

Intégrer un automate M340 ou TSX Premium dans un système Schneider Electric (Unity Pro)

- 10 - Unity Pro: les blocs fonction utilisateur DFB

1. Présentation des blocs fonctions utilisateur (DFB)

3

- ☐ Présentation
- ☐ Avantage d'utiliser un DFB
- ☐ Comparaison avec un sous-programme

2. Mise en œuvre d'un bloc fonction DFB

4

- ☐ Procédure de mise en œuvre
- ☐ Création du type DFB
- ☐ Description du type de DFB
- ☐ Utilisation des instances de DFB

3. Exemple d'application

7

- ☐ Présentation
- ☐ Configuration du projet
- ☐ Création et paramétrage du modèle « Types DFB »
- ☐ Programmation du modèle « Types DFB »
- ☐ Création d'une instance de DFB
- ☐ Exécution d'une instance de DFB
- ☐ Utilisation de la table d'animation de l'instance de DFB

1. Présentation des blocs fonctions utilisateur (DFB)

❑ Présentation

- ✓ Un DFB est un bloc de programme écrit afin de répondre aux spécificités de l'application. Il comprend:
 - une ou plusieurs sections écrites en langage à contacts (LD), en liste d'instructions (IL), en littéral structuré (ST) ou en langage à blocs fonctionnels (FBD),
 - des paramètres d'entrée/de sortie,
 - des variables internes publiques ou privées.
- ✓ Les blocs fonction permettent de structurer et d'optimiser l'application. Ils sont utilisés dès qu'une séquence de programme est répétée plusieurs fois dans l'application ou pour figer une programmation standard (par exemple, l'algorithme de commande d'un moteur incluant la prise en compte des sécurités locales).
- ✓ L'export puis l'import de ces blocs fonction permet leur utilisation par un groupe de programmeurs travaillant sur une même application ou dans des applications différentes.

❑ Avantage d'utiliser un DFB

- ✓ L'utilisation d'un bloc fonction DFB dans une application vous permet :
 - de simplifier la conception et la saisie du programme,
 - d'accroître la lisibilité du programme,
 - de faciliter la mise au point de l'application (toutes les variables manipulées par le bloc fonction sont identifiées sur son interface),
 - de diminuer le volume de code généré (le code correspondant au DFB est chargé une seule fois, quel que soit le nombre d'appels au DFB dans le programme, seules les données correspondants aux instances sont générées).

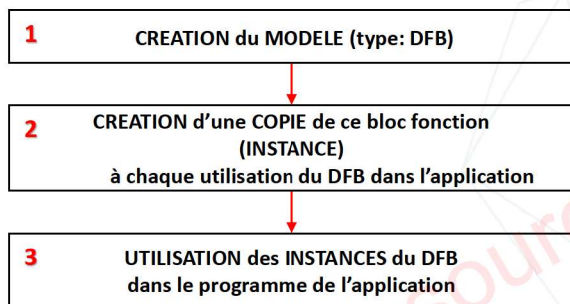
❑ Comparaison avec un sous-programme

- ✓ Par rapport à un sous programme, l'utilisation d'un DFB permet :
 - de paramétrer plus facilement le traitement,
 - d'utiliser des variables internes propres au DFB, donc indépendantes de l'application,
 - de tester son fonctionnement indépendamment de l'application.
- ✓ De plus, les langages LD et FBD permettent de visualiser de manière graphique les DFB, ce qui facilite la conception et la mise au point du programme.

2. Mise en œuvre d'un bloc fonction DFB

❑ Procédure de mise en œuvre

- ✓ Elle comporte **3 étapes**:

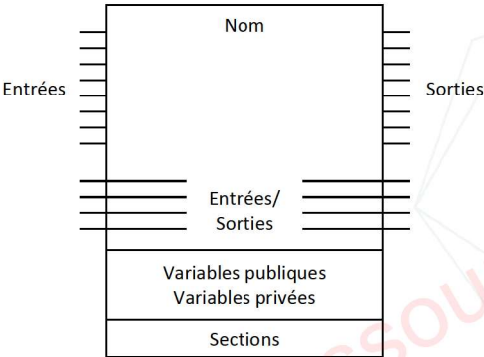


❑ Création du type DFB

- ✓ Cette opération consiste à concevoir un modèle du DFB qui sera utilisé dans l'application. Pour ce faire, l'éditeur DFB est utilisé pour définir et coder tous les éléments qui constituent le DFB :
 - description du bloc fonction : nom, type (DFB), activation du diagnostic, commentaire.
 - structure du bloc fonction : paramètres, variables, sections de code.
- ✓ Si un DFB existant dans la bibliothèque définie par l'utilisateur est modifié alors le nouveau type sera utilisé pour toute instance supplémentaire du projet ouvert. Cependant, la bibliothèque définie par l'utilisateur reste inchangée.

2. Mise en œuvre d'un bloc fonction DFB

❑ Description du type de « DFB »



Nom	nom du type DFB (32 caractères max.) Ce nom doit être unique dans les bibliothèques,
Entrées	paramètres d'entrée
Sorties	paramètres de sortie
Entrées/sorties	Paramètres d'entrées/sorties
Variables publiques	Ces variables internes du DFB peuvent être utilisées par le DFB, par le programme application et par l'utilisateur en mode réglage.
Variables privées	Ces variables internes du DFB peuvent être utilisées uniquement par ce bloc fonction et ne sont par conséquent pas accessibles par le programme application, mais ces types de variables sont accessibles via le tableau d'animation. Ces variables sont généralement des variables nécessaires à la programmation du bloc, mais sans intérêt pour l'utilisateur (résultat d'un calcul intermédiaire, par exemple).
Sections	sections de code DFB dans LD, IL, ST ou FBD.
Commentaire	1024 caractères maximum

2. Mise en œuvre d'un bloc fonction DFB

❑ Utilisation des instances de DFB

- ✓ Une instance DFB est utilisée comme suit
 - en tant que bloc fonction standard dans un langage à contacts (LD) ou à blocs fonction (FBD),
 - en tant que fonction élémentaire dans un langage littéral structuré (ST) ou liste d'instructions (IL).
- ✓ Une instance DFB peut être utilisée dans toutes les tâches du programme d'application, sauf les tâches événementielles ou les transitions de diagramme fonctionnel en séquence (SFC).

institut des ressources
industrielles
- AFPI Lyon -



3. Exemple d'application

□ Présentation

- ✓ L'objectif est de créer un bloc fonction utilisateur DFB utilisé pour commander et contrôler la commande d'un moteur à un sens de marche.
- ✓ **Les informations d'entrées:**
 - **Marche:** commande de mise en service du moteur
 - **Arrêt:** commande d'arrêt du moteur
 - **Contrôle:** information sur l'état du contacteur (désactivé, activé)
 - **Thermique:** défaut moteur
 - **Sécurité:** arrêt d'urgence
 - **Acquittement:** annulation des défauts
 - **TempoMa:** temps enveloppe pour l'activation du contacteur
 - **TempoAr:** temps enveloppe pour la désactivation du contacteur
- ✓ **Les ordres de sorties:**
 - **Contacteur:** commande du contacteur du moteur
 - **En Service:** voyant « moteur en service »
 - **Défaut:** voyant « défaut moteur »

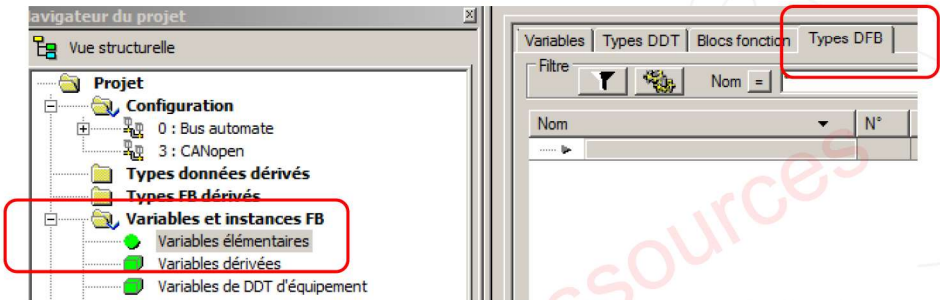
□ Configuration du projet

- ✓ Créer le projet DFB_Moteurs »
- ✓ Réaliser la configuration matérielle et logicielle
- ✓ Déclarer les variables élémentaires suivantes:

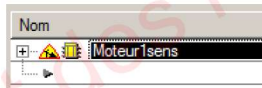
Variables Types DDT Blocs fonction Types DFB				
Filtre   Nom = <input type="text"/>				
Nom	Adresse	Type	Commentaire	
BpArret1	%I0.1.0	EBOOL	BP arrêt moteur 1	
BpArret2	%I0.1.5	EBOOL	BP arrêt moteur 2	
BpMa1	%I0.1.1	EBOOL	BP marche moteur 1	
BpMa2	%I0.1.6	EBOOL	BP marche moteur 2	
CtlKm1	%I0.1.2	EBOOL	Etat du contacteur KM1	
CtlKm2	%I0.1.7	EBOOL	Etat du contacteur KM2	
Rth1	%I0.1.3	EBOOL	Relais thermique du moteur 1	
Rth2	%I0.1.8	EBOOL	Relais thermique du moteur 2	
BpAcq1	%I0.1.4	EBOOL	BP acquittement défaut moteur 1	
BpAcq2	%I0.1.9	EBOOL	BP acquittement défaut moteur 2	
Hes1	%Q0.2.0	EBOOL	Voyant moteur 1 en service	
Hes2	%Q0.2.4	EBOOL	Voyant moteur 2 en service	
Hdef1	%Q0.2.1	EBOOL	Voyant défaut moteur 1	
Hdef2	%Q0.2.5	EBOOL	Voyant défaut moteur 2	
KM1	%Q0.2.2	EBOOL	Commande contacteur KM1	
KM2	%Q0.2.6	EBOOL	Commande contacteur KM2	
Au1	%Q0.2.3	EBOOL	Arrêt d'urgence	

3. Exemple d'application

- ❑ **Création et paramétrage du modèle « Types DFB »**
 - ✓ Ouvrir la fenêtre « Variables élémentaires » puis sélectionner « Types DFB »



- ✓ Nommer le DFB:



- ✓ Déclarer les variables du DFB:

Nom	N°	Type	Commentaire
Moteur1sens		<DFB>	
<entrées>			
Marche	1	EBOOL	Commande de mise en service du moteur
Arrêt	2	EBOOL	Commande d'arrêt du moteur
Contrôle	3	EBOOL	Info sur l'état du contacteur (activé, désactivé)
Thermique	4	EBOOL	Contact du relais magnétothermique
Securite	5	EBOOL	Arrêt d'urgence
Acquittement	6	EBOOL	Annulation du défaut
TempoMa	7	TIME	Temps enveloppe d'activation du contacteur
TempoAr	8	TIME	Temps enveloppe de désactivation du contacteur
<sorties>			
Contacteur	1	EBOOL	Commande du contacteur du moteur
EnService	2	EBOOL	Voyant "moteur en fonctionnement"
Defaut	3	EBOOL	Voyant "défaut moteur"
<entrées/sorties>			
<public>			
MemContacteur		BOOL	Mémoire "commande contacteur"
MemDefaut		BOOL	Mémoire "défaut moteur"

- ✓ Générer le projet

3. Exemple d'application

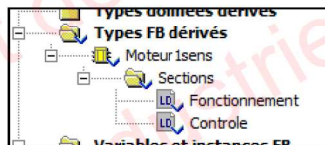
- ❑ **Programmation du modèle « Types DFB »**
 - ✓ Ouvrir l'onglet « sections » du Type DFB:

Nom	N
Moteur1sens	
+ <entrées>	
+ <sorties>	
+ <entrées/sorties>	
+ <public>	
+ <privé>	
+ <sections>	

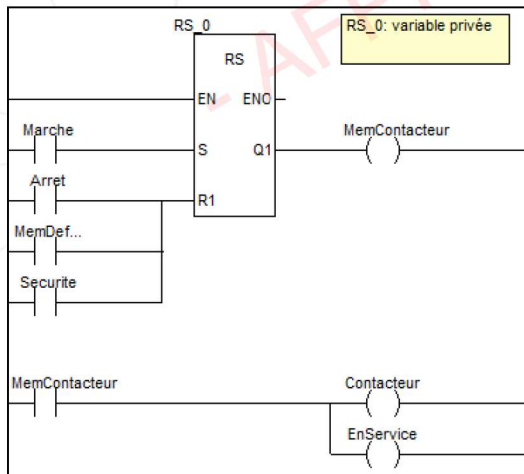
- ✓ Créer les sections « FONCTIONNEMENT » et « CONTROLE » en langage « LD »

<sections>		
+ Fonctionnement	<LD>	
+ Controle	<LD>	

- Le « Types DFB » apparaît dans l'onglet « **Types FB dérivés** » du navigateur



- Programmer les sections
 - ❖ Section « FONCTIONNEMENT »

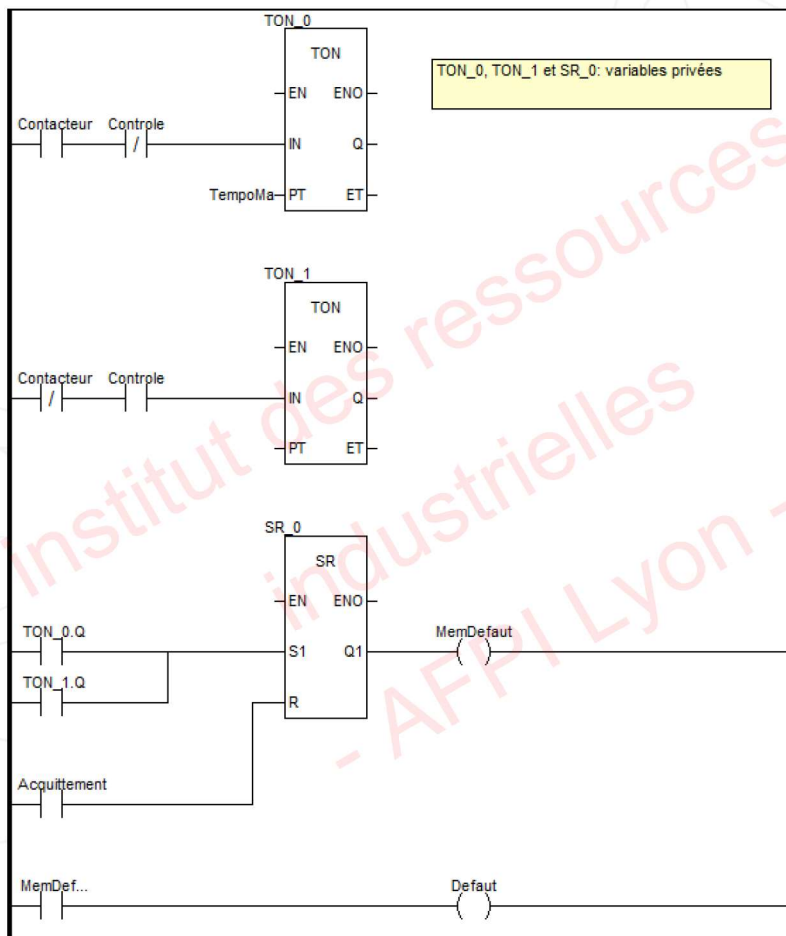


3. Exemple d'application

□ Programmation du modèle « Types DFB »

➤ Programmer les sections

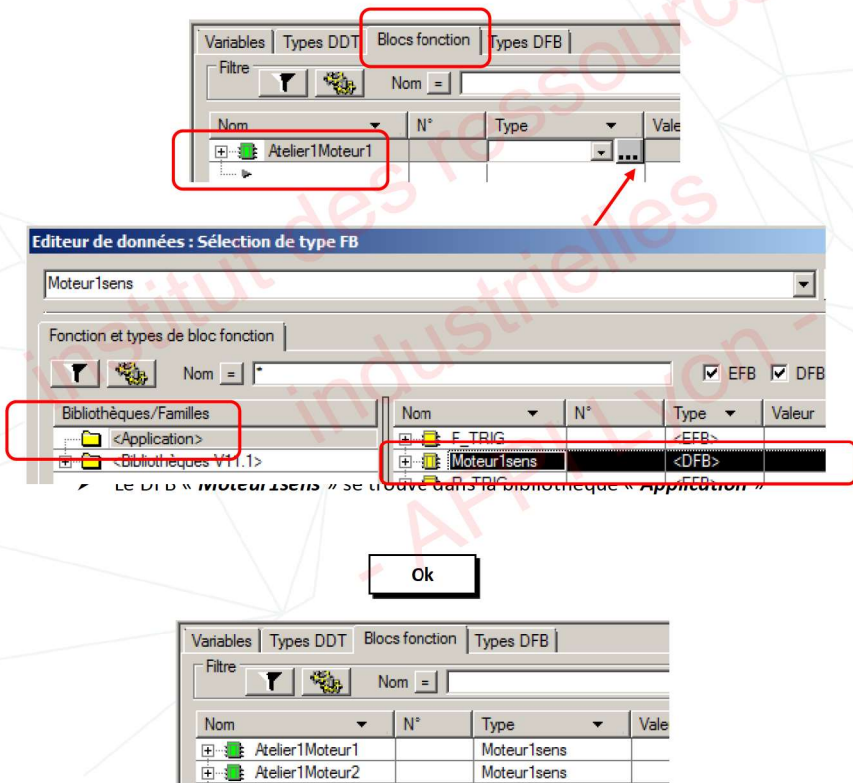
❖ Section « CONTROLE »



3. Exemple d'application

❑ Création d'une instance de DFB

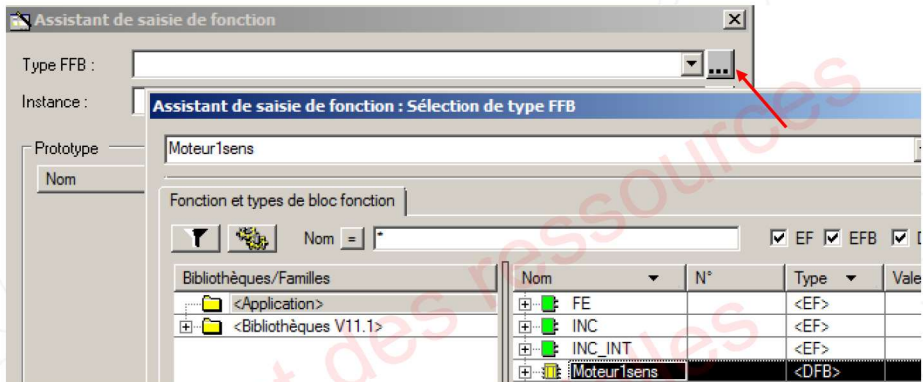
- ✓ Une instance de DFB est une copie du modèle de DFB (type de DFB):
 - elle exploite le code du type de DFB,
 - elle crée une zone de données spécifique à cette instance, qui est la recopie des paramètres et des variables du type de DFB. Cette zone est située dans l'espace données de l'application.
- ✓ Chaque instance de DFB créée est repérée par un nom de 32 caractères maximum. Le 1^{er} caractère doit être une lettre.
- ✓ Pour créer les instances de DFB des moteurs 1 et 2 de l'atelier 1, sélectionner « Bloc fonction » dans « variables élémentaires » puis déclarer les instances:



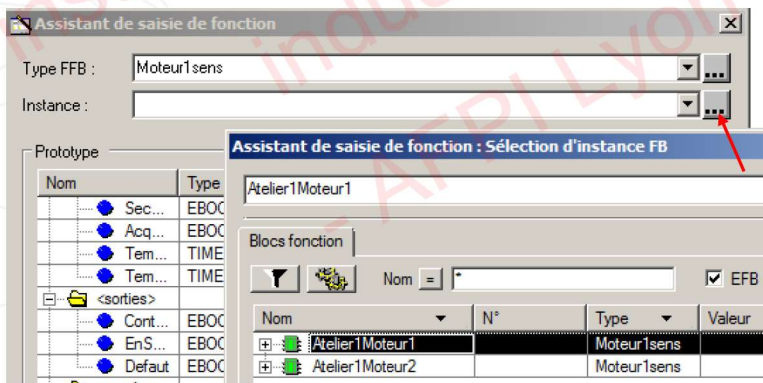
- Les instances « **Atelier1 Moteur1** » et « **Atelier1 Moteur2** » sont de type « **Moteur1sens** » et seront utilisées pour la commande et le contrôle des moteurs 1 et 2 de l'atelier 1.

3. Exemple d'application

- ❑ Exécution d'une instance de DFB
 - ✓ Utiliser l'assistant de saisie FFB pour programmer le bloc fonction.



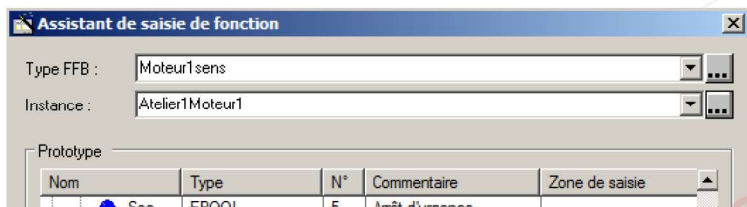
- ✓ Sélectionner le bloc fonction « **Moteur1sens** » puis l'instance « **Atelier1Moteur1** »



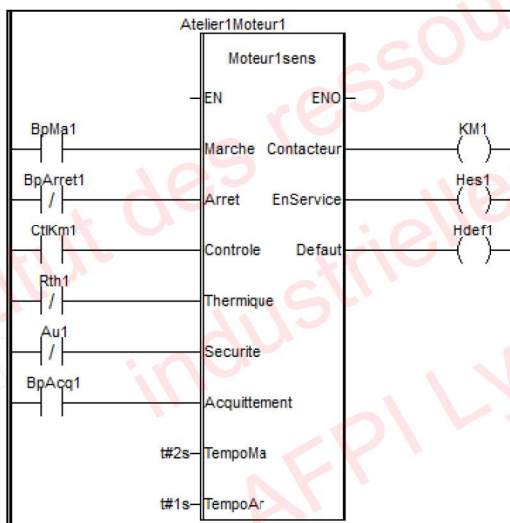
Ok

3. Exemple d'application

❑ Exécution d'une instance de DFB



- ✓ Poser le bloc sur la grille de la section « **Atelier1** » puis compléter le schéma:



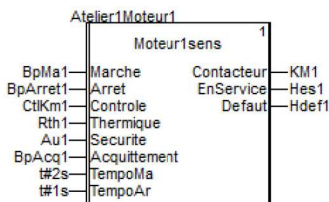
- ✓ Programmer le bloc fonction utilisateur « **Atelier1Moteur2** » en utilisant le même procédé.

3. Exemple d'application

❑ Exécution d'une instance de DFB

✓ Programmation en langage « FBD »

- Utiliser l'assistant de saisie FFB pour programmer le bloc fonction.



✓ Programmation en langage « IL »

```
(* Moteur 1 de l'atelier 1 *)  
  
CAL Atelier1Moteur1 (  
    Marche := BpMa1,  
    Arret := BpArret1,  
    Controle := CtlKm1,  
    Thermique := Rth1,  
    Securite := Au1,  
    Acquitement := BpAcq1,  
    TempoMa := t#2s,  
    TempoAr := t#1s,  
    Contacteur => KM1,  
    EnService => Hes1,  
    Default => Hdef1  
)
```

✓ Programmation en langage « ST »

```
(* Moteur 1 de l'atelier 1 *)  
  
Atelier1Moteur1 (Marche := BpMa1,  
    Arret := BpArret1,  
    Controle := CtlKm1,  
    Thermique := Rth1,  
    Securite := Au1,  
    Acquitement := BpAcq1,  
    TempoMa := t#2s,  
    TempoAr := t#1s,  
    Contacteur => KM1,  
    EnService => Hes1,  
    Default => Hdef1);
```

3. Exemple d'application

- ✓ Sélectionner le bloc fonction puis ouvrir la table d'animation du bloc:

