

# PROGRAMMATION AVEC STEP 7

## ***1. Introduction***

Dans cete partie nous allons détaillé comment faire pour créer un projet par différents langages de programmation et comment transférer et tester le programme dans la CPU et comment se fait le traitement des programmes en compte tenu le principe de conception d'une structure de programme avec des explication à différents blocs.

## **STEP 7 : Mode d'emploi**

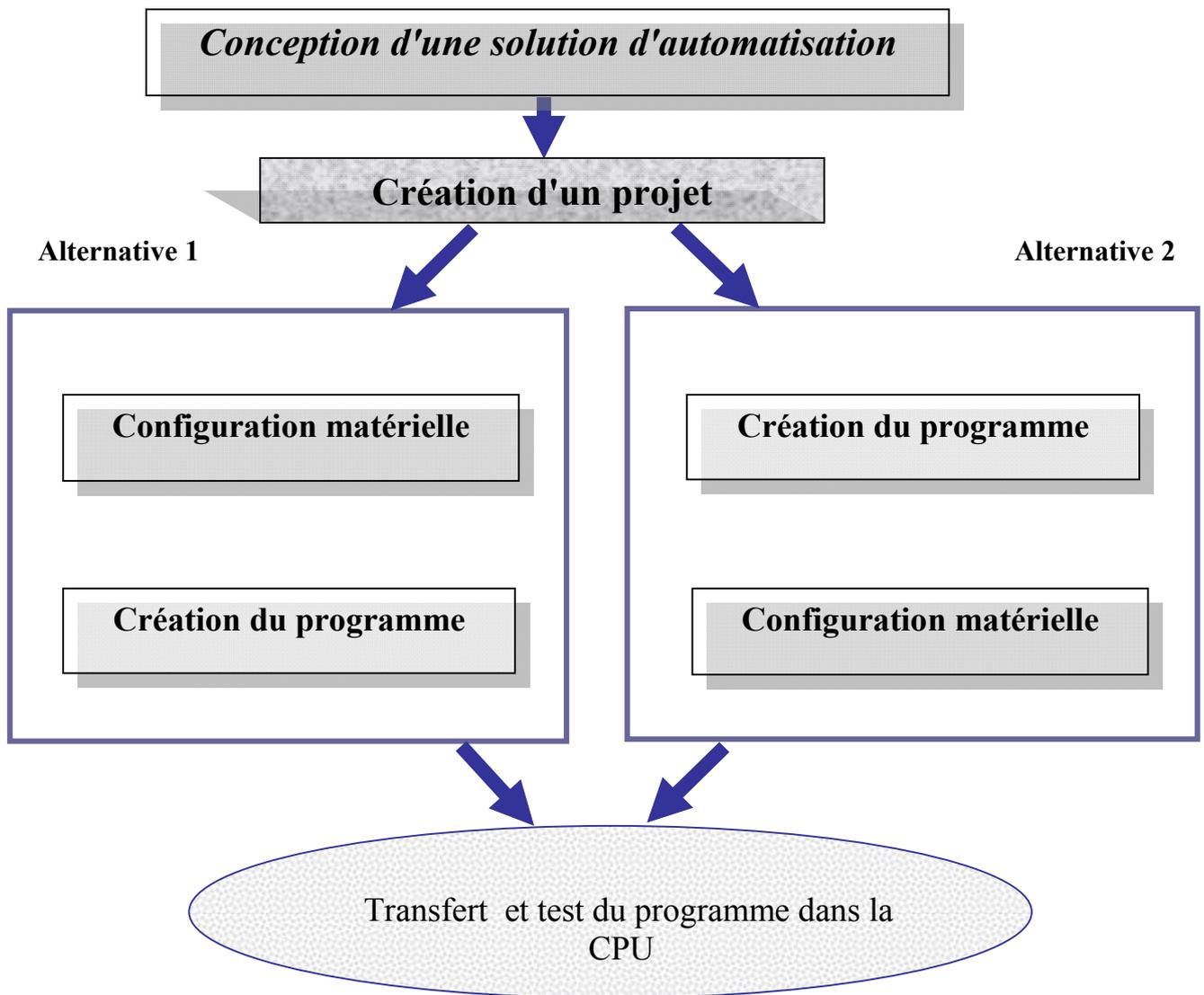
Avant de créer un projet, on peut envisager différentes approches. En effet, le logiciel STEP 7, offre une liberté, de choix de la procédure à adapter.

Dans le cas au un système contient beaucoup d'entrées et de sorties, il est préférable de commencer par configurer le matériel avant la création du programme. L'application de la configuration matérielle de STEP 7 présente l'avantage que les adresses y sont sélectionnées pour nous.

Si nous choisissons la seconde alternative (*voir la figure 1*), il nous faudra rechercher nous-même les adresses en fonctions des constituants choisis.

Vous ne pourrez alors pas bénéficier de la fonction d'adressage automatique de STEP 7.

La configuration matérielle nous permet non seulement de sélectionner les adresses, mais également de modifier les paramètres et les propriétés des modules.



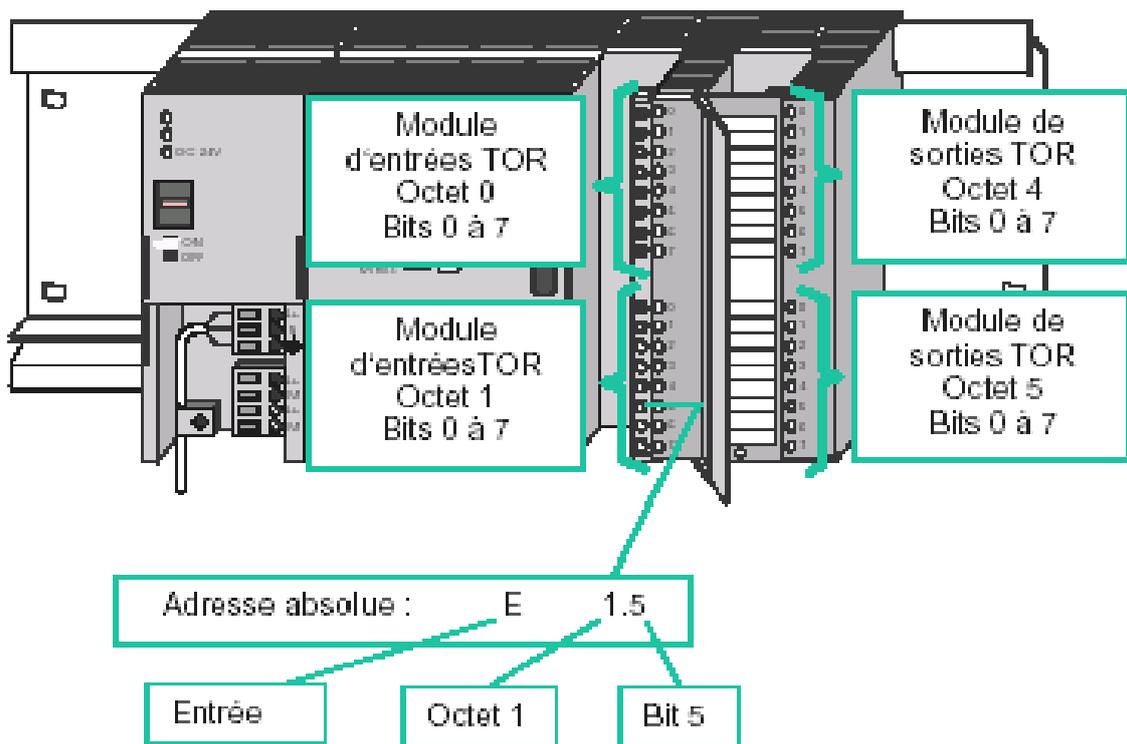
*Figure 1 : Les différentes étapes de programmation par Step7*

## 2 Programmation symbolique

### 2.1 Adresse absolue (voir Figure 2)

Chaque entrée et chaque sortie possèdent 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



**Figure 2 :** Les adresses absolues du modules entrées / sorties

## 2.2 Programmation symbolique

Nous affectons dans la table des mnémoniques un nom symbolique à toutes adresses absolues que nous voulons appeler dans le programme ainsi que le type de donnée, par exemple d'entrée E 0.1 le mnémonique commutateur 1. Ces noms valent pour toutes les sections du programme. C'est pour ça on les appelle des variables globales.

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

### 3 Edition et traitement du programme

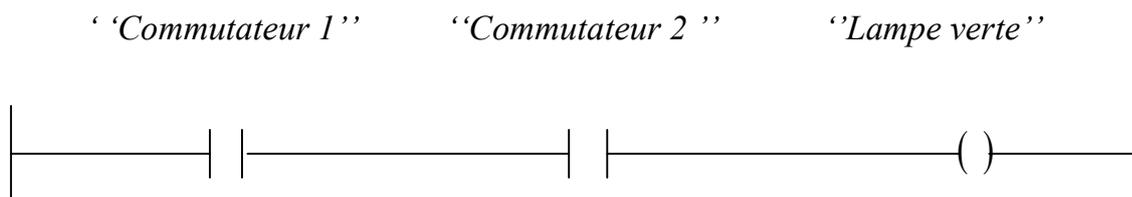
#### 3.1 Edition du programme

Pour créer notre programme S7, nous disposons dans STEP 7 de trois langages de programmation : CONT, LIST, LOG, et Grafcet (en option S7- Grafcet).

**Exemple :** Lampe verte : ‘ ‘Commutateur 1’ ’ & ‘ ‘Commutateur 2’ ’

##### 3.1.1 CONT (schéma à CONTACTs)

Pour l’habitué des schémas électrique, le schéma à contacts CONT est un langage de programmation graphique. La synthèse de ses instructions s’inspire des schémas à relais. CONT permet de suivre facilement le flux d’énergie circulant via des entrées, des sorties et des opérations entre les barres d’alimentation.



##### 3.1.2 LIST (LISTe d’instructions)

Pour l’informaticien. La liste d’instructions LIST est un langage de programmation pouvant être utilisé pour la création d’instruction des blocs de code. La syntaxe des instructions ressemble à celle du langage assembleur. Les opérations sont suivies d’opérandes.

<i>U</i>	<i>E 0.1</i>	‘ ‘Commutateur 1’ ’
<i>U</i>	<i>E 0.2</i>	‘ ‘Commutateur 2’ ’
=	<i>A 0.1</i>	‘ ‘Lampe verte’ ’
└──────────┘ └──────────┘		
<i>Opération</i>	<i>Opérande</i>	

### 3.1.3 LOG (LOGigramme)

Pour le spécialiste des circuits ou le programmeur préférant les opérations logiques, le logigramme LOG est un langage de programmation graphique qui utilise les boîtes logiques de l'algèbre de Boole pour représenter la logique. Il est également possible de représenter des fonctions complexes comme les fonctions mathématiques directement en liaison avec les boîtes logiques.



#### **Remarque :**

Il est possible, de manière générale, de représenter sans problème en LIST les programmes écrits en LOG ou en CONT. Lors de la conversion de programmes CONT en programme LOG et vice-versa, tout élément de programme ne pouvant être représenté dans le langage cible est affiché en LIST.

### 3.2 Traitement des programmes

Le traitement du programme par le processeur s'effectue dans la CPU. Cette dernière et les modules d'E/S sont reliés à un bus servant aux échanges de données.

Le processeur reçoit l'information sur l'état du signal de chaque capteur, au moyen des modules d'entrées. Il active les sorties concernées. Ces informations sont transmises au processus au moyen des modules de sorties.

Le microprocesseur se trouvant sur le module CPU, avec le logiciel système, assure le traitement du programme utilisateur.

## **4 Principe de conception d'une structure de programme**

### **4.1 Blocs dans le programme utilisation**

Le logiciel de programmation STEP 7 nous permet de structurer notre programme utilisateur, c'est à dire le subdiviser en différentes parties autonomes pour les avantages suivants :

- Ecrire des programmes importants mais clairs.
- Standardiser certaines parties du programme.
- Simplifier l'organisation du programme.
- Modifier facilement le programme.
- Faciliter la mise en service.

### **4.2 Types de bloc**

Nous pouvons utiliser différents types de blocs dans un programme utilisateur S7 :

#### ➤ **Bloc d'organisation (OB)**

Les OB déterminent la structure du programme utilisateur. Ils constituent l'interface entre le système d'exploitation et le programme utilisateur. Ils sont appelés par le système d'exploitation et gèrent le traitement de programme cyclique, assurent le déclenchement de l'alarme à la fin de chaque cycle, ainsi que le comportement à la mise en route de l'API et le traitement des erreurs.

#### ➤ **Blocs fonctionnels système (SFB) et fonctions système (SFC)**

Les SFB et SFC sont intégrés à la CPU S7 et nous permettent de réaliser quelques fonctions systèmes importantes.

#### ➤ **Blocs fonctionnels (FB)**

Les FB sont des blocs avec « mémoire » que nous programmons nous-même. Un bloc fonctionnel contient un programme qui est exécuté quand ce bloc fonctionnel est appelé par un autre bloc de code. Les blocs fonctionnels facilitent la programmation de fonctions complexes souvent utilisées.

➤ **Fonctions (FC)**

Les FC contiennent des routines de programmes pour les fonctions fréquemment utilisées. Une fonction contient un programme qui est exécuté quand cette fonction est appelée par un autre bloc de code. Nous pouvons faire appel à des fonctions pour :

- Renvoyer une valeur de fonction au bloc appelant (exemple : fonctions mathématiques)
- Pour exécuter une fonction technologique (exemple : commande individuelle avec combinaison binaire).

➤ **Blocs de données d'instance (DB d'instance)**

Un bloc de données d'instance est associé à chaque appel de bloc fonctionnel transmettant des paramètres. Ce bloc de données d'instance contient les paramètres effectifs et les données statiques du FB. Les variables déclarées dans le FB déterminent la structure du bloc de données d'instance.

Les blocs de données sont appelés par les blocs FB / SFB. Et générés automatiquement lors de la compilation.

➤ **Blocs de données (DB)**

Contrairement aux blocs de code, les blocs de données ne contiennent pas d'instructions STEP 7. Ils servent à l'enregistrement de données utilisateur, ils contiennent des données variables que le programme utilisateur utilise. Ces données utilisateur sont utilisables par tous les autres blocs. Les DB sont des zones de données dans lesquelles l'on enregistre les données utilisateur.

## **5. Programmation linéaire ou structurée**

Il est possible d'écrire notre programme utilisateur dans l'OB1 (programmation linéaire). Cela n'est toute fois recommandé que pour les programme simples s'exécutant sur des CPU S7-300 avec une mémoire peu importante.

Les automatismes complexes sont mieux traités si nous les subdivisons en tâches partielles qui sont représentées par des parties de programme (blocs) correspondantes (programmation structurée).

### *Appels des blocs*

Le programme appelle le deuxième bloc dont les opérations sont alors traitées dans leur intégralité. Une fois le bloc appelé achevé, le traitement se poursuit avec les opérations qui suivent l'appel du bloc appelé.