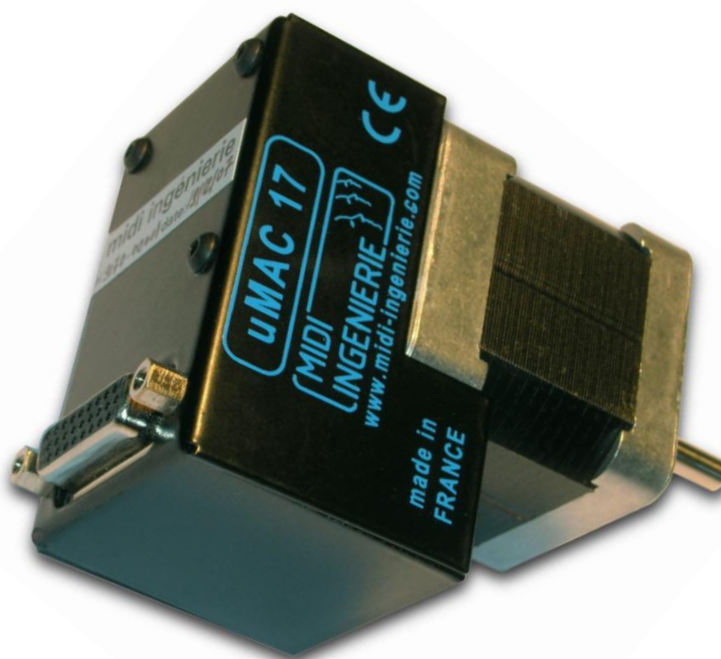


**NEXEYA Products Division**

Labège Innopole
Immeuble de Memphis
3509, route de Baziège
BP 48308
31683 LABÈGE Cedex
France
Tél. : 33 (0)5 61 39 96 18
Fax : 33 (0)5 61 39 17 58
www.nexeya-products.com

midi ingenierie
MICROMAC17
manuel utilisateur



Date : 15.11.11

Référence : umac17_v6_um_fr.pdf

Réf. MI : BLN1700858.DOC

Révision : 6

Auteur : B.LOPEZ

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



Sommaire

1.	Présentation du produit.....	4
1.1.	INTRODUCTION.....	4
1.2.	FONCTIONNALITES	4
1.2.1.	Architecture	4
1.2.2.	Mouvements	4
1.2.3.	Butées.....	5
1.3.	COUPLE ET PUISSANCE.....	6
1.3.1.	Couple maximum.....	6
1.3.2.	Couple permanent.....	7
1.3.3.	Entrées – Sorties.....	7
1.3.4.	Précautions d'emploi	7
1.3.4.1.	Règles générales	7
1.3.4.2.	Conditions de stockage	7
1.3.4.3.	Conditions d'utilisation.....	7
2.	Spécifications techniques	8
2.1.	SPECIFICATIONS GENERALES	8
2.2.	SPECIFICATIONS MECANQUES	8
2.3.	ALIMENTATION	9
3.	Câblage et configuration.....	9
3.1.	DESCRIPTION DU CONNECTEUR.....	9
3.2.	ENTREES/SORTIES	10
3.2.1.	Fonctionnalités des entrées/sorties.....	10
3.2.2.	Entrées logiques opto-isolées	10
3.2.3.	Entrées logiques non isolées	10
3.2.4.	Sorties logiques opto-isolées.....	11
3.2.5.	Entrée analogique	11
3.3.	CONFIGURATION DU PORT DE COMMUNICATION.....	11
3.3.1.	Interface série standard.....	11
3.3.2.	Interface série Option RS485	11
3.3.3.	Les protocoles séries.....	11
3.3.4.	Comment communiquer avec le module?.....	11
4.	COMMANDES ET séquences	12
4.1.	VARIABLES	12
4.1.1.	Définition des variables	12
4.1.2.	Manipulation des variables.....	13
4.1.2.1.	Lecture écriture	13
4.1.2.2.	Opérations et séquences	13
4.2.	SEQUENCES	13
4.2.1.	Organisation des séquences.....	13
4.2.2.	Ecriture et mémorisation des séquences	13
4.2.3.	Description des séquences	14
4.2.4.	Sélection et exécution des séquences.....	14
4.2.5.	Déroulement des séquences.....	14
4.2.5.1.	Enchaînement naturel	15
4.2.5.2.	Enchaînement conditionnel	15
4.2.5.3.	Sous séquences.....	15
5.	Langage micromac	16
5.1.	GESTION GENERALE ET CONFIGURATION	16
5.1.1.	SET_ADDRESS : adresse module	16
5.1.2.	SET_BAUDRATE : vitesse de dialogue série	17
5.1.3.	REQUEST_VERSION : relecture de version	17
5.1.4.	MODULE_RESET : réinitialisation	18
5.1.5.	READ : Relecture des variables.....	19
5.1.6.	POWER : Puissance moteur	19



5.2.	PARAMETRES.....	20
5.2.1.	Sauvegarde des paramètres	20
5.2.2.	Cohérence des vitesses	20
5.2.3.	#HIGH_SPEED : vitesse de consigne maximum	20
5.2.4.	#LOW_SPEED : vitesse d'approche	21
5.2.5.	#RAMPING_TIME : loi d'accélération	21
5.2.6.	#TORQUE_RATIO : Courant moteur	22
5.2.7.	#LOW_TORQUE : Courant de Standby du moteur	22
5.2.8.	#POSITIVE_END/NEGATIVE_END : butées logicielles	23
5.3.	ENTREES SORTIES	24
5.3.1.	#OUTPUT: sorties digitales	24
5.3.2.	#INPUT : entrées digitales.....	24
5.3.3.	#INPUT_ANALOG : Entrée analogique	25
5.3.4.	#SUPPLY_VOLTAGE : tension d'alimentation	25
5.3.5.	#TEMPERATURE : température du boîtier.....	25
5.4.	VARIABLES	26
5.4.1.	Paramètres, variables système et variables utilisateur	26
5.4.2.	Opérations sur les variables.....	26
5.4.3.	#Vn : variable utilisateur	26
5.4.4.	#Mn : variable utilisateur mémorisée.....	27
5.5.	GESTION DES MOUVEMENTS	28
5.5.1.	#POSITION : compteur de position absolue	28
5.5.2.	#PROFILE_SPEED : vitesse courante.....	28
5.5.3.	MOVE_TO : Mouvement absolu	29
5.5.4.	MOVE_ON : mouvement relatif.....	29
5.5.5.	MOVE_SPEED : Mouvement à vitesse imposée	30
5.5.6.	STOP : Arrêt avec décélération.....	31
5.5.7.	HALT : Arrêt d'urgence.....	31
5.6.	MODES.....	32
5.6.1.	STANDBY_MODE : Gestion du Standby	32
5.6.2.	HARD_ENDS : Butées hardware	32
5.6.3.	SOFT_ENDS : Butées software	33
5.6.4.	INVERSE_POLARITY : inversion de polarité.....	33
5.7.	GESTION DES SEQUENCES.....	34
5.7.1.	OPEN/CLOSE_SEQ : édition des séquences.....	34
5.7.2.	:n : déclaration de phases de séquence.....	35
5.7.3.	START_SEQUENCE : lancement d'une séquence	35
5.7.4.	#ON_RESET : séquence de démarrage.....	35
5.7.5.	WAIT : temporisation	36
5.7.6.	JUMP: saut entre phases	36
5.7.7.	JUMP_RELATIVE : saut relatif entre phases.....	37
5.7.8.	IF... JUMP : saut conditionnel.....	37
5.7.9.	IF... JRE : saut conditionnel relatif.....	38
5.7.10.	CALL/RETURN: sous séquence.....	38
5.7.11.	#LINE : ligne courante exécutée.....	39
5.7.12.	STEP : pas à pas	39
5.7.13.	\$upLoad : relecture de la mémoire séquence	39
5.8.	CONTROLE	40
5.8.1.	#STATUS: état du module.....	40
5.8.2.	#ERROR: Compte rendu d'erreur	41
6.	ANNEXES.....	42
6.1.	EXEMPLES DE SEQUENCES	42
6.1.1.	Exemple 1.....	42
6.1.2.	Exemple 2.....	42
6.1.3.	Exemple 3.....	42
6.1.4.	Exemple 4.....	43
6.1.5.	Exemple 4 bis	43
6.1.6.	Exemple 5.....	44
6.2.	RESUME DES COMMANDES ET VARIABLES	45
6.3.	DOCUMENTS ASSOCIES	46

1. PRESENTATION DU PRODUIT

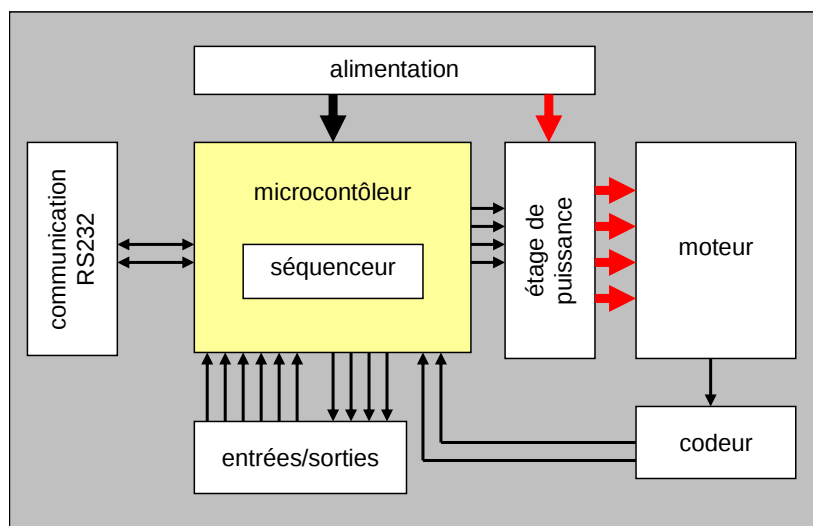
1.1. Introduction

Le MICROMAC est un module composé d'un moteur et d'une électronique de commande intégrée fonctionnant en mode auto commuté. Ce type de contrôle permet de garantir à tout moment la position du moteur grâce au codeur incrémental intégré. On obtient ainsi un système qui combine les avantages d'un moteur pas à pas et d'un moteur brushless.

Le module se présente sous la forme d'un boîtier compact avec une connectique simplifiée. Il se contrôle grâce à un protocole série spécifique utilisant le standard RS-232 V24.

1.2. Fonctionnalités

1.2.1. Architecture



1.2.2. Mouvements

Les mouvements se font avec une résolution de 1/2000ème de tour. Un incrément de position correspond donc à un angle 0,18°. La vitesse des mouvements peut être définie jusqu'à 1200t/mn.

Afin d'avoir un maximum de résolution dans les vitesses basses, toutes les vitesses sont exprimées en 100ème de tour par minute (10-2 tr/min). Les positions sont définies en incréments de la résolution (5 10-4tr).

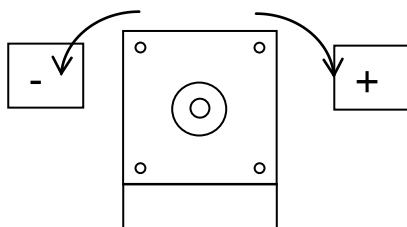
On distingue deux types de mouvements:

- Le mode Vitesse (MOVE_SPEED) dans lequel le moteur tourne à une vitesse de consigne.
- Le mode Position (MOVE_TO, 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.

Pour les deux 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. Le profil de vitesse est de type trapézoïdal et l'utilisateur peut spécifier la durée totale de la rampe.

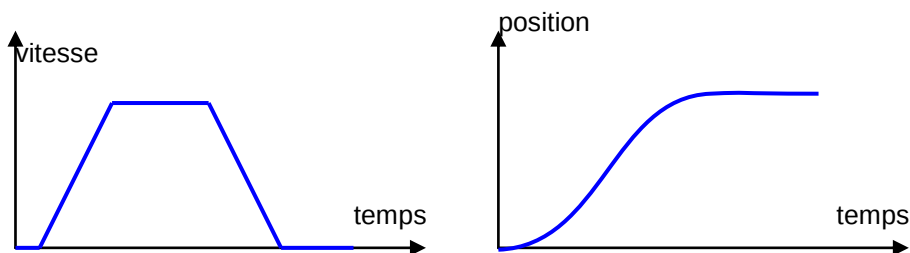
Pour les mouvements relatifs ou les mouvements à vitesses données, le sens des mouvements est défini par le signe de la consigne donnée. Pour les mouvements absolus le sens réel dépend du signe de l'écart entre la position courante et la position à atteindre.

Le sens de mouvement positif correspond à une rotation en sens horaire du rotor vu de face (coté arbre moteur).

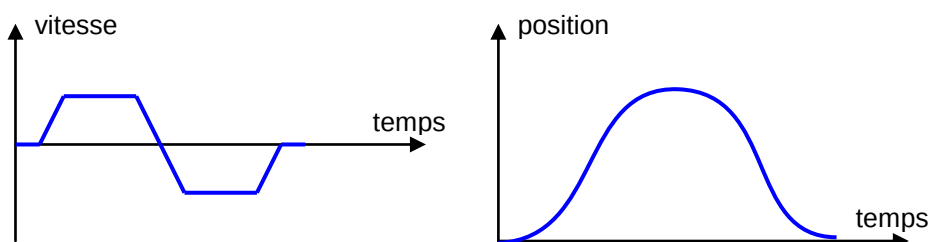


Exemples de mouvements:

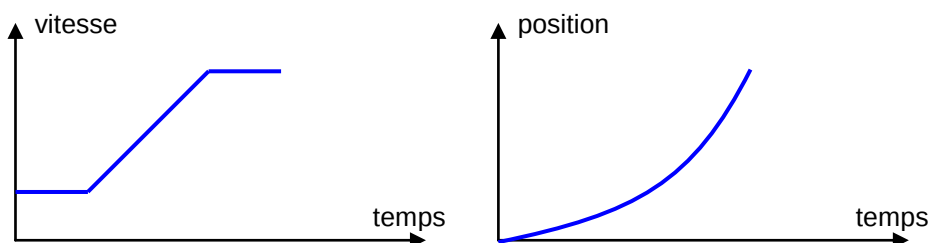
Mouvement simple avec rampes trapézoïdales:



Aller retour avec rampes :



Contrôle en vitesse, passage d'une vitesse à une autre:



1.2.3. Butées

Le MICROMAC dispose de quatre butées:

- Deux butées "matérielles" (aussi appelées butées Hard). Ce sont des butées électriques qui utilisent les entrées logiques IN5 et IN6. 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 plage de positions donnée. Cette plage est définie par les paramètres : #POSITIVE_END et #NEGATIVE_END.

Au passage d'une de ces butées, 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. L'arrêt sur butée peut être réalisé avec ou sans décélération suivant le flag HALT ou STOP utilisé dans les commandes HARD_ENDS ou SOFT_ENDS.

L'arrêt sur une des butées peut-être contrôlé par l'utilisateur en relisant le registre d'état via la commande READ #STATUS.

Remarque : L'utilisation des butées n'est pas obligatoire, les butées peuvent être autorisées indépendamment les unes des autres.

1.3. Couple et puissance

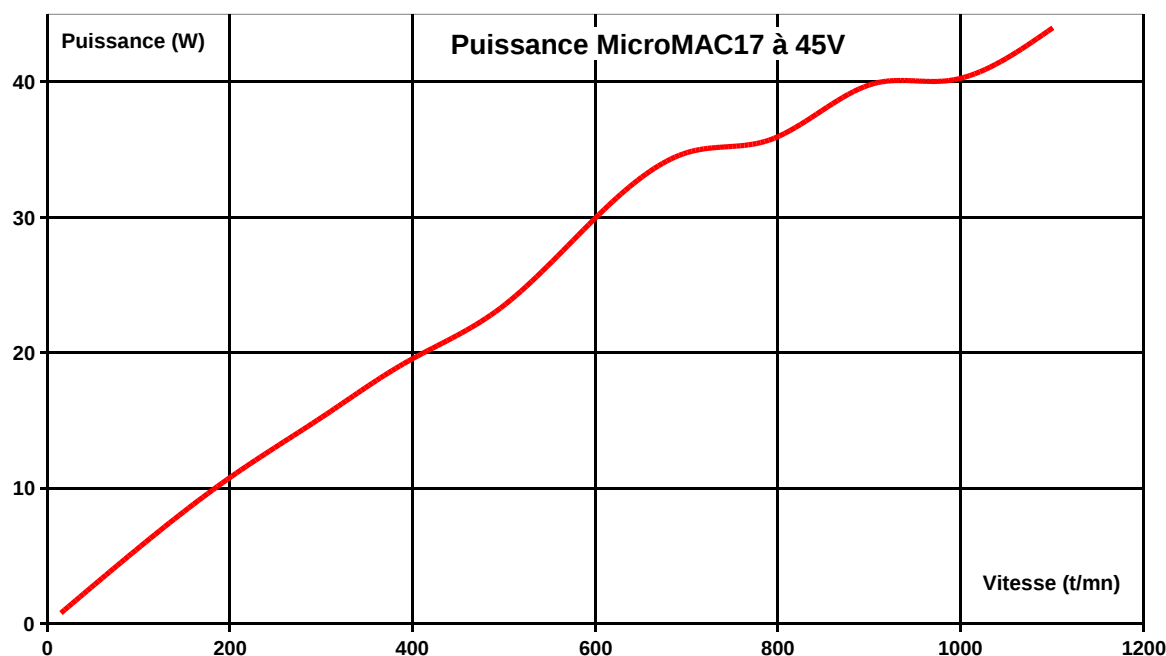
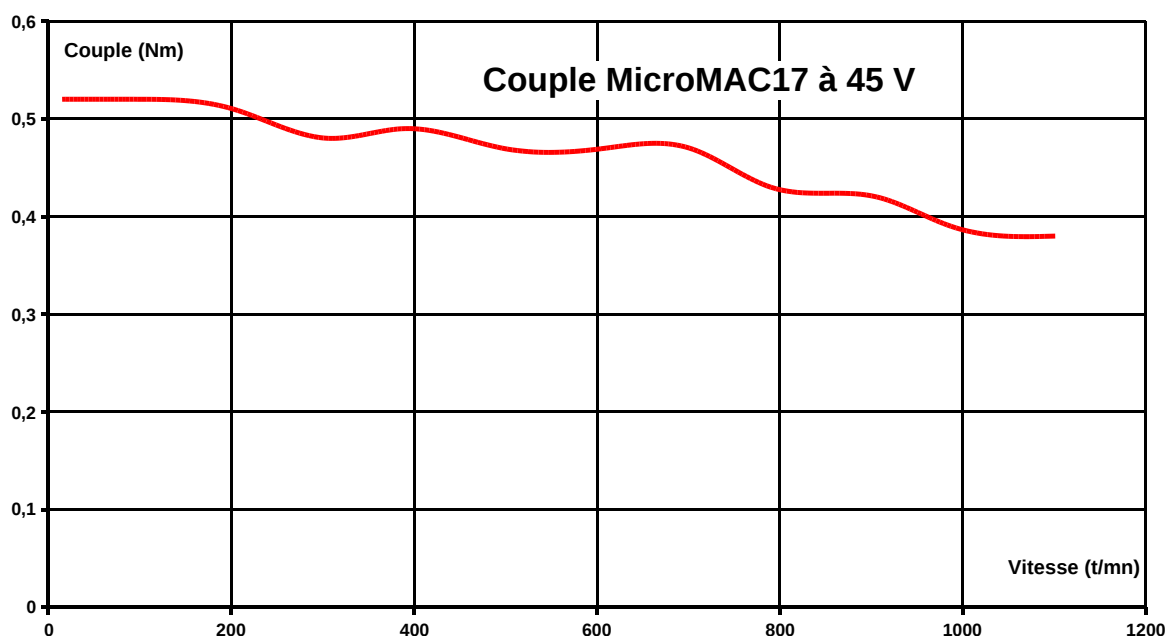
1.3.1. Couple maximum

Le paramètre #TORQUE_RATIO règle le couple maximum disponible en sortie du moteur jusqu'à 0,5Nm.

Le mode Standby permet, lors des phases d'arrêt, de gérer un courant différent de celui nécessaire au mouvement et donc d'injecter dans le moteur le seul courant réellement nécessaire au maintien de la position à l'arrêt afin de minimiser les pertes thermiques ; ce courant est réglé par le paramètre #LOW_TORQUE.

La puissance moteur peut être coupée ou rétablie grâce à la commande POWER ON/OFF. Elle est automatiquement rétablie au début de chaque mouvement. Le mode court circuit (POWER SC) permet de court-circuiter le moteur lorsqu'il est hors puissance et d'obtenir un couple résistant de type frottement visqueux.

Par construction, le couple du moteur est maximum à basse vitesse, et ce type de motorisation permet donc d'exercer du couple à l'arrêt et de maintenir une raideur contrôlée.



1.3.2. Couple permanent

Compte tenu de l'échauffement dû aux pertes joules du driver et aux pertes fer du moteur, le module MICROMAC ne peut, sans réducteur additionnel, maintenir le couple maximum disponible.

Le couple permanent utilisable est de 75% du couple maximum pour une température ambiante de 25°C sans ventilation et de 50% pour une température ambiante de 40°C (moteur vissé sur châssis métallique).

Une version MICROMAC avec radiateur peut être fournie sur demande.

1.3.3. Entrées – Sorties

Le MICROMAC dispose de 6 entrées et 4 sorties opto-isolées qui peuvent être utilisées conjointement avec l'automate intégré au module pour, par exemple, déclencher des mouvements ou visualiser une position.

Une entrée analogique unipolaire 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 de l'automate pour, par exemple, asservir la position ou la vitesse de l'axe à la position d'un potentiomètre.

1.3.4. Précautions d'emploi

1.3.4.1. Règles générales

- * Les moteurs sont qualifiés IP30, respecter les limites relatives à cet indice de protection. En particulier, le moteur 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 nucléaires.
- * Ne jamais ouvrir un module. Les tensions internes peuvent atteindre un niveau dangereux.
- * Ne pas toucher un module sous tension : risque de brûlure ou d'électrocution.
- * Ne pas toucher l'arbre moteur : risque de blessure.
- * Ne pas soumettre l'arbre moteur à un effort axial ≥ 10 N ou bien à un effort radial ≥ 20 N à 5 mm du flasque.

1.3.4.2. Conditions de stockage

- * Protéger le module contre les rayons solaires et l'humidité.
- * Le module doit être stocké ou transporté dans son emballage d'origine ou dans un conditionnement adapté.
- * La température doit être comprise entre -20°C et $+85^{\circ}\text{C}$.

1.3.4.3. Conditions d'utilisation

- * **Attention!** Le moteur peut atteindre une température de 85°C 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.
- * Utiliser une alimentation protégée en surintensité ou bien insérer un fusible 5A temporisés sur la ligne d'alimentation DC.
- * 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 MICROMAC à la masse générale de la machine.
- * Ne jamais introduire un corps étranger dans les orifices du produit.

2. SPECIFICATIONS TECHNIQUES

2.1. Spécifications générales

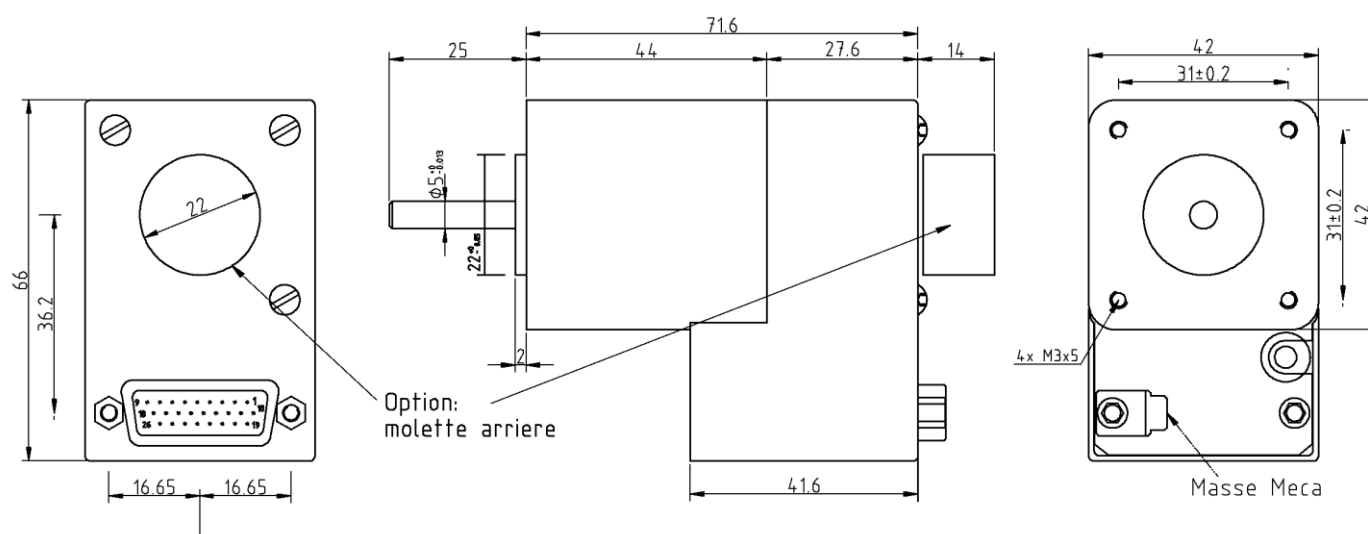
Dimensions			42mm x 66mm x 71.6mm
Masse			480 gr
Alimentation Continue	§ 2.2	Consommation (sans moteur ni actuateur)	12 V - 50 VDC < 0,2 A
Couple statique moteur	§5.2.6 et 5.2.7	Couple maximum disponible	Cnom=0.5Nm
Couple permanent	§5.2.6 et 5.2.7	@25°C de température ambiante @40°C de température ambiante	75% du couple max 50% du couple max
Résolution	§1.2.2		2000 position/tour (0,18°)
Vitesse			$1 < \omega < 1200$ t/mn
2 entrées butées	§3.2.3	seuil TTL	pull up entrée 10 K Ω à 5V
4 entrées logiques	§3.2.2	opto-isolées	4V $\leftrightarrow$ 30 V
4 sorties logiques	§3.2.4	opto-isolées (collecteur ouvert) $I \leq 80$ mA	< 0,6 V @ 1 mA < 1,1 V @ 5 mA
1 entrée analogique	§3.2.5		Résolution 10 bits Précision ± 2 LSB
Liaison série	§3.3	interface RS232V24	9600, 19200, 38400, 115000 bauds 8 bits sans parité
Sortie		+5V	Impédance 100 Ω
Température ambiante de fonctionnement			0 - 40°C
Divers		gestion repos et surcourant	Indépendants des sorties logiques

2.2. Spécifications mécaniques

Taille: 42mm x 66mm x 71,6mm (hors arbre moteur et connecteur)

Poids: à déterminer

Plan d'encombrement:



2.3. Alimentation

Le MICROMAC est conçu pour fonctionner avec une tension d'alimentation comprise entre 12 et 50V. Le courant total consommé par le module (y compris moteur actif) est inférieur à 2A. (**Attention!** A faible tension, le courant consommé est plus élevé). 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).

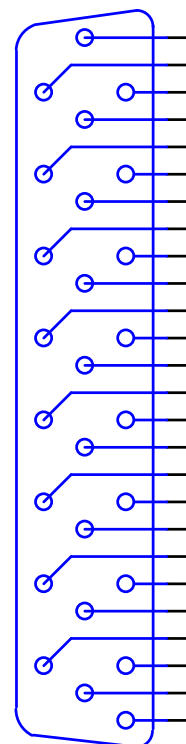
Plus la tension d'alimentation est élevée, meilleur est le couple à haute vitesse. Le couple à l'arrêt ou à faible vitesse est indépendant de la tension d'alimentation. Lorsque la tension descend en dessous de 12V, les mouvements sont arrêtés et le module se place en disjonction de sous-tension.

Lorsque le module fonctionne en frein, une partie de l'énergie est renvoyée vers l'alimentation, dans ce cas l'alimentation utilisée doit accepter cette réinjection d'énergie. La tension d'alimentation peut donc augmenter, lorsque elle dépasse le seuil de tension maximum admise par le module, celui ci disjoncte supprimant de fait le freinage et donc la réinjection d'énergie vers l'alimentation. Lorsque l'on ne peut accepter cette disjonction il convient d'utiliser un ballast pour dissiper l'énergie récupérée.

3. CABLAGE ET CONFIGURATION

3.1. Description du connecteur

Broche	Description	Connecteur SUBD haute densité 26 points Contacts femelles
10	+Valim	
19	0V analogique (référence analogique)	
1	+Valim	
11	0Valim	
20	Entrée analogique	
2	0Valim	
12	Réservé	
21	Butée + /entrée logique 6 (non isolée)	
3	Butée - /entrée logique 5 (non isolée)	
13	+ Entrée logique 1	
22	+ Entrée logique 2	
4	- Commun entrées logiques (0V IO)	
14	+ Entrée logique 3	
23	+ Entrée logique 4	
5	- Commun entrées logiques (0V IO)	
15	0V V24	
24	Tx_ext	
6	Tx_V24 (Z RS485)	
16	Rx_V24 ($\bar{Z}$ RS485)	
25	Rx_ext	
7	+ Commun sorties logiques (+V IO)	
17	- Sortie logique 1	
26	- Sortie logique 2	
8	+ Commun sorties logiques (+V IO)	
18	- Sortie logique 3	
9	- Sortie logique 4	



3.2. Entrées/sorties

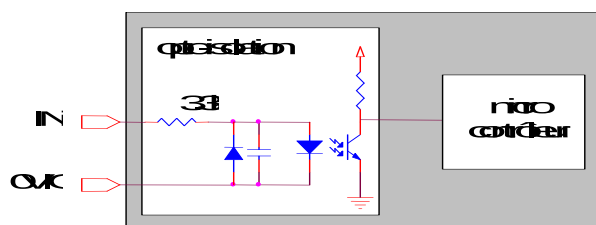
3.2.1. Fonctionnalités des entrées/sorties

Les entrées sorties utilisateur sont multiplexées avec des fonctions évoluées.

ENTREE	FONCTION
IN1	Entrée utilisateur (opto isolée)
IN2	Entrée utilisateur (opto isolée)
IN3	Entrée utilisateur (opto isolée)
IN4	Entrée utilisateur (opto isolée)
IN5	Butée - / Entrée utilisateur
IN6	Butée + / Entrée utilisateur

SORTIE	FONCTION
OUT1	Sortie utilisateur (opto isolée)
OUT2	Sortie utilisateur (opto isolée)
OUT3	Sortie utilisateur (opto isolée)
OUT4	Sortie utilisateur (opto isolée)

3.2.2. Entrées logiques opto-isolées



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

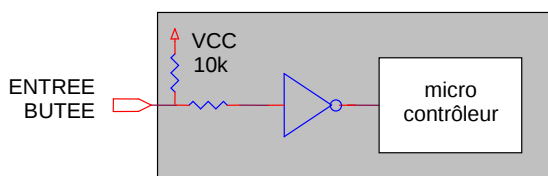
Tension nominale 5VDC à 24VDC

Tension admissible ±30VDC

Isolation galvanique 50V

L'état des entrées logiques est donné par la variable #INPUT et peut être relu avec la commande: READ #INPUT.

3.2.3. Entrées logiques non isolées



	min	max
V_{IL} (actif)	-0,3V	+0,8V
V_{IH} (inactif)	2,8V	5,5V

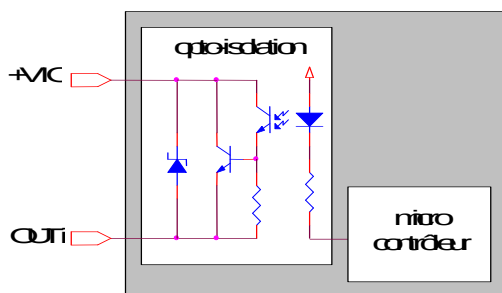
Tension nominale 5VDC (compatible TTL avec Pull up interne à 5V : 10KΩ)

Tension admissible -0,3 à +30 VDC

L'état des entrées logiques est donné par la variable #INPUT et peut être relu avec la commande READ #INPUT.

!!! Les entrées logiques non isolées sont rappelées à 5V et sont inactives lorsque elles sont laissées en l'air, elles sont donc actives à l'état électrique 0V

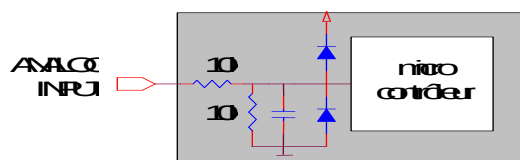
3.2.4. Sorties logiques opto-isolées



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

Courant de sortie max 50mA
Tension de sortie max 30V

3.2.5. Entrée analogique



Dynamique : $0V \leq V_{ana} \leq 10V$

Résolution : 10 bits (9,79 mV/lb)

La valeur en mV de l'entrée analogique est donnée par la variable #INPUT_ANALOG et peut donc être relue avec la commande READ #INPUT_ANALOG.

3.3. Configuration du port de communication

3.3.1. Interface série standard

Le module micromac dispose de deux interfaces série de type RS232 V24, la première permet de le piloter à partir d'un ordinateur ou d'un automate, la seconde est un port d'extension qui permet de chaîner plusieurs modules sur la même interface série. Les modules sont simplement différenciés par une adresse propre qu'il suffit de définir préalablement au moyen de la commande SET_ADDRESS (à défaut 0).

Plusieurs vitesses de dialogue sont utilisables et les modules doivent être configurés au besoin par la commande SET_BAUDRATE parmi les vitesses 9600,19200,38400 ou 115200 baud (à défaut 38400).

3.3.2. Interface série Option RS485

En option le module micromac peut être équipé d'une interface RS485 en lieu et place des deux liaisons série RS232 V24 décrit au paragraphe précédent, plusieurs modules peuvent ainsi être placés en parallèles sur la même liaison RS485. La définition des adresses et de la vitesse de dialogue est identique à celle de l'interface série standard V24. Le module peut gérer un temps de retournement de ligne de 0 à 4ms (à défaut 0) via la commande SET_BAUDRATE.



La liaison RS485 doit être référencée à la même masse que l'alimentation du module Micromac.

3.3.3. Les protocoles séries

Les caractéristiques précises des protocoles utilisables sont décrites dans la note d'application "liaison ordinateur : protocoles et syntaxe" (BLN1570828.DOC).

Seul le protocole XON/XOFF étendu est pris en compte par le module micromac.

3.3.4. Comment communiquer avec le module?

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

DrvMi.dll

Bibliothèque de fonctions accessibles depuis des programmes développés par l'utilisateur en C/C++, VisualBasic, Delphi, etc...

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.

4. COMMANDES ET SEQUENCES

4.1. Variables

Les modules MICROMAC proposent l'utilisation de variables afin d'augmenter les performances des automatismes réalisables : le comptage de cycle, la calibration, le paramétrage sont autant de fonctionnalités facilement réalisables grâce aux variables.

4.1.1. Définition des variables

Une variable est une donnée modifiable à tout moment à partir du dialogue série ou dans le déroulement même des séquences. On distinguera 2 types de variables, les variables utilisateur et les variables système:

- Les variables utilisateur

Ces variables sont totalement libres d'utilisation et peuvent être affectées à n'importe quelle grandeur au gré de l'utilisateur. Elles sont au nombre de 8 : 4 variables remises à zéro à chaque mise sous tension ou reset notées #V1 à #V4, et 4 variables dont les valeurs courantes restent mémorisées d'une mise sous tension à l'autre notées #M1 à #M4.

Ces données sont stockées sur 4 octets signés, elles peuvent représenter des valeurs numériques de consignes (entiers relatifs $-2^{31} < Var < 2^{31}$) ou des données binaires (ou hexa) comme un état de sortie logique.

-Les variables système

Si elles restent manipulables par l'utilisateur comme les précédentes, leurs affectations sont imposées par le module. Elles permettent d'accéder à certains paramètres internes du module. Les limites de ces variables dépendent évidemment de la nature des paramètres qu'elles représentent. Les variables système sont représentées par un mnémonique précédé du signe #.

Liste des variables système :

#POSITION	: position absolue du moteur
#HIGH_SPEED	: consigne de vitesse nominale des mouvements
#LOW_SPEED	: vitesse de rattrapage en fin de mouvement
#PROFILE_SPEED	: valeur théorique de la vitesse instantanée
#RAMPING_TIME	: durée de la rampe d'accélération
#TORQUE_RATIO	: couple moteur nominal
#LOW_TORQUE	: couple de maintien
#POSITIVE_END	: butée logique positive
#NEGATIVE_END	: butée logique négative
#INPUT_ANALOG	: entrée analogique
#SUPPLY_VOLTAGE	: Mesure de la tension d'alimentation en mV
#TEMPERATURE	: Mesure de la température interne du boîtier en °C
#INPUT	: entrées logiques
#OUTPUT	: sorties logiques
#ON_RESET	: adresse de démarrage automatique de séquence
#LINE	: n° de la ligne de séquence en cours d'exécution
#STATUS	: registre d'état
#ERROR	: compte rendu d'erreur

4.1.2. Manipulation des variables

4.1.2.1. Lecture écriture

Les valeurs des variables peuvent être modifiées, imposées ou relues par des commandes immédiates via la liaison série, la commande d'affectation ' := ' impose la valeur d'une variable.

#Var := xx(décimal)

#Var := hxx(hexadécimal)

#Var := bxx (binaire)

La commande READ #Vn retourne sur la liaison série la valeur courante d'une variable.

4.1.2.2. Opérations et séquences

Dans une séquence (cf §4.2) les variables peuvent être utilisées directement comme valeurs de consignes, ou manipulées par des opérations arithmétiques et logiques élémentaires : #Var1 := #Var2 op cst #Var1 := #Var2 op #Var3 avec les opérateurs op : +, -, *, /, >, >=, =, !=, < et <=

4.2. Séquences

Afin de réaliser des opérations complexes, les modules MICROMAC peuvent mémoriser des successions d'actions. Ces suites d'actions élémentaires sont appelées "séquences". Leur déroulement peut être conditionné à certains événements externes.

4.2.1. Organisation des séquences

Les séquences sont constituées d'actions élémentaires appelées phases ou lignes. Chaque phase peut contenir une commande identique à une commande directe (par exemple MOVE_ON ou POWER OFF). Les phases d'une séquence sont repérées par un numéro qui leur est propre.

Il est possible de définir plusieurs séquences dans un même module, les séquences sont simplement repérées par le numéro de leur première phase.

Un certain nombre de commandes permettent de préparer, sélectionner et exécuter les séquences.

Plusieurs séquences peuvent s'enchaîner automatiquement les unes aux autres avec ou sans condition.

4.2.2. Ecriture et mémorisation des séquences

Pour écrire ou éditer une séquence, le module doit être placé en mode édition de séquence via la directive OPEN_SEQ qui permet de préciser le numéro de la première phase à éditer. Les phases valides sont mémorisées en EEPROM au fur et à mesure de leur saisie, il est possible de préciser le numéro de la phase saisie en début de la ligne de commande par la directive :Ln où Ln est le numéro de la phase; sans précision de ligne, le numéro de phase s'incrémente automatiquement phase après phase.

Pour revenir au mode de fonctionnement nominal il convient de quitter le mode édition de séquence via la commande CLOSE_SEQ

Attention! Entre une commande d'ouverture et une commande de fermeture, toutes les commandes sont mémorisées en phases de séquences successives et ne sont donc pas exécutées immédiatement.

Exemple de séquence simple:

Le micromac tourne à la vitesse de 50tr/min et change de sens toutes les 12 secondes.

```

OPEN_SEQ 1                ;ouverture du mode Edition
:1 MOVE_SPEED 5000        ;rotation sens positif
:2 WAIT 12000             ;attente 12 secondes
:3 MOVE_SPEED -5000       ;rotation sens négatif
:4 WAIT 12000             ;attente 12 secondes
:5 JUMP 1                 ;retour à la ligne 1 (boucle)
CLOSE_SEQ                ;fermeture du mode Edition

START_SEQ 1               ;exécution de la séquence
    
```

4.2.3. Description des séquences

L'écriture des phases de la séquence s'effectue comme si l'on demandait l'exécution de commandes immédiates en les faisant précéder éventuellement des numéros de phases. A l'exception des directives d'édition, lancement et débogage des séquences, toutes les commandes immédiates sont utilisables en séquence, quelques commandes sont spécifiques des séquences et n'ont pas de signification en commande immédiate telles que les sauts et conditions de tests, ainsi que la commande de temporisation. Une phase de séquence s'écrit donc sous une des formes :

ou	[:n] Cde cst		
ou	[:n] Cde #Var	avec	$\left\{ \begin{array}{l} \text{Cde} : \text{le mnémonique d'une commande} \\ \text{cst} : \text{une valeur constante} \\ \text{op} : \text{opérateur arithmétique ou booléen} \\ \text{\#Var, \#Var1, \#Var2} : \text{variables utilisateur ou système} \end{array} \right.$
ou	[:n] #Var := #Var1 op cst		
ou	[:n] #Var := #Var1 op #Var2		

les phases conditionnelles s'écrivent sous la forme :

[:n] IF #Var1 op cst JUMP p ou [:n] IF #Var1 op #Var2 JUMP p

4.2.4. Sélection et exécution des séquences

L'exécution des séquences par les modules MICROMAC peut être lancée de plusieurs manières :

* Exécution immédiate de la séquence à partir de la phase n sous demande du calculateur hôte (éventuellement via WINSIM2 ou DRVMI) : commande START_SEQ n.

* Exécution différée de la séquence choisie à toute nouvelle mise sous tension (ou Reset) en définissant le paramètre #ON_RESET := n. (démarrage à la phase n)

L'opérateur est libre d'arrêter toute séquence en cours d'exécution ou d'en supprimer l'exécution automatique (voir commandes STOP, HALT et MODULE_RESET ...)

4.2.5. Déroulement des séquences

Il est possible de démarrer une séquence à n'importe quelle adresse de phase, ce qui permet d'écrire autant d'automatismes que désirés jusqu'à concurrence de la place mémoire disponible pour stocker les séquences. Les phases d'une séquence peuvent s'enchaîner, à partir de la première phase appelée, de trois manières distinctes :

- * Enchaînement naturel,
- * Enchaînement conditionnel.
- * Sous séquence

4.2.5.1. Enchaînement naturel

Si rien de particulier n'est précisé, les phases s'enchaînent chronologiquement depuis la première jusqu'à la dernière dans l'ordre naturel 1, 2, 3... n.

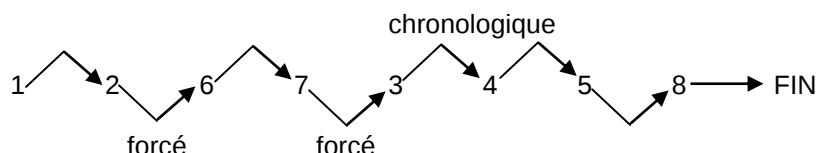
Cet ordre peut être modifié au moyen de l'instruction JUMP qui permet de préciser la phase suivante désirée.

Si la phase suivante n'est pas définie ou n'existe pas, la séquence se termine sur cette phase lors de l'exécution, même si d'autres phases correctement définies lui succèdent.

Exemple :

```
:1 Cde1
:2 JUMP 6
:3 Cde2
:4 Cde3
:5 JUMP 8
:6 Cde4
:7 JUMP 3
:8 ---
```

enchaînement :



Attention! L'instruction - - - n'existe pas en temps que telle, elle correspond à une phase de séquence vide et non définie préalablement, pour forcer la fin d'une séquence sans préalablement effacer la mémoire il convient d'utiliser l'instruction RETURN.

L'instruction NOP permet de définir une phase sans action mais ne termine pas la séquence.

4.2.5.2. Enchaînement conditionnel

Les entrées logiques et les variables permettent d'intervenir sur le déroulement chronologique des phases d'une séquence.

La valeur instantanée d'une variable et donc éventuellement des entrées logiques peut être comparée à une valeur constante ou à une autre variable. Si la condition définie par la phase est réalisée la séquence continue à la phase précisée dans l'instruction, sinon elle enchaîne la phase naturelle suivante.

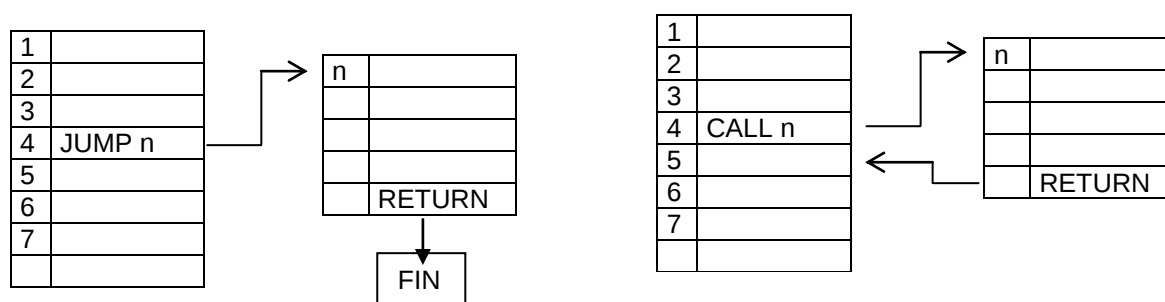
Exemple :

```
:23 IF #V2 > 50 JUMP 10
```

Si la variable #V2 est réellement supérieure à 50 la séquence continue à la phase 10 sinon à la phase 24.

4.2.5.3. Sous séquences

L'exécution d'une séquence peut être momentanément interrompue pour exécuter une sous séquence grâce à l'instruction CALL. A la différence de l'instruction JUMP, si la séquence est bien déroutée sur le numéro de phase précisé, celle ci revient à la phase suivant l'instruction CALL lorsqu'elle rencontre l'instruction RETURN. Lorsque l'instruction RETURN est exécutée en dehors d'un appel à sous séquence via l'instruction CALL elle provoque l'arrêt de la séquence en cours.



Le module MICROMAC accepte jusqu'à 5 niveaux de sous séquences.

5. LANGAGE MICROMAC

Le langage utilisé pour dialoguer avec les modules MICROMAC est basé sur un protocole de type ACK/NACK avec contrôle de flux de type XON/XOFF dit étendu. Ce protocole est parfaitement compatible avec le protocole de dialogue des modules MAC et SIMPA, ce qui permet de brancher indifféremment n'importe lequel de ces modules sur le même lien série sous réserve d'utiliser le même baudrate. La structure et la syntaxe des messages est totalement compatible avec le nouveau langage des modules MAC. La structure d'un message est de la forme :

STX nc [@] cde CS ETX

- STX et ETX : respectivement caractère de début (02h) et fin (03h) de message.
 nc : nombre total sur 3 digits codés décimal des caractères transmis hormis STX, ETX, nc et CS.
 [] : caractères ou paramètres optionnels.
 @ : adresse du module (00 à 63) auquel est destiné le message.
 Si l'adresse est omise, le message est destiné à l'ensemble des modules présents sur la ligne. Un module d'adresse 00 est indispensable dans toutes les configurations.
 Cde : libellé de la commande avec ses paramètres.
 CS : contrôle de validité de type "check sum". Codé sur deux caractères ASCII représentant chacun un des deux digits hexadécimaux obtenus en faisant la somme arithmétique modulo 256 de tous les caractères du message sauf STX, nc, CS et ETX.

L'utilisation de l'interface opérateur WINSIM2 ou de la DLL DRVMI ne nécessite que la connaissance du langage spécifique des modules, aussi, seul est décrit ici le contenu des commandes (**en gras**). Les différents protocoles utilisables pour dialoguer avec les modules sont décrits dans le document :

Famille SIMPA et MAC

Note d'application

Liaison calculateur : protocole et syntaxe

Remarque : Le langage a été conçu pour une écriture en clair des commandes, de façon à faciliter l'approche et la compréhension des commandes utilisées. Cependant, afin de minimiser les temps de traitement, et éventuellement les temps de dialogue, les modules acceptent une forme réduite sur 3 caractères de toutes les commandes et mnémoniques de variables. C'est cette dernière forme qui est utilisée par le module pour relire les variables.

Le langage MICROMAC ne fait pas la distinction entre majuscules et minuscules.

5.1. Gestion générale et configuration

5.1.1. SET_ADDRESS : adresse module

Syntaxe:	[@]SET_ADDRESS add
Mnémonique:	SAD
Paramètres:	add = dd nouvelle adresse du module $0 \leq add \leq 63$ à défaut 0
Description:	Définition de la nouvelle adresse du module sélectionné. Cette nouvelle adresse est prise en compte immédiatement et donc applicable dès le prochain dialogue.
Codes Erreur:	FI_hors_lim si la valeur fournie n'appartient pas à la liste donnée ci dessus. FI_non_num si la valeur fournie n'est pas une valeur numérique.

Remarque : Cette commande n'est pas utilisable en séquence

5.1.2. SET_BAUDRATE : vitesse de dialogue série

Syntaxe:	[@]SET_BAUDRATE v (RS232 et RS485) et [@]SET_BAUDRATE Trl (option RS485 uniquement)
Mnémonique:	SBA
Paramètres:	v = 9600,19200,38400 ou 115200 nouveau baudrate de la liaison série à défaut 38400 Trl = 0,1,2,3,4 délai en ms de retournement de ligne pour l'option RS485 à défaut 4
Description:	Définition de la nouvelle vitesse de dialogue. Cette nouvelle vitesse est prise en compte immédiatement et donc applicable dès le prochain dialogue.
Codes Erreur:	FI_hors_lim si la valeur fournie n'appartient pas à la liste donnée ci dessus. FI_non_num si la valeur fournie n'est pas une valeur numérique.

Remarque : Cette commande n'est pas utilisable en séquence

5.1.3. REQUEST_VERSION : relecture de version

Syntaxe: @REQUEST_VERSION

Mnémonique: RV

Paramètres: aucun

Description: la commande RV permet de demander la lecture des numéros de version et numéro de série du module

@EV vx.y 9170 "MI_uMAC17_9170-nnnnn_" x.x

numéro version Boot

numéro de version Applicatif

numéro de série

Codes Erreur: aucun

Remarque : Cette commande n'est pas utilisable en séquence

5.1.4. MODULE_RESET : réinitialisation

Syntaxe:	[@]MODULE_RESET [ALL]																
Mnémonique:	MRE																
Paramètres:	ALL optionnel : ALL => remise en configuration sortie usine.																
Description:	Cette commande est équivalente à une remise sous-tension du module : un mouvement en cours est interrompu,le compteur de position absolue est forcé à 0, (Attention! Compte tenu du suivi moteur le glissement éventuel de ce dernier peut provoquer un décalage de la position initiale) Les sorties sont forcées à l'état inactif, si une séquence de démarrage automatique est sélectionnée 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): <table><tr><td>#HIGH_SPEED</td><td>: 600 t/mn</td><td>#TORQUE_RATIO</td><td>: 50 %</td></tr><tr><td>#LOW_SPEED</td><td>: 0</td><td>#LOW_TORQUE</td><td>: 20 %</td></tr><tr><td>#RAMPING_TIME</td><td>: 1000 ms</td><td></td><td></td></tr><tr><td>#POSITIVE_END</td><td>: +20000 (+10 tours)</td><td>#NEGATIVE_END</td><td>: -20000 (-10 tours)</td></tr></table> les variables mémorisées (#Mn) sont remises à 0 le démarrage automatique est annulé (#ON_RESET = 0) le mode Standby automatique est activé : Standby_mode = ON la polarité nominale des entrées sorties est restituée (INVERSE_POLARITY = OFF) les butées sont désactivées (HARD_ENDS = OFF ; SOFT_ENDS = OFF) Nota : Les paramètres liaison série, baudrate et adresse, ne sont pas sont pas modifiés.	#HIGH_SPEED	: 600 t/mn	#TORQUE_RATIO	: 50 %	#LOW_SPEED	: 0	#LOW_TORQUE	: 20 %	#RAMPING_TIME	: 1000 ms			#POSITIVE_END	: +20000 (+10 tours)	#NEGATIVE_END	: -20000 (-10 tours)
#HIGH_SPEED	: 600 t/mn	#TORQUE_RATIO	: 50 %														
#LOW_SPEED	: 0	#LOW_TORQUE	: 20 %														
#RAMPING_TIME	: 1000 ms																
#POSITIVE_END	: +20000 (+10 tours)	#NEGATIVE_END	: -20000 (-10 tours)														
Codes Erreur:	FI_non_bool si la valeur fournie n'est pas une clé reconnue.																

Remarque : Cette commande n'est pas utilisable en séquence

5.2. Paramètres

5.2.1. Sauvegarde des paramètres

L'ensemble des paramètres décrits dans ce chapitre est sauvegardé lors des coupures d'alimentation. Les valeurs de ces paramètres peuvent être remises à leurs valeurs à défaut (sortie usine) par la commande :

MODULE_RESET ALL.

5.2.2. Cohérence des vitesses

Les vitesses sont générées au moyen d'un timer 16 bits, aussi pour couvrir la dynamique des vitesses proposées la base de temps est choisie selon la vitesse de consigne désirée. Cette base de temps est utilisée pour générer toutes les vitesses quel que soit le mouvement demandé; ainsi, compte tenu de la dynamique du timer précisée ci-dessus, la vitesse de consigne maximum donnée par l'utilisateur : #HIGH_SPEED impose une limite basse sur les vitesses réalisables.

Si l'on essaie d'imposer une vitesse inférieure à cette limite via le paramètre #LOW_SPEED ou la commande de mouvement à vitesse constante MOVE_SPEED il y a détection d'erreur matérialisée par le flag FI_err_cohé, et la valeur de vitesse est forcée à la valeur min réalisable. La commande est cependant acceptée, il n'y a pas de matérialisation de l'erreur autre que le flag précité de manière à ne pas provoquer de dysfonctionnement de séquence. Les limites de cohérence sont données par le tableau suivant :

Vitesse de consigne (1/100°/mn)	1	160	600	4800	38400	120000
Vitesse min (1/100°/mn)	1	3	12	92	766	

5.2.3. #HIGH_SPEED : vitesse de consigne maximum

Syntaxe:	[@]#HIGH_SPEED := w
Mnémonique:	#HSP
Paramètres:	w = dddddd : vitesse de consigne en 1/100e t/mn Non signé, Limites absolues du paramètre : $1 \leq w \leq 120000$ 10-2 t/mn Valeur à défaut 60000 (600t/mn)
Description:	Définition de la vitesse maximum de consigne de tous les prochains mouvements moteur point à point, y compris après réinitialisation. Cette valeur constitue la limite maximum de vitesse imposée au moteur quel que soit le mouvement demandé. Cette vitesse est utilisée pour définir les paliers à vitesse constante des mouvements de point à point. Remarque : La structure du générateur interne de vitesse impose sa propre résolution en temps, aussi la valeur de fréquence réellement générée peut différer légèrement de la valeur programmée. Lors d'une relecture via la commande READ #PROFILE_SPEED c'est la valeur réellement générée qui est retournée. Attention! La cohérence entre #HIGH_SPEED et #LOW_SPEED est testée au moment de la saisie, il est judicieux de laisser #LOW_SPEED à défaut à 0 lors de la saisie de la vitesse de consigne maximum, puis de préciser sa valeur désirée éventuellement en suite.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur du paramètre fourni est hors limites (donc éventuellement négative) FI_err_cohé si les valeurs et #HIGH_SPEED et #LOW_SPEED ne sont pas cohérentes entre elles (Cf §5.2.1).

5.2.4. #LOW_SPEED : vitesse d'approche

Syntaxe:	[@]#LOW_SPEED := w
Mnémonique:	#LSP
Paramètres:	w = dddddd : vitesse de consigne en 1/100e t/mn Non signé, Limites absolues du paramètre : $0 \leq w \leq \text{\#HIGH_SPEED}$ Valeur à défaut 0
Description:	Cette valeur définit la vitesse de recherche et d'asservissement à la position finale d'un mouvement en cas de dépassement. Lorsque LOW_SPEED = 0, la vitesse de recherche est égale à #HIGH_SPEED / 50 Attention! La cohérence entre #HIGH_SPEED et #LOW_SPEED est testée au moment de la saisie, il est judicieux de laisser #LOW_SPEED à défaut à 0 lors de la saisie de la vitesse de consigne maximum, puis de préciser sa valeur désirée éventuellement en suite.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur du paramètre fourni est hors limites (donc éventuellement négative) FI_err_cohé si les valeurs et #HIGH_SPEED et #LOW_SPEED ne sont pas cohérentes entre elles (Cf §5.2.2).

5.2.5. #RAMPING_TIME : loi d'accélération

Syntaxe:	[@]#RAMPING_TIME := tr
Mnémonique:	#RTI
Paramètres:	tr = dddddd : durée de la rampe en ms Non signé, Limites absolues des paramètres : $1 \leq tr \leq 12\,000$ ms Valeur à défaut 1000 ms
Description:	Ce paramètre définit les durées d'accélération et décélération de tous les mouvements. Ces durées s'entendent pour une excursion de vitesse de #LOW_SPEED à #HIGH_SPEED. Pour des dynamiques plus faibles les temps sont réduits au prorata de l'excursion réelle de vitesse ; de même sur une inversion de vitesse le temps réel est double. Tr permet de définir simultanément la même valeur pour les rampes d'accélération (ta) et décélération (td) : ta = td = tr. Attention! La rampe de décélération n'est pas utilisée pour la commande HALT et les arrêts sur butées en mode immédiat, la décélération est alors seulement définie par le couple moteur et dépend donc de la charge réelle de ce dernier. Exemple : soit à fixer les temps d'accélération et décélération à 150ms #RAMPING_TIME := 150 => ta = 150ms et td = 150ms
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur du paramètre fourni est hors limites (donc éventuellement négative)

5.2.6. #TORQUE_RATIO : Courant moteur

Syntaxe:	[@]#TORQUE_RATIO := Cm
Mnémonique:	#TRA
Paramètres:	Cm = ddd : amplitude du courant ou couple disponible moteur Non signé, Limite $0 \leq Cm \leq 100$ $C_{moteur} = (C_{nom} \times Cm) / 100$ (Cnom : couple maximal délivré par le module) La valeur par défaut à la première mise sous tension ou en sortie usine est 50%, soit la moitié du couple que peut fournir le moteur
Description:	Programmation de la valeur nominale du courant à appliquer au moteur et donc de son couple disponible.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale n'est pas comprise entre 0 et 100

5.2.7. #LOW_TORQUE : Courant de Standby du moteur

Syntaxe:	[@]#LOW_TORQUE := Cstb
Mnémonique:	#LTO
Paramètres:	Cstb = ddd : amplitude du couple de Standby moteur Non signé, Limite $0 \leq Cstb \leq 100$ $C_{standby} = (C_{nom} \times Cstb) / 100$ (Cnom : courant maximal délivré par le module) La valeur par défaut à la première mise sous tension ou en sortie usine est 20%
Description:	Programmation de la valeur de Standby du courant à appliquer au moteur à l'arrêt ou couple minimum disponible pour garantir une raideur minimum. Voir la commande STANDBY_MODE (§ 5.6.1) pour activer ou non le passage au courant de Standby.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale n'est pas comprise entre 0 et 100 Attention! Il n'y a pas de test de cohérence entre le courant nominal et le courant de Standby, il est donc possible d'imposer un courant à l'arrêt supérieur au courant en mouvement.

5.2.8. #POSITIVE_END/NEGATIVE_END : butées logicielles

Syntaxe:	[@] #POSITIVE_END := bp [@] #NEGATIVE_END := bn
Mnémonique:	#PEN #NEN
Paramètres:	bp ou bn = ± d d d d d d d d d position des butées logicielles Numérique Limite : $-2^{31} \leq bp < +2^{31}-1$ et $-2^{31} \leq bn < +2^{31}-1$ Valeurs à défaut : respectivement 20000 et -20000 (soit + ou - 10 tours moteur)
Description:	Fixent les positions des 2 butées logicielles (activées ou non par la commande SOFT_ENDS). Si $bp < bn$ les butées soft sont toujours détectées et le moteur ne peut tourner. Pour ne gérer qu'une seule butée il convient de placer l'autre valeur de butée à sa position extrême (respectivement $+2^{31}-1$ ou -2^{31}). Les butées logicielles sont activées simultanément par la commande SOFT_ENDS ON et désactivées par SOFT_ENDS OFF (§ 5.6.3)
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. Attention! Il n'y a pas de test de dépassement de capacité numérique : si np ou $nb \geq 2^{31}$ ou np ou $nb < -2^{31}$, ni de cohérence entre les 2 limites ($bp \leq bn$)

5.3. Entrées sorties

5.3.1. #OUTPUT: sorties digitales

Syntaxe:	[@]#OUTPUT := Out
Mnémonique:	#OUT
Paramètres:	En format hexadécimal: Out = hnnnn les 4 caractères hexa constituant les 32 bits du port En format binaire: Out = bnn.. nn les 32 bits du port de sortie. Remarque: Seuls les 4 bits de poids faibles sont réellement disponibles en sortie, les bits de poids fort peuvent cependant être relus et peuvent être utilisés à des fins particulières.
Description:	Cette commande permet de contrôler globalement les sorties logiques du module et donc d'activer par exemple certains actionneurs parallèlement au déroulement d'une séquence. Lorsque les poids forts d'un format hexa ou binaire sont omis ils sont pris à 0. Attention! La réponse à cette commande, peut être perturbée par le déroulement d'une séquence ou vice et versa, lorsque la séquence modifie les sorties logiques.
Codes Erreur:	Fl_non_num si la valeur fournie ne correspond pas à une valeur numérique.

5.3.2. #INPUT : entrées digitales

Syntaxe:	@READ #INPUT ⇒ @ #INP = In uniquement accessible en lecture
Mnémonique:	#INP
Paramètres:	In = bb..bb (hh) valeur des 6 bits du port d'entrées digital
Description:	Cette variable retourne les valeurs instantanées des signaux digitaux présents sur les ports d'entrées digitales disponibles. Remarque 1 : Comme le port réel d'entrée ne comprend pas 32 bits, les bits non utilisés sont forcés à 0. Remarque 2 : Les entrées butées hardware sont associées aux entrées logiques 5 et 6. Lorsque le mode de gestion de ces butées n'est pas activé, ces entrées se comportent comme les autres entrées logiques à ceci près qu'elles ne sont pas opto-isolées. De même si le mode butée est actif les capteurs de butées sont visibles via ces deux entrées.
Codes Erreur:	

5.3.3. #INPUT_ANALOG : Entrée analogique

Syntaxe:	@READ #INPUT_ANALOG $\Rightarrow$ @#IAN = Val uniquement accessible en lecture
Mnémonique:	#IAN
Paramètres:	Val = ddddd valeur de l'amplitude en entrée (en mV ; dynamique 0..10V)
Description:	Cette variable retourne la valeur instantanée du signal analogique présent sur l'entrée analogique. La résolution de lecture est toujours de 1 mV. Remarque : La résolution réelle est en fait de 10 bits pour la pleine échelle pleine échelle, soit 9,76mV/bit. La dynamique en tension réelle de l'entrée, elle, est de 0..10V.
Codes Erreur:	

5.3.4. #SUPPLY_VOLTAGE : tension d'alimentation

Syntaxe:	@READ #SUPPLY_VOLTAGE $\Rightarrow$ @# SVO = Val uniquement accessible en lecture
Mnémonique:	#SVO
Paramètres:	Val = ddddd valeur de la tension d'alimentation en mV
Description:	Cette variable retourne la valeur moyenne de la tension d'alimentation du module exprimée en mV.
Codes Erreur:	

5.3.5. #TEMPERATURE : température du boîtier

Syntaxe:	@READ #TEMPERATURE $\Rightarrow$ @#TEM = Val uniquement accessible en lecture
Mnémonique:	#TEM
Paramètres:	Val = +/-ddd valeur de la température en °C
Description:	Cette variable retourne la valeur de la température interne du module exprimée en °C. Remarque : La mesure de température n'est pas fonctionnelle lorsque l'entrée IN6 (But+) est activée. La protection thermique du module reste cependant opérationnelle.
Codes Erreur:	

5.4. Variables

5.4.1. Paramètres, variables système et variables utilisateur

Les paramètres du module et les entrées sorties décrits plus haut, sont des données modifiables et utilisables par l'utilisateur soit par des commandes directes soit par des séquences, à ce titre elle se comportent comme des variables appelées système par opposition aux variables utilisateur décrites ci dessous. Quelque soit leur type, les variables peuvent être modifiées, relues ou manipulées au moyen d'opérations.

5.4.2. Opérations sur les variables

Un certain nombre de fonctions permettent de réaliser des assignations, des opérations ou des tests sur les variables. Ces fonctions peuvent être indifféremment utilisées en commandes directes, ou dans les séquences. Elles prennent la forme suivante :

$\#n := \#m \text{ (op) } \#p$ et avec une constante $\#n := \#m \text{ (op) } \text{cst}$
 ou une simple assignation
 $\#n := \#m$ et avec une constante $\#n := \text{cst}$

Ou (op) peut prendre les significations suivantes :

+ addition
 - soustraction
 * multiplication
 / division entière
 & et logique (and)
 | ou logique (or)
 >, >=, =, !=, <, <= tests supériorité, égalité, inégalité et infériorité (= 1 si vrai, 0 sinon)

Lorsque ces fonctions sont utilisées dans une instruction de test, la variable résultat est omise. D'où la forme simplifiée :

$\#m \text{ (op) } \#p$ et avec une constante $\#m \text{ op } \text{cst}$

Attention! Il convient de séparer l'opérateur du nom des variables ou valeurs numériques par des espaces.

Note : Il n'est possible de définir qu'une seule opération par ligne de séquence par exemple :

$\#VAR := \#VAR1 * \#Var2 + 1$ provoque une erreur.

Une opération ne peut comprendre qu'une seule valeur constante.

5.4.3. #Vn : variable utilisateur

Syntaxe:	[@]#Vn := Val
Mnémonique:	#Vn n: numéro de la variable utilisateur ($1 \leq n \leq 4$)
Paramètres:	Val = ±ddddddddd : valeur de la variable $-2^{31} \leq \text{Val} < 2^{31}-1$ Hhhhhhhh Bbbbbbbb : éventuellement en hexa ou binaire Numérique Limite : $-2147483648 \leq \text{Val} \leq +2147483647$
Description:	Variables totalement libres d'emploi pour l'utilisateur, forcées à 0 à la mise sous tension ou au reset
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. Attention! Il n'y a pas de test de dépassement de capacité numérique si val $2^{31} \geq$ ou $< -2^{31}$

5.4.4. #Mn : variable utilisateur mémorisée

Syntaxe:	[@]#Mn := Val
Mnémonique:	Mn n: numéro de la variable utilisateur mémorisée (1 ≤ n ≤ 4)
Paramètres:	Val = ±ddddddddd : valeur de la variable $-2^{31} \leq \text{Val} < 2^{31}-1$ Hhhhhhhh Bbbbbbbb : éventuellement en hexa ou binaire Numérique Limite : $-2147483648 \leq \text{Val} \leq +2147483647$
Description:	Variables totalement libres d'emploi pour l'utilisateur, mémorisées d'une mise sous tension à l'autre ou au reset.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. Attention! Il n'y a pas de test de dépassement de capacité numérique si val $2^{31} \geq$ ou $< -2^{31}$

5.5. Gestion des mouvements

5.5.1. #POSITION : compteur de position absolue

Syntaxe:	[@]#POSITION := Pr
Mnémonique:	#POS
Paramètres:	Pr = ±ddddddddd : nouvelle valeur de la position absolue Numérique, Limite : $-2147483648 \leq Pr < +2147483647$
Description:	Contient la valeur de la position absolue du moteur, la modification de cette variable système permet de redéfinir sans mouvement une position de référence différente. Remarque : La modification de la position absolue est refusée si un mouvement de point à point est en cours d'exécution.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. Attention! Il n'y a pas de test de dépassement de capacité numérique : si $Pr \geq 2^{31}$ ou $Pr < -2^{31}$ la valeur retenue par le module sera aberrante.

5.5.2. #PROFILE_SPEED : vitesse courante

Syntaxe:	@READ #PROFILE_SPEED ⇒ @PSP= Vr uniquement accessible en lecture
Mnémonique:	#PSP
Paramètres:	Vr = ±dddddd : valeur de la vitesse calculée courante en 1/100° t/mn Numérique, Limite : $-120000 \leq Vr \leq +120000$
Description:	Retourne la valeur courante de la vitesse moteur en 1/100° t/mn. Attention! Cette valeur n'est pas exactement la vitesse réelle, mais seulement la vitesse recherchée du moteur. Elle tient cependant compte des phases d'accélération ou décélération. Ce n'est donc pas la vitesse de consigne #HIGH_SPEED, ou le paramètre de la commande MOVE_SPEED, notamment dans les phases d'accélération ou décélération.
Codes Erreur:	

5.5.3. MOVE_TO : Mouvement absolu

Syntaxe:	[@]MOVE_TO Pa (ou [@]MOVE_TO #VAR)
Mnémonique:	MTO
Paramètres:	Pa = ±ddddddddd : position absolue désirée Numérique Unité : incrément de position soit 1/2000° tr Limites : $-2147483648 \leq Pa \leq +2147483647$ (en incrément de position)
Description:	Exécution d'un mouvement de type absolu. Le module exécute le mouvement nécessaire pour positionner le moteur à la position absolue définie par la commande en respectant la loi d'accélération décélération et la vitesse de consigne définie par #RAMPING_TIME et #HIGH_SPEED. S'il s'agit d'un premier mouvement après mise sous tension ou reset, le module initialise préalablement l'autocommutateur avant d'effectuer le mouvement. Si le module se trouve déjà à la position requise, il n'y a pas de mouvement. Si le module est déjà en mouvement la commande est refusée.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_refus si le moteur est déjà en mouvement Attention! Il n'y a pas de test de dépassement de capacité numérique : si $Pa \geq 2^{31}$ ou $Pa < -2^{31}$

5.5.4. MOVE_ON : mouvement relatif

Syntaxe:	[@]MOVE_ON [±]n (ou [@]MOVE_ON #VAR)
Mnémonique:	MON
Paramètres:	n = dddddddd valeur de déplacement Numérique Unité : incrément de position soit 1/2000° tr Limite $ n \leq 2147483647$ incréments de position
Description:	Exécution d'un mouvement d'amplitude n dans le sens donné par le signe de la consigne. S'il n'y a pas de paramètre, il n'y a pas de mouvement. Si le module est déjà en mouvement, la commande est refusée.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_refus si le moteur est déjà en mouvement Attention! Il n'y a pas de test de dépassement de capacité numérique : si $Pa \geq 2^{31}$ ou $Pa < -2^{31}$

5.5.5. MOVE_SPEED : Mouvement à vitesse imposée

Syntaxe:	[@]MOVE_SPEED [±] v ([@]MOVE_SPEED #VAR)
Mnémonique:	MSP
Paramètres:	v = ddddd nouvelle valeur de vitesse Numérique Unité : 1/100 t/mn Limites $1 < v < 120000$ (1/100 t /mn) (0,033 < V < 4000 p/s) Le sens du mouvement est défini par le signe de la consigne que celui-ci soit explicite ou non
Description:	Lancement d'un mouvement accéléré (ou décéléré) jusqu'à la vitesse v puis maintien de la vitesse atteinte jusqu'à une commande d'arrêt ou d'initialisation (STOP, HALT, MODULE_RESET ou interruption par condition logique). Le mouvement est exécuté dans le sens donné par le signe de la consigne; si le moteur est déjà en mouvement dans un sens opposé, le moteur est décéléré jusqu' à vitesse nulle puis accéléré dans le sens désiré, si le sens du mouvement est identique le moteur est accéléré ou décéléré suivant le sens de l'écart par rapport à la vitesse courante pour l'amener à la vitesse désirée. Si la valeur de vitesse donnée est 0 la vitesse est ramenée à 0 en suivant la loi de décélération comme s'il s'agissait d'une commande STOP. Si la vitesse demandée est trop faible compte tenu de la cohérence avec la vitesse de consigne, c'est la vitesse minimum cohérente qui est utilisée. Si la valeur de vitesse demandée est supérieure à la vitesse de consigne définie par #HIGH_SPEED, la vitesse est bornée à cette dernière Remarque: Cette commande peut intervenir sur un mouvement de point à point, elle continue le mouvement à la vitesse demandée, sans tenir compte de la cible initiale.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur absolue du paramètre fourni est supérieure à #HIGH_SPEED FI_err_cohe si la valeur de vitesse demandée n'est pas cohérente avec la vitesse de consigne : #HIGH_SPEED Nota : Les deux dernières erreurs ne provoquent pas de refus de la commande, mais seulement une correction de la valeur de la consigne passée que ce soit en commande directe ou en séquence.

5.5.6. STOP : Arrêt avec décélération

Syntaxe:	[@]STOP [OUT]
Mnémonique:	STO
Paramètres:	OUT Permet en séquence de passer à la ligne d'instruction suivante sans attendre la fin du mouvement.
Description:	Le mouvement en cours (direct ou séquence) est décéléré jusqu'à vitesse minimum puis arrêté. Lorsque la gestion du courant de Standby est activée le courant moteur est forcé à sa valeur de repos. (Cf §5.6.1) Les prochains mouvements moteurs seront effectués à partir de la vitesse minimum. La commande STOP en commande immédiate termine la séquence éventuellement en cours.
Codes Erreur:	

Attention! La commande stop ne garantit pas forcément la fin du mouvement, notamment en cas de blocage ou en absence de puissance moteur. Le mouvement ne se termine réellement que lorsque le nombre de pas prévus pour la décélération est réellement réalisé.

5.5.7. HALT : Arrêt d'urgence

Syntaxe:	[@]HALT
Mnémonique:	HAL
Paramètres:	Aucun
Description:	Le mouvement en cours (direct ou séquence) est arrêté immédiatement sans décélération. Lorsque la gestion du courant de Standby est activée le courant moteur est forcé à sa valeur de repos (Cf §5.6.1). Les prochains mouvements moteurs seront effectués à partir de la vitesse minimum. La commande HALT utilisée en commande immédiate termine la séquence éventuellement en cours.
Codes Erreur:	

5.6. MODES

5.6.1. STANDBY_MODE : Gestion du Standby

Syntaxe:	[@]STANBY_MODE ON ou [@]STANBY_MODE OFF
Mnémonique:	SMO
Paramètres:	ON ou OFF
Description:	Activation ou désactivation de la gestion de courant de repos moteur, lorsque ce mode est appliqué: * Le courant moteur est forcé au courant de Standby lorsque le moteur est à l'arrêt (sauf en cas de coupure puissance). * Le courant nominal est restitué pendant les mouvements. Si ce mode est off le moteur est toujours alimenté au courant nominal (sauf hors puissance). La configuration du mode Standby est sauvegardée pendant les coupures d'alimentation.
Codes Erreur:	FI_non_bool si la valeur fournie n'est pas une clé reconnue.

5.6.2. HARD_ENDS : Butées hardware

Syntaxe:	[@]HARD_ENDS [OFF][POS][NEG][ALL][HALT][STOP]
Mnémonique:	HEN
Paramètres:	OFF, POS, NEG, ALL, HALT, STOP
Description:	Activation ou désactivation de la gestion des butées hardware globalement ou individuellement pour chaque sens de rotation. Les clés ALL et OFF activent ou désactivent la gestion des deux butées simultanément. Les clés POS et NEG activent respectivement les butées positive et négative. Dans le mode de fonctionnement butée, deux entrées logiques (E5 et E6) sont utilisées en tant que détection de butées. L'activation de l'entrée logique E6 (ou Butée+) lors d'un mouvement de sens horaire, provoque un arrêt d'urgence du mouvement, exactement comme la commande STOP si les butées sont configurées avec la clé STOP, ou comme la commande HALT si les butées sont configurées avec la clé HALT. Cette entrée n'a pas d'action sur les mouvements en sens anti-horaire. De même, l'activation de l'entrée logique E5 (ou Butée-) lors d'un mouvement de sens anti-horaire provoquera son arrêt d'urgence (mouvement ou séquence). La configuration du mode Butée est sauvegardée pendant les coupures d'alimentation. Remarque : La polarité de gestion des entrées butées est éventuellement modifiable via la commande INVERSE_POLARITY. Attention! Une commande ne peut comporter qu'une seule clé, aussi la configuration du mode d'arrêt d'urgence et des butées doit se faire au moyen de 2 commandes distinctes.
Codes Erreur:	FI_non_bool si la valeur fournie n'est pas une clé reconnue.

5.6.3. SOFT_ENDS : Butées software

Syntaxe:	[@]SOFT_ENDS [OFF][POS][HALT][STOP]
Mnémonique:	SEN
Paramètres:	ON, OFF, HALT, STOP
Description:	Activation ou désactivation de la gestion des butées software. Les limites de positions admissibles sont définies par les variables système #POSITIVE_END pour le sens horaire et #NEGATIVE_END pour le sens anti-horaire.(cf §5.2.8) Le type d'arrêt est géré par les directives HALT et STOP comme pour les butées hardware, ces directives définissent le type d'arrêt d'urgence commun à l'ensemble des butées. La configuration du mode Butée est sauvegardée pendant les coupures d'alimentation. Attention! Une commande ne peut comporter qu'une seule clé, aussi la configuration du mode d'arrêt d'urgence et des butées doit se faire au moyen de 2 commandes distinctes.
Codes Erreur:	FI_non_bool si la valeur fournie n'est pas une clé reconnue. Attention! Il n'y a pas de test de cohérence entre les 2 limites : (#POSITIVE_END ≤ #NEGATIVE_END)

5.6.4. INVERSE_POLARITY : inversion de polarité

Syntaxe:	[@]INVERSE_POLARITY [ALL][OFF][IN][OUT]
Mnémonique:	IPO
Paramètres:	ALL , OFF , IN ou OUT
Description:	Modification de la polarité des entrées et sorties digitales. La clé ALL modifie la polarité de l'ensemble des entrées et sorties logiques, la clé OFF restitue la polarité initiale de l'ensemble, les clés IN et OUT modifient respectivement seulement la polarité des entrées ou des sorties. La polarité à défaut des entrées/sorties correspond à 0 pour une entrée ou une sortie non connectée. Une entrée opto-isolée est active lorsque un courant y est forcé, une sortie opto-isolée active autorise le passage du courant. Attention! Cette commande n'est pas rétroactive, l'inversion de polarité ne met pas à jour l'état des sorties logiques. Si l'état réel des sorties doit être modifié il convient de réécrire la variable #OUTPUT. La configuration d'inversion de polarité est sauvegardée pendant les coupures d'alimentation.
Codes Erreur:	FI_non_bool si la valeur fournie n'est pas une clé reconnue.

5.7. Gestion des séquences

Le module MICROMAC permet de mémoriser jusqu'à 75 lignes de commandes sous la forme de séquences. Les lignes de séquence, lorsqu'elles sont valides, sont mémorisées au fur et à mesure de leur saisie en EEPROM de façon à être conservées d'une mise sous tension à l'autre. La taille mémoire nécessaire à la mémorisation des séquences dépend de la nature des commandes utilisées. Le nombre de lignes de séquences mémorisables est notamment plus faible si l'on utilise des opérations sur variables utilisant des constantes.

Attention! Il n'y a pas de gestion dynamique de l'espace mémoire, aussi la réécriture de lignes de séquence ne réutilise pas forcément la mémoire préalablement utilisée, il est donc conseillé d'effacer préalablement l'ensemble de la mémoire et de réécrire la totalité de la séquence lorsque l'on désire effectuer une modification, si l'on veut disposer de la totalité de l'espace mémoire.

5.7.1. OPEN/CLOSE_SEQ : édition des séquences

Syntaxe:	[@]OPEN_SEQ nl [@]CLOSE_SEQ
Mnémonique:	OSE CSE
Paramètres:	ns = dd : numéro de la première ligne de séquence à modifier Limites : $0 \leq \text{n° de ligne} \leq 75$ Si nl = 0 ou est omis, la mémoire séquence est préalablement effacée, ce qui permet de disposer de la totalité de la mémoire, l'écriture des lignes de séquence s'effectuera à partir de la ligne 1.
Description:	Ces commandes permettent respectivement d'ouvrir et fermer le mode d'édition de séquence pour permettre la saisie des phases. Entre ces 2 commandes toutes les commandes reçues sont mémorisées une à une dans les lignes de séquence successivement à partir de la ligne d'ouverture donnée, elles ne sont pas exécutées sur le moment. Seule la commande MODULE_RESET est directement interprétée et ne peut être utilisée en séquence. Il est possible de donner le numéro de ligne si désiré au moyen de la directive :n.
Codes Erreur:	FI_hors_seq si Le N° de ligne de Séquence fourni est hors limite. FI_out_mem la mémoire non volatile est saturée. FI_non_autorisé si le moteur est en mouvement ou si une séquence est en cours d'exécution pour OPEN_SEQ ou si le module n'est pas en édition de séquence pour CLOSE_SEQ.

5.7.2. :n : déclaration de phases de séquence

Syntaxe:	[@][:nl]
Mnémonique:	Aucun
Paramètres:	nl numéro de la ligne de séquence Limites : $1 \leq n^{\circ} \text{ de ligne} \leq 75$
Description:	Il n'y a pas de différence entre l'écriture d'une séquence et l'envoi de commande immédiate. Une fois le mode d'édition de séquence ouvert par la commande OPEN_SEQ, les phases sont écrites les unes après les autres comme s'il s'agissait de commandes immédiates, le numéro de phase est automatiquement incrémenté. Le numéro de ligne peut cependant être précisé pour plus de clarté surtout dans le cas où l'on désire changer de numéro de ligne ou revenir sur une ligne déjà écrite, il est alors possible d'imposer le numéro de ligne au moyen de la directive :n, où n est le numéro de la ligne où l'on veut écrire la commande . Exemple : 05:10 MOVE_TO 500 place la commande MOVE_TO 500 à la ligne 10 de la séquence du module d'adresse 5
Codes Erreur:	FI_hors_lim si Le N° de ligne de Séquence fourni est hors limite. FI_out_mem la mémoire non volatile est saturée. FI_refus si le module n'est pas en mode d'édition de séquence.

5.7.3. START_SEQUENCE : lancement d'une séquence

Syntaxe:	[@]START_SEQ nl
Mnémonique:	SSE
Paramètres:	nl = dd : numéro de la ligne de séquence à exécuter en premier Limites : $1 \leq n^{\circ} \text{ de ligne} \leq 75$
Description:	lance l'exécution d'une séquence à partir de la ligne d'adresse nl
Codes Erreur:	FI_hors_lim si un n° de ligne fourni est hors limite FI_refus si le moteur est déjà en mouvement ou si une séquence est en cours d'exécution

5.7.4. #ON_RESET : séquence de démarrage

Syntaxe:	[@]#ON_RESET := nl [@]#ON_RESET := 0
Mnémonique:	#ORE
Paramètres:	nl = dd : numéro de la première ligne de séquence à démarrer au reset Limites : $1 \leq n^{\circ} \text{ de ligne} \leq 75$
Description:	Définition du numéro de ligne à exécuter à la mise sous tension ou à tout reset. Si le numéro de ligne est 0 ou absent aucune séquence n'est lancée au démarrage
Codes Erreur:	FI_hors_lim si un n° de ligne fourni est hors limite

5.7.5. WAIT : temporisation

Syntaxe:	WAIT [time] WAIT [#var]
Mnémonique:	WAI
Paramètres:	time = dddd : durée de la temporisation en ms (valeur négative pour time out) Limites absolues des paramètres : $0 \leq t \leq 65\,000$ ms
Description:	Cette instruction permet d'imposer un délai dans le déroulement d'une séquence ; elle n'a aucun effet en commande immédiate. L'instruction WAIT sans paramètre ou avec une valeur nulle permet d'attendre la fin d'un éventuel mouvement en cours. Une valeur de paramètre négative effectue la même attente de fin de mouvement avec une valeur de time out correspondant à la valeur absolue du paramètre. Attention! Le dépassement du time out n'entraîne pas l'arrêt du mouvement en cours. Si nécessaire il convient de l'arrêter via une commande STOP ou HALT (cet arrêt préalable est nécessaire dans le cas où l'on désire enchaîner sur un mouvement de point à point si l'on ne veut pas que ce dernier soit refusé).
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale n'est pas comprise entre 0 et 65000 FI_out_mem la mémoire non volatile est saturée

5.7.6. JUMP: saut entre phases

Syntaxe:	JUMP nl
Mnémonique:	JUM
Paramètres:	nl = dd : numéro de la ligne de phase suivante à exécuter Limites : $1 \leq n^{\circ} \text{ de ligne} \leq 75$
Description:	Le séquenceur continue le déroulement de la séquence à la ligne nl. Un saut à la ligne 0 termine la séquence, y compris la séquence appelante, dans le cas d'une sous séquence.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale d'adresse de saut est hors limite FI_out_mem la mémoire non volatile est saturée

5.7.7. JUMP_RELATIVE : saut relatif entre phases

Syntaxe:	JUMP_RELATIVE nl
Mnémonique:	JRE
Paramètres:	nl = ±dd : valeur du nombre de lignes à sauter Limites : $-75 \leq \text{saut} \leq 75$
Description:	Le séquenceur continue le déroulement de la séquence à la ligne courante + nl. n peut prendre une valeur négative. Un saut relatif de 0 boucle indéfiniment sur la même ligne de séquence. Un saut à la ligne 0 termine la séquence, y compris la séquence appelante, dans le cas d'une sous séquence. Un saut à une ligne hors champ ou non définie à l'exécution provoque l'arrêt de la séquence, donc par exemple si le numéro de la ligne pointée (ligne courante + nl) est < 0.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale d'adresse de saut est hors limite FI_out_mem la mémoire non volatile est saturée

5.7.8. IF... JUMP : saut conditionnel

Syntaxe:	IF val1 cond val2 JUMP nl
Mnémonique:	IF JUM
Paramètres:	nl numéro de la phase suivante à exécuter si la condition est réalisée val1 et val2 sont soit des variables utilisateur ou système soit des constantes cond est le code opératoire servant de condition Limites : $0 \leq \text{saut} \leq 75$
Description:	La séquence continue à la phase chronologique suivante si la condition de test est fausse, si celle ci est vraie la séquence continue à la phase précisée par nl. Pour un calcul arithmétique ou logique la condition vraie correspond à un résultat différent de 0, dans le cas contraire la condition est fausse. Un saut à la ligne 0 termine la séquence, y compris la séquence appelante, dans le cas d'une sous séquence. Exemples : :11 IF #V3 = #V4 JUMP 25 saut à phase 25 si #V3=#V4 sinon continue à la phase chronologique suivante 12 : 45 IF #V4 + 45 JUMP 25 saut à phase 25 si en final #V4 + 45 <> 0
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale d'adresse de saut est hors limite FI_incoher si le champ de description ne respecte pas les syntaxes prévues FI_out_mem la mémoire non volatile est saturée

5.7.9. IF... JRE : saut conditionnel relatif

Syntaxe:	IF val1 cond val2 JRE nl
Mnémonique:	IF JRE
Paramètres:	nl = ±dd : valeur du nombre de lignes à sauter Limites : $-75 \leq \text{saut} \leq 75$
Description:	La séquence continue le déroulement de la séquence à la ligne courante +nl si la condition de test est vraie, si celle-ci est fausse il continue à la ligne suivante. Pour un calcul arithmétique ou logique, la condition correspond à un résultat différent de 0, dans le cas contraire la condition est fausse. Exemples : : 11 IF #V3 = #V4 JRE2 saut à la ligne 13 si #V3=#V4 : 45 IF #V4 + 45 JRE-3 saut à la ligne 42 si #V4 + 45 > 0
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_lim si la valeur finale d'adresse de saut est hors limite FI_incoher si le champ de description ne respecte pas les syntaxes prévues FI_out_mem la mémoire non volatile est saturée

5.7.10. CALL/RETURN: sous séquence

Syntaxe:	CALL nl appel de la sous-séquence RETURN retour
Mnémonique:	CAL RET
Paramètres:	nl = dd numéro de la première ligne de la sous-séquence à exécuter Limites : $1 \leq \text{n}^\circ \text{ de ligne} \leq 75$
Description:	Sur l'instruction CALL, le séquenceur continue le déroulement de la séquence à la ligne nl, lorsque l'instruction RETURN est rencontrée, le séquenceur reprend le déroulement de la séquence à la phase naturelle suivant l'instruction CALL appelante. La pile de gestion des sous séquences contient 5 niveaux. Nota : l'instruction RETURN sert aussi à revenir au mode commande directe en fin de séquence lorsque celle-ci a seulement été appelée par la commande de lancement de processus : START_SEQ.
Codes Erreur:	FI_non_num si la valeur fournie ne correspond pas à une valeur numérique. FI_hors_champ si la valeur finale n'est pas comprise entre 0 et 75 (à l'exécution) FI_out_mem la mémoire non volatile est saturée

5.7.11. #LINE : ligne courante exécutée

Syntaxe:	@READ #LINE => @#LIN = nl
Mnémonique:	#LIN
Paramètres:	nl : numéro de la ligne de séquence en cours d'exécution accessible en lecture seule
Description:	Cette variable retourne la valeur de la ligne de séquence en cours d'exécution.
Codes Erreur:	

5.7.12. STEP : pas à pas

Syntaxe:	@STEP [nl] exécution de la ligne de séquence nl
Mnémonique:	STE
Paramètres:	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 suivant la dernière ligne exécutée.
Codes Erreur:	

5.7.13. \$upLoad : relecture de la mémoire séquence

Syntaxe:	\$upload
Mnémonique:	\$upload
Paramètres:	néant
Description:	WinSim2 relit l'ensemble de la mémoire séquence et crée un fichier texte de l'ensemble des commandes nécessaires à recharger l'espace séquence. L'utilisateur peut facilement modifier le fichier grâce à l'éditeur intégré à WinSim et le sauvegarder (.CMW : extension à défaut).
Exemple:	<pre> ; ; Séquences et Datas récupérées par UpLoad ; 00MODULE_RESET ALL ; ; Identification du module UMAC17: ;00EV v4.7 9170 "MI_uMAC17_9170-00007_" ; séquences : 00OPEN_SEQ 00:01 #V1 := #INP 00:02 JUM 1 00CLOSE_SEQ ; listes des variables mémorisées non nulles : 00#M2 := -45 ; la phase de démarrage (si elle existe) : </pre>
Remarque:	Cette commande s'utilise sans adresse, celle-ci est automatiquement gérée par WinSim2.

5.8. Contrôle

5.8.1. #STATUS: état du module

Syntaxe:	@READ h#STATUS ⇒ @STA = hEtat uniquement accessible en lecture ou @READ b#STATUS ⇒ @STA = bEtat																																																																																																										
Mnémonique:	#STA																																																																																																										
Paramètres:	Etat = hhhhhhhh ensemble des flags d'état du module (à lire en hexa ou binaire) Etat = bbb...bbb																																																																																																										
Description:	liste des Flags d'état du module (état actif à 1) <table border="1"> <thead> <tr> <th></th><th>bit</th><th>Flag</th><th>Description</th></tr> </thead> <tbody> <tr> <td rowspan="8">poids fort</td><td>32</td><td>FI_interrupt</td><td>mouvement ou séquence interrompu</td></tr> <tr><td>31</td><td></td><td></td></tr> <tr><td>30</td><td></td><td></td></tr> <tr><td>29</td><td></td><td></td></tr> <tr><td>28</td><td></td><td></td></tr> <tr><td>27</td><td>FI_ass</td><td>vitesse asservie</td></tr> <tr><td>26</td><td>FI_mvt</td><td>mouvement en cours</td></tr> <tr><td>25</td><td>FI_pon</td><td>moteur sous puissance</td></tr> <tr> <td rowspan="8"></td><td>24</td><td>FI_warning</td><td></td></tr> <tr><td>23</td><td></td><td></td></tr> <tr><td>22</td><td></td><td></td></tr> <tr><td>21</td><td></td><td></td></tr> <tr><td>20</td><td>FI_etat_soft-</td><td>butée logicielle négative détectée</td></tr> <tr><td>19</td><td>FI_etat_soft+</td><td>butée logicielle positive détectée</td></tr> <tr><td>18</td><td>FI_etat_hard-</td><td>butée hard négative détectée</td></tr> <tr><td>17</td><td>FI_etat_hard+</td><td>butée hard positive détectée</td></tr> <tr> <td rowspan="8"></td><td>16</td><td>FI_edit_seq</td><td>Séquence en cours d'édition</td></tr> <tr><td>15</td><td>FI_exec_seq</td><td>séquence en cours d'exécution</td></tr> <tr><td>14</td><td>FI_inv_pol_out</td><td>inversion de la polarité des sorties</td></tr> <tr><td>13</td><td>FI_inv_pol_in</td><td>inversion de la polarité des entrées</td></tr> <tr><td>12</td><td></td><td></td></tr> <tr><td>11</td><td></td><td></td></tr> <tr><td>10</td><td>FI_mode_urg</td><td>arrêt sans décélération sur les butées</td></tr> <tr><td>9</td><td></td><td></td></tr> <tr> <td rowspan="8">poids faible</td><td>8</td><td></td><td></td></tr> <tr><td>7</td><td>FI_but_soft</td><td>gestion des butées logicielles activée</td></tr> <tr><td>6</td><td>FI_but_hard-</td><td>gestion de la butée hard négative activée</td></tr> <tr><td>5</td><td>FI_but_hard+</td><td>gestion de la butée hard positive activée</td></tr> <tr><td>4</td><td></td><td></td></tr> <tr><td>3</td><td></td><td></td></tr> <tr><td>2</td><td></td><td></td></tr> <tr><td>1</td><td>FI_mode_stb</td><td>gestion du Standby activée</td></tr> </tbody> </table>				bit	Flag	Description	poids fort	32	FI_interrupt	mouvement ou séquence interrompu	31			30			29			28			27	FI_ass	vitesse asservie	26	FI_mvt	mouvement en cours	25	FI_pon	moteur sous puissance		24	FI_warning		23			22			21			20	FI_etat_soft-	butée logicielle négative détectée	19	FI_etat_soft+	butée logicielle positive détectée	18	FI_etat_hard-	butée hard négative détectée	17	FI_etat_hard+	butée hard positive détectée		16	FI_edit_seq	Séquence en cours d'édition	15	FI_exec_seq	séquence en cours d'exécution	14	FI_inv_pol_out	inversion de la polarité des sorties	13	FI_inv_pol_in	inversion de la polarité des entrées	12			11			10	FI_mode_urg	arrêt sans décélération sur les butées	9			poids faible	8			7	FI_but_soft	gestion des butées logicielles activée	6	FI_but_hard-	gestion de la butée hard négative activée	5	FI_but_hard+	gestion de la butée hard positive activée	4			3			2			1	FI_mode_stb	gestion du Standby activée
	bit	Flag	Description																																																																																																								
poids fort	32	FI_interrupt	mouvement ou séquence interrompu																																																																																																								
	31																																																																																																										
	30																																																																																																										
	29																																																																																																										
	28																																																																																																										
	27	FI_ass	vitesse asservie																																																																																																								
	26	FI_mvt	mouvement en cours																																																																																																								
	25	FI_pon	moteur sous puissance																																																																																																								
	24	FI_warning																																																																																																									
	23																																																																																																										
	22																																																																																																										
	21																																																																																																										
	20	FI_etat_soft-	butée logicielle négative détectée																																																																																																								
	19	FI_etat_soft+	butée logicielle positive détectée																																																																																																								
	18	FI_etat_hard-	butée hard négative détectée																																																																																																								
	17	FI_etat_hard+	butée hard positive détectée																																																																																																								
	16	FI_edit_seq	Séquence en cours d'édition																																																																																																								
	15	FI_exec_seq	séquence en cours d'exécution																																																																																																								
	14	FI_inv_pol_out	inversion de la polarité des sorties																																																																																																								
	13	FI_inv_pol_in	inversion de la polarité des entrées																																																																																																								
	12																																																																																																										
	11																																																																																																										
	10	FI_mode_urg	arrêt sans décélération sur les butées																																																																																																								
	9																																																																																																										
poids faible	8																																																																																																										
	7	FI_but_soft	gestion des butées logicielles activée																																																																																																								
	6	FI_but_hard-	gestion de la butée hard négative activée																																																																																																								
	5	FI_but_hard+	gestion de la butée hard positive activée																																																																																																								
	4																																																																																																										
	3																																																																																																										
	2																																																																																																										
	1	FI_mode_stb	gestion du Standby activée																																																																																																								
Codes Erreur:																																																																																																											

Remarque : Les flags de détection de butées traduisent l'état instantané des butées, ils sont donc relâchés lorsque la butée correspondante n'est plus active.

Attention! Les bits non définis ont des valeurs indifférentes qui peuvent parfaitement évoluer durant le fonctionnement du module car ces bits peuvent correspondre à certains états internes.

5.8.2. #ERROR: Compte rendu d'erreur

Syntaxe:	@READ h#ERROR ⇒ @ERR = hError																																																																																																									
Mnémonique:	ERR																																																																																																									
Paramètres:	Error = hhhhhhhh ensemble des flags d'erreur du module (à lire en hexa ou binaire)																																																																																																									
Description:	liste des Flags d'erreur du module <table><thead><tr><th></th><th>bit</th><th>Flag</th><th>Description</th></tr></thead><tbody><tr><td rowspan="8">poids fort</td><td>32</td><td></td><td></td></tr><tr><td>31</td><td></td><td></td></tr><tr><td>30</td><td></td><td></td></tr><tr><td>29</td><td></td><td></td></tr><tr><td>28</td><td></td><td></td></tr><tr><td>27</td><td></td><td></td></tr><tr><td>26</td><td></td><td></td></tr><tr><td>25</td><td></td><td></td></tr><tr><td rowspan="7"></td><td>24</td><td></td><td></td></tr><tr><td>23</td><td>FI err typ</td><td>erreur sur le type de paramètre</td></tr><tr><td>22</td><td>FI hors champ</td><td>point hors champ des séquences</td></tr><tr><td>21</td><td>FI out mem</td><td>dépassement taille mémoire</td></tr><tr><td>20</td><td>FI hors seq</td><td>dépassement taille séquence</td></tr><tr><td>19</td><td>FI def soft</td><td></td></tr><tr><td>18</td><td>FI_non_autorise</td><td>commande non autorisée selon l'état du module</td></tr><tr><td></td><td>17</td><td>FI_req_add</td><td>adresse module requise</td></tr><tr><td rowspan="8"></td><td>16</td><td></td><td></td></tr><tr><td>15</td><td>FI refus</td><td>commande refusée selon l'état du module</td></tr><tr><td>14</td><td>FI err cohe</td><td>erreur de cohérence entre paramètres</td></tr><tr><td>13</td><td>FI incoher</td><td>syntaxe incohérente</td></tr><tr><td>12</td><td>FI non_def</td><td>paramètre non défini</td></tr><tr><td>11</td><td>FI nom_var</td><td>nom de variable erroné</td></tr><tr><td>10</td><td>FI non_bool</td><td>paramètre fourni non booléen</td></tr><tr><td>9</td><td>FI non_num</td><td>paramètre fourni non numérique</td></tr><tr><td rowspan="8">poids faible</td><td>8</td><td>FI err_calc</td><td>erreur de calcul (division par 0 ...)</td></tr><tr><td>7</td><td>FI hors_lim</td><td>paramètre ou variable hors limite</td></tr><tr><td>6</td><td></td><td></td></tr><tr><td>5</td><td>FI err_ov</td><td>erreur surtension</td></tr><tr><td>4</td><td>FI err_uv</td><td>erreur sous tension</td></tr><tr><td>3</td><td>FI err_cc</td><td>erreur court circuit moteur</td></tr><tr><td>2</td><td>FI err_dth</td><td>disjonction thermique</td></tr><tr><td>1</td><td></td><td></td></tr></tbody></table> Exemple #ERR = h200 => le paramètre fourni avec la commande doit être boolean		bit	Flag	Description	poids fort	32			31			30			29			28			27			26			25				24			23	FI err typ	erreur sur le type de paramètre	22	FI hors champ	point hors champ des séquences	21	FI out mem	dépassement taille mémoire	20	FI hors seq	dépassement taille séquence	19	FI def soft		18	FI_non_autorise	commande non autorisée selon l'état du module		17	FI_req_add	adresse module requise		16			15	FI refus	commande refusée selon l'état du module	14	FI err cohe	erreur de cohérence entre paramètres	13	FI incoher	syntaxe incohérente	12	FI non_def	paramètre non défini	11	FI nom_var	nom de variable erroné	10	FI non_bool	paramètre fourni non booléen	9	FI non_num	paramètre fourni non numérique	poids faible	8	FI err_calc	erreur de calcul (division par 0 ...)	7	FI hors_lim	paramètre ou variable hors limite	6			5	FI err_ov	erreur surtension	4	FI err_uv	erreur sous tension	3	FI err_cc	erreur court circuit moteur	2	FI err_dth	disjonction thermique	1		
	bit	Flag	Description																																																																																																							
poids fort	32																																																																																																									
	31																																																																																																									
	30																																																																																																									
	29																																																																																																									
	28																																																																																																									
	27																																																																																																									
	26																																																																																																									
	25																																																																																																									
	24																																																																																																									
	23	FI err typ	erreur sur le type de paramètre																																																																																																							
	22	FI hors champ	point hors champ des séquences																																																																																																							
	21	FI out mem	dépassement taille mémoire																																																																																																							
	20	FI hors seq	dépassement taille séquence																																																																																																							
	19	FI def soft																																																																																																								
	18	FI_non_autorise	commande non autorisée selon l'état du module																																																																																																							
	17	FI_req_add	adresse module requise																																																																																																							
	16																																																																																																									
	15	FI refus	commande refusée selon l'état du module																																																																																																							
	14	FI err cohe	erreur de cohérence entre paramètres																																																																																																							
	13	FI incoher	syntaxe incohérente																																																																																																							
	12	FI non_def	paramètre non défini																																																																																																							
	11	FI nom_var	nom de variable erroné																																																																																																							
	10	FI non_bool	paramètre fourni non booléen																																																																																																							
	9	FI non_num	paramètre fourni non numérique																																																																																																							
poids faible	8	FI err_calc	erreur de calcul (division par 0 ...)																																																																																																							
	7	FI hors_lim	paramètre ou variable hors limite																																																																																																							
	6																																																																																																									
	5	FI err_ov	erreur surtension																																																																																																							
	4	FI err_uv	erreur sous tension																																																																																																							
	3	FI err_cc	erreur court circuit moteur																																																																																																							
	2	FI err_dth	disjonction thermique																																																																																																							
	1																																																																																																									
Codes Erreur:																																																																																																										

Remarque : Les flags d'erreur ne sont pas effacés par une simple lecture de la variable d'erreur, il convient de remplacer les bits à 0 par une écriture de la variable #ERROR, ou de reseter le module via la commande MODULE_RESET ou une simple remise sous tension.

Attention! Les bits non définis ont des valeurs indifférentes.

6. ANNEXE

6.1. Exemples de séquences

6.1.1. Exemple 1

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

```

HALT                                ;arrêt des séquences éventuelles
OPEN_SEQ                            ;mode édition
IF #SUPPLY_VOLTAGE < 15000 JUMP 5    ;sortie OFF si la U < 15Volts
IF #SUPPLY_VOLTAGE > 20000 JUMP 5    ;sortie OFF si la U > 20Volts
#OUTPUT := 8                        ;sinon sortie ON
JUMP 1                              ;reboucle sur la phase 1
#OUTPUT := 0                        ;sortie OFF
JUMP 1                              ;reboucle sur la phase 1
CLOSE_SEQ                           ;sortie du mode édition
START_SEQ 1                         ;exécution

```

6.1.2. Exemple 2

La vitesse de rotation du MICROMAC est asservie à l'entrée analogique à laquelle on applique un offset et un gain.

```

HALT
OPEN_SEQ                            ;mode édition
:20 #V1:=#INPUT_ANALOG - 15000      ;ligne 20: entrée analog - 15000 dans V1
#V1 := #V1 * 5                      ;(ligne 21)gain de 5
MOVE_SPEED #V1                      ;commande de mouvement en vitesse
JUMP_REL -3                         ;reboucle sur la ligne 20 (23-3) (en relatif)
CLOSE_SEQ                           ;fermeture du mode édition
START_SEQ 20                        ;exécution

```

6.1.3. Exemple 3

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.

```

HALT
OPEN_SEQ                            ;mode édition
#V1 := #INPUT & 2                   ;l'entrée 2 est isolée dans #V1
#V2 := #INPUT & 4                   ;l'entrée 4 est isolée dans #V2
IF #V1 = 2 JUMP_REL +3              ;teste l'entrée 2
IF #V2 = 4 JUMP_REL +4              ;teste l'entrée 3
JUMP 1                              ;reboucle sur la phase 1 si e2 et e3 sont inactives
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 phase 1
CLOSE_SEQ                           ;fermeture du mode édition
START_SEQ 1                         ;exécution

```

6.1.4. Exemple 4

Appel d'une sous-séquence pour mettre à zéro la position si l'entrée 3 est inactive ou si la tension d'alimentation est supérieure à 30V

```

HALT                                ;arrêt des séquences en cours (éventuelle)
#POSITION := 12345                  ;une valeur de position
OPEN_SEQ 15                         ;mode édition à la phase 1

:15 #V1 := #INPUT & 4
    IF #V1 = 0 JUMP 20              ;Ligne 16: teste l'entrée 3
:17 IF #SUPPLY_VOLTAGE > 30000 JUMP 25 ;teste la tension d'alimentation
:18 JUMP 0                          ;arrêt de la séquence
:20 CALL 40                         ;l'entrée est active : appel sous_séquence 40
    JUMP 17                        ;retour en ligne 17
:25 CALL 40                         ;la tension d'alimentation est > à 30V
    JUMP 18                        ;retour en ligne 18

; sous_séquence
:40 #POSITION:=0                   ;Ligne 40: met la position à zéro
    RETURN                        ;retourne au call
CLOSE_SEQ                          ;fermeture du mode édition
START_SEQ 15                       ;exécution
    
```

6.1.5. Exemple 4 bis

Même exemple écrit avec les mnémoniques réduits :

```

;Même exemple écrit avec les mnémoniques réduits :
HAL                                ;arrêt des séquences en cours (éventuelle)
#POS := 12345
OSE 15                            ;mode édition à la phase 1

:15 #V1 := #INP & 4
    IF #V1 = 0 JUM 20              ;Ligne 15: teste l'entrée 3
:17 IF #SVO > 30000 JUM 25         ;teste la tension d'alimentation
:18 JUM 0
:20 CAL 40
    JUM 17
:25 CAL 40
    JUM 18
:40 #POS:=0                       ;Ligne 40: met la position à zéro
    RET                           ;retourne au call

CSE                               ;fermeture du mode édition
SSE 15
    
```



6.1.6. Exemple 5

Relectures de variables :

```
00read #position  
;réponse 00#POS= 12789 -> le module est à la position 12789  
  
00read h#input  
;réponse 00#INP=h00000012 -> les entrées logiques 5 et 2 sont actives  
  
00read b#INP  
;réponse 00#INP=b000000000000000000000000000010010 -> les entrées logiques 5 et 2  
sont actives  
  
00read #supply_voltage  
;réponse 00#SVO= 24560 -> la tension d'alimentation vaut 24,56 V  
  
#V1 := 156  
#M2 := -45  
#V3 := #V1 * #M2  
00read #V3  
;réponse 00#V3 = -7020 -> résultat du calcul précédant
```

6.2. Résumé des commandes et variables

Add	Commandes et variables	abrévés	description
	Général		
[@]	MODULE_RESET [ALL]	MRE	Réinitialisation du module. Efface toute la mémoire si clé: ALL
[@]	POWER [ON][OFF][SC]	POW	Puissance moteur on ou off
@	READ #xxx	REA	Lecture d'une variable (utilisateur ou système)
@	READ_VERSION	RV	Relecture version module et soft
	configuration		
[@]	SET_ADDRESS add	SAD	Définition de l'adresse module
[@]	SET_BAUDRATE v	SBA	Vitesse du dialogue série
	paramètres		
[@]	#HIGH_SPEED := Smax	#HSP	Définie la vitesse max (1 to 120000 1/100 t/mn) ⇔ 0,03 to 4000 p/s
[@]	#LOW_SPEED := Smin	#LSP	Définie la vitesse de démarrage (1 to 120000 1/100 t/mn)
[@]	#RAMPING_TIME := Tr	#RTI	Durée de l'accélération (1 to 65000ms)
[@]	#TORQUE_RATIO := Im	#TRA	Courant nominal du moteur = Inominal * Im/ 100
[@]	#LOW_TORQUE := Is	#LTO	Courant moteur au repos = Inominal * Im/ 100
[@]	#POSITIVE_END := b+	#PEN	position de la butée soft positive
[@]	#NEGATIVE_END := b-	#NEN	position de la butée soft négative
@	#STATUS	#STA	État du module
@	#ERROR	#ERR	Compte rendu d'erreur
	modes		
[@]	STANDBY_MODE [ON][OFF]	SMO	Sélection du mode de gestion du courant
[@]	HARD_ENDS [ON][OFF] [POS][NEG][HALT][STOP]	HEN	Autorisation et mode de gestion des butées hardware (un ou deux sens)
[@]	SOFT_ENDS [ON][OFF][HALT][STOP]	SEN	Autorisation et mode de gestion des butées soft
[@]	INVERSE_POLARITY [ALL][OFF][IN][OUT]	IPO	Gestion de la polarité des entrées et sorties
	mouvements		
[@]	MOVE_TO P0	MTO	Mouvement jusqu'à la position absolue P0
[@]	MOVE_ON d0	MON	Mouvement relatif sur la distance d0
[@]	MOVE_SPEED V0	MSP	Mouvement à vitesse constante
[@]	HALT	HAL	Arrêt immédiat du mouvement
[@]	STOP	STO	Arrêt du mouvement avec décélération
[@]	#POSITION	#POS	Position absolue du moteur
@	#PROFILE_SPEED	#PSP	Valeur de la vitesse instantanée (calculée)
	entrées sorties		
[@]	#OUTPUT := xx	#OUT	Positionne les sorties logique à xx
@	#INPUT	#INP	Valeur des entrées logiques
@	#INPUT_ANALOG	#IAN	Valeur de l'entrée analogique (mV)
@	#SUPPLY_VOLTAGE	#SVO	Valeur de la tension d'alimentation (mV)
@	#TEMPERATURE	#TEM	Valeur de la température interne du boîtier
	variables		
[@]	#Vn := v	#Vn	Charge la valeur v dans la variable n (décimal, hexa , binaire)
	#Mn := v	#Mn	Variable temporaire #Vn ou sauvegardée #Mn (1 ≤ n ≤ 4)
...	#n1 := #n2 [op #n3] ou #n1 := #n2 op cste ou #n1 := cste	+ - * / & > ≥ ≤ <	Fonctions logique et mathématique, opérateurs
	Séquences		
[@]	START_SEQ nl	SSE	Démarre la séquence à partir de la ligne nl
[@]	#ON_RESET	ORE	N° de la ligne de démarrage automatique de séquence (0 = off)
[@]	#LINE	#LIN	N° de la ligne de séquence en cours d'exécution
@	STEP nl	STE	Exécution de la ligne de séquence nl
[@]	OPEN_SEQ nl	OSE	Ouverture de séquence en mode édition (à partir de la ligne nl)
[@]	CLOSE_SEQ	CSE	Fermeture du mode édition
[@]	:np	:	Numéro de ligne de séquence
...	WAIT t	WAI	Tempo de t ms
...	IF #n1 ?? #n2 JUMP nl	IF	Saut conditionnel sur la valeur d'une variable
...	IF #n1 ?? #n2 JRE n	IF	Saut conditionnel relatif sur la valeur d'une variable
...	JUMP nl	JUM	Saut à la ligne nl
...	JUMP_RELATIVE n	JRE	Saut relatif
...	CALL nl	CAL	Appel de sous séquence
...	RETURN	RET	Retour de sous séquence (éventuellement fin de séquence)
...	NOP	NOP	Ligne sans action

6.3. Documents associés

Ces documents sont accessibles sur le site midi-ingenierie.com (connexion Internet indispensable).

1. Note d'application – Liaison calculateur : protocoles et syntaxe
2. Résumé des commandes DMAC et microMAC (français)
Commands Abstract – DMAC and microMAC (anglais)