

# SIEMENS

## SIMATIC

### Mise en route STEP 7

Getting Started


A la découverte de STEP 7	1
SIMATIC Manager	2
Programmation symbolique	3
Création d'un programme dans l'OB1	4
Création d'un programme avec FB et DB	5
Configuration des unités centrales	6
Chargement et test du programme	7
Programmation d'une fonction (FC)	8
Programmation d'un bloc de données global	9
Programmation d'un bloc multiinstance	10
Configuration de la périphérie décentralisée	11
Annexe A	A


Ce manuel est livré avec la documentation référencée :  
**6ES7810-4CA10-8CW0**


## Mentions légales

### Signalétique d'avertissement

Ce manuel donne des consignes que vous devez respecter pour votre propre sécurité et pour éviter des dommages matériels. Les avertissements servant à votre sécurité personnelle sont accompagnés d'un triangle de danger, les avertissements concernant uniquement des dommages matériels sont dépourvus de ce triangle. Les avertissements sont représentés ci-après par ordre décroissant de niveau de risque.

 <b>DANGER</b>
signifie que la non-application des mesures de sécurité appropriées <b>entraîne</b> la mort ou des blessures graves.

 <b>ATTENTION</b>
signifie que la non-application des mesures de sécurité appropriées <b>peut entraîner</b> la mort ou des blessures graves.

 <b>PRUDENCE</b>
accompagné d'un triangle de danger, signifie que la non-application des mesures de sécurité appropriées peut entraîner des blessures légères.

<b>PRUDENCE</b>
non accompagné d'un triangle de danger, signifie que la non-application des mesures de sécurité appropriées peut entraîner un dommage matériel.

<b>IMPORTANT</b>
signifie que le non-respect de l'avertissement correspondant peut entraîner l'apparition d'un événement ou d'un état indésirable.


En présence de plusieurs niveaux de risque, c'est toujours l'avertissement correspondant au niveau le plus élevé qui est reproduit. Si un avertissement avec triangle de danger prévient des risques de dommages corporels, le même avertissement peut aussi contenir un avis de mise en garde contre des dommages matériels.

### Personnes qualifiées

L'appareil/le système décrit dans cette documentation ne doit être manipulé que par du **personnel qualifié** pour chaque tâche spécifique. La documentation relative à cette tâche doit être observée, en particulier les consignes de sécurité et avertissements. Les personnes qualifiées sont, en raison de leur formation et de leur expérience, en mesure de reconnaître les risques liés au maniement de ce produit / système et de les éviter.

### Utilisation des produits Siemens conforme à leur destination

Tenez compte des points suivants:

 <b>ATTENTION</b>
Les produits Siemens ne doivent être utilisés que pour les cas d'application prévus dans le catalogue et dans la documentation technique correspondante. S'ils sont utilisés en liaison avec des produits et composants d'autres marques, ceux-ci doivent être recommandés ou agréés par Siemens. Le fonctionnement correct et sûr des produits suppose un transport, un entreposage, une mise en place, un montage, une mise en service, une utilisation et une maintenance dans les règles de l'art. Il faut respecter les conditions d'environnement admissibles ainsi que les indications dans les documentations afférentes.

### Marques de fabrique

Toutes les désignations repérées par ® sont des marques déposées de Siemens AG. Les autres désignations dans ce document peuvent être des marques dont l'utilisation par des tiers à leurs propres fins peut enfreindre les droits de leurs propriétaires respectifs.

### Exclusion de responsabilité

Nous avons vérifié la conformité du contenu du présent document avec le matériel et le logiciel qui y sont décrits. Ne pouvant toutefois exclure toute divergence, nous ne pouvons pas nous porter garants de la conformité intégrale. Si l'usage de ce manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition.

# Bienvenue dans STEP 7...

... le logiciel SIMATIC de base pour la conception de programmes pour systèmes d'automatisation SIMATIC S7-300/400 dans les langages de programmation CONT, LOG ou LIST.

## Quelques informations sur ce Getting Started

Vous apprenez dans ce livre les principes de SIMATIC STEP 7. Nous vous montrons à l'aide d'exercices pratiques les boîtes de dialogue et les techniques de programmation centrales. Ce manuel a été conçu de sorte que vous pouvez le prendre en cours et le commencer pour ainsi dire à chaque chapitre.

Vous trouvez dans chaque sous-chapitre une partie explicative repérée par une bande grise et une partie programmation repérée en vert. La séquence de programmation commence toujours par une flèche dans la marge verte gauche et peut se poursuivre sur plusieurs pages avant de se terminer par un point suivi d'un complément d'information.

Une expérience de Windows (maniement de la souris, technique multifenêtres ou utilisation de menus déroulants etc.) et des connaissances dans le domaine de l'automatisation sont utiles.

Vous avez la possibilité d'approfondir les connaissances acquises dans ce Getting Started au cours de stages de formation à STEP 7, dans lesquels vous apprenez à concevoir et à élaborer une solution d'automatisation dans toutes ses phases.

## Environnement requis pour travailler avec Getting Started

Pour réaliser les exercices pratiques sur STEP 7 présentés dans ce Getting Started vous avez besoin

- d'une console de programmation Siemens ou d'un PC,
- du logiciel de base STEP 7 et de la "License Key" correspondante et
- d'un système d'automatisation SIMATIC S7-300 ou S7-400 (pour le chapitre 7 "Charger et tester le programme")

## Autre documentation de STEP 7

- STEP 7 Connaissances fondamentales
- STEP 7 Manuels de référence

Après l'installation de STEP 7, vous trouvez les manuels électroniques dans le menu de démarrage sous **SIMATIC > Documentation**. Vous avez également la possibilité de les commander dans n'importe quelle filiale Siemens. Toutes les informations contenues dans les manuels de STEP 7 peuvent également être appelées dans l'aide en ligne.

Nous vous souhaitons un parcours agréable avec Getting Started !

Votre SIEMENS AG



# Sommaire

<b>1</b>	<b>A la découverte de STEP 7</b>	
1.1	Qu'apprendrez-vous dans ce manuel ?	7
1.2	Interaction du logiciel et du matériel	9
1.3	STEP 7 : Mode d'emploi	10
1.4	Installation de STEP 7	11
<b>2</b>	<b>SIMATIC Manager</b>	
2.1	Lancer SIMATIC Manager et créer un projet	13
2.2	Structure du projet dans SIMATIC Manager et appel de l'aide de STEP 7	16
<b>3</b>	<b>Programmation symbolique</b>	
3.1	Adresse absolue	19
3.2	Programmation symbolique	20
<b>4</b>	<b>Création d'un programme dans l'OB1</b>	
4.1	Ouvrir l'éditeur de programme dans la vue CONT, LIST ou LOG et l'ouvrir dans l'OB1	23
4.2	Programmation de l'OB1 en CONT	26
4.3	Programmation de l'OB1 en LIST	30
4.4	Programmation de l'OB1 en LOG	33
<b>5</b>	<b>Création d'un programme avec FB et DB</b>	
5.1	Créer et ouvrir un bloc fonctionnel	37
5.2	Programmation du bloc FB1 en CONT	39
5.3	Programmation du bloc FB1 en LIST	43
5.4	Programmation du bloc FB1 en LOG	46
5.5	Générer les blocs de données d'instance et modifier les valeurs effectives	50
5.6	Programmation d'un appel de bloc en CONT	52
5.7	Programmation d'un appel de bloc en LIST	55
5.8	Programmation d'un appel de bloc en LOG	57

Dans les chapitres 3 à 5, vous créez un programme simple.



# 1 A la découverte de STEP 7

## 1.1 Qu'apprendrez-vous dans ce manuel ?

Nous voulons vous montrer à l'aide d'exercices pratiques comme il est simple de programmer en CONT, LOG et LIST avec STEP 7.

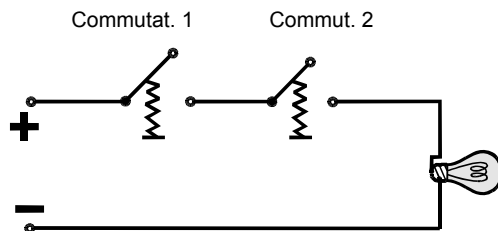
Vous apprendrez à utiliser les différentes applications de STEP 7 au cours des onze leçons suivantes.

### Création d'un programme à l'aide de fonctions binaires

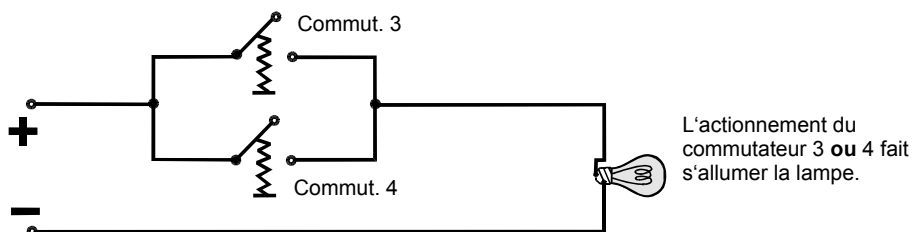
Dans les chapitres 2 à 7, vous créez un programme à l'aide de fonctions binaires permettant l'adressage des entrées et sorties de votre CPU si vous en avez une.

Les programmes-exemples de "Getting Started" utilisent pour l'essentiel trois fonctions binaires de base.

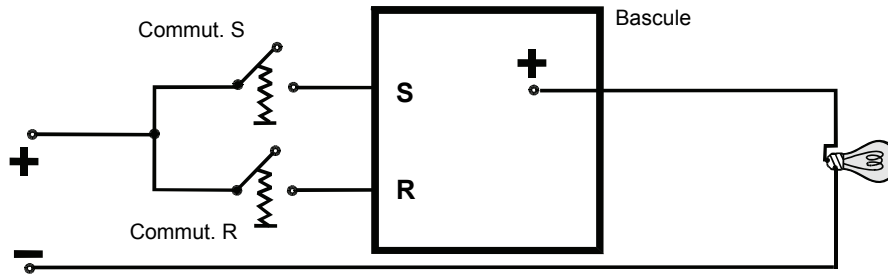
La fonction binaire que vous aurez en premier à programmer est la fonction ET. Cette dernière peut être représentée par un circuit électrique à deux commutateurs.



La seconde fonction binaire que nous serons amené à programmer est la fonction OU. On peut également la représenter par un circuit électrique.



La troisième fonction qui nous occupera est la bascule (fonction SR). Celle-ci réagit dans un circuit électrique à certains états de tension et a pour fonction de les transmettre à d'autres éléments du circuit.



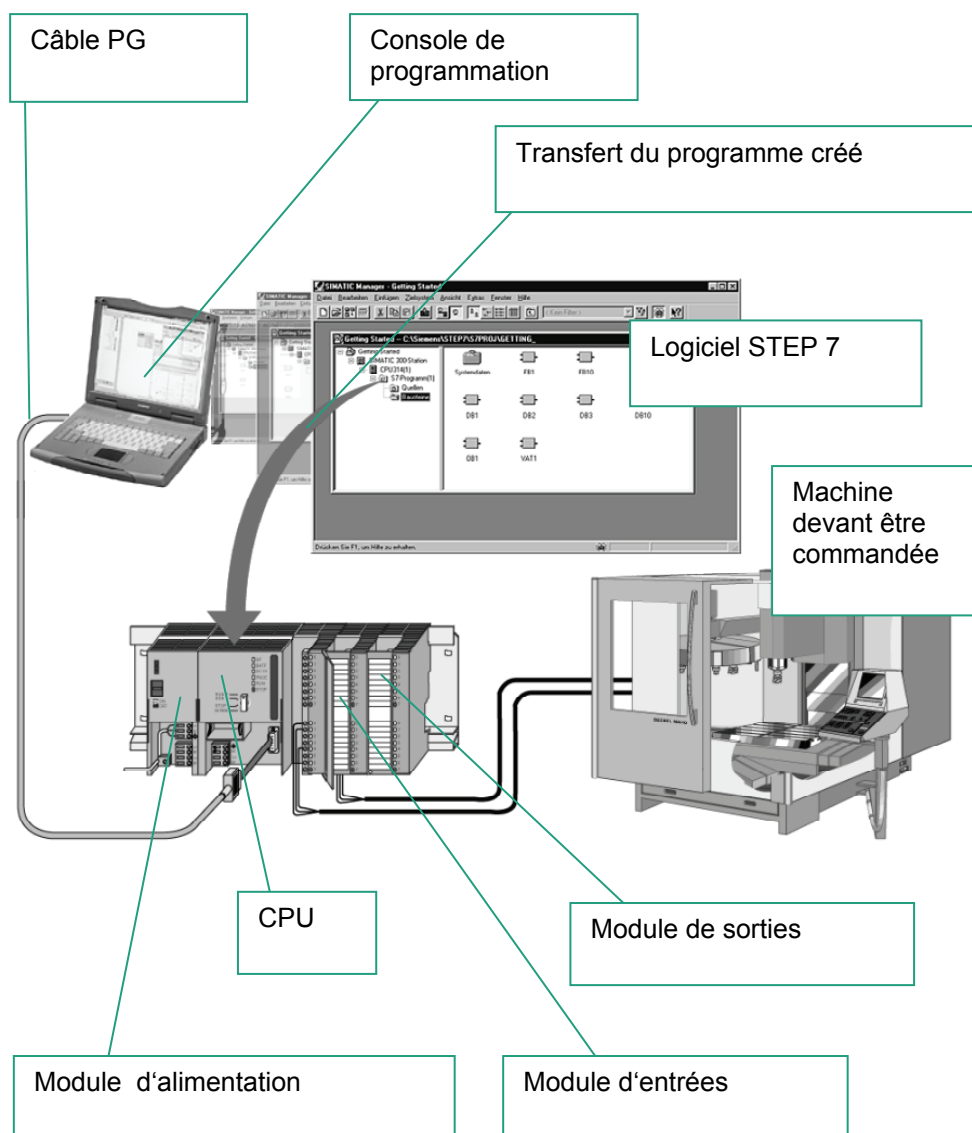
L'actionnement du commutateur S fait s'allumer la lampe qui reste allumée jusqu'à l'action du commutateur R.



## 1.2 Interaction du logiciel et du matériel

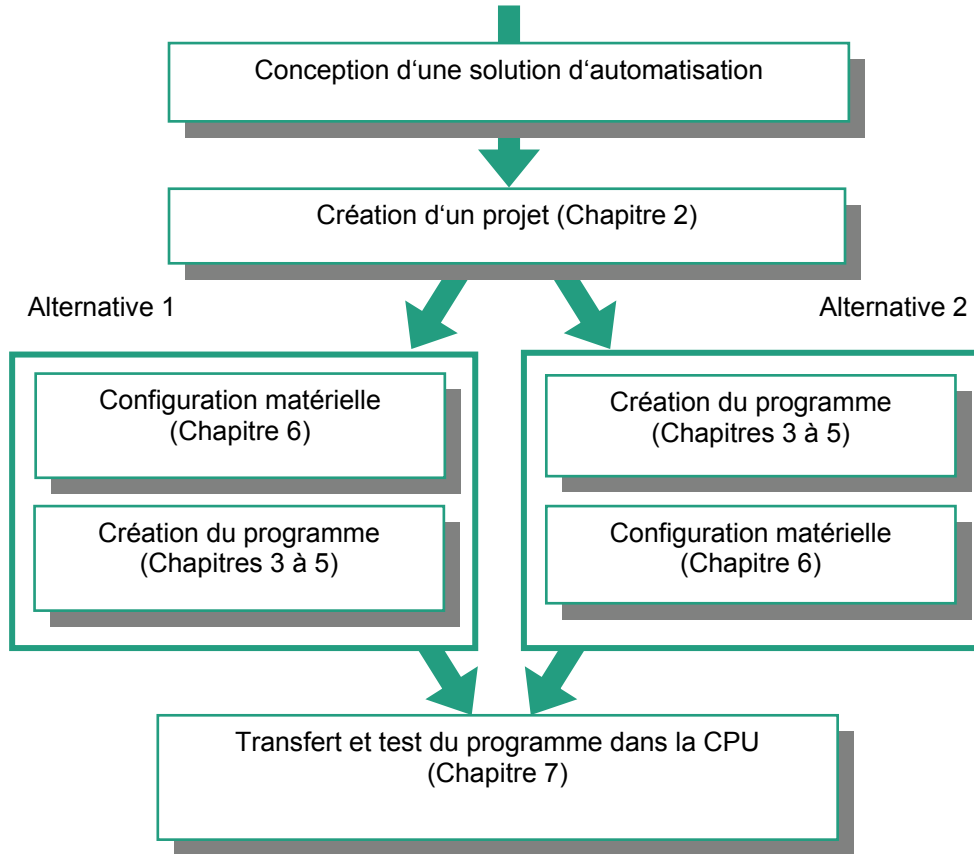
Vous créez à l'aide du logiciel STEP 7 votre programme S7 dans un projet. L'automate S7 est constitué d'un module d'alimentation, d'une CPU et de modules d'entrées ou de sorties (modules d'E/S).

L'automate programmable (AP) contrôle et commande à l'aide du programme S7 votre machine. L'adressage des modules d'E/S se fait par l'intermédiaire des adresses du programme S7.



### 1.3 STEP 7 : Mode d'emploi

Avant de créer votre projet, sachez que différentes approches sont possibles. En effet, vous êtes libre dans STEP 7 de procéder dans l'ordre qui vous convient.



Si votre programme contient beaucoup d'entrées et de sorties, nous vous recommandons de commencer par configurer le matériel, l'application de configuration matérielle de STEP 7 présentant l'avantage que les adresses y sont sélectionnées pour vous.

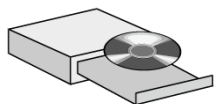
Si vous choisissez la seconde alternative, il vous faudra rechercher vous-même les adresses en fonction des constituants choisis. Vous ne pourrez alors pas bénéficier de la fonction d'adressage automatique de STEP 7.

La configuration matérielle vous permet non seulement de sélectionner les adresses, mais également de modifier les paramètres et les propriétés des modules. Pour la mise en œuvre de plusieurs CPU, il faut par exemple modifier les adresses MPI des CPU.

Comme nous n'avons pas besoin de beaucoup d'entrées et de sorties dans ce "Getting Started", sautons la configuration matérielle et passons directement à la programmation.

## 1.4 Installation de STEP 7

Que vous vouliez commencer par la programmation ou par la configuration matérielle, vous devez tout d'abord installer STEP 7, à moins que vous n'utilisiez une PG SIMATIC sur laquelle STEP 7 est déjà installé.



Pour installer le logiciel STEP 7 sur une PG/PC sans logiciel préinstallé, tenez compte de l'environnement logiciel et matériel requis. Vous trouvez ceux-ci décrits dans le fichier Lisezmoi qui se trouve sur le CD-ROM de STEP 7 sous **<Lecteur>:\STEP 7\Disk1**

Si vous devez d'abord installer STEP7, insérez le CD-ROM de STEP 7 dans le lecteur. Le programme d'installation est automatiquement lancé. Suivez les instructions affichées par celui-ci.

Si le lancement automatique du programme échoue, vous pouvez lancer ce dernier à partir du CD-ROM sous :

**<Lecteur>:\STEP 7\Disk1\setup.exe.**



SIMATIC Manager



Après l'installation et le redémarrage de l'ordinateur, l'icône du "SIMATIC Manager" s'affiche sur votre bureau.

En double-cliquant l'installation une fois achevée sur l'icône "SIMATIC Manager", vous lancez automatiquement l'assistant de STEP 7.

Vous trouverez de plus amples informations sur l'installation du logiciel dans le fichier Lisezmoi.wri qui figure sur le CD de STEP 7 sous **<Lecteur>:\STEP 7\Disk1\Lisezmoi.wri**



## 2 SIMATIC Manager

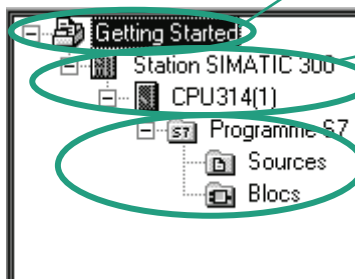
### 2.1 Lancer SIMATIC Manager et créer un projet

Le lancement de STEP 7 fait s'ouvrir le gestionnaire de projets SIMATIC Manager. L'assistant de STEP 7 est par défaut toujours activé. Celui-ci a pour but de vous assister dans la création de votre projet STEP 7. La structure du projet sert à ordonner les données et programmes créés au cours du projet.

Les données sont archivées dans le projet sous la forme d'objets en une structure hiérarchique.

La station SIMATIC et la CPU renferment les données de configuration et de paramétrage du matériel.

Le programme S7 contient tous les blocs des divers programmes qui serviront à commander la machine.

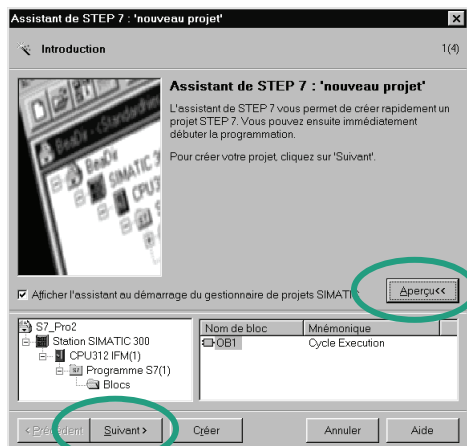


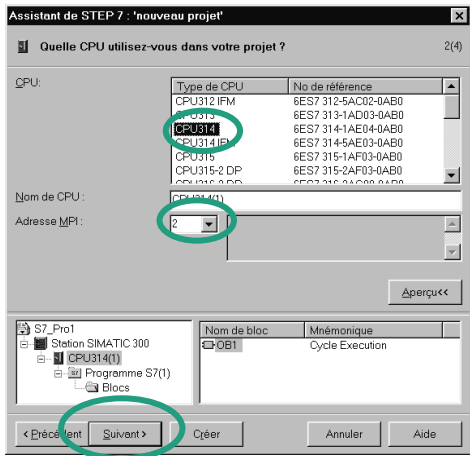
SIMATIC Manager

Double-cliquez sur l'icône **SIMATIC Manager** sur le bureau Windows. Choisissez la commande de menu **Fichier > Assistant "Nouveau projet"**, si l'assistant ne démarre pas automatiquement.

Avec **Aperçu**, vous pouvez afficher ou masquer la structure du projet créé.

Avec **Suivant**, vous passez à la feuille suivante de l'assistant.





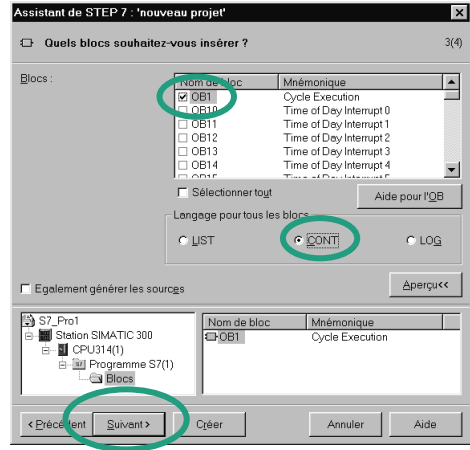
Sélectionnez pour l'exemple de projet de notre "Getting Started" la CPU 314. Cet exemple a été conçu de telle sorte que vous pouvez sélectionner la CPU qui vous a été livrée.

L'adresse MPI est réglée par défaut sur 2.

Confirmez vos sélections et passez au prochain dialogue avec **Suivant**.

Chaque CPU a des caractéristiques, comme la capacité de mémoire ou les plages d'opérandes qui lui sont propres. C'est pourquoi vous devez toujours sélectionner une CPU avant de programmer.

L'adresse MPI (Multi Point Interface) est requise pour la communication entre la CPU et la PG ou le PC.



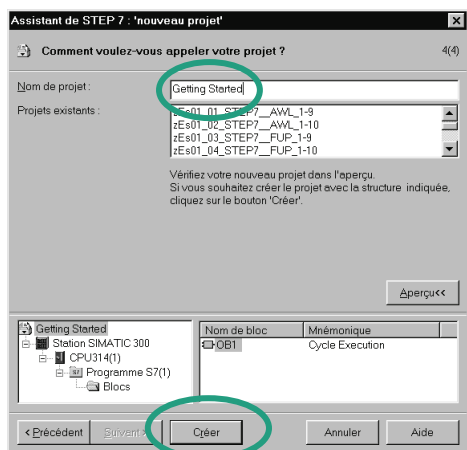
Sélectionnez le bloc d'organisation **OB1** (s'il n'est déjà sélectionné).

Choisissez votre langage de programmation : **CONT**, **LOG** ou **LIST**.

Confirmez vos sélections avec **Suivant**.

L'OB1 se trouve à la tête de la hiérarchie du programme. Tous les autres blocs du programme lui sont subordonnés.

Vous pouvez changer de langage de programmation à tout moment ultérieur.



Sélectionnez en double-cliquant dans la zone de texte "Nom du projet" le nom proposé et entrez à la place de celui-ci "Getting Started".

Si vous cliquez sur **Créer**, votre nouveau projet sera créé selon la structure que vous pouvez voir avec **Aperçu**.

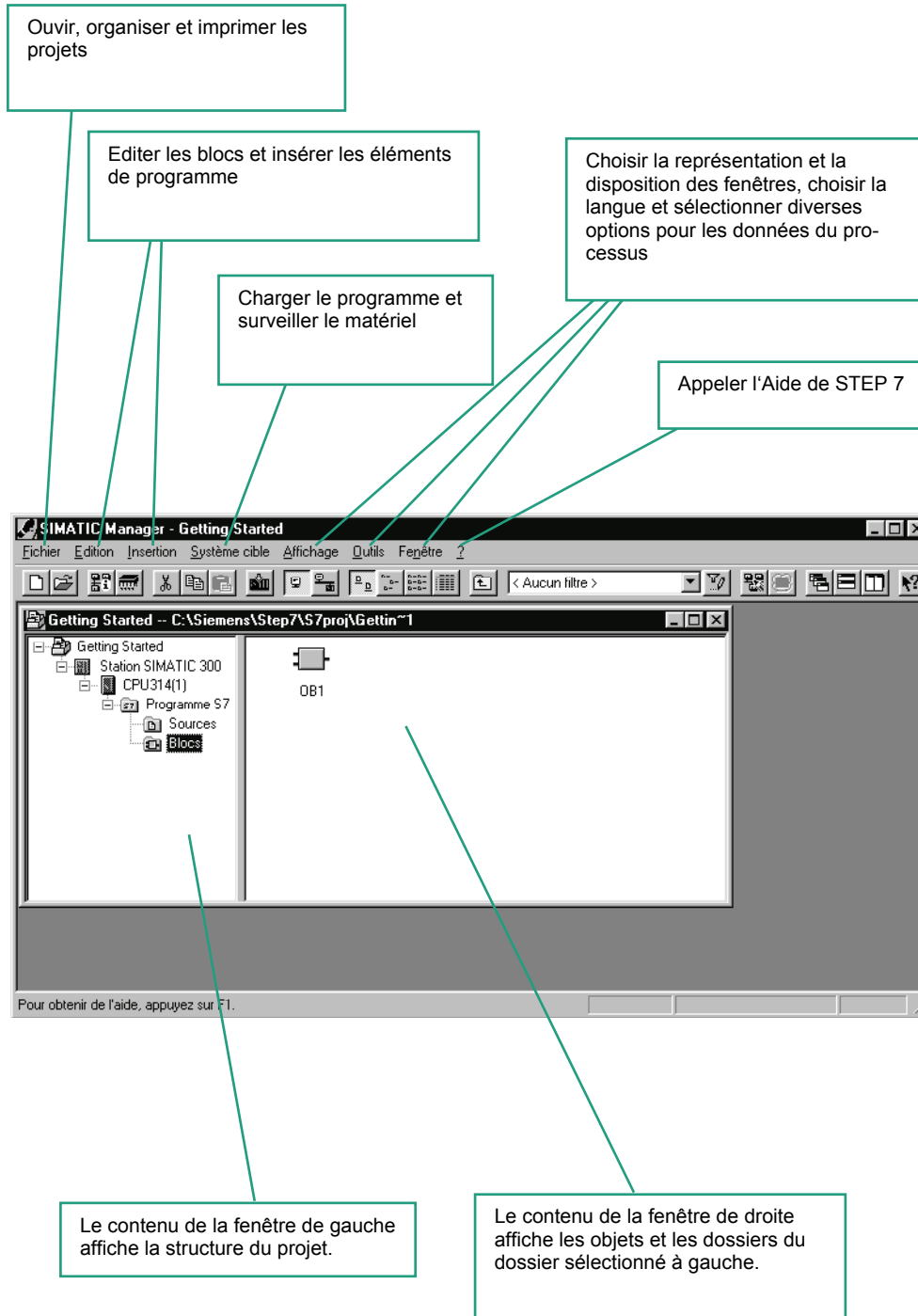
Après l'exécution de la commande **Créer**, SIMATIC Manager s'ouvre avec la fenêtre du projet "Getting Started" nouvellement créé. La signification et la manipulation des fichiers et dossiers créés sera expliquée dans les pages suivantes.

L'assistant de STEP 7 est activé par défaut à chaque nouveau lancement du programme. Si vous voulez le désactiver, vous pouvez le faire dans le premier dialogue de l'assistant. Sachez toutefois qu'il vous faudra créer manuellement chaque dossier du projet que vous créerez sans l'assistant.

Pour plus d'informations, référez-vous à la rubrique d'aide "Création et édition de projets" via la commande de menu ? > **Rubriques d'aide**.

## 2.2 Structure du projet dans SIMATIC Manager et appel de l'aide de STEP 7

Dès que l'Assistant est refermé, SIMATIC Manager apparaît de nouveau avec la fenêtre du projet "Getting Started" qui vient d'être créée ouverte. C'est à partir de cette fenêtre que vous allez appeler toutes les fonctions et les autres fenêtres de STEP 7.



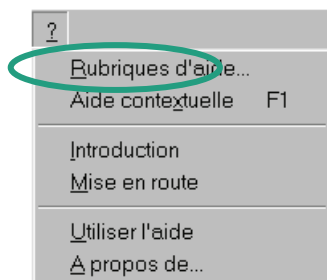


## Appeler l'Aide de STEP 7



## F1 Alternative 1 :

Sélectionnez une commande de menu quelconque et appuyez sur la touche de fonction **F1**. Une aide contextuelle s'affiche alors sur la commande en question.



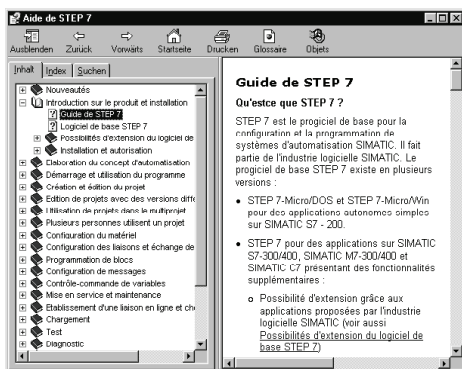
## Alternative 2 :

Cliquez dans la barre des menus sur ? et sélectionnez-y la commande **Rubriques d'aide**. Ceci ouvre le menu de l'Aide de STEP 7.

Dans la partie gauche de la fenêtre est affiché le sommaire avec toutes les rubriques traitées, dans la partie droite la rubrique sélectionnée.

Naviguez dans le sommaire jusqu'à la rubrique désirée en ouvrant éventuellement par un clic sur le signe + les livres pour afficher les rubriques qu'il contient. Quand vous sélectionnez une rubrique, son contenu s'affiche aussitôt dans la partie droite de la fenêtre

Avec **Index** et **Rechercher**, vous pouvez entrer vos critères de recherche afin de cibler la recherche.



## Alternative 3 :

Cliquez dans l'aide de STEP 7 sur l'icône "Page d'accueil".

Un portail d'information s'affiche. Il offre un accès rapide aux thèmes centraux de l'aide en ligne comme par exemple :

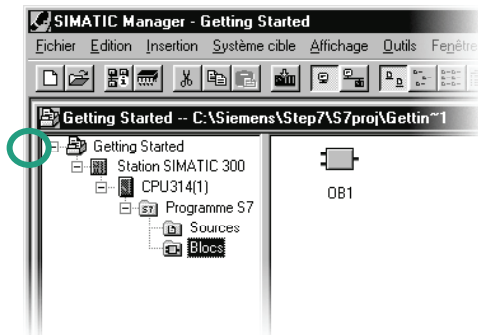
- Mise en route rapide de STEP 7
- Configuration et programmation
- Test et débogage
- SIMATIC sur Internet

## Alternative 4 :

Cliquez sur le curseur d'aide. Le prochain clic sur un objet quelconque affiche l'aide pour cet objet.



## Naviguer dans la structure du projet



La structure du projet nouvellement créé s'affiche avec la station S7 et la CPU sélectionnées.

Cliquez sur le signe + ou – pour ouvrir ou fermer les différents dossiers.

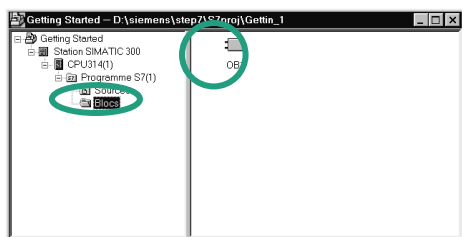
Vous appelez les autres fonctions en cliquant sur les icônes apparaissant dans la partie droite de la fenêtre.



Cliquez sur le dossier **Programme S7 (1)**. Il contient à son tour d'autres constituants du programme.

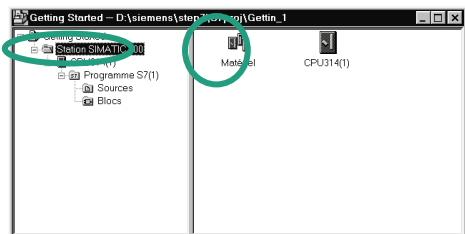
Via **Mnémoniques** vous ouvrez la table des mnémoniques décrite au chapitre 3 dans laquelle vous donnez aux adresses des noms symboliques.

Le dossier Sources sert à archiver vos programmes source. Ces derniers ne sont pas traités dans ce „Getting Started“.



Si vous cliquez sur le dossier **Blocs**, vous voyez l'unique bloc créé jusqu'ici l'**OB1**. Il contiendra tous les autres blocs qui viendront après lui.

Via les blocs vous parvenez à la programmation en CONT, LOG et LIST décrite aux chapitres 4 et 5.



Cliquez sur le dossier **Station SIMATIC 300**. Il contient toutes les données du projet servant au matériel.

Via **Matériel** vous spécifiez les paramètres de votre système d'automatisation comme décrit au chapitre 6.

Les logiciels optionnels servant à l'extension de votre tâche d'automatisation tels PLC-SIM (programme de simulation du matériel) ou S7-GRAPH (langage graphique de programmation) sont intégrés à STEP 7. Vous pouvez alors ouvrir leurs objets, par exemple un bloc fonctionnel S7-GRAPH depuis SIMATIC Manager.

Pour plus d'informations, voir les rubriques d'aide "Elaboration du concept d'automatisation" et "Principes de conception de la structure du programme".

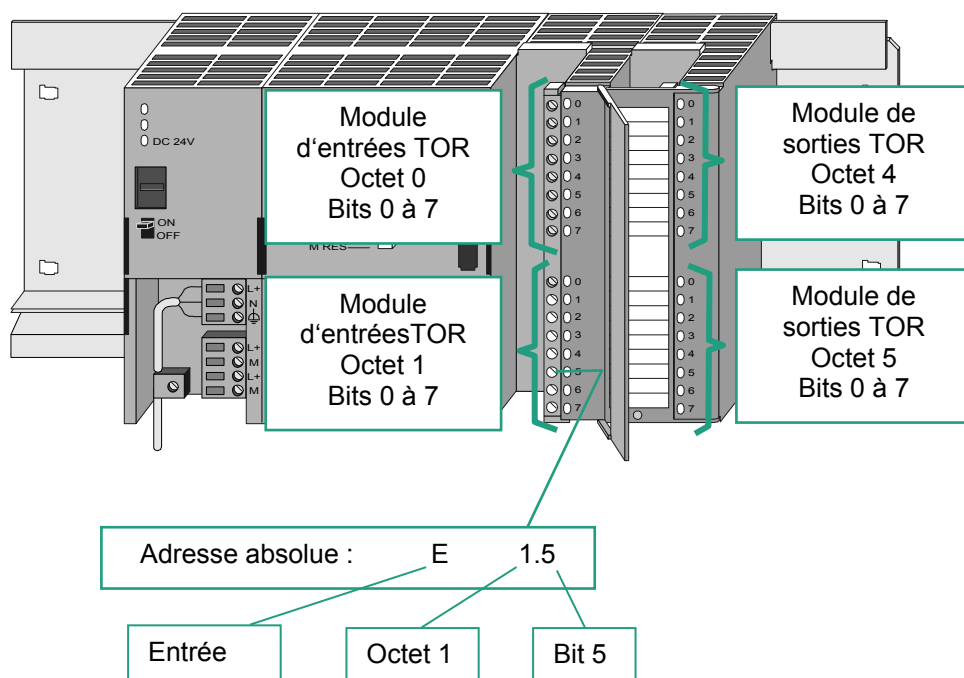
Pour plus d'informations sur les logiciels optionnels, voir le catalogue SIMATIC "Constituants pour l'intégration totale de systèmes automatisés" ST 70.

# 3 Programmation symbolique

## 3.1 Adresse absolue

Chaque entrée et chaque sortie possède par défaut une adresse absolue déterminée par la configuration matérielle. Celle-ci est indiquée de manière directe, c'est-à-dire absolue.

L'adresse absolue peut être remplacée par des noms symboliques pouvant être librement choisis.



N'utilisez la programmation absolue que si le nombre d'entrées et de sorties de votre programme est limité.

### 3.2 Programmation symbolique

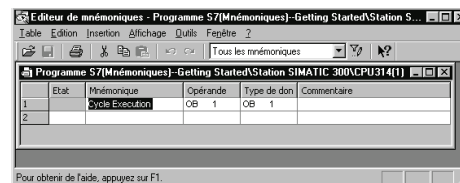
Vous affectez dans la table des mnémoniques un nom symbolique à toutes les adresses absolues que vous voulez appeler dans le programme ainsi que le type de données, par exemple pour l'entrée E0.1 le mnémonique Commutateur 1. Ces noms valent pour toutes les sections du programme. C'est pourquoi on les appelle des variables globales.

La programmation symbolique permet d'alléger l'écriture de votre programme qui y gagne en clarté.

#### Travailler avec l'éditeur de mnémoniques



Pour ouvrir celui-ci, naviguez dans la fenêtre de projet "Getting Started" jusqu'au **Programme S7 (1)** et double-cliquez sur **Mnémoniques**.



La table des mnémoniques ne contient pour l'instant que le bloc d'organisation défini par défaut, l'**OB1**.

	Etat	Mnémonique	Opérande	Type de don
1		Cycle Execution	OB 1	OB 1
2				

Cliquez sur **Cycle Execution** et écrivez à la place de celui-ci "Programme principal".

	Etat	Mnémonique	Opérande	Type de
1		Programme principal	OB 1	OB 1
2		Lampe verte	A 4.0	BOOL

Entrez dans la ligne 2 "Feu vert" et "A 4.0". Le type de données s'inscrit automatiquement dans la colonne du type.

Commentaire				

Cliquez dans la ligne 1 ou 2 sur la colonne du commentaire pour entrer éventuellement un commentaire de mnémonique. L'action de la touche **Entrée** clôt la ligne ou l'enregistrement et insère une nouvelle ligne de mnémonique.

	Etat	Mnémonique	Opérande	Type de
1		Programme principal	OB 1	OB 1
2		Lampe verte	A 4.0	BOOL
3		Lampe rouge	A 4.1	BOOL

Entrez dans la ligne 3 "Feu rouge" et "A 4.1" et confirmez la saisie avec **Entrée**.

Affectez de la même manière un nom symbolique à toutes les entrées et sorties du programme.



Enregistrez vos entrées ou vos modifications de la table des mnémoniques et fermez la fenêtre.

Comme le projet "Getting Started" contient beaucoup de noms, vous pouvez copier la table des mnémoniques dans votre projet comme décrit au chapitre ci-après.

Etat	Mnémonique	Opérande	Type de données	Commentaire
1	Automatique Marche	E 0.5	BOOL	Activation de la bascule
2	Commutateur 1	E 0.1	BOOL	Pour la connexion en série
3	Commutateur 2	E 0.2	BOOL	Pour la connexion en série
4	Commutateur 3	E 0.3	BOOL	Pour la connexion en parallèle
5	Commutateur 4	E 0.4	BOOL	Pour la connexion en parallèle
6	Diesel	DB 2	FB 1	Données du moteur diesel
7	Données_G	DB 3	DB 3	Bloc de données global
8	Essence	DB 1	FB 1	Données du moteur à essence
9	Exécution cyclique	OB 1	OB 1	Bloc contenant le programme utilis
10	Lampe rouge	A 4.1	BOOL	Résultat de l'interrogation OU
11	Lampe verte	A 4.0	BOOL	Résultat de l'interrogation ET
12	Manuel Marche	E 0.6	BOOL	Désactivation de la bascule
13	Marche_MotDies	A 5.4	BOOL	Commande de mise en marche du
14	Marche_MotEss	A 5.0	BOOL	Commande de mise en marche du
15	Mode automatique	A 4.2	BOOL	Bascule
16	MotDies_Arrêt	E 1.5	BOOL	Arrêt du moteur diesel
17	MotDies_Défaillance	E 1.6	BOOL	Défaillance du moteur diesel
18	MotDies_Marche	E 1.4	BOOL	Mise en marche du moteur diesel
19	MotDies_Ventil_activé	A 5.6	BOOL	Commande mise en marche ventil
20	MotDies_VitesseCourante	MW 4	INT	Vitesse réelle du moteur diesel
21	MotDies_Vitesse_atteinte	A 5.5	BOOL	Signalisation vitesse prescrite mo
22	MotEss_arrêt	E 1.1	BOOL	Arrêt du moteur à essence
23	MotEss_Défaillance	E 1.2	BOOL	Défaillance du moteur à essence
24	MotEss_marche	E 1.0	BOOL	Mise en marche du moteur à esse
25	MotEss_Ventil_activé	A 5.2	BOOL	Commande mise en marche ventil
26	MotEss_Vitesse_atteinte	A 5.1	BOOL	Signalisation vitesse prescrite mot
27	MotEss_VitesseCourante	MW 2	INT	Vitesse réelle du moteur à essence
28	Moteur	FB 1	FB 1	Commande de moteur
29	Retard_MotDies	T 2	TIMER	Retardement de l'arrêt du ventilateur du mot...
30	Retard_MotEss	T 1	TIMER	Retardement de l'arrêt du ventilateur du mot...
31	Ventilateur	FC 1	FC 1	Commande de ventilateur
32				

Vous voyez ci-contre la table des mnémoniques de l'exemple de programme S7 "Getting Started" pour LIST. De manière générale, une table des mnémoniques est générée pour chaque programme S7, et quel que soit le langage de programmation choisi. Tous les caractères pouvant être imprimés (lettres accentuées, espaces etc.) sont autorisés dans la table des mnémoniques.



Le type de données inscrit automatiquement dans la table des mnémoniques indique à la CPU le type de signal qu'elle a à traiter. STEP 7 utilise entre autres les types de données suivants :

BOOL BYTE WORD DWORD	Les données ayant ce type autorisent les opérations sur bits de 1 bit (type BOOL) à 32 bits (DWORD).
CHAR	Les données ayant ce type occupent exactement un caractère du jeu de caractères ASCII.
INT DINT REAL	Ces types de données servent au traitement de valeurs numériques (par exemple au calcul d'expressions arithmétiques).
S5TIME TIME DATE TIME_OF_DAY	Formats de temps existants dans STEP 7 pour indiquer une date ou entrer une valeur de temps.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Définir les mnémoniques" via la commande de menu ? > Rubriques d'aide.



## 4 Création d'un programme dans l'OB1

### 4.1 Ouvrir l'éditeur de programme dans la vue CONT, LIST ou LOG et ouvrir l'OB1

#### Choisissez votre langage de programmation : CONT, LIST ou LOG

Pour créer vos programmes S7, vous disposez dans STEP 7 de trois langages de programmation CONT, LIST ou LOG. Dans la pratique et pour ce chapitre, vous devez vous décider pour l'un de ces langages.

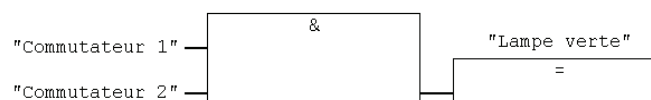
#### CONT (Schéma à CONTACTs) Pour l'habitué des schémas électriques.



#### LIST (LISTE d'instructions) Pour l'informaticien.

```
U "Commutateur 1"  
U "Commutateur 2"  
= "Lampe verte"
```

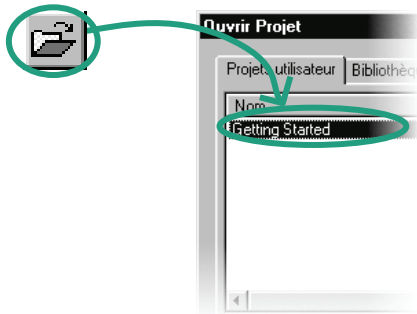
#### LOG (LOGigramme) Pour le spécialiste des circuits ou le programmeur préférant les opérations logiques.



Le bloc OB1 s'ouvre dans la vue du langage choisi lors de sa création avec l'assistant au projet. Vous pouvez toutefois modifier le langage par défaut à tout moment ultérieur.



**Copier la table des mnémoniques et ouvrir l'OB1**

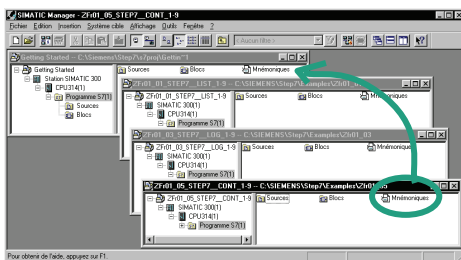


S'il n'est pas encore ouvert, ouvrez votre projet "Getting Started". Cliquez pour cela dans la barre d'outils sur le bouton **Ouvrir**, sélectionnez dans la liste proposée le projet créé "Getting Started" et confirmez avec **OK**.

Sélectionnez dans l'onglet "Projets-exemples" en outre l'un des projets suivants en fonction du langage de programmation choisi :

- ZFr01\_05\_STEP7\_KOP\_1-9 ou
- ZFr01\_01\_STEP7\_AWL\_1-9 ou
- ZFr01\_03\_STEP7\_FUP\_1-9.

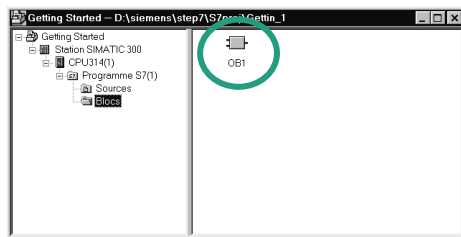
Vous pouvez voir ci-contre ces trois exemples de projet représentés.



Naviguez dans ZFr01\_XXX jusqu'à l'objet **Mnémoniques** et copiez celui-ci par glisser-lâcher dans le dossier **Programme S7** de la fenêtre de votre projet "Getting Started".

Fermez ensuite la fenêtre du projet ZFr01\_XXX.

Glisser-lâcher signifie sélectionner un objet en cliquant dessus avec la souris et le déplacer en maintenant le bouton de la souris appuyé. Le relâchement du bouton de la souris permet d'insérer l'objet à l'endroit désiré.



Double-cliquez dans le projet "Getting Started" sur l'**OB1**. L'éditeur de programme CONT/LIST/LOG s'ouvre.

L'OB1 de STEP 7 est exécuté de manière cyclique par la CPU. La CPU lit pour cela le programme ligne par ligne et en exécute les commandes. Lorsque la CPU est revenue à la première ligne du programme, elle a effectué un cycle. Le temps qu'elle a mis pour le faire est appelé le temps de cycle.

Pour poursuivre la programmation, reportez-vous si vous avez choisi le langage CONT au paragraphe 4.2, le langage LIST au paragraphe 4.3 et le langage LOG au paragraphe 4.4.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Création de blocs et de bibliothèques" via la commande de menu ? > **Rubriques d'aide.**



## L'éditeur de programme CONT/LIST/LOG

C'est dans l'éditeur de programme CONT/LIST/LOG que vous programmez les blocs. Vous voyez représentée ici à titre d'exemple la vue CONT.

The screenshot shows the SIMATIC Manager CONT editor interface. The main window displays a network structure on the left and a detailed view of a block on the right. The interface includes a menu bar, a toolbar, and a status bar at the bottom.

Callouts and their descriptions:

- Insérer un nouveau réseau**: Points to the 'Nouveau réseau' button in the toolbar.
- Afficher, masquer les éléments de programme et la structure du programme (la fenêtre peut être épinglée n'importe où dans la fenêtre de programme)**: Points to the 'Opérations sur bits' and 'Opérations sur bits' buttons in the toolbar.
- Modifier la vue du langage de programmation**: Points to the 'Eléments de programme' and 'Struct' buttons in the toolbar.
- Principaux éléments de programme CONT et LOG**: Points to the main content area of the editor.
- Eléments de programme (ici CONT) et structure d'appels**: Points to the left-hand pane showing the network structure.
- La vue d'ensemble des variables ou la vue de détail contient les paramètres et les variables locales du bloc.**: Points to the variable declaration table in the main editor.
- Titre et zone de commentaire du bloc ou du réseau**: Points to the comment area below the variable table.
- Ligne de saisie du programme (encore appelée Réseau ou Branche de courant).**: Points to the program input line below the comment area.
- Description succincte de l'élément de programme sélectionné**: Points to the status bar at the bottom of the editor.
- Les différents onglets de la fenêtre "Détails" permettent l'affichage de messages d'erreur et d'informations sur les opérands, sur l'édition des mnémoniques, le forçage des variables, la comparaison de blocs et l'édition de définitions d'erreur pour le diagnostic de processus.**: Points to the tabs at the bottom of the editor.

Nom	Type de données	Adresse	Commentaire
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 ...
OB1_SCAN_1			
OB1_PRIORITY			
OB1_OB_NUMBR			
OB1_RESERVED_1			
OB1_RESERVED_2			
OB1_RESERVED			
OB1_PREV_CYCLE			
OB1_MIN_CYCLE			
OB1_MAX_CYCLE			
OB1_DATE_TIME	Date_And_Ti...	12.0	Date and time OB1 started

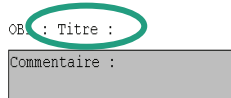
## 4.2 Programmation de l'OB1 en CONT

Vous apprenez dans les pages suivantes à programmer un circuit série, un circuit parallèle et une bascule Mise à 1 /Remise à 0 en langage de programmation CONT (Schéma à **CONTacts**).

### Programmation d'un circuit série en CONT



Si vous ne l'avez pas encore fait, sélectionnez via le menu **Affichage** le langage de programmation **CONT**.



Cliquez dans la zone **Titre** de l'OB1 et entrez comme titre pour celui-ci "Exécution cyclique".



Sélectionnez la position voulue de la branche de courant pour y insérer le premier élément.



Cliquez dans la barre d'outils sur le bouton représenté ici et insérez un contact à fermeture.



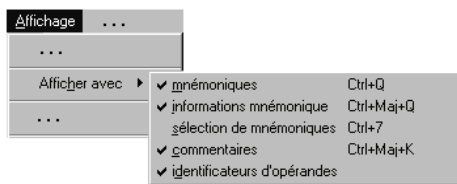
Insérez de la même manière un second contact à fermeture.



Insérez une bobine à l'extrémité droite de la branche de courant.



Pour achever notre circuit série, il manque encore les adresses des contacts et de la bobine.



Vérifiez si vous avez activé la représentation symbolique.



Cliquez sur **???** et entrez le nom symbolique "Commutateur 1" (entre guillemets !). Vous pouvez également choisir le nom dans la liste déroulante affichée. Confirmez avec la touche **Entrée**.



Introduisez pour le second contact à fermeture le nom symbolique "Commutateur 2".



Entrez pour la bobine le nom "Feu vert".



Votre circuit série est maintenant programmé.



Enregistrez le bloc lorsque le programme ne signale plus aucune erreur.

Les mnémoniques sont affichés en rouge s'ils ne sont pas contenus dans la table des mnémoniques ou s'il y a une erreur syntaxique dans le programme.





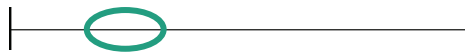
### Programmation d'un circuit parallèle en CONT

**Réseau 1** Titre :  
Commentaire :

Sélectionnez le **Réseau 1**.



Insérez un nouveau réseau.



Sélectionnez à nouveau la branche de courant.



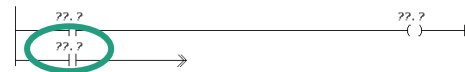
Insérez un contact à fermeture et une bobine.



Sélectionnez la branche verticale du réseau.



Insérez une branche parallèle.



Insérez dans la branche parallèle un second contact à fermeture.



Fermez la branche en cliquant le cas échéant sur l'extrémité de la flèche).



Il ne reste plus qu'à compléter les adresses.

Procédez pour cela comme pour le circuit série.



Entrez pour le contact du haut "Commutateur 3", pour le contact du bas "Commutateur 4" et pour la bobine "Feu rouge".



Enregistrez le bloc.

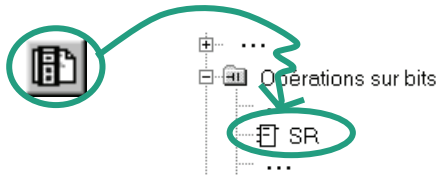
## Programmation d'une bascule en CONT



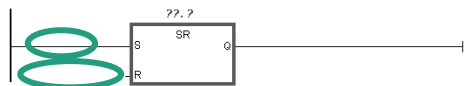
Sélectionnez le réseau 2, et insérez un troisième réseau.



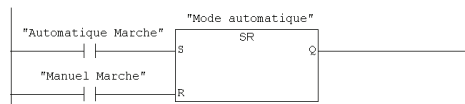
Sélectionnez ensuite la branche de courant.



Naviguez dans le catalogue des éléments de programme jusqu'à l'entrée **Opérations sur bits** et sélectionnez-y la bascule **SR**. Double-cliquez sur celle-ci pour l'insérer.



Insérez un contact à fermeture avant les entrées S et R.

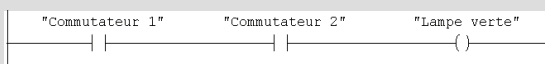


Entrez pour la bascule SR les noms symboliques suivants :  
 "Automatique Marche" pour le premier contact, "Manuel Marche" pour le second contact et entrez comme titre de bascule "Mode automatique".



Enregistrez le bloc et fermez l'éditeur de programme.

Si vous voulez voir la différence entre l'adressage absolu et l'adressage symbolique, désactivez l'affichage symbolique en choisissant dans le menu **Affichage** la commande **Afficher avec > mnémoniques**.



Exemple d'adressage symbolique en CONT



Exemple d'adressage absolu en CONT

Si les mnémoniques apparaissent coupés, vous pouvez agrandir la largeur du champ de l'opérande avec la commande de menu **Outils > Paramètres > CONT/LOG > Largeur du champ d'opérande** de l'éditeur de programme CONT/LIST/LOG.

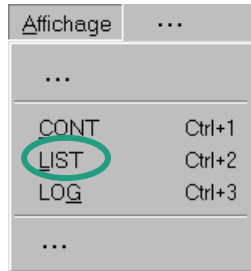
Celui-ci peut être élargi à une largeur de 10 à 26 caractères.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Edition d'instructions CONT" via la commande de menu **? > Rubriques d'aide**.

### 4.3 Programmation de l'OB1 en LIST

Vous apprenez dans les pages qui suivent à programmer une instruction ET, une instruction OU et des instructions de mise à 1 et de mise à 0 en LIST (**LISTE** d'instructions).

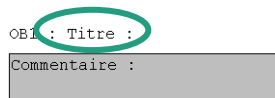
#### Programmation d'une instruction ET en LIST



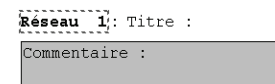
Si vous ne l'avez pas encore fait, sélectionnez dans le menu **Affichage** le langage de programmation **LIST**.



Vérifiez que la représentation symbolique est activée.



Cliquez dans la zone de **Titre** de l'OB1 et entrez par exemple pour titre "Exécution cyclique".



Sélectionnez la zone de la première instruction.

U "Commutateur 1"

Inscrivez dans la première ligne du programme un U (UND) pour ET suivi d'un espace et du mnémonique "Commutateur 1" (entre guillemets).

Cliquez à la fin de la ligne sur la touche **Entrée**. Le curseur saute à la ligne suivante.

U "Commutateur 1" Complétez de la même manière l'instruction UND (ET).  
 U "Commutateur 2"  
 = "Lampe verte"



Votre fonction ET est maintenant programmée. Enregistrez le bloc lorsque le programme ne signale plus aucune erreur.

Les mnémoniques sont affichés en rouge s'ils ne sont pas contenus dans la table des mnémoniques ou s'il y a une erreur syntaxique dans le programme.

### Programmation d'une instruction OU en LIST

Réseau 1: Titre :  
 Commentaire :

Sélectionnez le **Réseau 1**.



Insérez un nouveau réseau et sélectionnez à nouveau la zone de saisie.

○ "Commutateur 3" Entrez un O (ODER) pour OU suivi du mnémonique "Commutateur 3" (comme nous l'avons fait pour ET).  
 ○ "Commutateur 3"  
 ○ "Commutateur 4"  
 = "Lampe rouge" Complétez l'instruction ODER (OU) et enregistrez-la.

### Programmation d'une bascule en LIST



Sélectionnez le réseau 2 et insérez un troisième réseau.

U "Automatique Marche"

Inscrivez l'instruction U dans la première ligne avec pour mnémonique "Automatique Marche".

U "Automatique Marche"

Complétez l'instruction de bascule en vous orientant au modèle ci-contre et enregistrez-la. Fermez le bloc.

S "Mode automatique"

U "Manuel Marche"

R "Mode automatique"



Si vous voulez voir la différence entre l'adressage absolu et l'adressage symbolique, désactivez l'affichage symbolique en choisissant dans le menu **Affichage** la commande **Afficher avec > mnémoniques**.

```
U "Commutateur 1"
U "Commutateur 2"
= "Lampe verte"
```

Exemple d'adressage symbolique en LIST

```
U E 0.1
U E 0.2
= A 4.0
```

Exemple d'adressage absolu en LIST

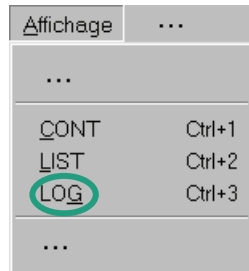
Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Edition d'instructions LIST" via la commande de menu ? > **Rubriques d'aide**.



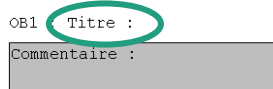
## 4.4 Programmation de l'OB1 en LOG

Vous apprenez dans les pages qui suivent à programmer une fonction ET, une fonction OU et une bascule en langage de programmation LOG (**LOG**igramme).

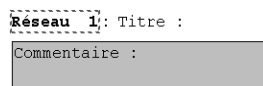
### Programmation d'une fonction ET en LOG



Si cela n'est déjà fait, sélectionnez le langage de programmation **LOG** dans le menu **Affichage**.



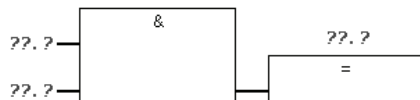
Cliquez dans la zone de titre de l'**OB1** et entrez comme titre "Exécution cyclique".



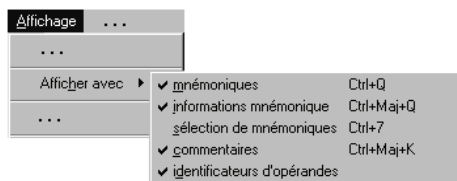
Sélectionnez la zone de saisie pour y entrer la fonction ET (sous la zone du commentaire).



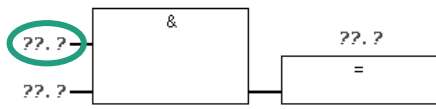
Insérez une boîte ET (&) et une affectation (=).



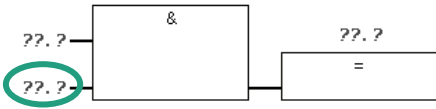
Il ne reste plus qu'à compléter les adresses des différents éléments de la fonction ET.



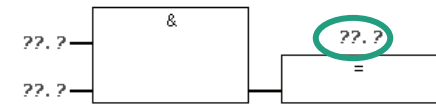
Vérifiez que la représentation symbolique est activée.



Cliquez sur **??.** et entrez le nom symbolique "Commutateur 1" (entre guillemets !). Vous pouvez également sélectionner le nom dans la liste déroulante affichée. Confirmez avec la touche **Entrée**.



Inscrivez pour la seconde entrée le mnémonique "Commutateur 2".



Entrez comme nom d'affectation "Feu rouge".



Votre fonction ET est maintenant programmée.



Lorsqu'aucun opérande n'est plus affiché en rouge, vous pouvez enregistrer.

Les mnémoniques sont affichés en rouge s'ils ne sont pas contenus dans la table des mnémoniques ou s'il y a une erreur syntaxique dans le programme.

## Programmation d'une fonction OU en LOG



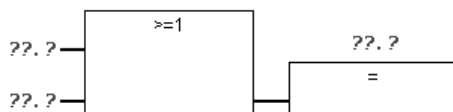
Insérez un nouveau réseau.

Réseau 2: Titre :  
Commentaire :

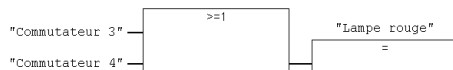
Sélectionnez à nouveau la zone de saisie pour y entrer la fonction OU.



Insérez une boîte OU ( $\geq 1$ ) et une affectation (=).



Il ne reste plus qu'à compléter les adresses. Procédez comme pour la fonction ET.



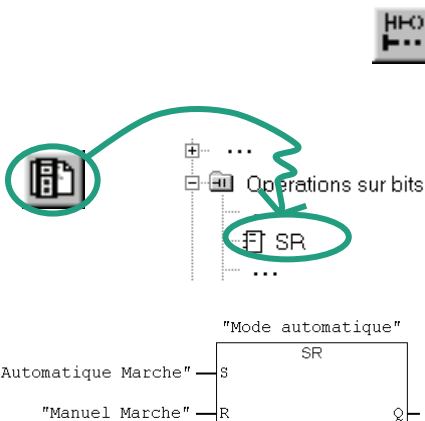
Entrez pour la première entrée partant du haut le mnémonique "Commutateur 3", pour la seconde entrée le mnémonique "Commutateur 4" et pour l'affectation le mnémonique "Feu rouge".



Enregistrez le bloc.



## Programmation d'une bascule en LOG



Sélectionnez le réseau 2 et insérez un troisième réseau. Sélectionnez de nouveau la zone de saisie (sous la zone du commentaire).

Naviguez dans le catalogue des éléments de programme jusqu'à l'entrée **Opérations sur bits** et sélectionnez-y une bascule **SR**. Un double clic sur celle-ci insère une bascule dans le réseau.

Introduisez aux entrées et sorties de la bascule les noms symboliques suivants : S "Automatique Marche", R "Manuel Marche", Mémento "Mode automatique".

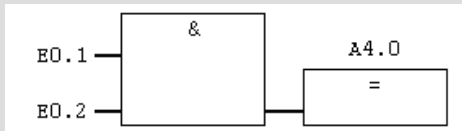


Enregistrez le bloc et fermez l'éditeur de programme.

Si vous voulez voir la différence entre l'adressage absolu et l'adressage symbolique, désactivez l'affichage symbolique en choisissant dans le menu **Affichage** la commande **Afficher avec > mnémoniques**.



Exemple d'adressage symbolique en LOG



Exemple d'adressage absolu en LOG

Si les mnémoniques apparaissent coupés, vous pouvez agrandir la largeur du champ de l'opérande avec la commande de menu **Outils > Paramètres > CONT/LOG > Largeur du champ d'opérande** de l'éditeur de programme CONT/LIST/LOG.

Celui-ci peut être élargi à une largeur de 10 à 26 caractères.

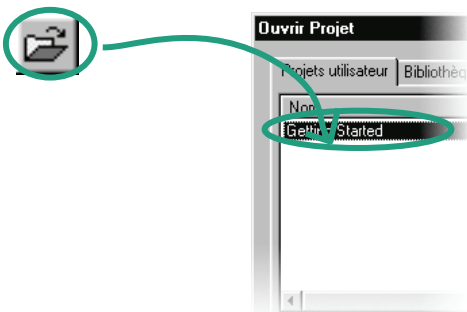
Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Edition d'instructions LOG" via la commande de menu ? > **Rubriques d'aide**.

## 5 Création d'un programme avec FB et DB

### 5.1 Créer et ouvrir un bloc fonctionnel

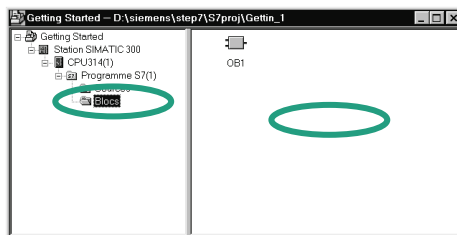
Le bloc fonctionnel (FB) est subordonné au bloc d'organisation. Il renferme une partie du programme qui peut être appelée autant de fois qu'on le veut dans l'OB1. Tous les paramètres formels et toutes les données statiques du bloc fonctionnel sont stockées dans un bloc de données DB séparé qui est associé au bloc fonctionnel.

Vous programmez le bloc fonctionnel (FB1 au nom symbolique "Moteur", voir la table des mnémoniques, page 21) dans l'éditeur de programme CONT/LIST/LOG déjà connu. Vous devez utiliser pour cela le même langage de programmation que celui que vous avez utilisé au chapitre 4 (Programmation de l'OB1).



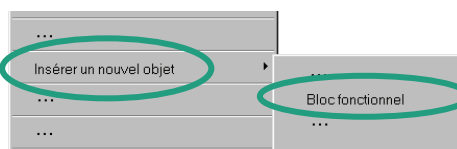
La table des mnémoniques doit pour cela avoir été copiée dans le projet "Getting Started". Si ce n'est pas le cas, référez-vous à la page 24, copiez la table des mnémoniques et revenez à cette page.

Ouvrez si le projet n'est pas encore ouvert la fenêtre du projet "Getting Started".

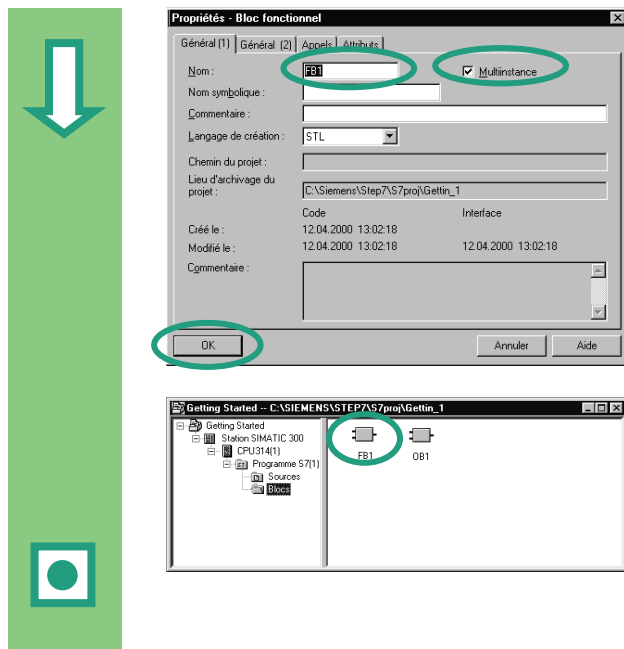


Naviguez jusqu'au dossier **Blocs** et ouvrez-le.

Cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre.



Le menu contextuel du bouton droit de la souris offre à nouveau les principales commandes de menu de la barre d'outils. Insérez comme nouvel objet un bloc fonctionnel.



Choisissez dans la boîte de dialogue des propriétés du bloc fonctionnel, le langage de création, activez la case d'option **Multiinstance** et validez toutes les autres options avec **OK**.

Le bloc fonctionnel **FB1** a été inséré dans le dossier **Blocs**.

Un double clic sur le bloc FB1 ouvre l'éditeur de programme CONT/LIST/LOG.

Si vous avez choisi le langage de programmation CONT, poursuivez au chapitre 5.2, si vous avez choisi le langage LIST au chapitre 5.3 et si vous avez choisi le langage LOG au chapitre 5.4.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Création de blocs et de bibliothèques" via la commande de menu ? > **Rubriques d'aide**.

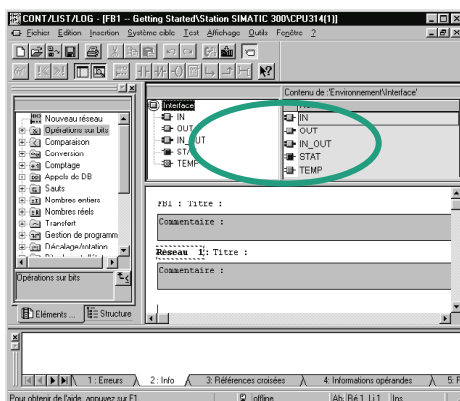
## 5.2 Programmation du bloc FB1 en CONT

Nous vous montrons comment programmer un bloc fonctionnel pouvant par exemple commander et surveiller un moteur à essence et un moteur Diesel grâce à ses deux blocs de données.

Tous les signaux spécifiques à un type de moteur sont transmis sous la forme de paramètres par le bloc d'organisation au bloc fonctionnel et doivent donc au préalable être déclarés comme paramètres d'entrée et de sortie ("in" et "out") dans la table de déclaration des variables.

Vous devez déjà connaître la programmation d'un circuit série, d'un circuit parallèle et d'une bascule avec STEP 7.

### Déclarer ou définir en premier les variables



La fenêtre de l'éditeur de programme CONT/LIST/LOG est ouverte et la vue CONT est activée (commande **Affichage > CONT**).

Vous voyez maintenant en titre de la fenêtre FB1 car vous avez ouvert l'éditeur de programme en double-cliquant sur ce bloc.

La section de déclaration des variables est constituée d'une vue d'ensemble des variables (volet gauche de la fenêtre) et de la vue de détail des variables (volet droit de la fenêtre).

Sélectionnez successivement dans la vue d'ensemble les types de déclaration "IN", "OUT" et "STAT" et entrez dans la vue de détails des variables les déclarations suivantes.

Cliquez dans les champs correspondants de la vue détaillée des variables et reprenez les données affichées dans les figures suivantes. Sélectionnez le type de données dans la liste déroulante affichée.



Contenu de : 'Environnement\Interface\IN'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Switch_On	Bool	0.0	FALSE	Mise en marche du moteur
Switch_Off	Bool	0.1	FALSE	Arrêt du moteur
Failure	Bool	0.2	FALSE	Défaillance du moteur provoquant l'arrêt
Actual_Speed	Int	2.0	0	Vitesse réelle du moteur

Contenu de : 'Environnement\Interface\OUT'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Engine_On	Bool	4.0	FALSE	Le moteur se met en marche
Preset_Speed_Reached	Bool	4.1	FALSE	Vitesse prescrite atteinte

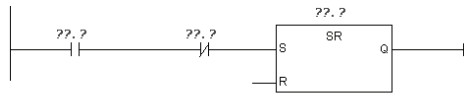
Contenu de : 'Environnement\Interface\STAT'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Preset_Speed	Int	6.0	1500	Vitesse de moteur prescrite

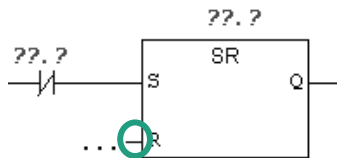
Pour afficher les colonnes invisibles, cliquez sur le bouton droit de la souris et choisissez les colonnes voulues dans le menu contextuel.

Seules les lettres, les chiffres et le caractère de soulignement sont autorisés pour l'introduction de noms dans la table de déclaration des variables.

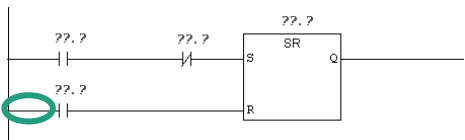
### Programmer la mise en marche et la mise à l'arrêt du moteur



Insérez dans le réseau 1 un contact à fermeture, un contact à ouverture et une bascule SR en série en cliquant sur les icônes correspondantes ou en les sélectionnant dans le catalogue des éléments de programme.



Sélectionnez ensuite la branche de courant suivant immédiatement l'entrée R.

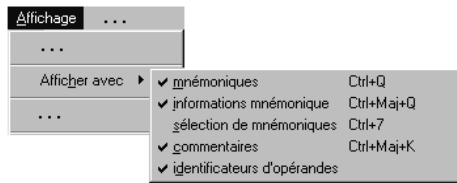


Insérez un autre contact à fermeture. Sélectionnez la branche de courant suivant immédiatement le contact à fermeture.



Insérez parallèlement au contact à fermeture un contact à ouverture.



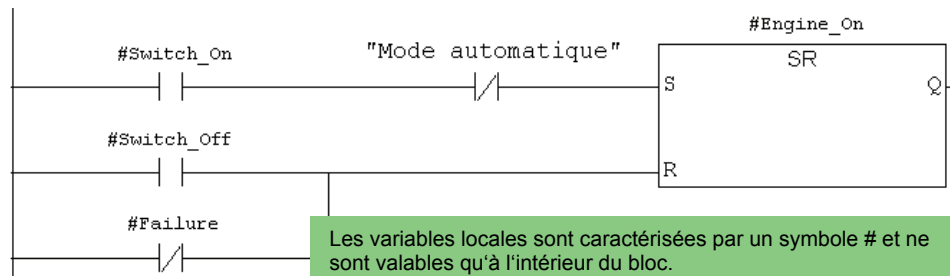


Vérifiez que la représentation symbolique est activée.

Sélectionnez les points d'interrogation et entrez les noms symboliques de la table de déclaration des variables (# est automatiquement attribué).

Entrez pour le contact à ouverture du circuit série le mnémorique "Mode automatique".

Enregistrez ensuite votre programme.



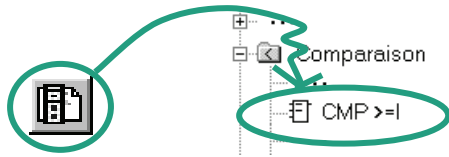
Les variables locales sont caractérisées par un symbole # et ne sont valables qu'à l'intérieur du bloc.

Les variables globales figurent entre des guillemets. Elles sont définies dans la table des mnémoniques et sont valables dans tout le programme.

L'état de signal "Mode automatique" est défini dans l'OB1 (réseau 3, voir pages 29) par une autre bascule SR et interrogé à présent dans le bloc FB1.



### Programmer une surveillance de vitesse



Insérez un nouveau réseau et sélectionnez la branche de courant.

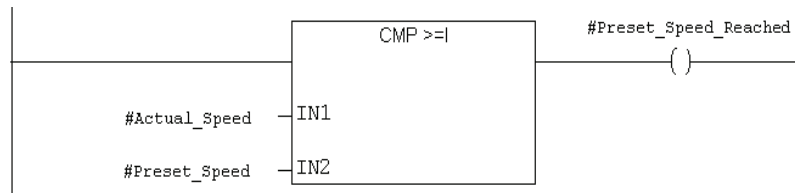
Naviguez ensuite dans le catalogue des éléments de programme jusqu'au dossier **Comparaison** et insérez le comparateur **CMP>=I**.



Insérez également une bobine à la fin de la branche de courant.

Sélectionnez de nouveau les points d'interrogation et intitulez la bobine et le comparateur en vous servant des noms correspondants dans la table de déclaration des variables.

Enregistrez en dernier lieu votre programme.



#### Quand le moteur se met-il en marche ou à l'arrêt ?

Le moteur est activé si la variable #Mise en marche a pour état de signal "1" et si la variable "Mode automatique" a pour état de signal "0". Nous réalisons cette fonctionnalité en niant la variable "Mode automatique" (contact à ouverture).

Si la variable #Mise à l'arrêt a pour état de signal "1" ou si la variable #Défaillance a pour état de signal "0", le moteur se met à l'arrêt. Nous réalisons cette fonction à nouveau en niant la variable #Défaillance (#Défaillance est donc un signal entrant en action quand sa valeur est nulle. Dans le cas normal, il aura la valeur 1, dans le cas d'une défaillance, il aura la valeur 0.).

#### Comment le comparateur surveille-t-il la vitesse du moteur ?

Le comparateur compare les variables #Actual\_Speed et #Preset\_Speed et inscrit ce résultat dans la variable #Preset\_Speed\_Reached (état de signal 1).

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Editer la table de déclaration des variables" ou "Editer les instructions CONT" via la commande de menu ? > **Rubriques d'aide**.

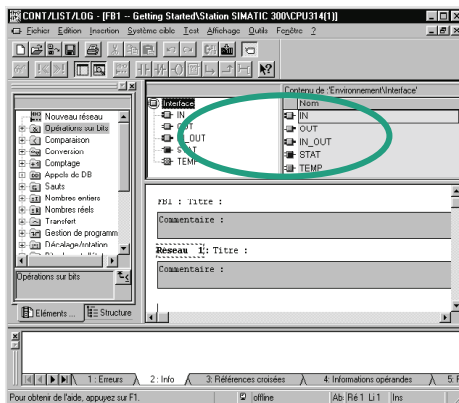
## 5.3 Programmation du bloc FB1 en LIST

Nous vous montrons comment programmer un bloc fonctionnel pouvant par exemple commander et surveiller un moteur à essence et un moteur Diesel grâce à ses deux blocs de données.

Tous les signaux spécifiques à un type de moteur sont transmis sous la forme de paramètres par le bloc d'organisation au bloc fonctionnel et doivent donc au préalable être déclarés comme paramètres d'entrée et de sortie ("in" et "out") dans la table de déclaration des variables.

Vous devez déjà connaître la programmation d'une instruction ET, d'une instruction OU et d'une bascule en LIST.

### Déclarer ou définir en premier les variables



La fenêtre de l'éditeur de programme CONT/LIST/LOG est ouverte et la vue CONT est activée (commande **Affichage > LIST**).

Vous voyez maintenant en titre de la fenêtre FB1 car vous avez ouvert l'éditeur de programme en double-cliquant sur ce bloc.

La section de déclaration des variables est constituée d'une vue d'ensemble des variables (volet gauche de la fenêtre) et de la vue de détail des variables (volet droit de la fenêtre).

Sélectionnez successivement dans la vue d'ensemble les types de déclaration "IN", "OUT" et "STAT" et entrez dans la vue de détails des variables les déclarations suivantes.

Cliquez dans les champs correspondants de la vue détaillée des variables et reprenez les données affichées dans les figures suivantes. Sélectionnez le type de données dans la liste déroulante affichée.



Contenu de : 'Environnement\Interface\IN'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Switch_On	Bool	0.0	FALSE	Mise en marche du moteur
Switch_Off	Bool	0.1	FALSE	Arrêt du moteur
Failure	Bool	0.2	FALSE	Défaillance du moteur provoquant l'arrêt
Actual_Speed	Int	2.0	0	Vitesse réelle du moteur

Contenu de : 'Environnement\Interface\OUT'

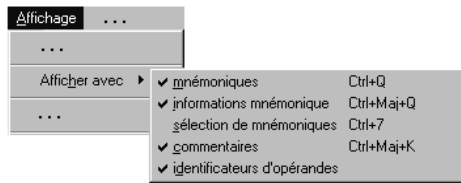
Nom	Type de données	Adresse	Valeur initiale	Commentaire
Engine_On	Bool	4.0	FALSE	Le moteur se met en marche
Preset_Speed_Reached	Bool	4.1	FALSE	Vitesse prescrite atteinte

Contenu de : 'Environnement\Interface\STAT'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Preset_Speed	Int	6.0	1500	Vitesse de moteur prescrite

Seules les lettres, les chiffres et le caractère de soulignement sont autorisés pour l'introduction de noms dans la table de déclaration des variables.

### Programmer la mise en marche et la mise à l'arrêt du moteur



Vérifiez que la représentation symbolique est activée.

```

U   #Switch_On
UN  "Automatic_Mode"
S   #Engine_On
O   #Switch_Off
ON  #Failure
R   #Engine_On
    
```

Entrez dans le réseau 1 les instructions requises.

Les variables locales sont caractérisées par un symbole # et ne sont valables qu'à l'intérieur du bloc.  
 Les variables globales figurent entre des guillemets. Elles sont définies dans la table des mnémoniques et sont valables dans tout le programme.  
 L'état de signal "Mode automatique" est défini dans l'OB1 (réseau 3, voir pages 29) par une autre bascule SR et interrogé à présent dans le bloc FB1.



## Programmer une surveillance de vitesse

```
L   #Actual_Speed
L   #Preset_Speed
>=I
=   #Preset_Speed_Reached
```

Insérez un nouveau réseau et entrez les instructions voulues. Enregistrez ensuite votre programme.



### Quand le moteur se met-il en marche ou à l'arrêt ?

Le moteur est activé si la variable #Mise en marche a pour état de signal "1" et si la variable "Mode automatique" a pour état de signal "0". Nous réalisons cette fonctionnalité en niant la variable "Mode automatique" (contact à ouverture).

Si la variable #Mise à l'arrêt a pour état de signal "1" ou si la variable #Défaillance a pour état de signal "0", le moteur se met à l'arrêt. Nous réalisons cette fonctionnalité également en niant la variable #Défaillance (#Défaillance est un signal qui entre en action quand sa valeur est nulle. Il a dans le cas normal la valeur 1, et en cas de défaillance la valeur 0.).

### Comment le comparateur surveille-t-il la vitesse du moteur ?

Le comparateur compare les variables #Actual\_Speed et #Preset\_Speed et inscrit ce résultat dans la variable #Preset\_Speed\_Reached (état de signal 1).

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Editer la table de déclaration des variables" ou "Editer les instructions LIST" via la commande de menu ? > **Rubriques d'aide.**

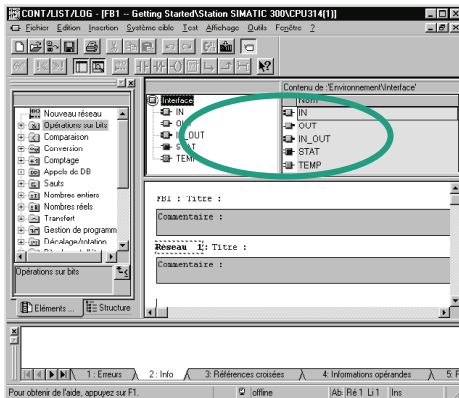
## 5.4 Programmation du bloc FB1 en LOG

Nous vous montrons comment programmer un bloc fonctionnel pouvant par exemple commander et surveiller un moteur à essence et un moteur Diesel grâce à ses deux blocs de données.

Tous les signaux spécifiques à un type de moteur sont transmis sous la forme de paramètres par le bloc d'organisation au bloc fonctionnel et doivent donc au préalable être déclarés comme paramètres d'entrée et de sortie ("in" et "out") dans la table de déclaration des variables.

Vous devez pour cela déjà savoir programmer une fonction ET, une fonction OU et une bascule en LOG.

### Déclarer ou définir en premier les variables



La fenêtre de l'éditeur de programme CONT/LIST/LOG est ouverte et la vue LOG est activée (commande **Affichage > LOG**).

Vous voyez maintenant en titre de la fenêtre FB1 car vous avez ouvert l'éditeur de programme en double-cliquant sur ce bloc.

La section de déclaration des variables est constituée d'une vue d'ensemble des variables (volet gauche de la fenêtre) et de la vue de détail des variables (volet droit de la fenêtre).

Sélectionnez successivement dans la vue d'ensemble les types de déclaration "IN", "OUT" et "STAT" et entrez dans la vue de détails des variables les déclarations suivantes.

Cliquez dans les champs correspondants de la vue détaillée des variables et reprenez les données affichées dans les figures suivantes. Sélectionnez le type de données dans la liste déroulante affichée.



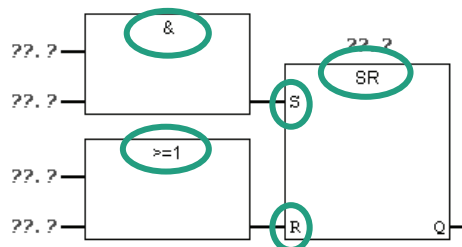
Contenu de 'Environnement\Interface\IN'					
Nom	Type de données	Adresse	Valeur initiale	Commentaire	
Switch_On	Bool	0.0	FALSE	Mise en marche du moteur	
Switch_Off	Bool	0.1	FALSE	Arrêt du moteur	
Failure	Bool	0.2	FALSE	Défaillance du moteur provoquant l'arrêt	
Actual_Speed	Int	2.0	0	Vitesse réelle du moteur	

Contenu de 'Environnement\Interface\OUT'					
Nom	Type de données	Adresse	Valeur initiale	Commentaire	
Engine_On	Bool	4.0	FALSE	Le moteur se met en marche	
Preset_Speed_Reached	Bool	4.1	FALSE	Vitesse prescrite atteinte	

Contenu de 'Environnement\Interface\STAT'					
Nom	Type de données	Adresse	Valeur initiale	Commentaire	
Preset_Speed	Int	6.0	1500	Vitesse de moteur prescrite	

Seules les lettres, les chiffres et le caractère de soulignement sont autorisés pour l'introduction de noms dans la table de déclaration des variables.

### Programmer la mise en marche et la mise à l'arrêt d'un moteur



Insérez dans le réseau 1 une fonction SR (dossier Opérations sur bits) que vous sélectionnez dans le catalogue des éléments de programme.

Reliez l'entrée S (mise à 1) à une boîte ET et l'entrée R (remise à 0) à une boîte OU.

Affichage ...  
 ...  
 Afficher avec mnémoniques Ctrl+Q  
 informations mnémonique Ctrl+Maj+Q  
 sélection de mnémoniques Ctrl+7  
 commentaires Ctrl+Maj+K  
 identificateurs d'opérandes

Vérifiez que la représentation symbolique est activée.

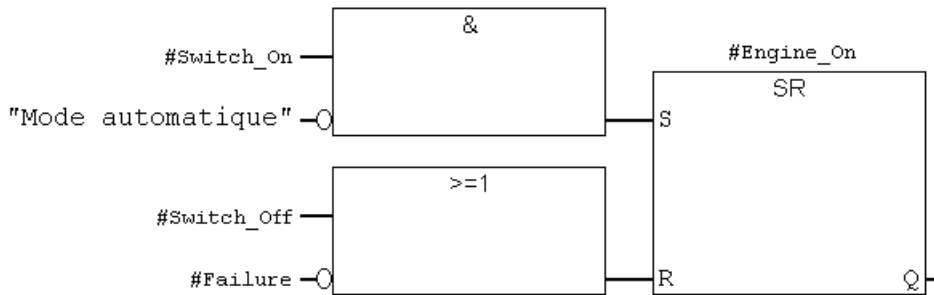


Cliquez sur les points d'interrogation ??? et entrez à leur place les noms appropriés de la table de déclaration des variables (# est automatiquement entré par le programme).

Veillez à ce qu'une entrée de la fonction ET ait pour adresse le nom symbolique "Mode automatique".

Il ne vous reste plus qu'à nier les entrées "Mode automatique" et #Defaillance en cliquant dans la barre d'outils sur le bouton servant à la négation.

Enregistrez ensuite votre programme.



Les variables locales sont caractérisées par un symbole # et ne sont valables qu'à l'intérieur du bloc.

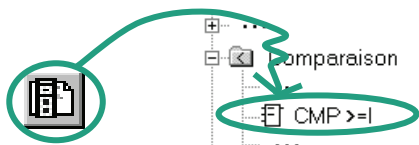
Les variables globales figurent entre des guillemets. Elles sont définies dans la table des mnémoniques et sont valables dans tout le programme.

L'état de signal "Mode automatique" est défini dans l'OB1 (réseau 3, voir pages 29) par une autre bascule SR et interrogé à présent dans le bloc FB1.





## Programmer une surveillance de vitesse

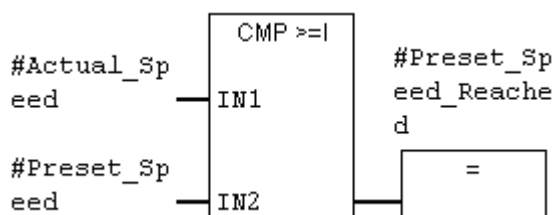


Insérez un nouveau réseau et sélectionnez la zone de saisie.

Naviguez ensuite dans le catalogue des éléments de programme jusqu'au dossier **Comparaison** et sélectionnez un comparateur **CMP>=I**.

Insérez après le comparateur une affectation de sortie et entrez aux adresses les mnémoniques de la table de déclaration des variables.

Enregistrez ensuite votre programme.



### Quand le moteur se met-il en marche ou à l'arrêt ?

Le moteur est activé si la variable #Mise en marche a pour état de signal "1" et si la variable "Mode automatique" a pour état de signal "0". Nous réalisons cette fonctionnalité en niant (contact à ouverture) la variable "Mode automatique".

Si la variable #Mise à l'arrêt a pour état de signal "1" ou si la variable #Défaillance a pour état de signal "0", le moteur se met à l'arrêt. Nous réalisons cette fonctionnalité également en niant la variable #Défaillance (#Défaillance est un signal entrant en action quand sa valeur est nulle. Il a dans le cas normal la valeur 1, et en cas de défaillance la valeur 0).

### Comment le comparateur surveille-t-il la vitesse du moteur ?

Le comparateur compare les variables #Actual\_Speed et #Preset\_Speed et inscrit ce résultat dans la variable #Preset\_Speed\_Reached (état de signal 1).

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Editer la table de déclaration des variables" ou "Editer les instructions LOG" via la commande de menu ? > **Rubriques d'aide**.

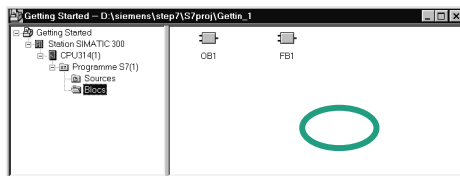
## 5.5 Générer les blocs de données d'instance et modifier les valeurs effectives

Vous avez programmé le bloc fonctionnel FB1 ("Moteur") et défini les paramètres spécifiques à chaque moteur dans la table de déclaration des variables.

Pour pouvoir programmer l'appel (CALL) du FB dans l'OB1, vous devez générer son bloc de données (DB). Un FB est toujours affecté à un DB d'instance.

Le FB doit commander et surveiller un moteur à essence ou un moteur Diesel. Les vitesses prescrites des moteurs sont stockées dans deux DB distincts dans lesquels seule la valeur effective (#Vitesse\_prescrite) change.

En ne programmant le bloc fonctionnel qu'une seule fois, vous réduisez le temps de programmation.

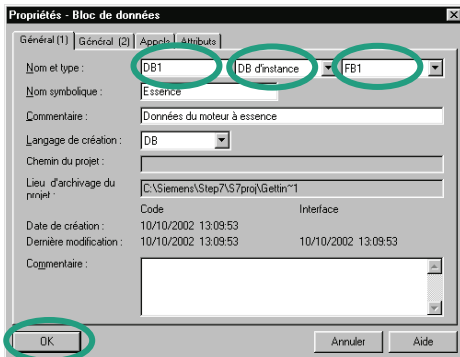


Le projet "Getting Started" est ouvert dans SIMATIC Manager.

Naviguez jusqu'au dossier **Blocs** et cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre.

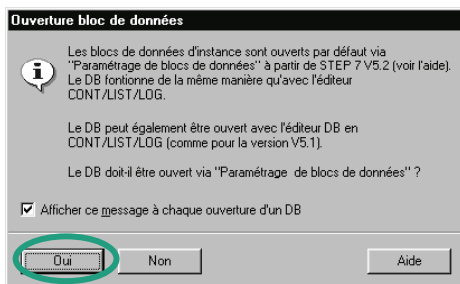


Insérez avec le menu contextuel du bouton droit de la souris un **Bloc de données**.



Reprenez dans la boîte de dialogue "Propriétés – Bloc de données" le nom DB1, choisissez dans la liste déroulante le type "DB d'instance" et reprenez le nom du bloc fonctionnel "FB1". Confirmez vos entrées avec **OK**.

Le bloc de données DB1 est inséré dans le projet "Getting Started".



Adresse	Décl.	Nom	Type	Valeur initial	Valeur en co	Commentaire
1	0.0 in	Switch_On	RBOOL	FALSE	FALSE	Mise en marche du moteur
2	0.1 in	Switch_Off	BOOL	FALSE	FALSE	Arrêt du moteur
3	0.2 in	Failure	BOOL	FALSE	FALSE	Défillance du moteur provoquant l'arrêt
4	2.0 in	Actual_Speed	INT	0	0	Vitesse réelle du moteur
5	4.0 out	Engine_On	BOOL	FALSE	FALSE	Le moteur se met en marche
6	4.1 out	Presst_Speed_Reached	BOOL	FALSE	FALSE	Vitesse prescrite atteinte
7	6.0 stat	Presst_Speed	INT	1500	1500	Vitesse de moteur prescrite

Adresse	Décl.	Nom	Type	Valeur initial	Valeur en co	Commentaire
1	0.0 in	Switch_On	RBOOL	FALSE	FALSE	Mise en marche du moteur
2	0.1 in	Switch_Off	BOOL	FALSE	FALSE	Arrêt du moteur
3	0.2 in	Failure	BOOL	FALSE	FALSE	Défillance du moteur provoquant l'arrêt
4	2.0 in	Actual_Speed	INT	0	0	Vitesse réelle du moteur
5	4.0 out	Engine_On	BOOL	FALSE	FALSE	Le moteur se met en marche
6	4.1 out	Presst_Speed_Reached	BOOL	FALSE	FALSE	Vitesse prescrite atteinte
7	6.0 stat	Presst_Speed	INT	1500	1200	Vitesse de moteur prescrite



Ouvrez le bloc **DB1** par un double clic.

Confirmez dans le dialogue suivant par **OUI** pour paramétrer le bloc de données d'instance.

Entrez à présent pour le moteur à essence la valeur "1500" dans la colonne de la valeur effective (dans la ligne "Vitesse\_Prescrite"). Vous venez ainsi de définir la vitesse maximale du moteur.

Enregistrez le DB1 et fermez l'éditeur de programme.

Générez de la même manière un second DB pour le FB1 que vous appellerez DB2.

Entrez cette fois pour la valeur effective du moteur Diesel "1200".

Enregistrez le DB 2 et fermez la fenêtre de programme.

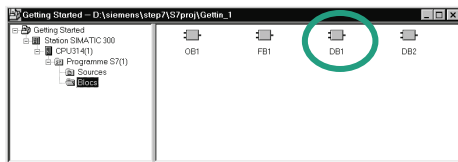
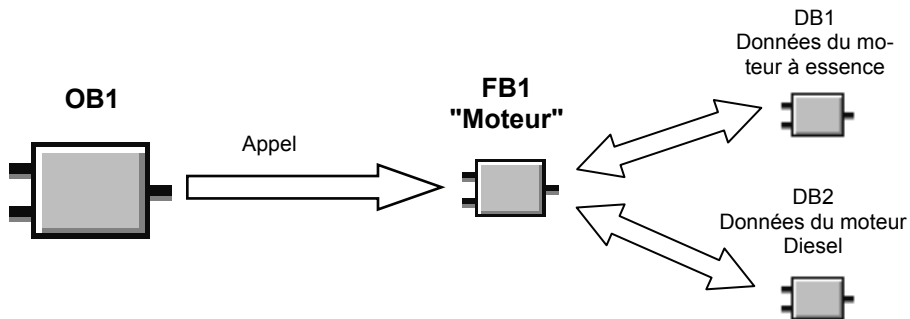
Avec la modification des valeurs effectives, nous en avons terminé avec les préparatifs de notre bloc fonctionnel destiné à commander deux moteurs. Pour commander d'autres moteurs, il nous suffirait de générer d'autres blocs de données.

Pour programmer maintenant l'appel du FB dans l'OB1, reportez-vous si votre langage de programmation est CONT au paragraphe 5.6, si votre langage de programmation est LIST au paragraphe 5.7 et si vous avez comme langage de programmation LOG au paragraphe 5.8.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Création de blocs de données" via la commande de menu ? > Rubriques d'aide.

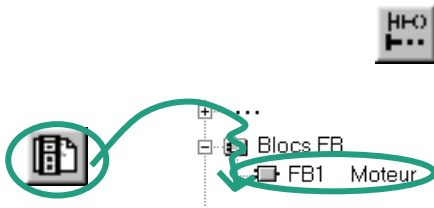
## 5.6 Programmation d'un appel de bloc en CONT

Toute la programmation du FB resterait sans effet si son appel n'était pas programmé dans l'OB1. Un bloc de données est utilisé pour chaque appel du FB et servira à commander un moteur différent.



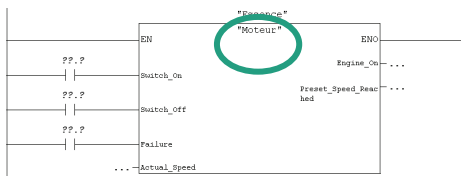
SIMATIC Manager est ouvert avec le projet "Getting Started".

Naviguez jusqu'au dossier **Blocs** et ouvrez l'**OB1**.



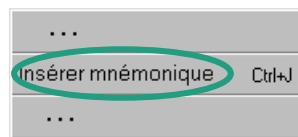
Sélectionnez dans la fenêtre de programme CONT/LIST/LOG le réseau 3 et insérez le réseau 4.

Naviguez ensuite dans l'onglet "Éléments de programme" jusqu'au **FB1** et insérez-le dans votre programme par double clic.

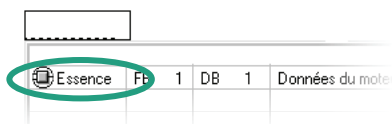


Insérez un contact à fermeture avant les paramètres Mise en marche, Mise à l'arrêt et Défaillance.

Cliquez sur les points d'interrogation ??? au-dessus de "Moteur", puis aussitôt dans le cadre de saisie avec le bouton droit de la souris.

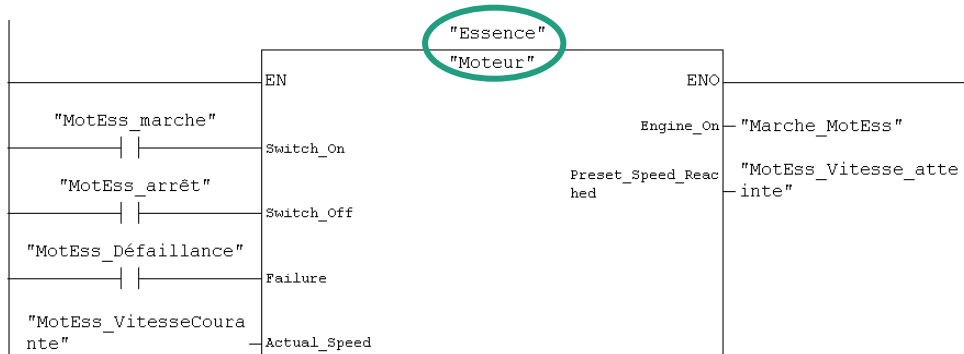


Cliquez dans le menu contextuel du bouton droit de la souris sur **Insérer mnémonique**. Ceci fait s'ouvrir une liste déroulante.



Double-cliquez sur le bloc de données **Essence**. Il s'affiche automatiquement entre guillemets dans le cadre de saisie.

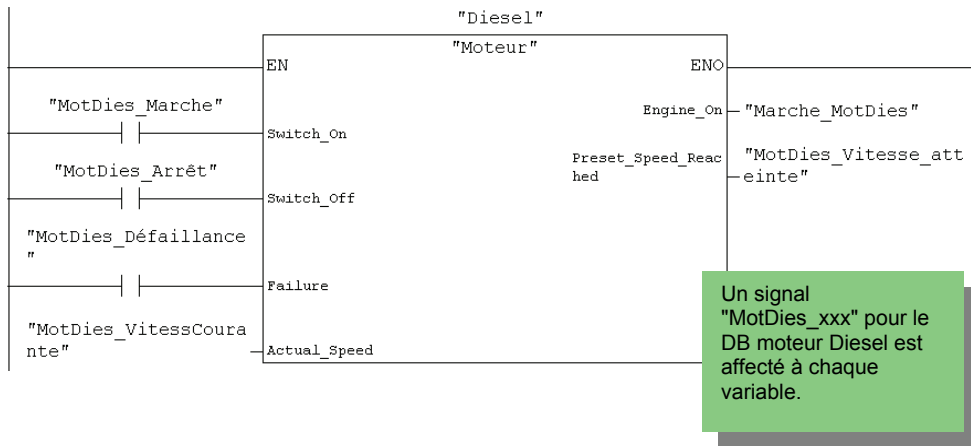
Cliquez sur les points d'interrogation et entrez une adresse qui doit figurer entre guillemets. Entrez pour les autres paramètres du bloc fonctionnel les mnémoniques appropriés que vous sélectionnez dans la liste déroulante.



Les variables d'entrée et de sortie (déclaration "in" et "out") spécifiques au moteur sont affichées dans le FB "Moteur". Ces variables reçoivent chacune un signal "MotEss\_xxx" signalant leur appartenance au DB Moteur à essence.



Programmez dans un nouveau réseau l'appel du bloc fonctionnel "Moteur" (FB1) avec le bloc de données "Diesel" (DB2) en sélectionnant pour chaque paramètre l'opérande dans la liste déroulante.



Enregistrez votre programme et fermez le bloc.

Si vous créez des structures de programme avec des OB, des FB et des DB, vous devez programmer l'appel d'un bloc subordonné (par exemple un FB1) dans le bloc hiérarchique supérieur (par l'exemple l'OB1). La procédure reste la même.

Vous pouvez donner des noms symboliques aux différents blocs dans la table des mnémoniques (FB1 a par exemple pour nom "Moteur" et le DB1 le nom "Essence").

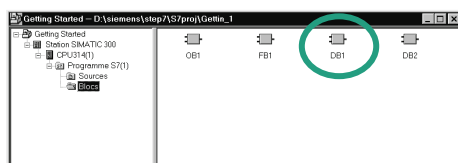
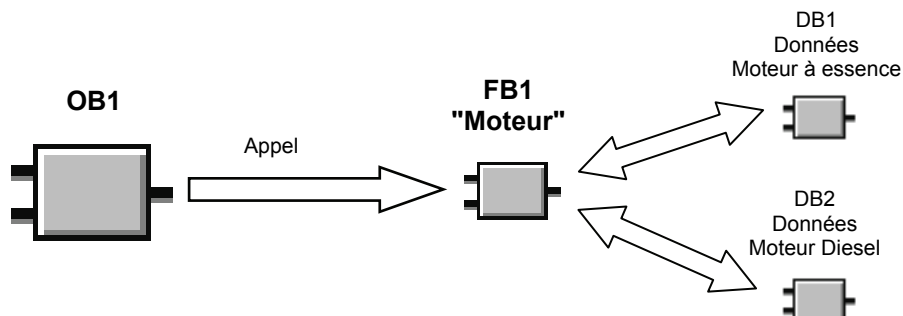
Les blocs programmés peuvent être à tout moment archivés ou imprimés. Vous trouvez les fonctions correspondantes dans SIMATIC Manager sous les commandes de menu

**Fichier > Archiver** ou **Fichier > Imprimer**.

Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage CONT" et "Gestion du programme" via la commande de menu ? > **Rubriques d'aide**.

## 5.7 Programmation d'un appel de bloc en LIST

L'entière programmation du bloc fonctionnel resterait sans effet si son appel n'était pas programmé dans l'OB1. Un bloc de données différent est utilisé à chaque appel du bloc fonctionnel commandant à chaque fois un moteur différent.



SIMATIC Manager est ouvert avec le projet "Getting Started".

Naviguez jusqu'au dossier **Blocs**, et ouvrez l'OB1.



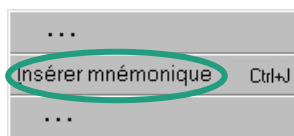
Sélectionnez dans la fenêtre de programme CONT/LIST/LOG le réseau 3 et insérez le réseau 4.

```
CALL "Moteur" , "Essence"
Switch_On      :=
Switch_Off     :=
Failure        :=
Actual_Speed   :=
Engine_On      :=
Preset_Speed_Reached :=
```

Entrez dans la section des instructions **CALL "Moteur", "Essence"** et appuyez sur la touche **Entrée**.

Tous les paramètres du bloc fonctionnel "Essence" sont affichés.

Positionnez le curseur après le signe d'égalité suivant Mise en marche et cliquez avec le bouton droit de la souris.



Sélectionnez dans le menu contextuel qui s'ouvre alors la commande **Insérer mnémonique**. Ceci fait s'afficher une liste déroulante.



MotEss_arrêt	BOOL	E	1.1	A
MotEss_Défaillance	BOOL	E	1.2	D
MotEss_marche	BOOL	E	1.0	M
MotEss_vitesse_activé	BOOL	A	5.2	C
MotEss_Vitesse_atteinte	BOOL	A	5.1	S

```

CALL "Moteur" , "Essence"
Switch_On      := "MotEss_marche"
Switch_Off     := "MotEss_arrêt"
Failure        := "MotEss_Défaillance"
Actual_Speed   := "MotEss_VitesseCourante"
Engine_On      := "Marche_MotEss"
Preset_Speed_Reached := "MotEss_Vitesse_atteinte"

CALL "Moteur" , "Diesel"
Switch_On      := "MotDies_Marche"
Switch_Off     := "MotDies_Arrêt"
Failure        := "MotDies_Arrêt"
Actual_Speed   := "MotDies_ViteessCourante"
Engine_On      := "Marche_MotDies"
Preset_Speed_Reached := "MotDies_Vitesse_atteinte"
    
```

Cliquez sur le mnémorique **MotEss\_marche**. Celui-ci vient s'insérer automatiquement avec les guillemets dans votre programme.

Affectez à toutes les variables du bloc fonctionnel l'opérande approprié que vous sélectionnez dans la liste déroulante.

Chaque variable se voit affecter le signal "MotEss\_xxx " signalant son appartenance au DB Moteur à essence.

Programmez dans un nouveau réseau l'appel du bloc fonctionnel "Moteur" (FB1) avec le bloc de données "Diesel" (DB2). Procédez comme pour l'appel précédent.



Enregistrez votre programme et fermez le bloc.

Si vous créez des structures de programme avec des OB, des FB et des DB, vous devez programmer l'appel d'un bloc subordonné (par exemple un FB1) dans le bloc supérieur (par exemple l'OB1). La procédure reste la même.

Vous pouvez donner des noms symboliques aux différents blocs dans la table des mnémoniques (FB1 a par exemple pour nom "Moteur" et le DB1 le nom "Essence").

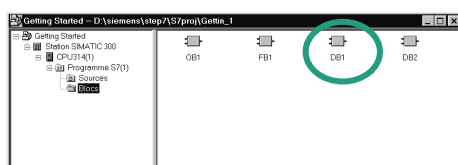
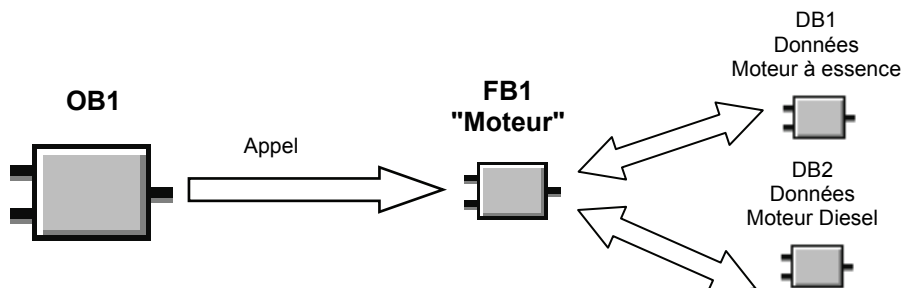
Les blocs programmés peuvent être à tout moment archivés ou imprimés. Vous trouvez les fonctions correspondantes dans SIMATIC Manager sous les commandes de menu **Fichier > Archiver** ou **Fichier > Imprimer**.

Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage LIST" et "Gestion du programme" via la commande de menu ? > **Rubriques d'aide**.



## 5.8 Programmation d'un appel de bloc en LOG

L'entière programmation du bloc fonctionnel resterait sans effet si son appel n'était pas programmé dans l'OB1. Un bloc de données différent est utilisé à chaque appel du bloc fonctionnel commandant à chaque fois un moteur différent.



SIMATIC Manager est ouvert avec le projet "Getting Started".

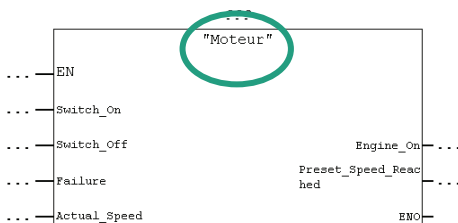
Naviguez jusqu'au dossier **Blocs**, et ouvrez l'OB1.



Sélectionnez dans la fenêtre de programme CONT/LIST/LOG le réseau 3 et insérez le réseau 4.

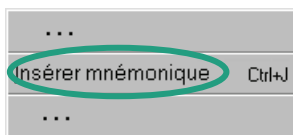


Naviguez ensuite dans l'onglet "Eléments de programme" jusqu'au **FB1** et insérez-le dans votre programme par double clic.



Toutes les variables d'entrée et de sortie spécifiques au moteur sont affichées.

Cliquez sur les points d'interrogation ??? au-dessus de "Moteur" et cliquez aussitôt avec le bouton droit de la souris dans le cadre de saisie.

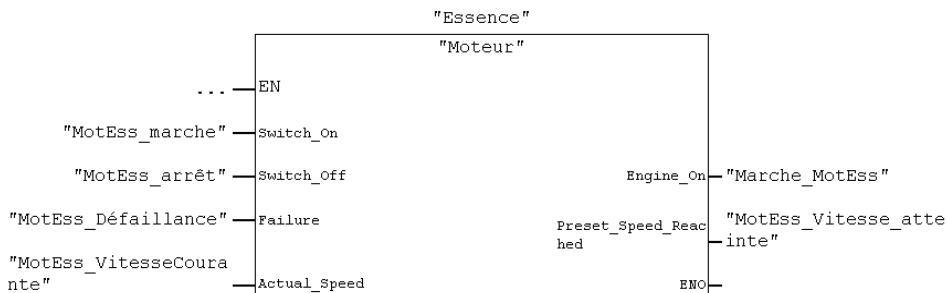


Sélectionnez dans le menu contextuel qui s'ouvre alors la commande **Insérer mnémonique**. Ceci fait s'afficher une liste déroulante.



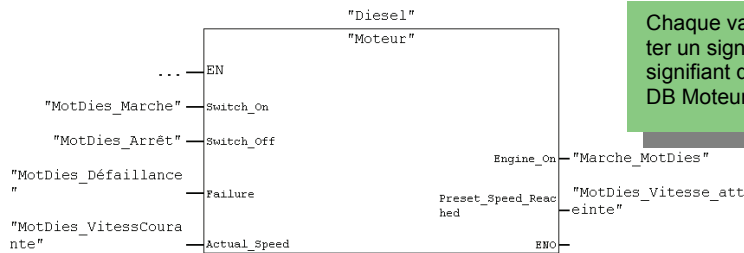
Double-cliquez dans la liste déroulante sur le bloc de données **Essence**. Il est automatiquement repris avec les guillemets dans le cadre de saisie.

Affectez de la même manière à chaque paramètre du bloc fonctionnel l'opérande symbolique approprié en le sélectionnant dans la liste déroulante.



Chaque variable se voit affecter un signal "MotEss\_xxx" signifiant son appartenance au DB Moteur à essence.

Programmez dans un nouveau réseau l'appel du bloc fonctionnel "Moteur" (FB1) avec le bloc de données "Diesel" (DB2) en sélectionnant à chaque fois l'opérande approprié dans la liste déroulante.



Enregistrez votre programme et fermez le bloc.

Si vous créez des structures de programme avec des OB, des FB et des DB, vous devez programmer l'appel d'un bloc subordonné (par exemple un FB1) dans le bloc hiérarchique supérieur (par l'exemple l'OB1). La procédure reste la même.

Vous pouvez donner des noms symboliques aux différents blocs dans la table des mnémoniques (FB1 a par exemple pour nom "Moteur" et le DB1 le nom "Essence").

Les blocs programmés peuvent être à tout moment archivés ou imprimés. Vous trouvez les fonctions correspondantes dans SIMATIC Manager sous les commandes de menu

**Fichier > Archiver** ou **Fichier > Imprimer**.

Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage LOG" et "Gestion du programme" via la commande de menu **? > Rubriques d'aide**.

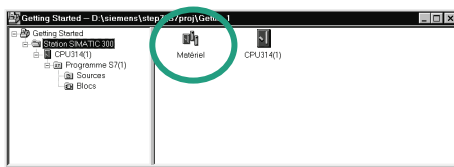


# 6 Configuration des unités centrales

## 6.1 Configuration matérielle

Pour pouvoir configurer le matériel, vous devez avoir au préalable créé un projet avec une station SIMATIC. La structure du projet créée à l'aide de l'Assistant de STEP 7 au chapitre 2.1 remplit toutes ces conditions.

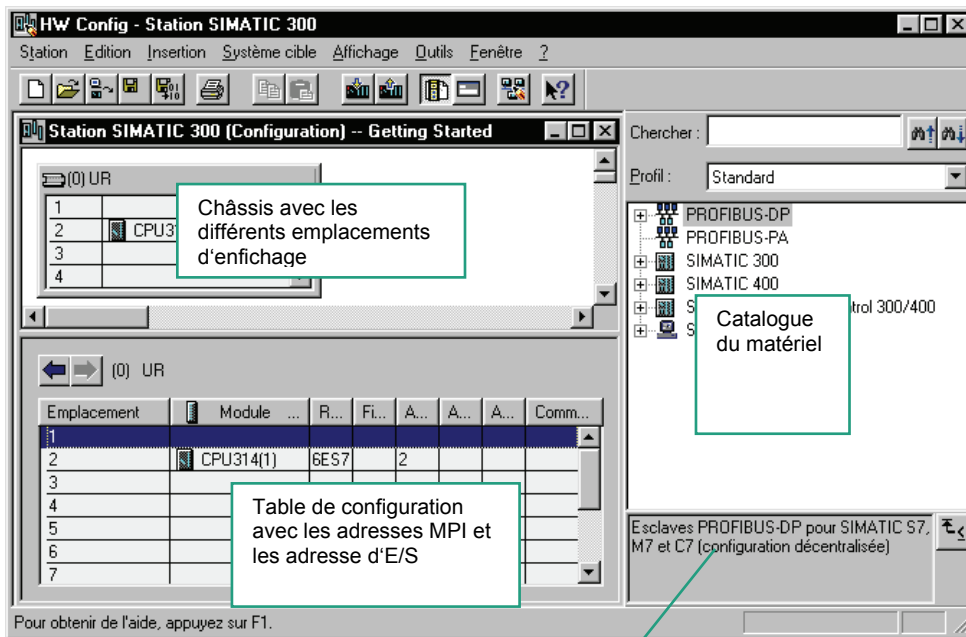
Vous configurez le matériel avec STEP 7. Ces données de configuration sont ensuite chargées (voir le chapitre 7 "Chargement") dans le système d'automatisation.



Le point de départ de la configuration est toujours SIMATIC Manager avec le projet "Getting Started" ouvert.

Ouvrez le dossier **Station SIMATIC 300**, et double-cliquez sur l'icône **Matériel**.

La fenêtre "HW Config" s'ouvre. La CPU qui a été sélectionnée à la création du projet est affichée. Il s'agit pour notre "Getting Started" de la CPU314.



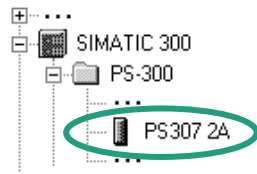
Châssis avec les différents emplacements d'enfichage

Table de configuration avec les adresses MPI et les adresse d'E/S

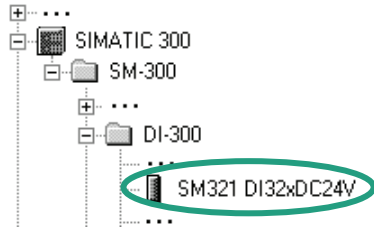
Catalogue du matériel

Esclaves PROFIBUS-DP pour SIMATIC S7, M7 et C7 (configuration décentralisée)

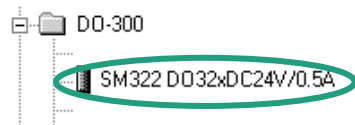
Information succincte sur l'élément sélectionné



Vous avez tout d'abord besoin d'un module d'alimentation. Naviguez dans le catalogue jusqu'au module d'alimentation **PS307 2A** et enfichez ce dernier par glisser-lâcher sur l'emplacement 1.



Sélectionnez ensuite un module d'entrées TOR (DI, Digital Input) **SM321 DI32xDC24V** et enfichez-le sur l'emplacement 4. L'emplacement 3 reste libre.

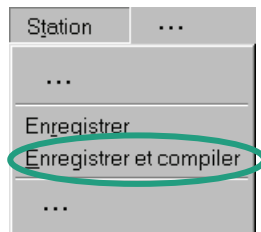


Enfichez de la même manière sur l'emplacement 5 le module de sorties **SM322 DO32xDC24V/0.5A**.

Pour modifier les paramètres (par exemple l'adresse) d'un module à l'intérieur d'un projet, il vous suffit d'ouvrir celui-ci par double-clic. Mais ne modifiez les paramètres que lorsque vous connaissez les répercussions que celles-ci peuvent avoir sur votre automate.

Aucune modification de paramètres n'est requise pour l'exemple de projet "Getting Started".

Emplacement	Module	Référence	Adresse MPI	Adresse d'entrée	A...	Commentaire
1	PS307 2A	6ES7 307-1BA00-0AA0				
2	CPU314(1)	6ES7 314-1AE04-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4...7	
6						
7						
8						
9						
10						
11						



Les données sont aussitôt préparées pour le transfert dans la CPU avec la commande **Enregistrer et compiler**.

Après avoir fermé "HW Config", vous pouvez voir une nouvelle icône dans le dossier Blocs. Il s'agit des Données système.



Vous pouvez en outre vérifier la configuration en choisissant la commande de menu **Station > Vérifier la cohérence** qui permet de rechercher les erreurs formelles dans le programme. STEP 7 vous offre diverses solutions si des erreurs sont trouvées.

Pour plus d'informations, référez-vous aux rubriques "Configuration du matériel" et "Configuration des unités centrales" via la commande de menu ? > **Rubriques d'aide**.

# 7 Chargement et test du programme

## 7.1 Etablir la liaison en ligne

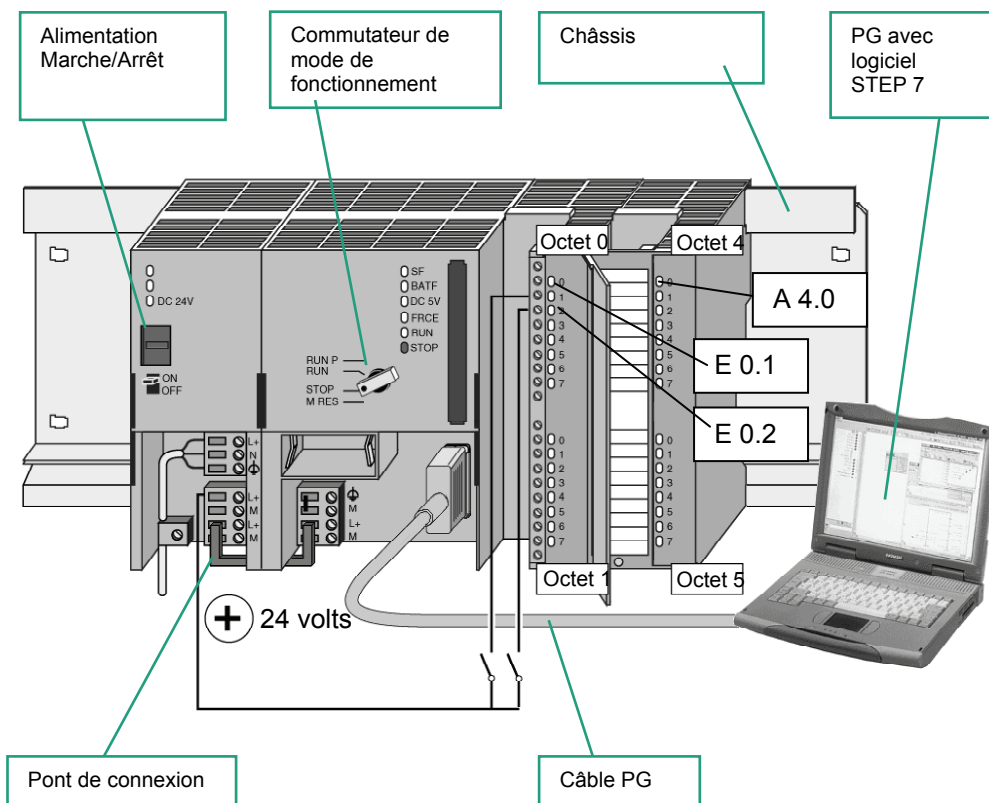
Nous allons vous montrer à l'aide du projet-exemple livré "zFr01\_06\_STEP7\_CONT\_1\_10" ou du projet déjà créé "Getting Started" et un montage de test simple comment charger votre programme dans le système d'automatisation (AP) pour ensuite le tester.

Vous devez avoir :

- configuré le matériel du projet "Getting Started" (voir le chapitre 6)
- monté le matériel conformément au manuel

Exemple de circuit série (fonction ET) :

La diode à la sortie A 4.0 du module de sorties TOR ne doit s'allumer que si les deux commutateurs E 0.1 et E 0.2 sont appuyés. Effectuez le montage de test en vous aidant de câbles et de la CPU.





### Monter le matériel

Pour monter un module sur le profilé support, procédez comme suit :

- Enfichez le module sur le connecteur du bus
- Accrochez le module et faites-le pivoter vers le bas
- Vissez à fond le module
- Montez les modules restants
- Après avoir monté tous les modules, enfichez la clé dans la CPU.



Le test peut être effectué avec un matériel différent de celui décrit ci-dessus. Seul l'adressage des entrées et sorties doit être conservé.

STEP 7 offre différentes possibilités de test telles que la visualisation du programme ou la table des variables.

Pour plus d'informations sur le montage des unités centrales, référez-vous aux manuels "S7-300 – Installation et configuration ; Caractéristiques de la CPU" ou "S7-400/M7-400 – Installation et configuration"



## 7.2 Chargement du programme dans le système cible

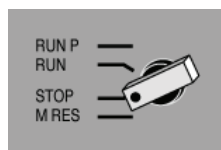
Le chargement du programme n'est possible que si une liaison en ligne à la CPU a été établie.



### Appliquer la tension

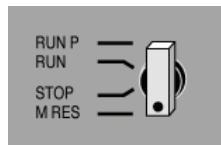


Appliquez la tension en activant le commutateur ON/OFF. La diode "DC 5V" s'allume sur la CPU.



Mettez le commutateur de mode de fonctionnement sur STOP (s'il ne s'y trouve pas déjà). La LED "STOP" s'allume en rouge.

### Effacement général de la CPU et passage à RUN



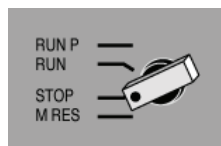
Mettez le commutateur de mode sur **MRES** et maintenez-le en cette position durant au moins 3 secondes jusqu'à ce que la LED "STOP" clignote en rouge.

L'effacement général efface toutes les données sur la CPU. La CPU se trouve maintenant dans son état initial.

Relâchez le commutateur de mode et remettez-le après 3 secondes maximum en position **MRES**. Quand la LED "STOP" clignote rapidement, la CPU a été remise à zéro.

Si ce n'est pas le cas, recommencez la procédure.

### Charger le programme dans la CPU

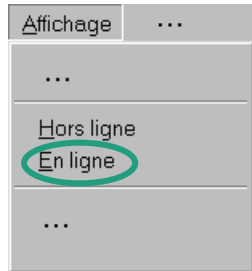


Pour charger le programme, le commutateur de mode doit à nouveau se trouver sur STOP.

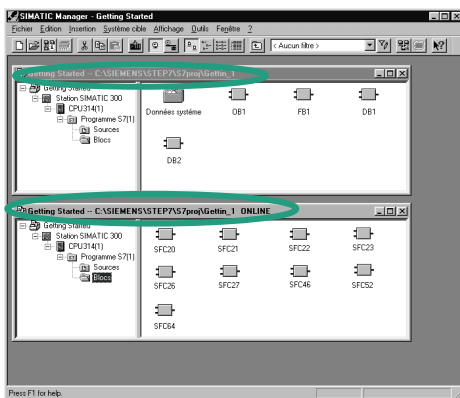




Démarrez SIMATIC Manager et ouvrez le projet "Getting Started" s'il n'est déjà ouvert via la boîte de dialogue "Ouvrir".



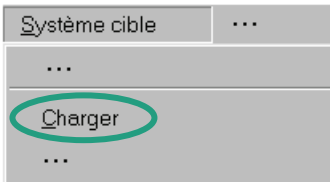
Appelez la vue en ligne du projet en plus de la vue hors ligne déjà ouverte. Vous pouvez les distinguer à leur barre de titre de couleur différente.



Naviguez dans les deux fenêtres jusqu'au dossier **Blocs**.

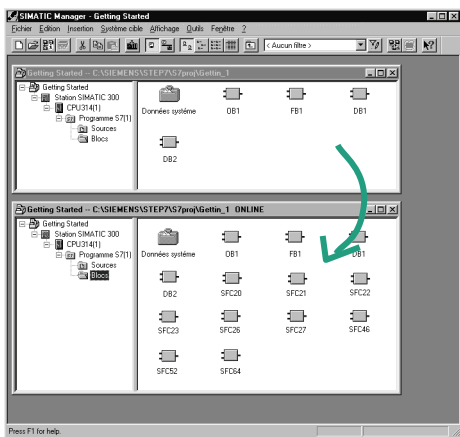
La fenêtre "Hors ligne" montre ce qui se passe sur la PG, la fenêtre "En ligne" montre ce qui se passe dans la CPU.

Des fonctions système (SFC) se trouvent encore sur la CPU même après l'effacement général de celle-ci. Ces fonctions sont mises à disposition par le système d'exploitation de la CPU. Elles n'ont pas besoin d'être chargées, mais



Sélectionnez le dossier **Blocs** dans la fenêtre "Hors ligne" et chargez ensuite le programme dans la CPU via la commande **Système cible > Charger**.

Répondez à la demande de confirmation par **OK**.

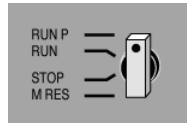


Après le chargement, les blocs du programme s'affichent dans la fenêtre "En ligne".

Vous pouvez encore appeler la commande **Système cible > Charger** via le bouton correspondant de la barre d'outils ou via le menu contextuel en cliquant avec le bouton droit de la souris.



## Mettre en marche la CPU et vérifier le mode de fonctionnement de celle-ci



Mettez le commutateur de mode sur **RUN-P**. La LED verte de "RUN" s'allume et la LED rouge de "STOP" s'éteint. La CPU est prête à fonctionner.

Lorsque la LED verte est allumée, vous pouvez commencer à tester votre programme.

Si la LED rouge ne s'éteint pas, c'est qu'il y a une erreur. Évaluez alors la mémoire tampon de diagnostic pour en rechercher la cause.

### Chargement de blocs isolés

Pour réagir rapidement dans la pratique aux erreurs, vous avez la possibilité de transférer par glisser-lâcher des blocs un par un dans la CPU.

Le commutateur de mode doit se trouver pour le chargement soit sur "RUN-P", soit sur "STOP". En mode "RUN" les blocs chargés sont aussitôt activés. Tenez compte ce faisant des points suivants :

L'écrasement de blocs exempts d'erreur par des blocs défectueux peut entraîner un mauvais fonctionnement de votre installation. Pour éviter ceci, testez vos blocs avant de les charger dans la CPU.

- Si l'ordre de chargement des blocs n'a pas été respecté – d'abord les blocs de niveaux inférieurs, ensuite les blocs de niveaux supérieurs – la CPU passe en STOP. Pour éviter ceci, chargez le programme entier dans la CPU.

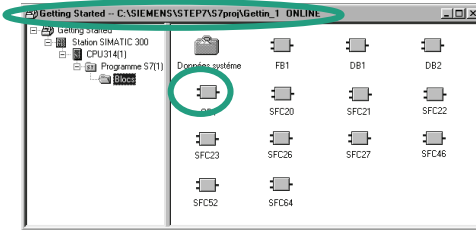
### Programmer en ligne

Il peut s'avérer dans la pratique nécessaire de modifier les blocs déjà chargés dans la CPU. Cliquez pour cela sur le bloc voulu dans la fenêtre "En ligne" pour ouvrir l'éditeur de programme CONT/LIST/LOG. Programmez ensuite le bloc comme vous êtes habitué à la faire. Notez que le bloc programmé est immédiatement activé dans la CPU.

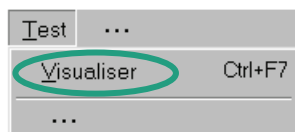
Pour plus d'informations, référez-vous aux rubriques "Chargement" et " Etablissement de liaisons en ligne et choix de la CPU" via la commande de menu ? > **Rubriques d'aide**.

### 7.3 Tester le programme avec la fonction de visualisation

La fonction de visualisation permet de tester le bloc d'un programme. Une liaison en ligne doit avoir été établie à la CPU, la CPU doit être en mode RUN ou RUN-P et le programme doit avoir été chargé dans la CPU.

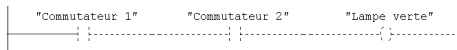


Ouvrez l'**OB1** dans la fenêtre en ligne du projet.  
L'éditeur de programme CONT/LIST/LOG s'ouvre.



Activer la fonction **Test > Visualiser**.

#### Tester avec CONT



Le circuit série du réseau 1 est affiché dans la vue CONT. Le trajet du courant est représenté par un trait continu, indiquant le passage de la tension.

#### Tester avec LIST

	RLG	ETA	STANDARD
U "Commutateur 1"	0	0	0
U "Commutateur 2"	0	0	0
= "Lampe verte"	0	0	0

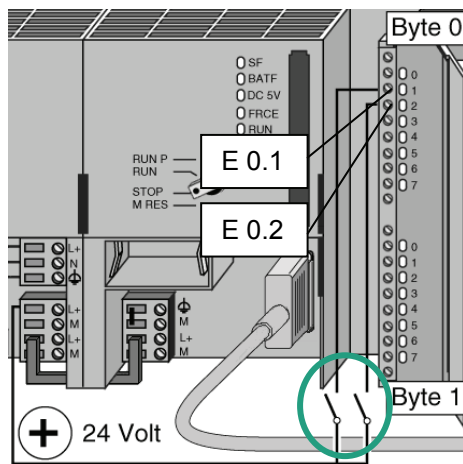
Dans LIST, les  
– résultat logique (RLG),  
– bit d'état (ETA) et  
– état standard (STANDARD)  
sont représentés sous la forme d'une table.

#### Tester avec LOG



L'état de signal est représenté dans LOG par un "0" ou un "1". Une ligne en pointillés signifie qu'il n'y a pas de résultat logique.

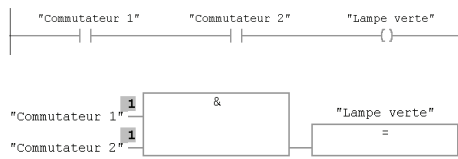
Vous pouvez modifier l'affichage du langage de programmation durant le test avec la commande **Outils > Paramètres**.



Fermez maintenant les deux commutateurs de votre circuit de test.

Les diodes aux entrées E 0.1 et E 0.2. du module d'entrées s'allument.

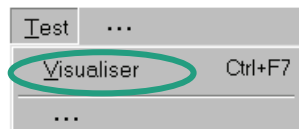
La diode de la sortie A 4.0 du module de sorties s'allume aussi.



	RLG	ETA	STANDARD
U "Commutateur 1"	1	1	0
U "Commutateur 2"	1	1	0
= "Lampe verte"	1	1	0

Dans les langages de programmation graphiques CONT et LOG, vous pouvez suivre le trajet du courant à l'écran et voir les valeurs du réseau programmé changer durant le test. Le changement de couleur indique que le résultat logique est rempli jusqu'ici.

Dans le langage de programmation LIST, vous voyez les valeurs changer dans les colonnes ETA et RLG lorsque le résultat logique est satisfait.



Désactivez la fonction **Test > Visualiser** et fermez la fenêtre.

Après quoi, vous pouvez refermer la fenêtre "En ligne" dans le SIMATIC Manager.

Nous vous recommandons de ne jamais charger, ni d'exécuter de programmes volumineux dans la CPU, la détection des erreurs étant rendue difficile par les multiples sources d'erreur. Il est recommandé pour un test plus rapide et efficace de charger et de tester les blocs un par un.

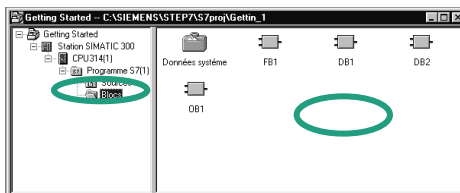
Pour plus d'informations, référez-vous aux rubriques "Test" et "Test avec la fonction de visualisation" via la commande de menu ? > **Rubriques d'aide**

## 7.4 Tester le programme avec la table des variables

Vous testez des variables isolées du programme en les visualisant et en les forçant. Il faut pour cela qu'une liaison en ligne à la CPU existe, que la CPU se trouve en mode RUN-P et que le programme soit chargé.

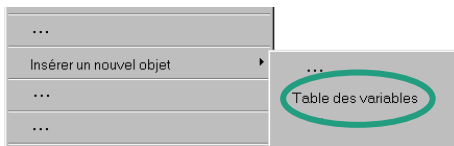
Comme dans la visualisation du programme, vous pouvez visualiser l'état des entrées et sorties du réseau 1 (circuit série ou fonction ET) dans la table des variables. Vous pouvez en outre tester le comparateur de vitesse de moteur du FB1 en entrant une vitesse réelle.

### Créer la table des variables

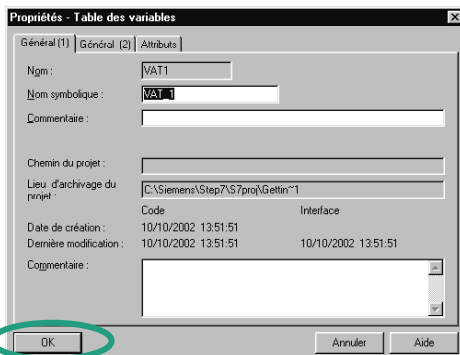


Vous devez pour cela vous trouver à nouveau dans la fenêtre du projet "Getting Started" ouverte hors ligne.

Naviguez jusqu'au dossier **Blocs**, et cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre.

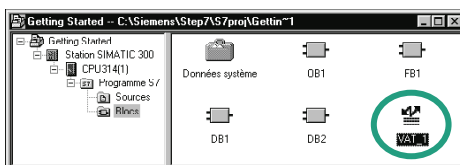


Sélectionnez dans le menu contextuel du bouton droit de la souris l'objet **Table des variables**.



Validez les options par défaut de la boîte de dialogue des propriétés avec **OK**.

Une autre méthode consiste à entrer un nom de mnémonique et à lui attribuer un commentaire.



Une table de variables (par défaut VAT1) est insérée dans le dossier Blocs.

Ouvrez **VAT1** par double clic. La fenêtre "Visualisation et forçage des variables" s'ouvre.



La table des variables est d'abord vide. Entrez les opérands et les mnémoniques pour l'exemple "Getting Started" comme représenté ci-dessous. Le programme complète les autres colonnes une fois que vous avez confirmé la saisie avec la touche Entrée.

Changez le format de la valeur d'état en format décimal pour toutes les valeurs de vitesse. Cliquez pour cela sur la cellule voulue et choisissez dans le menu contextuel (clic sur le bouton droit de la souris) le format DECIMAL.

	Opérande	Mnémonique	Format d'	Valeur d'état	Valeur de forçage
1	E 0.1	"Commutateur 1"	BOOLEEN		
2	E 0.2	"Commutateur 2"	BOOLEEN		
3	A 4.0	"Lampe verte"	BOOLEEN		
4					
5	MW 2	"MotEss_VitesseCourante"	DEC		
6	DB1.DBW 6	"Essence".Preset_Speed	DEC		
7	A 5.1	"MotEss_Vitesse_atteinte"	BOOLEEN		
8					
9	MW 4	"MotDies_VitessCourante"	DEC		
10	DB2.DBW 6	"Diesel".Preset_Speed	DEC		
11	A 5.5	"MotDies_Vitesse_atteinte"	BOOLEEN		
12					

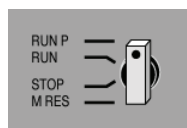


Enregistrez votre table des variables.

### Commuter la table des variables en ligne



Etablissez la liaison en ligne à la CPU configurée. Le mode de la CPU s'affiche dans la barre d'état.



Mettez le commutateur de mode de la CPU sur **RUN-P** (si vous ne l'avez pas encore fait).





### Visualiser les variables



Cliquez sur l'icône **Visualiser la variable**.

	Opérande	Mnémonique	Format d'	Valeur d'état	Valeur de forçage
1	E 0.1	"Commutateur 1"	BOOLEEN	true	
2	E 0.2	"Commutateur 2"	BOOLEEN	true	
3	A 4.0	"Lampe verte"	BOOLEEN	true	
4					
5	MW 2	"MotEss_VitesseCourante"	DEC	0	

Fermez les commutateurs 1 et 2 de votre circuit de test et observez le résultat dans la table des variables.

Les valeurs d'état passent de false à true dans la table des variables.

### Forcer des variables

Entrez la valeur "1500" pour l'opérande MW2 et "1300" pour l'opérande MW4 dans la colonne Valeur de forçage de la table des variables.

	Opérande	Mnémonique	Format d'	Valeur d'état	Valeur de forçage
1	E 0.1	"Commutateur 1"	BOOLEEN	true	
2	E 0.2	"Commutateur 2"	BOOLEEN	true	
3	A 4.0	"Lampe verte"	BOOLEEN	true	
4					
5	MW 2	"MotEss_VitesseCourante"	DEC	0	1500
6	DB1.DBW 6	"Essence".Preset_Speed	DEC	1500	
7	A 5.1	"MotEss_Vitesse_atteinte"	BOOLEEN	true	
8					
9	MW 4	"MotDies_VitessCourante"	DEC	0	1300
10	DB2.DBW 6	"Diesel".Preset_Speed	DEC	1200	
11	A 5.5	"MotDies_Vitesse_atteinte"	BOOLEEN	true	
12					



Transférez les valeurs de forçage dans votre CPU.





Après leur transfert, ces valeurs sont traitées par la CPU. Vous pouvez alors observer le résultat de la comparaison.

Fermez la fenêtre Visualisation et forçage des variables. Répondez à une demande de confirmation éventuelle par **Oui** ou par **OK**.

	Opérande	Mnémonique	Format d'	Valeur d'état	Valeur de forçage
1	E 0.1	"Commutateur 1"	BOOLEEN	true	
2	E 0.2	"Commutateur 2"	BOOLEEN	true	
3	A 4.0	"Lampe verte"	BOOLEEN	true	
4					
5	MW 2	"MotEss_VitesseCourante"	DEC	1500	1500
6	DB1.DBW 6	"Essence".Preset_Speed	DEC	1500	
7	A 5.1	"MotEss_Vitesse_atteinte"	BOOLEEN	true	
8					
9	MW 4	"MotDies_VitesseCourante"	DEC	1300	1300
10	DB2.DBW 6	"Diesel".Preset_Speed	DEC	1200	
11	A 5.5	"MotDies_Vitesse_atteinte"	BOOLEEN	true	
12					

MPI = 2 (direct)      MARCHÉ      Abs < 5.2



Il arrive fréquemment qu'une table des variables de taille importante ne puisse pas être affichée dans sa totalité à l'écran en raison des dimensions limitées de ce dernier.

Si vous avez de grandes tables de variables, vous vous recommandons d'en créer plusieurs avec STEP 7 pour un même programme S7. Vous pouvez ainsi créer vos tables de variables en fonction de vos besoins de test.

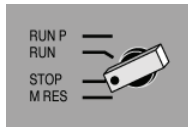
Vous pouvez leur donner comme aux blocs un nom individuel (par exemple le nom OB1\_Réseau1 au lieu de VAT1). Vous pouvez renommer vos tables VAT dans la table des mnémoniques.

Pour plus d'informations, référez-vous aux rubriques "Test" et "Tester avec la table des variables" via la commande de menu ? > **Rubriques d'aide**.

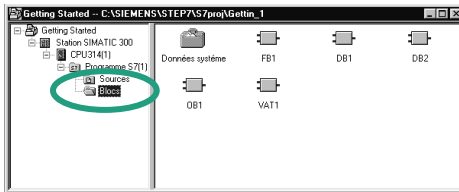
## 7.5 Evaluer la mémoire tampon de diagnostic

Pour le cas où la CPU passerait en STOP durant le traitement d'un programme S7 ou que la CPU ne se laisse plus commuter sur RUN après le chargement du programme, vous pouvez lire les messages du tampon de diagnostic pour rechercher la cause de l'erreur.

Il faut pour cela qu'une liaison en ligne à la CPU existe et que la CPU se trouve à l'état de fonctionnement "STOP".



Commutez d'abord la CPU sur "STOP" avec le commutateur de mode.

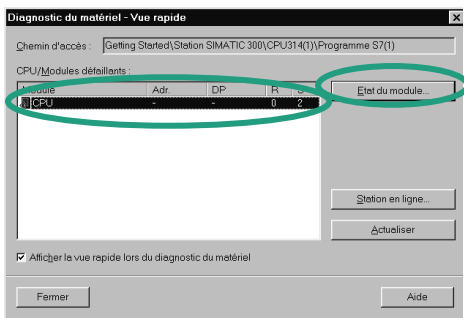


Le point de départ est de nouveau le projet "Getting Started" ouvert hors ligne dans SIMATIC Manager.

Sélectionnez le dossier **Blocs**.



S'il y a plusieurs CPU dans votre projet, vérifiez d'abord quelle est la CPU à l'arrêt.



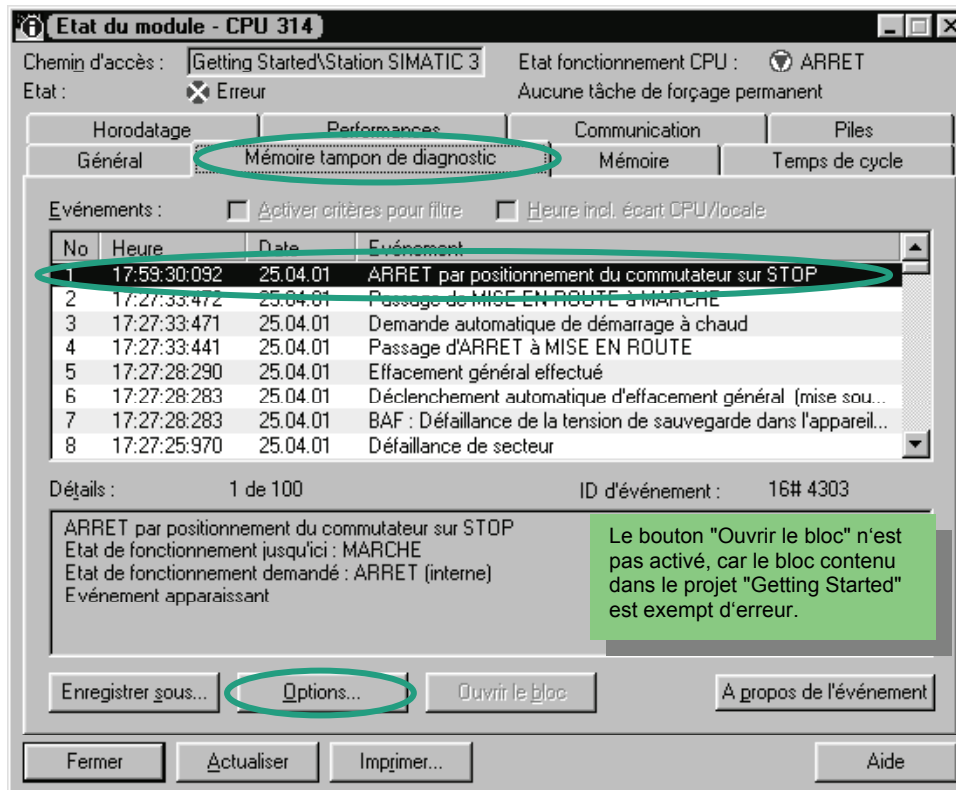
Toutes les CPU adressables sont affichées dans la boîte de dialogue "Diagnostic du matériel". La CPU qui se trouve en STOP y est sélectionnée.

Le projet "Getting Started" n'a qu'une CPU. Vous ne voyez donc qu'une CPU affichée.

Cliquez sur le bouton **Etat du module** pour lire la mémoire tampon de diagnostic de la CPU.

S'il n'y a qu'une CPU de connectée, vous pouvez lire directement l'état du module via la commande de menu **Système cible > Diagnostic/Réglage > Etat du module**.

La fenêtre "Etat du module" vous renseigne sur les propriétés et les paramètres de votre CPU. Cliquez à présent sur l'onglet **Mémoire tampon de diagnostic** pour rechercher la cause du passage à STOP de la CPU.



L'événement le plus récent (No 1) est toujours affiché dans la première ligne. La cause du passage à STOP est affichée. Fermez toutes les fenêtres, sauf celle de SIMATIC Manager.



Si une erreur de programmation est à l'origine du passage à STOP de la CPU, sélectionnez l'événement et cliquez sur le bouton **Ouvrir le bloc**.

Le bloc est alors ouvert dans l'éditeur CONT/LIST/LOG connu et le réseau dans lequel se trouve l'erreur est affiché.

Vous avez dans ce chapitre achevé de créer et de tester le projet-exemple "Getting Started". Dans les chapitres suivants, vous pourrez approfondir votre savoir par des exercices choisis.

Pour plus d'informations, référez-vous aux rubriques "Diagnostic" et "Fonctions de renseignements sur l'état du module" via la commande de menu ? > **Rubriques d'aide**.



## 8 Programmation d'une fonction (FC)

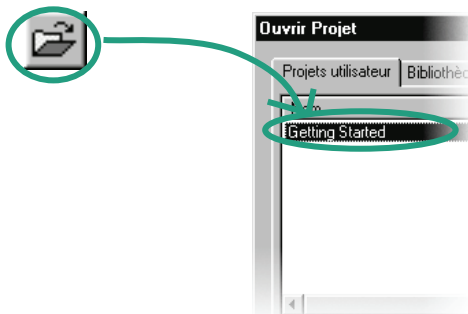
### 8.1 Créer et ouvrir une fonction

La fonction est comme le bloc fonctionnel subordonnée au bloc d'organisation. Afin qu'elle puisse être traitée par la CPU, il faut également l'appeler dans le bloc supérieur. A l'opposé du bloc fonctionnel, elle n'a pas besoin de bloc de données.

Les paramètres de la fonction sont aussi déclarés dans la table de déclaration des variables, mais les données locales statiques ne sont pas autorisées.

Vous programmez la fonction comme le bloc fonctionnel dans l'éditeur de programme CONT/LIST/LOG.

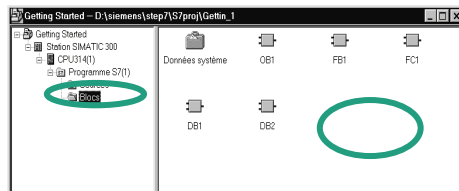
Vous devez déjà être familiarisé avec la programmation en CONT, LIST ou LOG (voir les chapitres 4 et 5) et la programmation symbolique (voir le chapitre 3).



Si vous avez exécuté le projet -exemple "Getting Started" (chapitres 1 – 7), ouvrez-le à présent.

Créez sinon un nouveau projet dans SIMATIC Manager avec **Fichier > Assistant "Nouveau projet"**. Procédez comme décrit dans la paragraphe 2.1 et nommez le projet "Fonction Getting Started".

Nous parlerons dans la suite de ce chapitre du projet "Getting Started". Mais vous pouvez exécuter chacune des étapes décrites avec n'importe quel nouveau projet.

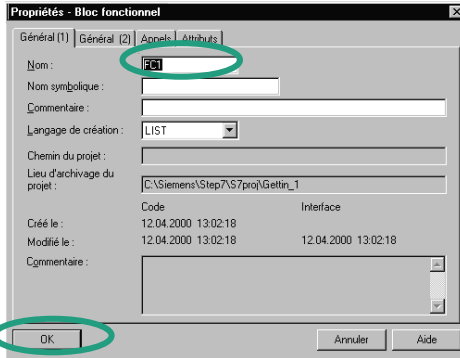


Naviguez jusqu'au dossier **Blocs** et ouvrez-le.

Cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre.

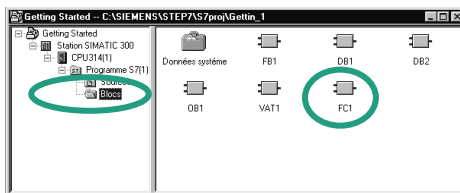


Insérez avec le menu contextuel du bouton droit de la souris une **Fonction** (FC).



Validez le nom FC1 dans la boîte des propriétés de la fonction et sélectionnez votre langage de création.

Confirmez les options restantes avec **OK**.



La fonction FC1 a été insérée dans le dossier Blocs.

Ouvrez la fonction **FC1** par un double clic.

A l'opposé du bloc fonctionnel, il n'est pas possible de définir de données statiques dans la table de déclaration des variables d'une fonction.

Les données statiques définies dans un bloc fonctionnel sont conservées après le traitement du bloc. Il peut s'agir par exemple des mémentos utilisés pour les valeurs limites de "Vitesse" (voir chapitre 5).

Vous pouvez, comme vous en avez l'habitude, avoir recours aux mnémoniques de la table des mnémoniques pour programmer la fonction.

Pour plus d'informations, référez-vous aux rubriques "Elaboration du concept d'automatisation", "Conception d'une structure du programme" et "Blocs dans le programme utilisateur" via la commande de menu ? > **Rubriques d'aide**.

## 8.2 Programmer la fonction

Nous allons programmer dans notre exemple une fonction de temporisation. Celle-ci aura pour fonction d'activer parallèlement à la mise en marche du moteur un ventilateur qui continuera à fonctionner (retard à la retombée) durant quatre secondes après l'arrêt du moteur.

Il nous faut préalablement déclarer les paramètres d'entrée et sortie (déclaration "in" et "out") de la fonction dans la vue de détail des variables.

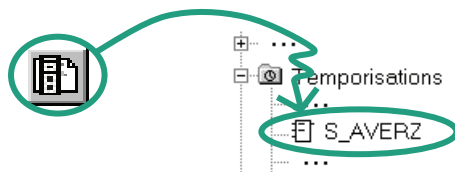
Vous avez ouvert pour cela la fenêtre de l'éditeur de programme CONT/LIST/LOG. Vous remplissez cette vue de détail des variables comme vous l'avez fait pour le bloc fonctionnel (voir chapitre 5).

Entrez les déclarations suivantes.

Contenu de : 'Environnement\Interface\IN'			
	Nom	Type de données	Commentaire
+	Engine_On	Bool	Signal de la mise en marche du moteur
+	Timer_Function	Timer	Temporisation utilisée pour le retard à la retombée

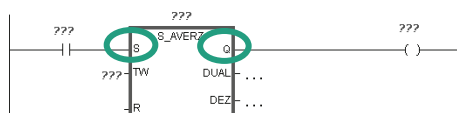
Contenu de : 'Environnement\Interface\OUT'			
	Nom	Type de données	Commentaire
+	Fan_On	Bool	Signal de mise en route du ventilateur

### Programmer une fonction de temporisation en CONT



Sélectionnez la branche de courant afin d'y insérer le nouvel élément CONT.

Naviguez dans le catalogue des éléments de programme jusqu'à l'élément **S\_AVERZ** (temporisation sous forme de retard à la retombée) et insérez-le dans le réseau.



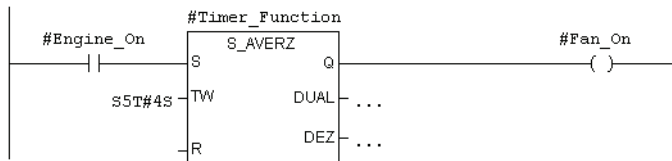
Insérez un contact à fermeture avant l'entrée **S** et ajoutez une bobine après la sortie **Q**.



Sélectionnez les points d'interrogation, entrez un caractère "#" et choisissez le nom voulu.

Entrez le temps de retard de la temporisation à l'entrée TW de S\_AVERZ. Une constante au type de données S5Time# (S5T#) d'une durée de 4 secondes (4s) sera par exemple définie comme suit : S5T#4s.

Enregistrez ensuite la fonction et fermez l'éditeur.



Le paramètre d'entrée "#Moteur\_Marche" permet de lancer la "#Fonction de temporisation". Celle-ci recevra à son appel dans l'OB1 d'abord les paramètres du moteur à essence, puis les paramètres du moteur Diesel (par exemple T1 pour "Retard\_MotEss"). Les mnémoniques de ces paramètres devront encore être entrés dans la table des mnémoniques.

### Programmer une fonction de temporisation en LIST

<pre> U   #Engine_On L   S5T#4s SA  #Timer_Function U   #Timer_Function =   #Fan_On                 </pre>	<p>Si vous programmez en LIST, sélectionnez la zone de saisie du réseau et entrez l'instruction ci-contre.</p> <p>Enregistrez ensuite la fonction et fermez l'éditeur.</p>
--	--



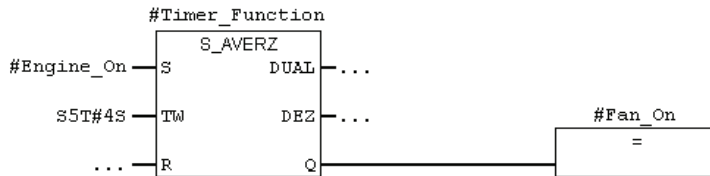




## Programmer une fonction de temporisation en LOG

Si vous programmez en LOG, sélectionnez la zone de saisie du réseau et entrez le programme LOG ci-dessous pour la fonction de temporisation.

Enregistrez ensuite la fonction et fermez l'éditeur.



Il faut encore programmer l'appel de la fonction dans le bloc supérieur (dans notre exemple l'OB1) si l'on veut que la temporisation soit exécutée dans le programme.

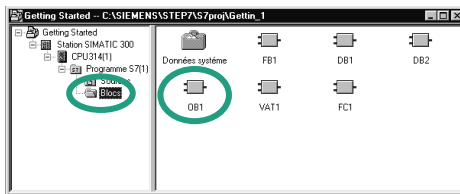
Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage CONT/LOG/LIST" et "Temporisations" via la commande de menu ? > **Rubriques d'aide**.

### 8.3 Appel de la fonction dans l'OB1

L'appel de la fonction FC1 dans l'OB1 est similaire à celui du bloc fonctionnel. Les paramètres de la fonction reçoivent dans l'OB1 les opérandes correspondants du moteur à essence ou Diesel.

Comme nous n'avons pas encore défini ces opérandes dans la table des mnémoniques, nous allons le faire maintenant.

Dans une instruction STEP 7, l'opérande est l'élément sur lequel l'opération du processeur doit porter. L'adressage d'opérandes peut être absolu ou symbolique.



SIMATIC Manager est ouvert avec le projet "Getting Started" ou le projet nouvellement créé.

Naviguez jusqu'au dossier **Blocs** et ouvrez l'**OB1**.

La fenêtre de l'éditeur de programme CONT/LIST/LOG s'ouvre.

#### Programmer l'appel de la fonction dans LOG

Vous vous trouvez dans la vue **CONT**. Sélectionnez le réseau 5 et insérez un nouveau réseau (No 6).



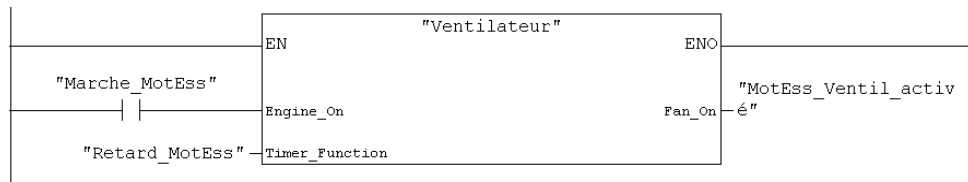
Naviguez ensuite dans le catalogue des éléments de programme jusqu'à la fonction FC1, et insérez la fonction dans votre réseau.



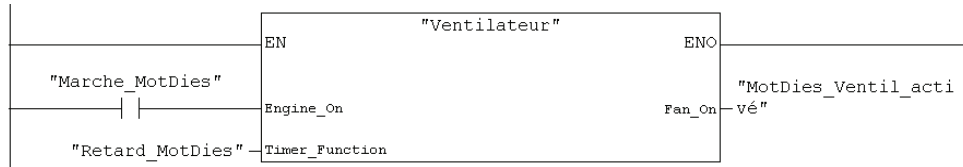
Insérez avant "Moteur\_Marche" un contact à fermeture.

Vous pouvez passer de l'affichage absolu à l'affichage symbolique avec la commande de menu **Affichage > Afficher avec > mnémoniques**.

Cliquez sur les points d'interrogation de l'appel FC1 et entrez les mnémoniques suivants.



Programmez un appel de fonction FC1 dans le réseau 7 avec les opérands du moteur Diesel. Procédez ce faisant comme pour le réseau précédant (les opérands du moteur Diesel doivent déjà avoir été entrés dans la table des mnémoniques).



Enregistrez le bloc et fermez l'éditeur.

Affichez les informations mnémonique avec la commande de menu **Affichage > Afficher avec > informations mnémonique**.

Pour voir plusieurs réseaux en même temps sur l'écran, masquez les commentaires avec la commande **Affichage > Afficher avec > commentaires** et les informations mnémonique avec la commande **Affichage > Afficher avec > Informations mnémonique**.

Vous pouvez changer l'échelle de représentation des réseaux avec la commande de menu **Affichage > Facteur d'agrandissement**.





### Programmer un appel de fonction en LIST

**Réseau 6 :** Ventilateur pour moteur à essence

```
CALL "Ventilateur"
Engine_On := "Marche_MotEss"
Timer_Function := "Retard_MotEss"
Fan_On := "MotEss_Ventil_activé"
```

**Réseau 7 :** Ventilateur pour moteur diesel

```
CALL "Ventilateur"
Engine_On := "Marche_MotDies"
Timer_Function := "Retard_MotDies"
Fan_On := "MotDies_Ventil_activé"
```

Si vous programmez en LIST, sélectionnez la zone de saisie d'un nouveau réseau et entrez les instructions LIST suivantes.

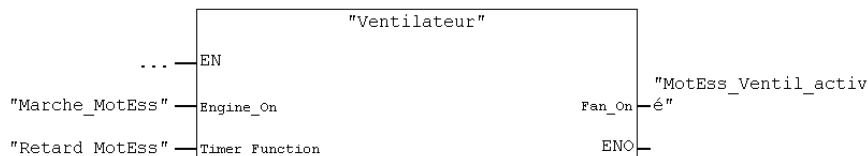
Enregistrez ensuite l'appel de fonction et fermez l'éditeur.

### Programmer un appel de fonction en LOG

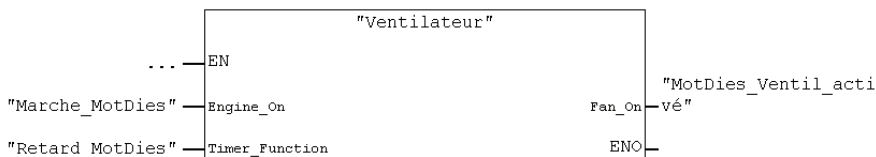
Si vous programmez en LOG, sélectionnez la zone de saisie d'un nouveau réseau et entrez les instructions LOG suivantes.

Enregistrez ensuite l'appel de fonction et fermez l'éditeur.

**Réseau 6 :** Ventilateur pour moteur à essence



**Réseau 7 :** Ventilateur pour moteur diesel



L'appel des fonctions a été programmé dans notre exemple comme un appel inconditionnel, cela signifie que la fonction sera toujours exécutée.

Vous pouvez si vous en avez besoin dans votre tâche d'automatisation programmer les appels de FC et FB en les faisant dépendre de conditions : en les reliant par exemple à une entrée ou à un autre circuit en amont. Vous programmez les conditions à l'entrée EN ou à la sortie ENO de la boîte de la fonction.

Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage CONT/LOG/LIST" via la commande de menu ? > **Rubriques d'aide.**

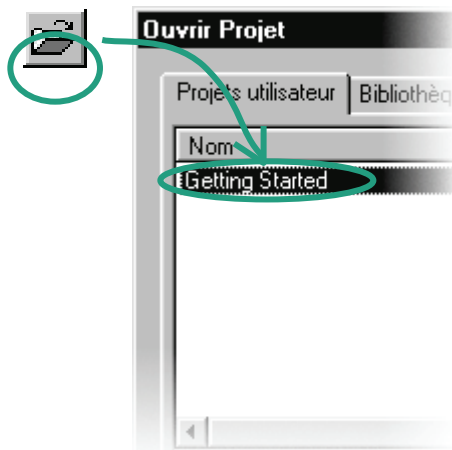
## 9 Programmation d'un bloc de données global

### 9.1 Créer et ouvrir un bloc de données global

Si le nombre des mémentos internes (cellules de mémoire) d'une CPU ne suffit plus à intégrer le stock de données, vous avez la possibilité d'archiver des données sélectionnées dans un bloc de données global.

Les données du bloc de données global sont mises à disposition de tous les autres blocs. Un bloc de données d'instance en revanche est affecté à un bloc fonctionnel précis, ses données ne sont disponibles que dans ce bloc fonctionnel (voir le paragraphe 5.5), c'est-à-dire qu'elles sont locales.

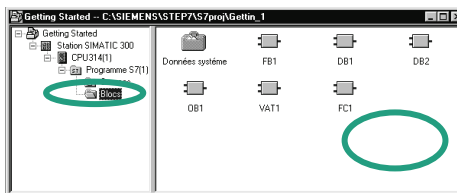
Vous devez déjà être familiarisé avec la programmation en CONT, LIST ou LOG (voir les chapitres 4 et 5) et avec la programmation symbolique (voir le chapitre 3).



Si vous avez exécuté le projet-exemple "Getting Started" (chapitres 1-7), ouvrez-le à présent.

Créez sinon un nouveau projet dans SIMATIC Manager avec **Fichier > Assistant "Nouveau projet"**. Procédez comme décrit dans le paragraphe 2.1 et nommez le projet ainsi créé "DB global Getting Started".

Nous parlerons dans la suite de ce chapitre du projet "Getting Started". Mais vous pouvez exécuter chaque étape avec n'importe quel nouveau projet.

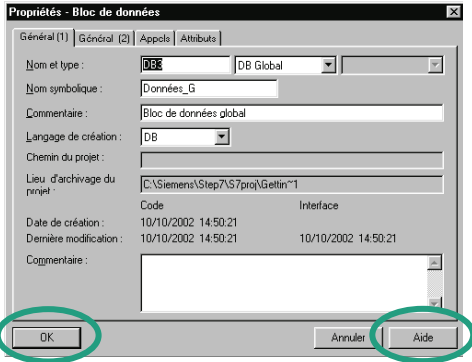


Naviguez jusqu'au dossier **Blocs** et ouvrez-le.

Cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre.



Insérez à partir du menu contextuel un **bloc de données**.



Validez les options par défaut dans la boîte de dialogue des propriétés du bloc avec **OK**.

Appelez l'aide sur cette boîte de dialogue pour plus d'informations.

Le bloc de données DB3 a été inséré dans le dossier **Blocs**.

Ouvrez le **DB3** avec un double clic.

Dans la boîte de dialogue "Nouveau bloc de données" qui s'ouvre alors, activez l'option **Bloc de données**. Fermez la boîte de dialogue par **OK**.

Rappel :  
 Dans le paragraphe 5.5 vous avez créé un bloc de données d'instance en sélectionnant l'option "Bloc de données associé à un bloc fonctionnel". Avec l'option "Bloc de données", vous créez un bloc de données global.

### Programmer des variables dans

Adresse	Nom	Type	Valeur initiale	Commentaire
0.0		STRUCT		
+0.0	DB VAR	INT	0	Variable temporaire de réservation

Entrez dans la colonne du nom "MotEss\_Vitesse\_courante".

Faites dérouler le menu contextuel **Type de données > simple > INT**.

A titre d'exemple, trois données globales ont été définies dans le DB3. Déclarez ces données dans la table de déclaration des variables comme dans le tableau ci-dessous.

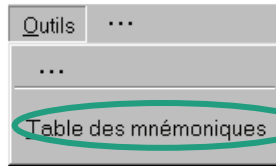
Adresse	Nom	Type	Valeur initiale	Commentaire
0.0		STRUCT		
+0.0	PE_Actual_Speed	INT	0	Vitesse courante du moteur à essence
+2.0	DE_Actual_Speed	INT	0	Vitesse courante du moteur diesel
+4.0	Preset_Speed_Reached	BOOL	FALSE	Les deux moteurs ont atteint la vitesse prescrite
=6.0		END_STRUCT		

Les variables des vitesses courantes du bloc de données "Vitesse\_courante\_MotEss" et "Vitesse\_courante\_MotDies" sont traitées de la même manière que les mots de memento MW2 (Vitesse\_courante\_MotEss) et MW4 (Vitesse\_courante\_MotDies). Ceci sera montré au chapitre suivant.



Enregistrez le bloc de données global.

## Affectation de mnémoniques



Il est également possible de donner un nom symbolique aux blocs de données.

Ouvrez la **Table de mnémoniques** et entrez pour le bloc de données DB3 le mnémonique "Données\_G".

Si vous avez copié au chapitre 4 la table des mnémoniques d'un projet-exemple (Exemple\_CONT, exemple\_LIST ou Exemple\_LOG) dans votre projet "Getting Started", vous n'avez pas besoin de définir de mnémoniques.

Mnémonique	Opérande	e de donn	Commentaire
...	...	...	...
Données_G	DB 3	DB 3	Bloc de données global



Enregistrez la table des mnémoniques et fermez l'éditeur de mnémoniques.

Fermez également le bloc de données global.



### DB global dans la table de déclaration des variables :

Avec **Affichage > Vue des données** vous pouvez voir les valeurs courantes de type INT changer dans la table du bloc de données global (se référer au chapitre 5.5).

### DB global dans la table des mnémoniques :

A l'opposé du DB d'instance, le type de données du DB global est toujours l'adresse absolue, dans notre exemple le type de données DB3. Le type de données du bloc de données d'instance est en revanche le bloc FB associé.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Création de blocs de données" via la commande de menu ? > **Rubriques d'aide**.





# 10 Programmation d'un bloc multiinstance

## 10.1 Créer et ouvrir un bloc fonctionnel

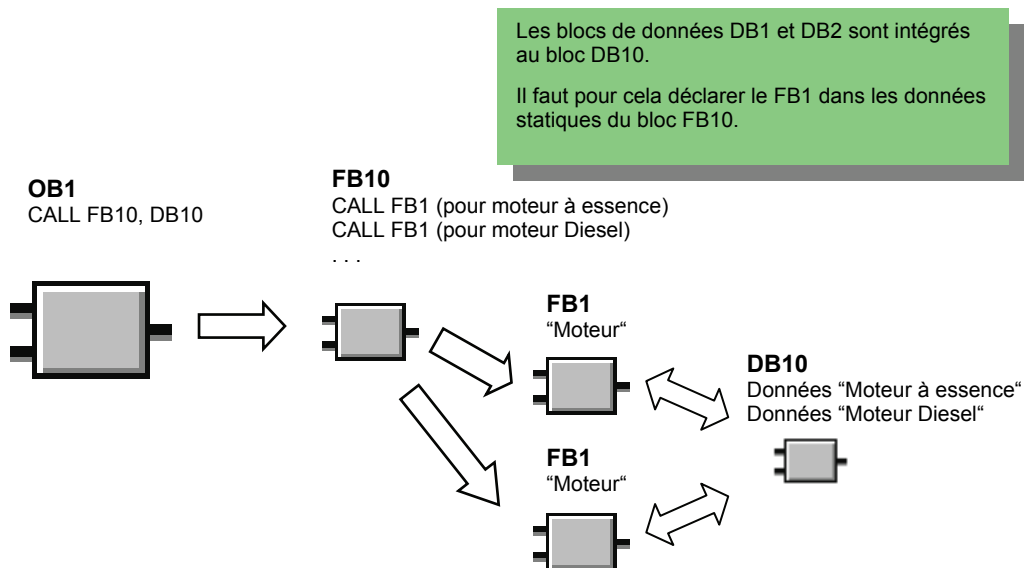
Vous avez programmé dans le chapitre 5 une commande de moteur à l'aide du bloc fonctionnel "Moteur" (FB1). Les blocs de données "Essence" (DB1) et "Diesel" (DB2) étaient utilisés lors de l'appel du bloc fonctionnel FB1 dans le bloc d'organisation OB1.

Chaque bloc de données contenait les données spécifiques à chaque moteur (par exemple #Vitesse\_Prescrite).

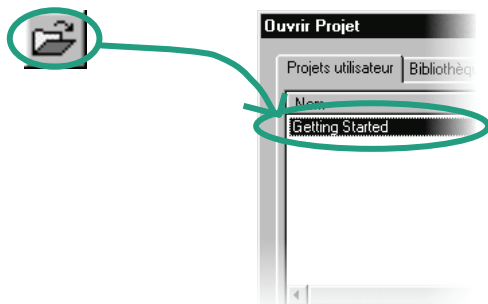
Imaginons-nous maintenant que notre tâche d'automatisation ait à commander d'autres moteurs, par exemple un moteur à l'huile de colza ou un moteur à hydrogène etc.

En appliquant la méthode pratiquée jusqu'ici, vous affecteriez pour chaque nouveau moteur un nouveau DB avec les données de ce moteur au FB utilisé jusqu'ici. Un DB3 pour commander le moteur à huile de colza et un DB4 pour commander le moteur à hydrogène etc. Le nombre de blocs augmenterait alors avec chaque nouvelle commande de moteur.

Vous pouvez réduire le nombre de blocs en utilisant un bloc multiinstance. Créez pour cela un nouveau bloc FB (dans notre exemple le bloc FB10) et appelez dans celui-ci le bloc FB1 tel qu'il est comme "instance locale". Le bloc FB1 transfère à chaque appel ses données dans le bloc de données DB10 du bloc supérieur FB10. Ainsi, il n'y a plus besoin d'affecter différents DB au bloc fonctionnel. Tous les FB, s'il y en a plusieurs, utilisent un seul bloc de données (ici le DB10).

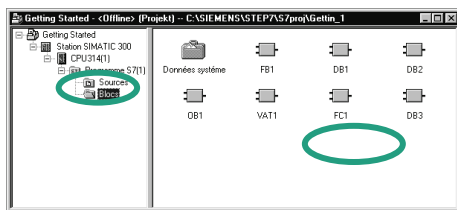


Vous devez déjà être familiarisé avec la programmation en CONT, LIST ou LOG (voir les chapitres 4 et 5) et avec la programmation symbolique (chapitre 3).



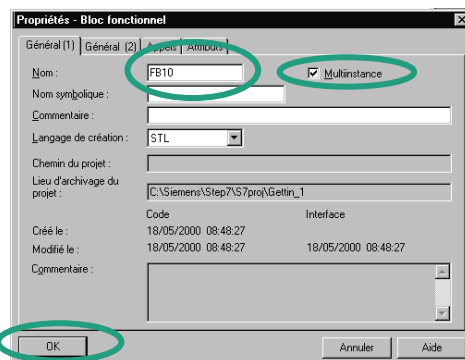
Si vous avez exécuté l'exemple "Getting Started" (chapitres 1–7), ouvrez le projet "Getting Started".

Si ce n'est pas le cas, ouvrez dans SIMATIC Manager le projet ZFr01\_05\_STEP7\_CONT, ZFr01\_01\_STEP7\_LIST\_1-9 ou ZFr01\_03\_STEP7\_LOG\_1-9.



Naviguez ensuite jusqu'au dossier **Blocs** et ouvrez-le.

Cliquez avec le bouton droit de la souris dans la partie droite de la fenêtre et sélectionnez un bloc fonctionnel dans le menu contextuel.



Nommez ce bloc "FB10" et choisissez votre langage de création.

Activez si elle ne l'est déjà la case d'option **Multiinstance**, et confirmez le reste des options avec **OK**.

Le bloc **FB10** a été inséré dans le dossier Blocs. Double-cliquez sur celui-ci pour l'ouvrir.

Vous pouvez créer des multiinstances pour tous les types de blocs fonctionnels, même pour les commandes de valves par exemple. Sachez si vous utilisez ce genre de bloc, que non seulement les blocs fonctionnels appelés mais également les blocs appelants peuvent avoir des multiinstances.

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs" et "Création de blocs et de bibliothèques" via la commande de menu ? > **Rubriques d'aide**.

## 10.2 Programmer le bloc FB10

Une variable statique à laquelle on donnera à chaque fois un nom différent est déclarée dans la table de déclaration des variables pour chaque appel prévu du bloc FB1 que l'on veut appeler comme instance locale dans le bloc FB10. On inscrira dans la colonne du type de données FB1 ("Moteur").

### Déclarer ou définir les variables

Le FB 10 est ouvert dans la fenêtre de programme CONT/LIST/LOG. Reportez les déclarations de la figure suivante dans la vue de détail des variables. Choisissez pour cela successivement dans la vue d'ensemble des variables les types de déclaration "OUT", "STAT" et "TEMP" et faites vos entrées dans la vue de détail. Choisissez pour le type de déclaration "STAT" le type de données "<No> FB" dans la liste déroulante et remplacez la suite de caractères "<No>" par le chiffre "1".

Contenu de : 'Environnement\Interface\OUT'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Preset_Speed_Reached	Bool	0.0	FALSE	Les deux moteurs ont atteint la vitesse prescrite

Contenu de : 'Environnement\Interface\STAT'

Nom	Type de données	Adresse	Valeur initiale	Commentaire
Petrol_Engi...	Moteur	2.0		Première instance locale du FB 1 "Moteur"
Diesel_Eng...	Moteur	2.0		Deuxième instance locale du FB 1 "Moteur"

Contenu de : 'Environnement\Interface\TEMP'

Nom	Type de données	Adresse	Commentaire
PE_Preset_Speed_Reached	Bool	0.0	Vitesse prescrite atteinte (moteur à essence)
DE_Preset_Speed_Reached	Bool	0.1	Vitesse prescrite atteinte (moteur diesel)

Les instances locales déclarées apparaissent ensuite dans le catalogue des éléments de programme sous la rubrique "Multiinstances".

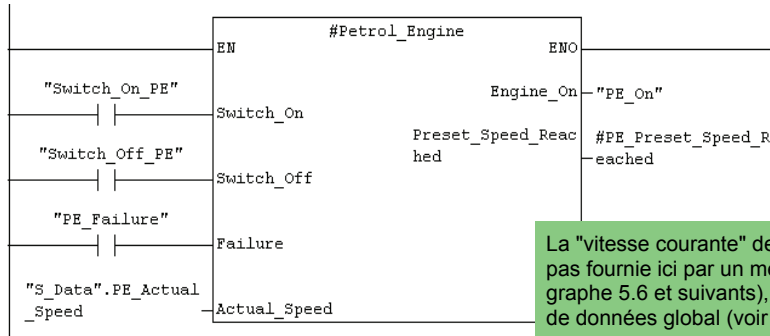


### Programmer le FB10 en CONT



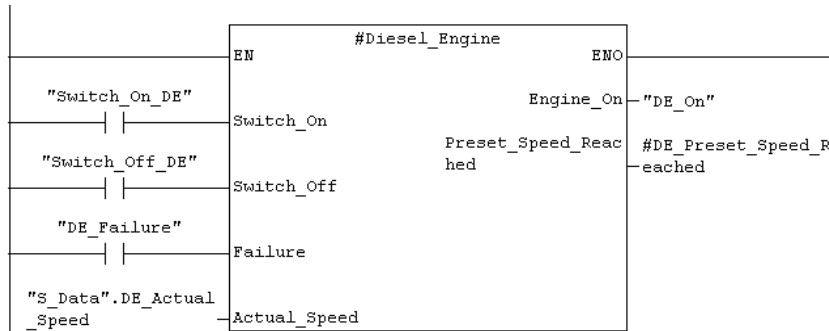
Insérez l'appel du bloc "MotEss" comme bloc multiinstance dans le réseau 1.

Insérez ensuite le contact à fermeture qui manque encore et remplacez les points d'interrogation par les mnémoniques.



La "vitesse courante" des moteurs n'est pas fournie ici par un memento (voir paragraphe 5.6 et suivants), mais par un bloc de données global (voir paragraphe 9.1).  
Les règles d'adressage sont les suivantes :  
"NomDBglobal".Opérande, par ex.  
"Données\_G".MotEss\_Vitesse\_courante

Insérez un nouveau réseau et programmez l'appel du moteur Diesel. Procédez pour cela comme pour le réseau 1.



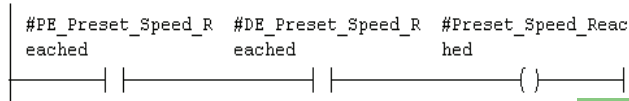


Insérez un nouveau réseau et programmez un circuit série, puis complétez ses adresses. Enregistrez ensuite votre programme et fermez le bloc.



Utilisez à chaque fois les variables temporaires. Vous reconnaissez celles-ci au symbole ci-contre.

Enregistrez votre programme et fermez le bloc.



Les variables temporaires ("MotEss\_Vitesse\_atteinte" et "MotDies\_Vitesse\_atteinte") sont transférées au paramètre de sortie "Vitesse\_atteinte" qui est ensuite traité dans l'OB1.

### Programmer le FB10 avec LIST

```
CALL #Petrol_Engine
  Switch_On      := "Switch On PE"
  Switch_Off     := "Switch Off PE"
  Failure        := "PE Failure"
  Actual_Speed   := "S_Data".PE_Actual_Speed
  Engine_On      := "PE On"
  Preset_Speed_Reached := #PE_Preset_Speed_Reached

CALL #Diesel_Engine
  Switch_On      := "Switch On PE"
  Switch_Off     := "Switch Off PE"
  Failure        := "PE Failure"
  Actual_Speed   := "S_Data".PE_Actual_Speed
  Engine_On      := "PE On"
  Preset_Speed_Reached := #PE_Preset_Speed_Reached

U   #PE_Preset_Speed_Reached
U   #DE_Preset_Speed_Reached
=   #Preset_Speed_Reached
```

Si vous programmez en LIST, sélectionnez la zone de saisie du nouveau réseau et entrez les instructions LIST ci-contre.

Enregistrez votre programme et fermez le bloc.

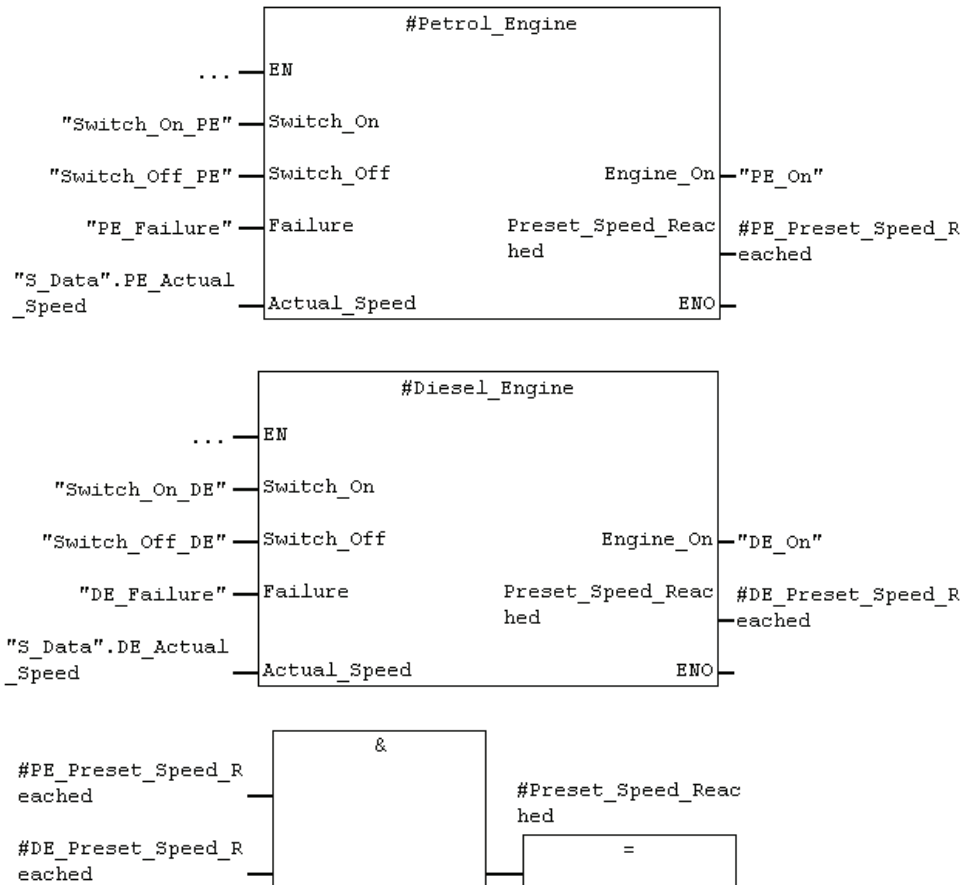




### Programmer le FB10 en LOG

Si vous programmez en LOG, sélectionnez la zone de saisie d'un nouveau réseau et entrez les instructions LOG suivantes.

Enregistrez ensuite votre programme et fermez le bloc.



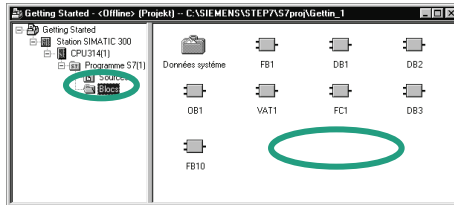
Pour que les deux appels du FB1 dans le FB10 soient traités, il faut également appeler ce dernier.

Vous ne pouvez programmer de multiinstances que pour les blocs fonctionnels. La création de multiinstances n'est pas possible pour les fonctions (FC).

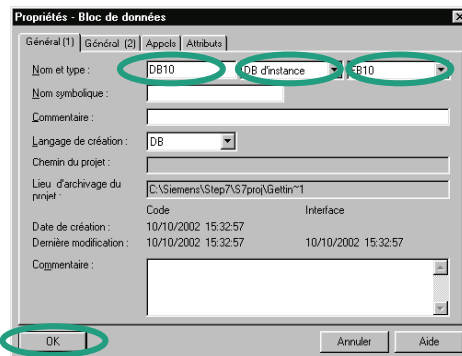
Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de code" et "Multiinstances dans la déclaration des variables" via la commande de menu ? > **Rubriques d'aide.**

### 10.3 Générer un DB10 et modifier la valeur effective

Le bloc de données DB10 remplacera les blocs de données DB1 et DB2. Le DB10 renferme les données du moteur à essence et Diesel requises plus tard lors de l'appel du FB10 dans l'OB1 (voir l'appel du FB1 dans l'OB1 dans le paragraphe 5.6 et les suivants).

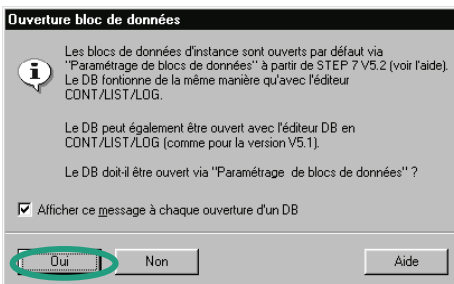


Générez un bloc de données DB dans le dossier **Blocs** du projet "Getting Started" avec le menu contextuel du bouton droit de la souris DB10.



Modifiez pour cela dans la boîte de dialogue "Propriétés – Bloc de données" le nom du bloc de données DB10, sélectionnez dans la liste déroulante correspondante le type "DB d'instance". Choisissez dans la liste déroulante de droite le bloc fonctionnel "FB10" auquel il doit être affecté et confirmez vos entrées avec **OK**.

Le bloc de données DB10 est maintenant inséré dans le projet "Getting Started".



Ouvrez le DB10 par double clic.

Confirmez la boîte de dialogue suivante avec **OUI** pour paramétrer le bloc de données d'instance. Activez la **vue des données**.

La vue des données affiche toutes les variables du DB10, y compris les variables "internes" des deux appels du FB1 ("instances locales").  
La vue de déclaration montre les variables telles que vous les avez déclarées dans le FB10.

Entrez "1300" à la place de la valeur effective du moteur Diesel. Enregistrez le bloc de données et fermez-le.

	Adress	Décl.	Nom	Type	Valeur initial	Valeur en co	Commentaire
1	0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Les deux moteurs ont atteint la vitesse prescrite
2	2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Mise en marche du moteur
3	2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Arrêt du moteur
4	2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Défaillance du moteur provoquant l'arrêt
5	4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Vitesse réelle du moteur
6	6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Le moteur se met en marche
7	6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Vitesse prescrite atteinte
8	8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Vitesse de moteur prescrite
9	10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Mise en marche du moteur
10	10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Arrêt du moteur
11	10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Défaillance du moteur provoquant l'arrêt
12	12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Vitesse réelle du moteur
13	14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Le moteur se met en marche
14	14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Vitesse prescrite atteinte
15	16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Vitesse de moteur prescrite



La table de déclaration des variables du DB10 contient à présent toutes les variables. Dans la partie supérieure de la table, vous voyez les variables de l'appel du bloc fonctionnel "Moteur essence" et dans la partie inférieure de la table l'appel du bloc fonctionnel "Moteur Diesel" (voir paragraphe 5.5).

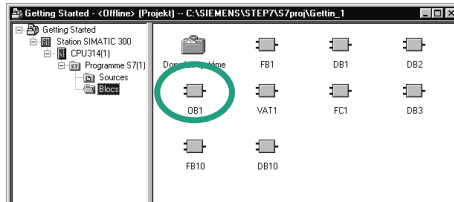
Les variables internes du FB1 conservent leurs noms symboliques, par exemple "Marche". Il est seulement précédé du nom de l'instance locale, par exemple "MotEss.Marche".

Pour plus d'informations, référez-vous aux rubriques "Programmation de blocs", "Création de blocs de données" via la commande de menu ? > **Rubriques d'aide.**



## 10.4 Appel du FB10 dans l'OB1

Dans notre exemple, le FB10 est appelé dans l'OB1. Il s'agit de la même fonctionnalité que celle nous avons déjà vue lorsque nous avons programmé et appelé le bloc FB1 dans l'OB1 (paragraphes 5.6 et suivants). L'utilisation d'un bloc fonctionnel multiinstance permet de remplacer les réseaux 4 et 5 programmés au chapitre 5.



Ouvrez l'OB1 dans lequel vous venez de programmer le bloc FB10.

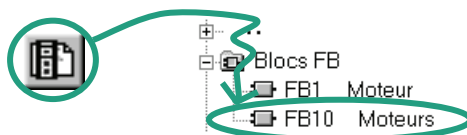
### Définir les mnémoniques

La fenêtre de l'éditeur de programme CONT/LIST/LOG est ouverte. Ouvrez la table des mnémoniques avec **Outils > Table des mnémoniques** et entrez les noms symboliques du bloc fonctionnel FB10 et du bloc de données DB10 dans la table.

Enregistrez la table des mnémoniques et fermez la fenêtre.

Mnémonique	Opérande	e de donn	Commentaire
...	...	...	...
Moteurs	FB 10	FB 10	Exemple de multiinstances
Données_Moteurs	DB 10	FB 10	Bloc de données d'instance de FB10
...	...	...	...

### Programmer l'appel dans CONT



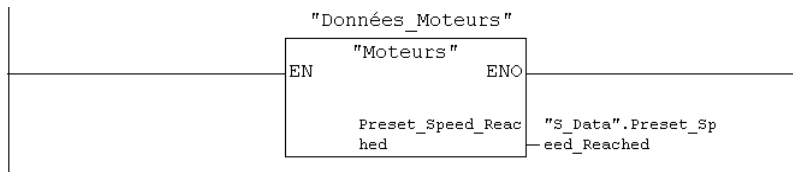
Insérez à la fin de l'OB1 un nouveau réseau et programmez l'appel du FB10 ("Moteurs").



Complétez l'appel en inscrivant les mnémoniques voulus comme dans la figure ci-dessous.

Effacez l'appel du FB1 dans l'OB1 (réseaux 4 et 5 des paragraphes 5.6 et suivants), car le bloc FB1 ne devra plus être appelé que centralement par l'intermédiaire du bloc FB10.

Enregistrez ensuite votre programme et fermez le bloc.



Le signal de sortie "Vitesse\_atteinte" du FB10 ("Moteurs") est transmis à la variable du bloc de données.

### Programmer l'appel dans LIST

Si vous programmez en LIST, cliquez dans la zone de saisie du nouveau réseau et entrez les instructions LIST suivantes. Sélectionnez pour cela le **FB10 "Moteurs"** dans les blocs FB du catalogue des éléments de programme.

Effacez l'appel du FB1 dans l'OB1 (réseaux 4 et 5 des paragraphes 5.6 et suivants), car le bloc FB1 ne devra plus être appelé que centralement par l'intermédiaire du FB10.

Enregistrez ensuite votre programme et fermez le bloc.

```

CALL "Moteurs" , "Données_Moteurs"
Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
  
```



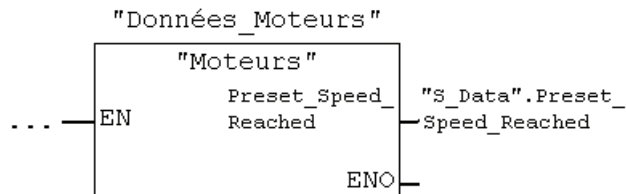
## Programmer l'appel dans LOG



Si vous programmez en LOG, cliquez dans la zone de saisie du nouveau réseau et entrez les instructions LOG comme dans la figure ci-dessous. Sélectionnez pour cela dans le catalogue des éléments du programme le FB10 "Moteurs" parmi les blocs FB.

Effacez l'appel du bloc FB1 dans l'OB1 (réseaux 4 et 5 des paragraphes 5.6 et suivantes), car le bloc ne devra plus être appelé que centralement par l'intermédiaire du bloc FB10.

Enregistrez ensuite votre programme et fermez le bloc.



Si vous avez besoin dans votre solution d'automatisation d'autres commandes de moteurs, par exemple pour des moteurs à gaz naturel ou gaz biologique, vous pouvez les programmer comme multiinstance et les appeler dans le FB10.

Vous déclarez pour cela les nouveaux moteurs dans la table de déclaration des variables du FB10 ("Moteurs") et programmez l'appel du FB1 (multiinstance dans le catalogue des éléments de programme) dans le bloc FB10. Si vous voulez utiliser l'adressage symbolique, vous devez définir les nouveaux mnémoniques, par exemple pour la mise en marche et la mise à l'arrêt des moteurs, dans la table des mnémoniques.

Pour plus d'informations, référez-vous aux rubriques "Appel des aides de référence", "Description du langage CONT/LOG/LIST" via la commande de menu ? > **Rubriques d'aide.**



# 11 Configuration de la périphérie décentralisée

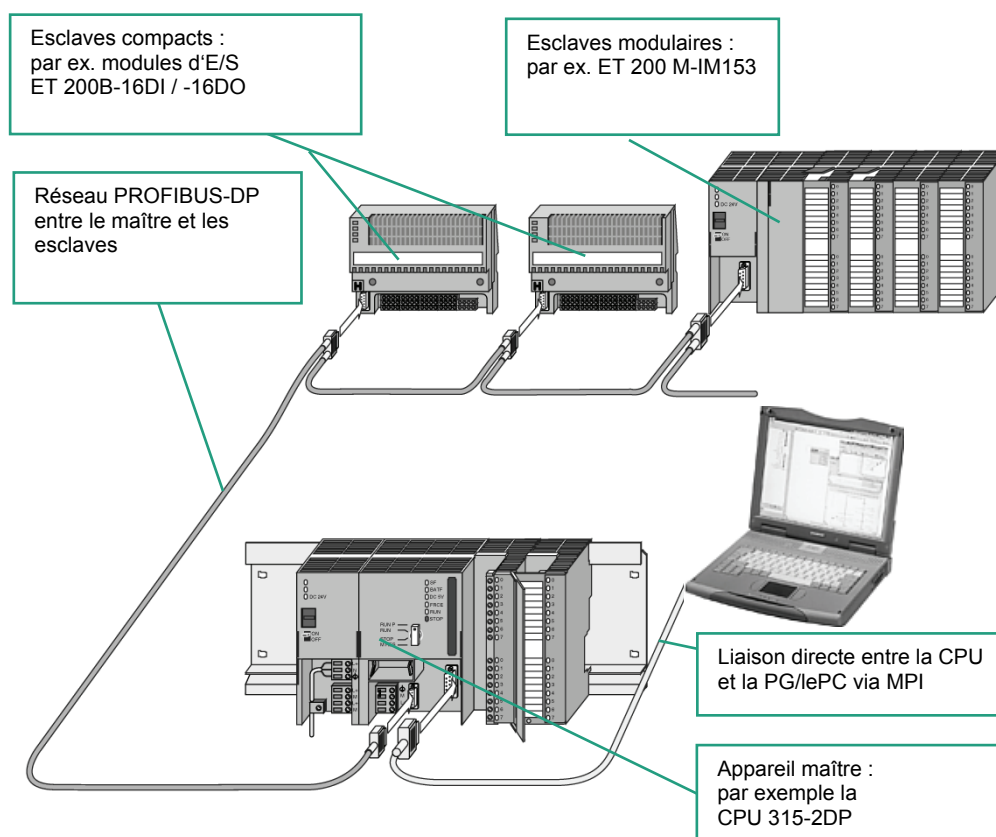
## 11.1 Installer et configurer la périphérie décentralisée avec PROFIBUS-DP

Dans la configuration traditionnelle d'une installation d'automatisation, les câbles de liaison des capteurs et des actionneurs sont enfilés directement dans les modules d'entrées/sorties de l'appareil de base, entraînant des coûts et un temps de câblage importants.

En configuration décentralisée, les coûts de câblage peuvent être considérablement réduits en plaçant les modules d'entrées/sorties à proximité des capteurs et actionneurs. Le bus de terrain PROFIBUS-DP fait la liaison entre le système d'automatisation, les modules de périphérie et les appareils de terrain.

Vous avez pu apprendre la programmation utilisée pour la configuration conventionnelle au chapitre 6. La configuration décentralisée ne requiert pas de programmation particulière. Vous choisissez vos modules dans le catalogue du matériel, les disposez sur un châssis et adaptez leurs propriétés en fonction de vos besoins.

Vous devez savoir comment créer un projet et configurer une installation centralisée (voir chapitre 6 et paragraphe 2.1).

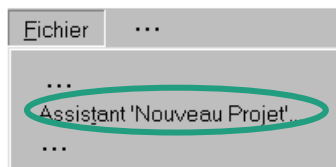


## Créer un nouveau projet

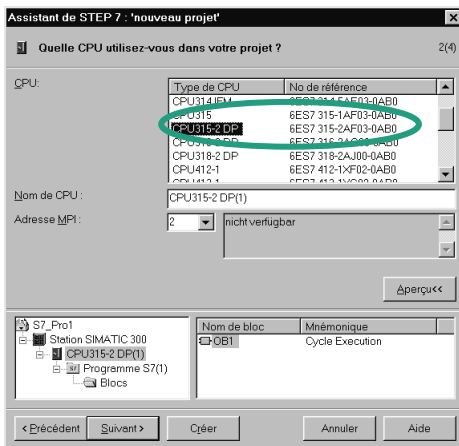


Vous pouvez à nouveau partir du SIMATIC Manager.

Fermez éventuellement les projets encore ouverts pour plus de clarté.



Créez un **nouveau projet**.

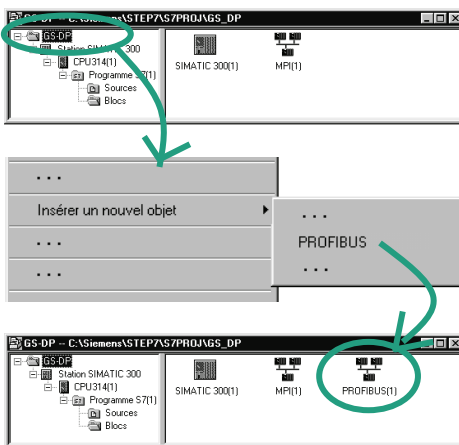


Sélectionnez lorsque l'Assistant vous le demande la **CPU 315-2DP** (CPU avec réseau PROFIBUS-D).

Procédez autrement comme au paragraphe 2.1 et entrez comme nom de projet "GS-DP" (Getting Started – Périphérie décentralisée).

Si vous désirez tout de suite créer votre propre configuration, indiquez à cet endroit votre CPU. Attention ! Elle doit prendre DP en charge.

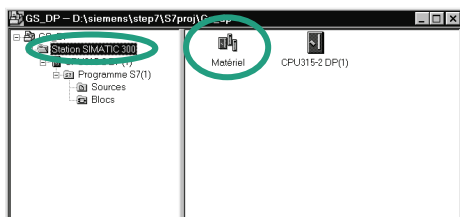
## Insérer un réseau PROFIBUS



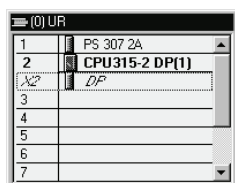
Sélectionnez le dossier **GS-DP**.

Insérez un réseau **PROFIBUS** via le menu contextuel du bouton droit de la souris.

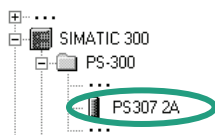
## Configurer la station



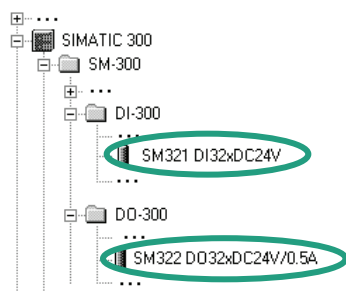
Sélectionnez le dossier **Station SIMATIC 300** et double-cliquez sur **Matériel**. Ceci ouvre la fenêtre "HW Config" (voir le paragraphe 6.1).



La CPU 315-2DP est déjà enfichée sur le châssis. S'il n'est pas visible, sélectionnez la commande de menu **Affichage > Catalogue du matériel** pour faire apparaître le catalogue ou cliquez sur le bouton correspondant.



Sélectionnez et faites glisser un module d'alimentation **PS307 2A** à l'emplacement 1.

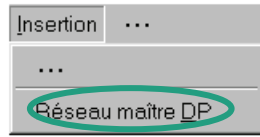


Enfichez de la même manière sur les emplacements 4 et 5 les modules d'entrées/sorties **DI32xDC24V** et **DO32xDC24V/0.5A**.

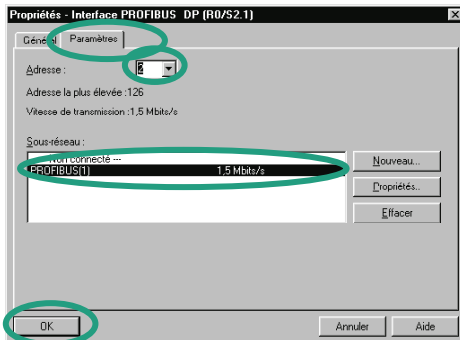
Il est possible de placer aussi des unités centrales sur le même rack en plus de la CPU-DP (ceci ne sera pas expliqué ici).



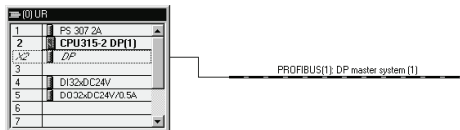
### Configurer le réseau maître DP



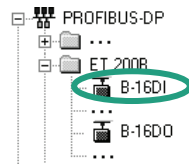
Sélectionnez le maître DP à l'emplacement 2.1 et insérez un **réseau maître DP**.



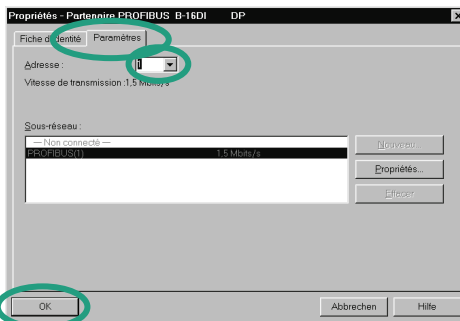
Acceptez l'adresse proposée dans la boîte de dialogue qui s'affiche. Sélectionnez dans le champ "Sous-réseau" l'entrée "PROFIBUS(1)" et confirmez vos entrées avec **OK**.



Vous pouvez déplacer tous les objets qui se trouvent sur le réseau maître DP en les sélectionnant et en les faisant glisser tout en maintenant le bouton de la souris enfoncé.



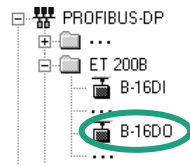
Naviguez dans le catalogue du matériel jusqu'au module **B-16DI** et insérez-le par glisser-lâcher dans le réseau maître (attendez que le curseur change d'aspect et relâchez-le).



L'adresse réseau du module inséré peut être alors modifiée dans la page d'onglet **Paramètres** de ses propriétés.

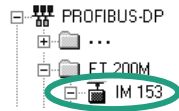
Confirmez l'adresse proposée avec **OK**.





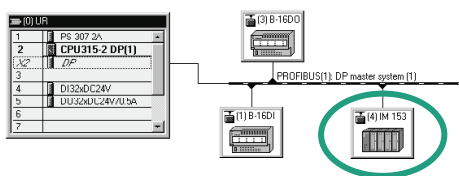
Amenez de la même manière le module **B-16DO** sur le réseau maître DP.

Son adresse de réseau sera automatiquement modifiée dans les propriétés. Confirmez celle-ci par **OK**.



Faites glisser le coupleur **IM 153** sur le réseau maître DP et validez l'adresse réseau proposée avec **OK**.

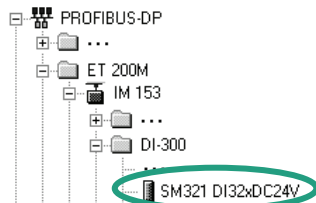
Nous conservons dans notre exemple les adresses par défaut. Celles-ci peuvent toutefois être modifiées pour les besoins de l'installation.



Sélectionnez le module **ET 200M** dans le réseau. Vous pouvez voir dans la table de configuration les emplacements vides de l'ET 200M.

[4] IM 153	
Emplacement	Module
4	
5	
6	
7	
8	
9	
10	
11	

Sélectionnez l'emplacement **4**.

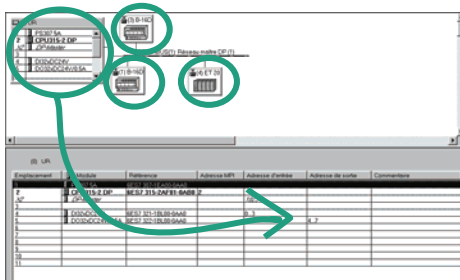


Le module ET 200M peut recevoir à son tour d'autres modules d'entrées/sorties. Choisissez par exemple le module **DI32xDC24V** pour l'emplacement 4 et insérez le module par double clic.

Vérifiez avant de sélectionner des modules dans le catalogue du matériel que vous êtes dans le bon dossier, par exemple dans le dossier ET 200M si vous voulez sélectionner des modules ET 200M.

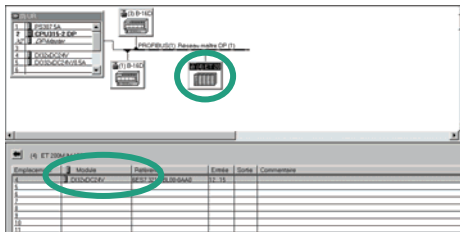


### Modifier l'adresse réseau



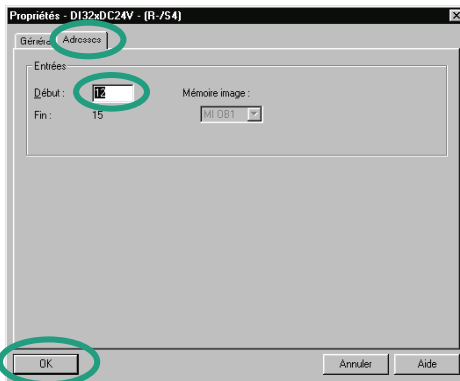
Dans notre exemple, nous n'avons pas eu à modifier l'adresse réseau. Dans la pratique, vous aurez souvent à le faire.

Sélectionnez l'un après l'autre les autres partenaires de réseau et vérifiez leurs adresses d'entrée et de sortie. Les adresses ont été modifiées dans la configuration matérielle, il n'y a pas d'adresses attribuées en double.

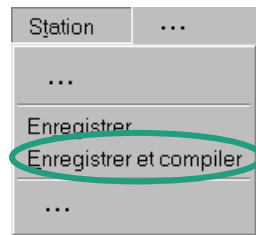


Admettons que vous vouliez modifier l'adresse du ET 200M :

Sélectionnez le **ET 200M** et double-cliquez sur le module **DI32xDC24V** (emplacement 4).



Modifiez à présent dans la page d'onglet **Adresses** des propriétés les adresses d'entrée de 6 à 12. Fermez la boîte de dialogue avec **OK**.



Enregistrez finalement votre configuration de périphérie décentralisée avec la commande : **Enregistrer et compiler.**

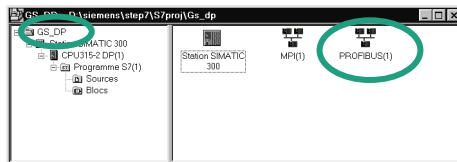
Fermez la fenêtre.

Avec la commande Enregistrer et compiler, une vérification de la cohérence de votre configuration a automatiquement lieu. Lorsque cette vérification n'a détecté aucune erreur, les données système sont générées et chargées dans le système cible.

Avec la commande Enregistrer, vous pouvez enregistrer une configuration inachevée ou comportant encore des erreurs. Il n'est pas possible de charger les données dans le système cible.

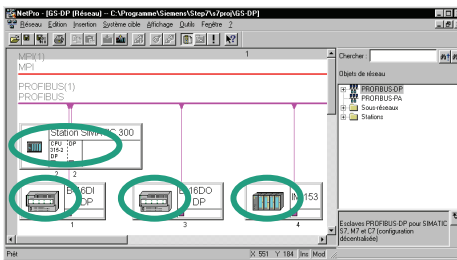


### Alternative : configuration du réseau



La configuration de périphérie décentralisée peut également être effectuée dans la configuration des réseaux.

Double-cliquez dans SIMATIC Manager sur le réseau **PROFIBUS (1)**.



La fenêtre "NetPro" s'ouvre.

Vous pouvez ajouter par glisser-lâcher d'autres esclaves DP que vous sélectionnez dans le catalogue des objets de réseau au bus PROFIBUS-DP.

Double-cliquez sur un élément quelconque pour le configurer. La fenêtre "Configuration matérielle" s'ouvre.



Avec **Station > Vérifier la cohérence** (fenêtre de la configuration matérielle) et **Réseau > Vérifier la cohérence** (fenêtre de la configuration de réseau), vous pouvez lancer une vérification formelle du programme avant qu'il soit enregistré. Les erreurs trouvées par STEP 7 sont affichées et des solutions vous sont proposées.

Pour plus d'informations, référez-vous aux rubriques "Configuration du matériel" et "Configuration de la périphérie décentralisée" via la commande de menu ? > **Rubriques d'aide.**

**Félicitations !** Vous êtes arrivé en fin de parcours de ce "Getting Started" et avez abordé les thèmes centraux, appris les techniques de programmation et fait un tour d'horizon des fonctions principales de STEP 7. Vous pouvez à présent vous lancer dans votre premier projet.

Pour le cas où vous auriez besoin d'aide dans la recherche de fonctions précises ou auriez oublié des manipulations, n'oubliez pas de recourir à l'aide étendue de STEP 7.

Pour vous permettre d'approfondir vos connaissances sur STEP 7, nous vous proposons des stages de formation. Votre partenaire Siemens dans nos filiales se tient à votre disposition pour toute question.

Nous vous souhaitons beaucoup de succès pour la conception de vos projets !

Votre Siemens AG

# A. Annexe A

## A.1 Vue d'ensemble des exemples de projet relatifs au manuel Getting Started

- **ZFr01\_02\_STEP7\_\_LIST\_1-10 :**  
Les chapitres programmés de 1 à 10 y compris la table des variables du langage de programmation LIST.
- **ZFr01\_01\_STEP7\_\_LIST\_1-9 :**  
Les chapitres programmés de 1 à 9 y compris la table des variables du langage de programmation LIST.
- **ZFr01\_06\_STEP7\_\_CONT\_1-10 :**  
Les chapitres programmés de 1 à 10 y compris la table des variables du langage de programmation CONT.
- **ZFr01\_05\_STEP7\_\_CONT\_1-9 :**  
Les chapitres programmés de 1 à 9 y compris la table des variables du langage de programmation CONT.
- **ZFr01\_04\_STEP7\_\_LOG\_1-10 :**  
Les chapitres programmés de 1 à 10 y compris la table des variables du langage de programmation LOG.
- **ZFr01\_03\_STEP7\_\_LOG\_1-9 :**  
Les chapitres programmés de 1 à 9 y compris la table des variables du langage de programmation LOG.
- **ZFr01\_07\_STEP7\_\_DezP\_11 :**  
Le chapitre programmé 11 et la périphérie décentralisée.



# Index

## A

Adresse absolue 19  
Adresse réseau  
  modifier 106  
Appel de bloc dans CONT 52  
Appel de la fonction 82  
Appel de l'Aide de STEP 7 17  
Appliquer la tension 65

## B

Beispielprojekte 109  
Bloc de données  
  programmer 85  
Bloc de données global  
  créer 85  
  ouvrir 85  
Bloc de données global dans la table des variables 87  
Bloc fonctionnel  
  créer 37  
  ouvrir 37  
Blocs de données  
  générer les blocs de données d'instance 50

## C

Chargement de blocs isolés 67  
Chargement du programme dans le système cible 65  
Choix du langage de programmation 23  
Commutation de la table des variables en ligne 71  
Configuration de la périphérie décentralisée 101  
Configuration des unités centrales 61  
Configuration du réseau 107  
Configuration du réseau maître DP 104  
Configuration matérielle 61, 63  
Configuration PROFIBUS-DP 101  
CONT  
  appel de bloc 52  
  programmation du FB1 39  
  programmer un circuit 28  
  programmer un circuit série 26  
  programmer une bascule 29  
  programmer une fonction de temporisation 79  
  tester 68  
Copier la table des mnémoniques 24  
Création d'un programme avec FB et DB 37  
Création d'un programme dans l'OB1 23  
Création de la table des variables 70  
Création d'un bloc de données global 85  
Création d'un projet 13  
Création d'une fonction 77

Créer un bloc fonctionnel 37

## E

Éditeur de mnémoniques 20  
Éditeur de programme CONT/LIST/LOG 25  
Effacement général de la CPU et passage à RUN 65  
Etablissement de la liaison en ligne 63  
Evaluer la mémoire tampon de diagnostic 74

## F

Fonction  
  appeler 82  
  créer 77  
  ouvrir 77  
Forçage des variables 72

## G

Générer les blocs de données d'instance 50

## I

Insérer > Mnémonique 31, 34  
Insertion > Mnémonique 27  
Interroger l'état du module 74

## L

Lancement de SIMATIC Manager 13  
Liaison en ligne  
  établir 63  
LIST  
  Appel de bloc 55  
  programmation du bloc FB1 43  
  programmer une bascule 32  
  programmer une fonction de temporisation 80  
  programmer une instruction ET 30  
  programmer une instruction OU 31  
  tester 68  
LIST  
  insérer un mnémonique 31  
LIST  
  représentation symbolique 32  
LOG  
  appel de bloc 57  
  Programmation du bloc FB1 46  
  programmer une bascule 36  
  programmer une fonction de temporisation 81  
  programmer une fonction OU 35  
  tester 68

LOG

- insérer un mnémonique 34
- programmer une fonction ET 33

LOG

- représentation symbolique 36
- Logiciels optionnels SIMATIC 18

**M**

- Matérielle configuration 61
- Mémoire tampon de diagnostic
  - évaluer 74
- Mise en marche de la CPU 67
- Modification de l'adresse réseau 106
- Modifier les valeurs effectives 50
- Multiinstance
  - programmer 89

**N**

- Navigation dans la structure du projet 18

**O**

- OB1
  - ouvrir 24
- Ouvrir l'OB1 24
- Ouvrir un bloc de données global 85
- Ouvrir un bloc fonctionnel 37
- Ouvrir une fonction 77

**P**

- Périphérie décentralisée
  - configurer 101
- Programmation d'un appel de bloc en LIST 55
- Programmation d'un appel de bloc en LOG 57
- Programmation d'un bloc de données global 85
- Programmation d'un circuit série en CONT 26
- Programmation d'une bascule en LIST 32
- Programmation d'une fonction (FC) 77
- Programmation d'une fonction ET en LOG 33
- Programmation d'une instruction ET en LIST 30
- Programmation d'une instruction OU en LIST 31
- Programmation d'une multiinstance 89
- Programmation du bloc FB1 en LIST 43
- Programmation du bloc FB1 en LOG 46
- Programmation du FB1 en CONT 39
- Programmation d'un circuit parallèle en CONT 28
- Programmation d'une bascule en CONT 29
- Programmation d'une bascule en LOG 36
- Programmation d'une fonction de temporisation en CONT 79

- Programmation d'une fonction de temporisation en LIST 80

- Programmation d'une fonction de temporisation en LOG 81

- Programmation d'une fonction OU en LOG 35

- Programmation en ligne 67

- Programmation symbolique 20

**R**

- Remplir la table de déclaration des variables
  - CONT 39
  - LIST 43
  - LOG 46
- Représentation symbolique
  - CONT 29
- Réseau > Vérifier la cohérence 107
- Réseau maître DP
  - configurer 104

**S**

- Station > Vérifier la cohérence 107
- STEP 7
  - installer 11
- STEP 7
  - mode d'emploi 10
- STEP7
  - Assistent
    - nouveau projet 13
- Structure du projet dans SIMATIC Manager 16

**T**

- Table des mnémoniques 20
- Table des mnémoniques
  - copier 24
- Table des variables
  - commuter en ligne 71
  - créer 70
- Test avec CONT 68
- Test avec LIST 68
- Test avec LOG 68
- Type de données 21

**V**

- Variables
  - forcer 72
  - visualiser 72
- Vérification du mode de fonctionnement 67
- Visualisation des variables 72
- Vue de déclaration 95
- Vue des données 95