Document de formation
pour une solution complète d'automatisation
Totally Integrated Automation (T I A)

MODULE B5
Programmation structurée avec les blocs fonctionnels (FB)

Nous remercions l'entreprise Michael Dziallas Engineering et les enseignants d'écoles professionnelles ainsi que tous ceux qui ont participé à l'élaboration de ce document.

PAGE :

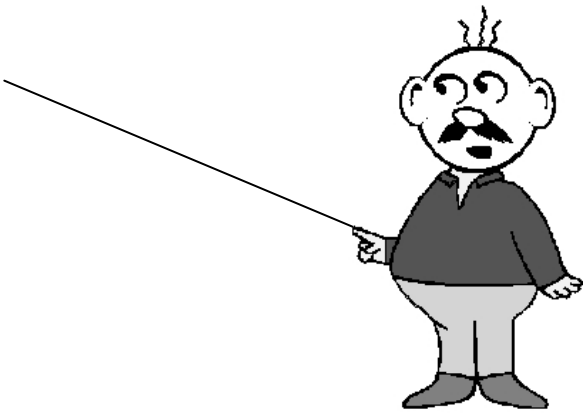
Légende des symboles utilisés dans ce module :

Information

Programmation

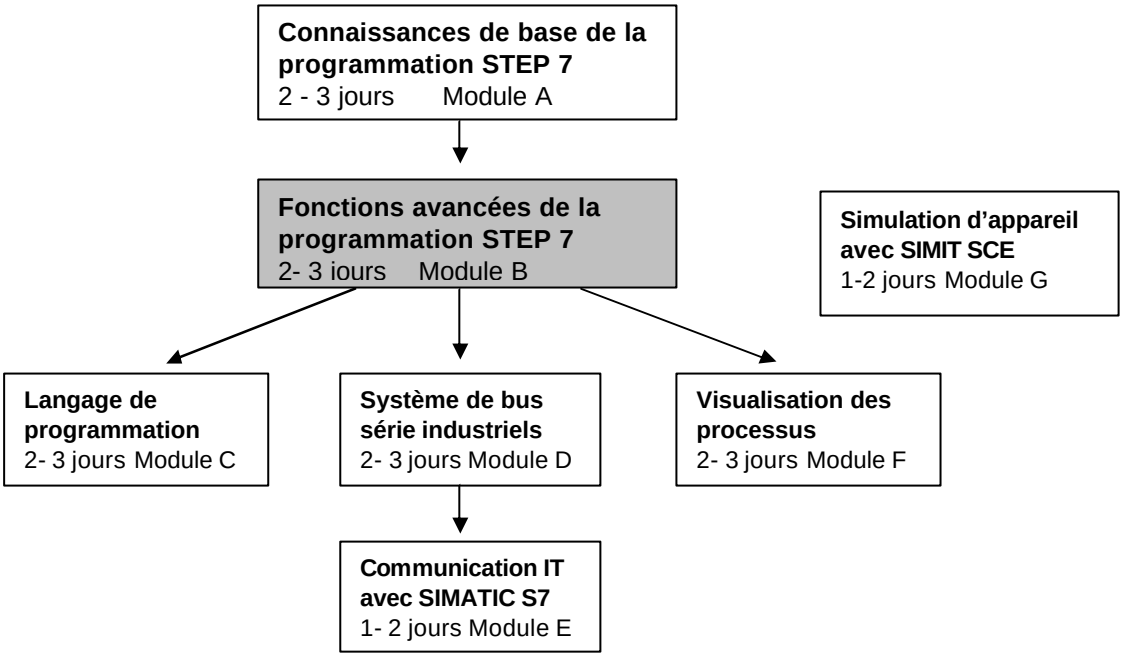
Exemple

Indications



1. AVANT-PROPOS

Le contenu du module B5 est assigné à l'unité **‚Fonctions avancées de la programmation STEP7'.**



Objectifs :

Dans ce module, le lecteur apprendra à créer et utiliser les blocs fonctionnels avec variables internes dans le cadre d'une programmation structurée.

- Création d'un bloc fonctionnel (FB)
- Définition de variables internes
- Programmation d'un bloc fonctionnel avec variables internes
- Appel et paramétrage d'un bloc fonctionnel à partir de l'OB1

Pré-requis :

Les connaissances suivantes sont requises pour l'étude de ce module :

- Systèmes d'exploitation : Windows 95/98/2000/ME/NT4.0/XP
- Bases en programmation SPS avec STEP7 (Ex : Module A3 ‚Startup', programmation SPS avec STEP 7)
- Connaissances de base en programmation structurée

Configurations matérielles et logicielles requises

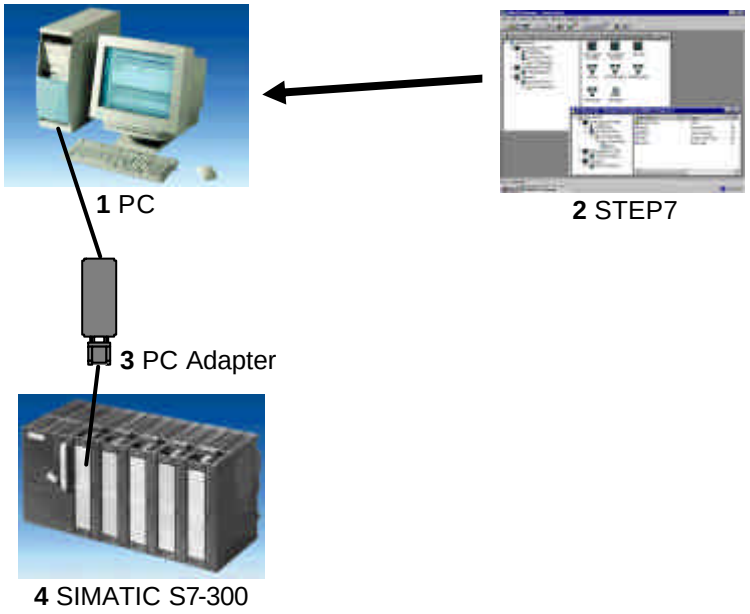
1 PC, système d'exploitation : Windows 95/98/2000/ME/NT4.0/XP avec
- Minimum : 133MHz et 64Mo RAM, 65 Mo d'espace disponible
- Optimal : 500MHz et 128Mo RAM, 65 Mo d'espace disponible

2 Logiciel STEP 7 V 5.x

3 Interface ordinateur MPI (Ex : PC- Adapter)

4 SPS SIMATIC S7-300

Exemple de configuration :
- Bloc d'alimentation : PS 307 2A
- CPU : CPU 314
- Entrées numériques : DI 16x DC24V
- Sorties numériques : DO 16x DC24V / 0,5 A



2. REMARQUES SUR LA PROGRAMMATION STRUCTUREES A L'AIDE DE FCS ET FBS

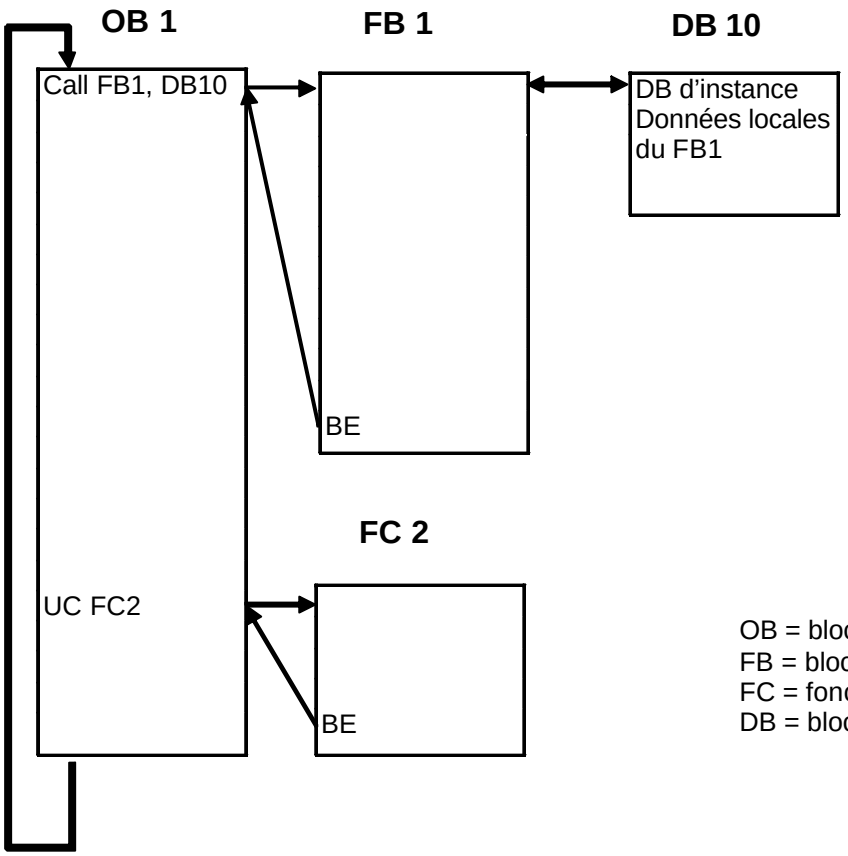
STEP 7 permet de répartir le programme utilisateur en différents blocs de programme. Le bloc d'organisation OB1 est présent par défaut.
Ce bloc constitue l'interface avec le système d'exploitation de la CPU, il est automatiquement appelé par celui-ci et est exécuté de façon cyclique.
Dans le cas de grosses applications, le programme pourra être subdivisé en blocs de programme plus petits, formant ainsi de petites parties autonomes, et assurant entre autre, une meilleure lisibilité.
Ces blocs seront appelés à partir du bloc d'organisation grâce aux commandes d'appel de bloc (Call xx / UC xx / CC xx). La fin d'un bloc est signalisée par la commande BE, à la rencontre de celle-ci, le programme utilisateur poursuit son exécution dans le bloc appelant juste après la commande d'appel de bloc.

STEP 7 propose différents types de blocs utilisateurs pour la programmation structurée :

- FB (bloc fonctionnel) :
  un FB dispose d'un espace mémoire. Lors de l'appel d'un FB, un bloc de données (DB) peut lui être alloué, il aura alors accès aux données contenues dans ce DB d'instance local. Plusieurs DBs peuvent être alloués à un même FB. D'autres FBs ou FCs peuvent être appelés (grâce aux commandes Call / UC / CC) à l'intérieur d'un FB.

- FC (fonction) :
  une FC ne dispose pas d'espace mémoire, les données locales sont perdues après l'exécution de la fonction. D'autres FBs ou FCs peuvent être appelés (grâce aux commandes Call / UC / CC) à l'intérieur d'une FC.

Un tel programme pourrait avoir la structure suivante :



Indication : Pour utiliser plusieurs fois un même bloc utilisateur, il est également possible de programmer ces FCs et FBs sous la forme de blocs standards utilisant des variables internes. Ceux-ci peuvent alors être appelés autant de fois que souhaités, bien qu'un nouveau DB d'instance doit être alloué à chaque appel de l'FB.

3. CREATION D'UN BLOC FONCTIONNEL (FB) AVEC DECLARATION DE VARIABLES

Il peut être intéressant de programmer dans Step7 des blocs de type « black box » qui permettent de réaliser une fonctionnalité précise de manière transparente, et indépendamment du contexte ou du programme appelant. Ce type de bloc ne peut naturellement pas contenir d'adressage absolu d'entrées/sorties, de mémentos, timers ou compteurs… etc. L'utilisation de variables, de constantes et le passage de paramètres prennent alors tout leur sens.

L'exemple suivant explicite la création d'un bloc fonctionnel assurant la commande d'un moteur en fonction du type de fonctionnement choisi, il contiendra également un compteur de cycle.

Le bouton poussoir 'S0' permet de sélectionner le type de fonctionnement 'manuel' du moteur, tandis que le bouton poussoir 'S1' le type de fonctionnement 'automatique'.
En fonctionnement ‚manuel', le moteur tourne tant que l'on reste appuyé sur le bouton poussoir 'S2' et que le bouton poussoir 'S3' n'est pas actionné.
En fonctionnement ‚automatique', le moteur est mis en route avec le bouton 'S2' et est arrêté avec le bouton 'S3'.

Les cycles d'exécution du programme sont comptabilisés dans un mémento de type double (MD).
L'exemple utilise les adresses définies ci-dessous :

Entrées :
- Bouton de commande manuelle S0 = E 0.0
- Bouton de commande automatique S1 = E 0.1
- Bouton de mise en route S2 = E 0.2
- Bouton d'arrêt S3 = E 0.3

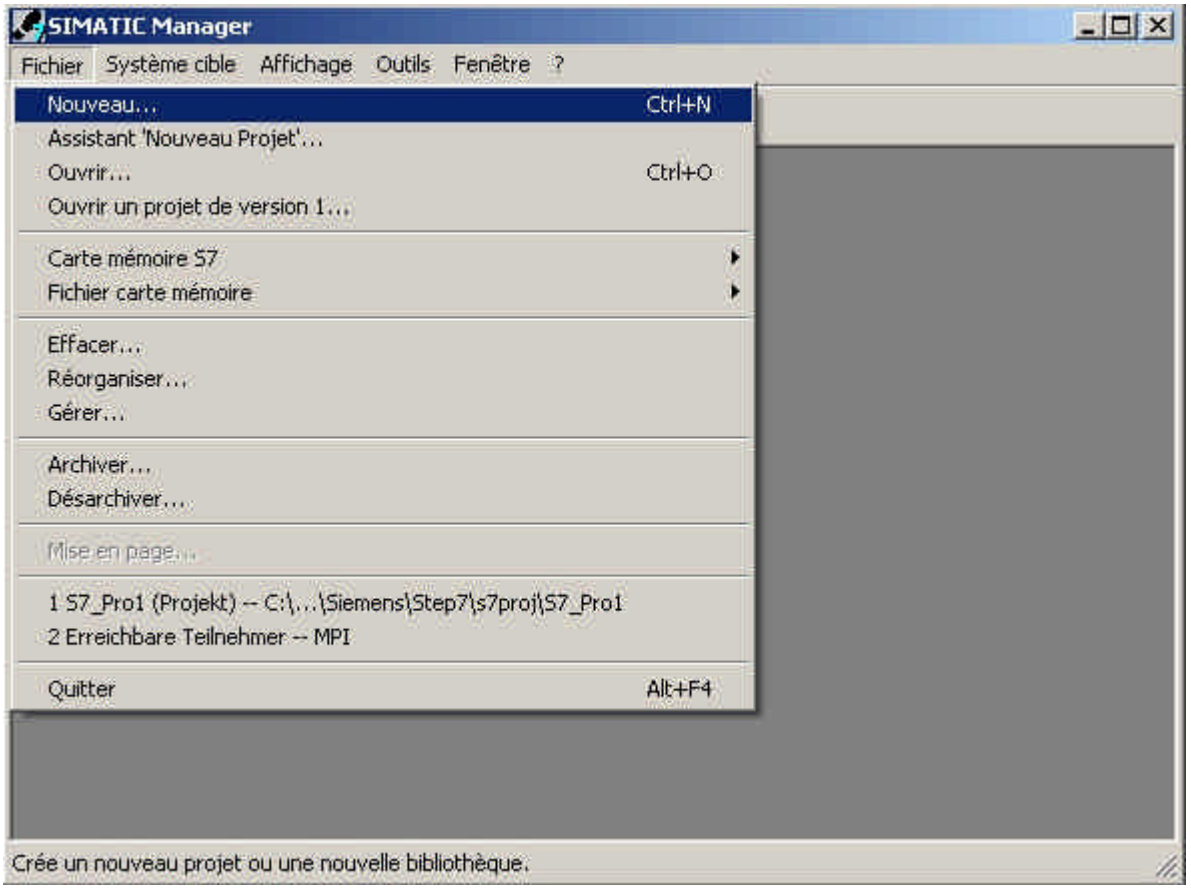
Sorties :
- Moteur = A 4.0

Mémentos :
- Compteur de cycles = MD20

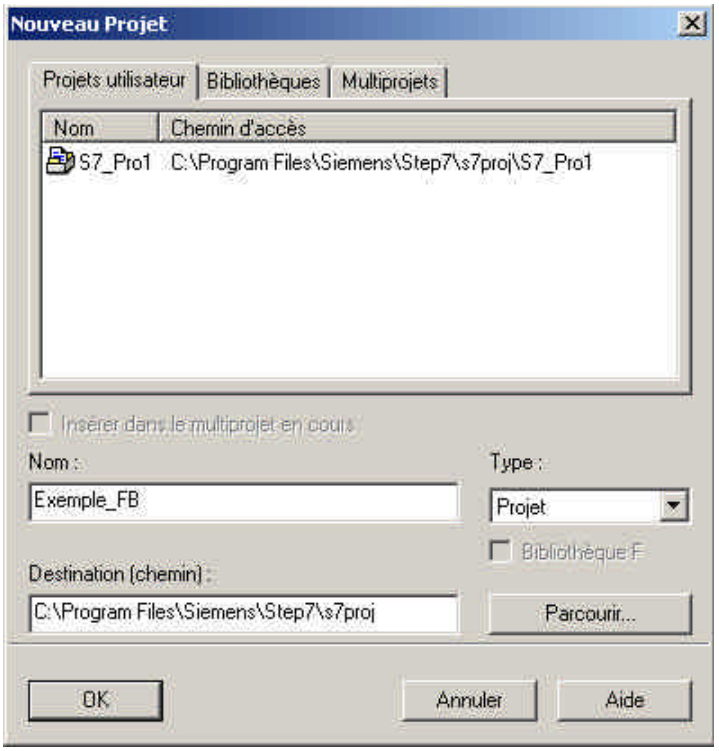
Pour la réalisation de cet exemple, veuillez suivre les étapes décrites ci-dessous :

1. Ouvrir **‚SIMATIC Manager'** (double clique sur l'icône)

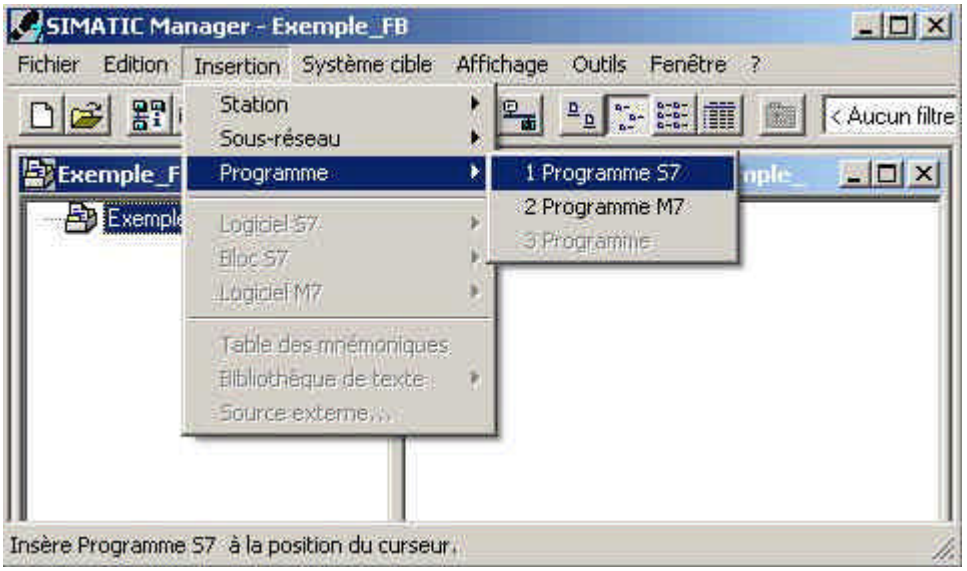
2. Ouvrir un nouveau Projet (→ Fichier → Nouveau)



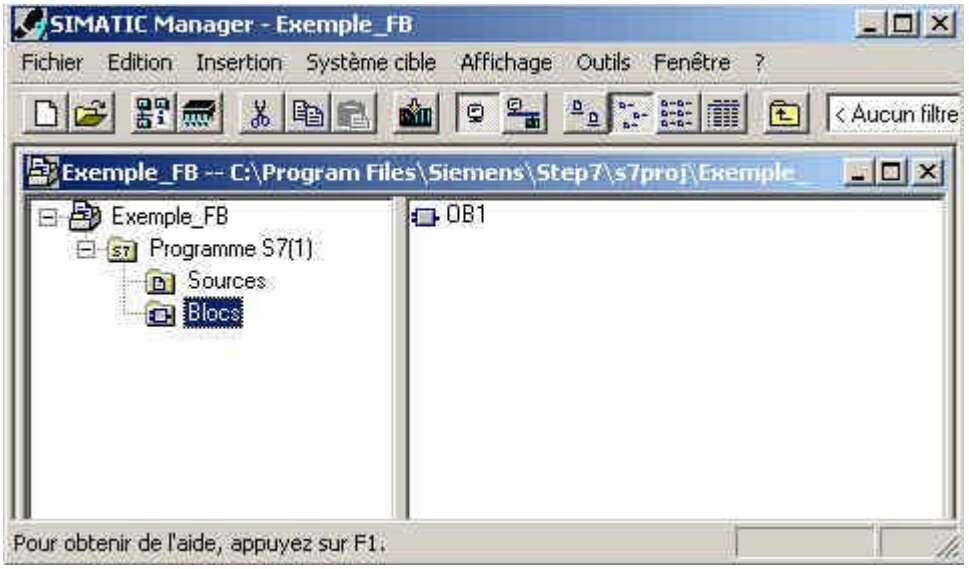
3. Créer un nouveau projet, choisir le chemin et donner le nom du projet **‚Exemple_FB'** (→ Exemple_FB)



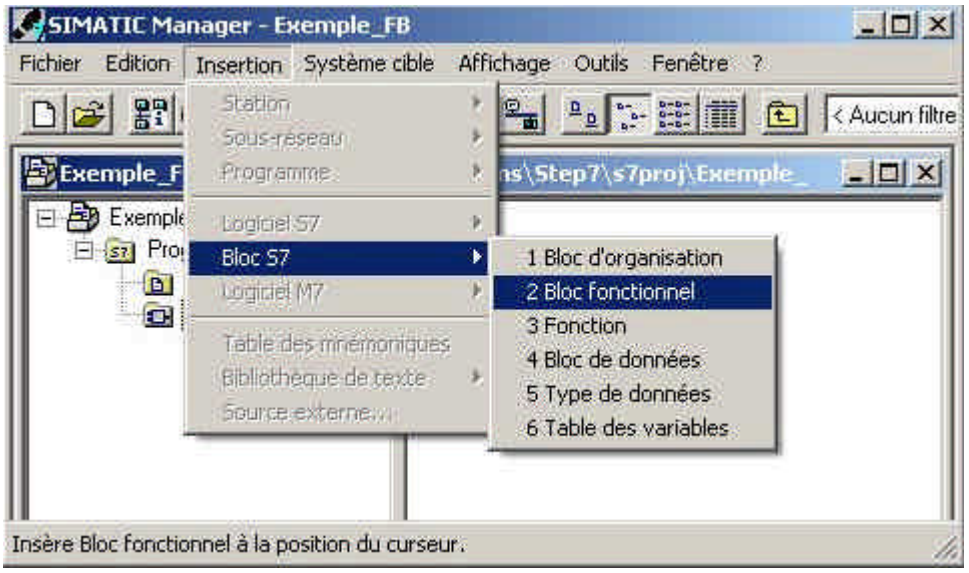
4. Insérer un nouveau **‚Programme S7'** (→ Insertion → Programme → Programme S7).



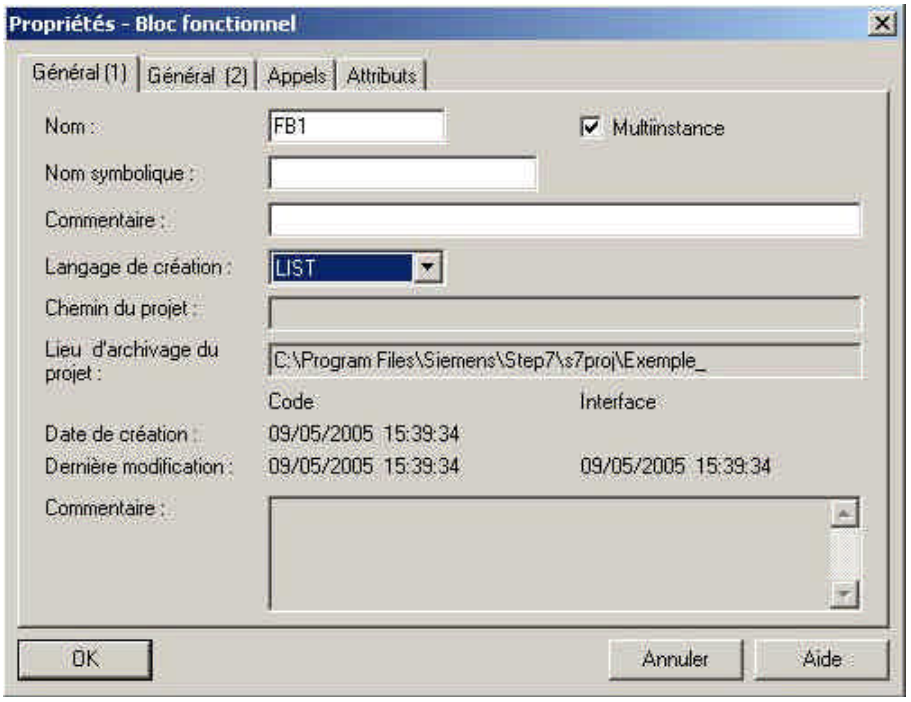
5. Sélectionner le dossier **‚Blocs'** (→ Blocs).



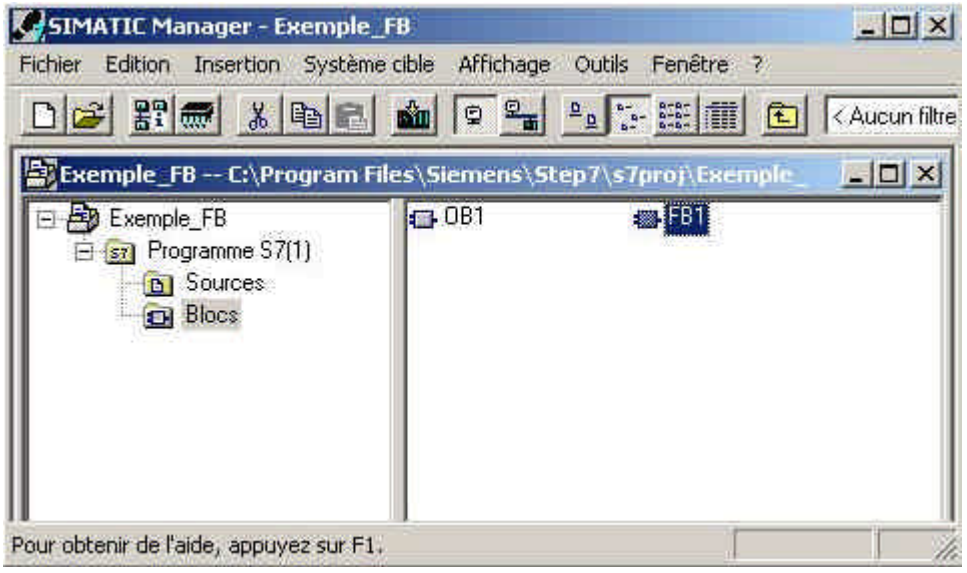
6. Insérer un **‚Bloc fonctionnel'** (→ Insertion → Bloc S7 → Bloc fonctionnel).



7. Accepter le nom et les propriétés en cliquant sur **‚OK'**. (→ FB1 → OK).



8. Double clique sur **‚FB1'** pour ouvrir le bloc fonctionnel. (→ FB1)



9. L'éditeur **'CONST, LIST, LOG'** vous permet de modifier votre bloc fonctionnel (FB).
Des variables de type 'in', 'out', 'in_out' et 'temp' peuvent être définies, ceci se fait dans la table de déclaration des variables située en haut du FB1.

Paramètres d'entrée (IN) pour les FBs, FCs, SFBs et SFCs
Grâce aux paramètres d'entrées, des données peuvent être transmises au bloc de programme.

Paramètres de sortie (OUT) pour les FBs, FCs, SFBs et SFCs
Les résultats du bloc appelé sont transmis grâce aux paramètres de sortie.

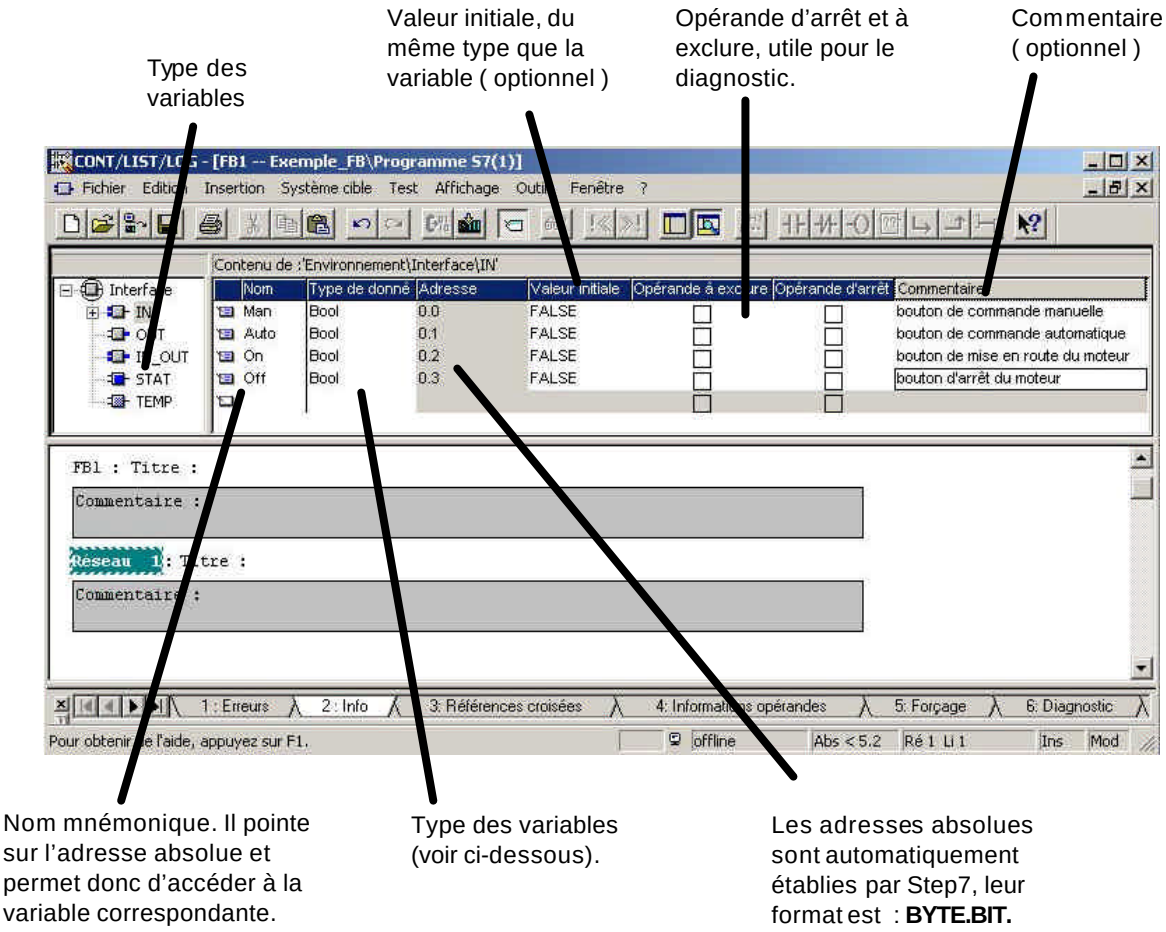
Paramètres d'entrée/sortie (IN_OUT) pour les FBs, FCs, SFBs et SFCs
Grâce aux paramètres d'entrée/sortie, des données peuvent être transmises au bloc, où elles seront utilisées. Cette même variable contiendra ensuite le résultat du bloc appelé.

Variable statique (STAT) pour les FBs et SFBs
Les Variables statiques sont les variables locales d'un bloc fonctionnel contenues dans le DB d'instance correspondant. Celles-ci restent inchangées jusqu'au prochain appel du bloc fonctionnel.

Variable temporaire (TEMP) pour tous les blocs
Les variables temporaires sont les variables locales d'un bloc contenues dans la pile locale (L-Stack). Celles-ci sont détruites après l'exécution du bloc.

Indication : C'est ici que réside la différence entre FB/SFB et FC/SFC : les FCs n'ont pas de variable statique à disposition alors que pour les FBs, ces variables sont stockées dans le DB d'instance correspondant jusqu'au prochain appel de l'FB.
C'est pourquoi les FBs vont être utilisés dans les programmes où des données doivent subsister durant plusieurs cycles d'exécution du programme.

La définition des variables se fait en donnant un nom et un type. Une valeur initiale ainsi qu'une description peuvent être ajoutées. Prenons pour exemple les paramètres de type **‚IN' :**

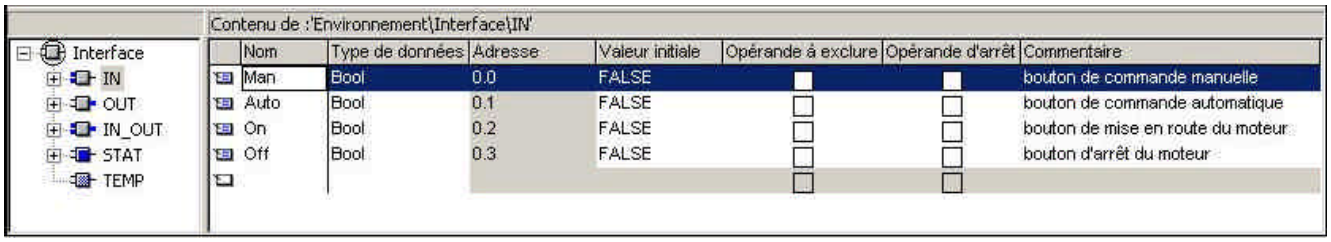


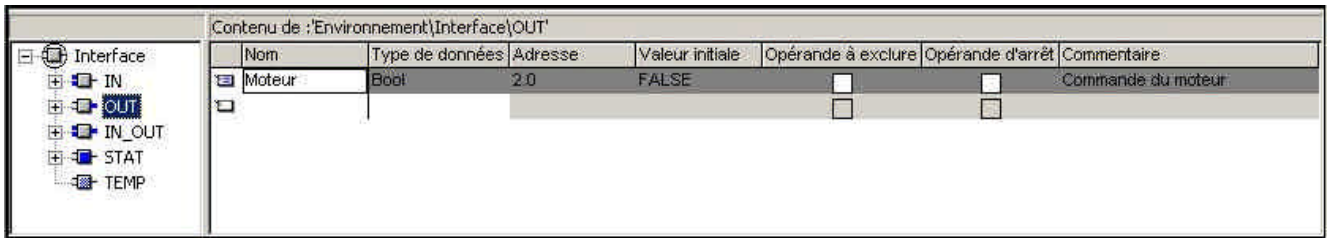
Indication : Dans la table de déclaration des variables sont listés tous les types de variables disponibles pour le bloc en question. Pour un FC les variables de type : **'in', 'out'**, **'in_out'** et **'temp'.** Pour un FB les variables de type : **'in', 'out', 'in_out', 'stat'** et **'temp'**. Lorsqu'une variable est complètement définie apparaît une nouvelle ligne vide pour pouvoir déclarer une nouvelle variable.

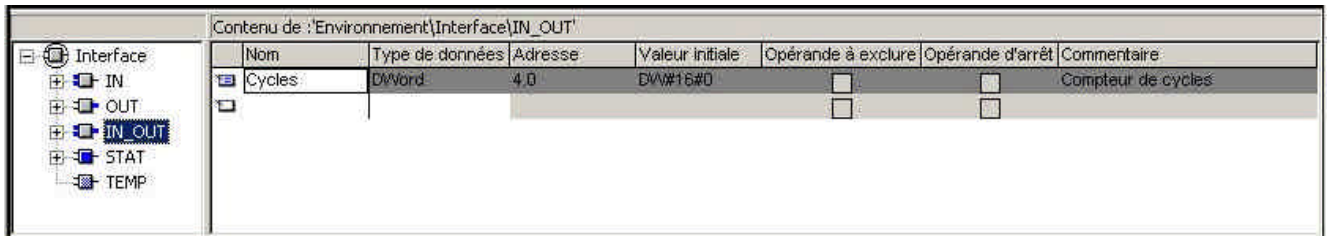
Les données d'une table de déclaration de variables doivent être typées, exactement comme pour un bloc de données (DB).
Step7 propose un certains nombre de types de données standards (ou type simple) prédéfinis :

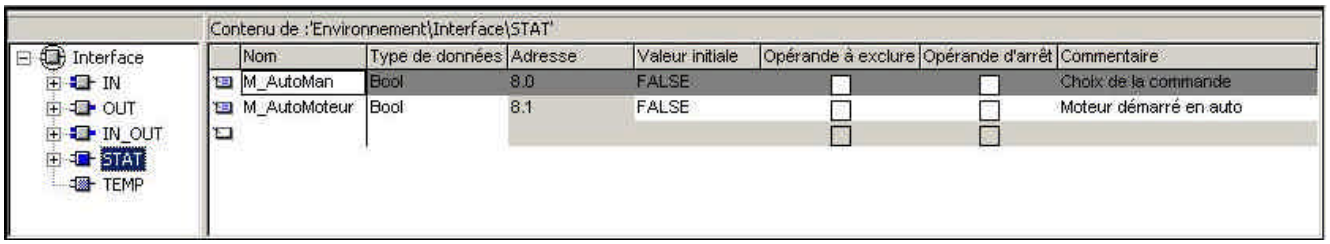
Type et description	Taille en bits	Options pour le format :	Plage et représentation des nombres (valeur minimale à valeur maximale)	Exemple
BOOL (bit)	1.	Texte booléen	TRUE/FALSE	TRUE
BYTE (octet)	8	Nombre hexadécimal	B#16#0 à B#16#FF	L B#16#10 L byte#16#10
WORD (mot)	16	Nombre en binaire pur Nombre hexadécimal BCD Nombre décimal non signé	2. 0 à 2#1111_1111_1111_1111 W#16#0 à W#16#FFFF C#0 à C#999 B#(0,0) à B#(255,255)	L 2#0001_0000_0000_0000 L W#16#1000 L word#16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (double mot)	32	Nombre en binaire pur Nombre hexadécimal Nombre décimal non signé	2#0 à 2#1111_1111_1111_1111_ 1111_1111_1111_1111 DW#16#0000_0000 à DW#16#FFFF_FFFF B#(0,0,0,0) à B#(255,255,255,255)	2#1000_0001_0001_1000_ 1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (entier)	16	Nombre décimal signé	32768 à 32767	L 1
DINT (nombre entier de 32 bits)	32	Nombre décimal signé	L#-2147483648 à L#2147483647	L L#1
REAL (nombre à virgule flottante)	32	IEEE nombre à virgule flottante	Limite supérieure : ±3.402823e+38 Limite inférieure : ±1.175 495e-38	L 1.234567e+13
S5TIME (durée SIMATIC)	16	Durée S7 en pas de 10 ms (valeur par défaut)	S5T#0H_0M_0S_10MS à S5T#2H_46M_30S_0MS et S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (durée CEI)	32	Durée CEI en incréments de 1 ms, entier signé	-T#24D_20H_31M_23S_648MS à T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (date CEI)	16	Date CEI en incréments de 1 jour	D#1990-1-1 à D#2168-12-31	L D#1994-3-15 L DATE#1994-3-15
TIME_OF_DAY (heure)	32	Heure en pas de 1 ms	TOD#0:0:0.0 à TOD#23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
CHAR (caractère)	8	Caractères ASCII	'A','B' etc.	L 'E'

La déclaration des Variables de types **‚IN'** ,**OUT', ‚IN_OUT', ‚STAT'** et **‚TEMP'** se fera comme suit :

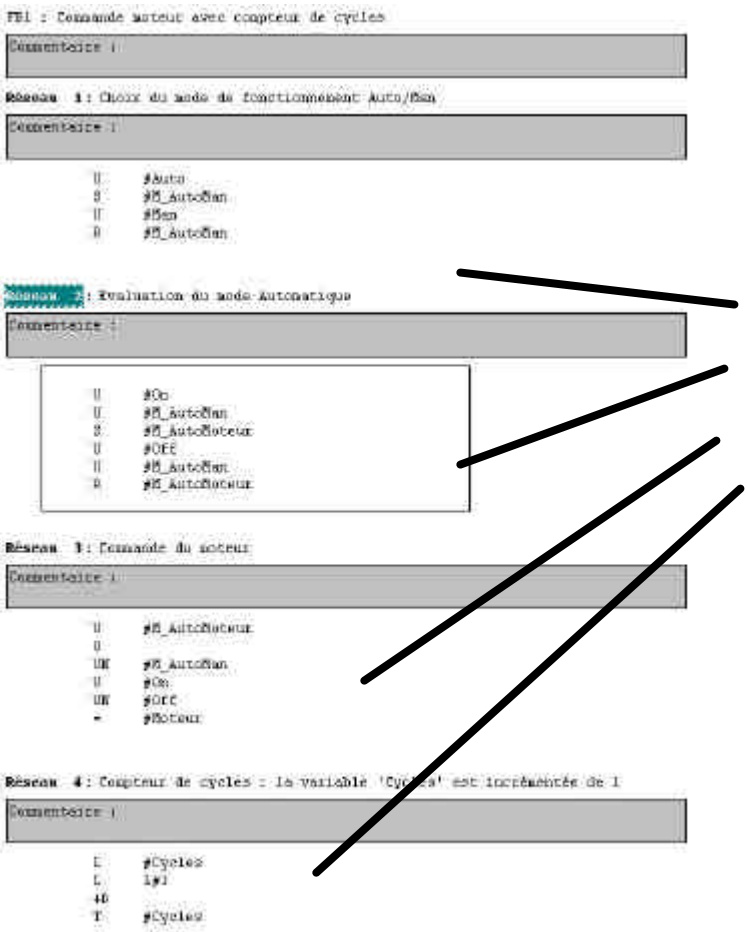






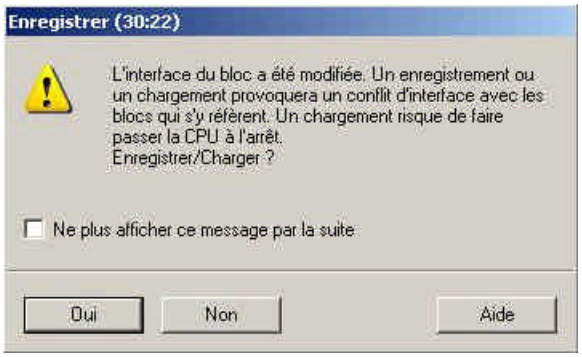


10. Les variables peuvent être utilisées dans le programme en les faisant précéder du caractère ‚#'. Ci-dessous, un exemple de programme en 'LIST' pour notre application. Le bloc fonctionnel FB1 peut ensuite être enregistré ‚' et chargé dans la CPU ‚'. Le commutateur d'état de la CPU est sur la position ‚Stop' (→ →)

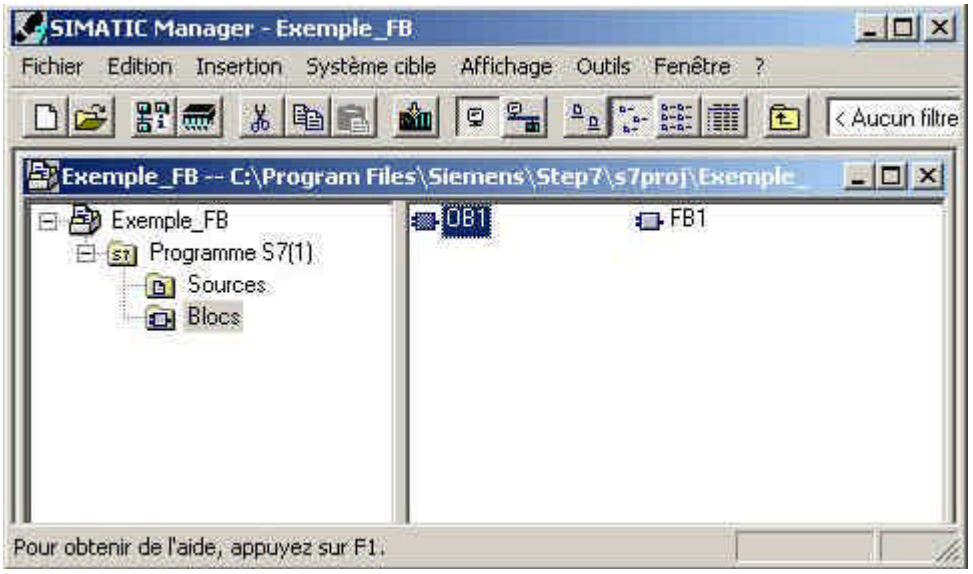


On accède directement aux variables en faisant précéder le nom correspondant par le caractère **‚#'**.

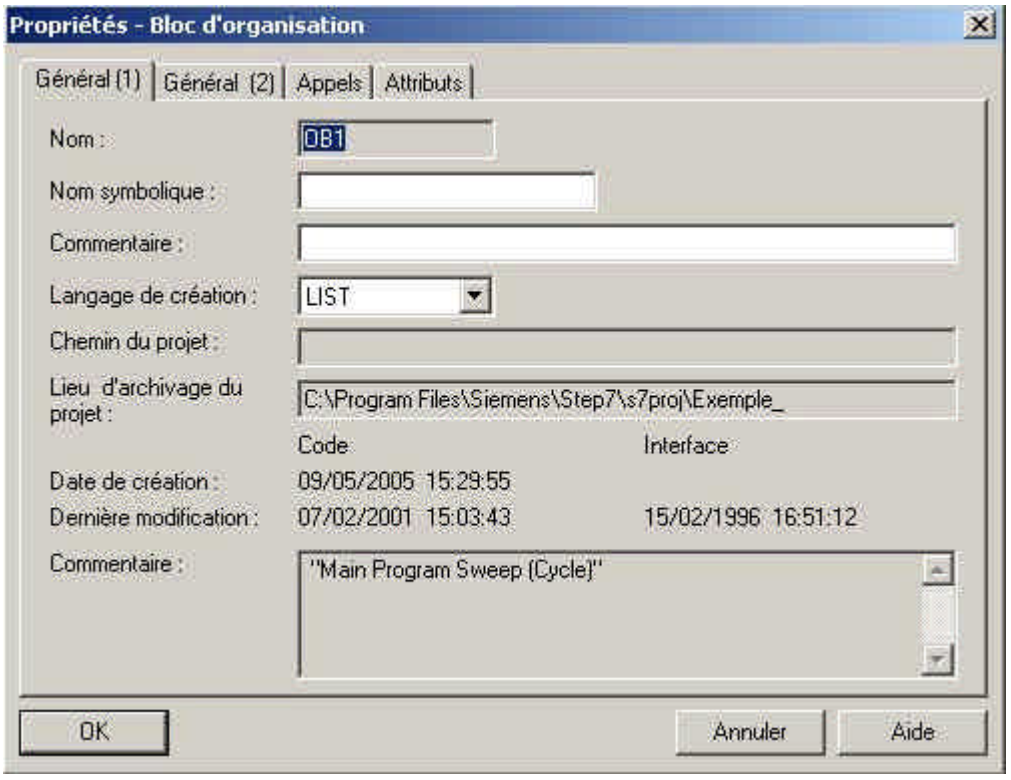
11. Le message d'alerte informe de la modification de l'interface d'appel du bloc fonctionnel. Répondre par **‚Oui'**. (→ Oui).



12. Dans **‚SIMATIC Manager',** ouvrir l'**‚OB1'** pour programmer l'appel du FB1 (→ OB1).



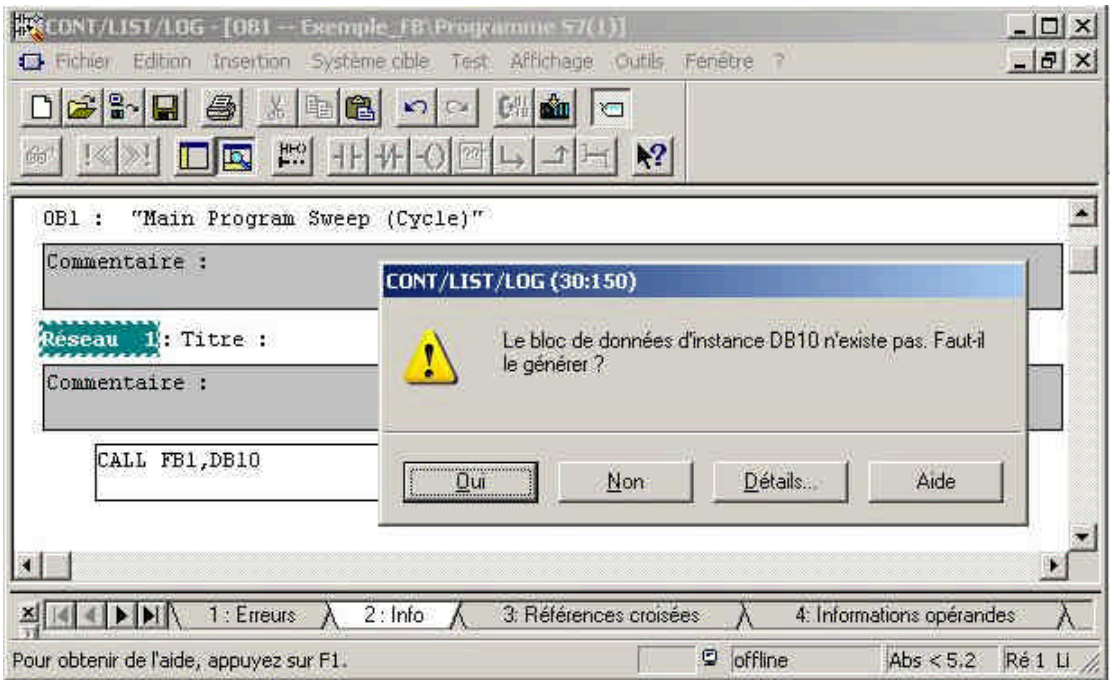
13. Cliquer sur **‚OK'** pour accepter les propriétés (→ OK).



14. La programmation de l'OB1 se fait dans l'éditeur 'CONST/LIST/LOG'. Le FB1 doit être appelé avec le DB d'instance (ou DB local) correspondant. Ceci est réalisé grâce à la commande suivante :

CALL FB1, DB10 <Enter>

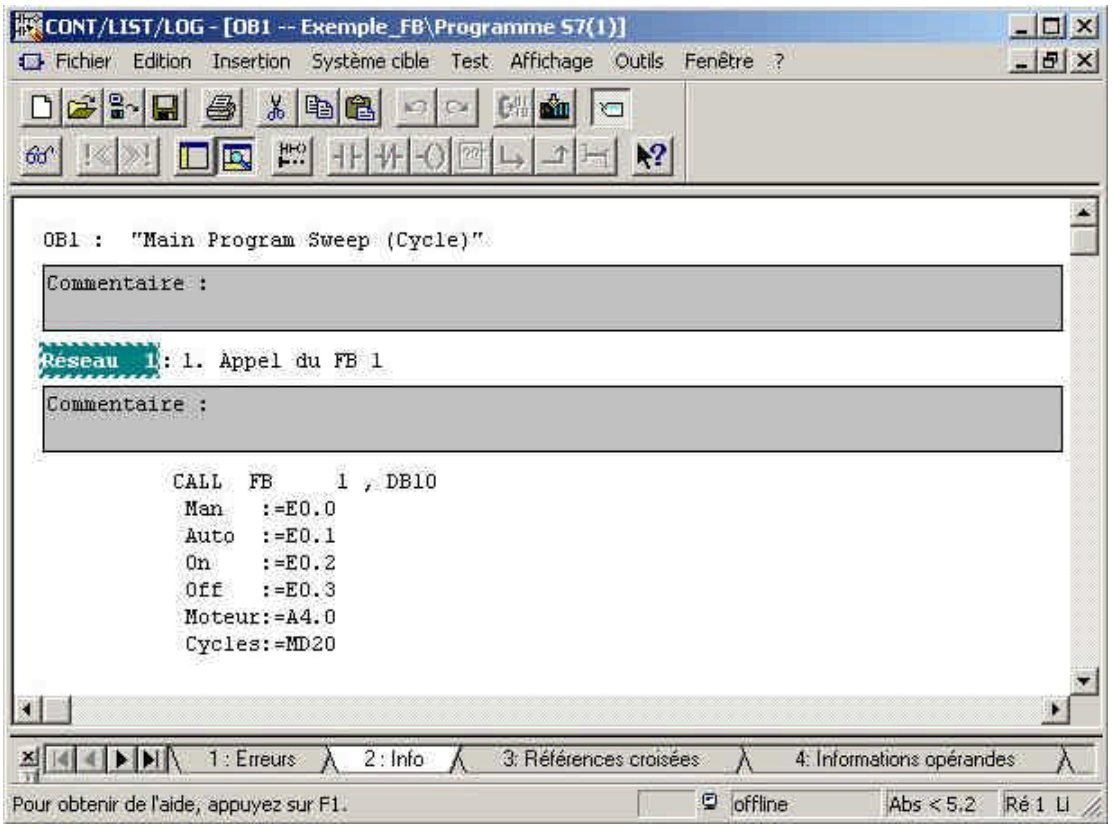
En répondant ,**Oui**' à la question, le DB d'instance (dans notre cas : DB10) sera automatiquement généré par Step7 (→ Call FB1,DB10 → Oui)



15. L'interface d'appel du FB est affichée, les paramètres de type **'in', 'out'** et **'in_out'** peuvent alors être affectés avec les valeurs souhaitées (par ex. : E 0.0, MW2 …etc).

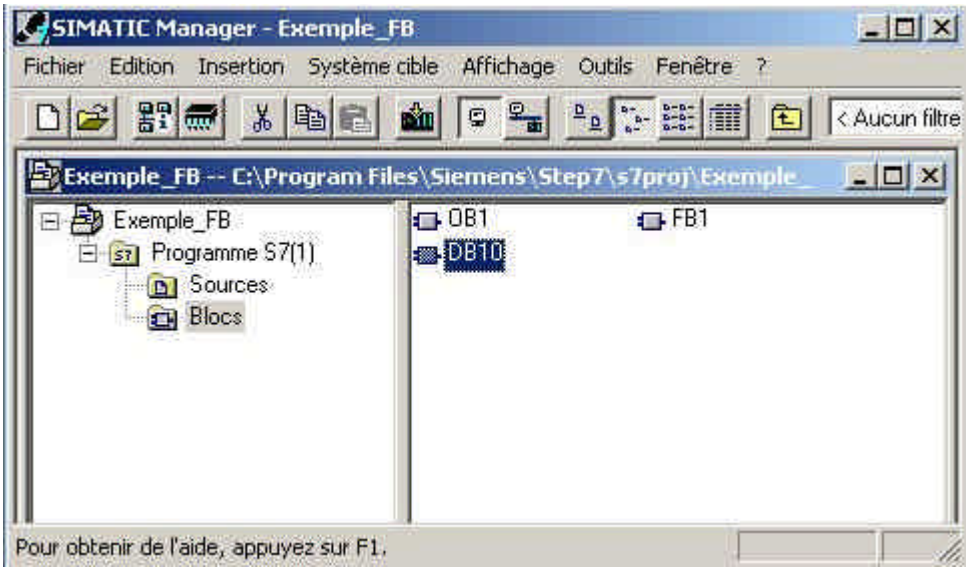
```
CALL  FB     1 , DB10
 Man   :=
 Auto  :=
 On    :=
 Off   :=
 Moteur:=
 Cycles:=
```

16. Les paramètres d'appel du FB1 doivent être affectés comme suit. Le bloc d'organisation OB1 peut ensuite être enregistré ,  ' et chargé dans la CPU ,  '. Le commutateur d'état de la CPU est sur la position ‚Stop' (→  → )



Indication : Tel qu'il a été programmé, le FB1 peut être appelé avec différents DBs et différentes entrées/sorties. Il peut donc être considéré comme un « bloc standard » spécifique à notre application.

17. Dans ‚SIMATIC Manager' sélectionner le DB d'instance (ou DB local) ‚DB10' et le charger dans la CPU à l'aide du bouton ‚  '. Le commutateur d'état de la CPU se trouve en position ‚STOP' ! (→ DB10 → )



18. Lancer l'exécution du programme en positionnant le commutateur d'état de la CPU sur ‚RUN'. Le moteur démarre lorsque l'on appuie sur le bouton E0.0 et s'arrête quand on appuie sur le bouton E0.1. Le mémento MD20 compte le nombre de fois où le FB1 est appelé depuis l'OB1. Il est ainsi possible de connaître la durée d'un cycle de l'OB1. Comme notre programme est très court, la fréquence d'appel de FB1 sera élevée.