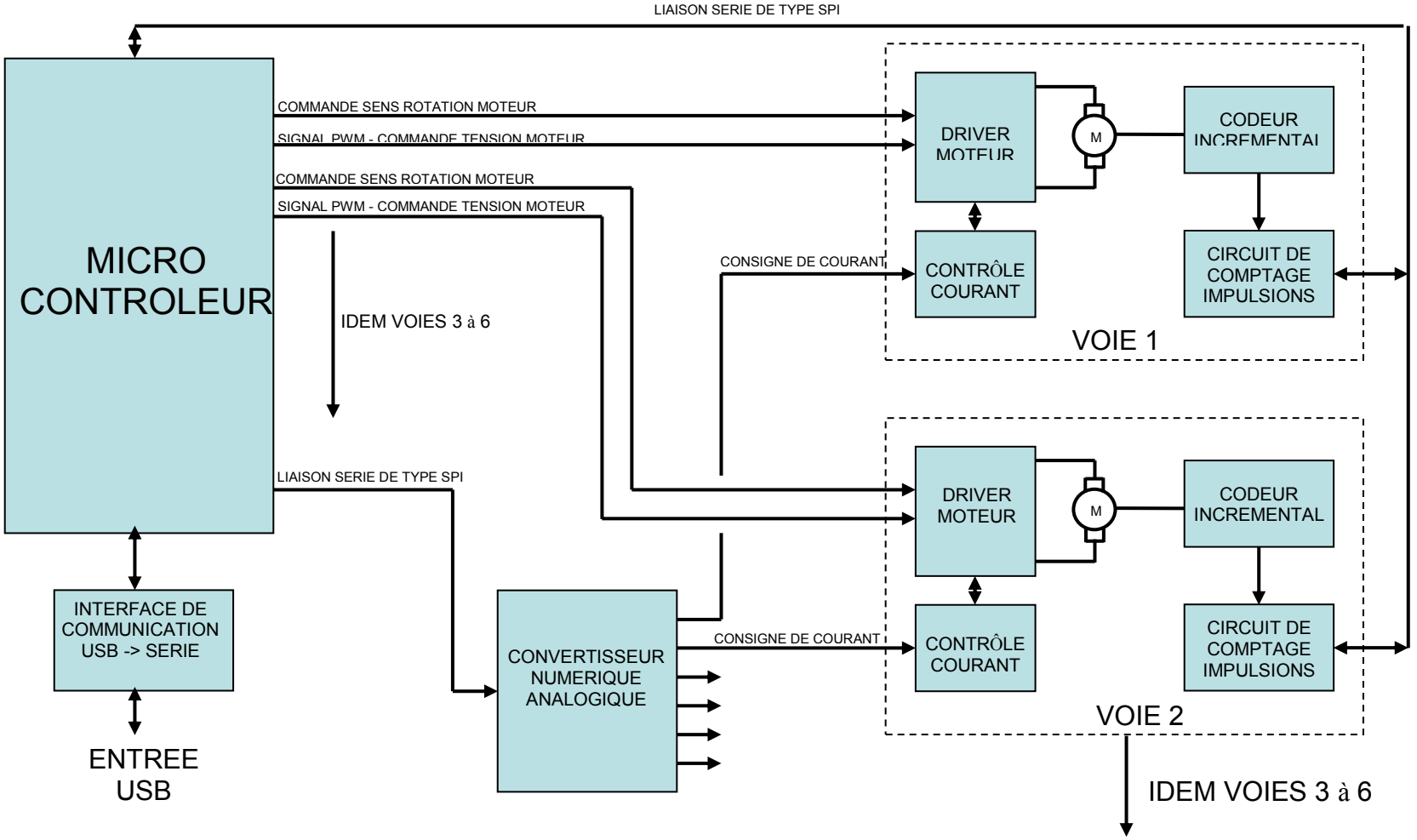
NOTICE UTILISATION MAINS ROBOTISEES

1. Description générale

1.1. Vue d'ensemble



1.2. Présentation

L'électronique de la main robotisée est principalement composée des éléments suivants

- Un microcontrôleur principal
- Un circuit d'interface USB vers COM
- Six circuits de comptage des impulsions codeurs
- Six drivers moteurs
- Un convertisseur de numérique en analogique à 8 sorties
- Six systèmes de contrôle du courant moteur

1.3. Microcontrôleur principal

Le microcontrôleur principal est de type ARM7 cadencé à 72 Mhz. Il réalise l'interface entre le PC et les différents composants nécessaires à la commande des moteurs.

Il intègre entre autres :

- Un module PWM avec 6 sorties. La largeur d'impulsion est individuellement réglable pour chaque sortie.
- Un module UART pour la communication avec le PC
- Deux modules SPI pour la communication avec les composants périphériques

Le programme qu'il embarque a été développé en langage C à l'aide de l'interface de développement "IAR Embedded Workbench"

A ce jour, ce programme permet deux modes de fonctionnement :

Un mode nommé : Mode manuel
Dans ce mode, le PC envoie des consignes de tension, de polarité ou de limite de courant au microcontrôleur qui les applique directement aux moteurs indiqués.

Un mode nommé : Mode consignes de positions
Dans ce mode le PC envoie des consignes de positions au microcontrôleur qui déroule un algorithme visant à placer et maintenir chaque moteur à la position indiquée. Cet algorithme prend en compte un certain nombre de paramètres permettant d'ajuster les mouvements selon les besoins de l'application. Les paramètres sont modifiables à tout moment depuis le PC. En parallèle il met à disposition du PC un certain nombre de registres indiquant l'état en cours.

Par ailleurs, une fonction particulière permet d'enregistrer un nouveau programme dans la mémoire du microcontrôleur. Ce programme est transmis par le PC au travers du port de communication sans qu'aucun démontage ne soit nécessaire.

1.4. Circuit d'interface USB – COM

Un FT232 de la société FTDI réalise l'interface entre un port USB du PC et un port UART du microcontrôleur de sorte que la liaison soit vue de part et d'autre comme une liaison série de type port COM.

1.5. Circuits de comptage des impulsions codeurs

Les capteurs de positions sont des codeurs incrémentaux. En exploitant les fronts montants et descendants de leurs 2 sorties on obtient 32768 positions par tour.
Les circuits de comptage sont au nombre de 6 (un par codeur). Ce sont de petits microcontrôleurs programmés spécifiquement pour assurer cette fonction. Ils sont vus par le microcontrôleur principal comme des périphériques dialoguant par une liaison type SPI.
Le microcontrôleur principal les adresse tour à tour pour obtenir la position de chaque codeur. Les lectures sont espacées d'environ 166 microsecondes, ainsi la position de chaque codeur est relue une fois par milliseconde.

1.6. Drivers moteurs

Ce sont des circuits intégrés destinés à la commande des moteurs. Ils sont au nombre de 6 et sont commandés en appliquant des signaux logiques sur deux de leurs entrées.

Entrée "ENABLE"
Lorsqu'un état logique 1 est appliqué sur cette entrée, la tension d'alimentation (12V) est appliquée aux bornes du moteur.
Lorsque cet état logique passe à 0, plus aucune tension n'est appliquée aux bornes du moteur.
Cette entrée est raccordée à une sortie PWM (ou MLI) du microcontrôleur, de sorte qu'en appliquant une impulsion plus ou moins large on obtienne une tension moyenne plus ou moins élevée aux bornes du moteur.

Entrée "SENS"
L'état logique sur cette entrée détermine la polarité de la tension appliquée aux bornes du moteur et par conséquent le sens de rotation de celui-ci.

1.7. Convertisseur de numérique en analogique à 8 sorties

Ce composant convertit des valeurs numériques reçues en entrée en tensions analogiques sur ses sorties. Les tensions sur chaque sortie sont commandées indépendamment les unes des autres et sont modifiables à tout moment depuis le PC.
Le microcontrôleur communique avec le convertisseur au travers d'une liaison série de type SPI.
Six des huit sorties sont utilisées comme tensions de consignes pour les systèmes de contrôle du courant.

1.8. Recherche des positions initiales

Les capteurs de position sont des codeurs incrémentaux, par conséquent, à la mise sous tension, le microcontrôleur n'a aucune information quant à la position réelle des doigts.
La recherche des positions initiales consiste à placer chaque doigt en butée mécanique (fermeture maximale) puis à initialiser le compteur d'impulsion à la valeur correspondant au déplacement maximal. Cette opération nécessite qu'il n'y ait aucun obstacle (objet ou doigt) sur les courses d'ouverture et fermeture.
Cette procédure est initiée par le PC et doit impérativement être effectuée au moins une fois après la mise sous tension pour une utilisation normale de la main.
Tant que cette procédure n'a pas été faite, les consignes de position sont ignorées et les consignes de tension ne peuvent pas dépasser +/- 3 Volts.

1.9. Réinitialisation des paramètres

Cette commande permet d'initialiser tous les paramètres aux valeurs par défaut.
Elle affecte aussi le paramètre indiquant au microcontrôleur s'il s'agit de la main droite ou de la main gauche. Ce paramètre est transmis dans la commande, par conséquent, il est important de s'assurer de sa cohérence.

Pour info :

Pour des raisons pratiques, mains droite et gauche utilisent des circuits imprimés identiques sur lesquels le raccordement des moteurs et codeurs diffère.
Le microcontrôleur utilise le paramètre main gauche / main droite pour aiguiller les entrées et sorties vers les voies physiques adéquates.

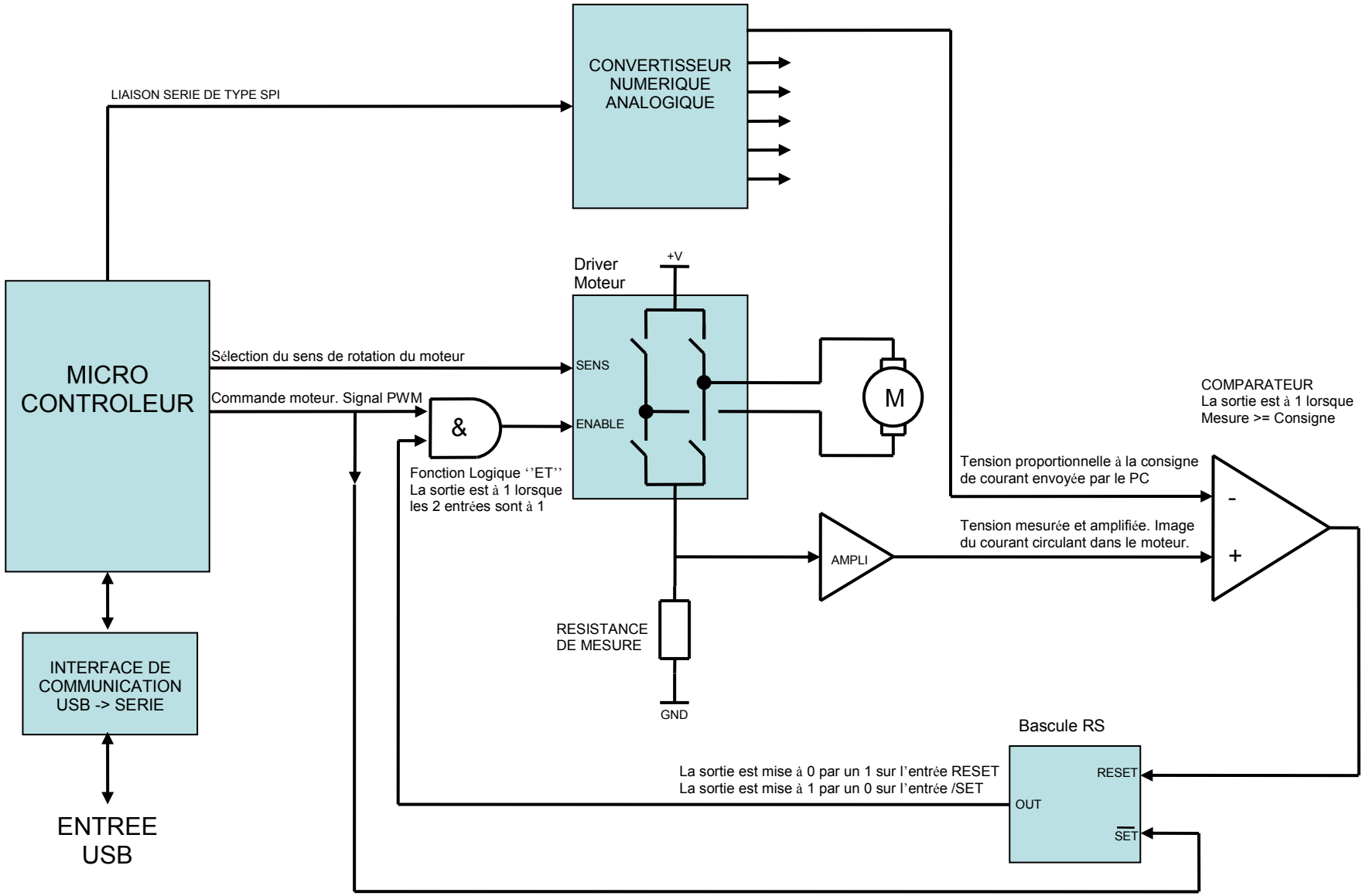
1.10. Systèmes de contrôle du courant moteur

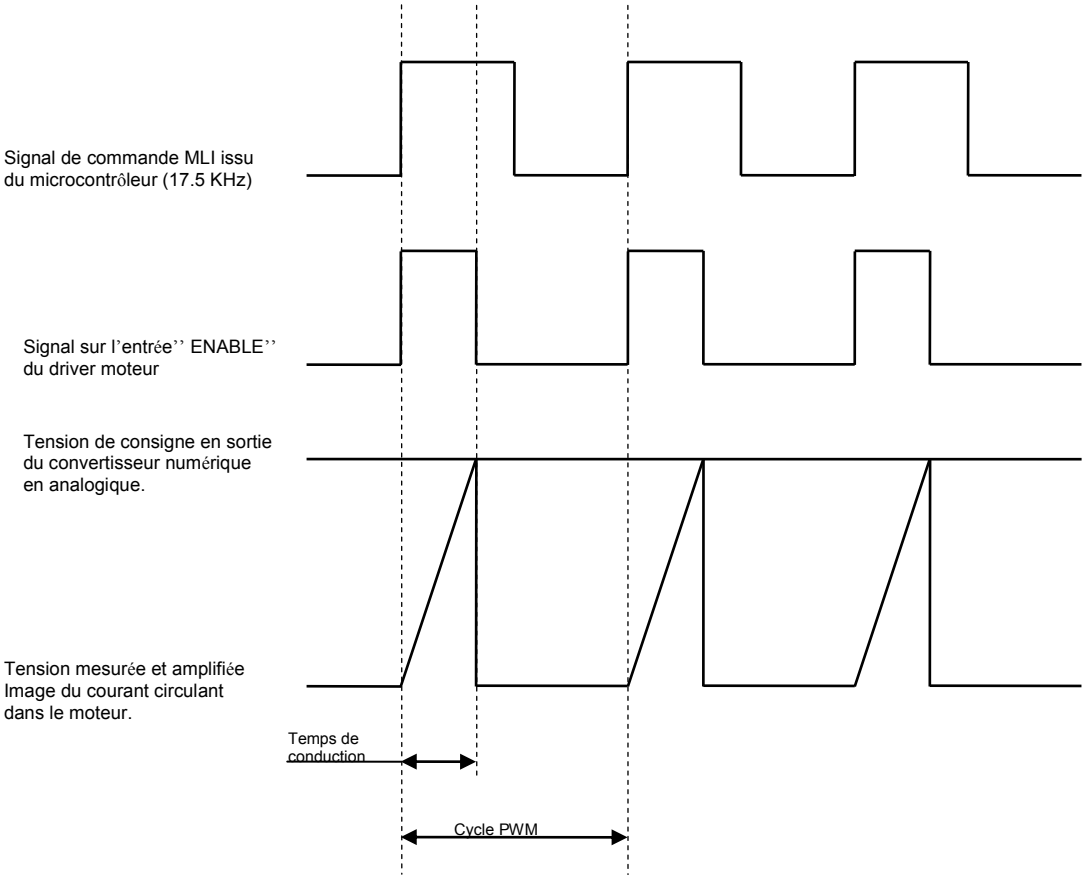
Chaque moteur est commandé par une sortie PWM (ou MLI) du microcontrôleur. La fréquence du signal PWM est fixée à environ 17,5KHz et la largeur d'impulsion est réglable avec une résolution de 12 bits (4096 valeurs possibles)

Le système de contrôle du courant utilise le fait que lorsque l'on applique une tension aux bornes du moteur, le courant s'établit progressivement dans celui-ci.
A chaque cycle du PWM, la tension aux bornes du moteur est coupée dès lors que le courant atteint une valeur de consigne, on obtient ainsi un courant moyen dans le moteur qui est fonction de cette consigne.

Pour ce faire, le système mis en œuvre va mesurer le courant circulant dans le moteur, le comparer à la valeur de consigne et désactiver le signal PWM dès que le courant mesuré atteint la consigne. Le signal PWM restera désactivé jusqu'à la fin du cycle et l'opération sera renouvelée à chaque cycle.

CONTRÔLE DU COURANT MOTEUR - SCHEMA DE PRINCIPE





Fonctionnement :

Lorsque le signal PWM issu du microcontrôleur passe à 1, la sortie de la bascule RS est aussi à 1.

L'entrée ENABLE du driver moteur passe donc à 1 et la tension d'alimentation (12V) est appliquée aux bornes du moteur.

Le courant circulant dans le moteur traverse la résistance de mesure et une tension image de ce courant apparaît aux bornes de la résistance.

Après amplification, cette tension est appliquée sur l'entrée "+" du comparateur.

La sortie du comparateur passe à 1 lorsque la tension sur son entrée "+" devient supérieure à celle sur son entrée "-". (Tension mesurée > tension de consigne)

Cette transition en sortie du comparateur place la sortie de la bascule "RS" à 0, ce qui fait passer la sortie de la "porte &" à 0.

L'entrée "ENABLE" du driver moteur passe donc à 0 et l'alimentation du moteur est coupée.

Lorsque le signal PWM repasse à 0 la sortie de la bascule "RS" est remise à 1 autorisant ainsi la commande du driver au prochain front montant du signal PWM.

Le signal PWM devra toujours être inférieur à 100 % puisque son front descendant est utilisé pour réinitialiser le système. Cette précaution a été prise au niveau du logiciel.

2. Protocole de communication

2.1. Introduction

Chaque moteur est mécaniquement couplé à un capteur de position de type codeur incrémental. L'ensemble ainsi formé est appelé voie. Les voies sont numérotées de 0 à 5 comme suit :

- Voie 0 = Rotation Pouce
- Voie 1 = Pouce
- Voie 2 = Index
- Voie 3 = Majeur
- Voie 4 = Annulaire
- Voie 5 = Auriculaire

A chaque voie est affecté un ensemble de registres définissant son fonctionnement. Ces registres sont regroupés dans un tableau (1 par voie) dans un ordre défini. Le N° d'ordre dans le tableau d'un registre donné est appelé N° de registre. (La numérotation des registres commence à 0)

L'accès à un registre particulier d'une voie donnée se fait par l'intermédiaire de sa position mémoire. La position mémoire est calculée de la manière suivante :

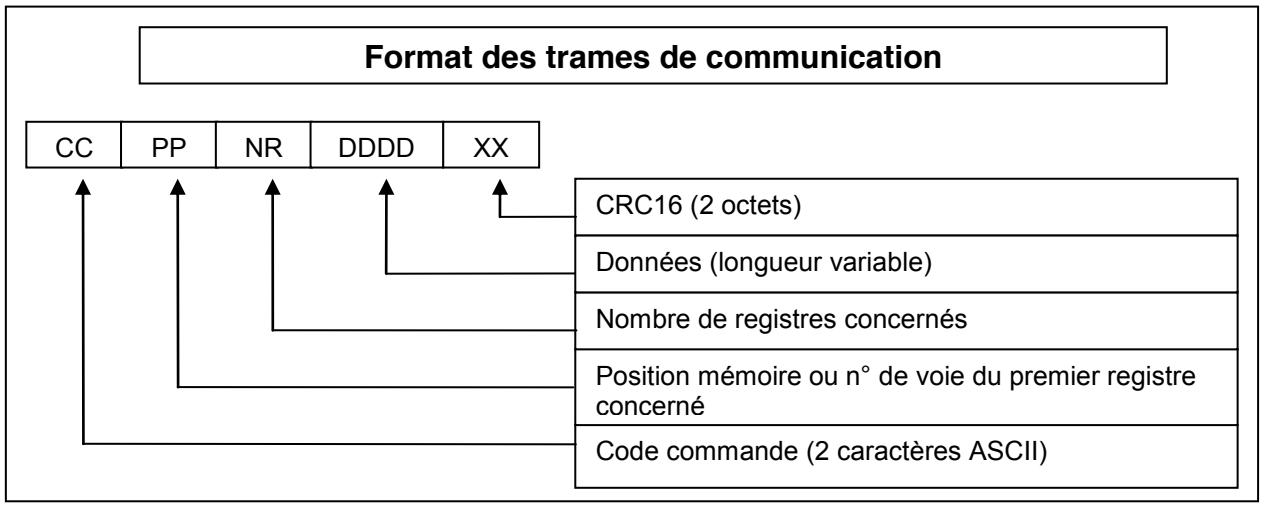
- Position mémoire = ((N° de Voie + 1) * 1000) + N° du registre.
- Exemple : La position mémoire du registre N°5 de la voie 2 est : ((2 + 1) * 1000) + 5 = 3005

Par ailleurs, il existe 2 registres indépendants des numéros de voie qui sont situés aux positions mémoire 100 et 200. Ils permettent pour l'un de démarrer la procédure de recherche de la position initiale et pour l'autre de charger le jeu de paramètres par défaut.

2.2. Généralités

La main robotisée intègre un composant qui réalise l'interface entre un port USB du PC et un port UART du microcontrôleur de sorte que la liaison soit vue de part et d'autre comme une liaison série de type port COM. La vitesse de communication est de **460 800** bauds.

La communication entre le PC et la main est basée sur des lectures et écritures de registres. Elle est toujours initiée par le PC qui envoie une trame de requête à laquelle la main répond par une trame de réponse.



Les trames commencent toujours par un code de commande servant à identifier le type de requête. Les codes de commandes implémentés à ce jour sont :

- RD : Lecture de "n" registres à partir d'une position mémoire donnée

- WR : Ecriture de "n" registres à partir d'une position mémoire donnée

- W1 : Ecriture des registres "MODE_CMD_MOTEUR" de "n" voies consécutives et lecture des registres "POSITION_CODEUR" des "n" voies.

- W2 : Ecriture des registres "CONSIGNE_TENSION_POSITION" de "n" voies consécutives et lecture des registres "POSITION_CODEUR" des "n" voies.

- W3 : Ecriture des registres "LIMITE_COURANT" de "n" voies consécutives et lecture des registres "POSITION_CODEUR" des "n" voies.

- W4 : Ecriture des registres "MODE_CMD_MOTEUR" de "n" voies consécutives et lecture des registres "VITESSE_MOTEUR" des "n" voies.

- W5 : Ecriture des registres "CONSIGNE_TENSION_POSITION" de "n" voies consécutives et lecture des registres "VITESSE_MOTEUR" des "n" voies.

- W6 : Ecriture des registres "LIMITE_COURANT" de "n" voies consécutives et lecture des registres "VITESSE_MOTEUR" des "n" voies.

- BL : Demande de chargement d'un nouveau programme dans le microcontrôleur.

Les commandes RD et WR permettent de lire ou d'écrire une suite de registres en indiquant la position mémoire du premier registre à lire ou écrire.

Les commandes W1 à W6 effectuent des lectures et écritures de registres en un seul échange. D'une manière générale, elles permettent d'écrire l'un des registres de n voies consécutives et d'obtenir en réponse l'un des registres de ces n voies.

Toutes les valeurs numériques sont transmises directement en hexadécimal en commençant par l'octet de poids le plus faible et en allant jusqu'à l'octet de poids le plus fort.

Les transmissions sont sécurisées par l'ajout d'un CRC16 en fin de trame. (Voir exemples de programme de calcul en fin de chapitre)

A réception d'une trame, le microcontrôleur vérifie que :

- Le CRC16 reçu est identique à celui qu'il a recalculé à partir des octets reçus.
- Le code commande est connu.
- Les positions de départ et de fin sont contenues dans sa zone mémoire
- La commande demandée est autorisée dans la zone mémoire spécifiée. (Pas d'écriture des registres en lecture seule)

Si toutes ces conditions sont remplies, la trame est prise en compte et une réponse est retournée.

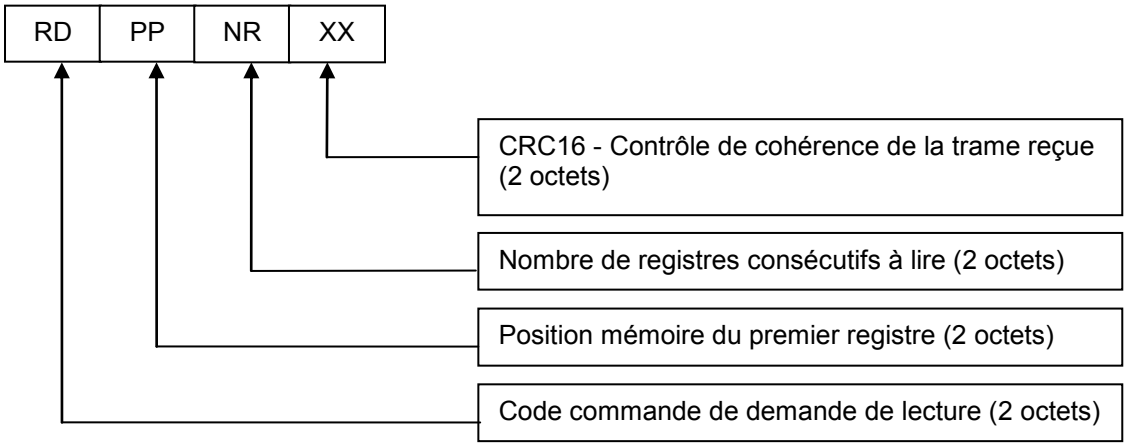
Le microcontrôleur détecte la fin d'une trame lorsqu'il ne reçoit plus de données pendant 100us.

Lorsque aucune trame n'est reçue du PC pendant 500ms, les voies configurées pour fonctionner en mode consigne de tension sont automatiquement désactivées.
Les registres (MODE_CMD_MOTEUR) et (CONSIGNE_TENSION_POSITION) sont remis à 0.

2.3. Format des trames de communication

Code commande RD - Format de la trame émise par le PC

Demande de lecture de n registres consécutifs à partir d'une position mémoire



Exemple : Demande de lecture de 2 registres consécutifs à partir de la position 1000 :

Octets émis : 0x52, 0x44, 0xE8, 0x03, 0x02, 0x00, 0x39, 0x66

0x52 = Code ascii du caractère R
0x44 = Code ascii du caractère D

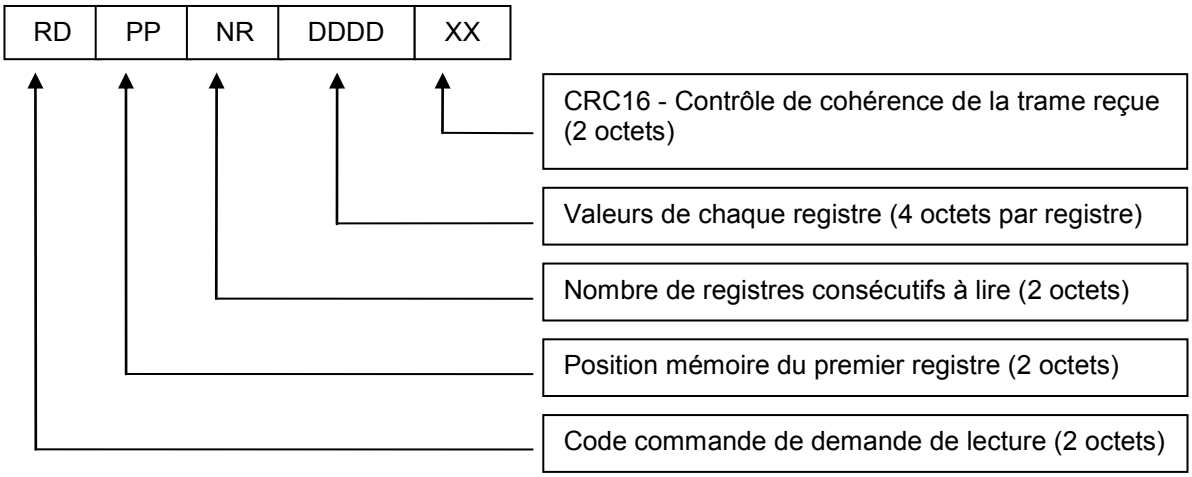
0xE8 = Poids faible de la première position mémoire
0x03 = Poids fort de la première position mémoire

0x02 = Poids faible du nombre de registres
0x00 = Poids fort du nombre de registres

0x39 = Poids faible du CRC16
0x66 = Poids fort du CRC16

Code commande RD - Format de la trame renvoyée par la main

Réponse à une demande de lecture de n registres consécutifs à partir d'une position mémoire



Exemple : Réponse à une demande de lecture de 2 registres consécutifs à partir de la position 1000

Octets reçus : 0x52, 0x44, 0xE8, 0x03, 0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0xA8, 0x61, 0x00, 0x00, 0x75, 0x0A

0x52 = Code ascii du caractère R
0x44 = Code ascii du caractère D

0xE8 = Poids faible de la première position mémoire
0x03 = Poids fort de la première position mémoire

0x02 = Poids faible du nombre de registre
0x00 = Poids fort du nombre de registre

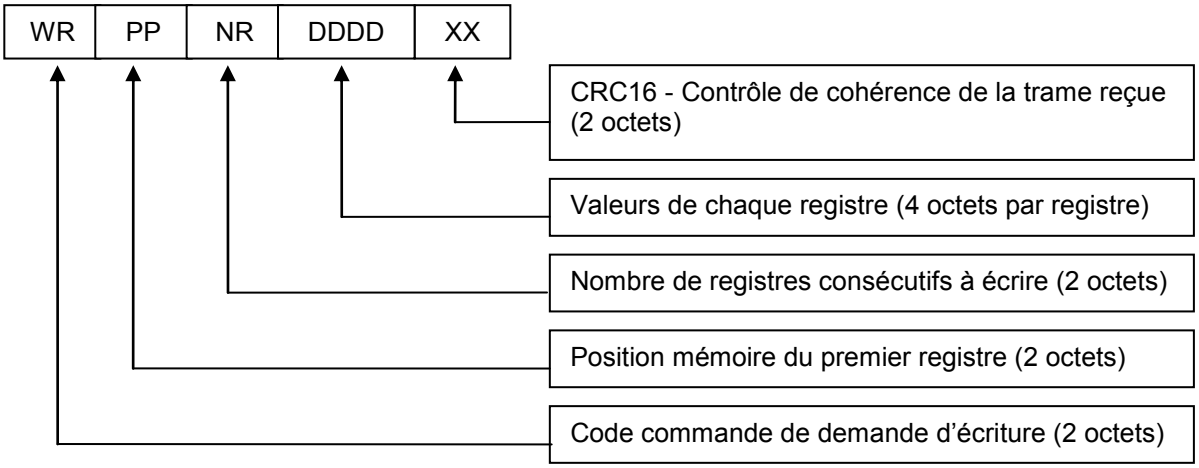
0x01 = Poids le plus faible de la valeur du registre situé à l'adresse 1000
0x00
0x00
0x00 = Poids le plus fort de la valeur du registre situé à l'adresse 1000

0xA8 = Poids le plus faible de la valeur du registre situé à l'adresse 1001
0x61
0x00
0x00 = Poids le plus fort de la valeur du registre situé à l'adresse 1001

0x75 = Poids faible du CRC16
0x0A = Poids fort du CRC16

Code commande WR - Format de la trame émise par le PC

Demande d'écriture de n registres consécutifs à partir d'une position mémoire



Exemple : Demande d'écriture de 2 registres consécutifs à partir de la position 1000
(Valeur 1 à la position 1000 et 25000 à la position 1001)

Octets émis : 0x57, 0x52, 0xE8, 0x03, 0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0xA8, 0x61, 0x00, 0x00, 0x47, 0x59

0x57 = Code ascii du caractère W
0x52 = Code ascii du caractère R

0xE8 = Poids faible de la première position mémoire
0x03 = Poids fort de la première position mémoire

0x02 = Poids faible du nombre de registre
0x00 = Poids fort du nombre de registre

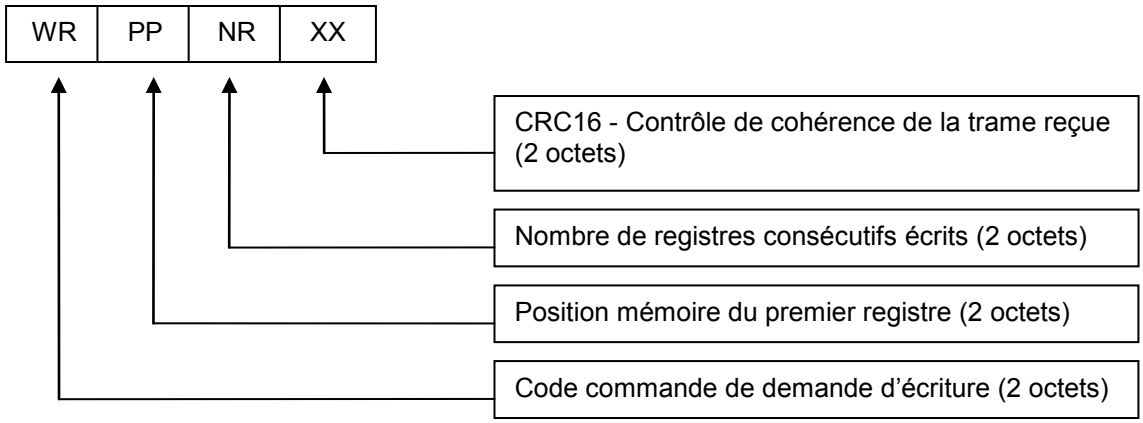
0x01 = Poids le plus faible de la valeur à écrire dans le registre à la position 1000
0x00
0x00
0x00 = Poids le plus fort de la valeur à écrire dans le registre à la position 1000

0xA8 = Poids le plus faible de la valeur à écrire dans le registre à la position 1001
0x61
0x00
0x00 = Poids le plus fort de la valeur à écrire dans le registre à la position 1001

0x47 = Poids faible du CRC16
0x59 = Poids fort du CRC16

Code commande WR - Format de la trame renvoyée par la main

Réponse à une demande d'écriture de n registres consécutifs à partir d'une position mémoire



Exemple : Réponse à une demande d'écriture de 2 registres consécutifs à partir de la position 1000

Octets reçus : 0x57, 0x52, 0xE8, 0x03, 0x02, 0x00, 0x70, 0xF0

0x57 = Code ascii du caractère W
0x52 = Code ascii du caractère R

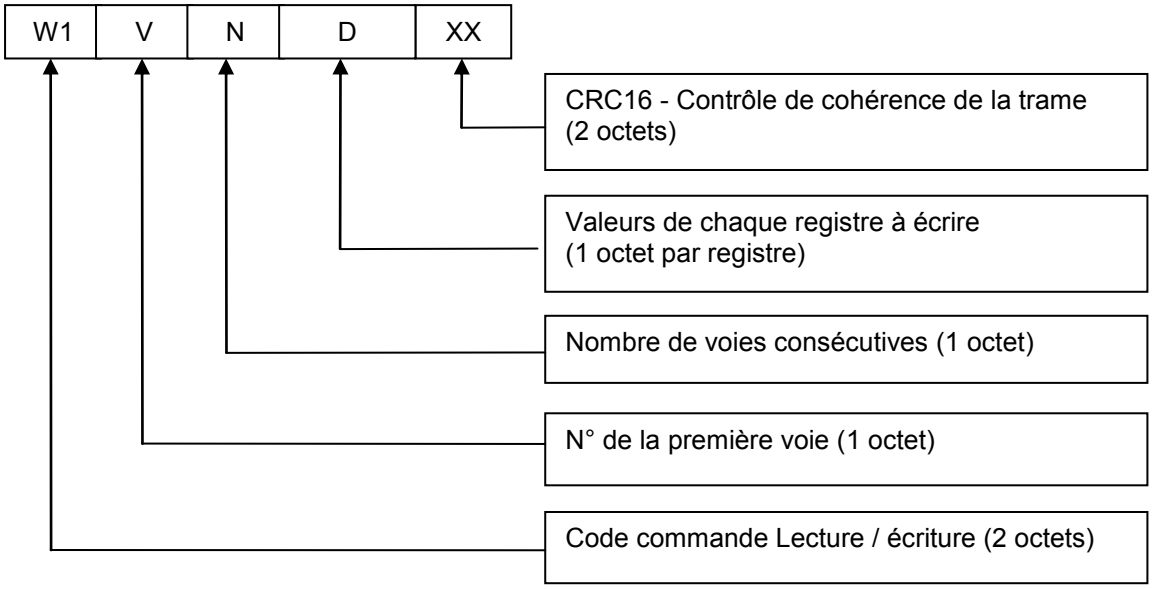
0xE8 = Poids faible de la première position mémoire
0x03 = Poids fort de la première position mémoire

0x02 = Poids faible du nombre de registres
0x00 = Poids fort du nombre de registres

0x70 = Poids faible du CRC16
0xF0 = Poids fort du CRC16

Code commande W1 - Format de la trame émise par le PC

Demande d'écriture du registre "MODE_CMD_MOTEUR" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Demande d'écriture des registres "MODE_CMD_MOTEUR" des voies 1 à 3 et lecture des registres " POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x31, 0x01, 0x03, 0x02, 0x01, 0x00, 0xA4, 0x30

0x57 = Code ascii hexadécimal du caractère W
0x31 = Code ascii hexadécimal du caractère 1

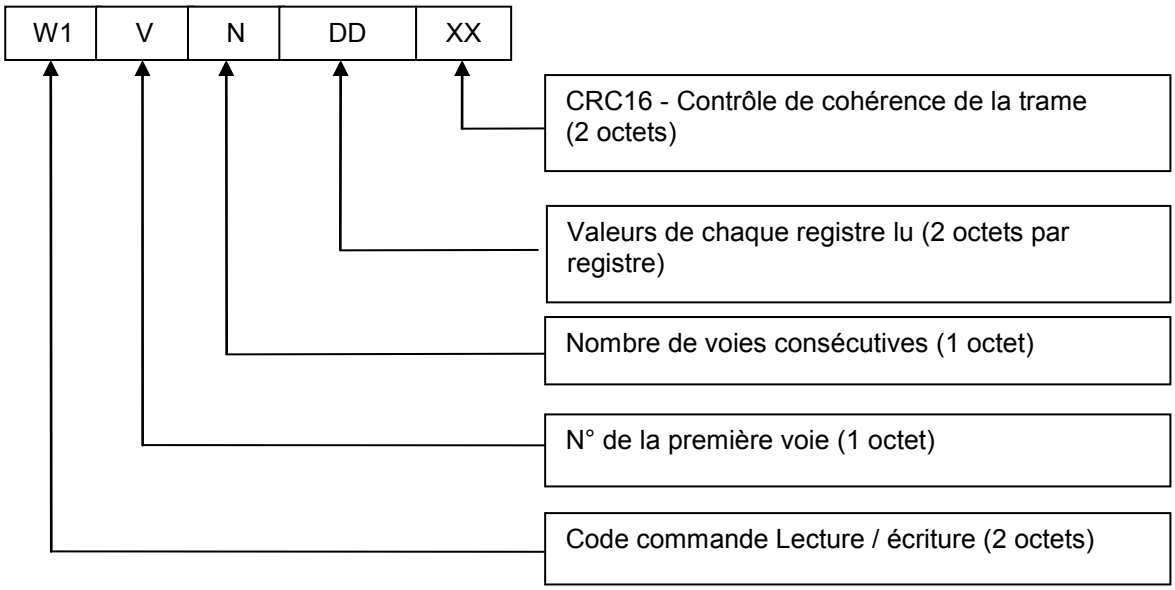
0x01 = N° de la première voie
0x03 = Nombre de voies

0x02 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 1
0x01 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 2
0x00 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 3

0xA4 = Poids faible du CRC16
0x30 = Poids fort du CRC16

Code commande W1 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "MODE_CMD_MOTEUR" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres "MODE_CMD_MOTEUR" des voies 1 à 3 et lecture des registres "POSITION_CODEUR" des voies 1 à 3

Octets émis 0x57, 0x31, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0xA3, 0x1D

0x57 = Code ascii du caractère W
0x31 = Code ascii du caractère 1

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "POSITION_CODEUR" de la voie 1
0x03 = Poids fort du registre "POSITION_CODEUR" de la voie 1

0x00 = Poids faible du registre "POSITION_CODEUR" de la voie 2
0x00 = Poids fort du registre "POSITION_CODEUR" de la voie 2

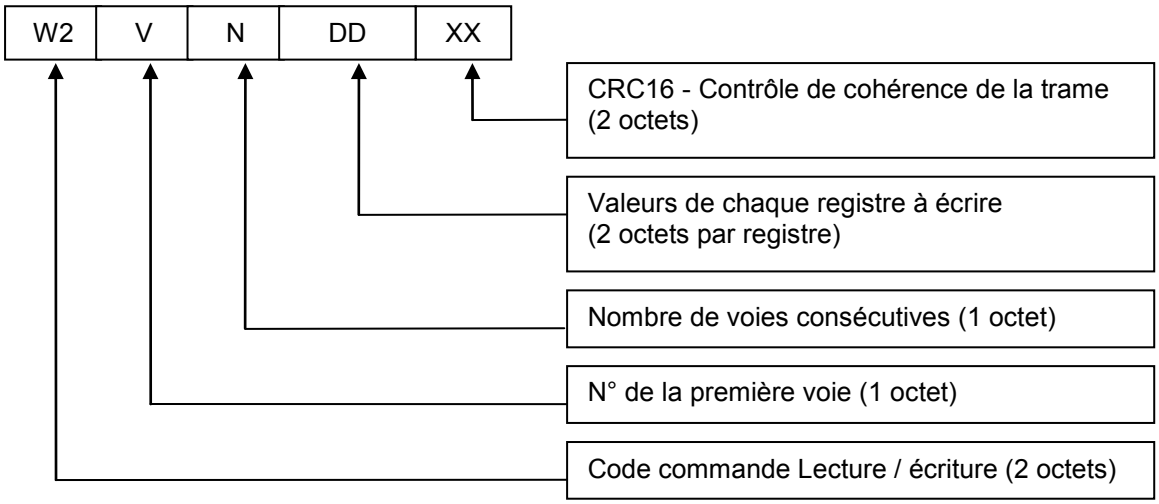
0x20 = Poids faible du registre "POSITION_CODEUR" de la voie 3
0x4E = Poids fort du registre "POSITION_CODEUR" de la voie 3

0xA3 = Poids faible du CRC16
0x1D = Poids fort du CRC16

Attention : Pour ce type commande, la valeur de "POSITION_CODEUR" est retournée sur 2 octets et doit être considérée comme positive lorsqu'elle est inférieure à 60535 et négative lorsqu'elle est supérieure à cette valeur

Code commande W2 - Format de la trame émise par le PC

Demande d'écriture du registre "CONSIGNE_TENSION_POSITION" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Demande d'écriture des registres "CONSIGNE_TENSION_POSITION" des voies 1 à 3 et lecture des registres " POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x32, 0x01, 0x03, 0x20, 0x03, 0x00, 0x00, 0xF4, 0x01, 0xB9, 0x51

0x57 = Code ascii hexadécimal du caractère W
0x32 = Code ascii hexadécimal du caractère 2

0x01 = N° de la première voie
0x03 = Nombre de voies

0x20 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 1
0x03 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 1

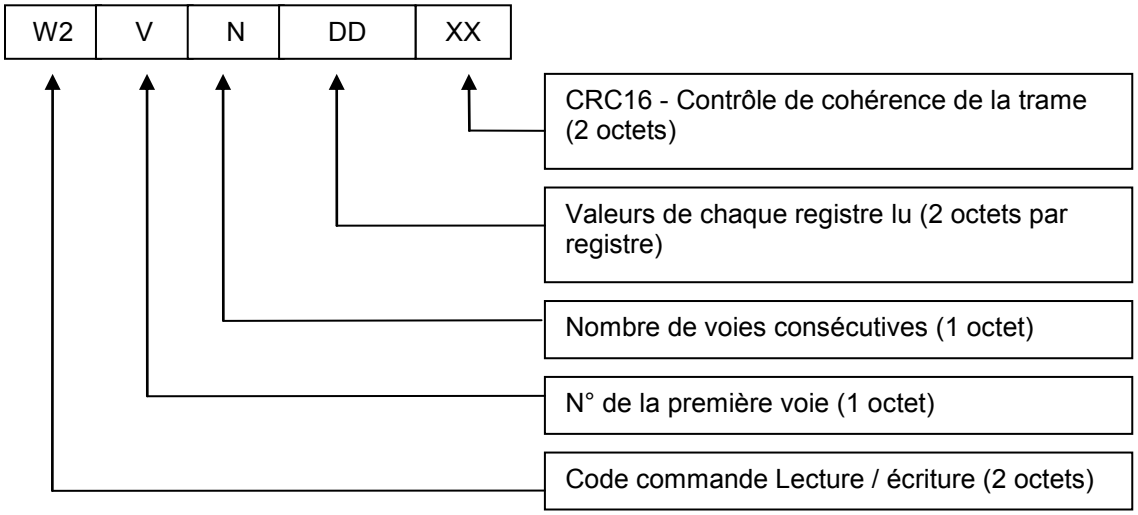
0x00 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 2
0x00 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 2

0xF4 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 3
0x01 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 3

0xB9 = Poids faible du CRC16
0x51 = Poids fort du CRC16

Code commande W2 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "CONSIGNE_TENSION_POSITION" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres " CONSIGNE_TENSION_POSITION " des voies 1 à 3 et lecture des registres "POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x32, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0xB7, 0xED,

0x57 = Code ascii du caractère W
0x32 = Code ascii du caractère 2

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "POSITION_CODEUR" de la voie 1
0x03 = Poids fort du registre "POSITION_CODEUR" de la voie 1

0x00 = Poids faible du registre "POSITION_CODEUR" de la voie 2
0x00 = Poids fort du registre "POSITION_CODEUR" de la voie 2

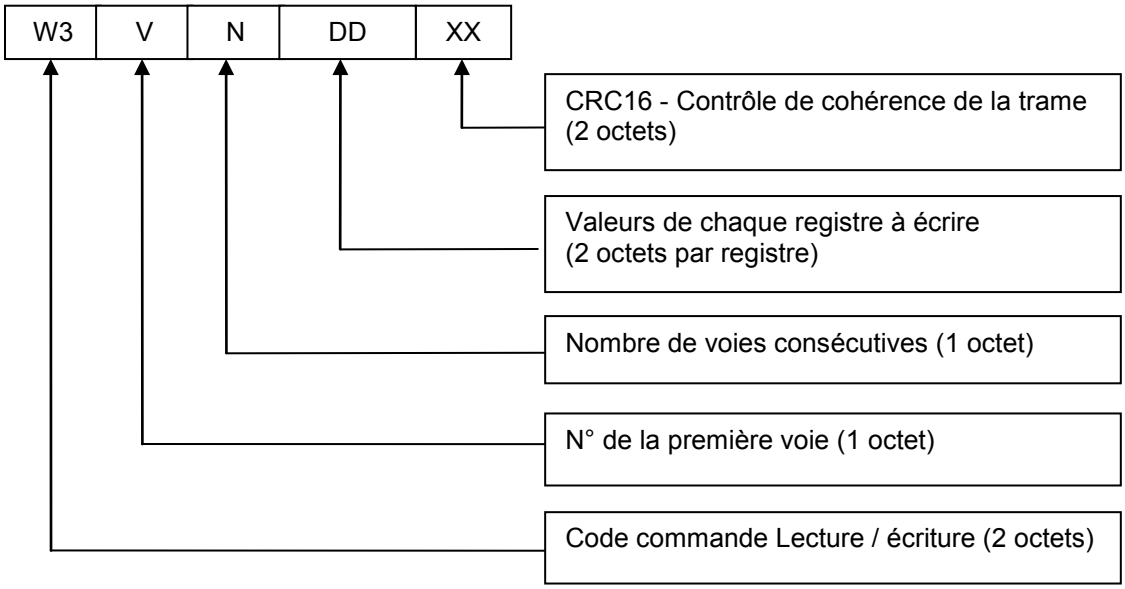
0x20 = Poids faible du registre "POSITION_CODEUR" de la voie 3
0x4E = Poids fort du registre "POSITION_CODEUR" de la voie 3

0xB7 = Poids faible du CRC16
0xED = Poids fort du CRC16

Attention : Pour ce type commande, la valeur de "POSITION_CODEUR" est retournée sur 2 octets et doit être considérée comme positive lorsqu'elle est inférieure à 60535 et négative lorsqu'elle est supérieure à cette valeur

Code commande W3 - Format de la trame émise par le PC

Demande d'écriture du registre "LIMITE_COURANT" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Demande d'écriture des registres " LIMITE_COURANT" des voies 1 à 3 et lecture des registres " POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x33, 0x01, 0x03, 0x30, 0x75, 0x00, 0x00, 0x20, 0x4E, 0x61, 0x6E

0x57 = Code ascii du caractère W
0x33 = Code ascii du caractère 3

0x01 = N° de la première voie
0x03 = Nombre de voies

0x30 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 1
0x75 = Poids fort de la valeur à écrire dans le registre "LIMITE_COURANT " de la voie 1

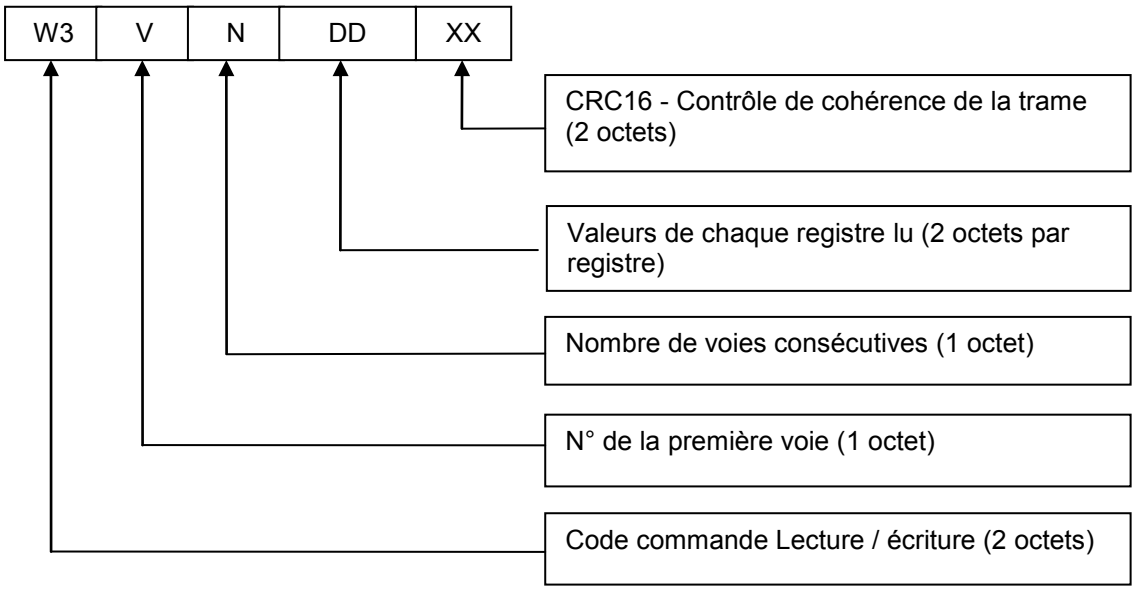
0x00 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT " de la voie 2
0x00 = Poids fort de la valeur à écrire dans le registre " LIMITE_COURANT" de la voie 2

0x20 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 3
0x4E = Poids fort de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 3

0x61 = Poids faible du CRC16
0x6E = Poids fort du CRC16

Code commande W3 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "LIMITE_COURANT" de n voies consécutives et lecture du registre "POSITION_CODEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres "LIMITE_COURANT" des voies 1 à 3 et lecture des registres "POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x33, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0xBA, 0x7D

0x57 = Code ascii du caractère W
0x33 = Code ascii du caractère 3

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "POSITION_CODEUR" de la voie 1
0x03 = Poids fort du registre "POSITION_CODEUR" de la voie 1

0x00 = Poids faible du registre "POSITION_CODEUR" de la voie 2
0x00 = Poids fort du registre "POSITION_CODEUR" de la voie 2

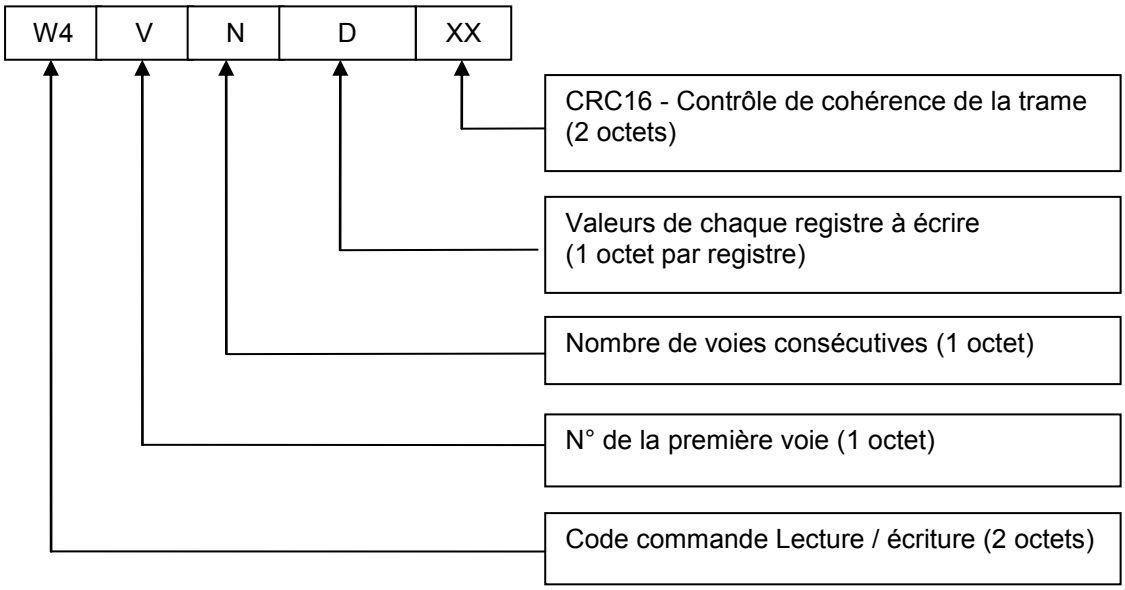
0x20 = Poids faible du registre "POSITION_CODEUR" de la voie 3
0x4E = Poids fort du registre "POSITION_CODEUR" de la voie 3

0xBA = Poids faible du CRC16
0x7D = Poids fort du CRC16

Attention : Pour ce type commande, la valeur de "POSITION_CODEUR" est retournée sur 2 octets et doit être considérée comme positive lorsqu'elle est inférieure à 60535 et négative lorsqu'elle est supérieure à cette valeur

Code commande W4 - Format de la trame émise par le PC

Demande d'écriture du registre "MODE_CMD_MOTEUR" de n voies consécutives et lecture du registre "VITESSE_MOTEUR" des n voies



Exemple : Demande d'écriture des registres "MODE_CMD_MOTEUR" des voies 1 à 3 et lecture des registres "VITESSE_MOTEUR" des voies 1 à 3

Octets émis : 0x57, 0x34, 0x01, 0x03, 0x02, 0x01, 0x00, 0xA4, 0x65

0x57 = Code ascii du caractère W
0x34 = Code ascii du caractère 4

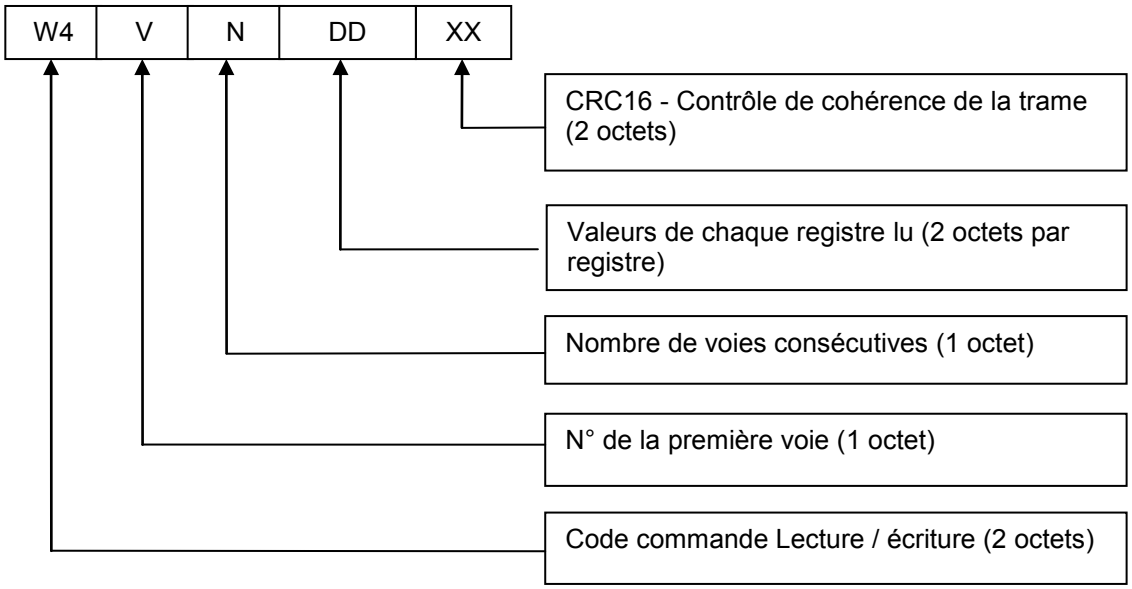
0x01 = N° de la première voie
0x03 = Nombre de voies

0x02 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 1
0x01 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 2
0x00 = Valeur à écrire dans le registre "MODE_CMD_MOTEUR" de la voie 3

0xA4 = Poids faible du CRC16
0x65 = Poids fort du CRC16

Code commande W4 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "MODE_CMD_MOTEUR" de n voies consécutives et lecture du registre "VITESSE_MOTEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres "MODE_CMD_MOTEUR" des voies 1 à 3 et lecture des registres "VITESSE_MOTEUR" des voies 1 à 3

Octets émis : 0x57, 0x34, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0x9C, 0x4D

0x57 = Code ascii du caractère W
0x34 = Code ascii du caractère 4

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "VITESSE_MOTEUR" de la voie 1
0x03 = Poids fort du registre "VITESSE_MOTEUR" de la voie 1

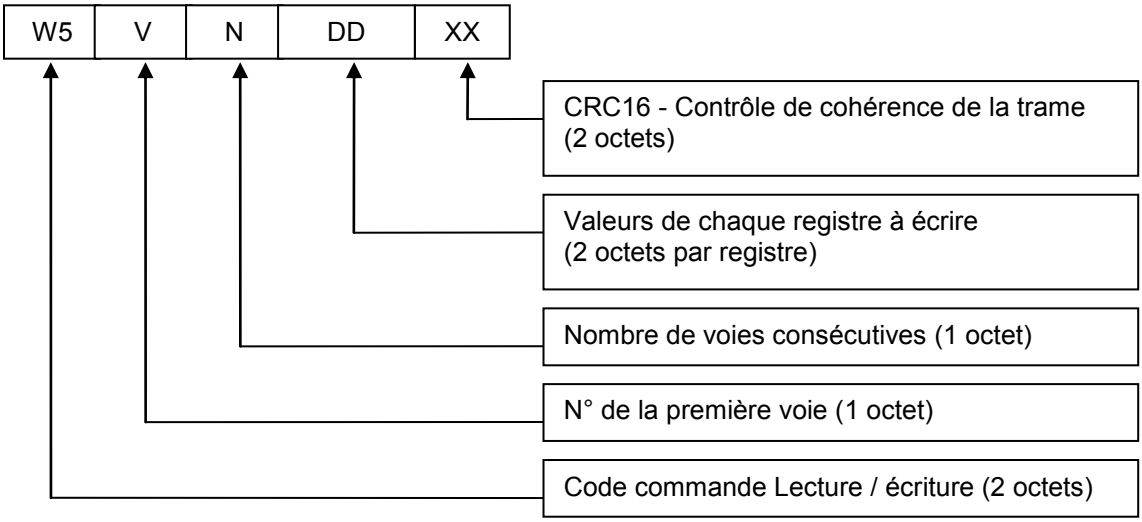
0x00 = Poids faible du registre "VITESSE_MOTEUR" de la voie 2
0x00 = Poids fort du registre "VITESSE_MOTEUR" de la voie 2

0x20 = Poids faible du registre "VITESSE_MOTEUR" de la voie 3
0x4E = Poids fort du registre "VITESSE_MOTEUR" de la voie 3

0x9C = Poids faible du CRC16
0x4D = Poids fort du CRC16

Code commande W5 - Format de la trame émise par le PC

Demande d'écriture du registre "CONSIGNE_TENSION_POSITION" de n voies consécutives et lecture du registre "VITESSE_MOTEUR" des n voies



Exemple : Demande d'écriture des registres "CONSIGNE_TENSION_POSITION" des voies 1 à 3 et lecture des registres " POSITION_CODEUR" des voies 1 à 3

Octets émis : 0x57, 0x35, 0x01, 0x03, 0x20, 0x03, 0x00, 0x00, 0xF4, 0x01, 0x9F, 0x61

0x57 = Code ascii du caractère W
0x35 = Code ascii du caractère 5

0x01 = N° de la première voie
0x03 = Nombre de voies

0x20 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 1
0x03 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 1

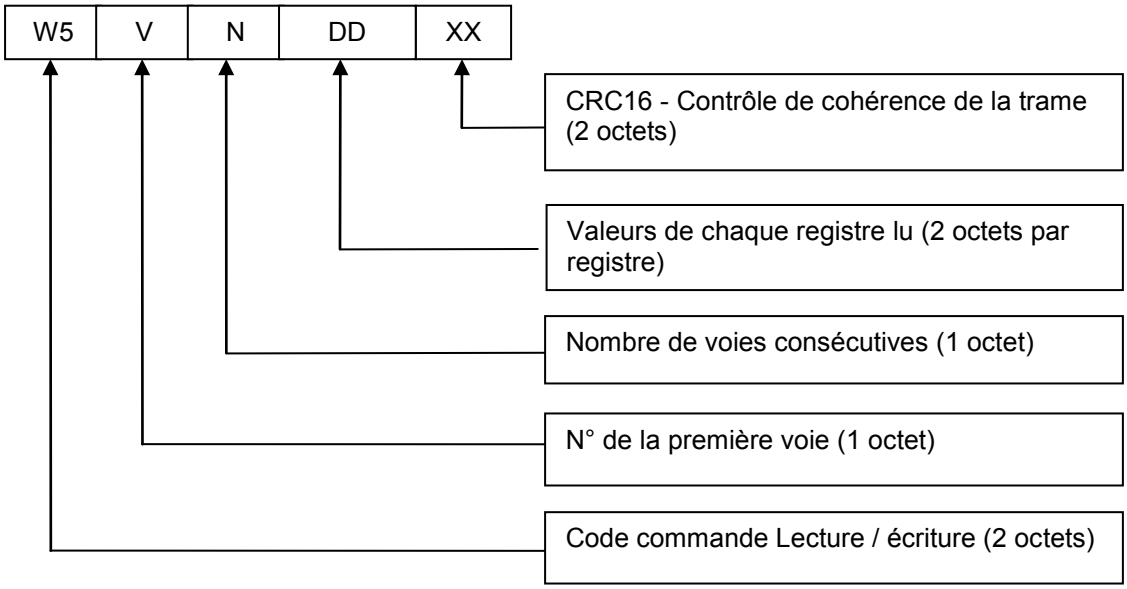
0x00 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 2
0x00 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 2

0xF4 = Poids faible de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 3
0x01 = Poids fort de la valeur à écrire dans le registre " CONSIGNE_TENSION_POSITION" de la voie 3

0x9F = Poids faible du CRC16
0x61 = Poids fort du CRC16

Code commande W5 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "CONSIGNE_TENSION_POSITION" de n voies consécutives et lecture du registre "VITESSE_MOTEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres "CONSIGNE_TENSION_POSITION" des voies 1 à 3 et lecture des registres "VITESSE_MOTEUR" des voies 1 à 3

Octets émis : 0x57, 0x35, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0x91, 0xDD

0x57 = Code ascii du caractère W
0x35 = Code ascii du caractère 5

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "VITESSE_MOTEUR" de la voie 1
0x03 = Poids fort du registre "VITESSE_MOTEUR" de la voie 1

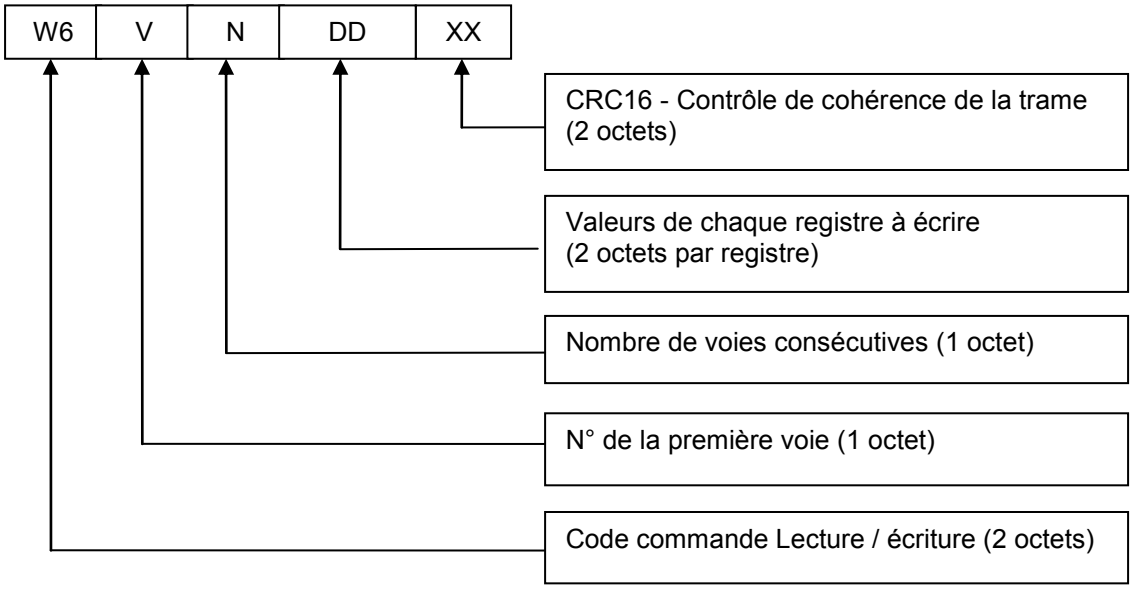
0x00 = Poids faible du registre "VITESSE_MOTEUR" de la voie 2
0x00 = Poids fort du registre "VITESSE_MOTEUR" de la voie 2

0x20 = Poids faible du registre "VITESSE_MOTEUR" de la voie 3
0x4E = Poids fort du registre "VITESSE_MOTEUR" de la voie 3

0x91 = Poids faible du CRC16
0xDD = Poids fort du CRC16

Code commande W6 - Format de la trame émise par le PC

Demande d'écriture du registre "LIMITE_COURANT" de n voies consécutives et lecture du registre "VITESSE_MOTEUR" des n voies



Exemple : Demande d'écriture des registres " LIMITE_COURANT" des voies 1 à 3 et lecture des registres "VITESSE_MOTEUR" des voies 1 à 3

Octets émis : 0x57, 0x36, 0x01, 0x03, 0x30, 0x75, 0x00, 0x00, 0x20, 0x4E, 0x5E, 0x3E,

0x57 = Code ascii du caractère W
0x36 = Code ascii du caractère 6

0x01 = N° de la première voie
0x03 = Nombre de voies

0x30 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 1
0x75 = Poids fort de la valeur à écrire dans le registre "LIMITE_COURANT " de la voie 1

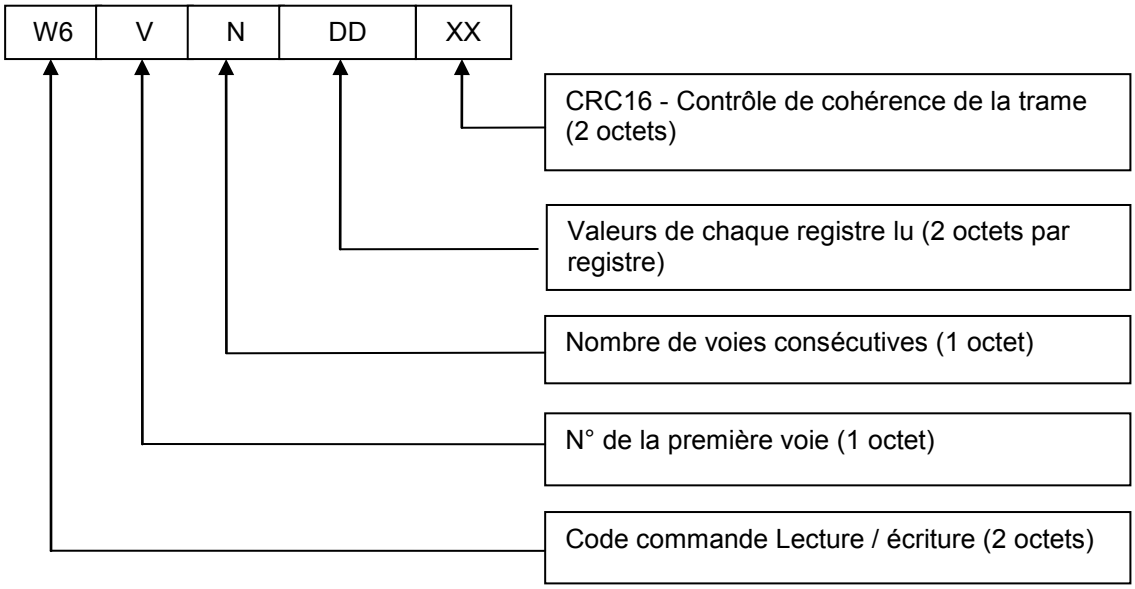
0x00 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT " de la voie 2
0x00 = Poids fort de la valeur à écrire dans le registre " LIMITE_COURANT" de la voie 2

0x20 = Poids faible de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 3
0x4E = Poids fort de la valeur à écrire dans le registre "LIMITE_COURANT" de la voie 3

0x5E = Poids faible du CRC16
0x3E = Poids fort du CRC16

Code commande W6 - Format de la trame renvoyée par la main

Réponse à une demande d'écriture du registre "LIMITE_COURANT" de n voies consécutives et lecture du registre " VITESSE_MOTEUR" des n voies



Exemple : Réponse à une demande d'écriture des registres "LIMITE_COURANT" des voies 1 à 3 et lecture des registres "VITESSE_MOTEUR" des voies 1 à 3

Octets émis : 0x57, 0x36, 0x01, 0x03, 0xE8, 0x03, 0x00, 0x00, 0x20, 0x4E, 0x85, 0x2D

0x57 = Code ascii du caractère W
0x36 = Code ascii du caractère 6

0x01 = N° de la première voie
0x03 = Nombre de voies

0xE8 = Poids faible du registre "VITESSE_MOTEUR" de la voie 1
0x03 = Poids fort du registre "VITESSE_MOTEUR" de la voie 1

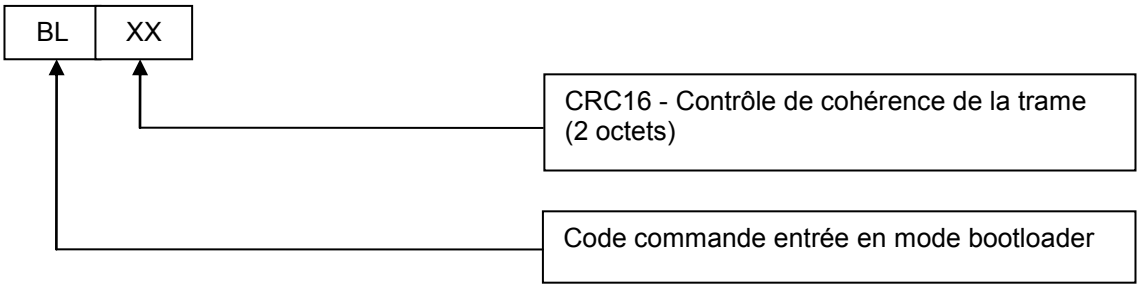
0x00 = Poids faible du registre "VITESSE_MOTEUR" de la voie 2
0x00 = Poids fort du registre "VITESSE_MOTEUR" de la voie 2

0x20 = Poids faible du registre "VITESSE_MOTEUR" de la voie 3
0x4E = Poids fort du registre "VITESSE_MOTEUR" de la voie 3

0x85 = Poids faible du CRC16
0x2D = Poids fort du CRC16

Code commande BL - Format de la trame envoyée par le PC

Demande d'entrée en mode de chargement d'un nouveau programme



Exemple : Demande d'entrée en mode de chargement d'un nouveau programme

Octets émis : 0x42, 0x4C, 0x30, 0xE5

0x42 = Code ascii du caractère B
0x4C = Code ascii du caractère L

0x30 = Poids faible du CRC16
0xE5 = Poids fort du CRC16

- Aucune réponse n'est retournée par la main à réception de ce code de commande.
- Le microcontrôleur est directement configuré de manière à recevoir un nouveau programme.
- Le chargement du programme se fait à l'aide de l'utilitaire "Flash Magic" disponible sur Internet
- Le PC doit impérativement libérer le port "COM" de manière à ce qu'il soit disponible pour "Flash Magic"

2.4. Calcul de CRC16 (méthode 1)

Cette méthode utilise peu de place en mémoire mais nécessite un temps d'exécution plus important que la méthode 2.

```
UVAR_16 APP_Crc16_Calc (UVAR_8 *buf, UVAR_16 len)
{
    UVAR_8 byte_index, bit_index;
    UVAR_16 crc16 = 0xFFFF;
    for(byte_index = 0; byte_index < len; byte_index ++)
    {
        crc16 ^= *(buf + byte_index);
        for (bit_index = 0 ; bit_index < 8 ; bit_index ++ )
        {
            if(crc16 & 0x0001)  { crc16 >>= 1; crc16 ^= 0xA001; }
            else crc16 >>= 1;
        }
    }
    return crc16;
}
```

2.5. Calcul de CRC16 (méthode 2)

Cette méthode utilise plus de place en mémoire mais nécessite un temps d'exécution moins important que la méthode 1.

```
/* Table of CRC values for high–order byte */
const UVAR_8 TABLE_CRC_HI[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81,0x40 } ;
```

```
/* Table of CRC values for low–order byte */
const UVAR_8 TABLE_CRC_LO[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,0x40 };



UVAR_16 APP_Crc16_Calc( UVAR_8 *buffer, UVAR_32 len)
{
   UVAR_8 crc_hi = 0xFF ; /* high byte of CRC initialized */
   UVAR_8 crc_lo = 0xFF ; /* low byte of CRC initialized */
   UVAR_8 index = 0; /* will index into CRC lookup table */
   UVAR_16 crc;
   while (len--)
   {
        index = crc_lo ^ *buffer++ ;
        crc_lo = crc_hi ^ TABLE_CRC_HI[index] ;
        crc_hi = TABLE_CRC_LO[index] ;
    }
    crc = crc_hi;
    crc = (crc * 256) + crc_lo;
    return crc ;
}
```

3. Registres internes

3.1. Introduction

La main robotisée est commandée depuis le PC. Elle peut fonctionner suivant 2 modes distincts. Le choix du mode de fonctionnement s'effectue par l'écriture de valeurs spécifiques dans un registre nommé MODE_CMD_MOTEUR. Le mode de fonctionnement peut être configuré indépendamment d'une voie à l'autre.

3.2. Mode consigne de position

Dans ce mode, le PC écrit les consignes de positions dans le registre nommé "CONSIGNE_TENSION_POSITION" de chaque voie. Le microcontrôleur va alors placer et maintenir chaque moteur à la position indiquée à l'aide de l'algorithme suivant :

- Lecture de la position actuelle "POSITION_CODEUR"

- Calcul de l'écart actuel : (Consigne de position – Position actuelle)
  ECART_POSITION = CONSIGNE_TENSION_POSITION – POSITION_CODEUR

- Calcul du cumul des écarts : (Cumul des Ecarts + Ecart Actuel)
  SOMME_ECARTS = SOMME_ECARTS + ECART_POSITION

- Calcul de la variation de l'écart : (Ecart Actuel – Ecart Précédent)
  DELTA_ECARTS = ECART_POSITION - MEMO_ECARTS

- Calcul de la tension à appliquer au moteur
  (Coéf_p * Ecart Actuel) + (Coéf_I * Cumul des Ecarts) + (Coéf_d * Variation de l'écart)

  SORTIE_PWM = (ECART_POSITION * (COEF_P / 1000)) +
  (POS_ERROR_SUM * (COEF_I / 10000))+
  (DELTA_ECARTS * (COEF_D / 100))

- Appliquer la nouvelle tension au moteur.

Les opérations ci-dessus sont répétées à raison d'une fois par milliseconde pour chaque voie et à 166,6 microsecondes d'intervalle entre 2 voies consécutives.

Remarques :

La position d'origine est fixée à 0 lorsque le doigt est ouvert au maximum et augmente lorsque le doigt se ferme.
La tension effectivement appliquée au moteur est limitée en négatif à la valeur du paramètre MIN_SORTIE_PWM et en positif à celle de MAX_SORTIE_PWM. La valeur du registre SORTIE_PWM n'est pas modifiée lorsque ces limites sont appliquées.

La valeur de SOMME_ECARTS est limitée en négatif à la valeur du paramètre MIN_SOMME_ECARTS et en positif à celle de MAX_SOMME_ECARTS

3.3. Mode consigne de tension

Dans ce mode, le PC écrit les consignes de tension dans le registre nommé "CONSIGNE_TENSION_POSITION" de chaque voie.

Le microcontrôleur applique directement ces tensions aux moteurs. Une tension négative ouvre le doigt alors qu'une tension positive le ferme.

Aucun contrôle de position minimale ou maximale n'est effectué dans ce mode

3.4. Mémorisation des registres

Les registres de paramètres sont enregistrés en mémoire permanente de manière à ne pas être perdus lors de la mise hors tension de la main.

Les registres de consignes susceptibles d'être modifiés à des cadences élevées ne sont pas enregistrés en mémoire.

Ainsi à la mise sous tension :
- les registres MODE_CMD_MOTEUR et CONSIGNE_TENSION_POSITION sont initialisés à 0.
- Le registre LIMITE_COURANT est initialisé avec la valeur de LIMITE_COURANT_DEFAUT.

Les registres d'états ne sont pas non plus enregistrés en mémoire.

Pour info :
Les mémoires utilisées pour la main autorisent un nombre de cycles d'écriture relativement élevé (entre 1 et 10 millions selon les données constructeur). Toutefois, ce nombre de cycles sera atteint rapidement si la cadence d'écriture est élevée.

Par exemple : A raison d'une écriture toute les 10 ms, un million de cycles est atteint en un peu moins de 28 heures.

3.5. Registres de commandes

Les registres de commande sont indépendants du numéro de voie. Ils sont au nombre de 2 et sont situés aux positions mémoire 100 et 200. Ils permettent pour l'un de démarrer la procédure de recherche de la position initiale et pour l'autre de charger le jeu de paramètres par défaut.

POSITION MEMOIRE	NOM	TYPE	ACCES	MODE	EEPROM
100	INIT_POSITION	UVAR_32	R/W	TOUS	NON

Procédure de recherche des positions initiales :

En écriture :
- La valeur 1 lance la procédure
- La valeur 0 stoppe la procédure si elle est en cours

En lecture :
- La valeur 0 indique que la procédure n'a pas été faite ou qu'elle est en cours
- La valeur 1 indique que la procédure a été faite

POSITION MEMOIRE	NOM	TYPE	ACCES	MODE	EEPROM
200	INIT_DEFAUT_PARAM	UVAR_32	W	TOUS	NON

Initialisation des paramètres aux valeurs part défaut :

Ecriture seule:
- La valeur 1 : Initialisation des paramètres et configuration en main droite
- La valeur 2 : Initialisation des paramètres et configuration en main gauche

3.6. Registres de consignes et paramétrage

N°	NOM	TYPE	ACCES	MODE	EEPROM
0	**MODE_CMD_MOTEUR**	**UVAR_32**	**R/W**	**TOUS**	**NON**

Ce registre permet de sélectionner le mode de fonctionnement de la voie. La valeur qu'il contient indique au microcontrôleur la manière de commander la voie.
Trois modes de fonctionnement sont possibles :

0 = Mode STOP. Aucune commande n'est appliquée au moteur

1 = Mode consigne de POSITION.
Le registre "CONSIGNE_TENSION_POSITION" contient la consigne de position

2 = Mode consigne de TENSION.
Le registre "CONSIGNE_TENSION_POSITION" contient la consigne de tension

Avant de passer du mode position au mode tension (ou inversement) il est préférable de passer par le mode stop afin de modifier la consigne en conséquence.

N°	NOM	TYPE	ACCES	MODE	EEPROM
1	**CONSIGNE_TENSION_POSITION**	**VAR_32**	**R/W**	**TOUS**	**NON**

Ce registre contient la valeur de consigne de la voie. Le registre "MODE_CMD_MOTEUR" détermine le type de cette consigne

- "MODE_CMD_MOTEUR" = 1 (MODE CONSIGNE DE POSITION)

Le registre "CONSIGNE_TENSION_POSITION" contient la consigne de position à atteindre et maintenir.
La consigne est exprimée en points (codeur incrémental)
Valeur minimale = CONSIGNE_POSITION_MIN
Valeur maximale = CONSIGNE_POSITION_MAX

- "MODE_CMD_MOTEUR" = 2 (MODE CONSIGNE DE TENSION)

Le registre "CONSIGNE_TENSION_POSITION" contient la consigne de tension à appliquer
Une tension négative ouvre le doigt alors qu'une tension positive le ferme
La tension est exprimée en centièmes de volt. (Exemple : 825 correspond à 8,25V)
La valeur minimale est de -1150 (-11,5V)
La valeur maximale est de 1150 (11,5V)

N°	NOM	TYPE	ACCES	MODE	EEPROM
2	LIMITE_COURANT	UVAR_32	R/W	TOUS	NON

Ce registre contient la consigne de limite de courant de la voie.

La valeur contenue dans ce registre représente le courant efficace fourni au moteur. Elle est exprimée en dixièmes de milliampère. (Exemple : 562 correspond à 56,2mA)

- Valeur minimale = 0
- Valeur maximale = 750 (75,0 mA)

A la mise sous tension, ce registre est initialisé avec la valeur du registre LIMITE_COURANT_DEFAUT.

N°	NOM	TYPE	ACCES	MODE	EEPROM
3	LIMITE_COURANT_DEFAUT	UVAR_32	R/W	TOUS	OUI

Ce registre est utilisé pour initialiser le registre LIMITE_COURANT à la mise sous tension de la main.
- Valeur minimale = 0
- Valeur maximale = 750 (75,0 mA)

N°	NOM	TYPE	ACCES	MODE	EEPROM
4	NON UTILISE	UVAR_32	R/W	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
5	NON UTILISE	UVAR_32	R/W	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
6	DELAI_MODE_PI	UVAR_32	R/W	POSITION	OUI

Consigne de temps pour passage en mode PI
Ce paramètre permet d'améliorer la stabilité lorsque le capteur de position est de type analogique.

Sur la main robotisée il ne présente aucun avantage car le capteur de position est numérique.
Configuré à la valeur 100, il n'a quasiment aucun effet sur le fonctionnement de la main robotisée.

N°	NOM	TYPE	ACCES	MODE	EEPROM
7	DELTA_MODE_PI	VAR_32	R/W	POSITION	OUI

Consigne d'écart pour passage en mode PI
Ce paramètre permet d'améliorer la stabilité lorsque le capteur de position est de type analogique.

Sur la main robotisée il ne présente aucun avantage car le capteur de position est numérique.
Configuré à la valeur 2, il n'a quasiment aucun effet sur le fonctionnement de la main robotisée.

N°	NOM	TYPE	ACCES	MODE	EEPROM
8	COEF_P	UVAR_32	R/W	POSITION	OUI

Coefficient utilisé pour le calcul de la part de commande proportionnelle à l'écart entre la consigne et la position.

CALCUL_P = ECART_POSITION * (COEF_P / 1000)

N°	NOM	TYPE	ACCES	MODE	EEPROM
9	COEF_I	UVAR_32	R/W	POSITION	OUI

Coefficient utilisé pour le calcul de la part de commande proportionnelle à la somme des écarts entre la consigne et la position.

CALCUL_I = SOMME_ECARTS * (COEF_I / 10 000)

N°	NOM	TYPE	ACCES	MODE	EEPROM
10	COEF_D	UVAR_32	R/W	POSITION	OUI

Coefficient utilisé pour le calcul de la part de commande proportionnelle à la variation de l'écart entre la consigne et la position.

CALCUL_D = DELTA_ECARTS * (COEF_D/ 100)

N°	NOM	TYPE	ACCES	MODE	EEPROM
11	CONSIGNE_POSITION_MIN	VAR_32	R	POSITION	OUI

Limite négative de la consigne de position. (Registre CONSIGNE_TENSION_POSITION en mode consigne de position)

N°	NOM	TYPE	ACCES	MODE	EEPROM
12	CONSIGNE_POSITION_MAX	VAR_32	R	POSITION	OUI

Limite positive de la consigne de position. (Registre CONSIGNE_TENSION_POSITION en mode consigne de position)

N°	NOM	TYPE	ACCES	MODE	EEPROM
13	MIN_SORTIE_PWM	VAR_32	R/W	POSITION	OUI

Limite négative de la tension appliquée au moteur (SORTIE_PWM en mode consigne de position).
La valeur est exprimée en points PWM.
Tension en Volt = MIN_SORTIE_PWM *12 V / 4095

N°	NOM	TYPE	ACCES	MODE	EEPROM
14	MAX_SORTIE_PWM	VAR_32	R/W	POSITION	OUI

Limite positive de la tension appliquée au moteur (SORTIE_PWM en mode consigne de position).
La valeur est exprimée en points PWM.
Tension en Volt = MAX_SORTIE_PWM *12 V / 4095

N°	NOM	TYPE	ACCES	MODE	EEPROM
15	MIN_SOMME_ECARTS	VAR_32	R/W	POSITION	OUI

Limite négative du cumul des écarts (Registre SOMME_ECARTS)

N°	NOM	TYPE	ACCES	MODE	EEPROM
16	MAX_SOMME_ECARTS	VAR_32	R/W	POSITION	OUI

Limite positive du cumul des écarts (Registre SOMME_ECARTS)

N°	NOM	TYPE	ACCES	MODE	EEPROM
17	NON UTILISE	UVAR_32	R/W	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
18	NON UTILISE	UVAR_32	R/W	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
19	DIR_MOTEUR_CODEUR	UVAR_32	R	TOUS	OUI

Le bit 0 de ce registre indique le sens de commande du moteur.
0 = les consignes de tension sont appliquées directement au moteur.
1 = les consignes de tension sont inversées avant d'être appliquées au moteur.

Le bit 1 de ce registre indique le sens de comptage des impulsions codeur.
0 = sens direct
1 = sens inverse

La valeur de ce registre dépend de la configuration matérielle et ne doit pas être modifiée.

N°	NOM	TYPE	ACCES	MODE	EEPROM
20	TEMPS_CALCUL_VITESSE	UVAR_32	R/W	TOUS	OUI

Période de calcul de la vitesse de rotation exprimée en millisecondes. La vitesse de rotation est obtenue par la moyenne des mouvements enregistrés pendant une période définie par ce paramètre.
La plage de réglage autorisée est comprise entre 1 et 200 ms.

N°	NOM	TYPE	ACCES	MODE	EEPROM
21	RESERVE_RW2	UVAR_32	R	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
22	RESERVE_RW3	UVAR_32	R	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
23	RESERVE_RW4	UVAR_32	R	AUCUN	OUI

Non utilisé

N°	NOM	TYPE	ACCES	MODE	EEPROM
24	ID_DROITE_GAUCHE	UVAR_32	R	TOUS	OUI

Identification de la main droite ou gauche
1 = Main droite
2 = Main gauche

N°	NOM	TYPE	ACCES	MODE	EEPROM
25	EMPLACEMENT_DIR_MOT_COD	UVAR_32	R	TOUS	OUI

Ce registre est utilisé en interne. Il contient le numéro de registre de DIR_MOTEUR_CODEUR.

LISTE DES REGISTRES DE CONSIGNES ET DE PARAMETRAGE						
N° de registre	Type de variable	Accès	Mode	Mémorisation	NOM REGISTRE	Description succincte
0	UVAR_32	R/W	Tous	Non	MODE_CMD_MOTEUR	Mode de commande du moteur : (Arrêt, mode position, mode tension)
1	VAR_32	R/W	Tous	Non	CONSIGNE_TENSION_POSITION	Consigne moteur : Position (en mode position) / tension (en mode tension)
2	UVAR_32	R/W	Tous	Non	LIMITE_COURANT	Consigne de courant
3	UVAR_32	R/W	Tous	Oui	LIMITE_COURANT_DEFAUT	Consigne de courant par défaut à la mise sous tension
4	UVAR_32	R/W	Aucun	Oui	NON UTILISE	Non utilisé
5	UVAR_32	R/W	Aucun	Oui	NON UTILISE	Non utilisé
6	UVAR_32	R/W	Position	Oui	DELAI_MODE_PI	Consigne de temps pour passage en mode PI
7	VAR_32	R/W	Position	Oui	DELTA_MODE_PI	Consigne d'erreur pour passage en mode PI
8	UVAR_32	R/W	Position	Oui	COEF_P	Coefficient utilisé pour le calcul de la part de commande proportionnelle à l'écart
9	UVAR_32	R/W	Position	Oui	COEF_I	Coefficient utilisé pour le calcul de la part de commande proportionnelle au cumul des écarts
10	UVAR_32	R/W	Position	Oui	COEF_D	Coefficient utilisé pour le calcul de la part de commande proportionnelle à la différence entre l'écart actuel et l'écart précèdent
11	VAR_32	R	Position	Oui	CONSIGNE_POSITION_MIN	Valeur minimum de la consigne de position
12	VAR_32	R	Position	Oui	CONSIGNE_POSITION_MAX	Valeur maximum de la consigne de position
13	VAR_32	R/W	Position	Oui	MIN_SORTIE_PWM	Butée négative de la commande moteur
14	VAR_32	R/W	Position	Oui	MAX_SORTIE_PWM	Butée positive de la commande moteur
15	VAR_33	R/W	Position	Oui	MIN_SOMME_ECARTS	Butée négative du cumul des erreurs
16	VAR_34	R/W	Position	Oui	MAX_SOMME_ECARTS	Butée positive du cumul des erreurs
17	UVAR_32	R/W	Aucun	Oui	NON UTILISE	Non utilisé
18	UVAR_32	R/W	Aucun	Oui	NON UTILISE	Non utilisé
19	UVAR_32	R	Tous	Oui	DIR_MOTEUR_CODEUR	Sens de commande du moteur et sens de comptage des tops codeur
20	UVAR_32	R/W	Tous	Oui	TEMPS_CALCUL_VITESSE	Période de calcul de la moyenne de vitesse
21	UVAR_32	R	Aucun	Oui	RESERVE_RW2	Non utilisé
22	UVAR_32	R	Aucun	Oui	RESERVE_RW3	Non utilisé
23	UVAR_32	R	Aucun	Oui	RESERVE_RW4	Non utilisé
24	UVAR_32	R	Tous	Oui	ID_DROITE_GAUCHE	Identification main droite ou gauche
25	UVAR_32	R	Tous	Oui	EMPLACEMENT_DIR_MOT_COD	N° du registre contenant DIR_MOTEUR_CODEUR

3.7. Registres d'états

N°	NOM	TYPE	ACCES	MODE	EEPROM
26	POSITION_CODEUR	UVAR_32	R	TOUS	NON

Position du codeur incrémental. Cette valeur est mise à jour une fois par milliseconde et exprimée en points

N°	NOM	TYPE	ACCES	MODE	EEPROM
27	VITESSE_MOTEUR	UVAR_32	R	TOUS	NON

Valeur absolue de la vitesse de rotation de l'arbre moteur (après moto réducteur) exprimée en centièmes de tour par minute. (Vitesse en tr/mn = VITESSE_MOTEUR / 100)

La vitesse de rotation est obtenue par la moyenne des mouvements enregistrés pendant une période définie par le paramètre TEMPS_CALCUL_VITESSE

N°	NOM	TYPE	ACCES	MODE	EEPROM
28	MEMO_POSITION	VAR_32	R	TOUS	NON

Mémorisation de la position pour calcul de la vitesse

N°	NOM	TYPE	ACCES	MODE	EEPROM
29	ECART_POSITION	VAR_32	R	POSITION	NON

Ecart entre la consigne de position et la position

N°	NOM	TYPE	ACCES	MODE	EEPROM
30	POS_ERRORS_SUM	VAR_32	R	POSITION	NON

Cumul des écarts entre la consigne de position et la position

N°	NOM	TYPE	ACCES	MODE	EEPROM
31	DELTA_ECARTS	VAR_32	R	POSITION	NON

Variation de l'écart : (Ecart Actuel – Ecart Précédent)
DELTA_ECARTS = ECART_POSITION – MEMO_ECARTS

N°	NOM	TYPE	ACCES	MODE	EEPROM
32	MEMO_ECARTS	VAR_32	R	POSITION	NON

Enregistrement de l'écart actuel pour calcul de la prochaine variation de l'écart

N°	NOM	TYPE	ACCES	MODE	EEPROM
33	SORTIE_PWM	VAR_32	R	POSITION	NON

Valeur calculée de la tension de commande à appliquer au moteur. La tension effectivement appliquée est limitée en négatif à MIN_SORTIE_PWM et en positif à MAX_SORTIE_PWM.
La valeur est exprimée en points PWM.

Tension en Volt = SORTIE_PWM * 12 V / 4095

N°	NOM	TYPE	ACCES	MODE	EEPROM
34	TEMPO_MODE_PI	UVAR_32	R	POSITION	NON

Temporisation (ms) : Temps restant avant de ne plus tenir compte de CALCUL_D pour le calcul de SORTIE_PWM

N°	NOM	TYPE	ACCES	MODE	EEPROM
35	CALCUL_P	VAR_32	R	POSITION	NON

Part de la commande moteur (SORTIE_PWM) proportionnelle à l'écart entre la consigne et la position

CALCUL_P = ECART_POSITION * (COEF_P / 1000)

N°	NOM	TYPE	ACCES	MODE	EEPROM
36	CALCUL_I	VAR_32	R	POSITION	NON

Part de la commande moteur (SORTIE_PWM) proportionnelle au cumul des écarts entre la consigne et la position

CALCUL_I = POS_ERRORS_SUM * (COEF_I / 10000)

N°	NOM	TYPE	ACCES	MODE	EEPROM
37	CALCUL_D	VAR_32	R	POSITION	NON

Part de la commande moteur (SORTIE_PWM) proportionnelle à la variation de l'écart entre la consigne et la position

CALCUL_D = POS_ERRORS_DELTA * (COEF_D / 100)

N°	NOM	TYPE	ACCES	MODE	EEPROM
38	POSITION_MIN_ATTEINTE	VAR_32	R	POSITION	NON

Position minimale atteinte. Ce registre est réinitialisé à chaque nouvelle consigne de position. Il peut être utilisé pour calculer le dépassement lorsque la nouvelle consigne est inférieure à la position

N°	NOM	TYPE	ACCES	MODE	EEPROM
39	POSITION_MAX_ATTEINTE	VAR_32	R	POSITION	NON

Position maximale atteinte. Ce registre est réinitialisé à chaque nouvelle consigne de position. Il peut être utilisé pour calculer le dépassement lorsque la nouvelle consigne est supérieure à la position

N°	NOM	TYPE	ACCES	MODE	EEPROM
40	ETAPE_INIT_DOIGT	UVAR_32	R	AUCUN	NON

N° de l'étape en cours dans la procédure d'initialisation. L'initialisation du doigt est complète lorsque cette valeur est égale à 6

N°	NOM	TYPE	ACCES	MODE	EEPROM
41	VERSION	UVAR_32	R	AUCUN	NON

Version logicielle du microcontrôleur. Ce registre contient aussi l'information main droite ou gauche.

Bits 7 à 0 – Indice de révision
Bits 15 à 8 – Numéro de version
Bits 31 à 30 – information main
01 Main droite
02 Main gauche

LISTE DES REGISTRES D'ETATS

N° de registre	Type de variable	Accès	Mode	Mémorisation en EEPROM	NOM REGISTRE	Description succincte
26	VAR_32	R	Tous	Non	POSITION_CODEUR	Position codeur
27	UVAR_32	R	Tous	Non	VITESSE_MOTEUR	Vitesse de l'arbre moteur (après moto réducteur)
28	VAR_32	R	Tous	Non	MEMO_POSITION	Mémorisation de la dernière position (utilisé pour le calcul de vitesse)
29	VAR_32	R	Position	Non	ECART_POSITION	Ecart entre consigne de position et position
30	VAR_32	R	Position	Non	SOMME_ECARTS	Cumul des écarts entre consigne de position et position
31	VAR_32	R	Position	Non	DELTA_ECARTS	Différence entre l'écart actuel et l'écart précèdent
32	VAR_32	R	Position	Non	MEMO_ECARTS	Mémorisation du dernier écart
33	VAR_32	R	Position	Non	SORTIE_PWM	Consigne de tension du moteur avant application de la butée SORTIE_PWM = CALCUL_P + CALCUL_I + CALCUL_D
34	UVAR_32	R	Position	Non	TEMPO_MODE_PI	Temporisation en cours pour passage en mode PI
35	VAR_32	R	Position	Non	CALCUL_P	Part de la commande moteur proportionnelle à l'écart CALCUL_P = ECART_POSITION * (COEF_P / 1000)
36	VAR_32	R	Position	Non	CALCUL_I	Part de la commande moteur proportionnelle au cumul des écarts CALCUL_I = SOMME_ECARTS * (COEF_I / 10000)
37	VAR_32	R	Position	Non	CALCUL_D	Part de commande moteur proportionnelle à la différence entre l'écart actuel et l'écart précèdent. CALCUL_D = DELTA_ECARTS * (COEF_D / 100)
38	VAR_32	R	Position	Non	POSITION_MIN_ATTEINTE	Position minimale atteinte. Réinitialisé à chaque nouvelle consigne de position. Peut être utilisé pour calculer le dépassement de consigne
39	VAR_32	R	Position	Non	POSITION_MAX_ATTEINTE	Position maximale atteinte. Réinitialisé à chaque nouvelle consigne de position. Peut être utilisé pour calculer le dépassement de consigne
40	UVAR_32	R	Tous	Non	ETAPE_INIT_DOIGT	Non utilisé
41	UVAR_32	R	Tous	Non	VERSION	Version logicielle du microcontrôleur et info main droite ou gauche

R : Lecture seule
R/W : Lecture et écriture
VAR_32 : Variable entière signée codée sur 32 bits
UVAR_32 : Variable entière non signée codée sur 32 bits