

Sommaire

	Page
1. Généralités	2
2. Installation du programme S7 et des unités de programmation (POU)	3
2.1 Intégration du projet.....	3
2.2 Configuration matérielle.....	3
2.3 Fichiers GSD / GSE.....	5
2.4 Organisation des POU.....	7
3. Description des Fonctions	8
3.1 Profil DRIVECOM	8
3.1.1 Mot de commande DRIVECOM.....	8
3.1.2 Mot d'état DRIVECOM.....	9
3.2 Fonction FC30	10
3.2.1 Réseau 1, Données process \leftarrow Variateur	11
3.2.2 Réseau 2, Démarrer l'entraînement, mise sous tension	11
3.2.3 Réseau 3, 4 et 5, RFR, HLG (Générateur de rampes) et QSP \Rightarrow Variateur ..	11
3.2.4 Réseau 6, DRIVECOM \Rightarrow Variateur	11
3.2.5 Réseau 7, Mots de données process \Rightarrow Variateur	12
3.2.6 Réseau 8, PZD \Rightarrow Variateur	12
3.2.7 Réseau 9, Paramètres S7 DP \Rightarrow Variateur	13
4. Fonction FC 127	14
5. Bloc de données DB31	15
5.2 Exemple DB31.....	16
5.1 Utilisation de la fonction FC30 avec le bloc de données DB31.....	17

1. Généralités

Les POU développées par Lenze pour S7 sont des exemples d'application globalement valides dans des utilisations avec Profibus-DP. Ces POU doivent aider à la mise en œuvre de projets Profibus-DP avec les produits Lenze. Ils sont gracieusement mis à disposition par la société Lenze. L'utilisateur ne peut donc exiger aucun droit de garantie ou de mise à jour. Aussi, Lenze rejette expressément toute responsabilité de dommages intervenus à la suite de l'utilisation de ces POU.

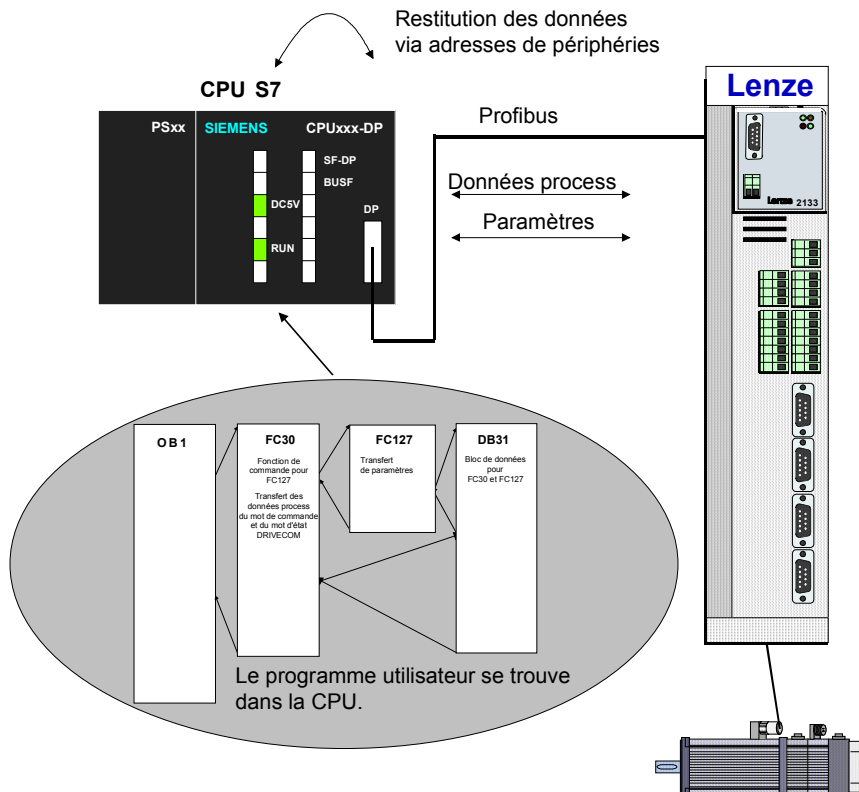
POU : Unité d'organisation de programme

But:

Le réseau Profibus-DP possède un canal paramètres et un canal process. Les POU proposées permettent la lecture/écriture de paramètres, ainsi que l'envoi/réception de données process.

Canal paramètres : Le canal paramètres permet la lecture et l'écriture de paramètres. Ces données ne doivent pas exiger de contraintes temporelles pour les différents participants du bus. Il s'agit, par exemple, de paramètres de fonctionnement (ex : C0011 – N_{max}), d'informations de diagnostic (ex : C0161 – défaut actuel) ou de données moteur (ex : C0015 – fréquence nominale U/f).

Canal process : Les données process doivent être échangées le plus rapidement possible de manière cyclique. Des données prépondérantes, comme le mot de commande, le mot d'état, les consignes et les valeurs réelles sont transmises. La taille du canal process est déterminée lors de la configuration matérielle.



Matériel nécessaire :

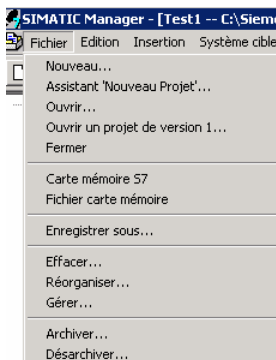
- Variateur avec interface Profibus (ex : AIF (2131 ou 2133), FIF (E82ZAFPC))
- S7-CPU 3xx/4xx avec interface Profibus maître

2. Installation du programme S7 et des unités de programmation (POU)

2.1 Intégration du projet

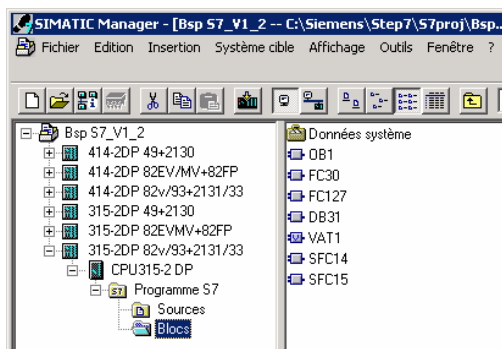
1. Choisir dans SIMATIC-Manager ⇒ **Fichier** ⇒ **Désarchiver**.
2. Sélectionner le fichier exemple **E_S7V1_2.zip**.
3. Sélectionner le répertoire cible (par exemple sous-dossier “\Step7\s7proj”).

Un projet possède plusieurs composants et peut comporter une ou plusieurs stations (S7-300/400). Chaque station dispose d'une CPU (ex : 315-2 DP) et d'un ou plusieurs programmes S7 avec à chaque fois un dossier de POU, un dossier source et une table de symboles.



Via le menu ⇒ **Ouvrir...** du „SIMATIC Manager“ on peut sélectionner le projet.

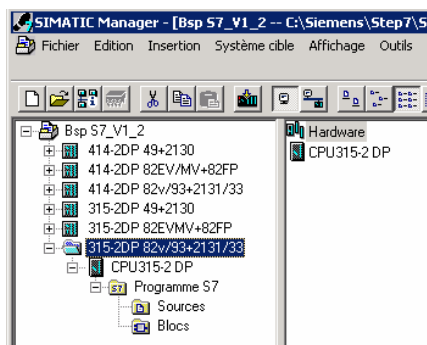
Un nouveau projet peut être créé via le menu ⇒ **Nouveau....**



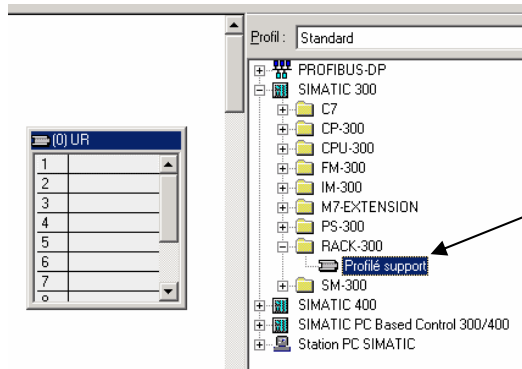
Dans ce projet, on retrouve des sous-parties pour une CPU 315-2 DP et pour une CPU 414-2 DP, ainsi que pour les différents modules Profibus (2130, 2131, etc.). Toute ces sous-parties travaillent avec les mêmes blocs, qui peuvent se distinguer par des longueurs différentes de données process. L'exemple décrit ci-après ne dispose que d'un esclave Profibus. Si plusieurs participants sont présents, les blocs sont utilisables une fois par participant. Comme on ne peut pas travailler plusieurs fois avec les mêmes numéros de blocs, les blocs doivent être renommés pour chaque participant.

2.2 Configuration matérielle

La mise à point de la configuration matérielle est présentée au moyen d'un exemple. Dans cet exemple, on utilise Step 7 Version 5.1 + Hotfix 2 Version K 5.1.0.2.

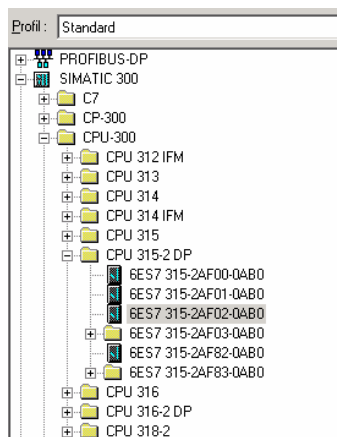


Après avoir créé un nouveau projet et inséré une station (S7-300/400) par le menu ⇒ **Insertion** ⇒ **Station**, un double Click sur **Hardware** dans la fenêtre de droite permet d'accéder à la configuration matérielle.



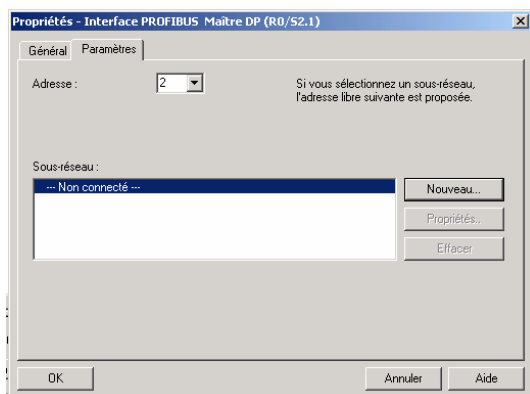
Si le catalogue matériel n'est pas visible, l'activer par le menu ⇒ **Affichage** ⇒ **Catalogue**.

En premier lieu, un 'Profilé support' doit être sélectionné dans le catalogue, sous **RACK-300** et doit être glissé par Drag and Drop dans la fenêtre de gauche.

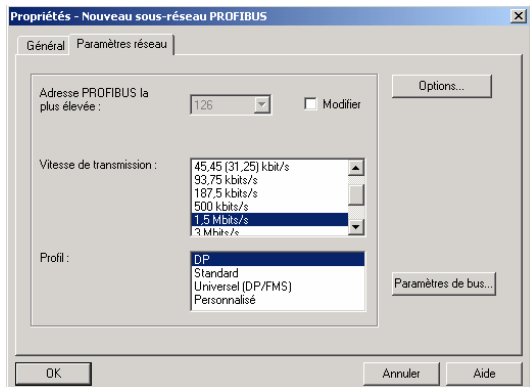


La CPU et le type de bus utilisés peuvent ensuite être rajoutés par Drag and Drop.

Conseil : ⇒ Dans le catalogue, on trouve plusieurs types pour la CPU 315-2 DP. Il faut toujours comparer le numéro de version de la CPU avec celui du catalogue pour éviter les conflits matériels. Le numéro de version de la carte se trouve sur le capot avant.



Si on utilise une CPU avec un Maître DP, la fenêtre „Propriétés – Interface Profibus Maître DP“ s'ouvre automatiquement. On peut ici régler l'adresse du Maître DP et créer un réseau Profibus grâce au bouton ⇒ **Nouveau**.



Dans la fenêtre qui apparaît alors, la vitesse de transmission peut être réglée dans l'onglet 'Paramètres réseau'. Valider ensuite toutes les fenêtres par **OK**. A ce moment, la configuration matérielle est composée d'une alimentation, d'une CPU Maître DP et d'un réseau Profibus.

2.3 Fichiers GSD / GSE

Si vous travaillez pour la première fois avec des variateurs Lenze en relation avec Profibus ou que vous souhaitez insérer des nouveaux fichiers GSD, il faut tout d'abord importer ceux-ci dans le catalogue du matériel.

⇒ Conseil : Les versions actualisées des fichiers Lenze GSD / GSE sont régulièrement mises à jour sur le site Internet de Lenze.

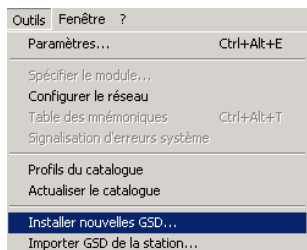
www.lenze.fr (⇒ Service ⇒ Téléchargements ⇒ Versions actualisées ⇒ Profibus)

Installer les fichiers GSD/GSE :

Une configuration ne peut pas être chargée à partir du configurateur de matériel.

Sélectionner le menu ⇒ **Outils** ⇒ **Installer nouvelles GSD**. Sélectionner les nouveaux GSD dans la fenêtre de recherche et valider par **OK**.

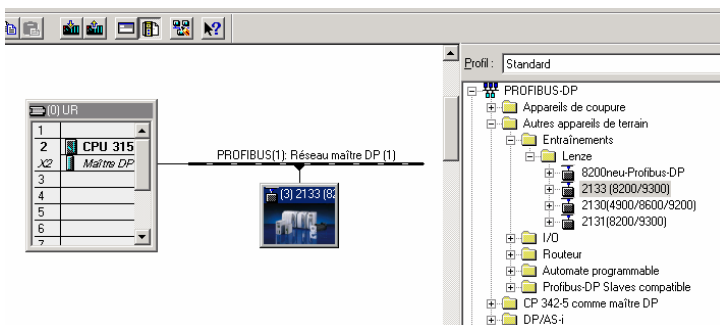
On peut également copier tous les nouveaux fichiers GSD dans le dossier ⇒ **Siemens** ⇒ **Step7** ⇒ **S7data** ⇒ **gsd** et ensuite choisir dans le configurateur le menu ⇒ **Outils** ⇒ **Actualiser le catalogue**.

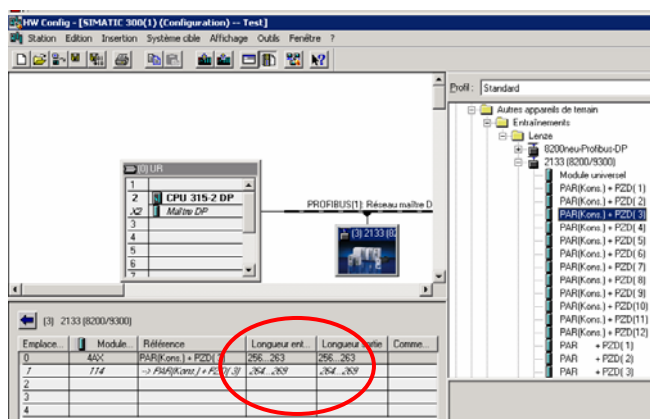


Génération d'un esclave DP :

Les modules Lenze esclaves DP-Slave se trouvent (après installation) dans le catalogue, sous ⇒ **PROFIBUS DP** ⇒ **Autres appareils de terrain** ⇒ **Entraînements** ⇒ **Lenze**

Pour connecter un esclave, il faut glisser par Drag and Drop le dossier 2133 (8200/9300) sur la ligne du réseau Profibus. Dans le champ suivant, l'adresse Profibus doit être réglée. Dans cet exemple, le module 2133 possède l'adresse 3.





En cliquant sur le signe + de chacun des modules de communication, on peut accéder au format des données à échanger. Pour sélectionner le format désiré, il suffit de le marquer, puis de le faire glisser par Drag and Drop dans la table se trouvant sous le matériel sélectionné.

Dans la table correspondant à l'esclave, 2 lignes sont alors remplies.

La première ligne correspond au canal paramètre, la seconde au canal process.

Les adresses d'entrée et de sortie des périphériques sont importantes (rond rouge).

⇒ Indication : La communication peut intervenir par différents formats de données process.

Extrait du catalogue matériel :

- Catalogue
- PROFIBUS DP
- :
- Autres appareils de terrain
- Entraînements
- Lenze
- 8200neu-Profibus DP
- 2133 (8200/9300)
- :
- Par(Kons.) + PZD(1) **AR**
- :
- PPO1**
- :
- Par(Kons.) + PZD(1)
- :
- 2130 (4900/8600/9200)
- 2131 (8200/9300)

Profil	Nom	Remarque
AR	Commande appareil Lenze	Accès direct à AIF-CTRL et STAT
PPO	Profidrive	Etats machine Profidrive
Sans sigle	Drivecom	Etats machine Drivecom

Pour connaître la description du mot de commande et du mot d'état, veuillez consulter les instructions de mise en service des différents modules de communication.

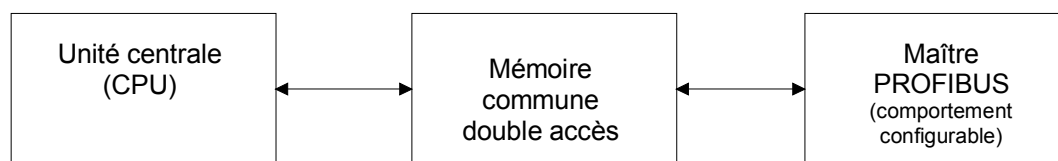
Le suffixe (Kons.) signifie que les données (paramètre et/ou process) sont échangées de manière consistante.

⇒ Conseil : Si des paramètres sont transmis par la fonction FC127 , il faut assurer la consistance des données.

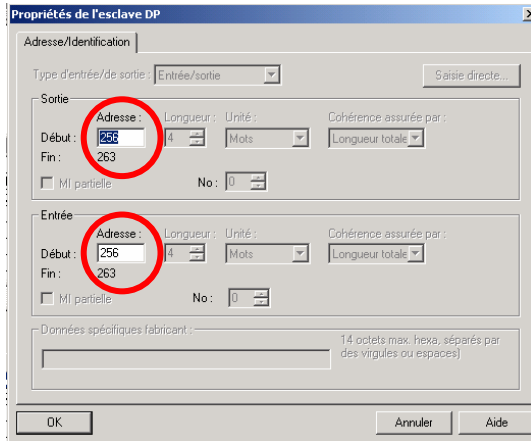
Qu'est-ce que la consistance ?

On parle de consistance, lorsqu'on veut échanger des données de longueur supérieure à un mot (WORD) ou un octet (BYTE). Assurer la „consistance“ consiste à éviter un échange de données erronées entre la CPU et le Maître Profibus via la Mémoire commune doubleaccès (Dual Port Memory). Dans le cas où la consistance n'est pas assurée, la CPU peut, lors d'une requête de lecture, recevoir des données qui ne font pas partie du même bloc et donc interpréter une mauvaise information.

Exemple: un paramètre doit être lu



Le Maître Profibus attend que les données soient complètement lues. Ensuite, il va écrire en une fois l'intégralité des données dans la mémoire commune. Il n'écrit que lorsque la CPU ne lit pas.

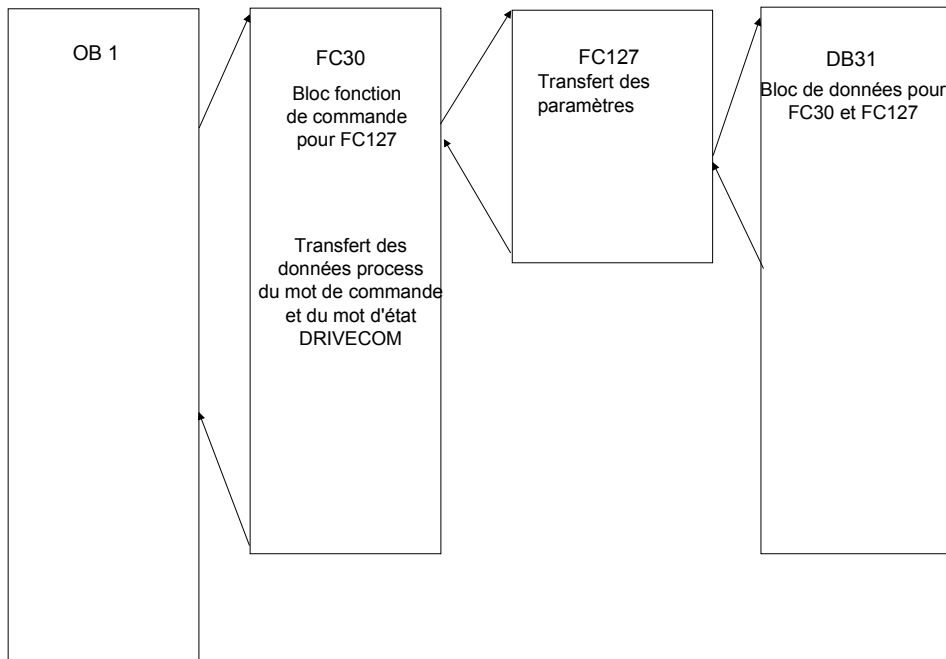


Les adresses de début de périphérie sont à utiliser dans le programme automate associé.

Par un double Click sur chacune des lignes, une fenêtre de dialogue s'ouvre, permettant de modifier les différents domaines d'adresses. Le domaine d'adresses pour une ligne (entrée et sortie) doit toujours être identique (rond rouge); la programmation future en est ainsi facilitée.

2.4 Organisation des POU

Le bloc d'organisation OB1 appelle le bloc de fonction FC30 de manière cyclique.



La fonction FC30 est un exemple. Elle présente les conditions à réaliser pour assurer le transfert des données paramètre et process.

Ce bloc assure, via le canal process, l'écriture du mot de commande DRIVECOM et la lecture du mot d'état DRIVECOM. Il travaille donc selon les états machine DRIVECOM et pilote, en conséquence, le déblocage du variateur via PROFIBUS, à la condition que l'état machine DRIVECOM soit activé. Le bloc FC30 montre ensuite comment la fonction FC127 doit être utilisée et donc comment les paramètres peuvent être écrits ou lus par Profibus. Les codes correspondants, qui doivent être écrits ou lus, sont alors saisis dans le bloc de données DB31.

Pour le transfert des paramètres; les blocs système SFC 14 et SFC 15 sont à charger. Ils assurent ainsi un échange de données consistant.

3. Descriptions des fonctions

Les différents blocs utilisés vont être explicités. Ils sont écrits en langage IL (liste d'instructions).

3.1 Profil DRIVECOM

Le bloc FC30 utilise le profil DRIVECOM. Ceci signifie que le premier mot process de sortie automate (d'entrée variateur) est le mot de commande DRIVECOM, et que le premier mot process d'entrée automate (de sortie variateur) est le mot d'état DRIVECOM. Selon ce profil, un ordre bien précis de commandes doit être respecté lors du déblocage du variateur. Le bloc FC30 traite automatiquement cette succession de commandes.

3.1.1 Mot de commande DRIVECOM

Les numéros de réseaux (NR) de FC30 dans lesquels les bits sont traités sont notés entre parenthèses.

Bit	0 := Mise en service	<- bit géré par le bloc fonction (NR 6) (ne doit pas être géré par le programmeur)
	1 := Blocage tension	<- bit géré par le bloc fonction (NR 6) (ne doit pas être géré par le programmeur)
	2 := Arrêt rapide	<- bit géré par le bloc fonction (NR 6) (ne doit pas être géré par le programmeur)
	3 := Activer le fonctionnement	=> Blocage := 0; Déblocage := 1, (NR 3)
	4 := Arrêt GdR	=> Arrêt GdR (AR) := 0; Arrêt GdR désactivé := 1, (NR 5)
	5 := libre DRIVECOM	Voir instruction de mise en service Profibus (ex : 2131, 2133) ATTENTION ! Réglage usine avec la plupart des variateurs: BLOCAGE_GdR := 0; BLOCAGE_GdR désactivé := 1 (NR 4)
	6 := libre DRIVECOM	Voir instruction de mise en service Profibus (ex : 2131, 2133) ATTENTION ! Réglage usine avec la plupart des variateurs: GdR_ZERO := 0; GdR_ZERO désactivé := 1 (NR 4)
	7 := Réarmement défaut	Front montant de 0 à 1
	8 := réservé DRIVECOM	
	9 := réservé DRIVECOM	
	10 := réservé DRIVECOM	
	11 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	12 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	13 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	14 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	15 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)

3.1.2 Mot d'état DRIVECOM

L'état instantané de l'appareil est décrit dans le mot d'état DRIVECOM.

Bit	0 := Prêt à fonctionner	-> Information pour le bloc fonction
	1 := En service	-> Information pour le bloc fonction
	2 := En cours de fonctionnement->	Information pour le bloc fonction
	3 := Défaut	<= aucun défaut (TRIP) := 0; Défaut (TRIP) := 1
	4 := Blocage tension	-> Information pour le bloc fonction
	5 := Arrêt rapide	Arrêt rapide DRIVECOM := 0; pas d'arrêt rapide := 1
	6 := Blocage	-> Information pour le bloc fonction
	7 := Avertissement	Pas d'avertissement := 0; avertissement := 1
	8 := Message	Pas de message := 0; message := 1
	9 := Remote	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	10 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	11 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	12 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	13 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	14 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)
	15 := libre DRIVECOM	Voir instructions de mise en service Profibus (ex : 2131, 2133)

Les bits libres des mots de commande et d'état sont librement configurables. Reportez-vous aux différentes instructions de mise en service, car l'affectation par défaut des bits peut être différente selon l'appareil utilisé et même selon la configuration présélectionnée.

Quelle documentation technique décrit quoi ?

Instructions de mise en service des modules	N°	Module Lenze pour Profibus
2130	381616	4800, 4900, 8600 et 9200
2131 (max 3 mots process)	410779	8200, 8200vector, 8200motec, Drive PLC et 9300
2133 (max 12 mots process)	412941	
„Modules de fonction bus de terrain pour convertisseur“	411387	8200vector et 8200motec

3.2 Fonction FC30

FC30 est appelé soit par OB1 soit par un autre bloc. Cette fonction va être décrite ici, ligne par ligne, pour le module bus de terrain 2131.

Conseil : Si on veut assurer la consistance de l'échange de données, les blocs système SFC 14 et SFC 15 doivent être chargés dans l'automate.

Le bloc travaille avec deux structures de données temporaires dans la table des variables.

data_receive (Réception des données)
data_send (Envoi des données)

Dans cet exemple, le bloc FC30 est configuré pour une communication avec 3 mots process (PZD). Dans la table de variables, on a donc une longueur de 6 octets (3 mots) par structure.

Indication: Vous trouverez dans la documentation du maître Profibus, si l'échange de données se fait par octets ou par mots.

La table de variables doit être adaptée au nombre de mots process.

Exemple :

Adresse	Déclaration	Nom	Type	Commentaire
+2.0	temp	Mot_Process_1	INT	Mot_Process_1 <- Variateur
+4.0	temp	Mot_Process_2	INT	Mot_Process_2 <- Variateur
+6.0	temp	Mot_Process_3	INT	Mot_Process_3 <- Variateur
+8.0	temp	Mot_Process_4	INT	Mot_Process_4 <- Variateur
=8.0			END_STRUCT	

La table montre la structure data_receive avec 4 PZD. La même opération doit être effectuée pour la structure data_send.

FC30 comporte en tout 9 réseaux et correspond au profil DRIVECOM. Chacun des réseaux va maintenant être explicité.

3.2.1 Réseau 1, Données process \leftarrow Variateur

Réseau 1

```
// 1. #####
// Consistent transfer via 1 process data word
// process data word 0 (status word)
L "PEW 264" // 1. I/O address for process data transfer 264
T LW 0 // temp. local variable
// process data word 1 (actual value)
L "PEW266"
T #data_receive.processdataword_1 // temp. local variable

// 2. #####
// Consistent transfer via 2 process data words
// L PED 264 // 1. I/O address for process data transfer 264
// T LD 0 // temp. local variable

// 3. #####
// Consistent transfer via 2 and more process data words
// CALL "DPRD_DAT"
// LADDR :=W#16#108 // 1. I/O address for process data transfer 264
// RET_VAL:=#data_receive_ret_val //
// RECORD :=#data_receive // process data <- controller
```

Dans le réseau 1, trois transmissions consistantes de données process vers le variateur sont préparées.

1. Transfert consistant via 1 PZD
2. Transfert consistant via 2 PZD
3. Transfert consistant via plus que 2 PZD

Sélectionnez le type de transfert que vous avez choisi lors de la configuration matérielle et supprimez (ou mettez en commentaires) les autres types.

\Rightarrow **Attention**: les adresses des périphériques doivent coïncider avec celles définies lors de la configuration matérielle et seront rentrées en format hexadécimal.

3.2.2 Réseau 2, Variateur déverrouillé, sous puissance

Ce réseau peut être utilisé pour des liaisons avec le programme utilisateur.

3.2.3 Réseaux 3, 4 et 5, RFR, HLG (Générateur de rampes) et QSP \Rightarrow Variateur

Réseau 3

```
SET //replaces links of the application program
= #data_send.control_word.operation_enable
```

Dans ces réseaux, les ordres de commande concernant le déblocage, l'arrêt rapide et le générateur de rampes sont mis dans le bon état. Les ordres SET remplacent des liens du programme utilisateur.

Réseau 4

```
// "RFG stop / enable"
SET // replaces links of the application program
= #data_send.control_word.RFG_no_stop
// "RFG zero/ enable"
= #data_send.control_word.RFG_not_zero
```

Réseau 5

```
SET // replaces links of the application program
= #data_send.control_word.RFG_inhibit
```

3.2.4 Réseau 6, DRIVECOM \Rightarrow Variateur

Réseau 6

```
### Drivecom -> controller
// => "switched on"
SET
S #data_send.control_word.DRIVECOM_switch_on
S
#data_send.control_word.DRIVECOM_inhibit_voltage
S #data_send.control_word.DRIVECOM_quick_stop
// => "Ready to switch on"
UN
#data_receive.status_word.DRIVECOM_ready_to_switch
UN #data_receive.status_word.fault_TRIP
R #data_send.control_word.DRIVECOM_switch_o
```

Le réseau ne peut pas être modifié. Après le déblocage des différentes commandes dans les réseaux 3, 4 et 5 et la saisie d'une consigne de vitesse, le variateur doit passer de l'état DRIVECOM „PRET A FONCTIONNER“ à l'état „EN SERVICE“. Le moteur tourne ensuite à la vitesse pré-définie, à condition que le variateur soit déblocqué par la borne 28.

3.2.5 Réseau 7, Mots de données process ⇒ Variateur

Réseau 7

```
// process data word 1
L 0 // replaces user date
T #data_send.processdataword_1
```

Dans ce réseau, on définit le contenu des données process à envoyer au variateur.

⇒ Indication : La table des variables est à adapter dans le cas où on travaille avec plus de 3 mots process.

3.2.6 Réseau 8, PZD ⇒ Variateur

Réseau 8

```
// 1. #####
// Consistent transfer via 1 process data word
// process data word 0 (control word)
L LW 4 // temp. local variable
T "PAW 264" // 1. I/O address for process data transfer 264
// process data word 1 (setpoint)
L #data_send.processdataword_1 // temp. local variable
T "PAW266"

// 2. #####
// Consistent transfer via 2 process data words
// l LD 4 // temp. local variable
// t PAD 264 // 1. I/O address for process data transfer 264

// 3. #####
// Consistent transfer via 2 and more process data words
// CALL "DPWR_DAT"
// LADDR :=W#16#108 // 1. I/O address for process data transfer 264
// RECORD :=#data_send // process data <- controller
// RET_VAL:=#data_send_ret_val // error code from the SFCs
```

Dans ce réseau, on écrit les données process à envoyer au variateur (correspondant au réseau 7).

Parallèlement au réseau 1, on retrouve les trois possibilités pour le transfert des données. Les propositions non retenues doivent être effacées (ou mises en commentaires).

Les adresses de périphériques sont également à adapter à la configuration matérielle.

Données process – Résumé :

Si on travaille avec 3 PZD (à chaque fois consistant par mot), seuls les réseaux 1 et 8 sont à adapter. Les autres réseaux sont à remplir (2,3,4,5 et 7). Si on travaille avec plus ou moins de PZD, il faut en plus :

- a.) Adapter la table des variables.
- b.) Adapter le réseau 7.

3.2.7 Réseau 9, Paramètres DP S7 ⇒ Variateur

```

### activate request =>
//# 01) C 011 -> controller
//- setpoint -> request list
  L L#3145 // replaces selection of the application program
  L L#10000 // consider 4 decimal positions
  *D // => 3145 min-1
  T "DB31".telegram_001.data
//- activate request
  SET // replaces selection of the application program
  = DB31.DBX 8.6

//# 02) C 051 <- controller
//- ? request ready without error
  O DB31.DBX 16.6 // activate request
  O DB31.DBX 16.7 // request with error
  SPB nOK2
//- actual value <- request list
  L "DB31".telegram_002.data
  L L#10000 // consider 4 decimal positions
  /D // => in min-1
// T xx // replaces selection of the application program
nOK2: NOP 0
//- activate request
  SET // replaces selection of the application program
  = DB31.DBX 16.6

//# 03) C 012 -> controller
//- setpoint -> request list
  L L#123 // replaces selection of the application program
  L L#1000 // 4 decimal places considern
  *D // => 12.3sec
  T "DB31".telegram_003.data
//- activate request
  SET // replaces selection of the application program
  = DB31.DBX 24.6

//# 04) C 013 -> controller
//- setpoint -> request list
  L L#123 // replaces selection of the application program
  L L#100 // 4 decimal places considern
  *D // => 1.23sec
  T "DB31".telegram_004.data
//- request activate
  SET // replaces selection of the application program
  = DB31.DBX 32.6

### process request
CALL FC 127
peripherieadr_1st_byte:=256 // 1. I/O address
DB_transferlist := "DB31" // DB with request list
begin_DB_transferlist := 0 // beginning of the request list in the DB
enable_transfer := TRUE // enable parameter transfer
timeout_timer := T35 // time monitoring

```

Dans le réseau 9, on trouve une liste de requêtes, dans laquelle 4 requêtes de paramètres sont préparées, pour les codes :

C011 (fdmax),
 C051 (fdact),
 C012 (Accelération) et
 C013 (Deceleration).

Une requête (lecture ou écriture) de paramètres a une taille de 8 octets (ou 4 mots). Les différentes requêtes sont écrites dans le bloc de données DB31. A la fin de la liste des requêtes, on appelle FC127, qui va traiter les requêtes dans l'ordre.

Chaque requête est à activer individuellement via l'octet de service Lenze (voir description du DB31, chapitre 5, page 15).

4. Fonction FC 127

Cette fonction traite successivement les requêtes paramètres. Elle ne travaille qu'avec une configuration des paramètres consistante, car les paramètres doivent à priori être transférés de manière consistante.

Attention ⇒ Dans la configuration matérielle, la consistance doit être activée pour l'ensemble du domaine correspondant aux adresses périphériques du canal paramètres.

CALL FC127

peripherieadr_1st_byte	:=256	Adresse de début des périphériques en décimal
DB_transferlist	:=„DB31“	DB avec la liste des requêtes
begin_DB_transferlist	:=0	Début de la liste des requêtes dans le DB
enable_transfer	:=TRUE	Libération du transfert de paramètres, est remis à FALSE après réception d'un accusé de réception sans erreur
timeout_timer	:=T35	Surveillance du temps de traitement de la requête

begin_DB_transferlist: On vient définir ici, à partir de quel numéro de requête du DB31, les requêtes doivent être traitées. A priori, on doit toujours commencer par zéro.

enable_transfer: On démarre le transfert de paramètres grâce à ce bit.

timeout_timer: Le Timer surveille temps de traitement de la requête actuellement traitée. Si la requête n'a pas été traitée correctement au bout de 5s, le bit 7 de l'octet de service est passé à 1. La prochaine requête est traitée, ensuite.

5. Bloc de données DB31

Le DB31 se compose par défaut de 3 requêtes de paramètres mais peut être étendu à volonté. Le format d'une requête est toujours identique. Les 4 premiers mots du bloc ne doivent pas être modifiés.

Les 8 octets d'une requête peuvent être décrits comme suit :

Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8
Octet de service	Sous-index	Index Octet poids fort	Index Octet poids faible	Données 4 / Erreur 4	Données 3 / Erreur 3	Données 2 / Erreur 2	Données 1 / Erreur 1

Octet 1: Octet de service (Servicebyte) => Via cet octet, on vient préciser la longueur des données à transmettre et s'il s'agit d'une requête de lecture ou d'écriture.

Bit 7 6 5 4 3 2 1 0

0 1 => Requête de lecture

1 0 => Requête d'écriture

↳ pas utilisé

↳ réservé

0 0 => Longueur 1 octet

0 1 => Longueur 2 octets

1 0 => Longueur 3 octets

1 1 => Longueur 4 octets

La longueur des données est définie pour chaque paramètre dans la table des attributs du manuel système (chapitre „table des codes“).

↳ **Bit Handshake :**

Pour activer une requête de lecture ou d'écriture, ce bit doit être fixé par l'utilisateur.

Le bloc remet à zéro ce bit après une transmission correcte (avec ou sans erreur)

↳ Bit d'erreur: Dans le cas d'une erreur lors de la requête, ce bit est mis à 1.

Si la requête suivante se déroule sans erreur, le bit est remis à zéro après le traitement.

Octet 2 : Sous-index du code, qui doit être lu ou écrit.

Octet 3-4 : Index du code, qui doit être lu ou écrit, Calcul : $24575 - N^{\circ} \text{ Code}$.

Octet 5-8 : Données utiles, qui doivent être écrites dans le variateur lors d'une requête d'écriture ou qui doivent être envoyées à la commande lors d'une requête de lecture.

Données :

- Vous trouverez la longueur des données et le format du code dans la table d'attribut du manuel système correspondant au variateur. Le format le plus répandu est le format à virgule fixe avec 4 décimales.

Pour ce format, les paramètres sont à multiplier par 10000.

Dans le bloc de données DB31, les informations suivantes sont transmises pour chacune des requêtes de paramètres:

- Choix d'une requête de lecture ou d'écriture
- Longueur des données
- Index (correspondant au numéro de code du paramètre)
- Sous-index du numéro de code du paramètre
- Données utiles

Conseils lors de l'occurrence d'erreurs

- Le bit d'erreur (bit 7) de l'octet de service doit être surveillé. Si une erreur est signalée lors d'une requête de paramètre, un code d'erreur se trouve dans les octets de données (Octets 5-8). La description du contenu des données de l'éventuel message d'erreur se trouve dans la documentation technique du module Profibus utilisé.
- Lors de l'occurrence d'une erreur, on trouve „error“ (mot double) à l'adresse 4 du bloc de données DB31 (réseau 4 et 6) du dernier code d'erreur qui est apparu. Cette plage d'adresse est définie dans la mémoire, de telle façon que le contenu ne soit écrasé que lors de l'occurrence d'une nouvelle erreur.

5.2 Exemple DB31

La valeur du code C012 doit être réglée à 12,3.

Marche à suivre:

	telegram_001STRUCT			##### C 012 -> Variateur
1.0	servicebyte	BYTE	B#16#32	Octet de service -> Lenze
2.0	subindex	BYTE	B#16#0	Sous-index (Sous-code)
4.0	index	INT	24563	Index = 24575 – N°Code.
8.0	data	DINT	L#123000	Données utiles (attention au format décimal)
=8.0	END_STRUCT			

Précision:

Dans la première colonne, on trouve le domaine de mémoire correspondant aux types de données utilisés.

BYTE	:= une place mémoire
WORD/INT	:= deux places mémoires
DINT/DWORD	:= quatre places mémoires

Dans l'octet de service, on doit déclarer une requête d'écriture pour une longueur de données de 4 octets. Le code qui doit être écrit ne possède pas de sous-code (=0).

$$24575 - 12 = 24563$$

Le temps d'accélération doit être réglé à 12,3 s. Dans le domaine „données“, on doit trouver la valeur correspondante.

$$12,3 \times 10000 = 123000 \text{ (format à 4 décimales)}$$

5.1 Interaction de la fonction FC30 avec le bloc de données DB31

Le bloc de données DB31 est renseigné par les requêtes à réaliser.
Deux alternatives sont alors proposées.

- 1.) La liste de requêtes est traitée dans FC30 et les données sont écrites à partir de là dans le DB31.
- 2.) Dans le DB31, toutes les données nécessaires sont déclarées en tant que valeurs initiales.

Pour lancer une requête, il faut mettre à 1 le bit Handshakebit (bit 6) de l'octet de service. Cette opération s'effectue dans le FC30 (réseau 9), qui commande FC127.

Les lignes de programme marquées en gras permettent de fixer le bit Handshake.

```
//# 03) C 012 -> controller
//- setpoint -> request list
L   L#123           // replaces selection of the application program
L   L#1000          // 4 decimal places considern
*D                               // => 12.3sec
T   "DB31".telegram_003.data
//- activate request
SET                // replaces selection of the application program
=   DB31.DBX 24.6
```

La fonction FC127 inverse automatiquement l'état du bit Handshake après un transfert correct, à la condition que la réponse soit positive. Si le paramètre doit être lu de manière cyclique, il faut donc forcer en permanence le Handshake à 1 (SET) dans la fonction de commande (par exemple FC30).

La fonction FC127 parcourt le bloc de données DB31 et vérifie si une requête a été activée. Le FC127 recommence un nouveau traitement du bloc de données, si l'information „30 hex“ est présent dans l'octet de service de la dernière requête.