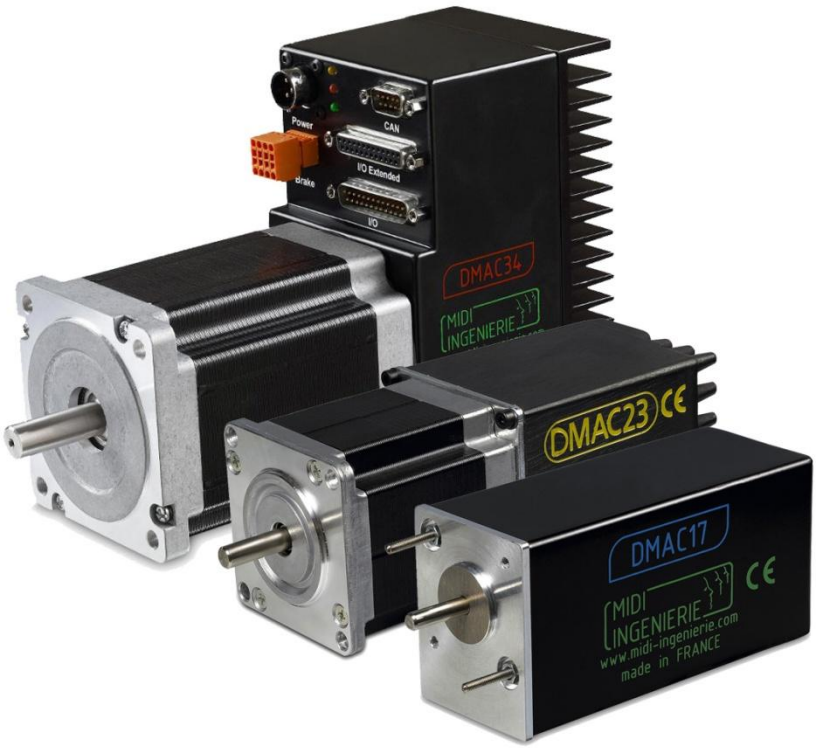
midi ingénierie
NEXEYA Products Division

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

midi ingenierie

DMAC17, DMAC23, DMAC34
Manuel utilisateur



Date : 23.11.12

Référence : dmac_v2_um_fr.pdf
Réf. MI : CMN1480889.docx

Révision : 2

Auteur : C.MARTY
http://www.midi-ingenierie.com

afaq ISO 9001 Qualité AFNOR CERTIFICATION

S.A.S. au capital social de 713 046€
333 373 108 RCS Toulouse
Siret 333 373 108 00013
APE 7112B
TVA FR60 333 373 108

GROUPE NEXEYA

Sommaire

1. Présentation du produit

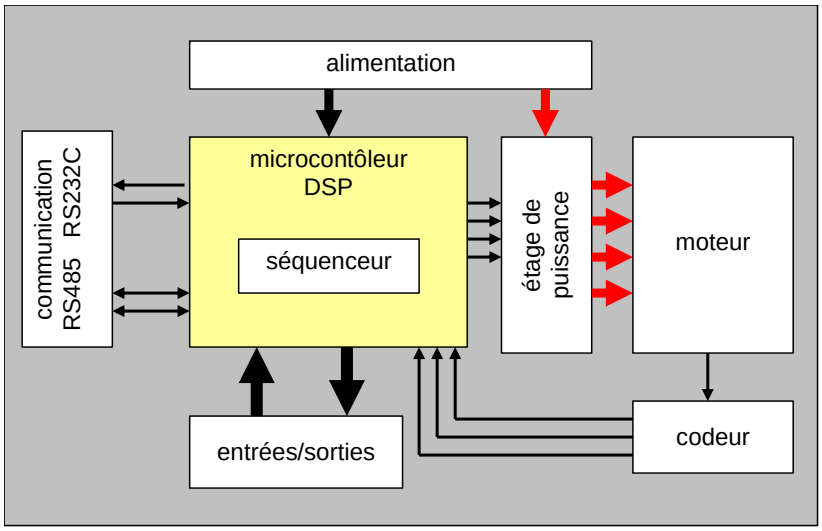
1.1. Introduction

Le DMAC est un module composé d'un moteur et d'une électronique de commande intégrée fonctionnant en mode autocommuté. 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 : contrôle de la position, de la vitesse mais aussi du couple.

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 RS232 (applications monoaxes) ou RS485 (applications multiaxes, meilleure immunité au bruit). Une version CanOpen motion control DSP402 est également disponible.

1.2. Fonctionnalités

1.2.1. Architecture



1.2.2. Mouvements

Les mouvements se font avec une résolution de 1/10000ème de tour. Un incrément de position correspond donc à un angle 0,036°.
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) et les positions en incréments (10^{-4}tr).

On distingue trois types de mouvements:

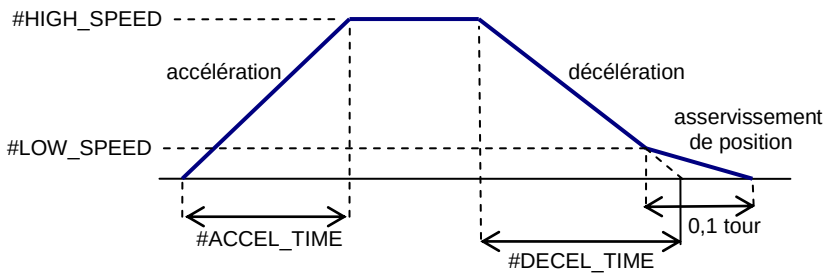
- Le mode Vitesse (`MOVE_SPEED`) dans lequel le moteur tourne à une vitesse de consigne.
- Le mode Position (`MOVE_TO` et `MOVE_ON`) dans lequel le moteur tourne jusqu'à une position de consigne définie en absolu par rapport à une origine ou en relatif par rapport à la position courante.
- Le mode Interpolation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

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

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

Un mouvement complet se décompose donc en quatre étapes successives:
Accélération de la vitesse courante (qui peut être nulle) jusqu'à la vitesse de consigne.
Mouvement à vitesse constante.
Décélération de la vitesse de consigne jusqu'à la vitesse `#LOW_SPEED`.
Asservissement de position. Le moteur passe de la vitesse `#LOW_SPEED` jusqu'à atteindre une vitesse nulle sur la position d'arrivée.

Exemple de mouvement:



1.2.3. Butées

Le DMAC dispose de quatre butées:
Deux butées "matérielles" (aussi appelées butées Hard). Ce sont des butées électriques qui utilisent les entrées logiques IN1 et IN2 sur DMAC17 ou DMAC23, et IN9 et IN10 sur DMAC34. On peut y connecter un capteur, un contact sec, etc.
Deux butées "virtuelles" (aussi appelées butées Soft). Ce sont des butées gérées par le logiciel qui empêchent de dépasser une position donnée. Cette position est paramétrable grâce aux variables `#POSITIVE_END` et `#NEGATIVE_END`.
Au passage d'une de ces butées, si elles sont activées, le mouvement en cours est immédiatement arrêté et tout mouvement dans le même sens est interdit. Seuls les mouvements dans le sens inverse, qui permettent de se dégager de la butée sont autorisés.
La relecture de la variable `#STATUS` permet de savoir si le module est arrêté sur une butée.

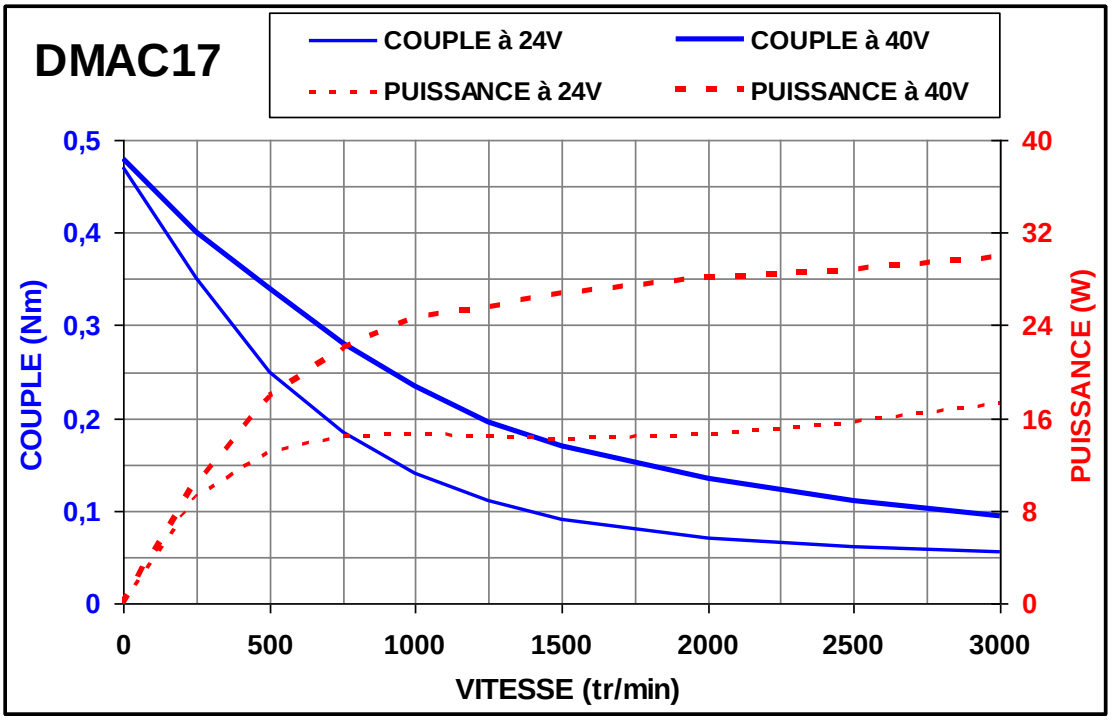
1.2.4. Séquenceur

Le DMAC peut mémoriser puis exécuter jusqu'à 500 commandes, ce qui permet de réaliser des automatismes qui pourront être exécutés en autonome (sans automate ni PC de commande).
L'écriture des séquences est simple et intuitive. Les commandes mémorisées sont exécutées successivement au rythme d'une toutes les millisecondes.
Des commandes évoluées spécifiques (`IF`, `JUMP`, `CALL`, `WAIT`, etc.) permettent de contrôler le déroulement des séquences pour conditionner l'exécution à des évènements extérieurs (entrées logiques et analogique, position, variables internes, etc.…)

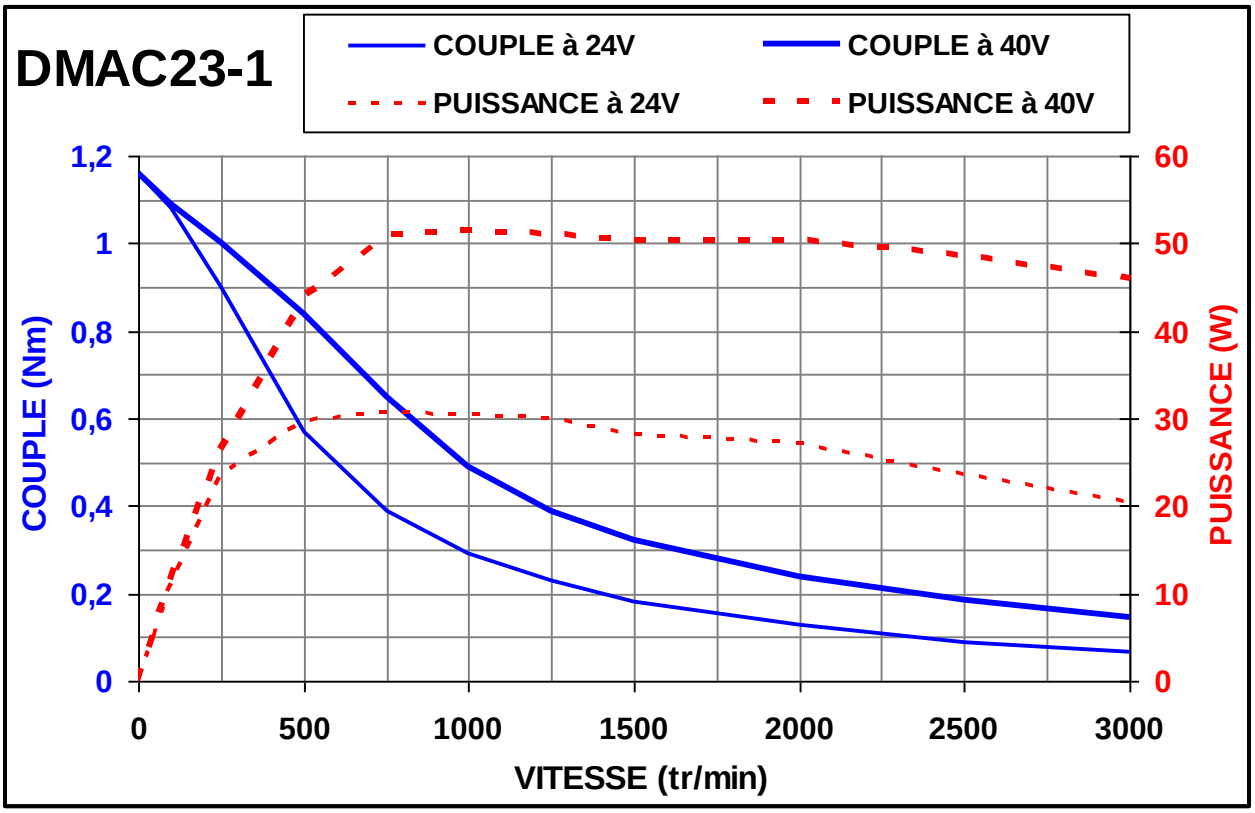
1.2.5. Couple et puissance

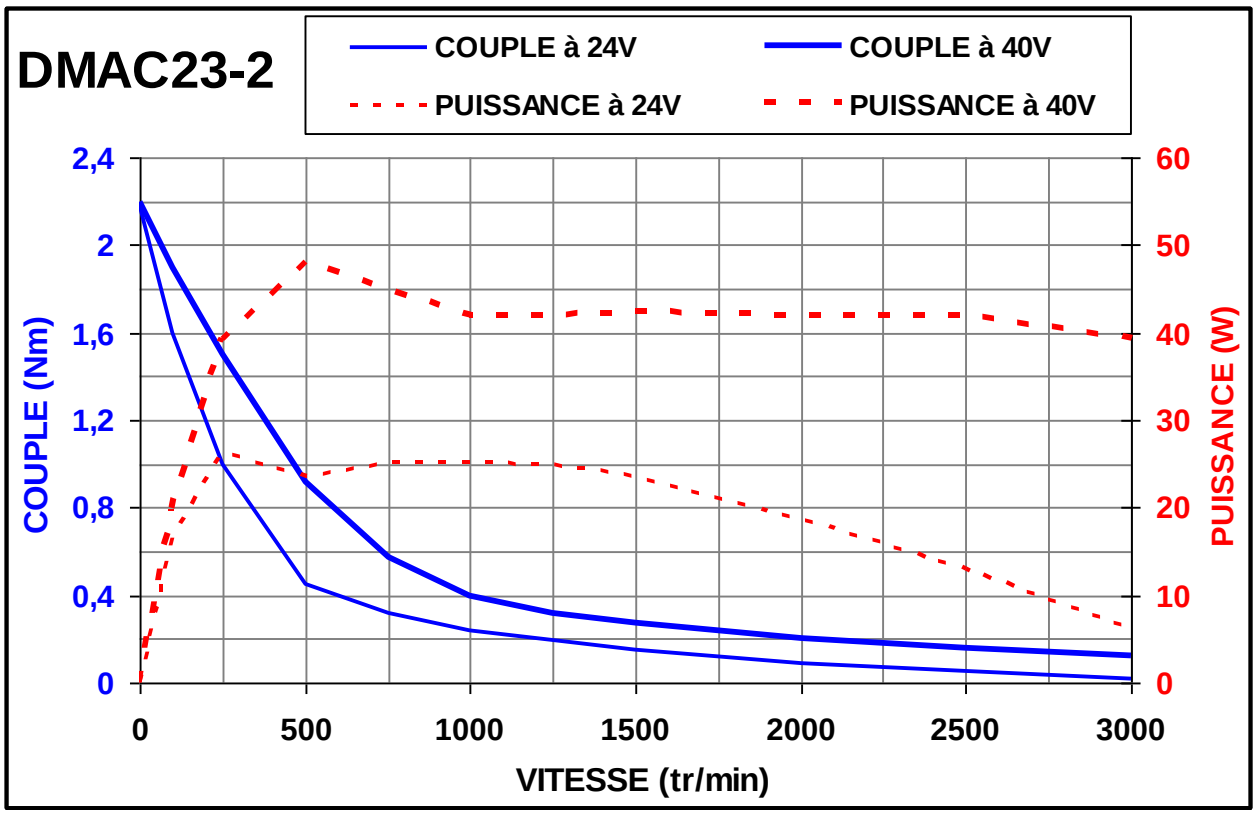
Le couple maximum disponible en sortie du moteur est réglable par la variable `#TORQUE_RATIO` jusqu'à 0,5Nm (DMAC17), 1,2Nm (DMAC23-1), 2,2Nm (DMAC23-2), 7Nm (DMAC34-1) ou 10Nm (MAC34-2).
Un mode optimisé (`OPTIMIZED_CURRENT`) permet d'injecter dans le moteur le courant réellement nécessaire afin de minimiser les pertes thermiques.
La puissance moteur peut être coupée ou rétablie grâce à une commande (`POWER ON` ou `POWER OFF`). Elle est automatiquement rétablie au début de chaque mouvement.
Par construction, le couple du moteur est maximum à basse vitesse, et cette motorisation peut exercer du couple à l'arrêt (positionnement).

Courbes DMAC17

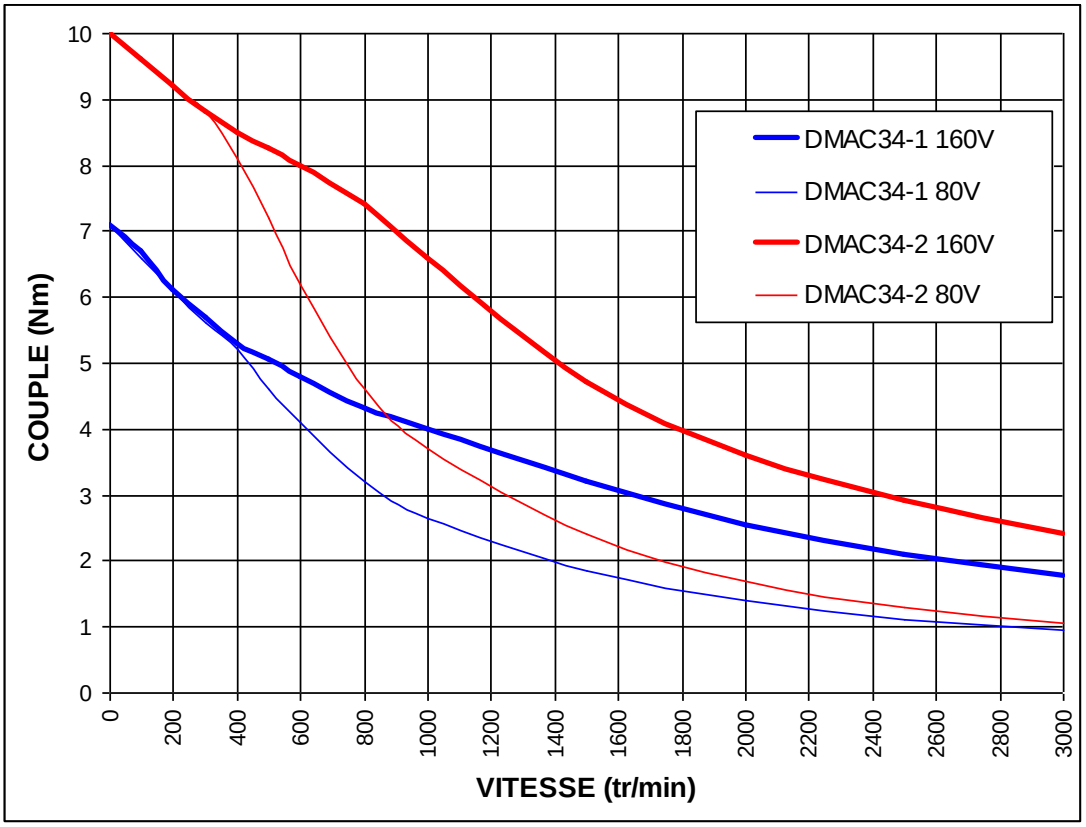


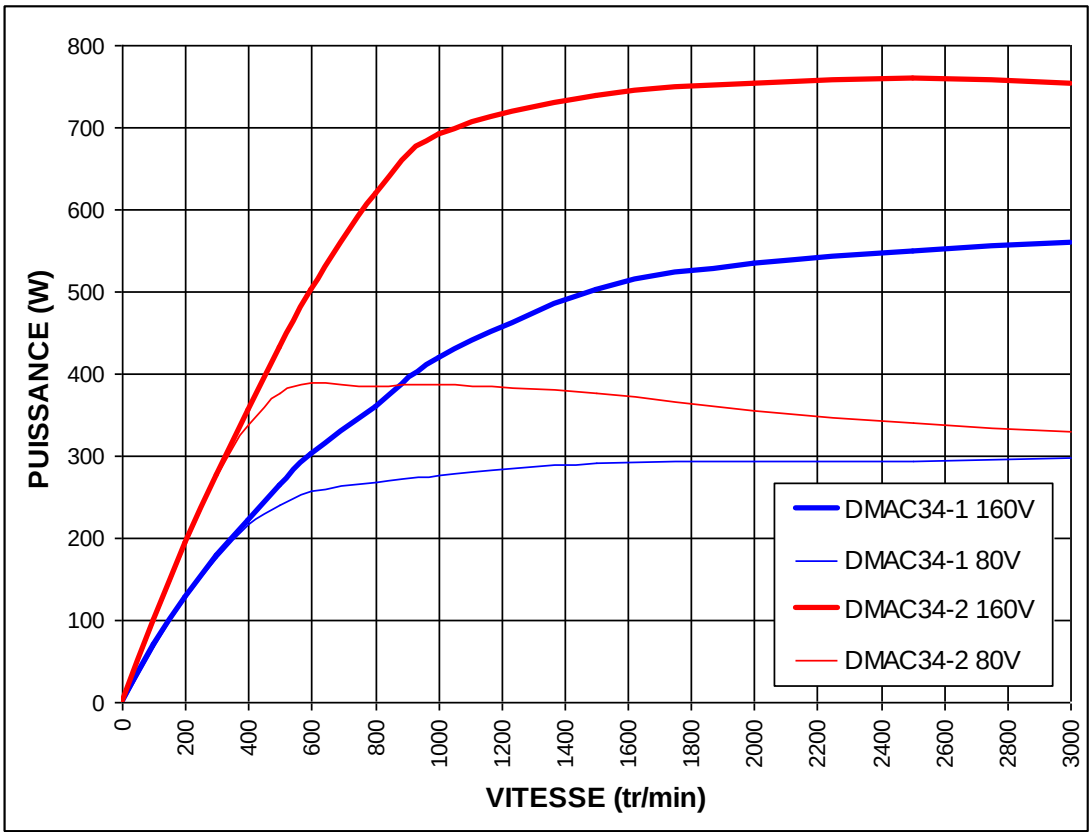
Courbes DMAC23





Courbes DMAC34





1.2.6. Entrées – Sorties

Les DMAC17 et 23 disposent de 6 entrées et 4 sorties opto-isolées. Le DMAC34 dispose de 8 entrées et 8 sorties opto-isolées. Elles peuvent être utilisées conjointement avec le séquenceur pour, par exemple, déclencher des mouvements prédéfinis. Deux entrées butées (non isolées sur DMAC34) peuvent être utilisées comme des entrées logiques banalisées lorsque les fonctions de butées ne sont pas utiles.

Une (DMAC17, DMAC23) ou deux (DMAC34) entrées analogiques différentielles permettent de connecter des capteurs ou des potentiomètres. Comme pour les entrées logiques, les valeurs des entrées analogiques peuvent être utilisées dans le module par l'intermédiaire du séquenceur pour, par exemple, asservir la position de l'axe à la position d'un potentiomètre.

Sur DMAC34, deux sorties analogiques sont également disponibles pour contrôler des actuateurs proportionnels, ou fournir une information de type recopie.

1.3. Précautions d'emploi

1.3.1. 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.
- 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 du DMAC34 à un effort axial ≥ 50N ou bien à un effort radial ≥ 250N à 2 mm du flasque.
- Ne pas soumettre l'arbre moteur du DMAC17 ou DMAC23 à un effort axial ≥ 10N ou bien à un effort radial ≥ 20N à 5 mm du flasque.

1.3.2. Conditions de stockage

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

1.3.3. 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 limitée en courant ou bien insérer un fusible 5A temporisé (DMAC17 ou DMAC23) **ou** 10A temporisé **(DMAC34)** 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°C.
- Le câble du DMAC ne doit pas être soumis à des flexions répétitives. Installer le câble de manière fixe par rapport au DMAC.
- 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 DMAC à la masse générale de la machine.
- Ne jamais introduire un corps étranger dans les orifices du produit.

2. Spécifications techniques

2.1. Alimentation

	Tension d'alimentation	Courant consommé maximum
DMAC17	12 à 45Vdc	2A
DMAC23	12 à 45Vdc	3A
DMAC34	20 à 160Vdc	9A

Attention ! Plus la tension est faible, plus le courant consommé augmente.
Le courant réellement nécessaire dépend de la puissance mécanique totale requise :
$P = C * \omega$ (P est la puissance en Watts, C est le couple en Nm, ω est la vitesse de rotation du moteur en rad/s).

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

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

Lorsque la tension descend en dessous de 12V (DMAC17, DMAC23) ou 20V (DMAC34), les mouvements sont arrêtés et le module se place en disjonction de sous-tension.

Remarque DMAC17 et DMAC23 *: Lors des phases de freinage, l'énergie cinétique récupérée est renvoyée vers l'alimentation, celle-ci doit donc accepter un éventuel courant inverse. Dans ce cas de figure, la tension d'alimentation peut se mettre à monter (charge des condensateurs de sortie de l'alimentation). Les modules DMAC17 et 23 possèdent une sécurité interne qui coupe le freinage lorsque la tension d'alimentation dépasse 46V. Une protection complémentaire coupe définitivement la puissance moteur si la tension dépasse 49V. Le module se met alors en défaut jusqu'à coupure d'alimentation, reset par la commande "MODULE_RESET" ou écriture #ERROR:=0.*

Quelle que soit la valeur nominale de l'alimentation utilisée, la tension générée par le DMAC17 ou 23 peut atteindre 49 Vdc.

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

Dans tous les cas, si l'alimentation ne supporte pas la tension nominale du module en récupération il convient d'insérer une diode (75V/5A) en série dans le circuit d'alimentation du module. Cette diode est intégrée dans le bornier DMAC.

Remarque DMAC34 *: Lors des phases de freinage, l'énergie cinétique récupérée est renvoyée vers l'alimentation, celle-ci doit donc accepter un éventuel courant inverse. Dans ce cas de figure, la tension d'alimentation peut se mettre à monter (charge des condensateurs de sortie de l'alimentation). Les modules DMAC34 possèdent une sécurité interne qui coupe le freinage lorsque la tension d'alimentation dépasse 165V. Une protection complémentaire coupe définitivement la puissance moteur si la tension dépasse 170V. Le module se met alors en défaut (led rouge allumée) jusqu'à coupure d'alimentation, reset par la commande "MODULE_RESET" ou écriture #ERROR:=0.*

Quelle que soit la valeur nominale de l'alimentation utilisée, la tension générée par le DMAC34 peut atteindre 170 Vdc.

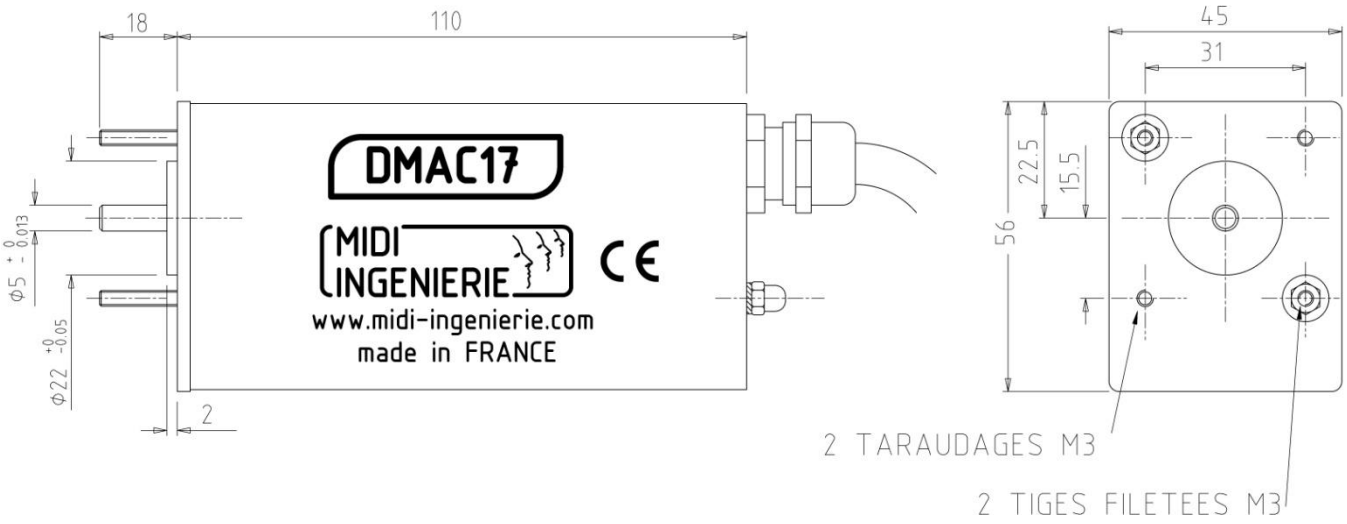
L'option Ballast permet de dissiper l'énergie récupérée dans des résistances de puissance de façon à ne jamais atteindre le seuil de protection.

Dans tous les cas, si l'alimentation ne supporte pas la tension nominale du module en récupération il convient d'insérer une diode (200V/15A) en série dans le circuit d'alimentation du module.

2.2. Spécifications mécaniques

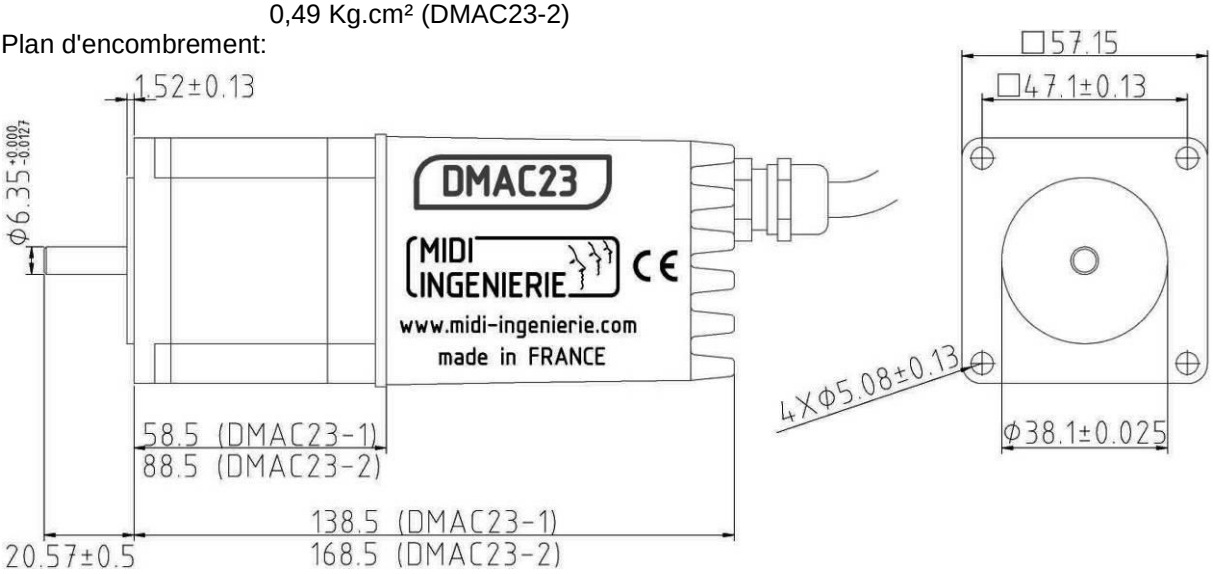
2.2.1. DMAC17

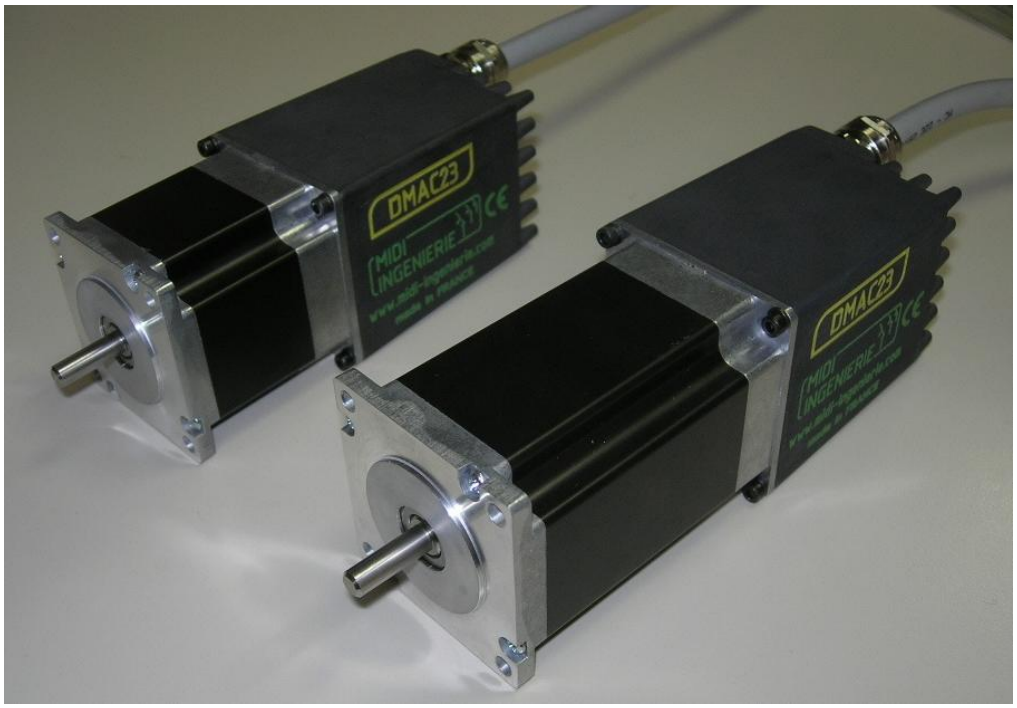
Taille:	45mm x 56mm x 110mm (hors câble et arbre moteur)
Poids:	800g
Inertie rotor :	0,08 Kg.cm²
Plan d'encombrement:	



2.2.2. DMAC23

Taille: 57.15 x 57.15 x 138.5mm (DMAC23-1 hors câble et arbre moteur)
57.15 x 57.15 x 168.5mm (DMAC23-2 hors câble et arbre moteur)
Poids: 1.5kg
Inertie rotor : 0,25 Kg.cm² (DMAC23-1)
0,49 Kg.cm² (DMAC23-2)
Plan d'encombrement:





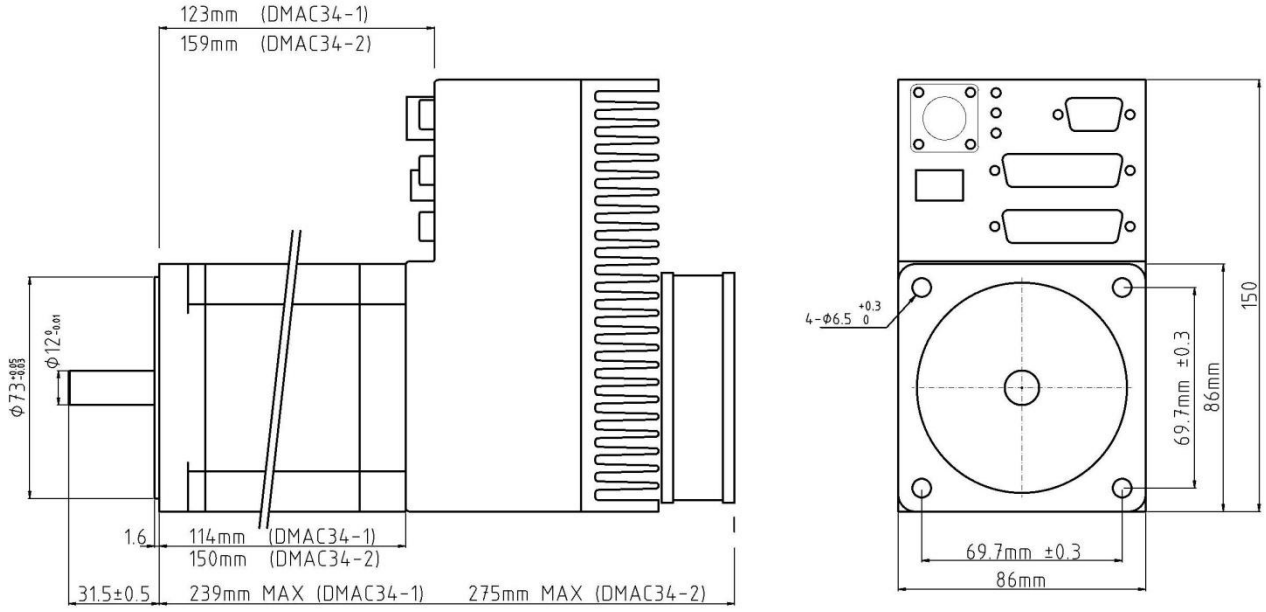
2.2.3. DMAC34

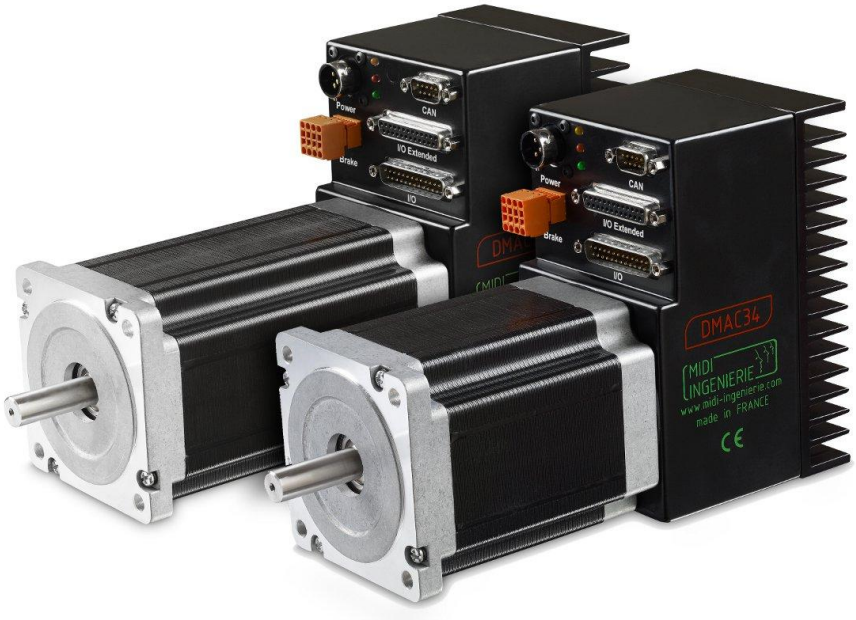
Taille: 239 x 86 x 150mm (DMAC34-1 hors câble et arbre moteur)
275 x 86 x 150mm (DMAC34-2 hors câble et arbre moteur)

Poids: 4,8 Kg (DMAC34-1)
6,1 Kg (DMAC34-2)

Inertie rotor : 2,7 Kg.cm² (DMAC34-1)
4,05 Kg.cm² (DMAC34-2)

Plan d'encombrement:

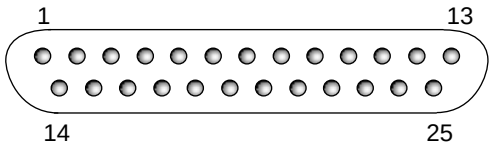




3. Câblage et configuration

3.1. Description du connecteur du DMAC17

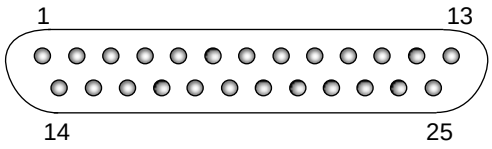
Broche	Description
1	Réservé
2	Communication RS485. /Z
3	Communication RS485. Z
4	Réservé
5	Communication RS485. 0V
6	Entrée logique numéro 1 (IN1)
7	Entrée logique numéro 2 (IN2)
8	Entrée logique numéro 3 (IN3)
9	Entrée logique numéro 4 (IN4)
10	Entrée logique numéro 5 (IN5)
11	Entrée logique numéro 6 (IN6)
12	Réservé
13	Réservé
14	Entrée analogique (-)
15	Entrée analogique (+)
16	Masse des entrées logiques (0V IO)
17	Point commun +V des sorties (+V IO)
18	Sortie logique numéro 1 (OUT1)
19	Sortie logique numéro 2 (OUT2)
20	Sortie logique numéro 3 (OUT3)
21	Sortie logique numéro 4 (OUT4)
22	Réservé
23	Réservé
24	0V alimentation
25	+V alimentation



Connecteur SUB-D mâle du MAC17
(vue de face)

3.2. Description du connecteur du DMAC23

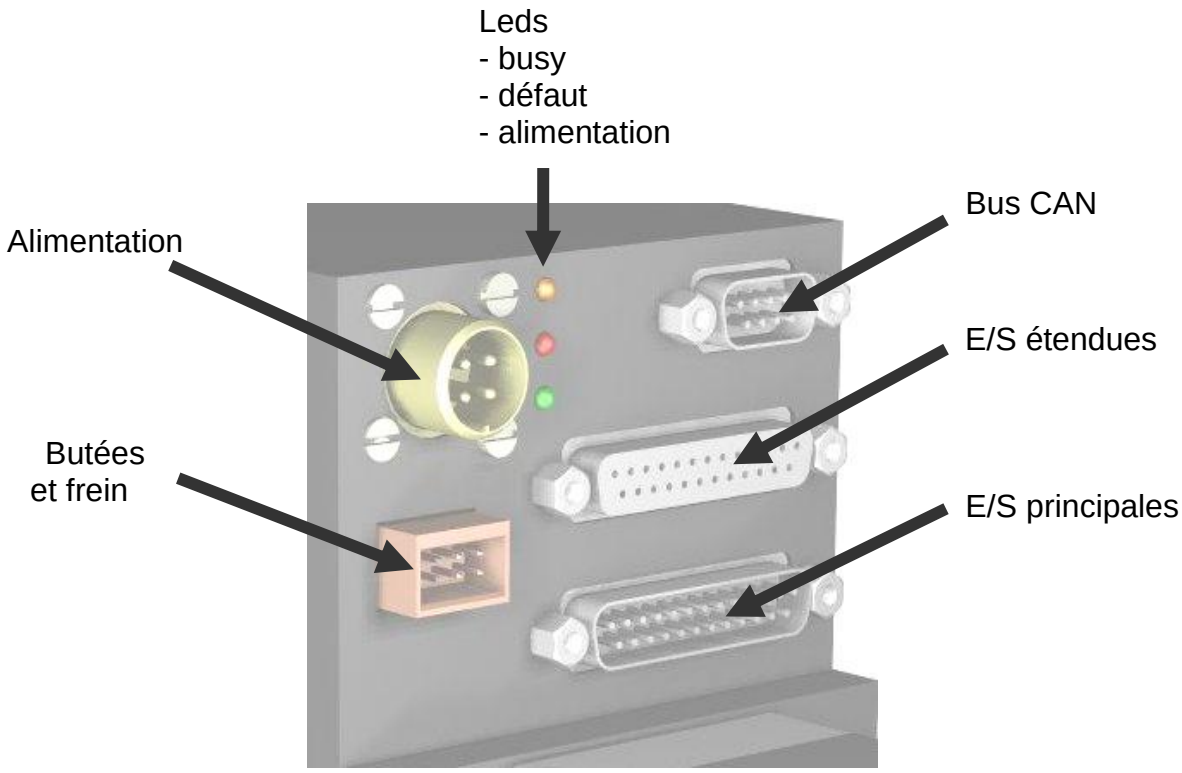
Broche	Description
1	Communication RS232. TXD
2	Communication RS485. /Z
3	Communication RS485. Z
4	Communication RS232. RXD
5	Communication 0V
6	Entrée logique numéro 1 (IN1)
7	Entrée logique numéro 2 (IN2)
8	Entrée logique numéro 3 (IN3)
9	Entrée logique numéro 4 (IN4)
10	Entrée logique numéro 5 (IN5)
11	Entrée logique numéro 6 (IN6)
12	Réservé
13	Réservé
14	Entrée analogique (-)
15	Entrée analogique (+)
16	Masse des entrées logiques (0V IO)
17	Point commun +V des sorties (+V IO)
18	Sortie logique numéro 1 (OUT1)
19	Sortie logique numéro 2 (OUT2)
20	Sortie logique numéro 3 (OUT3)
21	Sortie logique numéro 4 (OUT4)
22	Réservé
23	Réservé
24	0V alimentation
25	+V alimentation



Connecteur SUB-D mâle du MAC17
(vue de face)

3.3. Description des connecteurs du DMAC34

3.3.1. Vue générale de la connectique DMAC34



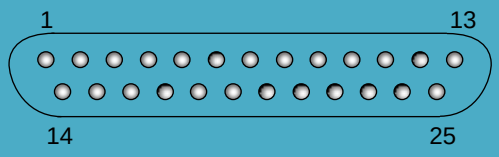
3.3.2. Connecteur d'alimentation

Prise circulaire 4 pts : alimentation de puissance			
A	*+Valim*	**C**	*Masse mécanique*
B	*0Valim*	**D**	*Réservé*

Références du connecteur à utiliser (1 exemplaire fourni):
- fiche 4 points pour contacts femelles
  réf. TR1004PFS1NB (ITT CANNON)
  réf 1186713 (Farnell)
- contacts femelles à sertir jauge 16
  réf. 192991-0073 (ITT CANNON)
  réf 1186778 (Farnell)
- serre câble pour fiche 4 points réf. TR10ASR1N (ITT CANNON) ou 1188282 (Farnell)

3.3.3. Connecteur d'entrées/sorties principales

Subd25 mâle : E/S vers bornier DMAC



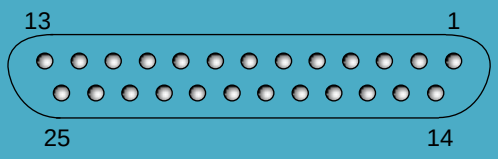
SUB-D25 mâle du DMAC34 (vue de face)

1	*TXD RS232*	**14**	*–IANA1*
2	*/Z RS485*	**15**	*+IANA1*
3	*Z RS485*	**16**	*0V IO*
4	*RXD RS232*	**17**	*+V IO*
5	*0V RS232_485*	**18**	*OUT1*
6	*IN1*	**19**	*OUT2*
7	*IN2*	**20**	*OUT3*
8	*IN3*	**21**	*OUT4*
9	*IN4*	**22**	*Réservé*
10	*IN5*	**23**	*Réservé*
11	*IN6*	**24**	*0Vaux*
12	*Réservé*	**25**	*+24Vaux*
13	*Réservé*		

Cette prise est compatible point à point avec le bornier DMAC. La tension +24Vaux fournie par le DMAC34 entre les broches 24 et 25 permet d'alimenter le bornier sous réserve que la puissance consommée soit inférieure à 2W. Dans le cas contraire, ne pas connecter ces points et alimenter le bornier séparément.

3.3.4. Connecteur d'entrées/sorties étendues

SubD25 femelle : E/S, fonctions étendues

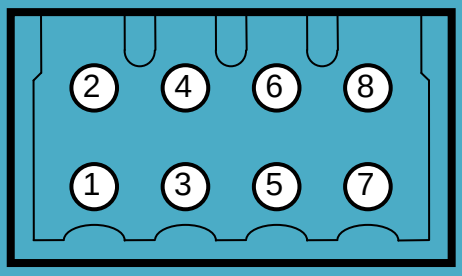


SUB-D25 mâle du DMAC34 (vue de face)

1	*OUTA1*	**14**	*–IANA2*
2	*OUTA2*	**15**	*+IANA2*
3	*0Vana*	**16**	*0V IO*
4	*0Vana*	**17**	*+V IO*
5	*Réservé*	**18**	*OUT5*
6	*IN7*	**19**	*OUT6*
7	*IN8*	**20**	*OUT7*
8	*Réservé*	**21**	*OUT8*
9	*Réservé*	**22**	*COD A*
10	*COD /A*	**23**	*COD B*
11	*COD /B*	**24**	*0Vaux*
12	*0V_COD*	**25**	*+24Vaux*
13	*Réservé*		

3.3.5. Connecteur butées et frein

Bornier débrochable 8 points : butées et frein



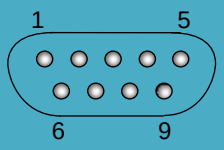
Connecteur 8 points du DMAC34 (vue de face)

1	*24Vaux*	***2***	*BUT+ (IN9)*
3	*0Vaux*	***4***	*24Vaux*
5	*BUT- (IN10)*	***6***	*0Vaux*
7	*+FREIN*	***8***	*-FREIN*

Références du connecteur à utiliser (1 exemplaire fourni):
Bornier débrochable 2 x 4 points 3.5mm femelle droit
réf. B2L3.5/8 (Weidmuller)
réf 382-9602 (Radiospares)

3.3.6. Connecteur bus CAN

SubD9 Mâle : bus CAN



SUB-D25 mâle du DMAC34 (vue de face)

1	*Réservé*	**4**	*Réservé*	**7**	*CANH*
2	*CANL*	**5**	*Masse mécanique*	**8**	*Réservé*
3	*0V CAN*	**6**	*Réservé*	**9**	*Réservé*

3.4. Fonctionnalités des entrées/sorties

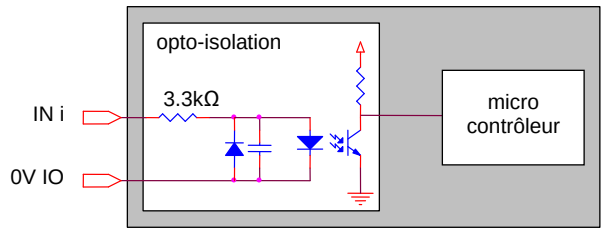
Les entrées sorties commandées par l'utilisateur sont multiplexées avec des fonctions évoluées.

	ENTREE	FONCTION
	IN1	Entrée utilisateur ou Butée sens positif sur DMAC17 et DMAC23
	IN2	Entrée utilisateur ou Butée sens négatif sur DMAC17 et DMAC23
	IN3	Entrée utilisateur
	IN4	Entrée utilisateur
	IN5	Entrée utilisateur et capture de position ou référence
	IN6	Entrée utilisateur ou Synchro interpolation
DMAC34 seulement	IN7	Entrée utilisateur
DMAC34 seulement	IN8	Entrée utilisateur
DMAC34 seulement	IN9	Entrée utilisateur ou Butée sens positif
DMAC34 seulement	IN10	Entrée utilisateur ou Butée sens négatif

	SORTIE	FONCTION
	OUT1	BUSY ou Synchro interpolation ou Sortie utilisateur (suivant `#OUTPUT_CONFIG.1`)
	OUT2	DEFAUT ou Sortie utilisateur (suivant `#OUTPUT_CONFIG.2`)
	OUT3	Sortie utilisateur
	OUT4	Sortie utilisateur
DMAC34 seulement	OUT5	Sortie utilisateur
DMAC34 seulement	OUT6	Sortie utilisateur
DMAC34 seulement	OUT7	Sortie utilisateur
DMAC34 seulement	OUT8	Sortie utilisateur

3.5. Spécification des entrées-sorties

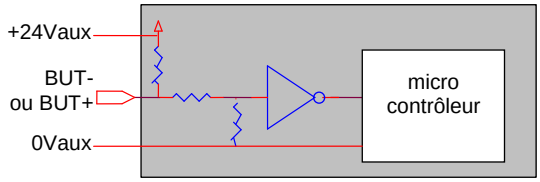
3.5.1. Entrées logiques opto-isolées



	min	max
VI_L (inactif) VI_H (actif)	-30V 4V	1V +30V
II_L II_H	 1,1mA	25µA

Tension nominale 5VDC à 24VDC
Tension du signal (inactif) 0VDC à 1VDC
Tension du signal (actif) 4VDC à 30VDC
Tension admissible ±30VDC
Isolation galvanique 50V
La référence 0VIO est commune à toutes les entrées logiques.
Les entrées logiques peuvent être relues avec la commande `READ #INPUT`.

3.5.2. Entrées logiques non isolées (butées du DMAC34)



	Min	max
VI_L (actif) VI_H (inactif)	-0,3V +11V	+3V 30V

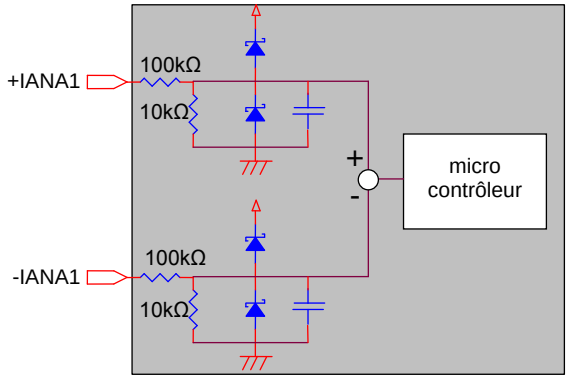
Tension nominale 24VDC (Pull up interne à 24V : source de courant 2mA)
Tension admissible -0,3 à +30 VDC
L'état de ces entrées est également donné par la variable #INPUT et peut être relu avec la commande READ #INPUT.

Les entrées logiques non isolées sont rappelées à 24V et sont inactives lorsqu'elles sont laissées en l'air, elles sont donc actives à l'état électrique 0V.

Plusieurs montages sont possibles :
- contact sec ou sortie optocoupleur directement connecté entre le signal BUT et la broche 0Vaux
- capteur 3 fils « NPN » alimenté par les broches +24Vaux/0Vaux et sortie sur le signal BUT
- ces entrées sont également compatibles avec des capteurs de proximité bifilaires qui s'alimentent sur le signal de commande via une source de courant. Les capteurs utilisés doivent respecter les critères suivants : courant inactif <0.5mA, tension active < 3V

3.5.3. Entrées analogiques

Schéma de principe pour une entrée différentielle :

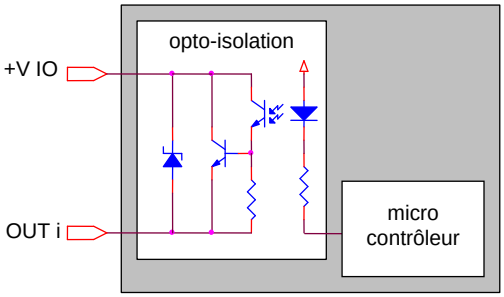


Sur DMAC17 ou 23, la variable `#INPUT_ANALOG` indique la tension différentielle entre les points +IANA et –IANA, exprimée en mV.

Sur DMAC34, la variable `#INPUT_A1` indique la tension différentielle entre les points +IANA1 et –IANA1, exprimée en mV. De même, `#INPUT_A2` donne la tension différentielle entre les entrées +IANA2 et –IANA2, exprimée en mV.

	DMAC17 ou 23	DMAC34
Plage de tension d'entrée / 0Valim	0 à 35V	±15V
Plage de tension d'entrée différentielle	±35V	±15V
Fréquence de coupure	1KHz	1KHz
Impédance d'entrée différentielle	220KOhms	220KOhms
Précision	±3%	±3%
Résolution	8,6mV	7,3mV

3.5.4. Sorties logiques opto-isolées



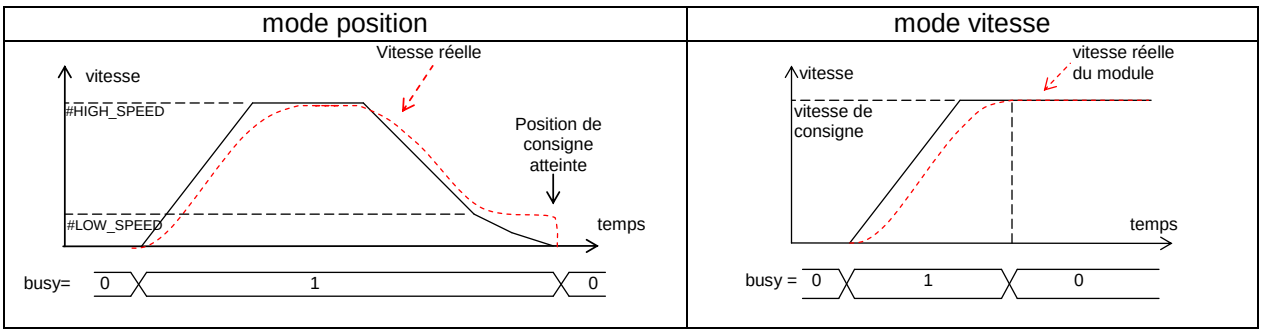
	min	max
Io_{off}		0,1mA
Vo_{on}		0,6V @1mA 1,1V @5mA 1,3V @50mA

Courant de sortie max 50mA
Tension de sortie max 30V
La tension +VIO est commune à toutes les sorties logiques.
Un niveau logique 1 (`#OUTPUT:=1`) fait conduire l'optocoupleur.
Un niveau logique 0 (`#OUTPUT:=0`) ouvre l'optocoupleur.

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

La sortie OUT1 peut matérialiser l'information BUSY si le bit correspondant de `#OUTPUT_CONFIG` est à 1.

Le signal Busy est actif lorsque le moteur ne parvient pas à suivre sa consigne de vitesse en mode vitesse ou tant que la position de consigne n'est pas atteinte en mode position, comme l'indique le diagramme ci-après.



La sortie OUT2 peut indiquer le signal DEFAUT si le bit correspondant de `#OUTPUT_CONFIG` est à 1.
Le signal DEFAUT est actif lorsque l'un des évènements suivant s'est produit :
Disjonction surcourant moteur
Disjonction surtension si la tension d'alimentation monte au-delà de la tension nominale maximum.
Disjonction sous-tension si la tension d'alimentation chute au dessous de la tension nominale minimum.
Disjonction thermique si la température interne moteur ou électronique atteint 85°C ou bien si la température du processeur atteint 120°C.

3.5.5. Sorties analogiques (DMAC34 seulement)

Les 2 sorties analogiques OUTA1 et OUTA2 sont référencées par rapport au 0Vana (non isolé).
Elles se pilotent directement avec la valeur souhaitée exprimée en mV :
`00#OUTPUT_A1 := 4520` ; force la sortie analogique n°1 à 4,52V
`00#OUTPUT_A2 := 2125` ; force la sortie analogique n°2 à 2,125V

Plage de tension de sortie :	0 à +10V
Fréquence de coupure:	1kHz
Résolution :	2,6mV
Précision :	2%
Charge maximale	0,1 mA
Courant maximum admissible	-1mA < I_0 < 1 mA

3.5.6. Sortie commande frein (DMAC34 seulement)

Cette sortie (+FREIN, -FREIN) non isolée permet de piloter un frein à absence de courant. La commande `BRAKE ON` coupe l'alimentation du frein, celui-ci est donc bloqué. La commande `BRAKE OFF` alimente le frein sous 24Vdc ce qui permet de le libérer.

Tension nominale du frein : 24Vdc
Courant maximum : 600mA
Précautions d'usage : respecter la polarité du frein et connecter une diode de roue libre en antiparallèle au niveau du frein pour prévenir tout problème en cas de déconnexion intempestive du bornier 8 points.

Un frein qui s'intercale à l'avant du moteur est proposé en option avec le DMAC34.

3.5.7. Sortie recopie codeur (DMAC34 seulement)

Cette sortie recopie l'état du codeur incrémental biphasé 500points/tour interne au DMAC34.
Elle est fournie sous la forme d'une liaison différentielle RS422 avec les signaux CODA, COD/A, CODB, COD/B qui sont référencés au 0V_COD (non isolé).

3.5.8. Exemple de câblage des entrées-sorties

Bornier DMAC **Câble 25 points blindé** **DMAC34**

***Nota** : Ne pas connecter une alimentation au bornier car il est alimenté via le +24Vaux du DMAC34.*

3.6. Visualisations (DMAC34 seulement)

Trois LEDs résument l'état du module DMAC34 :

- la led jaune "Busy" matérialise l'activité du moteur en signalant tout mouvement ou séquence en cours,
- la led rouge "Défaut" indique un défaut : surtension, sous-tension, température excessive ou disjonction moteur. Par ailleurs, cette led clignote à la mise sous tension, à l'initialisation et à la mise à zéro du module
- la led verte "alimentation" signale que le DMAC34 est sous tension. Attention, cette led reste active même si la tension d'alimentation est trop forte ou trop faible par rapport à la plage nominale du DMAC34.

3.7. Configuration du port de communication

Les modules DMAC peuvent communiquer en RS232 ou en RS485. Le protocole RS232 est conseillé dans les applications monoaxes en environnement non bruité. Le protocole RS485 est conseillé dans les applications multiaxes et présente une meilleure immunité au bruit.

3.7.1. Paramétrage du module

Pour paramétrer le DMAC, entrer les valeurs adéquates:
`SET_BAUDRATE` (vitesse de transfert en bauds 9600, 19200, 38400 ou 115200)
`SET_ADDRESS` (adresse du module 0 à 63)
`#LINE_DELAY` (délai de retournement de ligne RS485 en microsecondes, valeurs 100 à 3000)

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

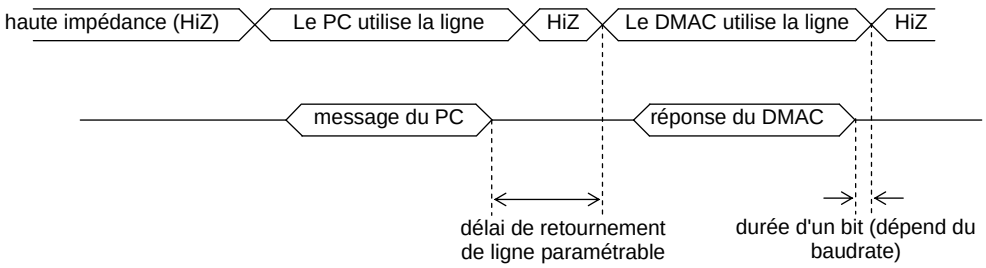
```
SET_BAUDRATE 38400
SET_ADDRESS 0
#LINE_DELAY:=3000
```

3.7.2. Le protocole RS-485

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

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

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



Comment communiquer avec le module?
Il existe plusieurs méthodes pour s'interfacer avec un module DMAC, chacune utilisant des couches d'abstraction différentes:

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

4. Commandes

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

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

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

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

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

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

Les commandes sont séparées des paramètres par un espace.

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

Dans la suite du document, on utilise les conventions suivantes:
@ représente l'adresse du module
[ ] représente un paramètre optionnel

4.1. BRAKE

Syntaxe:	[@]BRAKE paramètre
Mnémonique:	BRA
Paramètre:	ON (freine), OFF (relâche le frein)
Description:	Cette commande permet de piloter le frein à absence de courant disponible en option sur certains modules. La commande `BRAKE ON` bloque le frein, la commande `BRAKE OFF` alimente le frein ce qui permet de le libérer.
Exemple:	`08BRAKE ON` Active le freinage du module d'adresse 08.

4.2. CALL / RETURN

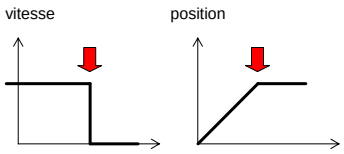
Syntaxe:	CALL paramètre RETURN
Mnémonique:	CAL RET
Paramètres:	numéro de la première ligne de la routine à exécuter pour CALL 1 < paramètre < 500
Description:	Appel d'une routine. A la fin de l'exécution de la sous routine (`RETURN`), le séquenceur continuera l'exécution linéaire à la ligne qui suit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

Exemple:

```
:4   CALL 12        Appel de la routine située à la ligne 12
     NOP
     NOP

:12  NOP            Ligne 12: début de la sous-routine
     NOP
     RETURN         Retour à la ligne 5 (qui suit le CALL)
```

4.3. HALT

Syntaxe:	[@]HALT [paramètre]
Mnémonique:	HAL
Paramètre:	Aucun, MOUV ou SEQ
Description:	Le module s'arrête sans suivre une rampe de vitesse. Le couple de freinage est maximum, la valeur de la décélération est imposée par le rapport entre le couple et l'inertie de la charge. Le suivi de position est conservé.



La commande HALT sans paramètre arrête à la fois le mouvement en cours et la séquence en cours. Il est possible de n'arrêter que le mouvement (commande `HALT MOUV`) en poursuivant l'exécution d'une séquence.
Dans le cas d'un mouvement en mode interpolé, l'interpolation est stoppée et la FIFO d'interpolation est vidée.

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

Exemple:

```
05HALT
```

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

4.4. HARD_ENDS

Syntaxe:	[@]HARD_ENDS paramètre
Mnémonique:	HEN
Paramètre:	ALL (butées actives) OFF (butées inactives) POS (active la butée positive uniquement) NEG (active la butée négative uniquement)
Description:	Active ou désactive les butées matérielles. La butée positive arrête et empêche les mouvements dans le sens positif. La butée négative arrête et empêche les mouvements dans le sens négatif. **Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.**
Remarques:	Attention, l'inversion de polarité (commande `INVERSE_POLARITY`) intervient sur les butées matérielles au même titre que sur les entrées logiques.
Exemple:	`03HARD_ENDS ALL` Active les butées matérielles du module d'adresse 03.

4.5. IF

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

4.6. INVERSE_POLARITY

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

4.7. JUMP / JUMP_REL

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

4.8. MODULE_RESET

Syntaxe:	[@]MODULE_RESET [paramètre]
Mnémonique:	MRE
Paramètre:	Aucun ou ALL => remise en configuration sortie usine.

Description:	Réinitialise le module. Le moteur est mis hors puissance puis freiné en court-circuit jusqu'à l'arrêt complet. L'état du module est ensuite similaire à l'état après mise sous tension. Les sorties sont forcées à l'état inactif. Les variables non mémorisées (#Vn) sont remises à 0. Si une séquence de démarrage automatique est programmée, celle-ci est exécutée.
	
	La directive ALL permet de replacer l'ensemble des paramètres à leurs valeurs de sortie usine (hormis les paramètres de liaison série):
	`#HIGH_SPEED := 60000`
	`#LOW_SPEED := 6000`
	`#TORQUE_RATIO := 50`
	`#ACCEL_TIME := 1000`
	`#DECEL_TIME := 1000`
	`#POSITIVE_END := +10000`
	`#NEGATIVE_END := -10000`
	`#POSITION := 0`
	`#OUTPUT_CONFIG := 3`
	`#ON_RESET := 0`
	Les variables mémorisées (#Mn) sont remises à 0.
	La polarité nominale des entrées sorties est restituée (`INVERSE_POLARITY OFF`).
	Les butées sont désactivées (`HARD_ENDS OFF; SOFT_ENDS OFF`).
	La configuration des mouvements est restituée `(S_CURVE OFF, OPTIMIZED_CURRENT OFF).`
	
	Nota : Les paramètres liaison série (baudrate, adresse et `#LINE_DELAY`) ne sont pas modifiés.

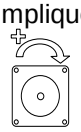
Exemple:	`02MODULE_RESET` Réinitialise le moteur d'adresse 2.

4.9. MOVE_INTERPOL

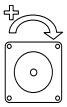
Syntaxe	[@]MOVE_INTERPOL paramètre
Mnémonique:	MIN
Paramètre:	Déplacement relatif à effectuer (en incréments moteur: 10^{-4} tour) pendant le temps `#INTERPOL_TIME` . [-2 147 483 648; +2 147 483 647]
Description:	En mode interpolé, chaque axe décrit une trajectoire définie par une succession de "segments".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

```
#INTERPOL_FIFOSIZE:=64    ;FIFO de 64 segments
#INTERPOL_TIME:=100       ;segments de 100ms
#INTERPOL_MODE:=0         ;mode SimpleSYNC

00MOVE_INTERPOL 100      ;1er segment
01MOVE_INTERPOL 750      ;(mémorisé en FIFO mais pas
02MOVE_INTERPOL 100      ;encore lancé)

SYNCHRO INTERPOL         ;départ synchronisé

00MOVE_INTERPOL 100      ;2ème segment
01MOVE_INTERPOL -50
02MOVE_INTERPOL 100

…                         ;3ème segment puis suite…
```

4.10. MOVE_ON

Syntaxe:	[@]MOVE_ON paramètre
Mnémonique:	MON
Paramètre:	Amplitude du déplacement en incrément moteur (10^{-4}tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	Le moteur effectue un mouvement du nombre d'incréments voulu suivant le profil de vitesse défini par les paramètres `#ACCEL_TIME`, `#DECEL_TIME` et `#HIGH_SPEED`. Le sens du mouvement est défini par le signe du paramètre (un paramètre positif implique une rotation dans le sens horaire lorsque l'on regarde l'axe du moteur). 
Remarques:	Cette commande force toujours une mise sous puissance du moteur.
Exemple:	`03MOVE_TO -2500` l'axe d'adresse 03 se déplace d'un quart de tour dans le sens anti-horaire

4.11. MOVE_SPEED

Syntaxe:	[@]MOVE_SPEED paramètre
Mnémonique:	MSP
Paramètre:	Vitesse de rotation du module exprimée en 100èmetour/min [-400000 ; 400000]
Description:	Le moteur tourne dans le sens donné par le signe du paramètre (un paramètre positif implique une rotation dans le sens horaire lorsque l'on regarde l'axe du moteur) à la vitesse donnée par le paramètre. Le profil de vitesse passe de la vitesse courante à la vitesse de consigne en un temps défini proportionnellement aux paramètres `#ACCEL_TIME` et `#DECEL_TIME`.
Remarques:	 La consigne de vitesse est bornée (en valeur absolue) par la vitesse maximale définie par la variable `#HIGH_SPEED`. Un mouvement avec une consigne supérieure ne renvoie pas d'erreur mais sera effectué à la vitesse maximale.
Exemple:	`00MOVE_SPEED 50000` l'axe d'adresse 00 tourne dans le sens positif à 500tr/min

4.12. MOVE_TO

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

4.13. OPEN_SEQ / CLOSE_SEQ

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

4.14. OPTIMIZED_CURRENT

Syntaxe:	[@]OPTIMIZED_CURRENT paramètre
Mnémonique:	OCU
Paramètre:	ON (mode optimisé) ou OFF (mode nominal)
Description:	Active ou désactive le mode "courant optimisé". Ce mode permet d'adapter le courant moteur au couple réellement nécessaire. Cette option permet de diminuer les pertes thermiques, notamment lorsque le moteur ne tourne pas en continu. **Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.**
Exemple:	`06OPTIMIZED_CURRENT ON` Active le mode optimisé du module d'adresse 06.

4.15. POWER

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

4.16. READ

Syntaxe:	@READ paramètre
Mnémonique:	REA
Paramètre:	Nom de la variable à relire (toutes les variables sont décrites dans la suite du document)
Description:	Relit une variable du module et renvoie sa valeur sur la liaison série vers le contrôleur (PC ou automate). Il est possible de préciser le format de relecture en faisant précéder le nom de la variable à relire de 'H' pour une relecture en hexadécimal ou de 'B' pour une relecture en binaire. H et B peuvent être indifféremment en majuscule ou minuscule.
Remarques:	Les relectures hexadécimales ou binaires renvoient 4 octets signés. La commande `READ` ne peut être utilisée qu'en commande directe. Son utilisation dans une séquence provoque une erreur. La chaîne totale renvoyée par le module est composée de l'adresse du module, du mnémonique de la variable relue, du caractère '=' et de la valeur de la variable dans le format adéquat.

Exemple:	`00READ #POSITION`
	renvoie la position courante du module d'adresse 00:
	réponse du module: `00#POS=12345`
	
	`01READ h#OUTPUT`
	renvoie l'état des sorties du module d'adresse 01:
	réponse du module: `01#OUT=h00000007`
	=> les sorties OUT1, OUT2 et OUT3 sont actives, OUT4 est inactive (ou le contraire si la polarité est inversée par la commande `INVERSE_POLARITY`).
	
	`02READ b#INPUT`
	renvoie l'état des entrées logiques du module d'adresse 02:
	réponse du module: `02#INP=b00000000 00000000 00000000 00001010`
	=> les entrées IN2 et IN4 sont actives, les entrées IN1, IN3,IN5 et IN6 sont inactives (ou le contraire si la polarité est inversée par la commande `INVERSE_POLARITY`).

4.17. READ_SEQ

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

4.18. REFERENCE

Syntaxe:	[@]REFERENCE paramètre
Mnémonique:	REF
Paramètre:	ON (mode actif) ou OFF (mode inactif)
Description:	Active ou désactive le mode "référence". Ce mode permet d'initialiser la position sur un capteur de position. La position est mise à zéro si le moteur tourne dans le sens positif et qu'un front montant (transition 0 à 1) est détecté sur l'entrée IN5. Le mode est automatiquement désactivé une fois que la position a été initialisée.
Exemple:	`06REFERENCE ON` Active le mode référence du module d'adresse 06.

4.19. REQUEST_VERSION

Syntaxe: [@]REQUEST_VERSION

Mnémonique: RV OU RVE

Paramètre: Aucun

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

Détail du code logiciel:

CODE	PRODUIT ET LOGICIEL
9148	DMAC17 standard
A148	DMAC17 boîtier
B148	DMAC17 horloge & sens
K148	DMAC17 CAN
F138	DMAC23 standard
J138	DMAC23 horloge & sens
K138	DMAC23 CAN
H142	DMAC34 standard
J142	DMAC34 horloge & sens
K142	DMAC34 CAN

Exemple: `04 REQUEST_VERSION`
pourrait renvoyer, par exemple, la chaîne suivante:
04EV v1.7 H142 "MIDI-INGENIERIE_DMAC34-1_H142-10145_25/01/05_12/04/06" PHASE:6A BOOT:v1.1
Il s’agit donc du module DMAC34-1 n° H142-10145 fabriqué le 21/01/05 et révisé le 12/04/06. Il est programmé avec le logiciel standard H142 en version 1.7.

4.20. SET_ADDRESS

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

Exemple:	`04SET_ADDRESS 3`
	Le module d'adresse 4 devient le module d'adresse 3
	
	`04SET_ADDRESS 0`
	Le module d'adresse 4 devient le module d'adresse 0. Il va à présent répondre aux commandes générales.
	
	`SET_ADDRESS 6`
	Attention, tous les modules sont renommés à l'adresse 6. Il y a conflit si plusieurs modules sont présents.

4.21. SET_BAUDRATE

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

4.22. S_CURVE

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

4.23. SOFT_ENDS

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

4.24. START_SEQ

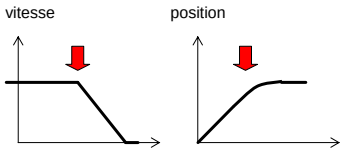
Syntaxe:	[@]START_SEQ [paramètre]
Mnémonique:	SSE
Paramètre:	Numéro de la première ligne à exécuter (1 à défaut).
Description:	Exécution du séquenceur à partir de la ligne précisée en argument. Si aucun argument n'est précisé, le séquenceur commence à la ligne 1.

Exemple:	`START_SEQ 15`
	Lancement du séquenceur à la ligne 15 pour tous les modules présents.
	
	`02START_SEQ`
	Lancement du séquenceur à la ligne 1 pour le module 2.

4.25. STEP

Syntaxe:	[@]STEP [nl]
Mnémonique:	STE
Paramètre:	nl : numéro de la ligne de séquence à exécuter
Description:	Si nl > 0 le module exécute la commande décrite par la ligne de séquence donnée. Si nl n'est pas donné ou nul, le module exécute la ligne de séquence qui suit la dernière ligne réalisée.
Exemple:	`04STEP 25` Le module 4 effectue la commande placée à la ligne 25.

4.26. STOP

Syntaxe:	[@]STOP [paramètre]
Mnémonique:	STO
Paramètre:	Aucun, MOUV ou SEQ
Description:	Le module s'arrête en passant de la vitesse courante à une vitesse nulle en un temps défini proportionnellement au paramètre `#DECEL_TIME`.  La commande STOP sans paramètre arrête à la fois le mouvement en cours et la séquence en cours. Il est possible de n'arrêter que le mouvement (commande `STOP MOUV`) en poursuivant l'exécution d'une séquence. Il est également possible de n'arrêter que la séquence (commande `STOP SEQ`) et de poursuivre le mouvement en cours. NB : dans le cas d'un mouvement en mode interpolé, la commande `STOP` est équivalente à la commande `HALT` (arrêt immédiat). Dans les deux cas la FIFO d'interpolation est vidée.

Exemple:	`04STOP`
	l'axe d'adresse 04 décélère jusqu'à l'arrêt. Si une séquence était en cours d'exécution, elle est arrêtée.
	`03STOP MOUV`
	l'axe d'adresse 03 décélère jusqu'à l'arrêt. Si une séquence était en cours d'exécution, elle continue.

4.27. SYNCHRO

Syntaxe:	SYNCHRO [paramètre]
Mnémonique:	SYN
Paramètre:	ON, OFF, TOP ou INTERPOL
Description:	Ce jeu de commandes permet un démarrage synchronisé de l'ensemble des axes. `SYNCHRO ON` passe en mode synchrone (inutile en cas d'interpolation) `SYNCHRO OFF` quitte le mode synchrone `SYNCHRO TOP` lance le mouvement (hors interpolation) `SYNCHRO INTERPOL` lance le mouvement d'interpolation Note : si nécessaire, le moteur est automatiquement mis sous tension au démarrage du mouvement. Pour une meilleure synchronisation il est toutefois conseillé de le mettre sous puissance au préalable avec la commande `POWER ON`.

Exemple:	1) mouvement non interpolé
	`SYNCHRO ON` ; entrée en mode synchrone sur tous les axes
	`00MOVE_SPEED 4000`
	`03MOVE_SPEED -12500`
	`SYNCHRO TOP` ; départ des axes 0 et 3
	`SYNCHRO OFF` ; quitte le mode synchrone sur tous les axes (facultatif)
	2) interpolation (voir commande MOVE_INTERPOL chapitre 4.4)

4.28. WAIT

Syntaxe:	WAIT paramètre
Mnémonique:	WAI
Paramètres:	temps d'attente en millisecondes [-3600000 ; 3600000]
Description:	Temporisation. L'exécution du séquenceur est suspendue jusqu'à ce que le délai soit écoulé. Si le temps d'attente spécifié est nul ("`WAIT 0`"), l'exécution est suspendue jusqu'à la fin du mouvement en cours (asservissement de position inclus). Si le temps d'attente spécifié est négatif ("`WAIT -1000`"), l'exécution est suspendue jusqu'à la fin du mouvement en cours avec un timeout équivalent à la valeur absolue du paramètre.

Exemple:	`WAIT 2000`
	Le séquenceur attend 2 secondes avant de passer à la ligne suivante.
	`WAIT 0`
	Le séquenceur attend que le mouvement en cours soit terminé avant de passer à la ligne suivante.
	`WAIT -2000`
	Le séquenceur attend au maximum 2 secondes que le mouvement en cours soit terminé avant de passer à la ligne suivante.

5. Variables internes

5.1. Généralités

La forme générale des commandes d'accès aux variables est la suivante:
[@]#VARIABLE[.BIT]:= [- ! H B]VALEUR pour une écriture
[@]READ [H B]#VARIABLE pour une lecture.

La réponse à la relecture d'une variable est du type:
@MNEMONIQUE=VALEUR

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

5.1.1. Format décimal

Le format par défaut est le format décimal.
Le caractère "+" devant les valeurs positives ou nulles est facultatif. Il est en revanche toujours spécifié dans les relectures.
Exemples:

```
00#ACCEL_TIME:=123
00#V20:=-40
00READ #ACCEL_TIME → 00#ATI=+123
```

5.1.2. Format hexadécimal

En faisant précéder le nom de la variable du caractère "H", on peut la saisir et la relire au format hexadécimal.
Les valeurs négatives sont notées en "complément à deux" sur quatre octets (exemple: -10 se note HFFFFFFF6).
Il est facultatif de préciser les zéros non significatifs. La relecture se fait toujours sur quatre octets.

Exemples:

```
00#ACCEL_TIME:=H100
00#V20:= HFFFFFFD8
00READ h#ACCEL_TIME → 00#ATI=h00000100
```

5.1.3. Format binaire

En faisant précéder le nom de la variable du caractère "B", on peut la saisir et la relire au format binaire.
Les valeurs négatives sont notées en "complément à deux" sur quatre octets.
Il est facultatif de préciser les zéros non significatifs. La relecture se fait toujours sur trente-deux bits, chaque octet étant séparé par un espace.

Exemples:

```
00#ACCEL_TIME:=B1100100
00READ b#ACCEL_TIME → 00#ATI=b00000000 00000000 00000000 01100100
```

5.1.4. Opposé et complément

En faisant précéder le nom de la variable du caractère "-" ou "!", on accède respectivement à l'opposé ou au complément binaire de sa valeur.
Cette notation est particulièrement utile dans les opérations ou les tests en séquence.

Exemples:

```
00#V21:=-#V1
00#V23:=#V15 & !#STATUS
00MOVE_TO -#POSITION
```

5.1.5. Format bit-à-bit

Pour accéder à un seul bit d'une variable, on peut faire suivre le nom de la variable d'un point et du numéro du bit (de 1 à 32, 1 étant le bit de poids faible et 32 étant le bit de poids fort).
Le bit de la variable joue alors le rôle d'une variable à part entière. Sa valeur peut être 1 ou 0.
On peut également utiliser cette notation dans un calcul ou un test.

Exemples:

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

5.1.6. Opérations sur variables

Il est possible d'effectuer un certain nombre d'opérations et de tests sur les variables.
La syntaxe est la suivante:
[@]#VARIABLE:=OPERANDE1 OPERATEUR OPERANDE2

Il est nécessaire d'inclure au moins un espace avant et après l'opérateur.
Les opérandes peuvent être soit une variable interne, soit une constante (éventuellement notée en binaire ou en hexadécimal comme décrit précédemment).

L'opérateur est un des opérateurs suivants:

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

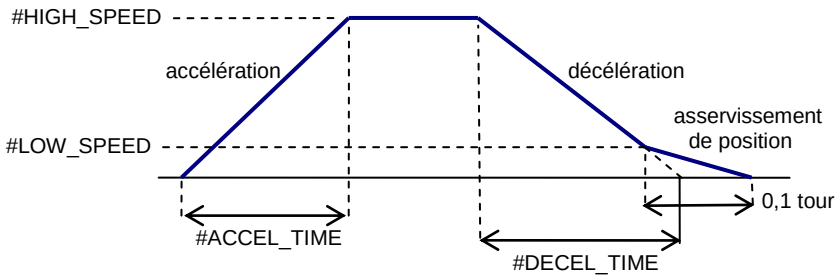
Dans le cas des opérateurs de test (<,>,>=,<=,!), le résultat de l'opération vaut 1 si le test est VRAI et 0 si le test est FAUX. Ces opérateurs servent en particulier aux sauts conditionnels en séquence (IF…JUMP).

5.1. #ACCEL_TIME, #DECEL_TIME

Mnémonique #ATI, #DTI

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

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



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

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

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

Le temps d'accélération effectif correspond donc à:

$$\#ACCEL\_TIME \times \frac{|Vitesse\ Cible - Vitesse\ Courante|}{\#HIGH\_SPEED}$$

De même, le temps de décélération effectif est de :

$$\#DECEL\_TIME \times \frac{|Vitesse\ Cible - Vitesse\ Courante|}{\#HIGH\_SPEED}$$

Exemple:
`#ACCEL_TIME:=1000` (1s)
`#HIGH_SPEED:=60000` (600tr/min)
`MOVE_SPEED 30000` → le temps d'accélération sera de 0.5s

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

Exemple: `00#ACCEL_TIME:=1000`
Le module mettra 1 seconde pour passer d'une vitesse nulle à `#HIGH_SPEED`.

5.2. #CAPTURE (lecture seule)

Mnémonique	#CAP
Paramètre:	Valeur de la variable `#POSITION` lors du dernier front montant de IN5
Description:	A chaque front montant (0→1) de l'entrée logique IN5, la position courante est mémorisée dans la variable `#CAPTURE`.
Exemple:	`03READ #CAPTURE` → `03#CAP=27895` Le module d'adresse 3 était à la position 27895 lors du dernier front montant de IN5.

5.3. #CPU_TEMPERATURE (lecture seule)

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

5.4. #DRIVER_TEMPERATURE, #MOTOR_TEMPERATURE (lecture seule, DMAC34)

Mnémonique	#DTE (DMAC34 seulement) #MTE (DMAC34 seulement)
Paramètre:	#DTE indique la température de l'étage de puissance du DMAC34 (unité = 0,1°C) #MTE indique la température du moteur du DMAC34 (unité = 0,1°C)
Exemple:	`02READ #MOTOR_TEMPERATURE` → `02#MTE=650` Le bloc moteur du DMAC34 d'adresse 2 a une température de 65°C.

5.5. #ERROR

Mnémonique #ERR

Paramètre: Aucun

Description: La relecture des erreurs du DMAC renvoie une valeur dont les bits représentent l'état du module:

poids fort	bit 32	
	…	
	bit 12	NonDef: Commande ou Variable non définie (syntaxe)
	bit 11	
	bit 10	
	bit 9	
	bit 8	Calcul (division par zéro, etc.…)
	bit 7	Limite (paramètre de commande hors limite)
	bit 6	
	bit 5	Surtension
	bit 4	Sous-tension
	bit 3	Court-circuit
	bit 2	Disjonction thermique
poids faible	bit 1	

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

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

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

5.6. #HIGH_SPEED

Mnémonique #HSP

Paramètre: Vitesse de consigne en 100ème tour/min
[0 ; 400000]

Description: Vitesse de consigne de tous les déplacements en mode position (`MOVE_TO` et `MOVE_ON`) et la vitesse maximale des déplacements en mode vitesse (`MOVE_SPEED`).

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

Remarques: Lors de petits déplacements, la vitesse de consigne n'est pas forcément atteinte.

Exemple:
```
04#HIGH_SPEED:=20000
```
Lors des prochains mouvements en mode position, les mouvements de l'axe 04 se feront à 200tr/min.

5.7. #INPUT (lecture seule)

Mnémonique #INP

Paramètre: Valeur qui représente bit à bit l'état des entrée logiques (en binaire actif à 1, inactif à 0)

Description: Relit les entrées logiques IN1 (bit de poids faible) à IN6 ou IN10 suivant les modules (bit de poids fort).
Si la polarité des entrées est inversée (commande `INVERSE_POLARITY` représentée par le bit `#STATUS.13`), l'état logique de `#INPUT` est inversé par rapport à l'état physique des entrées.

Exemple: `1) DMAC23 d'adresse 01` (avec `INVERSE_POLARITY OFF`)

`01READ #INPUT` → `01#INP=19`

19 (décimal) =	0	0	0	1	0	0	1	1	(binaire)
			OFF	ON	OFF	OFF	ON	ON	
	----	----	IN6	IN5	IN4	IN3	IN2	IN1	

`2) DMAC34 d'adresse 05` (avec `INVERSE_POLARITY OFF`)

`05READ #INPUT` → `05#INP=19`
`05READ h#INPUT` → `05#INP=h13`
`05READ b#INPUT` → `05#INP=b00000000000000000000000000010011`

19 (décimal) =	0	0	0	0	0	1	0	0	1	1	(binaire)
	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON	ON	
	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	

5.8. #INPUT_ANALOG, #INPUT_A1, #INPUT_A2 (lecture seule)

Mnémonique #IAN (DMAC17 et DMAC23)
#IA1, #IA2 (DMAC34)

Paramètre: Tension de l'entrée analogique différentielle en mV.

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

`04READ #INPUT_A2` → `04#IA2=1500`
Le DMAC34 d'adresse 4 a une tension de 1,5V sur l'entrée analogique n°2.

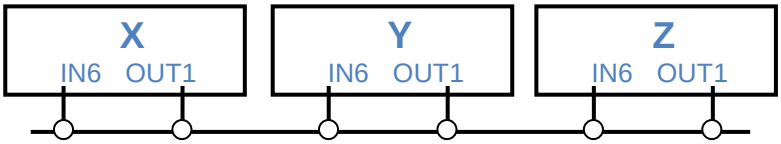
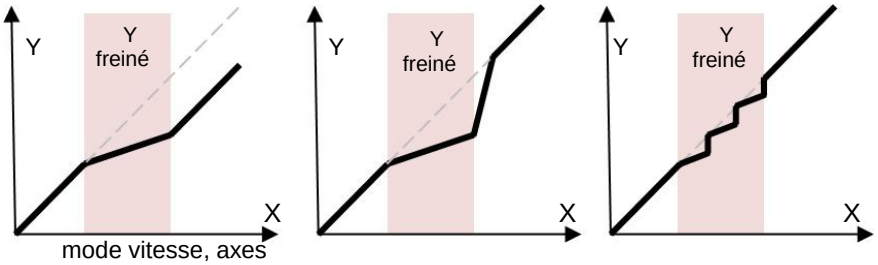
5.9. #INTERPOL_COUNT (lecture seule)

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

5.10. #INTERPOL_FIFOSIZE

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

5.11. #INTERPOL_MODE

Syntaxe	[@]#INTERPOL_MODE paramètre
Mnémonique:	#IMO
Paramètre:	0 (SimpleSYNC) ou -1 (UltraSYNC)
Description:	Cette variable permet de choisir entre les deux types de synchro disponibles pour le mode "Trajectoires Interpolées". **Mode SimpleSYNC** (`#INTERPOL_MODE:=0`) Dans ce mode, la synchronisation multi-axe est assurée par le démarrage simultané du mouvement sur tous les axes (envoi de la commande globale `SYNCHRO INTERPOL`). La durée des segments est ensuite assurée sur chaque axe indépendamment au moyen d'un timer interne de précision. **Mode UltraSYNC** (`#INTERPOL_MODE:=-1`) Dans ce mode, la synchronisation multi-axe est assurée par un chaînage des entrées-sorties de tous les axes. En connectant l'entrée IN6 et la sortie OUT1 de tous les axes du réseau, ils se synchronisent automatiquement, et si un des axes est freiné ou ralenti (frottements, point dur, etc…) les autres axes attendent qu'il ait fini son segment pour passer au suivant. 
Exemple:	Exemple dans le cas d'un système 2 axes X-Y:  mode vitesse, axes indépendants (non interpolé) SimpleSYNC UltraSYNC

5.12. #INTERPOL_TIME

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

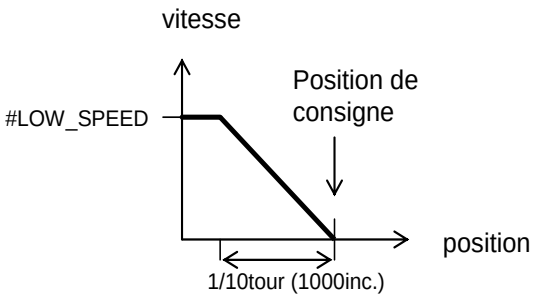
5.13. #LINE_DELAY

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

5.14. #LINE

Mnémonique	#LIN
Paramètre:	Ligne du séquenceur en cours d'exécution [0 ; 500]
Description:	La variable `#LINE` représente la ligne qui est en train d'être exécutée par le module.
Remarques:	Lorsque le séquenceur n'est pas démarré, la variable `#LINE` vaut zéro. La mise à zéro de cette variable (`#LINE:=0`) arrête le séquenceur.
Exemple:	`READ #LINE` → `00#LIN:=182` Le module d'adresse 0 est en train d'effectuer la ligne 182 de la séquence.

5.15. #LOW_SPEED

Mnémonique	#LSP
Paramètre:	Vitesse d'approche en 100ème tour/min [0 ; 400000]
Description:	Vitesse d'approche utilisée en fin de déplacement pour la gestion du positionnement en mode position (`MOVE_TO` et `MOVE_ON`). Cette vitesse est également utilisée pour revenir à la position de consigne si l'axe s'en écarte. Le profil de vitesse utilisé lors de l'asservissement de position est défini uniquement par la valeur `#LOW_SPEED`:  **Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.**
Remarques:	Cette variable permet de régler le temps de réponse de l'asservissement de position et sa stabilité. Le temps de réponse diminue lorsque `#LOW_SPEED` augmente. Par contre si `#LOW_SPEED` est trop élevée il y a des risques d'oscillations notamment si le moteur est mécaniquement chargé par une forte inertie avec peu de frottements.
Exemple:	`04#LOW_SPEED:=2000` Lors des prochains mouvements en mode position, l'asservissement de position en fin de mouvement s'effectuera à 20tr/min maximum.

5.16. #M1 à #M8

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

5.17. #NEGATIVE_END

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

5.18. #ON_RESET

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

5.19. #OUTPUT

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

5.20. #OUTPUT_A1, #OUTPUT_A2 (DMAC34 seulement)

Mnémonique	#OA1, #OA2
Paramètre:	Valeur de la tension analogique à présenter sur la sortie analogique 1 ou 2, celle-ci étant exprimée en mV dans la plage [0 ; + 10 000].
Description:	Pilotage des sorties analogiques du module.
Remarques:	Cette fonction est disponible sur DMAC34 uniquement. Consulter le chapitre de description des entrées/sorties pour plus de détails.
Exemple:	`00#OUTPUT_A1:=4520` force la sortie analogique n°1 à 4,52V

5.21. #OUTPUT_CONFIG

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

5.22. #POSITION

Mnémonique	#POS
Paramètre:	Nouvelle position de l'axe en incrément moteur (10^{-4}tour) [-2 147 483 648 ; + 2 147 483 647]
Description:	Permet de forcer la position de référence interne du module. On peut, en particulier, utiliser cette commande pour définir l'origine d'un système mécanique en donnant 0 pour paramètre.
Remarques:	**Cette variable est mémorisée à la coupure d'alimentation et restituée à la mise sous tension. Sa valeur initiale n'est donc pas valable si l'axe moteur a été tourné hors tension ou bien si la coupure d'alimentation est intervenue en cours de mouvement.** Attention à l'utilisation de cette commande lorsque les butées virtuelles sont activées: spécifier une position au-delà d'une butée a pour effet d'arrêter le mouvement.
Exemple:	`04#POSITION:=0` Tous les mouvements en mode position du module d'adresse 04 seront référencés par rapport à la position courante.

5.23. #POSITIVE_END

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

5.24. #SPEED, #PROFILE_SPEED (lecture seule)

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

5.25. #STATUS (lecture seule)

Mnémonique #STA

Paramètre: Aucun

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

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

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

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

5.26. #SUPPLY_VOLTAGE (lecture seule)

Mnémonique	#SVO
Paramètre:	Mesure de la tension d'alimentation du module en mV
Exemple:	`02READ #SUPPLY_VOLTAGE` → `02#SVO=32000` Le module d'adresse 2 est alimenté avec une tension de 32V.

5.27. #TIMER_1 à #TIMER_3

Mnémonique	#T1 à #T3
Paramètre:	Tempo en millisecondes [0 ; + 2 147 483 647]
Description:	Les variables `#TIMER_1` à `#TIMER_3` permettent d'introduire des temporisations dans une séquence. A partir de la valeur initiale définie par l'utilisateur, la valeur des variables est décrémentée toutes les millisecondes jusqu'à zéro. Ces variables peuvent être relues (`READ #TIMER_1`) ou testées (`IF #TIMER_1=0 JUMP…`) pour savoir si le délai est écoulé ou non.
Exemple:	`#TIMER_3:=2000` La variable `#TIMER_3` est chargée avec la valeur 2000. Elle sera décrémentée toutes les millisecondes pendant deux secondes.

5.28. #TORQUE_RATIO

Mnémonique	#TRA
Paramètre:	Pourcentage du couple total disponible. [0 ; 100]
Description:	Cette variable permet de réduire le courant injecté dans le moteur dans le cas où le couple maximal n'est pas nécessaire. La consommation sur l'alimentation et l'échauffement du moteur seront d'autant plus faibles. **Ce paramètre est mémorisé et se conserve en cas de coupure d'alimentation.**
Remarques:	Les performances maximales du moteur (vitesse de rotation et puissance) nécessitent une valeur `#TORQUE_RATIO` réglée à 100.
Exemple:	`00#TORQUE_RATIO:=70` Le module 0 dispose de 70% du couple nominal.

5.29. #V1 à #V32

Mnémonique:	#V1 à #V32
Paramètre:	Valeur signée comprise entre -2^{31} et $2^{31}-1$ [-2 147 483 648 ; + 2 147 483 647]
Description:	Les variables `#V1` à `#V32` peuvent être utilisées dans une ligne de séquence ou en commande directe pour stocker des valeurs ou pour faire des calculs. Ces variables sont initialisées à 0 (zéro) à chaque mise sous tension ou reset du module.
Exemple:	`00#V13:=1234` Mémorise la valeur 1234 dans la variable `#V13` du module d'adresse 0.

6. Séquenceur

6.1. Présentation des Fonctionnalités

Le DMAC offre la possibilité de mémoriser puis d'exécuter une suite de commandes.
Chaque ligne est référencée par son "numéro de ligne" (de 1 à 500).
Lorsque le séquenceur est à la ligne 0 (`#LINE=0`), le séquenceur est OFF.
Lorsque le séquenceur est démarré (commande "`START_SEQ n`" où `n` est la première ligne à être exécutée), le module exécute une ligne toutes les millisecondes puis passe à la ligne suivante (incrémentation automatique de `#LINE`).

6.2. Mémorisation des lignes

Pour mémoriser les lignes:

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

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

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

L'automatisme mémorisé sera donc:
Ligne 1 : `MOVE_ON 10000`
Ligne 2 : `WAIT 2000`
Ligne 3 : `JUMP 1`

6.3. Exécution du séquenceur

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

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

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

Exemple: pour exécuter une séquence qui commence à la ligne 12 à la mise sous tension du module, envoyer la commande " `#ON_RESET:=12` ".
Pour désactiver la séquence de démarrage, entrer simplement `#ON_RESET:=0`

6.4. Exemples de séquences

6.4.1. Exemple 1

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

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

6.4.2. Exemple 2

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

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

6.4.3. Exemple 3

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

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

6.4.4. Exemple 4

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

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

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

7. ANNEXES

7.1. Résumé des commandes

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

(*) DMAC34 seulement

7.2. Résumé des variables

Variable	Mnémo.	Mémorisé	Lecture seule	Valeur usine ou suite à MRE ALL
#ACCEL_TIME	#ATI	◊		1 000
#CAPTURE	#CAP		◊	
#CPU_TEMPERATURE	#CTE		◊	
#DECEL_TIME	#DTI	◊		1 000
#DRIVER_TEMPERATURE (*)	#DTE		◊	
#ERROR	#ERR			
#HIGH_SPEED	#HSP	◊		60 000
#INPUT	#INP		◊	
#INPUT_ANALOG	#IAN		◊	
#INPUT_A1, #INPUT_A2 (*)	#IA1, #IA2		◊	
#INTERPOL_COUNT	#ICO		◊	
#INTERPOL_FIFOSIZE	#IFI			64
#INTERPOL_MODE	#IMO			0
#INTERPOL_TIME	#ITI			100
#LINE_DELAY	#LDE	◊		3 000
#LINE	#LIN			0
#LOW_SPEED	#LSP	◊		6 000
#M1 à #M8	#M1 à #M8	◊		0
#MOTOR_TEMPERATURE (*)	#MTE		◊	
#NEGATIVE_END	#NEN	◊		-100 000
#ON_RESET	#ORE	◊		0
#OUTPUT	#OUT			0
#OUTPUT_A1, #OUTPUT_A2 (*)	#OA1, #OA2			0
#OUTPUT_CONFIG	#OCO	◊		3
#POSITION	#POS	◊		0
#POSITIVE_END	#PEN	◊		+100 000
#PROFILE_SPEED	#PSP		◊	
#SPEED	#SPE		◊	
#STATUS	#STA		◊	
#SUPPLY_VOLTAGE	#SVO		◊	
#TIMER_1 à #TIMER_3	#T1 à #T3			0
#TORQUE_RATIO	#TRA	◊		50
#V1 à #V32	#V1 à #V32			0

(*) DMAC34 seulement

7.3. Documents associés

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

- Note d'application – Liaison calculateur : protocoles et syntaxe
- Note d'application – DMAC Horloge & Sens
- Résumé des commandes DMAC et microMAC (français)
  Commands Abstract – DMAC and microMAC (anglais)
- Notice d'utilisation du bornier DMAC / microMAC
- Note d'application DMAC CAN