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

MODULE B4

Blocs de données

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 :

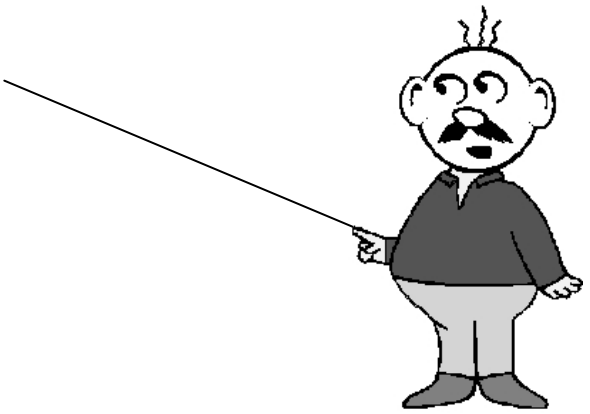
Les symboles suivants seront utilisés dans ce module :

Information

Programmation

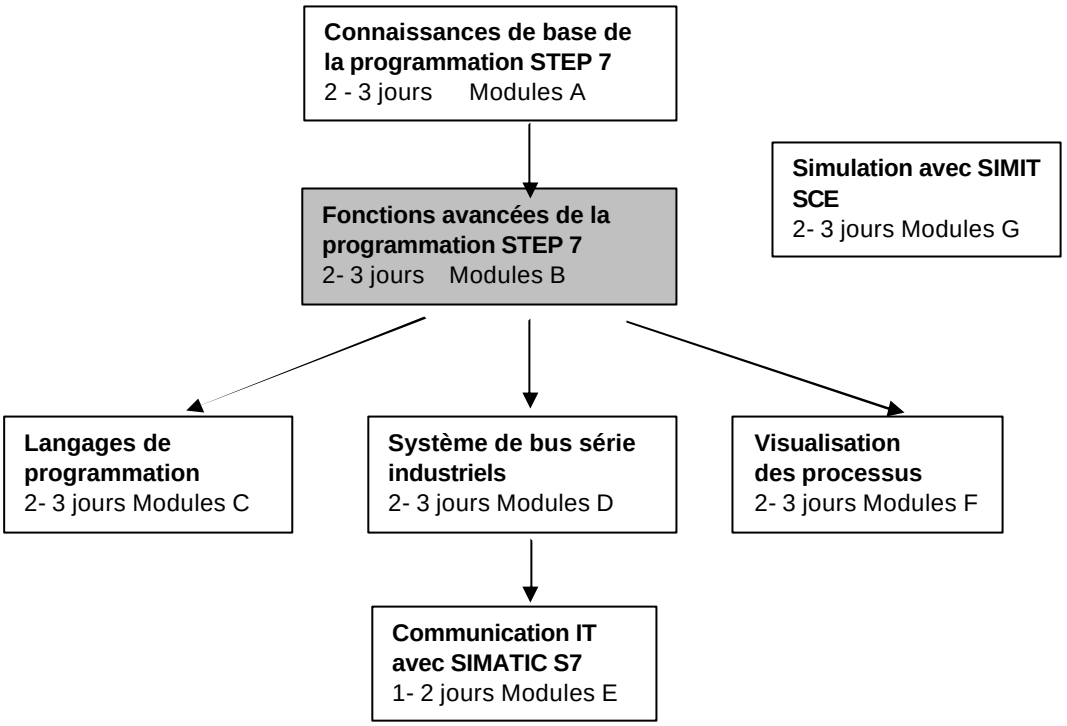
Exemple d'application

Indications



1. AVANT-PROPOS

Le contenu du module B1 est assigné à l'unité **‚Fonctions avancées de la programmation STEP7'.**



Objectif :

Dans ce module, le lecteur va apprendre comment un bloc de données peut être utilisé pour sauvegarder des données.

- Création de blocs de données
- Détermination de la structure d'un bloc de données
- Accès aux éléments de données dans le programme STEP 7

Pré-requis :

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

- Systèmes d'exploitation : Windows 95/98/2000/ME/NT4.0/XP
- Base en programmation SPS avec STEP7 (Ex : Module A3 ‚Startup', programmation SPS avec STEP 7)
- Base en programmation structurée (Ex : Annexe I – Bases de SPS – Programmation avec SIMATIC S7-300)

Configurations matérielles et logicielles requises

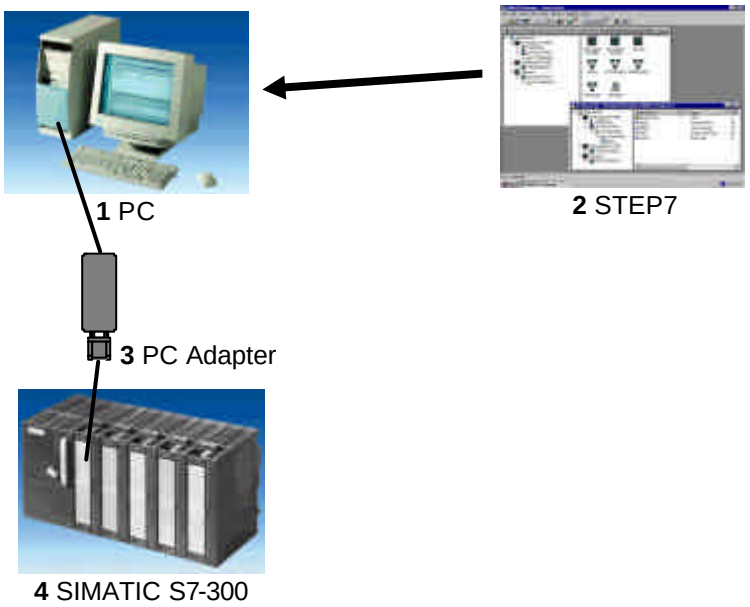
1 PC, système d'exploitation : Windows 95/98/2000/ME/NT4.0/XP avec
- Minimum : 133MHz et 64Mo RAM, 65 Mo d'espace disponible
- Optimal : 500MHz et 128Mo RAM, 65 Mo d'espace disponible

2 Logiciel STEP 7 V 5.x

3 Interface ordinateur MPI (Ex : PC- Adapter)

4 SPS SIMATIC S7-300 avec au moins un module d'entrées numériques et un module de sorties numériques. Les entrées doivent sortir sur une unité fonctionnelle.

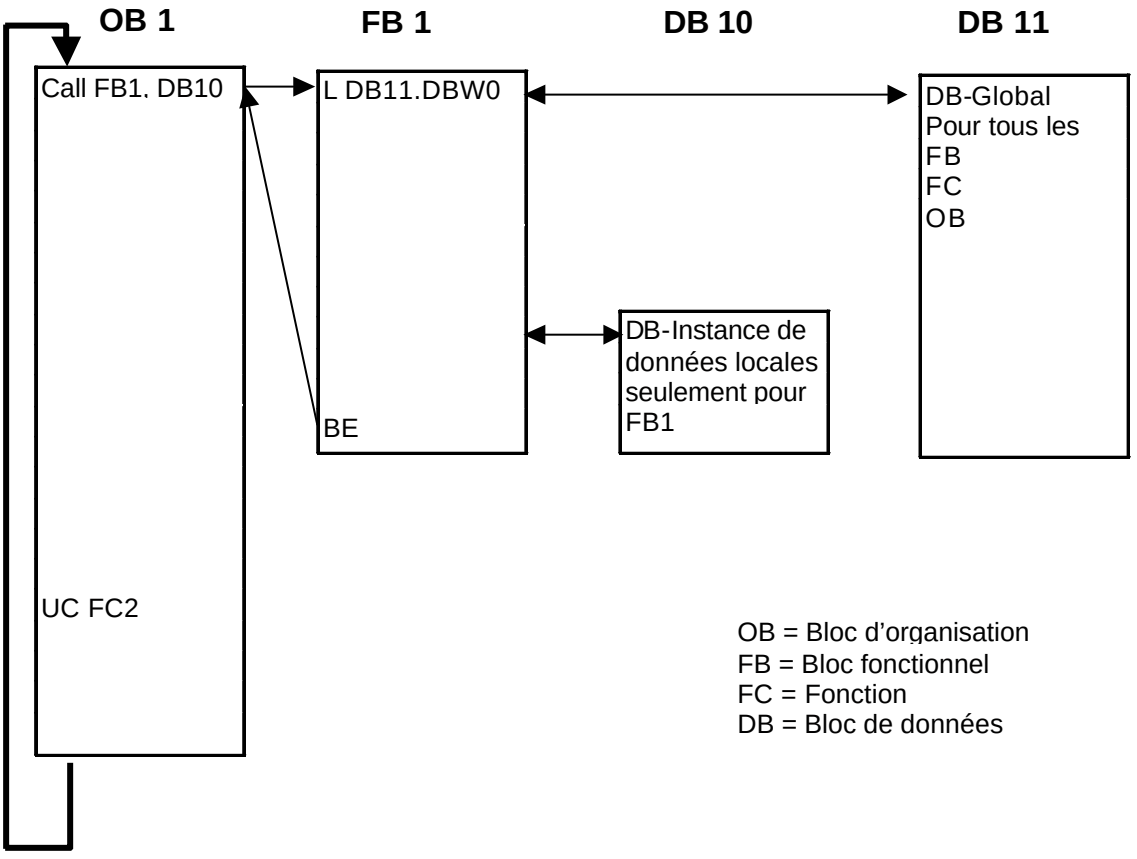
Exemple de configuration :
- Bloc d'alimentation : PS 307 2A
- CPU : CPU 314
- Entrées numériques : DI 16x DC24V
- Sorties numériques : DO 16x DC24V / 0,5 A



2. INDICATIONS SUR LES BLOCS DE DONNEES

Les blocs de données (DB) peuvent être utilisés à travers votre programme pour sauvegarder les données dans la CPU. Votre emplacement mémoire s'élève, selon la CPU, jusqu'à 8 Koctets (8192 octets) ou plus.
Il existe deux catégories de blocs de données. Le DB-**Global**, où tous les OB, FB et FC peuvent lire les données sauvegardées ou bien écrire lui-même les données dans le DB. Le DB-**Instance locale**, qui sont attribués à certains FB.
Dans le DB, différents types de données (par exemple : BOOL ou WORD) peuvent être sauvegardés dans un ordre quelconque.
Cette structuration d'un DB résulte par l'entrée dans un tableau avec l'outil **,CONT/LIST/LOG Programmation de blocs S7'**.

Les blocs de données dans la structure du programme STEP 7 sont comme suit :



3. CREATION DE BLOCS DE DONNEES

Les blocs de données sont créés et ouverts comme un blocs de programme avec l'outil **'CONT/LIST/LOG Programmation de blocs S7'**. Ils servent, par exemple, à la sauvegarde des données et des états du dispositif.

La création d'un exemple de programme simple avec l'emploi de blocs de données globales, va vous être décrite ci-après :

On choisit pour cela avec les commutateurs de 'S0' à 'S7' des valeurs dans un bloc de données et on les affiche dans un module de sortie « Affichage ». Lors de l'actionnement simultané de plusieurs commutateurs, la représentation de la valeur du commutateur S7 a la plus haute priorité et celle de S1 la priorité la plus basse.

L'exemple concerne les adresses suivantes :

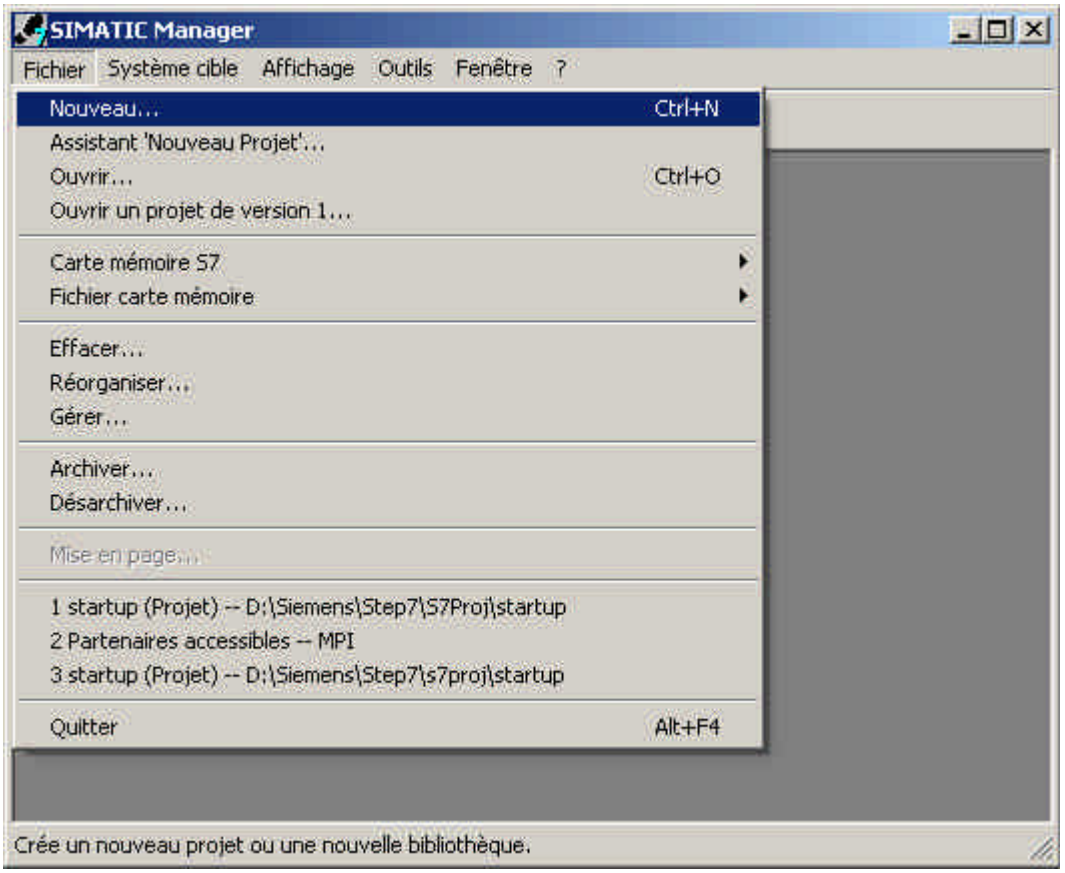
Entrées :
- Interrupteur S0 = E 0.0
- Interrupteur S1 = E 0.1
- Interrupteur S2 = E 0.2
- Interrupteur S3 = E 0.3
- Interrupteur S4 = E 0.4
- Interrupteur S5 = E 0.5
- Interrupteur S6 = E 0.6
- Interrupteur S7 = E 0.7

Sorties :
- Indicateur = AW4

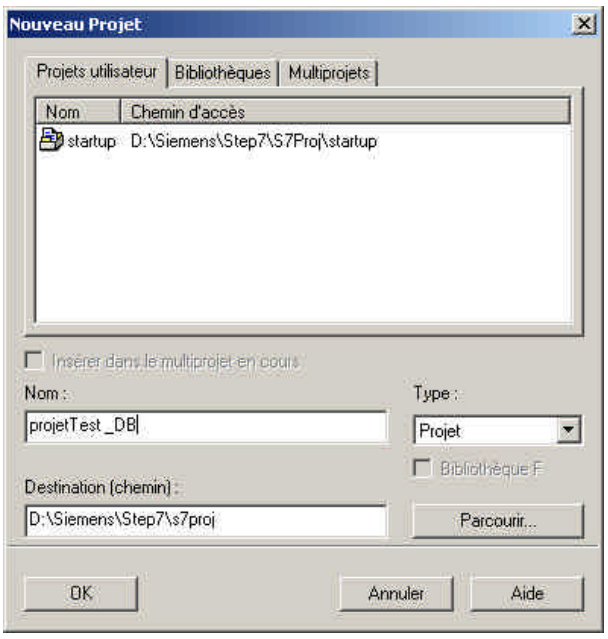
Afin de créer cet exemple de programme, suivez les étapes suivantes (L'élaboration d'une configuration matérielle ne sera pas effectuée.) :

1. Ouvrez **‚SIMATIC Manager'** en double-cliquant (→ SIMATIC Manager)

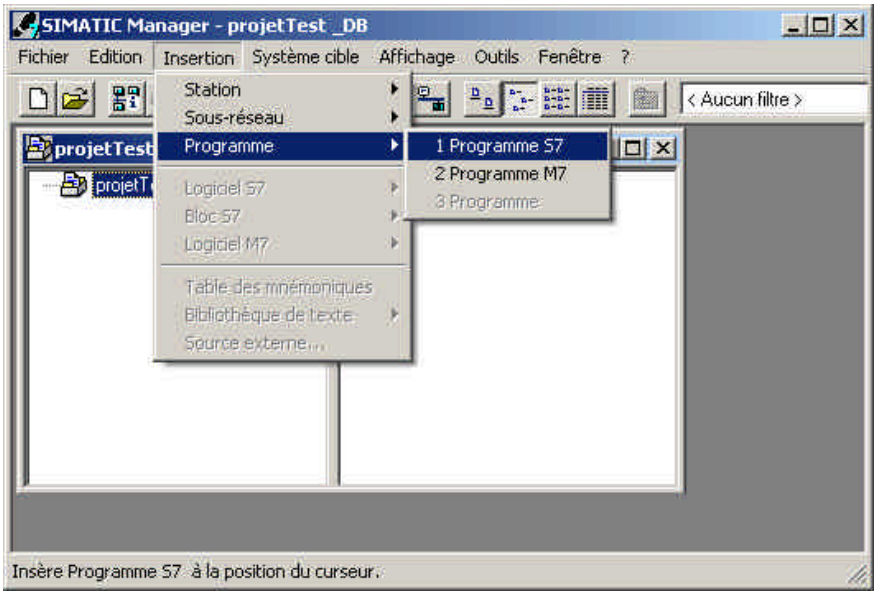
2. Créez un nouveau projet (→ Fichier → Nouveau)



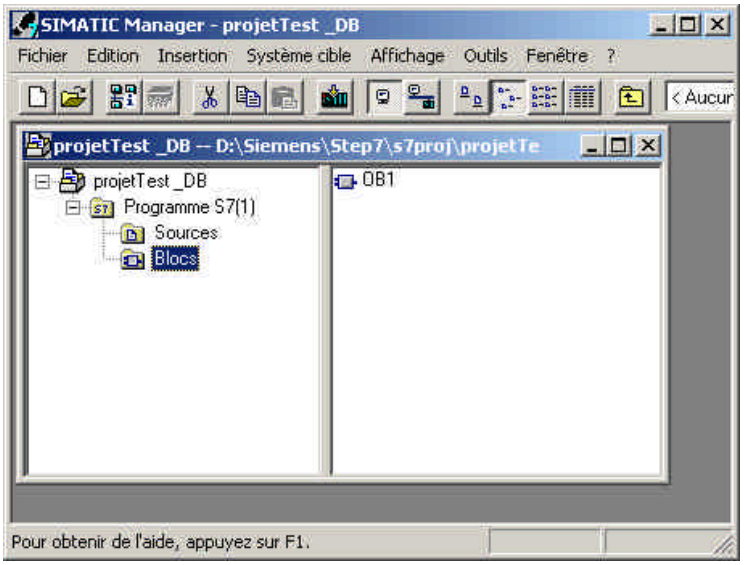
3. Donnez au projet le nom de **‚projetTest_DB'**.
(→ 'projetTest _DB' → OK)



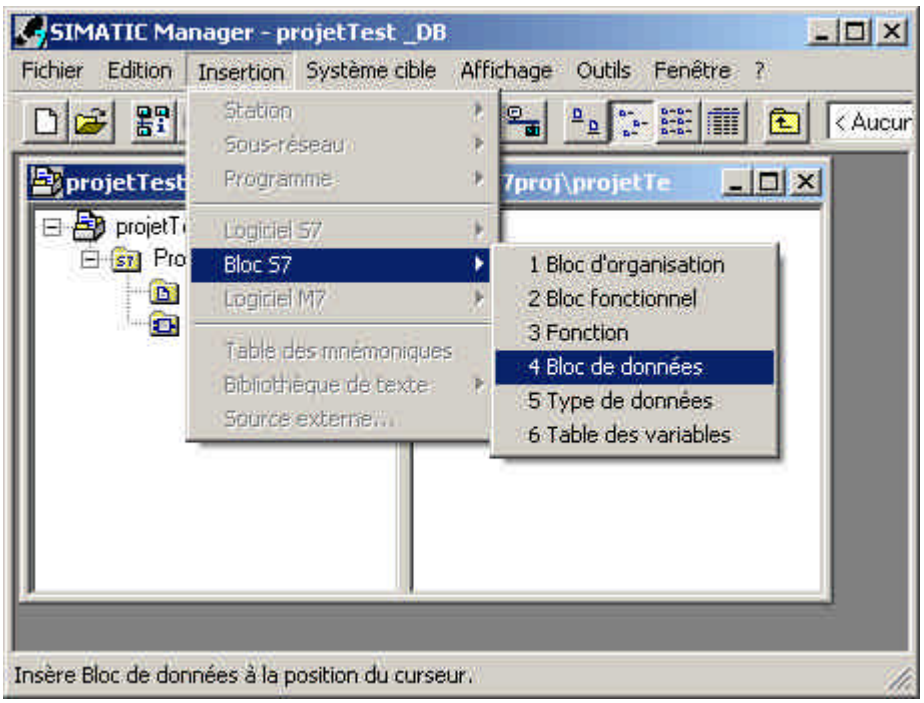
4. Insérez un nouveau **‚programme S7'** (→ Insertion → Programme → programme S7).



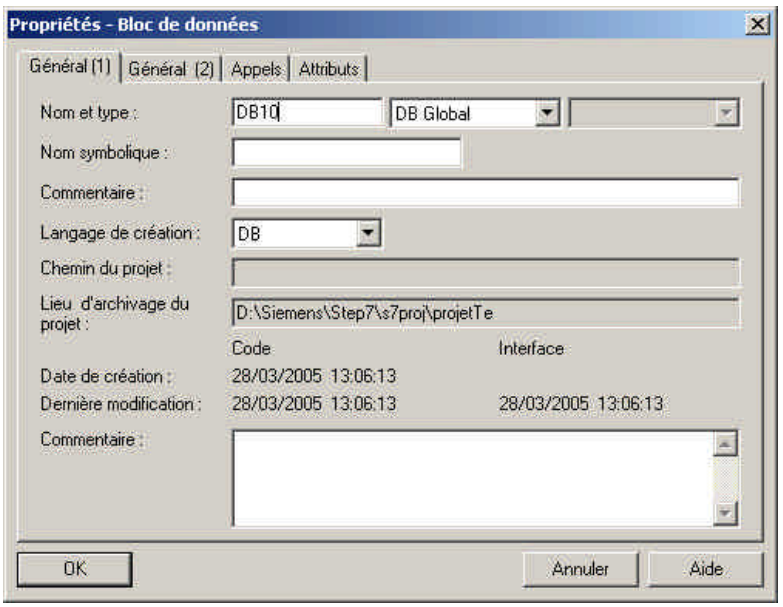
5. Sélectionnez le répertoire **‚Blocs'**. (→ Blocs)



6. Insérez un **‚Bloc de données'** (→ Insertion → Bloc S7 → Bloc de données).

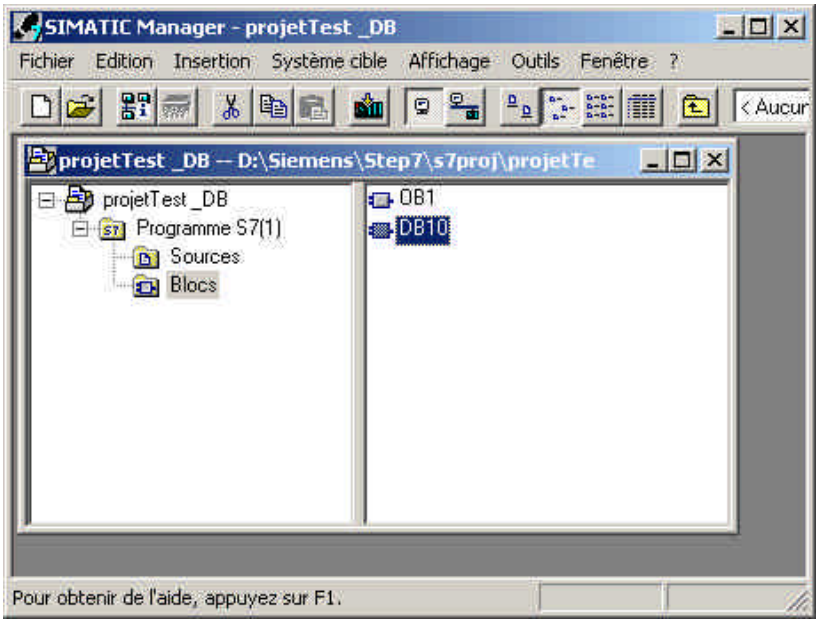


7. Entrez le numéro **‚DB10'** pour le bloc de données. Puis choisissez **‚DB-Global'** en tant que type. Validez avec **‚OK'**. (→ DB10 → DB-Global → OK)



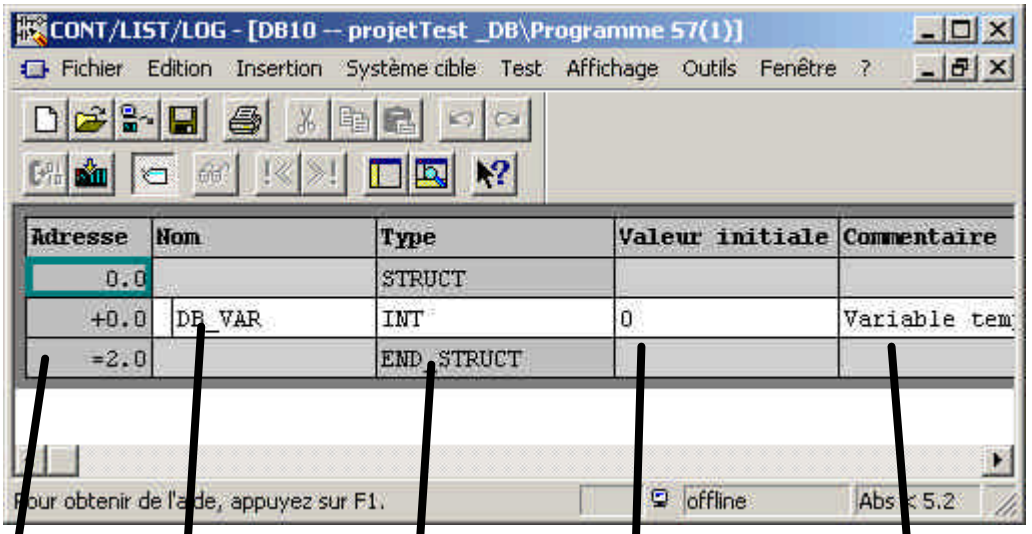
Indications: Comme types peuvent être choisis "DB global", "instance DB" respt. "DB de type".
Les blocs de données "instances DB" à FB attribués sont créés automatiquement à l'appel des FB correspondants. La plupart du temps, il est donc inutile de les créer soi-même.
S'il existe dans le dossier des blocs FBs ou SFBs, on peut choisir dans cette liste relevable du type "instance DB" et les attribuer dans la prochaine liste relevable à un FB ou un SFB.
Les blocs de données à type de données personnalisé (UDT) sont des blocs de données, dont la structure a déjà été imposée préalablement dans cet UDT.
S'il existe dans le dossier des blocs UDTs, on peut choisir dans la liste relevable du type "DB du type " et les attribuer dans la prochaine liste relevable.

8. Ouvrez le bloc de données **‚DB10'** en double-cliquant. (→ DB10).



9. Le bloc de données est créé, et pour chaque élément de données, on doit saisir un **‚Mnémonique'**, le **‚Type'**, une **‚valeur initiale'** ainsi qu'un **‚commentaire'** (optionnel). L'adresse est générée automatiquement et ne peut pas être changée.

   Ensuite le bloc de données peut être enregistré ‚  ' et chargé dans le SPS ‚  '. Pour cela, l'interrupteur à clé de la CPU doit se trouver sur STOP ! (→ Nom → Type → Valeur initiale → Commentaire →  → )



Nom symbolique, avec lequel est renvoyée l'adresse absolue.

Type de données souhaité (voir ci-dessous) pour votre élément de

Valeur de début, qui doit être compatible avec le type de données. (optionnel)

Commentaire pour la documentation. (optionnel)

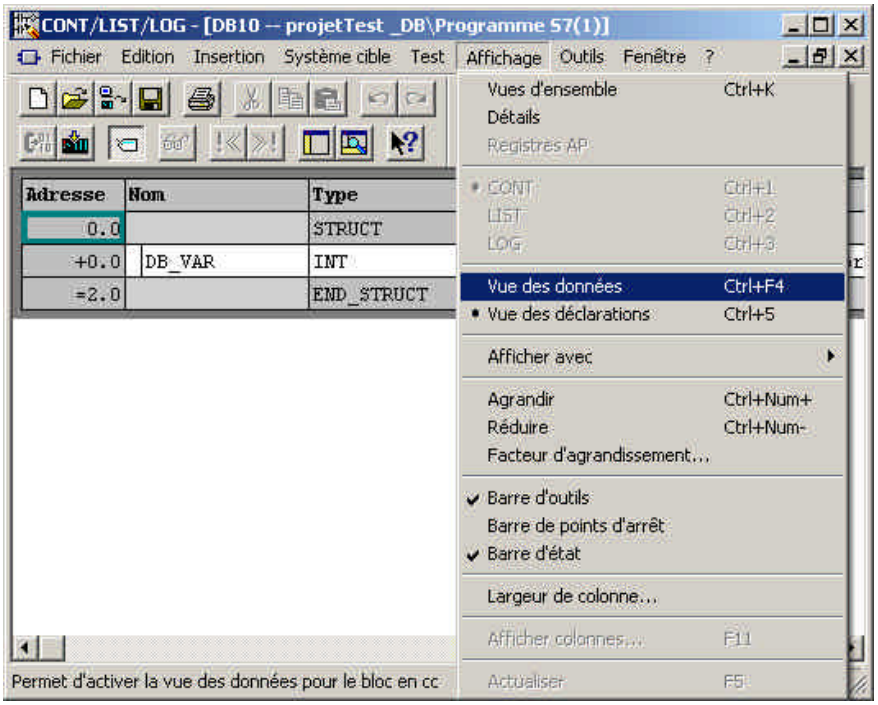
L'adresse absolue est automatiquement générée par STEP 7, si la DB est interprétée ou enregistrée. Le format de l'adresse est **OCTET.BIT.** Par cette adresse, on peut accéder aux éléments de données (par ex. par opérations de chargement et de transfert ou dans une boîte logique.).

Indications: Si le bloc de données est attribué comme instance DB local à un FB, alors le tableau de déclaration du FB impose la structure du DB.

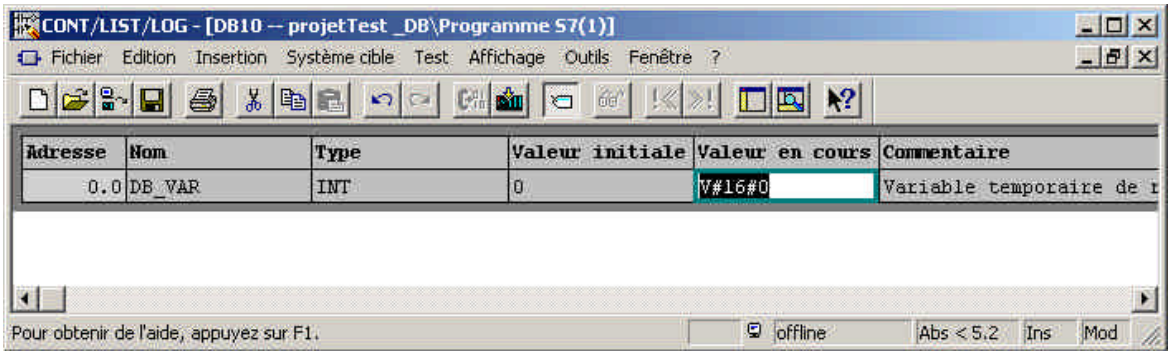
Les données dans un bloc de données doivent être déterminées par des types de données.
Les types de données standards suivants sont définis entre autres dans S7 :

Type et description	Taille en Bits	Format-option	Domaine et système de représentation, intervalle de valeurs possibles	Exemple
BOOL (bit)	1	Booléen	TRUE/FALSE	TRUE
BYTE (octet)	8	Hexadécimal	De B#16#0 jusqu'à B#16#FF	B#16#10
WORD (mot)	16	Nombre binaire	De 2#0 jusqu'à 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Nombre hexadécimal	De W#16#0 jusqu'à W#16#FFFF	W#16#1000
		BCD	De C#0 jusqu'à C#999	C#998
		Nombre décimal (o.V.)	De B#(0,0) jusqu'à B#(255,255)	B#(10,20)
DWORD (mot double)	32	Nombre binaire	De 2#0 jusqu'à 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Nombre hexadécimal	De DW#16#0000_0000 jusqu'à DW#16#FFFF_FFFF	DW#16#00A2_1234
		Nombre décimal (o.V.)	De B#(0,0,0,0) jusqu'à B#(255,255,255,255)	B#(1,14,100,120)
INT (entier)	16	Nombre décimal	De -32768 jusqu'à 32767	1
DINT (entier,32 bits)	32	Nombre décimal	De L#-2147483648 jusqu'à L#2147483647	L#1
REAL (Flottant)	32	Nombre flottant IEEE	Limite supérieure : +/-3.402823e+38 Limite inférieure : +/-1.175495e-38	1.234567e+13
S5TIME (Durée Simatic)	16	Durée S7 en pas de 10 ms	De S5T#0H_0M_0S_10MS jusqu'à S5T#2H_46M_30S_0MS et S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (Durée IEC)	32	Durée IEC en pas de 1ms, entier signé	De -T#24D_20H_31M_23S_648MS jusqu'à T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (date IEC)	16	Date IEC en pas de 1 jour	De D#1990-1-1 jusqu'à D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Heure)	32	Heure en pas de 1ms	De TOD#0:0:0.0 jusqu'à TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Caractère)	8	Caractère ASCII	´A´, ´B´ etc.	´B´

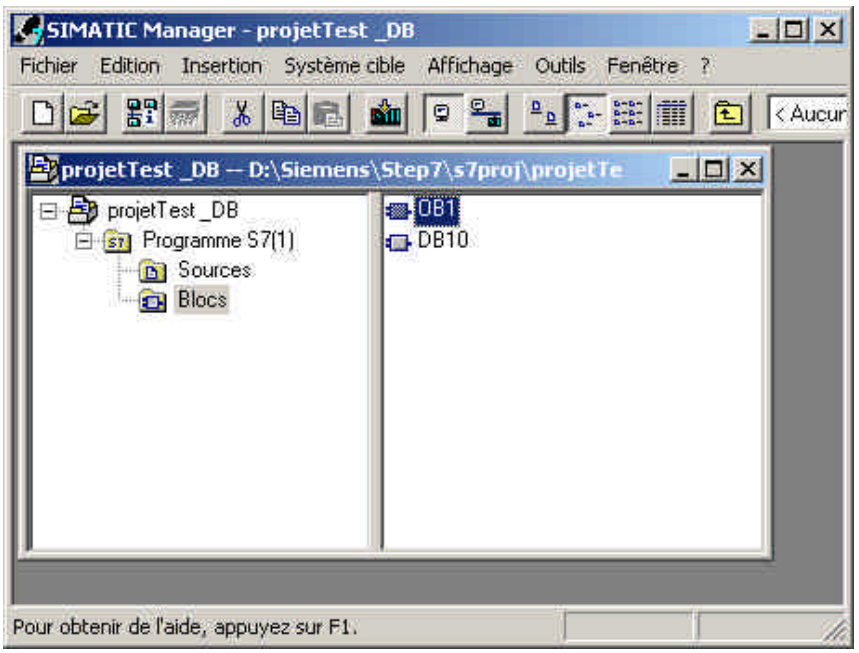
10. Si vous devez changer ultérieurement les valeurs dans le bloc de données, il ne suffit pas de changer celui ci dans le champ **‚Valeur de démarrage'**. Cela est seulement encore possible en basculant sur l'affichage **« Données »**. (→ Affichage → Vue des données)



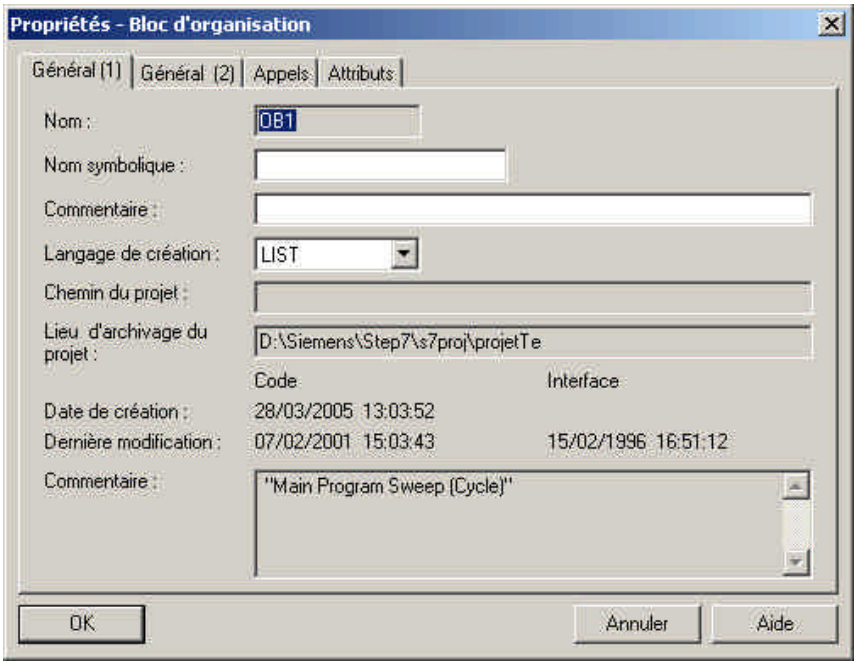
11. Maintenant entrez une nouvelle valeur dans le champ **‚valeur actuelle'** puis sauvegardez sur le disque dur avec le bouton d'enregistrement ‚  '. Effectuez également le transfert dans la CPU avec le bouton de chargement ‚  '. (→ Valeur actuelle →  → )



12. Pour créer le programme avec accès aux éléments de données, **‚OB1'** doit être ouvert dans **‚SIMATIC Manager'** par un double-clic. (→ SIMATIC Manager → OB1)



13. Ensuite validez le paramétrage avec **‚OK'**. (→ OK)



Il existe trois possibilités d'accéder aux éléments de données :

1. Accès à adressage direct :

On peut accéder directement aux éléments de données élémentaires dans un bloc de données avec les commandes suivantes :

Exemples :

L	DB 20. DBB2	Charge l'octet de données 2 depuis DB20 dans ACCU 1
L	DB 22. DBW4	Charge le mot de données 4 depuis DB22 dans ACCU 1
U	DB 2. DBX5.6	Effectue une opération logique ET avec le bit de données 5.6 depuis DB2.

2. Accès aux éléments de données dans les blocs de données préalablement ouverts :

Pour pouvoir accéder aux éléments de données élémentaires, le DB est tout d'abord ouvert avec les commandes AUF DB ou AUF DI. Ensuite, les bits de données élémentaires (DBX/DIX), les octets de données (DBB/DIB), les mots de données (DBW/DIW) ou les doubles mots de données (DBD/DID) sont traités de manière numérique ou par des opérations binaires.
On emploie pour cela principalement AUF DI pour l'ouverture d'instances DB. Il peut aussi être employé pour les DB globaux si deux DB doivent être ouverts simultanément.

Exemple :

AUF	DB 20	Ouverture de DB20
AUF	DI 22	Ouverture de DB22
L	DBW 0	Charge le mot de données 0 depuis DB20 dans ACCU 1
T	MW 1	Transfère contenu depuis ACCU 1 dans le mot mémoire interne 1
U	DIX 0.0	ET logique du bit de données 0.0 depuis DB22
U	E 1.0	Bit d'entrée 1.0
=	A 4.0	Attribution du résultat au bit de sortie 4.0

3. Accès aux données de l'instance locale DB à l'appel d'un bloc de fonction :

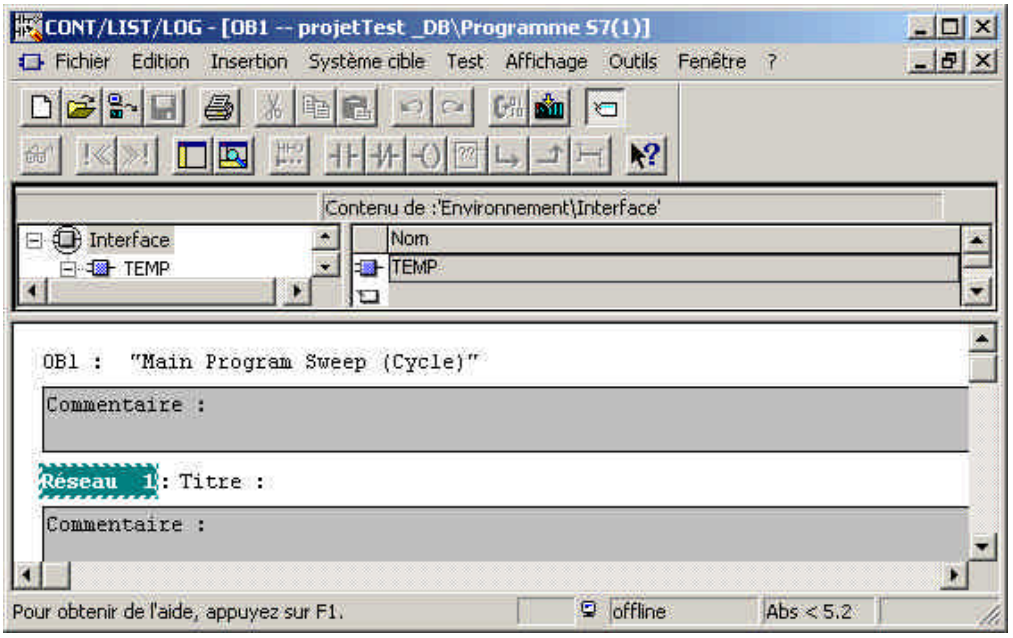
Déjà au niveau de l'appel d'un FB on peut transmettre des données d'une instance correspondante de bloc de données par la commande CALL FB1, DB19. L'attribution des variables, qui sont définies dans le tableau de déclaration de FB et dont les valeurs sont dans DB aux adresses absolues (par ex : EW0, M 10.0 ou AW4) s'effectue directement par la commande CALL.

Exemple :

CALL	FB1, DB19	
ZAHL:=	EW 0	La variable ZAHL est attribuée à EW 0 comme adresse absolue.
AUS:=	A 4.0	La variable AUS est attribuée à A 4.0 comme adresse absolue.

Indications : Le format de la variable et de l'adresse absolue attribuée doivent être identiques.

14. Vous avez maintenant avec **'CONT/LIST/LOG Programmation de blocs S7'** un éditeur qui vous offre la possibilité de créer votre programme STEP7.
Le schéma pour le choix des éléments de données par les boutons est semblable pour les trois premières entrées comme représenté ci-dessous.
Si le bloc d'organisation OB1 est correctement conçu pour tous les boutons, de S0 à S7, il doit être sauvegardé ,  ' et chargé dans le SPS ,  '. . L'interrupteur à clé amovible de la CPU doit être sur la position STOP ! (→  → )



15. En mettant l'interrupteur à clé amovible de la CPU sur RUN, le programme démarre.
Si un des boutons de S0 à S7 est maintenant actionné, les valeurs du modules de sortie à l'adresse AW4 du bloc de données s'affichent.