institut des ressources industrielles

# Intégrer un automate Siemens S7-1200 ou S7-1500 dans un système (STEP7 - TIA Portal)

## - 05 - La scrutation du programme



Formation Professionnelle VeriSelect CERTIFICATION DE SERVICES
L'AFPI LYON est certifiée VeriSelect Formation Professionnelle

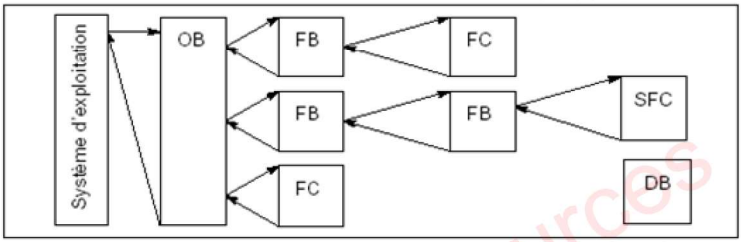
UIMM PÔLE FORMATION · LYON LA FABRIQUE DE L'AVENIR

# La scrutation du programme

## 1. Les blocs du programme utilisateur

### ❑ Introduction

✓ Le programme utilisateur est constitué de blocs de programme permettant de structurer et de décomposer le programme en différentes tâches.




### ❑ Les bloc d'ORGANISATION


✓ Les blocs d'organisation (OB) constituent l'interface entre le système d'exploitation et le programme utilisateur. Ils sont appelés par le système d'exploitation et commandent par exemple les opérations suivantes :

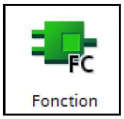
- comportement de démarrage du système d'automatisation,
- traitement cyclique du programme,
- exécution du programme déclenchée par des alarmes (cyclique, processus, diagnostic,…),
- traitement des erreurs.

✓ Pour que le traitement du programme démarre, le projet doit posséder au moins un OB cyclique (par exemple l'OB 1).

# La scrutation du programme

## 1. Les blocs du programme utilisateur

### ❑ Les blocs FONCTION




- ✓ Une fonction contient un programme qui est exécuté lorsque la fonction est appelée par un autre bloc de code. Les fonctions peuvent par exemple servir dans les cas suivants :
  - ➢ retourner des valeurs de fonction au bloc appelant, par ex. pour les fonctions mathématiques,
  - ➢ exécuter des fonctions technologiques, par ex. commandes uniques avec combinaisons binaires.
- ✓ Une fonction peut être appelée plusieurs fois à différents endroits d'un programme. Ainsi vous simplifiez la programmation de fonctions utilisées fréquemment.

### ❑ Les blocs FONCTIONNELS


- ✓ Les blocs fonctionnels sont des blocs de code qui mémorisent durablement leurs paramètres d'entrée, de sortie et d'entrée/sortie dans des blocs de données d'instance afin qu'il soit possible d'y accéder même après le traitement de blocs. C'est pourquoi ils sont également appelés "Blocs avec mémoire".
- ✓ Les blocs fonctionnels contiennent des sous-programmes qui sont exécutés lorsqu'un bloc fonctionnel est appelé par un autre bloc de code. Un bloc fonctionnel peut être appelé plusieurs fois à différents endroits d'un programme. Ainsi vous simplifiez la programmation de fonctions utilisées fréquemment.
- ✓ Un appel d'un bloc fonctionnel est désigné par le terme "instance". Pour chaque instance d'un bloc fonctionnel, il faut un bloc de données d'instance dans lequel sont mémorisées des valeurs spécifiques à l'instance pour les paramètres formels déclarés dans le FB.
  - ➢ **Exemple**: bloc fonctionnel « CDE_MOTEUR » de gestion de la commande d'un moteur appelé pour le « MOTEUR 1 » et pour le « MOTEUR 2 ».


Bloc fonctionnel


Blocs de données

- ✓ Les blocs fonctionnels peuvent aussi travailler avec des variables temporaires. Cependant, les variables temporaires ne sont pas enregistrées dans la DB d'instance mais disponibles uniquement tout le temps d'un cycle.

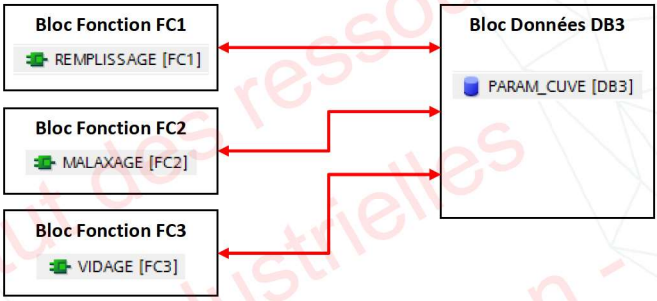
# La scrutation du programme

## 1. Les blocs du programme utilisateur

### ❑ Les blocs de DONNÉES

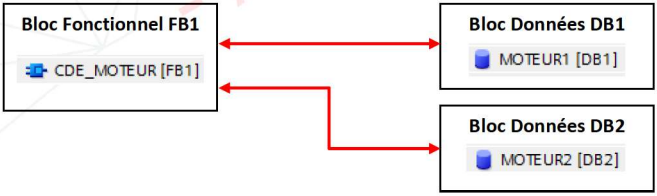

#### ✓ Les blocs de DONNÉES GLOBAUX

- Les blocs de données servent à mémoriser les données de programme. Les blocs de données contiennent donc des données variables qui sont utilisées dans le programme utilisateur.
- Les blocs de données globaux enregistrent des données qui peuvent être utilisées par tous les autres blocs.




#### ✓ Les blocs de DONNÉES D'INSTANCE

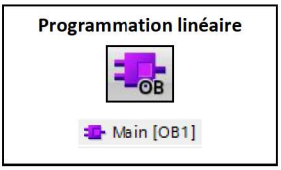
- L'appel d'un bloc fonctionnel est une instance. Les données avec lesquelles opère l'instance sont mémorisées dans un bloc de données d'instance.
- La taille maximale des blocs de données d'instance varie selon la CPU. Les variables déclarées dans le bloc fonctionnel déterminent la structure du bloc de données d'instance.




- ❖ Les blocs de données d'instance ne sont utilisés que par le bloc fonctionnel d'instance associé.

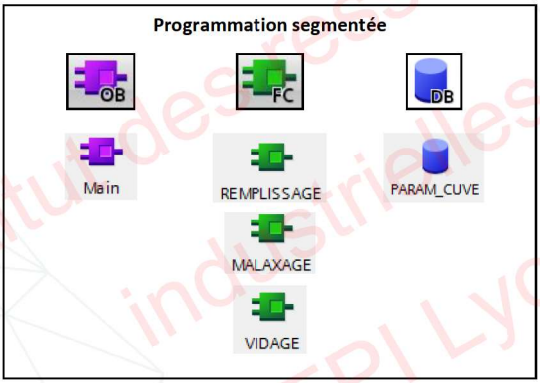
## 2. Les types de programmation

### ❑ La programmation linéaire




- ✓ Il est possible de résoudre de petites tâches d'automatisation en écrivant le programme utilisateur complet linéairement dans un **OB cyclique**.
- ✓ Toutes les instructions sont contenues dans un seul bloc (**OB1**).
- ✓ Cette démarche est recommandée uniquement pour des programmes simples.

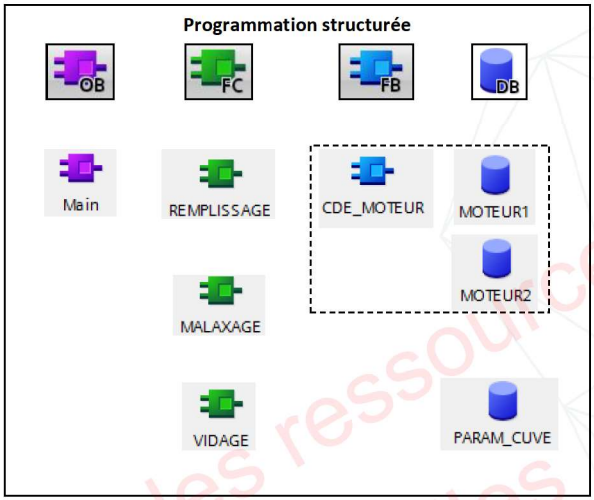
### ❑ La programmation segmentée




- ✓ La réalisation et la maintenance de tâches d'automatisation complexes sont plus simples si ces tâches sont divisées en plusieurs tâches partielles plus petites qui correspondent aux fonctions technologiques du processus d'automatisation ou qui peuvent être utilisées plusieurs fois. Dans le programme utilisateur, ces tâches partielles sont représentées par des blocs.
- ✓ Chaque bloc constitue une section indépendante du programme utilisateur.
- ✓ La structuration du programme offre les avantages suivants :
  - ➢ la programmation de programmes volumineux est plus claire,
  - ➢ l'organisation du programme est simplifiée,
  - ➢ il est plus facile de modifier le programme,
  - ➢ le test du programme est simplifié, car il peut s'effectuer section par section,
  - ➢ la mise en service est simplifiée.

## 2. Les types de programmation

### ❑ La programmation structurée




✓ En plus de la programmation segmentée, on utilise des blocs fonctionnels (FB) et des blocs de données d'instance (DB) pour programmer les fonctions réutilisables.

## 3. Le traitement du programme

### ❑ Introduction

- ✓ Il est fréquent de programmer le programme utilisateur avec un OB de cycle de programme, généralement dans l'OB 1.
- ✓ Dans les applications complexes, il est souvent nécessaire de respecter des temps de réaction courts exigés par l'application.
- ✓ Il est possible de répondre aux exigences de temps de réaction subdivisant le programme utilisateur en parties ayant des exigences différentes pour les temps de réaction.
- ✓ La CPU propose toute une série de types d'OB différents pour adapter les propriétés (priorité, fréquence...) aux exigences.

### ❑ L'organisation du programme

- ✓ **Traitement dans le programme cyclique de la CPU**
  - ➢ La CPU traite le programme utilisateur de manière cyclique.
  - ➢ Lorsque le traitement est arrivé à la fin d'un cycle, le traitement du programme recommence à nouveau dans le cycle suivant.
  - ➢ Dans le cas le plus simple, l'intégralité du programme utilisateur est exécutée dans le programme cyclique de la CPU. Toutes les tâches du programme utilisateur sont alors exécutées au même niveau. Ainsi, toutes les tâches sont également soumises aux mêmes temps de réaction.

- ✓ **Traitement déclenché par horloge**
  - ➢ Dans un programme utilisateur complexe, il y a souvent des parties présentant des exigences différentes en ce qui concerne le temps de réaction.
  - ➢ Il est possible de scinder ainsi les parties du programme ayant des exigences plus élevées en temps de cycle dans des blocs d'organisation (OB) de priorité supérieure avec des temps de cycle court (ex: **OB d'alarme cyclique**).
  - ➢ Ainsi, le traitement de ces parties peut avoir lieu à fréquence variable selon différentes priorités.

- ✓ **Traitement déclenché par un événement**
  - ➢ Selon les modules de périphérie utilisés, il est possible de configurer pour certains événements du processus (ex: changement de front sur une entrée TOR) des alarmes de processus qui entraînent l'appel de l'OB d'alarme de processus associé.
  - ➢ Les **alarmes de processus** ont une priorité supérieure et interrompent le programme cyclique de la CPU.
  - ➢ Le déclenchement direct du traitement du programme permet d'atteindre des temps de réaction très courts avec les alarmes de processus dans la périphérie centralisée.

## 3. Le traitement du programme

❑ **L'utilisation des mémoires images partielles**

- ✓ Lorsque le programme est réparti sur différents OB, à cause d'exigences différentes concernant les temps de réaction, il est souvent nécessaire d'affecter directement à ces OB l'actualisation des données de périphérie utilisées.
- ✓ Dans une mémoire image partielle, les données d'entrée et de sortie sont regroupées en fonction de leur utilisation dans le programme et affectées à l'OB.
- ✓ Une mémoire image des entrées partielle (**MIEP**) permet d'actualiser les données d'entrées d'un programme OB juste avant le démarrage du programme OB.
- ✓ Une mémoire image des sorties partielle (**MISP**) permet de fournir les données de sortie du programme OB, aux sorties juste après l'exécution du programme OB.
- ✓ **32 mémoires images partielles** (0 à 31)
- ✓ La périphérie est affectée par défaut à la mémoire image partielle 0 (paramétrage : "Actualisation automatique").
- ✓ La **mémoire image partielle 0** est affectée de manière fixe **au traitement cyclique**.

❑ **Les priorités des blocs « OB » et des activités système**

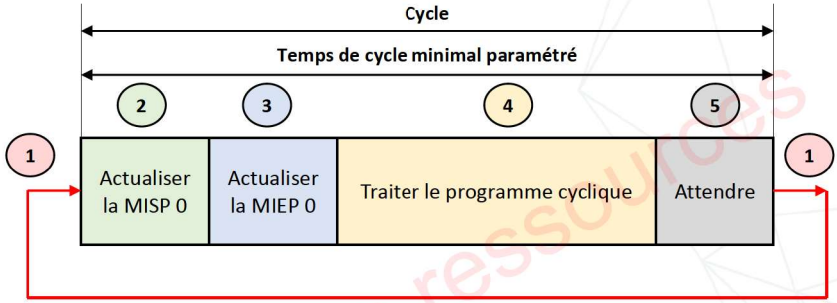
- ✓ Chaque bloc d'organisation est traité selon sa priorité affectée en fonction des exigences concernant le temps de réaction.
- ✓ Tous les OB de cycle de programme ont la priorité la plus basse égale à 1. La priorité la plus élevée est 26.
- ✓ Les **tâches de communication** ont toujours la **priorité 15**.
- ✓ Il est possible de modifier la priorité des blocs et sélectionner une priorité supérieure à celle de la communication.
- ✓ Les blocs d'organisation ou les activités système de priorité supérieure interrompent les blocs d'organisation et activités système de priorité plus faible.

## 4. Le traitement cyclique du programme

### ❑ Le cycle

✓ Un cycle comprend les opérations suivantes :

- ➢ Actualisation automatique de la mémoire image partielle 0 des sorties (MISP 0)
- ➢ Actualisation automatique de la mémoire image partielle 0 des entrées (MIEP 0)
- ➢ Traitement du programme cyclique




1. Point de contrôle du cycle auquel le système d'exploitation démarre la mesure du temps de cycle
2. La CPU écrit les états de la mémoire image des sorties dans les modules de sorties
3. La CPU lit l'état des entrées dans les modules d'entrées et écrit les données d'entrée dans la mémoire image des entrées
4. La CPU traite le programme utilisateur et exécute les opérations indiquées dans le programme
5. Phase d'attente jusqu'à la fin du temps de cycle minimal

### ❑ Le point de contrôle du cycle

✓ Lorsque le point de contrôle de cycle est atteint, la CPU a terminé le programme cyclique et elle n'exécute plus d'OB.

✓ A ce point, toutes les données utiles sont cohérentes. La condition est qu'aucune communication modifiant les données utiles ne soit active (ex: communication IHM).

✓ Le point de contrôle du cycle marque :

- ➢ la fin d'un cycle et ses statistiques de temps de cycle
- ➢ le début du cycle suivant et ses statistiques de temps de cycle
- ➢ le redémarrage de la surveillance du temps de cycle maximal paramétré (le compteur de dépassement haut de temps de cycle est réinitialisé)

✓ Le point de contrôle de cycle est atteint en fonction de celui des événements suivants apparaissant en dernier :

- ➢ Fin du dernier OB de cycle de programme
- ➢ Fin du temps de cycle minimal (si configuré)

✓ Une fois le point de contrôle de cycle atteint, la CPU effectue les étapes 2, 3, 4 (Exécution du premier OB de cycle de programme).

## 4. Le traitement cyclique du programme

### ❑ Le temps de cycle

✓ Le temps de cycle est le temps nécessaire à la CPU pour :

- ➢ l'actualisation de la mémoire image des entrées et des sorties
- ➢ le traitement du programme cyclique
- ➢ Le traitement de toutes les parties du programmes et les activités système qui interrompent ce cycle
- ➢ attendre la fin du temps d'exécution minimum (si ce dernier est paramétré et est plus long que le temps de traitement du programme)

### ❑ Les différents temps de cycle

✓ Le temps de cycle (Tcyc) n'est pas identique dans chaque cycle car les temps de traitement peuvent varier.

✓ Les temps d'exécution du programme peuvent varier en fonction:

- ➢ des boucles de programmes,
- ➢ des commandes conditionnelles,
- ➢ des appels de blocs conditionnels,
- ➢ des différences de chemin du programme.

✓ Les interruptions allongent le temps de cycle:

- ➢ Traitement d'alarme déclenché par horloge
- ➢ Traitement de l'alarme déclenché par un événement
- ➢ Communication

### ❑ Le temps de cycle minimal

✓ Le temps de cycle minimal par défaut des CPU non redondantes est d'une milliseconde

✓ Il est judicieux d'augmenter ce paramètre dans les cas suivants :

- ➢ Pour réduire la plage de variation du temps de cycle.
- ➢ Pour disposer du temps de calcul restant pour les tâches de communication. La CPU traite alors ces tâches de communication jusqu'à écoulement du temps de cycle minimum.

✓ La mise à disposition du temps de calcul restant pour les tâches de communication présente les avantages suivants :

- ➢ Des temps de cycle minimum plus long permettent d'éviter une actualisation inutilement fréquente des mémoires image et réduisent ainsi la charge du bus interne.
- ➢ Des temps de cycle minimum plus long assurent des performances de communication supérieures.

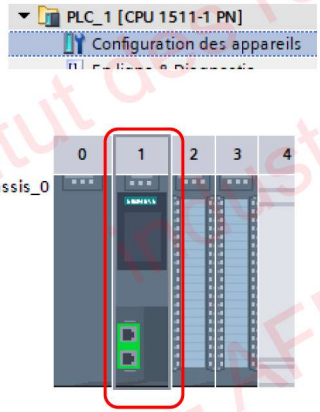
## 4. Le traitement cyclique du programme

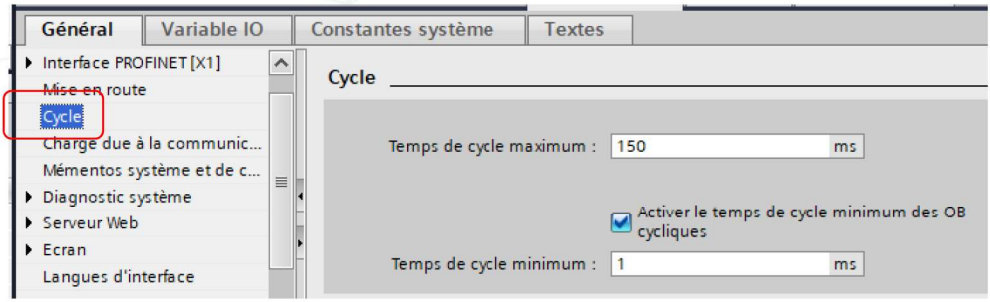
### ❑ Le temps de cycle maximal

- ✓ Le temps de cycle maximal est une limite supérieure paramétrable du temps d'exécution du programme cyclique.
- ✓ La tâche du temps de cycle maximal consiste à surveiller le temps de réaction requis pour le processus correspondant.
- ✓ Le temps de cycle maximal des CPU non redondantes est réglé par défaut sur **150 ms**.
- ✓ Le paramétrage de la CPU vous permet de régler cette valeur entre **1 ms et 6 000 ms**.
- ✓ Si le temps du cycle en cours de traitement dépasse le temps de cycle maximal, l'OB d'erreur de temps (**OB 80**) est appelé.
- ✓ Si l'OB 80 n'est pas programmé, la CPU passe en STOP dès le 1er dépassement du temps de cycle.
- ✓ Si l'OB 80 est programmé, , la CPU passe en STOP au 2ème dépassement du temps de cycle.

### ❑ Réglage du temps de cycle

- ✓ Ouvrir la configuration matérielle de la CPU



## 5. Les principaux OB de démarrage et d'alarme

- **Les OB cycliques (Main)**
  - ✓ Exemple: **OB1**
  - ✓ Le programme cyclique constitue la base essentielle du programme utilisateur. Ses sections de programme correspondantes sont traitées de manière cyclique : lorsque son traitement est terminé, le système d'exploitation en relance l'exécution.
  - ✓ Le programme cyclique comprend un ou plusieurs OB cycliques. Ceux-ci ont la priorité 1 (la plus basse).

- **Les OB de démarrage (Startup)**
  - ✓ Exemple: **OB100**
  - ✓ Le système d'exploitation appelle une seule fois chaque OB de démarrage lors du passage de l'état « STOP" à l'état « RUN".
  - ✓ Les valeurs de la mémoire image des entrées sont mises à 0.
  - ✓ S'il existe plusieurs OB de démarrage, ils sont appelés dans l'ordre de leur numéro d'OB en commençant par le plus petit.
  - ✓ Une fois le traitement du programme de démarrage terminé, le système d'exploitation lit la mémoire image des entrées et démarre le programme cyclique.

- **Les OB d'alarme cycliques (Cyclic interrupt)**
  - ✓ Exemple: **OB30**
  - ✓ Les OB d'alarme cyclique permettent de démarrer des programmes indépendamment du traitement cyclique du programme, dans des intervalles de temps périodiques.
  - ✓ L'intervalle de temps est l'intervalle entre deux appels. Il s'agit d'un multiple entier de l'impulsion de base de 1 µs.
  - ✓ Le décalage de phase est l'intervalle de temps duquel les heures de déclenchement sont décalées par rapport aux multiples de l'intervalle de temps.
  - ✓ **Exemple**:
    - ➢ Vous utilisez deux OB d'alarme cyclique dans votre programme : un avec un intervalle de temps de 20 ms, l'autre de 100 ms. Afin de garantir qu'ils ne soient pas appelés simultanément en cas de multiples entiers de 100 ms, utilisez un décalage de phase

- **Les OB d'alarme de processus (Hardware interrupt)**
  - ✓ Exemple: **OB40**
  - ✓ Les OB d'alarme du processus interrompent le traitement cyclique du programme en réponse à un événement matériel (ex: dépassement d'un seuil haut d'une entrée analogique).
  - ✓ Plusieurs événements peuvent être affectés à un OB d'alarme de processus.

- **Les OB d'erreur de temps (Time error interrupt)**
  - ✓ Exemple: **OB80**
  - ✓ Les OB d'erreur de temps interrompent le traitement cyclique du programme lorsque le temps de cycle maximum est dépassé.