

**NEXEYA Products Division**

3509, route de Baziège
31670 LABÈGE
France
Tél. : 33 (0)5 61 39 96 18
Fax : 33 (0)5 61 39 17 58
www.midi-ingenierie.com

midi ingénierie

BMAC RS485

Manuel utilisateur



Date : 23.11.12

Référence : bmac_v2_um_fr.pdf

Réf. MI : CMN1890890.docx

Révision : 2

Auteur : C.MARTY

<http://www.midi-ingenierie.com>





Sommaire

1. Présentation du produit	4
1.1. Introduction	4
1.2. Fonctionnalités	4
1.2.1. Architecture	4
1.2.2. Moteurs compatibles	5
1.2.3. Réglage du courant moteur	5
1.2.4. Codeur	6
1.2.5. Boucle ouverte ou autocommutation	6
1.2.6. Mouvements	7
1.2.7. Butées	8
1.2.8. Séquenceur	8
1.2.9. Entrées – Sorties	8
1.3. Précautions d'emploi	9
1.3.1. Règles générales	9
1.3.2. Conditions de stockage	9
1.3.3. Conditions d'utilisation	9
2. Spécifications techniques	10
2.1. Alimentation	10
2.2. Spécifications mécaniques	11
2.2.1. BMAC boîtier	11
2.2.2. BMAC rack	12
2.2.3. BMAC-H	13
3. Câblage et configuration	14
3.1. Brochage des connecteurs	14
3.1.1. BMAC boîtier	14
3.1.2. BMAC rack	15
3.1.3. BMAC-H	16
3.2. Fonctionnalités des entrées/sorties	17
3.3. Spécification des entrées-sorties	18
3.3.1. Entrées logiques opto-isolées	18
3.3.2. Entrées analogiques	18
3.3.3. Sorties logiques opto-isolées	19
3.4. Visualisations	19
3.5. Configuration du port de communication	20
3.5.1. Paramétrage du module	20
3.5.2. Le protocole RS-485	20
4. Commandes	21
4.1. CALL / RETURN	22
4.2. CLOSE_LOOP	22
4.3. HALT	23
4.4. HARD_ENDS	23
4.5. IF	24
4.6. INVERSE_POLARITY	24
4.7. JUMP / JUMP_REL	25
4.8. MODULE_RESET	26
4.9. MOVE_INTERPOL	27
4.10. MOVE_ON	28
4.11. MOVE_SPEED	28
4.12. MOVE_TO	29
4.13. OPEN_SEQ / CLOSE_SEQ	29
4.14. OPTIMIZED_CURRENT	30
4.15. POWER	30
4.16. READ	31
4.17. READ_SEQ	32
4.18. REFERENCE	32
4.19. REQUEST_VERSION	33
4.20. SET_ADDRESS	34
4.21. SET_BAUDRATE	34



4.22.	S_CURVE	35
4.23.	SOFT_ENDS	35
4.24.	START_SEQ	36
4.25.	STEP	36
4.26.	STOP	37
4.27.	SYNCHRO	38
4.28.	WAIT	38
5.	Variables internes	39
5.1.	Généralités	39
5.1.1.	Format décimal	39
5.1.2.	Format hexadécimal	39
5.1.3.	Format binaire	39
5.1.4.	Opposé et complément	39
5.1.5.	Format bit-à-bit	40
5.1.6.	Opérations sur variables	40
5.1.	#ACCEL_TIME, #DECEL_TIME	41
5.2.	#CAPTURE (lecture seule)	42
5.3.	#CURRENT_RATIO	42
5.4.	#CPU_TEMPERATURE (lecture seule)	43
5.5.	#DRIVER_TEMPERATURE (lecture seule)	43
5.6.	#ELECTRICAL_POSITION (lecture seule)	43
5.7.	#ENCODER	44
5.8.	#ERROR	44
5.9.	#HIGH_SPEED	45
5.10.	#INPUT (lecture seule)	45
5.11.	#INPUT_ANALOG (lecture seule)	45
5.12.	#INTERPOL_COUNT (lecture seule)	46
5.13.	#INTERPOL_FIFOSIZE	46
5.14.	#INTERPOL_MODE	47
5.15.	#INTERPOL_TIME	48
5.16.	#LINE_DELAY	48
5.17.	#LINE	49
5.18.	#LOW_SPEED	49
5.19.	#M1 à #M8	50
5.20.	#NEGATIVE_END	50
5.21.	#ON_RESET	50
5.22.	#OUTPUT	51
5.23.	#OUTPUT_CONFIG	51
5.24.	#POSITION	52
5.25.	#POSITIVE_END	52
5.26.	#RATED_CURRENT	53
5.27.	#SPEED, #PROFILE_SPEED (lecture seule)	53
5.28.	#STATUS (lecture seule)	54
5.29.	#SUPPLY_VOLTAGE (lecture seule)	55
5.30.	#TIMER_1 à #TIMER_3	55
5.31.	#V1 à #V32	55
6.	Séquenceur	56
6.1.	Présentation des Fonctionnalités	56
6.2.	Mémorisation des lignes	56
6.3.	Exécution du séquenceur	56
6.4.	Exemples de séquences	57
6.4.1.	Exemple 1	57
6.4.2.	Exemple 2	57
6.4.3.	Exemple 3	57
6.4.4.	Exemple 4	57
7.	ANNEXES	58
7.1.	Résumé des commandes	58
7.2.	Résumé des variables	59
7.3.	Documents associés	59

1. Présentation du produit

1.1. Introduction

Le module BMAC est un indexeur numérique avec amplificateur micropas et contrôleur DSP intégrés. Il est destiné au pilotage de moteurs pas à pas bipolaires (4, 6 ou 8 fils). La puissance de son unité intelligente le destine tout autant à des applications simples mono axe qu'à des systèmes multiaxes complexes.

Son étage amplificateur 45V 2.5Aeff le rend particulièrement adapté au pilotage de moteurs pas à pas de taille 17 et 23. La version haute puissance BMAC-H (45V 7Aeff) convient pour les moteurs nema 23, 24 et 34 qui nécessitent un courant plus élevé. Le mode de pilotage en courant sinusoïdal permet une grande souplesse de mouvement, un silence de fonctionnement et une faible sensibilité aux phénomènes de résonance. Le BMAC est protégé en thermique et en surcourant contre les courts-circuits entre phases ainsi qu'entre phase et 0V.

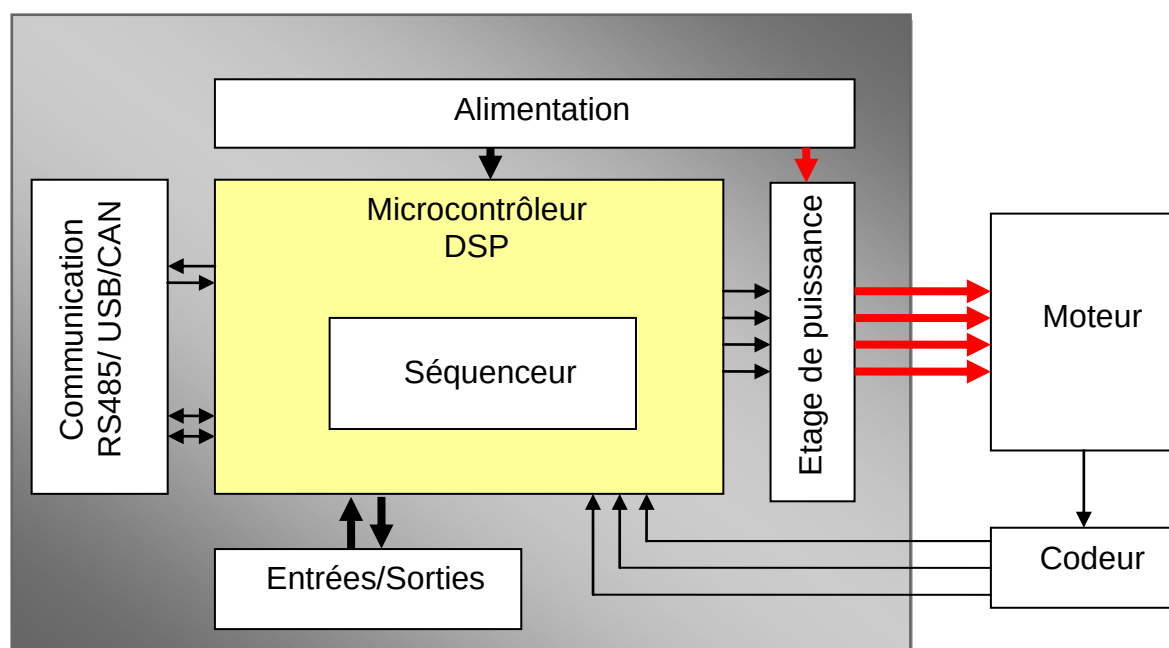
Le moteur peut être piloté en boucle ouverte ou bien en mode autocommuté avec un retour codeur. L'autocommutation empêche tout décrochement et permet le pilotage d'un moteur en position et en vitesse mais aussi en couple sans correcteur d'asservissement.

Le BMAC est équipé d'un séquenceur de commandes dédié au contrôle de l'axe, complété par 8 entrées-sorties logiques opto-isolées et une entrée analogique différentielle. Le module peut fonctionner de manière totalement autonome avec ses commandes enregistrées. 500 lignes de commande sont disponibles pour réaliser des automatismes.

Un protocole de communication basé sur les standards RS232/485 ou CANopen Motion Control permet la commande de l'axe et les communications multi-axes pour des applications 2D et 3D. La connectique sous forme de bornier débouchable simplifie le câblage, la mise en service et la maintenance.

1.2. Fonctionnalités

1.2.1. Architecture





1.2.2. Moteurs compatibles

Le BMAC permet de piloter un moteur pas à pas hybride bipolaire 4 fils dont le courant nominal est de 2.5Aeff maximum. Le BMAC-H atteint 7Aeff.

Le pilotage s'effectue en micropas avec une résolution de 50 micropas par pas.

Un moteur 200pas/tour est souhaitable pour la cohérence des vitesses (exprimées en 1/100 tour/min) et de la position (en micropas ou 1/10000 tour).

1.2.3. Réglage du courant moteur

Dans un premier temps il est nécessaire de programmer le courant nominal moteur via la variable `#RATED_CURRENT` exprimée en mA (valeur maximum 2500 sur BMAC et 7000 sur BMAC-H). Ce courant est indiqué sur la plaque signalétique du moteur ou bien dans sa fiche de spécification. Il s'agit du courant efficace par phase.

Ensuite le courant souhaité est défini via la variable `#CURRENT_RATIO` (0 à 1000) exprimée en pour mille de `#RATED_CURRENT`. Finalement, le courant efficace exprimé en mA est donc donné par la formule : $\#RATED_CURRENT * \#CURRENT_RATIO / 1000$

Note : la résolution du courant réellement généré est de 25mA sur BMAC et 70mA sur BMAC-H.

A tout instant, les courants de consigne des 2 phases sont donnés par les formules :

$$I_{\text{phaseA}} = \sqrt{2} * \#RATED_CURRENT * \#CURRENT_RATIO * \cos(2\pi * (\#ELECTRICAL_POSITION / 200))$$

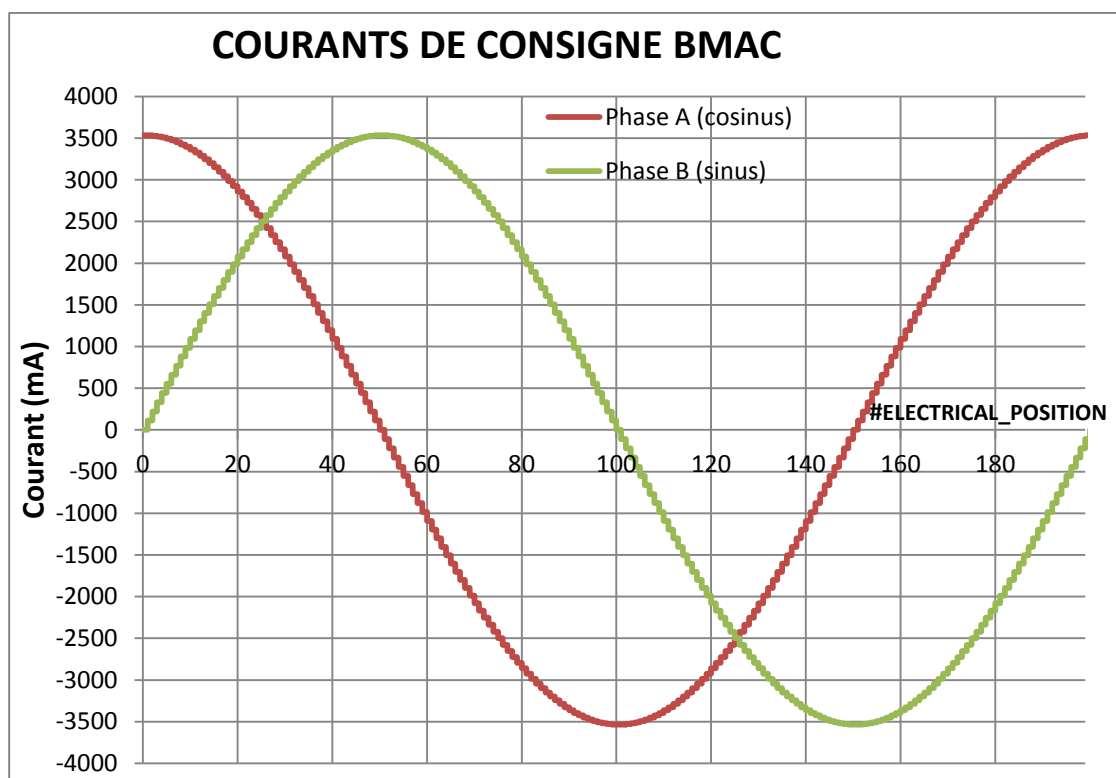
$$I_{\text{phaseB}} = \sqrt{2} * \#RATED_CURRENT * \#CURRENT_RATIO * \sin(2\pi * (\#ELECTRICAL_POSITION / 200))$$

avec $0 \leq \#ELECTRICAL_POSITION \leq 199$

(lors d'un mouvement de sens positif, `#ELECTRICAL_POSITION` s'incrémente modulo 200)

Les pas entiers correspondent aux valeurs 0, 50, 100 et 150 de la variable `#ELECTRICAL_POSITION`.

Exemple avec `#RATED_CURRENT:=2500` et `#CURRENT_RATIO:=1000` :





En fonctionnement autocommuté, un mode optimisé (`OPTIMIZED_CURRENT`) permet d'injecter dans le moteur le courant réellement nécessaire afin de minimiser les pertes thermiques.

La puissance moteur peut être coupée ou rétablie grâce à une commande (`POWER ON` ou `POWER OFF`). Elle est automatiquement rétablie au début de chaque mouvement.

1.2.4. Codeur

Le BMAC peut être connecté à un codeur incrémental biphasé avec les voies A, /A, B, /B, I, /I disposant de niveaux 5V RS422. Les 3 résistances de terminaison de ligne sont intégrées dans le BMAC.

Le codeur peut être alimenté via le +5V COD généré par le BMAC pourvu qu'il consomme moins de 80mA.

La variable `#ENCODER` indique la position du codeur incrémental biphasé, exprimée en quadruple résolution. Par exemple avec un codeur 10000points/tour il y aura 40000 incréments `#ENCODER` par tour.

L'entrée index (I, /I) peut être utilisée pour la prise d'origine (cf. commande `REFERENCE INDEX`).

Pour un fonctionnement en mode boucle fermée autocommutée, il est impératif d'utiliser un codeur 500 points associé à un moteur 200pas/tour en assurant également la cohérence des sens de rotation. Pour cela, effectuer un mouvement positif en mode boucle ouverte et vérifier que `#ENCODER` s'incrémente. Si ce n'est pas le cas, inverser une phase moteur ou codeur.

1.2.5. Boucle ouverte ou autocommutation

Le mode de fonctionnement est sélectionné par la commande `CLOSE_LOOP ON/OFF`. Il est tracé dans le bit `#STATUS.28` qui indique un fonctionnement BMAC en mode autocommuté. Cette sélection est mémorisée hors tension.

Par défaut (sortie usine ou commande `MODULE_RESET ALL`), le BMAC est en boucle ouverte.

Le tableau suivant indique les différences entre le mode boucle ouverte et le mode autocommuté.



	BOUCLE OUVERTE CLOSE_LOOP OFF	BOUCLE FERMEE CLOSE_LOOP ON
Codeur	Avec ou sans codeur Codeur de résolution quelconque et de sens de rotation quelconque. Le ratio angle moteur/ angle codeur peut varier (par exemple avec un codeur linéaire en sortie d'un système bielle/manivelle)	Codeur 500 points impératif, disposé sur l'axe moteur avec cohérence des sens de rotation moteur/codeur.
Autocommutation	Inactive	Fonctionnelle
Variable #POSITION	Indique la position électrique de consigne exprimée en micropas avec une résolution de 50µpas/pas. Cette variable est indépendante du suivi du codeur éventuel via #ENCODER	Indique la position physique de l'axe avec une résolution de 10000incrments/tour et une précision d'1/1000 de tour.
Gestion des mouvements	Réalisée sur la position électrique #POSITION.	Réalisée sur la base de la variable #POSITION.
asservissement de courant via OPTIMIZE_CURRENT	Sans objet	Fonctionnel

1.2.6. Mouvements

Les mouvements se font avec une résolution de $1/10000^{\text{ème}}$ de tour. Un incrément de position correspond donc à un angle $0,036^{\circ}$.

Afin d'avoir un maximum de résolution dans les vitesses basses, toutes les vitesses sont exprimées en $100^{\text{ème}}$ de tour par minute (10^{-2} tr/min) et les positions en incréments (10^{-4} tr).

On distingue trois types de mouvements:

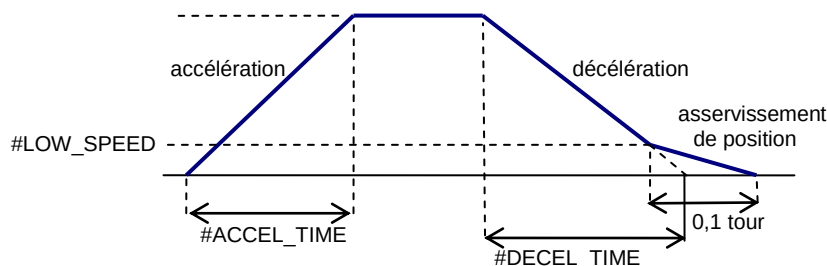
- Le mode Vitesse (MOVE_SPEED) dans lequel le moteur tourne à une vitesse de consigne.
- Le mode Position (MOVE_TO et MOVE_ON) dans lequel le moteur tourne jusqu'à une position de consigne définie en absolu par rapport à une origine ou en relatif par rapport à la position courante.
- Le mode Interpolation (MOVE_INTERPOL) permettant d'exécuter des déplacements coordonnés sur plusieurs axes via des segments chargés dans la mémoire tampon (FIFO).

Pour les modes Vitesse et Position, les mouvements se font suivant des profils de vitesses pour minimiser l'accélération subie par la charge et fluidifier le mouvement. Pour chaque profil de vitesse, l'utilisateur peut choisir une rampe trapézoïdale ou en « S » et peut spécifier la durée des phases d'accélération et de décélération.

Lorsque les mouvements sont commandés en "mode position" (commandes MOVE_TO et MOVE_ON), le module effectue un asservissement pour atteindre exactement la position de consigne avec une précision de $1/10000^{\text{ème}}$ de tour. Cet asservissement se fait à une vitesse proportionnelle à #LOW_SPEED et à l'écart de position.

Un mouvement complet se décompose donc en quatre étapes successives:

- ✓ Accélération de la vitesse courante (qui peut être nulle) jusqu'à la vitesse de consigne.
- ✓ Mouvement à vitesse constante.
- ✓ Décélération de la vitesse de consigne jusqu'à la vitesse #LOW_SPEED.
- ✓ Asservissement de position. Le moteur passe de la vitesse #LOW_SPEED jusqu'à atteindre une vitesse nulle sur la position d'arrivée.



1.2.7. Butées

Le BMAC dispose de quatre butées:

- ✓ Deux butées "matérielles" (aussi appelées butées Hard). Ce sont des butées électriques qui utilisent les entrées logiques IN7 et IN8. On peut y connecter un capteur, un contact sec, etc.
- ✓ Deux butées "virtuelles" (aussi appelées butées Soft). Ce sont des butées gérées par le logiciel qui empêchent de dépasser une position donnée. Cette position est paramétrable grâce aux variables #POSITIVE_END et #NEGATIVE_END.

Au passage d'une de ces butées, si elles sont activées, le mouvement en cours est immédiatement arrêté et tout mouvement dans le même sens est interdit. Seuls les mouvements dans le sens inverse, qui permettent de se dégager de la butée sont autorisés.

La relecture de la variable #STATUS permet de savoir si le module est arrêté sur une butée.

1.2.8. Séquenceur

Le BMAC peut mémoriser puis exécuter jusqu'à 500 commandes, ce qui permet de réaliser des automatismes qui pourront être exécutés en autonome (sans automate ni PC de commande).

L'écriture des séquences est simple et intuitive. Les commandes mémorisées sont exécutées successivement au rythme d'une toutes les millisecondes.

Des commandes évoluées spécifiques (IF, JUMP, CALL, WAIT, etc.) permettent de contrôler le déroulement des séquences pour conditionner l'exécution à des événements extérieurs (entrées logiques et analogique, position, variables internes, etc....)

1.2.9. Entrées – Sorties

Le BMAC dispose de 8 entrées/sorties opto-isolées. Elles peuvent être utilisées conjointement avec le séquenceur pour, par exemple, déclencher des mouvements prédéfinis. Deux entrées butées peuvent être utilisées comme des entrées logiques banalisées lorsque les fonctions de butées ne sont pas utiles.

Une entrée analogique différentielle permet de connecter un capteur ou un potentiomètre. Comme pour les entrées logiques, la valeur de l'entrée analogique peut être utilisée dans le module par l'intermédiaire du séquenceur pour, par exemple, asservir la position de l'axe à la position d'un potentiomètre.



1.3. Précautions d'emploi

1.3.1. Règles générales

- Le module est qualifiés IP30, respecter les limites relatives à cet indice de protection. En particulier, le module n'est pas étanche, il doit être protégé contre les projections de liquide et les ruissellements.
- Eviter les projections de solvants, acides, bases.
- Eviter l'exposition aux radiations.
- Ne jamais ouvrir un module. Les tensions internes peuvent atteindre un niveau dangereux.
- Ne pas toucher un module ou un moteur sous tension : risque de brûlure ou d'électrocution.
- Ne pas toucher l'arbre moteur : risque de blessure.
- Se référer à la spécification du moteur utilisé pour dimensionner le pilotage et vérifier les contraintes thermiques et mécaniques

1.3.2. Conditions de stockage

- Le module doit être stocké ou transporté dans son emballage d'origine ou dans un conditionnement adapté.
- Protéger le module contre les rayons solaires et l'humidité.
- La température doit être comprise entre -20°C et $+40^{\circ}\text{C}$.

1.3.3. Conditions d'utilisation

- **Attention ! Le moteur peut atteindre une température élevée lors du fonctionnement. Ne pas toucher le module amplificateur ou le moteur, même hors mouvement.**
- **Ne jamais intervenir sur les connexions d'un module sous tension. Couper l'alimentation et attendre 20s au minimum avant toute intervention.**
- Respecter l'affectation des broches sous peine de destruction du système.
- Le module doit se trouver à l'air libre avec une température ambiante comprise entre -10°C et $+40^{\circ}\text{C}$.
- Ne pas poser le produit sur un emplacement qui ne soit pas stable : le produit pourrait tomber et entraîner des blessures ou être endommagé.
- Relier la masse mécanique du BMAC à la masse générale de la machine ainsi qu'à la terre.
- Ne jamais introduire un corps étranger dans les orifices du produit.



2. Spécifications techniques

2.1. Alimentation

	Tension d'alimentation	Courant consommé maximum
BMAC	12 à 45Vdc	2A
BMAC-H	24 à 45Vdc	6A

Attention ! Plus la tension est faible, plus le courant consommé augmente.

Le courant réellement nécessaire dépend de la puissance mécanique totale requise :

$P = C * \omega$ (P est la puissance en Watts, C est le couple en Nm, ω est la vitesse de rotation du moteur en rad/s).

Plus la tension d'alimentation est élevée, meilleur est le couple à haute vitesse.

Le couple à l'arrêt ou à faible vitesse est indépendant de la tension d'alimentation.

Lorsque la tension descend en dessous de 12V, les mouvements sont arrêtés et le module se place en disjonction de sous-tension.

Remarque : Lors des phases de freinage, l'énergie cinétique récupérée est renvoyée vers l'alimentation, celle-ci doit donc accepter un éventuel courant inverse. Dans ce cas de figure, la tension d'alimentation peut se mettre à monter (charge des condensateurs de sortie de l'alimentation). Les modules BMAC possèdent une sécurité interne qui coupe le freinage lorsque la tension d'alimentation dépasse 46V. Une protection complémentaire coupe définitivement la puissance moteur si la tension dépasse 49V. Le module se met alors en défaut jusqu'à coupure d'alimentation, reset par la commande "MODULE_RESET" ou écriture #ERROR:=0.



Quelle que soit la valeur nominale de l'alimentation utilisée, la tension générée par le BMAC peut atteindre 49 Vdc.

L'option Ballast permet de dissiper l'énergie récupérée dans des résistances de puissance de façon à ne jamais atteindre le seuil de protection.

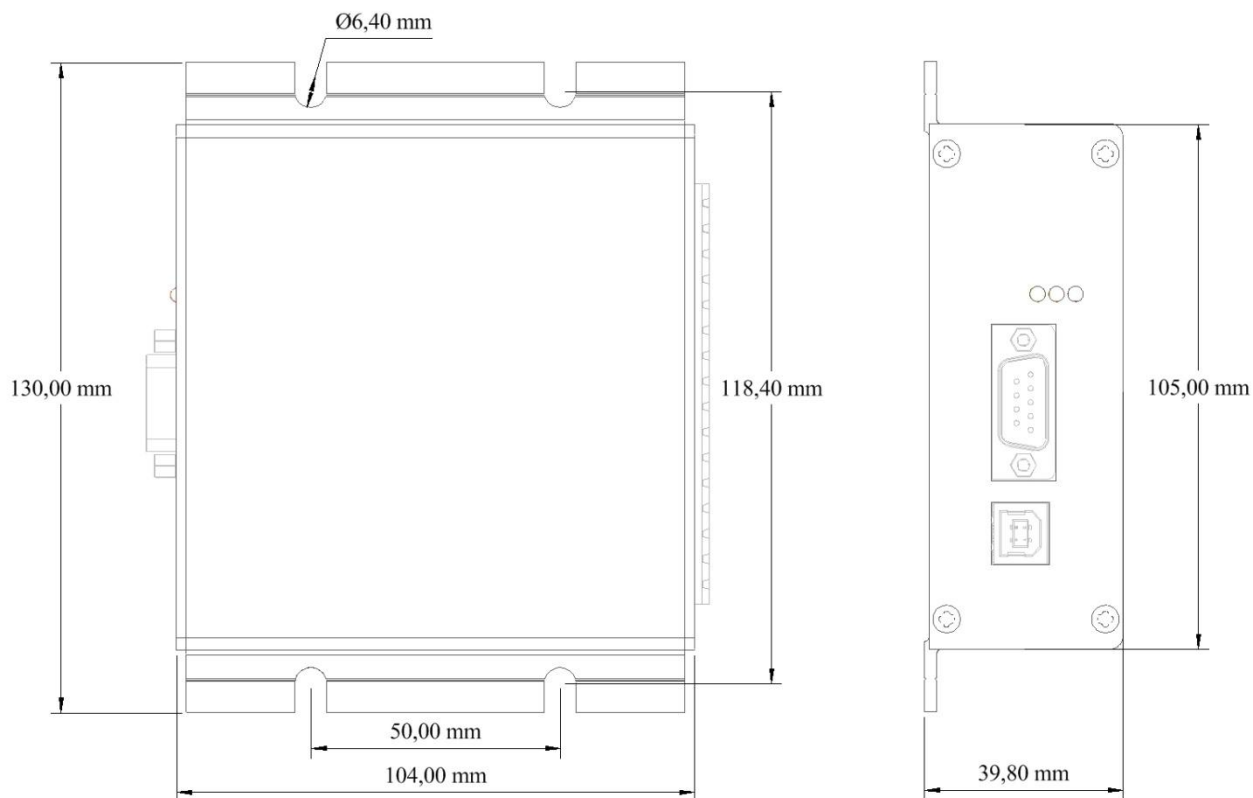
Dans tous les cas, si l'alimentation ne supporte pas la tension nominale du module en récupération il convient d'insérer une diode (75V/7A) en série dans le circuit d'alimentation du module.

2.2. Spécifications mécaniques

2.2.1. BMAC boîtier

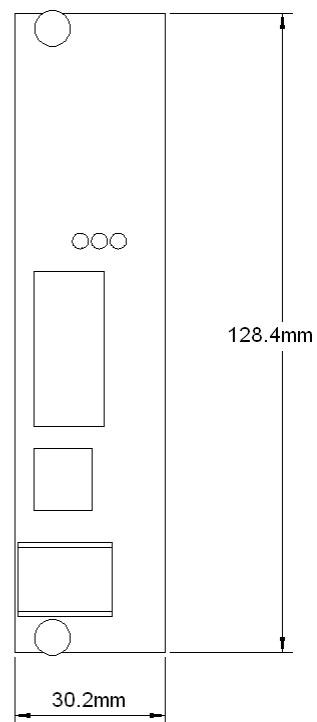
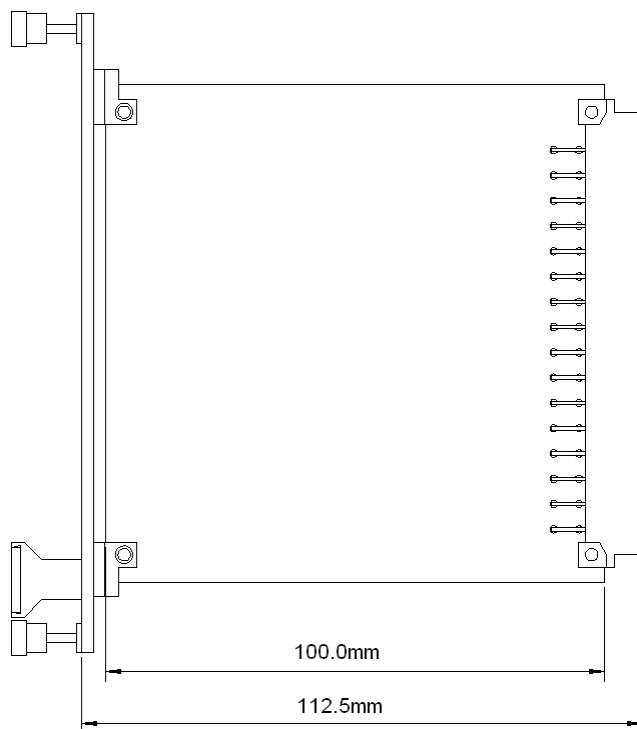
Masse: 470g

Plan d'encombrement:



2.2.2. BMAC rack

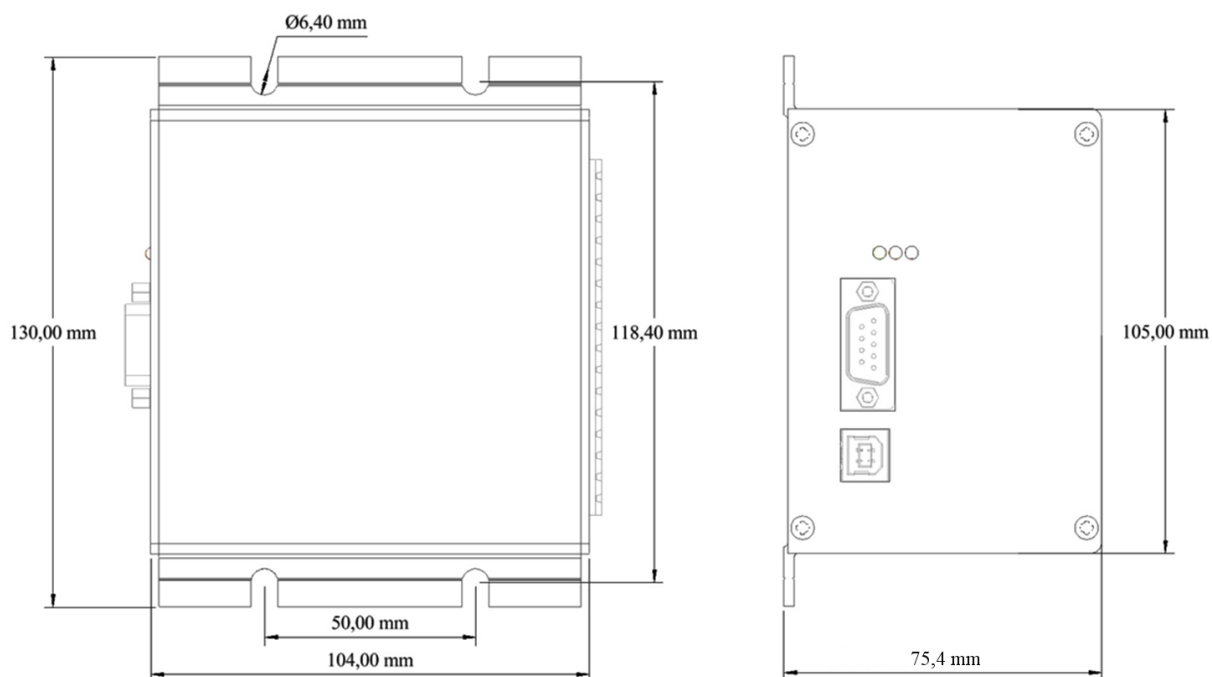
Plan d'encombrement:



2.2.3. BMAC-H

Masse: 745g

Plan d'encombrement:



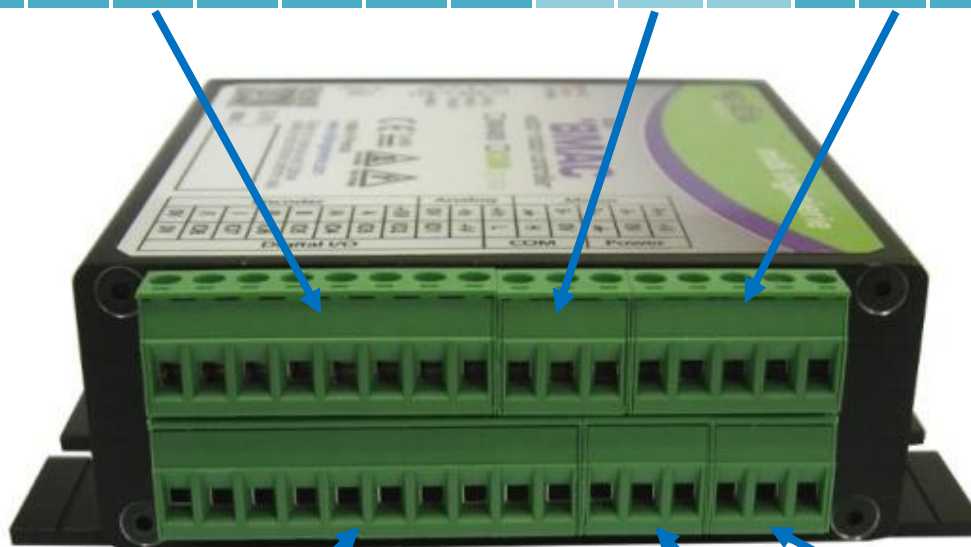


3. Câblage et configuration

3.1. Brochage des connecteurs

3.1.1. BMAC boîtier

ENCODER MSTB2.5/8-ST-5.08								ANALOG MSTB2.5/3-ST-5.08			MOTOR MSTB2.5/5-ST-5.08				
0V COD	COD /I	COD I	COD /B	COD B	COD /A	COD A	+5V COD	0V ANA	- IANA	+ IANA		B -	B +	A -	A +



DIGITAL I/O MSTBT2.5/10-ST-5.08										RS485 CAN MSTBT2.5/3-ST-5.08			POWER MSTBT2.5/3-ST-5.08		
0V_IO	I/O8	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	+V_IO	/Z CANL	Z CANH	0V		0V	+V

SubD9 Mâle : RS485 ou bus CAN					
1	Réservé	4	Réservé	7	Z CANH
2	/Z CANL	5		8	Réservé
3	0V485 CAN	6	Réservé	9	Réservé

3.1.2. BMAC rack

Brochage du connecteur DIN41612 D mâle :

ENCODER								ANALOG			MOTOR				
0V COD	COD /I	COD I	COD /B	COD B	COD /A	COD A	+5V COD	0V ANA	- IANA	+ IANA		B -	B +	A -	A +
32C	30C	28C	26C	24C	22C	20C	18C	16C	14C	12C	10C	8C	6C	4C	2C



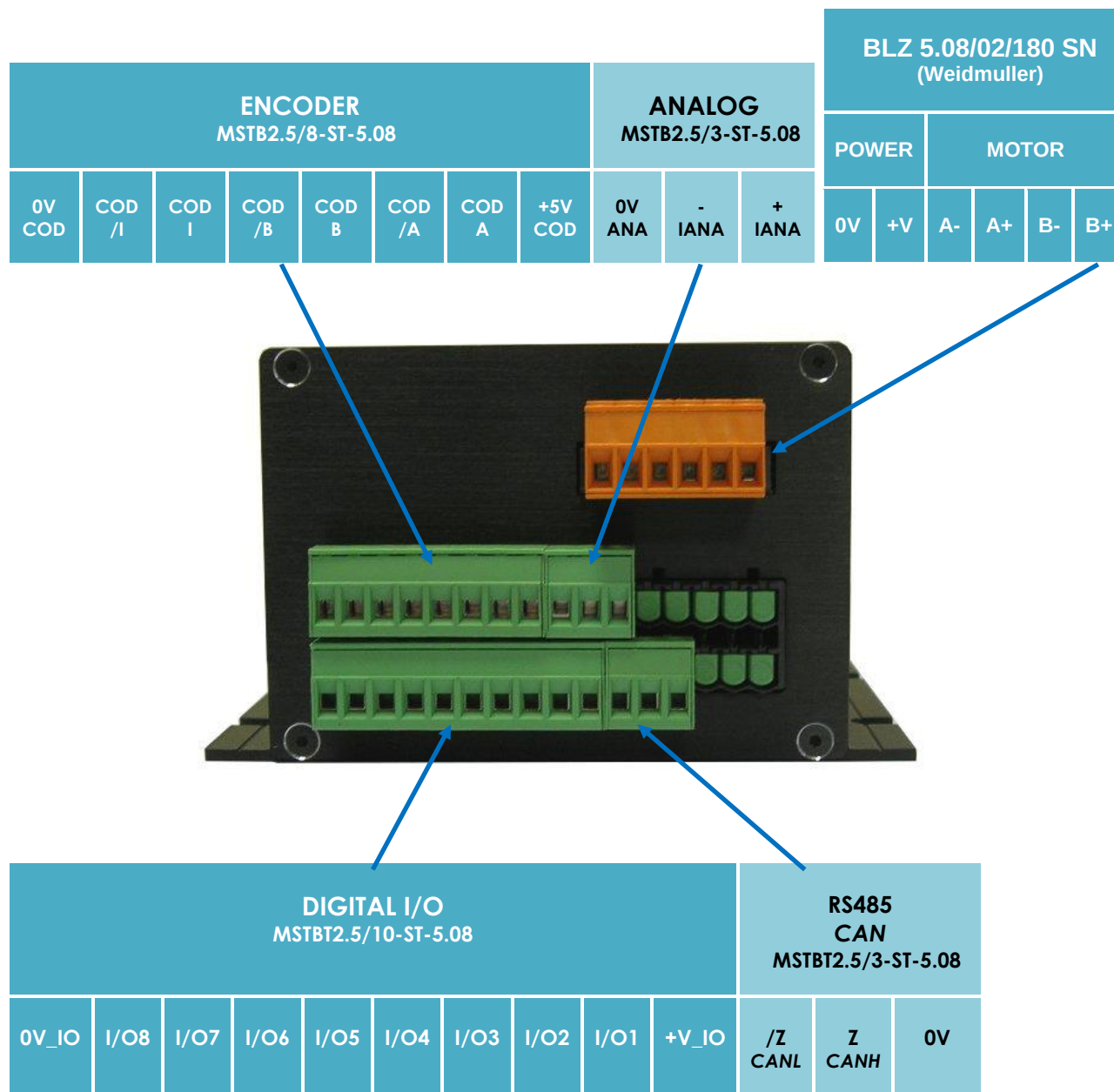
32A	30A	28A	26A	24A	22A	20A	18A	16A	14A	12A	10A	8A	6A	4A	2A
0V_IO	I/O8	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	+V_IO	/Z CANL	Z CANH	0V		0V	+V

DIGITAL I/O										RS485 CAN		POWER	
-------------	--	--	--	--	--	--	--	--	--	--------------	--	-------	--

SubD9 Mâle : RS485 ou bus CAN					
1	Réservé	4	Réservé	7	Z CANH
2	/Z CANL	5		8	Réservé
3	0V485 CAN	6	Réservé	9	Réservé



3.1.3. BMAC-H



SubD9 Mâle : RS485 ou bus CAN					
1	Réservé	4	Réservé	7	Z CANH
2	/Z CANL	5		8	Réservé
3	0V485 CAN	6	Réservé	9	Réservé

Pour la liaison de masse mécanique du boîtier, il est préconisé d'utiliser un montage vis + rondelle AZ + cosse à œillet + rondelle AZ sur une des 4 vis de fixation du boîtier.



3.2. Fonctionnalités des entrées/sorties

Le BMAC dispose de 8 entrées/sorties protégées en surcoursant.

Chacune des 8 broches peut être exploitée en entrée ou en sortie :

- pour une utilisation en sortie il suffit de piloter le bit correspondant, par exemple #OUTPUT.4 :=1. L'entrée associée recopie l'état de la sortie, dans cet exemple #INPUT.4 est lue à 1
- attention, pour utiliser une broche en entrée il est nécessaire que la sortie correspondante soit inactive (par exemple #OUTPUT.7 := 0). L'entrée peut alors être lue via la variable #INPUT (ici #INPUT.7)

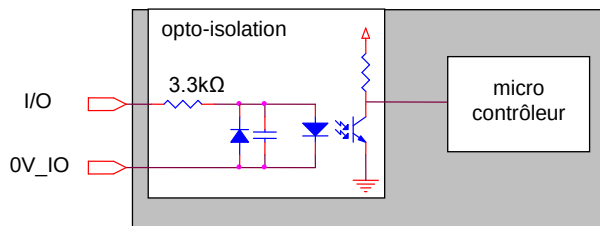
Les entrées sorties utilisateur sont multiplexées avec des fonctions évoluées selon #OUTPUT_CONFIG:

ENTREE/SORTIE	FONCTION
I/O 1	Entrée utilisateur OU sortie utilisateur ou sortie BUSY (suivant #OUTPUT_CONFIG.1) OU sortie Synchro interpolation (si mode interpolé synchronisé)
I/O 2	Entrée utilisateur OU sortie utilisateur ou sortie DEFAULT (suivant #OUTPUT_CONFIG.2)
I/O 3	Entrée utilisateur OU sortie utilisateur
I/O 4	Entrée utilisateur OU sortie utilisateur
I/O 5	Entrée utilisateur et entrée capture de position ou entrée référence OU sortie utilisateur
I/O 6	Entrée utilisateur ou entrée Synchro interpolation OU sortie utilisateur
I/O 7	Entrée utilisateur ou entrée Butée sens positif OU sortie utilisateur
I/O 8	Entrée utilisateur ou entrée Butée sens négatif OU sortie utilisateur



3.3. Spécification des entrées-sorties

3.3.1. Entrées logiques opto-isolées



	min	max
V_{L} (inactif)	-30V	1V
V_{H} (actif)	4V	+30V
I_{L}		25μA
I_{H}	1,1mA	

Tension nominale 5VDC à 24VDC

Tension du signal (inactif) 0VDC à 1VDC

Tension du signal (actif) 4VDC à 30VDC

Tension admissible ±30VDC

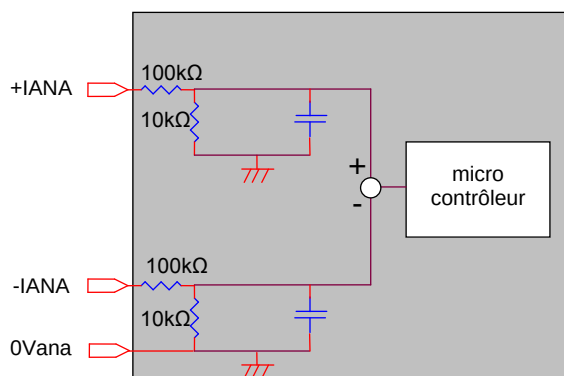
Isolation galvanique 50V

La référence 0VIO est commune à toutes les entrées logiques.

Les entrées logiques peuvent être relues avec la commande `READ #INPUT.`

3.3.2. Entrées analogiques

Schéma de principe pour une entrée différentielle :

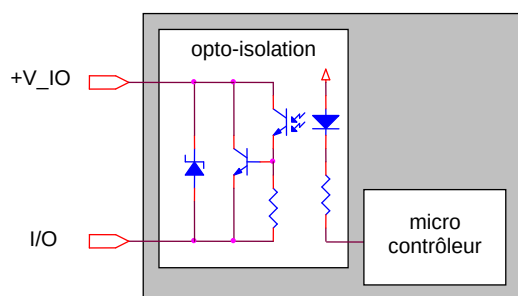


La variable `#INPUT_ANALOG` indique la tension différentielle entre les points +IANA et -IANA, exprimée en mV.

Plage de tension d'entrée / 0Vana	0 à 35V
Plage de tension d'entrée différentielle	±35V
Fréquence de coupure	1KHz
Impédance d'entrée différentielle	220KOhms
Précision	±3%
Résolution	8,6mV



3.3.3. Sorties logiques opto-isolées



	min	max
$I_{O_{off}}$		0,1mA
$V_{O_{on}}$		0,6V @1mA 1,1V @5mA 1,3V @50mA

Courant de sortie max 50mA

Tension de sortie max 30V

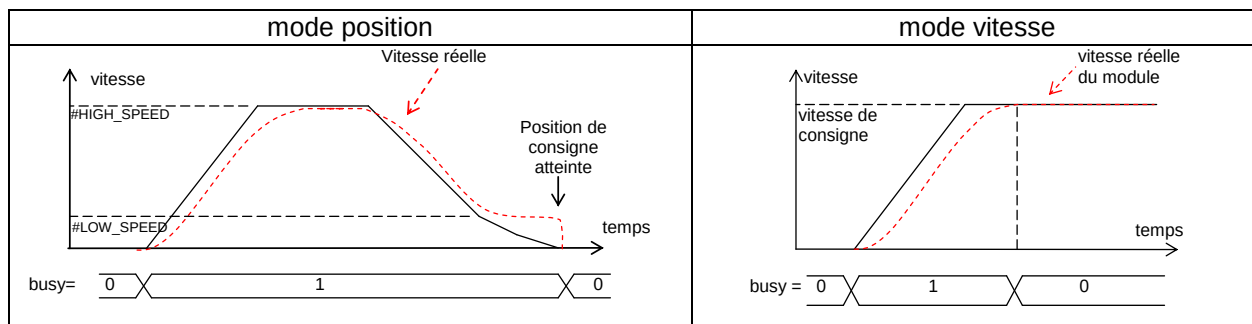
La tension +VIO est commune à toutes les sorties logiques.

Un niveau logique 1 (#OUTPUT:=1) fait conduire l'optocoupleur.

Un niveau logique 0 (#OUTPUT:=0) ouvre l'optocoupleur.

Les sorties logiques peuvent être positionnées avec la commande #OUTPUT, et peuvent être relues avec la commande READ #OUTPUT.

La sortie OUT1 peut matérialiser l'information BUSY si le bit correspondant de #OUTPUT_CONFIG est à 1. Le signal Busy est actif lorsque le moteur ne parvient pas à suivre sa consigne de vitesse en mode vitesse ou tant que la position de consigne n'est pas atteinte en mode position, comme l'indique le diagramme ci-après.



La sortie OUT2 peut indiquer le signal DEFAULT si le bit correspondant de #OUTPUT_CONFIG est à 1.

Le signal DEFAULT est actif lorsque l'un des événements suivant s'est produit (cf. tracé dans #ERROR)

- disjonction surcourant ou court-circuit moteur
- disjonction différentielle si court-circuit entre une phase moteur et le 0V ou la terre
- disjonction surtension si la tension d'alimentation monte au-delà de la tension maximum.
- disjonction sous-tension si la tension d'alimentation chute au dessous de la tension minimum.
- disjonction thermique si la température de l'étage de puissance atteint 85°C ou bien si la température du processeur atteint 120°C.

3.4. Visualisations

Trois LEDs résument l'état du module BMAC :

- la led jaune "Busy" matérialise l'activité du moteur en signalant tout mouvement ou séquence en cours,
- la led rouge "Défaut" indique un défaut : surtension, sous-tension, température excessive ou disjonction moteur. Par ailleurs, cette led clignote à la mise sous tension, à l'initialisation et à la mise à zéro du module. En dehors des cas de défaut elle est allumée en vert
- la led verte "alimentation" signale que le BMAC est sous tension. Attention, elle reste active même si la tension d'alimentation est trop forte ou trop faible par rapport à la plage nominale du produit. Elle devient orange en cas de court-circuit sur la sortie +5V COD.



3.5. Configuration du port de communication

Les modules peuvent communiquer en USB ou en RS485. Le protocole RS485 est conseillé dans les applications multi-axes et présente une meilleure immunité au bruit. Avant toute installation d'un nouveau BMAC en USB, veuillez consulter la note d'application « Installation du driver USB FTDI ».

3.5.1. Paramétrage du module

Pour paramétrer le BMAC, entrer les valeurs adéquates:

SET_BAUDRATE (vitesse de transfert en bauds 9600, 19200, 38400 ou 115200)

SET_ADDRESS (adresse du module 0 à 63)

#LINE_DELAY (délai de retournement de ligne RS485 en microsecondes, valeurs 100 à 3000)

Par défaut, en sortie d'usine, le module est configuré avec:

SET_BAUDRATE 38400

SET_ADDRESS 0

#LINE_DELAY:=3000

3.5.2. Le protocole RS-485

Qu'est-ce qu'un réseau RS485?

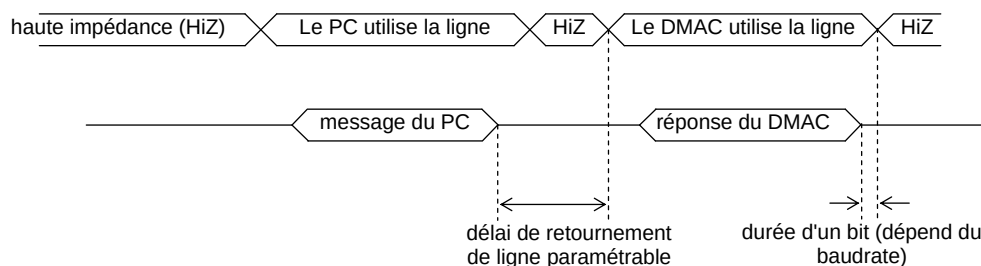
Le protocole RS485 permet la mise en réseau de plusieurs équipements (jusqu'à 32) qui peuvent communiquer entre eux en half-duplex sur un bus composé de deux contacts et de la masse. La longueur totale du bus peut aller jusqu'à 1200 mètres (Notons que la longueur maximale et le nombre d'équipements peut être augmenté en utilisant des répéteurs). Il est tout à fait possible de connecter des BMAC et DMAC de différents types (17, 23, 34) sur une même ligne de communication sous réserve d'utiliser le même baudrate et des adresses distinctes.

Comment fonctionne le matériel?

Les données sont transmises sur une paire différentielle (si possible blindée et torsadée), ce qui confère au signal une bonne immunité aux perturbations électromagnétiques, permettant ainsi d'accroître la distance maximale de transfert. L'émetteur et le récepteur sont connectés sur la même ligne. Un seul équipement peut "prendre la ligne", tous les autres équipements du réseau doivent être à l'état de haute impédance. Une résistance de terminaison 120ohm doit être disposée aux deux extrémités de la ligne.

Quel est le rôle du logiciel?

La conséquence directe du fonctionnement "3 états" de la ligne est le délai qu'il est nécessaire d'imposer entre la prise de ligne et le début de la communication, ainsi qu'entre la fin de la communication et la mise en haute impédance du driver. Tout ceci est géré par le logiciel embarqué dans le BMAC et par l'interface utilisée par le contrôleur (PC ou automate). Dans le cas courant de l'utilisation de la dll DrvMi ou de Winsim2, la gestion de la ligne est transparente pour l'utilisateur.





Comment communiquer avec le module?

Il existe plusieurs méthodes pour s'interfacer avec un module BMAC, chacune utilisant des couches d'abstraction différentes:

WinSim2 Logiciel sous Windows permettant de s'interfacer avec les modules et de communiquer de manière visuelle et simplifiée. WinSim2 offre de nombreuses possibilités de contrôle et de suivi. Il est particulièrement adapté au développement et à la maintenance.
DrvMi.dll Bibliothèque de fonctions accessibles depuis des programmes développés par l'utilisateur en C/C++, VisualBasic, Delphi, etc...

4. Commandes

Toutes les commandes sont envoyées au module sous la forme d'une chaîne de caractères formée de l'adresse, de la commande et des paramètres.

On peut envoyer indifféremment le nom complet de la commande (exemple: MOVE_TO) ou son mnémonique (exemple: MTO).

L'adresse du module est indiquée par les deux premiers caractères de la commande.

Si l'adresse n'est pas précisée, la commande est dite "générale" et tous les modules présents sur la ligne l'exécutent. Seul le module d'adresse 0 (zéro) répond à une commande générale.

On peut envoyer plusieurs commandes, séparées par des virgules, dans la même chaîne de caractère (exemple: "#HIGH_SPEED:=10000, MOVE_TO 1234"). La totalité de la commande ne doit cependant pas dépasser 256 caractères.

L'affectation numérique d'une commande peut être remplacée par une variable, par exemple :

```
#V12 :=10000
```

```
MOVE_SPEED #V12 ; équivaut ici à MOVE_SPEED 10000
```

Les commandes sont séparées des paramètres par un espace.

Certaines commandes peuvent être utilisées en mode direct (hors séquence) et en mode séquence. D'autres ne peuvent être utilisées que dans l'un ou l'autre des deux modes.

Dans la suite du document, on utilise les conventions suivantes:

@ représente l'adresse du module

[] représente un paramètre optionnel



4.1. CALL / RETURN

Syntaxe:	CALL paramètre RETURN	
Mnémonique:	CAL RET	
Paramètres:	numéro de la première ligne de la routine à exécuter pour CALL 1 < paramètre < 500	
Description:	Appel d'une routine. A la fin de l'exécution de la sous routine (RETURN), le séquenceur continuera l'exécution linéaire à la ligne qui suit le CALL.	
Remarques:	Il est possible d'imbriquer jusqu'à 5 routines Il est important que chaque routine appelée par un CALL se termine par un RETURN.	
Exemple:	<pre> : 4 CALL 12 Appel de la routine située à la ligne 12 NOP NOP : 12 NOP Ligne 12: début de la sous-routine NOP RETURN Retour à la ligne 5 (qui suit le CALL) </pre>	

4.2. CLOSE_LOOP

Syntaxe:	[@]CLOSE_LOOP paramètre	
Mnémonique:	CLO	
Paramètre:	ON (boucle fermée, autocommutation) ou OFF (boucle ouverte)	
Description:	Sélectionne le mode de fonctionnement du BMAC. Par défaut (sortie usine ou commande MODULE_RESET ALL), le BMAC est en boucle ouverte. Ce paramètre est tracé dans le bit #STATUS.28 qui indique un fonctionnement BMAC en mode autocommuté. Il est mémorisé et se conserve en cas de coupure d'alimentation.	
Exemple:	<pre> 03CLOSE_LOOP ON Active la boucle fermée du module d'adresse 03. </pre>	



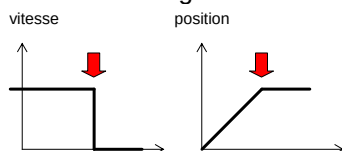
4.3. HALT

Syntaxe: `[@]HALT [paramètre]`

Mnémonique: HAL

Paramètre: Aucun, MOUV ou SEQ

Description: Le module s'arrête sans suivre une rampe de vitesse. Le couple de freinage est maximum, la valeur de la décélération est imposée par le rapport entre le couple et l'inertie de la charge. Le suivi de position est conservé.



La commande HALT sans paramètre arrête à la fois le mouvement en cours et la séquence en cours. Il est possible de n'arrêter que le mouvement (commande `HALT MOUV`) en poursuivant l'exécution d'une séquence.

Dans le cas d'un mouvement en mode interpolé, l'interpolation est stoppée et la FIFO d'interpolation est vidée.

Il est également possible de n'arrêter que la séquence (commande `HALT SEQ`) et de poursuivre le mouvement en cours.

Exemple: `05HALT`

l'axe d'adresse 05 s'arrête. Si une séquence était en cours d'exécution, elle est arrêtée.

4.4. HARD_ENDS

Syntaxe: `[@]HARD_ENDS paramètre`

Mnémonique: HEN

Paramètre: ALL (butées actives) ou OFF (butées inactives)
POS (active la butée positive uniquement) ou NEG (active la butée négative uniquement)

Description: Active ou désactive les butées matérielles.
La butée positive arrête et empêche les mouvements dans le sens positif.
La butée négative arrête et empêche les mouvements dans le sens négatif.
Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.

Remarques: Attention, l'inversion de polarité (commande `INVERSE_POLARITY`) intervient sur les butées matérielles au même titre que sur les entrées logiques.

Exemple: `03HARD_ENDS ALL`

Active les butées matérielles du module d'adresse 03.



4.5. IF

Syntaxe:	IF test JUMP paramètre IF test CALL paramètre IF test JUMP_REL paramètre
Mnémonique:	IF
Paramètres:	"test" est une opération de test sur une variable dont le résultat est VRAI ou FAUX. "paramètre" est le numéro de la ligne à atteindre si le test est VRAI Si le test est FAUX, l'exécution continue linéairement. [0 ; 500]
Description:	Saut conditionnel de ligne. Cette instruction permet de rompre l'exécution linéaire pour sauter directement à une ligne donnée. Le saut peut être de type JUMP, CALL ou JUMP_REL (voir détails dans la description de chacune de ces commandes)
Exemple:	:10 IF #POSITION > 1234 JUMP 56 Si la position courante est supérieure à 1234, la prochaine ligne exécutée sera la ligne 56. Dans le cas contraire, l'exécution continue linéairement à la ligne 11.

4.6. INVERSE_POLARITY

Syntaxe:	[@]INVERSE_POLARITY paramètre
Mnémonique:	IPO
Paramètre:	OFF (entrées et sorties en mode standard) ALL (entrées et sorties en mode inversé) IN (entrées en mode inversé) OUT (sorties en mode inversé)
Description:	Spécifie si l'état physique des entrées et sorties du module est inversé par rapport à l'état logique de #INPUT et #OUTPUT. Pour les entrées: en mode standard, état physique actif (alimentée) ⇔ #INPUT à 1 en mode inversé, état physique actif (alimentée) ⇔ #INPUT à 0 Pour les sorties: en mode standard, #OUTPUT à 1 ⇔ état physique actif en mode inversé, #OUTPUT à 0 ⇔ état physique actif
Exemple:	03INVERSE_POLARITY OUT A partir de maintenant, l'état physique des sorties est inversé par rapport à la variable #OUTPUT.



4.7. JUMP / JUMP_REL

Syntaxe:	JUMP param1 JUMP_REL param2
Mnémonique:	JUM JRE
Paramètre:	0 ≤ param1 < 500 numéro de la ligne à exécuter pour JUMP -500 < param2 < 500 nombre de lignes à sauter pour JUMP_REL
Description:	Saut de ligne. Ces instructions permettent de rompre l'exécution linéaire pour sauter directement à une ligne de séquenceur donnée. L'instruction JUMP saute à une ligne indépendamment de la ligne courante, tandis que l'instruction JUMP_REL saute à un nombre de ligne donné de la ligne courante. Le paramètre de JUMP_REL peut être négatif pour pouvoir revenir en arrière.
Remarques:	L'instruction JUMP 0 a pour effet d'arrêter l'exécution du séquenceur.
Exemple:	JUMP 10 Saut de la ligne courante vers la ligne 10 :15 JUMP_REL -1 Boucle sur la ligne précédente (ligne 14). JUMP_REL +2 Saut deux lignes plus loin



4.8. MODULE_RESET

Syntaxe: `[@]MODULE_RESET [paramètre]`

Mnémonique: MRE

Paramètre: Aucun ou ALL => remise en configuration sortie usine.

Description: Réinitialise le module. Le moteur est mis hors puissance puis freiné en court-circuit jusqu'à l'arrêt complet. L'état du module est ensuite similaire à l'état après mise sous tension. Les sorties sont forcées à l'état inactif. Les variables non mémorisées (#Vn) sont remises à 0. Si une séquence de démarrage automatique est programmée, celle-ci est exécutée.

La directive ALL permet de replacer l'ensemble des paramètres à leurs valeurs de sortie usine (hormis les paramètres de liaison série):

```
#HIGH_SPEED := 60000
#LOW_SPEED := 6000
#CURRENT_RATIO := 500
#ACCEL_TIME := 1000
#DECEL_TIME := 1000
#POSITIVE_END := +10000
#NEGATIVE_END := -10000
#POSITION := 0
#OUTPUT_CONFIG := 3
#ON_RESET := 0
```

Les variables mémorisées (#Mn) sont remises à 0.

La polarité nominale des entrées sorties est restituée (INVERSE_POLARITY OFF).

Les butées sont désactivées (HARD_ENDS OFF; SOFT_ENDS OFF).

La configuration des mouvements est restituée (S_CURVE OFF,
OPTIMIZED_CURRENT OFF, CLOSE_LOOP OFF).

Nota : Les paramètres liaison série (baudrate, adresse et #LINE_DELAY) ne sont pas modifiés.

Exemple: `02MODULE_RESET`
Réinitialise le moteur d'adresse 2.



4.9. MOVE_INTERPOL

Syntaxe	[@]MOVE_INTERPOL paramètre
Mnémonique:	MIN
Paramètre:	Déplacement relatif à effectuer (en incréments moteur: 10^{-4} tour) pendant le temps #INTERPOL_TIME . [-2 147 483 648; +2 147 483 647]
Description:	En mode interpolé, chaque axe décrit une trajectoire définie par une succession de "segments". Chaque segment est défini par le déplacement relatif à effectuer dans un temps imparti (#INTERPOL_TIME). A chaque commande MOVE_INTERPOL, un segment est mémorisé dans une FIFO (de taille variable #INTERPOL_FIFOSIZE). Lorsque le mouvement est lancé par la commande globale SYNCHRO INTERPOL, tous les axes exécutent le mouvement mémorisé dans la FIFO de manière synchronisée (voir variable #INTERPOL_MODE). Lorsque la FIFO est pleine, la commande MOVE_INTERPOL est rejetée par l'émission sur la ligne série du caractère XON_ERREUR (17h). Dans le cas contraire, lorsque la FIFO est vide (toute la trajectoire a été parcourue) le mouvement s'arrête et il sera nécessaire de renvoyer la commande SYNCHRO INTERPOL pour relancer un nouveau mouvement interpolé. Pendant le mouvement, il est à la charge de l'utilisateur d'entretenir le mouvement par l'émission des commandes MOVE_INTERPOL à un rythme supérieur à l'exécution de la trajectoire.
Remarque:	Le logiciel PC MISHell permet de répéter automatiquement une commande qui aurait été refusée par un module (FIFO pleine). Il suffit pour cela de la faire précéder du caractère _ (underscore): _01MOVE_INTERPOL 200
Exemple:	Commandes à envoyer pour décrire une trajectoire interpolée sur 3 axes: <pre>#INTERPOL_FIFOSIZE:=64 ;FIFO de 64 segments #INTERPOL_TIME:=100 ;segments de 100ms #INTERPOL_MODE:=0 ;mode SimpleSYNC 00MOVE_INTERPOL 100 ;1^{er} segment 01MOVE_INTERPOL 750 ;(mémorisé en FIFO mais pas 02MOVE_INTERPOL 100 ;encore lancé) SYNCHRO INTERPOL ;départ synchronisé 00MOVE_INTERPOL 100 ;2^{ème} segment 01MOVE_INTERPOL -50 02MOVE_INTERPOL 100 ... ;3^{ème} segment puis suite...</pre>



4.10. MOVE_ON

Syntaxe: `[@]MOVE_ON paramètre`

Mnémonique: MON

Paramètre: Amplitude du déplacement en incrément moteur (10^{-4} tour)
[-2 147 483 648 ; + 2 147 483 647]

Description: Le moteur effectue un mouvement du nombre d'incréments voulu suivant le profil de vitesse défini par les paramètres #ACCEL_TIME, #DECEL_TIME et #HIGH_SPEED.

Le sens du mouvement est défini par le signe du paramètre (un paramètre positif implique une rotation dans le sens horaire lorsque l'on regarde l'axe du moteur).



Remarques: Cette commande force toujours une mise sous puissance du moteur.

Exemple: `03MOVE_TO -2500`
l'axe d'adresse 03 se déplace d'un quart de tour dans le sens anti-horaire

4.11. MOVE_SPEED

Syntaxe: `[@]MOVE_SPEED paramètre`

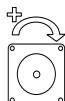
Mnémonique: MSP

Paramètre: Vitesse de rotation du module exprimée en $100^{\text{ème}}$ tour/min
[-400000 ; 400000]

Description: Le moteur tourne dans le sens donné par le signe du paramètre (un paramètre positif implique une rotation dans le sens horaire lorsque l'on regarde l'axe du moteur) à la vitesse donnée par le paramètre.

Le profil de vitesse passe de la vitesse courante à la vitesse de consigne en un temps défini proportionnellement aux paramètres #ACCEL_TIME et #DECEL_TIME.

Remarques:



La consigne de vitesse est bornée (en valeur absolue) par la vitesse maximale définie par la variable #HIGH_SPEED. Un mouvement avec une consigne supérieure ne renvoie pas d'erreur mais sera effectué à la vitesse maximale.

Exemple: `00MOVE_SPEED 50000`
l'axe d'adresse 00 tourne dans le sens positif à 500tr/min



4.12. MOVE_TO

Syntaxe:	[@]MOVE_TO paramètre
Mnémonique:	MTO
Paramètre:	Position de consigne en incrément moteur (10^{-4} tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	Le moteur se déplace de la position courante à la position de consigne suivant le profil de vitesse défini par les paramètres #ACCEL_TIME, #DECEL_TIME et #HIGH_SPEED.
Remarques:	Si le module est déjà à la position de consigne, il n'y a aucun mouvement. Cette commande force toujours une mise sous puissance du moteur.
Exemple:	00MOVE_TO 123456 L'axe d'adresse 00 se déplace jusqu'à la position +123456

4.13. OPEN_SEQ / CLOSE_SEQ

Syntaxe:	[@]OPEN_SEQ [@]CLOSE_SEQ	
Mnémonique:	OSE CSE	
Paramètre:	Aucun	
Description:	OPEN_SEQ ouvre le "mode édition" du séquenceur. CLOSE_SEQ ferme le "mode édition" du séquenceur. Le bit STATUS.16 est positionné pour indiquer que le mode Edition est actif. En mode édition, toutes les commandes saisies ne sont pas exécutées mais mémorisées pour une utilisation future dans des lignes de séquenceur successives. Par défaut, l'édition commence à la ligne 1, mais il est possible de forcer l'édition à une autre ligne en faisant précéder la commande de ":n" où n est le numéro de la ligne. La commande OPEN_SEQ commence par effacer entièrement l'automatisme mémorisé dans le module. Le module reste dans le "mode édition" jusqu'à ce qu'il reçoive la commande CLOSE_SEQ	
Exemple:	OPEN_SEQ Ouverture du "mode édition" pour tous les modules présents. NOP NOP Les commandes sont mémorisées dans le séquenceur NOP CLOSE_SEQ Ferme le "mode édition".	



4.14. OPTIMIZED_CURRENT

Syntaxe:	[@]OPTIMIZED_CURRENT paramètre
Mnémonique:	OCU
Paramètre:	ON (mode optimisé) ou OFF (mode nominal)
Description:	Active ou désactive le mode "courant optimisé". Ce mode permet d'adapter le courant moteur au couple réellement nécessaire. Cette option permet de diminuer les pertes thermiques, notamment lorsque le moteur ne tourne pas en continu. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Ce mode nécessite l'autocommutation, il n'a aucune action si le BMAC est configuré en boucle ouverte.
Exemple:	06OPTIMIZED_CURRENT ON Active le mode optimisé du module d'adresse 06.

4.15. POWER

Syntaxe:	[@]POWER paramètre
Mnémonique:	POW
Paramètre:	ON pour placer le moteur sous tension OFF pour couper totalement la puissance moteur SC pour placer les bobines du moteur en court circuit
Description:	Met sous puissance ou hors puissance le moteur. La mise hors puissance du moteur implique un courant et donc un couple nul. La mise en court-circuit permet au moteur d'exercer un couple résistant s'il se trouve entraîné à faible vitesse.
Remarques:	La commande POWER ON est implicitement réalisée à chaque commande de mouvement.
Exemple:	02POWER ON Mise sous puissance du moteur d'adresse 2.



4.16. READ

Syntaxe: @READ paramètre

Mnémonique: REA

Paramètre: Nom de la variable à relire (toutes les variables sont décrites dans la suite du document)

Description: Relit une variable du module et renvoie sa valeur sur la liaison série vers le contrôleur (PC ou automate).

Il est possible de préciser le format de relecture en faisant précéder le nom de la variable à relire de 'H' pour une relecture en hexadécimal ou de 'B' pour une relecture en binaire. H et B peuvent être indifféremment en majuscule ou minuscule.

Remarques: Les relectures hexadécimales ou binaires renvoient 4 octets signés.

La commande READ ne peut être utilisée qu'en commande directe. Son utilisation dans une séquence provoque une erreur.

La chaîne totale renvoyée par le module est composée de l'adresse du module, du mnémonique de la variable relue, du caractère '=' et de la valeur de la variable dans le format adéquat.

Exemple: 00READ #POSITION
renvoie la position courante du module d'adresse 00:
réponse du module: 00#POS=12345

01READ h#OUTPUT
renvoie l'état des sorties du module d'adresse 01:
réponse du module: 01#OUT=h00000007
=> les sorties OUT1, OUT2 et OUT3 sont actives, OUT4 est inactive (ou le contraire si la polarité est inversée par la commande INVERSE_POLARITY).

02READ b#INPUT
renvoie l'état des entrées logiques du module d'adresse 02:
réponse du module: 02#INP=b00000000 00000000 00000000 00001010
=> les entrées IN2 et IN4 sont actives, les entrées IN1, IN3, IN5 et IN6 sont inactives (ou le contraire si la polarité est inversée par la commande INVERSE_POLARITY).



4.17. READ_SEQ

Syntaxe:	[@]READ_SEQ paramètre
Mnémonique:	RSE
Paramètre:	adresse de la ligne à relire [1 ; 500]
Description:	Relecture d'une ligne mémorisée dans le module. Cette commande sert à la mise au point de la séquence. Attention, elle renvoie une écriture qui peut être légèrement différente de la forme entrée par l'utilisateur.
Exemple:	00READ_SEQ 3 Relecture de la ligne 3 (exemple de réponse: 00:003 MTO +2000, la ligne 3 du module zéro contient la commande "MOVE_TO 2000")

4.18. REFERENCE

Syntaxe:	[@]REFERENCE paramètre
Mnémonique:	REF
Paramètre:	ON (mode actif sur entrée IN5) INDEX (mode actif sur index codeur) OFF (mode inactif)
Description:	Active ou désactive un mode « référence ». Dans tous les cas, le bit #STATUS.30 indique qu'un des modes de prise de référence est actif. Il retourne à 0 dès que la référence est effectuée. REFERENCE ON Ce mode permet d'initialiser la position sur un capteur de position connecté à l'entrée IN5. Les variables #POSITION et #ENCODER sont forcées à zéro si le moteur tourne dans le sens positif et qu'un front montant (transition 0 à 1) est détecté sur l'entrée IN5. Attention, l'entrée IN5 doit être positionnée à 1 pendant au moins 1ms lors du mouvement de recherche de référence. REFERENCE INDEX Ce mode permet d'initialiser la position sur l'entrée index du codeur. Les variables #POSITION et #ENCODER sont forcées à zéro dès qu'un front montant (transition 0 à 1) est détecté sur l'index codeur. Attention, cette impulsion sur l'index codeur doit durer au minimum 50µs afin d'être prise en compte. ⇒ effectuer le mouvement de recherche de référence à 600tours/min maximum pour un codeur 500points dont l'index serait positionné sur 1/2000 tour ⇒ effectuer le mouvement de recherche de référence à 30tours/min maximum pour un codeur 10000points dont l'index serait positionné sur 1/40000 tour
Exemple:	06REFERENCE INDEX Active le mode référence sur l'index codeur du module d'adresse 06.



4.19. REQUEST_VERSION

Syntaxe: [@] REQUEST_VERSION

Mnémonique: RV OU RVE

Paramètre: Aucun

Description: Relit une chaîne de caractère interne descriptive du module.
Permet l'identification de chaque module.
La réponse du module est de la forme :
@EV vV.RR CODE "MIDI-INGENIERIE_produit_ n°série _ date de fabrication _date
révision" BOOT:vX.Y
avec V = version logicielle, RR = révision logicielle,
CODE = identification du logiciel
Les dates sont au format jj/mm/aa.

Détail du code logiciel:

CODE	LOGICIEL
9189	RS485
J189	horloge & sens
K189	CAN

Exemple: 04 REQUEST_VERSION
pourrait renvoyer, par exemple, la chaîne suivante:
04EV v3.07 9189 "MIDI-INGENIERIE_BMAC _9189-0014_25/09/11_12/10/11" BOOT:v2.0
Il s'agit donc du module BMAC n° 9189-0014 fabriqué le 25/09/11 et révisé le 12/10/11. Il est programmé avec le logiciel standard 9189 en version 3.07



4.20. SET_ADDRESS

Mnémonique	SAD
Paramètre:	Nouvelle adresse du module [0 ; 63] Valeur usine = 0.
Description:	Change l'adresse d'un module. Le module répondra uniquement aux commandes qui sont adressées à sa nouvelle adresse. Les commandes générales (non adressées) seront exécutées sans réponse sauf pour le module d'adresse 0. L'adresse est mémorisée et se conserve en cas de coupure d'alimentation.
Remarques:	Deux modules ne peuvent pas avoir la même adresse (risque de conflit). C'est à l'utilisateur de gérer l'adressage des modules dans un ordre qui convient. Il est interdit de renommer un module si la nouvelle adresse correspond à celle d'un module existant. Ne pas envoyer une commande SET_ADDRESS comme une commande générale, sans préciser l'adresse du module visé. ATTENTION: en sortie d'usine, tous les modules ont pour adresse 0. Il convient alors de les connecter un par un sur la ligne de communication pour leur affecter une adresse différente.
Exemple:	04SET_ADDRESS 3 Le module d'adresse 4 devient le module d'adresse 3 04SET_ADDRESS 0 Le module d'adresse 4 devient le module d'adresse 0. Il va à présent répondre aux commandes générales. SET_ADDRESS 6 Attention, tous les modules sont renommés à l'adresse 6. Il y a conflit si plusieurs modules sont présents.

4.21. SET_BAUDRATE

Mnémonique	SBA
Paramètre:	définit la vitesse de communication en bauds (bits par secondes) 115200, 38400, 19200, 9600 (toutes les autres valeurs sont sans effet) Valeur usine = 38400.
Description:	Spécifie la vitesse de communication entre l'hôte (PC ou automate) et les modules. Le baudrate est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Il est déconseillé de mettre des baudrates différents pour chaque module. La commande SET_BAUDRATE doit donc être envoyée en commande globale. La valeur du baudrate est par défaut de 38400bauds. Le changement de baudrate est pris en compte dès la commande suivante.
Exemple:	SET_BAUDRATE 38400 Tous les modules communiquent maintenant à 38400bauds



4.22. S_CURVE

Syntaxe:	[@]S_CURVE paramètre
Mnémonique:	SCU
Paramètre:	ON (rampes en S) ou OFF (rampes trapézoïdales)
Description:	Active ou désactive les rampes en S. Par rapport aux rampes trapézoïdales, les rampes en S minimisent les à-coups d'accélération (donc de couple) qui peuvent intervenir au début et à la fin des rampes de vitesse. Le mouvement est alors plus fluide. Attention l'accélération instantanée est plus importante au milieu de la rampe. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Pour interdire les rampes de vitesse et se ramener à un fonctionnement dit "de start-stop", il suffit de définir un temps d'accélération nul (#ACCEL_TIME:=0)
Exemple:	05S_CURVE OFF Le module d'adresse 5 utilisera des rampes trapézoïdales pour tous les prochains mouvements.

4.23. SOFT_ENDS

Syntaxe:	[@]SOFT_ENDS paramètre
Mnémonique:	SEN
Paramètre:	ALL (butées actives) ou OFF (butées inactives)
Description:	Active ou désactive les butées virtuelles. La butée #POSITIVE_END arrête et empêche les mouvements dans le sens positif au-delà de sa valeur. La butée #NEGATIVE_END arrête et empêche les mouvements dans le sens négatif en deçà de sa valeur. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Exemple:	02SOFT_ENDS OFF Désactive les butées virtuelles du module d'adresse 02.



4.24. START_SEQ

Syntaxe: `[@]START_SEQ [paramètre]`

Mnémonique: `SSE`

Paramètre: Numéro de la première ligne à exécuter (1 à défaut).

Description: Exécution du séquenceur à partir de la ligne précisée en argument.
Si aucun argument n'est précisé, le séquenceur commence à la ligne 1.

Exemple: `START_SEQ 15`
Lancement du séquenceur à la ligne 15 pour tous les modules présents.

`02START_SEQ`
Lancement du séquenceur à la ligne 1 pour le module 2.

4.25. STEP

Syntaxe: `[@]STEP [nl]`

Mnémonique: `STE`

Paramètre: `nl` : numéro de la ligne de séquence à exécuter

Description: Si `nl > 0` le module exécute la commande décrite par la ligne de séquence donnée.
Si `nl` n'est pas donné ou nul, le module exécute la ligne de séquence qui suit la dernière ligne réalisée.

Exemple: `04STEP 25`
Le module 4 effectue la commande placée à la ligne 25.



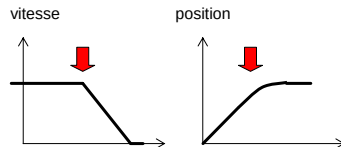
4.26. STOP

Syntaxe: `[@]STOP [paramètre]`

Mnémonique: `STO`

Paramètre: Aucun, MOUV ou SEQ

Description: Le module s'arrête en passant de la vitesse courante à une vitesse nulle en un temps défini proportionnellement au paramètre `#DECEL_TIME`.



La commande `STOP` sans paramètre arrête à la fois le mouvement en cours et la séquence en cours.

Il est possible de n'arrêter que le mouvement (commande `STOP MOUV`) en poursuivant l'exécution d'une séquence.

Il est également possible de n'arrêter que la séquence (commande `STOP SEQ`) et de poursuivre le mouvement en cours.

NB : dans le cas d'un mouvement en mode interpolé, la commande `STOP` est équivalente à la commande `HALT` (arrêt immédiat). Dans les deux cas la FIFO d'interpolation est vidée.

Exemple: `04STOP`

l'axe d'adresse 04 décélère jusqu'à l'arrêt. Si une séquence était en cours d'exécution, elle est arrêtée.

`03STOP MOUV`

l'axe d'adresse 03 décélère jusqu'à l'arrêt. Si une séquence était en cours d'exécution, elle continue.



4.27. SYNCHRO

Syntaxe: SYNCHRO [paramètre]

Mnémonique: SYN

Paramètre: ON, OFF, TOP ou INTERPOL

Description: Ce jeu de commandes permet un démarrage synchronisé de l'ensemble des axes.

SYNCHRO ON passe en mode synchrone (inutile en cas d'interpolation)

SYNCHRO OFF quitte le mode synchrone

SYNCHRO TOP lance le mouvement (hors interpolation)

SYNCHRO INTERPOL lance le mouvement d'interpolation

Note : si nécessaire, le moteur est automatiquement mis sous tension au démarrage du mouvement. Pour une meilleure synchronisation il est toutefois conseillé de le mettre sous puissance au préalable avec la commande POWER ON.

Exemple: 1) mouvement non interpolé

SYNCHRO ON ; entrée en mode synchrone sur tous les axes

00MOVE_SPEED 4000

03MOVE_SPEED -12500

SYNCHRO TOP ; départ des axes 0 et 3

SYNCHRO OFF ; quitte le mode synchrone sur tous les axes (facultatif)

2) interpolation (voir commande MOVE_INTERPOL chapitre 4.4)

4.28. WAIT

Syntaxe: WAIT paramètre

Mnémonique: WAI

Paramètres: temps d'attente en millisecondes [-3600000 ; 3600000]

Description: Temporisation. L'exécution du séquenceur est suspendue jusqu'à ce que le délai soit écoulé.

Si le temps d'attente spécifié est nul ("WAIT 0"), l'exécution est suspendue jusqu'à la fin du mouvement en cours (asservissement de position inclus).

Si le temps d'attente spécifié est négatif ("WAIT -1000"), l'exécution est suspendue jusqu'à la fin du mouvement en cours avec un timeout équivalent à la valeur absolue du paramètre.

Exemple: WAIT 2000

Le séquenceur attend 2 secondes avant de passer à la ligne suivante.

WAIT 0

Le séquenceur attend que le mouvement en cours soit terminé avant de passer à la ligne suivante.

WAIT -2000

Le séquenceur attend au maximum 2 secondes que le mouvement en cours soit terminé avant de passer à la ligne suivante.



5. Variables internes

5.1. Généralités

La forme générale des commandes d'accès aux variables est la suivante:

[@]#VARIABLE[.BIT]:= [- ! H B]VALEUR pour une écriture

[@]READ [H B]#VARIABLE pour une lecture.

La réponse à la relecture d'une variable est du type:

@MNEMONIQUE=VALEUR

Toutes les variables du BMAC sont mémorisées dans un format "32 bit signés".

5.1.1. Format décimal

Le format par défaut est le format décimal.

Le caractère "+" devant les valeurs positives ou nulles est facultatif. Il est en revanche toujours spécifié dans les relectures.

Exemples:

00#ACCEL_TIME:=123

00#V20:=-40

00READ #ACCEL_TIME → 00#ATI=+123

5.1.2. Format hexadécimal

En faisant précéder le nom de la variable du caractère "H", on peut la saisir et la relire au format hexadécimal.

Les valeurs négatives sont notées en "complément à deux" sur quatre octets (exemple: -10 se note HFFFFFFF6).

Il est facultatif de préciser les zéros non significatifs. La relecture se fait toujours sur quatre octets.

Exemples:

00#ACCEL_TIME:=H100

00#V20:= HFFFFFFD8

00READ h#ACCEL_TIME → 00#ATI=h00000100

5.1.3. Format binaire

En faisant précéder le nom de la variable du caractère "B", on peut la saisir et la relire au format binaire.

Les valeurs négatives sont notées en "complément à deux" sur quatre octets.

Il est facultatif de préciser les zéros non significatifs. La relecture se fait toujours sur trente-deux bits, chaque octet étant séparé par un espace.

Exemples:

00#ACCEL_TIME:=B1100100

00READ b#ACCEL_TIME → 00#ATI=b00000000 00000000 00000000 01100100

5.1.4. Opposé et complément

En faisant précéder le nom de la variable du caractère "-" ou "!", on accède respectivement à l'opposé ou au complément binaire de sa valeur.

Cette notation est particulièrement utile dans les opérations ou les tests en séquence.

Exemples:

00#V21:=-#V1

00#V23:=#V15 & !#STATUS

00MOVE_TO -#POSITION



5.1.5. Format bit-à-bit

Pour accéder à un seul bit d'une variable, on peut faire suivre le nom de la variable d'un point et du numéro du bit (de 1 à 32, 1 étant le bit de poids faible et 32 étant le bit de poids fort).

Le bit de la variable joue alors le rôle d'une variable à part entière. Sa valeur peut être 1 ou 0.

On peut également utiliser cette notation dans un calcul ou un test.

Exemples:

```
00#V10.3:=1 ;met à 1 le 3ème bit de #V10
00#V10.12:=0 ;met à 0 le 12ème bit de #V10
00READ #STATUS.5 → 00#STA.5=1 ;relit le 5ème bit de #STATUS
00#V12.1:=#STATUS.12 & #ERROR.2 ;ET logique entre deux bits
00IF #V1.24 = 1 JUMP 3 ;test d'un bit en séquence
```

5.1.6. Opérations sur variables

Il est possible d'effectuer un certain nombre d'opérations et de tests sur les variables.

La syntaxe est la suivante:

[@]#VARIABLE:=OPERANDE1 OPERATEUR OPERANDE2

Il est nécessaire d'inclure au moins un espace avant et après l'opérateur.

Les opérandes peuvent être soit une variable interne, soit une constante (éventuellement notée en binaire ou en hexadécimal comme décrit précédemment).

L'opérateur est un des opérateurs suivants:

opérateur	opération	exemple
+	addition	#V1:=#POSITION + 12000
-	soustraction	#LOW SPEED:=#HIGH SPEED - #V3
*	multiplication	#V2:=3 * -#SUPPLY VOLTAGE
/	division entière	#M3:=#POSITION / 10000
&	ET bit à bit	#V2:=#STATUS & H00000200
	OU bit à bit	#M1:=!#V4 H00000240
>	test supérieur	#V1:=#POSITION > 123
<	test inférieur	#V1:=#POSITION < 123
>=	test supérieur ou égal	#V1:=#POSITION >= 123
<=	test inférieur ou égal	#V1:=#POSITION <= 123
!=	test différent	#V2:=#POSITION != 0

Dans le cas des opérateurs de test (<,>,>=,<=,!), le résultat de l'opération vaut 1 si le test est VRAI et 0 si le test est FAUX. Ces opérateurs servent en particulier aux sauts conditionnels en séquence (IF...JUMP).

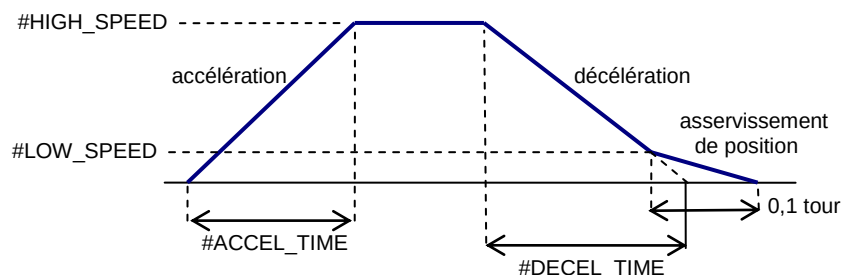


5.1. #ACCEL_TIME, #DECEL_TIME

Mnémonique #ATI, #DTI

Paramètre: temps d'accélération et de décélération, exprimés en millisecondes. [0 ; 12000]

Description: #ACCEL_TIME: Temps nécessaire pour atteindre la vitesse de consigne
#DECEL_TIME: Temps d'arrêt lorsqu'on est à la vitesse de consigne.



Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.

Remarques: La variable #ACCEL_TIME correspond au temps qu'il faut pour passer d'une vitesse nulle (module arrêté) à la vitesse de consigne #HIGH_SPEED.

La variable #DECEL_TIME correspond au temps qu'il faut pour passer de la vitesse de consigne #HIGH_SPEED à une vitesse nulle.

Dans le cas de mouvements à des vitesses différentes de la vitesse de consigne, le module recalcule automatiquement les temps de rampe proportionnellement à l'écart entre la vitesse cible et la vitesse courante. Ceci permet de conserver des accélérations constantes.

Le temps d'accélération effectif correspond donc à:

$$\#ACCEL_TIME \times \frac{|Vitesse\ Cible - Vitesse\ Courante|}{\#HIGH_SPEED}$$

De même, le temps de décélération effectif est de :

$$\#DECEL_TIME \times \frac{|Vitesse\ Cible - Vitesse\ Courante|}{\#HIGH_SPEED}$$

Exemple:

#ACCEL_TIME:=1000 (1s)

#HIGH_SPEED:=60000 (600tr/min)

MOVE_SPEED 30000 → le temps d'accélération sera de 0.5s

Note: Ces paramètres règlent indirectement l'accélération du module.

Exemple: 00#ACCEL_TIME:=1000

Le module mettra 1 seconde pour passer d'une vitesse nulle à #HIGH_SPEED.



5.2. #CAPTURE (lecture seule)

Mnémonique #CAP

Paramètre: Valeur de la variable #POSITION lors du dernier front montant de IN5

Description: A chaque front montant (0→1) de l'entrée logique IN5, la position courante est mémorisée dans la variable #CAPTURE.

Exemple: 03READ #CAPTURE → 03#CAP=27895

Le module d'adresse 3 était à la position 27895 lors du dernier front montant de IN5.

5.3. #CURRENT_RATIO

Mnémonique #CRA

Paramètre: Courant moteur, exprimé en pour mille du courant nominal #RATED_CURRENT [0 ; 1000]

Le courant injecté exprimé en mA vaut :
 $\#RATED_CURRENT * \#CURRENT_RATIO / 1000$

Description: Cette variable permet de régler le courant injecté dans le moteur. Une valeur élevée donne une réserve de couple, une valeur plus faible permet de réduire la consommation électrique et l'échauffement du moteur.

Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.

Remarques: Attention, dans un premier temps il est nécessaire de programmer le courant nominal moteur via la variable #RATED_CURRENT exprimée en mA (valeur maximum 2500 sur BMAC et 7000 sur BMAC-H).

Ensuite le courant souhaité est défini via la variable #CURRENT_RATIO (0 à 1000) exprimée en pour mille de #RATED_CURRENT.

Les performances maximales du moteur (vitesse de rotation et puissance) nécessitent une valeur #CURRENT_RATIO réglée à 1000.

La résolution du courant réellement généré est de 25mA sur BMAC et 70mA sur BMAC-H.

En fonctionnement autocommuté, un mode optimisé (OPTIMIZED_CURRENT) permet d'injecter dans le moteur le courant réellement nécessaire afin de minimiser les pertes thermiques.

Exemple: 00#CURRENT_RATIO:=750

Le moteur du module 0 dispose de 750/1000 soit 75% de son courant nominal.



5.4. #CPU_TEMPERATURE (lecture seule)

Mnémonique	#CTE
Paramètre:	Mesure de la température interne du processeur du module (unité = 0,1°C)
Exemple:	02READ #CPU_TEMPERATURE → 02#CTE=520 Le module d'adresse 2 a une température de 52°C.

5.5. #DRIVER_TEMPERATURE (lecture seule)

Mnémonique	#DTE
Paramètre:	indique la température de l'étage de puissance du BMAC (unité = 0,1°C)
Exemple:	02READ #DRIVER_TEMPERATURE → 02#DTE=650 L'étage de puissance du BMAC d'adresse 2 a une température de 65°C.

5.6. #ELECTRICAL_POSITION (lecture seule)

Mnémonique	#EPO
Paramètre:	Accès en lecture seule à la position électrique de consigne. [0 ; 199]
Description:	#ELECTRICAL_POSITION indique la position électrique de consigne exprimée en micropas avec une résolution de 50µpas/pas. Les courants de consigne en mA sont donnés par les formules : $I_{\text{phaseA}} = I_{\text{crête}} * \cos (2\pi * (\#ELECTRICAL_POSITION / 200))$ $I_{\text{phaseB}} = I_{\text{crête}} * \sin (2\pi * (\#ELECTRICAL_POSITION / 200))$ avec $I_{\text{crête}} = \sqrt{2} * \#RATED_CURRENT * \#CURRENT_RATIO$
Remarques:	Cette variable est mémorisée à la coupure d'alimentation et restituée à la mise sous tension. Les pas entiers correspondent aux valeurs 0, 50, 100 et 150 de la variable #ELECTRICAL_POSITION.
Exemple:	00READ #ELECTRICAL_POSITION → 00#EPO=0 ⇒ Le courant phase A est à sa valeur maximum, le courant phase B est à 0.



5.7. #ENCODER

Mnémonique	#ENC
Paramètre:	Nouvelle position de l'axe en incrément codeur (1/2000 de tour si codeur 500points) [-2 147 483 648 ; + 2 147 483 647]
Description:	Permet d'initialiser le suivi de position codeur. On peut, en particulier, utiliser cette commande pour définir l'origine d'un système mécanique en donnant 0 pour paramètre.
Exemple:	04#ENCODER:=2000 Initialise le suivi codeur à +2000.

5.8. #ERROR

Mnémonique	#ERR																																						
Paramètre:	Aucun																																						
Description:	La relecture des erreurs du BMAC renvoie une valeur dont les bits représentent l'état du module:																																						
	<table><tr><td rowspan="10">poids fort</td><td>bit 32</td><td></td></tr><tr><td>bit 31</td><td>Erreur de l'amplificateur de puissance du BMAC-H (alimentation < 20V ou disjonction court-circuit ou différentielle ou thermique)</td></tr><tr><td>bit 30</td><td>Disjonction différentielle (court-circuit entre phase et 0V)</td></tr><tr><td>bit 29</td><td>Disjonction court-circuit d'une phase moteur</td></tr><tr><td>...</td><td></td></tr><tr><td>bit 25</td><td>Disjonction thermique étage de puissance > 85°C</td></tr><tr><td>...</td><td></td></tr><tr><td>bit 12</td><td>NonDef: Commande ou Variable non définie (syntaxe)</td></tr><tr><td>...</td><td></td></tr><tr><td>bit 9</td><td></td></tr><tr><td rowspan="9">poids faible</td><td>bit 8</td><td>Calcul (division par zéro, etc....)</td></tr><tr><td>bit 7</td><td>Limite (paramètre de commande hors limite)</td></tr><tr><td>bit 6</td><td></td></tr><tr><td>bit 5</td><td>Surtension</td></tr><tr><td>bit 4</td><td>Sous-tension</td></tr><tr><td>bit 3</td><td>Court-circuit (détail dans les bits 29 et 30)</td></tr><tr><td>bit 2</td><td>Disjonction thermique (étage de puissance si le bit 25 est activé, processeur > 120°C sinon)</td></tr><tr><td>bit 1</td><td></td></tr></table>	poids fort	bit 32		bit 31	Erreur de l'amplificateur de puissance du BMAC-H (alimentation < 20V ou disjonction court-circuit ou différentielle ou thermique)	bit 30	Disjonction différentielle (court-circuit entre phase et 0V)	bit 29	Disjonction court-circuit d'une phase moteur	...		bit 25	Disjonction thermique étage de puissance > 85°C	...		bit 12	NonDef: Commande ou Variable non définie (syntaxe)	...		bit 9		poids faible	bit 8	Calcul (division par zéro, etc....)	bit 7	Limite (paramètre de commande hors limite)	bit 6		bit 5	Surtension	bit 4	Sous-tension	bit 3	Court-circuit (détail dans les bits 29 et 30)	bit 2	Disjonction thermique (étage de puissance si le bit 25 est activé, processeur > 120°C sinon)	bit 1	
poids fort	bit 32																																						
	bit 31		Erreur de l'amplificateur de puissance du BMAC-H (alimentation < 20V ou disjonction court-circuit ou différentielle ou thermique)																																				
	bit 30		Disjonction différentielle (court-circuit entre phase et 0V)																																				
	bit 29		Disjonction court-circuit d'une phase moteur																																				
	...																																						
	bit 25		Disjonction thermique étage de puissance > 85°C																																				
	...																																						
	bit 12		NonDef: Commande ou Variable non définie (syntaxe)																																				
	...																																						
	bit 9																																						
poids faible	bit 8	Calcul (division par zéro, etc....)																																					
	bit 7	Limite (paramètre de commande hors limite)																																					
	bit 6																																						
	bit 5	Surtension																																					
	bit 4	Sous-tension																																					
	bit 3	Court-circuit (détail dans les bits 29 et 30)																																					
	bit 2	Disjonction thermique (étage de puissance si le bit 25 est activé, processeur > 120°C sinon)																																					
	bit 1																																						

L'acquittement d'une erreur se fait directement en écrivant 0 dans #ERROR

Exemple:

```
01READ b#ERROR
→ 01#ERR=b00000000 00000000 00000000 00010000
```

Relecture de l'état du module d'adresse 1 au format binaire (surtension).



5.9. #HIGH SPEED

Mnémonique	#HSP
Paramètre:	Vitesse de consigne en 100 ^{ème} tour/min [0 ; 400000]
Description:	Vitesse de consigne de tous les déplacements en mode position (MOVE_TO et MOVE_ON) et la vitesse maximale des déplacements en mode vitesse (MOVE_SPEED).
	Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Lors de petits déplacements, la vitesse de consigne n'est pas forcément atteinte.
Exemple:	04#HIGH_SPEED:=20000 Lors des prochains mouvements en mode position, les mouvements de l'axe 04 se feront à 200tr/min.

5.10. #INPUT (lecture seule)

Mnémonique	#INP
Paramètre:	Valeur qui représente bit à bit l'état des entrée logiques (en binaire actif à 1, inactif à 0)
Description:	Relit les entrées logiques IN1 (bit de poids faible) à IN6 ou IN10 suivant les modules (bit de poids fort). Si la polarité des entrées est inversée (commande <code>INVERSE_POLARITY</code> représentée par le bit <code>#STATUS.13</code>), l'état logique de <code>#INPUT</code> est inversé par rapport à l'état physique des entrées.
Exemple:	BMAC d'adresse 05 (avec <code>INVERSE_POLARITY OFF</code>) <pre> 05READ #INPUT → 05#INP=19 05READ h#INPUT → 05#INP=h13 05READ b#INPUT → 05#INP=b000000000000000000000000010011 19 (décimal) = 0 0 0 1 0 0 1 1 (binaire) OFF OFF OFF ON OFF OFF ON ON IN8 IN7 IN6 IN5 IN4 IN3 IN2 IN1 </pre>

5.11. #INPUT_ANALOG (lecture seule)

Mnémonique	#IAN
Paramètre:	Tension de l'entrée analogique différentielle en mV.
Exemple:	02READ #INPUT_ANALOG → 02#IAN=-3200 Le module d'adresse 2 a une tension de -3,2V sur l'entrée analogique.



5.12. #INTERPOL_COUNT (lecture seule)

Mnémonique	#ICO
Paramètre:	Nombre de segments présents en mémoire FIFO [0 ; #INTERPOL_FIFOSIZE]
Description:	Cette variable en lecture seule indique le nombre de segments mémorisés en mémoire FIFO. Lorsque la FIFO est vide, #INTERPOL_COUNT=0 et le mouvement s'arrête après exécution du dernier segment. (se référer à la commande MOVE_INTERPOL pour un descriptif détaillé du mode "position interpolée")
Remarques:	L'espace disponible en FIFO est donné par la différence #INTERPOL_FIFOSIZE - #INTERPOL_COUNT
Exemple:	02READ #INTERPOL_COUNT pourrait renvoyer 02#ICO=23 => 23 segments actuellement mémorisés dans l'axe n°2.

5.13. #INTERPOL_FIFOSIZE

Mnémonique	#IFI
Paramètre:	Dimension de la mémoire FIFO [1 ; 64]
Description:	Paramètre la taille de la mémoire FIFO entre 1 et 64 segments. (se référer à la commande MOVE_INTERPOL pour un descriptif détaillé du mode "position interpolée")
Remarques:	Une taille importante permet de tolérer des temps de latence dans la transmission des messages de type MOVE_INTERPOL. Une taille réduite sera utile pour avoir un faible temps de réaction dans le cas d'une trajectoire non déterminée à l'avance, par exemple en poursuite asservie sur une cible.
Exemple:	#INTERPOL_FIFOSIZE :=64 ; valeur maximum, appliquée à tous les axes



5.14. #INTERPOL_MODE

Syntaxe: `[@]#INTERPOL_MODE paramètre`

Mnémonique: #IMO

Paramètre: 0 (SimpleSYNC) ou -1 (UltraSYNC)

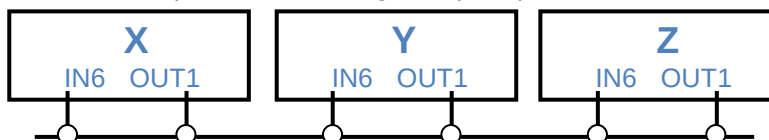
Description: Cette variable permet de choisir entre les deux types de synchro disponibles pour le mode "Trajectoires Interpolées".

Mode SimpleSYNC (#INTERPOL_MODE:=0)

Dans ce mode, la synchronisation multi-axe est assurée par le démarrage simultané du mouvement sur tous les axes (envoi de la commande globale `SYNCHRO INTERPOL`). La durée des segments est ensuite assurée sur chaque axe indépendamment au moyen d'un timer interne de précision.

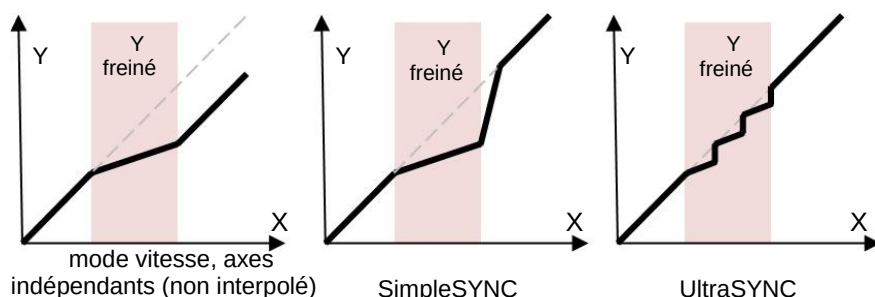
Mode UltraSYNC (#INTERPOL_MODE:=-1)

Dans ce mode, la synchronisation multi-axe est assurée par un chaînage des entrées-sorties de tous les axes. En connectant l'entrée IN6 et la sortie OUT1 de tous les axes du réseau, ils se synchronisent automatiquement, et si un des axes est freiné ou ralenti (frottements, point dur, etc...) les autres axes attendent qu'il ait fini son segment pour passer au suivant.



Exemple:

Exemple dans le cas d'un système 2 axes X-Y:





5.15. #INTERPOL_TIME

Mnémonique	#ITI
Paramètre:	Durée des segments d'interpolation en ms [2 ; 138]
Description:	Cette durée sera appliquée pour tous les segments ultérieurement transmis via MOVE_INTERPOL. (se référer à la commande MOVE_INTERPOL pour un descriptif détaillé du mode "position interpolée") En général, il convient de travailler avec une durée de segment constante et identique pour tous les axes.
Remarques:	Si la durée doit varier au long de la trajectoire, il convient d'avoir des durées identiques pour les segments de même niveau chronologique sur l'ensemble des axes afin de conserver le synchronisme.
Exemple:	#INTERPOL_TIME:=50 ; tous les segments mémorisés par la suite auront une durée de consigne de 50ms sur tous les axes

5.16. #LINE_DELAY

Mnémonique	#LDE
Paramètre:	Délai de retournement de ligne en µs [100 ; 3000] Valeur usine = 3000.
Description:	Temps de retournement de la ligne RS485 entre la fin de la commande du contrôleur (PC ou automate) et le début de la prise de ligne du BMAC. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Il est déconseillé de mettre des temps de retournement de ligne différents pour chaque module.
Exemple:	#LINE_DELAY:=1000 Les modules attendent 1ms après la fin de la commande pour prendre la ligne et répondre.



5.17. #LINE

Mnémonique #LIN

Paramètre: Ligne du séquenceur en cours d'exécution [0 ; 500]

Description: La variable #LINE représente la ligne qui est en train d'être exécutée par le module.

Remarques: Lorsque le séquenceur n'est pas démarré, la variable #LINE vaut zéro.
La mise à zéro de cette variable (#LINE:=0) arrête le séquenceur.

Exemple: READ #LINE → 00#LIN:=182

Le module d'adresse 0 est en train d'effectuer la ligne 182 de la séquence.

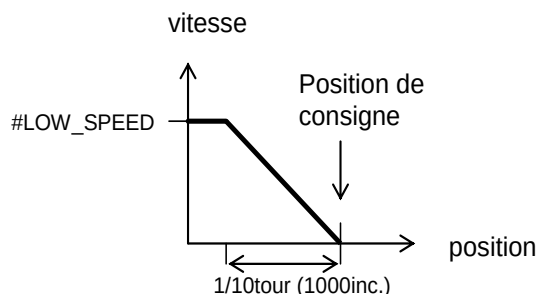
5.18. #LOW_SPEED

Mnémonique #LSP

Paramètre: Vitesse d'approche en 100^{ème} tour/min
[0 ; 400000]

Description: Vitesse d'approche utilisée en fin de déplacement pour la gestion du positionnement en mode position (MOVE_TO et MOVE_ON). Cette vitesse est également utilisée pour revenir à la position de consigne si l'axe s'en écarte.

Le profil de vitesse utilisé lors de l'asservissement de position est défini uniquement par la valeur #LOW_SPEED:



Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.

Remarques: Cette variable permet de régler le temps de réponse de l'asservissement de position et sa stabilité. Le temps de réponse diminue lorsque #LOW_SPEED augmente. Par contre si #LOW_SPEED est trop élevée il y a des risques d'oscillations notamment si le moteur est mécaniquement chargé par une forte inertie avec peu de frottements.

Exemple: 04#LOW_SPEED:=2000

Lors des prochains mouvements en mode position, l'asservissement de position en fin de mouvement s'effectuera à 20tr/min maximum.



5.19. #M1 à #M8

Mnémonique	#M1 à #M8
Paramètre:	Valeur signée comprise entre -2^{31} et $2^{31} - 1$ [-2 147 483 648 ; + 2 147 483 647]
Description:	Les variables #M1 à #M8 ont le même rôle que les variables #Vn, mais leur contenu est mémorisé à la mise hors tension du module, ce qui peut servir à la sauvegarde de paramètres.
Exemple:	#M2:=H100 Mémorise la valeur hexadécimale 0x100 dans la variable #M2 de tous les modules.

5.20. #NEGATIVE_END

Mnémonique	#NEN
Paramètre:	Position de la butée virtuelle négative (10^{-4} tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	Les mouvements qui dépassent cette consigne sont arrêtés, un bit d'erreur est positionné, et seuls les mouvements dans le sens positif (horaire) sont autorisés. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	La valeur de la position des butées software n'est pas modifiée lorsque la position de référence #POSITION est modifiée.
Exemple:	03#NEGATIVE_END:=-80000 Le module d'adresse 3 ne pourra pas aller au-delà de 8 tours en sens négatif de la position d'origine.

5.21. #ON_RESET

Mnémonique	#ORE
Paramètre:	Numéro de la ligne de démarrage [0 ; 500]
Description:	La variable #ON_RESET représente la ligne qui doit être exécutée à chaque mise sous tension ou reset du module. Pour désactiver le démarrage automatique du séquenceur, entrer #ON_RESET:=0.
Exemple:	#ON_RESET:=1 Le module lance le séquenceur à partir de la ligne 1 à chaque reset.



5.22. #OUTPUT

Mnémonique	#OUT
Paramètre:	Valeur qui représente bit à bit l'état des sorties logiques
Description:	Sorties logiques du module. Si la polarité des sorties est inversée (commande <code>INVERSE_POLARITY</code> représentée par le bit <code>#STATUS.14</code>), l'état logique de <code>#OUTPUT</code> est inversé par rapport à l'état physique des sorties.
Remarques:	Les 32 bits de la variable <code>#OUTPUT</code> sont accessibles, mais seuls les bits de poids faible correspondent à des sorties logiques physiquement accessibles (par exemple <code>#OUTPUT.1</code> à <code>#OUTPUT.4</code> pour les modules dotées de 4 sorties, <code>#OUTPUT.1</code> à <code>#OUTPUT.8</code> pour les modules disposant de 8 sorties). Les sorties <code>OUT1</code> et <code>OUT2</code> peuvent être affectées aux fonctions Busy et Défaut en utilisant la variable <code>#OUTPUT_CONFIG</code> . L'état de la sortie logique correspondante est alors imposé par l'état du module et n'est plus modifié par <code>#OUTPUT</code> .
Exemple:	<code>00#OUTPUT:=4</code> Active la sorties <code>OUT3</code> , désactive les autres sorties 4 (décimal) = 0 0 0 0 0 1 0 0 (binaire) OUT8 OUT7 OUT6 OUT5 OUT4 OUT3 OUT2 OUT1

5.23. #OUTPUT_CONFIG

Mnémonique	#OCO
Paramètre:	Valeur qui représente bit à bit l'état des sorties logiques
Description:	Multiplexage des sorties logiques. Les sorties logiques 1 et 2 peuvent être utilisées en sortie standard ou en fonctions évoluées: OUT1 peut représenter l'information BUSY OUT2 peut représenter l'information DEFAULT Si le bit de <code>#OUTPUT_CONFIG</code> est à 0, la sortie est utilisée en "sortie utilisateur". L'état est imposé par la variable <code>#OUTPUT</code> . Si le bit de <code>#OUTPUT_CONFIG</code> est à 1, l'état de la sortie est imposé par la fonction évoluée correspondante.
Remarques:	#OUTPUT_CONFIG est mémorisé et se conserve en cas de coupure d'alimentation. La valeur par défaut en sortie d'usine est 3 (<code>OUT1</code> et <code>OUT2</code> représentent respectivement BUSY et DEFAULT).
Exemple:	<code>00#OUTPUT_CONFIG:=0</code> Toutes les sorties sont en sortie utilisateur <code>00#OUTPUT_CONFIG:=1</code> <code>OUT1</code> = BUSY, <code>OUT2</code> = sortie utilisateur <code>00#OUTPUT_CONFIG:=2</code> <code>OUT1</code> = sortie utilisateur, <code>OUT2</code> = DEFAULT <code>00#OUTPUT_CONFIG:=3</code> <code>OUT1</code> = BUSY, <code>OUT2</code> = DEFAULT



5.24. #POSITION

Mnémonique	#POS
Paramètre:	Nouvelle position de l'axe en incrément moteur (10^{-4} tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	En mode boucle ouverte, #POSITION indique la position électrique de consigne exprimée en micropas avec une résolution de 50µpas/pas. En mode boucle fermée, cette variable indique la position physique de l'axe avec une résolution de 10000incréments/tour et une précision d'1/1000 de tour.
Remarques:	Cette variable est mémorisée à la coupure d'alimentation et restituée à la mise sous tension. Sa valeur initiale n'est donc pas valable si l'axe moteur a été tourné hors tension ou bien si la coupure d'alimentation est intervenue en cours de mouvement. Attention à l'utilisation de cette commande lorsque les butées virtuelles sont activées: spécifier une position au-delà d'une butée a pour effet d'arrêter le mouvement.
Exemple:	04#POSITION:=0 Tous les mouvements en mode position du module d'adresse 04 seront référencés par rapport à la position courante.

5.25. #POSITIVE_END

Mnémonique	#PEN
Paramètre:	Position de la butée virtuelle positive (10^{-4} tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	Les mouvements qui dépassent cette consigne sont arrêtés, un bit d'état est positionné, et seuls les mouvements dans le sens négatif (anti-horaire) sont autorisés. Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	La valeur de la position des butées software n'est pas modifiée lorsque la position de référence #POSITION est modifiée.
Exemple:	#POSITIVE_END:=100000 Les modules ne pourront pas aller au-delà de 10 tours de la position d'origine.



5.26. #RATED_CURRENT

Mnémonique	#RCU
Paramètre:	Courant nominal du moteur, exprimé en mA efficaces par phase. [0 ; 2500] sur BMAC [0 ; 7000] sur BMAC-H
Description:	Cette variable permet de régler le courant maximum qui sera injecté dans le moteur. Par la suite, le courant réellement injecté dépendra du paramètre #CURRENT_RATIO avec la formule : Courant (mA) = #RATED_CURRENT * #CURRENT_RATIO /1000 Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.
Remarques:	Ce courant est indiqué sur la plaque signalétique du moteur ou bien dans sa fiche de spécification. Il s'agit du courant efficace par phase.
Exemple:	00#RATED_CURRENT:=1200 Paramètre le BMAC pour piloter un moteur 1,2Aeff

5.27. #SPEED, #PROFILE_SPEED (lecture seule)

Mnémonique	#SPE, #PSP
Paramètre:	Vitesse de rotation de l'axe en 100 ^{ème} de tr/min. Valeur signée
Description:	La variable #SPEED permet de connaître la vitesse de rotation du moteur (mesurée). La variable #PROFILE_SPEED permet de connaître la vitesse de consigne théorique instantanée générée par le module.
Remarques:	La valeur de #SPEED est déterminée par échantillonnage de la position de l'axe. Il est donc possible que sa valeur instantanée soit légèrement différente de la vitesse réelle.
Exemple:	04READ #SPEED → 04#SPE=-20000 Relecture de la vitesse de l'axe 4. L'axe 4 tourne à 200 tr/min dans le sens négatif.



5.28. #STATUS (lecture seule)

Mnémonique #STA

Paramètre: Aucun

Description: La relecture de l'état du BMAC renvoie une valeur dont les bits représentent l'état du module:

poids fort	bit 32	Mouvement interrompu anormalement
	bit 31	Erreur (contrôler la variable #ERROR)
	bit 30	mode référence
	bit 29	Busy
	bit 28	fonctionnement en mode autocommuté (1) ou non (0)
	bit 27	Asservissement de position en cours
	bit 26	Mouvement en cours
	bit 25	Puissance ON (1) ou OFF (0)
	bit 24	
	bit 23	mode Synchro
	bit 22	
	bit 21	Frein actif (1) ou désactivé (0, cf. commande BRAKE OFF)
	bit 20	Etat Butée Soft négative
	bit 19	Etat Butée Soft positive
	bit 18	Etat Butée Hard négative
	bit 17	Etat Butée Hard positive
	bit 16	Séquenceur en mode Edition
	bit 15	Séquenceur en cours d'exécution
	bit 14	Polarité des sorties standard (0) ou inversée (1)
	bit 13	Polarité des entrées standard (0) ou inversée (1)
	bit 12	Rampes en « S » (1) ou Trapèze (0)
	bit 11	
	bit 10	
	bit 9	
	bit 8	Inversion de polarité des entrées butées hard
	bit 7	Autorisation des butées soft
	bit 6	Autorisation de la butée hard Inférieure
	bit 5	Autorisation de la butée hard Supérieure
	bit 4	Mode Courant Optimisé
	bit 3	
	bit 2	
	bit 1	
poids faible		

Exemple: 03READ h#STATUS → 03#STA=h13000800

Relecture de l'état du module d'adresse 3 au format hexadécimal.



5.29. #SUPPLY_VOLTAGE (lecture seule)

Mnémonique: #SVO

Paramètre: Mesure de la tension d'alimentation du module en mV

Exemple: 02READ #SUPPLY_VOLTAGE → 02#SVO=32000

Le module d'adresse 2 est alimenté avec une tension de 32V.

5.30. #TIMER_1 à #TIMER_3

Mnémonique: #T1 à #T3

Paramètre: Tempo en millisecondes [0 ; + 2 147 483 647]

Description: Les variables #TIMER_1 à #TIMER_3 permettent d'introduire des temporisations dans une séquence.
 A partir de la valeur initiale définie par l'utilisateur, la valeur des variables est décrétementée toutes les millisecondes jusqu'à zéro. Ces variables peuvent être relues (READ #TIMER_1) ou testées (IF #TIMER_1=0 JUMP...) pour savoir si le délai est écoulé ou non.

Exemple: #TIMER_3:=2000

La variable #TIMER_3 est chargée avec la valeur 2000. Elle sera décrétementée toutes les millisecondes pendant deux secondes.

5.31. #V1 à #V32

Mnémonique: #V1 à #V32

Paramètre: Valeur signée comprise entre -2^{31} et $2^{31}-1$
 [-2 147 483 648 ; + 2 147 483 647]

Description: Les variables #V1 à #V32 peuvent être utilisées dans une ligne de séquence ou en commande directe pour stocker des valeurs ou pour faire des calculs.
 Ces variables sont initialisées à 0 (zéro) à chaque mise sous tension ou reset du module.

Exemple: 00#V13:=1234

Mémore la valeur 1234 dans la variable #V13 du module d'adresse 0.



6. Séquenceur

6.1. Présentation des Fonctionnalités

Le BMAC offre la possibilité de mémoriser puis d'exécuter une suite de commandes.

Chaque ligne est référencée par son "numéro de ligne" (de 1 à 500).

Lorsque le séquenceur est à la ligne 0 (#LINE=0), le séquenceur est OFF.

Lorsque le séquenceur est démarré (commande "START_SEQ n" où n est la première ligne à être exécutée), le module exécute une ligne toutes les millisecondes puis passe à la ligne suivante (incrément automatique de #LINE).

6.2. Mémorisation des lignes

Pour mémoriser les lignes:

1. Passer le module en "mode édition" avec la commande OPEN_SEQ.
2. Entrer les commandes qui doivent être exécutées, comme s'il s'agissait de commandes directes. Le numéro de ligne est automatiquement incrémenté ou peut être forcé en faisant précéder la commande de " :n " où n est le numéro de la ligne.
(exemple: "00:120 MOVE_TO 15000")
3. Quitter le "mode édition" avec la commande CLOSE_SEQ

Exemple: mémorisation d'une séquence qui fait faire au module un tour toutes les deux secondes

```
OPEN_SEQ           ;ouverture mode édition
MOVE_ON 10000      ;1 tour dans le sens positif
WAIT 2000          ;tempo
JUMP 1             ;retour à la ligne 1
CLOSE_SEQ          ;fermeture mode édition
```

L'automatisme mémorisé sera donc:

Ligne 1: MOVE_ON 10000

Ligne 2: WAIT 2000

Ligne 3: JUMP 1

6.3. Exécution du séquenceur

Pour lancer l'exécution, utiliser la commande START_SEQ en précisant en paramètre le numéro de la première ligne à devoir être exécutée.

Pour arrêter l'exécution, envoyer la commande STOP (ou STOP_SEQ pour n'arrêter que la séquence et pas le mouvement). La ligne 0 correspond à un séquenceur arrêté. Lorsque le module est en ligne 0, il y reste jusqu'à ce qu'il reçoive une commande de type START_SEQ.

Il est possible de forcer l'exécution d'une séquence à la mise sous tension ou au reset du module. Pour cela, noter le numéro de la première ligne de la séquence dans la variable #ON_RESET.

Exemple: pour exécuter une séquence qui commence à la ligne 12 à la mise sous tension du module, envoyer la commande " #ON_RESET:=12 ".

Pour désactiver la séquence de démarrage, entrer simplement #ON_RESET:=0



6.4. Exemples de séquences

6.4.1. Exemple 1

Dans cet exemple simple, la sortie 4 s'allume si la tension d'alimentation est comprise entre 15V et 20V

```

OPEN_SEQ                                ;mode édition
IF #SUPPLY_VOLTAGE < 15000 JUMP 5        ;sortie OFF si U < 15Volts
IF #SUPPLY_VOLTAGE > 20000 JUMP 5        ;sortie OFF si U > 20Volts
#OUTPUT.4:=1                             ;sinon sortie ON
JUMP 1                                   ;reboucle sur la ligne 1
#OUTPUT.4:=0                             ;sortie OFF
JUMP 1                                   ;reboucle sur la ligne 1
CLOSE_SEQ                                ;sortie du mode édition
START_SEQ 1                             ;exécution
    
```

6.4.2. Exemple 2

Dans cet exemple, la vitesse de rotation du BMAC est asservie à l'entrée analogique à laquelle on applique un offset et un gain. (cet exemple utilise les mnémoniques de commandes plutôt que la forme complète)

```

OSE                                      ;mode édition (OPEN_SEQ)
:50 #V1:=#IAN - 15000                  ;ligne 50: #INPUT_ANALOG - 15000 dans V1
#V1:=#V1 * 5                           ;gain de 5
MSP #V1                                ;mouvement en vitesse (MOVE_SPEED)
JRE -3                                  ;reboucle sur la ligne 50 (JUMP_REL)
CSE                                     ;fermeture du mode édition (CLOSE_SEQ)
SSE 50                                 ;exécution (START_SEQ)
    
```

6.4.3. Exemple 3

Dans cet exemple, l'entrée logique 2 commande un mouvement de 1 tour dans le sens positif et l'entrée 3 commande un mouvement de 1 tour dans le sens négatif.

```

OPEN_SEQ                                ;mode édition
IF #INPUT.2 = 1 JUMP_REL +3             ;teste l'entrée 2
IF #INPUT.3 = 1 JUMP_REL +4             ;teste l'entrée 3
JUMP 1                                   ;reboucle sur la ligne 1
MOVE_ON 10000                           ;1 tour dans le sens positif
JUMP_REL 2                               ;saute à attente de fin de mouvement
MOVE_ON -10000                           ;1 tour dans le sens négatif
WAIT 0                                   ;attente de fin de mouvement
JUMP 1                                   ;reboucle sur la ligne 1
CLOSE_SEQ                                ;fermeture du mode édition
START_SEQ 1                             ;exécution
    
```

6.4.4. Exemple 4

Cet exemple utilise l'appel à une sous-routine pour mettre à zéro la position si l'entrée 3 est inactive ou si la tension d'alimentation est supérieure à 30V

```

OPEN_SEQ                                ;mode édition à la ligne 1
:150 IF #INPUT.3 = 0 CALL 160            ;Ligne 150: teste l'entrée 3
IF #SUPPLY_VOLTAGE > 30000 CALL 160      ;teste la tension d'alimentation
JUMP 0                                   ;arrête le séquenceur

:160 #POSITION:=0                       ;Ligne 160: met la position à zéro
RETURN                                   ;retourne au call
CLOSE_SEQ                                ;fermeture du mode édition
START_SEQ 150                           ;exécution
    
```



7. ANNEXES

7.1. Résumé des commandes

Commande	Mnémo.	Paramètre	Direct	Séquence	Etat usine ou suite à MRE ALL
CALL	CAL	ligne seq		◇	
CLOSE_LOOP	CLO	ON / OFF	◇	◇	OFF
CLOSE_SEQ	CSE	-	◇		
HALT	HAL	- / SEQ / MOUV	◇	◇	
HARD_ENDS	HEN	OFF / ALL / POS / NEG	◇	◇	OFF
IF	IF	test		◇	
INVERSE_POLARITY	IPO	OFF / ALL / IN / OUT	◇	◇	OFF
JUMP	JUM	ligne seq		◇	
JUMP_REL	JRE	ligne seq		◇	
MODULE_RESET	MRE	- / ALL	◇		
MOVE_INTERPOL	MIN	position relative	◇		
MOVE_ON	MON	position relative	◇	◇	
MOVE_SPEED	MSP	vitesse	◇	◇	
MOVE_TO	MTO	position absolue	◇	◇	
OPEN_SEQ	OSE	-	◇		
OPTIMIZED_CURRENT	OCU	ON / OFF	◇	◇	OFF
POWER	POW	ON / OFF / SC	◇	◇	OFF
READ	REA	variable	◇		
READ_SEQ	RSE	ligne seq	◇		
REFERENCE	REF	INDEX / ON / OFF	◇	◇	OFF
RETURN	RET	-		◇	
REQUEST_VERSION	RV	-	◇		
SET_ADDRESS	SAD	adresse	◇		0
SET_BAUDRATE	SBA	baudrate	◇		38400
S_CURVE	SCU	ON / OFF	◇	◇	OFF
SOFT_ENDS	SEN	ALL / OFF	◇	◇	OFF
START_SEQ	SSE	ligne seq	◇		
STEP	STE	ligne seq	◇		
STOP	STO	- / SEQ / MOUV	◇	◇	
SYNCHRO	SYN	ON / OFF / TOP / INTERPOL	◇		OFF
WAIT	WAI	temps		◇	



7.2. Résumé des variables

Variable	Mnémo.	Mémorisé	Lecture seule	Valeur usine ou suite à MRE ALL
#ACCEL_TIME	#ATI	◇		1 000
#CAPTURE	#CAP		◇	
#CPU_TEMPERATURE	#CTE		◇	
#CURRENT_RATIO	#CRA	◇		500
#DECEL_TIME	#DTI	◇		1 000
#DRIVER_TEMPERATURE	#DTE		◇	
#ELECTRICAL_POSITION	#EPO	◇	◇	0
#ENCODER	#ENC			0
#ERROR	#ERR			
#HIGH_SPEED	#HSP	◇		60 000
#INPUT	#INP		◇	
#INPUT_ANALOG	#IAN		◇	
#INTERPOL_COUNT	#ICO		◇	
#INTERPOL_FIFOSIZE	#IFI			64
#INTERPOL_MODE	#IMO			0
#INTERPOL_TIME	#ITI			100
#LINE_DELAY	#LDE	◇		3 000
#LINE	#LIN			0
#LOW_SPEED	#LSP	◇		6 000
#M1 à #M8	#M1 à #M8	◇		0
#NEGATIVE_END	#NEN	◇		-100 000
#ON_RESET	#ORE	◇		0
#OUTPUT	#OUT			0
#OUTPUT_CONFIG	#OCO	◇		3
#POSITION	#POS	◇		0
#POSITIVE_END	#PEN	◇		+100 000
#PROFILE_SPEED	#PSP		◇	
#RATED_CURRENT	#RCU	◇		2500
#SPEED	#SPE		◇	
#STATUS	#STA		◇	
#SUPPLY_VOLTAGE	#SVO		◇	
#TIMER_1 à #TIMER_3	#T1 à #T3			0
#V1 à #V32	#V1 à #V32			0

7.3. Documents associés

Ces documents sont accessibles sur le site midi-ingenierie.com.

- Note d'application – Installation du driver USB FTDI
- Note d'application – Liaison calculateur : protocoles et syntaxe
- Manuel utilisateur DMAC & BMAC CANopen