN %10.2 Automatic "-SO" mode_active 8 %0 3 "-S1" Start %10.1 "-KO" %10.4 Conveyor "-S2" -%10.5 Stop motor_ "-B1" -Enable_OK %Q0.0 automatic mode - "-Q1" 40 0 Safety "-A1 shutoff_active ENC 👸 🕉 👻 🐁 📰 🚍 💬 웹 ± 월 ± 달 😥 🧐 😘 웬 행 🌞 🖕 🍾 🔗 🕎 🔢 >=1 [??] - -ol - -[=] 8 Network 1: Control conveyor motor forwards in automatic mode Comment %D81 "MOTOR_AUTO_ DB" FB "MOTOR_AUTO" EN TRUE 40.2 Automatic "-SO" mode_active FALSE TRUE "-S1" Start -ко TRUE %10.4 TRUE "-S2" Stop Conveyor "-B1 Enable_OK TRUE TRUE automatic Q0.0 Safety_ shutoff_active mode -01 %10.0 -A1 ENO

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)

IVIC	TOR_AUTO_DB [DB1]		
Call	environment		
	Dependency structure	I Address	Details
1	Auto_DB*)	OB1	Main NW1 (Co
		Transfer to "adjuste	ed manually"
Man	ally adjusted call environment	Transfer to *adjuste	ed manually"
Man	ually adjusted call environment	Transfer to "adjuste	ed manually*
Man	ually adjusted call environment	Transfer to [*] adjuste	ed manually"
Man	ually adjusted call environment	Transfer to *adjuste	ed manually"

Calla	TOR_AUTO_DB [DB1]			
	Dependency structure	1	Address	Details
1	- Main ("MOTOR_AUTO_DB")		OB1	Main NW1 (Co
		Trans	fer to "adjust	ed manually"
		Trans	sfer to "adjust	ed manually"

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))

M Siemens - D:\00_DATA\SIEMENS\Unterlagen\	08_Ausbildungsunterl	age_TIA-Portal_R1503_e\SCE_EN_	032-100 FC-Programming\032_200_FB-Programming	g\032_200_FB-Programming 🗆
Project Edit View Insert Online Options	Tools Window Hel	p		Totally Integrated Automation
1 New) 1	: (* ± 🖥 🛄 🗓 🛢	🛛 📮 💋 Go online 🖉 Go offline		PORTAL
Migrate project	032 200 FB-Prog	ramming > CPU1516F [CPU 1	IS16F-3 PN/DP] • Program blocks • Main [OB	n] _ = = X
Close Ctrl+W				
Save Ctrl+5	x		AG C_ AH CH 475 L_ 3_ 0. 995 HH	
Save as Ctrl+Shift+S	КЯ КЯ = " = "	;=⊟ ≡ ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		
Delete project Ctrl+E			BIOCKITTEPTACE	
Archive	& >=1 ??	I −oI 🛏 -[=] SR RS		
Retrieve				A
Tard Reader/USB memory	 Network 1: 	Control conveyor motor forwards in au	tomatic mode	
The Memory card file	Comment			
Upgrade				
Print Ctrl+P			%	DB1
Print preview			"MOTOR_	AUTO_DB1"
D:100_DATA11032_200_FB-Programming			%	FB1
D:00_TIA032-100_FC-Programmierung			"МОТО	DR_AUTO"
D:l00_TIA_Pol032_100_FC-Programming			<u>—</u> EN	-
D:\\Abschlusspruefung_Teil1_Mechatr			%10.2 Automatic mode	L. L
Exit			"-S0" — active	
Watch and force tables		&	%10.3	
Online backups		%IO.1	"-S1" — Start	
Program info		"-ко" —	%10.4	
Device proxy data		%10.5	"-S2" — Stop	
PLC alarms		"-B1" — +	Enable_OK	
Text lists			%IO.0 Safety shutoff	
Local modules			"-A1" - active	
Common data			%1.0	
Languages & resources			"-B4" — Sensor_slide	Conveyor motor %00.0
Image: Second			%1.3 Sensor end of	automatic_mode — "-Q1"
Card Reader/USB memory	<			> 100%
> Details view				🖻 Properties 🔄 🗓 Info 🗓 🖫 Diagnostics 👘 💷 📥
Portal view Overview	- Main			Project 032_200_FB-Programming ope

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) \rightarrow -Q1 = 0	
5	Activer l'arrêt d'urgence $(-A1 = 0) \rightarrow -Q1 = 0$	
6	Mode manuel (-S0 = 0) \rightarrow -Q1 = 0	
7	Éteindre l'installation (-K0 = 0) \rightarrow -Q1 = 0	
8	Vérin non rentré (-B1 = 0) \rightarrow -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

N٥	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) \rightarrow -Q1 = 0	
5	Actionner brièvement le bouton arrêt automatique (-S2 = 0) \rightarrow -Q1 = 0	
6	Activer l'arrêt d'urgence $(-A1 = 0) \rightarrow -Q1 = 0$	
7	Mode manuel (-S0 = 0) \rightarrow -Q1 = 0	
8	Éteindre l'installation (-K0 = 0) \rightarrow -Q1 = 0	
9	Vérin non rentré (-B1 = 0) \rightarrow -Q1 = 0	
10	Le projet a été archivé avec succès	

10Informations 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