PilMot

DLL PilMot / Tango-Ctrl

Manuel d'utilisation

MUT/1107/0242C/NBA-PPI/GC

Juin 2014

ZI de la Vaure – BP 30940
42290 SORBIERS - France
Tél : 04 77 53 30 48 – Fax : 04 77 53 38 61
Email : contact@rsautomation.com

HISTORIQUE DU DOCUMENT

Indice	Date	Auteur	Approbateur	Description de la révision
A	07/2011	NBA	PPI	Création
B	11/2011	NBA	PPI	Ajout fonctions pour modules TOR et BFp
C	06/2014	NBA	BMO	Mise à jour pour module Bim2Ax-BO

SOMMAIRE

1. BUT

Ce document a pour objectif de présenter les fonctions de la Dll PilMot Tango Control.
Cette dll permet le contrôle de modules d'axes Pas à Pas BOCM ou BOC ou BO, de module d'axes moteur à courant continu BFp, et de modules TOR (12DI8RO, 16RO et 24DI).

Abréviations utilisées :

- Dll : Dynamic Link Library
- OpenMB : Open ModBus
- Maj : Mise à jour
- Err : Erreur
- RAZ : Remise à zéro
- N.S. : Non Significatif
- POM : Prise d'Origine Mécanique
- FDC : Fin De Course
- E/S : Entrées / Sorties
- TOR : Tout Ou Rien
- ANA : Analogique

Documents associés :

« Bim 2AX-BOCM – Manuel d'utilisation »
Réf. : MUT/0712/0116x/JCB-SPI/MS

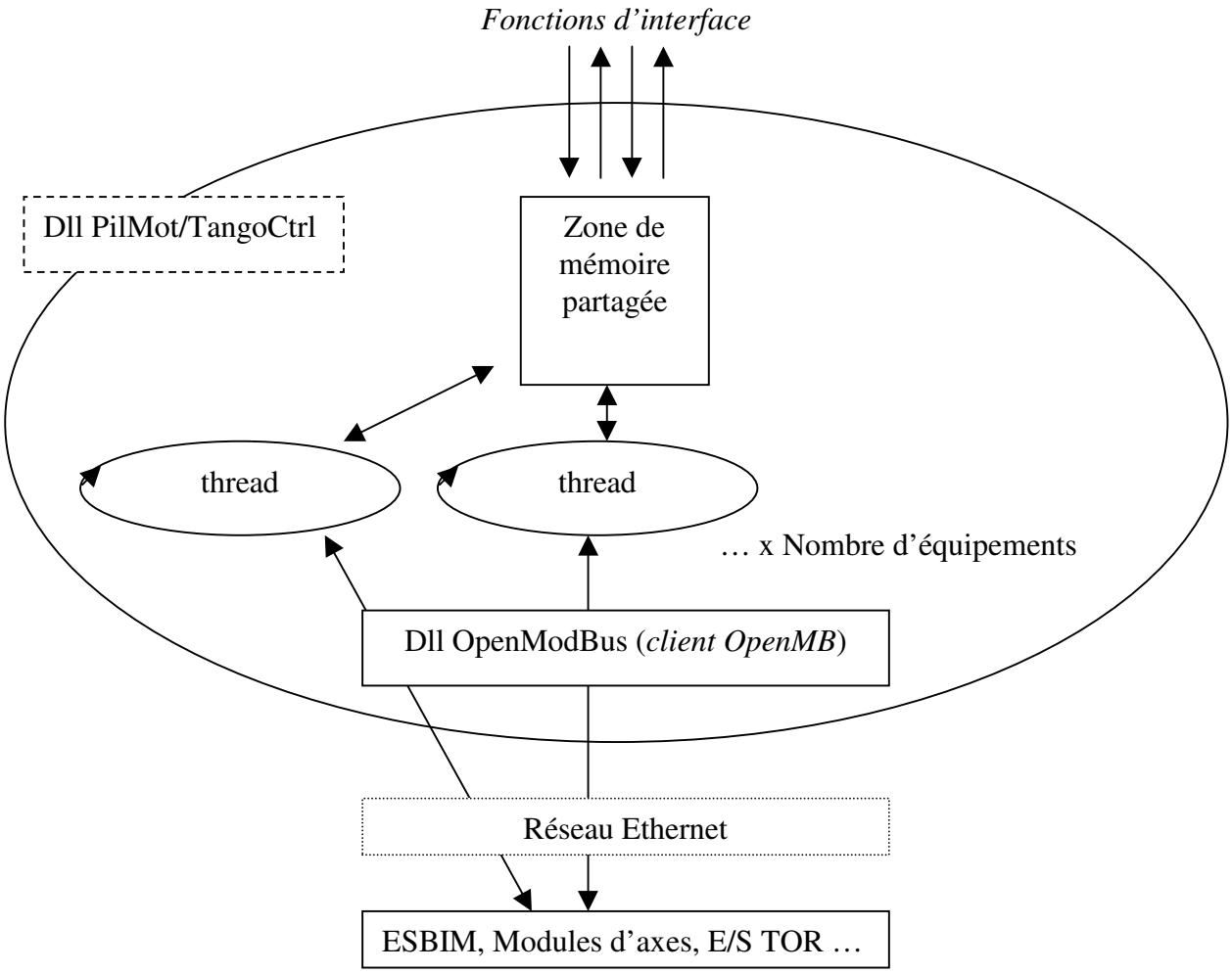
« Bim 2AX-BOC – Manuel d'utilisation »
Réf. : MUT/0205/0126x/GCA-MVE/GC

« Bim 2AX-BO – Manuel d'utilisation »
Réf. : MUT/9802/0042x/GCA-MVE/GC

« Bim 2AX-BFP – Manuel d'utilisation »
Réf. : MUT/0601/0180x/NBA-PPI/GC

2. PRESENTATION

La DLL PilMot Tango-Ctrl permet de communiquer avec les modules d'axes en s'affranchissant de la couche de gestion du protocole de communication OpenModBus.
Le schéma ci-dessous présente l'architecture générale de cette dll.



Caractéristiques :

La DLL PilMot Tango-Ctrl crée pour chaque équipement scruté, un thread client Open ModBus.
Chaque thread est cadencé à une période de 10ms ; il peut y avoir théoriquement jusqu'à 100 équipements définis pour une application.
En cas d'erreur de connexion avec son équipement, la connexion est rétablie au bout de 2 secondes.
Au niveau Open ModBus, le Time Out d'attente de réception d'une réponse est d'environ 500 ms.

Remarques :

Il n'y a pas de contrôle sur les valeurs des paramètres des fonctions de la DLL

3. FONCTIONS

3.1.1 Fonction « Start »

Description :

La fonction 'Start' permet l'initialisation et le lancement des thread de communication avec les équipement sur le réseau Ethernet / openMB.
Cette fonction doit être exécutée une seule fois au démarrage de l'application.

***Prototype* :**

int Start (int NbEquipt, char (*ListAdrIP) [16])

Paramètres en entrée :

int NbEquipt — Nombre d'équipements (ou ESBIM) à scruter
$1 \leq NbEquipt \leq 100$

char * ListAdrIP[16] — Liste des adresses IP des équipements à scruter

Chaque adresse IP est définie sous forme d'une chaîne de caractères terminée par '\0'
Exemple : Chaîne de caractères correspondant à l'adresse IP = « 192.8.79.130 »

Ind	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
val	'1'	'9'	'2'	'.'	'8'	'.'	'7'	'9'	'.'	'1'	'3'	'0'	'\0'	*N.S.*	*N.S.*	*N.S.*

Valeur retournée :

Compte-rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

3.1.2 Fonction « Stop »

Description :

La fonction 'Stop' permet de fermer les threads précédemment ouverts par la fonction 'Start'. Cette fonction doit être exécutée une fois avant de quitter le programme d'application.

***Prototype* :**

int Stop (void)

Paramètres en entrée :

Aucun

Valeur retournée :

Compte-rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

3.1.3 Fonction « rEtatConnexion »

Description :

La fonction 'rEtatConnexion' permet de lire l'état de la connexion Ethernet OpenMB avec l'équipement défini en paramètre.

***Prototype* :**

int rEtatConnexion (int NumEquipt)

Paramètres en entrée :

int NumEquipt — Numéro / Indice de l'équipement à lire
0 ≤ NumEquipt ≤ 99

(Cette valeur correspond à l'indice de la liste définie dans la fonction 'Start' (listAdrIP))

Valeur retournée :

Etat de la connexion :

- 0 : Etat REPOS — Pas de connexion
- 1 : Etat Initialisation — Connexion en cours d'initialisation
- 2 : Etat Identification modules — Lecture et maj de la config matérielle des cartes BIM
- 3 : Etat Echange données — Connexion Ok, échanges OpenMB Ok
- 4 : Etat Erreur connexion — Erreur de communication avec l'équipement
- 5 : Etat Close connexion — Fermeture de la connexion openMB
- 5 : Etat Attente Fin Timout — Erreur – Attente fin Timout avant essai reconnexion

3.1.4 Fonction « rPositionMot »

Description :

La fonction 'rPositionMot' permet de lire la valeur théorique du moteur en nombre de pas (ou ½ pas, ou micro pas ... selon le mode de configuration du module)
Cette fonction est valable pour les modules Pas à Pas BO/BOC/BOCM uniquement.

***Prototype* :**

int rPositionMot (int NumEquipt, int NumMot)

Paramètres en entrée :

int NunEquipt	Numéro / Indice de l'équipement à lire 0 ≤ NumEquipt ≤ 99
int NumMot	Numéro du moteur de l'équipement à lire 1 ≤ NumMot ≤ 14

Valeur retournée :

Position théorique du moteur en nombre de pas

3.1.5 Fonction « rPositionCod_Pt »

Description :

La fonction 'rPositionCod_Pt' permet de lire la position du codeur en nombre de points (⇔ valeur brute du codeur)
Cette fonction est valable pour les modules BOC/BOCM et les modules BFp.

***Prototype* :**

int rPositionCod_Pt (int NumEquipt, int NumCod)

Paramètres en entrée :

int NumEquipt	Numéro / Indice de l'équipement à lire 0 ≤ NumEquipt ≤ 99
int NumCod	Numéro du codeur de l'équipement à lire 1 ≤ NumCod ≤ 14

Valeur retournée :

Position du codeur en nombre de points

3.1.6 Fonction « rPositionCod_Pas »

Description :

La fonction 'rPositionCod_Pas' permet de lire la position du codeur en nombre de pas (= valeur brute du codeur × Para « Dividende » / para « Diviseur »)
Cette fonction est valable pour les modules Pas à Pas BOC/BOCM uniquement.

***Prototype* :**

int rPositionCod_Pt (int NumEquipt, int NumCod)

Paramètres en entrée :

int NumEquipt	Numéro / Indice de l'équipement à lire 0 ≤ NumEquipt ≤ 99
int NumCod	Numéro du codeur de l'équipement à lire 1 ≤ NumCod ≤ 14

Valeur retournée :

Position du codeur en nombre de pas

3.1.7 Fonction « rEtatMoteur »

Description :

La fonction 'rEtatMoteur' permet de lire le registre d'état d'un moteur
La description détaillée de ce registre est disponible dans le manuel d'utilisation du module BOCM ou dans le manuel d'utilisation du module BFp.
Cette fonction est valable pour les modules BO/BOC/BOCM et les modules BFp.

***Prototype* :**

Unsigned int rEtatMoteur (int NumEquipt, int NumMot)

Paramètres en entrée :

int NumEquipt — Numéro / Indice de l'équipement à lire
0 ≤ NumEquipt ≤ 99

int NumMot — Numéro du moteur de l'équipement à lire
1 ≤ NumMot ≤ 14

Valeur retournée :

Valeur du registre d'état du moteur

Module BOCM (+BOC + BO)			**Module BFp**		
Bit	**Commentaire**	**Val hexadécimal**	**Bit**	**Commentaire**	**Val hexadécimal**
0	Butée fin de course positive	**0001**	**0**	Erreur présente (détail reg Err)	**0001**
1	Butée fin de course négative	**0002**	**1**	Non Prêt à être Validé	**0002**
2	Butée logicielle positive	**0004**	**2**	Prêt à être Validé	**0004**
3	Butée logicielle négative	**0008**	**3**	Validé	**0008**
4	Prise origine faite	**0010**	**4**	Mouvement en Cours	**0010**
5	Axe en mouvement	**0020**	**5**	Servo hors fenêtre d'arrêt	**0020**
6	Axe en arrêt brutal	**0040**	**6**	Réservé	**0040**
7	Phase dévalidée	**0080**	**7**	Réservé	**0080**
8	Non utilisé	**0100**	**8**	Mode Vitesse	**0100**
9	Phase ouverte	**0200**	**9**	Mode Manuel	**0200**
10	Court circuit entre phase	**0400**	**10**	Mode Positionneur	**0400**
11	Puissance absente	**0800**	**11**	Réservé	**0800**
12	Non utilisé	**1000**	**12**	Réservé	**1000**
13	came origine active	**2000**	**13**	Prise d'origine faite	**2000**
14	Err. Envoi Ordre / cde	**4000**	**14**	Réservé	**4000**
15	Non utilisé	**8000**	**15**	Réservé	**8000**

3.1.8 Fonction « rEtatCodeur »

Description :

La fonction 'rEtatCodeur' permet de lire le registre d'état d'un codeur
La description détaillée de ce registre est disponible dans le manuel d'utilisation du module BOCM.
Cette fonction est valable pour les modules Pas à Pas BOC/BOCM uniquement.

***Prototype* :**

Unsigned int rEtatCodeur (int NumEquipt, int NumCod)

Paramètres en entrée :

int NumEquipt — Numéro / Indice de l'équipement à lire
$0 \leq NumEquipt \leq 99$

int NumCod — Numéro du codeur de l'équipement à lire
$1 \leq NumCod \leq 14$

Valeur retournée :

Valeur du registre d'état du codeur

Bit	Commentaire	Valeur hexadécimal
0	Diviseur = 0	**0001**
1	Défaut Codeur	**0002**
2	Défaut bouchon de configuration	**0004**
3	N.S.	**0008**
4	Au moins 1 correction effectuée	**0010**
5	Défaut Alarme (Codeur Endat)	**0020**
6	N.S.	**0040**
7	N.S.	**0080**
8	N.S.	**0100**
9	N.S.	**0200**
10	N.S.	**0400**
11	N.S.	**0800**
12	N.S.	**1000**
13	Présence Codeur Endat	**2000**
14	Présence Codeur SSI	**4000**
15	Présence Codeur Relatif	**8000**

3.1.9 Fonction « rAcqCde »

Description :

La fonction 'rAcqCde' permet de lire le registre d'acquittement des commandes moteur ; ce registre correspond à la dernière commande exécutée.
Cette fonction est valable pour les modules Pas à Pas BO/BOC/BOCM uniquement.

***Prototype* :**

Unsigned int rAcqCde (int NumEquipt, int NumMot)

Paramètres en entrée :

int NumEquipt — Numéro / Indice de l'équipement à lire
$0 \leq NumEquipt \leq 99$

int NumMot — Numéro du moteur de l'équipement à lire
$1 \leq NumMot \leq 14$

Valeur retournée :

Valeur du registre d'acquittement de commande

Bit	Commentaire	Valeur hexadécimal
0	POM	**0001**
1	Mouvement Abs / Rel	**0002**
2	N.S.	**0004**
3	Abandon mouvement	**0008**
4	Mouvement manuel sens PLUS	**0010**
5	Mouvement manuel sens MOINS	**0020**
6	Déplacement en FDC PLUS	**0040**
7	Déplacement en FDC MOINS	**0080**
8	Arrêt brutal	**0100**
9	Dévalidation des phases	**0200**
10	N.S.	**0400**
11	N.S.	**0800**
12	N.S.	**1000**
13	N.S.	**2000**
14	N.S.	**4000**
15	Test Moteur	**8000**

3.1.10 Fonction « rResCdeParam »

Description :

La fonction 'rResCdeParam' permet de lire le résultat d'une commande sur les paramètres. Cette fonction est valable pour les modules Pas à Pas BO/BOC/BOCM uniquement.

***Prototype* :**

unsigned int rResCdeParam(int nEquipt, int numMot, int *numPara, int *ValPara)

Paramètres en entrée :

int nEquipt — Numéro / Indice de l'équipement à lire
$0 \leq nEquipt \leq 99$

int NumMot — Numéro du moteur de l'équipement à lire
$1 \leq NumMot \leq 14$

Paramètres en sortie :

int *NumPara — Numéro du paramètre lu ou écrit

int *ValPara — Valeur du paramètre écrit ou lu

Valeur retournée :

Acquittement de la commande envoyée ; indique la dernière commande exécutée

Valeur	Commentaire
0x1000	Ecriture Paramètre
0x2000	Lecture Paramètre
0x4000	Lecture Bouchon de configuration

3.1.11 Fonction « rRegErreur »

Description :

La fonction 'rRegErreur' permet de lire le registre d'erreur d'un module BFp. Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int rRegErreur (int nEquipt, int numMot)

Paramètres en entrée :

int nEquipt — Numéro / Indice de l'équipement à lire
$0 \leq nEquipt \leq 99$

int NumMot — Numéro du moteur de l'équipement à lire
$1 \leq NumMot \leq 14$

Valeur retournée :

Valeur du registre d'erreur

Bit	Commentaire	Valeur hexadécimal
0	Réservé	**0001**
1	Valeur d'un paramètre statique refusée	**0002**
2	Ordre invalide	**0004**
3	Ordre en cours de mouvement	**0008**
4	Consigne non reçue	**0010**
5	Erreur de poursuite supérieure au Maxi	**0020**
6	Vitesse trop élevée	**0040**
7	Réservé	**0080**
8	Puissance absente	**0100**
9	Butée Fin de course PLUS	**0200**
10	Butée Fin de course MOINS	**0400**
11	Butée logicielle PLUS	**0800**
12	Butée logicielle MOINS	**1000**
13	Défaut Codeur	**2000**
14	Défaut Température pont puissance	**4000**
15	Entrée Origine	**8000**

3.1.12 Fonction « rErrPours »

Description :

La fonction 'rErrPours' permet de lire l'erreur de poursuite courante d'un axe d'un module BFp.

Erreur de Poursuite = position théorique – position actuelle

Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int rErrPours (int nEquipt, int numMot)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à lire $0 \leq nEquipt \leq 99$
int NumMot	Numéro du moteur de l'équipement à lire $1 \leq NumMot \leq 14$

Valeur retournée :

Valeur de l'erreur de poursuite (en nombre de points codeur)

3.1.13 Fonction « rSommeErrP »

Description :

La fonction 'rSommeErrP' permet de lire le cumul de toutes les erreurs de poursuite d'un axe d'un module BFp. (Elle doit être stable et le plus proche possible de zéro)

Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int rSommeErrP (int nEquipt, int numMot)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à lire $0 \leq nEquipt \leq 99$
int NumMot	Numéro du moteur de l'équipement à lire $1 \leq NumMot \leq 14$

Valeur retournée :

Valeur de la somme des erreurs de poursuite (en nombre de points codeur)

3.1.14 Fonction « rResCdeParaBFp »

Description :

La fonction 'rResCdeParaBFp' permet de lire le résultat d'une commande sur les paramètres. Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int rResCdeParaBFp (int nEquipt, int numMot, int *numPara, int *ValPara)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à lire 0 ≤ nEquipt ≤ 99
int NumMot	Numéro du moteur de l'équipement à lire 1 ≤ NumMot ≤ 14

Paramètres en sortie :

int *NumPara	Numéro du paramètre lu ou écrit
int *ValPara	Valeur du paramètre écrit ou lu

Valeur retournée :

Acquittement de la commande envoyée :

- '0' : commande correcte
- '1' : commande invalide

3.1.15 Fonction « rIdModule »

Description :

La fonction 'rIdModule' permet de lire le code Identificateur d'un module BIM

***Prototype* :**

unsigned int rIdModule (int nEquipt, int nModule)

Paramètres en entrée :

int nEquipt — Numéro / Indice de l'équipement à écrire
0 ≤ nEquipt ≤ 99

int nModule — Numéro du module BIM de l'équipement (ESBIM)
1 ≤ nModule ≤ 7

Valeur retournée :

Valeur du code identificateur lu.

Module BIM	Code Identificateur	Description
24DI	0x00	24 entrées TOR
16RO	0x01	16 sorties TOR
ANA4I4O	0x02	4 entrées ANA 4 Sorties ANA
4COD	0x03	4 Codeurs
6CT	0x04	6 entrées compteur
ANA8O	0x05	8 sorties ANA
12DI8RO	0x06	12 entrées TOR 8 sorties TOR
ANA8I	0x07	8 entrées ANA
12CT	0x08	12 entrées Compteur
ANA6I	0x12	6 entrées ANA
2AX_BO	0x4C	2 Axes Pas à Pas
2AX_BOC / BOCM	0x4E	2 Axes Pas à Pas + Codeur
2AX_BFP	0x4F	2 Axes Boucle fermé moteur CC
CDE_LOCALE	0xFC	Commande locale en cours (Type fictif)
INACCESSIBLE	0xFD	Module inaccessible (Type fictif)
INCONNU	0xFE	Module inconnu (Type fictif)
ABSENT	0xFF	Module absent / Emplacement libre (fictif)

3.1.16 Fonction « wCdeMot »

Description :

La fonction 'wCdeMot' permet d'envoyer une commande au moteur (Ecriture du registre de commande du moteur)
Cette fonction est valable pour les modules Pas à Pas BO/BOC/BOCM uniquement.

***Prototype* :**

unsigned int wCdeMot (int nEquipt, int nMot, unsigned int regCde, int consPos, int consVit)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à écrire 0 ≤ nEquipt ≤ 99
int nMot	Numéro du moteur de l'équipement à lire 1 ≤ nMot ≤ 14
unsigned int regCde	registre de commande (cf. description ci-dessous)
int consPos	Consigne de position pour la commande spécifiée
int consVit	Consigne de vitesse pour la commande spécifiée

Valeur retournée :

Compte rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

Valeur du registre de commande

Bit	Commentaire	Valeur hexadécimal
0	POM	**0001**
1	Mouvement Absolu	**0002**
2	Mouvement Relatif	**0004**
3	Abandon mouvement	**0008**
4	Mouvement manuel sens PLUS	**0010**
5	Mouvement manuel sens MOINS	**0020**
6	Déplacement en FDC PLUS	**0040**
7	Déplacement en FDC MOINS	**0080**
8	Arrêt brutal	**0100**
9	Dévalidation des phases	**0200**
10	RAZ position	**0400**
11	Preset Côte (non valable pour module BO)	**0800**
12	Forçage à la dernière position mémorisée	**1000**
13	N.S.	**2000**
14	N.S.	**4000**
15	Test Moteur	**8000**

3.1.17 Fonction « wCdeParam »

Description :

La fonction 'wCdeParam' permet d'envoyer une commande sur les paramètres ou sur le module. Cette fonction est valable pour les modules Pas à Pas BO/BOC/BOCM uniquement.

***Prototype* :**

unsigned int wCdeParam (int nEquipt, int nMot, unsigned int regCde, int numPara, int valPara)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à écrire 0 ≤ nEquipt ≤ 99
int nMot	Numéro du moteur de l'équipement à lire 1 ≤ nMot ≤ 14
unsigned int regCde	registre de commande (cf. description ci-dessous)
int numPara	numéro du paramètre à accéder
int valPara	Valeur du paramètre associée à la commande

Valeur retournée :

Compte rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

Valeur du registre de commande

Valeur	Commentaire
0x1000	Ecriture Paramètre
0x2000	Lecture Paramètre
0x4000	Lecture Bouchon de configuration (non valable pour les modules BO)
0x8000	Reset du module BIM (non valable pour les modules BO)

A noter :

- ***La description des paramètres est détaillée dans le manuel d'utilisation du module d'axes BOCM***
- ***Cette fonction accède aux paramètres du module d'axes sur lequel est connecté le moteur (2 moteurs sont définis par module)***

3.1.18 Fonction « wCdeMotCC »

Description :

La fonction 'wCdeMotCC' permet d'envoyer une commande au moteur d'un axe BFp (Ecriture de la zone principale des registres de commande du moteur)
Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int wCdeMotCC (int nEquipt, int nMot, unsigned int regCde, int consigne, int consVit, int AccelSens, int Decel)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à écrire (0 ≤ nEquipt ≤ 99)
int nMot	Numéro du moteur de l'équipement à lire (1 ≤ nMot ≤ 14)
unsigned int regCde	registre de commande (cf. description ci-dessous)
int consigne	Consigne pour la commande spécifiée (position, offset…)
int consVit	Consigne de vitesse pour la commande spécifiée (en pts cod/ms)
int AccelSens	Consigne d'accélération ou de sens pour la commande spécifiée
int Decel	Consigne de décélération pour la commande spécifiée (en ms)

Valeur retournée :

Compte rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

- Valeur du registre de commande

Bit	Commentaire	Val hexa
0	Demande d'initialisation de l'axe	**0001**
1	Demande de dévalidation de l'axe	**0002**
2	Demande de validation	**0004**
3	Réservé	**0008**
4	Commande de départ de mouvement	**0010**
5	Commande d'arrêt de mouvement	**0020**
6	Réservé	**0040**
7	Demande de mise à zéro de l'erreur de poursuite	**0080**
8	Sélection du mode Vitesse	**0100**
9	Sélection du mode Manuel	**0200**
10	Sélection du mode Positionneur	**0400**
11 à 13	Réservé	
14	Demande de Prise d'origine sur la position actuelle	**4000**
15	Demande de Prise d'origine sur entrée Origine	**8000**

- Valeur du registre « Consigne »

  Ce registre a différente signification selon le mode de fonctionnement sélectionné, et selon le mouvement demandé :

Registre « Consigne »	
Mode / Commande	**Signification du registre**
Mode Positionneur	Consigne de position, en nombre de points codeur
Prise d'origine	Offset, en nombre de points codeur
Mode Vitesse	Consigne de vitesse en pourcentage de la vitesse Maxi du moteur (valeur de -100 à +100 %)

- Valeur du registre « AccelSens »

  Ce registre a différente signification selon le mode de fonctionnement sélectionné et selon le mouvement demandé :

Registre « Accélération »	
Mode / Commande	**Signification du registre**
Mode Positionneur	Accélération en ms
Prise d'origine	Sens de déplacement de l'axe : 0 = sens NEGATIF 1 = sens POSITIF
Mode Manuel	

Remarque :

La valeur des registres 'Accélération' et 'Décélération' est limitée par les paramètres statiques « Accélération minimum » et « Décélération minimum ». Les valeurs écrites dans ces registres doivent être supérieures ou égales à celles des paramètres statiques correspondants pour être prises en compte.

La première écriture sur un module BFp n'est pas prise en compte, il faut une écriture pour initialiser le module

3.1.19 Fonction « wCdeParaBFp »

Description :

La fonction 'wCdeParaBFp' permet d'envoyer une commande sur les paramètres du module. Cette fonction est valable pour les modules BFp uniquement.

***Prototype* :**

unsigned int wCdeParam (int nEquipt, int nMot, unsigned int regCde, int numPara, int valPara)

Paramètres en entrée :

int nEquipt	Numéro / Indice de l'équipement à écrire (0 ≤ nEquipt ≤ 99)
int nMot	Numéro du moteur de l'équipement à lire (1 ≤ nMot ≤ 14)
unsigned int regCde	registre de commande (cf. description ci-dessous)
int numPara	numéro du paramètre à accéder
int valPara	Valeur du paramètre associée à la commande

Valeur retournée :

Compte rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

Valeur du registre de commande

Valeur	Commentaire
0x00	Ecriture Paramètre
0x01	Sauvegarde des paramètres + Initialisation du module
0x02	Lecture Paramètre

A noter :

- ***La description des paramètres est détaillée dans le manuel d'utilisation du module d'axes BFp***
- ***La première écriture sur un module BFp n'est pas prise en compte, il faut une écriture pour initialiser le module***

3.1.20 Fonction « r_TOR »

Description :

La fonction 'r_TOR' permet la lecture de l'état des entrées / sorties d'un module TOR.

***Prototype* :**

unsigned int r_TOR (int nEquipt, int nModule)

Paramètres en entrée :

int nEquipt — Numéro / Indice de l'équipement à lire
0 ≤ nEquipt ≤ 99

int nModule — Numéro du module TOR de l'équipement à lire
1 ≤ nModule ≤ 7

Valeur retournée :

- Registre de l'état des entrées Sorties (cf. ci-dessous)
- « -1 » (⇔ 0xFFFFFFFF) en cas d'erreur

Valeur de l'état des E/S :

Chaque bit du registre correspond à une entrées ou sortie TOR
Une valeur à 1 du bit indique un entrée ou une sortie active
Une valeur à 0 indique un entrée ou une sortie inactive
Exemple de valeur retournée pour un module BIM24DI avec les entrées n°3 et 10 actives :
Valeur lue = 0x0000204

Module	Masque
Bim24DI	Entrées 0x00000001 ⇔ entrée n°1 0x00000002 ⇔ entrée n°2 … jusqu'à n°24
Bim16RO	Sorties 0x00000001 ⇔ sortie n°1 0x00000002 ⇔ sortie n°2 … jusqu'à n°16
Bim12DI8RO	Entrées 0x00000001 ⇔ entrée n°1 0x00000002 ⇔ entrée n°2 … 0x00000800 ⇔ entrée n°12 Sorties 0x00010000 ⇔ sortie n°1 0x00020000 ⇔ sortie n°2 … 0x00800000 ⇔ sortie n°8

3.1.21 Fonction « w_TOR »

Description :

La fonction 'w_TOR' permet l'écriture des sorties TOR d'un module 16RO ou 12DI8RO.

***Prototype* :**

unsigned int w_TOR (int nEquipt, int nModule, unsigned int regCdeSortieTOR)

Paramètres en entrée :

int nEquipt Numéro / Indice de l'équipement à écrire
$0 \leq nEquipt \leq 99$

int nModule Numéro du module TOR de l'équipement à écrire
$1 \leq nModule \leq 7$

unsigned int regCdeSortieTOR registre de commande (cf. description ci-dessous)

Valeur retournée :

Compte rendu d'exécution de la fonction

- 1 : Exécution correcte
- 0 : Erreur

Valeur du registre de commande :

Chaque bit du registre correspond à une sortie TOR
Une valeur à 1 du bit active la sortie, la valeur 0 reset la sortie
Exemple de valeur du registre pour l'écriture des sorties n°1 et n°3 :
regCdeSortieTOR = 0x0005

Valeur	Sortie TOR correspondante
0x0001	Sortie n°1
0x0002	Sortie n°2
0x0004	Sortie n°3
0x0008	Sortie n°4
…	…

A noter :

- ***La première écriture d'un module TOR n'est pas prise en compte ; il faut une écriture du module pour l'initialiser***