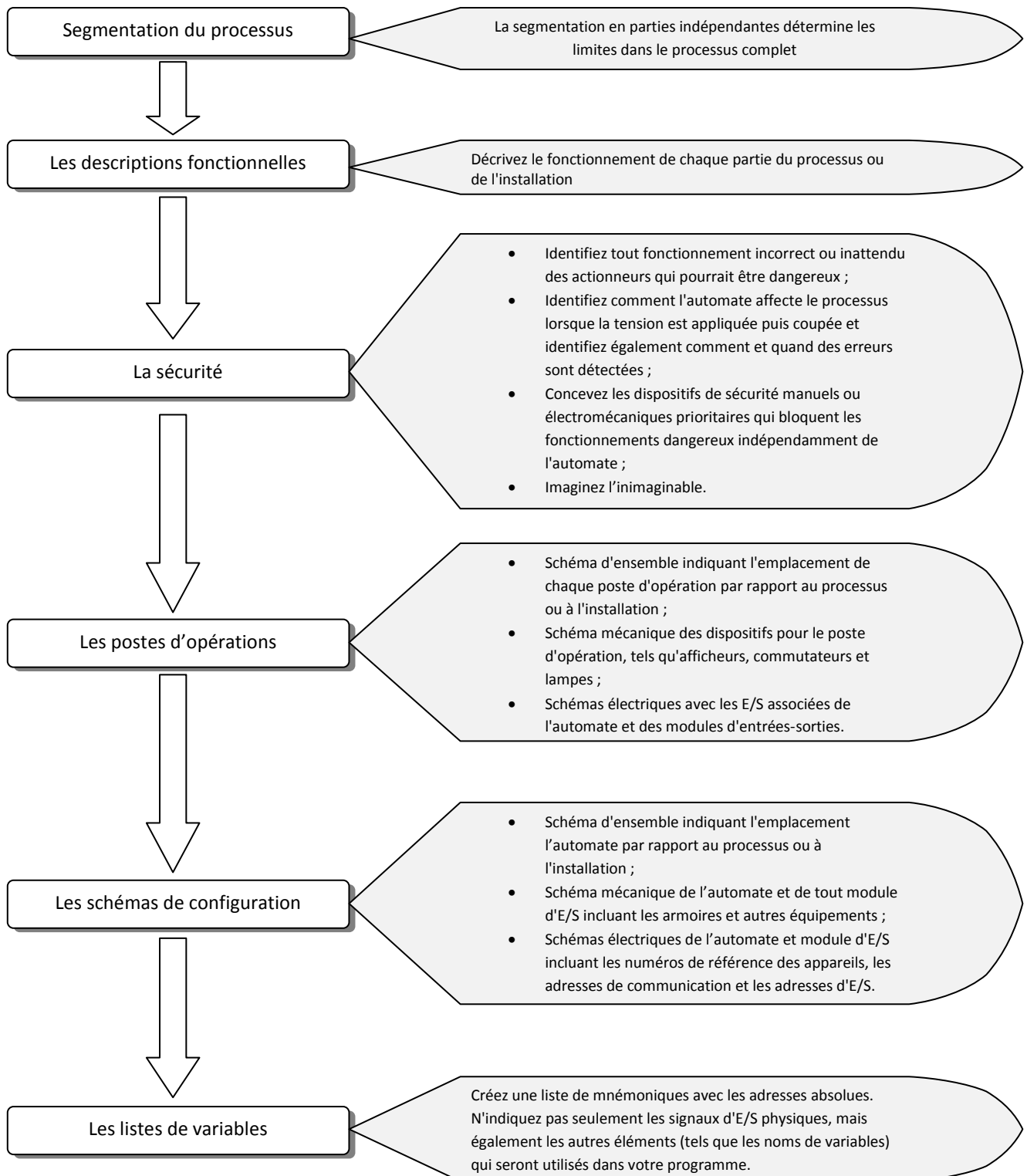




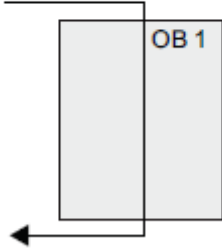
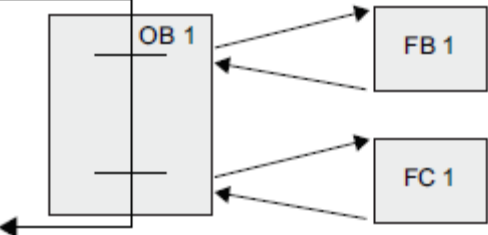
## 1. Analyse de l'automatisation





## 2. Organisation du programme

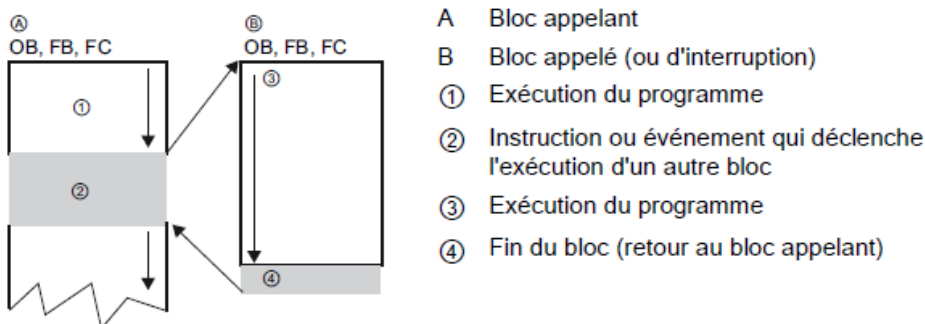
Selon les besoins de votre application, vous pouvez choisir soit une structure linéaire soit une structure modulaire pour votre programme utilisateur.

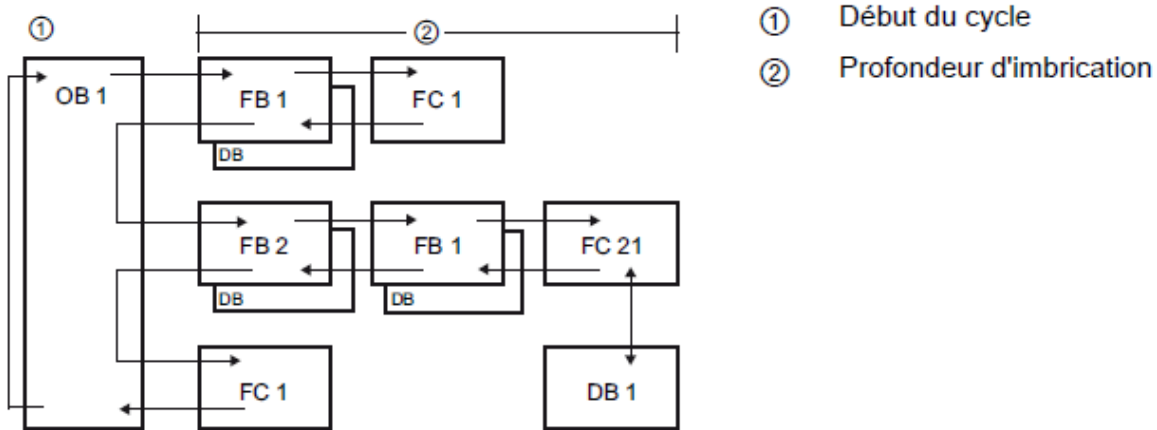
<p>Un programme linéaire exécute toutes les instructions pour vos tâches d'automatisation séquentiellement les unes après les autres. Avec un programme linéaire, vous placerez typiquement toutes les instructions dans l'OB d'exécution cyclique du programme (OB 1).</p>	<p><b>Structure linéaire :</b></p> 
<p>Un programme modulaire appelle des blocs de code spécifiques qui exécutent des tâches spécifiques. Pour créer une structure modulaire, vous divisez la tâche d'automatisation complexe en petites tâches subordonnées qui correspondent aux fonctions technologiques du processus. Chaque bloc de code fournit le segment de programme pour une tâche subordonnée. Vous structurez votre programme en appelant l'un des blocs de code à partir d'un autre bloc.</p>	<p><b>Structure modulaire :</b></p> 

## 3. Structuration du programme

En concevant des FB et des FC qui exécutent des tâches génériques, vous créez des blocs de code modulaires. Vous structurez ensuite votre programme en faisant appeler ces modules réutilisables par d'autres blocs de code. Le bloc appelant transmet des paramètres spécifiques de l'appareil au bloc appelé.

Lorsqu'un bloc de code appelle un autre bloc de code, la CPU exécute le code du programme dans le bloc appelé. Une fois l'exécution du bloc appelé achevée, la CPU reprend l'exécution du bloc appelant. Le traitement se poursuit par l'exécution de l'instruction qui suit l'appel de bloc.





### Bloc OB

OB1 est par défaut, le bloc d'organisation qui contient votre programme principal.

Vous pouvez inclure plus d'un OB de cycle de programme dans votre programme utilisateur.

A l'état MARCHE, les OB de cycle de programme s'exécutent au niveau de priorité le plus bas et peuvent être interrompus par tous les autres types de traitement de programme.

L'OB de démarrage n'interrompt pas l'OB de cycle de programme, car la CPU exécute l'OB de démarrage avant de passer à l'état MARCHE.

Les OB déclenchés sur événement.

La CPU exécute un OB suite à un événement, tel qu'une alarme de diagnostic ou un intervalle de temps.

Certains OB ont des événements déclencheurs et un comportement prédéfinis.

### Bloc FC

Une fonction (FC) est un bloc de code qui exécute typiquement une opération spécifique sur un ensemble de valeurs d'entrée.

Vous avez p. ex. recours à des FC pour effectuer des opérations standard et réutilisables (telles que des calculs mathématiques) ou des fonctions technologiques (telles que des contrôles individuels utilisant des opérations logiques sur bits).

Une FC peut également être appelée plusieurs fois en différents points d'un programme. Cette réutilisation simplifie la programmation de tâches revenant souvent.

Les données temporaires ne sont pas sauvegardées. Pour stocker les données de manière permanente, affectez la valeur de sortie à une adresse de mémoire globale, un memento M ou un DB global par exemple.

### Bloc FB

Un bloc fonctionnel (FB) est un bloc de code qui utilise un bloc de données d'instance pour ses paramètres et ses données statiques.

Les FB ont une mémoire de variables qui se situe dans un bloc de données (DB) appelé DB d'instance.

Le DB d'instance fournit un bloc de mémoire qui est associé à cette instance (ou appel) du FB et qui contient les données une fois le FB achevé.

Vous structurez votre programme en insérant dans un bloc de code l'appel d'un FB et d'un DB d'instance. La CPU exécute alors le code dans ce FB et sauvegarde les paramètres du bloc et les données locales statiques dans le DB d'instance. Une fois l'exécution du FB achevée, la CPU revient au bloc de code qui a appelé le FB. Le DB d'instance conserve les valeurs pour cette instance du FB. Ces valeurs sont disponibles pour des appels ultérieurs du bloc fonctionnel soit dans le même cycle, soit dans d'autres cycles.



## 4. Langages de programmation

### Codesys et la norme IEC 61131-3

CODESYS est un environnement de développement gratuit pour des automates programmables industriels (API) selon le standard CEI 61131-3 pour le développement d'applications dans l'automation industrielle. CODESYS est développé et commercialisé par 3S-Smart Software Solutions, une société de logiciels indépendante fondée en 1994 et située à Kempten en Allemagne. Le nom CODESYS est un acronyme et signifie Controller Development System. Le système de programmation est libre de droits et peut être installé et utilisé légalement sur tout ordinateur.

Tous les langages spécifiés de la CEI 61131-3 (Commission électrotechnique internationale) sont inclus dans CODESYS :

- IL (liste d'instructions), une sorte de langage assembleur ;
- ST (Texte structuré, inspiré par PASCAL) pour la programmation structurée ;
- LD (Langage ladder), en français aussi „schéma à contacts" ;
- FBD (Function Block Diagram), en français „boîtes fonctionnelles" ;
- SFC (Sequential Function Chart), proche du langage Grafcet.

Les automates IFM, ABB, Beckhoff, Schneider et bien d'autres sont compatibles Codesys

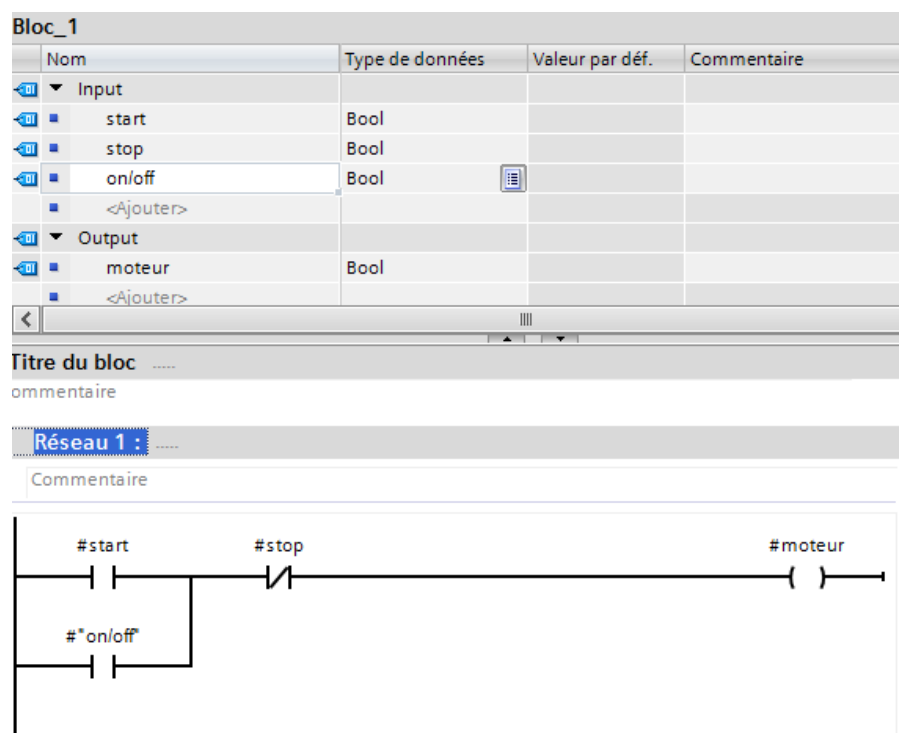
Codesys n'est pas directement compatible avec le matériel Siemens, mais les langages LD, FBD, et ST se retrouvent dans TIA Portal sous les noms CONT, LOG, et SCL.

### Schéma à contacts (CONT)

Les éléments d'un schéma de circuit, contacts à ouverture et à fermeture et bobines par exemple, sont reliés pour former des réseaux. Pour créer la logique pour des opérations complexes, vous pouvez insérer des branches formant des circuits parallèles. Les branches parallèles s'ouvrent vers le bas ou se connectent directement à la barre conductrice. Vous terminez les branches vers le haut.

CONT fournit des instructions sous forme de boîtes pour des fonctions variées, telles que les fonctions mathématiques, de temporisation, de comptage et de transfert.

Exemple :





## Logigramme (LOG)

Comme CONT, LOG est un langage de programmation graphique. La représentation de la logique repose sur les symboles logiques graphiques utilisés en algèbre booléenne.

Pour créer la logique pour des opérations complexes, insérez des branches parallèles entre les boîtes.

Les fonctions mathématiques et autres fonctions complexes peuvent être représentées directement avec des boîtes logiques.

**Bloc\_1**

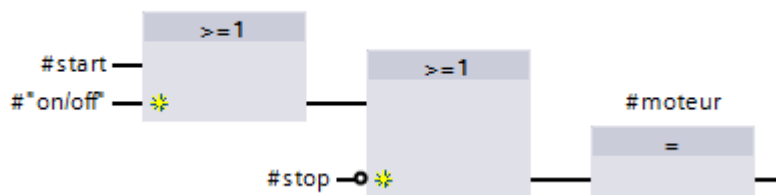
	Nom	Type de données	Valeur par déf.	Com
▼	Input			
■	start	Bool		
■	stop	Bool		
■	on/off	Bool		
	<Ajouter>			
▼	Output			
■	moteur	Bool		

Titre du bloc : .....

Commentaire

**Réseau 1 :** .....

Commentaire



## SCL (Structured Control Language)

SCL est un langage de programmation évolué basé sur PASCAL pour les CPU SIMATIC S7. SCL prend en charge la structure de blocs de STEP 7. Vous pouvez en outre inclure des blocs de programme écrits en SCL avec des blocs de programme écrits en CONT et LOG.

Les instructions SCL utilisent des opérateurs de programmation standard, par exemple pour l'affectation (:=) et les fonctions mathématiques (+ pour l'addition, - pour la soustraction, \* pour la multiplication et / pour la division). SCL utilise également des opérations de gestion de programme PASCAL standard, telles que IF-THEN-ELSE, CASE, REPEAT-UNTIL, GOTO et RETURN. Vous pouvez utiliser n'importe quelle référence PASCAL pour les éléments syntaxiques du langage de programmation SCL. Un grand nombre des autres instructions pour SCL, comme les temporisations et les compteurs, correspondent aux instructions CONT et LOG.

## Conclusions sur les langages

CONT et LOG sont très efficaces dans le traitement de la logique booléenne. Bien que SCL soit particulièrement efficace pour les calculs mathématiques complexes et pour les structures de commande de projet, vous pouvez aussi l'utiliser pour la logique booléenne.



Décrivez le processus d'analyse de l'automatisation d'une tâche.	
Décrivez les éléments de sécurité à analyser.	
Comparez la structure d'un programme linéaire avec la structure d'un programme modulaire.	
Dans quel cas allez-vous utiliser plusieurs OB ?	
Comparez l'utilisation d'un FC et d'un FB.	

Traduisez les équations booléennes suivantes en CONT et LOG

Equation	CONT	LOG
$(A+B).C$		
$(A.B)+C$		



$(A+B)+ /C$		
$(/A.B)+C$		
$A+(B./C)$		
$A \oplus (B.C)$		
$(A+B) \oplus /C$		