Albatros

3.2.3

Commande numérique

Tecnologie e Prodotti per l'Automazione

Sommaire

1 Introduction

1.1 Comment utiliser ce manuel

Ce fichier d'Aide décrit les fonctions de la commande numérique Albatros.
Le fichier d'Aide a été conçu de façon à aider l'opérateur à mieux comprendre le système et son utilisation.

Les points saillants des différentes sections sont les suivants :

- les fenêtres, les instruments de Albatros
- la description de l'architecture typique d'un système Albatros
- comment visualiser les dispositifs et agir sur ceux-ci en mode manuel et diagnostique, en utilisant le tableau synoptique
- comment visualiser et utiliser les Paramètres Technologiques et Géométriques et le Paramètres Outils.
- comment visualiser les Dispositifs et comment y agir avec les fonctions Manuel et Diagnostic.

Afin d'éviter d'alourdir excessivement ce fichier, il est conseillé à l'opérateur de consulter les manuels du Système d'Exploitation Windows®, pour approfondir ses connaissances quant à l'emploi de la *souris*, des *menus*, des *barres d'outils* et de toutes les fonctions opérationnelles classiques de Windows.

1.2 Les fenêtres de travail

Il existe de différents types de fenêtres de travail en fonction du type d'opération que l'on a l'intention d'accomplir et elles peuvent être toutes ouvertes en même temps. Il est donc possible d'ouvrir, par exemple, la fenêtre de la liste d'exécution et celle des dispositifs d'un ou de plusieurs modules.

Les types de fenêtres sont les suivants :

Fenêtre	**Description**
Principale	fenêtre principale de Albatros. Elle permet de rappeler des fonctions et elle contient d'autres fenêtres dont le contenu dépend de l'application spécifique qu'elles représentent.
Synoptique	contient une représentation graphique de la machine ou des parties de cette dernière, permettant ainsi d'y intervenir.
Paramètres technologiques	permet de visualiser et de modifier les paramètres technologiques.
Paramètres Outils	permet de visualiser et de modifier les paramètres d'outils.
Diagnostic	permet de visualiser l'état des dispositifs et, si cela est possible, d'y intervenir
Erreurs de Système	fenêtre avec la liste des dernières erreurs de système ayant eu lieu. Il est également possible de visualiser les erreurs de cycle et les messages.
Configuration de système	permet de visualiser et modifier les dispositifs logiques et physiques de la machine

2 Composition du Système

2.1 Droits d'accès au système

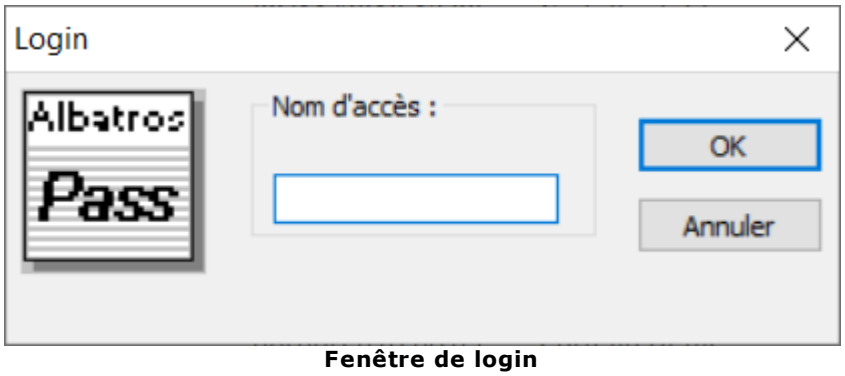
Albatros prévoit quatre niveaux d'accès au système :

- Utilisateur : Il s'agit du niveau qui présente le plus grand nombre de restrictions d'accès. Aucun des réglages relatifs aux dispositifs ne peut être modifié. Ce niveau est utilisé pour exécuter les usinages et les opérations normales de la machine. Quand le système est mis en marche, ce niveau d'accès est activé automatiquement.
- Maintenance : Il s'agit du niveau qui est utilisé pour effectuer l'entretien ordinaire de la machine. L'opérateur devrait pouvoir modifier certains paramètres de configuration, sans effectuer de modifications à la structure de la machine.
- Constructeur : Il s'agit du niveau qui est utilisé pour configurer les installations et les machines. Dans ce niveau, il est possible d'effectuer presque toutes les modifications exécutables. Il est utilisé par les programmateurs.
- Tpa : Il s'agit du plus haut niveau d'accès au système. Il sert à protéger l'accès à certains réglages particulièrement délicats et dont la modification réclame une connaissance profonde de Albatros. Il n'est utilisé que très rarement. Le mot de passe d'accès à ce niveau doit être demandé directement à la société TPA.

Pour accéder au système avec un niveau d'accès supérieur à celui de l'Utilisateur ou pour remettre le système au niveau Utilisateur, après avoir effectué des modifications à un niveau supérieur, il est nécessaire d'insérer le mot de passe correspondant.

Pour rappeler la fenêtre qui permet d'insérer le mot de passe, il est nécessaire d'utiliser la combinaison de touches **Ctrl + *** (astérisque). Alternativement à droite de la **Barre des applications** de Windows il y a l'icone . En cliquant avec le bouton droite de la souris sur l'icône il est possible de visualiser un menu dans lequel comparait l'option **Change pass level**.

La fenêtre qui s'ouvre se présente de la façon suivante :



Fenêtre de login

Il est alors possible d'insérer le mot de passe et d'appuyer sur le bouton **[OK]** pour confirmer. Les caractères qui composent le mot de passe seront substitués par des caractères "*", de façon à ce que personne ne puisse lire le mot de passe qui vient d'être déclaré.
Une fois le mot de passe inséré, on se trouve déjà au niveau d'accès correspondant. Pour avoir une confirmation du niveau d'accès, il est possible de sélectionner l'option **Informations sur Albatros** dans le menu **?**.

Lorsque le mot de passe inséré n'est pas correct, l'erreur est mise en évidence par le message "Attention ! Mot de passe Incorrect !".

2.2 Support multilingue

Albatros supporte plusieurs langues.

Changement de langue

Le changement de langue peut être effectué à n'importe quel niveau d'accès au système. Pour changer la sélection de la langue, il est nécessaire d'utiliser la combinaison de touches **Ctrl** + **/** ou de sélectionner l'icône  dans le "**barre des applications**" de Windows.
Il est alors possible de sélectionner la langue et de cliquer sur la touche **[OK]**.
Le changement de la langue n'est pas exécuté immédiatement, mais au nouvel redémarrage de Albatros.

2.3 Architecture typique d'un système

Étant donné que, dans les représentations graphiques et dans la structure des données de base de Machine, de nombreux aspects dépendent fortement du type de la machine, au-delà de quelques informations d'ordre général, ce fichier d'Aide fournit, à titre indicatif, une description de la composition d'un système classique.

Bien entendu, les indications réelles, les schémas et les pages graphiques du véritable système dépendent de l'application spécifique et sont donc programmées par le fabricant de la machine-outil.

La commande numérique Albatros se constitue d'une unité de contrôle, basée sur ordinateur personnel, qui montre l'interface Opérateur-Machine, et d'un nombre variable d'unités de traitement (de 1 a 16), pour le pilotage et le contrôle de toutes les ressources opérationnelles de la machine-outil ou de l'installation.
Il est donc possible d'avoir deux versions d'équipement:

Monomodule constituée d'un seul module raccordé directement au bus de l'ordinateur personnel.

Multimodule constituée d'un minimum de 1 à un maximum de 16 modules et prévue uniquement pour les applications effectuées sur des installations ou des lignes de plusieurs machines; dans ce cas, l'ordinateur personnel est matériellement séparé des unités de traitement qui peuvent se situer en différents endroits de la ligne ou de l'installation.

Dans les deux types d'architecture, les unités de traitement sont constituées d'une ou de plusieurs cartes axes pour le contrôle direct des axes de la machine et la gestion logique des dispositifs d'Entrée/Sortie. Dans la version monomodule, les cartes axes sont installées directement sur le PC Superviseur. En revanche, dans la version multimodule, elles sont installées sur un PC industriel (avec ou sans écran et clavier) qui est raccordé au PC Superviseur au moyen du réseau Ethernet. La figure suivante présente un schéma de raccordement entre le PC Superviseur et un module à distance (Clipper). Elle indique également les principales activités qui sont accomplies par les différents composants.

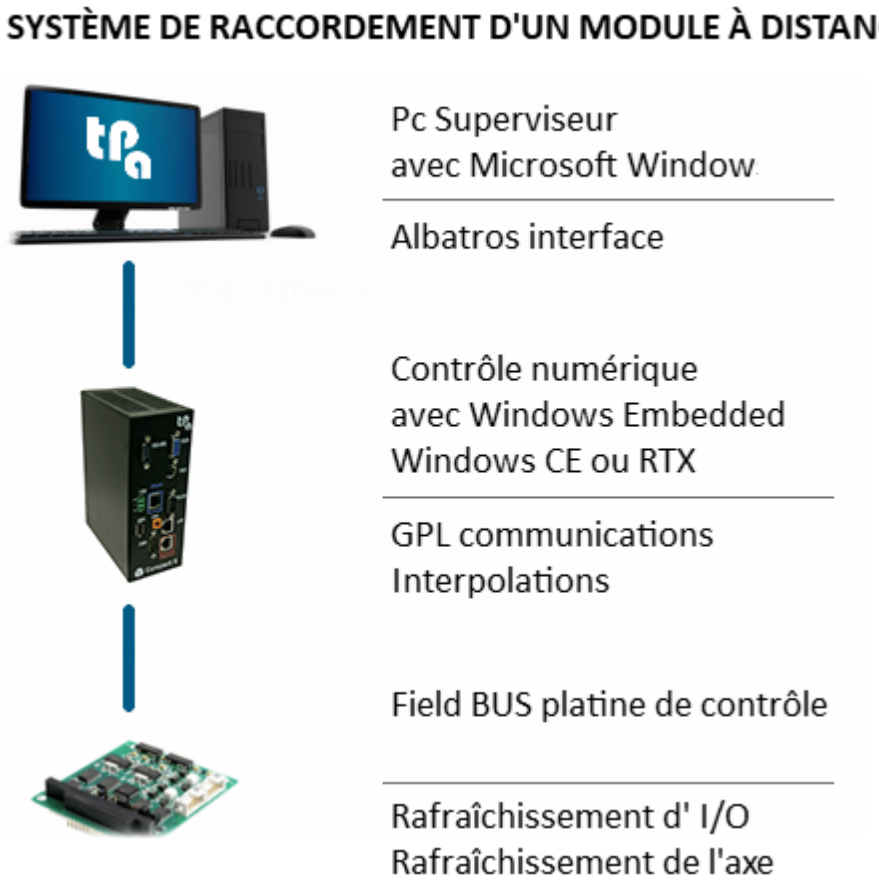


Schéma de raccordement d'un module à distance

Dispositifs à distance intelligents pilotent les dispositifs I/O et les axes (TRS-AX à distance) directement sur la machine. Ces dispositifs assurent la lecture des lignes d'entrée numériques (ON / OFF) ou analogiques et le rafraîchissement des lignes de sortie numériques ou analogiques et ils sont raccordés aux modules par GreenBUS (bus sériel RS485 - 1 Mbaud) et bus CAN et EtherCAT.
Le fonctionnement de Albatros dans la machine est protégé par une clé matérielle usb, configurée par TPA.

2.4 Organisation et configuration logique

Dans le système Albatros la structure descriptive de l'installation ou de la machine-outil est organisée dans des archives technologiques selon une structure hiérarchique

Cette conformation reflète le besoin de conserver, au niveau des données de configuration et de modalité d'accès, l'éventuelle structure modulaire des machines, en la classant en termes d'agrégation dynamique de différents modules, agrégats et dispositifs que l'on peut insérer ou exclure en fonction des différents aménagements possibles.

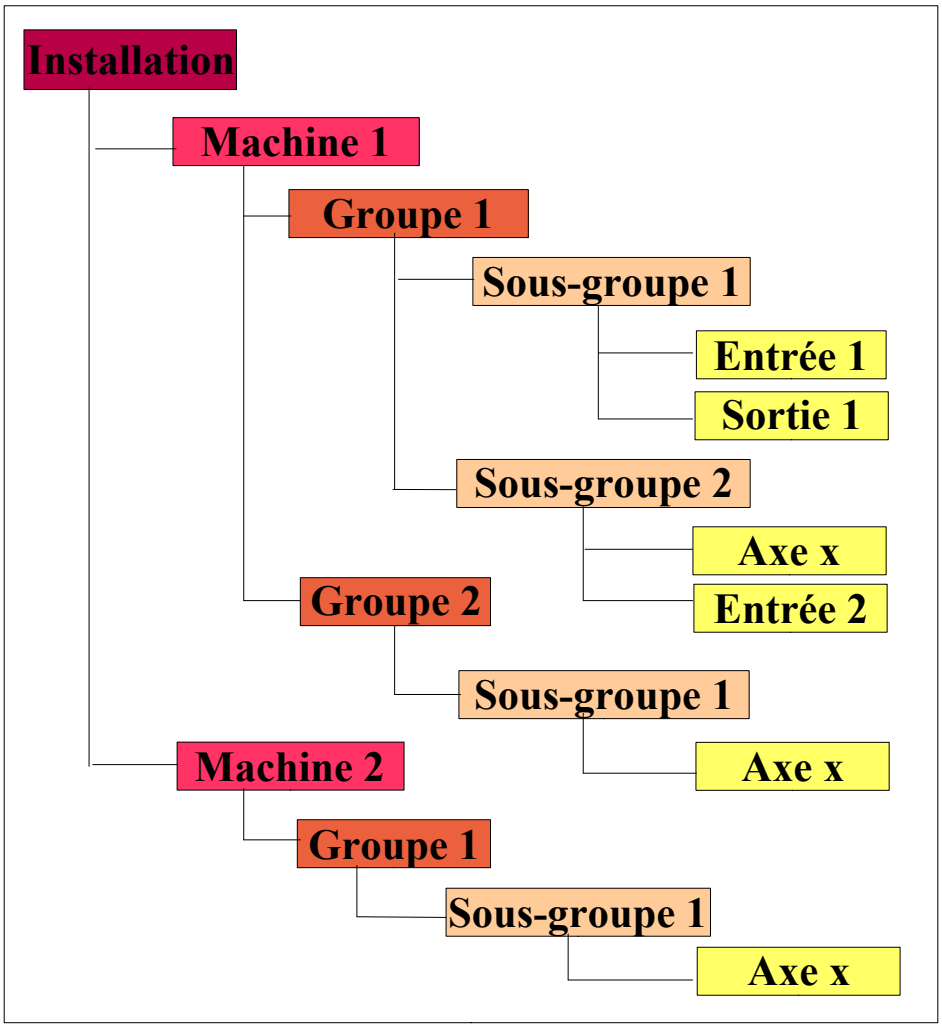
En suivant cette structure logique, dans le cas le plus général et le plus complexe, on obtient le niveau hiérarchique supérieur qui se compose de :

1. **Installation** Il s'agit d'un ensemble de machines. Elle constitue l'ensemble des éléments opérationnels gérés par la Commande Numérique. L'installation est toujours présente, même si elle ne se compose que d'une seule machine, et elle ne doit pas être définie de façon explicite.
2. **Machine** Elle est considérée d'un point de vue "logique" comme un ensemble de dispositifs (axes, temporisateurs, etc.) et de contrôles cycliques, c'est-à-dire d'un code écrit en GPL qui implémente les algorithmes de contrôle de la machine en question. Généralement, une machine contient un grand nombre de dispositifs qui sont organisés en groupes.
3. **Groupes** Ce sont des "conteneurs" qui permettent d'organiser les composants d'une machine selon des critères logiques. Par exemple, il est possible de définir un

groupe "axes" qui contient tous les axes de la machine, les commutateurs de fin de course, les contrôles cycliques qui exécutent le zérotage des axes, etc.

4. **Sous-groupes** Ils constituent une spécialisation ultérieure d'un groupe. Par exemple, le groupe "axes" pourrait être partagé en "axes-numériques" et en "axes-pas-pas".
5. **Dispositifs** Il s'agit du niveau le plus bas de la hiérarchie. Ils constituent une représentation logique des composants électriques et mécaniques de la machine et ils sont indépendants du matériel qui les soutient.

La figure suivante schématise la structure d'une installation imaginaire se composant de deux machines :



Exemple d'organisation hiérarchique d'un système.

REMARQUE : Les groupes peuvent aussi ne pas être partagés ultérieurement en sous-groupes et se composer directement de Dispositifs.

Pour accéder à certaines fonctions comme Diagnostic, Configuration de système, Paramétriques, en cas d'installation à plusieurs machines, il est nécessaire de sélectionner la machine dont on désire visualiser les données.

2.5 Les Dispositifs

Les dispositifs se regroupent en deux catégories : dispositifs physiques et dispositifs logiques. Dans le système, tous les dispositifs se repèrent à l'aide d'un nom qui identifie leur utilisation.

Dispositifs physiques

Les dispositifs physiques sont tous les éléments qui agissent sur les parties électriques ou pneumatiques de la machine ou qui en relèvent l'état ; il s'agit de :

Symbole	Dispositif	Fonction
	*Entrée numérique*	relève l'état "allumé" ou "éteint" d'un dispositif. Par exemple, l'interrupteur de sécurité d'une porte.
	*Sortie numérique*	active ou désactive un dispositif, en le mettant en état "allumé" ou "éteint". Elle peut être utilisée, par exemple, pour piloter une électrovanne
	*Entrée analogique*	relève la valeur d'une tension à l'entrée de la borne correspondante. Par exemple, la valeur générée par une dynamo tachymétrique.
	*Sortie analogique*	impose une tension à la sortie de la borne correspondante. Par exemple, elle peut être utilisée pour piloter un inverseur.
	*Porte d'entrées*	se compose de 8 lignes d'entrée numériques.
	*Porte de sorties*	se compose de 8 lignes de sortie numériques.
	*Axe*	gère le mouvement d'un axe électrique. Il est possible de gérer des axes de différents types : avec commande analogique, avec commande numérique, moteurs pas-pas, axes de comptage (uniquement lecture codeur).

Dispositifs logiques

Les dispositifs logiques sont des éléments qui agissent exclusivement au sein des programmes de travail et ils n'ont pas de contre-partie physique :

Symbole	Dispositif	Fonction
	*Minuterie*	instrument de mesure du temps. L'unité de mesure est la seconde. La résolution est de 4 ms. Elle ne peut prendre que des valeurs positives et la valeur maximale représentable est de 8.858.934 secondes (environ 99 jours) (avec real-time à 250 Hz). Sa valeur est enregistrée dans la mémoire non volatile de la carte axes.
	*Compteur*	élément comptant les opérations. Il peut prendre une valeur comprise entre -2.147.483.648 et +2.147.483.647. Sa valeur est enregistrée dans la mémoire non volatile de la carte axes.
	*Flag bit*	élément qui peut prendre la valeur "allumé" ou "éteint".
	*Flag switch*	ce sont des flags particuliers qui peuvent être associés à certains boutons de la barre d'outils, comme par exemple l'indicateur de Start.
	*Port d'indicateurs*	se compose de 8 lignes d'indicateurs
	*Variable*	variable globale type *integer* du code GPL.
	*Variable*	variable globale type *char* du code GPL.
	*Variable*	variable globale type *float* du code GPL.
	*Variable*	variable globale type *double* du code GPL.
	*Variable*	variable globale type *chaîne* du code GPL.
	*Variable*	variable globale type *array* du code GPL.
	*Variable*	variable globale type *matrice* du code GPL.

3 Tableau Synoptique

3.1 Utilisation du Tableau synoptique

Au cours de la phase d'exécution de la machine, il est possible d'ouvrir la fenêtre du *Synoptique* qui permet de visualiser l'état des dispositifs les plus importants.

Les informations qui se présentent dans les tableaux synoptiques sont les mêmes que celles de la fenêtre Diagnostic. Toutefois, alors que cette dernière présente les informations dans une structure arborescente (qui inclut tous les dispositifs de la machine), les synoptiques permettent de présenter les informations sous une forme graphique (affichant, par exemple, une image de la machine et présentant les cotes des axes à proximité de ces derniers). Ils permettent également de sélectionner les informations les plus importantes et de regrouper les autres dans des pages-écrans secondaires que l'utilisateur peut rappeler à sa guise.

3.2 Comment intervenir sur le tableau synoptique

A titre diagnostique, l'opérateur peut sélectionner les différentes pages qui composent le synoptique, en *double-cliquant* avec la souris sur l'une des zones de la machine délimitées dans l'image par un rectangle en pointillés et qu'on appelle également les "zones chaudes".

Pour pointer une "zone chaude", un dispositif ou un axe, il suffit de se positionner sur l'image à l'aide de la souris et de se déplacer ainsi jusqu'à l'objet graphique désiré. Tandis que l'on bouge la souris, la barre d'état fait apparaître le nom du dispositif sur lequel le pointeur de la souris est en train de passer.

Le pointeur de la souris prend des formes différentes selon le type d'objet qui est pointé, indiquant ainsi l'action qui est possible sur cet objet. Ces formes sont les suivantes :

 *loupe*	s'il s'agit d'une "zone chaude"
 *main*	s'il s'agit d'un dispositif de sortie
 *curseur de texte*	s'il s'agit d'une case où il est possible d'introduire une valeur

3.3 Comment intervenir sur les dispositifs

L'intervention sur les dispositifs est effectuée en pointant le dispositif désiré avec la souris, en complétant l'action comme il est indiqué ci-dessous ; bien entendu, (cette dernière varie en fonction du type de dispositif).

Mode de représentation	**Action**	**Dispositif**
Icône dispositif	pointer et cliquer	Sortie numérique Flag switch Flag bit
Case d'introduction valeur	pointer et cliquer et introduire la valeur	Sortie analogique Port de sortie Port d'indicateurs Cote axe Temporisateur Compteur

3.4 Mouvement des axes en Manuel

Pour accéder à la fonction de mouvement manuel des axes, il est nécessaire de posséder les droits d'accès. Ces droits sont attribués par le Constructeur de la machine.

Pour agir sur l'un des axes, il suffit de *double-cliquer* avec la souris au niveau du champ d'affichage de la cote de l'axe désiré. On obtient ainsi l'ouverture de la fenêtre du mouvement de l'axe. Pour les axes de type Virtuel, Pas-à-Pas et Comptage, la fenêtre présente une quantité réduite d'informations. Par exemple, si l'axe est de type Comptage, les seules données affichées sont la Cote Réelle et la Vitesse.

La fenêtre se compose de deux zones qui contiennent :

Zone de visualisation

- Trois cases présentant l'affichage de la *Cote absolue* de l'axe [mm], de sa *Vitesse Réelle* et de l'*Erreur de boucle* ou de poursuite.
- Deux boutons de sélection signalant l'*État* de l'axe (*Free* = à boucle ouverte, par ex. à cause d'une erreur de système, *Normal* = à boucle fermée, c'est-à-dire dans l'état de contrôle de position normal). Ces boutons permettent également de choisir l'état.
- Le signal, pendant le mouvement, de l'*État* de l'axe (ex. Accélération).
- Deux boutons permettant d'effectuer le déplacement de l'axe dans la *Direction* négative ⊟ ou positive ⊞.
- Le bouton ■**,** pour l'Arrêt à tout moment, du mouvement de l'axe au cours des déplacements en mode Absolu ou Step.

Zone Mouvement

- Deux cases pour insérer une *Cote Négative* et une *Cote Positive* qui seront exécutées en mode Absolu.
- Une case pour insérer la *Vitesse* à donner à l'axe pendant les mouvements en manuel.
- Trois boutons de sélection du mode avec lequel sera effectué le mouvement, à choisir entre : *Jog*, à cote *Absolue* ou *Step*.
- Une case permettant d'insérer la valeur du *Pas* à utiliser dans le mode *Step*.

Pour déplacer un axe, il est nécessaire de régler convenablement les paramètres décrits. Il faut choisir le mode de déplacement et appuyer sur le bouton ⊞ (pour déplacer l'axe dans la direction positive) ou sur le bouton ⊟ (pour déplacer l'axe dans la direction négative).
En mode *Jog*, l'axe se déplace aussi longtemps que le bouton ⊞ ou sur le bouton ⊟ reste appuyé.
En mode *Step*, l'axe se déplace de longueur indiquée dans la case "pas" chaque fois que l'on appuie sur le bouton ⊞ ou sur le bouton ⊟.
En mode *Absolu*, l'axe atteint directement la cote programmée dans la case Cote positive ou Cote négative.

A la place des boutons ⊞ , ⊟ et ■ il est possible d'utiliser les touches "+" (ou Ctrl+P), "-" (ou Ctrl+M) et la barre d'espacement du clavier.

4 Paramètres Technologiques et Outils

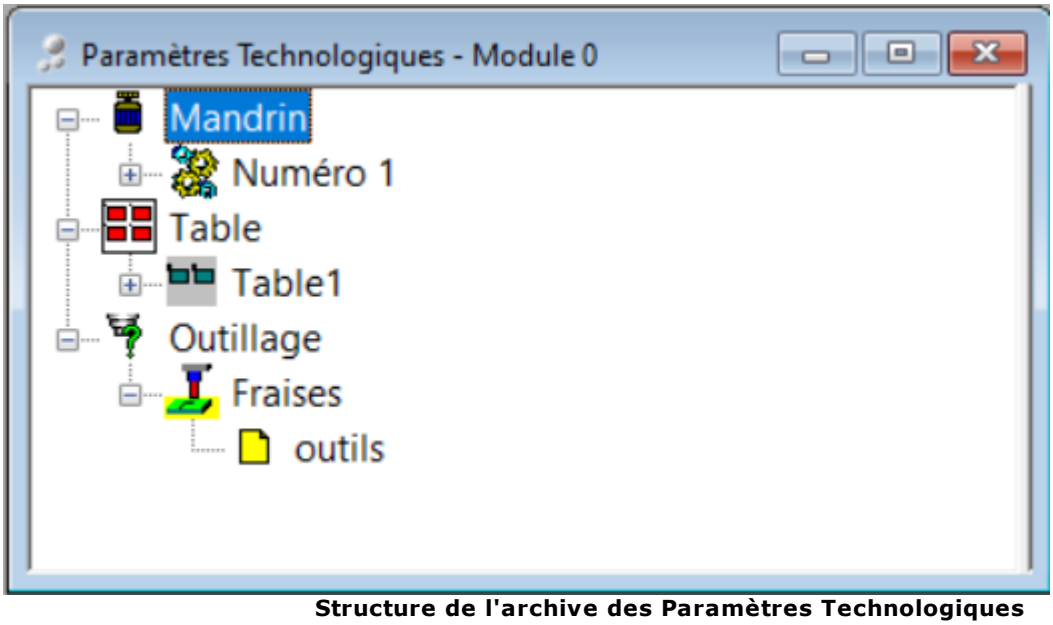
4.1 La fenêtre Paramètres Technologiques

Les archives des Paramètres Technologiques permettent d'enregistrer les informations de nature géométrique et technologique d'une machine. Ces informations sont nécessaires à la commande numérique pour pouvoir gérer correctement le fonctionnement de la machine.
Cette fenêtre s'ouvre dans le menu **Fichier->Ouvrir Paramètres Technologiques.**

Habituellement, les Paramètres Technologiques de la machine sont organisés en Groupes / Sous-groupes (en général, les groupes et les sous-groupes des Paramètres Technologiques ne sont pas liés aux groupes ou aux sous-groupes dans lesquels sont organisés les dispositifs de la machine). Les modes d'affichage sont établis par le Constructeur de la machine et ils diffèrent en fonction des applications spécifiques.

Habituellement, les valeurs indiquées sont établies par le fabricant au cours de la mise au point de la machine ; normalement elles ne peuvent être modifiées par l'utilisateur final qu'exceptionnellement. Il est donc possible que certaines données soient protégées par un mot de passe, de façon à éviter toute modification accidentelle qui pourrait nuire au bon fonctionnement du système.

La fenêtre inhérente aux Paramètres Technologiques présente dans une structure à arbre tous les Groupes et les Sous-groupes de paramètres qui composent la paramétrisation technologique et géométrique, comme il est indiqué dans la figure ci-dessous.



Structure de l'archive des Paramètres Technologiques

La fenêtre présente dans une structure à arbre les Groupes et leurs Sous-groupes de paramètres. La structure arborescente peut être étendue ou réduite à l'aide des boutons ⊞ et ⊟ de chaque nœud. L'ouverture et la fermeture des parties qui composent l'arbre peuvent également être effectuées à l'aide des touches : **+**, **-** , flèche directionnelle à droite et à gauche.

Comment intervenir sur les Paramètres Technologiques
Après avoir ouvert l'arbre du Groupe / Sous-groupe désiré, on accède à la fenêtre qui contient les données.
Les données peuvent apparaître sous la forme d'un tableau, ou de cases de texte ou de sélection ; cela dépend du type de données et de la manière avec laquelle elles ont été introduites par le fabricant.
Si l'on modifie des données, il est nécessaire d'appuyer sur le bouton **[OK]** pour que la modification devienne permanente.

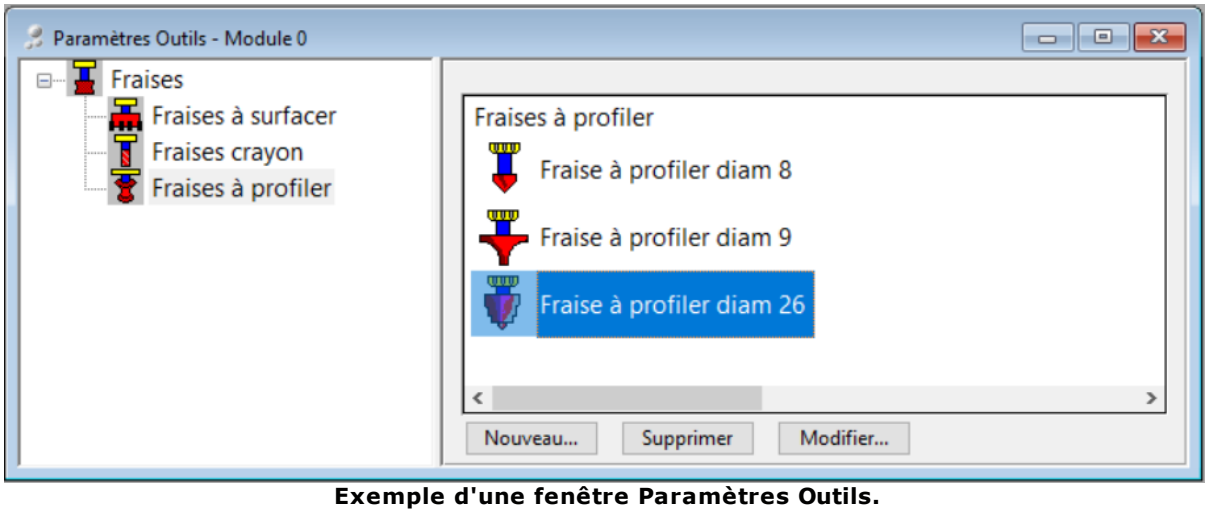
Outillage
Un cas particulier parmi les données relatives à une machine est celui des outillages. Traditionnellement, les informations relatives à l'ensemble des outils dont la machine est équipée (outillage) sont

enregistrées dans les archives des Paramètres Technologiques. En revanche, les informations relatives aux outils sont enregistrées dans les archives des Paramètres Outils.
Pour définir l'outillage de la machine, il est donc nécessaire de lier les informations des deux archives. Si l'application le prévoit, il est donc possible de rappeler, des archives des Paramètres Technologiques, des informations des archives des Paramètres Outils. Habituellement, le lien est implémenté au moyen d'un bouton présentant une icône semblable à la suivante.

Si l'on sélectionne l'icône et que l'on double-clique avec la touche gauche de la souris, on fait apparaître une fenêtre qui contient la liste des outils définis dans les archives des Paramètres Outils et qui permet de sélectionner l'outil désiré. La sélection étant effectuée, l'icône du bouton change pour devenir celle qui est associée à un outil spécifique.
Il est également possible de visualiser les données relatives à l'outil en double-cliquant sur l'icône avec la touche droite de la souris.

4.2 La fenêtre Paramètres Outils

La fenêtre des Paramètres Outils s'ouvre dans le menu **Fichier->Ouvrir Paramètres Outils.** Les paramétrisations outils, établies par le fabricant en fonction des usinages que la machine est en mesure d'exécuter, sont organisées comme il est indiqué dans la figure suivante :



Exemple d'une fenêtre Paramètres Outils.

La fenêtre Paramètres Outils se compose de deux zones :

- La *zone de gauche* présente dans une structure à arbre les Groupes et leurs Sous-groupes d'outils. La structure arborescente peut être étendue ou réduite à l'aide des boutons ⊞ et ⊟ de chaque nœud. Par exemple, il est possible d'avoir un Groupe Fraises composé de Sous-groupes de fraises présentant des caractéristiques différentes, comme les fraises profilées, les fraises à pivot, etc. Chacun de ces sous-groupes est associé à un ou à plusieurs outils dont les caractéristiques sont assignées dans une boîte de dialogue prévue par le Constructeur. Les outils contenus par chaque sous-groupe sont affichés dans la zone de droite de l'écran.
- La *zone de droite*, dont le titre est le nom du Sous-groupe sélectionné, contient la liste des Outils faisant partie du Sous-groupe. Les outils présentés dans cette zone ne sont pas obligatoirement montés sur la machine. L'association entre l'outil et la position qu'il occupe sur la machine (outillage) est normalement faite dans les archives des paramètres technologiques.

Comment intervenir sur les Paramètres Outils
Les opérations d'édition peuvent être exécutées à l'aide des *boutons* qui se trouvent dans la partie inférieure de la fenêtre :

[Nouveau...] Cette touche permet d'insérer un nouvel outil dans le Sous-groupe. Elle ouvre la boîte de dialogue "Nouvel outil" dans laquelle il est possible d'assigner les données suivantes :
- *Description :* il s'agit d'un message qui identifie l'outil. La description peut être choisie parmi celles qui sont déjà dans la liste, à condition qu'elle ne soit pas

encore attribuée à un autre outil ; il est également possible d'en définir une nouvelle.

- *Image :* il s'agit d'une icône qui identifie l'outil. Elle peut être choisie parmi celles qui sont déjà dans la liste ou être rappelée d'un répertoire en utilisant le bouton **[Image].** L'outil est inséré en respectant l'ordre alphabétique des descriptions.

[Supprimer] Cette touche permet d'éliminer un outil du Sous-groupe, après confirmation ; la description relative n'est pas éliminée et elle reste disponible pour un autre outil.

[Modifier...] Cette touche permet de remplacer la *description* ou l'*image* de l'outil sélectionné, En effet, la même fenêtre que celle qui a été décrite dans la commande **[Nouveau...]** apparaît.

5 Diagnostic

5.1 La fenêtre Diagnostic

Lorsque la machine est en marche, il est possible d'ouvrir la fenêtre *Diagnostic* qui permet à l'opérateur de contrôler l'état de fonctionnement de la machine, en surveillant l'état logique des signaux de I/O numériques, la valeur des I/O analogiques, celle des compteurs et des temporisateurs et l'éventuel mouvement d'un axe.
En fonction des droits d'accès assignés par le Constructeur, il est également possible de modifier l'état des dispositifs.

En temps réel, si le niveau d'accès le permet, il est possible d'avoir :

- la visualisation de l'état ON/OFF de tous les signaux d'entrée numériques ;
- la possibilité d'activer et de désactiver les signaux de sortie numériques ;
- l'affichage des valeurs courantes (dans l'intervalle +/-10V) des entrées analogiques ;
- la possibilité de forcer une valeur (dans l'intervalle +/-10V) pour toutes les sorties analogiques ;
- la possibilité de déplacer un axe en Manuel en sélectionnant sa vitesse, la valeur du pas ou la cote absolue finale, avec affichage de la position réelle, de la vitesse et de l'erreur de boucle ;
- affichage et modification des variables globales.

Les prochains paragraphes décrivent de façon détaillée les dispositifs ainsi que les variables globales et leur représentation graphique.

REMARQUE : Dans la fenêtre Diagnostic, seuls les dispositifs inhérents au niveau d'accès actuel sont affichés.

5.2 Composition de la fenêtre Diagnostic

En référence à la structure "Groupes / Sous-groupes" décrite dans le chapitre Organisation et configuration logique, il est possible d'accéder aux dispositifs qui sont ensuite présentés dans une structure à arbre.
Au sommet de la structure, on trouve le groupe représenté sous la forme de l'icône

 , suivie du nom et du commentaire de la Machine ou Groupe

La structure est agrandie ou rapetissée en cliquant sur le bouton ⊞ ou ⊟. L'ouverture et la fermeture des parties qui composent l'arbre peuvent également être effectuées à l'aide des touches : **+**, **-**, touche de direction droite et gauche.

Lorsqu'on ouvre un Groupe, l'arbre fait apparaître

- La "Liste des Dispositifs" du Groupe, représentée par le symbole 
- Les éventuels Sous-groupes qui composent le Groupe.

Lorsque l'on ouvre un des Sous-groupes, les dispositifs qui le composent apparaissent.

5.3 Représentation des Dispositifs

Pour chaque dispositif affiché, les informations indiquées ci-dessous sont montrées :

- le Symbole graphique ;
- l'État ou la valeur courantes ;
- le Nom ;
- le Commentaire.

On trouvera ci-dessous la représentation graphique des dispositifs, le type de dispositif et ce qui en est affiché en temps réel.

L'état des entrées et des sorties numériques est représenté graphiquement par une diode qui prend une certaine couleur selon que l'entrée est activée ou désactivée.

En cas de Ports, c'est-à-dire lorsque plusieurs lignes sont représentées en même temps (8), une rangée de diodes apparaît ; la première ligne du groupe est représentée par la diode la plus à droite et la dernière est la plus à gauche.

Dispositif	Symbole	Etat	Visualisation en temps réel
Entrée numérique			état: Active = VERTE, Désactivée = GRISE
Sortie numérique			état: Active = ROUGE, Désactivée = GRISE
Entrée analogique		22.000	valeur courante
Sortie analogique		22.000	valeur numérique courante en volts
Port d'entrées			état de chaque ligne (comme Entrée numérique). état: Active = VERTE, Désactivée GRISE
Port de sorties			état de chaque ligne (comme Sortie numérique). état: Active = ROUGE, Désactivée = GRISE
Axe		100.000	position absolue courante
Temporisateur		12.000	valeur courante en secondes
Compteur		58	valeur numérique courante
Flag bit			état: Actif = JAUNE, Désactivé = GRISE
Flag switch			état (comme Flag bit). état: Actif = JAUNE, Désactivé = GRISE
Port de flag			état de chaque ligne (comme Flag bit). état: Actif = JAUNE, Désactivé = GRISE
Variable globale		2	variable globale type Integer du code GPL
Variable globale		127	variable globale type Char du code GPL
Variable globale		50.00000000	variable globale type Float du code GPL
Variable globale		200.0000000	variable globale type Double du code GPL
Variable globale		Area	variable globale type Chaîne du code GPL
Variable globale		[256]	variable globale type Array du code GPL
Variable globale		[10][3]	variable globale type Matrice du code GPL

5.4 Interagir sur les Dispositifs

A titre diagnostique, il est possible d'interagir sur un dispositif pour en lire l'état ou pour en modifier la valeur.
Cela est impossible pour certains types de dispositifs, comme ceux d'entrée et ceux qui sont protégés par le fabricant. Dans ce cas, un message informe l'opérateur s'il tente d'accomplir des opérations sur le dispositif.

Le dispositif étant sélectionné, double-cliquer avec la souris, appuyer sur la touche **Entrée** ou sur la **Barre d'espacement**, pour accéder à la fenêtre qui permet de modifier l'état ou la valeur du dispositif.

S'il s'agit d'une **Sortie numérique** ou d'un **Flag bit**, aucune fenêtre n'apparaît, mais l'état du dispositif est modifié automatiquement. Le bon fonctionnement de la sortie est signalé par le changement de couleur de la diode qui en indique l'état.

En cas de **Port de Sorties**, pointer la souris sur la diode correspondant à la sortie désirée et double-cliquer pour effectuer la modification de l'état.
Il en va de même pour les **Flag switches** et pour le **Port de flag**.

Pour les **Sorties analogiques**, les **Temporisateurs** et les **Compteurs**, une Fenêtre de dialogue apparaît, qui visualise la valeur courante et qui permets l'introduction de la nouvelle valeur que l'on désire donner immédiatement au dispositif.

Le mode d'interaction avec un **Axe** est décrit dans le paragraphe Personnalisation mouvement axes.

5.5 Liste des touches pour naviguer à l'intérieur d'une arborescence

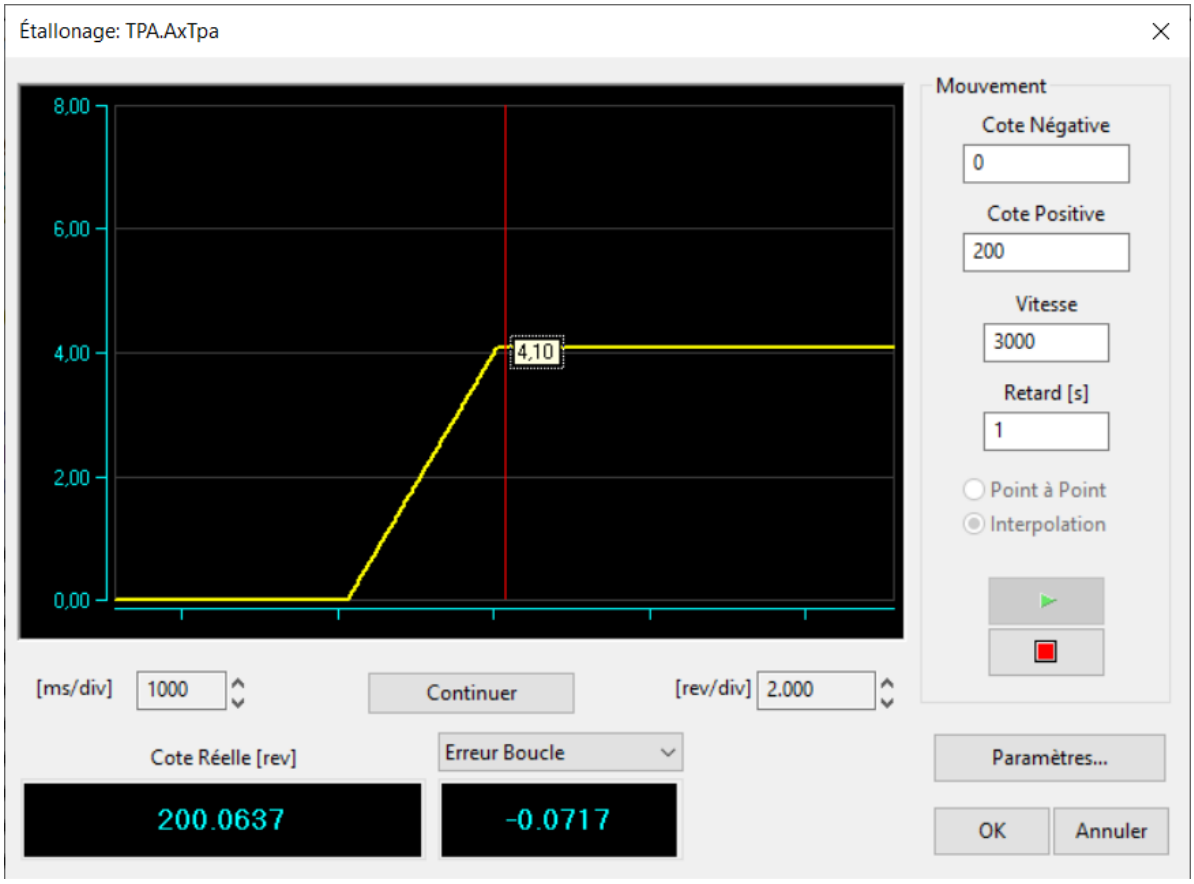
Touche	Description
Flèche en haut Flèche en bas	déplace la sélection jusqu'à la ligne immédiatement précédente ou suivante
Flèche droite	développe d'un niveau la branche sélectionnée et si déjà développée elle déplace le sélection dans la branche suivante
Flèche gauche	réduit la branche sélectionnée et si déjà réduite elle déplace la sélection dans la branche précédente
+	développe d'un niveau la branche sélectionnée
-	réduit la branche sélectionnée
*	développe tous les niveaux de la branche sélectionnée
Ctrl+Alt+Maj et Entrée Ctrl+Alt+Maj et Clic gauche	affiche les tableaux des correcteurs de linéarités associées à un axe. Si la combinaison de touches est activée, lorsqu' un axe dans l'arbre des dispositifs d'un module est sélectionné, tous les correcteurs associés à un axe s'affichent, comme s'ils étaient une matrice où les colonnes sont les axes associés (la première colonne est celle des correcteurs automatiques) et les lignes sont les valeurs de correction. Les valeurs éventuellement modifiés sont pris en charge pendant le mouvement de l'axe, mais pas stockées dans le disque.

5.6 Correcteurs de linéarité

En diagnostic, vous pouvez afficher et modifier les correcteurs de linéarité de l'axe sélectionné en ouvrant le menu contextuel et en sélectionnant **Correcteurs de linéarité**. Cette option n'est visible que si des correcteurs de linéarité ont été définis pour l'axe dans la configuration des correcteurs de linéarité. Comme alternative au menu, vous pouvez utiliser la combinaison de touches **[Ctrl+Shift+Entrée].**

5.7 Console d'étalonnage des axes

La console d'étalonnage des axes est un instrument qui permet de modifier les paramètres d'un axe, tout en déplaçant l'axe et en visualisant son comportement sur un oscilloscope virtuel.
Pour accéder à la console d'étalonnage, il faut disposer d'un niveau d'accès au système supérieur ou égal à "Constructeur". On accède à la console d'étalonnage à partir du mode diagnostique ou manuel, en double-cliquant sur l'axe que l'on veut calibrer et en maintenant la touche **[Maj]** ou à partir du menu contextuel en sélectionnant l'entrée "Étalonnage" enfoncée.
On accède ainsi à la console d'étalonnage qui est représentée sur la figure suivante :



Pour contrôler le comportement de l'axe en cas de variation des paramètres, le faire bouger de façon continue entre deux cotes limites intitulées **Cote Positive** et **Cote Négative**. Au-delà de ces paramètres, il est également nécessaire de régler la **Vitesse** de déplacement de l'axe. Au cours des premières phases de l'étalonnage, il est conseillé d'utiliser une vitesse basse. Il est également possible de programmer un **Retard** à appliquer entre un mouvement et l'autre.

La fenêtre de l'oscilloscope affiche le tracé de l'erreur de boucle ou de l'une des autres grandeurs de l'axe. Comme pour les oscilloscopes de table, il est possible de mettre le graphique à l'échelle et de l'adapter à la dimension de la fenêtre ou de mettre en évidence certains de ses détails.
Par l'intermédiaire de la souris et des touches contrôle il est possible de revoir la dernière minute d'étalonnage, d'afficher un ou deux curseurs pour mesurer et vérifier la donnée échantillonnée, agrandir une surface de la graphique pour analyser des détailles de la donnée échantillonnée, varier l'offset et l'échelle soit dans les abscisse soit dans les ordonnées.
En appuyant sur le bouton **Arrêter**, il est possible de bloquer le défilement du graphique pour l'observer tranquillement sans devoir arrêter l'axe.
En plus du graphique, on trouve deux cases qui présentent la cote réelle (à gauche) et la grandeur sur le graphique (à droite). Cette dernière peut également être réglée dans la combo-box qui se trouve au-dessus de la case d'affichage.

Pour modifier le paramétrage de l'axe, appuyez sur le bouton **[Paramètres...]**. Celui-ci active une fenêtre qui permet de modifier immédiatement la plupart des paramètres de l'axe. Après avoir effectué une modification sur un ou sur plusieurs paramètres, il est possible de la faire devenir réelle en appuyant sur le bouton **[Appliquer]**. Ce bouton permet de voir immédiatement l'effet du changement dans le comportement dynamique de l'axe.
Si vous appuyez sur **[Ok]** les modifications restent enregistrées; en appuyant sur **[Annuler]** les valeurs présentées avant d'ouvrir la fenêtre **[Paramètres...]** sont rétablies.

Les paramètres sur lesquels il est généralement nécessaire d'intervenir sont les suivants :

- Coefficient **Proportionnel**
- Coefficient **Intégratif**
- Coefficient **Dérivatif**
- **Feed Forward** : taux de la vitesse courante, fourni directement à l'actionnement (quelle que soit l'erreur de boucle)

- **Feed Forward Accél.** : taux de référence de vitesse, fourni directement à l'actionnement pendant les phases d'accélération et de décélération de l'axe (en plus du Feed Forward)
- **Accélération** : durée de la rampe d'accélération
- **Décélération** : durée de la rampe de décélération

Le curseur dans l'aire graphique

Le curseur est un outil qui permet la mesure et l'affichage de certaines données de la trace. Il consiste en une ligne vertical de la couleur associée à la trace sélectionnée. Il peut se déplacer à l'intérieure de la graphique à l'aide des boutons curseur ou de la souris. Dans une info-bulle liée à la ligne verticale on peut afficher des valeurs différentes, qui peuvent être sélectionnées d'un menu contextuel, qui peut être appelé en cliquant avec le bouton droit de la souris prés du curseur.

Dans le menu contextuel on peut choisir les options suivantes :

- **Canal** : présente la liste des traces affichées dans le graphique et, en cochant une case, il marque celle à laquelle le curseur se réfère. En outre, il est possible de choisir une nouvelle trace à associer au curseur.
- **Style** : présente une liste de données qui peuvent être affichées dans le rectangle de l'info-bulle.
- **Valeur X-Y** : affiche la valeur sur l'axe x et sur l'axe y du point de la trace où le curseur est placé.
- **Valeur X** : affiche la valeur sur l'axe x du point de la trace où le curseur est placé.
- **Valeur Y** : affiche la valeur sur l'axe y du point de la trace où le curseur est placé.
- **Période** : affiche la valeur de la distance entre les deux curseurs affichés, lorsqu'on mesure le long de l'axe X.
- **Pic**-**pic** : affiche la valeur de la distance entre deux curseurs visualisés, lorsque on mesure la distance entre deux points le long de l'axe Y.
- **Fréquence** : affiche la valeur inverse de la distance le long de l'axe X.
- **Options :** configure la façon d'affichage du curseur et de l'info-bulle associée.
- **Utiliser la couleur du canal :** si activée, le curseur est dessiné selon la couleur de la donnée échantillonnée à laquelle il se réfère, sinon on choisit la couleur d'affichage par hasard.
- **Cacher la suggestion au moment de la libération** : si cette commande est activée, l'info-bulle est affichée uniquement jusqu'à le bouton gauche de la souris est appuyé; ensuite elle est cachée.
- **Inverser l'alignement** : normalement l'info-bulle est affichée à droite du curseur. Si l'option est activée, le curseur est affiché à gauche du curseur.
- **Positionnement à l'échantillon :** s'il est activé, il ne positionne le curseur que sur les valeurs échantillonnées,

Étalonnage de l'axe
L'étalonnage d'un axe est une opération délicate qui doit donc être exécutée avec la plus grande attention et la plus grande prudence.

Par l'option CalibSampleTime dans la section [Albatros] en Tpa.ini il est possible de modifier la période d'échantillonnage des données des axes pour la fenêtre de Étalonnage. La valeur est exprimée en millisecondes et elle ne peut pas être inférieure à la fréquence de contrôle des axes et supérieure à 100.

Avant d'effectuer l'étalonnage de l'axe sur la console, il est nécessaire de régler tous les paramètres dans la configuration et d'indiquer la valeur de fin d'échelle de la vitesse de l'actionnement. La valeur de tension à laquelle correspond la vitesse maximale dans Albatros est de 9 volts.

Pour ne pas risquer d'endommager la machine en utilisant des paramètres erronés, il convient de programmer une vitesse peu élevée, par exemple 10% de la vitesse maximale prévue pour l'axe. De cette manière, même si l'on programme un gain trop élevé, l'on n'aura pas de réactions trop élevées de l'axe.

En général, on effectue d'abord l'étalonnage pour les mouvements point à point, puis l'étalonnage pour les mouvements interpolés.

La première opération (si elle n'a pas été préalablement faite dans la configuration) consiste à régler les temps d'accélération et de décélération. Plus ces temps sont élevés, plus faible sera l'accélération à laquelle l'axe est soumis.

Ensuite, régler un gain minimal permettant de déplacer l'axe. Cela permet de s'assurer que l'étalonnage de l'actionnement est correct. En effet, Albatros est configuré pour fournir une référence de 9 volts au niveau de la vitesse maximale programmée lors de la configuration de l'axe. Par exemple, si l'on déplace l'axe à une vitesse égale à 10% de la vitesse maximale et si l'actionnement est calibré correctement, la tension de référence affichée correspond à 10% de la tension maximale, c'est-à-dire 0,9 volt. Si l'on

n'obtient pas cette tension de référence, il est nécessaire de modifier la valeur de fin d'échelle de l'actionnement.

Après avoir calibré l'actionnement, commencer à élever peu à peu, et toujours avec la plus grande prudence, le gain du boucle de position. A chaque fois, s'assurer qu'il n'y a aucun phénomène de sur-oscillation ou d'instabilité. Au cours de cette phase, la vitesse doit toujours rester inférieure ou égale à 10% de la vitesse maximale. Au cours de cette phase, il convient toujours d'analyser méticuleusement le profil de la vitesse obtenue avec l'oscilloscope virtuel, en agrandissant le plus possible l'image de façon à en voir les détails.

Lorsque l'axe a un comportement à la fois prêt et stable, il est possible d'augmenter peu à peu la vitesse de déplacement. Il faut alors contrôler comment il se comporte et, selon les cas, ajuster le gain si ce dernier n'est pas satisfaisant. Le gain et la vitesse ne devront jamais être élevés brusquement. Des conditions de étalonnage apparemment stables à petite vitesse risquent de ne pas l'être à des vitesses supérieures.

Après avoir établi la valeur optimale du Gain, il est possible, si besoin est, d'élever progressivement les coefficients Intégratif et Dérivatif, puis le Feed Forward, pour réduire l'erreur de boucle et lui donner des valeurs acceptables pour ce qui est de la précision de l'axe. Le Feed Forward permet d'annuler presque complètement l'erreur de boucle pendant la phase de régime du mouvement, mais non pendant les phases d'accélération et de décélération. Pour réduire encore davantage l'erreur de boucle pendant ces phases, il est possible d'élever le Feed Forward d'Accélération. En général, des valeurs très basses données à ce paramètre suffisent pour garantir un bon résultat.

Pour ce qui est de l'étalonnage d'un axe pour les mouvements interpolés, il est possible d'utiliser les valeurs déterminées précédemment pour les mouvements point à point. Dans ce cas, il faut toutefois tenir compte de la présence des autres axes de la machine. En particulier, pour obtenir une bonne précision pendant les mouvements interpolés, il est nécessaire d'équilibrer les erreurs de boucle des axes. Ainsi, il est nécessaire de déterminer l'axe avec la plus grande erreur de boucle (à vitesse identique) et d'"empirer" (uniquement pour les paramètres d'interpolation) l'étalonnage des autres, de façon à ce que les erreurs de boucle soient identiques.

6 Erreurs et Signalisations

6.1 Introduction

Albatros gère les événements et les rapports d'erreur.

Erreurs de système

Il s'agit des erreurs que le système Albatros est à même de relever automatiquement, aussi bien pendant les phases d'exécution des programmes, que pendant les opérations d'entretien et de diagnostic de l'installation.

Ces erreurs sont de différentes natures et elles vont des problèmes de gestion d'un axe aux problèmes pouvant surgir lors de l'exécution du programme.

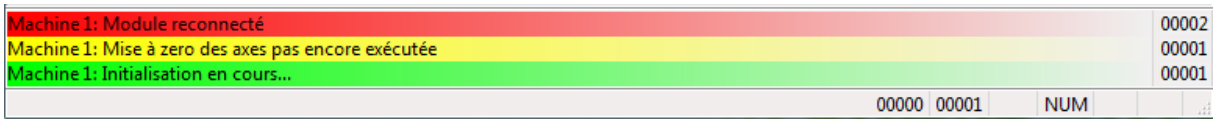
Les erreurs de système peuvent être gérées à l'intérieur des programmes de travail au moyen de l'instruction ONERRSYS. En cas contraire, l'exécution des programmes est interrompue sur le module où l'erreur a eu lieu. L'explication de chaque erreur système est décrite dans les pages suivantes de ce manuel.

Erreurs de cycle
Ce sont des erreurs qui se produisent pendant l'exécution du programme, mais qui lui permettent généralement de continuer après la suppression de l'erreur. Les erreurs de cycle peuvent être générées par l'instruction GPL ERROR.

Messages
Il s'agit de messages d'avertissement qui sont produits dans des situations particulières pendant l'exécution du programme, ou de signaux d'une demande d'intervention de l'opérateur, mais qui n'arrêtent pas l'exécution du programme. Les messages peuvent être générés par l'instruction GPL MESSAGE.

Signalisations
Il s'agit de signalisations génériques qui ne sont pas liées aux cycles de la machine et qui sont rédigées par Albatros. Elles ne s'affichent pas en Albatros. La description de chaque signalisation se trouve dans les pages suivantes de ce manuel.

La Barre d'erreur
Dans la barre d'erreur s'affiche le dernier erreur de système ayant lieu chronologiquement avec le dernier erreur de cycle et le dernier message.
Les **erreurs de système** sont signalées en rouge.
Les **erreurs de cycle** sont signalées en jaune. Il s'agit d'erreurs qui ont lieu pendant l'exécution du programme mais qui, généralement, lui permettent de continuer après avoir toutefois éliminé l'erreur.
Les **messages** sont signalés en vert.



Barre d'erreurs

Les erreurs qui se sont produites à partir du démarrage du système peuvent être affichées dans une fenêtre que l'on ouvre en double-cliquant sur la *Barre d'erreurs* ou du menu ***Affichage***. Cette fenêtre présente également des informations complémentaires sur les erreurs de système.

La fenêtre comprend les zones suivantes. La **partie supérieure** présente les informations suivantes :

- Type : identifie le type d'erreur (erreur de système, erreur di cycle et de message).
- Heure&Date : sont l'heure et la date où l'erreur a eu lieu.
- Description : est la description de l'erreur.
- Code : est le numéro du message d'erreur.
- Tâche : est le nom de la tâche qui a généré l'erreur (non présente dans la Barre d'erreurs).

En *double-cliquant* sur l'une de ces colonnes, les informations sont classées en fonction du contenu de la colonne.
Dans la **partie située dessous**, on trouve les cases :

- Erreurs de cycle : si la case est activée, elle montre également ces erreurs.
- Messages : si la case est activée, elle montre également les messages.
- Tous : si la case est activée, elle montre les messages de tous les modules du système, relatifs au type d'informations que l'on est en train de visualiser.

- Case nom module : elle montre le nom du module que l'on est en train de visualiser et elle permet également, en cas de système à plusieurs modules, de sélectionner le module dont on veut visualiser les informations.

Elle présente enfin les **boutons** de commande suivants :

- **[Supprimer tous]** : élimine de la mémoire toutes les informations visualisées, mais elle ne les efface pas des archives.
- **[Supprimer]** : permet d'éliminer de la mémoire l'information courante, sans l'effacer toutefois des archives.
- **[OK]** : permet de fermer la fenêtre.

Mémorisation des erreurs et des signalisations dans un fichier de rapport
Toutes les erreurs sont stockées dans un fichier pour leur reconstruction historique. Il s'agit d'un fichier de texte au format TSV. Le nom du fichier est MONTH (numéro du mois en cours).TER. La suite Albatros comprend un programme ViewRER, qui charge et affiche les fichiers .TER.

6.2 Erreurs de système

6.2.1 Erreurs générées par la gestion des axes

1 NomAxe : connexion codeur erronée

Cause :
L'axe étant arrêté, une différence a eu lieu entre la cote théorique et la cote réelle de l'axe, supérieure à 1024 pas de codeur.
En général, cela a lieu, pendant la mise en service de l'axe, lorsque les phases du codeur sont inversées. Pendant le fonctionnement normal, cette erreur a lieu quand, l'actionnement étant éteint, on déplace l'axe manuellement sans l'avoir mis préalablement en FREE ou quand, à cause d'un étalonnage incorrect, l'axe fait l'objet d'une suroscillation lors de l'arrivée à la cote (overshoot).
Après cette erreur, le signal de référence est réinitialisé et l'axe est mis en état FREE.

Solution :
Au cours de la mise en service de l'axe, contrôlez la connexion des phases du codeur de l'axe concerné (éventuellement, activer l'option Inversion Phases Codeur dans la configuration de l'axe).
Contrôlez l'étalonnage de l'axe à l'aide de la fonction prévue à cet effet en Diagnostic.

2 NomAxe : mouvement non terminé

Cause :
A la fin du déplacement, 5 secondes après la fin du mouvement théorique, la différence entre la cote théorique et la cote réelle de l'axe s'est avérée supérieure à la fenêtre indiquée dans Configuration. Cela peut, tout simplement, être dû au fait que l'actionnement est éteint ou qu'il n'est pas activé, ou encore à un réglage erroné de l'offset de l'actionnement. Cela peut cependant également dépendre de jeux mécaniques présents sur l'axe ou d'un gain de la bague de position de l'axe trop bas.

Solution :
Assurez-vous que l'actionnement est allumé et activé. Contrôlez l'étallonage de l'axe et régler l'offset de l'actionnement de l'axe concerné.

3 NomAxe : servoerror

Cause :
Pendant n'importe quel type de mouvement, la différence entre la cote théorique et la cote réelle de l'axe a dépassé l'erreur maximale indiquée dans Configuration ou celle qui est introduite dans l'instruction SETMAXER.
Normalement, cela est dû à un mauvais réglage du gain de la bague de position ou de la pleine-échelle de vitesse de l'actionnement ou à une inertie excessive de l'axe.

Solution :
Contrôlez le réglage du gain et la pleine-échelle de vitesse de l'actionnement.
Assurez-vous que le codeur et le groupe moteur/actionnement fonctionnent correctement.
Assurez-vous également qu'il n'y a pas de blocages mécaniques.

4 NomAxe : hors limite positive

Cause :
La cote théorique de l'axe a dépassé la cote limite positive introduite dans la Configuration ou introduite dans l'instruction SETLIMPOS.

Solution :
Corrigez dans le programme la cote qui dépasse la cote limite positive.

5 NomAxe : hors limite négative

Cause :
La cote théorique de l'axe a dépassé la cote limite négative introduite dans la Configuration ou introduite dans l'instruction SETLIMNEG.

Solution :
Corrigez dans le programme la cote qui dépasse la cote limite négative ou redéfinir les limites de cote de l'axe.

10 NomAxe : L'exécution en Temps Réel est plus vite que la construction du profil

Cause :
L'exécution en real-time du profil de mouvement est plus rapide que la génération GPL du profil lui-même. Le lookahead est vidangé plus vite que son remplissage. L'erreur peut être le résultat de deux causes généralement concomitantes :
- la vitesse d' interpolation est trop élevée par rapport aux dimensions des traits à parcourir
- les traits à parcourir sont trop courts.

Solution :
Vérifiez que la vitesse d'interpolation affichée n'est pas trop élevée par rapport aux dimensions des traits à parcourir et vérifiez que le traits d'interpolations à parcourir ne sont pas trop courts.

6.2.2 Erreurs générées par la gestion I/O à distance

2049 Récepteur numéro : configuration erronée

Cause :
Le récepteur à distance a reçu une configuration d'extension I/O différente de celle qui est relevée sur le champ. Cela se peut produire, si le contrôle à distance n'est pas égal à celui qui a été choisi dans la configuration de Albatros. Par exemple, le récepteur à distance est un Albre16 et en Albatros on a configuré un contrôle à distance Albre24 (GreenBUS v3.0) ou encore un TRS-IO avec un nombre erroné d' extensions TRS-IO-E (GreenBUS v4.0).

Solution :
Contrôlez la configuration matérielle.

2050 Récepteur numéro : déconnecté

Cause :
Le récepteur à distance ne répond pas aux commandes du transmetteur.

Solution :
Contrôlez l'alimentation du module d'I/O et le raccordement sériel.

2051 Récepteur numéro : reconnecté

Cause :
La connexion entre le transmetteur et le récepteur a été rétablie.

2052 Récepteur numéro : erreur dans le contrôle de Sortie pas connectée numéro NuméroSortie

Cause :
La sortie numérique indiquée est protégée ou en court-circuit et, de toute manière, dans un état différent de celui qui est prévu par le contrôle. La sortie n'est associée à aucun dispositif logique dans la Configuration Virtuel-Physique et il s'agit donc d'une incongruité entre la Configuration et le câblage réel de la machine.

Solution :
Contrôlez la Configuration Virtuel-Physique. Éliminez le court-circuit ou s'assurer que la charge qui est appliquée ne dépasse pas les limites maximales (consulter la documentation technique).

2054 Récepteur numéro : type erroné

Cause :
Pendant la phase d'initialisation des dispositifs à distancerémotes, à une certaine adresse, un module d'I/O différent de celui qui a été indiqué lors de la configuration a été détecté.

Solution :
Assurez-vous que la Configuration matérielle correspond aux réglages des modules d'I/O rémotes.

2055 Récepteur numéro : initialisé

Cause :
Le récepteur s'est connecté au transmetteur après une déconnexion due à une coupure d'alimentation.

2056 Récepteur numéro : erreur alimentation +24 Vcc

Cause :
L'alimentation du champ (+24 Vcc) d'un module d'I/O à distance n'est pas active ou ne fonctionne pas correctement.

Solution :
Contrôlez le fonctionnement de l'alimentation +24 Vcc.

2057 Erreur alimentation GreenBUS

Cause :
L'alimentation du bus qui connecte les modules d'I/O à distance avec le contrôle ne fonctionne pas correctement. Cette alimentation doit avoir une valeur nominale de +12 Vcc et elle est fournie par le contrôle.

Solution :
Assurez-vous que le GreenBUS est alimenté. Verifiez les câbles GreenBUS. Éteindre et rallumer. Si besoin est, remplacer la carte de contrôle.

2058 Récepteur numéro : erreur en relecture TypeDispositif NomDispositif

Cause :
L'état de la sortie indiquée ne correspond pas à celui qui est programmé. Cela peut dépendre de la présence d'un court-circuit, de l' intervention de la protection (charge excessive) ou, tout simplement, de la coupure de l'alimentation. La sortie spécifiée peut être une sortie digitale, une sortie analogique, une sortie de contrôle de l'axe. La type de sortie est spécifié en affichant l'erreur.

Solution :
S'il s'agit d'une sortie digitale, verifier l'alimentation +24V (côté champ), Supprimez tout court-circuit si besoin ou l' excessive absorption de la sortie (consultez la documentation technique). S'il s'agit d'une sortie analogique ou d'une sortie de contrôle des axes, vérifiez la présence et la valeur de la tension réglée sur la sortie (testeur ou oscilloscope), supprimez l'éventuelle situation de court-circuit ou l'excessive absorption de la sortie (consultez la documentation technique).

2059 Échec du test de la mémoire double port du transmetteur

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, l'initialisation du transmetteur GreenBUS (microcontrôleur i296) a échoué. Cela peut dépendre d'une mauvaise configuration des adresses d'I/O et IRQ de la carte ou d'un conflit avec d'autres périphériques présents dans le système. Cela peut également être dû au fait que la carte des axes est endommagée.

Solution :
Contrôlez la configuration de la carte, assurez-vous qu'il n'y a pas de conflits avec d'autres périphériques. Si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port du microcontrôleur i296. Si le problème se reproduit, contactez le Constructeur.

2060 Erreur lors de l'initialisation du transmetteur

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, l'envoi du firmware au transmetteur GreenBUS (microcontrôleur i296) a échoué. Cela peut dépendre d'une mauvaise configuration des adresses d'I/O et IRQ de la carte ou d'un conflit avec d'autres périphériques présents dans le système. Cela peut également être dû au fait que la carte des axes est endommagée.

Solution :
Contrôlez la configuration de la carte, assurez-vous qu'il n'y a pas de conflits avec d'autres périphériques. Si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port du microcontrôleur i296. Si le problème se reproduit, contactez le Constructeur.

2061 Erreur en transmission du firmware au transmetteur

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, l'envoi de la configuration des modules d'I/O remote au transmetteur GreenBUS (microcontrôleur i296) a échoué.

Solution :
Contrôlez la configuration matérielle, si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port du microcontrôleur i296. Si le problème se reproduit, contactez le Constructeur.

2062 Erreur en transmission de la configuration au transmetteur

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, l'initialisation des modules d'I/O remote a échoué.

Solution :
Contrôlez la configuration matérielle, si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port du microcontrôleur i296. Si le problème se reproduit, contactez le Constructeur.

2063 Erreur en transmission de la configuration au récepteur

Cause :
Une erreur a eu lieu pendant l'initialisation d'un module à distance.

Solution :
Contrôlez la configuration matérielle. Des techniciens peuvent effectuer un test matériel du module à distance. Si le problème se présente de nouveau, contactez le service de maintenance technique.

2064 Récepteur numéro : version de firmware non compatible

Cause :
Le récepteur à distance est muni d'une version de firmware non compatible avec le firmware du contrôle.

Solution :
Vérifiez l'installation du contrôle. Si le problème persiste, contactez le service de maintenance technique.

2065 Récepteur numéro : erreur dans une communication asynchrone

Cause :
Une erreur ou une réponse manquée s'est produite pendant la communication d'une commande vers un dispositif à distance (GreenBUS v4.0).

Solution :
Vérifiez les connexions et l'alimentation du GreenBUS. Si le problème persiste, contactez le service de maintenance technique.

2066 Récepteur numéro : erreur générique

Cause :
Une erreur générique s'est produite pendant la communication d'un événement ou d' une alarme du dispositif à distance (GrennBus v4.0)

Solution :
Vérifizr les connexions et l' alimentation du GreenBUS. Si le problème persiste, contactez le service de maintenance technique.

2067 Récepteur numéro : Erreur pendant la transmission de la configuration

Cause :
Une erreur de communication s'est produite dans la transmission des données de configuration à un contrôle à distance

Solution :
Vérifiez les connexions et l'alimentation du GreenBUS. Éteignez et allumez. Si le problème persiste, contactez le service de maintenance technique.

2068 Récepteur numéro : erreur interne no. NuméroErreur

Cause :
Une erreur interne s'est produite sur le dispositif à distance indiqué.

Solution :
Contactez le Constructeur

2069 Récepteur numéro : erreur alimentation +24 Vcc banc numéro

Cause :
L'alimentation du champ (+24 Vcc) d'un groupe de sortie connecté à la même borne d'alimentation électrique n'est pas active ou ne fonctionne pas correctement.

Solution :
Contrôlez le fonctionnement de l'alimentation +24 Vcc.

6.2.3 Erreurs générées par la gestion MECHATROLINK-II

2308 Carte NuméroCarte : Échec de l'initialisation à cause du réglage non correct d'un paramètre de configuration

Cause :
En Configuration Virtuel Physique aucun axe n'a été connecté (dispositif logique) à une carte avec bus MECHATROLINK-II (dispositif physique).

Solution :
Vérifiez les connexions en Configuration Virtuel-Physique.

2341 Carte NuméroCarte : le nombre de servocommandes dépasse la valeur maximale autorisée

Cause :
On a connecté à une carte avec bus MECHATROLINK-II un nombre de servocommandes plus élevé par rapport à la configuration programmée.

Solution :
Vérifiez dans la configuration du système la valeur de Fréquence Contrôle Axes. Dans la table suivante les valeurs corrects sont indiqueés pour la programmation selon le nombre de servocommandes gérés par la carte.

Carte	Fréquence Contrôle Axes (Hz)	Nombre maximum de servocommandes
AlbMech	1000	8
AlbMech	<=500	16
DualMech Mono	1000	8
DualMech Mono	500	20
DualMech Mono	250	30
DualMech	1000	16
DualMech	500	40
DualMech	250	60

2342 Carte NuméroCarte : l'adresse materielle de la servocommande SERVO est supérieure au maximum autorisé

Cause :
Dans une carte avec bus MECHATROLINK-II on a connecté un axe (dispositif logique) à un adresse matériel (dispositif physique) supérieure au nombre de servocommandes que la carte peut gérer.

Solution :
Vérifiez dans la configuration du système la valeur de Fréquence Contrôle Axes. Dans la table suivante les valeurs correctes sont indiquées pour la programmation selon le nombre de servocommandes gérés par la carte.

Carte	Fréquence Contrôle Axes (Hz)	Nombre maximum de servocommandes
AlbMech	1000	8
AlbMech	<=500	16
DualMech Mono	1000	8
DualMech Mono	500	20
DualMech Mono	250	30
DualMech	1000	16
DualMech	500	40
DualMech	250	60

Vérifiez en Configuration Virtuel-Physique la connexion entre dispositif logique et dispositif physique. Par exemple, si le nombre maximum des servocommandes est 8, la connexion entre dispositif logique et dispositif physique doit être entre le premiers 8 axes (de Ax1 à Ax8).

2349 Carte NuméroCarte : la servocommande SERVO n’est pas connecté

Cause :
La connexion physique au servo-entraînement de a platine est suspendue.

Solution :
Contrôlez les câblages du bus MECHATROLINK-II et le servo-entraînement.

6.2.4 Erreurs générées par la gestion CanBUS

2761 Nœud numéro : Déconnecté

Cause :
Il ressort que le nœud Can, indiqué actuellement, n'est pas connecté au bus de terrain, qui se réfère à la carte indiquée, bien qu'il soit présent dans la configuration.

2762 Nœud numéro : Reconnecté

Cause :
Il ressort que le nœud Can vient d'être reconnecté au bus de terrain, qui se réfère à la carte indiquée.

2763 Erreur : transmission échouée

Cause :
Erreur dans la carte indiquée. La transmission des données au nœud CAN indiqué a échouée.

Solution :
Contactez le Constructeur

2764 Nœud numéro : erreur : échec de réception

Cause :
La réception des données attendues par le nœud CAN indiqué a échouée.

Solution :
Vérifiez la connexion et l'alimentation du dispositif CAN indiqué. Vérifiez le câblage de toute la ligne CAN. Vérifiez la connexion de ligne à la commande numérique. Vérifiez la cohérence entre les réglages du protocole du dispositif CAN indiqué par rapport aux réglages du transmetteur installé sur le contrôle numérique (baud rate, adresse, réglages scientifiques du protocole adopté).

2765 Nœud numéro : initialisé

Cause :
Le nœud Can indiqué a été connecté au bus de terrain. Ensuite il a été initialisé correctement.

2766 Condition de panne dans l'interface CAN

Cause :
Signalisation d'une panne dans l'alimentation intérieure du dispositif de l'interface CAN.

Solution :
Contactez le Constructeur

2767 Erreur perte d'état CANopen

Cause :
En raison d'un problème grave le transmetteur CAN n'a été plus en état opérationnel.

Solution :
Contactez le Constructeur

2768 Nœud numéro : Réception PDO échouée

Cause :
Le contrôle numérique n'a pas reçu le PDO, qu'il attendait du nœud CAN.

Solution :
Vérifiez :
- l'alimentation du nœud ;

- que le nœud n'est pas resté en mode preopérationnel (pre-Operational) ;
- les données du PDO et du bus CAN, configurés en Albatros.

2769 Nœud numéro : erreur de réception d'un nœud non configuré

Cause :
Sur le réseau CAN on a détecté un nœud pas déclaré dans la configuration matérielle d'Albatros.

Solution :
Vérifiez l'adresse matérielle du nœud et l'adresse du nœud défini dans la configuration matérielle.
Vérifiez que le nœud a été déclaré dans la configuration matérielle, sinon il est nécessaire de l'ajouter.

2770 Nœud numéro : configuration incorrecte

Cause :
La description des données de RPDO et de TPDO est incorrecte.

Solution :
Corrigez la description des données des pdos de transmission et de réception dans la configuration matérielle.

2771 Nœud numéro : erreur dans la communication SDO

Cause :
Le nœud CAN indiqué n'a pas répondu dans une communication asynchrone (SDO).

Solution :
Vérifiez l'état de connexion du nœud. Si ce problème se reproduit, contactez le Constructeur.

2772 Délai d'expiration dans le cycle CAN d'interrogation des nœuds

Cause :
Une erreur de délai d'expiration dans le cycle CAN d'interrogation des nœuds s'est produite

Solution :
Modifiez dans la configuration matérielle la valeur fixée du moment de l'échantillonnage.

3073 Nœud numéro : erreur Emergency n° NuméroErreur

Cause :
Le dispositif CANopen a détecté une situation d'erreur dans le nœud, spécifiée par le code visualisé.
Le code d'erreur est un numéro hexadécimal.
Il s'agit d'une situation d'erreur concernant le nœud individuel et conforme au standard CiA DS301 - protocole EMERGENCY.

Solution :
Reportez-vous à la documentation concernant le nœud.

3074 Nœud numéro : erreur CAN générique n° NuméroErreur

Cause :
Sur le nœud indiqué s'est produite une erreur interne. Le code d'erreur est un numéro hexadécimal.

Solution :
Contactez le Constructeur.

3088 Carte CAN numéro : nœud NuméroNœud : erreur dans la communication SDO no. NuméroErreur – description

Cause :
Dans une instruction READDICTIONARY ou WRITEDICTIONARY une ou plusieurs demandes de lecture/écriture SDO ont échoué. L'échec des instructions peut être causé par exemple par la demande de lecture d'un objet CANOpen non implémenté dans le dispositif, auquel l'on se réfère ou par l'écriture dans un registre CANOpen d'une donnée non compatible avec le type de l'objet (par exemple tentative d'écriture d'une chaîne dans un objet de type integer). Le code d'erreur fourni est conforme aux spécification DS402 et en plus du code numérique la description en texte est fournie aussi.
Le code d'erreur est un numéro hexadécimal.

Solution :

Contrôlez que les paramètres de BaudRate, Temps d'échantillonnage, etc., définis dans la configuration matérielle et les paramètres d' éventuelles instructions READDICTIONARY et/ou WRITEDICTIONARY dans le codes GPL soient corrects.

6.2.5 Erreurs génerées par la gestion bus Ethercat

3329 Erreur lors de l'initialisation du socket de communication

Cause :
Le firmware n'a pas pu communiquer aver la carte réseau.

Solution :
Si la carte a été configurée dans un système RTX, vérifiez que le fichiers.ini dans la sous-dossier FW d'Albatros sont écrits correctement. Pour vérifier la syntaxe du fichier, voyez le manuel d'installation de RTX en Albatros (installationRTXGuide.pdf).

3330 Erreur pendant la balayage du réseau EtherCAT

Cause :
Pendant la balayage préliminaire du réseau EtherCAT, le maître n'a pas reçu aucune réponse par quelques ou par tous les esclaves configurés ou la configuration ne correspond pas à l'effectif réseau EtherCAT disponible dans le champ.

Solution :
Vérifiez le câblage entre maître EtherCAT et esclave.
Vérifiez les descriptions dans les dispositifs dans la configuration matérielle. Un aide pour identifier l'erreur est donnée par la fenêtre de Diagnostic du matériel. Ici on affiche les nœuds présents et, s'ils sont configurés d'une façon erronée, le nom du dispositif attendu est affiché en plus du nom du dispositifs trouvé.

3331 Erreur dans la configuration de la boîte aux lettres de transmission

Cause :
Le nœud EtherCAT n'a pas répondu à la commande donnée par le Maître. Causes; communication absente, nœud en panne...

Solution :
Vérifiez le câblage et le fonctionnement à distance

3332 Erreur dans la configuration de la boîte aux lettres de reception

Cause :
Le nœud EtherCAT n'a pas répondu à la commande donnée par le Maître. Causes: communication absente, nœud en panne...

Solution :
Vérifiez le câblage et le fonctionnement à distance

3333 Carte EtherCAT numéro : erreur dans le type des extensions du nœud NuméroNœud

Cause :
Le type d'extensions configurées sur un nœud EtherCAT dans la configuration matérielle ne correspond pas à la sorte d' extensions réellement présentes. (Par exemple, dans la configuration matérielle on a défini un TRS-CAT avec une extension TRS-IO-E, alors que dans le système il y a un TRS-CAT avec une expansion TRS-AN-E).

Solution :
Vérifiez que les dispositifs décrits dans la configuration matérielle correspondent à ceux présents.

3334 Erreur lors de configuration des PDOs

Cause :
Le nœud EtherCAT, pour lequel on a cherché de configurer les PDOs, n'est pas présent sur le réseau ou il est en panne.

Solution :
Vérifiez que la configuration du réseau EtherCAT décrite dans la configuration d' Albatros corresponds à la configuration physique du réseau.

3335 Nœud NuméroNœud en alerte NuméroErreur

Cause :
Le nœud indiqué se trouve dans un situation d'alerte.

Solution :
Vérifiez le code d'alerte dans le tableau suivante

Code alerte	Description
0x0001	Unspecified error
0x0002	No memory
0x0011	Invalid requested state change
0x0012	Unknown requested state
0x0013	Bootstrap not supported
0x0014	No valid firmware
0x0015	Invalid mailbox configuration
0x0016	Invalid mailbox configuration
0x0017	Invalid sync manager configuration
0x0018	No valid inputs available
0x0019	No valid outputs
0x001A	Synchronization error
0x001B	Sync manager watchdog
0x001C	Invalid Sync Manager Types
0x001D	Invalid Output Configuration
0x001E	Invalid Input Configuration
0x001F	Invalid Watchdog Configuration
0x0020	Slave needs cold start
0x0021	Slave needs INIT
0x0022	Slave needs PREOP
0x0023	Slave needs SAFEOP
0x0024	Invalid input mapping
0x0025	Invalid output mapping
0x0026	Inconsistent settings
0x0027	Free-Run not supported
0x0028	Synchronization not supported
0x0029	Free-Run needs 3 buffer mode
0x002A	Background watchdog
0x002B	No valid inputs and outputs
0x002C	Fatal Sync error
0x002D	No Sync error
0x0030	Invalid DC SYNCH Configuration
0x0031	Invalid DC Latch Configuration
0x0032	PLL Error
0x0033	Invalid DC IO Error
0x0034	Invalid DC Timeout Error

0x0035	DC Invalid Sync Cycle Time
0x0036	DC Sync0 Cycle Time
0x0037	DC Sync1 Cycle Time
0x0041	MBX_AOE
0x0042	MBX_EOE
0x0043	MBX_COE
0x0044	MBX_FOE
0x0045	MBX_SOE
0x004F	MBX_VOE
0x0050	EEPROM no access
0x0051	EEPROM error
0x0060	Slave restarted locally

3336 Carte EtherCAT numéro : Le nombre d'extensions du nœud NuméroNœud est incorrect

Cause :
Le nombre d' d'extensions configuré sur un nœud EtherCAT en Albatros ne correspond pas au nombre d' extensions réellement présentes. (Par exemple, dans la configuration matérielle on a défini un TRS-CAT avec deux extensions TRS-IO-E, alors que dans le système il n' y a qu'une extension).

Solution :
Vérifiez que le dispositifs décrits dans la configuration matérielle correspondent à ceux présents.

3337 Carte EtherCAT : nœud NuméroNœud déconnecté

Cause :
Le nœud EtherCAT n'a renvoyé aucune réponse au contrôle. Le nœud peut être déconnecté du réseau ou éteint.

Solution :
Verifiez le câblage et que le nœud est sous tension.

3338 Carte EtherCAT : nœud NuméroNœud reconnecté

Cause :
Le nœud EtherCAT indiqué a été reconnecté au réseau et a recommencé à répondre aux demandes du contrôle.

3340 Carte EtherCAT : le nœud NuméroNœud n'a pas répondu à la demande (Code)

Cause :
Le nœud EtherCAT, interrogé au moyen du service SDO, n’a pas répondu à la demande. En outre, si le dispositif a donné un code d’abandon, cela est signalé dans le message d'erreur indiqué *(code)*. Cette erreur peut apparaître lors de l'exécution des instruction SETPZERO, SETPFLY et FASTREAD sur les axes EtherCAT, car ils utilisent la communication SDO pour configurer l’ actionnement. Dans ce cas spécifique, le message d'erreur, en plus des codes d'abandon standard, peut contenir les codes suivants :

- 1 : Le délai d'expiration interne a expiré : le nœud n'a pas répondu dans les 250 ms.
- 00000005 : indice de boîte aux lettres interne incorrect.
- 00000006 : données reçues non conformes à la demande.

Solution :
Contactez le Constructeur.

3341 Carte EtherCAT : le nœud NuméroNœud n'existe pas

Cause :
On a cherché à faire exécuter une commande par un nœud inexistant.

Solution :
Verifiez le numero du nœud dans l'instruction GPL qui a généré l'erreur.
Connectez le nœud.

3342 Câble déconnecté

Cause :
Le câble EtherCAT n'est pas connecté au contrôle.

Solution :
Vérifiez que le câble est correctement connecté à la commande et qu'il n'est pas endommagé.

3343 Carte EtherCAT numéro : nœud NuméroNœud : Il ne passe pas à l'état SAFE-OPERATIONAL (Code)

Cause :
Le nœud EtherCAT indiqué n'est pas passé à l'état SAFE-OPERATIONAL.

Solution :
Contactez le fabricant et signalez le numéro d'erreur indiqué *(code)*, qui représente le code ALstatuscode.

3344 Carte EtherCAT numéro : nœud NuméroNœud : Il ne passe pas à l'état OPERATIONAL (Code)

Cause :
Le nœud EtherCAT indiqué n'est pas passé à l'état OPERATIONAL.

Solution :
Contactez le fabricant et signalez le numéro d'erreur indiqué *(code)*, qui représente le code ALstatuscode.

3345 Carte EtherCAT : communication instable

Cause :
La communication sur le réseau EtherCAT est instable en raison de perturbations ou de câbles ou nœuds défectueux.

4400 Trop d'axes actifs en FASTREAD (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a cherché à utiliser une instruction FASTREAD avec plus d'axes que le nombre d'axes autorisé. Veuillez noter que les axes avec codeurs supplémentaires sont équivalents à un axe double (voir l'instruction SWITCHENC).

Solution :
Réduisez le nombre d'axes avec le codeur supplémentaire utilisé.

6.2.6 Erreurs générées par l'initialisation

769 Configuration logiciel erronée

Cause :
La configuration matérielle du module à distance ne correspond pas à la configuration du logiciel spécifiée.

Solution :
Contrôlez la congruence entre les paramètres matériels du module à distance et les paramètres du software.

770 Numéro d'IRQ mal configuré

Cause :
L'IRQ de la carte des axes n'a pas été réglé correctement dans la configuration du Module. Il s'agit généralement d'un conflit matériel avec d'autres périphériques présents dans le système.

Solution :
Dans les réglages du BIOS de la carte mère, assurez-vous que l'IRQ utilisé par la carte des axes est dédié "Legacy ISA". Verifiez qu'il n'y a pas d'autres périphériques qui utilisent la même IRQ que celui qui est assignée à la carte des axes. Si possible, modifiez l'IRQ du périphérique qui est en conflit avec la carte des axes ; autrement, modifiez l'IRQ de la carte des axes.

772 Erreur lors de la lecture de la zone de mémoire tampon pendant l'initialisation

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, le test de la RAM tampon (Dallas) a échoué. Cela peut dépendre d'une mauvaise configuration des adresses d'I/O et IRQ de la carte ou d'un conflit avec d'autres périphériques présents dans le système. Cela peut également être dû au fait que la carte des axes est endommagée.

Solution :
Contrôlez la configuration matérielle. Des techniciens qualifiés peuvent effectuer un test matériel de la RAM du microcontrôleur i296. N'oubliez pas que le test matériel de la RAM implique l'élimination de toutes les données qui y sont enregistrées. La RAM tampon conserve les valeurs de certains dispositifs, comme les compteurs, les temporisateurs et les offsets des DAC des axes. Avant d'effectuer le test, il faut enregistrer ces valeurs.
Si le problème se reproduit, contactez le Constructeur.

773 Atteint le nombre maximum d'axes configurables

Cause :
On a essayé de configurer un nombre d'axes supérieur au nombre maximal consenti.

Solution :
Réduisez le nombre d'axes configuré. Contactez le Constructeur pour toute information complémentaire.

774 Le Real-time axes n'a pas été exécuté

Cause :
Le firmware de gestion des axes a été initialisé mais il ne fonctionne pas correctement. Il s'agit généralement d'un conflit matériel avec d'autres périphériques présents dans le système.

Solution :
Assurez vous qu'il n'y a pas de conflits avec d'autres périphériques, modifiez la configuration des périphériques qui provoquent le conflit ou les éliminer du système.

775 Temps insuffisant pour l'exécution GPL

Cause :
L'exécution d'une tâche real-time prend trop de temps de cycle. Cette erreur apparaît lorsqu'une tâche real-time ne s'achève pas avant le début du real-time axes suivant (ex. un cycle infini a été créé).

Solution :
Modifiez le code GPL de façon à réduire la durée de la tâche real-time.

776 Temps excessif d'exécution du Temps Réel

Cause :
L'exécution d'une tâche de real-time prend trop de temps de cycle. Le temps de l'exécution est légèrement supérieur à la limite maximale consentie.

Solution :
Modifiez le code GPL de façon à réduire la durée de la tâche real-time.

777 Watchdog expiré

Cause :
Le firmware de la carte des axes est bloqué.

Solution :
Contactez le Constructeur.

778 Le code de Main du firmware est bloqué

Cause :
Le firmware s'est bloqué pour plus de 5 real-time.

Solution :
Contactez le Constructeur.

1025 Carte NuméroCarte : Il ne répond pas à la commande

Cause :
La présence d'une carte des axes a été détectée à l'initialisation, mais elle ne répond pas correctement aux commandes.

Solution :
Des techniciens qualifiés peuvent effectuer un test matériel de la carte des axes. Si le problème se reproduit, contacter le Constructeur.

1026 Carte NuméroCarte : erreur lors de l'envoi du firmware à la carte des axes

Cause :
Impossible d'envoyer le firmware à la carte.

Solution :
Contactez le Constructeur.

1028 Carte NuméroCarte : firmware absent

Cause :
Les firmwares présents sur la carte ne sont pas corrects pour le type de carte relevée.

Solution :
Lancez le firmware correct.

1029 Carte NuméroCarte : Main bloqué

Cause :
Le firmware de la carte n'a pas commencé à fonctionner.

Solution :
Contactez le Constructeur.

1031 Carte NuméroCarte : erreur d'initialisation

Cause :
Une erreur a eu lieu pendant la procédure d'initialisation de la carte des axes.

Solution :
Contrôlez et corrigez les erreurs de système qui ont eu lieu dans les instants antérieurs à l'erreur courant. Ensuite initialisez le système.

1032 Carte NuméroCarte : échec du test de la mémoire double port

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, l'initialisation de la mémoire double port a échoué.
En général, cela est dû à un conflit matériel avec d'autres périphériques présents dans le système, mais cela peut également être dû au fait que la carte est endommagée.

Solution :
Contrôlez la configuration de la carte, assurez-vous qu'il n'y a pas de conflits avec d'autres périphériques. Si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port. Si le problème se reproduit, contactez le Constructeur.

1033 Carte NuméroCarte : le code de Boot du firmware n'est pas en exécution

Cause :
Le firmware de démarrage de la carte a été initialisé mais il ne fonctionne pas correctement. Il s'agit généralement d'un conflit matériel avec d'autres périphériques présents dans le système.

Solution :
Contrôlez la configuration de la carte, assurez-vous qu'il n'y a pas de conflits avec d'autres périphériques. Si l'on utilise un module à distance, retransmettez le firmware au module. Des techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port. Si le problème se reproduit, contactez le Constructeur.

1035 Carte NuméroCarte : non présent

Cause :
Une erreur a eu lieu pendant les tests exécutés lors de l'initialisation de la carte des axes. En particulier, la carte n'a pas été détectée.

Solution :
Assurez-vous que la carte est présente dans le système et qu'elle n'est pas endommagée. Des techniciens qualifiés peuvent effectuer un test matériel de la carte. Si le problème se reproduit, contactez le Constructeur.

1037 Carte NuméroCarte : échec de l'ouverture de la mémoire double port

Cause :
Échec de l'ouverture de la mémoire double port de la carte.

Solution :
Les techniciens qualifiés peuvent effectuer un test matériel de la carte. Contactez TPA S.r.l.

1039 Carte NuméroCarte : Watchdog expiré

Cause :
Le firmware de la carte des axes *NuméroPlug* est bloqué.

Solution :
Contactez le Constructeur.

1040 Carte NuméroCarte : erreur alimentation +24 Vcc

Cause :
L'alimentation du champ (+24 Vcc) des sorties n'est pas présent ou ne fonctionne pas correctement

Solution :
Contrôlez le fonctionnement de l'alimentation +24 Vcc du champ.

1047 Carte NuméroCarte : configuration software non permise

Cause :
Le dispositif a reçu une configuration qui n' est pas compatible avec le matériel actuel ou habilité. Par exemple, on demande la configuration d' une axe non habilitée ou qui n' existe pas sur ce dispositif.

Solution :
Contrôlez la correspondance entre les paramètres matériels de la carte et ceux du logiciel matériel.

1052 Carte NuméroCarte : le code d'amorçage est en cours d'exécution

Cause :
La carte est configurée en mode provisoire et est en train d'exécuter le code de démarrage.

Solution :
Contactez le Constructeur

1053 Carte NuméroCarte : Watchdog axes expiré

Cause :
Une erreur grave s' est produite pendant l' exécution du micrologiciel de la carte de contrôle des axes. Les axes sont désactivés et l' éventuel signal de SYSOK est abaissé. Ne réinitialisez pas le système.

Solution :
Contactez le Constructeur.

1055 Watchdog expiré sur la carte NuméroCarte

Cause :
Le firmware de la carte NuméroCarte est bloqué.

Solution :
Contactez le Constructeur.

1056 Carte NuméroCarte : Erreur d'alimentation interface CAN

Cause :
Le dispositif de transmission sur la ligne CanBus présent sur le plug a produit une erreur d' alimentation. Cela peut dépendre de la présence d'un court-circuit, d'une erreur de câblage du bus, ou de l' endommagement du plug.

Solution :
Vérifiez le câblage de toute la ligne Can. Vérifiez la connexion de ligne à la commande numérique. Supprimez la présence de l' éventuel court-circuit. Si la communication n'est pas rétablie, veuillez contacter le Constructeur.

1057 Carte NuméroCarte : Erreur interne numéro NuméroErreur

Cause :
Erreur dans le matériel du nœud.

Solution :
Contactez le Constructeur.

6.2.7 Erreurs générées par la gestion de la mémoire

1281 Erreur d'allocation de mémoire dans la zone de tas

Cause :
La mémoire RAM disponible est inférieure à celle qui est demandée, par exemple, par une matrice globale.

Solution :

Réduisez la dimension des variables globales situées dans la RAM.

1286 Erreur dans la gestion du tas

Cause :
Erreur de gestion de la mémoire de la part du firmware.

Solution :
Contactez le Constructeur.

1287 Trop de désallocations de memoire à partir du tas

Cause :
Erreur de gestion de la mémoire de la part du firmware.

Solution :
Contactez le Constructeur.

1289 Erreur en création des variables globales

Cause :
Le nombre de variables globales qui a été défini est trop élevé ou l'on a défini des matrices globales trop grandes.

Solution :
Réduisez le nombre des variables globales ou la dimension des matrices.

1290 Erreur de dimension des variables non volatiles

Cause :
Le nombre de variables non volatiles qui a été défini est trop élevé ou l'on a défini des matrices non volatiles trop grandes.

Solution :
Réduisez le nombre des variables non volatiles ou la dimension des matrices non volatiles.

1291 Erreur de dimension des variables en lecture seule

Cause :
Le nombre de variables en lecture seule qui a été défini est trop élevé ou l'on a défini des matrices en lecture seule trop grandes.

Solution :
Réduisez le nombre des variables en lecture seule ou la dimension des matrices en lecture seule.

6.2.8 Erreurs générées par les défaillances

1559 Trace de point d'arrêt

Cause :
Erreur grave du firmware.

Solution :
Contactez le Constructeur.

1569 Code opérationnel du microprocesseur non valide

Cause :
Le microprocesseur a rencontré une instruction inconnue. Il peut s'agir aussi bien de problèmes du matériel de l'ordinateur que de la corruption des fichiers qui contiennent le firmware de Albatros.

Solution :

En cas de module local, assurez-vous que les fichiers ne sont pas corrompus et, si en cas de besoin, réinstallez Albatros. Pour les modules clippers, actualisez le firmware. Effectuez un test du matériel de l'ordinateur et, plus particulièrement, de la RAM. Si le problème se reproduit, contactez le Constructeur.

1586 Valeur INTEGER divisée par zéro

Cause :
On a essayé de diviser un INTEGER par zéro.

Solution :
Contrôlez dans les fonctions GPL si toutes les divisions sont correctes.

1600 Dépassement dans le résultat d'une opération en virgule flottante

Cause :
Le résultat d'une opération entre FLOAT dépasse les capacités du destinataire:

± 3,402823E+38	pour les floats
± 1,79769313486231E+308	pour les doubles.

Solution :
Contrôlez dans les fonctions GPL si les calculs flottants sont corrects.

1601 Dépassement dans le résultat d'une opération en virgule flottante

Cause :
Le résultat d'une opération entre FLOAT est inférieur aux capacités du destinataire:

± 1,401298E-45	pour les float
± 4,94065645841247E-324	pour les double.

Solution :
Contrôlez dans les fonctions GPL si les calculs flottants sont corrects.

1602 Argument non valide pour une opération en virgule flottante

Cause :
Dans une opération en flottant, on a utilisé une opérande non flot.

Solution :
Contrôlez dans les fonctions GPL si les calculs flottants sont corrects.

1603 Valeur en virgule flottante divisée par zéro

Cause :
On a essayé de diviser un float ou un double par zéro. Il est généré aussi dans le cas où on exécute un logarithme de zéro.

Solution :
Contrôlez dans les fonctions GPL si toutes les divisions sont correctes.

1604 Résultat erroné d'une opération en virgule flottante

Cause :
Le résultat d'une opération entre float n'est pas correct.

Solution :
Contrôlez dans les fonctions GPL si les calculs flottants sont corrects.

1605 Une valeur en virgule flottante erronée a été utilisée

Cause :
On a utilisé une valeur en virgule flottante plus petite que le minimum représentable :

±1,401298E-45	pour les float
±4,94065645841247E-324	pour les double.

Solution :
Contrôlez dans les fonctions GPL si les calculs float sont corrects.

1728 Vous avez essayé d'accéder à une adresse non valide

Cause :
Le programme a exécuté un accès à une zone de mémoire non valable.

Solution :
Contrôlez la congruence des variables globales/locales; si le problème persiste, signalez l'anomalie.

1735 Exception générique

Cause :
Une exception non reconnue a eu lieu.

Solution :
Contactez le Constructeur.

1736 Données non alignées

Cause :
Erreur grave du firmware.

Solution :
Contactez le Constructeur.

1801 Alarme température

Cause :
La température de la CPU du contrôle a dépassé les maxima consentis.

Solution :
Assurez-vous qu'il n'y a pas de problèmes de ventilation ou de causes de surchauffe. Si le problème se reproduit, contactez le fabricant.

1802 Alarme ventilateur

Cause :
Le ventilateur de la CPU du contrôle ne fonctionne pas correctement. Le problème peut rapidement provoquer la surchauffe de la CPU.

Solution :
Contactez le Constructeur.

1803 La fréquence de la CPU est instable

Cause :
La fréquence de travail de la CPU n’est pas stable.

Solution :
Contactez le Constructeur.

6.2.9 Erreurs générées par la fonction GPL

4097 Le dispositif TypeDispositif NomDispositif n'est pas configuré

Cause :
Une instruction GPL a utilisé un dispositif non configuré, c'est-à-dire sans connexion VirFisic. Erreur pouvant être générée par toutes les instructions recevant un dispositif par paramètre.

Solution :
Dans les configurations de la commande, assurez-vous que tous les dispositifs utilisés par les fonctions ont une connexion VirFisic.
Retransmettez les configurations à la carte.

4098 La variable globale NomVariable n'existe pas

Cause :
Une instruction GPL a reçu comme argument une variable globale qui n'a pas été définie.
Cette erreur a généralement lieu lorsque le contrôle n'a pas été initialisé correctement.

Solution :
Recompilez tout le code GPL et réinitialisez le contrôle.

4099 Fonction NomFonction non trouvée

Cause :
Une fonction qui n'est pas présente sur la carte a été appelée.
Cette erreur peut se présenter quand l'initialisation du contrôle n'a pas été effectué après une modification du code GPL.

Solution :
Recompilez tout le code GPL et réinitialiser le contrôle.

4101 Gestion non congruente de l'axe NomAxe

Cause :
Un passage d'état illégal a été exécuté sur un axe. Pour les passages d'état, consulter la documentation prévue à cet effet.
L'erreur peut être générée par toutes les instructions qui gèrent les axes. Habituellement, elle se produit dans l'un des cas suivants:

- Si l'on tente d'interpoler, de bobiner ou de coordonner avec un axe déjà engagé dans un mouvement point par point (ou vice versa).
- Si l'on exécute une instruction Chain, SetPFly, SetPTZero sur un axe qui est en mode transparent.
- Si l'on tente d'interpoler, de bobiner ou de coordonner avec un axe qui est asservi à un autre.

Solution :
Assurez-vous que tous les mouvements d'axes s'achèvent par une instruction d'attente en cote, surtout si les axes alternent des mouvements de types différents (point par point, interpolation, etc.).

4105 Instruction non exécutable sur l'axe NomAxe

Cause :
On a essayé d'exécuter une instruction sur un axe qui ne la supporte pas. Par exemple, une instruction d'interpolation sur un axe pas-à-pas.

Solution :
Corrigez le code GPL.

4106 Le dispositif à distance lié à l'axe pas-à-pas NomAxe n'est pas connecté

Cause :
On a essayé d'agir sur un axe pas-à-pas qui n'est pas connecté au contrôle.

Solution :
Contrôlez la connexion du dispositif à distance qui contrôle l'axe.

4107 Instruction SYSOK avec arguments erronés

Cause :
On a exécuté une instruction SYSOK avec des arguments erronés. Cette erreur a lieu lorsqu'une ou plusieurs sorties numériques indiquées en tant qu'arguments dans l'instruction n'ont pas été configurées correctement.

Solution :
Contrôlez le code GPL et la configuration du Virtuel-Physique.

4108 L'axe NomAxe : Cote finale au-delà des limites software

Cause :
On a essayé de déplacer un axe au-delà des limites programmées dans la configuration ou par le code GPL.

Solution :
Corrigez le programme d'usinage qui a provoqué l'erreur. Si besoin est, corriger le code GPL ou la configuration de l'axe.

4110 Vitesse erronée

Cause :
On a essayé d'assigner une vitesse nulle ou négative à un axe.

Solution :
Corrigez le code GPL.

4111 Accélération axe NomAxe négative

Cause :
On a essayé d'assigner une accélération négative à un axe.

Solution :
Corrigez le code GPL.

4112 Décélération axe NomAxe négative

Cause :
On a essayé d'assigner une décélération négative à un axe.

Solution :
Corrigez le code GPL.

4114 Axe NomAxe : mise à zéro sur entrée rapide non effectuée

Cause :
La remise à zéro de la cote d'entrée rapide (zérotage éclair) n'a pas été effectuée correctement. Cette procédure permet de remettre à zéro la cote d'un axe en mouvement à partir du moment où l'entrée rapide correspondante change d'état. Si l'axe achève le mouvement en cours sans que la commutation de l'entrée n'ait eu lieu, l'erreur de système est générée. Cela peut être dû à un mauvais paramétrage du mouvement de l'axe ou à un problème de câblage de l'entrée rapide.

Solution :
Contrôlez le code GPL qui implémente le zérotage éclair, contrôler le câblage de l'entrée rapide.

4115 Axe NomAxe : cran de zéro introuvable

Cause :
La remise à zéro de la cote sur le cran de zéro du codeur n'a pas eu lieu correctement. Cette procédure permet de remettre à zéro la cote d'un axe en mouvement à partir du moment où le cran de zéro du codeur est détecté. Si l'axe atteint la cote de recherche du cran sans que ce dernier ne soit détecté, l'erreur de système est générée. Cela peut être dû à un mauvais paramétrage du mouvement de l'axe ou à un problème de câblage du signal du cran (phase C du connecteur de l'axe).

Solution :
Contrôlez le code GPL qui implémente le zérotage sur le cran, contrôlez le câblage de l'axe.

4353 Code opérationnel d'instruction inconnue (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction illégale a été effectuée pendant l'exécution d'une fonction GPL. En général, cette erreur indique que les fichiers qui contiennent le code GPL compilé sont corrompus.
Si vous avez actualisé le logiciel du contrôle, assurez-vous que vous avez recompilé le code GPL. La version précédente du logiciel de contrôle pourrait, en effet, contenir des instructions qui ne sont plus supportées par la nouvelle version.

Solution :
Recompilez tout le code GPL et initialisez le contrôle. Si le problème persiste, contactez le Constructeur.

4354 Opération mathématique erronée (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction GPL a essayé d'exécuter une opération mathématique erronée, par exemple une division par zéro, ou certaine données introduites dans les instructions GPL sont incongrues. Habituellement, cette erreur est donnée dans les interpolations car c'est la partie du micro-logiciel qui exécute plusieurs calculs mathématiques.

Solution :
Assurez-vous que les paramètres donnés aux instructions d'interpolation sont corrects. Si le problème persiste, signalez le dysfonctionnement au Constructeur.

4355 Adresse erronée de matrice ou vecteur (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction GPL a essayé d'accéder à un élément d'un ensemble ou d'une matrice qui dépasse la dimension maximale. Par exemple, elle a essayé d'accéder à l'élément 10 d'un ensemble de 5 éléments.
Erreur pouvant être générée par toutes les instructions qui acceptent un ensemble ou une matrice comme paramètre.

Solution :
Assurez-vous que tous les indices de matrice ou d'ensemble qui sont passés aux instructions sont compris à l'intérieur des dimensions de l'ensemble et de la matrice.

4356 Instruction RET non appelée par CALL (Fonction : NomFonction ligne : NuméroLigne)

Cause :
La fonction a exécuté une instruction RET et n'a pas trouvé l'adresse de retour relative sur la pile. Le plus souvent, cette erreur est due au fait que l'on a déclaré une sous-procédure avant l'instruction FRET de sortie d'une fonction sans l'avoir protégée avec un GOTO qui prévient son exécution accidentelle. Il peut également s'agir d'un saut involontaire au sein d'une sous-procédure.

Solution :
Contrôlez le flux du programme GPL. Positionnez de préférence les sous-procédures à la fin du corps des fonctions (après l'instruction FRET).

4357 Variable locale inexistante (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction GPL a essayé d'accéder à une variable locale qui n'a pas été attribuée.

Solution :
Remplissez de nouveau et retransmettez toutes les fonctions à la carte. Si le problème persiste, signalez le dysfonctionnement.

4358 Etiquette de saut inexistante (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction GPL a effectué un saut sur une étiquette de saut inexistante.
Erreur pouvant être générée par GOTO, CALL, FCALL, toutes les IF.

Solution :
Remplissez de nouveau et retransmettez toutes les fonctions à la carte. Si le problème persiste, signalez le dysfonctionnement.

4359 Argument macro erroné (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction GPL a reçu des arguments non valables. Erreur pouvant être générée par toutes les instructions. Cependant, dans l'écrasante majorité des cas, le système GPL essaie de corriger automatiquement cette situation, en exécutant des conversions automatiques de type (cast), qui impliquent cependant des pertes de temps. L'erreur est générée lorsque ces conversions ne sont pas possibles et surtout dans les cas suivants:

- instructions opérant sur des dispositifs spécifiques (SETTIMER, SETCOUNTER) et qui en reçoivent un d'un autre type.
- instructions opérant sur des bits qui reçoivent un numéro en virgule flottante (AND, OR, etc.)
- instructions opérant sur des matrices ou des ensembles qui reçoivent une variable (SORT, MOVEMAT, etc.).
- instructions opérant sur des chaînes qui ne reçoivent pas de chaînes.

L'erreur est générée lorsqu'on cherche à exécuter une instruction dans une carte qui ne gère pas cette instruction-ci (par exemple, une instruction SENDPDO ou une instruction RECEIVEPDO dans une carte différente de TMSCan ou de TMSCan+).

Solution :
Corrigez le code GPL.

4360 Échec de l'allocation de la mémoire lors de l'éxecution (Fonction : NomFonction ligne : NuméroLigne)

Cause :
La fonction GPL a essayé d'attribuer une zone de mémoire pour des emplois internes, mais elle n'a pas trouvé la mémoire disponible.
Il se peut que l'erreur indique une situation physiologique due, par exemple, au trop grand nombre de tâches mises en exécution simultanément ou à des variables globales aux dimensions trop grandes.

Solution :
Contrôlez les dimensions des variables globales et locales et essayer de les réduire. Assurez-vous qu'il n'y a pas trop de tâches mises en exécution en même temps et en diminuer le nombre si besoin est.

4361 Trop de tâches actives (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'exécuter plus de 256 tâches en même temps.

Solution :
Réduisez le nombre de tâches actives en même temps.

4362 Format de matrice erroné (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction opérant sur des matrices a trouvé un format non valable. Les instructions qui peuvent générer cette erreur de système sont les suivantes:
- MOVEMAT si le format de la matrice source ne corresponde pas au format de la matrice cible.
- CLEAR si l'on veut zéroter une ligne de matrice qui n'existe pas.
- GETAXIS si le format de la matrice, passé en tant que paramètre, ne correspond pas à celui auquel l'instruction s'attend (consulter la documentation du langage GPL).

Solution :
A l'intérieur de la tâche qui a généré l'erreur, contrôlez les instructions cités de côté. En particulier, assurez-vous que les matrices passées à MOVEMAT ont le même nombre de colonnes et que la matrice passée à GETAXIS présente le bon format.

4363 Trop d'instructions ONINPUT actives (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Plus de 128 instructions OnInput ont été activées.

Solution :
Réduisez le nombre d'ONINPUT.

4364 Axe déjà engagé avec une référence locale (Fonction:NomFonction ligne: NuméroLigne)

Cause :
L'erreur concerne l'activation des triades d'axes à déplacement rotatoire pour effectuer les interpolations sur plusieurs axes cartésiens.
On a essayé d'exécuter une SETRIFLOC en passant à l'instruction d'un axe étant déjà engagé dans une triade d'axes de référence. Cette erreur est également générée si l'on contrôle une RESRIFLOC sur un axe qui n'était engagé dans aucune triade d'axes. Le dernier cas est lorsqu'il n'existe plus de triades de référence disponibles (il ne peut y en avoir que 32 au maximum).

Solution :
Assurez-vous que les triades passées avec la SETRIFLOC n'ont pas d'axes en commun.
Contrôlez les RESRIFLOC.
Assurez-vous également qu'il n'y a pas d'instructions d'attente en cote avant la RESRIFLOC.
Tenez compte du fait que, aussi longtemps que l'interpolation n'est pas achevée, la RESRIFLOC n'est pas réellement exécutée.

4365 Instruction ONINPUT activée sur la même entrée (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction ONINPUT a reçu la même entrée plusieurs fois.

Solution :
Assurez-vous qu'il n'y a pas deux ONINPUT ayant reçu la même entrée comme paramètre.

4366 Trop d'instructions ONFLAG actives (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Plus de 128 instructions OnFlag ont été activées.

Solution :
Réduisez le nombre d'ONFLAG.

4367 Instruction ONFLAG activée sur le même flag (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une instruction ONFLAG a reçu le même indicateur plusieurs fois.

Solution :
Assurez-vous qu'il n'y a pas deux ONFLAG ayant reçu le même indicateur comme paramètre.

4368 Tentative d'écriture d'une variable de type ReadOnly (Fonction : NomFonction ligne: NuméroLigne)

Cause :
On a essayé d'accéder à l'écriture d'une variable de type en lecture seule.
Les variables de type readonly sont toujours globales et elles résident dans la flash de la commande. Elles sont indiquées comme "static" dans l'éditeur des variables globales. Si l'on essaie d'écrire sur l'une de ces variables globales, cette erreur de système est générée.
L'erreur est également générée si l'on utilise des variables résidant dans la RAM tamponnée ("non volatile") comme argument de certaines instructions en écriture. Par exemple dans l'instruction COORDIN la variable passée pour indiquer la ligne en cours d'élaboration doit se trouver dans la RAM.

Solution :
Contrôlez toutes les variables statiques et non volatiles.

4369 Trop d'axes maîtres actifs (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'activer plus de quatre axes en même temps comme maîtres.
Cette erreur ne se génère que pendant l'exécution de l'instruction CHAIN.

Solution :
Réduisez le nombre d'axes maîtres.

4370 Trop d'axes slave actifs (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'activer plus de huit axes asservis à un seul axe maître.
Cette erreur ne se génère que pendant l'exécution de l'instruction CHAIN.

Solution :
Réduisez le nombre d'axes asservis.

4372 Utilisation erronée d'une instruction (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Cette erreur est générée dans l'une des situations suivantes :

1. Vous êtes en train d'utiliser une instruction pour la gestion de la boîte aux lettres (sendmail, waitmail, endmail, Ifmail) ou l'instruction pour la gestion IPC (sendipc, testipc, waitipc) dans une fonction appelée par une instruction Errsys, Oninput ou OnFlag.
2. Vous êtes en train d'utiliser une instruction IfError ou une instruction IfMessage sans avoir activé la gestion des alarmes d'état.
3. Vous êtes en train d'utiliser Watchdog sans la présence de la platine TMSWD.
4. Les paramètres définis dans une instruction d'interpolation (linearinc, linearabs, circle, circinc, circabs, helicinc e helicabs) ne sont pas conformes. Par exemple, le nombre d'axes déclarés est différent du nombre de cotes déclarées ou vous avez déclaré un nombre d'axes supérieur de celui que l'instruction peut gérer.

Solution :
Voici les solutions pour chaque cause énumérée :

1. Déplacez l'instruction qui a provoqué l'erreur dans une autre fonction ou supprimer l'instruction.
2. Vérifiez que la gestion des alarmes d'état est activée. Dans le fichier tpa.ini dans la section [ALBATROS] sous AlarmsHaveState la valeur attribuée doit être 1.

3. Enlevez l'instruction Watchdog ou procurez-vous une platine TMSWD.
4. Vérifiez que les paramètres de l'instruction GPL sont corrects. Chaque axe doit correspondre à une cote. Le nombre d'axes déclarés dans l'instruction ne doit pas être supérieur au nombre d'axes que la fonction peut gérer. Par exemple, l'instruction LINEARABS peut gérer jusqu'à 6 axes. Si, parmi les paramètres, plus de 6 sont déclarés, l'instruction génère l'erreur du système.

4373 Lecture du taux d'alimentation impossible (Fonction : NomFonction ligne : NuméroLigne)

Cause :
L'instruction GETFEED a été utilisée sur une TmsBus ou TmsCan qui n'est pas une carte maître.

Solution :
Vérifiez en configuration matérielle que la carte sur la quelle le feedrate est connecté est une carte maître.

4374 Trop d'instructions de type IPC en exécution (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a franchi la limite maximale de 16 instructions IPC en exécution simultanée.

Solution :
Modifiez le code GPL.

4375 FASTREAD exécutée sur les axes de cartes différentes (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'exécuter une instruction FASTREAD en réglant les paramètres d'axes qui ne sont pas tous connectés à la même carte.

Solution :
Modifiez le code GPL ou la configuration Virtuel-Physique comme il se doit.

4378 Instruction non activée (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'utiliser une instruction dont l'exécution n'est pas validée pour la version de la commande numérique utilisée. Probablement la clé n'est pas insérée correctement ou manque.

Solution :
Contactez TPA S.r.l.

4379 L'instruction ne peut pas être utilisée dans les fonctions lancées par Interrupt (Fonction : NomFonction ligne: NuméroLigne)

Cause :
On a essayé d'utiliser une instruction qui n'est pas admise dans une fonction lancée par Interrupt. Les fonctions lancées par Interrupt sont celles qui sont données en tant que paramètres aux instructions ONERRSYS, ONINPUT et ONFLAG.

Solution :
Modifiez le code GPL. Consultez la liste des instructions non utilisables sur interrupt.

4380 Trop de demandes d'écriture dans la zone de mémoire tampon (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'exécuter trop d'opérations d'écriture en même temps sur la mémoire tampon (la mémoire tampon se caractérise par un temps d'accès relativement élevé).

Solution :
Contrôlez les instructions qui effectuent des opérations d'écriture sur les variables allouées dans la mémoire tampon : compteurs, temporisateurs, matrices et variables déclarées "non volatile".

4381 Impossible d'utiliser une ligne sérielle pas encore ouverte (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'exécuter une instruction qui agit sur un port sériel avant d'avoir exécuté l'instruction COMOPEN sur le port en question.

Solution :
Modifiez le code GPL.

4382 Il n'est pas possible d'ouvrir une ligne sérielle déjà ouverte (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'exécuter une instruction COMOPEN sur un port sériel déjà ouvert avec la même instruction.

Solution :
Modifiez le code GPL.

4383 Vous avez essayé d'ouvrir trop de processus auxiliaires (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'ouvrir plus de 4 processus auxiliaires en même temps.

Solution :
Modifiez le code GPL.

4384 Le processus auxiliaire n'est pas en exécution (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'accéder à un processus auxiliaire qui n'est pas en exécution.

Solution :
Modifiez le code GPL

4385 Vous avez essayé d'ouvrir un processus auxiliaire en partant d'une autre tâche (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé d'ouvrir un processus auxiliaire à partir d'une tâche qui n'est pas celle dont l'exécution a été lancée. Un processus auxiliaire ne peut être utilisé qu'à partir de la tâche dont l'exécution a été entreprise.

Solution :
Modifiez le code GPL.

4391 Erreur lors de l'activation du SYSOK (Fonction : NomFonction ligne : NuméroLigne)

Cause :
L'activation du signal de SYSOK a échoué. En général, cela peut dépendre d'un problème de fonctionnement du transmetteur du GreenBUS sur la carte des axes.

Solution :
Les techniciens qualifiés peuvent effectuer un test matériel de la mémoire double port du microcontrôleur i296. Si le problème se reproduit, contactez le Constructeur.

4394 Trop d'erreurs de cycle (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Il es actif plus de 4000 entre erreurs de cycle.

Solution :
Corrigez le code GPL en confinant le numéro de signaux.

4395 Trop de messages (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Plus de 2000 messages sont actifs.

Solution :
Corrigez le code GPL en limitant le nombre de signalisations.

4397 Dépassement de capacité de la pile (Fonction : NomFonction ligne : NuméroLigne)

Cause :
La pile d'une fonction GPL a franchi la limite maximale de 2Koctets.

Solution :
Recompilez le code GPL et, dans le rapport du compilateur, contrôler l'occupation de pile estimée de la fonction qui a provoqué l'erreur de système. Ensuite, diminuez le nombre de variables locales et de paramètres passés aux fonctions (en les remplissant, par exemple, par des variables globales). Diminuez le nombre de CALLS.

4398 Dépassement de capacité de la pile (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Cela ne peut se produire que sur une erreur grave du micro-logiciel, par exemple en cas de gestion erronée des paramètres d'une fonction ou des variables locales.

Solution :
Contactez le Constructeur.

4399 Paramètre hors de la plage (Fonction : NomFonction ligne : NuméroLigne)

Cause :
Une valeur non comprise dans les limites acceptables a été assignée à une variable GPL ou à un dispositif.

Solution :
Corrigez et recompiler le code GPL.

4865 La définition de la machine pour l'interpolation (G216 ou G217) est absente (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé de bouger les axes avec une interpolation ISO ou on a programmé les indices de configuration, sans avoir d'abord défini les matrices de configuration et les axes, qui composent la machine.

Solution :
Corrigez et recompilez le code GPL en utilisant les instructions ISOG216

4866 La définition des indices de la configuration de la machine sélectionnée (M6) est absente (Fonction : NomFonction ligne : NuméroLigne)

Cause :
On a essayé de bouger les axes avec une interpolation ISO sans avoir défini d'abord les indices des matrices de configuration de la machine.

Solution :
Corrigez et recompiler le code GPL en utilisant l'instruction ISOM6.

6.2.10 Erreurs générées par le driver des communications CNCTPA

16385 Module déconnecté

Cause :
La connexion entre l'ordinateur superviseur et un module a été coupée.
Les causes possibles sont les suivantes :
- non-alimentation du module à distance
- coupure (même temporaire) de la connexion des câbles Ethernet due à un faux contact des connecteurs ou à la détérioration des câbles en question
- non-alimentation ou mauvais fonctionnement hub Ethernet (si présent)
- blocage du firmware du module à distance dû à la détérioration des fichiers de configuration
- réinitialisation du processeur du module à distance à cause d'une surchauffe ou de dérangements EM

Solution :
Assurez-vous que le module est allumé. Contrôler les câbles et les connecteurs Ethernet. Actualisez le firmware du module à distance. Assurez-vous que le module à distance ne subit aucune surchauffe due à une ventilation insuffisante et qu'il est également exempt de dérangements EM. Si le problème persiste, contactez le Constructeur.

16386 Module connecté

Cause :
Un module à distance s'est connecté au PC Superviseur après la phase d'initialisation de Albatros. Pendant le lancement, Albatros essaie de connecter tous les modules fournis par la Configuration du Système. Cette phase dure environ 4 secondes. Les modules qui se connectent après ce laps de temps provoquent l'erreur de système.

16387 Module reconnecté

Cause :
Un module à distance s'est reconnecté à l'ordinateur superviseur après une déconnexion. Cette erreur a donc toujours lieu après l'erreur 16385 "Module déconnecté".

16388 Module Initialisé

Cause :
Un module à distance a été réinitialisé pendant le fonctionnement normal. Cela implique que le module a été déconnecté et reconnecté à l'ordinateur superviseur. Cette erreur a donc toujours lieu après l'erreur 16385 "Module déconnecté".
Cette erreur indique que le module a été réinitialisé en raison, par exemple, d'une panne de courant.

16389 Le module a interrompu la liaison

Cause :
Un module à distance a interrompu la connexion avec Albatros. Cela passe lorsque le module ne reçoit pas commandes ou interrogations de part du PC Superviseur pendant un temps prolongé. Cette erreur indique, donc, un problème (fort ralentissement ou bloc) sur le PC Superviseur.

Solution :
Contrôlez que sur le PC Superviseur il n'y a pas de programmes qui provoquent le bloc ou il ralentissement du système. Désactivez l'écran de veille sur le PC Superviseur. Si le problème se reproduit, contactez le Constructeur de la machine.

16641 Le firmware du contrôle ne répond pas aux commandes

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, le firmware ne répond pas correctement. Il se peut que le problème soit dû à une détérioration des fichiers de firmware.

Solution :
Essayez de réinitialiser le système et, si nécessaire, si nécessaire, réinstallez Albatros. Si le problème persiste, contactez le Constructeur.

16642 TpaSock ne répond pas aux commandes

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, le software de communication avec les modules à distance ne répond pas correctement. Il se peut que le problème soit dû à une détérioration des fichiers.

Solution :
Essayez de réinitialiser le système et, si nécessaire, réinstallez Albatros. Si le problème persiste, contactez le Constructeur.

16643 Le système d'exploitation ne permet pas d'utiliser RTX

Cause :
Le système d'exploitation installé sur l'ordinateur ne permet pas d'utiliser RTX et, en conséquence, il n'assure pas le bon fonctionnement des versions de Albatros qui prévoient sa présence.

Solution :
Actualisez le système d'exploitation de l'ordinateur. Consultez les conditions minimales requises par le système sur le manuel d'installation de Albatros (InstallationGuide.pdf).

16645 Erreur lors de l'envoi du code firmware

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, l'envoi d'un fichier de firmware à un module a échoué.

Solution :
Essayez de réinitialiser le contrôle. Si le problème persiste, contactez le Constructeur.

16646 On n'a pas pu remettre en exécution le code firmware

Cause :
Une erreur a eu lieu pendant la réinitialisation du système. En particulier, le lancement du firmware a échoué après un arrêt précédent.

Solution :
Essayez de réinitialiser le contrôle. Si le problème persiste, contactez le Constructeur.

16897 RTX n'a pas été installé

Cause :
La version de Albatros qui est installée oblige à ce que l'ordinateur soit équipé de RTX, qui, pourtant, en est absent.

Solution :
Installer RTX ou, s'il est déjà présent, l'installer à nouveau. Consultez le manuel d'installation de RTX Albatros (InstallationRTXGuide.pdf).

16898 L'utilisateur n'a pas les droits d'administrateur

Cause :
Albatros a été lancé par un utilisateur n'ayant pas les droits d'Administrateur de l'ordinateur. Les droits d'Administrateur sont nécessaires pour garantir le bon fonctionnement de Albatros.

Solution :
Fermer la session de travail en cours et accéder au système en tant qu'administrateur ou comme un autre utilisateur ayant les droits d'administrateur.

16899 La dimension de la RAM du module est erronée

Cause :
La quantité de RAM mesurée sur le module à distance ne correspond pas à celle qui est prévue. En général, cette erreur laisse entrevoir un problème du matériel informatique.

Solution :
Si le problème persiste, contactez le Constructeur.

16900 L'adresse IP du module est erronée

Cause:
Un module à distance dont l'adresse IP n'appartient pas au sous-réseau de l'ordinateur superviseur a été détecté. Albatros ne peut donc pas communiquer correctement avec le module à distance.

Solution :
Assurez-vous que le paramétrage du service AlbDHCP et de la carte de réseau de l'ordinateur est correct. Voir le manuel d'installation de Albatros (InstallationGuide.pdf).

16901 Ce module est déjà connecté à une autre installation

Cause :
Un module à distance qui est connecté à un autre ordinateur superviseur a été détecté. Cela peut dépendre de la présence dans le réseau d'un autre ordinateur sur lequel Albatros est en exécution et qui utilise le même module. Il se peut également qu'il y ait un problème de fonctionnement du logiciel de communication du module clipper.

Solution :
Assurez-vous que le module à distance n'est pas utilisé par un autre ordinateur superviseur. Réinitialisez le module. Si le problème persiste, contactez le Constructeur.

16902 Le module n'est pas configuré

Cause :
Un module qui n'est pas configuré dans la "Configuration de Système" d'Albatros a été détecté.

Solution :
Configurez le module.

16903 Les réglages du firewall ne permettent pas la comunication

Cause :
Il a été détecté un firewall installé sur le PC qui interdit la communication entre Albatros et les modules à distance.

Remarque : Albatros peut identifier la présence du firewall de Windows et pas d'autres firewall par exemple ces-là qui sont inclus dans quelque software antivirus.

Solution :
Modifiez les réglages du firewall ou désactivez-le.

16904 Carte de réseau non présente ou deshabilitée

Cause :
Il n'a pas été détecté une carte de réseau que peut être utilisée pour la connexion aux modules à distance.

Note: le fait qu'il est détecté une carte ne garantie pas que celle-ci soit bien configurée et connectée.

Solution :
Verifiez la présence e la correcte configuration de la carte de réseau. Si le problème se reproduit, contacter le Constructeur de la machine.

16905 Le code firmware du côntrole est absent

Cause :
Albatros ne trouve pas un fichier de firmware sur l'hard disk du PC. Le problème peut se manifester après avoir effacer de façon accidentelle quelque fichier de firmware ou bien après une mise à jour erronée.

Solution :
Verifiez que les fichier contenus dans le répertoire FW de l'installation de Albatros soient présents et de la version correcte. Contactez le Constructeur de la machine.

16906 Version RTX non compatible avec le code firmware du côntrole

Cause :
Il a été détecté une version de RTX pas compatible avec le firmware installé.

Solution :
Installez la version de RTX correcte ou faire le mise à jour du firmware. Contactez le Constructeur de la machine.

16907 Version du système d'exploitation incompatible avec le code firmware du côntrole

Cause :
La version du système d'exploitation du module à distance n'est pas compatible avec le firmware installé.

Solution :
Installez la version correcte du système d'exploitation ou faire le mise à jour du firmware. Contactez le Constructeur de la machine.

17153 TypeCarte : Absence du code firmware du transmetteur GreenBUS

Cause :
Le répertoire FW ne contient aucun fichier de firmware. En général, le problème est dû à l'élimination du fichier par inadvertance ou à une installation incomplète ou corrompue.

Solution :
Réinstallez Albatros après avoir sauvegardé le système. Consultez le Constructeur de la machine.

17154 TypeCarte : Absence de la partie du code firmware du transmetteur GreenBUS

Cause :
Le fichier qui contient le code firmware du transmetteur GreenBUS est présent dans le répertoire FW mais il est corrompu ou incomplet.

Solution :
Réinstallez Albatros après avoir sauvegardé le système. Contactez le Constructeur de la machine.

17155 TypeCarte : Erreur lors de l'envoi du code d'amorçage du transmetteur GreenBUS

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, l'envoi d'un fichier de firmware à un module a échoué.

Solution :
Essayez de réinitialiser le contrôle. Si le problème persiste, contactez le Constructeur.

17156 TypeCarte : Erreur lors de l'envoi du code de Main du transmetteur GreenBUS

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, l'envoi d'un fichier de firmware à un module a échoué.

Solution :
Essayez de réinitialiser le contrôle. Si le problème persiste, contactez le Constructeur.

17157 TypeCarte : Absence du code d'amorçage

Cause :
Le répertoire FW ne contient aucun fichier de firmware. En général, le problème est dû à l'élimination du fichier par inadvertance ou à une installation incomplète ou corrompue.

Solution :
Réinstallez Albatros après avoir sauvegardé le système. Contactez le Constructeur de la machine.

17158 TypeCarte : Absence du code de Main

Cause :
Le répertoire FW ne contient aucun fichier de firmware. En général, le problème est dû à l'élimination du fichier par inadvertance ou à une installation incomplète ou corrompue.

Solution :
Réinstallez Albatros après avoir sauvegardé le secours du système. Contactez le Constructeur de la machine.

17159 TypeCarte : Erreur lors de l'envoi du code d'amorçage

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, l'envoi d'un fichier de firmware à un module a échoué.

Solution :
Essayez de réinitialiser le contrôle. Si le problème persiste, contactez le Constructeur.

17160 TypeCarte : Erreur lors de l'envoi du code de Main

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, l'envoi d'un fichier de firmware à un module a échoué.

Solution :
Essayez de réinitialiser le système. Si le problème persiste, contactez le Constructeur.

17409 Impossible d'envoyer l'exécutable auxiliaire

Cause :
L'erreur peut avoir lieu pendant l'actualisation du firmware d'un module à distance. Elle peut être due à un problème de fonctionnement momentané du réseau ou à la détérioration du firmware du module. Le message d'erreur peut inclure un code d'erreur.

Solution :

Essayez d' éteindre et réallumez le module à distance et refaire la procédure d'actualisation. Si le problème persiste, contactez le Constructeur.

17410 Impossible de mettre l'exécutable auxiliaire en exécution

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, il n'a pas été possible d'exécuter un programme auxiliaire. Le message d'erreur indique également le nom du programme auxiliaire et, éventuellement, un code d'erreur.

Solution :
Essayez de réinitialiser le système. Si le problème persiste, contactez le Constructeur.

17667 NomDLL : Impossible d'exécuter le code firmware

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, il n'a pas été possible d'exécuter le code firmware. "*NOMDLL*" correspond au composant qui a provoqué l'erreur.

Solution :
Essayez de réinitialiser le système. Si le problème persiste, contactez le Constructeur.

17668 NomDLL : Impossible d'obténir le pointeur à la RAM partagée

Cause :
Une erreur a eu lieu pendant l'initialisation du système. En particulier, il n'a pas été possible d'ouvrir le canal de communication avec le firmware. *"NOMDLL*" correspond au composant qui a provoqué l'erreur.

Solution :
Essayez de réinitialiser le système. Si le problème persiste, contactez TPA S.r.l.

17921 Impossible d'envoyer NODETPA

Cause :
L'erreur peut avoir lieu pendant l'actualisation du firmware d'un module à distance. Elle peut être due à un problème de fonctionnement momentané du réseau ou à la détérioration du firmware du module. Le message d'erreur peut inclure un code d'erreur.

Solution :
Essayez d'éteindre et réallumer le module et refaire la procédure d'actualisation. Si le problème persiste, contactez le Constructeur.

17922 NODETPA n'est pas rédemarré

Cause :
L'erreur peut avoir lieu pendant l'actualisation du firmware d'un module à distance. Elle peut être due à un problème de fonctionnement momentané du réseau ou à la détérioration du firmware du module. Le message d'erreur peut inclure un code d'erreur.

Solution :
Essayez d'éteindre et réallumer le module à distance, puis répétez la procédure d' initialisation. Si le problème persiste, contactez le Constructeur.

17923 NODETPA n'est pas en exécution

Cause :
Un module à distance dont le logiciel de communication n'est pas en exécution a été détecté. En général, cela est dû à un problème de fonctionnement du logiciel de communication. Le message d'erreur peut inclure un code d'erreur.

Solution :
Essayez d'éteindre et réallumer le module. Si le problème persiste, contactez le Constructeur.

18177 NODETPA a cherché à accéder à un adresse non valide

Cause :
Le software de communication de un module à distance a généré une erreur. Le message d'erreur peut inclure un code d'erreur.

Solution :
Essayez d'éteindre et réallumer le module. Si le problème persiste, contactez le Constructeur.

6.3 Signalisations génériques

6.3.1 Albatros commence l'exécution

Il signale le démarrage d'Albatros et affiche quelques informations utiles sur la version du programme et l'environnement d'exécution.

6.3.2 Albatros termine l'exécution

Il signale qu'Albatros est sur le point de terminer l'exécution.

6.3.3 L'ordinateur entre en modalité de suspension

Il signale que l'ordinateur est sur le point de se mettre en veille. A partir de ce moment, Albatros n'est plus en mesure de répondre aux demandes et aux rapports de cycle de GPL.

6.3.4 L'ordinateur sort de la modalité de suspension

Il signale que l'ordinateur vient de sortir du mode veille. Albatros reprend son fonctionnement sans être redémarré.

6.3.5 Éteindre l'ordinateur

Il signale que l'ordinateur est sur le point d'être éteint alors qu'Albatros est encore en cours d'exécution.

6.3.6 Niveau d'accès actuel

Il signale un changement dans le niveau d'accès aux fonctionnalités d'Albatros, généralement pour effectuer une maintenance ou modifier le cycle ou la configuration.

6.3.7 Mise à jour logiciel des modules

Il signale qu'il vous a été demandé de mettre à jour le logiciel et le micrologiciel des télécommandes.

6.3.8 Envoie de la configuration aux modules

Il signale qu'il vous a été demandé de mettre à jour la configuration et le cyclique dans les télécommandes.

7 Configuration du Système

7.1 Introduction

Dans le chapitre consacré à la composition du système, nous avons vu qu'un système Albatros se compose d'un ou de plusieurs modules qui constituent une installation et que chacun de ces derniers est organisé selon une structure hiérarchique.
Pour configurer une machine selon Albatros, il est nécessaire d'effectuer une série d'opérations qui permettent de configurer les différents niveaux logiques et le Matériel qui les supporte.
L'ordre de principe à suivre pour configurer un système est le suivant:

- Configuration de l'installation
- Définition des Groupes et des Sous-groupes
- Configuration des Dispositifs
- Configuration du Système
- Configuration Matérielle
- Configuration Virtuel-Physique

En substance, les Configurations de Module, Groupe et Machine définissent la structure logique de la machine. Les Configurations de Système, Matériel et Virtuel-Physique en définissent la structure au niveau physique.
Les prochains paragraphes traiteront chacun de ces points de façon détaillée.

7.2 Configuration des Dispositifs

7.2.1 Introduction

Le chapitre consacré à la composition du système Albatros présente les différents types de dispositifs dont un module peut se composer. Ces dispositifs sont de nouveau présentés ci-dessous, mais du point de vue de leur configuration.

Comme il est indiqué dans la liste suivante, chaque type prévoit un nombre maximal de dispositifs configurables :

Type de dispositif	**Nombre max.**
Entrée analogique	128
Sortie analogique	128
Entrée digitale	4096
Entrée digitale	4096
Port d'entrée	512
Port de sortie	512
Axe	240
Temporisateur	128
Compteur	128
Flag Bit	1024
Flag Switch	256
Flag Port	256

7.2.2 Dispositif générique

La plupart des dispositifs réclament les mêmes paramètres de configuration. Voici la configuration d'une Entrée Numérique. Les mêmes considérations sont cependant également valables pour :

- Flag Bit
- Flag Switch
- Sorties analogiques
- Port d'Entrées
- Port de Sorties
- Port de Flags
- Temporisateurs
- Compteurs

Pour configurer l'un des dispositifs énumérés ci-dessus, il est nécessaire de configurer les paramètres suivants :

- **Nom :** nom du dispositif d'une longueur maximale de 40 caractères.

- **Commentaire :** description succincte du dispositif, pouvant être traduite en plusieurs langues.
- **Accès à la lecture :** il indique le niveau d'accès minimal demandé pour que le dispositif soit affiché dans la fenêtre de Diagnostic et dans les Tableaux Synoptiques.
- **Accès à l'écriture :** il indique le niveau d'accès minimal demandé pour pouvoir modifier l'état du dispositif.
- **Public :** il précise si l'état du dispositif peut être lu et modifié par un code GPL n'appartenant pas au groupe dans lequel le dispositif se trouve.

7.2.3 Sortie numérique

La sortie numérique a un paramètre de plus que les dispositifs standard : *Monostable*

Pour configurer une sortie numérique, il est nécessaire de configurer les paramètres suivants :
- **Nom :** nom du dispositif d'une longueur maximale de 40 caractères.
- **Commentaire :** description succincte du dispositif, pouvant être traduite en plusieurs langues.
- **Monostable :** si ce paramètre est sélectionné, il configure la sortie en monostable, ce qui revient à dire que lorsque la sortie est réglée sur ON, elle se remet automatiquement sur OFF 200 ms plus tard.
- **Accès à la lecture :** il indique le niveau d'accès minimal demandé pour que le dispositif soit affiché dans la fenêtre de Diagnostic et dans les Tableaux Synoptiques.
- **Accès à l'écriture :** il indique le niveau d'accès minimal demandé pour pouvoir modifier l'état du dispositif.
- **Public :** il précise si l'état du dispositif peut être lu et modifié par un code GPL n'appartenant pas au groupe dans lequel le dispositif se trouve.

7.2.4 Entrée analogique

L'entrée analogique a un paramètre de plus que les dispositifs standard : *Le type de tension à l'entrée.*

Pour configurer une entrée analogique, il est nécessaire de configurer les paramètres suivants :
- **Nom :** nom du dispositif d'une longueur maximale de 40 caractères.
- **Commentaire :** description succincte du dispositif, pouvant être traduite en plusieurs langues.
- **Type :** ce paramètre permet de sélectionner l'intervalle des tensions lues à l'entrée.
- **Accès à la lecture :** il indique le niveau d'accès minimal demandé pour que le dispositif soit affiché dans la fenêtre de Diagnostic et dans les Tableaux Synoptiques.
- **Accès à l'écriture :** il indique le niveau d'accès minimal demandé pour pouvoir modifier l'état du dispositif.
- **Public :** il précise si l'état du dispositif peut être lu et modifié par un code GPL n'appartenant pas au groupe dans lequel le dispositif se trouve.

7.2.5 Axe

Données de base

Les données de base à indiquer sont les suivantes :
- **Nom :** nom du dispositif d'une longueur maximale de 40 caractères.
- **Description :** description succincte du dispositif, pouvant être traduite en plusieurs langues; les espaces ne sont pas admis.
- **Résolution :** résolution du codeur ; elle dépend des caractéristiques du codeur et de l'unité de mesure adoptée. Ne pas oublier que les cartes des axes de Albatros comptent en tant qu'impulsions aussi bien les flancs de montée que les flancs de descente des deux phases du codeur (un codeur de 2500 impulsions par tour sera donc considéré comme un codeur de 10000 impulsions par tour).
- **Type :** type d'axe. Les différents types sont : ***Analogique*** (contrôle analogique), ***Pas à pas***, ***Digital***, ***Compte*** (uniquement lecture du codeur), ***Virtuel***.
- **Unité de mesure :** unité de mesure utilisée pour exprimer les cotes de l'axe. C'est de ce paramètre que dépendent toutes les mesures effectuées. Il est donc conseillé de le programmer avant tout autre paramètre.
- **Inversion phases :** cette fonction permet de compenser via logiciel une éventuelle inversion des câbles des phases du codeur.
- **Inversion référence :** cette fonction permet d'inverser la référence de vitesse de l'axe. Si on l'utilise en même temps que l'inversion des phases, elle permet d'inverser la direction de l'axe (si le câblage est correct).
- **Activation marque :** disponible uniquement pour les axes de comptage, cette fonction remet automatiquement à zéro la cote lorsque le cran du codeur est détecté.

Paramètres de mouvement

Paramètres utilisés pour le déplacement point à point de l'axe :

- **Vitesse maximale :** vitesse maximale de l'axe.
- **Accélération :** durée de la rampe d'accélération.
- **Décélération :** durée de la rampe de décélération.
- **Vitesse minimale :** réglable uniquement pour les axes pas à pas ; il s'agit de la vitesse que le moteur atteint en un seul pas.
- **Type de rampe :** type de rampe d'accélération et de décélération. Non réglable pour les moteurs pas à pas.
- **Proportionnel :** coefficient proportionnel du contrôleur PID de l'anneau de position.
- **Integrative:** coefficient integrative du contrôleur PID de l'anneau de position.
- **Dérivée :** coefficient dérivatif du contrôleur PID de l'anneau de position.
- **Feed Forward :** taux de feed forward. Donne une réduction correspondante de l'erreur de l'anneau à vitesse égale.
- **Feed Forward Accél. :** Taux de feed forward d'accélération. Donne une réduction de l'erreur résiduelle de l'anneau (non réduite par le feed forward) pendant les phases d'accélération et de décélération de l'anneau.
- **Étalons Intégrative :** Programme le nombre d'étalons de l'erreur de boucle utilisés pour le calcul de la composante integrale. Les valeurs valables sont comprises entre 1 et 200. La valeur de default est 50. Voir instruction GPL SETINTEGTIME.

Paramètres d'interpolation

Les paramètres utilisés pour le déplacement en interpolation de l'axe, à l'exception de la vitesse minimale, possèdent la même signification que les paramètres des réglages du Mouvement. Ils sont toutefois utilisés pour les mouvements interpolés.

N.B. : Les valeurs d' accélération et de décélération, réglés dans les paramètres d' interpolation, ne peuvent pas être inférieures aux valeurs correspondantes dans le paramètres de mouvement.

Autres paramètres

- **Vitesse Manuelle :** ce paramètre indique la vitesse maximale de configuration utilisable dans les mouvements manuels. Elle ne doit jamais être supérieure à la vitesse maximale programmée.
- **Servoerror dynamique :** active ou désactive le servoerror dynamique. La valeur par défaut est servoerror dynamique désactivé, donc le servoerror à seuil reste activé. La valeur par défaut est 0. Voir instruction GPL SETMAXERTYPE.
- **Vitesse de référence** et **Erreur de boucle** : ces deux valeurs sont utilisées pour calculer le rapport linéaire réel entre les deux grandeurs dans la machine. Pour que les deux valeurs soient prises en compte, elles doivent être à la fois positives et non nulles et le champ d'erreur **Servoerror Dynamique** doit être activé.
- **Attente axe arrêté** : paramètre activant ou désactivant la fonction de récupération. Il programme un temps d'attente de 50 ms à la fin de chaque mouvement.
- **Timeout déplacement Axe :** les valeurs valables sont comprises entre 0 et 1024. Voir instruction GPL ENABLESTARTCONTROL.
- **Limite connexion encodeur erronée :** les valeurs programmées sont exprimées avec l'unité de mesure avec laquelle la résolution de l'axe est exprimée. Les valeurs programmables doivent être comprises entre 128/résolution de l'axe et 16384/résolution de l'axe. Le défaut est calculé sur la base d'un nombre de pas égal à 1024, c'est-à-dire 1024/résolution de l'axe.
- **Limite servoerror positive :** valeur maximale de l'erreur de boucle pour la boucle de réglage en sens positif.
- **Limite servoerror négative :** valeur maximale de l'erreur de boucle pour la boucle de réglage en sens négatif.
- **Limite axe positive :** valeur maximale de la course de l'axe en sens positif.
- **Limite axe négative :** valeur maximale de la course de l'axe en sens négatif.
- **Fenêtre d'arrive en cote positive :** tolérance sur la cote d'arrivée en sens positif.
- **Fenêtre d'arrive en cote négative :** tolérance sur la cote d'arrivée en sens négatif.

Paramètres de référence

- **Référence :** valeur de la tension de référence à laquelle correspond la vitesse maximale.
- **Adjust automatique :** habilite ou déshabilite le calcule de récupération du offset automatique. Normalement il est habilité.

- **Offset initial :** Valeur à laquelle programmer l'offset initial de référence. La valeur doit être comprise entre –10 et 10. La valeur de default est 0.
- **Fréquence filtre d'Encoche :** Valeur de fréquence à filtrer. La valeur doit être comprise entre 0 et 500.
- **Tension minimale :** Programme les paramètres de tension minimum pour l'axe indiqué . La valeur négative doit être comprise entre –10 et 0, celle positive entre 0 et +10. Voir instruction GPL SETDEADBAND.
- **Seuil :** Programme les valeurs de seuil. Celles-ci sont toujours mineures ou égales aux respectives valeurs de tension minimum, donc la valeur négative de seuil doit être compris entre 0 et la valeur de tension minimum négative. La valeur maximum de seuil doit être compris entre 0 et la valeur de tension minimum positive.

Niveaux d'accès

- **Accès à la lecture :** il indique le niveau d'accès minimal demandé pour que l'axe soit affiché dans les fenêtres de Diagnostic et dans les Tableaux Synoptiques.
- **Accès à l'écriture :** il indique le niveau d'accès minimal demandé pour pouvoir modifier l'état de l'axe.
- **Public :** il précise si l'état de l'axe peut être lu et modifié par un code GPL n'appartenant pas au groupe dans lequel l'axe se trouve.

Enchaînement des axes

Paramètres d'enchaînement des axes. Ce sont les coefficients du contrôleur PID qui compense les différences d'erreur de boucle entre l'axe master et les axes slave :

- **Proportionnel :** coefficient proportionnel
- **Intégrative :** coefficient intégratif
- **Dérivée :** coefficient dérivatif

Correcteurs de linéarité

Configuration des correcteurs de linéarité de l'axe. Les correcteurs permettent de compenser les erreurs de positionnement d'un axe dues à un manque de précision de la mécanique de l'axe lui-même (correcteurs automatiques) et les erreurs dues à l'effet des autres axes de la machine (correcteurs croisés) et typiquement lié à la flexion de la structure. Les correcteurs ne sont pas validés automatiquement. Ils doivent être validés dans la fenêtre de modification des valeurs de correction (entretien **[Modifier...]**) et activés avec le code GPL, avec la commande ENABLECORRECTION.

- **Intervalle de Correction :** Ce paramètre permet de régler la distance entre une correction et la suivante. Le nombre de mesures est donné par la longueur de l'axe divisée par la longueur de l'intervalle de correction.
- **Nom fichier correcteurs :** Espace permettant d'entrer le nom du fichier d'enregistrement des valeurs de correction. Il s'agit d'un fichier ASCII dont les valeurs sont séparées par le caractère ";". Cela ne permet pas d'effectuer des modifications avec un éditeur de texte normal. L'extension du fichier ne doit pas être précisée. L'extension ".csv" (comma separated values) est assignée automatiquement.
- **Données de Correction :** Espace permettant de préciser la liste des axes pour laquelle l'axe courant génére correction. L'axe courant est toujours inclus dans la liste, ce qui revient à dire que le correcteur automatique est toujours présent. Il est possible de spécifier un maximum de 5 autres axes. Pour ajouter un axe, le sélectionner dans la liste de gauche et appuyer sur le bouton **[>>Ajouter]**. Pour éliminer un axe, le sélectionner dans la liste de droite et appuyer sur le bouton **[Supprimer<<]**. Pour préciser les valeurs de correction, sélectionner un axe de la liste de droite et appuyer sur le bouton **[Modifier...]**. La fenêtre qui s'ouvre présente un tableau dans lequel il est possible d'entrer les valeurs de correction.

REMARQUE : Il existe une limite qui fixe un maximum de ***235*** correcteurs de linéarité gérés par le système pour chaque axe. Par conséquent, compte tenu de la longueur de l'axe, l'intervalle de mesure doit avoir une longueur supérieure ou égale au deux cent trente-cinquième de la longueur de l'axe. Par exemple, pour un axe de 2500 mm de longueur, il faut donc programmer un intervalle de correction supérieur ou égal à 10,63 mm. Il existe également une limite qui fixe la valeur maximale d'une correction. Cette dernière doit être inférieure à 1024 pas de codeur. Par exemple, pour un axe ayant une résolution de 256 pas/mm, la correction maximale est de ± 4 mm.

7.3 Configuration Logique

7.3.1 Configuration de l'installation

Pour définir une nouvelle machine ou modifier une machine préexistante, il est nécessaire d'entrer dans la page-écran Configuration Module. La Configuration Module est la configuration des modules dont l'installation se compose.

L'ouverture de l'espace de Configuration n'est possible, si l'on est au niveau d'accès constructeur ou supérieur.

Accès à la Configuration

Sélectionner la fonction ***Ouvrir Configuration*** du menu ***Fichier***.

Si l'installation ne contient encore aucun module configuré, le système ouvre automatiquement la fenêtre Configuration Module. Autrement, il ouvre la fenêtre Configuration Machine. Dans ce cas, pour passer à la Configuration Module, il est possible de suivre la démarche indiquée ci-dessous :

Sélectionner la fonction ***Configuration Module*** du menu ***Modifier***.

Pour ajouter un module à l'installation, il suffit d'appuyer sur le bouton **[Nouveau].** Le bouton **[Modifier]** permet de modifier les données d'un module préexistant. Le bouton **[Supprimer]** permet d'éliminer un module et le bouton **[Fermer]** permet de sortir de la configuration de l'installation.

Les données identifiant une machine, qu'il faudra préciser, sont les suivantes :
- le numéro du module : un nombre entier progressif qui, s'il n'est pas précisé, est assigné par le système
- une description rapide.

Il y a quelques données inhérentes au matériel :

- **Fréquence Contrôle Axes** : indique la fréquence de l' échange périodique des données entre le contrôle numérique et les dispositifs qui lui sont connectés au moyen des bus de champ.
- **Numéro canaux interpolation** : indique le nombre maximal des canaux d'interpolation (c'est à dire des groupes d'axes qui réalisent un mouvement interpolé) qui peuvent être gérés simultanément.
- **Pourcentages d'emploi de la CPU** : indique la pourcentage de temps, concernant la période de contrôle axe (c'est à dire à l'inverse de la "Fréquence Contrôle Axes"), qui est réservée à l'exécution du firmware.

Vous pouvez ouvrir la même fenêtre du module à partir de la branche Module de la Configuration des groupes, de la branche Module de la Configuration de machine et de la branche Module de la Configuration Matérielle.

7.3.2 Configuration des groupes

La première fois que l'on conçoit une machine, il est nécessaire de définir chacun de ses éléments et d'écrire tous les cycles de contrôle, cependant la conception est accomplie à partir d'une machine déjà réalisée et qui est modifiée en fonction des caractéristiques de la machine nouvelle.

Création d'un Groupe

Pour créer un nouveau groupe, il est nécessaire d'entrer dans la page-écran de configuration des groupes. La première branche de l'arbre est le module, duquel proviennent tous le groupes, les sous-groupes et les dispositifs. Si vous appuyez sur la touche **[ENTRÉE]** ou le bouton **[Modifier]** une boîte de dialogue s'affiche afin de modifier les données du module.

Sélectionner la fonction ***Groupes*** du menu ***Modifier***.

À partir d'ici cette fonction permet de créer des nouveaux groupes, de modifier ou d'éliminer ceux qui existent déjà et de copier un groupe.

Liste des commandes pour créer, modifier, supprimer, copier et coller des groupes, sous-groupes et dispositifs.

Commande	Action

Créer un nouveau groupe, sous-groupe, dispositifs	[Ctrl+Entrée], Bouton [Nouveau], Modifier->Nouveau, menu contextuel
Modifier un groupe, sous-groupe, dispositifs	[Entrée], Bouton [Modifier], Modifier->Modifier.., menu contextuel
Annuler un groupe, un sous-groupe, dispositif	[Annuler], Bouton [Supprimer],Modifier->Supprimer, menu contextuel
Activer ou désactiver dans la machine l'utilisation d'un groupe	Menu contextuel, Bouton [Activer]
Copier un groupe, sous-groupe, dispositif	[Ctrl+C], Bouton [Copier], Modifier->Copier, menu contextuel
Coller un groupe, sous-groupe, dispositif	[Ctrl+V], Bouton [Coller], Modifier->Coller, menu contextuel

Quand un nouveau groupe est créé, la fenêtre ci dessous apparaît, dans laquelle on devra configurer
- le nom du groupe
- un commentaire (traduisibles tous les deux dans les différentes langues gérées par Albatros).

Il est également possible de marquer le groupe en tant qu'***Intergroupe***. Au moins un groupe doit être défini comme intergroupe, car cette sélection est utilisée par Albatros pour identifier le groupe "principal" de la machine. C'est le groupe dont la fonction principale (celle qui porte le même nom que le groupe) est lancée automatiquement au démarrage. Ce mécanisme est utilisé pour initialiser la machine et exécuter des tâches au démarrage qui vérifient que tout fonctionne correctement, avant de passer le contrôle à l'utilisateur.
Lorsque vous désactivez un groupe qui a des dispositifs connectés à des dispositifs physiques, le système vous demande si vous souhaitez supprimer le lien virtuel physique. Si vous choisissez de conserver les connexions, les codes pin des dispositifs physiques, auxquels elles sont connectées, seront affichées en gris dans la représentation graphique de l'interface du virtuel-physique.

Adjonction d'un sous-groupe à un groupe
Pour créer le sous-groupe d'un groupe l'opérateur doit se positionner sur le groupe.

Si l'on ne souhaite pas créer de sous-groupes, il est possible de sélectionner la case *Liste des Dispositifs* et d'appuyer sur le bouton **[OK]**. Le nom du sous-groupe est alors assigné automatiquement.

Il est alors possible d'insérer les différents dispositifs au sein du sous-groupe. Le principe est semblable à celui de la création des sous-groupes. Dans ce cas, une fenêtre apparaît pour présenter la liste des dispositifs disponibles.

Après avoir sélectionné le dispositif désiré, confirmer en appuyant sur le bouton **[OK]**.

Une nouvelle fenêtre s'ouvre pour entrer le nom, un commentaire et d'autres données qui diffèrent en fonction du dispositif sélectionné. Une description détaillée des types de dispositifs et de leurs réglages sera présentée dans le chapitre Configuration des Dispositifs.

Copie d'un dispositif

La fonction de copie d'un dispositif permet de copier un dispositif. En premier lieu, il faut sélectionner le dispositif, puis appuyer sur le bouton **[Copier].** Pour insérer le dispositif dans la liste, on doit sélectionner la branche dans la quelle coller le dispositif et activer la commande **[Coller].** Dans la boite de dialogue le nouveau nom du dispositif doit être inséré.

Copie d'un sous-groupe

La fonction de copie de sous-groupe vous permet de copier un sous-groupe comprenant tous les appareils qu'il contient. Pour insérer le sous-groupe, sélectionnez la branche dans laquelle vous voulez le coller et activez la commande **[Coller].** Le nouveau nom du sous-groupe doit être inséré dans la boite de dialogue.

Copie d'un groupe

La fonction de copie d'un groupe permet de copier un groupe avec tous les sous-groupes et tous les dispositifs qu'il contient. Le système copie également l'éventuel synoptique de groupe qui lui est associé (synoptique dont le nom est le même que celui du groupe).

Cela permet de créer rapidement des groupes qui ont une structure semblable à celle d'un groupe préexistant sans qu'il soit nécessaire de recréer tous les groupes l'un après l'autre. Pour copier un groupe, il est nécessaire de sélectionner le groupe que l'on veut copier, d'appuyer sur le bouton **[Copier]** et d'entrer le nom du nouveau groupe dans la boîte de dialogue.

La copie des dispositif, sous.-groupes et groupes peut être réalisée aussi entre des modules différents.

Choix de groupes appartenant à la machine

Après avoir créé l'archive de groupe, vous devez désactiver ou activer les groupes réellement présents. Les groupes sont tous présents dans la machine, sauf si vous décidez de les désactiver à l'aide du bouton **[Désactiver]** ou en sélectionnant la même commande dans le menu contextuel. Si un groupe a été désactivé, le mot **Non présent** apparaît à côté du nom du groupe.

Pour afficher uniquement les groupes présents dans la machine, choisissez l'entrée ***Machine*** dans le menu ***Modifier.***
Pour insérer un nouveau groupe, appuyer sur le bouton **[Insérer]**. Une fenêtre apparaîtra contenant une liste des groupes présents dans l' archive Groupes et non encore entrés dans la machine.

À ce stade, vous devez sélectionner le groupe choisi et **le faire glisser avec la souris** dans la fenêtre Configuration de la machine ou sélectionner le bouton **[Insérer]**.
Il est également possible de supprimer un groupe existant à l'aide du bouton **[Supprimer]** ou de rechercher le nom d'un groupe ou d'un périphérique dans l'arborescence de composition de la machine.

Une machine ne doit contenir qu'un seul intergroupe.

7.4 Configuration Physique

7.4.1 Configuration du système

La configuration du système permet d'associer des ressources physiques (unités de contrôle) à des modules définis dans la configuration logique. Ceci est possible dans la boîte de dialogue Configuration Système. Voici la liste des arguments: **modules** de l'installation et à chacun d'eux il est possible d'associer un **Nœud de réseau**.

- **Nœud local :** Systèmes "Locaux" dans lesquels le HW qui implémente le contrôle est monté directement sur l'ordinateur qui constitue l'interface utilisateur du système.
- **Nom d'un nœud de réseau :** Systèmes "À distance" dans lesquels le HW qui implémente le contrôle est raccordé à l'ordinateur via réseau.
- **Non configuré :** aucune configuration. Il s'agit du défaut initial. Si cette option reste, il sera possible par la suite dans la boîte de dialogue **Connexion des nœuds de réseau** d'associer un module distant.

On peut configure jusqu'à 16 modules et un seulement peut être configuré comme nœud local.
Pour assigner un module sélectionnez le bouton **[Modifier]** ou double-cliquer avec la souris sur le nœud de réseau à modifier. En ouvrant le menu déroulant on visualise la liste des modules à distance, la possibilité d'utiliser un nœud local ou d'organiser le module comme non configuré. Pour confirmer la sélection, on doit sélectionner le bouton  .

N.B. : Le fonctionnement de Albatros dans la machine est protégé par une clé matérielle USB, configurée par TPA.

7.4.2 Configuration matérielle

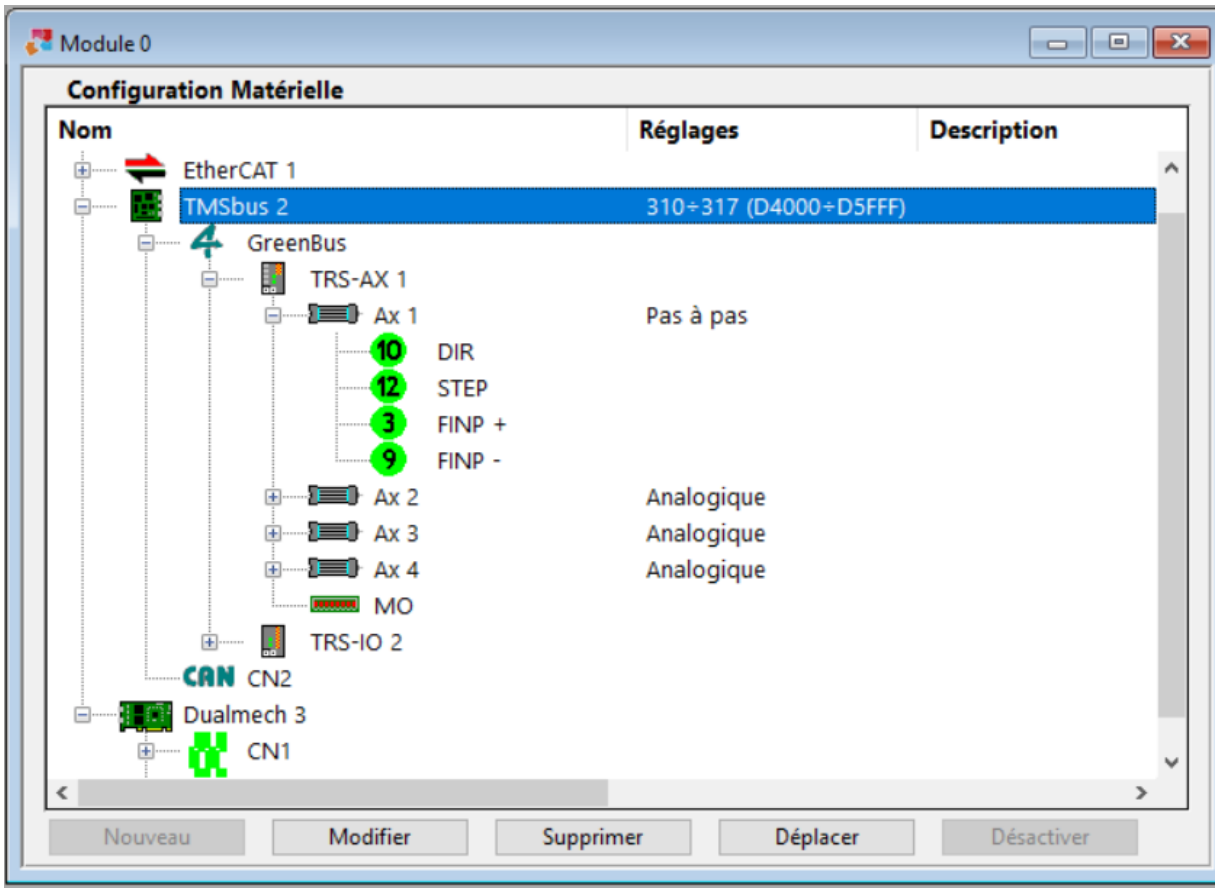
Dans la configuration matérielle, vous définissez les cartes et les nœuds qui composent le système.
La platine dans la première position de la liste s'appelle carte Maître.

Types de cartes configurables :

- TMSbus jusqu'à deux
- TMSbus+ jusqu'à quatre
- TMSCombo+ jusqu'à quatre
- DualMech jusqu'à quatre
- DualMech Mono jusqu'à quatre
- TMSCan jusqu'à deux
- MSCan+ jusqu'à quatre
- AlbMech jusqu'à deux
- EtherCAT un

Description de la fenêtre de configuration matérielle

La fenêtre de configuration matérielle s'affiche en sélectionnant du menu l'entrée ***Modifier->Matériel.*** Pour insérer une carte ou un modèle d'I/O à distance ou un nœud CAN ou EtherCAT, appuyez sur le bouton **[Nouveau]**. La fenêtre qui apparaît alors permet de sélectionner la carte ou le module d'I/O à distance pour le bus CAN et GreenBUS la position dans laquelle elle doit être insérée.
Généralement pour chaque module il est possible de configurer jusqu'à 4 cartes max. et selon le type de carte et de bus, un nombre variable de modules d'I/O à distance.



Configuration matérielle

Dans la colonne **Configurations** des informations liées à la carte ou au nœud sont assignées.
Grâce au bouton **[Déplacer]**, il est possible de déplacer une carte ou un nœud ou un'expansion d'un TRS-IO ou d'un TRS-CAT d'une position à une autre de l'arbre. Les expansions ne peuvent être déplacées qu'à l'intérieur d'un même nœud. Cette opération maintient les liens présents dans la configuration du Virtuel-Physique.

Il est également possible d'invalider un nœud ou une expansion d'un TRS-IO ou d'un TRS-CAT. L'invalidation fait en sorte que les liens sont maintenus dans la configuration du Virtuel-Physique.
Si le nœud appartient à un bus GreenBUS, le nœud et les dispositifs qui lui sont liés sont complètement ignorés par le système. Ainsi, aucune erreur n'est générée si le module n'est pas relevé pendant l'initialisation et aucune erreur n'est générée lorsqu'une instruction GPL est exécutée sur un dispositif associé au module.
Si le nœud ou l'expansion appartient au réseau EtherCAT ou l'expansion appartient à un bus GreenBUS, il ne doit pas être présent dans le réseau. Si en GPL l'on accède à un appareil non connecté, une erreur système se produit.
Une attention particulière doit être apportée à l'utilisation de cette fonction.

Pour désactiver un nœud ou un'expansion utilisez la commande **[Désactiver]**, pour réactiver un nœud utilisez la commande **[Activer]**.

Configurations prédéfinies

Plusieurs configurations prédéfinies sont disponibles. Du menu **Modifier->Modifier le type de contrôle** ou du menu contextuel de la branche du Module il est possible de sélectionner la configuration souhaitée. Si la configuration est nouvelle, l'arbre est peuplé des platines et des nœuds définis, sinon on vérifie que le matériel déjà présent est compatible avec le type de module sélectionné. Tout ce qui n'est pas compatible est supprimé.

Configurer le nœud d'un bus TPA

Types de modules d'I/O à distance configurables en GreenBUS (v3.0) :

- Albre8 8 entrées et 8 sorties digitales
- Albre16 16 canaux configurables via logiciel en tant qu'entrée ou sortie digitale
- Albre24 24 entrées et 24 sorties digitales
- Albre48 48 entrées et 48 sorties digitales
- Albrem 10 ports en entrée et 10 ports en sortie
- AlbSTEP 8 entrées et 6 sorties digitales, un moteur pas à pas
- AlbEV 20 ou 24 électrovalves (connecteur D-sub 25 pin)
- Albrea entrées et 4 sorties analogiques

Types de modules à distance configurables en GreenBUS (v4.0) :

- TRS-AX 4 axes analogiques ou pas à pas
- TRS-EV-24 24 électrovalves (connecteur D-sub 25 pin)
- TRS-16 16 canaux configurables via logiciel en tant qu'entrée ou sortie digitale
- TRS-IO 16 canaux configurables via logiciel en tant qu'entrée ou sortie digitale expansible avec modules type TRS-IO-E et modules TRS-AN-E jusqu'à un nombre total de 5 et modules TRS-AC-E
- TRS-IO-E 16 canaux configurables via logiciel en tant qu'entrée ou sortie digitale utilisables uniquement comme expansions d'un module TRS-IO
- TRS-AN-E 1 entrée analogique et 1 sortie analogique utilisables uniquement en tant qu'extensions d'un module TRS-IO
- TRS-REM dispositif générique à distance utilisé pour réaliser des modules spéciaux. On peut connecter non plus de 12 ports en entrée et 12 ports en sortie.
- TRS-AC-E 1 axe de comptage et 2 entrées digitales configurables en tant que cran de zéro et fastinput. La table suivante montre le nombre maximum de TRS-AC-E configurables dans un TRS-IO. Si dans l'ensemble il y a 3 extension de TRS-IO-E et TRS-AN-E, vous ne pouvez configurer qu'une seule extension TRS-AC-E. Si vous avez configuré une seule extension (TRS-IO-E ou TRS-AN-E), vous pouvez avoir jusqu'à 2 extensions TRS-AC-E.

Les modules à distance TRS-AX, TRS-IO, TRS-REM et TRS-16 ne peuvent être connectés qu'aux platines TMSbus, TMSbus+ et TMSCombo+.

Non plus que 4 modules à distance TRS-AX peuvent être connectés à chaque carte TMSbus et TMSbus+.
Types configurables de module à distance de Tpa en bus EtherCAT :

- TRS-CAT 16 canaux configurables via logiciel en tant qu'entrée ou sortie digitale expansible avec modules type TRS-IO-E, modules TRS-AN-E et modules TRS-AC-E.
- STAR-CAT Transforme une topologie de réseau EtherCAT linéaire en topologie en étoile par moyen d'un canal en entrée et jusqu'à 3 différents canaux en sortie.

Le tableau suivant montre le nombre maximum d'expansions configurables dans un TRS-CAT

Nombre extensions TRS-IO et TRS-AN-E	Nombre d'expansions TRS-AC-E
7	0
5	1

3	2
1	3

Pour les modules à distance TRS-AX, si le nombre de TRS-AX insérés augmente, le nombre de TRS-16 et TRS-IO pouvant être utilisés diminue.
Afin de calculer le nombre maximum de dispositifs à distance TRS-16 et TRS-IO qu'on peut insérer, on doit appliquer la formule suivante : numéro d'autres dispositifs à distance = 32(numéro de TRS-AX * 4).
Par exemple, si 3 TRS-AX sont connectés à une carte TMSbus, en appliquant la formule nous auront : numéro d'autres dispositifs à distance =32-(3*4), donc on peut insérer un numéro maximum de 20 modules à distance de type TRS-16 et/ou TRS/IO.
Le choix de la position du dispositif à distance doit être fait sur la base de l'adresse programmée par switch sur le module à distance. Veuillez vous référer à la documentation sur le matériel du dispositif à distance.

En cas de sélection d'un dispositif à distance TRS-AX, peut-être faut-il configurer le type d'axes géré.

Ci-dessous on décrit les types d'axes, qui peuvent être associés à de différents matériels.

- carte AlbMech — axes numériques
- carte DualMech — axes numériques
- carte DualMech Mono — axes numériques
- dispositif TRS-AX à distance — axes analogiques (si le connecteur spécifique est de type Analogique), axes de comptage (si le connecteur spécifique est de type Analogique), axes pas-pas (si le connecteur spécifique est de type Pas-pas)
- dispositif AlbStep à distance — axes pas-pas
- expansion TRS-AC-E — axes de comptage

Dans les cartes MECHATROLINK-II, chaque axe peut être géré en contrôle de position ou en contrôle de vitesse (par défaut). Le type de contrôle pour chaque axe peut être modifié dans la fenêtre de Configuration matérielle dans la colonne **Règlages.** Le nombre des axes configurables varie en fonction de la valeur de fréquence configurée du contrôle des axes :

Carte	**Fréquence Contrôle Axes (Hz)**	**Nombre maximal de servocommandes**
AlbMech	1000	8
AlbMech	<=500	16
DualMech Mono	1000	8
DualMech Mono	500	20
DualMech Mono	250	30
DualMech	1000	16
DualMech	500	40
DualMech	250	60

Configurer un nœud d'un bus CAN

La carte de contrôle du bus

Albatros peut gérer des dispositifs sur les bus de champ CAN au moyen de cartes Tpa équipées d'un connecteur pour bus CAN. Le bus CAN est présent sur les **cartes TMSbus, TMSbus +, TMSCan + et TMSCan** .

Configurer les données de base et les services
Les données de bus CAN sont définies dans la configuration matérielle. Sélectionnez le bus CAN dont les paramètres doivent être définis et cliquez sur le bouton **[Modifier].**
Les données de base sont les suivantes :

- **Temps d'échantillonnage (TIME)** : temps d'échantillonnage en ms. Il ne peut pas dépasser 60000 (60 secondes). La valeur par défaut est 2. Sur le bus S-CAN on n'accepte que la valeur 2.
- **Temps pour la communication synchrone du PDO (TIMEPDO)** : temps exprimé en ms. Il indique le temps consacré à la communication synchrone des PDO. La valeur programmée ne peut pas dépasser la valeur de TIME (il ne s'agit pas d'une valeur obligatoire).
- **Temps d'attente (TIMEAFTERRESET)** : temps exprimé en msec. indique le temps d'attente pendant la phase initiale après une réinitialisation du logiciel des nœuds dans le réseau. Il ne peut pas dépasser 60000 (60 secondes).
- **Nombre de cycles CAN sans réponse (LIFETIMEFACTOR)** : il s'agit du nombre de cycles CAN sans réponse à l'appel Node Guarding, avant que l'erreur de nœud déconnecté ne soit générée. La valeur ne peut pas dépasser 100 ni être inférieur à 1. La valeur par défaut est 3.
- **BaudRate (BAUDRATE) :** nombre vitesse de communication CAN en kilobits par seconde (la valeur peut être 1000, 500, 250, 125, 100).

Les services peuvent être activés ou désactivés en sélectionnant ou en désélectionnant l'option, qui fait référence au service. Le champ **Extra** peut se voir attribuer des valeurs ayant une signification établie par le fabricant de la machine. Aucune vérification n'est effectuée sur cette valeur. La valeur par défaut est 16.

Le nœud CAN

Insérer un nœud nouveau

Un nœud est inséré en sélectionnant la branche de CAN de l'arbre et en cliquant sur le bouton **[Nouveau]**. Le type de nœud est déduit du type de bus. Si le bus est du type CAN, le nœud est un nœud d'I/O, si le bus est du type S-CAN, le type de bus inséré est du type Servo. Dans la boîte de dialogue pour entrer le nœud doit être choisi :

- **Position** : est le numéro de nœud (à partir de 1).
- **AutoOp** : si cette option est sélectionnée, le dispositif l autorise la transition automatique vers l'état opérationnel après une reconnexions.

Configurer un nœud

Les PDO de transmission et les PDOs de réception peuvent être définis pour chaque nœud. Si le bus est CanOpen, vous pouvez définir jusqu'à 8 PDOs pour les cartes TMSBus et TMSBus + et jusqu'à 4 pour les cartes TMSCan et TMSCan +. Sélectionnez le nœud dans l' arborescence et cliquez sur le bouton **[Nouveau].** Les données de la boîte de dialogue sont les suivantes :

- **Type PDO** : choisissez **Transmission** ou **Réception,** selon que vous définissiez un TPDO ou un RPDO.
- **Dimension** : dimension du PDO de réception ou de transmission.
- **COB-ID** : valeur qui ne peut être définie que sur les cartes TMSBus et TMSBus +. La valeur est affichée et stockée en décimal. Pour afficher la valeur en notation hexadécimale, cochez la case **Hexadécimal** à côté de la valeur elle-même.
- **Asynchrone** : si cette option est activée, les PDOs synchrones sont configurés, c'est-à-dire qu'ils ne sont pas mis à jour à chaque cycle. Cette option est gérée uniquement sur les cartes TMSCan et TMSCan +. La réception des PDOs synchrones doit être effectuée dans le code GPL au moyen de l'instruction RECEIVEPDO.

Caractéristiques de la gestion EtherCAT en Albatros

Le mode de communication est toujours DC-Synchronous. Le premier nœud du réseau fournit l'horloge et il importe donc de s'assurer que ce nœud fournisse un horloge précis et stable, comment il est fournit par exemple par TRS-CAT TRS-CAT. Il n'est pas possible d'utiliser des modes différents, tels que par exemple Free-Run.
Les protocoles gérés sont : CoE (CAN application protocol over EtherCAT) e EoE (Ethernet over EtherCAT). À l'intérieur de CoE, les dispositifs DS401 e DS402 son gérés au moyen du mode d'exploitation par défaut du contrôle de l'axe *cyclic synchronous velocity mode*.
Le nombre maximal de nœuds EtherCAT est 200.

Préface

À chaque dispositif physiqueEtherCAT un fichier ESI (EtherCAT Slave Information) est associé, qui décrit les caractéristiques et fonctionnalités montrées par le dispositif. Le fichier è est en format XML. Pour chaque dispositif ne doit exister qu'une seul fichier ESI. D'une manière générale, les fichiers ESI peuvent être téléchargés du site internet du Constructeur. Albatros recherche ces fichiers dans le dossier défini en Tpa.ini dans la section [tpa] sous DirESIFiles. Le réglage par défaut est le sous-dossier "\EtherCAT" de SYSTEM.

Des fichiers ESI Albatros obtient les informations sur le dispositif, en analysant tous les éléments "/Devices/Device/Type". Chaque dispositif est identifié par un Vendor ID, un Product ID et un Revision Number.

Toujours des fichiers ESI on obtient les informations sur les expansions (appelées aussi modules) des dispositifs. Albatros trouve les informations sur les expansions en cherchant les éléments "Modules/Module" dans le fichier ESI du dispositif.

Configuration du matèriel EtherCAT

Le matériel se configure en décrivant les cartes maître et pour chaque carte la liste des dispositifs physiques connectés avec celle carte sur le bus. Les dispositifs physiques sont appelés aussi "nœud" du bus de terrain. Pour EtherCAT la carte maître n'est pas une spécifique carte de contrôle du bus, mais elle utilise une connexion de réseau du module.

En ce qui concerne les modules locaux, la connexion au réseau doit se réaliser parmi celles qui sont gérées par RTX, tandis que pour les modules à distance une connexion spécifique au réseau du module est utilisée parmi celles gérées par Windows CE 6.0. Pour chaque module local où a distance, il est possible de configurer un seul maître.

La carte maître doit être insérée dans la configuration matérielle en sélectionnant la branche Module et en cliquant sur le bouton **[Nouveau].**

Le nœud EtherCAT

Pour insérer un nœud sélectionnez la branche de l'arbre et cliquez sur le bouton **[Nouveau]**. Les indices des nœuds doivent être consécutifs. Ainsi, pour chaque commande **[Nouveau],** un nœud est ajouté après toutes les autres. Les données suivantes doivent être définies pour chaque nœud :

- **Nom du dispositif :** sélectionnez le nom du périphérique parmi ceux répertoriés. Seuls les appareils avec la révision la plus récente sont affichés. si vous souhaitez voir toutes les révisions, vous devez sélectionner l'élément **Visualiser les révisions**.
- **Mode axe :** sélectionnez le mode d' exploitation à utiliser pour les nœuds de type actionnement, tel que spécifié dans le règlement DS402 par rapport à l'objet 6060_{16} "mode of operation". Le choix est entre la position **Cyclic synchronous position** et **Cyclic synchronous velocity**. Ce dernier est la valeur par défaut.
- **Forcer en tant que nœud de I/O :** si activé, force le contrôle numérique à considérer l'actionnement comme un nœud d' I/O. Cet attribut ne s'applique qu' aux nœuds qui supportent (servocommandes).

Le mode axe est la seule donnée pouvant être modifiée après l'insertion du nœud.

Un nœud EtherCAT peut être supprimé, déplacé, désactivé et copié.Les commandes sont disponibles dans le menu contextuel de chaque branche, dans le menu principal d'Albatros et dans certains boutons de la fenêtre de configuration du matériel.
La commande **Supprimer** supprime le nœud sélectionné, toutes ses extensions et ses liens physiques virtuels. Tous les nœuds suivants sont déplacés à l'adresse précédente.
La commande **Déplacer** déplace le nœud sélectionné dans une nouvelle adresse. Les nœuds qui suivent l'adresse sélectionnée sont également déplacés d'une position.
La commande **Copier** stocke les données du nœud sélectionné pour les utiliser dans une commande Coller ultérieure.
La commande **Coller** insère le nœud précédemment copié dans l'arborescence du dispositif. Le nœud est inséré à la position choisie par l'utilisateur. Tous les nœuds suivants sont déplacés.

L'extension EtherCAT

Pour insérer une extension, sélectionnez le nœud auquel vous voulez ajouter l'extension et cliquez sur le bouton **[Nouveau]**. Les indices des expansions doivent être consécutifs, de sorte que à chaque commande **[Nouveau]** s'ajoute une expansion à la fin des autres. Les objets PDO des extensions ne sont pas modifiables.

Déscription d'un PDO

On peut définir jusqu'à 16 PDOs envoyés du noeud (TxPDO) et jusqu'à 16 PDOs reçus du nœud (RxPDO). Chaque RxPDO décrit un seul PDO que le nœud reçoit du maître, donc sorties numériques et analogiques pour des noeuds de I/O ou vitesse cible et controlword pour des nœuds d'axe. Chaque TxPDO ne décrit qu'un PDO que le nœud envoie au maître, donc des entrées numériques et analogiques pour des nœuds de I/O ou position courante et statusword pour des nœuds d'axe .

Pour la liste et la description des PDOs et des objets mappables sur un PDO il faut se référer à la documentation spécifique du dispositif spécifique EtherCAT et à son fichier ESI.

Il y a trois manières pour décrire les PDOs d'un nœud :

- N'indiquer aucun PDO.
  Ainsi, la commande numérique utilise les PDOs configurés par défaut dans le dispositif. De cette façon, la commande utilise les PDOs configurés par défaut dans le dispositif.
- N'indiquer que les PDOs sans donner la liste des objets.
  À utiliser lorsqu'un CN a plusieurs autres PDOs non programmables.
- Décrivez en détail le PDO, en indiquant l'objet de communication et la liste des objets à mapper.
  Ce mode vous donne le plus de contrôle sur l'information qui est envoyée et reçue par le nœud.

Chaque objet est décrit par son indice dans le dictionnaire d'objets CN, suivi éventuellement d'un sous-indice.
Si le sous-indice n'est pas présent, il est considéré comme étant 0.
L'objet du dictionnaire (object dictionary) est le cœur de chaque dispositif. Il habilite l'accès à tous les types de données du dispositif, aux paramètres de communication et aux paramètres de configuration et traitement de données.

Attention: tous les objets dans le dictionnaire d'objet ne peuvent pas être mappés dans une PDO.

Modifier le PDO d'un actionnement

Pour ce que concerne les nœuds servocommandes il y a un PDO pour chaque axe, de manière que l'énième TxPDO du nœud se référent au nième axe du nœud. Les premiers deux objets de chaque RxPDO et TxPDO ont une signification et une dimension pré-assignées, c'est à dire : le tableau suivant explique comment configurer les PDO associés aux axes du même nœud, contrôlés en mode Axe de **Cyclic synchronous velocity**.

	RxPDO		TxPDO	
1° actionnement	1° objet 16 bit Controlword	2° objet 32 bit Target velocity	1° objet 16 bit Statusword	2° objet 32 bit Actual position
1° axe	6040_{16}	$60FF_{16}$	6041_{16}	6064_{16}
2° axe	6840_{16}	$68FF_{16}$	6841_{16}	6864_{16}
nième axe	Ajouter 800 à chaque objet de l'axe actionnement qui le précède.			

Le tableau suivant explique comment configurer les PDO associés aux axes d'un même nœud, contrôlés avec le mode d'axe **Cyclic synchronous position.**

	RxPDO		TxPDO	
Actionnement	1° objet 16 bit	2° objet 32 bit	1° objet 16 bit	2° objet 32 bit

	Controlword	Target position	Statusword	Actual position
1° axe	6040_{16}	$607A_{16}$	6041_{16}	6064_{16}
2° axe	6840_{16}	$687A_{16}$	6841_{16}	6864_{16}
nième axe	Ajouter 800_{16} à chaque objet de l'axe qui le précède.			

Pour modifier les objets d'un axe, sélectionnez-le dans l'arborescence et cliquez sur le bouton **[Modifier]**. La boîte de dialogue affiche l'indice des PDOs de transmission, l'indice du PDO de réception et la liste des objets configurés.
Plus précisément les données de la boîte de dialogue sont les suivantes :

- **Indice PDO de transmission** : c'est la valeur du PDO pour la transmission de l'axe. Ce n'est pas une valeur éditable.
- **Objets** : liste des objets du PDO de transmission. Les deux premiers objets sont établis en fonction du mode d'axe choisi.
- **Liste d'objets** : liste des objets du PDO de transmission pouvant être utilisés. Pour chaque objet, le sous-indice et la description, s'ils sont présents, sont définis en plus du sommaire.
- **Indice PDO de réception** : c'est la valeur du PDO pour la transmission de l'axe. Ce n'est pas une valeur éditable.
- **Objets** : liste des objets du PDO de réception. Les deux premiers objets sont établis en fonction du mode d'axe choisi.
- **Liste d'objets** : liste des objets du PDO de réception pouvant être utilisés. Pour chaque objet, le sous-index et la description, s'ils sont présents, sont définis en plus de l'index.

Les boutons **[Haut]** et **[Bas]** permettent de modifier l'ordre des objets dans la liste de la fenêtre **Objets**.
Le bouton **[Ajouter]** vous permet d'insérer un objet qui ne figure pas dans la **Liste d'objets**.
Pour modifier la valeur d'un objet, sélectionnez-le et double-cliquez avec la souris. La syntaxe qui permet de définir un objet nouvel est :

- **objet numéro** : c'est le numéro d'objet au format hexadécimal.
- **sous-index** : c'est le numéro du sous-indice. Il doit être séparé du numéro d'objet par le caractère '.' (point). Il s'agit d'une valeur facultative qui, si elle n'est pas définie, est supposée être 0 comme valeur par défaut. Il est au format décimal.
- **L** : dimension de l'objet en bits. Il doit être un multiple de 8.

Les valeurs absentes sont extraites du dictionnaire d'objet ou sont associées à des valeurs par défaut.
Exemple :
objet complet également avec sous-indice et longueur : 1600.1L16
objet sans sous-indice : 1601L24
objet simple : 1603

Ces valeurs peuvent ensuite être lues par GPL au moyen de l'instruction, que nous vous prions de considérer. En outre, il est possible de tracer des objets supplémentaires de la fenêtre d'étallonage et de l'oscilloscope.

Plus généralement, de GPL il est possible d'accéder en lecture et écriture à des objets spécifiques à l'intérieure d'un PDO au moyen des instructions GETPDO et SETPDO que nous vous prions de considérer.

PDOs supplémentaires

Pour définir des PDOs supplémentaires de transmission et réception, sélectionnez nœud le dans l'arborescence et cliquez sur le bouton **[Modifier]** .

Dans la boîte de dialogue pour ajouter un PDO de transmission, cliquez sur le bouton **[TPDOs additionnels]**. Pour ajouter un PDO de réception, cliquez sur le bouton **[RPDOs additionnels]** .

La boîte de dialogue permettant de configurer un objet PDO supplémentaire est similaire à la boîte de dialogue permettant d'insérer des objets.

Dans cette fenêtre, vous pouvez sélectionner l'indice PDO et les objets à associer. Si le PDO a des objets obligatoires et non modifiables, ceux-ci sont affichés et tous les boutons d'édition sont désactivés. Le nouveau PDO est ajouté à l'arborescence à la fin des unités. Pour modifier les données, sélectionnez et cliquez sur le bouton **[Modifier]**. Pour supprimer un PDOs supplémentaire, sélectionnez le dans l'arborescence et cliquez sur le bouton **[Supprimer].**

Acquisition automatique des nœuds EtherCAT

Si le bus EtherCAT est présent dans la configuration matérielle, les nœuds connectés peuvent être acquis à partir du réseau grâce à la commande **Acquisition automatique des nœuds** qui peut être activée à partir du menu contextuel. Pour que la commande soit exécutée, les données de configuration du matériel et les données présentes sur la commande doivent être alignées et les fichiers ESI décrivant le nœud du réseau doivent être présents dans le dossier défini dans Tpa.ini dans la section [tpa] sous DirESIFiles.

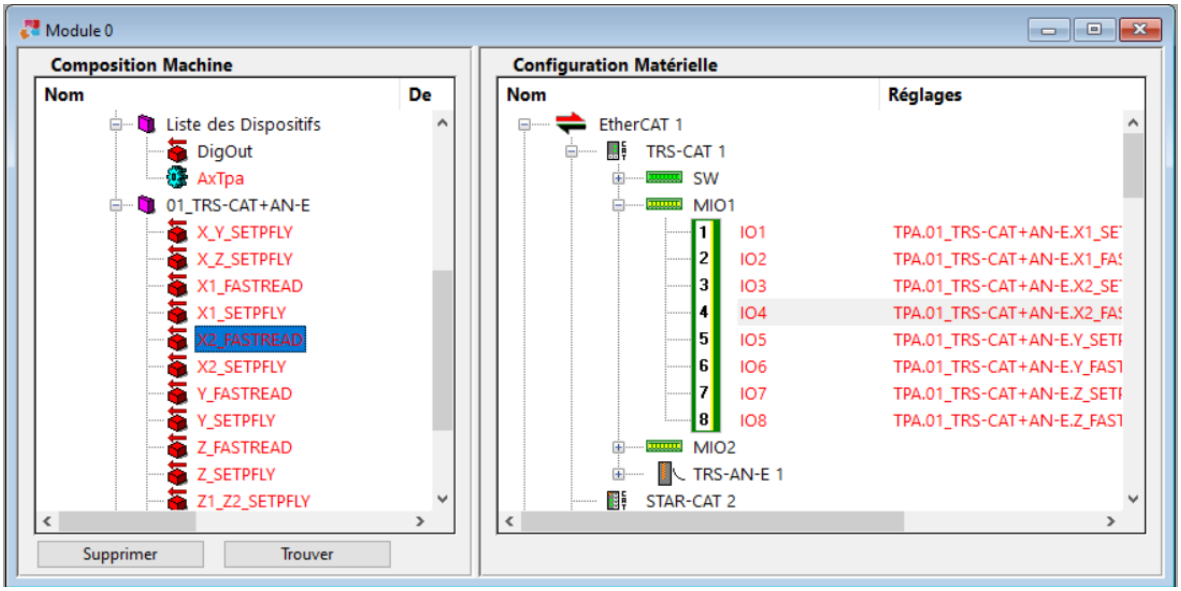
L'acquisition de nœuds EtherCAT à partir du réseau suit les règles ci-dessous :

- aucun nœud n'est configuré dans la configuration matérielle : les données relatives aux nœuds sont lues à partir du réseau et affichées dans la configuration
- les nœuds EtherCAT ont déjà été insérés dans la configuration matérielle :
  - si le nœud lu à partir du réseau a la même adresse, le même ID fournisseur, le même ID produit et le même Numéro de révision du nœud configuré, alors le nœud configuré avec tous ses PDOs est maintenu ;
  - Si le nœud lu par le réseau a la même adresse que le nœud configuré mais un ID fournisseur différent ou un ID produit différent, alors le nœud de configuration est supprimé, ainsi que son virtuel physique et est remplacé par le nœud du réseau. Si le nœud de réseau est un actionnement, les PDO sont lus à partir du fichier ESI, sinon ils sont lus à partir du nœud de réseau ;
  - si le nœud lu à partir du réseau a la même adresse, le même ID fournisseur, le même ID produit et un numéro de révision différent du nœud configuré, alors seul le numéro de révision est remplacé et les PDOs sont maintenus ;
  - si le nœud défini en configuration n'a pas de nœuds correspondant sur le réseau EtherCAT occupant la même adresse, alors il est désactivé en configuration et le virtuel physique est maintenu.

7.4.3 Configuration Virtuel-Physique

La Configuration virtuel-physique est la dernière étape de la configuration. Elle consiste à associer les dispositifs logiques aux composants du matériel.

Deux fenêtres apparaissent lors de l'ouverture de la Configuration Virtuel-Physique : à gauche, la fenêtre Composition de la Machine (virtuel) et, à droite, la fenêtre Composition du Matériel (physique). Ces deux fenêtres représentent tous les éléments qui composent le système de façon graphique et dans une structure arborescente.



Configuration Virtuel-Physique

Les connexions du Virtuel-Physique déjà réalisées sont marquées dans la fenêtre "Composition Machine" par le nom du dispositif en rouge et dans la fenêtre "Configuration Matérielle" par le nom du type de signal, qui se trouve après le numéro du borne, lui aussi en rouge.

Pour chaque axe d'une Platine MECHATROLINK-II on peut configurer en virtuel-physique 6 entrées et une sortie numérique. Lisez, s'il vous plaît, la description détaillée dans le chapitre **Langage GPL->Instructions-> MECHATROLINK-II->MECGESTATUS.**
Si dans un module le bus EtherCAT est present, en tous cas il est possible de configurer des platines pour le bus MECHATROLINK-II, mais avec des limitations: avec real-time à 1 ms on ne peut pas connecter plus de six axes MECHATROLINK-II (par chaque bus); avec real-time à 2 ms la limite s'élève à 16 axes.

Les dispositifs ou les bornes devant encore être raccordés sont en noir. En gris sont représentés les appareils qui appartiennent à un groupe desactivée dont vous vouliez conserver la configuration dans le virtuel physique.

Dans la fenêtre "Composition Matérielle", les signaux relatifs aux axes sont précédés par un rectangle dont la couleur correspond à celle de la gaine du fil situé à l'intérieur du câble de raccordement.
Une connexion peut être mise en évidence en sélectionnant un dispositif logique (ou un composant matériel) et en appuyant sur la barre d'espacement. La connexion est indiquée par une ligne rouge qui unit le dispositif au composant matériel. Il est également possible de laisser les connexions toujours affichées en utilisant la commande **[Alt+Entrée]**.
Pour visualiser quel est le dispositif logique qui est raccordé au composant matériel, sélectionner le composant matériel et double-cliquer.
Pour sélectionner le dispositif logique et le dispositif physique à connecter deux différentes procédures sont possibles :
première procédure

- Faire apparaître, dans la fenêtre "Composition matèrielle", la borne physique à laquelle on veut raccorder le dispositif
- Sélectionner, ou pointer, le dispositif logique désiré dans la fenêtre "Composition de la Machine"

deuxième procédure

- Sélectionner, ou pointer, le dispositif virtuel désiré dans la fenêtre "Composition de la Machine".
- Sélectionner la commande du menu **Modifier->Trouver le dispositif physique convenable** ou le raccourcis clavier **[Ctrl+espace]**. Automatiquement Albatros affiche dans la fenêtre "Composition matérielle" le premier dispositif physique libre auquel le dispositif logique peut être connecté.

troisième procédure possible

- Sélectionner, ou pointer, un dispositif virtuel dans la fenêtre "Composition de la Machine".
- Sélectionner la commande du menu **Modifier->Trouver le dispositif suivant non connecté** ou le raccourcis clavier **[Ctrl+NumPad+]** ou la commande **Modifier->Trouver le dispositif précédent non connecté** ou le raccourcis clavier **[Ctrl+NumPad-].**

Pour effectuer les deux dispositif sélectionnés :

- Sur le dispositif logique à connecter, appuyer sur la touche gauche de la souris et, en la maintenant enfoncée, le faire glisser vers la direction de la borne sélectionnée. Une ligne rouge apparaît pour indiquer la connexion en cours. Une fois que l'on a atteint la ligne de la borne, relâcher la touche pour achever l'opération ou
- sélectionner la commande **Connecter !** du menu **Modifer** ou le raccourci clavier **[Ctrl+L]**.

Pour éliminer une connexion, sélectionnez le dispositif ou le composant concerné et appuyez sur le bouton **[Supprimer]** ou sur la touche **[Suppr]** du clavier.

7.4.4 Cartes de câblage

Après avoir effectué la connexion entre les dispositifs virtuels et les dispositifs physiques correspondants, il est possible d'imprimer des cartes ou des listes qui indiquent l'association entre les dispositifs physiques et virtuels.

Pour qu'il soit possible d'effectuer cette opération, le système doit être muni de MS-Word (version 6 ou suivante). En effet, Albatros en utilise les fonctions pour formater les cartes.
Il est également nécessaire de configurer convenablement le système. Cela signifie que le système doit être muni des fichiers de modèle utilisés pour remplir les cartes. Il s'agit d'un ensemble de fichiers à extension ".doc" qui se trouvent normalement dans le répertoire de l'installation (le répertoire "Cartes").

L'important est que le répertoire dans lequel ces fichiers se trouvent corresponde à celui qui est précisé dans le fichier ***TPA.INI*** avec le mot clé "DirMaps". Par exemple :

```
[TPA]
DirMaps=C:\Albatros\Cartes
```

Pour imprimer les cartes de câblage, sélectionnez n'importe quel composant matériel situé dans la fenêtre à droite de la Configuration Virtuel-Physique ou dans la fenêtre de Configuration Matérielle.

En appuyant sur le bouton "Imprimer" de la barre d'outils ou en sélectionnant la fonction ***Imprimer*** du menu ***Fichier***, on fait apparaître la fenêtre de sélection des options d'impression habituelle. Après avoir fait ses choix et confirmé en appuyant sur le bouton **[OK]**, une fenêtre présente la liste des composants matériels présents dans la configuration.
Cette fenêtre permet de sélectionner les composants dont on veut imprimer la carte de câblage. Il est possible de sélectionner plusieurs composants en maintenant la touche "**Ctrl**" enfoncée et en sélectionnant les composants à l'aide de la souris.
Enfin, en appuyant sur le bouton **[OK]**, on imprime les cartes de câblage. Si l'on désélectionne l'option **Impression sur papier**, les cartes sont enregistrées sous la forme de documents MS-Word dans le répertoire du module courant (Mod. 0, etc.).

Compte tenu du grand nombre de pages devant être imprimées, il est conseillé de faire un essai en imprimant la carte d'un seul composant matériel et en veillant à ce que tout fonctionne comme il se doit. Si, à la place des cartes, l'on imprime une liste des dispositifs logiques, il est probable que l'on n'a pas sélectionné de composant (par exemple une carte d'axe ou un remote) dans la fenêtre relative au matériel. Un composant n'est sélectionné que si son nom est indiqué dans une barre bleu clair.

7.5 Liste des touches pour naviguer à l'intérieure d'une arborescence

Touche	**Description**
Flèche haut Flèche bas	Déplace la sélection jusqu'à la ligne immédiatement précédente ou suivante
Flèche droite	développe d'un niveau la branche sélectionnée et, si déjà développée, elle déplace le sélection dans la branche suivante
Flèche gauche	réduit la branche sélectionnée et, si déjà réduite, elle déplace la sélection dans la branche précédente
+	développe d'un niveau la branche sélectionnée
-	réduit la branche sélectionnée
*	développe tous les niveaux de la branche sélectionnée

8 Outils de développement

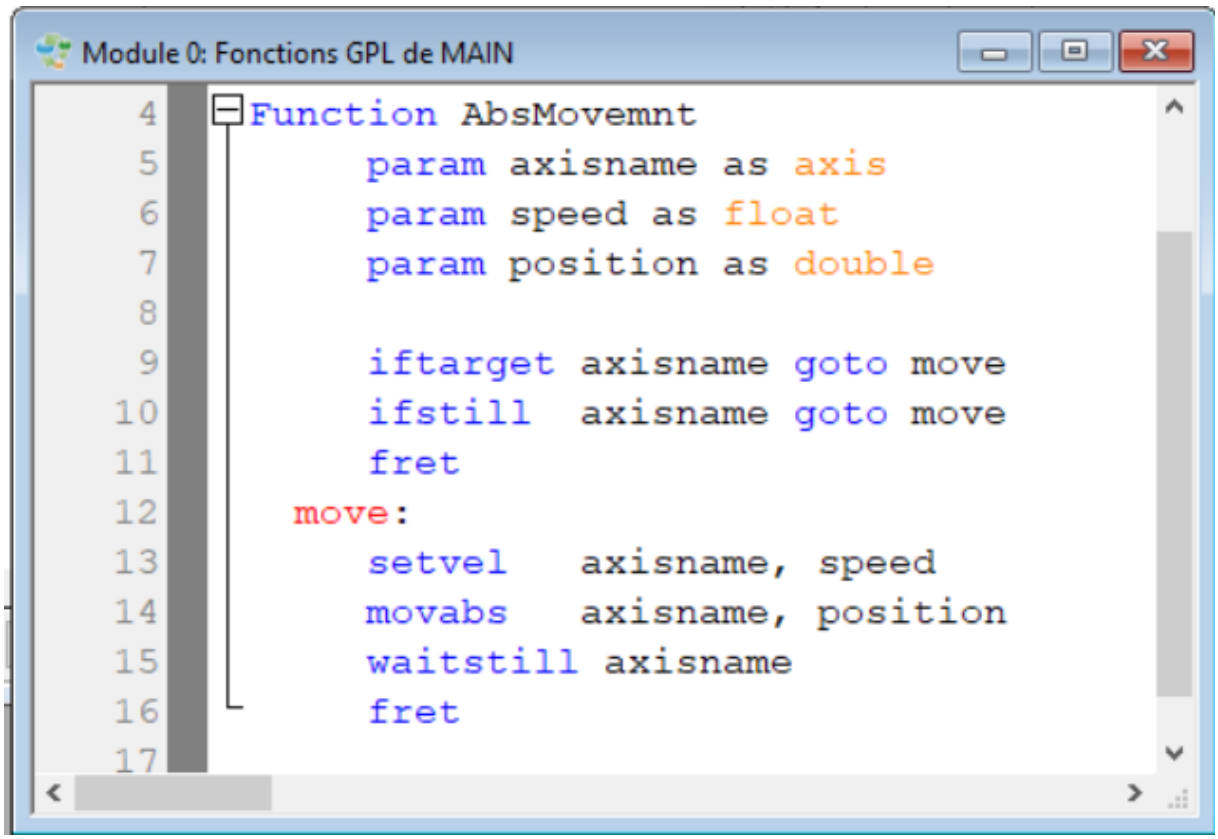
8.1 Editeur GPL

8.1.1 Fonctions de l'éditeur GPL

L'éditeur GPL est l'outil qui permet de créer et de modifier des fichiers contenant le code GPL de Albatros. Cette fonction n'est active qu'avec le mot de passe de niveau égal ou supérieur à Constructeur. Des informations visualisables dans le menu ***Fichier->Informations*** sont assignées à chaque fichier de fonctions.
Les modes de fonctionnement sont les modes classiques d'un éditeur de texte. On y trouve donc les commandes ***Copier, Coller, Rechercher, Remplacer***, etc. Toutes ces commandes peuvent être sélectionnées du menu ***Modifier***

Annuler	Il permet, lorsqu'il est possible, d'annuler la dernière opération effectuée. La situation immédiatement avant la dernière opération effectuée est restaurée.
Répéter	Il rétablit la situation avant la dernière commande d' Annuler
Couper	Le texte ou les données sélectionnes sont élimines et copiés dans une mémoire temporaire, telle qu'on puisse éventuellement les insérer de nouveau avec la commande *Coller.*
Copier	Le texte ou l' élément est copié dans une mémoire temporaire afin d'être inséré de nouveau avec la commande *Coller*
Coller	Le contenu de la mémoire temporaire est inséré en utilisant des critères différents selon la fonction active.
Supprimer	Le texte ou les ligne ou l'élément sélectionné est effacé. Il est possible de récupérer ce qui a été supprimé en agissant immédiatement sur la commande *Annuler.*
Sélectionner tout	Il permet de sélectionner tout le texte du fichier actif. Aux lignes sélectionnées on peut appliquer les commandes de Copier, Couper, Coller.
Rechercher...	Il recherche un texte dans le document courant. Il est possible de programmer des critères à utiliser au moment de la recherche tels que la direction de recherche et la distinction entre caractères majuscule et minuscule, la recherche du mot entier et la recherche au moyen d'expressions régulières.
Rechercher suivant	Il permet de répéter une recherche précédente en donnant la possibilité de modifier les critères de recherche programmés avec la commande Rechercher.
Remplacer	Il permet de rechercher du texte dans le document courant et de le remplacer avec un texte différent.
Insérer dispositif	Il insère un dispositif en le sélectionnant de la liste des dispositifs. Cette fonctionnalité est particulièrement commode quand on travaille avec un élevé nombre de dispositifs et pour cette raison il peut être difficile de s'en rappeler les noms. Seulement les dispositifs du module courant, qui peuvent être rappelés et tous les dispositifs publics des autres modules sont affichés.
Insérer fonction	À partir de la position du curseur, il insère une fonction ou partie d' une fonction, qui est lue par un fichier prototype, écrit par le Constructeur de la machine. On peut écrire plusieurs fichiers prototype. Le fichier prototype est un fichier dont le nome doit commencer avec le préfixe GPL et son extension doit être TXT. Il doit être mémorise dans le répertoire dans le quel sont archivées les librairies (généralement system\lib). Si on désigne plusieurs fichiers prototype, en sélectionnant la commande une boite de dialogue apparaît dans la quelle la liste des noms des fichiers prototype est affichée sans préfixe et sans extension. Les fichiers prototype peuvent contenir, par exemple, des définitions de const communément utilisées, des fonctions de gestion d' erreurs de système, des fonctions génériques, des codes qui implémentent des algorithmes généralement employés, etc. Ils peuvent aussi contenir des commentaires. On peut créer le fichier prototype en enregistrant le texte sélectionné dans le fichier des fonctions GPL La commande n'est disponible que comme accélérateur de clavier **[**Ctrl+Maj+C**]**. Une boîte de dialogue s'ouvre pour insérer le nom qu'on veut attribuer au fragment du code.
Insérer message	Il insère dans le texte GPL le code numérique associé au message choisi. Cela permet d'insérer des nouveaux message dans le fichier de langue.
Activer/désactiver saut de page	Il insère ou enlève un saut de page  . Le saut de page peut être utilisé comme un signet pour sauter à des positions significatives dans le fichier des fonctions.

Atteindre saut de page suivant	Déplace le curseur d'édition sur la ligne du saut de page suivante par rapport à sa position
Atteindre saut de page antécédent	Déplace le curseur d'édition sur la ligne du saut de page antécédent par rapport à sa position



Editeur GPL

Le contrôle de la syntaxe est effectué lors de l'archivage, lorsque le texte est également rempli. Le programmeur peut toutefois effectuer une première analyse visuelle, dans la mesure où le texte est affiché avec une couleur particulière en fonction de ce qu'il représente. Par exemple, les instructions sont en bleu, les commentaires sont en vert et les étiquettes en rouge.
Il est possible de modifier la valeur des tabulations avec la fonction ***Options->Tabulations...*** On peut définir deux typologies de tabulations :

- tabulations absolues : déterminent la position initiale du code GPL, la position initiale du premier argument des instructions et la position initiale du commentaire.
- tabulation relative : (espaces) déterminent le numéro d'espaces qui correspondent à une tabulation

Les tabulations constituent une aide qui rend le code GPL plus compréhensible au niveau visuel.

Pour chaque instruction ou mot clé, il existe une aide en ligne pour la rédaction de la fonction. Pour rappeler l'aide, il suffit de positionner le curseur sur l'instruction et appuyer sur la touche **[F1].**

Une seule instruction peut être écrite sur chaque ligne de texte. Il est possible de poursuivre l'instruction à la ligne suivante en appuyant sur le caractère '_' (Précédé d'un espace) comment le dernier de la ligne. Cela permet d'insérer des commentaires au milieu d'une instruction :

```
Message                                                    _
    1000    ;code du message qui sera affiché              _
    3       ;case du synoptique où il sera affiché  [Entrée]
```

Utilisation des expressions régulières

Dans la fenêtre **Trouver** et dans la fenêtre **Remplacer** des expressions régulières peuvent être utilisées. Une expression régulière est une séquence de caractères numériques et alphanumériques utilisés pour chercher et remplacer des parties de texte dans un texte plus grand. Albatros utilise la

grammaire des expressions régulières d'ECMAScript. Ce paragraphe décrit les principales utilisations des expressions régulières dans Albatros.

L'expression régulière la plus simple est un seul caractère. Il existe des exceptions, représentées par les caractères suivants :

- **. (point)** : le point trouve n'importe quel caractère. Par exemple, "A..." trouve toutes les occurrences de la lettre majuscule A suivie de deux caractères quelconques. .

- **[] (crochets)**: les crochets permettent de spécifier une liste de caractères à l'intérieur de ceux-ci. Le texte recherche des occurrences pour chacun des caractères présents. Si le premier caractère est **^ (accent circonflexe)**, tous les caractères seront recherchés, à l'exception de ceux figurant entre crochets.

  Par exemple :
  [<>] : on recherche toutes les occurrences du caractère < et du caractère >.
  [.]AX : on recherche toutes les occurrence qui contiennent la chaîne "AX".
  [a-d] : on recherche toutes les occurrence qui contiennent les caractères a, b, c e d. Le trait d'union indique un ensemble de caractères.
  [\[\]] : on recherche toutes les occurrences du caractère [et du caractère].
  [^+] : on recherche tous les caractères sauf le caractère +.

- ***(astérisque)**: toutes les occurrences d'un caractère (ou groupes de caractères) trouvées avant l'astérisque sont recherchées. L'astérisque n'affecte que le caractère qui le précède : pour le faire agir sur un groupe de caractères, il faut utiliser des parenthèses rondes.

  Par exemple :
  ;-*: on recherche toutes les occurrences des caractères **;** e **;-** e **;--------**.

- **+ (signe plus)** : recherche une ou plusieurs occurrences du caractère qui le précède. Il diffère de l'utilisation d'un astérisque, car le caractère précédant le signe + doit toujours être présent. Le signe plus n'exerce aucun effet que sur le caractère qui le précède : pour le faire agir sur un groupe de caractères, il faut utiliser des parenthèses rondes. Prenons l'exemple utilisé pour le caractère astérisque :

  ;-+ : on recherche toutes les occurrences des caractères **;-** e **;---**. Le caractère seul n'est pas recherché **; (point-virgule)**.

- **? (point d'interrogation)**: rend facultatif le personnage précédent, qui peut donc être présent au maximum une fois.
  Par exemple :
  Setfeedi?: on recherche toutes les occurrences du mot Setfeed et du mot Setfeedi.

- **{} (accolades)**: spécifient combien de fois un caractère (ou un groupe de caractères) doit être présent dans le texte.
  Par exemple :
  ee{2} : on recherche toutes les occurrences de deux ee consécutives.

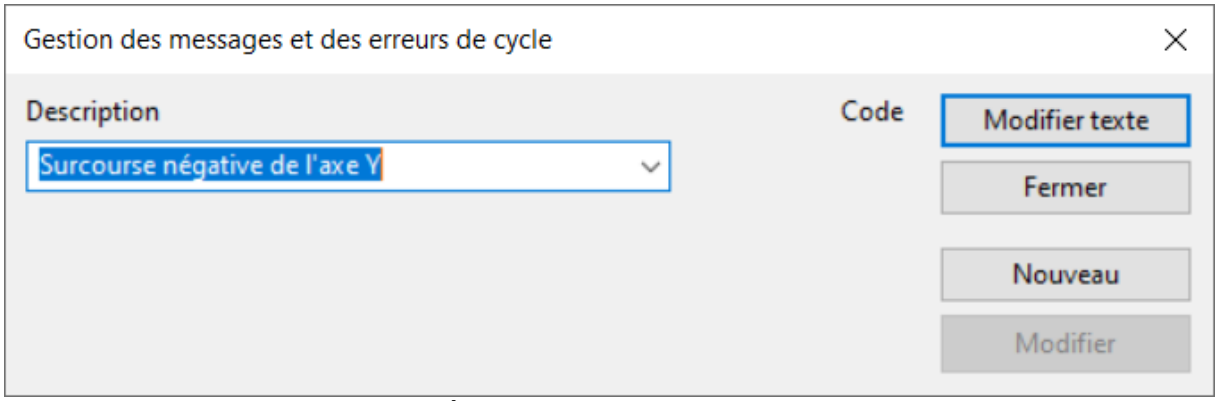
- **^. (accent circonflexe et point)**: trouve le premier caractère de chaque ligne.

- **^ (accent circonflexe)**: trouve le terme recherché uniquement s'il se trouve en début de ligne.
- **| (barre verticale):** trouve les termes qui apparaissent avant et après le caractère '|'.
  Par exemple :
  Send|Const: on recherche toutes les occurrences du mot Send et du mot Const.
- **\ (barre oblique inverse)**: la barre oblique inverse possède une fonction double :
  1) transforme le caractère normal en une fonction
  \b trouve le bord de début ou de fin du mot
  \B trouve le mot sauf le bord de début
  \d ne trouve que les chiffres
  \D trouve tout sauf les chiffres
  \s trouve l'espace
  \S trouve tout sauf l'espace
  Exemples :

\bi trouve dans le text tous les mots, qui commencent par la lettre i.
2) transforme un caractère spécial en un caractère normal. Par exemple, pour rechercher dans un texte le caractère spécial **^ (accent circonflexe),** il suffit de mettre devant lui la barre oblique inverse.

8.1.2 Insérer un message

Dans Albatros on trouve deux types de messages : les messages de module et les messages de groupe. On peut sélectionner la commande du menu ***Modifier->Insérer Message.***
Les messages de groupe sont insérés directement dans l'éditeur pendant l'écriture du code GPL, au moyen de l'instruction DEFMSG. Ils ne sont visibles et utilisables qu'au sein du groupe dans lequel ils sont définis. Il est donc possible d'avoir la même définition de message dans plusieurs groupes, sans qu'il y ait superposition.
Contrairement aux messages de groupe, les messages de module peuvent être utilisés par tous les groupes. Ils peuvent être insérés au moyen de la boîte de dialogue qui permet de rappeler un message existant déjà dans le fichier de langue ou d'en insérer un nouveau.



Fenêtre de gestion des messages

De cette manière, il n'est pas obligatoire de modifier le fichier. Le message sera inséré dans la langue courante et devra être, par la suite, traduit dans les autres langues (en utilisant le programme TpaLangs).
Tous les messages présents dans le fichier de langue sont énumérés dans la fonction **Description.** Pour insérer un message au sein même de la fonction, il est nécessaire de sélectionner le bouton **[Modifier Texte]** après avoir choisi le texte.
Pour modifier un message existant **[Modifier]** ou en créer un nouveau **[Nouveau]**, il faut tout d'abord taper la modification ou le nouveau texte et appuyer ensuite sur le bouton correspondant.

8.1.3 Cryptage

Dans Albatros, il est possible, grâce au cryptage, de rendre impossible l'affichage du texte source des fonctions.

Pour activer le cryptage il faut imposer en tpa.ini la fonction Tele+=1. La valeur par défaut est 0, ce qui signifie que le cryptage n'est pas activé.

Lorsqu'on enregistre un fichier de fonctions et le cryptage est activé il sera affiché le message suivant: "Voulez-vous encoder le fichier?" Si la réponse est [Oui], le fichier est crypté. Il est possible d'encoder un fichier précédemment enregistré sans cryptage, alors que un fichier crypté restera toujours comme ça.

Lorsque le cryptage est activé, si le mot de passe est de type Constructeur journalier, un fichier sans cryptage ne sera pas crypté.
Un fichier des fonctions encodées peut être affiché ou modifié en Albatros seulement par l'utilisateur qui l'avait précédemment enregistré. Le propriétaire d'un fichier de fonctions cryptées ne peut pas changer!

Si l'on veut éliminer le cryptage d'un fichier on doit utiliser un programme externe appelé SBIANCA, qui se trouve dans le répertoire Bin de Albatros.

L'interface graphique est un moyen d'utiliser le programme Sbianca. Sélectionnez les fichiers que vous voulez décrypter. Les propriétés "**État**" et "**Informations d'identifications**" sont indiquées pour chaque fichier. L' **État** peut être **Texte brut,** si le fichier est sans cryptage ou **Crypté** si le fichier est crypté. La propriété **Informations d'identifications** donne des informations sur la visibilité du fichier:

Lisible signifie que le fichier peut être affiché du niveau de mot de passe courant, autrement **Bloqué** signifie que le fichier ne peut pas être affiché.
Une fois que vous avez choisi les fichiers, vous devez cliquer sur [**Décrypter !**] pour les décrypter ou **[Crypter !]** pour les crypter.

Une deuxième façon de l'utiliser est la ligne de commande. Les commandes et les fichiers sont transmis au programme en tant qu'arguments. Si aucun argument n'est présent, le mode d'utilisation est activé par l'interface graphique. La syntaxe est :
[-l|-e|-d] file1 file2 ...
Avec:

- -l, ou aucune option, pour afficher l'état de chaque fichier
- -d pour décrypter les fichiers
- -e pour crypter les fichiers

À la fin de l'exécution, le programme envoie les résultats de l'opération demandée.
La sortie du programme est en format de markdown.

8.1.4 Liste des touches de raccourci disponibles

Effacer le texte

Touche	Description
Retour arrière	élimine un caractère à gauche ou élimine le texte sélectionné
Ctrl+Retour arrière	élimine le mot à gauche
Suppr	élimine un caractère à gauche ou élimine le texte sélectionné
Ctrl+T	élimine les mots ou les espaces à droite
Ctrl+Suppr	élimine le mot à droite et tous les espaces suivants jusqu'au début du mot nouvel

Commentaire de plusieurs lignes de texte

Touche	Description
Ctrl+'+'. Dans les claviers italiens on doit appuyer aussi sur la touche [Majuscule]	ajoute ou enlève le caractère de commentaire aux lignes sélectionnées

Positionner le curseur

Touche	Description
Flèche haut	déplace le curseur dans la direction sélectionnée
Flèche bas	
Flèche droite	
Flèche gauche	
Accueil	déplace le curseur au début de la ligne alternativement au début de la ligne et au premier caractère de la ligne
Fin	déplace le curseur à la fin de la ligne
Ctrl+Accueil	déplace le curseur au début du document
Ctrl+Fin	déplace le curseur à la fin du document
Ctrl+Flèche gauche	déplace le curseur d'un mot à gauche
Ctrl+Flèche droite	déplace le curseur d'un mot à droite
Ctrl Entrée	déplace le curseur sur le premier caractère de la ligne suivante

Sélectionner

Touche	Description
[Maj+Accueil]	sélectionne jusqu'à la ligne à partir de la position du curseur
Ctrl+Maj+Accueil	sélectionne jusqu'au début du document à partir de la position du curseur
Ctrl+Maj+Fin	sélectionne jusqu'à la fin du document à partir de la position du curseur
Ctrl+Maj+Flèche gauche	sélectionne le mot ou les espaces à gauche du curseur
Ctrl+Maj+Flèche droite	sélectionne le mot ou les espaces à droite du curseur
Maj+Page précédente	sélectionne la page précédente à partir de la position courante du curseur
Maj+Page suivante	sélectionne la page suivante à partir de la position courante du curseur
Ctrl+W	sélectionne le mot sur lequel le curseur est placé

Ctrl+A	sélectionne le document entier

Sélection rectangulaire

Touche	Description
Alt+ Maj+Page précédente Maj+Page suivante Maj+Flèche gauche Maj+Flèche droite	sélectionne un bloc rectangulaire de code

Tabulations

Touche	Description
Tab	s'il n'y a pas du texte sélectionné, cette touche insère l'espacement entre des caractères, comme défini en ***Options->Tabulations.*** Si plusieurs de lignes ont été sélectionnées, Tab insère l'espacement affiché pour la tabulation relative.
[Maj+Tab]	s'il n'y a pas du texte sélectionné, cette touche déplace le curseur à gauche de l'espacement défini en ***Options->Tabulations.*** Si on a sélectionné une ou plusieurs de lignes, elles sont déplacées à gauche de l'espacement affiché pour la tabulation relative.

Copier et coller

Touche	Description
Ctrl+C Ctrl+Ins	copie le texte sélectionné dans le presse-papiers
Ctrl+X [Maj+Suppr]	élimine le texte sélectionné, que est copié dans le presse-papiers
Ctrl+V [Maj+Ins]	insère le contenu des presse-papiers à partir de la position du curseur
Ctrl+Y	élimine la ligne sur la quelle le curseur est placé et en copie son contenu dans le presse-papiers
Drag'n'drop (avec souris)	traîne le texte sélectionné qui lors qu'on a relâché la souris est déplacé dans sa nouvelle position
Ctrl+Drag'n'drop (glisser-déplacer) (avec souris)	traîne le texte sélectionné qui lors qu'on a relâché la souris est copié dans sa nouvelle position

Annuler/Récupérer

Touche	Description
Ctrl+Z Alt+ Retour arrière	annule la dernière saisie
Ctrl+Maj+Z	restaure la dernière saisie

Recherche et Remplacement

Touche	Description
Ctrl+F3	recherche vers le bas dans le document entier le mot sur le quel le curseur est placé.
Ctrl+Maj+F3	recherche vers le haut dans le document entier le mot sur le quel le curseur est placé.
F3	recherche le terme suivant. La boîte de dialogue **Trouver** doit être fermée.
Maj+F3	recherche le terme précédent. La boîte de dialogue **Trouver** doit être fermée.
Alt+F3	ouvre la boîte de dialogue **Trouver** et il affiche comme texte à rechercher le mot sur lequel le curseur est placé

Affichage erreurs de compilation

Touche	Description
Double click sur l'erreur	positionne le curseur sur la ligne de la fonction GPL dans laquelle l'erreur décrite s'est produite.
F4	positionne le curseur sur la ligne de la fonction GPL dans laquelle l'erreur après la dernière erreur sélectionnée.

Maj+F4	positionne le curseur sur la ligne de la fonction GPL dans laquelle l'erreur qui précède la dernière erreur sélectionnée s'est produite.

Création d'un fichier prototype

Touche	Description
Ctrl+Maj+C	enregistre le texte sélectionné dans le fichier de fonction GPL. Une boîte de dialogue s'ouvre pour insérer le nom à attribuer au fragment du code.

Gestion du pliage

Touche	Description
Ctrl+M	ouvre ou ferme le pliage sélectionné.

8.2 Bibliothèques

Une bibliothèque est un ensemble de fonctions GPL qu'il est possible de rappeler au sein du code GPL Custom et qui ne sont pas liées à une configuration particulière. Les bibliothèques sont très utiles dans la mesure où il est facile de les copier d'une machine à l'autre. Cela permet de ne pas devoir réécrire le code commun lorsque l'on implémente de nouvelles machines. Par exemple, il est possible de créer une bibliothèque de fonctions géométriques et mathématiques.

Les fichiers de bibliothèque sont stockés dans le répertoire system\lib. Ils sont compilés lorsque l'on exécute l'une des commandes suivantes : ***CNC->Initialisation***, ***Fichier->Compiler tout***, ***Enregistrer*** le fichier de bibliothèque o le fichier des variables globales.

Si un nom de fonction ou de variable existant déjà dans une bibliothèque est assigné dans le code GPL d'une machine, celui de la machine aura toujours la priorité lors de la compilation. Si le même nom existe dans deux bibliothèques différentes, pour identifier celui que l'on désire, il convient d'utiliser, dans le code GPL, la syntaxe longue suivante : **nombibliotheque.nomfonction**. Par exemple, si la fonction LongueurSegment existe aussi bien dans la bibliothèque LIBGEO que dans la bibliothèque LIBMAT et que l'on veut utiliser la fonction de la bibliothèque LIBGEO, il faut alors écrire : LIBGEO.LongueurSegment.

Toutes les opérations relatives à une bibliothèque sont gérées par le biais d'une boîte de dialogue. Il est possible de créer une **[Nouvelle]** bibliothèque. Le nom assigné est reporté dans la liste des bibliothèques installées. Il est possible d'importer des bibliothèques préexistantes ou de transformer des fichiers de groupes en bibliothèques, en les rappelant à travers une boîte de dialogue que l'on ouvre avec le bouton **[Importer..]**. Les bibliothèques éliminées avec le bouton **[Supprimer]** se déplacent dans la corbeille de Windows.

Pour modifier le code d'une bibliothèque, sélectionner le bouton **[Modifier]**. La bibliothèque est alors ouverte par l'éditeur GPL. Pendant la rédaction des fonctions de bibliothèque, il convient de se rappeler quelques règles fondamentales :

- Il n'est pas possible d'accéder aux dispositifs, aux fonctions et aux variables de la configuration au sein de laquelle on est en train d'écrire la fonction
- Il est possible de rappeler des fonctions et des variables publiques d'autres bibliothèques
- Les fonctions déclarées à l'intérieur d'une d'une bibliothèque sont définies privées par défaut. Pour qu'elles puissent être rappelées par d'autres fichiers de fonction, il est nécessaire de les déclarer de type PUBLIC.

La possibilité de modifier une bibliothèque est subordonnée au niveau d'accès de celui qui utilise Albatros. Il est possible d'assigner ou de modifier les autorisations d'accès à une bibliothèque en sélectionnant le bouton **[Propriétés]**.

Les variables globales déclarées dans une bibliothèque sont utilisées en Diagnostic dans une section dédiée. L'affichage des éléments de la bibliothèque est lié aux droits d'accès de celui qui utilise Albatros.

8.3 Débogage

8.3.1 Le débogueur

Le débogage est une fonction de Albatros qui permet de suivre pas à pas la succession des instructions d'une tâche GPL. Il permet donc de comprendre et de corriger les éventuelles erreurs logiques et les comportements indésirables du code.

Cette fonction n'est active qu'avec un password d'un niveau égal ou supérieur à celui du Constructeur. Le débogueur permet, par exemple, de :

- assigner des points d'interruption
- interrompre l'exécution d'une tâche et de visualiser la valeur d'une variable
- suivre la séquence d'exécution d'une fonction
- contrôler la valeur prise par une variable locale
- s'assurer que, en cas d'instruction If, la branche choisie est la bonne

Les commandes utilisables avec la fonction débogage sont sélectionnées dans le menu ***Débogage.*** Les principales sont les suivantes :

Continuer	reprend l'exécution d'une tâche bloquée. La tâche poursuivra jusqu'à sa fin, jusqu'à elle n'est pas bloquée de nouveau ou jusqu'à son point d'interruption.
Démarrer	Il démarre le débogage des tâches en exécution
Interrompre maintenant	Il arrête l' exécutions des tâches dont on est en train d'exécuter le débogage. Le curseur se place à la ligne où l'instruction a été interrompue. Une fois bloquée une tâche, il est possible de piloter son exécution et de vérifier l'état des variables locales.
Démarrer fonction	Exécution d'une instruction GPL individuelle. Il faut que la tâche ait été précédemment interrompue.
Démarrer jusqu'à la fin de fonction	Il exécute chaque instruction jusqu'à la première instruction qui vient après de la fonction courante.
Démarrer instruction/fonction	Il exécute une instruction GPL individuelle ou si l'instruction est un appel de fonction, il exécute la fonction entière.
Démarrer jusqu'au curseur	Il exécute les instructions jusqu'à la position du curseur
Terminer le débogage	Il achève les fonctions de débogage. Le fichier de fonctions dans le quel on était en train d'exécuter le débogage est ouvert in mode d'édition.

Pour accéder au débogueur, il est nécessaire d'afficher la liste des tâches en exécution (De menu ***Débogage->Tâches en exécution***) ou la liste de toutes les tâches (De menu ***Débogage->Toutes les tâches***) et d'y sélectionner la tâche que l'on veut déboguer.

Avant d'exécuter le débogage, s'assurer qu'il n'y a pas d'erreurs dans la compilation des fonctions (par exemple : erreurs de syntaxe, variables non déclarées) et que le module que l'on veut déboguer a été initialisé convenablement.

La fenêtre de débogage est semblable à celle de l'éditeur GPL. Toutefois, il n'est pas possible de modifier le code. Le fond de la fenêtre est gris et la ligne en cours d'exécution est présentée en jaune.

Remarque : Il n'est pas possible d'exécuter en même temps le débogage de plusieurs tâches appartenant à un même module.

8.3.2 Tâche en exécution

La commande peut être sélectionnée du menu ***Débogage->Tâche en exécution.*** Cette fenêtre présente la liste des tâches en exécution associées à une machine ou à un module. En sélectionnant une tâche, il est possible de la déboguer ou d'en achever l'exécution. Pour ce faire, utiliser respectivement les boutons **[Débogage]** ou **[Terminer].**

8.3.3 Toutes les tâches

Une boîte de dialogue affiche toutes les tâches qui ont été définies dans le code GPL. Au niveau graphique, elles sont représentées dans une structure arborescente. Lorsque l'on sélectionne une fonction, le fichier dans lequel elle est définie s'ouvre et le curseur se positionne au niveau de sa première instruction. De cette manière, il est possible de programmer des Points d'interruption avant même d'entreprendre l'exécution.
Il est possible de déboguer n'importe quelle tâche et fonction sans paramètres.

Voici la définition de tous les symboles utilisés pour la composition de l'arbre d'exécution des tâches. Un symbole particulier est celui qui indique la fonction récursive, qui indique une fonction qui, à son intérieur, rappelle la fonction d'où elle est appelée.

Symbole	Description
	tâche de la fonction principale de l'Intergroupe
	tâche autorun

tâche générique
tâche real-time
fonction de groupe
fonction de groupe exécutée par des instructions comme ONINPUT, ONFLAG
fonction de bibliothèque
fonction de bibliothèque exécutée par des instructions comme ONINPUT, ONFLAG.
fonction récursive

8.3.4 Appels de fonction

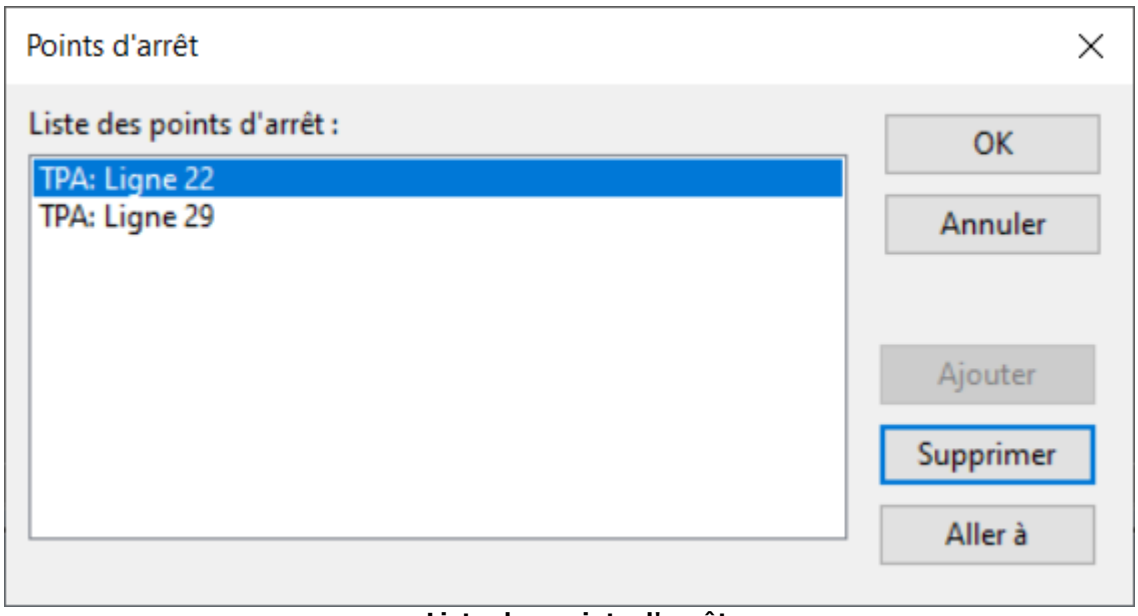
Pendant le débogage, il est possible de visualiser la liste des fonctions qui ont été appelées mais qui ne sont pas encore rentrées (c'est-à-dire, toutes les fonctions dont l'instruction FRET n'a pas encore été exécutée). Une boîte de dialogue énumère les appels de fonction qui ont abouti à l'instruction courante. La fonction qui se trouve en haut de la liste est celle qui a été exécutée le plus récemment.

Pour observer le comportement d'un appel de fonction :

- Mettre le curseur sur la position désirée à l'intérieur de la fonction.
- Sélectionner ***Débogage->Démarrer jusqu'au curseur*** de façon à mettre l'exécution du programme à la position voulue.
- Sélectionner ***Débogage->Appels*** de fonction, ou la touche de raccourci Ctrl+K.
- Dans la boîte de dialogue Appels, il est possible de sélectionner le nom d'une fonction. Le curseur se déplace sur la première instruction de la fonction sélectionnée.

8.3.5 Points d'interruption

Un point d'interruption permet de voir de façon plus détaillée les différentes étapes de l'exécution des instructions, d'examiner ou de modifier les variables et les dispositifs, d'examiner la liste des appels de fonction, etc.
L'exécution d'une tâche se bloque lorsque l'instruction où le point d'interruption a été inséré est atteinte. Les points d'interruption peuvent être programmés avant qu'une tâche donnée ne soit mise en exécution ou bien pendant son exécution (du menu ***Débogage->Points d'arrêt***). Il est également possible d'éliminer des points d'interruption lorsqu'ils ne sont plus nécessaires.

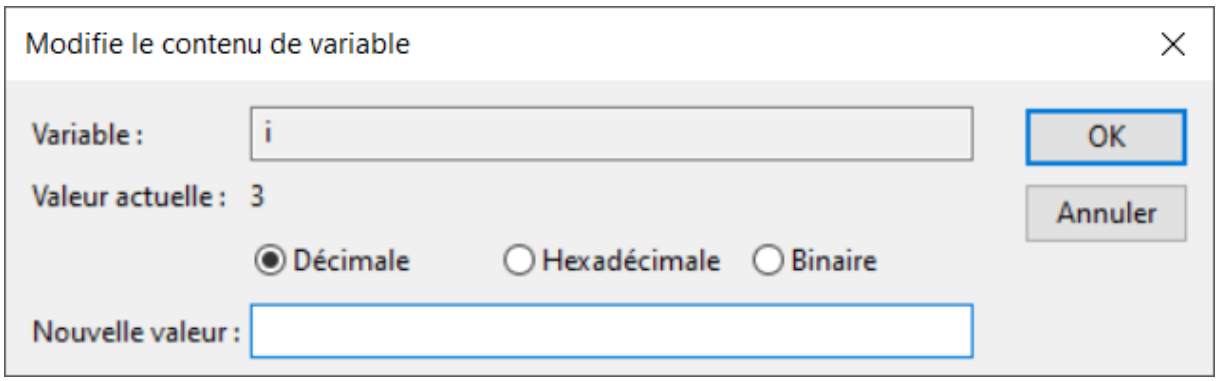


Liste des points d'arrêt

Dans certaines situations, bien que l'on n'ait pas introduit de points d'interruption, la tâche ne se bloque pas dans la mesure où l'exécution n'atteint jamais le point d'interruption. Dans de tels cas, il est possible de bloquer la tâche avec la commande ***Débogage->Interrompre maintenant.*** Le curseur se positionne alors sur l'instruction GPL qui était sur le point d'être exécutée au moment de l'interruption.

8.3.6 Contenu de variable

La commande peut être sélectionnée du menu ***Débogage->Contenu de variabile.***
Après avoir interrompu l'exécution d'une tâche, il est possible de visualiser :
- la valeur des variables locales déclarées dans la fonction où la tâche est immobilisée
- les variables globales
- la valeur assumé par une expression
- l'état des dispositifs et des paramètres type dispositif.



Visualisation/Modification du contenu d'une variable

Si la variable (ou le dispositif) n'est pas en lecture seule, son contenu est modifiable. Naturellement, toute modification aura une influence sur l'accomplissement de la tâche.
La modification de la valeur d'une variable ou d'un dispositif permet d'essayer l'exécution dans des conditions différentes de celles de l'exécution normale, de corriger les éventuelles erreurs se présentant et de poursuivre l'exécution des instructions suivantes.
Il est possible d'afficher le contenu d'une variable, d'un dispositif ou d'une constante, même si vous déplacez la souris sur la variable, sur le nom du dispositif ou sur la constante. Un tooltip apparaît, dans le quel le type, le nom et la valeur de la donnée sont affichés. Si vous sélectionnez une expression, le résultat apparaît. Si le curseur de la souris se trouve dans une sélection, la solution entière est utilisée, autrement seulement le mot sur le quel le curseur de la souris est placé. Si le curseur de la souris et se trouve pas dans un mot, on utilise l'argument entier.
Par exemple, pour voir la valeur de la matrice Mx[3][colonne], si le curseur de la souris est placé sur "3" dans le tooltip s'affiche 3; si le curseur est placé sur "colonne", s'affiche la valeur de la colonne; s'il est placé sur "matrice" il n'affiche rien; s'il est placé sur un crochet, la valeur de Mx[3][colonne] est affichée.

8.3.7 Liste des touches de raccourci disponibles

Pour activer les commandes du **Débogage** il est possible de sélectionner les options du menu ***Débogage*** ou taper directement sur le clavier.
Les touches à utiliser sont les suivantes :

Touche	**Description**
Ctrl+F5	ouvre le boîte de dialogue avec la liste des tâches en exécution.
Ctrl+Maj+F5	ouvre le boîte de dialogue avec la liste des toutes les tâches.
Ctrl+B	ouvre le boîte de dialogue avec l'insertion ou l'effacement des point d'interruption.
Ctrl+F9	insère ou élimine un point d'interruption sur la ligne ou le curseur est placé.
Ctrl+K	ouvre la boîte de dialogue pour afficher la liste des fonctions qui ont été appelées, mais qui ne sont pas encore retournées.
Maj+F9	ouvre une boîte de dialogue pour l'affichage du contenu d'une variable.
F8	exécute l'instruction. Si cette-ci est une fonction, elle entre dans la fonction.
Maj+F7	exécute toutes les instructions de la fonction.
F10	exécute l'instruction. Si cette-ci est une fonction, elle l'exécute sans y entrer.
F7	exécute toutes les instructions jusqu'a l'instruction sur laquelle le curseur est placé. Le curseur doit être placé sur une instruction à l'intérieur de la fonction.
Alt+Interr	interrompt l'exécution du code de la dernière instruction exécutée.
F5	réactive l'exécution du code après une interruption.
Maj+F5	termine la tâche courente et la exécute de nouveau.
Alt+F5	termine le débogage.

8.4 Initialisation du contrôle

8.4.1 Connexions de réseau

N.B. Le fonctionnement de Albatros dans la machine est protégé par un clé matérielle usb, configurée par TPA S.r.l.
La commande peut être sélectionnée du menu ***CNC->Connexions du réseau.*** Cette fenêtre affiche l'état des modules à distance connectés au système. Si un module n'est pas connecté, le symbole qui le représente est marqué d'un X rouge.
Deux champs sont présents pour chaque module. Le premier champ est le nom du module et le troisième est le nom de la station de réseau. Habituellement, ce dernier champ a les premiers caractères fixes "TPA" suivis du numéro de série du module à distance.

Assignation d'un nœud de réseau (module à distance) à un module logique
Il est possible d'assigner un nœud de réseau à un module en sélectionnant l'expression "Non configuré" ou en appuyant la touche **[Modifier]**. Quelques secondes plus tard, une fenêtre fait apparaître la liste des modules à distance disponibles sur le réseau (chaque module à distance doit être allumé et il doit avoir correctement reçu une adresse IP).

Il est alors possible de sélectionner le nœud de réseau que l'on veut lier au module logique et de confirmer le choix en appuyant sur le bouton  .

Cette opération peut être accomplie avec le mot de passe à niveau "Maintenance", sans qu'il soit nécessaire d'accéder à la configuration de Système de Albatros qui, en revanche, demande le mot de passe à niveau "Constructeur".

Mise à jour du logiciel d'un module à distance

En sélectionnant **[Mise à jour]** vous pouvez mettre a jour totalement le logiciel du contrôle qui se trouve dans le stockage interne du module à distance. Avant de demander la mise à jour, il est nécessaire de verifier que le module à distance soit connecté avec Albatros.

8.4.2 Diagnostic du matériel

La commande peut être sélectionnée du menu ***Cnc->Diagnostic du matériel***. Le Diagnostic du matériel affiche la liste et l'état des modules configurés, des cartes et des nœuds qui leur appartiennent, tels qu'ils sont définis dans la configuration matérielle. Si le symbole de la carte ou d'un nœud est marqué d'un X rouge, cela signifie que la carte n'a pas été détectée dans le matériel présent sur le contrôle ou qu'il n'a pas été possible de l'initialiser correctement. S'il est marqué d'un point d'interrogation jaune, cela signifie que le système a détecté la présence d'une carte ou d'un nœud, mais que cette dernière ne correspond pas au type défini dans la configuration.

Topologie du réseau EtherCAT

Dans la fenêtre de diagnostic du matériel, lorsqu'un nœud du réseau EtherCAT est sélectionné dans l'*arborescence*, le bouton **[Détails]** est activé, ce qui permet à son tour d'afficher la topologie du réseau EtherCAT sous forme graphique.

Ce graphique affiche les informations des nœuds physiquement présents sur le réseau : l'état de chaque nœud, l'état des axes si le nœud est un servocommande et la qualité de la communication. Chaque nœud et chaque axe est représenté par un rectangle, dont la couleur définit le statut.

Lorsque vous déplacez le curseur de la souris sur le rectangle, une info-bulle apparaît, décrivant l'état du nœud et les erreurs de communication ou l'état de l'axe.

Affichage et modification des objets dans les nœuds

Dans la fenêtre de diagnostic du matériel, lorsqu'un nœud du réseau EtherCAT est sélectionné dans l'arbre, le bouton [**Objet du dictionnaire**] est activé pour afficher et modifier les objets du nœud. La modification des données n'est possible qu'au niveau du **Constructeur**.

Les objets son regroupés en fonction de leur adresse :

Adresse initial	Adresse final	Nom de la zone

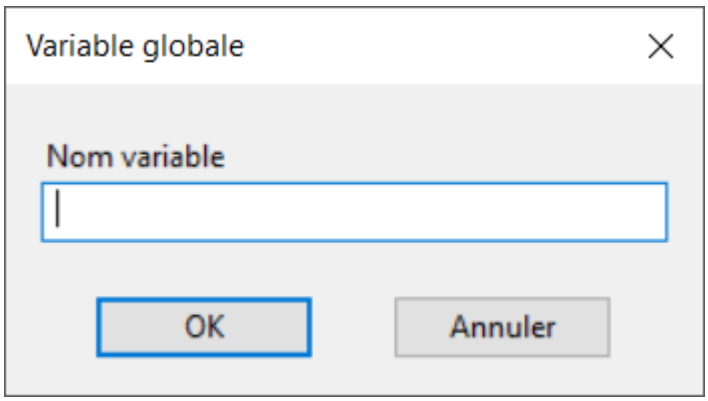
0x0000	0x0FFF	Data type area
0x1000	0x1FFF	Communication area
0x2000	0x5FFF	Manufacture specific area
0x6000	0x6FFF	Input area
0x7000	0x7FFF	Output area
0x8000	0x8FFF	Configuration area
0x9000	0x9FFF	Information area
0xA000	0xAFFF	Diagnosis area
0xB000	0xBFFF	Service transfer area
0xC000	0xEFFF	Reserved area
0xF000	0xFFFF	Device area

Les valeurs numériques entières sont affichées en décimal et en hexadécimal. Lors de la modification de leur valeur, le nombre peut également être saisi en hexadécimal, en utilisant la notation $...H, et en binaire, avec $...B. (comme pour le GPL).

8.5 Test

8.5.1 Enregistrer une variable globale

La commande peut être sélectionnée du menu ***Test->Impression variable globale sur le disque.*** Enregistrement du contenu d'une variable globale sur disque sous la forme d'un fichier de texte formaté. Le nom du fichier est composé par *nomvariable*.txt et il peut être enregistré dans le répertoire *Report*. L'opération n'est possible que si le niveau d'accès en lecture de la variable globale est compatible avec le niveau d'accès courant.



Enregistrement d'une variable globale

8.5.2 Démarrer une fonction

La commande peut être sélectionnée du menu ***Test->Démarrer la fonction.***
Elle démarre une fonction de façon indépendante par rapport au reste du système. La tâche qui est créée commence son exécution par la fonction sélectionnée dont elle prend le nom. Les fonctions exécutables sont uniquement celles qui n'ont pas de paramètres en entrée et dont le niveau d'accès en lecture est compatible avec le niveau d'accès courant. Si la fonction qui est exécutée est la fonction principale de l'intergroupe, toutes les tâches autorun sont exécutées par la suite.

8.5.3 Importation de messages

Les messages de groupe assignés à l'aide de l'instruction GPL DEFMSG sont stockés dans un fichier au format xmlng. Les messages du fichier peuvent être édités, ajoutés ou supprimés. Pour visualiser les changements dans le code GPL, vous devez utiliser la commande ***Test->Importation messages de groupe,*** qui n'est activée que lorsqu'aucune fenêtre n'est ouverte.
Pour exécuter une importation tout le code GPL doit être compilé sans erreurs. Si il y a des erreurs l'utilisateur est avisé avec le message "Pas tout le code GPL résulte compilé".
Il ne sont pas importés les messages de groupe qui appartiennent au fichiers encodés (Voir chapitre ***Outils de développement->Éditeur GPL->Cryptage***) pour lequel l'utilisateur n'a pas le permis de visualisation en clair.

Il sont importés seulement les messages qui ont été déjà définis dans le code GPL. La modification du texto GPL n'est pas effectuée s'il y a au moins une DEFMSG successive a une instruction IFDEF.
Pensant l'importation il est possible de relever des erreurs quand:

- entre les textes d'un particulier message de groupe, l'identificateur d'une langue est présent plus d'un fois
- n'import quel texte est vide (ou bien : "")
- le nom d' un groupe ou d'une librairie est défini plus d'une fois.

Au terme de l'importation il est exécuté la compilation de tous les modules dans lesquels ont été modifiés groupes ou librairies.

Toutes les langue ci dessous peuvent être utilisées dans le fichiers XMLNG :
"AFK" Afrikaans
"ARA" Arabe
"AZE" Azéri
"BAS" Bachkir
"BEL" Biélorusse
"BGR" Bulgare
"BSB" Bosnien (alphabet latin)
"BSC" Bosnien (alphabet cyrillique)
"BRE" Breton
"CAT" Catalan
"CHS" Chinois simplifié
"CHT" Chinois traditionnel
"COS" Corse
"CSY" Tchèque
"CYM" Gaélique
"DAN" Danois
"DEA" Allemand (Autriche)
"DEU" Allemand (Allemagne)
"ELL" Grec

"ENG" Anglais
"ENU" Anglais (États-Unis)
"ESP" Espagnol
"ETI" Estonien
"EUQ" Basque
"FAR" Persan
"FIN" Finnois
"FRA" Français
"FPO" Filipino
"FRB" Français (Belgique)
"FYN" Frison
"GLC" Galicien
"HAU" Haoussa
"HEB" Hébreu
"HRB" Croate (Bosnie-Herzégovine)
"HRV" Croate (Croatie)
"HUN" Hongrois
"IBO" Ibo
"IND" Indonésien
"IRE" Irlandais
"ISL" Islandais
"ITA" Italien
"JPN" Japonais
"KAL" Groenlandais
"KOR" Coréen
"SAH" Iakoute
"KYR" Kirghize
"LVI" Letton
"LTH" Lituanien
"LBX" Luxembourgeois
"MNN" Mongol
"NON" Norvégien Nynorsk
"NOR" Norvégien Bokmål
"NLB" Néerlandais (Belgique)
"NLD" Néerlandais (Pays-Bas)
"OCI" Occitan
"PLK" Polonais
"PTB" Portugais (Brésil)
"PTG" Portugais (Portugal)
"RMC" Romanche
"ROM" Roumain
"RUS" Russe
"SKY" Slovaque
"SLV" Slovène
"SQI" Albanais
"SRM" Serbe (alphabet latin, Serbie)
"SRN" Serbe (alphabet cyrillique, Bosnie-Herzégovine)
"SRO" Serbe (alphabet cyrillique, Serbie)
"SRP" Serbe (alphabet latin, Monténégro)
"SRQ" Serbe (alphabet cyrillique, Monténégro)
"SRS" Serbe (alphabet latin, Bosnie-Herzégovine)
"SVE" Suèdois
"TAJ" Tadjik
"TRK" Turque
"TTT" Tatar
"TUK" Turkmène
"UKR" Ukrainien
"URD" Ourdou
"UZB" Ouzbek
"VIT" Vietnamien
"WOL" Wolof
"XHO" Xhosa
"YOR" Yorouba
"ZUL" Zulu

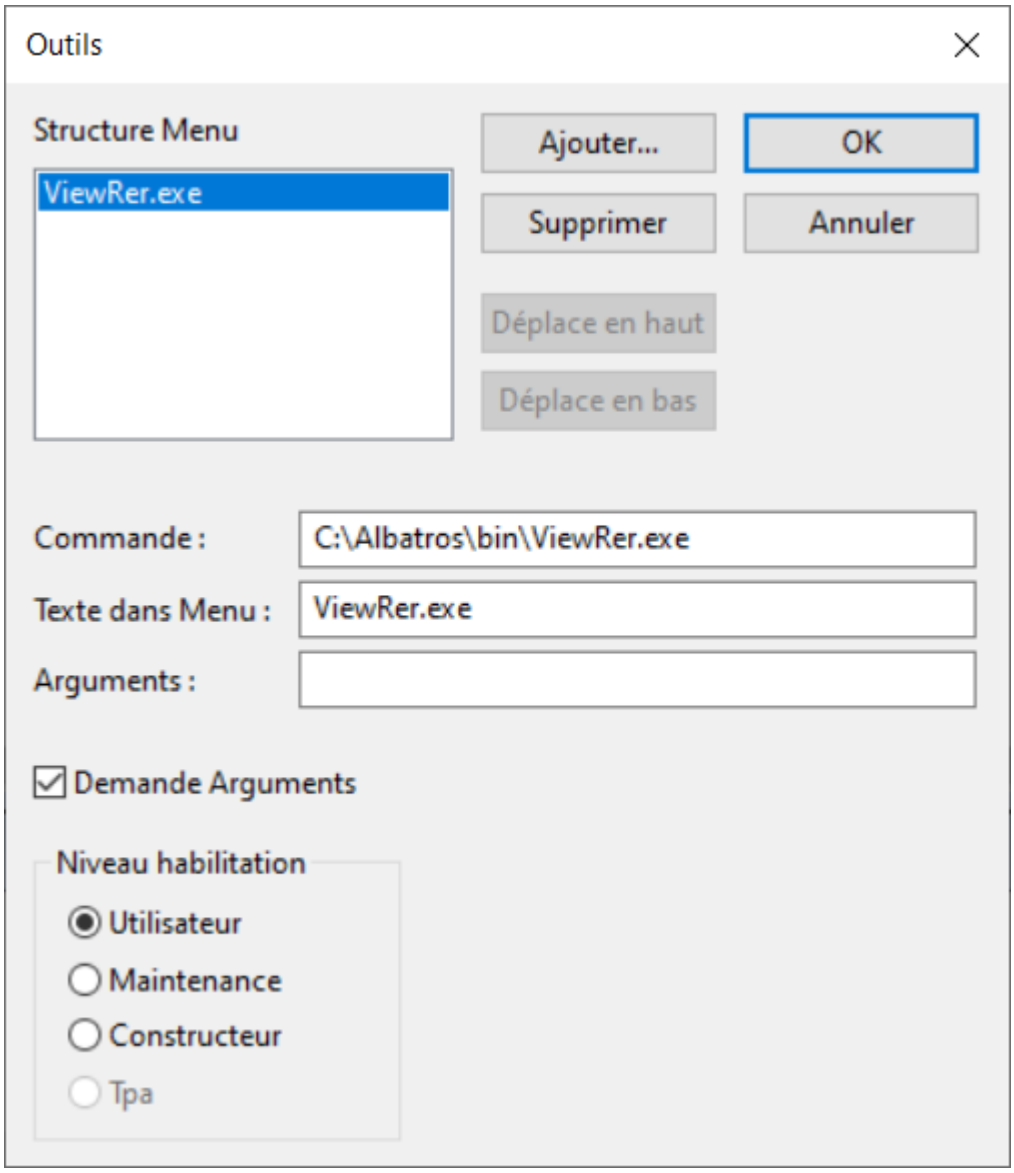
8.5.4 Note de l'utilisateur dans le fichier de rapport d'alarme

Grâce à la commande **Insérer une remarque** du menu **Test** vous pouvez entrer un rapport dans le fichier de rapport d'alarme du mois en cours (MONTHxx.TER). L'élément de menu est activé avec un niveau de mot de passe égal ou supérieur à l'assistance.

8.6 Outils

8.6.1 Personnaliser...

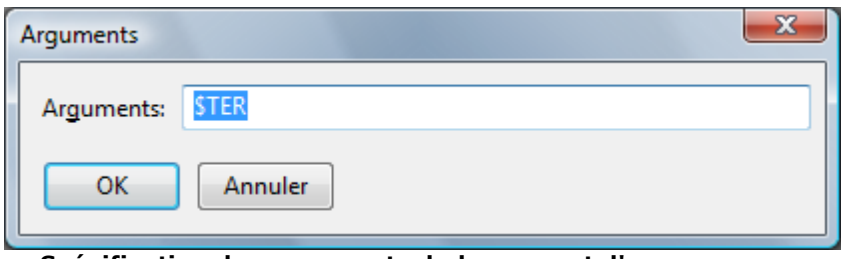
La commande peut être sélectionnée du menu ***Outils->Personnaliser...***
Cette fonction permet d'introduire un nombre maximum de 20 programmes dont l'exécution pourra être lancée dans le menu ***Outils*** de Albatros.



Configuration du menu Outils

Structure Menu :	énumère les programmes affichés dans le menu Outils.
Commande :	Nom du programme qui doit être exécuté. En option, il est également possible de définir le répertoire dans lequel est stocké le programme, s'il différent du répertoire d'où l'on a exécuté Albatros ou des répertoires où le système d'exploitation les fichiers exécutables (variable d'environnent windows PATH).
Texte dans menu :	C'est le nom que l'on veut faire apparaître dans le menu Outils pour identifier le programme exécutable.
Arguments :	toute combinaison d'arguments de la ligne de commande dont le programme a besoin pour garantir une exécution correcte. Il est possible d'insérer arguments dynamiques. Par exemple en utilisant la chaîne $TER en exécution de ViewRER le fichier de report du mois courant se ouvre. Voici la liste des arguments :
$File	Path name complet du fichier courant
$FileName	Nom fichier et extension du fichier courant.
$FileDir	Disque et répertoire du fichier courant
$Ter	Chemin complet du nom du fichier de report erreurs du mois en cours.
$DirModule	Disque et répertoire contenant MODx du fichier courant.
$Module	Disque et répertoire contenant les exécutables de Albatros numéro module du fichier courant.
$Bin	Disque et répertoire contenant les exécutables de Albatros
$TpaIni	Chemin complet du fichier d' initialisation TPA.INI
$ReqDirModule	Chemin (disque et répertoires) du module d'Albatros. Si plusieurs modules sont configurés, la boîte de dialogue du module s'ouvre
$ReqModule	Numéro du module d'Albatros. Si plusieurs modules sont configurés, la boîte de dialogue du numéro de module s'ouvre
$ReqFile	Nom du fichier. Une fenêtre s'ouvre pour demander le nom du fichier. Le fichier sélectionné sera passé parmi les arguments du programme qui sera executé.

Demande arguments : si cette option est sélectionnée, chaque fois que l'exécution du programme est demandée, une boîte de dialogue apparaît pour que l'on introduise les arguments, différents de ceux que l'on a insérés dans la fonction Arguments et pouvant varier en fonction du mode de lancement du programme.



Spécification des arguments de lancement d'un programme

Niveau habilitation : indique le niveau de visualisation du programme dans le menu ***Outils***. Pour les programmes de test ou de modification de données de Albatros on attribue normalement un niveau Constructeur. Les programmes pour l'édition d'usinages de machine sont plutôt associés à un niveau d'habilitation utilisateur.

L'édition de certains champs peut également se faire en utilisant le bouton **[Ajouter...].** Ce dernier permet d'ouvrir la boîte de dialogue **Ajouter Outil** pour sélectionner le programme qui doit être exécuté. Les types de fichiers exécutables sont .EXE, .COM, .PIF, .BAT.

Lorsque l'on ferme la boîte de dialogue après avoir confirmé des données, le programme est inséré dans la fenêtre **Structure Menu,** le nom du programme et du répertoire auquel il appartient apparaît sur la ligne **Commande**.
Les autres boutons présents sont **[Supprimer], [Déplace en haut], [Déplace en bas]** et ils sont utilisés respectivement pour effacer un programme et reclasser la liste des programmes.

8.7 Fureteur

8.7.1 Le fureteur

Le fureteur est un système de Albatros qui utilise les informations générées par le compilateur pour créer une base de données permettant de rechercher rapidement les symboles définis dans les fonctions.
Cette fonction n'est active qu'avec un mot de passe d'un niveau égal ou supérieur à Constructeur. Les commandes utilisables sont sélectionnables dans le menu ***Débogage*.**
Le fureteur permet de :

- positionner le curseur à la ligne où une fonction, une variable ou une constante de module, de groupe ou de bibliothèque est définie pour la première fois (de menu ***Débogage->Aller à la définition***)
- positionner le curseur sur les lignes où est référencée une fonction, un dispositif, une variable de module ou de groupe, une instruction GPL (à l'exception des instructions FCALL et FRET); à partir de ***Débogage->Aller à la référence***, pour afficher la référence précédente ou suivante sélectionner du menu respectivement les options ***Débogage->Précédent*** ou ***Débogage->Suivant***).

Les variables de groupe sont gérées uniquement dans la fenêtre d'édition du groupe auquel elles appartiennent.
Pour actualiser un fureteur lorsque l'on passe à une nouvelle version, il est conseillé d'enregistrer d'abord les variables globales et puis d'exécuter la commande ***Fichier->Compiler Tout.***
Pendant la phase d'édition des fonctions la correspondance entre le texte et les symboles recherchés est perdue, puis elle est rétablie lors du stockage.

8.7.2 Chercher l'identificateur

La commande peut être sélectionnée du menu ***Débogage->Chercher identificateur.*** Une boîte de dialogue s'ouvre pour que l'on y introduise le nom du symbole à rechercher dans le code GPL. En fonction du **Type de recherche** programmé, c'est la définition ou la première référence du symbole qui est identifiée.

Le nom inséré peut présenter les caractéristiques suivantes :

- il ne contient aucun caractère '.' (point) : il est recherché dans tous les fichiers de fonction ;
- il contient un seul caractère '.' (point) : le nom qui précède le point est considéré comme étant le nom du groupe et le symbole n'est recherché que dans ce groupe. Par exemple, s'il existe deux fonctions appelées VisError, une dans le groupe Main et l'autre dans le groupe ASSI, si l'identificateur que l'on écrit est ASSI.VisError, le curseur se positionne sur la première ligne de la fonction VisError du groupe ASSI ;
- il contient deux caractères '.' (point) : le nom qui précède le premier point est le nom du groupe, celui qui précède le deuxième point est le nom du sous-groupe et le symbole est alors recherché uniquement dans ce sous-groupe ;
- s'il s'achève par le caractère '*' (astérisque), tous les symboles qui commencent par les caractères qui précèdent l'astérisque sont pris en ligne de compte.

En cas d'ambiguïté dans la recherche du symbole, une boîte de dialogue présente tous les symboles ayant le nom demandé. Cette dernière permet ainsi de choisir le symbole désiré.

Voici la signification des symboles particuliers qui sont utilisés pour la sélection de l'identificateur.

Symbole	**Description**
	instruction GPL
	constante de module, de groupe ou de bibliothèque
	variable de module ou de groupe
	variable de bibliothèque
	vecteur de bibliothèque
	matrice de bibliothèque
	fonction de bibliothèque
	message de groupe
	étiquette
	variable locale

vecteur local
matrice locale
paramètre simple
paramètre vecteur
paramètre matrice

8.7.3 Liste des touches de raccourci disponibles

Pour activer les commandes du fureteur, il est possible de sélectionner les options du menu ***Débogage*** ou taper directement sur le clavier.
Les touches à utiliser sont les suivantes :

Touche	**Description**
F2	positionne le curseur sur la ligne où le symbole sélectionné est défini. Si plusieurs symboles ayant le même nom sont définis dans la base de données du fureteur, une boite de dialogue permet à l' utilisateur de choisir le symbole demandé.
Maj + F2	positionne le curseur à la première référence du symbole sélectionné. En cas d'ambiguité, la boite de dialogue s'ouvre, permettant de sélectionner le symbole desirè.
Ctrl+F2	ouvre une boite de dialogue permettant de sélectionner le symbole choisi.
Ctrl+'+'ou Ctrl+Page Précédente	positionne le curseur à la référence suivante (utiliser le '+' du pavé numérique).
Ctrl+'-' ou Ctrl+Page Suivante	positionne le curseur sur la référence précédente (utiliser le '-' du pavé numérique).

9 Programmes accessoires

9.1 XConfMerge : programme pour la fusion du fichier de configuration

XConfMerge est un outil qui fusionne les fichiers de configuration.
Il fonctionne à partir de la ligne de commande du répertoire bin, car XConfMerge lit le fichier tpa.ini.

Les fichiers lus par XConfMerge sont les suivants :

- hardware.xconf: contient les données du virtuel-physique et les données de la configuration matérielle (physiques)
- devices.xconf: contient les données de la configuration des groupes, sousgroupes et dispositifs (logiques)
- devices.xmlng: contient les messages que l'on peut traduire. En tout cas, tous les fichiers linguistiques présents dans le dossier sont considérés.
- addresses.xdb: contient les adresses logiques des dispositifs.

Les arguments à passer sont les suivants :

- le dossier dans lequel lire les fichiers nouveaux
- le numéro du module auquel le la personnalisation se réfère. Si aucun numéro est indiqué, 0 est indiqué comme valeur par défaut.

Le chemin de lecture du fichier à mettre à jour, qui est ensuite le même chemin pour écrire le fichier de fusion, est déduit de l'ensemble des données dans tpa.ini

Attention : les fichiers à mettre à jour sont écrasés et aucune sauvegarde automatique n'est effectuée.

Règles de fusion des fichiers de configuration des groupes :

1. Si le même groupe, sous-groupe ou dispositif existe dans les deux fichiers, les données et l'activation de l'ancien fichier seront conservées.

2. Si le groupe, le sousgroupe ou dispositif n'existe que dans le fichier nouveau, les données et les activations seront copiées.

3. Si le groupe, le sousgroupe ou dispositif n'existe que dans le fichier ancien, ceci est supprimé.

Règles de fusion du fichier de configuration matérielle et du virtuel-physique :

1. si le même matériel existe dans les deux fichiers, le matériel et le virtuel physique du nouveau fichier se maintiennent avec les activations définies dans l'ancien fichier.

2. Si le matériel n'existe que dans le fichier nouveau, le nouveau matériel avec son activation et le nouveau virtuel physique se maintiennent.

3. Si le matériel n'existe que dans l'ancien fichier, il est supprimé avec le virtuel physique.

Règles de fusion du fichier des adresses logiques :

1. si le fichier ne se trouve pas dans le dossier des fichiers à importer, alors les adresses logique de la personnalisation sont conservées et de nouvelles adresses sont attribuées aux nouveaux dispositifs, s'il y en a.

2. Si le fichier se trouve dans le dossier des fichier à importer, alors les adresses logiques qu'il contient sont lues et utilisées.

Règles de fusion des fichiers des messages :

1. Aucune fusion n'est effectuée, mais le nouveau fichier de message est copié dans le dossier du module.

À la fin de l'exécution, l'outil de fusion renvoie les valeurs suivantes :

0	tout va bien
1	la fusion des fichiers a échoué
2	aucun argument n'a été défini
3	le numéro de module passé en argument est erroné
4	le dossier indiqué comme premier argument n'existe pas

9.2 XParMerge : programme pour la fusion de deux fichiers de paramètres

XParMerge est un outil qui fusionne les deux fichiers paramétriques.

On l'exécute à partir de la ligne de commande du répertoire bin, car XParMerge lit le fichier tpa.ini. Les arguments à passer sont les suivants : le nom du nouveau fichier et le numéro du module auquel il se réfère.

Le chemin de lecture du fichier à mettre à jour et le chemin d'écriture du fichier, obtenus avec la procédure de fusion, sont déduits de l'ensemble des données dans tpa.ini.

Attention : aucune sauvegarde automatique de l'ancien fichier paramétrique n'est effectuée, mais celui-ci est écrasé.

Les règles de fusion des fichiers de Paramètres Technologiques sont les suivants:

1) si le même contrôle existe dans les deux fichiers, la valeur de l'ancien fichier se maintient, mais les autres paramètres qui le définissent (désactivé, visible, nom de variable GPL...) seront mis a jour, en les prenant dans le nouveau fichier ;

2) si un contrôle qui n'existe pas dans l'ancien dossier est défini dans le nouveau dossier, le contrôle se maintient ;

3) si dans l'ancien fichier est défini un contrôle qui n'existe pas dans le nouveau fichier, le contrôle est supprimé.

Règles pour fusionner les fichiers de Paramètres des Outils :

1) si le dialogue de référence du nouveau fichier est différent du dialogue de référence de l'ancien fichier, tous les dialogues du nouveau fichier sont conservés, le cas échéant, et ceux de l'ancien fichier sont mis à jour (nouveaux contrôles supprimés ou ajoutés) ;

2) si le dialogue de référence du nouveau fichier est le même que le dialogue de référence de l'ancien fichier, les dialogues du nouveau fichier sont ajoutés aux dialogues de l'ancien fichier ;

3) si le dialogue de référence n'est présent que dans l'ancien fichier, celui-ci et ses dialogues sont supprimés.

10 Langage GPL

10.1 Concepts de base

10.1.1 Introduction au langage GPL

Le langage GPL (General Purpose Language) est le langage utilisé pour la création des fonctions du système Albatros.
Sa structure est partiellement semblable à celle du langage BASIC, mais il se caractérise par la présence de nombreuses instructions dédiées au contrôle des dispositifs.
Le langage se compose de plus de 200 instructions qui, pour simplifier leur utilisation, sont réunies en groupes composés d'instructions ayant des fonctions similaires.
Le langage est également multitâche, c'est-à-dire qu'il peut exécuter simultanément plusieurs tâches.

Syntaxe typique d'une instruction GPL
Toutes les instructions GPL ont une structure semblable qui reflète le schéma suivant :

nominstruction paramètre-1, paramètre-2, ... paramètre-N

Le nombre de paramètres réellement présents dépend de l'instruction et du contexte dans lequel elle est utilisée. En général, le nombre maximum de paramètres admis pour une instruction ou fonction GPL est de 120. Dans certains cas, il peut même n'y avoir aucun paramètre.

Le bloc minimum de code GPL est la fonction.

Partage du code en groupes
Le code GPL est partagé en blocs qui reflètent la répartition logique de la machine en groupes. Pour chaque groupe, il y a donc un fichier contenant le code qui lui est associé. Les fichiers qui contiennent le code des groupes présents dans la machine se complètent d'un fichier de variables et de constantes globales, visibles avec le code GPL de tous les groupes, et les bibliothèques. Ces dernières contiennent un code délié de la configuration de la machine et qu'il est donc facile de déplacer d'une machine à une autre.

10.1.2 Conventions et terminologie

Principaux termes adoptés

ARGUMENT	C'est l'un des arguments de l'instruction; il peut être défini en tant que *constante*, *variable* ou *paramètre*, en fonction du type d'instruction. S'il est inscrit entre crochets (**[ ]**), cela signifie qu'il peut être omis, ce qui implique bien entendu que l'instruction sera effectuée différemment.
MOT CLÉ	C'est un argument à choisir parmi des arguments ayant une valeur prédéfinie et généralement présentée en majuscules ; la liste des mots clé est fournie dans une page d'aide dédiée.
PARAMETRE	C'est l'argument d'une instruction qui n'est pas définie à l'intérieur de l'instruction elle-même, mais qui est passée, justement en tant que paramètre, à la fonction lorsque cette dernière est exécutée ; on l'appelle également parfois *argument paramétré.*
CONSTANTE	C'est un argument défini de façon fixe au moyen de la métacommande CONST ou qui est fixé de façon rigide au sein de l'instruction.
VARIABLE	C'est un argument qui a été défini en tant que variable globale de module ou de groupe ou au moyen de l'instruction LOCAL. Il peut s'agir d'une variable simple, d'un vecteur ou d'une matrice. Voir Les variables
PARAMÈTRE DE CONFIGURATION	C'est un argument qui a été défini dans la configuration ; par exemple, les paramètres d'un axe.

Arguments les plus fréquents dans la description des instructions

Voici la liste des termes relatifs aux arguments les plus fréquents dans la syntaxe des instructions GPL, suivis d'une description succincte. Lorsqu'un argument peut prendre des valeurs différentes de celles qui sont mentionnées ci-dessous, sa description est reprise dans la section *Arguments* de la page d'aide de l'instruction.

nomentrée	nom de dispositif d'entrée numérique
nomsortie	nom de dispositif de sortie numérique
nomflag	nom de dispositif flag switch ou flag bit
nomport	nom de dispositif port d'entrée, port de sortie ou flag port
nomtemporisateur	nom de dispositif temporisateur
nomcompteur	nom de dispositif compteur
nomfonction	nom d'une fonction (également valable comme paramètre dispositif dans le cas de ERRSYS.)
nomsousprogramme	nom d'un sous-programme, il équivaut à *étiquette,* à laquelle nous renvoyons pour l'explication ; l'appel en sous-programme se fait au moyen de l'instruction "CALL nomsousprogramme".
axe	nom d'un axe
constante	un caractère ou un nombre entier ou double ou un mot clé
valeur	constante ou variable (le *type* dépend de l'instruction)
variable	nom de : variable, élément de vecteur ou élément de matrice
variabledevice	nom de *paramètre dispositif*
matrice	nom d'une matrice
vecteur	nom d'un vecteur
étiquette	nom de l'étiquette de saut ou le nom d'un sous-programme
état	état logique, peut être ON ou OFF ou bien 1 ou 0
timeout	quantité de temps pendant lequel un certain événement doit avoir lieu, ou un temps de retard (constante ou variable)
cote	valeur de la cote (constante double ou variable double)
rayon	valeur du rayon (constante double ou variable double)
angle	valeur de l'angle (constante double ou variable double)
nombtours	nombre de tours (constante double ou variable double)
vitesse	valeur de vitesse (constante float ou variable float)
sens	rotation dans le sens des aiguilles d'une montre ou contraire (variable ou constante : CW ou CCW)
opérande	(constante ou variable ou nomdevice)
résultat	résultat de l'opération (variable ou nomdevice)
nomdevice	nom d'un type de dispositif quelconque (ou paramètre dispositif)
constantestr	succession de caractères contenue dans des guillemets doubles (ex. "chaîne")
variablestr	le nom d'un vecteur de caractères, c'est-à-dire une chaîne
opérateur	opérateurs de comparaison, éventuellement associés les uns aux autres : > (supérieur à) = (égal à) < (inférieur à) ils peuvent être combinés en les approchant l'un de l'autre, ex. >= (supérieur ou égal à)
type	type de constante ou variable: "char" (8 bits), "integer" (32 bits), "float" (32 bits), "double" (64 bits), "string"
paramètre dispositif	c'est une variable qui représente un dispositif. Les dispositifs sont définis dans Configuration.

Principaux termes utilisés pour un axe

cote théorique (ou cible)	Position courante "théorique" imposée, d'instant en instant, par la commande numérique, en fonction de l'algorithme de génération du profil de vitesse.
cote réelle	Position réelle de l'axe tel qu'il détecté par le transducteur de position. Elle diffère de la cote théorique d'une ampleur dite "erreur de poursuite" ou "erreur de boucle".

cote finale	Elle correspond à la cote d'arrivée programmée d'un mouvement. L'algorithme de calcul du profil de vitesse permet à la cote théorique d'atteindre exactement la valeur finale.
fenêtre arrivée cote	Intervalle programmable dont le point central est la cote théorique finale : lorsque la cote réelle se trouve à son intérieur, le mouvement réel est considéré achevé.
fenêtre grande arrivée cote	fenêtre d'arrivée en cote multiplié par un facteur réglable à travers l'instruction SETBIGWINFACTOR.
erreur de poursuite	Différence, instant après instant, entre la cote théorique et la cote réelle d'un axe : elle est normalement proportionnelle à la vitesse de translation et inversement proportionnelle au "gain d'anneau proportionnel".
gain [d'anneau] proportionnel	Paramètre de réglage de l'axe, programmable : il détermine le rapport entre la vitesse courante et l'erreur de poursuite relative.
feed forward	Paramètre de réglage de l'axe, programmable : il détermine un apport direct (proportionnel à la vitesse programmée) donné à la commande de vitesse de l'actionnement. Il permet de réduire, à vitesse et à gain proportionnel identiques, la valeur de l'erreur de poursuite.
feed rate override	Taux de la vitesse programmée. Ce paramètre permet de réduire la vitesse d'exécution par rapport à la vitesse programmée, selon un pourcentage pouvant varier de 0 à 100%.
tolérance	Valeur de déplacement selon laquelle l'axe se détache de sa trajectoire originale dans une interpolation multiaxe à cheval entre deux blocs de déplacement consécutifs.
jeu mécanique	Espace entre creux et dent d'une paire d'engrenages

10.1.3 Les variables

Les variables contiennent des informations. Elles sont utilisées par le langage GPL qui y dépose les valeurs nécessaires à l'accomplissement du programme.
Les variables se caractérisent par un "type" qui reflète les caractéristiques de l'information qui y sera déposée. Chaque variable est également associée à un mode de visibilité qui définit quels sont les ensembles ou les sous-ensembles de code qui peuvent y agir (lire et écrire).

Types de données

DONNÉES SIMPLES ou SCALAIRES
Le GPL supporte les types de données simples et agrégées. Les types de données simples sont semblables à ceux que l'on trouve dans la plupart des langages de programmation :

Char
C'est un nombre entier avec signe compris dans l'intervalle [-128 ; +127] et ayant une longueur égale à 1 octet.
La déclaration d'une variable type char se fait avec la syntaxe suivante :

```
NomVariable as char
```

Integer
C'est un nombre entier avec signe compris dans l'intervalle [-2147483647 ; +2147483647] et ayant une longueur égale à 4 octets (correspond au type long du C).
La déclaration d'une variable type Integer se fait avec la syntaxe suivante :

```
NomVariable as integer
```

Float

C'est un nombre à virgule mobile compris dans les intervalles [-3,402823 E+38 ; -1,401298 E-45] et [+1,401298 E-45 ; +3,402823 E+38] et ayant une longueur égale à 4 octets (utilisé habituellement pour représenter les vitesses).
La déclaration d'une variable type Float se fait avec la syntaxe suivante :

```
NomVariable as float
```

Double C'est un nombre à virgule mobile compris dans les intervalles [-1,79769313486231 E+308 ; -4,94065645841247 E-324] et [4,94065645841247 E-324 ; 1,79769313486231 E+308] et ayant une longueur égale à 8 octets (utilisé habituellement pour représenter les cotes)
La déclaration d'une variable type Double se fait avec la syntaxe suivante :

```
NomVariable as double
```

Ces types de données peuvent être utilisés simultanément dans une même expression. Le GPL exécute une conversion automatique sans donner aucun message d'avertissement. Il convient donc de veiller à ne pas perdre d'informations, ce qui risque de se produire lorsque l'on utilise des types différents de données dans la même expression.
Dans certaines situations, la conversion n'est pas possible. Dans de tels cas, le compilateur donne généralement des avertissements ou des erreurs de système.

DONNÉES AGREGÉES

L'array
C'est un ensemble de variables de type simple, toutes du même type, obtenu en associant un indice al nom de la variable. L'indice est écrit entre crochets. Si l'array s'appelle, par exemple, "paramètres", le premier élément de l'ensemble est indiqué avec "paramètres[1]", le deuxième avec "paramètres[2]" et ainsi de suite.
L'array a un nombre fixe d'éléments qui doit être défini au moment de la déclaration. Une déclaration typique d'un array suit le syntaxe :

```
paramètres[10] as integer
```

Dans laquelle *paramètres[10]* précise que le nom de l'array est "paramètres" et qu'il est composé de 10 éléments, *as integer* définit le type de donnée simple des différents éléments de l'array qui, dans ce cas, est integer.
Les arrays peuvent se constituer de données de type simple ou de chaînes.
Un array peut être composé au maximun par 262144 éléments.
On peut initialiser les vecteurs directement dans le code GPL au moment de leur déclaration. La syntaxe GPL peut être:
[READONLY] vecteur[nombrelignes] as integer = 1,2,3,4
[READONLY] vecteur[nombrelignes] as string = "un","deux","trois","quatre"

Les matrices
Les matrices sono des arrays bidimensionnelles, c'est-à-dire des variables auxquelles sont assignés deux indices. Il est possible d'associer mentalement une matrice à un tableau de données divisées en lignes et en colonnes. Pour indiquer une des cases du tableau, il est possible de définir dans quelle ligne et dans quelle colonne du tableau elle se trouve. Le premier indice de la matrice correspond au numéro de ligne, le deuxième au numéro de colonne.
Contrairement aux arrays, les matrices peuvent contenir des types de données différents les uns des autres. La seule limite est la suivante : il est possible d'utiliser un type de données simple différent pour chaque colonne, mais il n'est pas possible de modifier le type à l'intérieur de la colonne elle-même.
Par exemple, il est possible de définir une matrice dont la première colonne est de type integer et la deuxième de type float. Il n'est pourtant pas possible d'avoir une matrice dans laquelle la première ligne présente un integer et un float et, dans la deuxième ligne, un char et un double. Les lignes doivent être toutes identiques pour ce qui est du type de données des éléments qui les composent.
La déclaration d'une matrice peut se faire avec l'une des syntaxes suivantes :

```
offset[10] as double double double
```

```
dim_pièce[50] as float:longueur float:largeur float:épaisseur
```

Dans le deuxième type de déclaration, une étiquette ou un nom symbolique est attribué à chaque colonne. Les noms symboliques des colonnes s'avèrent très utiles lorsque l'on travaille avec des matrices de grosses dimensions. En effet, dans de telles situations, il est difficile de se rappeler la signification des valeurs enregistrées à l'intérieur de chaque colonne de la matrice. Le nom symbolique nous permet d'identifier immédiatement le type de donnée que l'on est en train de traiter. Par exemple, "Offset[1][3]" est moins clair que "Offset[1].axe_X".

Les matrices ne peuvent contenir que des données de type simple. Par exemple, il n'est pas possible de créer des matrices contenant des chaînes. Une matrice peut se composer de 262144 lignes au maximum.
On peut initialiser les matrices directement dans le code GPL au moment de leur déclaration.
La syntaxe GPL peut être:
[READONLY] nommatrice[nombreligne] as double double integer double = _
1.1, 2.2, 3, 0.1 _
1.2, 3.4, 5, 0.1 _
2.1, 5.6, 6, 0.1

Les chaînes
Les chaînes sont des ensembles de caractères, c'est-à-dire de données de type char qui sont toutefois gérées d'une manière particulière, dans la mesure où elles constituent un texte lisible. Une chaîne est très semblable à un array de char. La principale différence repose sur le fait que chaque chaîne s'achève par un caractère terminateur qui lui est ajouté automatiquement. Le GPL fournit également quelques instructions qui permettent de manipuler les chaînes.
Habituellement, les chaînes sont utilisées pour écrire des messages, lisibles par l'utilisateur, sur l'écran ou dans un fichier de rapport.
La déclaration d'une variable type Chaîne peut se faire avec l'une des syntaxes suivantes :

NomVariable as String

NomVariable as String[20]

Dans le premier type de déclaration, une dimension de 256 octets est attribuée par défaut. Dans le deuxième cas, une dimension maximale est définie pour la chaîne.
Les valeurs de chaîne sont des séquences de caractères entre guillemets.
Pour insérer le caractère '"' (guillemets) il faut l'insèrer deux fois. Exemple : "Appuyer sur le bouton "Démarrer"".
Pour insérer des caractères en utilisant le code numérique, écrivez dans les caractères de la chaîne les caractères \ u suivis de la valeur numérique du caractère hexadécimal. Exemple : \u20ac est le symbol de l'Euro. Si vous écrivez "\u20ac 15,6" vous obtenez € 15,6.

Conversion des données

Dans toutes les expressions mathématiques, à l' exception de l' instruction EXPR, les types de données des opérandes sont converties en le type de données de la variable résultat et ensuite l' opération est exécutée. Il est important de faire très attention à la déclaration des types de données, parce-qu'ils peuvent influencer le résultat. Le tableau suivant est un exemple de comment les résultats peuvent varier selon le type de donnée:

DIV	Opérande 1 (Integer)	Opérande 2 (Double)	Résultat (char)
	3	5.0	0
	5	1.9	5
	1200	107.2	Indéfini
	1200	250.0	Indéfini

Dans l'instruction EXPR, si les opérandes ne sont pas tous du même type, on fait une conversion automatique et le type du résultat de l'opération est identique à celui qui est majeur entre les deux, selon la règle suivante :
- char <integer
- float < double
- char ou integer < float ou double.

Une fois calculée l'expression, le résultat est converti dans le type de la variable résultat.

EXPR	**Opérande 1 (Double)**	**+**	**Opérande 2 (Integer)**	**/**	**Opérande 3 (Float)**	**Résultat (Integer)**
	900.0	+	100	/	400	900

EXPR	**Opérande 1 (Double)**	**+**	**Opérande 2 (Integer)**	/	**Opérande 3 (Float)**	**Résultat (Double)**
	900.0	+	100	/:	400.0	900.25

Déclaration et visibilité des variables

Les déclarations des variables et des constantes ne peuvent être exécutées que dans certains points particuliers du code GPL.
Il est possible de définir les variables suivantes :

- Globales de module
- Globales de groupe
- Locales (uniquement variables)
- Globales de bibliothèque

Le nombre maximum de variables globales (de module et de groupe) déclarables est de 2048.

Il est possible de définir des *modificateurs* qui attribuent des caractéristiques complémentaires aux variables.

Variables globales de module

Les variables globales de module sont contenues dans un file particulier auquel on accède en sélectionnant l'option du menu **Fichier->Ouvrir variables globales**.
Comme nous l'avons vu dans les paragraphes précédents, la déclaration se fait en précisant le nom de la variable, suivi du mot clé "AS", suivi du type de donnée (ou des types de donnée en cas de matrices).
Ces variables sont visibles directement avec le code de tous les groupes.

Variables globales de groupe

Les variables globales de groupe sont définies au début du code relatif au groupe. Elles doivent être déclarées avant les fonctions GPL.
Ces variables sont visibles directement avec tout le code contenu à l'intérieur du groupe. En outre, il est possible d'étendre la visibilité de ces variables à l'extérieur du groupe en les déclarant en tant que variables "Publiques".
Les variables publiques ne sont pas accessibles directement de l'extérieur du groupe. Pour pouvoir y accéder, il est nécessaire d'utiliser leur nom précédé du nom du groupe auquel elles appartiennent.
Par exemple, si l'on veut modifier la variable publique "offset" du groupe "axes" du code du groupe "main", on écrira "SETVAL 10 axes.offset".
Pour déclarer une variable globale de groupe, on utilise la même syntaxe que pour les variables globales de module. La principale différence repose sur la définition des variables publiques.
Pour définir une ou plusieurs variables publiques et privées, on utilise les étiquettes "Public:" et "Private:". Par exemple :

```
Public:
  offset as double
  vitesse as float
Private:
  outil as integer
```

Variables locales

Les variables locales sont déclarées à l'intérieur du corps d'une fonction. Elles doivent être déclarées avant toute autre instruction, à l'exception de la déclaration des paramètres de la fonction.
Les variables locales ne sont accessibles qu'à l'intérieur de la fonction elle-même.
Ces variables ne sont créées avec la valeur 0 (la mémoire nécessaire est allouée) qu'au début de l'exécution de la fonction elle-même et elles sont détruites (la mémoire est relâchée) à la fin de l'exécution. En revanche, les variables globales sont créées à l'initialisation du module et elles sont toujours visibles dans le "Diagnostic".
La déclaration d'une variable locale suit la syntaxe que nous avons déjà vue, mais elle doit être précédée du mot clé "LOCAL".
Par exemple :

```
Function usinage
local    cote_centre as double
movabs X,cote_centre
fret
```

Variables globales de bibliothèque

Les variables globales de bibliothèque sont contenues dans les bibliothèques de code GPL. Elles sont identiques aux variables globales de groupe.

Modificateurs

Modificateurs: READONLY
Les variables globales de module et globales de groupe peuvent être déclarées en tant que READONLY.
Une variable readonly est une variable dont la valeur ne peut pas être modifiées avec le code GPL, mais elle peut être modifiée "de l'extérieur", c'est-à-dire à partir des archives des paramètres technologiques de Albatros.
Les archives des paramètres technologiques sont une base de données où sont conservées des valeurs qui caractérisent la machine, mais qui peuvent changer à long terme à la suite des modifications de la machine en question ou d'opérations d'entretien extraordinaires. Normalement, ces informations sont insérées dans une matrice GPL pendant l'initialisation du contrôle.
Des exemples de ce type d'informations sont constitués par les offsets des zones d'usinage ou par les dimensions et les paramètres technologiques des outils.
Si l'on déclare ces variables en tant que readonly, on se protège des modifications accidentelles des d'informations qui, en règle générale, ne doivent pas changer pendant le fonctionnement normal de la machine.
La dimension maximale d'une variable readonly est de 128 Koctets.
La déclaration d'une variable readonly se fait avec la syntaxe suivante :

readonly NomVariable as type

Modificateurs: NONVOLATILE

Les variables déclarées **NONVOLATILE** sont enregistrées sur une RAM non volatile (munie d'une pile) plutôt que sur une RAM normale. Par conséquent, les valeurs stockées dans ces variables ne sont perdues lorsque l'on éteint la commande numérique.
La déclaration d'une variable non volatile se fait avec la syntaxe suivante:

nonvolatile NomVariable as type

Par exemple:
nonvolatile OffsetAree[2] as double:offsetX double:offsetY double:offsetZ

Seules les variables globales de groupe et de module peuvent être définies "nonvolatile".
La dimension maximale totale des variables enregistrées sur une RAM non volatile est de 65536 octets.
La dimension maximale d'une matrice non volatile est de 1024 octets.

Assignation de RANGE

Lors de la déclaration d'une variable, il est possible de lui attribuer une fourchette de valeurs. Pour l'heure, aucun contrôle du respect des limites n'est effectué lors de l'exécution. Il n'existe qu'un contrôle qui est effectué par le compilateur lorsque l'on attribue des valeurs constantes (par exemple, pour initialiser la variable).
Le principal avantage repose donc sur une sorte d'auto-documentation du code.
La définition des fourchettes se fait avec la syntaxe suivante :

NomVariable Range: valmin..valmax AS type

Par exemple :
NuméroOutil Range: 1..100 as integer

Droits de Lecture / Écriture

Les droits de lecture et écriture permettent de préciser le niveau d'accès minimal au système nécessaire pour en visualiser (droit de lecture) et en modifier (droit d'écriture) la valeur.
La syntaxe est la suivante :

NomVariable Read=S Write=M AS type

Les mots clé utilisés pour indiquer les droits sont :
- READ lecture
- WRITE écriture

Les valeurs attribuables sont les suivantes :
- U ou USER utilisateur

- S ou SERVICE maintenance
- M ou MANUFACTURER constructeur
- T ou TPA tpa

Les défauts des valeurs sont les suivants :
- READ lecture pour maintenance (S ou SERVICE)
- WRITE écriture pour Constructeur (M ou MANUFACTURER) et Tpa (T ou TPA)

10.1.4 Les constantes

Le GPL prévoit quatre types de constantes :
- Integer
- Double
- Char
- String

Les constantes type Char sont déclarées en utilisant les guillemets simples comme ci-dessous :

```
Const COD = 'A'
```

Les constantes de type String sont déclarées en utilisant les doubles guillemets comme ci-dessous :

```
Const MSG = "Début d'usinage"
```

Les constantes de type Integer et type Double sont déclarées avec la syntaxe suivante :

```
Const PI = 3.14
Const MSGBOX = 12
```

Pour les constantes de type integer, il est également prévu une notation binaire et hexadécimale :

```
Const MASK = $11001001b ; binaire
Const MASK = $F5h       ; hexadécimale
```

Les constantes de groupe et de bibliothèque peuvent également être publiques ou privées.
La syntaxe est similaire à celle vue pour les variables.
Exemple :

```
Public :
  Const PI = 3.14
  Const MSGBOX = 12
Private :
  Const MASK = $11001001b
```

REMARQUE : Il n'y a pas de constantes Float. Les valeurs décimales devront évidemment être déclarés comme Double. Parfois, cela peut provoquer des messages d'avertissement de la part du compilateur (lorsque l'on utilise des instructions GPL optimisées pour l'utilisation de types Float).

Vous pouvez definir les constantes à la suite d'expressions de calcul avec la syntaxe suivante :

```
Const a  = 10
Const b =  20
Const c = a + b
```

Les opérateurs admis sont les mêmes que ceux utilisés dans l'instruction EXPR.

Constantes prédéfinies avec valeur préassignée

Le langage GPL prévoit quelques constantes prédéfinies. Ces dernières peuvent être utilisées directement sans qu'il soit nécessaire de les définir.
Les constantes prédéfinies et les valeurs correspondantes sont les suivantes :

ON	**1**
OFF	**0**
UP	**+1**
DOWN	**-1**
POSITIVE	**+1**

NEGATIVE	**-1**
CW	**1**
CCW	**0**
TRUE	**1**
FALSE	**0**
NOWAIT	**0**
WAIT	**1**
WAITACK	**2**
STORE	**1**
NOSTORE	**0**
NOPLACE	**0**
COM1	**0**
COM2	**1**
COM3	**2**
COM4	**3**
COM5	**4**
COM6	**5**
COM7	**6**
COM8	**7**
NOPARITY	**0**
ODDPARITY	**1**
EVENPARITY	**2**

Constantes prédéfinies avec une valeur prédéfinie au démarrage d' Albatros

Le langage GPL fournit quelques constantes prédéfinies dont la valeur est définie au démarrage d'Albatros. Ils peuvent être utilisés dans l'instruction IFDEF.

_ID_MODULE	numéro du module courant. Le numéro de module est compris entre 0 et 15.
_REMOTE_MODULE	Type de module. La constante vaut 1 si le module est distant, vaut 0 si le module est local, et n'est pas définie si le module n'est pas configuré dans la Configuration du système.
_VER_MAJOR	Numéro de la version principale de l'Albatros. Si la version d'Albatros est 3.2.1, la valeur de la version principale est 3.
_VER_MINOR	Numéro de la version secondaire d'Albatros. Si la version d'Albatros est 3.2.1, la valeur de la version secondaire est 2.
_VER_REVISION	Numéro de révision de l'Albatros. Si la version d'Albatros est 3.2.1, la valeur de révision est 1.
_VER_SP	Chaîne décrivant le Service Pack s'il est installé, (par exemple "Service Pack 1f"), sinon indéfinie
_VER_FULL	Numéro de la version complète. Dans le cas de la version 3.2.1, il s'agit de $00030201H.

10.1.5 Mots clé

Les mots clé sont des identificateurs à utilisation réservée et ne pouvant être utilisés en aucune autre manière.

Les mots clé disponibles sont les suivants :

Tous les noms des instructions GPL	Voir la partie du manuel "Instructions" pour la description de toutes les instructions GPL
Tous les types de donnée	Voir Les variables
Les paramètres type dispositif	Voir Les paramètres de type dispositif
EXIST	mot utilisé dans l'instruction IFDEF pour contrôler l'existence d'un groupe. Voir l'instruction IFDEF
NOTEXIST	mot utilisé dans l'instruction IFDEF pour contrôler la non-existence d'un groupe. Voir l'instruction IFDEF

LINKED	mot utilisé dans l'instruction IFDEF pour activer la compilation des blocs de code, si le dispositif est connecté en virtuel-physique. Voir l'instruction IFDEF
UNLINKED	mot utilisé dans l'instruction IFDEF pour activer la compilation des blocs de code, si le dispositif n'est pas connecté en virtuel-physique. Voir l'instruction IFDEF
FUNCTION	Déclaration d'une fonction. Voir Les fonctions
AS	Mot utilisé pour la déclaration des variables. Voir Les variables
PUBLIC	Attribut d'une fonction. Voir Les fonctions
AUTORUN	Attribut d'une fonction. Indique que la fonction est à départ automatique. Voir Les fonctions
R= o READ	Attribut d'une fonction ou d'une variable. Signale le niveau d'accès à la lecture. Voir Les fonctions et Les variables et Droits d'accès
W=o WRITE	Attribut d'une fonction ou d'une variable. Signale le niveau d'accès à l'écriture. Voir Les fonctions et Les variables et Droits d'accès
CONST	Mot permettant d'assigner un nom significatif, appelé constante symbolique, à la place d'un nombre, d'un caractère ou d'une chaîne. Voir Les variables
READONLY	Attribut d'une variable globale. Voir Les variables
NONVOLATILE	Attribut d'une variable globale. Voir Les variables
PRIVATE	Attribut d'une fonction. Voir Les fonctions
RANGE	Mot utilisé pour la définition d'un intervalle de valeurs pour une variable. Voir Les variables
USER	Attribut d'une fonction ou d'une variable. Signale le type d'accès. Dans ce cas, l'utilisateur. Voir Les fonctions ou Les variables.
SERVICE	Attribut d'une fonction ou d'une variable. Signale le type d'accès. Dans ce cas, l'entretien. Voir Les fonctions ou Les variables.
MANUFACTURER	Attribut d'une fonction ou d'une variable. Signale le type d'accès. Dans ce cas, le Constructeur. Voir Les fonctions ou Les variables.
TPA	Attribut d'une fonction ou d'une variable. Signale le type d'accès. Dans ce cas, TPA. Voir Les fonctions ou Les variables.

10.1.6 Les fonctions

Les fonctions constituent le bloc minimal du code GPL. Les instructions GPL ne peuvent pas être insérées de façon séquentielle dans un fichier mais elles sont regroupées en fonctions. On peut déclarer non plus que 8191 fonctions.

Du point de vue du compilateur, tous les blocs de code GPL qui commencent par une ligne dont le premier mot est FUNCTION sont des fonctions. Toutefois, il n'y a pas de mot clé qui indique la fin du texte d'une fonction : la fonction s'achève à la ligne qui précède le début d'une autre fonction ou à la fin du fichier qui contient les fonctions.

La syntaxe qui permet de définir une fonction est la suivante :

```
FUNCTION    NomFonction  Attributs
   Paramètres
   Variables locales
   Liste des instructions GPL
```

Une fonction est également un type particulier de dispositif de Albatros. Elle partage des propriétés les dispositifs: un nom univoque (non traduisible), un indicateur de visibilité (si le dispositif est public ou non), un niveau d'accès en lecture et un niveau d'accès en écriture (voir le paragraphe suivant).

Droits d'accès
Les droits d'accès permettent de préciser le niveau d'accès minimal au système qui en donne la visibilité (droit de lecture) et l'exécution (droit d'écriture).
La syntaxe est la suivante:

```
Function    NomFonction    READ=S WRITE=M
```

Les droits sont identifiés par les mots clé READ (lecture) et WRITE (exécution)
Les valeurs associables, qui correspondent aux niveaux d'accès, sont les suivants :

- U ou USER utilisateur
- S ou SERVICE maintenance
- M ou MANUFACTURER Constructeur
- T ou TPA tpa

Les défauts des valeurs sont les suivants :

- READ lecture pour maitenance (S ou SERVICE) et utilisateur (U ou USER)
- WRITE écriture pour Constructeur (M ou MANUFACTURER) et tpa (T ou TPA)

Fonctions Autorun
Une fonction type autorun est exécutée automatiquement lors de l'initialisation de la machine.
Les fonctions autorun présentent la caractéristique d'être relancées automatiquement lorsqu'elles sont interrompues à la suite d'une erreur de système.
La syntaxe est la suivante:

```
Function NomFonction autorun
```

Il suffit donc d'ajouter le modificateur "autorun" à la déclaration de la fonction.

Fonctions Public
Normalement, une fonction peut être mise en exécution (appelée) uniquement par le code résidant à l'intérieur du fichier de groupe. Pour faire en sorte qu'une fonction puisse être mise en exécution par le code GPL d'un autre groupe, la fonction doit être définie de type **public.** La syntaxe qui permet de définir une fonction publique est la suivante:

```
Function NomFonction public
```

Il suffit donc d'ajouter le modificateur "public" à la déclaration de la fonction.
Font exception à cette règle les fonctions qui appartiennent à l'intergroupe, ces dernières étant toujours de type **public.**

Fonctions de Sous-groupe
Une fonction peut être associée à un sous-groupe tout simplement en faisant précéder le nom de la fonction par le nom du sous-groupe. Le nom du sous-groupe et celui de la fonction doivent être séparés par un point ".". Par exemple, la fonction suivante appartient au sous-groupe "X" du groupe "Axes".

```
Function       X.Zérotage
   local       vel as float
   movabs      X,100
   waitstill   X
Fret
```

Fonctions asynchrones

Une fonction asynchrone est rappelée automatiquement par la commande numérique lorsque l'événement auquel la fonction est liée est généré.
Il existe trois types d'événements :

- Changement d'état d'une entrée numérique : instruction ONINPUT
- Changement d'état d'un bit indicateur ou un flag switch : instruction ONFLAG
- Génération d'une erreur de système : instruction ONERRSYS

Lorsque l'événement est généré, la fonction est appelée (non en tant que tâche autonome mais dans le cadre de la tâche dans laquelle l'instruction ON... correspondante a été exécutée) en tant que FCALL implicite, dès que l'instruction courante a achevé l'exécution.
Habituellement, les fonctions asynchrones servent à gérer les situations d'urgence et elles doivent être très rapides. C'est la raison pour laquelle ces fonctions ne peuvent pas utiliser n'importe quelle instruction GPL, mais un sous-ensemble garantissant des temps d'exécution brefs.

Fonctions avec paramètres en entrée (paramétriques)
Une fonction peut avoir des paramètres déclarés en entrée, tandis qu'elle ne restitue en aucun cas une valeur.
Les paramètres peuvent être considérés comme des variables locales particulières dont la valeur est initialisée de l'extérieur au moment où la fonction est mise en exécution. Les paramètres sont déclarés avec le mot clé PARAM et ils adoptent la même syntaxe que celle qui est utilisée pour les variables locales. Les paramètres doivent être énumérés sur les premières lignes du corps de la fonction, avant toute autre instruction et avant même les variables locales.

Il existe deux manières de passer les paramètres :

- ***par valeur :*** sont passés par valeur tous les types de donnée simples, c'est-à-dire CHAR, INTEGER, FLOAT et DOUBLE. Le passage par valeur comporte la création d'une copie de la valeur originale. Les modifications dont le paramètre a fait l'objet ne sont valables que dans le cadre de la fonction.
- ***par référence :*** sont passés par référence les types structurés, c'est-à-dire ARRAY, MATRICES et CHAÎNES. Le passage par référence comporte l'utilisation de la variable d'origine, par conséquent les modifications dont le paramètre a fait l'objet sont valables dans le cadre de la fonction appelante. Cette caractéristique peut être utilisée pour restituer les valeurs de retour à la fonction appelante.

Habituellement, une fonction est mise en exécution avec l'instruction FCALL. S'il s'agit d'une fonction paramétrique, après le nom de la fonction, il est nécessaire de préciser la liste des valeurs à attribuer aux paramètres.
L'exemple qui suit présente une fonction paramétrique qui exécute un usinage de perçage. Les cotes du centre du trou et la vitesse d'avance de l'axe Z sont passées à la fonction en tant que paramètres.

```
Function Perçage
   Param Qx as Double      ; cote X du centre du trou
   Param Qy as Double
   Param vel as Float      ; vitesse d'avance

   Movabs                  X, Qx, Y, Qy
   Waitstill               X,Y
    ....
    Fret
```

L'appel de cette fonction, pour effectuer, par exemple, un trou aux cotes (12.5 , 25.7) et avec une vitesse d'avance de 3 m par minute, peut être effectué de la manière suivante:

```
Fcall   Perçage 12.5, 25.7, 3.0
```

Les paramètres passés à la fonction doivent correspondre, pour leur numéro et leur type, à ceux qui sont déclarés dans la fonction appelée. L'exécution de la fonction appelante repart à la fin de l'exécution de la fonction appelée.
En tant que paramètre d'une fonction, il est également possible de déclarer un dispositif. Cela permet d'écrire des fonctions d'utilisation générale, par exemple, une fonction de zérotage utilisable avec tous les axes présents sur la machine:

```
Function ZEROTAGE PUBLIC
         param      axe as Axis
         movabs     axe,100
Fret

Function MAIN
        .....
        Axes.Zérotage x
Fret
```

La fonction zérotage appartient au groupe Axes et elle est déclarée PUBLIC de façon à ce qu'elle puisse également être vue par les fonctions déclarées dans d'autres groupes. La fonction Main appelle la fonction zérotage du groupe des axes, en indiquant l'axe que l'on veut déplacer en guise de paramètre d'entrée.

10.1.7 Les paramètres de type dispositif

Les types Dispositif sont des variables particulières qui permettent de se référer à un dispositif de la machine.
Ces typologies de donnée peuvent être utilisées **exclusivement** dans la déclaration des paramètres d'une fonction. Il n'est pas possible de déclarer des variables de ce type. La définition des noms et des autres caractéristiques des dispositifs reste du ressort de la Configuration du système.
Le tableau présenté ci-dessous indique les types Dispositif et les mots clé correspondants à utiliser pour la déclaration des paramètres correspondants.

Type	**Mot clé**
Entrée numérique	INPUTDIG
Sortie numérique	OUTPUTDIG
Entrée analogique	INPUTANALOG
Sortie analogique	OUTPUTANALOG
Axe	AXIS
Temporisateur	TIMER
Compteur	COUNTER
Bit indicateur	FLAGBIT
Flag switch	FLAGSWITCH
Flag port	FLAGPORT
Port d'entrée	INPUTPORT
Port de sortie	OUTPUTPORT
Fonction	FUNCTION
Dispositif générique	DEVICE
Tâche	TASK

Dans l'exemple suivant, le paramètre déclaré et utilisé est un paramètre de type axe :

```
Function essai
  Param axe as axis

    MovAbs    axe,100
    WaitStill axe

Fret
```

10.1.8 Le Multitâche

Le système Albatros est de type multitâche, ce qui implique qu'il peut exécuter plusieurs tâches GPL en même temps. La tâche est un processus de gestion d'une entité logique (un groupe, généralement).

Il existe deux types de tâches : les tâches normales et les tâches real-time.

Tâches normales
Le multitâche se base sur un algorithme coopératif basé sur des priorités. Il permet à toutes les tâches d'être exécutées de façon cyclique et de modifier leurs priorités. L'algorithme de programmation prévoit l'exécution d'une instruction pour chaque tâche active (état running). Chaque tâche est associée à une priorité introduite avec l'instruction SETPRIORITYLEVEL. La priorité est identifiée par un nombre compris entre 0 (priorité maximale) et 255 (priorité minimale). Pour les tâches ayant une priorité 0 (zéro) une instruction est effectuée à chaque début de programmation. Pour les tâches ayant une priorité 1, l'instruction est exécutée à tous les cycles de programmation et ainsi de suite jusqu'aux tâches ayant la priorité 255, pour lesquelles une instruction est effectuée tous les 256 cycles de programmation.

L'exécution des tâches normales est asynchrone par rapport à la fréquence de rafraîchissement des axes. Cela signifie qu'il n'est pas garanti qu'une fonction GPL soit achevée dans le laps de temps s'écoulant entre deux actualisations de l'état des axes.

Une tâche est identifiée avec le nom de la fonction GPL d'où son exécution est partie.
L'exécution d'une tâche peut être lancée :

- automatiquement lors de l'initialisation du système : fonction principale de l'intergroupe et fonctions autorun.
- à la suite de l'exécution d'une instruction STARTTASK.
- à la suite du lancement en mode manuel sur l'interface graphique de Albatros.

Chaque tâche se caractérise par un état interne:

RUNNING	la tâche est en exécution
HOLD	la tâche est suspendue
BREAK	la tâche est interrompue par le débogueur

La hiérarchie des tâches est arborescente. Chaque tâche est créée par une autre tâche dont elle est la fille, ce qui implique que si la tâche mère est terminée, toutes les tâches filles sont achevées.
Le nombre maximal de tâches pouvant être exécutées en même temps est de 500.
Il faut tenir compte qu'un nombre élevé de tâches en exécution comporte une diminution de la vitesse à laquelle chaque tâche est effectuée.

Si l' application que l'on va réaliser comporte l'utilisation d'un nombre de tâches plus important de 200, il faut utiliser un disque dur adéquat comme les modules Cn2128**.**

Tâches Real-Time
Les tâches real-time diffèrent des précédentes par le fait qu'elles ne font l'objet d'aucun mécanisme de programmation et qu'elles ne sont pas associées à une priorité. En effet, elles sont exécutées complètement lors de chaque actualisation de l'état des axes (real-time axes).

Il est absolument nécessaire que l'exécution de ces tâches s'achève dans un certain laps de temps parce que l'exécution des tâches GPL décrites précédemment reste suspendue pendant l'exécution des tâches real-time.
Le système exécute des contrôles du temps d'exécution des tâches real-time et une erreur de système est générée si ce temps maximal n'est pas respecté.
Il est donc déconseillé de créer des cycles infinis (ex. avec GOTO) à l'intérieur de ces tâches. Par ailleurs, ces cycles ne sont pas nécessaires étant donné que l'exécution du code recommence depuis le début à chaque axes real-time.
Toujours pour éviter des temps d'exécution trop longs, les tâches realtime font l'objet de restrictions quant à l'emploi de certaines instructions GPL. Les instructions dont l'utilisation est impossible sont celles qui ne sont pas utilisables sur Interrupt.

Il est conseillé d'utiliser les tâches real-time uniquement pour les activités qui doivent nécessairement être effectuées en mode synchrone avec actualisation des cotes axes. Pour la plupart des activités de contrôle, il convient plutôt d'utiliser des tâches normales.

Les tâches real-time sont lancées avec l'instruction STARTREALTIMETASK et elles peuvent être interrompues avec l'instruction ENDREALTIMETASK. Il est possible d'activer un maximum de 256 tâches real-time en même temps.
Il n'est plus valable le mécanisme d'hérédité, donc si la tâche qui a envoyé une tâche real-time finit, celle-ci restera en exécution.
Les variables locales déclarées dans la tâches real-time ne sont initialisées qu'en démarrage de tâche et ensuite elles maintiennent la valeur de la dernière exécution.

Les tâches real-time ne présentent pas les états prévus pour les tâches normales. Il est possible d'effectuer le débogage d'une tâche real-time mais, dans ce cas, le système déclasse automatiquement la tâche qui devient une "tâche normale" pendant tout le débogage.

Si une erreur de système se présente dans une tâche real-time, la tâche est déclassée en devenant une tâche normale et elle est mise en état HOLD pour que le débogueur puisse l'analyser.

10.1.9 Les Communications

Les communications entre le GPL et le monde extérieur se font de deux manières distinctes :
- SEND / RECEIVE
- Communications sérielles
- IPC

Send / Receive
Les instructions SEND et RECEIVE implémentent un mécanisme de communication destiné aux messages.

La communication peut se faire à l'intérieur du même module (peu avantageux), entre plusieurs modules d'une ligne ou entre les modules et le superviseur Albatros ou bien avec des applications OLE. Le fonctionnement est semblable à celui de la poste ; chaque message est associé à un destinataire, un identificateur de l'information transmise (ou demandée), information elle-même plus informations de service. Albatros recueille et trie les informations et, dans certains cas, fournit directement les informations demandées.
Normalement, ce mode de communication est utilisé pour l'envoi des programmes d'usinage entre le superviseur et les modules de contrôle, pour la synchronisation des activités des machines d'une ligne et pour l'interfaçage avec des applications externes (serveur OLE).

Communications sérielles
Le langage GPL fournit quelques instructions, par exemple COMREAD et COMWRITE, qui permettent d'envoyer et de recevoir des données par le biais des ports sériels de la commande. Il est donc possible de mettre en interface la commande avec des dispositifs externes (inverseurs, terminaux ou PLC). Judicieusement utilisées, ces instructions permettent d'implémenter des protocoles de communication sérielle comme MODBUS-RTU etc.

IPC
L'IPC ou Inter Process Communication est un mode de communication entre des processus. En particulier, ce mode permet de définir une zone de mémoire partagée entre deux ou plusieurs processus et pouvant être utilisée pour l'échange d'informations.
Habituellement, il est utilisé lorsque l'on doit transmettre un très grand nombre de données ou, plus généralement, dans lorsque les performances fournies par l'interface OLE de Albatros sont inadéquates. Sur le côté GPL, la communication IPC est implémentée avec les instructions SENDIPC, WAITIPC et TESTIPC. Dans le cas de module local les processus externes peuvent se référer aux API fournis RTX ou à la composante COM **gplipc2.dll** fournie par TPA qui en simplifie l'utilisation.

Dans le cas de module à distance les processus qui se déroulent dans le superviseur utilisent la composante COM **gplipc2net.dll**.
Pour toute information complémentaire, contactez TPA.

10.1.10 Variables à utiliser pour la programmation

La plupart des instructions ont été réalisées de façon à pouvoir travailler avec différents types de variables (CHAR, INTEGER, FLOAT, DOUBLE). Toutefois, chaque instruction a un type de variable préférée en fonction de laquelle elle a été optimisée. Pour garantir de meilleurs performances pendant l'exécution du code GPL, il est conseillé d'utiliser les types de variable conseillées dans la description de chaque instruction. En général, il est conseillé de respecter le tableau qui, présenté ci-dessous, associe les types optimaux aux principales valeurs utilisées dans la programmation :

valeur	***type***
cote	*double*
vitesse	*float*
temps	*double*
compteur	*integer*
valeur port / flag port	*integer*
timeout	*double*
entrée / sortie analogique	*float*
cosinus directeurs	*double*
caractère de contrôle chaîne	*char*
accélérations / décélérations	*integer*

10.1.11 Les axes

Le nom "axe" sert généralement à indiquer un système électromécanique dont le but est d'assurer le déplacement contrôlé d'un organe d'une machine-outil.
Il est possible de décrire ce système selon les éléments qui le composent, ces derniers pouvant être répartis en fonction de leurs caractéristiques technologiques.
Il existe donc des pièces mécaniques, dont :

- le bâti portant
- les guides
- les paliers
- les vis + roulements à recirculation de billes

le rôle duquel est de lutter contre les forces qui s'exercent, de diminuer les frottements et le transformer le mouvement rotatoire en un mouvement translatif, etc.
Il existe également des pièces électriques et électroniques, dont :

- le moteur
- les interrupteurs de fin de course

- le codeur
- la dynamo tachymétrique

Leur rôle est de fournir la puissance nécessaire au mouvement et de détecter l'état du système.
Ces éléments sont reliés les uns aux autres, de façon à garantir l'exécution des mouvements de façon contrôlée.

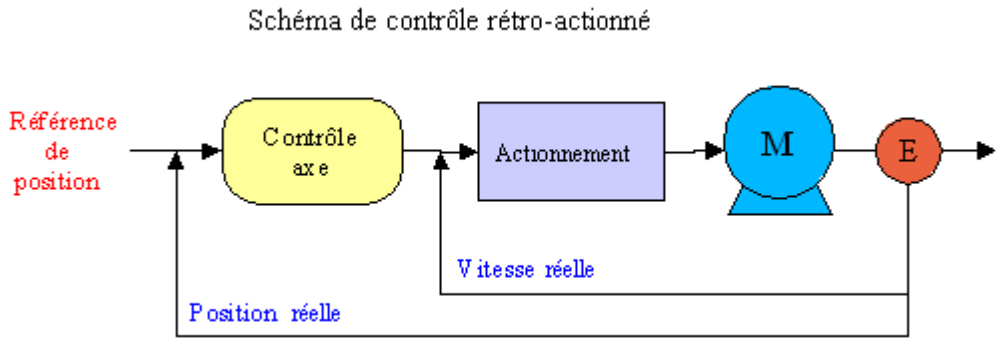


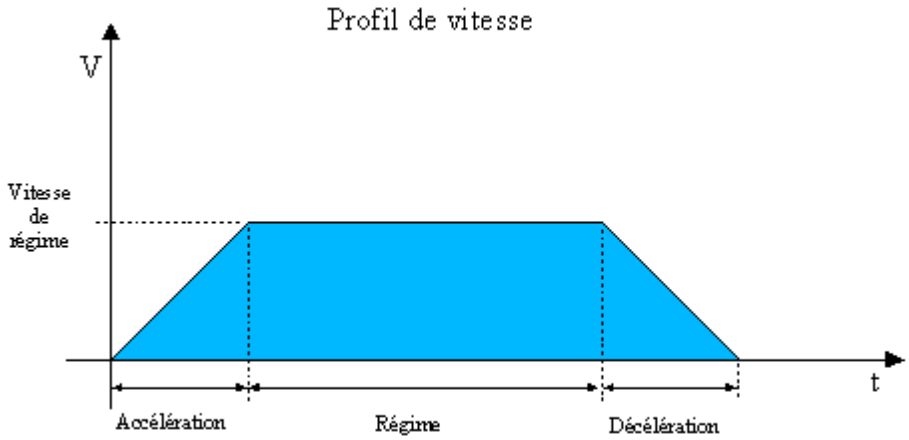
Schéma d'un axe avec commande rétroactionnée

Le rôle de la commande numérique est de contrôler la position et le mouvement d'un axe.

Le mouvement d'un axe se décompose en 5 phases :

Accélération	phase initiale pendant laquelle l'axe augmente progressivement de vitesse jusqu'à obtention de celle qui est désirée
Régime	phase intermédiaire pendant laquelle l'axe se déplace à vitesse constante (cette phase peut ne pas exister si l'espace à parcourir est inférieur à l'espace parcouru pendant l'accélération et la décélération)
Décélération	phase pendant laquelle l'axe diminue progressivement de vitesse jusqu'à 0
Fenêtre	phase d'attente que l'erreur de boucle s'abaisse à la valeur indiquée dans la configuration sous "fenêtre d'arrivée en cote"
Cote	fin du mouvement

A la fin du mouvement, l'axe doit être positionné à l'intérieur d'un intervalle que l'on appelle fenêtre d'arrivée en cote (définit la tolérance sur le positionnement de l'axe). Si cela ne se produit pas dans les 5 secondes qui suivent le moment où le mouvement aurait dû s'arrêter, le système génère une erreur de système "mouvement non fini".



Profil de vitesse du mouvement d'un axe

Pour chaque mouvement, la commande numérique calcule un profil de vitesse comme celui qui est représenté sur la figure, puis elle calcule les cotes cible en partageant le profil de vitesse en intervalles de temps égaux au temps de rafraîchissement de l'axe et en calculant la surface de chaque partie. La surface constitue l'incrément de cote que l'axe doit atteindre pendant ce laps de temps pour respecter le profil de vitesse vue ci-dessus.

Le contrôle de l'axe est implémenté par un contrôleur PID qui pourvoit à "fermer l'anneau de position", c'est-à-dire à fournir une référence de vitesse à l'actionnement calculé en fonction de la cote que l'on veut atteindre (cote cible) et de la cote réelle lue par le codeur. La différence entre la cote réelle et la cote cible est appelée ***Erreur de boucle***.

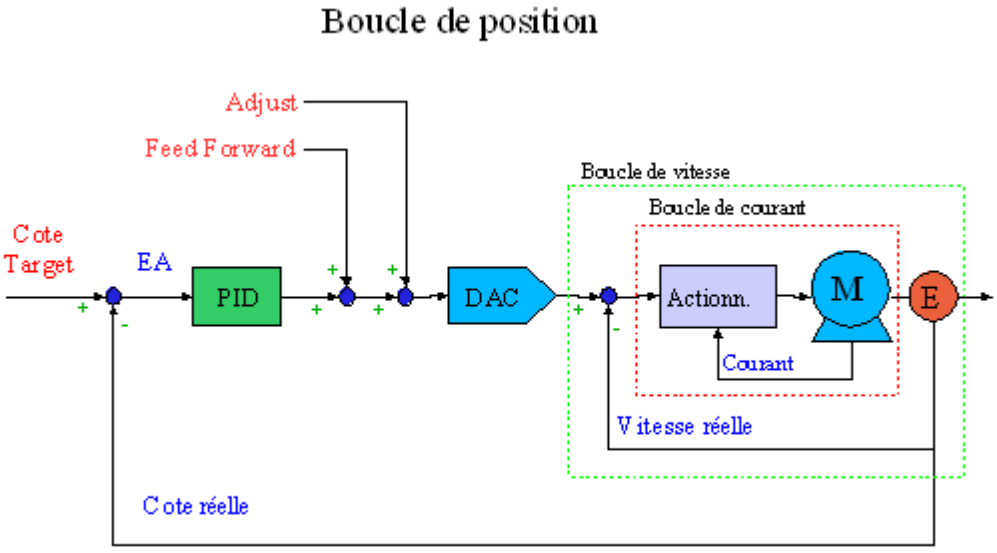


Schéma du contrôle d'axe de Albatros

10.1.12 Correcteurs de linéarité

Le tableau des correcteurs de linéarité d'un axe est vu comme une matrice, dont le nom est *NomGroupe.NomSousgroupe.NomAxe#*correctors ou *NomGroupe.NomAxe#*correctors et peut etre utilisé dans toutes les instructions, dans lesquelles les matrices, les elements de matrice et les lignes de matrice sont accessibles.

Le nombre de colonnes de la matrice correspond au nombre d'axes définis dans la fenêtre des correcteurs de linéarité dans les configurations d'axes. Les correcteurs automatiques sont insérés dans la première colonne. Toutes les valeurs de correction sont de type flottant (float). Le nombre total de lignes de la matrice peut être obtenu au moyen de l'instruction GPL LASTELEM.

Ces matrices sont accessibles en mode lecture et écriture et, si elles sont modifiées, leurs valeurs sont immédiatement utilisées pour la correction des dimensions, uniquement si la correction a été activée.

Exemple :

```
Function ReadCorr
local i as integer
local j as integer
local row as integer
local column as integer
local firstvalue as float

; la lecture de la première valeur autocorrectrice de l'axe AX
firstvalue = X.Ax#correctors[1][1]
; nombre d'axes dans la fenêtre du correcteur de linéarité
setval 3 column
lastelem X.AX#correctors  row
; augmente d'une valeur constante toutes les corrections
for i 1 row
     for j 1 column
        X.Ax#correctors[i][j] = X.Ax#correctors[i][j] + 0.025
      next
next

fret
```

10.1.13 Gestion des messages en langue

Comme il est décrit dans le chapitre Composition du système, Albatros dispose de la visualisation des messages de texte en plusieurs langues.

Ce support est fourni grâce à l'utilisation de TpaLangs qui est un programme situé à l'extérieur de Albatros qui gère des archives de messages. Ce programme supporte l'activité de traduction des messages dans le langues différentes.

Texte associé à Erreurs de Cycle et Messages
Un type particulier de textes, qui sont normalement affichés par Albatros sont les Messages et les Erreurs de Cycle générés par le code GPL.
Normalement, ces derniers sont définis par ceux qui élaborent le code GPL pendant l'écriture du code lui-même. Pour faciliter le travail du programmateur, l'éditeur du GPL permet d'insérer le texte d'un message directement à travers Albatros, c'est-à-dire sans qu'il soit nécessaire d'utiliser TpaLangs.
Une deuxième possibilité de gérer les messages en langue est donnée par l'utilisation de l'instruction GPL DEFMSG.

10.1.14 Gestion des erreurs de système

Quand il se présent une erreur de système (Voir chapitre ***Erreurs de Système->Introduction aux Erreurs de Système)*** le comportement normal du contrôle est cela de terminer tous les tâches : la gestion des erreurs de système permet d'éviter la terminaison de tâches pour lesquelles a été activé. Erreurs de système générées par défaut, par débordement négatif de la pile et par dépassement de la pile sont gerées directement par le contrôle sans rappeler la fonction de gestion des erreurs de système : la tâche est mise en état HOLD.

La fonction de gestion des erreurs

A l'intérieur du code GPL il doit être défini une o plus fonctions pour l'analyse de l'erreur de système et pour la conséquente définition de opportunes actions pour mettre en sécurité la machine. La fonction à rappeler est passé comme paramètre à l'instruction GPL ONERRSYS. (Voir chapitre ***Langage GPL->Instructions->Gestion de flux->ONERRSYS).***

Si arrive une erreur de système la tâche qui a généré l'erreur est mise en état HOLD. Si les tâches autorun gênèrent erreurs de système ils sont relancés seulement si l'erreur de système n'est pas un FAULT.
Si l'erreur de système est générés sans nombre de tâche il est mis en HOLD la tâche courante.

10.2 Fonctions spéciales

10.2.1 Personnalisation mouvement axes

L'interface graphique du système Albatros prévoit la possibilité de déplacer les axes en manuel et donne un support graphique à l'étalonnage de ces-ci.
Le mouvement en manuel des axes se fait au moyen de la console de déplacement axes, l'étalonnage peut être effectué au moyen de la console d'étalonnage. Les deux peuvent être appelées par la fonction de Diagnostique et par les tableaux synoptiques.
Dans les deux cas le mouvement de l'axe est commandé par un group de fonctions GPL dont l'exécution reste invisible pour l'utilisateur.
Le système a un ensemble prédéfini de ces fonctions qui résultent adéquates dans la majeure partie des cas. Toutefois, il peut être nécessaire les personnaliser, par exemple pour introduire des restrictions au déplacement des axes liés a l'état de la machine ou pour gérer dispositifs auxiliaires comme par exemple les freins axe.

La personnalisation se fait au moyen de la création de deux fonctions GPL pour chaque axe : une pour le déplacement en manuel et un pour le étalonnage. Ces fonctions sont optionnelles, donc si elles sont détectées par le système, elles sont utilisées ; si non, les fonctions standards sont utilisées. En plus, il y a également une personnalisation partielle des fonctions de mouvement.

Déplacement en manuel

Les fonctions de *mouvement manuel* personnalisées doivent respecter les spécifications suivantes :

- La fonction doit appartenir au même sous-group auquel il appartient l'axe dont elle fait référence.
- Le nom de la fonction doit être **MoveAx#*nom_axe*** où nom_axe doit être substitué avec le nom de l'axe défini dans la configuration. Par exemple pour l'axe X le nom de la fonction serait : MoveAx#X .
- La fonction doit prévoir les paramètres suivants :

1. ***Action demandée***. Peut être un mouvement sur une cote absolue, un mouvement incrémental, un stop etc. Les actions sont identifiées par un nombre entier, le compilateur GPL prévoit des constantes prédefinies associées aux actions :

_MOVAXABS	mouvement sur cote absolue
_MOVAXINC	mouvement sur cote incrémental
_MOVAXSET	programmation de la cote
_MOVAXFREE	programmation état normal
_MOVAXNORMAL	rétablissement état axe sur fin mouvement (il ne sert pas pour arrêter l'axe)

2. ***Résultat***. Il sert au système pour savoir si l'action demandée est gérée par la fonction personnalisée. Si l'action n'est pas gérée, il est utilisé la fonction standard correspondante. Il s'agit donc d'une valeur de retour que la fonction personnalisée doit programmer et dans ce but elle est définie comme un paramètre passé par référence (array d'un seul élément).
3. ***Vitesse***. Elle est significative seulement quand l'action demandé est un mouvement et elle programme sa vitesse.
4. ***Cote***. Significative seulement pour les actions de mouvement et de programmation cote.

Exemple de fonction de déplacement personnalisée :

```
Function MoveAx#X
   param action as integer
   param result[1] as integer
   param speed as float
```

```
    param position as double

    setval      1,result[1]

    select action
    case _MOVAXEND
          fcall EndMovement
    case _MOVAXABS
          fcall AbsMovement X, speed, position
    case _MOVAXINC
          fcall IncMovement X, speed, position
    case _MOVAXSET
          fcall PositionSet X, position
    case _MOVAXFREE
          fcall FreeAxis
    case _MOVAXNORMAL
          fcall NormalAxis
    case else
          call  Unknown
    endselect

    fret

Unknown:
    setval      0, result[1]
    ret
```

Les fonctions EndMovement, AbsMovement, etc. (les noms n'ont pas de liens) doivent exécuter la gestion personnalisée des actions demandées. Pour faciliter le travail du programmeur on montre les fonctions standards de déplacement qui peuvent servir comme trace pour le développement des fonctions personnalisées.

Étalonnage

Les fonctions d'*étalonnage* personnalisées doivent respecter les spécifications suivantes :

- La fonction doit appartenir au même sous-group auquel il appartient l'axe dont elle fait référence.
- Le nom de la fonction doit être **CalibAx*#nom_axe*** où nom_axe doit être substitué avec le nom de l'axe défini dans la configuration. Par exemple, pour l'axe X le nom de la fonction sera : CalibAx#X .
- La fonction doit prévoir les paramètres suivants :

  1. ***Action demandée***. Peut être un mouvement point à point ou un mouvement interpolé.
  2. ***Résultat***. Il sert au système pour savoir si l'action demandée est gérée par la fonction personnalisée. Si l'action n'est pas gérée, la fonction standard correspondante est utilisée.
  3. ***Vitesse***. Vitesse de mouvement pendant le étalonnage.
  4. ***Cote Positive***. Cote positive du mouvement alterné d'étalonnage.
  5. ***Cote Négative***. Cote négative du mouvement alterné d'étalonnage.
  6. ***Attente***. Attente entre un mouvement et le suivant.

NOTE: On doit considérer que dans certains cas actions effectuées dans la console d'étalonnage comportent l'exécution de la fonction de déplacement axe. Par exemple à la fin du mouvement d'étalonnage (quand on presse le bouton de stop) est effectuée une opération de rétablissement de l'axe pour laquelle est appelée la fonction de déplacement axe personnalisée avec le paramètre action programmé à _MOVAXEND. De la même façon, lorsque on modifie la cote de l'axe da la console d'étalonnage, la fonction de déplacement axe est appelée avec le paramètre action programmé sur _MOVAXSET.

Exemple de fonction d'étalonnage personnalisée :

```
Function CalibAx#X
param    action as integer
param    result[1] as integer
param    speed as float
```

```
param   PosPosition as double
param   NegPosition as double
param   WaitTime as float

   setval      1,result[1]

   select action
   case _CALAXPP
         fcall PPCalibration X, speed, PosPosition, NegPosition,
WaitTime
   case _CALAXINT
         fcall IntCalibration X, speed, PosPosition, NegPosition,
WaitTime
   case else
         call  Unknown
   endselect

   fret

Unknown:
   setval      0, result[1]
   ret
```

Les fonctions PPCalibration et IntCalibration (les noms n'ont pas de liens) doivent exécuter la gestion personnalisée des actions demandées. Pour faciliter le travail du programmeur on montre les fonctions standards d'étalonnage qui peuvent servir comme trace pour le développement des fonctions personnalisées.

Interaction avec la fenêtre de Mouvement axe en manuel

Les fonctions pour l'interaction avec la fenêtre de mouvement manuel doivent respecter les spécifications suivantes :

- La fonction devra appartenir au même sous-groupe que celui auquel l'axe de référence appartient.
- Le nom de la fonction devra être **MoveAx#nom_axe#Action** où nom_axe doit être remplacé par le nom de l'axe défini lors de la configuration, et Action peut désigner l'une des définitions suivantes :

OPEN	indique que l'utilisateur vient d'ouvrir la fenêtre de mouvement axe
CLOSE	indique que l'utilisateur va fermer la fenêtre de mouvement axe
ACTIVE	signale que la fenêtre de mouvement axe est active
INACTIVE	signale que la fenêtre de mouvement axe n'est pas active
JOG	indique qu'un mouvement par déplacement géré runtime par l'opérateur est inséré
STEP	indique qu'un mouvement avec déplacement de pas prédéfini est inséré
ABSOLUTE	indique qu'un mouvement avec déplacement à cote établie est inséré

Par exemple, si la fenêtre de mouvement axe pour l'axe X a été ouverte, la fonction apparaîtra avec le nom MoveAx#X#Open.

Modifications à la fenêtre de mouvement axe en manuel

Il est possible d'ajouter jusqu'à 4 touches à la fenêtre de mouvement axe. Dans le même sous-groupe où est défini l'axe concerné, il faut définir des fonctions GPL à nom fixe MoveAx#NomeAsse#BUTTONtext. NomAxe représente le nom de l'axe concerné et text représente le texte qui apparaîtra sur la touche. Le texte peut contenir le caractère '&' pour introduire un accélérateur de clavier. Si le texte commence par un nombre compris entre 1 et 4, ce nombre est considéré comme la position où sera inséré la touche dans la fenêtre de mouvement axe. Le texte de la touche peut être traduit en introduisant une DEFMSG, dans le groupe où se trouve l'axe, qui ait comme identificateur MOVEAX#BUTTONtesto. La pression de la touche personnalisée comporte l'exécution de la fonction GPL associée. Il n'y a aucune attente de fin d'exécution, ni aucun contrôle de début d'exécution de la fonction.

10.2.2 Fonctions standards de déplacement et étallonage

Les fonctions suivantes sont celles standards utilisées par la console de déplacement axes et la console de calibrage.
Le fonctions changent selon le type d'axe concerné : comptage, pas à pas, etc.
Cettes fonctions peuvent être personalizzate.

Fonctions standards de déplacement manuel

Mouvement à cote absolue

```
; pour axes pas à pas
Function AbsMovement
     param axisname as axis
     param speed as float
     param position as double

     ifstill     axisname goto move
     fret
move:
     setvel      axisname, speed
     movabs      axisname, position
     waitstill   axisname
     fret

; pour tous les autres types d'axes
Function AbsMovement
     param axisname as axis
     param speed as float
     param position as double

     iftarget    axisname goto move
     ifstill     axisname goto move
     fret
move:
     setvel      axisname, speed
     movabs      axisname, position
     waitstill   axisname
     fret
```

Mouvement incremental

```
; pour axes pas à pas
Function IncMovement
     param axisname as axis
     param speed as float
     param position as double

     ifstill     axisname goto move
     fret
move:
     setvel      axisname, speed
     movinc      axisname, position
     waitstill   axisname
     fret

; pour tous les autres types d'axes
Function IncMovement
     param axisname as axis
     param speed as float
```

```
     param position as double

     iftarget    axisname goto move
     ifstill     axisname goto move
     fret
move:
     setvel      axisname, speed
     movinc      axisname, position
     waitstill   axisname
     fret
```

Programmation cote

```
; pour axes de comptage
Function PositionSet
     param axisname as axis
     param position as double

     setquote    axisname, position
     fret

; pour axes pas à pas
Function PositionSet
     param axisname as axis
     param position as double

     ifstill     axisname goto set
     fret
set:
     setquote    axisname, position
     fret

; pour tous les autres types d'axes
Function PositionSet
     param axisname as axis
     param position as double

     iftarget    axisname goto set
     ifstill     axisname goto set
     fret
set:
     setquote    axisname, position
     fret
```

Programmation état free

```
Function FreeAxis
     param axisname as axis

     free        axisname
     fret
```

Programmation état normal

```
Function NormalAxis
     param axisname as axis

     normal      axisname
     fret
```

Fonctions standards de calibrage

Calibrage en point-point

```
; pour axes pas à pas
Function PPCalibration
     param axisname as axis
     param speed as float
     param PosPosition as double
     param NegPosition as double
     param WaitTime as float

     setvel      axisname, speed
loop:
     movabs      axisname, PosPosition
     waitstill   axisname
     delay       WaitTime
     movabs      axisname, NegPosition
     waitstill   axisname
     delay       WaitTime
     goto        loop
     fret

; pour tous les autres types d'axes
Function PPCalibration
     param axisname as axis
     param speed as float
     param PosPosition as double
     param NegPosition as double
     param WaitTime as float

     setvel      axisname, speed
loop:
     movabs      axisname, PosPosition
     waitstill   axisname
     ifquotet    axisname,<>,PosPosition goto exit
     delay       WaitTime
     movabs      axisname, NegPosition
     waitstill   axisname
     ifquotet    axisname,<>,NegPosition goto exit
     delay       WaitTime
     goto        loop
exit:
     fret
```

Calibrage en interpolation

```
Function IntCalibration
     param axisname as axis
     param speed as float
     param PosPosition as double
     param NegPosition as double
     param WaitTime as float

     setveli     axisname, speed
loop:
     linearabs   axisname, PosPosition
```

```
    waitstill   axisname
    ifquotet    axisname,<>,PosPosition goto exit
    delay       WaitTime
    linearabs   axisname, NegPosition
    waitstill   axisname
    ifquotet    axisname,<>,NegPosition goto exit
    delay       WaitTime
    goto        loop
exit:
    fret
```

10.2.3 Fonction OnUIEnd#

La fonction "OnUIEnd#" est exécutée, si disponible, par Albatros avant de terminer tous les tâches d'un module. Elle doit être définie dans le fichier de fonctions de l'intergroupe. La durée maximale d'exécution de la fonction "OnUIEnd#" est 2 secondes, après quoi Albatros va terminer toutes les tâches.

Le temps maximal pour attendre que cette fonction achève l'exécution est configurable en Tpa.ini, dans la section [Albatros], sous Timeout.OnUIEnd=*valeur*, où *valeur* est en millisecondes et ne peut pas dépasser 60000.

10.2.4 Fonction OnUIPlugged#

La fonction OnUIPlugged# est exécutée lorsqu'on a besoin de savoir, par exemple après l'allumage d'une installation, si Albatros a communiqué avec le module à distance. Cette fonction doit être définie dans l'intergroupe.

10.2.5 Fonction OnUIUnplugged#

La fonction "OnUIUnplugged#" est exécutée avant de terminer l'exécution de Albatros (et donc avant que Albatros se déconnecte d'un module). Cette fonction doit être définie dans l'intergroupe. Albatros exécute cette fonction entre max. 2 secondes, pendant lesquels on lit :

- erreurs de cycle
- erreurs de système
- messages

À la fin de l'exécution Albatros s'arrête.

Le temps maximal pour attendre que cette fonction achève l'exécution est configurable en Tpa.ini, dans la section [Albatros], sous Timeout.OnUIEnd=*valeur*, où *valeur* est en millisecondes et ne peut pas dépasser 60000.

10.3 Instructions

10.3.1 Conventions

Les pages suivantes se présentent sous la forme de fiches qui décrivent, pour chaque instruction :

- la syntaxe
- une description des arguments : type de donnée et valeurs admises
- une description du fonctionnement
- d'éventuelles notes
- d'éventuels exemples

Les instructions de la même nature sont regroupées pour faciliter l'apprentissage et la consultation.

10.3.2 Type des instructions du langage GPL

Le langage se compose d'instructions qui peuvent être groupées comme il suit :

Instructions pour la gestion des Entrées/Sorties (Input/Output)

GETFEED	lit le feed rate override
INPANALOG	lit une entrée analogique
INPFLAGPORT	lit un flag port
INPPORT	lit un port numérique
MULTIINPPORT	lit jusqu'à 4 ports de sortie
MULTIOUTPORT	règle jusqu'à 4 ports de sortie
MULTISETFLAG	règle plusieurs flags à 1
MULTISETOUT	règle plusieurs sorties à 1
MULTIRESETFLAG	règle plusieurs flags à 0
MULTIRESETOUT	règle plusieurs sorties à 0
MULTIWAITFLAG	attend l'état d'un bit indicateur ou flag switch
MULTIWAITINPUT	attend l'état de plusieurs entrées
OUTANALOG	modifie une sortie analogique
OUTFLAGPORT	modifie un flag port
OUTPORT	modifie un port numérique
RESETFLAG	règle un flag à 0
RESETOUT	règle une sortie à 0
SETFLAG	règle un flag à 1
SETOUT	règle une sortie à 1
WAITFLAG	attend l'état d'un flag ou flag switch
WAITINPUT	attend l'état d'une entrée
WAITPERSISTINPUT	attend l'état persistant d'une entrée

Instructions pour la gestion des Axes

CHAIN	enchaîne un axe à un autre
CIRCABS	interpolation circulaire absolue
CIRCINC	interpolation circulaire incrémentielle
CIRCLE	exécute un cercle
COORDIN	mouvement des axes coordonnés
DISABLECORRECTION	désactive la correction linéaire pour l' axe spécifié
EMERGENCYSTOP	force l'arrêt d'urgence des axes
ENABLECORRECTION	désactive la correction linéaire pour l' axe spécifié
ENDMOV	achève le mouvement d'un axe
FASTREAD	lecture rapide des cotes des axes
FREE	met l'axe en free
HELICABS	interpolation hélicoïdale absolue
HELICINC	interpolation hélicoïdale incrémentielle
JERKCONTROL	active ou désactive le contrôle sur les mouvements en interpolation
JERKSMOOTH	raccorde avec continuité d'accélération et de vitesse les profils de vitesse des axes pendant les mouvements en contournage
LINEARABS	interpolation linéaire absolue
LINEARINC	interpolation linéaire incrémentielle
MOVABS	mouvement des axes en mode absolue
MOVINC	mouvement des axes en mode incrémentiel
MULTIABS	interpolation linéaire multiaxe en mode absolu
MULTIINC	interpolation linéaire multiaxe en mode incrémentiel
NORMAL	enlève le free d'un axe
RESRIFLOC	rétablit les références initiales
SETINDEXINTERP	associe à un axe une variable pour le comptage des blocs d'interpolation exécutés
SETLABELINTERP	associe à un axe une variable pour l'identification d'un bloc de déplacement
SETPFLY	zérotage éclair
SETPFLYCHAINSTRAT	vérifie le comportement de l'axe slave ci-contre une instruction setpfly sur le maître
SETPZERO	zérotage sur cran de zéro
SETPZEROCHAINSTRAT	vérifie le comportement de l'axe esclave ci-contre une instruction setpfly sur le maître

SETQUOTE	règle la cote
SETQUOTECHAINSTRAT	contrôle le comportement de l'axe esclave face à une instruction setquote sur le maître
SETRIFLOC	règle les références spatiales
SETTOLERANCE	règle les valeurs de tolérance pour l'interpolation linéaire
START	reprend le mouvement d'un axe
STARTINTERP	force le début d'une interpolation
STOP	suspend le mouvement d'un axe
SWITCHENC	permet de remplacer le codeur d'un axe avec celui d'un autre axe
WAITACC	attend que l'axe soit en accélération
WAITCOLL	attend que l'axe supère une coordonnée de laquelle on commence à vérifier la présence d'une collision
WAITDEC	attend que l'axe soit en décélération
WAITREG	attend que l'axe soit en régime
WAITSTILL	attend que la cote finale soit égale à la cote cible
WAITTARGET	attend que l'axe soit en cible
WAITWIN	attend que l'axe soit en fenêtre

Istructions pour la gestion des Paramètres de l'axe Lecture/Écriture

DEVICEID	écrit l'adresse logique associé à un dispositif
GETAXIS	lit une ou plusieurs données d'un axe

Mouvement Point-Point

SETACC	règle l'accélération
SETDEC	règle la décélération
SETDERIV	règle le coefficient d'action dérivée
SETFEED	règle le feed rate point-point
SETFEEDF	règle le feed forward
SETFEEDFA	règle le feed forward d'accélération
SETINTEG	règle le coefficient d'action intégrale
SETMULTIFEED	règle la valeur de pourcentage de feed rate override des axes indiqués
SETPROP	règle le coefficient d'action proportionnelle
SETSLOPE	règle le type de rampe dans les mouvements rapides
SETVEL	règle la vitesse

Mouvement interpolé

LOOKAHEAD	règle le lookahead d'interpolation
SETACCI	règle l'accélération par interpolation
SETACCLIMIT	active et désactive le calcul automatique de la vitesse de régime d'interpolation
SETACCSTRATEGY	sélectionne le type d'accélération
SETAXPARTYPE	change l'ensemble de paramètres de l' axe actuellement utilisé
SETCONTORNATURE	règle l'angle de contournage
SETDECI	règle la décélération par interpolation
SETDERIVI	règle le coefficient d'action dérivée d'interpolation
SETFEEDFAI	règle le feed forward d'accélération en interpolation
SETFEEDI	règle le feed rate en interpolation
SETFEEDFI	règle le feed forward en interpolation
SETINTEGI	règle le coefficient d'action intégrale d'interpolation
SETPROPI	règle le coefficient d'action proportionnelle d'interpolation
SETSLOPEI	règle le type de rampe dans les mouvements dans l'interpolation
SETSLOWPARAM	modifie les paramétrés nécessaires à calculer la vitesse de décélération si la fonctionnalité de décélération en contournage est active
SETVELI	règle la vitesse
SETVELILIMIT	règle chaque composant de vitesse de l'axe spécifié

Mouvement coordonné

SETFEEDCOORD	règle la valeur en pourcentage de variation maximale instantanée du feed rate de l'axe
SETOFFSET	règle un offset de cote

Mouvement enchaîné

RATIO	programme le rapport de chaînage d'un axe slave par rapport à son maître
SETDYNRATIO	change de manière dynamique, pendant le mouvement de l'axe maître, le rapport d'enchaînement

Paramètres génériques

DYNLIMIT	active ou désactive le test sur le dépassement des limites de l'axe de manière dynamique
ENABLESTARTCONTROL	active et configure le timeout pour le contrôle du manqué départ ou imprévu arrêt de l'axe
NOTCHFILTER	règle la fréquence de coupe du filtre notch pour l'axe spécifié
RESLIMNEG	désactive la limite négative de l'axe
RESLIMPOS	désactive la limite positive de l'axe
SETADJUST	règle l'ajustement d'un axe
SETBACKLASH	diminue ou élimine les effets des jeux mécaniques sur la trajectoire de l'axe
SETBIGWINFACTOR	modifie le facteur de multiplication pour le calcul de la fenêtre grande sur l'axe choisi
SETDEADBAND	règle les paramètres de voltage minimal pour l'axe indiqué
SETENCLIMIT	règle les limites de connexion codeur erronée
SETINDEXEN	active ou désactive sur l'axe le zérotage de la cote qui correspond au cran de zéro
SETINTEGTIME	règle le nombre de étalons d'erreurs de boucle utilisés pour calculer la composante intégrale
SETIRMPP	règle la vitesse de début de rampe
SETLIMNEG	règle la limite négative de l'axe
SETLIMPOS	règle la limite positive de l'axe
SETMAXER	règle la valeur maximale de poursuite tolérée
SETMAXERNEG	règle la valeur maximale de poursuite tolérée (direction négative)
SETMAXERPOS	règle la valeur maximale de poursuite tolérée (direction positive)
SETMAXERTYPE	règle le type de test sur la servoerror
SETPHASESINV	active ou désactive sur l'axe indiqué l'inversion des phases
SETREFINV	active ou désactive sur l'axe indiqué l'inversion de référence de vitesse
SETRESOLUTION	change la résolution d'un axe

Instructions pour la gestion des Compteurs

DECOUNTER	décrémente un compteur
INCOUNTER	incrémente un compteur
SETCOUNTER	règle un compteur

Instructions pour la gestion des Temporisateurs

HOLDTIMER	bloque un temporisateur
SETTIMER	règle un temporisateur
STARTTIMER	fait partir le temporisateur

Instructions pour la gestion des Matrices

CLEAR	zérotage de variable, vecteur, matrice
FIND	recherche d'un élément
FINDB	recherche d'un élément dans un vecteur ou matrice ordonné de façon croissante
LASTELEM	dernier élément d'un vecteur ou matrice
LOCAL	déclaration de variable, vecteur, matrice locale
MOVEMAT	copie une ligne de matrice dans une autre
PARAM	déclaration d'un paramètre de fonction

SETVAL	modifie une variable
SORT	classement de vecteur ou matrice

Instructions pour la gestion des Chaînes

ADDSTRING	enchaîne deux chaînes
CONTROLCHAR	règle un caractère de contrôle dans une variable chaîne
LEFT	extrait les premiers caractères
LEN	lit la longueur d'une chaîne
MID	extrait des caractères
RIGHT	extrait les derniers caractères
SEARCH	recherche dans une chaîne
SETSTRING	modifie une variable chaîne
STR	convertit de nombre en chaîne
VAL	convertit de chaîne en nombre

Instructions pour la gestion des Communications

CLEARRECEIVE	vide la liste de RECEIVE à satisfaire
COMCLEARRXBUFFER	vide le buffer de réception d'un port sériel
COMCLOSE	ferme un port sériel
COMGETERROR	lit le code d'erreur
COMGETRXCOUNT	lit le nombre d'octets présents dans le buffer de réception
COMOPEN	ouvre un port sériel
COMREAD	lit à partir du port sériel
COMREADSTRING	lit une chaîne à partir du port sériel
COMWRITE	écrit sur le port sériel
COMWRITESTRING	écrit un chaîne sur le port sériel
RECEIVE	réception de données de l'extérieur
SEND	envoi de données à l'extérieur
SENDIPC	envoie une information IPC
WAITIPC	attend l'arrivée d'une information IPC
WAITRECEIVE	réception de données de l'extérieur avec attente

Instructions pour la gestion Mathématique

ABS	valeur absolue
ADD	addition
AND	AND binaire
ARCCOS	arc cosinus
ARCSIN	arc cosinus
ARCTAN	arc tangente
COS	cosinus
DIV	division
EXP	exponentiel
EXPR	reçoit expressions mathématiques
LOG	logarithme naturel
LOGDEC	logarithme à base 10
MOD	module
MUL	multiplication
NOT	NOT binaire
OR	OR binaire
RANDOM	génère un numéro aléatoire
RESETBIT	règle un bit à 0
ROUND	arrondissement
SETBIT	règle un bit à 1
SHIFTL	tourne les bits à gauche
SHIFTR	tourne les bits à droite
SIN	sinus
SQR	racine carrée
SUB	soustraction
TAN	tangente
TRUNC	tronquement
XOR	XOR binaire

Instructions pour la gestion Multitâche

ENDMAIL	signale la fin de l'exécution d'une commande
ENDREALTIMETASK	achève une tâche realtime
ENDTASK	achève une tâche
GETPRIORITYLEVEL	lit le niveau de priorité de la tâche courante
GETREALTIME	retourne le temps passé depuis le début du RealTime axes
GETREALTIMECOUNT	retourne le nombre de RealTime passés
HOLDTASK	suspend l'exécution d'une tâche
RESUMETASK	reprend l'exécution d'une tâche
SENDMAIL	envoie une commande à la boîte aux lettres 'mail'
SETPRIORITYLEVEL	règle le niveau de priorité pour la tâche courante
STARTREALTIMETASK	lance une tâche realtime
STARTTASK	lance l'exécution d'une tâche
STOPTASK	suspend l'exécution d'une tâche et arrête le mouvement des axes associés
WAITMAIL	reçoit une commande de la boîte aux lettres 'mail'
WAITTASK	attend qu'une tâche achève l'exécution

Instructions pour la gestion de Flux

CALL	appel à sous-programme
DELONFLAG	désactive la gestion de l'urgence sur flag bit ou flag switch
DELONINPUT	désactive la gestion de l'urgence sur entrée numérique
ENDREP	fin de répétition de bloc avec REPEAT
FCALL	appel de fonction
FOR	extension de REPEAT
FRET	retour d'appel de fonction
GOTO	saut à étiquette
IF	test sur une variable
IFACC	test si l'axe est en accélération
IFAND	test sur opération de AND
IFBIT	test sur bit
IFBLACKBOX	test si l'enregistrement de l'activité des dispositifs logiques est active.
IFCHANGEVEL	teste si l'axe est en changement de vitesse
IFCOUNTER	test sur un compteur
IFDEC	test si l'axe est en accélération
IFDIR	test sur direction de l'axe
IFERRAN	test sur erreur de boucle
IFERROR	test sur l'erreur de cycle actif
IFFLAG	test sur un flag
IFINPUT	test sur une entrée
IFMESSAGE	test sur le message actif.
IFOR	test sur opération de OR
IFOUTPUT	test sur une sortie
IFQUOTER	test sur cote réelle
IFQUOTET	test sur cote cible
IFRECEIVED	test sur réception de données
IFREG	teste si l'axe est à régime
IFSAME	vérifie que les deux arguments se réfèrent à la même donnée
IFSTILL	teste si l'axe est arrêté
IFSTR	test sur chaîne
IFTARGET	teste si l'axe est en cible
IFTASKHOLD	teste si la fonction parallèle est suspendue
IFTASKRUN	teste si la fonction parallèle est en exécution
IFTIMER	test sur un minuteur
IFVALUE	test sur une variable
IFVEL	test sur la vitesse de l'axe
IFWIN	teste si l'axe est en fenêtre
IFXOR	test sur opération de OR
NEXT	fin de répétition de bloc avec FOR
ONERRSYS	règle l'appel d'une fonction sur une erreur de système
ONFLAG	urgence sur bit indicateur ou flag switch
ONINPUT	urgence sur entrée numérique
REPEAT	répétition d'un bloc d'instructions
RET	retour de sous-programme

SELECT	sélection multiple avec saut
TESTIPC	contrôle la présence d'une information IPC
TESTMAIL	test et réception d'une commande

Instructions Diverses

CLEARERRORS	élimine toutes les erreurs de cycle du module
CLEARMESSAGES	élimine tous les messages du module
DEFMSG	définit un message de groupe
DELAY	bloque la fonction courante pendant un laps de temps
DELERROR	élimine une erreur de cycle précédente
DELMESSAGE	élimine un message précédent
ERROR	envoie une erreur de cycle à l'ordinateur
IFDEF/ELSEDEF/ENDDEF	test pour la compilation conditionnelle
MESSAGE	envoie un message à l'ordinateur
SYSFAULT	désactive le signal de SYSOK
SYSOK	active le signal de SYSOK
TYPEOF	type de l'argument
WATCHDOG	active, met à jour, désactive le chien de garde du GPL sur le module matériel TMSWD

Instructions pour la gestion MECHATROLINK-II

MECCOMMAND	envoie une commande à l'actionnement de l'axe
MECGETPARAM	lit un paramètre de l'axe indiqué.
MECSETPARAM	écrit un paramètre dans l'axe indiqué
MECGETSTATUS	lit les valeurs de STATUS, ALARAM et IO_MON

Instructions pour la gestion des bus de champ standard

AXCONTROL	règle une valeur pour ControlWord
AXSTATUS	rend la valeur contenue dans StatusWor
CNBYDEVICE	rend le numéro de carte et CN d'un dispositif
READDICTIONARY	lit le contenu d'on objet dans le dictionnaire
WRITEDICTIONARY	écrit le contenu d'on objet dans le dictionnaire

Instructions pour la gestion EtherCAT

ACTIVATEMODE	règle un mode d'exploitation
ECATGETREGISTER	rend le contenu d'un registre de l'ESC (EtherCAT Slave Controller)
ECATSETREGISTER	écrit le contenu d'un registre de l'ESC (EtherCAT Slave Controller)
GETPDO	rend un objet dans un PDO EtherCAT
SETEOE	active ou désactive le Sniffer
SETPDO	règle un objet dans un PDO EtherCAT

Instructions pour la gestion du bus CAN

GETCNSTATE	rend l'état du protocole NMT pour un nœud d'une carte CANOpen.
GETSDOERROR	rend la dernière erreur qui s'est produite
GETMNSTATE	rend l'état du protocole NMT pour le nœud maître de la carte CANOpen.
RECEIVEPDO	lit le contenu d'un PDO asynchrone
SENDPDO	écrit le contenu d'un PDO asynchrone
SETNMTSTATE	règle l'état du protocole NMT pour le nœud de la carte CANOpen.

Instructions pour la Simulation

DISABLE	désactive un ou plusieurs axes

DISABLEFORCEDINPUT	active le forcement des entrées
ENABLE	active un ou plusieurs axes
ENABLEFORCEDINPUT	active le forcement des entrées
RESETFORCEDINPUT	force une entrée à OFF
SETFORCEDANALOG	force une entrée numérique
SETFORCEDINPUT	force une entrée à ON
SETFORCEDPORT	force un port d'entrées

Instructions pour les fonctionnalités de "BlackBox"

ENDBLACKBOX	achève la fonctionnalité d'enregistrement
PAUSEBLACKBOX	suspend la fonctionnalité d'enregistrement
STARTBLACKBOX	démarre la fonctionnalité d'enregistrement

Instructions pour la gestion ISO

ISOG0	règle le mouvement rapide
ISOG1	règle le mouvement interpolé
ISOG9	règle l'arrêt forcé du mouvement
ISOG90	règle l'interprétation des cotes comme cotes absolues.
ISOG91	règle l'interprétation des cotes comme cotes absolues.
ISOG93	règle l'interprétation des vitesses comme inverse du temps d'exécution.
ISOG94	règle l'interprétation des vitesses comme unité de mesure par minute.
ISOG216	définit les matrices pour la paramétrisation de la machine
ISOG217	décrit les axes physiques et les axes virtuels qui composent la machine
ISOM2	libère les axes du mouvement ISO
ISOM6	sélectionne les indices des matrices de paramétrisation
ISOSETPARAM	règle des paramètres, qui caractérisent la fluidité du mouvement interpolé ISO
KINEMATICEXPR	règle les expressions individuelles de cinématique inverse et directe

10.3.3 Input/Output

GETFEED

Syntaxe

GETFEED **variable**

Arguments

variable	feed rate

Description

Cette instruction copie la valeur du feed rate, lue par la carte d'I/O à distance, dans la **variable** indiquée. La valeur de feedrate est comprise entre 0 et 100 et est une valeur pourcentage.
Elle intervient sur une entrée analogique non visible dans la configuration.
Toutes les cartes T.P.A qui gèrent le feed rate disposent d'un connecteur, qui leur est consacré.

INPANALOG

Syntaxe

INPANALOG **nominpanalogique, variable**

Arguments

nominpanalogique	nom dispositif type entrée analogique
variable	variable

Description

Cette instruction copie la valeur de l'entrée analogique indiquée de **nominpanalogique** dans la **variable** indiquée.

INPFLAGPORT

Syntaxe

INPFLAGPORT **nomflagport, variable**

Arguments

nomflagport nom de dispositif type port de flag
variable variable

Description

Cette instruction copie l'état du port de flag indiqué par **nomflagport** dans la **variable** indiquée. Le port de flag est considéré tel un masque de bit. Un bit est associé à flag du port. Si un flag si trouve dans l'état "ON", le bit correspondant est réglé à 1.

INPPORT

Syntaxe

INPPORT **nomport, variable**

Arguments

nomport nom de dispositif type port d'entrée
variable variable integer ou char

Description

Cette instruction copie l'état du port d'entrée **nomport** dans la **variable** indiquée.
Le port d'entrée est considéré tel un masque de bit. Si une entrée du port se trouve en état "ON", le bit correspondant est réglé à 1.

MULTIINPPORT

Syntaxe

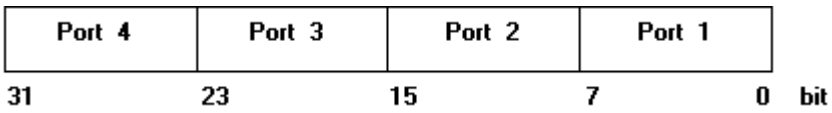
MULTIINPPORT **port1[,...,port4],variable**

Arguments

port1 fournit les bits de 0 à 7
port2 fournit les bits de 8 à 15
port3 fournit les bits de 16 à 23
port4 fournit les bits de 24 à 31
variable variable integer qui reçoive les ports d'entrée

Description

Il lit jusqu'à 4 ports de input simultanément et il les écrit dans une **variable**. La lecture se déroule d'une façon atomique, ce qui garantit que la lecture se déroule à l'intérieur du même real-time. Le port1 correspond au byte plus bas, tandis que le port4 au celui plus haut.



MULTIOUTPORT

Syntaxe

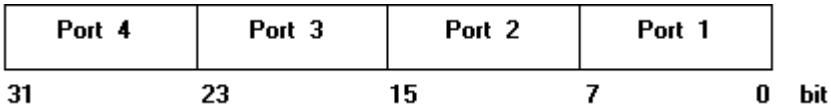
MULTIOUTPORT **valeur, nomport1[,...,nomport4]**

Arguments

valeur? nombre ou variable integer à écrire dans les ports de sortie
nomport1 reçoive les bits de 0 à 7
nomport2 reçoive les bits de 8 à 15
nomport3 reçoive les bits de 16 à 23
nomport4 reçoive les bits de 24 à 31

Description

Il écrit **valeur** en quatre ports de sortie simultanément. L'écriture des ports indiqués se déroule d'une façon atomique, ce qui garantit que la lecture se déroule à l'intérieur du même real-time. Si **nomport2**, **nomport3**, **nomport4** ne sont pas spécifiés, la valeur de son byte est 0.



MULTIRESETFLAG

Syntaxe

MULTIRESETFLAG **masque, nomflag1[, ..., nomflag32]**

Arguments

masque masque des flags concernés - constante ou variable
nomflag1 nom de dispositif type flag

Description

Parmi tous les flags **nomflag** (1÷32), cette instruction désactive, c'est-à-dire qu'elle met sur "OFF", ceux dont le bit est mis sur 1 dans l'argument **masque**.
Le bit 0 de **masque** (celui qui a le poids le plus bas) correspond à **nomflag1**.

MULTIRESETOUT

Syntaxe

MULTIRESETOUT **masque, nomsortie1[, ..., nomsortie32]**

Arguments

masque masque des sorties concernées - constante ou variable
nomsortie1 nom de dispositif type sortie

Description

Parmi toutes les sorties **nomsortie** (1÷32), cette instruction désactive celles dont le bit est mis à 1 dans l'argument **masque**. Le bit 0 de **masque** (celui qui a le poids le plus bas) correspond à **nomsortie1**.

MULTISETFLAG

Syntaxe

MULTISETFLAG **masque, nomflag1[, ..., nomflag32]**

Arguments

masque masque des flags concernés - constante ou variable
nomflag1 nom de dispositif type flag

Description

Parmi tous les flags **nomflag** (1÷32), cette instruction met sur "ON" ceux dont le bit est mis à 1 dans l'argument **masque**. Le bit 0 de **masque** (celui qui a le poids le plus bas) correspond à **nomflag1**.

MULTISETOUT

Syntaxe

MULTISETOUT **masque, nomsortie1[, ..., nomsortie32]**

Arguments

masque masque des sorties concernées - constante ou variable
nomsortie1 nom de dispositif type sortie

Description

Parmi toutes les sorties **nomsortie** (1÷32), cette instruction active celles dont le bit est mis à 1 dans l'argument **masque**.
Le bit 0 de **masque** (celui qui a le poids le plus bas) correspond à **nomsortie1**. Si la sortie est de type monostable, elle est désactivée automatiquement à la fin d'un timeout fixe de 200 millisecondes.

MULTIWAITFLAG

Syntaxe

MULTIWAITFLAG	**masque, flag1[, ..., flag32], état [, timeout [, GOTO étiquette]]**
MULTIWAITFLAG	**masque, flag1[, ..., flag32], état [, timeout [, CALL nomsousprogramme]]**
MULTIWAITFLAG	**masque, flag1[, ..., flag32], état [, timeout [, nomfonction]]**

Arguments

masque	constante ou variable. Masque des flags concernés
flag1[,...flag32]	nom de dispositif de type flag
état	constante prédéfinie. Les valeurs admises sont les suivantes:
ON	état flag actif
OFF	état flag désactif
timeout	constante ou variable. C'est le temps limite d'attente
étiquette	étiquette de saut (GOTO)
nomsousprogramme	étiquette de sous-programme (CALL)
nomfonction	nom de la fonction

Description

L'instruction attend que les flags indiquées de **flag1...flag32** se trouvent dans l'état indiqué par le paramètre **état** (ON/OFF).
Parmi tous les flags, elle contrôle ceux dont le bit est activé (ON) dans l'argument **masque**. Le bit 0 de l'argument **masque** (celui qui a le poids le plus bas) correspond au bit défini par **flag1**, le bit 1 correspond au bit défini par **flag2** et ainsi de suite jusqu'au bit défini par le **flag32**.
Le paramètre **timeout** permet de programmer un timeout différent de celui qui est donné par défaut (une seconde).
Lorsque **étiquette** ou **nomsousprogramme** ou **nomfonction** est présent, le programme saute à **étiquette** ou il appelle **nomsousprogramme** ou **nomfonction** à la fin du timeout.

MULTIWAITINPUT

Syntaxe

MULTIWAITINPUT	**masque, entrée1[, ..., entrée32], état [, timeout [, GOTO étiquette]]**
MULTIWAITINPUT	**masque, entrée1[, ..., entrée32], état [, timeout [, CALL nomsousprogramme]]**
MULTIWAITINPUT	**masque, entrée1[, ..., entrée32], état [, timeout [, nomfonction]]**

Arguments

nomfonction	nom de la fonction
masque	constante ou variable. Masque des entrées concernées
entrée1[,...entrée32]	nom entrée
état	constante prédéfinie. Les valeurs admises sont les suivantes:
ON	état flag actif
OFF	état flag désactif
timeout	constante ou variable. C'est le temps limite d'attente
étiquette	étiquette de saut (GOTO)
nomsousprogramme	étiquette de sous-programme (CALL)

Description

Cette instruction attend que les entrées indiquées de **entrée1...entrée32** se trouvent dans l'état indiqué par le paramètre **état** (ON/OFF).
Parmi toutes les entrées, elle contrôle celles dont le bit est activé (ON) dans l'argument **masque**. Le bit 0 de l'argument **masque** (celui qui a le poids le plus bas) correspond au bit défini par **entrée1**, le bit 1 correspond au bit défini par **entrée2** et ainsi de suite jusqu'au bit défini par **entrée32**.
Si aucun argument n'est spécifié en option, une seconde après le début de l'exécution de l'instruction (timeout par défaut), le message paramétré "Wait input ON/OFF" est généré automatiquement. Le nom de l'entrée qui est signalé est celui de la première entrée activée qui n'a pas satisfait l'état. Si le paramètre **timeout** est présent, le message décrit ci-dessus est généré à la fin du timeout demandé. Si, le timeout s'étant écoulé, la condition requise se présente, un message paramétré est généré automatiquement pour éliminer celui qui avait été envoyé précédemment.
Lorsque **étiquette** ou **nomsousprogramme** ou **nomfonction** est présent, le programme saute à **étiquette** ou il appelle **nomsousprogramme** ou **nomfonction** à la fin du timeout.

OUTANALOG

Syntaxe

OUTANALOG **nomoutanalogique, valeur**

Arguments

nomoutanalogique nom de dispositif type sortie analogique ou axe
valeur constante ou variable

Description

Règle la voltage indiquée par **valeur** sur la sortie analogique ou dans l'axe exprimé par **nomoutanalogique**.

OUTFLAGPORT

Syntaxe

OUTFLAGPORT **nomflagport, valeur**

Arguments

nomflagport nom de dispositif type port de flag
valeur constante ou variable

Description

Cette instruction copie la **valeur** dans le port de flag indiqué par **nomflagport**.
Le paramètre **valeur** est considéré comme un masque de bit. Un flag du port est associé à chaque bit. Si un bit vaut 1, le flag correspondant est réglé à l'état "ON".

OUTPORT

Syntaxe

OUTPORT **nomport, valeur**

Arguments

nomport nom de dispositif type port de sortie
valeur constante ou variable, integer ou char

Description

Cette instruction copie la **valeur** dans le port de sortie **nomport**.
Le port de sortie est considéré comme un masque de bit. Si un bit vaut 1, la sortie correspondante est réglée sur "ON".

RESETFLAG

Syntaxe

RESETFLAG **nomflag**

Arguments

nomflag nom de dispositif type flag

Description

Cette instruction désactive (met sur OFF) le flag **nomflag**.

RESETOUT

Syntaxe

RESETOUT **nomsortie**

Arguments

nomsortie nom de dispositif type sortie numérique

Description

Cette instruction désactive (met sur OFF) la sortie **nomsortie**.

SETFLAG

Syntaxe

SETFLAG	**nomflag**

Arguments

nomflag	nom de dispositif type flag

Description

Cette instruction active (met sur ON) le flag **nomflag**.

SETOUT

Syntaxe

SETOUT	**nomsortie**

Arguments

nomsortie	nom de dispositif type sortie numérique

Description

Cette instruction active (met sur ON) la sortie **nomsortie**.
Si la sortie est configurée monostable, elle est désactivée automatiquement à la fin d'un timeout fixe de 200 millisecondes.

WAITFLAG

Syntaxe

WAITFLAG	**nomflag, état [, timeout [, GOTO étiquette]]**
WAITFLAG	**nomflag, état [, timeout [, CALL nomsousprogramme]]**
WAITFLAG	**nomflag, état [, timeout [, nomfonction]]**

Arguments

nomflag	nom de dispositif type flag
état	constante prédéfinie. Les valeurs admises sont les suivantes : - **ON** état du flag actif - **OFF** état du flag non actif
timeout	constante ou variable. C'est le temps limite d'attente
étiquette	étiquette de saut (GOTO)
nomsousprogramme	étiquette de sous-programme (CALL)
nomfonction	nom de la fonction

Description

Cette instruction attend que le flag **nomflag** prenne l'état indiqué par le paramètre **état** (ON/OFF).
Si **timeout** est le seul des arguments facultatifs présent, l'erreur de cycle "Flag **nomflag** en attente d'**état**" est généré à la fin du timeout.
Si la condition a lieu après la fin du timeout, l'erreur de cycle envoyée précédemment par cette tâche est éliminée automatiquement.
Lorsque **étiquette** ou **nomsousprogramme** ou **nomfonction** est présent, le programme saute à a **étiquette** ou il appelle **nomsousprogramme** ou **nomfonction** à la fin du timeout sans générer aucun message automatique.

Remarque

Afin d'éviter une situation d'attente d'un flag pendant un cycle de travail, il est conseillé de programmer un timeout.

WAITINPUT

Syntaxe

WAITINPUT	**nomentrée, état [, timeout [, GOTO étiquette]]**
WAITINPUT	**nomentrée, état [, timeout [, CALL nomsousprogramme]]**
WAITINPUT	**nomentrée, état [, timeout [, nomfonction]]**

Arguments

nomentrée	nom entrée
état	constante prédéfinie. Les valeurs admises sont les suivantes : **- ON** état de l'entrée : actif - **OFF** état de l'entrée : non actif
étiquette	étiquette de saut (GOTO)
nomsousprogramme	étiquette de sous-programme (CALL)
nomfonction	nom de la fonction

Description

Cette instruction attend que l'entrée **nomentrée** prenne l'état indiqué par le paramètre **état** (ON/OFF).
Si aucun argument facultatif n'est spécifié, 20 secondes après le début de l'exécution de l'instruction, l'erreur de cycle : "Entrée numérique **nomentrée** en attente d'**état**" est générée automatiquement.
Si **timeout** est le seul des arguments facultatifs présent, le message mentionné ci-dessus est généré à la fin du timeout.
Si la condition a lieu après la fin du timeout, l'erreur de cycle envoyée précédemment par cette tâche est éliminée automatiquement.
Lorsque **étiquette** ou **nomsousprogramme** ou **nomfonction** est présent, le programme saute à a **étiquette** ou il appelle **nomsousprogramme** ou **nomfonction** à la fin du timeout sans générer aucun message automatique.

Remarque

Afin d'éviter une situation d'attente d'un signal d'entrée, pendant un cycle de travail, il est conseillé de programmer un timeout inférieur au temps donné tempo par défaut (20 secondes).

Exemple

Routine de Zérotage d'un axe

WAITPERSISTINPUT

Syntaxe

WAITPERSISTINPUT **nomentrée, état, timepersist [, timeout [, GOTO étiquette]]**
WAITPERSISTINPUT **nomentrée, état, timepersist [, timeout [, CALL nomsousprogramme]]**
WAITPERSISTINPUT **nomentrée, état, timepersist [, timeout [, nomfonction]]**

Arguments

nomentrée	nom de dispositif type entrée numérique
état	constante prédéfinie. Les valeurs admises sont les suivantes : **-ON** état de l'entrée : actif **-OFF** état de l'entrée : non actif
timepersist	constante ou variable
timeout	constante ou variable. C'est le temps limite d'attente
étiquette	étiquette de saut (GOTO)
nomsousprogramme	étiquette de sous-programme (CALL)
nomfonction	nom de la fonction

Description

Cette instruction attend que l'entrée **nomentrée** se trouve dans l'état indiqué par le paramètre **état** (ON/OFF) et reste dans cet état pendant le temps indiqué dans **timepersist** (unité de mesure: secondes).
Si aucun argument facultatif n'est spécifié, 20 secondes après le début de l'exécution de l'instruction, l'erreur de cycle : "Entrée numérique **nomentrée** en attente d'**état**" est générée automatiquement.
Si **timeout** est le seul des arguments facultatifs présent, le message mentionné ci-dessus est généré à la fin du timeout.
Si la condition a lieu après la fin du timeout, l'erreur de cycle envoyée précédemment par cette tâche est éliminée automatiquement.
Lorsque **étiquette** ou **nomsousprogramme** ou **nomfonction** est présent, le programme saute à a **étiquette** ou il appelle **nomsousprogramme** ou **nomfonction** à la fin du timeout sans générer aucun message automatique.

Remarque

Afin d'éviter une situation d'attente d'un signal d'entrée, pendant un cycle de travail, il est conseillé de programmer un timeout inférieur au temps donné tempo par défaut (20 secondes).

nom entrée
état

10.3.4 Axes

CHAIN

Syntaxe

CHAIN **axe_master, axe_slave1 [, ...axe_slave5]**

Arguments

axe_master	nom du dispositif type axe qui servira de master
axe_slave1...axe_slave5	nom du dispositif type axe qui servira de slave

Description

Après l'exécution de cette instruction, les axes_**slave** (1÷5) exécuteront des mouvements liés à ceux de l'axe master par le rapport de chaînage programmé avec l'instruction RATIO. Il y a donc chaînage aussi bien des mouvements point-point que des mouvements interpolés.
L' **axe_slave1** n'est pas un paramètre facultatif, mais il doit toujours être défini.
Pour pouvoir être enchaîné, l'axe slave ne doit pas être engagé dans une interpolation et il ne peut être, à son tour, le master d'autres slaves.
L'axe master ne peut pas être, à son tour, slave d'autres axes.
L'enchaînement peut être effectué aussi bien avec les axes en cote qu'avec les axes en mouvement.
Pour désactiver l'enchaînement des axes, il suffit d'exécuter l'instruction NORMAL sur l'axe master.
Cette dernière opération peut être effectuée aussi bien avec les axes en cote qu'avec les axes en mouvement. Lorsque l'enchaînement est désactivé pendant un mouvement des axes, l'axe slave effectue une rampe de décélération et il s'arrête.
Il es possible de definir un maximum de 8 axes master maîtres en même temps.
On peut exécuter cette instruction-ci aussi avec des axes pas-pas (stepper) à condition qu'ils soient gérés par TRS_AX.
En outre, tous les axes doivent posséder en codeur réale et non simulé. Dans le cas contraire l'erreur de système "4101 – Gestion non congruente de l'axe NomAxe" est générée.
Voir également RATIO.

Exemple

```
; enchainer l'axe Y à l'axe X
CHAIN           X, Y
; déplacement de l'axe X.
; Y réplique le mouvement de X
MOVINC          X, 100
```

CIRCABS

Syntaxe

CIRCABS **[étiquette],axe1, cote1, axe2, cote2, sens, ±rayon [, angle]**

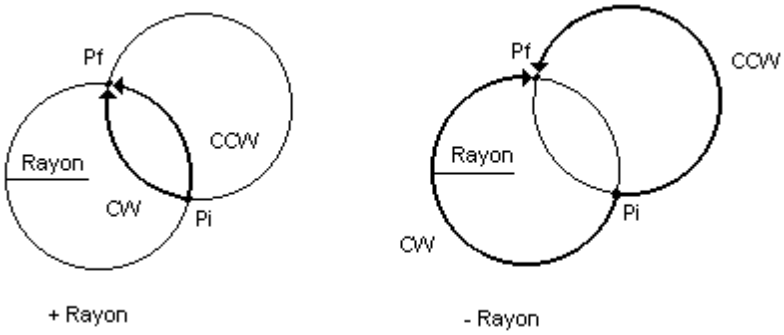
Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1, axe2	noms de dispositifs type axe
cote1, cote2	constante ou variable. Représente la cote de déplacement absolu costante
sens	variabile integer. Précise le type de rotation, les valeurs admissibles sont les suivantes : **CW** dans le sens des aiguilles d'une montre **CCW** dans le sens inverse des aiguilles d'une montre
rayon	constante ou variable. Représente la valeur du rayon du cercle
angle	constante ou variable. Représente l'angle de départ

Description

Interpolation circulaire à 2 axes avec *déplacement absolu* basé sur les cotes **cote1** et **cote2** programmées.
L'arc est déterminé par le point initial (point courant), par le point final, par la valeur du **rayon** et par le **sens** de parcours.
Le signe appliqué au **rayon** permet de sélectionner l'arc mineur (+rayon) ou l'arc majeur (-rayon).
Dans le cas particulier où la cote initiale de l'axe 1 coïncide avec la cote finale **cote1** et la cote initiale de l'axe2 coïncide avec la cote finale **cote2**, il s'agit d'un cercle complet : dans ce cas, il est nécessaire d'indiquer l'argument **angle**, avec la même signification que l'instruction CIRCLE (à laquelle l'on se réfère).

Le paramètre angle sert à déterminer sans équivoque le centre du cercle, avec la même signification que l'instruction CIRCLE. Il n'est pris en ligne de compte que lorsque, avant l'exécution de l'instruction, **cote1** et **cote2** correspondent aux cotes courantes des axes. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX.
Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l'axe.



CIRCINC

Syntaxe

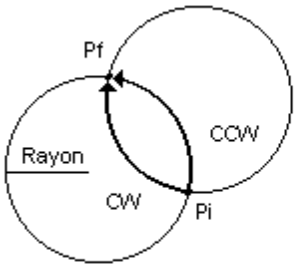
CIRCINC **[étiquette],axe1, cote1, axe2, cote2, sens, ±rayon [, angle]**

Arguments

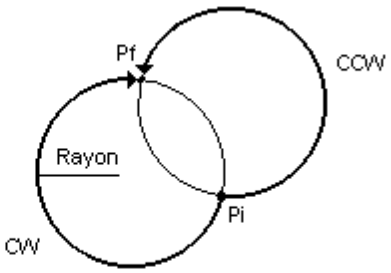
étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1, axe2	noms de dispositifs type axe
cote1, cote2	constante ou variable. Représente la cote de déplacement incrémentiel
sens	variable integer. Précise le type de rotation, les valeurs admissibles sont les suivantes : **CW** dans le sens des aiguilles d'une montre **CCW** dans le sens inverse des aiguilles d'une montre
rayon	constante ou variable. Représente la valeur du rayon du cercle
angle	constante ou variable. Représente l'angle de départ

Description

Interpolation circulaire à 2 axes avec *déplacement incrémentiel* basé sur les cotes **cote1** et **cote2** programmées.
L'arc est déterminé par le point initial (point courant), par le point final, par la valeur du **rayon** et par le **sens** de parcours.
Le signe appliquée au **rayon** permet de sélectionner l'arc mineur (+rayon) ou l'arc majeur (-rayon).
Dans le cas particulier où cote1 = cote2 = 0, il s'agit d'un cercle complet : dans ce cas, il est nécessaire d'indiquer l'argument **angle**, avec la même signification que l'instruction CIRCLE (à laquelle l'on se réfère).
Le paramètre angle sert à déterminer sans équivoque le centre du cercle, avec la même signification que l'instruction CIRCLE. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l' axe.



+ Rayon



- Rayon

CIRCLE

Syntaxe

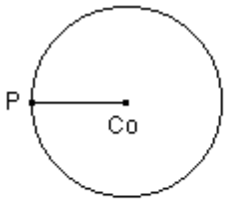
CIRCLE **[étiquette],axe1, axe2, sens, rayon, angle**

Arguments

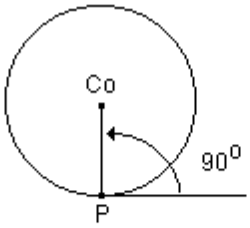
étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1, axe2	nos de dispositifs type axe
sens	variable integer. Précise le type de rotation, les valeurs admissibles sont les suivantes : **CW** dans le sens des aiguilles d'une montre **CCW** dans le sens inverse des aiguilles d'une montre
rayon	constante ou variable. Représente la valeur du rayon du cercle
angle	constante ou variable. Représente l'angle de départ

Description

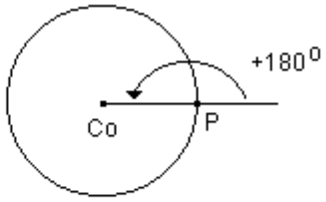
Interpolation circulaire complète.
Cette instruction crée un cercle avec **axe1** et **axe2**, dans le **sens** indiqué, avec un **rayon** égal à la valeur attribuée et en fonction de l'**angle** de départ programmé.
Rayon ne peut prendre que des valeurs positives.
L'**angle** doit être indiqué avec une convention trigonométrique positive dans le sens inverse des aiguilles d'une montre en partant de l'axe X. En effet, la position du centre C_0 du cercle reste déterminée en précisant l'angle formé par le rayon passant par le point initial P programmé (point courant) et la direction horizontale X+. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l' axe.



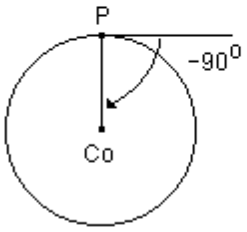
Angle = 0



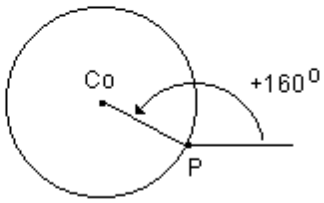
Angle = +90°



Angle = +180°



Angle = -90° (o +270°)



Angle = +160°

COORDIN

Syntaxe

COORDIN **matrice, valeur deltaT, direction, ini, fin, masque, axe1, n° _colonne_axe1**
[, (axe2, n°_colonne_axe2) ÷ (axe32, n°_colonne_axe32)]

Arguments

matrice	matrice de données
valeur deltaT	constante ou variable. Base temps.
direction	constante prédéfinie. Direction de lecture des données dans la matrice **UP** de la ligne plus basse à la plus haute **DOWN** de la ligne plus haute à la p,us basse
ini	variable globale integer. C'est le numéro de la ligne initiale
fin	variable globale integer. C'est le numéro de la ligne finale
masque	masque des axes à activer
axe1 [...axe32]	noms de dispositifs type axe
n° colonne axe1 [...n colonne axe32]	numéro de la colonne matrice se référant à l'axe

Description

Cette instruction permet d'effectuer des mouvements synchronisés des axes **axe1**, **axe2**, etc., à travers des déplacements incrémentiels (microvecteurs) définis par une **matrice** de données.
Les paramètres **axe1** et **n_colonne_axe1** doivent être toujours définis.
Les valeurs contenues dans la **matrice** indiquent les cotes absolues au fur et à mesure qu'elles sont atteintes par les différents axes.
Les déplacements incrémentiels relatifs (différence des cotes entre la ligne (n) et la ligne (n-1)) sont exécutés dans un laps de temps égal à un **multiple** de la base temps (1 ms = real time de rafraîchissement des axes), défini par l'argument **valeur** Δ**t**, qui doit donc être exprié par un compteur.
Cette valeur de temps étant préétablie, l'entité des différents déplacements de chaque axe en détermine la vitesse. Cette instruction assure le déplacement coordonné d'un maximum de 32 axes le long de n'importe quel parcours curviligne dans l'espace, comme l'un de ceux qui sont générés avec les techniques SPLINE.
Il n'est pas nécessaire d'attendre que l'instruction soit complétée et l'instruction STARTINTERP n'est pas nécessaire pour le démarrage. Cependant, il faut poser une instruction WAITSILL à sa fin pour attendre la correcte arrivée à la cote des axes. Eventuelles modifications de feedrate override doivent être faites au moyen de l'instruction SETFEDI et gérées au moyen l'instruction SETFEEDCORD.

Les constantes **direction** permet de déterminer la direction de parcours de la matrice, ce qui permet d'exécuter la trajectoire dans les deux sens.
Les colonnes de la matrice à analyser peuvent être de type float ou de type double, mais pas des deux types en même temps.
Outre le déplacement des axes long un parcours fini (défini par le nombre de lignes de la matrice) il est possible de réaliser un déplacement infini en utilisant:

- une matrice d'une ligne seulement. Avec cette modalité de fonctionnement le contrôle lit toujours la seule ligne de la matrice et applique aux axes les cotes qui sont ici écrites. Afin que les axes se déplacent il sera nécessaire modifier opportunément la ligne de la matrice, de préférence en utilisant une tâche real-time garantissant la synchronisation de la mise à jour des cotes avec la fréquence de rafraîchissement des axes. Dans cette façon on peut réaliser cames électroniques ou enchaînements avec rapport divers de 1:1. Pour activer cette modalité de fonctionnement on définira **ini** = 1**, fin** = 0 et **direction** = UP. Quand on utilise cette modalité de fonctionnement on NE doit PAS utiliser l'instruction STOP.
- une matrice de plus lignes. Il est possible de scander la matrice avec boucles depuis la première ligne jusqu'au la dernière ligne en programmant les valeurs **ini** = 1, **fin** = 0 et **direction** = UP. Dans le cas où on veut exécuter une ligne seulement d'une matrice multilignes, il est nécessaire de programmer les paramètres **ini**, **fin** et **direction** dans la suivante façon : **ini** = numéro ligne qui on veut exécuter, **fin** = numéro ligne précédente la ligne que on veut exécuter, **direction** = UP. Dans le cas différents il est généré une erreur de système.
  Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX.

DISABLECORRECTION

Syntaxe

DISABLECORRECTION **axe [, axe1, ..., axe6]**

Arguments

axe	noms de dispositifs type axe
axe1, ..., axe6	noms de dispositifs type axe

Description

Cette instruction désactive la correction linéaire pour l'axe indiqué.
Le premier paramètre est l'axe pour lequel on veut désactiver la correction. S'il s'agit du seul paramètre précisé, toutes les corrections présentes dans la configuration sont désactivées. Les paramètres suivants permettent de préciser quelles corrections on désire désactiver. Si l'un d'eux coïncide avec le premier paramètre, l'autocorrection est désactivée.
Pour avoir une description plus détaillée, voir ENABLECORRECTION.

Exemple

```
; désactive uniquement l'autocorrection pour l'axe X
DISABLECORRECTION     X, X

; désactive la correction croisée (vers X et Y) pour l'axe Z,
; mais pas l'autocorrection
DISABLECORRECTION     Z, X, Y
```

EMERGENCYSTOP

Syntaxe

EMERGENCYSTOP **axe, temps**

Arguments

axe	nom du dispositif type axe
temps	constante ou variable integer. Temps de ramp en ms

Description

Il arrête l'axe spécifié et éventuellement avec lui tous ceux impliqués dans le mouvement interpolé.
Le mouvement est arrêté avec une rampe de décélération dans le temps indiqué par la variable **[temps].**
Dans les mouvement point-point, si le temps configuré est supérieur au temps de décélération indiquée dans la configuration, ce dernier est utilisé.
Dans le mouvements interpolés, si le temps configuré est supérieur à la valeur maximale de décélération de tous les axes impliqués, on utilise le temps maximal configuré dans la configuration.

Le mouvement pourra être repris par une instruction START.
Cette instruction ne peut pas être utilisée, si **[axe]** est un axe esclave.
Cette instruction peut générer les erreurs de système suivants :
- "4101 – Gestion non congruente de l'axe" lorsque **[axe]** est en train d'exécuter un mouvement synchronisé ou une interpolation multilinéaire ou un mouvement ISO.
- "4105 – Instruction non exécutable sur l'axe" lorsque **[axe]** est un axe de comptage.
- "4399 – Paramètre hors de la plage" si le **[temps]** indiqué est égal ou inférieur à 0.

ENABLECORRECTION

Syntaxe

ENABLECORRECTION **axe [, axe1, ..., axe6]**

Arguments

axe	noms de dispositifs type axe
axe1, ..., axe6	noms de dispositifs type axe

Description

Cette instruction active la correction linéaire pour l'axe indiqué. La correction se compose de l'autocorrection et de la correction croisée. L'autocorrection est une correction de la cote réelle d'un axe en fonction de sa position, la correction croisée est une correction de la cote réelle d'un axe en fonction de la position d'autres axes. Il est possible de définir un maximum de cinq correcteurs croisés.
Le premier paramètre est l'axe pour lequel on veut activer la correction. S'il s'agit du seul paramètre précisé, toutes les corrections présentes dans la configuration sont activées.
Les paramètres suivants permettent de préciser quelles corrections on désire activer. Si l'un d'eux coïncide avec le premier paramètre, l'autocorrection est activée.
Voir également DISABLECORRECTION.

REMARQUE: Afin que l'instruction soit exécutée, la correction doit être activée également dans la configuration.

Exemple

```
;active toutes les corrections prévues dans la configuration pour l'axe X
ENABLECORRECTION      X

;active uniquement l'autocorrection pour l'axe X
ENABLECORRECTION      X, X

;active l'autocorrection et la correction croisée (vers X et Y)
;pour l'axe Z
ENABLECORRECTION      Z, X, Y, Z
```

ENDMOV

Syntaxe

ENDMOV **axe [, cote]**

Arguments

axe	noms de dispositifs type axe
cote	constante ou variable.

Description

Cette instruction arrête le mouvement de l'axe indiqué. Elle diffère d'une instruction STOP, par le fait que le mouvement est achevé et qu'il ne pourra pas être repris par une éventuelle instruction START.
Si cela est spécifié, le paramètre **cote** permet de régler la position où l'axe achève le mouvement, autrement le point où l'axe s'arrête dépend de la vitesse courante et de la dernière décélération programmée. Si nécessaire, pour atteindre le point de fin de mouvement, le contrôle effectue une inversion de mouvement de l'axe.

Remarque

Le paramètre cote est utilisé seulement s'il s'agit d'un mouvement point-point. En cas d'un mouvement rapide, le mouvement de l'axe s'arrête sans tenir compte de la valeur de la **cote**.

Exemple

```
; arrête le mouvement courant et mène l'axe à la cote 0.0
ENDMOV    X, 0.0
```

FASTREAD

Syntaxe

FASTREAD	**axe1, état, variable1 [,axe2, variable2],[..., axe8, variable8]**

Arguments

axe1...[...axe8]	noms de dispositifs type axe. Axe1 est l'axe master.
état	constante prédéfinie. Prend les valeurs suivantes: **ON** front de montée **OFF** front de descente
variable1...[...variable8]	variable ou élément de matrice/vecteur type double. Cote enregistrée

Description

Les cotes des axes indiqués sont lues et enregistrées dans les **variables** au moment où l'entrée rapide de l'**axe1** (axe Master) commute dans l'état programmé.

Si les axes indiqués sont analogiques, elles doivent appartenir à la même carte (4 pour TRS-AX).
Si les axes indiqués sont digitaux, le signal d'entrée se trouve directement sur l'actionnement; donc, en cas de FASTREAD multiple le signal doit être connecté en parallèle sur les différents dispositifs.
Avec le bus EtherCAT, le nombre maximum d'axes autorisé est réduit d'un pour chaque axe faisant l'objet d'un échange codeur (voir instruction SWITCHENC). Si cette limite est dépassée, l'erreur système "4400 Trop d'axes actifs en FASTREAD" est générée. En outre, le codeur supplémentaire doit être raccordé à une extension TRS-AC-E sur Trs-CAT. Sinon, l'erreur système "4375 FASTREAD exécuté sur des axes de cartes différentes" est générée.
L'instruction s'achève lorsque l'entrée commute dans l'**état (ON/OFF)** indiqué.
Si une instruction STOP est exécutée avant la commutation de l'entrée rapide, ces instructions restent actives et elles reprennent après l'instruction START.
Il est possible d'activer plusieurs lectures rapides en même temps sur le même carte d'axes.

Pendant l'exécution de l'instruction il n'est pas possible d'exécuter dans le même temps des instructions SETPZERO et SETPFLY sur le même axe, s'il est connecté à des cartes avec bus MECHATROLINK-II.

Remarque

L'entrée rapide pour axes de type digital sur carte avec bus Mechatronink II est présente sur l'entrée **EXTI2** et il n'est pas nécessaire de la configurer en virtuel physique. Les entrées rapides des axes digitaux MECHATROLINK-II doivent être court-circuiter parce-que la mémorisation de la cote d'un axe est fait seulement en référence à la propre entrée rapide.
L'entrée rapide est celle qui est présente sur le connecteur de l'**axe1** et il n'est pas nécessaire de la configurer en virtuel-physique.

FREE

Syntaxe

FREE	**axe [, voltage]**

Arguments

axe	nom de dispositif type axe
voltage	constante float ou variable float. Voltage de référence

Description

Cette instruction met **l'axe** en condition de "boucle ouvert" (Free), en désactivant ainsi son contrôle de position. Si l'**axe** est esclave en enchaînement avec d'autres axes, la contrainte est rompue et le déplacement de l'**axe** est arrêté.
Si le paramètre **voltage** est réglé, le voltage de référence de l'axe prend sa valeur.
Elle ne peut être utilisée que pour les axes de mesure des cotes ou pour les axes dont le mouvement peut être forcé par des organes mécaniques externes qui en altéreraient la position.
Pendant le fonctionnement, la cote de l'axe est mesurée et actualisée, ce qui permet de positionner l'axe de façon absolue après la réactivation du contrôle de position (instruction NORMAL).

HELICABS

Syntaxe

HELICABS **[étiquette],axe1, cote1, axe2, cote2, axe3, cote3, sens, ±rayon[,angle [, nombtours [, axe4, cote4 [, ..., axe6, cote6]]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1...axe3[...axe6]	noms de dispositifs type axe
cote1...cote3[...cote6]	constante ou variable. Cote de déplacement absolu
sens	variable integer. Type de rotation dans le sens des aiguilles d'une montre/sens contraire (CW/CCW)
rayon	constante ou variable. Rayon de l'hélice
angle	constante ou variable. Angle de départ
nombtours	constante ou variable. Nombre de tours

Description

Interpolation hélicoïdale avec déplacement absolu égal aux cotes **cote1**, **cote2** et **cote3** programmées. Le mouvement se constitue d'une interpolation circulaire associée aux axes **axe1** et **axe2** (avec les mêmes règles syntaxiques que CIRCABS / CIRCINC, en ce qui concerne les arguments **sens**, **±rayon** et **angle**), et d'une linéaire associée à l'**axe3** (éventuellement **axe4**, **axe5** et **axe6**). Le mouvement hélicoïdal peut se développer sur plusieurs tours indiqués par l'argument **nombtours**. La cote relative à l'axe à mouvement linéaire (comme les éventuelles cotes **axe4**, **axe5**, **axe6**) se réfère au déplacement total (et donc pas au déplacement/tour). Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l' axe.

1. Remarque

1. La condition de contournage est évaluée seulement sur les premiers trois axes, ceux qui forment le système de référence. En ajoutant un successif et éventuellement en le modifiant on a une erronée gestion du profil de vitesse. Pour obtenir un mouvement correct, il faut interposer une instruction WAITSTILL entre une instruction HELICABS et l'autre.
2. se on établit un système de référence locale en utilisant l'instruction SETRIFLOC, les trois axes qui définissent le nouveau système de référence doivent toujours être indiqués entre les paramètres de l'instruction HELICABS même si ils n'exécutent pas déplacements.

HELICINC

Syntaxe

HELICINC **[étiquette],axe1, cote1, axe2, cote2, axe3, cote3, sens, ± rayon [,angle [,nombtours [, axe4, cote4 [, ..., axe6, cote6]]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1...axe3[...axe6]	noms de dispositifs type axe
cote1...cote3[...cote6]	constante ou variable. Cote de déplacement incrémentiel
sens	variable integer. Type de rotation dans le sens des aiguilles d'une montre/sens contraire (CW/CCW)
rayon	constante ou variable. Rayon de l'hélice
angle	constante ou variable. Angle de départ
nombtours	constante ou variable. Nombre de tours

Description

Interpolation hélicoïdale avec déplacement incrémentiel égal aux cotes **cote1**, **cote2** et **cote3** programmées.
Le mouvement se constitue d'une interpolation circulaire associée aux axes **axe1** et **axe2** (avec les mêmes règles syntaxiques que CIRCABS /CIRCINC, , en ce qui concerne les arguments **sens**, **±rayon** et **angle**), et d'une linéaire associée à l'**axe3** (éventuellement **axe4**, **axe5** et **axe6**).
Le mouvement hélicoïdal peut se développer sur plusieurs tours indiqués par l'argument **nombtours**.

La cote relative à l'axe à mouvement linéaire (comme les éventuelles cotes **axe4**, **axe5**, **axe6**) se réfère au déplacement total (et donc pas au déplacement/tour). Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l' axe.

Remarque

1. La condition de contournage est évaluée seulement sur les premiers trois axes, ceux qui forment le système de référence. En ajoutant un successif et éventuellement en le modifiant on a une erronée gestion du profil de vitesse. Pour obtenir un mouvement correct, il faut interposer une instruction WAITSTILL entre une instruction HELICINC et l'autre.
2. se on établit un système de référence locale en utilisant l'instruction SETRIFLOC, les trois axes qui définissent le nouveau système de référence doivent toujours être indiqués entre les paramètres de l'instruction HELICINC même si ils n'exécutent pas déplacements.

JERKCONTROL

Syntaxe

JERKCONTROL **axe, état**

Arguments

axe	nom de dispositif type axe
état	constante prédéfinie. Les valeurs admises sont les suivantes: **ON** état flag actif **OFF** état flag désactif

Description

Cette instruction active (si le paramètre **état** est réglé sur ON) ou désactive (si le paramètre **état** est réglé sur OFF) le contrôle du jerk sur les mouvements en interpolation et point-point de l'**axe**. Le contrôle du jerk est habilité seulement avec les axes qui ont configuré une rampe d'accélération et décélération à Esse. Si l'axe a configuré une rampe Linéaire le contrôle du jerk n'est pas effectué.

JERKSMOOTH

Syntaxe

JERKSMOOTH **axe, valeur**

Arguments

axe	noms de dispositifs type axe
valeur	constante ou variable type float.

Description

Au cours des mouvements interpolés classiques les axes peuvent se déplacer en conditions de contournage, c'est à dire sans s'arrêter entre deux blocs de déplacement consécutifs, quand l'angle entre les tangentes à la trajectoire est inférieur au paramètre "Angle maximal contour" (valeur par défaut 15 degrés, qu'on peut modifier avec instruction SETCONTORNATURE). Dans le cas contraire les axes s' arrêtent sur le point d' arête des deux blocs avec décélération contrôlée et repartent le long du nouveau bloc avec accélération et vitesse contrôlées. Cependant arrêt et départ réduisent les performances du mouvement de la machine. Dans les cas où l' angle de contournage prend des valeurs consistantes, par exemple une valeur de discontinuité de tangence supérieure à 5 degrés, des remarquables sauts de vitesse se produisent pour les axes de mouvement, avec conséquentes valeurs infinies d'accélération, jerk et remarquables contraintes mécaniques, qui se peuvent répercuter aussi sur la qualité de l'usinage.
L'instruction JERKSMOOTH permet en regard d'une valeur décidée de l'utilisateur, de raccorder d'une façon lisse, ou plutôt avec continuité d'accélération et de vitesse, les profils de vitesse des axes dans les mouvements en contournage. Il faut remarquer que ce raccord lisse insère des petites changements dans la trajectoire exécutée, par rapport à la trajectoire théorique, puisque'autour du point du contournage les axes présentent un profil de vitesse différent du celui théorique.
La variable **valeur** qui est exprimée par une valeur de pourcentage entre 0 et 100, définit la valeur du raccord de façon lisse des profils de vitesse. Une valeur équivalente à 0 maintient le profil théorique, en créant des discontinuités dans les accélérations et dans les profils de vitesse. Une valeur équivalente à 100 obtient des profils raccordés doucement, une meilleur performance, mais

aussi un écart maximal de la trajectoire théorique, proportionnel à la vitesse le long de la trajectoire.

Note

L'instruction ne peut être appliquée que dans les mouvements avec interpolation classique (instructions LINEARABS, LINEARINC, CIRCABS, CIRCINC, HELICABS, HELICINC). Elle n'est pas appliquée en mouvements d'interpolation multiaxe (instruction MULTIABS et MULTIINC).

LINEARABS

Syntaxe

LINEARABS **[étiquette],axe1, cote1, [axe2, cote2 [, axe3, cote3 [, ..., axe6, cote6]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1[...axe2[...axe6]]	noms de dispositifs type axe
cote1[...cote2[...cote6]]	constante ou variable. Cote de déplacement absolu

Description

Interpolation linéaire, avec *déplacement absolu*, aux cotes indiquées par **cote1**, **cote2,** etc. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l'axe.

Remarque

1. La condition de contournage est évaluée seulement sur les premiers trois axes, ceux qui forment le système de référence. En ajoutant un successif et éventuellement en le modifiant on a une erronée gestion du profil de vitesse. Pour obtenir un mouvement correct, il faut interposer une instruction WAITSTILL entre une instruction LINEARABS et l'autre.
2. se on établit un système de référence locale en utilisant l'instruction SETRIFLOC, les trois axes qui définissent le nouveau système de référence doivent toujours être indiqués entre les premiers trois paramètres de l'instruction LINEARABS même si ils n'exécutent pas déplacements.

LINEARINC

Syntaxe

LINEARINC **[étiquette],axe1, cote1, [axe2, cote2 [, axe3, cote3 [, ..., axe6, cote6]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1...axe2[...axe6]	noms de dispositifs type axe
cote1...cote2[...cote6]	constante ou variable. Cote de déplacement incrémentiel

Description

Interpolation linéaire, avec *déplacement incrémentiel*, égal aux cotes indiquées par **cote1**, **cote2,** etc. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l'axe.

Remarque

1. La condition de contournage est évaluée seulement sur les premiers trois axes, ceux qui forment le système de référence. En ajoutant un successif et éventuellement en le modifiant on a une erronée gestion du profil de vitesse. Pour obtenir un mouvement correct, il faut interposer une instruction WAITSTILL entre une instruction LINEARINC et l'autre.

2. se on établit un système de référence locale en utilisant l'instruction SETRIFLOC, les trois axes qui définissent le nouveau système de référence doivent toujours être indiqués entre les paramètres de l'instruction LINEARINC même si ils n'exécutent pas déplacements.

MOVABS

Syntaxe

MOVABS **axe1, valeur1 [, axe2, valeur2 [, ..., axe6, valeur6]]**

Arguments

axe1...[...axe6] noms de dispositifs type axe
valeur1...[...valeur6] constante ou variable. Cote de déplacement absolu

Description

Cette instruction fait exécuter aux axes indiqués un *mouvement absolu* égal aux cotes indiquées par **valeur1 [,...valeur6]**.
Pour exécuter le mouvement, l'axe ne doit pas être engagé dans un mouvement interpolé et il doit se trouver en cote ou en fenêtre. Le mouvement de l'axe commence dès que l'instruction est exécutée. Si plusieurs instructions de déplacement point-point sont exécutées dans une même tâche, elles sont enchaînées. Si une deuxième tâche tente d'exécuter des instructions point-point sur un axe qui est engagé dans un autre mouvement, cette tâche reste à attendre que le mouvement commandé par la première tâche soit achevé.
Il est également possible de modifier la vitesse entre une mouvement point-point et le suivant avec une instruction SETVEL. Les deux mouvements seront raccordés par une rampe de vitesse sans arrêter les axes. S'il n'est pas utilisé l'instruction SETVEL, la vitesse maximale possible est représentée par la valeur de vitesse manuelle définie en configuration.
Un mouvement point-point peut être interrompu avec l'instruction STOP et être repris par la suite avec l'instruction START. Pendant l'interruption du mouvement, l'axe reste en état "à régime" même s'il ne bouge pas matériellement.
Un mouvement peut être avorté avec l'instruction ENDMOV. Dans ce cas, il ne peut plus être repris.

REMARQUE:

1) Auparavant, les mouvements point-point :
- ne permettaient pas de modifier la vitesse si l'axe n'était pas arrêté. Le comportement actuel est semblable à celui des mouvements interpolés.
- lorsqu'ils étaient interrompus par une instruction STOP, l'axe correspondant se mettait en état "en cote".

2) Nous suggérons d'utiliser les instructions d'interpolation linéaire au lieu des instructions de mouvement point-point, quand le nombre des blocs de déplacement dépasse 32 et les blocs sont constitués par micro-segments. Pour de plus amples informations, il y a lieu de se référer au document "Limiti Firmware Movimento Punto Punto.doc" disponible chez T.P.A.

Exemple 1

Routine de Zérotage sur Interrupt

Exemple 2

```
; changement de vitesse
Function cambiovel
    setvel     X, 20
    setvel     X, 20
    movabs     X, 100, Y, 200
    movabs     X, 150, Y, 180
    setvel     X, 5
    movabs     X, 80, Y, 100
    waitstill  X, Y
fret
```

MOVINC

Syntaxe

MOVINC **axe1, valeur1 [, axe2, valeur2 [, ..., axe6, valeur6]]**

Arguments

axe1...[...axe6] noms de dispositifs type axe
valeur1...[...valeur6] constante ou variable. Cote de déplacement incrémentiel

Description

Cette instruction fait exécuter à tous les axes un *mouvement incrémentiel* égal à la **valeur** correspondante.
Pour exécuter le mouvement, l'axe ne doit pas être engagé dans un mouvement interpolé et il doit se trouver en cote ou en fenêtre. Le mouvement de l'axe commence dès que l'instruction est exécutée. Si plusieurs instructions de déplacement point-point sont exécutées dans une même tâche, elles sont enchaînées. Si une deuxième tâche tente d'exécuter des instructions point-point sur un axe qui est engagé dans un autre mouvement, cette tâche reste à attendre que le mouvement commandé par la première tâche soit achevé
Il est également possible de modifier la vitesse entre une mouvement point-point et le suivant avec une instruction SETVEL. Les deux mouvements seront raccordés par une rampe de vitesse sans arrêter les axes.
S'il n'est pas utilisé l'instruction S'il n'est pas utilisé l'instruction SETVEL, la vitesse maximale possible est représentée par la valeur de vitesse manuelle définie en configuration, la vitesse maximale possible est représentée par la valeur de vitesse manuelle définie en configuration
Un mouvement point-point peut être interrompu avec l'instruction STOP et être repris par la suite avec l'instruction START. Pendant l'interruption du mouvement, l'axe reste en état "à régime" même s'il ne bouge pas matériellement.
Un mouvement peut être avorté avec l'instruction ENDMOV. Dans ce cas, il ne peut plus être repris.

REMARQUE:

1) Auparavant, les mouvements point-point :
- ne permettaient pas de modifier la vitesse si l'axe n'était pas arrêté. Le comportement actuel est semblable à celui des mouvements interpolés.
- lorsqu'ils étaient interrompus par une instruction STOP, l'axe correspondant se mettait en état "en cote".

2) Nous suggérons d'utiliser les instructions d'interpolation linéaire au lieu des instructions de mouvement point-point, quand le nombre des blocs de déplacement dépasse 32 et les blocs sont constitués par micro-segments. Pour de plus amples informations, il y a lieu de se référer au document "Limiti Firmware Movimento Punto Punto.doc" disponible chez TPA.

Exemple 1

Routine de Zérotage d'un axe

Exemple 2

```
; changement de vitesse
Function cambiovel
    setvel     X, 20
    setvel     X, 20
    movinc     X, 100, Y, 200
    movinc     X, 150, Y, 180
    setvel     X, 5
    movinc     X, 80, Y, 100
    waitstill  X, Y
fret
```

MULTIABS

Syntaxe

MULTIABS **[étiquette],axe1, valeur1, [axe2, valeur2 [, axe3, valeur3 [, ..., axe16, valeur 16]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement
axe1…axe16	noms de dispositifs type axe
valeur1…[…valeur16]	constante ou variable. Valeur de la cote théorique de fin bloc de déplacement

Description

Interpolation multilinéaire absolue jusqu'à 16 axes. Ce mouvement d'interpolation permet d'anticiper des profils de vitesse, en réglant sur les axes, au moyen de l'instruction SETTOLERANCE, les tolérances respectives (par tolérance d'un axe, entendons la portion de parcours où un constant rapport d'interpolation peut n'exister pas). L'ordre d'insertion des axes dans l'instruction MULTIABS **doit** toujours être le même et **tous** les axes concernés par le mouvement doivent être présents. Les blocs de déplacement sont mis à la suite dans le lookahead normal, et le mouvement part à l'exécution d'une instruction WAITSTILL, STARTINTERP ou au remplissage de ce même lookahead. Parmi les axes concernés par le mouvement, un peut être utilisé comme collider au moyen de l'instruction WAITCOLL. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l' axe.

Note

Avec ce type d'interpolation, il n'est pas possible d'utiliser des systèmes de référence virtuels (instruction SETRIFLOC et RESRIFLOC). Il est possible d'utiliser des mouvements avec axes enchaînées (en CHAIN). Les axes impliqués dans le mouvement interpolé multiaxe doivent être déclarés master d'autres axes non impliqués dans le mouvement. De plus, il est possible d'appliquer le FeedRate Override.

Exemple

```
setquote        x, 0
setquote        y, 0
setquote        z, 0
; premier bloc
setveli         x, velx1
setveli         y, vely1
setveli         z, velz1
multiabs        x, quotax1, y,quotay1, z,quotaz1
; deuxième bloc
settolerance    x,tollx2,  y,tolly2,  z,tollz2
setveli         x, velx2
setveli         y, vely2
setveli         z, velz2
multiabs        x,quotax2,  y,quotay2,  z,quotaz2
;troisième bloc
settolerance    x,tollx3,  y,tolly3,  z,tollz3
setveli         x, velx3
setveli         y, vely3
setveli         z, velz3
multiabs        x,quotax3, y,quotay3, z,quotaz3
;quatrième bloc
settolerance    x,tollx4,  y,tolly4,  z,tollz4
setveli         x, velx4
setveli         y, vely4
setveli         z, velz4
multiabs        x,quotax4, y,quotay4, z,quotaz4
waitstill       x,  y,  z
```

MULTIINC

Syntaxe

MULTIINC	**[étiquette],axe1, valeur1, [axe2, valeur2 [, axe3, valeur3 [, …, axe16, valeur 16]]]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement

axe1...axe16	noms de dispositifs type axe
valeur1...[...valeur16]	constante ou variable. Valeur d'augmentation de la cote théorique de fin bloc de déplacement

Description

Interpolation multilinéaire incrémentielle jusqu'à 16 axes. Ce mouvement d'interpolation permet d'anticiper des profils de vitesse, en réglant opportunément sur les axes, au moyen de l'instruction SETTOLERANCE, les tolérances respectives (par tolérance d'un axe, entendons la portion de parcours où un constant rapport d'interpolation peut n'exister pas). L'ordre d'insertion des axes dans l'instruction MULTIINC **doit** toujours être le même et **tous** les axes concernés par le mouvement doivent être présents. Les blocs de déplacement sont mis à la suite dans le lookahead normal, et le mouvement part à l'exécution d'une instruction WAITSTILL, STARTINTERP ou au remplissage de ce même lookahead. Parmi les axes concernés par le mouvement, un peut être utilisé comme collider au moyen de l'instruction WAITCOLL. Le paramètre optional **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'une façon univoque le bloc de déplacement.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX. Dans ce cas il ne faut pas oublier que le terme interpolation désigne le mouvement coordonné de plusieurs d'axes atteint d'une erreur discrète, due à la méthode de pilotage de l'axe.

Note

Avec ce type d'interpolation, il n'est pas possible d'utiliser des systèmes de référence virtuels (instruction SETRIFLOC et RESRIFLOC). Il est possible d'utiliser des mouvements avec axes enchaînées (en CHAIN). Les axes impliqués dans le mouvement interpolé multiaxe doivent être déclarés master d'autres axes non impliqués dans le mouvement. De plus, il est possible d'appliquer le FeedRate Override.

NORMAL

Syntaxe

NORMAL	**axe**

Arguments

axe	nom de dispositif type axe

Description

Cette instruction active le contrôle de position sur l'**axe** et elle désactive l'enchaînement axes.
Lors de l'allumage du système, tous les axes configurés sont mis dans l'état free et ils changent à l'état normal cuand il est exécuté cette instruction ou cuand il est effectué le premier déplassement.
Il est toutefois conseillé d'exécuter cette instruction avant d'effectuer la procédure de zérotage des axes, pour acquitter les éventuelles conditions d'urgence.

RESRIFLOC

Syntaxe

RESRIFLOC	**axe1, axe2, axe3**

Arguments

axe1...axe3	noms de dispositifs type axe

Description

Cette instruction rétablit le système de référence absolue pour les axes X Y Z (**axe1**, **axe2**, **axe3**).
Normalement, cette instruction est utilisée après la programmation d'un système de référence rototranslaté avec une instruction SETRIFLOC.

SETINDEXINTERP

Syntaxe

SETINDEXINTERP	**axe, nomvar**

Arguments

axe	nom de dispositif type axe
nomvar	nom variable globale de type integer

Description

Cette instruction définit un indice qui compte le nombre de blocs d'interpolation exécutés par un axe. Pendant un mouvement en interpolation, à chaque changement bloc, la valeur de la variable **nomvar** augmente de 1.

Remarque

La variable utilisée comme indice doit être une variable globale de groupe ou globale de machine.

SETLABELINTERP

Syntaxe

SETLABELINTERP **axe, valeur**

Arguments

Axe	nom de dispositif type axe
valeur	variable globale de type integer

Description

Dans la variable **valeur** dPendant un mouvement en interpolation, à chaque changement bloc, la valeur de l' étiquette du nouveau bloc est assignée. L' étiquette est définie dans les instructions de mouvement interpolé.

Note

La variable **valeur** doit être une variable globale de groupe ou globale de module.

SETPFLY

Syntaxe

SETPFLY **axe, état, vitesse, cote,[erreur]**

Arguments

axe	nom de dispositif type axe
état	constante prédéfinie. Elle indique l'état du micro-interrupteur à contrôler. Les valeurs de réglage sont les suivantes : **ON** **OFF**
vitesse	constante ou variable float.
cote	constante ou variable.
erreur	variable integer. Code d'erreur

Description

Cette instruction permet de remettre instantanément à zéro la cote axe "au vol". Le zérotage est piloté par un switch raccordé à l'entrée rapide du connecteur de l'axe (sur des cartes avec bus MECHATROLINK-II on fait référence a EXTI1).
Pendant le mouvement de l'**axe**, exécuté par moyen d'une instruction MOVABS, l'instruction attend que le micro de zérotage relatif commute dans l'**état** indiqué. Cette transition étant interceptée, la cote réelle de l'axe est remise à zéro sans arrêter le mouvement et la **cote** cible d'arrivée et la **vitesse** sont automatiquement redéfinies de façon dynamique. Donc le mouvement correct est reconstruit pour obtenir la position d' arrivée. S'il est nécessaire, on invertit son mouvement. Si la cote programmée est atteinte sans que la commutation de l'entrée ne soit détectée et que le paramètre **erreur** n'a pas été réglé, une erreur de système est générée. Si un paramètre **erreur** a été réglé, il contiendra le code numérique de l'erreur de système correspondante.
Dans ce cas, le zérotage n'a pas été exécuté et il est nécessaire exécuter l'instruction SETQUOTE pour programmer de nouveau la recherche du micro.
Pour interrompre l'exécution du zérotage éclair, il suffit d'exécuter une instruction NORMAL sur l'axe ou d'achever la tâche qui a réclamé l'exécution du zérotage.

Pendant l'exécution de l'instruction il n'est pas possible d'exécuter dans le même temps des instructions SETPZERO et FASTREAD sur le même axe, s'il est connecté à des cartes avec bus MECHATROLINK-II.

Exemple

Routine de Zérotage sur Interrupt

SETPFLYCHAINSTRAT

Syntaxe

SETPFLYCHAINSTRAT **axe, type**

Arguments

axe	nom de dispositif type axe
type	constante type integer. Les valeurs admises sont les suivantes : 0 = seulement l'axe master met à zéro la cote, l'axe slave garde la cote précédent Autre que 0 = tous deux l'axe master et l'axe slave mettent à zéro la cote de façon synchrone

Description

Cette instruction permet d'organiser la façon avec laquelle l'axe slave indiqué se comporte après une instruction SETPFLY sur l'axe master. L'instruction doit être exécuter sur l'axe slave. Si la variable **type** est omise, la valeur prise par défaut est équivalente à 0.

SETPZERO

Syntaxe

SETPZERO **axe, cote [,erreur]**

Arguments

axe	nom de dispositif type axe
cote	constante o variable. C'est une cote incrémentielle
erreur	variable integer. Code d'erreur

Description

Cette instruction fait partir un mouvement incrémentiel de l'axe à la **cote** indiquée et attend que le cran de zéro du codeur soit piloté (avant que la cote indiquée ne soit atteinte).
Au moment où le cran est détecté, la cote réelle est mise à zéro et l'axe est arrêté.
Si la cote programmée est atteinte sans avoir intercepté le cran de zéro et que le paramètre **erreur** n'a pas été réglé, une erreur de système est générée. Si un paramètre **erreur** a été réglé, il contiendra le code numérique de l'erreur de système correspondante. Dans ce cas, le zérotage n'a pas été exécuté et il est nécessaire d'exécuter l'instruction SETQUOTE pour programmer de nouveau la recherche du cran.
Le mouvement des axes, généré par cette instruction, peut être interrompu par une instruction STOP et réactivé par une START.
Si on exécute l'instruction avec des axes S-CAN et avec des axes EtherCAT il est nécessaire d'exécuter avant tout une instruction FREE.

Pendant l'exécution de l'instruction il n'est pas possible d'exécuter dans le même temps des instructions SETPZERO et FASTREAD sur le même axe, s'il est connecté à des cartes avec bus MECHATROLINK-II.

Exemple

```
FREE     X
SETPZEROX, 100
```

SETPZEROCHAINSTRAT

Syntaxe

SETPZEROCHAINSTRAT **axe, [valeur]**

Arguments

axe	nom de dispositif type axe
valeur	type integer. Les valeurs admises sont: 0 = seul l'axe master met la cote à zéro, l'axe slave garde la cote précédente Autre que 0 = l'axe master et l'axe slave mettent la cote à zéro en mode synchrone

Description

Cette instruction permet d'introduire la façon dont l'axe slave indiqué se comportera face à une instruction SETPZERO sur le master.
L'instruction doit être réalisée sur l'axe slave.
Si la variable **valeur** est omise, la valeur prise par défaut est équivalente à 0.

SETQUOTE

Syntaxe

SETQUOTE **axe, cote**

Arguments

axe	nom de dispositif type axe
cote	constante ou variable

Description

Cette instruction force simultanément la cote théorique et la cote réelle d'un axe à la valeur indiquée par **cote**. Si l'axe est en cours de déplacement, cela provoque un arrêt brusque de l'axe, étant donné que, pour le contrôle, l'axe se trouve soudain en cote (la cote réelle coïncide avec la cote cible). Il est donc déconseillé d'utiliser cette instruction sur un axe en mouvement, à moins qu'il ne soit à une vitesse très basse.

Exemple

Routine de Zérotage d'un axe

SETQUOTECHAINSTRAT

Syntaxe

SETQUOTECHAINSTRAT **axe, [valeur]**

Arguments

axe	nom de dispositif type axe
valeur	variable type integer. Les valeurs admises sont: 0 = seul l'axe master initialise sur la nouvelle cote, l'axe slave garde la cote précédente Autre que 0 = les cotes de l'axe slave sont initialisées en mode synchrone avec les cotes de l'axe master

Description

Cette instruction permet d'introduire la façon dont l'axe slave indiqué se comportera face à une instruction SETQUOTE sur le master.
L'instruction doit être réalisée sur l'axe slave.
Si la variable **valeur** est omise, la valeur prise par défaut est équivalente à 0.

SETRIFLOC

Syntaxe

SETRIFLOC **cote1_ax1, cote2_ax1, cote3_ax1,**
cote1_ax2, cote2_ax2, cote3_ax2,
cote1_ax3, cote2_ax3, cote3_ax3,
axe1, axe2, axe3

Arguments

cote1_ax1...cote3_ax3	cosinus directeurs des trois axes
axe1...axe3	noms de dispositifs type axe

Description

Cette instruction permet d'activer un système de référence cartésien X' Y' Z' rototranslaté par rapport au système de référence absolue de machine X Y Z, représenté par le axes matériels **axe1**, **axe2** et **axe3**.
Les neuf arguments indiquent les Cosinus Directeurs des trois axes locaux par rapport aux absolus,

$\cos\alpha_1$	$\cos\beta_1$	$\cos\gamma_1$
$\cos\alpha_2$	$\cos\beta_2$	$\cos\gamma_2$

$\cos\alpha_3$ $\cos\beta_3$ $\cos\gamma_3$
qui constituent la matrice de transformation des coordonnées.
L'origine du nouveau système de référence est prise sur le point courant.
Toutes les instructions de mouvement interpolé, concernant les axes X, Y et Z, se réfèrent à ce système de référence, jusqu'à l'exécution de l'instruction RESRIFLOC.

SETTOLERANCE

Syntaxe

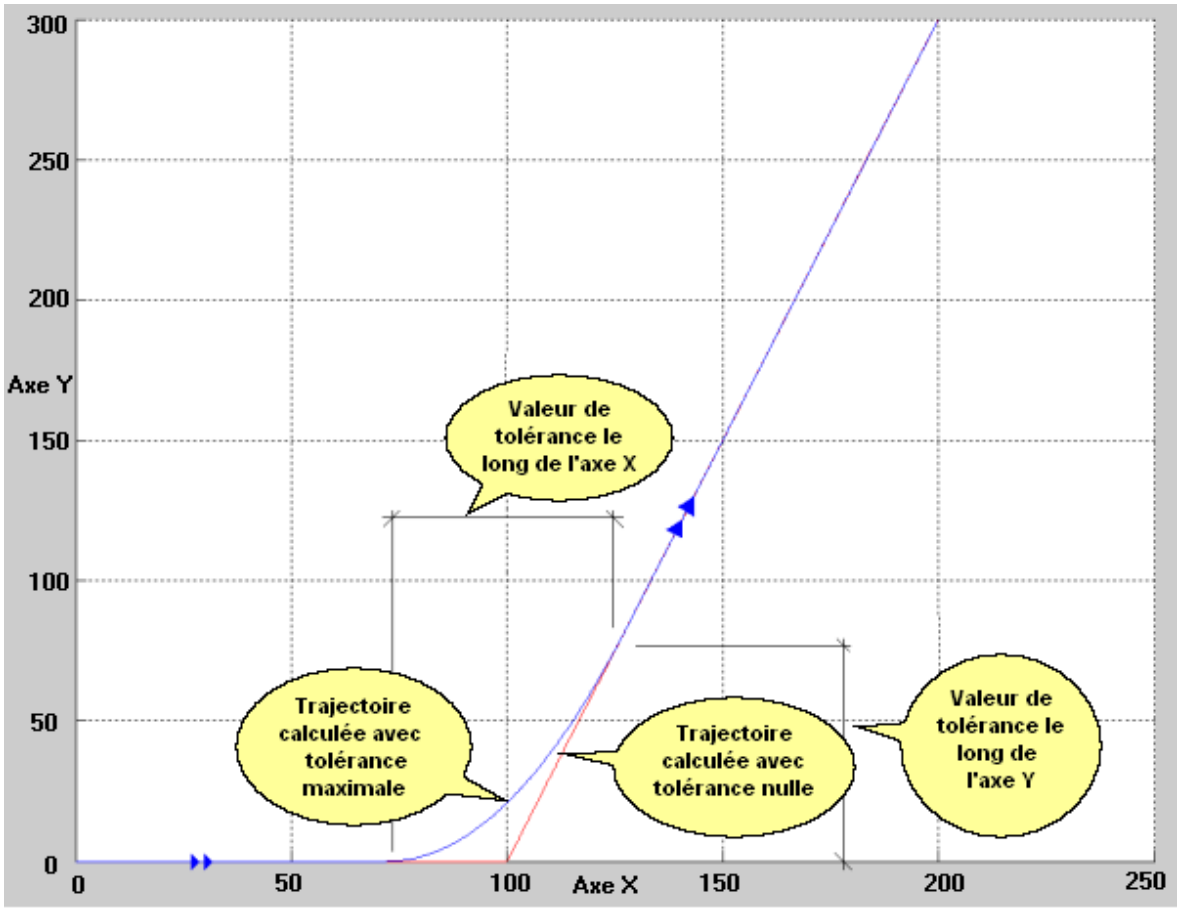
SETTOLERANCE **axe1, valeur1, [axe2, valeur2 [, axe3, valeur3 [, …, axe16, valeur 16]]]**

Arguments

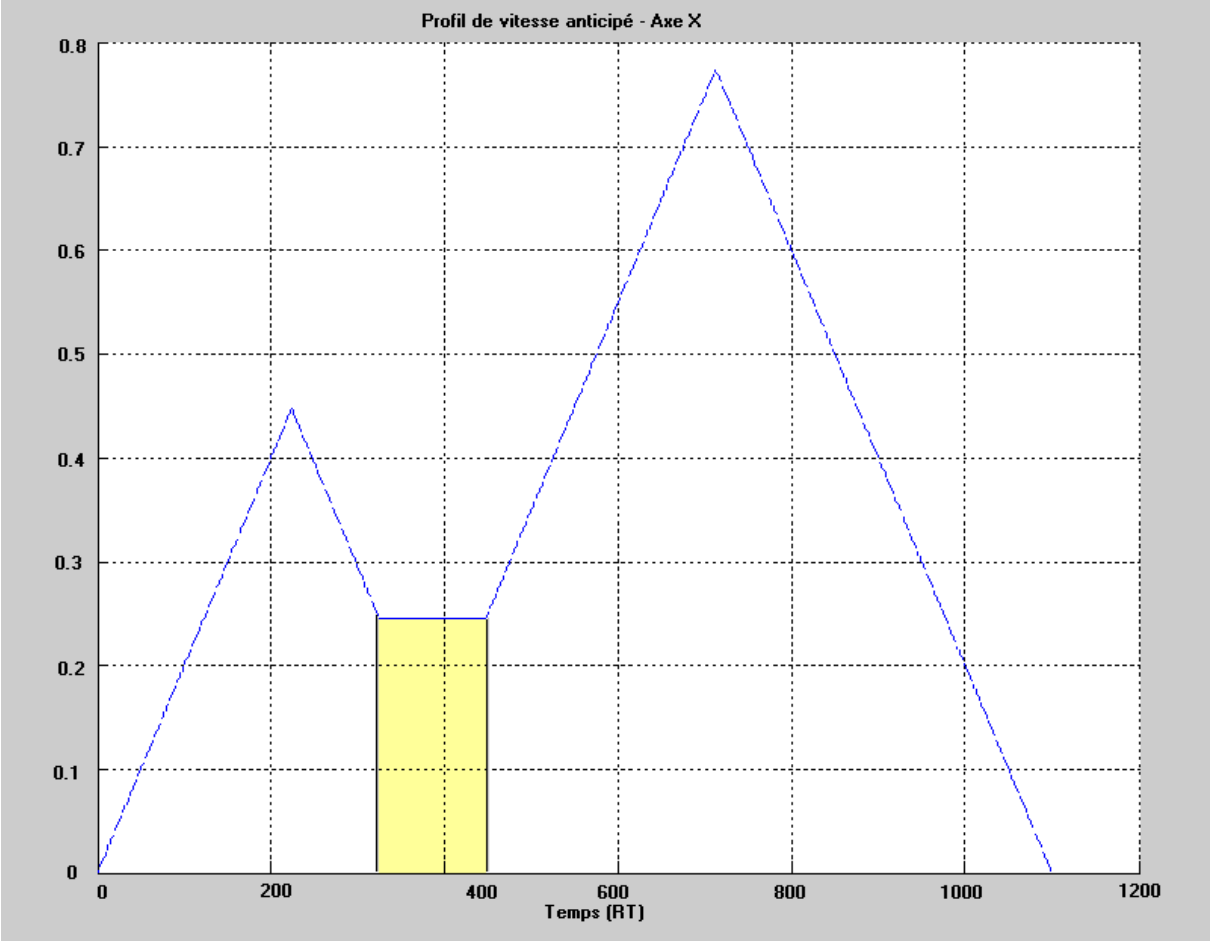
axe1…axe16	noms de dispositifs type axe
valeur1…[…valeur16]	constante ou variable. Valeur de la tolérance maximum applicable à l'axe

Description

Elle règle pour chaque **axe** défini la **valeur** de tolérance à appliquer dans les mouvements d'interpolation multiaxe. La valeur de tolérance est la valeur de déplacement selon laquelle l'axe se détache de la trajectoire originale dans une interpolation multiaxe.
Il faut régler la tolérance pour chaque **axe** concerné par l'interpolation, et le système fournira en avance des profils de vitesse en respectant les tolérances sur tous les axes et en ne dépassant pas l'espace de rampe, qui représente la limite maximale pour l'anticipation des profils. Si la tolérance n'est pas assignée avant une instruction multiaxe, celle qui s'appliquera sera la dernière tolérance réglée sur ce même axe. Dans le cas où une valeur de tolérance n'a jamais été assigné à un axe déterminé, celui-ci est considéré à tolérance nulle. Dans ce cas, tout mouvement multiaxe qui impliquera cet axe, ne prévoira aucune anticipation de rampe. Il est donc conseillé d'utiliser systématiquement cette instruction pour tout axe concerné, avant une instruction d'un mouvement d'interpolation multiaxe.



La figure ci-dessus représente une trajectoire multiaxe classique, composée de deux blocs de mouvement, dont le premier consiste dans un déplacement de 100 de l'axe X, tandis que le deuxième consiste dans un mouvement de l'axe Y de 300 et toujours un de 100 de l'axe X. La ligne rouge met en évidence la trajectoire en cas de tolérance nulle, tandis que la bleue est la trajectoire en cas de tolérance maximale.
La tolérance peut être vue également comme l'aire tendue par le profil de vitesse pendant l'intervalle d'anticipation, comme le montre la figure suivante



START

Syntaxe

START **axe**

Arguments

axe nom de dispositif type axe

Description

Reprise du mouvement de l'**axe** après le stop.

STARTINTERP

Syntaxe

STARTINTERP **axe**

Arguments

axe nom de dispositif type axe

Description

Cette instruction lance une interpolation dont le canal est identifié par **axe.** Normalement, le mouvement des axes associés à un canal d'interpolation commence lorsque le buffer de l'interpolateur (512 instructions) est complètement rempli ou quand une instruction WAITSTILL, qui achève donc le mouvement, est atteinte. Cela permet à l'algorithme de l'interpolateur de déterminer des profils de vitesse optimaux, dans la mesure où il dispose les informations relatives à un grand nombre (ou tous) de traits du mouvement en interpolation. L'instruction STARTINTERP permet de forcer le lancement du mouvement des axes même si les conditions que nous venons de décrire ne sont pas réunies.

STOP

Syntaxe

STOP **axe**

Arguments

axe nom de dispositif type axe

Description

Arrêt du mouvement de l'**axe**. L'axe exécute une rampe de décélération dont la durée dépend de la vitesse actuelle et des paramètres de configuration.

Exemple

Routine de Zérotage d'un axe

SWITCHENC

Syntaxe

SWITCHENC **axe1, [axe2, [direction, cote]]**

Arguments

axe1 Nom du dispositif type axe
axe2 noms de dispositifs type axe. Montre une axe de comptage.
direction constante prédéfinie.
UP = échange codeur en dépassant la cote en direction positive
DOWN = échange codeur en dépassant la cote en direction positive
cote constante double ou variable double

Description

Permet de remplacer le codeur de **l'axe1** le codeur de **l'axe2**. Le codeur est échangé en dépassant en **direction** positive (UP) ou négative (DOWN) la **cote** indiquée.
Si les paramètres **direction** et **cote** sont omis, le codeur est échangé tout de suite, en dehors de la position des axes.

Si seulement **axe1** est déclaré**,** le fonctionnement avec codeur individuel est rétabli.

Axe1 ne peut pas être de type pas-pas, comptage et virtuel, **Axe2** ne peut être qu'un axe de comptage. En outre, **l'axe1** comme **l'axe2** ne peuvent pas être engagés en mouvement en chaîne comme axe slave.

L'instruction génère l'erreur de système "4101 – Gestion non congruente de l'axe NomAxe", quand **axe1** ou **axe2** est déclaré slave dans un mouvement en chaîne ou ou lorsque **axe1** exécute une instruction FASTREAD ou une instruction SETPFLY. En outre, l'erreur de système "4105 – Instruction non exécutable sur l'axe NomAxe" peut être générée, lorsque le type d'axe déclaré n'es pas inclus dans ceux possibles.

WAITACC

Syntaxe

WAITACC **axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6] nom de dispositif type axe

Description

Attente de l'état d'accélération ou un des états suivants sur tous les **axes** (1÷6) indiqués.
La tâche dans laquelle est exécutée l'instruction est mise en attente jusqu'à l'axe se trouve dans l'état d' accélération ou dans un des états suivants.

Les états de l'axe sont identifié par un nombre entier:
- accélération = 1
- régime = 2

- décélération = 3
- cote = 4
- attente fenêtre grande = 5
- attente axe arrêté = 6
- attente fenêtre petite = 7

WAITCOLL

Sintaxe

WAITCOLL **axe, valeur, timeout, delta**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Valeur de coordonnée absolue
timeout	constante ou variable. Il est le temps d'attente avec axe arrêté.
delta	constante ou variable. Il est la valeur de fenêtre pour obtenir axe arrêté

Description

Pendant le mouvement d'un axe l'atteinte d'une coordonnée programmée peut être empêchée par un obstacle de nature mécanique, représenté quelque fois aussi par la même pièce en usinage. Le système dans ce cas engendre une erreur de système de "servoerror" ou "mouvement pas finit".
Avec cette instruction se définit une **valeur** de coordonnée en correspondance de laquelle le système commence à vérifier la présence d'une collision, le temps de attente (**timeout**) avant que l'**axe**, après la passée collision, soit mis dans l'été "coordonnée" et le **delta** qui définie la tolérance sur le positionnement de l'axe. Quand l'axe dépasse la coordonnée définie en **valeur** est vérifié si l'axe est encore en mouvement. Intercepté l'obstacle est individué la situation critique et, aussi en garantissant
La poussée du moteur, n'est plus vérifié le dépassement du limite de l'erreur de boucle. La direction de mouvement sur lequel est vérifiée la passée collision est la même direction du dernier mouvement accoué. Le timeout est exprimé en secondes, la valeur **delta** doit être majeur de 0.001 mm et mineur de la différence entre la coordonnée de arrivée programmée et la coordonnée **valeur**.
L'instruction peut être utilisée avec l'interpolateur multiaxe du moment que dans cet interpolateur on accepte la perte temporaire du lien d'interpolation.
L'instruction est applicable aussi aux axes Master d'un mouvement en Chain.
Est engendré une erreur de système quand**:**

- l'axe est en train d'exécuter un mouvement interpolé classique (voir les instructions LINNEARBS, LINEARINC, CIRCABS, CIRCINC, HELICABS, HELICINC) ou en mouvement coordonné ou mouvements interpolés ISO.
- l'axe **est** un axe slave
- l'axe **est** un axe de comptage ou un axe paso à pas ou un axe virtuel
- la **valeur** programmée est supérieure à la coordonnée du fin mouvement

Exemple

```
; programme la coordonnée de l'axe X
SETQUOTE                X, 0.0
; déplace l'axe X à la coordonnée absolue 1000
MOVABS                  X, 1000.0
; attende la coordonnée de collision , attende 2 secondes
; avant de mettre l'axe dans l'été "coordonnée"  après
; avoir intercepté une collision avec une précision de 0.01 mm
WAITCOLL                X, 980.0,2.0,0.01
```

WAITDEC

Syntaxe

WAITDEC **axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6]	nom de dispositif type axe

Description

Attente de l'état de décélération ou un des états suivants sur tous les **axes** (1÷6) indiqués.
La tâche dans laquelle est exécutée l'instruction est mise en attente jusqu'à l'axe se trouve dans les états de décélération, cote, attente fenêtre. grande, attente fenêtre petite et attente axe arrêté.

Les états de l'axe sont identifié par un nombre entier :
- accélération = 1
- régime = 2
- décélération = 3
- cote = 4
- attente fen. grande = 5
- attente axe arrêté = 6
- attente fen. petite = 7

WAITREG

Syntaxe

WAITREG **axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6] nom de dispositif type axe

Description

Attente de l'état de régime ou un des états suivants sur tous les **axes** (1÷6) indiqués.
La tâche dans laquelle est exécutée l'instruction est mise en attente jusqu'à l'axe se trouve dans les états de régime, décélération, cote, attente fenêtre grande, attente fenêtre petite et attente axe arrêté.

Les états de l'axe sont identifié par un nombre entier :
- accélération = 1
- régime = 2
- décélération = 3
- cote = 4
- attente fenêtre grande = 5
- attente axe arrêté = 6
- attente fenêtre petite = 7

WAITSTILL

Syntaxe

WAITSTILL **axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6] nom de dispositif type axe

Description

Pour tous les **axes** (1÷6) indiqués, attente de la fin du mouvement (état Cote).

Exemple

Routine de Zérotage d'un axe

WAITTARGET

Syntaxe

WAITTARGET **axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6] nom de dispositif type axe

Description

Pour tous les **axes** (1÷6) indiqués, attente que la cote théorique courante soit égale à la cote cible.
La cote réelle différera de la cote théorique jusqu'à ce que l'erreur de boucle ne soit acquittée.

WAITWIN

Syntaxe

WAITWIN	**axe1 [, ..., axe6]**

Arguments

axe1 [...,axe6]	nom de dispositif type axe

Description

Attente de l'état fenêtre ou un des états suivants sur tous les **axes** (1÷6) indiqués.
La tâche dans laquelle est exécutée l'instruction est mise en attente jusqu'à l'axe se trouve dans les états de attente fen. grande, attente fen. petite et attente axe arrêté.

Les états de l'axe sont identifié par un nombre entier :
- accélération = 1
- régime = 2
- décélération = 3
- cote = 4
- attente fen. grande = 5
- attente axe arrêté = 6
- attente fen. petite = 7

Paramètres Axe

Lecture/Écriture

DEVICEID

Syntaxe

DEVICEID	**dispositif, variable**

Arguments

dispositif	nom du dispositif ou paramètre dispositif
variable	variable integer qui reçoit l'adresse logique

Description

Il écrit en **variable** l'adresse logique associé à **dispositif** de type quelconque.
Cette instruction donne la possibilité d'avoir une "clé" univoque associée au dispositif, comme indice ou clé de recherche en structure de données.

GETAXIS

Syntaxe

GETAXIS	**axe, nomdonnée, nomvar**
GETAXIS	**axe, nomdonnée1, nomdonnée2, [...,nomdonnée20,] matrice[ligne]**

Arguments

axe	nom du dispositif type axe
nomdonnée	constante prédéfinie (voir la liste ci-après). Paramètre axe (1÷20)
nomvar	variable ou nom de dispositif
ligne	constante ou variable integer. Numéro de ligne de la matrice
matrice	nom de la matrice

Description

Dans la première forme, l'instruction lit une des données (**nomdonnée**) d'un axe et l'enregistre dans une variable.
Dans la deuxième forme, cette instruction lit, au même instant, plusieurs données d'un axe (de 1 à 20) et les enregistre, dans l'ordre où ils ont été demandés dans les éléments de la ligne de matrice indiquée.
Dans ce cas, le nombre de colonnes de la matrice doit être égal au nombre de données demandées.
La liste présentée ci-après comprend toutes les constantes prédéfinies qui peuvent être attribuées au paramètre **nomdonnée.**
La première colonne est le nom de la constante.
La deuxième colonne est la description de la grandeur de l'axe qui est lue par l'instruction.

La troisième est le format de la donnée qui est rendu dans la variable **nomvar** ou dans **matrice[ligne],** où :

- **d** signifie **double**,
- **f** signifie **float,**
- **i** signifie **integer,**
- **b** signifie **char**.

Si la déclaration de la variable, dans laquelle la donnée sera enregistrée, est différente de la valeur rendue par l'instruction, le compilateur exécute une transformation (cast) de la donnée dans le type demandé par l'utilisateur. Cela implique parfois une perte importante de données. Par exemple, une valeur double égale à 12.345 transformée en un nombre entier devient 12. Il est donc conseillé de respecter le type demandé lors de la déclaration des variables **nomvar** et **matrice[ligne]**.
La dernière colonne décrit la valeur de retour ou l'unité de mesure du paramètre correspondant.
Les constantes qui commencent par "_CFG" permettent de lire les valeurs de configuration, c'est-à-dire celles qui sont réglées à l'allumage de la machine.

constante	description	type	valeur de retour
_CFGTYPE	type axe	i	1=analogique,3=pas à pas,4=digital,5=comptage,6=non utilisé,7=virtuel
_CFGUM	unité de mesure	i	0=millimètres,1=pouces, 2=degrés, 3=tours
_CFGRIS	résolution	d	impulsions par_UM
_CFGVMAX	vitesse maximale	f	m/1' ou pou/1" ou degrés /1" ou tours/1'
_CFGVMAXD	vitesse maximale en manuel	f	m/1' ou pou/1" ou degrés/1" ou tours/1'
_CFGVMAXI	vitesse maximale d'interpolation	f	m/1' ou pou/1" ou degrés/1" ou tours/1'
_CFGPHINV	inversion phases codeur	b	0=aucune inversion, 1=inversion
_CFGRFINV	inversion référence	b	0=aucune inversion, 1=inversion
_CFGZIND	activation réinitialisation cote cran de zéro	b	0=désactivé, 1=activé
_CFGSLOPE	type de rampe pour accélérations/décélérations dans le point-point	b	0=linéaire, 1=en 'S' , 2= en double S
_CFGKFFA	feed forward d'accélération	f	
_CFGKFFAI	feed forward d'interpolation d'accélération	f	
_CFGSRPP	vitesse début rampe pas à pas	f	m/1' ou pou/1" ou degrés/1" ou tours/1'
_CFGACC	temps d'accélération de 0 à _CFGVMAX	i	msec
_CFGDEC	temps de décélération de _CFGVMAX à 0	i	msec
_CFGACCI	temps d'accélération de _CFGVMAXI à _VMAXI	i	msec
_CFGDECI	temps de décélération de _CFGVMAXI à 0	i	msec
_CFGQLP	limite axe en positif	d	cote
_CFGQLN	limite axe en négatif	d	cote
_CFGKP	coefficient proportionnel	f	
_CFGKI	coefficient intégratif	f	
_CFGKD	coefficient dérivatif	f	
_CFGKFF	feed forward	f	pourcentage
_CFGKPS	coefficient proportionnel axe slave	f	
_CFGKIS	coefficient intégratif axe slave	f	

constante	description	type	valeur de retour
_CFGKDS	coefficient dérivatif axe slave	f	
_CFGQEAP	erreur de boucle en positif	d	cote
_CFGQEAN	erreur de boucle en négatif	d	cote
_CFGKPI	coefficient proportionnel d'interpolation	f	
_CFGKII	coefficient intégratif d'interpolation	f	
_CFGKDI	coefficient dérivatif d'interpolation	f	
_CFGTMINP	tension minimum positive	f	volt
_CFGTMINN	tension minimum négative	f	volt
_CFGSTMINP	tension de seuil positive	f	volt
_CFGSTMINN	tension de seuil négative	f	volt
_CFGESC	timeout déplacement Axe	i	msec
_CFGDSE	habilitation servoerror dynamique	b	0=désactivé, 1=activé
_CFGAEN	habilitation adjust automatique	b	0=désactivé, 1=activé
_CFGOFFSET	tension de adjust - offset initial	f	volt
_CFGCEE	coordonnée de enchaînement encoder erroné	d	cote
_CFGNOTCH	fréquence filtre de Notch	i	Hz
_CFGBUFI	dimension buffer de calcule integratif	i	[1, 200]
_CFGQAP	seuil arrête en cote positive	d	
_CFGQAN	seuil arrête en cote negative	d	
_CFGSLOPEI	type de rampe pour accélérations/décélérations dans l'interpolations	f	0=linéaire, 1=en 'S' , 2= en double S
_CFGKFFI	feed forward interpolation	f	pourcentage
_CFGAAF	attente axe arrêté	b	0=désactivé, 1 =activé
_CFGENCTYPE	type de codeur	i	0=simulé ou absent, 1=réel
_SRPP	vitesse début rampe pas à pas	f	m/1' ou pou/1" ou degrés/1" ou tours/1'
_ACC	temps d'accélération de 0 à _VMAX	i	msec
_DEC	temps de décélération de _VMAX à 0	i	msec
_ACCI	temps d'accélération de 0 à _VMAXI en interpolation	i	msec
_DECI	temps de décélération de _VMAXI à 0 en interpolation	i	msec
_QLP	limite axe en positif	d	cote
_QLN	limite axe en négatif	d	cote
_KP	coefficient proportionnel	f	
_KI	coefficient intégratif	f	
_KD	coefficient dérivatif	f	
_KFF	feed forward	f	pourcentage
_KPS	coefficient proportionnel axe slave	f	
_KIS	coefficient intégratif axe slave	f	
_KDS	coefficient dérivatif axe slave	f	
_QEAP	erreur de boucle en positif	d	cote
_QEAN	erreur de boucle en négatif	d	cote

constante	description	type	valeur de retour
_VEL	vitesse point-point	f	m/1' ou pou/1" ou degrés/1" ou tours/1'
_VELI	vitesse d'interpolation	f	m/1' o pou/1" ou degrés/1" ou tours/1'
_MODE	mode de fonctionnement axe	b	1=normal, 2=libre, 8=interpol., 10=coord.
_PHINV	inversion phases codeur	b	0=aucune inversion, 1=inversion
_RFINV	inversion référence	b	0=aucune inversion, 1=inversion
_ZIND	activation réinitialisation cote cran de zéro	b	0=désactivé, 1=activé
_KPI	coefficient proportionnel interpolation	f	
_KII	coefficient intégratif interpolation	f	
_KDI	coefficient dérivatif d'interpolation	f	
_KFFI	feed forward interpolation	f	pourcentage
_KFFA	feed forward d'accélération	f	pourcentage
_KFFAI	feed forward d'accélération d'interpolation	f	pourcentage
_ESC	timeout déplacement axe	i	msec
_CEE	coordonnée de enchaînement encoder erroné	d	cote
_NOTCH	fréquence filter de Notch	i	Hz
_BUFI	dimension buffer de calcule intégratif	i	[1,200]
_QAP	seuil arrête en cote positive	d	
_QAN	seuil arrête en cote négative	d	
_QEAPINV	limite erreur de boucle positive en inversion	d	
_QEANINV	limite erreur de boucle négative en inversion	d	
_OFSCOORD	coordonnée offset mouvement coordonné	d	
_MS	type axe master ou slave	b	0=non dans la chaîne, 4=maître, 5=esclave
_QENC	cote codeur	d	cote
_QR	cote réelle	d	cote
_RIS	résolution utilisée par l'axe	d	
_ST	état axe	b	1=accél.,2=régime,3=décél.,4=cote,5=attente fenêtre grande, 6=attente axe arrêté, 7=attente fenêtre petite, 8=départ
_QT	cote théorique	d	quota
_EA	erreur de boucle	d	quota
_FF	feed forward	i	
_VC	vitesse courante	f	
_P	correcteur proportionnel	i	
_I	correcteur intégratif	i	
_D	correcteur dérivatif	i	
_FLGS	flags axes	b	
_VCR	vitesse courante réelle	f	
_ADJUST	compensation offset axe	i	valeur entier qui représente la tension, à passer à l'actionnement, vue du côté de la fiche des axes du dac. Le plein-

constante	description	type	valeur de retour
			échelle pour l'actionnement est 10 volt et pour le dac est 32767
_DAC	DAC	i	valeur entier, qui représente la tension, à passer à l'actionnement, vue du coté de la fiche des axes du dac. Le plein-échelle pour l'actionnement est 10 volt et pour le dac est 32767.
_ACCINST	valeur d'accélération instantanée	f	
_FFA	feed forward d'accélération	i	
_GONETIME	temps passé du début du mouvement	f	sec (0 pour les axes esclaves et axes pas à pas)
_RESTIME	temps résidu à la fin du mouvement. Les valeurs sont relatives au mouvement alloué dans le buffer au moment de la demande	f	sec (0 pour les axes esclaves, axes en mouvement coordonné, axes pas-à-pas et mouvement ISO)
_TARGETTIME	temps nécessaire pour générer la cote cible	f	microsecondes
_GONESPACE	chemin spatial depuis le début du mouvement. Les valeurs sont relatives au mouvement alloué dans le buffer au moment de la demande	f	pourcentage (100 pour les axes esclaves, axes en mouvement coordonné et axes pas-à-pas)
_RESSPACE	chemin spatial depuis le début du mouvement. Les valeurs sont relatives au mouvement alloué dans le buffer au moment de la demande	f	pourcentage (0 pour les axes esclaves, axes en mouvement coordonné et axes pas-à-pas)
_AXESJERK	habilitation sur l'axe du contrôle du jerk	b	1=contrôle activé , 0=contrôle non activé
_MOVEJERK	habilitation du contrôle du jerk sur le mouvement sur lequel est engagé l'axe	b	1=contrôle activé, 0=contrôle non activé
_MOVETYPE	type de mouvement sur lequel l'axe est engagé	b	1=mouvement interpolé classique, 2= mouvement interpolé classique multiaxe , 3=mouvement coordonné, 4= mouvement point-point, 5=mouvement in chain (uniquement axes esclaves)
_PARTYPESET	type de paramètres axe utilisés pendant le mouvement	i	1=interpolation, 0=point-point
_AXINRIFLOC	Axes utilisés dans un système de référence local	i	1=oui, 0=non
_QTARGETTOOL	cote cible de l'axe. En cas d' interpolation ISO cote cible de la coordonnée du foret de l'outil de l'axe	d	
_QREALTOOL	cote réelle de l'axe. En cas d' interpolation ISO cote réelle de la coordonnée du foret de l'outil de l 'axe	d	
_BACKLASH	valeur de la joue mécanique définie pour l'axe	d	
_DISABLED	désactivation d'un axe	b	1=axe désactivé, 0=axe activé
_DYNLIMIT	Habilitation du contrôle dynamique des limites axe	b	1=contrôle activé, 0=contrôle non activé

constante	description	type	valeur de retour
_AXESFEED	Valeur de feedrate override appliqué couramment à l'axe	f	
_CORRLIN	type de correcteur de linéarité utilisé	i	0=pas de compteur en service, 1=correcteur automatique, 2=correcteur croisé, 3=correcteur automatique avec correcteur croisé
_VELISO	Vitesse du bec de l'outil pendant le mouvement ISO	f	
_ISOSTOPS	nombre d'arrêts forcés du mouvement interpolé en raison de situations limite dans la gestion du lookahead	i	
_CURRATIO	valeur du rapport d'enchaînement actuellement utilisée	d	
_DYNRATIO	retourne, si un changement dynamique de rapport enchaînement est en cours.	i	0=non, 1 = oui
_RESBLOCK	nombre de blocs de déplacement encore à exécuter	i	
_EXECBLOCK	nombre de blocs de déplacement exécutés	i	
_TOTALBLOCK	nombre total de blocs de déplacement mis en file d'attente dans le mouvement (valeur courante)	i	
_SWITCHENC	contrôle si le codeur est en train d'être échangé	i	-1=l'axe n'utilise pas l'instruction SWITCHENC, 0=une instruction SWITCHENC a été exécutée, mais l'axe est en train d'utiliser son codeur, 1= une instruction SWITCHENC a été exécutée et l'axe est en train d'utiliser le codeur de l'axe de comptage
_QOFSENC	valeur de l'offset codeur	d	
_LENSETPZERO	distance parcourue pour atteindre le cran de zéro	d	cote
_TORQUEINST	valeur instantané de torque	i	
_CURRSLOPE	renvoie le type de rampe actuellement utilisé dans les mouvements rapides	i	0=linéaire, 1=en S, 2=en double S
_CURRSLOPEI	renvoie le type de rampe actuellement utilisé dans les mouvements interpolés	i	0=linéaire, 1=en S, 2=en double S
_REALSLOPEI	renvoie le type de rampe actuellement utilisé dans les mouvements interpolés	i	0=linéaire, 1=en S, 2=en double S
_AXISPAR1...AXISPAR8	paramètres optionnels supplémentaires pour l'axe EtherCAT	i	numéro
_ISOMOVETYPE	Type de mouvement ISO en exécution	i	0 = mouvement ISO rapide, 1 = mouvement ISO interpolé, -1 = autre
_QMAINENC	Cote réelle du codeur principal principale lorsque le codeur secondaire est utilisé	d	

Mouvement point-point

SETACC

Syntaxe

SETACC **axe, [valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Temps de accélération

Description

Cette instruction attribue à l'axe le temps d'accélération identifié par **valeur**. Le temps d'accélération est exprimé en millisecondes.
Si l'argument **valeur** est omis, le paramètre de configuration est adopté. Si l'instruction est posée entre deux instructions MOVABS ou MOVINC, la première instruction de mouvement est exécutée (le mouvement s'arrête), en utilisant les paramètres d'accélération et décélération précédemment programmés. La seconde instruction est exécutée en appliquant les nouveaux paramètres d'accélération. L'instruction SETACC n'a effet que pour les mouvements aprés son exécution.
Si la **valeur** indiquée est plus petite que celle de la configuration, cette dernière est adoptée.

Voir également SETDEC, SETACCI et SETDECI.

SETDEC

Syntaxe

SETDEC **axe, [valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Temps de décélération

Description

Cette instruction attribue à l'axe le temps de décélération identifié par **valeur**. Le temps de décélération est exprimé en millisecondes. Si l'argument **valeur** est omis, le paramètre de configuration est adopté. Si l'instruction est posée entre deux instructions MOVABS ou MOVINC, la première instruction de mouvement est exécutée (le mouvement s'arrête), en utilisant les paramètres d'accélération et décélération précédemment programmés. La seconde instruction est exécutée en appliquant les nouveaux paramètres de décélération. L'instruction SETACC n'a effet que pour les mouvements aprés son exécution.
Si la **valeur** indiquée est plus petite que celle de la configuration, cette dernière est adoptée.

Voir également SETACC, SETACCI et SETDECI.

SETDERIV

Syntaxe

SETDERIV **axe [, valeur]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Coefficient d'action dérivée. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action dérivée*.
Si l'argument **valeur** est omis, le coefficient d'action dérivée de la configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETDERIVI.

SETFEED

Syntaxe

SETFEED **axe, valeur**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Représente le pourcentage di feed rate override

Description

Cette instruction modifie la **valeur** pourcentage de feed rate override de l'axe pour ce qui est des *mouvements point-point*. Voir également SETFEEDI.

SETFEEDF

Syntaxe

SETFEEDF **axe [, valeur]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Pourcentage de feed forward

Description

Cette instruction attribue à l'axe la **valeur** *pourcentage de feed forward.*
Si l'argument **valeur** est omis, le coefficient de feed forward de configuration est adopté.
Si l'instruction est appliquée à un moteur pas à pas, une erreur de système est générée. Il en va de même si on attribue à la variable **valeur** une valeur non comprise entre 0 et 100.
Voir également les instructions SETFEEDFI, SETFEEDFA, SETFEEDFAI.

SETFEEDFA

Syntaxe

SETFEEDFA **axe [, valeur]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Pourcentage de feed forward

Description

Cette instruction attribue à l'axe la **valeur** *pourcentage de feed forward* d'accélération pour les mouvements point-point.
Si l'argument **valeur** est omis, le coefficient de feed forward de configuration est adopté.
Si l'instruction est appliquée à un moteur pas à pas, une erreur de système est générée. Il en va de même si on attribue à la variable **valeur** une valeur non comprise entre 0 et 100.
Voir également les instructions SETFEEDF, SETFEEDFI, SETFEEDFAI.

SETINTEG

Syntaxe

SETINTEG **axe [, valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Coefficient d'action intégrale. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action intégrale*.
Si l'argument **valeur est omis**, le coefficient d'action intégrale de configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETINTEGI.

SETMULTIFEED

Syntaxe

SETMULTIFEED	**axe1, valeur1, axe2, valeur2 [, axe3, valeur3 [, ..., axe16, valeur16]]]**

Arguments

axe1...axe16	noms de dispositifs type axe
valeur1...[...valeur16]	constante ou variable. Elle représente le pourcentage de feed rate override

Description

Modifie la **valeur** de pourcentage de feed rate override des **axes** indiqués en ce que concerne les *mouvements point-point*. Pour chaque axe une valeur différente peut être réglée.

SETPROP

Syntaxe

SETPROP	**axe [, valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Coefficient d'action proportionnelle. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action proportionnelle*.
Si l'argument **valeur** est omis, le coefficient d'action proportionnelle de configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETPROPI.

SETSLOPE

Syntaxe

SETSLOPE	**axe [, valeur]**

Arguments

axe	Nom du dispositif type axe
valeur	constante ou variable entière. Type de rampe

Description

Cette option configure le type de rampe à utiliser pour le mouvement rapide:

- 0 rampe linéaire
- 1 rampe en esse
- 2 rampe en double esse

Si **valeur** est omise, la rampe de configuration est restaurée.
On peut changer le type de rampe seulement si l'axe est à l'arrêt dans l'état COTE. Dans le cas contraire l'erreur de système "4101 – Gestion non congruente de l'axe NomAxe" est générée.
Conjointement avec cette instruction on peut vérifier le type de rampe de rampe actuellement utilisée par l'axe au moyen de l'instruction GETAXIS avec le paramètre _CURRSLOPE.

Voir également l'instruction SETSLOPEI.

SETVEL

Syntaxe

SETVEL	**axe [, vitesse]**

Arguments

axe	nom de dispositif type axe
vitesse	constante float ou variable float

Description

Cette instruction règle la **vitesse** maximale de l'axe pour les mouvements point-point.

La vitesse est exprimée dans l'unité de mesure de l'axe, indiquée dans la configuration.
Si la **vitesse** programmée dépasse la vitesse maximale de configuration, cette-ci est utilisée.
Si l'argument **vitesse** est omis, la valeur adoptée est celle de la configuration. **Vitesse** n'admet que des valeurs positives.
Voir l'instruction SETVELI.

Exemple

Routine de Zérotage d'un axe

Mouvement interpolé

LOOKAHEAD

Syntaxe

LOOKAHEAD **[valeur]**

Arguments

valeur constante ou variable. Valeur de lookahead

Description

Cette instruction règle la valeur de lookahead de l'interpolateur. Le lookahead est le nombre de blocs d'interpolation qui sont traités avant le début du mouvement des axes. Le lookahead permet de générer des profils de mouvement optimaux, en particulier lorsque l'on utilise des rampes en "S".
Si le paramètre **valeur** n'est pas réglé, le système utilise un lookahead de 512 blocs (défaut).
La valeur maximale admise est égale à **4096/nombrecanaux** où **nombrecanaux** est le nombre de canaux d'interpolation définis dans la configuration de module. La valeur minimale admise est de 256.

Remarque

Le bloc d'interpolation est l'ensemble d'informations associées à n'importe quelle instruction de mouvement interpolé (par ex. : LINEARABS).

Exemple

```
LookAhead 1024
```

SETACCI

Syntaxe

SETACCI **axe1 [, ..., axe6] [, valeur]**

Arguments

axe1,[...axe6] nom de dispositif type axe
valeur constante ou variable. Temps de accélération

Description

Cette instruction attribue aux axes **axe1**, **axe2** le temps d'accélération pour les mouvements interpolés identifié par **valeur**. Le temps est exprimé en millisecondes. Si l'argument **valeur** est omis, le paramètre de configuration est adopté.

Voir également SETACC, SETDEC et SETDECI.

SETACCLIMIT

Syntaxe

SETACCLIMIT **axe,[valeur]**

Arguments

axe nom du dispositif type axe
valeur constante de temps de l'actionnement

Description

Cette instruction active et désactive le calcul automatique de la vitesse de régime d'interpolation en fonction des accélérations tolérées par les axes. Le paramètre **valeur** est une constante de temps utilisée pour déterminer le degré de vitesse toléré par l'axe, en millisecondes. Ce paramètre est

facultatif. Si le paramètre est omis, l'instruction désactive le calcul automatique. Une valeur standard pour ce paramètre est de 30 millisecondes. Si l'on réduit ce laps de temps, le profil est ralenti et le mouvement est plus faible. Si on l'augmente, on obtient l'effet contraire. Cette instruction ne peut pas être appliqué aux interpolations hélicoïdaux.

SETACCSTRATEGY

Syntaxe

SETACCSTRATEGY **axe, [valeur]**

Arguments

axe	nom du dispositif type axe
[valeur]	variable ou constante integer

Description

Cette instruction permet de sélectionner le type d'accélération que l'on désire pour les mouvements d'interpolation suivants. L'instruction doit être exécutée pour tous les axes concernés par l'interpolation.
Les valeurs admises pour le paramètre **valeur** sont 0, 1 et 2. Si la valeur passée est 0, on adopte la stratégie habituelle d'accélération (l'accélération de profil choisie est la plus petite de tous les axes concernés par l'interpolation). En cas de valeur égale à 2 et interpolation linéaire, l'accélération maximale que les différents axes peuvent supporter est adoptée (en tenant compte des différents composants de toutes les axes, linéaires et/ou rotatifs), la gestion de l'accélération en cas d'interpolation circulaire reste inchangée. Le cas de la valeur 1 invoque une gestion obsolète qui est maintenue pour la compatibilité.

SETAXPARTYPE

Syntaxe

SETAXPARTYPE **axe, [valeur]**

Arguments

axe	Nom du dispositif type axe
[valeur]	variable ou constante integer.

Description

Pendant l'exécution d'une interpolation multilinéaire, cette instruction-ci permet de changer la série de paramètres des axes utilisés, de ceux qui sont typiques de l'interpolation (**valeur** =1) à ceux qui sont utilisés pour le mouvement point-point (**valeur** = 0). Si la variable **valeur** est omise, les paramètres utilisés sont ceux de l'interpolation.
On peut changer la série de paramètres seulement si l' axe est arrêtée dans l'état COTE. Dans le cas contraire l'instruction génère l'erreur de système 4101 "Gestion non congruente de l'axe NomAxe".

SETCONTORNATURE

Syntaxe

SETCONTORNATURE **[valeur1,[valeur2]]**

Arguments

valeur1	constante ou variable. Angle maximal de contournage
valeur2	constante ou variable. Angle maximal de décélération

Description

Cette instruction règle l'angle minimal entre les tangentes de deux trajectoires exécutées en interpolation, au-delà duquel la machine n'exécute pas le contournage, ce qui revient à dire que les axes s'arrêtent à la fin de la première trajectoire et qu'ils repartent le long de la deuxième. Pour cela on définit un *angle maximale de contournage* **valeur1**, qui représente l'angle maximale entre deux traits de déplacement au-dessous duquel le mouvement se s'arrête pas. Si l'angle entre deux blocs de déplacement est supérieur à l'angle maximal de contournage, le mouvement s'arrête. Pour éviter l'arrête on peut programmer une valeur d'angle maximal de décélération **valeur2**. Si l'angle entre deux blocs de déplacement est compris entre *l'angle maximal de contournage et* l'*angle maximal de décélération,* le mouvement n'est pas arrête, mais seulement ralenti. L'*angle maximal de décélération* réprésent donc l'angle au-delà duquel le mouvement est obligé à s'arrêter. Pour des angles inférieurs à l'angle maximal de contournage le mouvement **n'est pas ralenti,** pour des angles compris entre l'angle maximal de contournage et l'angle maximal de décélération le mouvement le remplacement **est ralenti,** pour des angles supérieurs à l'angle maximal de contournage le mouvement **s'arrête.**.

Valeur1 et **valeur2** sont paramètres optionnels. Si les deux ne sont pas programmés, la valeur par défaut q'on prend est 15 degrés. Si seulement le premier paramètre est configuré, on suppose que l'*angle maximal de décélération* est égal à *l'angle maximal de contour.* La fonctionnalité de décélération est désactivée lorsque l'*angle maximal de décélération* est inférieur ou égal à *l'angle maximale de contournage*. L'*angle maximal de décélération* est égal a 180 degrés. En programmant une valeur supérieure se genère une erreur de système 4399: "Paramètres hors des limites".
La fonctionnalité de décélération est habilité seulement lorsque l'instruction JERKSMOOTH est active, tandis que le contournage est toujours active.

Remarque

L'utilisation de cette instruction est associée à l'emploi des instructions JERKSMOOTH et SETSLOWPARAM et n'agit que dans les les mouvements avec interpolation classique (instructions LINEARABS, LINEARINC, CIRCABS, CIRCINC, HELICABS, HELICINC).

SETDECI

Syntaxe

SETDECI **axe1 [, ..., axe6] [, valeur]**

Arguments

axe1,[...axe6]	nom de dispositif type axe
valeur	constante ou variable. Temps de décélération

Description

Cette instruction attribue aux axes **axe1**, **axe2** le temps de décélération pour mouvements interpolés identifié par **valeur**. Le temps est exprimé en millisecondes. Si l'argument **valeur** est omis, le paramètre de configuration est adopté.

Voir également SETACC, SETDEC et SETACCI.

SETDERIVI

Syntaxe

SETDERIVI **axe [, valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Coefficient d'action dérivée. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action dérivée* utilisée pendant les mouvements d'interpolation des axes.
Si l'argument **valeur** est omis, le coefficient d'action dérivée de configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETDERIV.

SETFEEDFAI

Syntaxe

SETFEEDFAI **axe [, valeur]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Pourcentage de feed forward

Description

Cette instruction attribue à l'axe la **valeur** *pourcentage de feed forward* d'accélération pour les mouvements interpolés.
Si l'argument **valeur** est omis, le coefficient de feed forward de configuration est adopté.
Si l'instruction est appliquée à un moteur pas à pas, une erreur de système est générée. Il en va de même si on attribue à la variable **valeur** une valeur non comprise entre 0 et 100.
Voir également les instructions SETFEEDF, SETFEEDFI, SETFEEDFA.

SETFEEDI

Syntaxe

SETFEEDI **axe, valeur**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Elle représente le pourcentage de feed rate override

Description

Cette instruction modifie la **valeur** pourcentage de feed rate override de l'axe pour ce qui est des *mouvements en interpolation*. Voir également l'instruction SETFEED.

SETFEEDFI

Syntaxe

SETFEEDFI **axe [, valeur]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Pourcentage de feed forward

Description

Cette instruction attribue à l'axe la **valeur** *pourcentage de feed forward* pour les mouvements en interpolation.
Si l'argument **valeur** est omis, le système prend le pourcentage de feed forward qui est adopté dans les paramètres de configuration du dispositif de l'axe concerné.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Pour la variable **valeur**, les valeurs admises sont comprises entre 0 et 100.
Voir également l'instruction SETFEEDF, SETFEEDFA, SETFEEDFAI.

SETINTEGI

Syntaxe

SETINTEGI **axe [, valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Coefficient d'action intégrale. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action intégrale* utilisée pendant les mouvements d'interpolation des axes.
Si l'argument **valeur** est omis, le coefficient d'action intégrale de configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETINTEG.

SETPROPI

Syntaxe

SETPROPI **axe [, valeur]**

Arguments

axe	nom de dispositif type axe
valeur	constante ou variable. Coefficient d'action intégrale. Les valeurs type char et integer ne sont pas admises

Description

Cette instruction attribue à l'axe la **valeur** *coefficient d'action proportionnelle* utilisée pendant les mouvements d'interpolation des axes.

Si l'argument **valeur** est omis, le coefficient d'action proportionnel de configuration est adopté.
L'instruction ne peut pas être appliquée pour un moteur pas à pas.
Voir également l'instruction SETPROP.

SETSLOPEI

Syntaxe

SETSLOPEI **axe [, valeur]**

Arguments

axe	Nom du dispositif type axe
valeur	constante ou variable entier. Type de rampe

Description

Cette option configure le type de rampe à utiliser pour le mouvement dans l'interpolation (dans les cas où est autorisé) :

- 0 rampe linéaire
- 1 rampe en esse
- 2 rampe en double esse

Si **valeur** a été omise, la rampe de configuration est restaurée.

Vous pouvez changer le type de rampe seulement si l'axe n'est pas encore occupé dans un canal d'interpolation. Dans le cas contraire l'erreur de système "4101 – Gestion non congruente de l'axe NomAxe" est générée.
Conjointement avec cette instruction au moyen de l'instruction GETAXIS, en utilisant le paramètre _CURRSLOPEI on peut vérifier le type de rampe actuellement configurée pour l'axe et en utilisant le paramètre _REALSLOPEI on peut voir le type de rampe utilisée (celle du canal dans laquelle l'axe est occupé) par l'axe.

Voir également l'instruction SETSLOPE.

SETSLOWPARAM

Syntaxe

SETSLOWPARAM **axe, [valeur1, valeur2]**

Arguments

axe	nom de dispositif type axe
valeur1	constante ou variable double. Facteur de réduction générale
valeur2	constante ou variable double. Facteur de réduction inversion

Description

L'instruction modifie les valeurs nécessaires à calculer la vitesse de décélération, si la fonctionnalité de décélération en contournage est active (voir instruction SETCONTORNATURE).
La vitesse de décélération est calculée initialement d'une façon théorique d'axe en axe. Dans le cas d'inversion de mouvement on peut la réduire en utilisant dans les calcules la **valeur2.** Ensuite, entre toutes les vitesses calculées on considère la vitesse minimale de manière à respecter la dynamique de l'axe plus limitante. Il est possible enfin de réduire encore plus la vitesse de décélération d'un facteur qui dépend de **valeur1**.

Si **valeur1** o **valeur2** sont omises, on prend des valeurs par défaut, qui correspondent à une des leurs actions qui est nulle. Le paramètre **valeur1** représente la valeur de pourcentage de réduction de la vitesse de décélération théorique. La vitesse de décélération appliquée est égal à (100 - **valeur1**) / 100 fois la vitesse de décélération théorique. La valeur maximale de réduction est égal a 100. En ce cas la vitesse résultante est égal a 100. Au contraire, quand la valeur est zéro ou est omise, on considéré la valeur par défaut, c'est à dire on considère la vitesse théorique entière.
Le paramètre **valeur2** représente le pourcentage de réduction de 1 à 10 fois, de la vitesse de décélération théorique, quand l'axe invertit son mouvement. En particulier, on suppose donc que quand **valeur2** est 100 la vitesse est réduite de 10 fois. Au contraire, quand la valeur est zéro ou o est omise, la vitesse ne se réduit pas.
L'instruction génère une erreur de système 4399 "Paramètre hors de la plage", quand la valeur programmée est inférieure à zéro ou supérieure a 100. Il est important de se rappeler que si le paramètre **valeur1 est omis** on doit omettre le paramètre **valeur2** aussi.

Remarque

L'emploi de cette instruction est associé à l'utilisation des instructions JERKSMOOTH et SETCONTORNATURE et n'agit que dans les les mouvements avec interpolation classique (instructions LINEARABS, LINEARINC, CIRCABS, CIRCINC, HELICABS, HELICINC).

SETVELI

Syntaxe

SETVELI **axe1 [, ..., axe6] [, vitesse]**

Arguments

axe1 [...axe6]	nom du dispositif type axe à interpoler
vitesse	constante float ou variable float

Description

Cette instruction règle la **vitesse** des axes **axe1**, **axe2**, pour les mouvements interpolés.
La vitesse est exprimée dans l'unité de mesure des axes, indiquée par le paramètre de configuration. Si l'argument **vitesse** est omis, la valeur maximale de configuration est adoptée.
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un remote TRS-AX.
Voir l'instruction SETVEL.

SETVELILIMIT

Syntaxe

SETVELILIMIT **axe , vitesse**

Arguments

axe	nom du dispositif type axe
vitesse	constante float ou variable float

Description

Règle les différents composants de **vitesse** de l'**axe** indiqué, pour les mouvements interpolés.
La vitesse est exprimé avec l'unité de mesure de l'axe.

Mouvement coordonné

SETFEEDCOORD

Syntaxe

SETFEEDCOORD **axe, valeur1, valeur2**

Arguments

axe	nom du dispositif type axe
valeur1	constante ou variable double. Représente le pourcentage maximum de variation instantanée du feedrate
valeur2	constante ou variable integer. Représente le numéro de real-time où ne s'applique aucune variation des feedrate

Description

Modification de la **valeur1** en pourcentage de variation maximum instantanée du feed rate de l'**axe**.
Le feed rate ne subit plus de variations pendant l'intervalle de temps, exprimé en real-time, défini dans la variable **valeur2**. En d'autres termes, après avoir appliqué une variation de feedrate override, au maximum égal à **valeur1** par un numéro de real-times égal à **valeur2** aucune nouvelle variation de feedrate peut être appliée.
La combinaison de ces deux paramètres définit une sorte d'accélération/décélération que l'axe peut supporter. En modulant ces deux paramètres, il est possible d'obtenir des rampes "en escalier" de l'inclinaison voulue.

Note

Pour chaque axe concerné par le mouvement coordonné, il faut configurer la valeur de feedrate et l'intervalle de temps, sinon les valeurs par défaut, **valeur1**=100 et **valeur2**=1 sont prises. Pendant l'exécution du mouvement coordonné (instruction COORDIN) le système recalcule les paramètres **valeur1** et **valeur2** à appliquer au mouvement, en fonction des paramètres de tous les axes

concernés. Les axes à l'arrêt sont exclus du contrôle. Les deux paramètres sont calculés de façon à ce que:
valeur1 : valeur minimum configurée sur les axes qui sont en mouvement
valeur2 : valeur obtenue en divisant **valeur1** pour le rapport minimum **valeur1/valeur2.**

Exemple

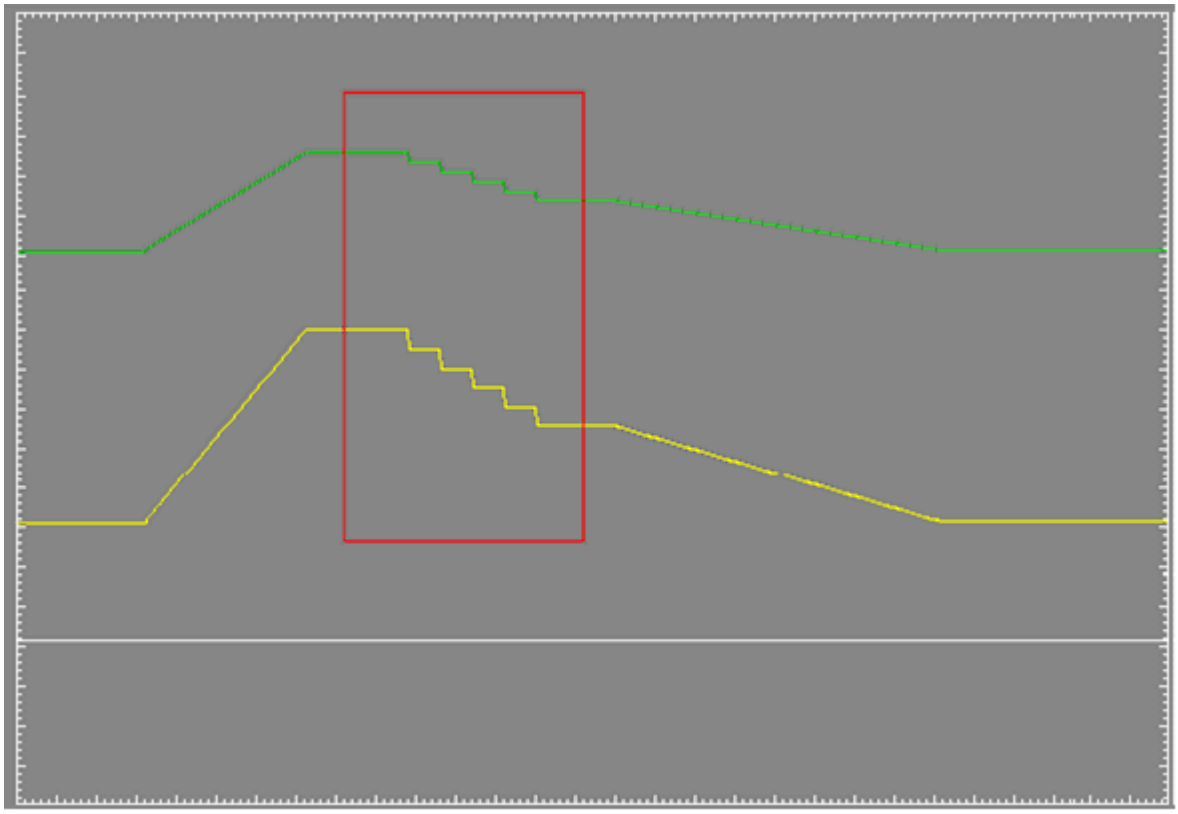
```
;
Fonction MouvementCoordonné


      Setquote              X,0
      Setquote              Y,0
      Setquote              Z,0


      setFeedCoord          X, 20, 80
      setFeedCoord          Y, 10, 1
      setFeedCoord          Z, 3, 3


      coordin               matrix, deltaT, UP, rigaInit, rigaEnd,mask,_
                            X,columnX, Y,columnY, Z,columnZ
      waitstill             x,y,z
```

fret

Supposons qu'à un passage déterminé du mouvement coordonné, l'axe Z ne se déplace pas. Les paramètres configurés résultent être

Variation_Max = **10**
Delta_T = 10 / 0.25 = **40**

Nous obtenons donc le tracé d'oscilloscope suivant, avec en vert le profil de vitesse de l'axe X et en jaune celui de l'axe Y.



SETOFFSET

Syntaxe

SETOFFSET	**axe, cote**

Arguments

axe	nom du dispositif type axe
cote	constante ou variable. Offset pour mouvement coordonné

Description

Cette instruction permet d'appliquer un offset aux cotes d'un mouvement coordonné.
L'offset indiqué par le paramètre **cote** est utilisé dans les mouvements coordonnés suivants, en ajoutant la cote indiquée à toutes les cotes présentes dans le tableau.
Voir également l'instruction COORDIN.

Mouvement enchaîné

RATIO

Syntaxe

RATIO	**axe, [valeur]**

Arguments

axe	noms de dispositifs type axe.
valeur	constante ou variable. Rapport de réduction.

Description

Programme le rapport de chaînage d'un axe slave par rapport à son master. Les mouvements de l'axe slave sont décalés par rapport à ceux du master selon le rapport de chaînage programmé. Si le paramètre **valeur** est omis, le rapport est rétabli à 1.0 (mouvements identiques). L'instruction génère une erreur de système si elle est exécutée lorsque l'axe n'est pas en état slave et que l'axe master correspondant n'est pas en état cote.
Voir l'instruction CHAIN.

Exemple

```
CHAIN       X, Y
RATIO       Y, 0.5   ; rapport de réduction 1/2

MOVABS      X, 100   ; l'axe Y se déplace à la cote 50
WAITSTILL   X
```

SETDYNRATIO

Syntaxe

SETDYNRATIO	**axe, valeur**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable double

Description

Cette instruction permet de changer d'un façon dynamique, pendant le mouvement de l'axe maître, le rapport d'enchaînement. Il est possible d'appliquer une nouvelle valeur du rapport d'enchaînement, même si la variation précédent n'est pas terminée. L'**axe** déclarée doit être une axe esclave.
Dans le cas où l'instruction est exécutée avec axe maître en état POSITION, la nouvelle valeur du rapport d'enchaînement **valeur** est appliquée instantanément.
La variation du rapport de enchaînement se produit par moyen d' une rampe de accélération (ou décéleratione) lineaire. La valeur d'accélération utilisée est donnée par l'accélération de l'axe Master courante pour le mouvement point-point. Cela signifie qu'il est également possible de modifier cette rampe par moyen d'une nouvelle valeur d'accélération, en utilisant l'instruction SETACC.
Cette instruction peut générer l'erreur de système suivante :

- "4101 – Gestion non congruente de l'axe", dans le cas où l'axe déclarée n'est pas une axe esclave.

Paramètres Génériques

DYNLIMIT

Syntaxe

DYNLIMIT **axe, état**

Arguments

axe	Nom du dispositif type axe
état	constante prédéfinie. Les valeurs admises sont : **ON** habilitation contrôle dynamique limites axe **OFF** déshabilitation contrôle dynamique limites axe

Description

Habilite ou déshabilite le test dynamique sur le dépassement des limites axe.
Le test dynamique de dépassement des limites axe se différencie du test statique de dépassement des limites axe dans la mesure où il vérifie à chaque real-time que l'axe dépasse ses limites, en fonction de sa vitesse actuelle et de sa décélération maximale. En revanche, le test de type statique contrôle, moment par moment, que la cote d'arrivée actuelle de chaque axe se trouve à l'intérieur des limites axe positif ou négatif configurés. Par ailleurs, avant le départ du mouvement, le test de type statique contrôle si les cotes passées à travers les instructions de mouvement dépassent les limites configurées.

Avant d'une instruction DYNLIMIT, on doit programmer les instructions SETLIMPOS et SETLIMNEG, à fin qu'elle définissent les nouvelles limites.

Exemple

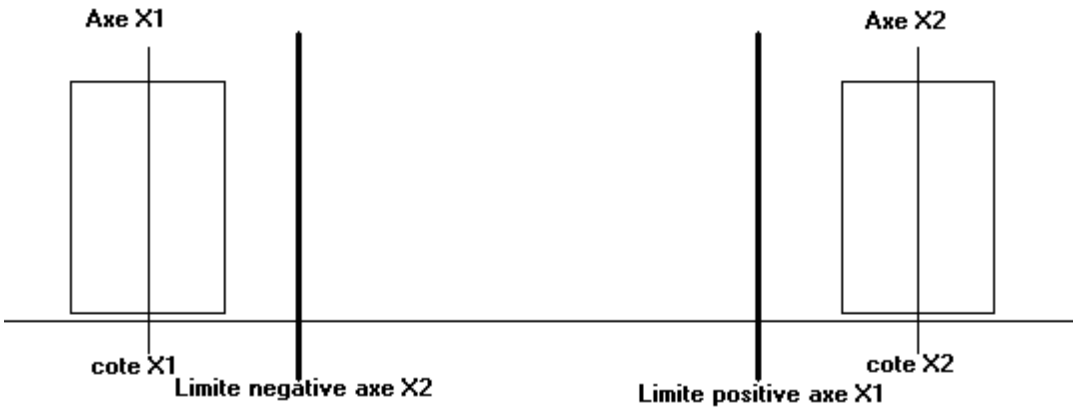
Vérification des limites axes selon les deux typologies de test statique et dynamique, avec les axes sur la même direction de mouvement.

Test statique

Dans un mouvement générique, l'**Axe X1** ne peut pas être supérieur à la limite positive initiale étant donné la cote de l'**Axe X2**. La vérification des limites axes génère une erreur de système no. 4108 "Axe X1: cote finale au-delà des limites software".

Test dynamique

Dans un mouvement générique, contrôler que la cote instantanée **cote X1** soit comprise dans les limites axe diminuées, avec le signe opportun selon la direction de mouvement de l'axe, de l'espace minimum d'arrêt de l'axe même. L'espace minimum d'arrêt est calculé en fonction de la vitesse instantanée et de la décélération réglée en configuration pour le mouvement point-point. Par ailleurs, le contrôle initial du dépassement des limites réglées par les cotes d'après les instructions de mouvement est déshabilité.



ENABLESTARTCONTROL

Syntaxe

ENABLESTARTCONTROL **axe, [timeout]**

Arguments

axe	nom du dispositif type axe

timeout	variable ou constante integer. Il s'agit du délai d'attente exprimé en real-time

Description

Cette instruction permet d' habiliter et programmer le **timeout** pour le contrôle du manqué départ ou subit stop de l'axe.
Si l'axe non bouge pas de ou du moins 2 pas en 200 real-time après une demande de mouvement est généré une erreur de système no. 3 "Servoerror".
Si le paramètre **timeout** est programmé à zéro il est désactivé le contrôle. L'instruction n'a pas d'effect si la vitesse théorique est mineur de deux pas en 200 real-time ou si le mouvement finit en moins de 200 real-time.

Exemple

```
; timeout de début axes de 10 real-time
ENABLESTARTCONTROL x, 10
```

NOTCHFILTER

Syntaxe

NOTCHFILTER	**axe, [valeur]**

Arguments

axe	noms de dispositifs type axe
valeur	constante ou variable. Valeur de fréquence [Hz]. Valeurs admises comprises entre **0** et **500**.

Description

Règle la fréquence de coupe du filtre notch pour l'axe spécifié. Si le paramètre **valeur** est égal à 0, le filtre est invalidé. Au cas où le paramètre **valeur** serait omis, l'on utilisera la valeur introduite dans la configuration.

Exemple

```
; coupe de la   97 Hz
fréquence
NOTCHFILTER     x, 97
```

RESLIMNEG

Syntaxe

RESLIMNEG	**axe**

Arguments

axe	nom de dispositif type axe

Description

Cette instruction désactive le test sur la limite negative de l'axe identifié.
Habituellement, ces instructions sont utilisées dans les routines de zérotage pour la recherche des switches de zérotage, ce qui permet ainsi de dépasser les valeurs de configuration programmées.
Voir également les instructions SETLIMNEG, SETLIMPOS, RESLIMPOS.

Exemple

Routine de Zérotage d'un axe

RESLIMPOS

Syntaxe

RESLIMPOS	**axe**

Arguments

axe	nom de dispositif type axe

Description

Cette instruction désactive le test sur la limite positive de l'axe identifié.
Habituellement, ces instructions sont utilisées dans les routines de zérotage pour la recherche des switches de zérotage, ce qui permet ainsi de dépasser les valeurs de configuration programmées.
Voir également les instructions RESLIMNEG, SETLIMPOS, SETLIMNEG.

Exemple

Routine de Zérotage d'un axe

SETADJUST

Syntaxe

SETADJUST **axe, état, [valeur]**

Arguments

axe	nom de dispositif type axe
état	constante prédéfinie. Les valeurs pouvant être prises sont les suivantes : • **ON** actif • **OFF** non actif
[valore]	variable ou constante float. Tension [V]

Description

Cette instruction active ou désactive, sur l'axe indiqué, le calcul de récupération automatique des offsets, c'est-à-dire l'adjust.
L'adjust permet de compenser de légers offsets de position en fin de mouvement de l'axe.
Normalement, l'adjust est activé.

Il peut s'avérer intéressant de désactiver l'adjust pour les axes actionnés par des moteurs qui présentent une forte hystérésis de position et qui ne tireraient donc aucun avantage de l'utilisation de cette fonction du contrôle.

Lorsque l'adjust est réactivé après une désactivation, le contrôle ne prend pas en compte la valeur calculée précédemment. L'instruction peut alors être utilisée pour réinitialiser le réglage accumulé d'un axe sans réinitialiser la commande.

Lorsque le troisième paramètre est présent, l' offset est réglé à la **valeur** indiquée indépendamment de l'activation ou de la désactivation de l'adjust automatique. Cette utilisation de l'instruction permet de compenser tout décalage de référence de vitesse par rapport au logiciel au lieu de le compenser sur le variateur, même si la compensation sur le variateur est préférée à la compensation par le logiciel.
L'instruction n'est applicable qu'aux axes à commande analogique.

SETBACKLASH

Sintaxe

SETBACKLASH **axe, valeur**

Arguments

axe	nom du dispositif type axe
valeur	variable ou constante float. Valeur du jeu (backlash).

Description

Cette instruction permit de diminuer ou éliminer les effets des jeux (backlash) sur la trajectoire de l'axe. La **valeur** du jeu qui peut être programmé doit être comprise entre 0.0 et 3.0. Cette valeur est indépendante de l'unité de mesure sélectionnée.
Situations particulières se passent dans les cas suivants :

- avec axe déshabilité la fonction de récupère jeux, même si demandé, n'est pas appliquée.
- avec axe vertical, vue la particulière configuration, il n'y a pas de jeux qui se manifeste.
- avec axe avec cargaison de grande inertie, il peut se vérifier une partiale ou quelque fois totale compensation de la cargaison. Il est possible en effet que, du à la masse de la cargaison, le mouvement des axes s'arrête en retard respect à l'arrêt du moteur. Le positionnement résultant des dents de l'engrenage réducteur respect au positionnement des dents de l'engrenage moteur peut être donc tel de réduire ou même annuler le jeu.

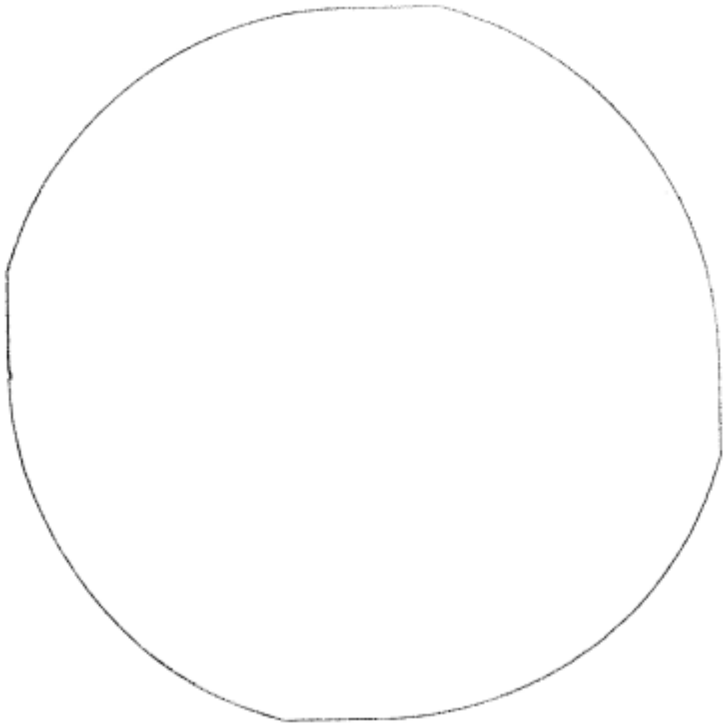
- la visualisation des coordonnées réel et encoder de l'axe, échantillonnées avec l'oscilloscope, dans les points dans lesquels est activé le récupère jeu (inversion du mouvement), présent un pic égal à la valeur du jeu même.

L'instruction engendre erreur de système si utilisée :
- sur axes pas à pas non gérés par dispositifs à distance TRS-AX, de comptage, virtuels
- sur axes pas à pas gérés par dispositifs à distance TRS avec codeur simulé.

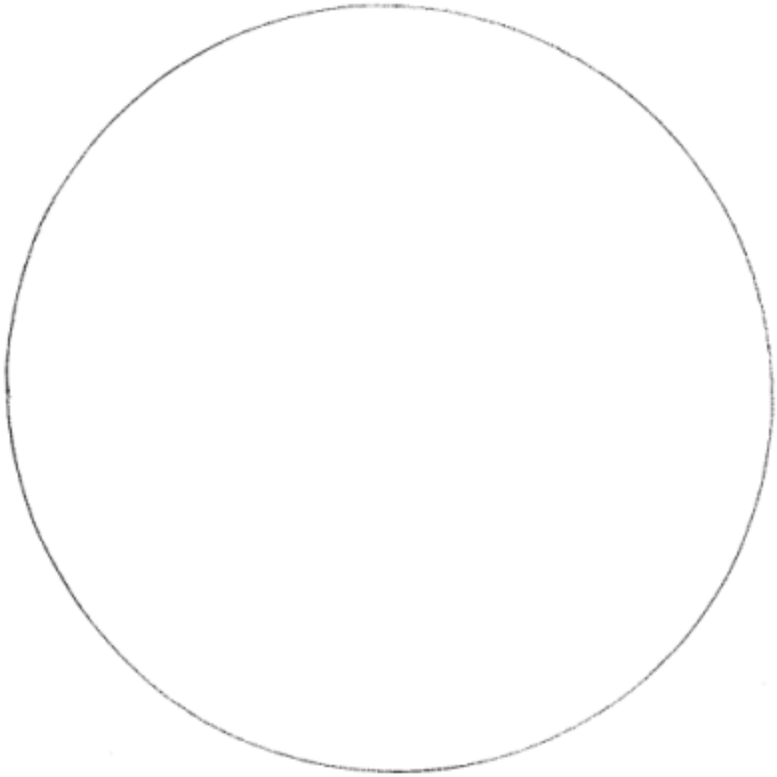
Exemple

```
;   Fonction avec récupère jeux déshabilité

   SETQUOTE                X, 0
   SETQUOTE                y, 0
   SETVELI                 X, 1.0
   CIRCLE                  X,Y,cw,100,90
   WAITSTILL               X,Y

; Fonction avec récupère récupère jeux habilité
   SETQUOTE                X, 0
   SETQUOTE                y, 0
   SETVELI                 X, 1.0
   SETBACKLASH             X, 1.9
   SETBACKLASH             y, 1.8
   CIRCLE                  X,Y,cw,100,90
   WAITSTILL               X,Y
```

L'exécution de deux fonctions engendre deux traces différentes.
La première figure montre l'interpolation de deux axes qui présentent un jeu dans la couple moteur-réducteur.



La deuxième figure représente la même interpolation, mais avec l'utilisation de l'instruction de récupère jeux.



SETBIGWINFACTOR

Syntaxe

SETBIGWINFACTOR **axe, valeur**

Arguments

axe	nom du dispositif type axe
valeur	variable ou constante double. Facteur de multiplication pour le calcul de la fenêtre grande.

Description

Cette instruction permet de modifier le facteur de multiplication pour le calcul de la fenêtre grande sur l'**axe** choisi. Pour le calcul de la fenêtre grande, il faut multiplier la variable **valeur** par le paramètre défini en configuration des axes de fenêtre d'arrivée en cote. La **valeur** qui peut être réglée doit être comprise entre 1 et 257, excluant les valeurs extrêmes. Par défaut elle est de 4.0.

SETDEADBAND

Syntaxe

SETDEADBAND **Axe,VMinPos,VMinNeg,VSeuilPos,VSeuilNeg**

Arguments

Axe	nom de dispositif type axe
VMinPos	variable ou constante float. Voltage minimal positif [V]
VMinNeg	variable ou constante float. Voltage minimal négatif [V]
VSeuiPos	variable ou constante float. Seuil positif [V]
VSeuilNeg	variable ou constante float. Seuil négatif [V]

Description

Règle les paramètres de voltage minimal pour l'axe indiqué. Les valeurs de voltage minimal (positif/négatif) sont additionnées au voltage de référence théorique (positif/négatif) si la valeur de ce-ci dépasse la valeur de seuil (positive/négative) réglée. Si le voltage de référence théorique se trouve à l'intérieur des valeurs de seuil, le voltage de référence réel est forcé à zéro. Il est possible

de désactiver la gestion du voltage minimal en réglant tous les valeurs à zéro. Les valeurs de seuil doivent être toujours inférieures ou égales aux respectives valeurs de voltage minimal.
À l'initialisation du système la gestion du voltage minimal est désactivé.

SETENCLIMIT

Syntaxe

SETENCLIMIT **axe [, valeur]**

Arguments

axe nom de dispositif type axe
valeur constante ou variable double

Description

Cette instruction permet de modifier la limite de enchaînement encoder incorrect . Le paramètre est exprimé avec l'unité de mesure de l'axe. Les valeurs admises doivent être comprises en un intervalle équivalent à 128 – 16384 pas encoder. Si le paramètre n'est pas présent il est rétabli la valeur de default qui corresponde à 1024 pas.
Par exemple, pour un axe avec résolution de 1000 impulsions/mm les valeurs admises seront compris entre 0,128 et 16,384 mm.
Si le paramètre **valeur** est programmé à zéro il est désactivé le contrôle de la limite de enchaînement encoder incorrect.

Exemple

```
; imposé une limite de enchaînement encoder de 3.5

  SETENCLIMIT X, 3.5
```

SETINDEXEN

Syntaxe

SETINDEXEN axe, état

Arguments

axe nom du dispositif type axe
état constante predefinie. Les valeurs pouvant etre prises sont les suivantes :
ON état cran de zéro actif
OFF état cran de zéro non actif

Description

Active ou désactive sur l'axe indiqué le cran de zéro.
Pour exécuter cette instruction l'axe doit être de type comptage.

SETINTEGTIME

Syntaxe

SETINTEGTIME **axe [, valeur]**

Arguments

axe nom du dispositif type axe
valeur constante integer ou variable

Description

Règle le nombre de étalons d'erreurs d'anneau utilisés pour calculer la composante intégrale. Les valeurs valables sont compris entre 1 et 200. La variation de ce paramètre au vol est possible mais elle peut provoquer des discontinuités sur la référence de vitesse de l'axe. Il est conseillé, donc, d'agir sur ce paramètres avec les axes arrêtés et déactivés ou de préférence en free.

SETIRMPP

Syntaxe

SETIRMPP **axe, vitesse**

Arguments

axe	nom du dispositif type axe
vitesse	constante float ou variable float. Vitesse de début de rampe

Description

Cette instruction attribue à l'axe la valeur de **vitesse** *de début de rampe*. C'est la vitesse minimale du moteur pas à pas.
Cette instruction est utilisée pour les axes actionnés par des moteurs pas à pas.

SETLIMNEG

Syntaxe

SETLIMNEG **axe [, cote]**

Arguments

axe	nom du dispositif type axe
cote	constante ou variable. Limite négative

Description

Cette instruction attribue à l'axe la **cote** limite négative.
Si le paramètre **cote** est omis, la limite négative de configuration est adoptée.
Habituellement, ces instructions sont utilisées dans les routines de zérotage pour la recherche des switches de zérotage, ce qui permet ainsi aux axes de dépasser les valeurs de configuration programmées.
Voir également les instructions RESLIMNEG, SETLIMPOS, RESLIMPOS.

Exemple

Routine de Zérotage d'un axe

SETLIMPOS

Syntaxe

SETLIMPOS **axe [, cote]**

Arguments

axe	nom du dispositif type axe
cote	constante ou variable. Limite positive

Description

Cette instruction attribue à l'axe la **cote** limite positive.
Si le paramètre **cote** est omis, la limite positive de configuration est adoptée.
Habituellement, ces instructions sont utilisées dans les routines de zérotage pour la recherche des switches de zérotage, ce qui permet ainsi aux axes de dépasser les valeurs de configuration programmées.
Voir également les instructions RESLIMNEG, RESLIMPOS, SETLIMNEG.

Exemple

Routine de zérotage d'un axe

SETMAXER

Syntaxe

SETMAXER **axe, valeur [, direction]**

Arguments

axe	nom du dispositif type axe
valeur	constante ou variable. Erreur maximale de suivi
direction	constante prédefinie. Direction de l'axe. Peut prendre les valeurs suivantes : **POSITIVE, NEGATIVE**

Description

Ce paramètre attribue à l'**axe :**

- la **valeur** maximale de poursuite admise dans le cas de test statique du servoerror;
- la **valeur**, qui ajoutée à l'erreur théorique proportionnelle à la vitesse, détermine la valeur maximale de poursuite tolérée dans le cas de test dynamique du servoerror.

Si la **direction** est omise, la valeur de maximum de poursuite est réglée dans les deux directions.

SETMAXERNEG

Syntaxe

SETMAXERNEG	**axe, retard , avance**

Arguments

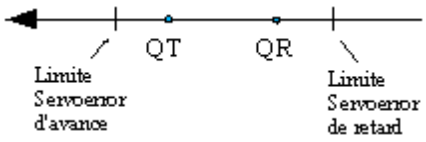
axe	nom du dispositif type axe
retard	constante ou variable. Erreur maximale de poursuite
avance	constante ou variable. Erreur maximale d'avance

Description

Cette instruction attribue à l'**axe** les valeurs maximales d'erreur de poursuite **(retard)** et d'avance **(avance)** admises par le contrôle, seulement en direction négative, avant de générer le signal de "servoerror".

- **retard** est la valeur maximale de poursuite tolérée dans le cas de test statique de la servoerror ou, dans le cas du test dynamique de servoerror, la valeur qui, ajoutée à l'erreur théorique proportionnelle à la vitesse, détermine la valeur de poursuite maximale tolérée.
- **avance** est la valeur maximale de poursuite tolérée pendant l'inversion de mouvement du mouvement négatif au mouvement positif.

L'erreur de poursuite est la différence entre la cote théorique (où l'axe devrait se trouver) et la cote réelle. Lorsque l'axe se déplace dans la direction négative, une erreur de poursuite de signe négatif indique une condition de retard de l'axe ; une erreur de poursuite de signe positif indique une condition d'avance. Si l'on n'utilise pas cette instruction, le contrôle utilise les valeurs maximales d'erreur de poursuite présentes dans la configuration de l'axe. Dans ce cas, la limite d'avance est égale au quart de la limite de retard.



Exemple

```
SETMAXERNEG Assi.X, 10, 5

;Le retard maximal de l'axe est de 10 mm, l'avance maximale est
;de 5 mm
```

SETMAXERPOS

Syntaxe

SETMAXERPOS	**axe, retard , avance**

Arguments

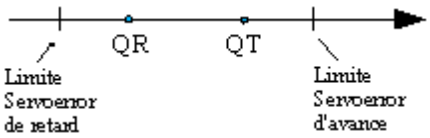
axe	nom du dispositif type axe
retard	constante ou variable. Erreur maximale de poursuite
avance	constante ou variable. Erreur maximale d'avance

Description

Cette instruction attribue à l'**axe** les valeurs maximales d'erreur de poursuite (retard) et d'avance (avance) admises par le contrôle, seulement en direction positive, avant de générer le signal de "servoerror".

- **retard** est la valeur maximale de poursuite tolérée dans le cas de test statique de la servoerror ou, dans le cas du test dynamique de servoerror, la valeur qui, ajoutée à l'erreur théorique proportionnelle à la vitesse, détermine la valeur de poursuite maximale tolérée.
- **avance** est la valeur maximale de poursuite tolérée pendant l'inversion de mouvement du mouvement négatif au mouvement positif.

L'erreur de poursuite est la différence entre la cote théorique (où l'axe devrait se trouver) et la cote réelle. Lorsque l'axe se déplace dans la direction positive, une erreur de poursuite de signe positif indique une condition de retard de l'axe, alors qu' une erreur de poursuite de signe négatif indique une condition d'avance. Si l'on n'utilise pas cette instruction, le contrôle utilise les valeurs maximales d'erreur de poursuite présentes dans la configuration de l'axe. Dans ce cas, la limite d'avance est égale au quart de la limite de retard.



Exemple

```
SetMaxErPos Assi.X, 10, 5

;Le retard maximal de l'axe est de 10 mm, l'avance maximale est
;de 5 mm
```

SETMAXERTYPE

Syntaxe

SETMAXERTYPE **axe, type**

Arguments

axe	nom du dispositif type axe
type	constante type integer. Les valeurs admises sont les suivantes : 0 = programme servoerror de seuil (valeur de défaut) 1 = programme servoerror dynamique

Description

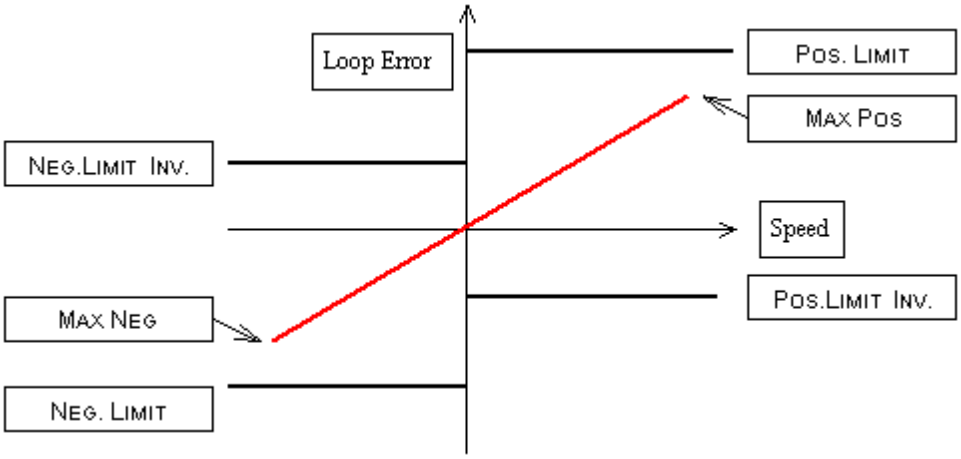
Cette instruction permet de programmer le **type** de test sur le servoerror. La gestion traditionnelle du servoerror prévoit une couple de limites (positive et négative) constantes avec la variation de la vitesse de l'axe. Ce type de gestion porte à dimensionner les limites en fonction de la vitesse maximale de l'axe, ou bien on programme une limite qui permet de ne pas déclencher l'erreur en conditions de normal fonctionnement. Pourtant pendant les basses vitesses l'erreur de boucle a valeurs généralement plus bas pas rapport à la limite programmée, cela comporte un retard dans l'identification d'une condition d'erreur.

La gestion du servoerror à fenêtre est basée sur le calcul de l'erreur de boucle théorique. Les limites de servoerror positive et négative sont calculées en fonction de ce-ci en leur additionnant et soustrayant une valeur de seuil. Si l'erreur de boucle réel sort de ce seuil est généré l'erreur de servoerror.

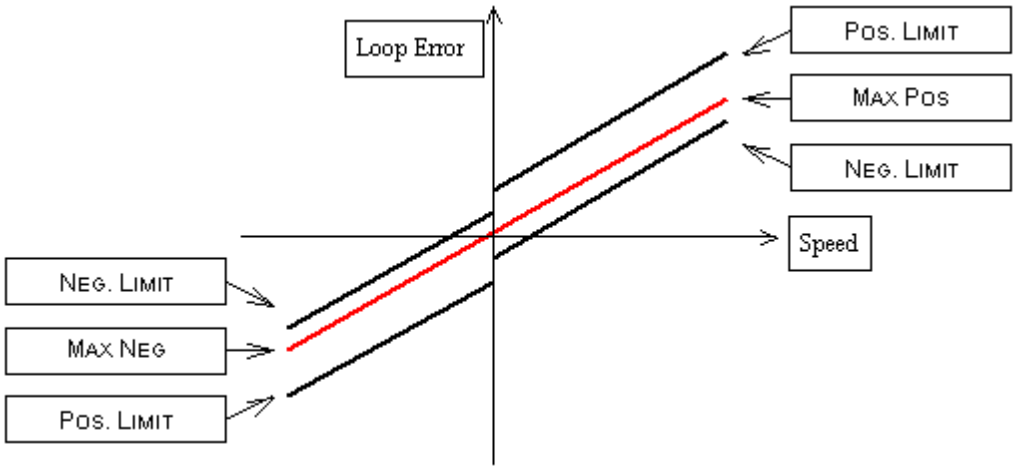
Remarque

Se on établit le test sur le servoerror dynamique est généralement nécessaire modifier les valeurs de Limite servoerror positive et Limite servoerror négative programmés en configuration axes pour le servoerror à seuil. Cela parce-que les susdits valeurs sont utilisés comme valeurs initiales pour le calcule de l'erreur de boucle.

Limite ServoError "Classique":



Limite ServoError "à Fenêtre":



SETPHASESINV

Syntaxe

SETPHASESINV **axe, état**

Arguments

axe nom du dispositif type axe
état constante prédéfinie. Les valeurs pouvant être prises sont les suivantes :
ON état inversion phases actif
OFF état inversion phases désactif

Description

Active ou désactive sur l'**axe** indiqué l'inversion phases. Il permet de compenser via software une éventuelle inversion de câblage des phases encoder. Si utilisé avec l'inversion référence permet d'invertir le vers de l'axe (si le câblage est correct).
Pour exécuter cette instruction l'axe doit être dans l'état FREE.

SETREFINV

Syntaxe

SETREFINV **axe, état**

Arguments

axe nom de dispositif type axe
état constante prédéfinie. Les valeurs pouvant être prises sont les suivantes :

ON active l'inversion de la référence de vitesse
OFF désactive l'inversion de la vitesse

Description
Il active ou désactive sur l'**axe** indiqué l'inversion de la référence de vitesse. Si utilisé avec l'inversion phases, il permet d'invertir le vers de l'axe (si le câblage est correct).
Pour exécuter cette instruction l'axe doit être dans l'état FREE.
Voir aussi SETPHASESINV.

SETRESOLUTION

Syntaxe
SETRESOLUTION **axe [, valeur]**

Arguments
axe nom du dispositif type axe
valeur constante ou variable double

Description
La résolution de l'axe indiqué change Si **valeur** est omis, on utilise la valeur de résolution programmée dans la configuration. La modification de la valeur de résolution ne peut être exécutée que si l'axe est arrêté (état axe = cote). Autrement, l'erreur de système no.4101 "Gestion non congruente de l'axe" s'affiche.

10.3.5 Compteurs

DECOUNTER

Syntaxe
DECOUNTER **nomcompteur [, valeur]**

Arguments
nomcompteur nom de dispositif type compteur
valeur constante, variable ou dispositif type compteur

Description
Cette instruction décrémente le compteur **nomcompteur** de la **valeur** indiquée. Si le paramètre **valeur** est omis, la valeur prise est 1. Voir également les instructions SETCOUNTER et INCOUNTER.

INCOUNTER

Syntaxe
INCOUNTER **nomcompteur [, valeur]**

Arguments
nomcompteur nom de dispositif type compteur
valeur constante ou variable ou dispositif type compteur

Description
Cette instruction incrémente le compteur **nomcompteur** de la **valeur** indiquée. Si le paramètre **valeur** est omis, la valeur prise est 1. Voir également les instructions SETCOUNTER et DECOUNTER.

SETCOUNTER

Syntaxe
SETCOUNTER **nomcompteur, valeur**

Arguments
nomcompteur nom de dispositif type compteur
valeur constante ou variable ou dispositif type compteur

Description
Cette instruction règle le compteur **nomcompteur** à la **valeur** indiquée.

Voir également INCOUNTER et DECOUNTER.

10.3.6 Temporisateurs

HOLDTIMER

Syntaxe

HOLDTIMER **nomtemporisateur**

Arguments

nomtemporisateur nom de dispositif type temporisateur

Description

Cette instruction bloque l'actualisation du temporisateur **nomtemporisateur**.
Voir également STARTTIMER et SETTIMER.

SETTIMER

Syntaxe

SETTIMER **nomtemporisateur, temps**

Arguments

nomtemporisateur nom de dispositif type temporisateur
temps constante ou variable ou dispositif type temporisateur

Description

Cette instruction règle **nomtemporisateur** au **temps** indiqué (en secondes).
Seulement les valeurs supérieures à zéro sont admises. La précision maximale des temporisateurs est de 4 ms.
Voir également STARTTIMER et HOLDTIMER.

Exemple

```
;La Function règle un temporisateur
;Le temporisateur TimeOut a été fixé
;à la valeur 20 secondes
SETTIMER    Timeout,20
; le temporisateur démarre en mode descendant. Lorsqu'il atteint 0, il
s'arrête

STARTTIMER      Timeout,DOWN
```

STARTTIMER

Syntaxe

STARTTIMER **nomtemporisateur [, direction]**

Arguments

nomtemporisateur nom de dispositif type temporisateur
direction constante prédéfinie. Les valeurs pouvant être définies sont les suivantes:
UP croissante
DOWN décroissante

Description

Cette instruction fait partir le temporisateur **nomtemporisateur** dans le mode éventuellement indiqué par **direction**.
Si le paramètre **direction** est omis, le mode adopté est **DOWN**.
Lorsqu'un temporisateur (lancé en mode décroissant) arrive à zéro, il s'arrête automatiquement.
Voir également HOLDTIMER et SETTIMER.

10.3.7 Variables, Vecteurs, Matrices

CLEAR

Syntaxe

CLEAR	**nomvar ou vecteur ou matrice[lignematrice]**

Arguments

nomvar	nom de variable
vecteur	nom de vecteur
matrice	nom de matrice
lignematrice	constante ou variable ou compteur. Ligne de la matrice

Description

Cette instruction initialise à la valeur 0 l'espace de mémoire réservé aux variables (**nomvar**), aux vecteurs (**vecteur**), aux matrices (**matrice**) ou aux éléments d'une ligne de matrice.

FIND

Syntaxe

FIND	**matrice, colonne, limite_min, limite_max, valeur, variable**
FIND	**vecteur, limite_min, limite_max, valeur, variable**

Arguments

matrice	nom de la matrice. Matrice dans laquelle la recherche doit être exécutée
vecteur	nom du vecteur. Vecteur dans lequel la recherche doit être exécutée
colonne	constante ou variable entière ou nomcompteur. Numéro de colonne de la matrice dans laquelle la recherche doit être exécutée
limite_min	constante ou variable. Indice minimal du vecteur ou de la matrice d'où la recherche doit commencer
limite_max	constante ou variable. Indice maximal du vecteur ou de la matrice où la recherche doit s'achever
valeur	constante ou variable. Valeur à rechercher
variable	variable. Résultat de la recherche

Description

Cette instruction exécute une recherche séquentielle d'une valeur à l'intérieur d'un **vecteur** ou d'une **colonne** de la **matrice** et met l'indice de l'élément en **variable**.
Si la valeur n'a pas été trouvée, la **variable** contiendra la valeur -1.

FINDB

Syntaxe

FINDB	**matrice, colonne, limite_min, limite_max, valeur, variable**
FINDB	**vecteur, limite_min, limite_max, valeur, variable**

Arguments

matrice	nom de la matrice. Matrice dans laquelle la recherche doit être exécutée
vecteur	nom du vecteur. Vecteur dans lequel la recherche doit être exécutée
colonne	constante ou variable entière ou nomcompteur. Numéro de colonne de la matrice dans laquelle la recherche doit être exécutée.
limite_min	constante ou variable. Indice minimal du vecteur ou de la matrice d'où la recherche doit commencer
limite_max	constante ou variable. Indice maximal du vecteur ou de la matrice où la recherche doit s'achever
valeur	constante ou variable. Valeur à rechercher
variable	variable. Résultat de la recherche

Description

Cette instruction exécute une recherche rapide d'une valeur à l'intérieur d'un **vecteur** ou d'une **colonne** de la matrice et met l'indice de l'élément en **variable**. Pour que la recherche soit réussie, il est nécessaire que le **vecteur** ou la **colonne** de la **matrice** aient été ordonnés précédemment avec l'instruction SORT selon un système croissant.
Si la valeur n'a pas été trouvée, **variable** contiendra la valeur e -1.

LASTELEM

Syntaxe

LASTELEM	**vecteur, elementsvet**
LASTELEM	**matrice, lignesmat**

Arguments

matrice	nom de la matrice
vecteur	nom de vecteur
elementsvect	variable. Nombre d'éléments du vecteur
lignesmat	variable. Nombre de lignes de la matrice

Description

Cette instruction écrit dans la variable **elementsvect** le nombre d'éléments du **vecteur** ou dans la variable **lignesmat** le nombre de lignes de la **matrice**.

LOCAL

Syntaxe

LOCAL	**nomvar AS type**
LOCAL	**vecteur[n. éléments] AS type**
LOCAL	**matrice[n. lignes] AS type, type, type, etc.**
LOCAL	**matrice[n. lignes] AS type:nomcol1, type: nomcol2, type:nomcol3, etc.**

Arguments

nomvar	nom de variable
[n. éléments]	variable ou constante (argument obligatoire). Nombre d'éléments du vecteur
[n. lignes]	constante ou variable (argument obligatoire). Nombre de lignes de la matrice
type	char, integer (32 bits), float (32 bits), double (64 bits), string, timer
nomcol1...nomcolN	nom de colonne, étiquette

Description

Déclaration d'une variable locale. Avant cette instruction, il se peut qu'apparaisse seulement l'instruction PARAM qui définit les paramètres de la fonction.
Pour toute information complémentaire sur les variables locales, voir Les variables locales.

MOVEMAT

Syntaxe

MOVEMAT	**nommatsour, nommatdest**
MOVEMAT	**nommatsour[lignesour], nommatdest[lignedest]**
MOVEMAT	**nommatsour[lignesour], nommatdest[lignedest], nombrelignes**

Arguments

nommatsour	nom de matrice source
lignesour	nombre de lignes de départ pour la copie de la matrice source (argument obligatoire)
nommatdest	nom de matrice destinataire
lignedest	nombre de lignes de départ pour la copie dans la matrice destinataire (argument obligatoire)
nombrelignes	nombre de lignes à copier

Description

Copie du contenu d'une matrice entière **nommatsour** dauns une autre matrice **nommatdest** une ou plusieurs lignes **[nombrelignes]** de la ligne de la matrice **nommatsour[lignesour]** dans la ligne de la matrice **nommatdest[lignedest]**.Si la paramètre **nombrelignes** n'est pas spécifié, une seule ligne est copiée. Les deux matrices doivent avoir le même type de structure (nombre de colonnes et type de

donnée de chaque colonne) et dans le cas de matrices intières aussi le même nombre de lignes. Il est possible de déplacer les lignes de données à l'intérieur de la même matrice.

Exemple

```
Movemat    Mx1, Mx2          ; copie la matrice Mx1 dans la Mx2

; copie la ligne 10 de la matrice Mx1 dans la ligne 3 de Mx2
Movemat    Mx1[10], Mx2[3]

; copie la ligne 1 de la matrice dans la ligne 7 de Mx1
Movemat    Mx1[1], Mx1[7]
; copie 6 lignes à partir de la ligne 2 de la matrice Mx1 dans la
matrice
; Mx2 à partir de la ligne 8
Movemat    Mx1[2], Mx2[8],6

; copie 4 lignes à partir de la ligne 2
; de la matrice Mx1 dans la même matrice
; Mx1 à partir de la ligne 10
Movemat  Mx1[2], Mx1[10],4
```

PARAM

Syntaxe

[PARAM]	**nomvar AS type**
[PARAM]	**vecteur[n° éléments] AS type**
[PARAM]	**matrice[n° lignes] AS type, type, type, etc**
[PARAM]	**matrice[n° lignes] AS type:alias, type:alias, type:alias, etc**

Arguments

nomvar	nom de variable
[n. éléments]	constante (argument obligatoire)
[n. lignes]	constante (argument obligatoire)
type	char, integer (32 bits), float (32 bits), double (64 bits), string

Description

Les paramètres agissent comme les variables locales (voir LOCAL), mais ils sont initialisés par qui appelle la fonction. La syntaxe de la déclaration des paramètres est la même que celle qui est utilisée pour les variables locales. Les paramètres sont passés par valeur ou par référence en fonction de leur type. Voir "Les fonctions".
Ils sont déclarés avant toute autre instruction.
Pour toute information complémentaire, voir Les variables locales.

SETVAL

Syntaxe

SETVAL	**valeur, nomvar**

Arguments

valeur	constante, variable ou nomdevice
nomvar	variable ou nomdevice

Description

Cette instruction attribue la **valeur** indiquée à la variable **nomvar** ou au énième élément d'un vecteur ou d'une matrice.

SORT

Syntaxe

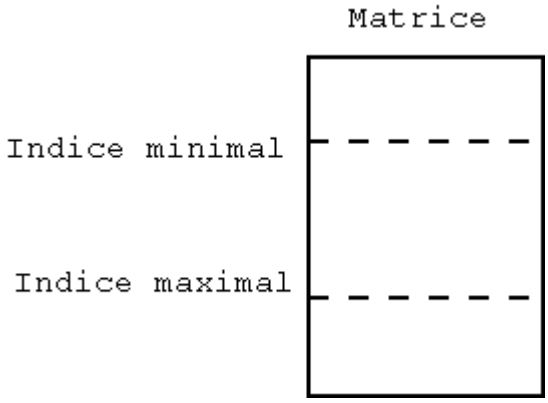
SORT	**matrice, colonne [, ordre], limite_min, limite_max**
SORT	**vecteur [,ordre], limite_min, limite_max**

Arguments

matrice	nom de matrice
vecteur	nom de vecteur
colonne	constante ou variable. Numéro de la colonne de la matrice
	ordre constante prédéfinie. Indique la modalité de classement
	Les valeurs admises sont les suivantes :
	UP ordination croissante
	DOWN ordination décroissante
limite_min	constante ou variable. Indice minimal du vecteur ou de la matrice d'où la recherche doit commencer
limite_max	constante ou variable. Indice maximal du vecteur ou de la matrice où la recherche doit s'achever

Description

Cette instruction effectue un classement des valeurs présentes à l'intérieur d'un **vecteur** ou d'une **matrice**, selon un sens défini par la constante **ordre**.
Dans le cas d'une matrice, le classement des lignes est dicté par la disposition croissante (UP) ou décroissante (DOWN) des valeurs contenues dans la **colonne** sélectionnée.
Si l'argument **ordre** est omis, le mode UP est pris automatiquement.



10.3.8 Chaînes

ADDSTRING

Syntaxe

ADDSTRING **nomchaîne1, nomchaîne2, nomchaîne3**

Arguments

nomchaîne1	constante chaîne ou variable chaîne. Chaîne source
nomchaîne2	constante chaîne ou variable chaîne. Chaîne à ajouter
nomchaîne3	variable chaîne. Chaîne résultat

Description

Enchaînement de deux chaînes.
Cette instruction ajoute la chaîne identifiée par **nomchaîne2** à la chaîne identifiée par **nomchaîne1** et met le résultat dans la chaîne identifiée par **nomchaîne3**.
La dimension maximale d'une chaîne est de 255 caractères+ le terminateur, ce qui implique que le résultat de l'enchaînement des deux premières chaînes ne devra pas dépasser cette limite.

Exemple

Opérations sur les chaînes

CONTROLCHAR

Syntaxe

CONTROLCHAR **valeur, nomchaîne**

Arguments

valeur	constante char ou integer ou variable char ou integer. Valeur à convertir
nomchaîne	variable chaîne. Chaîne résultat

Description

Cette instruction convertit la valeur identifiée par **valeur** en caractères ASCII et met le résultat dans la chaîne **nomchaîne** (c'est-à-dire dans le premier byte).
Ce que la chaîne contenait précédemment est perdu. Cette instruction est utile si l'on doit insérer des caractères de contrôle ou non imprimables (par exemple le caractère NULL = 0x00) dans une chaîne.
Il accepte chaîne de au moins 2 caractères: 1 caractère + le terminateur. Si la chaîne est de seulement caractères array[1] as char il est signalé un erreur de système "Argument macro erroné".

Exemple

Opérations sur les chaînes

LEFT

Syntaxe

LEFT	**nomchaînesour, numcaractères, nomchaînedest**

Arguments

nomchaînesour	constante chaîne ou variable chaîne. Chaîne source
nombcaractères	constante ou variable. Nombre de caractères à copier
nomchaînedest	variable chaîne. Chaîne destination

Description

Copie des premiers **nombcaractères** de la chaîne **nomchaînesour** dans la chaîne **nomchaînedest**.
Dans le concret, la partie gauche de la chaîne source est prélevée. Voir également les instructions MID et RIGHT.

Exemple

Opérations sur les chaînes

LEN

Syntaxe

LEN	**nomchaîne, variable**

Arguments

nomchaîne	variable chaîne. Chaîne
variable	variable

Description

Cette instruction calcule le nombre de caractères contenus dans la chaîne **nomchaîne** (à l'exception du terminateur) et le met dans **variable**.

Exemple

Opérations sur les chaînes

MID

Syntaxe

MID	**nomchaînesour, premiercar [, numcaractères], nomchaînedest**

Arguments

nomchaînesour	constante chaîne ou variable chaîne. Chaîne source
nombcaractères	constante ou variable. Nombre de caractères à copier
nomchaînedest	variable chaîne. Chaîne destination
premiercar	constante ou variable. Position caractère départ copie

Description

Cette instruction ôte de la chaîne identifiée par **nomchaînesour** un nombre de caractères identifiés par **nombcaractères**, à partir de la position **premiercar**.
La sous-chaîne ôtée est mise dans la chaîne identifiée par **nomchaînedest**.
Si **nombcaractères** est omis, **chaînesour** est copié à partir de la position **premiercar** jusqu'à la fin. Dans le concret, la partie centrale de la chaîne source est prélevée. Voir également les instructions LEFT et RIGHT.

Exemple

Voir exemple Opérations sur les chaînes

RIGHT

Syntaxe

RIGHT	**nomchaînesour, numcaractères, nomchaînedest**

Arguments

nomchaînesour	constante chaîne ou variable chaîne. Chaîne source
numcaractères	constante ou variable. Numéro de caractères à copier
nomchaînedest	variable chaîne. Chaîne destination

Description

Copie des derniers **nombcaractères** de la chaîne **nomchaînesour** dans la chaîne **nomchaînedest**.
Dans le concret, la partie droite de la chaîne source est prélevée. Voir également les instructions LEFT et MID

Exemple

Opérations sur les chaînes

SEARCH

Syntaxe

SEARCH	**nomchaîne, caractères, variable**

Arguments

nomchaîne	variable chaîne.
caractères	constante char ou constante chaîne ou variable chaîne. Caractère ou chaîne à chercher
variable	variable

Description

Cette instruction recherche la position du caractère ASCII identifié par **caractère** (qui peut également être une chaîne) à l'intérieur de la chaîne **nomchaîne** et met l'indice du résultat dans la **variable**.
Si **caractère** n'a pas été trouvé, **variable** contiendra la valeur -1.

Exemple

Opérations sur les chaînes

SETSTRING

Syntaxe

SETSTRING	**"valeur", nomchaîne**

Arguments

valeur	constante chaîne ou variable chaîne (entre doubles guillemets)
nomchaîne	chaîne destination

Description

Copie d'une chaîne.
Cette instruction exécute une copie des caractères ASCII présents dans la chaîne identifiée par **"valeur"** dans la chaîne identifiée par **nomchaîne**.
Pour insérer des caractères non imprimables dans une chaîne, voir l'instruction CONTROLCHAR.

Exemple

Opérations sur les chaînes

STR

Syntaxe

STR **valeur, nomchaîne**

Arguments

valeur constante ou variable. Valeur source à convertir
nomchaîne variable chaîne. Chaîne destination

Description

Cette instruction convertit la **valeur** en caractères ASCII et met le résultat dans la chaîne **nomchaîne**. Elle peut être utilisée pour transformer une variable type integer en une chaîne. Par exemple, le nombre 10 est transformé en chaîne "10".

Exemple

Opérations sur les chaînes

VAL

Syntaxe

VAL **nomchaîne, résultat**

Arguments

nomchaîne variable chaîne. Chaîne à convertir
résultat variable. Chaîne transformée

Description

Cette instruction transforme le contenu de la chaîne **nomchaîne** en un nombre décimal et met le résultat dans la **variable.**
Par exemple, la chaîne "123" devient la valeur 123.

Exemple

Opérations sur les chaînes

10.3.9 Communications

CLEARRECEIVE

Syntaxe

CLEARRECEIVE

Arguments

aucun argument

Description

Cette instruction vide la liste des RECEIVE exécutés mais pas encore satisfaits.

COMCLEARRXBUFFER

Syntaxe

COMCLEARRXBUFFER **numéroCOM**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**

Description

L'instruction vide le buffer de réception de la ligne sérielle **numéroCOM**. Toutes les données, éventuellement présentes sont éliminées.

COMCLOSE

Syntaxe

COMCLOSE **numéroCOM**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**

Description

Cette instruction ferme la ligne sérielle **NuméroCOM** ouverte par une **COMOPEN**. Il est nécessaire de fermer la ligne sérielle même lorsque la tâche qui a ouvert un port sériel est interrompue pour toute raison quelle qu'elle soit.

COMGETERROR

Syntaxe

COMGETERROR **numéroCOM, variable**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**

variable variable integer. Résultat de la dernière opération exécutée sur la sérielle

Description

L'instruction lit le code de retour de la dernière instruction de communication sérielle appelée sur le port **numéroCOM**. Cette instruction permet de savoir si une opération de lecture ou d'écriture est réussie et, en cas d'échec, de connaître le code d'erreur revenu.
Voici la liste des codes d'erreur :

Retour normal	0
Buffer de transmission plein	2
Dispositif déjà ouvert	3
Port non valable ou non configuré	6
Echec de l'activation du port I/0	7
Connexion à l'interrupt impossible	8
Port sériel (com) pas encore ouvert	9
Le dispositif sériel (com) est occupé	12
Connexion à RTX impossible	14

COMGETRXCOUNT

Syntaxe

COMGETRXCOUNT **numéroCOM, numchar**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**

numchar nombre de caractères présents dans le buffer

Description

L'instruction restitue le nombre de caractères présents dans le buffer de réception. Elle permet de savoir si le port sériel a reçu des caractères.

COMOPEN

Syntaxe

COMOPEN **numéroCOM, baudrate, wordsize,stopbits,parity**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**

baudrate	débit en bauds de communication. Les valeurs attribuables sont les suivantes : 2400, 4800, 9600, 19200, 38400, 57600, 115200
wordsize	dimension du mot de données. Les valeurs attribuables sont les suivantes : 5, 6, 7, 8.
stopbits	bits d'arrêt. Les valeurs attribuables sont les suivantes : 1, 2
parity	constante prédéfinie. Parité. Les valeurs qui peuvent être attribuées sont les suivantes : **NOPARITY**, **ODDPARITY** et **EVENPARITY**

Description

Cette instruction ouvre une ligne sérielle. Elle doit être exécutée avant toute autre instruction qui gère la ligne sérielle. Si une autre instruction relative à la même ligne sérielle est exécutée avant COMOPEN, une erreur de système est générée. Les paramètres passés doivent être des valeurs comprises dans celles qui sont indiquées ci-dessus.
Le canal de communication avec la ligne sérielle est associé à la tâche qui a exécuté l' instruction COMOPEN. Si la tâche finit, le canal de communication est fermé automatiquement.
Voir également COMCLOSE, COMREAD, COMWRITE, COMREADSTRING, COMWRITESTRING.

Note

Le nombre de lignes sérielles disponibles depend de l'environnement matériel du contrôle numérique (voir documentation). Dans l'environnement RTX uniquement COM1 et COM2 sont disponibles.

COMREAD

Syntaxe

COMREAD	**numéroCOM, buffer, nombcharàlire, nombcharlus [,timeout]**

Arguments

numéroCOM	constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes: de **COM1** à **COM8.**
buffer	vecteur de char. C'est le vecteur où sont déposées les données lues
nombcharàlire	nombre de caractères que l'on désire lire sur la ligne sérielle
nombcharlus	nombre de caractères réellement lus
timeout	timeout d'attente (en secondes)

Description

L'instruction lit des caractères sur la sérielle **NuméroCOM**. Les caractères lus sont enregistrés dans la variable **buffer**. Le champ **nombcharàlire** indique le nombre de caractères que l'instruction devra lire. S'il y a moins de caractères dans le buffer de réception de la sérielle et si le paramètre **timeout** n'est pas réglé, l'instruction s'achève immédiatement en indiquant le nombre de caractères réellement lus dans le paramètre **nombcharlus**. Si le paramètre **timeout** est réglé, l'instruction doit alors attendre au maximum le nombre de secondes indiqué dans la variable, en attendant que d'autres caractères arrivent. Si le **timeout** s'achève, l'instruction s'interrompt en indiquant toujours en **nombcharlus** le nombre de caractères réellement copié dans le **buffer.**

COMREADSTRING

Syntaxe

COMREADSTRING	**numéroCOM, buffer, nombcharlus [,terminateur [,timeout]]**

Arguments

numéroCOM	constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**
buffer	vecteur de char. C'est le vecteur où sont déposées les données.
nombcharlus	nombre de caractères réellement lus
terminateur	caractère de terminaison de transmission
timeout	timeout d'attente (en secondes)

Description

L'instruction lit des caractères sur la sérielle **NuméroCOM**. Elle diffère de la **COMREAD** dans la mesure où elle lit sur la sérielle jusqu'à ce qu'elle trouve le caractère terminateur. Les caractères lus sont enregistrés dans la variable **buffer**. Cette variable doit être type vecteur de char. Le champ **nombcharlus** indique le nombre de caractères que l'instruction a réellement lus sur la ligne sérielle et qu'elle a donc copiés dans le **buffer**. Le paramètre **terminateur** indique le caractère qui servira de terminateur de la transmission. Pratiquement, l'instruction doit lire les caractères sur la sérielle jusqu'à ce qu'elle trouve un caractère identique à celui qui est indiqué dans ce paramètre. Ce paramètre est facultatif. S'il est absent, le caractère de terminaison est considéré comme étant le zéro. Le caractère zéro ne sera pas copié dans le buffer passé en tant que paramètre, tandis qu'un éventuel caractère de terminaison specifié dans l'instruction sera copié dans le buffer. Le **timeout** est

un autre paramètre qui indique combien de secondes l'instruction doit attendre l'arrivée de nouveaux caractères, si elle a épuisé le buffer de réception sans avoir trouvé le caractère de terminaison. Si le paramètre **timeout** n'est pas réglé, l'instruction s'interrompt dès qu'elle a vidé le buffer de réception.

COMWRITE

Syntaxe

COMWRITE **numéroCOM, buffer, àécrire**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**
buffer vecteur de char. C'est le vecteur qui contient les données à écrire
àécrire nombre de caractères à écrire

Description

L'instruction écrit les caractères présents dans la variable **buffer** sur la ligne sérielle **numéroCOM**. Le paramètre **àécrire** indique le nombre de caractères réellement à écrire.

COMWRITESTRING

Syntaxe

COMWRITESTRING **numéroCOM, buffer [,terminateur]**

Arguments

numéroCOM constante prédéfinie. Numéro du port sériel. Les valeurs qui peuvent être attribuées sont les suivantes : de **COM1** à **COM8.**
buffer vecteur de char. C'est le vecteur qui contient les données à écrire.
terminateur caractère de terminaison de transmission

Description

L'instruction écrit les caractères présents dans la variable buffer sur la ligne sérielle **numéroCOM**. Elle diffère de la COMWRITE dans la mesure où elle écrit sur la sérielle jusqu'à ce qu'elle trouve le caractère **terminateur**. Le paramètre **terminateur** est facultatif. S'il n'est pas réglé, l'instruction transmet jusqu'à ce qu'elle trouve un caractère de zéro. Le zéro ne sera pas transmis, tandis qu'un éventuel caractère de contrôle indiqué le sera.

RECEIVE

Syntaxe

RECEIVE **[source,] identificateur, flags [, conteneur]**

Arguments

source constante type chaîne
identificateur constante type chaîne
flags constante type integer
conteneur nom de dispositif ou variable (numérique ou chaîne)

Description

Avec la SEND, cette instruction est utilisée pour l'échange d'informations entre les modules de l'installation et l'ordinateur superviseur. L'instruction SEND est utilisée pour envoyer des informations, tandis que la RECEIVE les reçoit. Les informations peuvent être demandées à Albatros ou à un programme externe (Server OLE Automation). Dans le deuxième cas, la demande sera toutefois reçue par Albatros qui pourvoira à l'envoyer, à son tour, au programme externe.

Le paramètre **source** est une chaîne qui permet de préciser à qui est adressée la demande d'information. Il existe trois catégories de sources :

- sources qui commencent par le caractère "@" (voir la liste donnée par la suite). La source est, en réalité, Albatros ou mieux, l'une de ses fonctions.
- Sources qui ne commencent pas par le caractère "@". Elles sont considérées en tant que Serveur OLE, à la réception de la première demande d'information qui leur est adressée, Albatros tâchera de les mettre en exécution, puis de leur passer la demande d'information transmise par le module.
- Source non indiquée (en effet, il s'agit d'un paramètre facultatif). Dans ce cas, l'information est lue dans un tableau conservé par Albatros. Si l'information n'est pas présente dans le tableau, la demande reste en suspens et elle sera exécutée dès que l'information sera disponible (fournie par un autre module ou par un programme externe).

Le paramètre **identificateur** est le nom de l'information demandée. Il ne peut pas être omis et il prend plusieurs significations en fonction de la source :
- si la source est Albatros, ce sera une commande liée à la fonction à laquelle l'on est en train d'accéder ;
- si la source est un Serveur OLE, ce sera une propriété de l'objet OLE demandé ;
- si la source n'est pas indiquée, ce sera l'étiquette qui identifie l'information à l'intérieur du tableau conservé par Albatros.

Le paramètre **flags** permet de préciser comment l'information demandée doit être traitée par Albatros. Les valeurs admises et les effets qu'elles impliquent sont les suivants :

valeur	***commande***	***description***
$0008H	CancelAfter	L'information sera éliminée après avoir été lue.
$0800H	UpdateFlags	Modifie l'état de l'information (lue / à lire) sans modifier les données.
$8000H	Delete	Elimine l'information

Enfin, le paramètre **conteneur** est la variable (ou le dispositif) dans lequel sera enregistrée l'information que l'on est en train de demander. Il peut être omis et, dans ce cas, on demande la notification d'un événement (il peut être utilisé pour synchroniser l'exécution du code GPL sur des modules différents).

Liste des **sources** gérées par Albatros et des commandes relatives :

"@List"
permet de gérer les commandes Simulation et Mise à zero. Les commandes suivantes (paramètre **identificateur**) sont permises :
- Sim,0,conteneur: demande l'été du bouton Simulé, qui est écrit sur le flagSwitch Simulé. La variable de retour **conteneur** vaut 1 si on n'a pas trouvé d'erreurs, autrement vaut 0.
- Setp,0,conteneur: demande l'été du bouton Setpoint, qui est écrit sur le flagSwitch CmdSetP. La variable de retour **conteneur** vaut 1 si on n'a pas trouvé d'erreurs, autrement vaut 0.
- Esc,0, conteneur: demande l'état du bouton Setpoint, qui est écrit sur le flagSwitch Escluso. La variable de retour **conteneur** vaut 1 si on n'a pas trouvé d'erreurs, autrement il vaut 0.

"@Environ"
Elle permet d'obtenir des informations sur l'état du système : niveau d'accès de l'utilisateur, modules raccordés au superviseur etc. L'information demandée est enregistrée dans le paramètre **conteneur**. Les valeurs admises pour le paramètre **identificateur** et les réponses correspondantes sont les suivantes :
- "Access niveau d'accès au système 0=utilisateur, 1=maintenance, 2=constructeur, 3=tpa Level"
- "MaskC masque des modules configurés onfModu les"
- "MaskAcmasque des modules raccordés tiveMod ules"
- "Currentmodule d'où provient la demande Module"
- "*mod*:N nom de l'ordinateur correspondant au module "*mod*". (*mod* compris entre 0 et 15) amePC"
- "LocalD date et heure du PC dans le format AAAA/MM/JJ HH:MM:SS ateTime "
- Le paramètre **conteneur** recevra la date et l'heure du PC selon un format lié à son type :
- char : numéro du jour de la semaine
- integer: numéro de secondes à partir du 1/1/1970
- float: numéro de jours et fractions de jours à partir du 1/1/1900
- double : numéro de jours et fractions de jour à partir du 1/1/1900
- string: texte "AAAA/MM/JJ hh:mm:ss"

Les masques des modules raccordés et configurés sont des masques de bit. Le bit de poids le plus bas correspond au module 0. Le bit de chaque module sera 1 si le module est raccordé ou configuré. En cas de "NamePC", le numéro du module est facultatif ; s'il est omis, le numéro qui est pris est celui du module d'où provient la demande.

"@Syn"

Communication entre GPL et l'afficheur des tableaux synoptiques. Elle assure l'ouverture ou la fermeture de tableaux synoptiques commandée par GPL et la demande d'informations d'une case d'un synoptique. Les commandes admises sont les suivantes (paramètre **identificateur**) :

- "Open:*nomfichier*" ouverture du synoptique *nomfichier.xsyn*
- "Close:*nomfichier*. fermeture du synoptique *nomfichier.xsyn*
- "*nomcase*" case dans laquelle l'information demandée est lue.

Il est possible d'avoir des informations sur la fenêtre de déplacement des axes selon les spécifications définies aussi par le paramètre **source** "@Devices reporté ci-dessous.

"@FileName"

Enregistre un association d'une une chaîne constante avec un nom de fichier qui se peut composer avec des variables chaîne. Dès que Albatros reçoit la communication de l' association il replace tous les noms suivants de fichier avec le nom reçu par cette instruction. Le paramètre **identificateur** est le nom du fichier. Le nom du fichier est la variable châine. Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué, Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. La valeur du paramètre identificateur est enregistré en tpa.ini dans la section [GPLFileName] à l'entrée Log, de façon que l'on peut réutiliser même dans les exécutions de Albatros. Pour supprimer l'association on doit configurer une chaîne vide comme paramètre identificateur. L'association qui a été définie de cette façon est valable pour chaque module.

"@FileDelete"

Elimination d'un fichier. Le paramètre **identificateur** est le nom du fichier qui sera éliminé (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. Le nom du fichier peut être défini selon les règles décrites en correspondance du paramètre **source** @FileRead**.** Le paramètre **conteneur** contiendra la valeur :

- 1, si le fichier a été éliminé,
- 0 dans le cas contraire.

"@FileRead"

Lecture du contenu d'un fichier. Le paramètre **identificateur** est le nom du fichier qui sera lu (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué, Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. Le nom du fichier peut être construit pendant l'exécution de l'instruction. Si l'identificateur commence et finit avec un caractère de % la séquence à l'intérieur est cherchée en tpa.ini dans la section [tpa] et utilisée comme nom du fichier. A l'intérieur du nom, il est possible d'insérer des caractères qui seront remplacés pendant l'exécution de l'instruction :

- %n numéro du module qui effectue l'instruction RECEIVE
- %h heure courante (format 00-23)
- %d jour courant (format 01-31)
- %m mois courant (format 01-12)
- %y année courante (format à quatre chiffre)

Si le paramètre **conteneur** est défini comme variable char il contiendra byte lu par le fichier, s'il est défini comme séquence, il contiendra une ligne entière de fichier texte, s'il est défini comme integer il contiendra le nombre de byte qui manquent à la fin du fichier (0 = fin du fichier).

Pour placer le pointeur de fichier au début du fichier, le paramètre **conteneur** doit être omis.

"@FileExist"

Contrôle de l'existence d'un fichier. Le paramètre **identificateur** est le nom du fichier qui sera lu (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. Le nom du fichier peut être défini selon les règles décrites en correspondance du paramètre **source** @FileRead. Le paramètre **conteneur** contiendra la valeur :

- autre que 0 si le fichier existe
- 0 si le fichier n'existe pas.

"@FileLastWrite"

Obtient la date de la dernière modification d'un fichier. Le paramètre **identificateur** est le nom du fichier (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué, Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] à l'entrée *dirreport*. Le nom du fichier peut être défini selon les règles décrites en correspondance du paramètre **source** @FileRead. Le

paramètre **conteneur** contiendra la valeur de la dernière modification du fichier selon un format lié au type de paramètre :
- char : numéro du jour de la semaine
- integer : numéro de secondes à partir du 1.er janvier 1970
- float : numéro de jours et fractions de jour à partir du 1.er janvier 1900
- double : numéro de jours et fractions de jour à partir du 1.er janvier 1900
- string : texte en format "AAAA/MM/GG hh:mm:ss"

"@FileInfo"

Lit des information d'un fichier. Le paramètre **identificateur** doit être exprimé dans la forme "propriété:nomfichier", ou **propriété** indique le nom de la propriété à lire et **nomefichier** est le nom du fichier. Le nom du fichier peut être défini au moyen de "Name" ("Nom"). Le paramètre **conteneur** contiendra les données lues par le fichier.
Liste des propriétés :
- "version:": renvoie un type de donnée integer dans un conteneur. Les quatre numéros identifiant la version se trouvent dans le 4 octets de la variable conteneur. Si une erreur se produit, la valeur de la variable conteneur est 0.
- "size:": renvoie un type de donnée integer ou float ou double dans un conteneur. La donnée est la dimension du fichier. Si une erreur se produit, la valeur de la variable conteneur est -1.

"@Devices"

Demande d'ouverture ou de fermeture de la fenêtre Diagnostic relative au module qui envoie l'information. Le paramètre identificateur peut prendre les valeurs suivantes:
- "Open" ouverture du Diagnostique,
- "Close" fermeture du Diagnostique.

Pour interagir avec la fenêtre de mouvement, le paramètre **identificateur** peut prendre les valeurs suivantes:

"MoveAX#nome_asse #HasFocus"	ile paramètre **conteneur** contiendra 1, si la fenêtre de mouvement d'axe spécifiée est active, sinon il contiendra 0.
"MoveAX#nome_asse #Jog"	paramètre **conteneur** contiendra 1, si le mouvement pour déplacements gérés runtime par l'opérateur est réglé, sinon il contiendra 0.l
"MoveAX#nome_asse #Step"	le paramètre **conteneur** contiendra 1, si le mouvement pour déplacements de pas prédéfini est réglé, sinon il contiendra 0.
"MoveAX#nome_asse #Absolute"	le paramètre **conteneur** contiendra 1, si le mouvement avec déplacements à cote établie est réglé, sinon il contiendra 0.

où nom_axe représente le nom de l'axe affiché dans la fenêtre. Par exemple si l'on souhaite vérifier si la fenêtre de mouvement axe X est active, le paramètre **identificateur** sera "@MoveAX#X#HasFocus". Le nom de l'axe peut être dans l'une des formes suivantes :
1. Nom_Groupe.Nom_Sous-groupe.Nom_Axe ou Nom_Groupe.Nom_Axe : le parcours complet de l'axe est fournit.
2. Nom_Axe : pour identifier l'axe correct, il faut que les contrôles suivants soient faits dans l'ordre :
   - si la tâche d'où arrive la commande est une fonction de sous-groupe, l'axe est cherché dans ce sous-groupe.
   - si la tâche d'où arrive la commande est une fonction du sous-groupe principal, l'axe est cherché dans tout le groupe. S'il y a plus d'un axe avec ce nom, la recherche échoue.
   - si les contrôles précédents ont échoué, l'axe est recherché dans tous les groupes du module. S'il y a plus d'un axe avec le Nom_Axe la recherche n'a pas de résultat positif.

"@Vars"

Demande d'actualisation d'une variable globale GPL. Elle permet d'effectuer un rafraîchissement des données des Paramétrages technologiques et des outils. Normalement, les données de paramétrage sont envoyées au GPL pendant l'initialisation de la machine. Le paramètre **identificateur** prend la signification du nom de la variable globale (de machine ou de groupe) qui a demandé l'actualisation. Le paramètre **conteneur** contiendra la valeur :
- 1 si la variable a été actualisée correctement,
- 0 dans le cas contraire.

"@Application"

Interaction avec Albatros. Elle permet de visualiser des boîtes de messages sur l'écran et de fermer Albatros. Les valeurs admises pour le paramètre **identificateur** sont les suivantes:

"Quit"	ferme Albatros
"IsLocked"	il vérifie si la sortie de Albatros est bloquée. Le paramètre **conteneur** contiendra 1, si l'interface est bloquée, 0 si il est possible de sortir de Albatros.

"MsgBox" ouvre une boîte de message.

Le paramètre **conteneur** permet de savoir, en cas de boîte de messages, sur quel bouton l'opérateur a appuyé :

- 1 bouton "OK"
- 2 bouton "Annuler"
- 4 bouton "Réessayer"
- 6 bouton "Oui"
- 7 bouton "Non"

En cas de commande "Quit", le paramètre **conteneur** contiendra la valeur :

- 1 si Albatros à été fermé correctement,
- 0 dans le cas contraire.

"@Param"

Il permet de connaître le numéro progressif d'enregistrement des fichiers de paramétrique Partec.xpar et Partool.xpar. L'information requise est enregistrée dans le paramètre **contenant**. Les valeurs admises pour le paramètre **identificateur** sont:

- "partec" requiert le progressif d'enregistrement de partec.xpar
- "partool" requiert le progressif d'enregistrement de partool.xpar

"@Ini"

lit un raccourci de clé=valeur du fichier tpa.ini. Le paramètre **identificateur** est le nom de la clé à lire dans le fichier tpa.ini à la section [Tpa]. Pour lire d'une section spécifique, au nom de la clé on doit ajouter le nom de la section entre crochets.("[Section]Clé").

"@ShellExecute"

Il demande au système d'exploitation d'ouvrir un fichier en utilisant le logiciel associé à l'extension du fichier. Il est possible aussi de lancer un exécutable. Le paramètre **identificateur** est le nome du fichier à ouvrir ou le nom du logiciel à lancer. Le nom du fichier peut être déclaré avec un nom du chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom du fichier est aussi cherché dans ceux qui sont définis par "@FileName". Le paramètre **conteneur** contiendra la valeur 0, si aucun erreur s'est produite dans l'ouverture du fichier, sinon il contiendra le code d'erreur.

"@StartProg"

Il exécute le programme défini dans le paramètre **identificateur**. Il n'est pas possible de transmettre des arguments au programme à lancer. Le nom du fichier doit contenir le chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom dulogiciel est aussi cherché dans ceux qui sont définis par "@FileName". Le paramètre **conteneur** contiendra la valeur 0, si aucun erreur s'est produite dans l'ouverture du fichier, sinon il contiendra le code d'erreur. Si le programme avait déjà été lancé, le code d'erreur est 1056.

"@ProgRunning"

Il vérifie si le logiciel lancé avec "@StartProg" est encore en exécution. Le nom du logiciel est définie dans le paramètre **identificateur**. Le nom du fichier doit contenir le chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom dulogiciel est aussi cherché dans ceux qui sont définis par "@FileName". Le paramètre **conteneur** contiendra la valeur 1, si le programme est encore en exécution, sinon il contiendra valeur 0.

"@TermProg"

Il termine le programme défini dans le paramètre **identificateur** et lancé avec "@StartProg" . Le nom du fichier doit contenir le chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom du programme est aussi cherché dans ceux qui sont définis par "@FileName". Le paramètre **conteneur** contiendra la valeur 0, si aucun erreur s'est produite dans l'ouverture du fichier, sinon il contiendra le code d'erreur. Si le programme avait déjà été lancé, le code d'erreur est 1056.

"@DialogFile"

Il ouvre la boite de dialogue de Fichier Ouvrir ou de Fichier Enregistrer pour permettre de choisir le nom du fichier. Pour ouvrir la fenêtre de Fichier Ouvrir, configurer **identificateur** = "Open", pour ouvrir la fenêtre de Fichier Enregistrer configurer le paramètre **identificateur** = "Enregistrer". Le nom du fichier sélectionné estenregistré dans le paramètre **conteneur**.

"@AxisCorrectors"

Il remplace la table des correcteurs de linéarité pour un axe par un tableau nouveau, chargée à partir des lignes, qui doit toujours avoir le même nombre de correcteurs et les mêmes axes pour les correcteurs croisés.

Le paramètre **identificateur** est le nom du fichier, qui a généralement l'extension .csv et se trouve dans le dossier ...\Mod.n\Config (le nom du fichier est "normalisé" comme par exemple "@FileExist"). Le paramètre **conteneur** est défini comme une variable integer et contiendra 1, si de nouveaux correcteurs ont été envoyés, sinon 0.

"@Language"

Reçoit un texte traduisible correspondant à un message de groupe, de bibliothèque ou de module, qui est associé à une instruction MESSAGE ou une instruction ERROR. Les valeurs autorisées pour le paramètre **identificateur** sont les suivantes :

- DEFMSG:numéro", où le nombre est une séquence de chiffres. Albatros écrit dans le **conteneur** le texte du formulaire de message numéro "numéro".
- "DEFMSG:name", où "name" est le nom, y compris le groupe et la bibliothèque, d'un DEFMSG. Albatros écrit dans un **conteneur** le texte du message indiqué. Si le nom du groupe ou de la bibliothèque est manquant, le nom de la tâche qui a envoyé le RECEIVE est utilisé.
- "DEFMSG:*", Albatros écrit dans le **conteneur** le texte du message du groupe ou du module préalablement indiqué par l'instruction SEND.

Exemple

```
; en GPL
RECEIVE "@Param", "partec", 0, prog
RECEIVE "@Param", "partool", 0, prog

; en GPL
; lit la valeur de la clé Radix dans la section [Albatros] du fichier
                          tpa.ini
RECEIVE "@INI", "[Albatros]Radix", 0, valeur

; ouvre la fenêtre du Fichier Ouvrir et mémorise le nom du fichier dans
                          la variable NomFichier
RECEIVE "@DialogFile", "Open", 0, NomFichier


; lecture complète d'un fichier
Function ReadProperties
 PARAM file       AS STRING
 LOCAL version AS INTEGER
 LOCAL size      AS DOUBLE

 SEND “@FileName” “theFile” 0 file
 WAITRECEIVE “@FileInfo”, “version:theFile”, 0,version
 WAITRECEIVE “@FileInfo”,”size:theFile”, 0, size
```

SEND

Syntaxe

SEND	**[destinataire,] identificateur, flags [, information]**

Arguments

destinataire	constante type chaîne
identificateur	constante type chaîne
flags	constante type integer
information	nom de dispositif ou variable (numérique ou chaîne)

Description

Avec RECEIVE, cette instruction est utilisée pour échanger des informations entre les modules de l'installation et l'ordinateur superviseur. L'instruction SEND est utilisée pour envoyer des informations, tandis que la RECEIVE les reçoit. Les informations peuvent être demandées à Albatros ou à un programme externe (Server OLE Automation). Dans le deuxième cas, la demande sera toutefois reçue par Albatros qui pourvoira à l'envoyer, à son tour, au programme externe.

Le paramètre **destinataire** est une chaîne qui permet de préciser à qui est adressée la demande d'information. Il existe trois catégories de destinataires :

- destinataires qui commencent par le caractère " @ " (voir la liste donnée par la suite). Le destinataire est, en réalité, Albatros ou mieux, l'une de ses fonctions.
- Destinataires qui ne commencent pas par le caractère " @ ". Ils sont considérés en tant que Serveur OLE, à la réception de la première information qui leur est adressée, Albatros tâchera

de les mettre en exécution, puis de lui passer la demande d'information transmise par le module.
- Destinataire non indiqué (en effet, il s'agit d'un paramètre facultatif). Dans ce cas, l'information est conservée par Albatros dans un tableau et elle reste à la disposition de qui en fera la demande (un autre module ou un programme externe).

Le paramètre **identificateur** est le nom de l'information. Il ne peut pas être omis et il prend plusieurs significations en fonction du destinataire :
- si le destinataire est Albatros, ce sera une commande liée à la fonction à laquelle on est en train d'accéder ;
- si le destinataire est un Serveur OLE, ce sera une propriété de l'objet OLE demandé ;
- si le destinataire n'est pas indiqué, ce sera l'étiquette qui identifie l'information à l'intérieur du tableau conservé par Albatros.

Le paramètre **flags** permet de préciser comment l'information doit être traitée par Albatros. Les valeurs admises et les effets qu'elles impliquent sont les suivants :

valeur	**commande**	**description**
$0001H	Broadcast	Envoi normal de l'information.
$0008H	CancelAfter	L'information sera éliminée après avoir été lue.
$0020H	ReadOnly	L'information ne peut être éliminée que par l'expéditeur.
$1000H	UpdateFlags	Modifie l'état de l'information (lue / à lire) sans modifier les données.
$8000H	Delete	Elimine l'information.

Enfin, le paramètre **information** est l'information que l'on est en train d'envoyer. Elle peut être omise. Dans ce cas, l'envoi d'une information vide prend la signification d'une notification d'un événement (il peut être utilisé pour synchroniser l'exécution du code GPL sur des modules différents). Le paramètre information admet tous les dispositifs (sauf les axes), les variables simples du GPL et les chaînes.

Liste des destinataires gérés par Albatros et des commandes relatives :

" @List "
permet de gérer les commandes Simulation et Mise à zero.
Ils sont admis les suivants commandes (paramètre **identificateur**) :
- Sim : notifie le changement d'état du flag switch Simulato. Sur la base de l'état du flag est visualisé pressé ou relâché le bouton qui l'identifie dans la barre d'Usinages (1=sélectionné, 0=désélectionné).
- Setp : notifie le changement d'état du flag switch CmdSetp. Sur la base de l'état du flag est visualisé pressé ou relâché le bouton qui l'identifie dans la barre d'Usinages (1=sélectionné, 0=désélectionné).
- Esc : notifie le changement d'état du flag switch "Escluso". Sur la base de l'état du flag est visualisé pressé ou relâché le bouton qui l'identifie (le même du flag switch CmdSetp) dans la barre des Usinages(1=sélectionné, 0=désélectionné).
- End : termine l'exécution de la liste. Cette commande abaisse les boutons de Start et Stop et déshabilite les options de menu Start et Stop.
- Hold: abaisse le bouton de Stop et habilite l'option de menu Stop. Il élève le bouton de Start et déshabilite l'option de menu Start.

"@Syn"
Communication entre GPL et l'afficheur des tableaux synoptiques. Elle assure l'ouverture ou la fermeture de tableaux synoptiques commandée par GPL et l'envoi d'informations dans une case d'un synoptique. Les commandes admises sont les suivantes (paramètre **identificateur**) :

" Open : ouverture du synoptique nomfichier.xsyn
nomfichier
"
" Close : fermeture du synoptique nomfichier.xsyn
nomfichier
"
" Open " ouverture d'un synoptique. Le nom du fichier est lu par la variable **information**
" Close " fermeture d'un synoptique. Le nom du fichier est lu par la variable **information**
"*nomcase* case où est affichée l'information envoyée
"

Il est possible 'avoir des informations sur la fenêtre de mouvement des axes selon les spécifications définies également par le paramètre **destinataire** "@Devices", reporté ci-dessous.

"@File"

Ecriture sur fichier. Cette fonction permet de créer des fichiers journal personnalisés qui enregistrent les opérations exécutées par une machine. Ce sont des fichiers de texte (ASCII).
Le paramètre **identificateur** est le nom du fichier sur lequel l'écriture sera effectuée.
Si dans le paramètre identificateur le chemin d' accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué, Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*.
Le nom du fichier peut être:
Si l'identificateur commence et finit avec un caractère de % la chaîne è l'intérieur est cherché en tpa.ini dans le secteur [tpa] et utilisé comme nom du fichier. A l'intérieur du nom on peut insérer des caractères qui seront remplacés pendant l'exécution de l'instruction :

- %n numéro du modulo qui exécute l'instruction SEND
- %h heure courante (format 00-23)
- %d jour courant (format 01-31)
- %m mois courant (format 01-12)
- %y année courante (format è quatre chiffres)

Voir l'exemple.
Les opérations d'écriture se font en mode Append (les données sont ajoutées à la fin du fichier).
Un fichier peut recevoir des données numériques (qui sont converties automatiquement en ASCII) ou des chaînes. Il est possible d'écrire des chaînes en format date et heure en utilisant les caractères du format %d pour la date et %t pour l'heure.
Pour l'heure on utilise le format "HH:mm:ss" (c'est à dire: heures, minutes et secondes séparés par " ") et pour la date un format en fonction des réglages de chaque nation. Il est possible d'utiliser un autre format en réglant en tpa.ini dans la section [Albatros] l'option "LogNoLocale=1" (le réglage par défaut est LogNoLocale=0, c'est à dire utilisation du format actuel). Il est possible aussi de régler le format à utiliser pour la date et l'heure, quelle que soit le format réglé dans Windows, en réglant toujours en tpa.ini dans la section [Albatros] les options "LogDateFormat=" e "LogTimeFormat=" et en assignant une chaine de caractères selon le tableau ci-dessous. Si ces options ne sont pas présentes ou si elles sont vides, utilisez les formats configurés par Windows.

Format de l'heure

h	heure au format 12 heures sans zéros initiaux
hh	heure au format 12 heures avec zéros initiaux
H	heure au format 24 heures sans zéro initiaux
HH	heure au format 24 heures avec zéro initiaux
m	minutes sans zéros initiaux
mm	minutes avec zéros initiaux
s	secondes sans zéros initiaux
ss	secondes avec zéros initiaux
t	un seul caractère pour indiquer le marqueur de temps, par exemple A ou P
tt	plusieurs de caractères pour indiquer le marqueur de temps, par exemple AM ou PM

Remarque : les formats "t" et "tt" utilisent le marqueur de temps indiqué dans le panneau de configuration de l'utilisateur actuel. Ils ne sont pas nécessairement "AM" e "PM".
Exemple : s'il est 11:29 de l'après-midi et la chaîne se compose de "hh':'mm':'ss tt", on va écrire"11:29:40 PM".

Format du jour

d	jour du mois sans zéros initiaux, affiché en chiffre
dd	jour du mois avec zéros initiaux, affiché en chiffre
ddd	jour de la semaine, affiché en caractères et réduit à trois lettres
dddd	jour de la semaine, affiché en caractères avec son nom complet
M	mois sans zéros initiaux, affiché en chiffres
MM	mois sans zéros initiaux, affiché en chiffres
MMM	mois, affiché en caractères et réduit à trois lettres
MMMM	mois, affiché en caractères avec son nom complet
y	an de deux chiffres sans zéros initiaux pour ans inférieurs a 10
yy	an de deux chiffres avec zéros initiaux pour ans inférieurs a 10
yyyy	an affiché avec quatre ou cinq chiffres selon le calendrier actuel
yyyyy	an affiché avec quatre ou cinq chiffres selon le calendrier actuel

Exemple : s'il est mercredi 31 Août 1994 et la chaine se compose de "ddd',' MMM dd yy", on va écrire "Mer, Août 31 94"
Si l'information est omise, un "retour à la ligne" est ajouté au fichier.

"@FileName"

Enregistre un association d'une chaîne constante avec un nom de fichier qui se peut composer avec des variables chaîne. Dès que Albatros reçoit la communication de l'association il replace tous les noms suivants de fichier avec le nom reçu par cette instruction. Le paramètre **identificateur** est le nom du fichier dans le quel on va écrire. Le nom du fichier est la variable châine. Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué, Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. La valeur du paramètre identificateur est enregistré en tpa.ini dans la section [GPLFileName] à l'entrée Log, de façon que l'on peut réutiliser même dans les exécutions de Albatros. Pour supprimer l'association on doit configurer une chaîne vide comme paramètre identificateur. L'association qui a été définie de cette façon est valable pour chaque module.

"@FileDelete "

Elimination d'un fichier. Le paramètre **identificateur** est le nom du fichier qui sera éliminé (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. Le nom du fichier peut être défini selon les règles décrites en correspondance du paramètre **destinataire** @File.

"@FileRead"

Situation du pointeur du fichier sur le début du fichier. Le paramètre **identificateur** est le nom du fichier (parcours complet). Si dans le paramètre identificateur le chemin d'accès complet, dans le quel on peut archiver le fichier, n'est pas indiqué Albatros considère comme chemin d'accès celui défini en tpa.ini dans la section [tpa] sous *dirreport*. Le nom du fichier peut être défini selon les règles décrites en correspondance du paramètre **destinataire** @FileRead.

"@Axis"

Interagit avec la fenêtre de mouvement des axes selon les spécifications définies aussi pour le paramètre destinataire "@Devices" ci-dessous. Si une fenêtre qui règle le mouvement de l'axe indiquée est déjà ouverte, la commande agit sur cette fenêtre, quand elle est ouverte dans un synoptique et quand elle est ouverte dans la diagnostique. Si la fenêtre est fermée, la commande cherche de l'ouvrir dans la diagnostique ou dans un des synoptiques déjà ouverts et qui contiennent celle axe-là.

"@Devices"

Demande d'ouverture ou de fermeture de la fenêtre Diagnostic relative au module qui envoie l'information. Exécution de commandes à l'intérieur de la fenêtre de mouvement axe en diagnostic. Le paramètre **identificateur** peut prendre les valeurs suivantes :

- " Open " ouverture du Diagnostic
- " Close " fermeture du Diagnostic

Le paramètre **identificateur,** lorsque vous voulez interagir avec la fenêtre de mouvement de l'axe, peut prendre les valeurs suivantes :

Pour interagir avec la fenêtre de mouvement axe, le paramètre **identificateur** peut prendre les valeurs suivantes:

- "MoveAX#nome_asse#Open" ouverture de la fenêtre de mouvement axe.
- "MoveAX#nome_asse#Close" fermeture de la fenêtre de mouvement axe.
- "MoveAX#nome_asse#Plus" pression de la touche de mouvement axe en direction positive
- "MoveAX#nome_asse#Minus" pression de la touche de mouvement axe en direction négative
- "MoveAX#nome_asse#Stop" pression de la touche d'arrêt du mouvement
- "MoveAX#nome_asse#Jog" règle la modalité de mouvement pour les déplacements gérés runtime par l'opérateur
- "MoveAX#nome_asse#Step" règle la modalité de mouvement pour les déplacements de pas prédéfini
- "MoveAX#nome_asse#Absolute" règle la modalité de mouvement avec déplacement à cote établie de l'axe

où nom_axe représente le nom de l'axe affiché dans la fenêtre. Par exemple si l'on souhaite ouvrir la fenêtre de mouvement axe X, le paramètre **identificateur** sera "@MoveAX#X#Open". Le nom de l'axe peut être dans l'une des formes suivantes:

1.Nom_Groupe.Nom_Sous-groupe.Nom_Axe ou Nom_Groupe.Nom_Axe: le parcours complet de l'axe est fournit.

2.Nom_Axe: pour identifier l'axe correct, il faut que les contrôles suivants soient faits dans l'ordre :

- si la tâche d'où arrive la commande est une fonction de sous-groupe, l'axe est cherché dans ce sous-groupe ;

- si la tâche d'où arrive la commande est une fonction du sous-groupe principal, l'axe est cherché dans tout le groupe. S'il y a plus d'un axe avec ce nom, la recherche échoue ;
- si les contrôles précédents ont échoué, l'axe est recherché dans tous les groupes du module. S'il y a plus d'un axe avec le Nom_Axe la recherche n'a pas de résultat positif.

Il est possible d'empêcher l'utilisateur d'agir sur les touches de mouvement axe de toutes les fenêtres de mouvement axe du module en Diagnostic, en configurant le paramètre **identificateur** de la façon suivante :

- "MoveAX##UIENABLE", si le paramètre **information** est configuré sur 0, le mouvement d'axes de Albatros est désactivé ; s'il est configuré sur 1e le mouvement d'axes de Albatros est activé.

La désactivation du mouvement d'axes de Albatros est conseillée quand les axes sont déplacés à partir du clavier de la machine.

"@Vars"
Demande l'enregistrement du contenu d'une variable globale GPL dans l'archive des paramètres technologiques ou outils. Le paramètre **identificateur** est le nom de la variable globale (de machine ou de groupe ou de librairie) pour lequel on demande la mise à jour.

"@Application"
Interaction avec Albatros. Elle permet de fermer Albatros ou d'afficher des "boîtes de message" dans l'écran afin d'informer l'utilisateur ou de lui demander son consentement pour des activités ultérieures. Les valeurs admises pour le paramètre **identificateur** sont les suivantes :
"Quit" ferme Albatros
"Lock" interdit la fermeture de Albatros de ***Fichier->Quitter*** ou des raccourcis clavier [ALT+F4] ou de bouton de fermeture.
"UnLock"rétablit la possibilité de fermer Albatros.
"MsgBox:*flags*" ouvre un message box.
Le comportement des boîtes de messages est contrôlé par la partie "*flags*" de la chaîne **identificateur**. Cette dernière peut être une séquence des caractères suivants (ils peuvent être soit en majuscules, soit en minuscules) :

"O"	bouton "OK"
"C"	bouton "Annuler"
"Y"	bouton "Oui"
"N"	bouton "Non"
"R"	bouton "Réessayer"
"S"	icône de l'affiche Stop
"?"	icône icône informative, composé d'un "i" minuscule à l'intérieur d'un cercle
"!"	icône avec point d'exclamation
"*"	icône d'information
"1"	le premier bouton est celui qui est indiqué par défaut
"2"	le deuxième bouton est celui qui est indiqué par défaut
"3"	le troisième bouton est celui qui est indiqué par défaut
"4"	le quatrième bouton est celui par défaut

S'il n'est pas indiqué, le bouton par défaut est le premier
Par exemple, "MsgBox:?YN2" définit une boîte de messages avec l'icône informative, deux boutons "Oui" et "Non" dont le deuxième est celui qui est indiqué par défaut. Le paramètre **information** peut être une chaîne contenant le texte à visualiser ou un compteur qui est interprété comme le code d'un message de module géré par TpaLangs.exe, ou l'étiquette d'un message de groupe défini avec l'instruction DEFMSG.
Quant au texte, s'il contient un caractère de retour chariot, "\u000A", il sera divisé en deux et la première partie sera affichée comme le texte de la boîte de message, tandis que la deuxième partie sera affichée comme une explication, ou un détail, du texte.
La langue dans laquelle les boutons sont affichés est la langue de Windows.

"@Help"
Ouverture d'un fichier d'aide. Cette fonction permet de commander l'affichage d'un fichier d'aide en précisant l'argument à visualiser. Les valeurs admises pour le paramètre **identificateur** sont les suivantes :

- "Open:*nomfichier*" ouverture d'un fichier d'aide,
- "Close:*nomfichier*" fermeture d'un fichier d'aide.

La partie "*nomfichier*" de la chaîne indique le nom du fichier d'aide à ouvrir.
Le paramètre **information** peut être une chaîne ou un nombre et il prend respectivement la signification de clé ou de nombre de contexte (servant à identifier la page ou l'argument de l'aide que l'on veut visualiser).

@Report"

Ajoute signalisations au fichier de report de Albatros (MONTH(n. mois).TER). Le paramètre **identificateur** est :

- "Add"

Le paramètre **information** peut être:

- une variable chaîne ou une constante chaîne : dans ce cas il est mémorisé le texte contenu dans la chaîne
- une variable integer ou une valeur numérique integer : dans ce cas il est mémorisé le texte défini dans l'instruction DEFMSG.

"@Ini"
écrit un raccourci de clé=valeur dans le fichier tpa.ini. Le paramètre **identificateur** est le nom de la clé à ajouter dans le fichier tpa.ini dans la section [Tpa]. Pour écrire une section spécifique, au nom de la clé on doit ajouter le nom de la section entre crochets ("[Section]Clé").
Le paramètre **information** peut être une variable chaîne ou numérique, une constante chaîne ou numérique.

"@ShellExecute"
Il demande au système d'exploitation d'ouvrir un fichier, en utilisant le programme associé à l'extension du fichier. Il est possible aussi de lancer un exécutable. Le paramètre **identificateur** est le nome du fichier à ouvrir ou le nom du programme à lancer. Le nom du fichier peut être déclaré avec un nom de chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom du fichier est aussi cherché dans ceux qui sont définis par "@FileName".

"@StartProg"
Il exécute le programme défini dans le paramètre **identificateur**. Il n'est pas possible de transmettre des arguments au programme à lancer. Le nom du fichier doit contenir le chemin d'accès complet, sinon il est cherche dans le dossier courant de Albatros. Le nom du programme est aussi cherché dans ceux qui sont définis par "@FileName".

"@TermProg"
Il termine le programme défini dans le paramètre **identificateur** et lancé avec "@StartProg". Le nom du fichier doit contenir le chemin d'accès complet, sinon il est cherché dans le dossier courant de Albatros. Le nom du programme est aussi cherché dans ceux qui sont définis par "@FileName".

"@DialogFile"
Permet de configurer des paramètres concernant la boîte de dialogue du fichier Ouvrir ou du fichier Enregistrer.
Les valeurs admises pour le paramètre **identificateur** sont les suivantes :

"Extension"	si l'utilisateur n'insère pas une extension, l'extension définie dans le paramètre **information** (variable ou constante chaîne)
"Filter"	configure le filtre dans les types de fichier à utiliser. Le paramètre **information** peut être une variable chaîne ou une constante chaîne et dans ce cas on utilise comme filtre le texte contenu dans la chaine, une variable "integer" ou une valeur numérique "integer" et dans ce cas on utilise comme filtre le texte défini dans l'instruction DEFMSG.
"Flags"	configure le flag d'initialisation. Pour la liste des valeurs à configurer dans le champ **information** (variable ou constante integer) veuillez faire référence à la documentation officielle Microsoft concernant le membre Flags de la structure OPENFILENAME
"InitalDir"	configure le dossier de début, défini dans le champ **information** (variable ou constante chaîne)
"Title"	configure le titre de la fenêtre. Le paramètre **information** peut être une variable chaîne ou une constante chaîne et dans ce cas on utilise comme titre le texte contenu dans la chaine, une variable "integer" ou une valeur numérique "integer" et dans ce cas on utilise comme titre le texte défini dans l'instruction DEFMSG.

"@Language"
Définit le numéro de message du groupe, du module ou de la bibliothèque qui sera utilisé dans le prochain RECEIVE avec le même identifiant. La valeur autorisée pour le paramètre **identificateur** est "DEFMSG:*". Le paramètre **information** peut être une variable entière ou une constante entière et, dans ce cas, il définit le numéro du message de groupe à afficher. Il peut s'agir d'un type de chaîne ou d'une constante de chaîne et, dans ce cas, il définit le nom du DEFMSG.

Exemple

```
; Exemple d'instruction send file avec nom construit en exécution.
; Suppose que la date dans laquelle est exécuté l'instruction
; soit 31-01-2000.

; en GPL
SEND "@File", "%Log%", 0, "Début exécution"
; ajoute un "retour chariot"
; dans le fichier tpa.ini à la section [TPA] on ajoute
SEND "@File", "%Log%", 0
Log=c:\Albatros\report\%y\Rep%m%d.txt

; Le nom du fichier qui suit est:
c:\Albatros\report\2000\Rep0131.txt

; Exemple d'instruction send Vars
; on définit une variable Var_SendVars
; as double dans le fichier de variables globales
; en paramètres Technologiques on insère Var_SendVars
; dans le champ Nom Matrice
; en GPL
SETVAL  100.0,Var_SendVars
; envoie la valeur 100.0 au paramètres Technologiques
; associé à la variable Var_SendVars
SEND "@Vars", "Var_SendVars", 0

; Exemple d'instruction send INI
; on écrit en tpa.ini la clé Radix dans la section [Albatros]
; pour configurer una base numérique d'affichage décimal
SEND "@INI", "[Albatros]Radix", 0;1


; Exemple de configuration d'une association de chaîne constante GPL
; avec le nom d'un fichier.

; déclaration d' une variable chaîne
nomfichier as string
; composition du nom du fichier
setstring "C:\albatros\report\LogFile.txt",nomfichier
; association
SEND "@FileName", "LOG",0,nomfichier
; toutes les opérations d'écriture de ce moment sont
; exécutées dans le fichier défini par la variable nomfichier
 SEND "@File", "LOG",0,"Écriture dans le fichier LOG"
```

SENDIPC

Syntaxe

SENDIPC	**nomIPC, attente [, nomvar1 [, nomvarN, ...]]**
SENDIPC	**nomIPC, attente , matrice[ligne]**
SENDIPC	**nomIPC, attente , vecteur**
SENDIPC	**nomIPC, attente , matrice**

Arguments

nomIPC	constante chaîne. Nom de la IPC
attente	constante prédéfinie. Mode d'attente de lecture de la commande. Les valeurs admises sont les suivantes : **WAIT** attend la lecture de la commande **NOWAIT** n'attend pas la lecture de la commande
nomvar1[...nomvarN]	constante ou variable. Noms variables 1÷N
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice
vecteur	Nom de vecteur
matrice	Nom de matrice

Description

Cette instruction envoie une commande IPC à la mémoire partagée **"nomIPC"**.

La première fois qu'une instruction de SENDIPC est exécutée, il y a allocation de la mémoire partagée, dont la dimension est calculée en fonction de la dimension des données qui sont envoyées. La dimension maximale de la mémoire partagée est de 64 Ko. On peut definir au maximum 48 mémoires partagées, identifiées par 48 noms univoques.
Un sémaphore est associé à la mémoire partagée. Ce dernier permet de synchroniser l'exécution des tâches qui y ont lieu. La tâche qui écrit les données active le sémaphore à la fin de l'écriture, la tâche qui lit les données le désactive à la fin de la lecture.
Si le paramètre **attente** a été réglé sur WAIT, la tâche qui a écrit les données attend que ces dernières soient lues (le sémaphore est désactivé) avant de poursuivre l'exécution.
Une instruction SENDIPC sans données n'est plus qu'un synchronisme entre des tâches. Dans ce cas, la mémoire partagée n'est pas allouée.

IPC intermodule
Deux modules à distance peuvent échanger des données au moyen de IPCs. Ces IPCx sont appelés IPC intermodule. Pour definir un IPC intermodule il faut écrire le **nomIPC** selon le formalisme suivant:
numéro du module source, "->", numéro du module destinataire, ":", ensuite les autres caractères du nom dans l'IPC.
Par exemple "0->1:Paramètres de Base".

Voir également WAITIPC et TESTIPC.

WAITIPC

Syntaxe

WAITIPC	**nomIPC [, nomvar1 [, nomvarN, ...]]**
WAITIPC	**nomIPC, matrice[ligne]**
WAITIPC	**nomIPC, vecteur**
WAITIPC	**nomIPC, matrice**

Arguments

nomIPC	constante chaîne. Nom de la IPC
nomvar1[...nomvarN]	constante ou variable. Noms variables 1÷N
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice
vecteur	Nom de vecteur
matrice	Nom de matrice

Description
Cette instruction reçoit une commande IPC de la mémoire partagée **"nomIPC"**.
La première fois qu'une instruction de WAITIPC est exécutée, il y a allocation de la mémoire partagée dont la dimension est calculée en fonction de la dimension des données qui sont envoyées. La dimension maximale de la mémoire partagée est de 64 Ko. On peut definir au maximum 48 mémoires partagées, identifiées par 48 noms univoques.
Un sémaphore est associé à la mémoire partagée. Ce dernier permet de synchroniser l'exécution des tâches qui y ont lieu. La tâche qui lit les données attend que le sémaphore soit activé par la tâche qui écrit les données, lit les données et désactive le sémaphore.
Une instruction WAITIPC sans données n'est plus qu'un synchronisme entre des tâches. Dans ce cas, la mémoire partagée n'est pas allouée.
Voir également SENDIPC et TESTIPC.

WAITRECEIVE

Syntaxe

WAITRECEIVE	**[source,] identificateur, flags [, conteneur]**

Arguments

source	constante type chaîne
identificateur	constante type chaîne
flags	constante type integer
conteneur	nom de dispositif ou variable (numérique ou chaîne)

Description
Cette instruction attend que l'information demandée (indiquée par **identificateur**) soit arrivée avant de poursuivre l'exécution du programme GPL. Pour son utilisation, se référer à la documentation de l'instruction RECEIVE.

10.3.10 Mathématiques

ABS

Syntaxe

ABS **opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction calcule la valeur absolue d'**opérande** et la met dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL -10,op ; attribue -10 à la variable op
ABS    op,var ; La valeur donnée à la variable var est 10
```

ADD

Syntaxe

ADD **opérande1, opérande2, résultat**

Arguments

opérande1	constante, variable ou nom de dispositif
opérande2	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une addition entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal      5,op1       ; assigne 5 à la variable opl
Add         op1,3,var

;La valeur donnée à la variable var est 8
```

AND

Syntaxe

AND **opérande1, opérande2, résultat**

Arguments

opérande1	constante, variable ou nom de dispositif
opérande2	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une opération de AND binaire (*entre deux bits, 1 seulement s'ils valent tous les deux 1*) entre **opérande1 et opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
;La valeur donnée à la variable var est 1
;(Notation binaire: 5 = 0101, 3 = 0011, 1 = 0001)

AND   5,3,var
```

ARCCOS

Syntaxe

ARCCOS **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération d'arccosinus sur **opérande** et met la valeur (en degrés) dans **résultat**. La valeur du résultat est donnée dans la fourchette 0°÷180°. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

ARCSIN

Syntaxe

ARCSIN **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération d'arcsinus sur **opérande** et met la valeur (en degrés) dans **résultat**. La valeur du résultat est donnée dans la fourchette -90°÷+90°. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

ARCTAN

Syntaxe

ARCTAN **opérande1 [, opérande2], résultat**

Arguments

opérande1...[opérande2] constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Si **opérande2** est omis, cette instruction exécute une opération d'arc tangente sur **opérande1** et met la valeur (en degrés) dans **résultat**.
Si **opérande2** est présent, l'angle considéré sera celui dont le sinus est donné par **opérande1** et dont le cosinus est donné par **opérande2**.
Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

COS

Syntaxe

COS **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération de cosinus sur **opérande** et met la valeur dans **résultat**. L'argument **opérande** est exprimé en degrés avec une éventuelle partie fractionnelle centésimale (ex.: 30° 15" = 30,25). Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal          60,op       ; assigne 60 à la varible op
Cos             op,var

;La valeur donnée à la variable var est 0.5.
```

DIV

Syntaxe

DIV	**opérande1, opérande2, résultat**

Arguments

opérande1	constante, variable ou nom de dispositif
opérande2	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une division entre **opérande1** et **opérande2** et met le résultat dans **résultat**.
L'instruction peut générer une erreur de système lorsque **opérande2** est égal à 0. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal      10,op1      ; assigne 10 à la variable op1
SetVal      5,op2       ; assigne  5 à la variable op2
Div         op1,op2,var

;La valeur donnée à la variable var est 2
```

EXP

Syntaxe

EXP	**opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction calcule l'exponentielle d'**opérande** et met la valeur dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      2.302585093,op      ;attribue 2.302585093 à la variable op
EXP          op,var

;La valeur donnée à la variable var est 10
```

EXPR

Syntaxe

EXP	**variable = expression**

Arguments

variable	nom du dispositif ou variable
expression	ensemble d' opérateurs

Description

Cette instruction permet de calculer des expressions mathématiques. Les opérateurs peuvent être constants, noms de dispositifs ou variables. La syntaxe de l'instruction impose que entre chaque opérateur et chaque opérande on doit ajouter une espace. Si les opérandes ne sont pas tous du même type, on fait une conversion automatique et le type du résultat de l'opération est identique à celui qui est majeur entre les deux, selon la règle suivante:

- char <integer
- float < double
- char ou integer < float ou double.

Une fois calculée l'**expression** le résultat est converti dans le type **variable**.
Les opérateurs admis, regroupés par niveaux de priorité sont les suivants :

()	parenthèse
-	opérateur de changement signe
ABS	valeur absolue d'opérande
ROUND	arrondissement à l'unité
TRUNC	tronquement à un nombre entier
LOG	logarithme naturel
LOGDEC	logarithme en base 10
EXP	exponentielle
SRQ	opération de racine carrée
SIN	opération de sinus. L' opérande est exprimé en degrés avec une éventuelle partie fractionnelle centésimale (ex. : 30° 15" = 30,25.)
COS	opération de cosinus. L' opérande est exprimé en degrés avec une éventuelle partie fractionnelle centésimale (ex. : 30° 15" = 30,25.)
TAN	opération de tangent. L' opérande est exprimé en degrés
ARCSIN	opération d'arcsinus. Le résultat est exprimé en degrés et sa valeur est donnée dans la fourchette -90°÷+90°
ARCCOS	opération d'arccosinus. Le résultat est exprimé en degrés et sa valeur est donnée dans la fourchette 0°÷180°
ARCTAN	opération d' arctangent. Voir ARCTAN
^	opérateur de puissance

*	multiplication
/	division
%	reste de la division (module)
+	somme
-	soustraction

Cette instruction permet de simplifier l'écriture du code GPL dans les cas où on doit effectuer calcules mathématiques, en substituant chaque instruction GPL correspondante aux opérateurs énumerés dans le tableaux. Les instructions restent disponibles par compatibilité.

Exemple

```
; calcul de la distance entre deux points

EXPR dist = SQR ( ( Xb - Xa ) ^ 2 + ( Yb - Ya ) ^ 2 )
```

LOG

Syntaxe

LOG **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction calcule le logarithme naturel d'**opérande** et met la valeur dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal      10,op          ; attribue 10 à la variable op
log         op,var

; La valeur dans la variable var sera 2.302585093
```

LOGDEC

Syntaxe

LOGDEC	**opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction calcule le logarithme en base 10 d'**opérande** et met la valeur dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      10,op       ; attribue 10 à la variable op
LOGDEC      op,var

;La valeur donnée à la variable var est 1
```

MOD

Syntaxe

MOD	**opérande1, opérande2, résultat**

Arguments

opérande1	costante ou variable integer ou nom de dispositif
opérande2	costante ou variable integer ou nom de dispositif
résultat	variable integer ou nom de dispositif

Description

Cette instruction exécute une opération de module entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Le module est le reste de la division entre le premier et le deuxième opérande. L'instruction peut générer une erreur de système lorsque **opérande2** est égal à 0. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      20,op1      ; attribue 20 à la variable op1
SETVAL      3,op2       ; attribue 3 à la variable op2
MOD         op1,op2,var

 ;La valeur donnée à la variable var est 2
```

MUL

Syntaxe

MUL	**opérande1, opérande2, résultat**

Arguments

opérande1	constante, variable ou nom de dispositif
opérande2	constante, variable ou nom de dispositif

résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération de module entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal    5,op1          ; attribue 5 à la variabile op1
SetVal    2,op2          ; attribue 2 à la variabile op2
Mul       op1,op2,var


 ;La valeur donnée à la variable var est 10
```

NOT

Syntaxe

NOT **opérande**

Arguments

opérande variable ou nom de dispositif

Description

Cette instruction exécute une opération de NOT binaire (*les différents bits sont inversés*) de la valeur exprimée par **opérande**. Le résultat est enregistré dans **opérande**.

Exemple

```
SETVAL          5,var ; attribue la valeur 5 à "var"
NOT             var

; Le résultat sera var = -6
; Notation binaire: 5 = 0000 0101,
; Notation hexadécimal  5 = 0000 0000 0000  0005
; Notation hexadécimal 10 = 0000 0000 0000 000A
; en exécutant un NOT sur valeur 5 on obtient 0xFFFF FFFF FFFF
; FFFA = -6
```

OR

Syntaxe

OR **opérande1, opérande2, résultat**

Arguments

opérande1 constante, variable ou nom de dispositif
opérande2 constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération de OR binaire (entre deux bits, 1 si au moins un des deux est égal à 1) entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
;La valeur donnée à la variable var est 7
;(Notation binaire: 5 = 0101, 3 = 0011, 7 = 0111)

OR     5,3,var
```

RANDOM

Syntaxe

RANDOM **min, max, résultat**

Arguments

min	constante ou variable
max	constante ou variable
résultat	variable ou nom de dispositif

Description

Cette instruction donne dans **résultat** un nombre pseudoaléatoire compris entre **min** et **max** (extrêmes inclus).
En exécutant l'instruction à plusieurs reprises, on obtient une série de nombres pseudoaléatoires.
Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SetVal      2,op1          ;assigne 2 à la variable op1
SetVal      100,op2        ;assigne 100 à la variable op2
Random      op1,op2,var    ;la valeur dans la variable var
                           ;sera un numéro aléatoire
                           ;entre 2 et 100
```

RESETBIT

Syntaxe

RESETBIT **masque, nbit**

Arguments

masque	constante ou variable entière ou nomcompteur ou nomport. Représente la valeur à modifier (max. 32 bits)
nbit	constante ou variable entière ou nomcounter. Numéro du bit à modifier

Description

Cette instruction règle à 0 un bit donné, indiqué par **nbit**, du **masque** de bit passé. L'argument **masque** doit pouvoir correspondre à une valeur entière avec un maximum de 32 bits. Le nombre de bits, **nbit**, va de 1 à 32.

Exemple

Etat du port avant l'exécution du code

Etat du port après l'exécution du code


```
;-------------------------------------------------------
; Exemple pour désactiver une ligne d'un port de flag:
;-------------------------------------------------------

SETVAL      2,nbit
RESETBIT    FlagPort,nbit

; désactive la ligne 2 du port de flag
```

ROUND

Syntaxe

ROUND **opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une opération d'arrondissement à l'unité sur **opérande** et met la valeur dans **résultat**.

Exemple

```
;La valeur donnée à la variable var sera 5


SETVAL  5.7,op    ;attribue 5.7 à la variable op
ROUND   op,var

;la valeur dans la variable var sera 6

SETVAL  5.2,op    ;attribue 5.2 à la variable op
ROUND   op,var


;la valeur dans la variable var sera 5
```

SETBIT

Syntaxe

SETBIT **masque, nbit**

Arguments

masque	constante ou variable entière ou nomcompteur ou nomport. Représente la valeur à modifier (max. 32 bits)
nbit	constante ou variable entière ou nomcompteur. Numéro du bit à modifier

Description

Cette instruction règle à 1 un bit donné, indiqué par **nbit**, du **masque** de bit passé. L'argument **masque** doit pouvoir correspondre à une valeur entière avec un maximum de 32 bits. Le nombre de bits, **nbit,** va de 1 à 32.

Exemple

Etat du port avant l'exécution du code

Etat du port après l'exécution du code


```
;-------------------------------------------------------
; Exemple per activer une ligne d'un port de flag:
;
;-------------------------------------------------------

SETVAL      2,nbit
SETBIT      FlagPort,nbit

; active la ligne 2 du port de flag
```

SHIFTL

Syntaxe

SHIFTL **opérande1 [, opérande2]**

Arguments

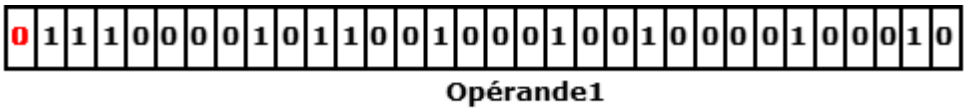
opérande1 variable (integer ou char) ou nom de dispositif
opérande2 variable (integer ou char) ou nom de dispositif

Description

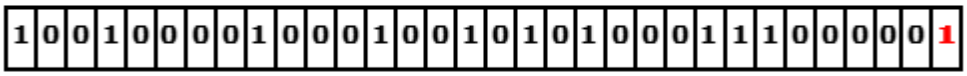
Si **opérande2** n'est pas spécifié, cette instruction exécute une opération de défilement à gauche des bits qui composent **l'opérande1**. Si le deuxième opérande aussi est spécifié , cette instruction exécute une opération de rotation entre **opérande2,** utilisé comme valeur 0 ou différent de 0 et les bits de **opérande1**. En ce cas à la fin de cette opération **opérande2** contiendra le retenue de l'opération et le bit de poids plus faible de **opérande1** deviendra 0 ou 1 en fonction de la valeur initiale de **opérande2** (0 ou différent de zéro).

Exemple

Rotation d' operands de typ integer (shift à gauche avec report)
Avant la rotation



Opérande1



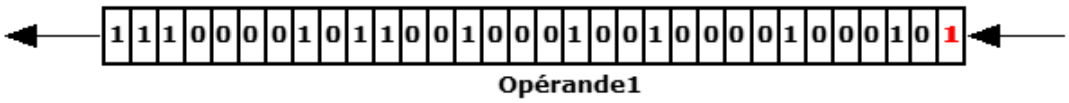
Opérande2

Après la rotation

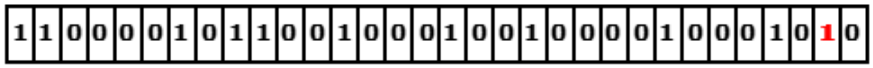
Opérande1

Opérande2

Shift (shift à gauche sans report)



Opérande1



Opérande1
après shift à
gauche

SHIFTR

Syntaxe

SHIFTR **opérande1 [, opérande2]**

Arguments

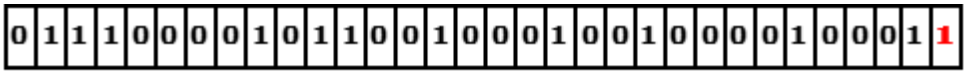
opérande1 variable (integer ou char) ou nom de dispositif
opérande2 variable (integer ou char) ou nom de dispositif

Description

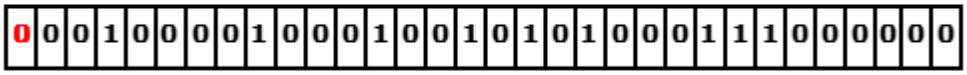
Si **opérande2** n'est pas spécifié, cette instruction exécute une opération de défilement à droite des bits qui composent **l'opérande1**. Si l'**opérande1** est déclaré de type char, le bit haut en entrée est toujours 0**.** Si operande1 est déclaré de type integer, le bit 32 en entrée est le bit de signe.
Si le deuxième opérande est également spécifié, il effectue une opération de rotation entre l'opérande2 destinée à prendre la valeur 0 ou différente de 0 et les bits de l'opérande1. A la fin de l'opération, l'opérande2 contiendra le report de l'opération et le bit de poids le plus élevé de l'opérande1 deviendra 0 ou 1, selon la valeur initiale de l'opérande2 (0 ou 1)

Exemple

Rotation d'opérandes de type integer (shift à droite avec retenue)
Avant la rotation

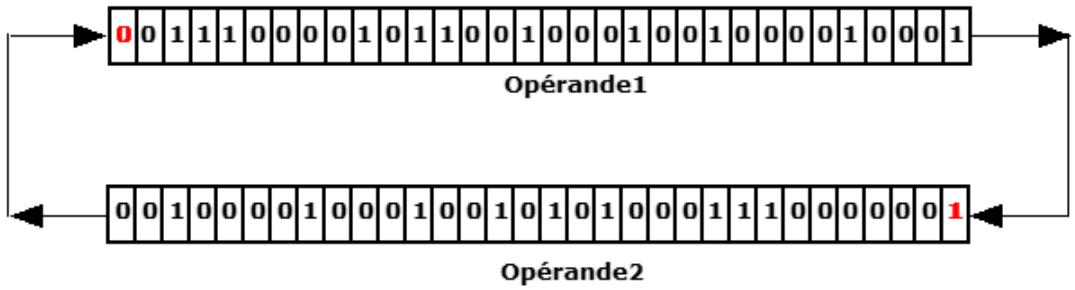


Opérande1

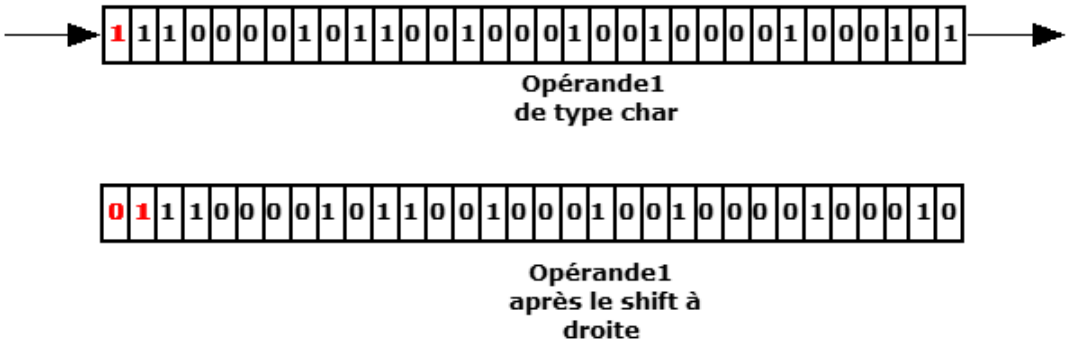


Opérande2

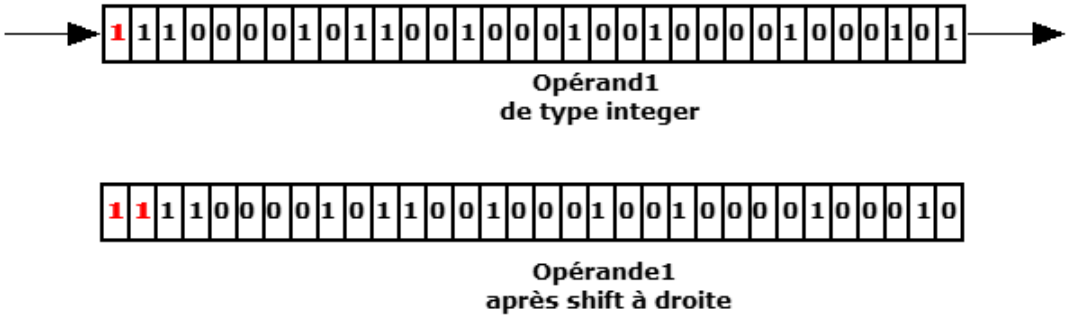
Après la rotation



Défilement à droite d'un char (shift a droite sans report)



Défilement à droite d'un integer (shift à droite sans report)



SIN

Syntaxe

SIN	**opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une opération de sinus sur **opérande** et met la valeur dans **résultat**. L'argument **opérande** est exprimé en degrés avec une éventuelle partie fractionnelle centésimale (ex. : 30° 15" = 30,25). Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      30,op          ;attribue 30 à la variable op
SIN         op,var

;La valeur donnée à la variable var est 0.5
```

SQR

Syntaxe

SQR **opérande, résultat**

Arguments

opérande	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une opération de racine carrée sur **opérande** et met la valeur dans **résultat**.
Le paramètre **opérande** n'admet que des valeurs positives. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      81,op        ;attribue 81 à la variable op
SQR         op,var

;La valeur donnée à la variable var est 9
```

SUB

Syntaxe

SUB **opérande1, opérande2, résultat**

Arguments

opérande1	constante, variable ou nom de dispositif
opérande2	constante, variable ou nom de dispositif
résultat	variable ou nom de dispositif

Description

Cette instruction exécute une soustraction entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      10,op1       ; attribue 10 à la variable op1
SETVAL      4,op2        ; attribue 4 à la variable op2
SUB         op1,op2,var

;La valeur donnée à la variable var est 6
```

TAN

Syntaxe

TAN **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération de tangente sut **opérande** et met la valeur dans **résultat**. L'argument **opérande** est exprimé en degrés. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      45,op       ;attribue 45 à la variable op
TAN         op,var

;La valeur donnée à la variable var est 1
```

TRUNC

Syntaxe

TRUNC **opérande, résultat**

Arguments

opérande constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Tronque à un nombre entier la valeur exprimée par opérande et met la valeur dans résultat. (La partie décimale est perdue). Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
SETVAL      5.7,op      ;attribue 5.7 à la variable op
TRUNC       op,var

;La valeur donnée à la variable var est 5
```

XOR

Syntaxe

XOR **opérande1, opérande2, résultat**

Arguments

opérande1 constante, variable ou nom de dispositif
opérande2 constante, variable ou nom de dispositif
résultat variable ou nom de dispositif

Description

Cette instruction exécute une opération de XOR binaire (entre deux bits, 1 si un seul des deux est égal à 1) entre **opérande1** et **opérande2** et met le résultat dans **résultat**. Pour la conversion des données, selon le type de donnée déclaré, il est fait référence au chapitre Conversion des données.

Exemple

```
XOR   5,3,var

;La valeur donnée à la variable var est 6
;(Notation binaire: 5 = 0101, 3 = 0011, 6 = 0110)
```

10.3.11 Multitâche

ENDMAIL

Syntaxe

ENDMAIL **mail**

Arguments

mail constante ou variable type integer. Numéro de la boîte aux lettres (1÷256)

Description

Cette instruction signale la fin de l'exécution d'une commande associée à un message prélevé de la boîte aux lettres **mail**.
La tâche qui avait envoyé le message (avec l'instruction SENDMAIL) et qui attendait l'exécution de la commande (argument d'attente WAITACK) peut ainsi poursuivre son exécution. Cette instruction ***n'est valable que si elle est exécutée par la tâche*** qui a reçu précédemment le message (avec l'instruction WAITMAIL ou TESTMAIL).

Voir également les instructions SENDMAIL, WAITMAIL et TESTMAIL

Exemple

Serveur de déplacement des axes

ENDREALTIMETASK

Syntaxe

ENDREALTIMETASK **nomfonction**

Arguments

nomfonction nom de fonction

Description

Cette instruction achève l'exécution d'une tâche real-time. Voir également STARTREALTIMETASK.

ENDTASK

Syntaxe

ENDTASK **[nomtâche]**

Arguments

nomtâche nom tâche

Description

Cette instruction achève l'exécution d'une tâche et de toutes les tâches que cette dernière a activées (tâches filles).
Cette instruction suspend également le mouvement des axes, annule les RECEIVE en suspens et ferme les éventuelles connexions aux ports sériels.
Si la variable **nomtâche** est omise, l'instruction interrompt l'exécution de la tâche courante.

GETPRIORITYLEVEL

Syntaxe

GETPRIORITYLEVEL **niveau [,nomfonction]**

Arguments

niveau variable. Niveau de priorité d'exécution
nomfonction nom de fonction

Description

Cette instruction redonne à la variable **niveau** la valeur de priorité de la tâche définie par **nomfonction**. Cette valeur est un nombre compris entre 1 et 255, où 1 représente le niveau de priorité le plus haut et où 255 est le plus bas. Si **nomfonction** n'est pas défini, la valeur de priorité

de la tâche courante, c'est-à-dire de la fonction dans laquelle l'instruction GETPRIORITYLEVEL est exécutée, est rendue.
Voir également SETPRIORITYLEVEL.

GETREALTIME

Syntaxe

GETREALTIME **nomvar**

Arguments

nomvar variable integer

Description

Cette instruction redonne à la variable **nomvar** le temps passé depuis le début du dernier real-time de gestion des axes. Le temps rendu est exprimé en microsecondes. Voir également GETREALTIMECOUNT.

GETREALTIMECOUNT

Syntaxe

GETREALTIMECOUNT **nomvar**

Arguments

nomvar variable integer

Description

Cette instruction redonne à la variable **nomvar** le nombre de real-time de gestion des axes exécutés depuis la dernière initialisation de la commande numérique. Voir également GETREALTIME.

HOLDTASK

Syntaxe

HOLDTASK **[nomtâche]**

Arguments

nomtâche nom tâche

Description

Cette instruction suspend l'exécution de la tâche définie dans **nomtâche.** Cette instruction ne suspend pas le mouvement des axes qui seront arrêtés avec les instructions de STOP.
Si **nomtâche** est omis, l'exécution de la tâche courante est suspendue.

RESUMETASK

Syntaxe

RESUMETASK **[nomtâche]**

Arguments

nomtâche nom tâche

Description

Cette instruction reprend l'exécution des tâches définies dans **nomtâche**. Si **nomtâche** est omis, l'exécution de la tâche courante est reprise. Si la tâche avait été suspendue avec l'instruction STOPTASK, même les éventuels mouvements des axes sont repris.

SENDMAIL

Syntaxe

SENDMAIL **mail, attente [, nomvar1 [,..nomvarN]]**
SENDMAIL **mail, attente, matrice[ligne]**

Arguments

mail	constante ou variable type integer. Numéro de la boîte aux lettres (1÷256).
attente	constante prédéfinie. Mode d'attente de lecture de la commande Les valeurs admises sont les suivantes : - **WAIT** attend la lecture de la commande - **NOWAIT** n'attend pas la lecture de la commande - **WAITACK** attend l'exécution de la commande
nomvar1[...nomvarN]	constante ou variable. Noms variables 1÷20
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice

Description

Cette instruction envoie un message (ou commande) à la boîte aux lettres **mail**. Les messages peuvent être utilisés pour synchroniser et échanger des informations entre deux ou plusieurs tâches.

Si la boîte aux lettres **mail** n'existe pas, c'est-à-dire si une instruction WAITMAIL ou TESTMAIL n'a pas encore été effectuée, l'instruction est ignorée.

Si la tâche réceptrice n'attend pas de message (instruction WAITMAIL) ou qu'elle est engagée, les données (**nomvar** (1÷20) ou la ligne de matrice indiquée de **matrice[ligne]**) passées par l'instruction sont enregistrées dans une queue. Dans ce cas :

1. si l'argument d'attente est **NOWAIT**, l'exécution continue avec l'instruction suivante ;
2. si l'argument d'attente est **WAIT**, l'exécution attend que le message soit lu par la tâche réceptrice ;
3. si l'argument d'attente est **WAITACK**, l'exécution attend que le message soit lu et que l'exécution de la commande soit confirmée par la tâche réceptrice (par le biais de l'instruction ENDMAIL ou une nouvelle WAITMAIL).

Il est très important que le nombre de variables passées et leurs types coïncident avec ceux qui ont été utilisés pour créer la boîte aux lettres avec l'instruction WAITMAIL. Le contrôle ne permet pas d'utiliser des types différents et il n'effectue pas de conversions de type automatiques (cast) comme à l'accoutumée.

Une instruction SENDMAIL sans les paramètres facultatifs (données) n'est plus qu'un simple mécanisme de synchronisation entre des tâches.

Exemple

Serveur de déplacement des axes

SETPRIORITYLEVEL

Syntaxe

SETPRIORITYLEVEL **niveau [, nomfonction]**

Arguments

niveau	constante ou variable. Niveau de priorité d'exécution.
nomfonction	nom de fonction

Description

Cette instruction règle, dans la variable **niveau**, la valeur de priorité de la tâche définie dans **nomfonction**. Cette valeur est un nombre compris entre 0 et 255, où 0 représente le niveau de priorité le plus haut et où 255 est le plus bas. Si le nom de la tâche n'est pas défini dans la variable **nomfonction**, la valeur de priorité de la tâche courante, c'est-à-dire le niveau d'exécution de la fonction dans laquelle l'instruction est exécutée, est modifiée.
Voir également GETPRIORITYLEVEL.

STARTREALTIMETASK

Syntaxe

STARTREALTIMETASK **nomfonction**

Arguments

nomfonction	nom de fonction

Description

Cette instruction active l'exécution d'une tâche real-time. Cette tâche est exécutée avec la même fréquence que le real-time de contrôle des axes. Contrairement aux tâches normales GPL, chaque real- time est exécuté entièrement, de la première instruction de la fonction à la première instruction FRET. Voir également ENDREALTIMETASK.

Note:
Les variables locales déclarées dans la tâches real-time ne sont initialisées qu' en démarrage de tâche et ensuite elles maintiennent la valeur de la dernière exécution.

STARTTASK

Syntaxe

STARTTASK	**nomtâche [, paramètres]**

Arguments

nomtâche	nom de la tâche
paramètres	éventuels paramètres nécessaires pendant l'exécution de la tâche

Description
Cette instruction active l'exécution de la tâche définie dans la variable **nomtâche**.
Cette tâche peut recevoir les éventuels **paramètres** utilisés pendant l'exécution. Le nombre et le type des paramètres passés doivent correspondre à ceux qui sont déclarés par la fonction qui implémente la tâche. Si la tâche est déjà en cours d'exécution, l'instruction n'a aucun effet.

Exemple
Exécution Séquentielle / Parallèle

STOPTASK

Syntaxe

STOPTASK	**nomtâche**

Arguments

nomtâche	nom de fonction

Description
Cette instruction arrête l'exécution d'une tâche et de toutes les autres tâches que cette dernière a activées (tâches filles), en interrompant le mouvement des axes (si ces derniers étaient en cours de déplacement).
Si **nomtâche** est omis, l'exécution de la tâche courante est interrompue. L'exécution de la tâche et le mouvement des axes peuvent être réactivés avec l'instruction RESUMETASK.

WAITMAIL

Syntaxe

WAITMAIL	**mail [, nomvar1 [,..nomvarN]]**
WAITMAIL	**mail, matrice[ligne]**

Arguments

mail	constante ou variable integer. Numéro de la boîte aux lettres (1÷256)
nomvar1[...nomvarN]	constante ou variable integer. Noms variables 1÷20
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice

Description
Cette instruction reçoit un message de la boîte aux lettres **mail**. Des données peuvent être associées à ce message.
Les données reçues avec le message sont enregistrées dans les variables **nomvar** (1÷20) indiquées ou dans la ligne de la matrice indiquée par **matrice[ligne]**.
Si, au moment où l'instruction WAITMAIL est exécutée, il y a déjà des messages en attente de lecture, la tâche est mise en état HOLD, dont elle ne sort que lorsqu'une autre tâche envoie un message à la boîte aux lettres avec l'instruction SENDMAIL.
La compatibilité entre les données passées et celles qui sont attendues par l'instruction est contrôlée pendant l'exécution de l'instruction.
Une instruction WAITMAIL sans paramètres facultatifs n'est plus qu'un simple mécanisme de synchronisation entre des tâches.

Voir également les instructions SENDMAIL, ENDMAIL et TESTMAIL.

Exemple

Serveur de déplacement des axes

WAITTASK

Syntaxe

WAITTASK **nomtâche**

Arguments

nomtâche nom tâche

Description

Cette instruction attend que la tâche **nomtâche** ait terminé l'exécution.

Exemple

Exécution Séquentielle / Parallèle

10.3.12 Gestion de flux

CALL

Syntaxe

CALL **nomsousprogramme**

Arguments

nomsousprogramme nom de sous-programme, étiquette

Description

Cette instruction exécute le sous-programme défini par l'étiquette **nomsousprogramme**.
Pour pouvoir revenir à l'instruction qui suit la CALL, chaque sous-programme doit s'achever, au point de sortie, par l'instruction RET.

Remarque

Avec RET, cette instruction constitue l'une sources d'erreurs de programmation les plus classiques. Il est conseillé de prêter la plus grande attention à son utilisation. En particulier, il est conseillé de positionner les sous-procédures à la fin du corps d'une fonction (après l'instruction FRET), de façon à éviter l'exécution accidentelle du code de la sous-procédure comme s'il faisait partie intégrante du code principal. Le résultat de cette situation est, dans le meilleur des cas, une erreur de système. Dans d'autres cas, on obtient un comportement anormal de la machine qu'il est difficile d'identifier.

DELONFLAG

Syntaxe

DELONFLAG **nomflag**

Arguments

nomflag nom de dispositif flag

Description

Cette instruction désactive la gestion de l'interruption de logiciel sur l'état d'un bit indicateur ou d'un flag switch qui avait été précédemment activé avec ONFLAG.

DELONINPUT

Syntaxe

DELONINPUT **nomentrée**

Arguments

nomentrée nom d'entrée

Description

Cette instruction désactive la gestion de l'interruption de logiciel sur l'état d'une entrée précédemment activée avec l'instruction ONINPUT.

FCALL

Syntaxe

[FCALL] nomfonction [, paramètres]
nomfonction [paramètres]

Arguments

nomfonction	nom de la fonction à rappeler
paramètres	éventuels paramètres passés à la fonction

Description

Cette instruction effectue un appel de fonction, c'est-à-dire que la fonction **nomfonction** est exécutée.
D'éventuels paramètres peuvent être passés à la fonction **paramètres.** Ces derniers doivent correspondre en nombre et en type à ceux qui sont déclarés dans la fonction appelée.
L'exécution de la fonction appelante (celle où est exécutée la FCALL) repart à la fin de l'exécution de la fonction appelée (celle qui est indiquée par le paramètre **nomfonction**).

Remarquer la différence avec l'instruction STARTTASK qui met en exécution une autre fonction en parallèle à l'appelante (utilisée pour avoir plusieurs tâches en exécution simultanée).

Exemple

Exécution Séquentielle / Parallèle

FOR / NEXT

Syntaxe

FOR indice, début, fin [, step]
instruction
instruction
...
NEXT

Arguments

indice	variable ou nomcompteur
début	constante, variable ou nomcompteur. Valeur initiale
fin	constante, variable ou nomcompteur. Valeur finale
step	constante, variable ou nomcompteur. Pas d'incrément ou de décrément

Description

Cette instruction répète de façon cyclique l'exécution des instructions contenues entre l'instruction FOR et l'instruction NEXT.
Pendant le premier cycle, la variable **indice** est initialisée à la valeur de la variable **début**. Au deuxième cycle, la variable **indice** aura une valeur égale à (**début**+**pas)** et ainsi de suite jusqu'à ce que la variable **indice** ne devienne supérieure (ou inférieure en cas de valeur négative de la variable **step**), à la variable **fin.** Si la variable **step** est omise, la valeur qui est prise par défaut est égale à +1.
Les instructions contenues entre FOR et NEXT peuvent modifier le nombre de répétitions en modifiant **indice**.
Lorsque les répétitions sont terminées, l'instruction qui suit NEXT est exécutée.

Exemple

```
Function Loop
 local    i As integer
 local    vecteur[10] as integer

 For      i,1,10
   Setval      i, vecteur[i]; remplit les éléments du vecteur
```

```
                                ; avec les nombres 1, 2, .... 10
 Next
 Fret



Function loop2
 local    j As integer
 local    vecteur[10] as integer

 For      j,1,10,2

   Setval      27, vecteur[j]; insère la valeur 27 dans les éléments
                               ; suivants du vecteur: 1,3,5,7,9
  Next
  Fret
```

FRET

Syntaxe

FRET

Arguments

Aucun argument

Description

Retour d'une fonction. Il provoque la fin de l'exécution de la fonction et le relâchement de la mémoire allouée aux variables locales. Si la fonction avait été mise en exécution avec une FCALL, l'exécution de la fonction appelante repart de l'instruction suivante.
Si l'on a précédemment exécuté des WAITTASK avec la fonction courante (celle où est exécutée la FRET) comme argument, les tâches en suspens sont débloquées.

GOTO

Syntaxe

GOTO **étiquette**

Arguments

étiquette étiquette

Description

Cette instruction effectue un saut inconditionné à l'étiquette indiquée par le paramètre **étiquette**.
Une étiquette est définie par un mot clé immédiatement suivi du caractère ":".
L'étiquette doit se trouver à l'intérieur du corps de la fonction où l'instruction GOTO est exécutée.

Remarque

Le corps d'une fonction est compris entre l'instruction FONCTION qui déclare le nom de la fonction et l'instruction analogue qui définit la fonction suivante (ou à partir de la fin du fichier). On en déduit qu'il est possible d'exécuter des sauts du corps principal de la fonction vers d'éventuels sous-procédures (voir instructions CALL et RET). Ce style de programmation est vivement déconseillé car il est à l'origine de nombreuses erreurs difficiles à identifier.

Exemple

```
 ; Fonction qui fait clignoter un flag
 ; (ex.: un voyant d'alarme sur un tableau synoptique)

Function Loop

loop:
 Setflag    alarme     ; active le flag
 delay      1
 resetflag  alarme     ; désactive le flag
 delay      1
```

goto loop
Next
Fret

IF/IFVALUE/IF-THEN-ELSE

Syntaxe

IF **nomvar, opérateur de comparaison, valeur, GOTO étiquette**
IF **nomvar, opérateur de comparaison, valeur, CALL nomsousprogramme**
IF **nomvar,opérateur de comparaison, valeur, nomfonction**

IF **nomvar, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ENDIF

IF **nomvar, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

nomvar	constante ou variable ou nomdevice
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont: **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomdevice
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Les instructions IF et IFVALUE sont synonymes. Il est conseillé d'utiliser la forme brève.
L'instruction permet d'effectuer une comparaison entre **nomvar** et **valeur** et d'entreprendre une action en fonction du résultat de la comparaison.
Dans les premières trois formes, si la comparaison est positive, un saut à l'étiquette (GOTO), un appel de sous-programme (CALL) ou un appel de fonction (nomfonction) est effectué. A la fin de l'exécution de la fonction ou du sous-programme appelé, l'exécution repart à la ligne suivante. En revanche, si la comparaison est négative, l'exécution du programme se poursuit.
L'élaboration IF...THEN permet d'exécuter une ou plusieurs instructions de façon conditionnée. Les instructions comprises entre les mots clé THEN et ENDIF sont exécutées si la comparaison entre **nomvar** et **valeur** est positive.
L'élaboration IF...THEN...ELSE permet de définir deux blocs d'instructions, dont un seul sera exécuté. Si la comparaison entre **nomvar** et **valeur** est positive, les instructions comprises entre les mots clé THEN et ELSE sont exécutées ; si elle est négative, les instructions qui sont exécutées sont celles qui sont comprises entre les mots clé ELSE et ENDIF. Dans un cas comme dans l'autre, l'exécution se poursuit avec l'instruction qui suit ENDIF.

Remarque

L'instruction IFVALUE est maintenue pour garantir la compatibilité avec les versions précédentes de GPL.

IFACC

Syntaxe

IFACC	**axe, GOTO étiquette**
IFACC	**axe, CALL nomsousprogramme**
IFACC	**axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** est dans l'état d'accélération.
Si la condition est vérifiée, elle saute à **étiquette** ou appelle **nomsousprogramme** ou **nomfonction.**

IFAND

Syntaxe

IFAND **opérande1, opérande2, valeurvérification, GOTO étiquette**
IFAND **opérande1, opérande2, valeurvérification, CALL nomsousprogramme**
IFAND **opérande1, opérande2, valeurvérification, nomfonction**

IFAND **opérande1, opérande2, valeurvérification THEN**
instruction
instruction
...
ENDIF

IFAND **opérande1, opérande2, valeurvérification THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

opérande1	constante, variable ou nomdevice
opérande2	constante, variable ou nomdevice
valeurvérification	constante. Valeur utilisée pour le contrôle du résultat de l'opération. Les valeurs pouvant être prises sont les suivantes: **TRUE 1, FALSE 0**
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Deux comparaisons sont effectuées, la première entre **opérande1** et **opérande2**, la deuxième entre le résultat de la première comparaison et **valeurvérification**.
La première comparaison consiste en un AND binaire entre **opérande1** et **opérande2**. Les deux opérandes sont considérés comme des masques de bit. Si le résultat de l'AND binaire a au moins un bit différent de 0, le résultat de la première comparaison est TRUE. Il est ensuite comparé avec **valeurvérification**. Si les deux valeurs coïncident, il y a un saut à étiquette ou un appel de fonction ou de sous-programme.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFBIT

Syntaxe

IFBIT **masque, nbit, état, GOTO étiquette**
IFBIT **masque, nbit, état, CALL nomsousprogramme**
IFBIT **masque, nbit, état, nomfonction**

IFBIT **masque, nbit, état THEN**
instruction
instruction
...
ENDIF

IFBIT **masque, nbit, état THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

masque	constante ou variable entière ou nomcompteur ou nomport. Valeur à vérifier
nbit	constante ou variable entière ou nomcompteur. Numéro du bit (1÷32)
état	constante prédéfinie. Etat à vérifier sur masque Les valeurs admises sont les suivantes : **ON** bit choisi sur 1 **OFF** bit choisi sur 0
étiquette	étiquette de saut (GOTO)
nomsousprogramme	appel de sous-programme (CALL)
nomfonction	nom de fonction

Description

Test sur un bit donné du **masque** de bit passé. L'argument **masque** doit pouvoir correspondre à une valeur entière avec un maximum de 32 bits. Le nombre à attribuer à la variable **nbit** pour identifier le bit à contrôler va de 1 à 32. Si la condition indiquée dans **état** est vérifiée, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFBLACKBOX

Syntaxe

IFBLACKBOX **GOTO étiquette**
IFBLACKBOX **CALL nomsousprogramme**
IFBLACKBOX **nomfonction**

Arguments

étiquette	nom de l'étiquette 'etichetta a cui saltare
nomsousprogramme	nom del sottoprogramma
nomfonction	nom della funzione

Description

Si l'enregistrement est actif, la registrazione è attiva, saute à **étiquette** ou appelle **nomsousprogramme** ou **nomfonction.** Voyez aussi STARTBLACKBOX, PAUSEBLACKBOX et ENDBLACKBOX.

IFCHANGEVEL

Syntaxe

IFCHANGEVEL	**axe [, état], GOTO étiquette**
IFCHANGEVEL	**axe [, état], CALL nomsousprogramme**
IFCHANGEVEL	**axe [, état], nomfonction**

Arguments

axe	nom de dispositif type axe
état	type di variation. Les valeurs admises sont les suivantes: **POSITIVE** **NEGATIVE**
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste s'il y a eu une variation de vitesse d'un axe.
Si l'axe défini par la variable **axe** est en train d'effectuer un changement de vitesse pendant un déplacement, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Le paramètre **état** indique si la vitesse a augmenté (POSITIVE) ou diminué (NEGATIVE).

IFCOUNTER

Syntaxe

IFCOUNTER **nomcompteur, opérateur de comparaison, valeur, GOTO étiquette**

IFCOUNTER **nomcompteur, opérateur de comparaison, valeur, CALL nomsousprogramme**

IFCOUNTER **nomcompteur, opérateur de comparaison, valeur, nomfonction**

IFCOUNTER **nomcompteur, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ENDIF

IFCOUNTER **nomcompteur, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

nomcompteur	nom de compteur
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont: **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomcompteur
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

L'instruction exécute le test d'un compteur.

Si le contenu du compteur défini dans la variable **nomcompteur** contrôle la condition indiquée par l'**opérateur de comparaison** avec la valeur exprimée par la variable **valeur**, il y a un saut à l'étiquette indiquée par **étiquette** ou un appel du sous-programme défini par **nomsousprogramme** ou de fonction définie par **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFDEC

Syntaxe

IFDEC **axe, GOTO étiquette**
IFDEC **axe, CALL nomsousprogramme**
IFDEC **axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** est en état de décélération.
Si la condition est vérifiée, il y a un saut à étiquette ou un appel du **nomsousprogramme** ou de **nomfonction.**

IFDIR

Syntaxe

IFDIR **axe, direction, GOTO étiquette**
IFDIR **axe, direction, CALL nomsousprogramme**
IFDIR **axe, direction, nomfonction**

IFDIR **axe, direction THEN**
instruction
instruction
...
ENDIF

IFDIR **axe, direction THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

axe	nom de dispositif type axe
direction	direction axe. Les valeurs admises sont les suivantes: **POSITIVE** direction axe positive **NEGATIVE** direction axe négative
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom di sous-programme
nomfonction	nom di fonction

Description

Test sur la direction courante d'un axe.
Si l'axe se déplace dans la direction indiquée par la variable **direction**, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFERRAN

Syntaxe

IFERRAN **axe, opérateur de comparaison, valeur, GOTO étiquette**
IFERRAN **axe, opérateur de comparaison, valeur, CALL nomsousprogramme**
IFERRAN **axe, opérateur de comparaison, valeur, nomfonction**

IFERRAN **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ENDIF

IFERRAN **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

axe	nom de dispositif type axe
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomcompteur
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test sur la valeur de l'erreur de poursuite (erreur de boucle) de l'axe défini dans la variable **axe.**
Si l'erreur de boucle de l'**axe** vérifie la condition exprimée par l'**opérateur de comparaison** avec la valeur exprimée par **valeur**, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFERROR

Syntaxe

IFERROR **numéro, IDposiz, GOTO étiquette**
IFERROR **numéro, IDposiz, CALL étiquette**
IFERROR **numéro, IDposiz, nomfonction**
IFERROR **nomedevice, état,IDposiz, GOTO étiquette**
IFERROR **nomedevice, état,IDposiz, CALL étiquette**
IFERROR **nomedevice, état,IDposiz, nomfonction**

Arguments

numéro	DEFMSG ou constante ou variable integer
nomedevice	nom de dispositif
état	constante prédéfinie. Les valeurs pouvant être prises sont les suivantes : **ON, OFF**
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomfonction	nom de fonction

Description

Test sur l'erreur de cycle actif.

Si l'erreur de cicle, identifié par **numéro** et **IDposiz** ou par **nomedevice, état** et **IDposiz** est actif il est effectué un saut à **étiquette** o un appel à la fonction **nomfonction.**
Le paramètre **numéro** peut identifier une erreur de cycle de module (donc une valeur numérique entière) ou de groupe (dans ce cas, on utilise une DEFMSG). Le paramètre **nomedevice** est le nom d'un dispositif et le paramètre **état** représente l'état ON/OFF où il faut trouver le dispositif au moment de la générateur de l'erreur. Le paramètre **IDposiz** est un paramètre en option qui spécifie la valeur numérique utilisée dans les tableaux synoptiques pour distribuer les erreurs de cycle dans des différentes cases. Il doit correspondre à la valeur spécifiée dans le Constructeur synoptique pour cette case particulière d'affichage. S'il n'est pas utile d'indiquer une case spécifique, il faut attribuer la constante prédéfinie NOPLACE. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023. Si l'instruction est utilisée sans que la gestion des alarmes d'état ait été activée, une erreur de système est générée.

Voir également l'instruction ERROR.

IFFLAG

Syntaxe

```
IFFLAG                  nomflag, état, GOTO étiquette
IFFLAG                  nomflag, état, CALL nomsousprogramme
IFFLAG                  nomflag, état, nomfonction
IFFLAG                  nomflag, état THEN
        instruction
        instruction
        ...
ENDIF

IFFLAG                  nomflag, état THEN
        instruction
        instruction
        ...
ELSE
        instruction
        instruction
        ...
ENDIF
```

Arguments

nomflag	nom de dispositif flag
état	constante prédéfinie. Etat à vérifier. Les valeurs admises sont les suivantes : **ON** active **OFF** désactif
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test de l'état logique d'un flag.
Si le flag défini par la variable **nomflag** se trouve dans l'**état** indiqué, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFINPUT

Syntaxe

```
IFINPUT                 nomentrée, état, GOTO étiquette
IFINPUT                 nomentrée, état, CALL nomsousprogramme
IFINPUT                 nomentrée, état, nomfonction


IFINPUT                 nomentrée, état THEN
        instruction
        instruction
        ...
ENDIF
```

IFINPUT **nomentrée, état THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

nomentrée	nom entrée
état	constante prédéfinie. État à vérifier Les valeurs admises sont les suivantes : **- ON** actif **- OFF** non actif
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test de l'état logique d'une entrée.
Si l'entrée définie dans la variable **nomentrée** se trouve dans l'**état** indiqué, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFMESSAGE

Syntaxe

IFMESSAGE **numéro, IDposiz, GOTO étiquette**
IFMESSAGE **numéro, IDposiz, CALL étiquette**
IFMESSAGE **numéro, IDposiz, nomfonction**

Arguments

numéro	DEFMSG ou constante ou variable integer
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomfonction	nom de fonction

Description

Test sur le message actif.
Si le message, identifié par **numéro** et **IDposiz** est actif il est il est effectué un saut à **étiquette** o un appel à la fonction **nomfonction.**
Le paramètre **IDposiz** est un paramètre en option qui spécifie la valeur numérique utilisée dans les tableaux synoptiques pour distribuer les erreurs de cycle dans des différentes cases. Il doit correspondre à la valeur spécifiée dans le Constructeur synoptique pour cette case particulière d'affichage. S'il n'est pas utile d'indiquer une case spécifique, il faut attribuer la constante prédéfinie NOPLACE. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023.
Pour utiliser l'instruction il doit être activé la gestion des alarmes à états.
Voir également l'instruction MESSAGE.

IFOR

Syntaxe

IFOR **opérande1, opérande2, valeurvérification, GOTO étiquette**
IFOR **opérande1, opérande2, valeurvérification, CALL nomsousprogramme**
IFOR **opérande1, opérande2, valeurvérification, nomfonction**

IFOR **opérande1, opérande2, valeurvérification THEN**
instruction
instruction

...
ENDIF

IFOR **opérande1, opérande2, valeurvérification THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

opérande1	constante ou variable ou nomdevice
opérande2	constante ou variable ou nomdevice
valeurvérification	constante. C'est la valeur utilisée pour vérifier le résultat de l'opération Les valeurs pouvant être prises sont les suivantes : **TRUE** 1 **FALSE** 0
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Deux comparaisons sont effectuées, la première est entre **opérande1** et **opérande2**, la deuxième entre le résultat de la première comparaison et **valeurvérification**.
La première comparaison consiste en un OR binaire entre **opérande1** et **opérande2**. Les deux opérandes sont considérés comme des masques de bit. Si le résultat de l'OR binaire a au moins un bit différent de 0, le résultat de la première comparaison est TRUE. Il est ensuite comparé avec con **valeurvérification**. Si les deux valeurs coïncident, il y a un saut à étiquette ou un appel de fonction ou de sous-programme.
Pour toute information complémentaire sur la construction IF-THEN-ELSE voir IF / IFVALUE / IF-THEN-ELSE.

IFOUTPUT

Syntaxe

IFOUTPUT **nomsortie, état, GOTO étiquette**
IFOUTPUT **nomsortie, état, CALL nomsousprogramme**
IFOUTPUT **nomsortie, état, nomfonction**

IFOUTPUT **nomsortie, état THEN**
instruction
instruction
...
ENDIF

IFOUTPUT **nomsortie, état THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

nomsortie	nom sortie
état	constante prédéfinie. Elle représent l'état à vérifier sur la sortie. Les valeurs pouvant être prises sont les suivantes :
ON	actif
OFF	déactif
étiquette	nom del'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme

nomfonction	nom de fonction

Description

Test de l'état logique d'une sortie.
Si la sortie définie dans la variable **nomsortie** se trouve dans l'état indiqué, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFQUOTER

Syntaxe

IFQUOTER **axe, opérateur de comparaison, valeur, GOTO étiquette**
IFQUOTER **axe, opérateur de comparaison, valeur, CALL nomsousprogramme**
IFQUOTER **axe, opérateur de comparaison, valeur, nomfonction**

IFQUOTER **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ENDIF

IFQUOTER **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

axe	nom de dispositif type axe
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomcompteur
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test sur la cote réelle exprimée par la variable **axe.**
Si la valeur de la variable **axe** vérifie la condition exprimée par l'**opérateur de comparaison** avec la valeur exprimée par **valeur**, il y a un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFQUOTET

Syntaxe

IFQUOTET **axe, opérateur de comparaison, valeur, GOTO étiquette**
IFQUOTET **axe, opérateur de comparaison, valeur, CALL nomsousprogramme**
IFQUOTET **axe, opérateur de comparaison, valeur, nomfonction**

IFQUOTET **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...

ENDIF

IFQUOTET **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

axe	nom de dispositif type axe
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomcompteur
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test sur la cote théorique exprimée par la variable **axe.**
Si la valeur de la variable **axe** vérifie la condition exprimée par l'**opérateur de comparaison** avec la valeur exprimée par **valeur**, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFRECEIVED

Syntaxe

IFRECEIVED	**[source,] identificateur, GOTO étiquette**
IFRECEIVED	**[source,] identificateur, CALL nomsousprogramme**
IFRECEIVED	**[source,] identificateur, nomfonction**

Arguments

source	constante type chaîne
identificateur	constante type chaîne
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction contrôle si une instruction RECEIVE a été satisfaite.
Si une RECEIVE précédente particulière a été satisfaite, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.
Voir également les instructions RECEIVE, WAITRECEIVE, SEND.

IFREG

Syntaxe

IFREG	**axe, GOTO étiquette**
IFREG	**axe, CALL nomsousprogramme**
IFREG	**axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** est dans l'état de régime.
Si la condition est vérifiée, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction.**

IFSAME

Syntaxe

IFSAME	**opérande1, opérande2, GOTO étiquette**
IFSAME	**opérande1, opérande2, CALL nomsousprogramme**
IFSAME	**opérande1, opérande2, nomfonction**

Arguments

opérande1	variable ou nomdevice
opérande2	variable ou nomdevice
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test entre deux opérandes.
L'instruction s'assure que la valeur définie dans **opérande1** et **opérande2** se réfère au même dispositif ou à la même zone de mémoire.
Si le test entre les deux opérandes est vérifié, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.

IFSTILL

Syntaxe

IFSTILL	**axe, GOTO étiquette**
IFSTILL	**axe, CALL nomsousprogramme**
IFSTILL	**axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** est arrêtée, c'est-à-dire en "cote".
Si la condition est vérifiée, il y a saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction.**
Voir également IFTARGET et IFWIN.

IFSTR

Syntaxe

IFSTR	**chaîne1, opérateur de comparaison, chaîne2, GOTO étiquette**
IFSTR	**chaîne1, opérateur de comparaison, chaîne2, CALL nomsousprogramme**
IFSTR	**chaîne1, opérateur de comparaison, chaîne2, nomfonction**

```
IFSTR                   chaîne1, opérateur de comparaison, chaîne2 THEN
        instruction
        instruction
        ...
ENDIF

IFSTR                   chaîne1, opérateur de comparaison, chaîne2 THEN
        instruction
        instruction
        ...
ELSE
        instruction
```

instruction
...
ENDIF

Arguments

chaîne1	variable string. C'est la première chaîne ASCII
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
chaîne2	variable string. C'est la deuxième chaîne ASCII
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de l sous-programme
nomfonction	nom de fonction

Description

Test sur les chaînes ASCII.
Si la chaîne définie dans **chaîne1** vérifie la condition exprimée par l'**opérateur de comparaison** avec la chaîne contenue dans **chaîne2**, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFTARGET

Syntaxe

IFTARGET	**axe, GOTO étiquette**
IFTARGET	**axe, CALL nomsousprogramme**
IFTARGET	**axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** a atteint la cote finale programmée. Même s'il a atteint la cote cible finale, l'axe n'est pas nécessairement arrêté et, en général, il devra récupérer l'erreur de boucle. Si la condition est vérifiée, il y a saut à **étiquette** ou appel de **nomsousprogramme** ou de **nomfonction.**
Voir également IFSTILL et IFWIN.

IFTASKHOLD

Syntaxe

IFTASKHOLD	**nomtâche, GOTO étiquette**
IFTASKHOLD	**nomtâche, CALL étiquette**
IFTASKHOLD	**nomtâche, nomfonction**

Arguments

nomtâche	nom tâche parallèle
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction contrôle si la tâche est suspendue (état hold).
Si la tâche **nomtâche** est suspendue, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.

IFTASKRUN

Syntaxe

IFTASKRUN	**nomtâche, GOTO étiquette**
IFTASKRUN	**nomtâche, CALL nomsousprogramme**
IFTASKRUN	**nomtâche, nomfonction**

Arguments

nomtâche	nom tâche parallèle
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction contrôle si la tâche est en exécution.
Si la tâche définie dans **nomtâche** est en exécution, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.

IFTIMER

Syntaxe

IFTIMER		**nomtemporisateur, opérateur de comparaison, valeur, GOTO étiquette**
IFTIMER		**nomtemporisateur, opérateur de comparaison, valeur, CALL nomsousprogramme**
IFTIMER		**nomtemporisateur, opérateur de comparaison, valeur, nomfonction**
IFTIMER		**nomtemporisateur, opérateur de comparaison, valeur THEN**
	instruction	
	instruction	
	...	
ENDIF		
IFTIMER		**nomtemporisateur, opérateur de comparaison, valeur THEN**
	instruction	
	instruction	
	...	
ELSE		
	instruction	
	instruction	
	...	
ENDIF		

Arguments

nomtemporisateur	nom de dispositif temporisateur
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : **<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou variable ou nomtemporisateur. C'est la valeur sur laquelle exécuter la comparaison
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test d'un temporisateur.
Si le contenu du temporisateur **nomtemporisateur** vérifie la condition exprimée par l'**opérateur de comparaison** avec la valeur exprimée par **valeur**, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFVEL

Syntaxe

IFVEL **axe, opérateur de comparaison, valeur, GOTO étiquette**
IFVEL **axe, opérateur de comparaison, valeur, CALL nomsousprogramme**
IFVEL **axe, opérateur de comparaison, valeur, nomfonction**

IFVEL **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ENDIF

IFVEL **axe, opérateur de comparaison, valeur THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

axe	nom de dispositif type axe
opérateur de comparaison	Les symboles qui on peut utiliser pour exécuter la comparaison sont : (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test sur la vitesse courante d'un axe.
Si la vitesse de l'axe vérifie la condition exprimée par l'**opérateur de comparaison** avec la valeur exprimée par **valeur**, il y a un saut à **étiquette** ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pour toute information complémentaire voir la construction IF-THEN-ELSE.

IFWIN

Syntaxe

IFWIN **axe, GOTO étiquette**
IFWIN **axe, CALL nomsousprogramme**
IFWIN **axe, nomfonction**

Arguments

axe	nom de dispositif type axe
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Cette instruction teste si l'axe défini par la variable **axe** est entré à l'intérieur de la fenêtre d'arrivée en cote (voir Conventions et terminologie).
Si la condition est vérifiée, il y a saut à **étiquette** ou appel de **nomsousprogramme** ou de **nomfonction.**
Voir également IFTARGET et IFSTILL.

IFXOR

Syntaxe

IFXOR **opérande1, opérande2, valeurvérification, GOTO étiquette**
IFXOR **opérande1, opérande2, valeurvérification, CALL nomsousprogramme**
IFXOR **opérande1, opérande2, valeurvérification, nomfonction**

IFXOR **opérande1, opérande2, valeurvérification THEN**
instruction
instruction
...
ENDIF

IFXOR **opérande1, opérande2, valeurvérification THEN**
instruction
instruction
...
ELSE
instruction
instruction
...
ENDIF

Arguments

opérande1	constante ou variable ou nomdevice
opérande2	constante ou variable ou nomdevice
valeurverification	constante. C'est la valeur utilisée pour la vérification du résultat de l'opération. Les valeurs pouvant être prises sont les suivantes : **TRUE** 1, **FALSE** 0
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Deux comparaisons sont effectuées, la première est entre **opérande1** et **opérande2**, la deuxième entre le résultat de la première comparaison et **valeurvérification**.
La première comparaison consiste en un XOR binaire entre **opérande1** et **opérande2**. Les deux opérandes sont considérés comme des masques de bit. Si le résultat de l'XOR binaire a au moins un bit différent de 0, le résultat de la première comparaison est TRUE. Il est ensuite comparé avec **valeurvérification**. Si les deux valeurs coïncident, il y a un saut à étiquette ou un appel de fonction ou de sous-programme.
Pour toute information complémentaire sur la construction IF-THEN-ELSE voir IF-THEN-ELSE.

ONERRSYS

Syntaxe

ONERRSYS **nomfonction**

Arguments

nomfonction	nom de fonction

Description

Cette instruction active la gestion des erreurs de système. La réaction normale du contrôle, lorsqu'une erreur de système a lieu, consiste à terminer toutes les tâches. La gestion des erreurs de système permet d'éviter l'interruption des tâches pour lesquelles elle a été activée.
Lorsqu'une erreur de système a lieu, la fonction **nomfonction** est mise en exécution. La tâche de cette fonction est d'analyser l'erreur de système et d'entreprendre les actions qui s'imposent pour mettre la machine en sécurité.

La fonction **nomfonction** est soumise à deux contraintes.
En premier lieu, elle doit accepter les paramètres suivants :

- le numéro de l'erreur de système, comme Integer
- la tâche dans laquelle l'erreur a eu lieu, comme Fonction
- le dispositif qui a généré l'erreur, comme device

Deuxièmement, elle ne peut pas contenir un certain nombre d'instructions GPL. Voir la liste des instructions non utilisables sur Interrupt.

En cas d'Erreurs de Système multiples, la fonction est appelée une fois pour chaque erreur générée, de façon séquentielle. Si la fonction elle-même génère une Erreur de Système, toutes les tâches sont interrompues.

Pendant l'exécution de la fonction, la tâche pour laquelle la gestion des erreurs a été activée est arrêtée et ne repart qu'à la fin de la première fonction appelée par l'instruction ONERRSYS. En particulier, la tâche reprendra son exécution en exécutant à nouveau l'instruction interrompue par l'erreur système.

Exemple

Cycle de Main avec gestion des Erreurs

ONFLAG

Syntaxe

ONFLAG **nomflag, [état,] nomfonction[,arguments]**

Arguments

nomflag	nom de dispositif flag
état	constante prédéfinie. Etat à vérifier. Les valeurs admises sont les suivantes : **ON** actif **OFF** non actif
nomfonction	nom fonction
arguments	éventuels arguments de la fonction

Description

Cette instruction active l'interruption de logiciel, associée à l'état du flag indiqué, de la tâche où elle est exécutée. Lorsque le flag commute dans l'état indiqué (interrupt), l'exécution de la tâche est interrompue et la fonction indiquée par **nomfonction** est mise en exécution. À la fin de cette dernière, l'exécution de la tâche repart de là où elle avait été interrompue. La fonction exécutée lorsque l'interrupt a lieu est soumise à des limitations. En particulier, toutes les instructions GPL ne peuvent pas être incluses dans le corps de la fonction. Cette limite sert à éviter les situations de blocage critique du code GPL ou d'attentes prolongées. Voir la liste des instructions non exécutables sur Interrupt.
Si l'argument **état** est omis, la fonction est appelée à chaque changement d'état du flag.
Le test sur l'état du flag est exécuté toutes les 5 ms. Ainsi, entre la variation du flag et l'exécution de la fonction, il peut y avoir un temps maximal de défaillance de 5 ms.
Il n'est pas possible de définir plusieurs ONFLAG sur le même flag.
Pour les arguments de la fonction définie dans **nomfonction**, il n'est pas possible d'utiliser des vecteurs ou des matrices locales.
Voir également les instructions DELONFLAG, ONINPUT, DELONINPUT.

ONINPUT

Syntaxe

ONINPUT **nomentrée, [état,] nomfonction [,arguments]**

Arguments

nomentrée	nom entrée
état	constante prédéfinie. Etat à vérifier. Les valeurs admises sont les suivantes : **ON** actif **OFF** non actif
nomfonction	nom fonction
arguments	éventuels arguments de la fonction

Description

Cette instruction active l'interruption de logiciel, associée à l'état de l'entrée indiquée, de la tâche où elle est exécutée. Lorsque l'entrée commute dans l'état indiqué (interrupt), l'exécution de la tâche est interrompue et la fonction indiquée par **nomfonction** est mise en exécution. A la fin de cette dernière, l'exécution de la tâche repart de là où elle avait été interrompue. La fonction exécutée

lorsque l'interrupt a lieu est soumise à des limitations. En particulier, toutes les instructions GPL ne peuvent pas être incluses dans le corps de la fonction. Cette limite sert à éviter les situations de blocage critique du code GPL ou d'attentes prolongées. Voir la liste des instructions non exécutables sur Interrupt.
Si l'argument **état** est omis, la fonction est appelée à chaque changement d'état de l'entrée.
Le test sur l'état de l'entrée est exécuté toutes les 5 ms auxquelles s'ajoutent 4 ms de filtrage anti-rebondissement sur la gestion des entrées. Ainsi, il peut y avoir un temps maximal de défaillance de 9 ms avant le lancement de la fonction. Il n'est pas possible de définir plusieurs ONINPUT sur la même entrée.
Voir également les instructions DELONINPUT, ONFLAG et DELONFLAG.

REPEAT / ENDREP

Syntaxe

REPEAT **valeur**
instruction
instruction
...
ENDREP

Arguments

valeur constante, variable ou nomcompteur. Nombre de répétitions

Description

Cette instruction répète l'exécution des instructions contenues entre l'instruction REPEAT et l'instruction ENDREP pour un nombre de fois indiqué par la variable **valeur**.
Lorsque le programme rencontre l'instruction ENDREP, le compteur de répétitions est décrémenté et s'il n'est pas inférieur ou égal à zéro, le bloc d'instructions est réexécuté à partir de l'instruction de la ligne qui suit REPEAT. Les instructions sont donc exécutées au moins une fois (même si, au départ, le paramètre valeur est nul ou négatif).
Lorsque les répétitions sont terminées, l'instruction qui suit ENDREP est exécutée.
Voir également l'instruction FOR/NEXT.

Exemple

```
; exemple de cycle qui actionne un axe
; entre deux cotes pendant 10 fois
Function Cyclo
    Repeat   10
      MovAbs       axe,100
      waitinput    switch,ON
      Movabs       axe,-100
      Waitinput    switch, OFF
    EndRep
Fret
```

RET

Syntaxe

RET

Arguments

aucun argument

Description

Cette instruction achève l'exécution du sous-programme avec retour à l'instruction qui suit immédiatement la CALL d'appel.
Voir également l'instruction CALL.

Remarque

Avec la CALL, cette instruction constitue l'une sources d'erreurs de programmation les plus classiques. Il est conseillé de prêter la plus grande attention à son utilisation. En particulier, il est conseillé de positionner les sous-procédures à la fin du corps d'une fonction (après l'instruction FRET), de façon à éviter l'exécution accidentelle du code de la sous-procédure comme s'il faisait partie intégrante du code principal. Le résultat de cette situation est, dans le meilleur des cas, une

erreur de système. Dans d'autres cas, on obtient un comportement anormal de la machine qu'il est difficile d'identifier.

SELECT

Syntaxe

SELECT nomevar
CASE valeur
instruction
CASE valeur1 TO valeur2
instruction
CASE IS < => valeur
instruction
CASE ELSE
instruction
ENDSELECT

instruction à remplacer par une des valeurs suivantes :
GOTO étiquette
CALL nomsousprogramme
[FCALL] nomfonction [paramètre1,...paramètreN]
[EXPR] variable = expression
[EXPR] dispositif = expression

Arguments

nomvar	constante ou variable integer ou nomcompteur
valeur, valeur1, valeur2	constantes
étiquette	nom de l'étiquette de saut
nomsousprogramme	nom du sousprogranmme
nomfonction	nom de la fonction
paramètre1...paramètreN	paramètre passé à la fonction appel
variable	nom de la variable
dispositif	nom du dispositif
expression	ensemble d'opérateurs

Description

Sélection multiple sur la base de la **valeur** de la variable **nomevar**. Le code qui se trouve dans la CASE de l'état vérifié est exécuté. La branche CASE - ELSE est exécutée si aucun cas précédent n'est satisfait. Il ne peut y avoir qu'une seule instruction GOTO, CALL, FCALL ou EXPR pour chaque CASE (facultative).
Il doit y avoir au moins une CASE entre SELECT et ENDSELECT. Cette dernière signale la fin de l'instruction SELECT.
Après chaque CALL, FCALL ou EXPR, l'exécution de la fonction se poursuit à l'instruction qui suit ENDSELECT.

Exemple

Serveur de déplacement des axes

TESTIPC

Syntaxe

TESTIPC	**nomIPC, [, nomvar1 [, nomvarN, ...]], GOTO étiquette**
TESTIPC	**nomIPC, [, nomvar1 [, nomvarN, ...]], CALL nomsousprogramme**
TESTIPC	**nomIPC, [, nomvar1 [, nomvarN, ...]], nomfonction**
TESTIPC	**nomIPC, matrice[ligne], GOTO étiquette**
TESTIPC	**nomIPC, matrice[ligne], CALL nomsousprogramme**
TESTIPC	**nomIPC, matrice[ligne], nomfonction**
TESTIPC	**nomIPC, vecteur, GOTO étiquette**
TESTIPC	**nomIPC, vecteur, CALL nomsousprogramme**
TESTIPC	**nomIPC, vecteur, nomfonction**

Arguments

nomIPC	constante chaîne. Nom de la IPC
nomvar1[..nomvarN]	constante ou variable. Noms variables 1÷N
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice
vecteur	nom de vecteur
matrice	nom de la matrice
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test et réception d'une commande IPC.
La première fois qu'une instruction de TESTIPC est exécutée, il y a allocation de mémoire partagée, dont la dimension est calculée en fonction de la dimension des données qui sont envoyées. La dimension maximale de la mémoire partagée est de 64 Ko.
Un sémaphore est associé à la mémoire partagée. Ce dernier permet de synchroniser l'exécution des tâches qui y accèdent. La tâche qui y accède s'assure qu'il y a un sémaphore actif, lit les données de la mémoire partagée et désactive le sémaphore. Ensuite, il y a un saut à instruction ou un appel de la fonction ou du sous-programme décrit en tant que dernier paramètre de l'instruction TESTIPC.
Voir également SENDIPC et WAITIPC.

TESTMAIL

Syntaxe

TESTMAIL	**mail, [nomvar1 [,..nomvarN]], GOTO étiquette**
TESTMAIL	**mail, [nomvar1 [,..nomvarN]], CALL étiquette**
TESTMAIL	**mail, [nomvar1 [,..nomvarN]], nomfonction**
TESTMAIL	**mail, matrice[ligne], GOTO étiquette**
TESTMAIL	**mail, matrice[ligne], CALL nomsousprogramme**
TESTMAIL	**mail, matrice[ligne], nomfonction**

Arguments

mail	constante ou variable integer (1÷256). Numéro du mailbox
[nomvar1[,..nomvarN]]	variable integer. Noms variables 1÷20
matrice[ligne]	constante ou variable integer. Numéro de ligne de la matrice
étiquette	nom de l'étiquette vers laquelle il faut sauter
nomsousprogramme	nom de sous-programme
nomfonction	nom de fonction

Description

Test et réception d'un message.
Le premier TESTMAIL sur la boîte aux lettres **mail** crée la boîte aux lettres.
Si le message est présent dans la boîte aux lettres **mail**, les données envoyées avec le message sont enregistrées dans les variables **nomvar** (1÷20), uniquement si ces dernières sont indiquées, ou dans la ligne de la matrice indiquée par **matrice[ligne]**. Il y a également un saut à étiquette ou un appel de **nomsousprogramme** ou de **nomfonction**.
Pendant l'exécution, la compatibilité entre les données passées et celles qui sont attendues par l'instruction est contrôlée.
Voir également les instructions SENDMAIL, WAITMAIL et ENDMAIL.

10.3.13 Divers

CLEARERRORS

Syntaxe

CLEARERRORS	**[IDposiz]**

Arguments

IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques

Description

Cette instruction demande à l'ordinateur superviseur d'éliminer toutes les erreurs de cycle, relatives au module qui exécute l'instruction, envoyées précédemment avec l'instruction ERROR. Le paramètre **IDposiz** est un paramètre facultatif qui précise la valeur numérique utilisée dans les synoptiques pour trier les erreurs de cycle dans différentes cases. Il doit correspondre à la valeur indiquée dans

le Constructeur de synoptiques pour cette case d'affichage particulière. Albatros utilise cet identificateur pour gérer les erreurs de cycle dans des queues séparées. Une queue est créée pour chaque IDposiz. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023. Si le paramètre **IDposiz** n'est pas spécifié, toutes les erreurs de cycle sont effacées et dans la queue par défaut et dans les éventuelles autres queues.
Voir également les instructions ERROR et DELERROR.

CLEARMESSAGES

Syntaxe

CLEARMESSAGES **[IDposiz]**

Arguments

IDposiz constante ou variable. C'est une valeur numérique utilisée dans les synoptiques

Description

Cette instruction demande à l'ordinateur superviseur d'éliminer tous les messages relatifs au module qui exécute l'instruction, envoyés précédemment avec l'instruction MESSAGE. Le paramètre **IDposiz est** un paramètre facultatif qui précise la valeur numérique utilisée dans les synoptiques pour trier les messages dans différentes cases. Il doit correspondre à la valeur indiquée dans le Constructeur de synoptiques pour cette case d'affichage particulière. Albatros utilise cet identificateur pour gérer les messages dans des queues séparées. Une queue est créée pour chaque IDposiz. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023. Si le paramètre **IDposiz** n'est pas spécifié, toutes les messages sont effacées et dans la queue par défaut et dans les éventuelles autres queues.
Voir également les instructions MESSAGE et DELMESSAGE.

DEFMSG

Syntaxe

DEFMSG **étiquette [, préfixelangue1], "chaînemessage" , ... , [, préfixelangueN, "chaînemessage"]**

Arguments

étiquette nom mnémonique du message à visualiser
préfixelangue constante prédéfinie. Langue dans laquelle le message est écrit
chaînemessage message à visualiser. Il doit être écrit entre doubles guillemets ("")

Description

Cette instruction attribue une **étiquette** à un message. L'instruction DEFMSG doit être déclarée avant l'implémentation des fonctions. La définition du message peut être utilisée uniquement à l'intérieur du fichier (ou groupe) dans lequel elle est déclarée. Il est possible d'insérer des messages en plusieurs langues en utilisant la constante prédéfinie **préfixelangue** (pour la liste des préfixes de langue voir le chapitre "Importation de messages"). Dans ce cas, une instruction MESSAGE affiche le message dans la langue correspondant à celle qui est alors utilisée pour Albatros. Un message auquel est associé un préfixe est utilisé lorsque la langue en cours d'utilisation ne correspond à aucun des préfixes existants.
Les étiquettes des différents langues doivent être écrit ou toutes sur la même ligne ou sur plus de lignes. Il faut changer ligne au moyen dule caractère "_" précédé d'un espace.
L'instruction DEFMSG peut être passée comme paramètre à une fonction. De cette façon la fonction qui la reçoit la peut utiliser comme un de trois arguments d'ERROR et MESSAGE. (Voir exemple 2).
Voir également les instructions MESSAGE, DELMESSAGE, ERROR, DELERROR.

Exemple 1 :

```
;attribution à une étiquette d'une chaîne de message
;sans sélection de langue
DEFMSG      MSG_GRU_1   "Message groupe 1"

;utilisation de la définition
MESSAGE     MSG_GRU_1   ;affiche: "Message groupe 1"


;attribution à une étiquette d'une chaîne de message
;avec sélection langue
DEFMSG      MSG_GRU_1   ITA "Messaggio gruppo 1"
```

```
                                ENG "Message group 1"


      ;utilisation de la définition lorsque la langue
      ;de Albatros EST ENG
      MESSAGE     MSG_GRU_1   ;affiche: "Message group  1"
```

Exemple 2:

```
      ;Dans un groupe:
      DEFMSG  MSG_TEST  "Erreur d'exécution"

      FUNCTION ChiamaTest
        Test MSG_TEST
      FRET

      ; Dans une bibliothèque :
       DEFMSG  MSG_BASE   "Signal d'erreur:  $1"
          ...
      FUNCTION Test Public
          PARAM  codice AS integer
          ERROR  MSG_BASE NOPLACE NOSTORE code
      FRET
      L'erreur de boucle visualisé est : Signal d'erreur : Erreur
  d'exécution
```

DELAY

Syntaxe

DELAY **valeur**

Arguments

valeur	constante ou variable. Temps de retard exprimé en secondes

Description

L'instruction attend que le temps indiqué par **valeur** se soit écoulé. À la fin de l'intervalle de temps l'instruction suivante est exécutée. La valeur minimum programmable est 4 msec (0,004 seconde).

DELERROR

Syntaxe

DELERROR **nomdevice [,état [, IDposiz [log]]**
DELERROR **numéro [, IDposiz [log]]**

Arguments

nomdevice	nom du dispositif
numéro	DEFMSG ou constante ou variable
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques
état	constante prédéfinie. Les valeurs pouvant être prises sont les suivantes: **ON, OFF**
log	constante numérique prédéfinie ou variable integer. Elle peut prendre les valeurs suivantes : **STORE** erreur enregistré dans le fichier, **NONSTORE** erreur non enregistré dans le fichier.

Description

Cette instruction demande à l'ordinateur superviseur d'éliminer une erreur de cycle envoyée précédemment avec l'instruction ERROR.
Si un nom de dispositif est spécifié, à la place du numéro, l'ordinateur reçoit le type et l'adresse logique du dispositif. Afin que l'élimination soit effectuée, toutes les valeurs insérées dans les paramètres doivent coïncider avec celles qui ont été utilisés pour générer l'erreur. En réglant le paramètre **log** à **STORE,** l'erreur de cycle est enregistrée dans le fichier de rapport d'erreur pour le mois en cours**.** En effet, une erreur enregistrée dans les fichiers de rapport n'est pas éliminée du fichier, mais seulement de la fenêtre des erreurs. Un nouvel enregistrement de l'élimination de l'erreur est ajouté au fichier.
Le paramètre **IDposiz** est un paramètre en option qui spécifie la valeur numérique utilisée dans les tableaux synoptiques pour distribuer les erreurs de cycle dans des différentes cases. Il doit

correspondre à la valeur spécifiée dans le Constructeur synoptique pour cette case particulière d'affichage. S'il n'est pas utile d'indiquer une case spécifique, il faut attribuer la constante prédéfinie NOPLACE. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023.
Si les erreurs de cycle ont été gérer comme signaux il sont envoyé tous les appels à effacer. Si les alarmes sont gérés comme états l'élimination des erreurs de cyclo est envoyé seulement si l'erreur de cycle était actif, sinon l'instruction DELERROR est ignoré.
Voir également les instructions ERROR, CLEARERRORS.

DELMESSAGE

Syntaxe

DELMESSAGE **numéro [, IDposiz]**

Arguments

numéro	DEFMSG ou constante ou variable
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques

Description

Cette instruction demande à l'ordinateur d'éliminer un message envoyé précédemment avec l'instruction MESSAGE.
Si les messages sont gérés comme signaux sont effacés tous les messages correspondants. Si les messages sont gérer comme états l'élimination du message est envoyé seulement si son état était actif sino l'instruction DELMESSAGGE est ignoré.
Le paramètre **IDposiz** est un paramètre en option qui spécifie la valeur numérique utilisée dans les tableaux synoptiques pour distribuer les erreurs de cycle dans des différentes cases. Il doit correspondre à la valeur spécifiée dans le Constructeur synoptique pour cette case particulière d'affichage. S'il n'est pas utile d'indiquer une case spécifique, il faut attribuer la constante prédéfinie NOPLACE. La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023.
Voir également l'instruction MESSAGE.

ERROR

Syntaxe

ERROR **nomdevice [,état [, IDposiz [, log]]]**
ERROR **numéro [, IDposiz [, log [, arg1, ..., arg3]]]**

Arguments

nomdevice	nom dispositif
numéro	DEFMSG ou constante ou variable
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques
état	constante prédéfinie. Les valeurs pouvant être prises sont les suivantes : **ON** **OFF**
log	constante numérique prédéfinie ou variable integer. Elle peut prendre les valeurs suivantes : **STORE** erreur sauvegardée sur fichier **NOSTORE** erreur non sauvegardée sur fichier
arg1, ..., arg3	constante ou dispositif ou variable

Description

Cette instruction génère une erreur de cycle. L'erreur est identifiée par le paramètre **numéro** ou par le nom d'un dispositif. Le paramètre **numéro** peut identifier une erreur de cycle de module (donc une valeur numérique entier) ou de groupe (dans ce cas on utilise une DEFMSG). Si un nom de dispositif est précisé, à la place du numéro, l'ordinateur reçoit le type et l'adresse logique du dispositif. L'erreur de cycle est envoyée à l'ordinateur superviseur et affichée sur la barre d'erreurs de Albatros.
Le paramètre **IDposiz** est utilisé dans les tableaux synoptiques pour trier les erreurs de cycle dans différentes cases. Il doit correspondre à la valeur indiquée dans le Constructeur de synoptiques pour cette case d'affichage particulière. Albatros utilise cet identificateur pour gérer les erreurs de cycle dans des queues séparées. Une queue est créée pour chaque IDposiz. Si le IDposiz n'est pas indiqué, ou que l'on utilise la constante prédéfinie NOPLACE, l'erreur de cycle va dans la queue par défaut (IDposiz=0). La plage des valeurs pouvant être configurées est comprise entre -1 (NOPLACE) et 1023.

Le réglage du paramètre **log** à **STORE** implique l'enregistrement de l'erreur de cycle dans le fichier de rapport des erreurs du mois courant. La génération d'un élevé nombre d'erreurs (ou leur suppression) peut modifier le niveau de performance des modules à distance. En effet l'ordinateur de supervision doit gérer tous les erreurs envoyées (et leur éventuelle suppression) et cela peut ralentir la transmission au contrôle de données importantes, particulièrement les programmes d'usinage.
Les paramètres en option **arg1, ..., arg3** permettent de définir des messages paramétriques. Dans la chaîne de définition du message, l'on insérera des marqueurs qui, lorsque le message sera généré, seront remplacés par la valeur ou par le nom du dispositif ou de la variable passée en tant que paramètre. Les marqueurs à introduire dans la chaîne sont :

- $1, ... $2 remplacés par le **nom** du dispositif ou de la variable ($1 correspond à arg1 etc.)
- $(1), ..., $(3) remplacés par la **valeur** du dispositif ou de la variable

Les types de données admises pour les paramètres arg1, ..., arg3 sont :

- CHAR
- INTEGER
- FLOAT
- DOUBLE (qui est toutefois converti automatiquement en FLOAT)
- numéro de message (ou étiquette de DEFMSG)
- dispositif
- variable globale ou locale
- paramètre de fonction. Il est possible d'utiliser comme paramètre de fonction aussi l'étiquette definie avec une instruction DEFMSG.

Il n'est pas possible d'utiliser paramètres les chaînes, les matrices et les vecteurs (les éléments de vecteur ou de matrice sont admis). Pour les variables locales, il est possible de décoder uniquement la valeur et pas le nom.
Pour l'élimination d'un message avec l'instruction DELERROR les paramètres arg1, ...arg3 sont ignorés.

On définit deux modalités de gestion des erreurs de cycle établies par le Constructeur de la machine:
Alarms gérés comme signaux : envoyés tous les erreurs de cycle. Albatros maintient une queue des 100 dernières erreurs de la queue indiquée et des 100 dernières erreurs de la queue par défaut.
Alarms gérés comme états : l'erreur es considerée active ou non active. Si active chaque ultérieur envoi du même erreur de cycle (par instruction ERROR) est ignoré.
Voir également les instructions DELERROR, CLEARERRORS.

Exemple 1

```
DEFMSG    ERR_TOOL    "Ousinage pas present"
DEFMSG    ERR_TOOL_P  "Charger l'outil $(1) sur la position $(2)"

; tag pour les synoptique
CONST           TOOLCHANGE = 5

; erreur visualisée sur la barre d'erreurs
; ou dans les cases de synoptique pas marquées.
ERROR           ERR_TOOL

; erreur enregistrées dans le report e visualisée
; dans les cases de synoptique marquées avec le code 5
ERROR           ERR_TOOL, TOOLCHANGE, STORE

; erreur enregistrées dans le report mais pas associée
; aux cases de synoptique
ERROR           ERR_TOOL, NOPLACE, STORE

; erreur avec paramètres
ERROR           ERR_TOOL_P, NOPLACE, NOSTORE, MxUtensili[3].Cod, 5
```

Exemple 2

```
; definie dans un groupe
DEFMSG    MSG_ERR_CARICO    "Chargement outil n'a pas été exécuté"


Function MontreMessage
    MsgTool  MSG_ERR_CARICO MxUtensili[3].Cod
fret
```

```
Function MontreErreur
  ErrTool STORE MSG_ERR_CARICO MxUtensili[3].Cod
Fret


; definie dans une librairie
DEFMSG    MSG_ERR_TOOL      "Erreur outil": $1 $(2)"


Function MsgTool public
PARAM parameter1 as integer
PARAM parameter2 as integer

       MESSAGE MSG_ERR_TOOL NOPLACE parameter1 parameter2
fret


Function ErrTool public
PARAM log as integer
PARAM argument1 as integer
PARAM argument2 as integer

  ERROR MSG_ERR_TOOL NOPLACE log argument1 argument2
fret
```

IFDEF/ELSEDEF/ENDDEF

Syntaxe

IFDEF **constante**
instruction
...
ENDDEF

IFDEF **constante, opérateur de comparaison, valeur**
instruction
...
ENDDEF

IFDEF **LINKED, nomdispositif**
instruction
...
ENDDEF

IFDEF **UNLINKED, nomdispositif**
instruction
...
ENDDEF

IFDEF **EXIST, nomgroupe**
instruction
...
ENDDEF

IFDEF **constante, opérateur de comparaison, valeur**
instruction
...
ELSEDEF
instruction
...
ENDDEF

Arguments

constante	constante type integer, char, double, chaîne
nomvar	constante type integer, char, double, chaîne
opérateur de	Les symboles qui on peut utiliser pour exécuter la comparaison sont :

comparaison	**<** (Inférieur) **=** (égal) **>** (Supérieur) **=<** (Inférieur ou égal) **>=** (Supérieur ou égal) **<>** (Différent)
valeur	constante ou nom de dispositif
nomgroupe	constante chaîne ou nom de group
nomdispositif	constante chaîne ou nom d'un dispositif

Description

La compilation conditionnelle permet de contrôler quelles sont les parties d'un fichier de fonction GPL qu'il faut compiler et exécuter. Le compilateur s'assure que la condition demandée comme argument de l'instruction IFDEF est vérifiée. Dans ce cas, le code compris entre l'instruction IFDEF et l'instruction ENDDEF ou ELSEDEF est compilé. Si l'instruction ELSEDEF est présente et que la condition n'est pas vérifiée, le code compilé est celui qui est compris entre l'instruction ELSEDEF et l'instruction ENDDEF.

La condition de compilation peut être exprimée de différentes manières :

- après l'instruction IFDEF, avec indication du nom d'une constante. Dans ce cas, la condition est vérifiée s'il existe une constante globale ou un groupe courant avec le nom indiqué.
- après l'instruction IFDEF, avec indication d'une relation entre deux opérateurs et un opérande. Le premier opérande doit être une constante. Dans ce cas, la condition est vérifiée si la relation est vraie (ex. : MAX_TOOLS = 100).
- après l'instruction IFDEF, avec indication du mot clé EXIST ou NOTEXIST suivi du nom d'un groupe de la machine ou d'une chaîne contenant le nom d'un groupe de la machine ou le nom de la bibliothèque. Dans ce cas, la condition est vérifiée si Configuration de Machine contient – ou ne contient pas – un groupe ayant le même nom.
- après l'instruction IFDEF le mot clé LINKED ou UNLINKED est spécifié, suivi par le nom d'un dispositif. Dans le cas précis, la condition est vérifiée, si le dispositif est connecté (LINKLED) ou s'il n'est pas connecté (UNLINKED) en virtuel-physique. On peut exprimer le nom du dispositif sous la forme : Nom_Groupe.Nom_Sousgroupe.Nom_Dispositif ou Nom_Groupe.Nom_Dispositif ou Nom_Sousgroupe_NomDispositif ou Nom_Dispositif. Si le dispositif n'existe pas dans la configuration, il est considéré comme non connecté.

Il est possible d'enficher plusieurs instructions IFDEF, en tenant compte du fait qu'une instruction ENDDEF doit correspondre à chaque instruction IFDEF.

Exemple 1

```
; l'exécution du code GPL change en fonction de la présence du
groupe
; FRAISE sur la machine
Const GroupFraise = "Fraise"
IFDEF Exist GroupFraise
 instruction
 instruction
ELSEDEF
 instruction
 instruction
ENDDEF
```

Exemple 2

```
; l'exécution du code GPL change en fonction du module
IFDEF _ID_MODULE = 1     ; rédige les instructions pour le module
1
 instruction
 instruction
ELSEDEF       ; rédige les instructions pour les autres modules
 instruction
 instruction
ENDDEF

; rédige le code pour la version 3.2.0 de Albatros

IFDEF _VER_MAJOR  = 3
 IFDEF _VER_MINOR = 2
       IFDEF _VER_REVISION = 0
             instruction
             instruction
```

```
        ENDDEF
  ENDDEF
ENDDEF

; rédige le code pour la version de service pack 10 Albatros
IFDEF _VER_SP  = "Service Pack 10"
 instruction
ENDDEF

; rédige le code uniquement si le système est configuré pour
; un module à distance
IFDEF _REMOTE_MODULE  = 1   ; 1 = module à distance,
sinon 0 = module local
 instruction
ENDDEF


; rédige le code uniquement si le système est configuré pour
; un module Clipper
IFDEF _REMOTE_MODULE  = 1   ; 1 = Clipper, sinon 0
 instruction
ENDDEF

; rédige le code pour la version  2.4 service pack 10 Albatros
IFDEF _VER_FULL  = $0002040AH _
 instruction
ENDDEF
```

Exemple

```
; l'exécution du code GPL change
; si le dispositif est connecté en virtuel-physique
IFDEF LINKED  out1   ; si Out1 est connecté, le code est
exécuté
 instruction
 instruction
 instruction
ENDIF
```

MESSAGE

Syntaxe

MESSAGE	**numéro [, IDposiz [, arg1, ..., arg3]]**

Arguments

numéro	DEFMSG ou constante ou variable
IDposiz	constante ou variable. C'est une valeur numérique utilisée dans les synoptiques
arg1, ..., arg3	constante ou dispositif ou variable.

Description

Cette instruction génère un message pour l'opérateur. Le paramètre **numéro** peut identifier un message de module (donc une valeur numérique entier) ou de groupe (dans ce cas on utilise une DEFMSG). En option, il est possible de passer un argument représenté par **IDposiz**. Il indique sur quelle fenêtre de synoptique le message doit être affiché. Il doit correspondre à la valeur indiquée dans le Constructeur de synoptiques pour cette case d'affichage particulière. Albatros utilise cet identificateur pour gérer les messages dans des queues séparées. Une queue est créée pour chaque IDposiz. Si IDposiz n'est pas indiqué, le message va dans la queue par défaut (IDposiz=0). La plage des valeurs pouvant être configurées est comprise entre 0 (NOPLACE) et 1023.
Les paramètres en option **arg1, ..., arg3** permettent de définir des messages paramétriques. Dans la chaîne de définition du message, l'on insérera des marqueurs qui, lorsque le message sera généré, seront remplacés par la valeur ou par le nom du dispositif ou de la variable passée en tant que paramètre. Les marqueurs à introduire dans la chaîne sont :

- $1, ... $2 remplacés par le **nom** du dispositif ou de la variable ($1 correspond à arg1 etc.)
- $(1), ..., $(3) remplacés par la **valeur** du dispositif ou de la variable

Les types de données admises pour les paramètres arg1, ..., arg3 sont :
- CHAR
- INTEGER
- FLOAT
- DOUBLE (qui est toutefois converti automatiquement en FLOAT)
- numéro de message (ou étiquette de DEFMSG)
- dispositif
- variable globale ou locale
- paramètre de fonction. Il est possible d'utiliser comme paramètre de fonction aussi l'étiquette définie avec une instruction DEFMSG

On définit deux modalités de gestion des erreurs de cycle établies par le Constructeur de la machine :
Messages gérés comme signaux : envoyés tous les messages. Albatros maintient une queue des 100 derniers messages de la queue indiquée et des 100 derniers messages de la queue par défaut. Lorsque la queue des messages est pleine, le message le plus ancien est écrasé. Si le message précédent de la queue est identique à celui que l'on est en train d'envoyer, le message n'est pas envoyé (tâche identique, numéro identique, argument identique).
Messages gérés comme états : le message est considéré actif ou non actif. Si actif chaque ultérieur envoi du même message (par instruction MESSAGE) est ignoré.

Il n'est pas possible d'utiliser paramètres les chaînes, les matrices et les vecteurs (les éléments de vecteur ou de matrice sont admis). Pour les variables locales, il est possible de décoder uniquement la valeur et pas le nom.
Pour l'élimination d'un message avec l'instruction DELMESSAGE les paramètres arg1, ...arg3 sont ignorés.

Voir également les instructions DELMESSAGE et CLEARMESSAGES.

Exemple 1

```
DEFMSG    MSG_TOOL    "Effectuer changement outil"
DEFMSG    MSG_TOOL_P  "Outil n° $(1) chargé"

; tag pour les synoptique
CONST           TOOLCHANGE = 7

; erreur visualisée sur la barre d'erreurs
; ou dans les cases de synoptique pas marquées.
MESSAGE   MSG_TOOL

; erreur enregistrées dans le report e visualisée
; dans les cases de synoptique marquées avec le code 7
MESSAGE   MSG_TOOL, TOOLCHANGE

; message avec paramètres
MESSAGE   MSG_TOOL_P, NOPLACE, MxUtensili[3].Cod
```

Exemple 2

```
; definie dans un groupe
DEFMSG    MSG_CARICO  "Changement exécuté"


Function MostraMessaggio
    MsgTool  MSG_CARICO MxUtensili[3].Cod
fret

; definie dans une librairie
DEFMSG    MSG_TOOL    "outil": $(1) $2"


Function MsgTool public
PARAM parameter1 as integer
PARAM parameter2 as integer

       MESSAGE MSG_TOOL NOPLACE parameter1 parameter2
fret
```

SYSFAULT

Syntaxe

SYSFAULT

Arguments

aucun argument

Description

Cette instruction désactive le signal de SYSOK.
Ce signal est désactivé pour indiquer que la machine n'est plus en sécurité (ex. : les tâches GPL qui gèrent les urgences ne sont plus en cours d'exécution).
Voir également l'instruction SYSOK.

SYSOK

Syntaxe

SYSOK **[nomsortie1 [, ... nomsortie8]]**

Arguments

nomsortie1 [...nomsortie8] nom dispositif type sortie numérique

Description

Il indique à la commande numérique quelles sont les sorties connectées aux circuits de la machine (il peut s'agir d'une sortie connectée à un relais de sécurité qui contrôle l'alimentation en puissance de la machine). Les sorties sont activées, quand la commande numérique a complété l'initialisation de la machine et a activé tous les tâches de gestion des urgences. La machine peut alors être considérée en condition de sécurité. On peut définir jusqu'à un nombre total de 8 sortie digitales. Dans chaque remote on ne peut activer qu'une seule sortie. La liste des sorties déclarées dans la première utilisation de l'instruction SYSOK ne peut pas être modifiée dans les éventuels appels au Sysok, jusqu'à on ha initialisé le contrôle.
Si l'on exécute cette instruction sans paramètres, le signal de SYSOK est restauré.
Voir également l'instruction SYSFAULT.

Nota

L'instruction SYSYOK peut être activée :

- dans tous le modules à distance GreenBUS v3.0 avec sorties numériques ;
- dans les modules à distance GreenBUS v4.0 de type TRS-IO ;
- dans les modules à distance TRS-CAT, seulement sur la base et non sur les extensions avec une version ne firmware 1.2 ou supérieure (rev 1.00 du module à distance).

TYPEOF

Syntaxe

TYPEOF **nom, résultat**

Arguments

nom nom de dispositif, constante, nomfonction, variable, vecteur, matrice ou ligne de matrice

résultat variable integer. Type du premier argument

Description

Cette instruction redonne à la variable **résultat** le type de l'argument **nom**.

WATCHDOG

Syntaxe

WATCHDOG **état**

Arguments

état constante prédéfinie. Elle peut prendre les valeurs suivantes : **ON et OFF.**

Description

Cette instruction active l'utilisation du watchdog connecté au matériel TMSWD. Elle permet d'identifier des situations d'erreur qui ont lieu pendant l'exécution du code GPL.
La première fois que cette instruction est exécutée avec le paramètre état à ON, l'instruction permet l'utilisation du watchdog. Toutes les fois suivantes on doit assigner ON au paramètre **état** pour mettre à jour le compteur de la platine.
Si la mise à jour n'est pas réalisée, le watchdog se déclenche et le module TMSWD désactive la sortie de décours de la machine.
Pour terminer l'usage du watchdog on doit assigner OFF au paramètre **état.**

Cette instruction ne peut être utilisée qu'avec des cartes TMSbus+, TMSCan+ e TMSCombo+ version FPGA 2.0 ou supérieure et avec le module matériel TMSWD monté.

Exemple

```
Function TestWatchDog autorun

   watchdog ON              ; active la gestion du watchdog

loop:

   watchdog ON              ; réalise la mise à jour de la carte

   goto loop

fret
```

10.3.14 MECHATROLINK-II

MECCOMMAND

Syntaxe

MECCOMMAND **axe,commande,paramètres,réponse,erreur**

Arguments

axe	noms de dispositifs type axe numérique
commande	constante integer
paramètres	vecteur de integer
réponse	vecteur de integer
erreur	variable integer. Code d'erreur

Description

Envoie à l'actionnement de l'**axe** indiqué une **commande** et il attende la réponse. Les données nécessaires pour l'exécution de la commande sont insérés dans le vecteur **paramètres**, tandis que les données retournées de l'exécution de l'instruction sont enregistrées dans le vecteur **réponse**. Les vecteurs **paramètres** et **réponse** doivent avoir la même dimension et un numéro de éléments maximum de 14. La valeur considerée est le byte plus bas de entiers uniques. Le paramètre **erreur** contient les codes des éventuelles erreurs générées pendant l'opération. Les codes de erreurs doivent être gérés par Gpl comme erreurs de cycle.
Les codes d'**erreur** rendus sont :

Code Erreur	**Message**
-40	Commande non permise dans les actuelles conditions de fonctionnement
-41	Erreur de timeout pendant l'exécution d'une commande MECHATROLINK-II
-44	Erreur de timeout pendant l'exécution d'une souscommande MECHATROLINK-II
-45	Erreur de connexion de l'actionnement

Pour les valeurs à assigner aux paramètres **commande**, **paramètres**, **réponse** et **erreur** faire référence à la documentation officielle Yaskawa MECHATROLINK-II. Ici les valeurs à assignés pour la commande sont décrits à partir de l'indice 2 jusqu'à l'indice 15. Les valeurs à assignés pour la souscommande sont décrits à partir de l'indice 18 jusq'à l'indice 32
Les valeurs de **commande** peuvent varier comme de suite par rapport à la documentation officielle:
Les commandes peuvent être distinguées en :

- commande. Elles ont code compris entre 0x00 et 0xFF. Pour raisons de sécurité elles sont exécutées seulement avec servo axe deshabilité.

- souscommande. Les commandes pris comme souscommandes doivent additionner à la valeur documentée le code 0x100. Par exemple la commande NOP a code documenté 0x00, pris comme souscommande est 0x100.
- procédure. Les commandes pris comme procédure ont code avec valeur à partir de 0x200. Actuellement sont prévues ces procédures:
    - 201H procédure de prise en charge paramètres offline (à utiliser avec axe désactivé)

Cette instruction peut être utilisée seulement avec cartes AlbMech, DualMech et DualMech Mono. Pour toute information complémentaire inhérente à l'utilisation de cette instruction, consulter TPA.

Remarque
Cette instruction agit sur le comportement de axes numériques et donc elle doit être utilisée dans un contexte contrôlé

MECGETPARAM

Syntaxe

MECGETPARAM	**axe,paramètre,dimension,donné,erreur**

Arguments

axe	noms de dispositifs type axe numérique.
paramètre	constant ou variable integer
dimension	constant ou variable integer
donné	variable integer
erreur	variable integer. Code d'erreur

Description
Lit un paramètre de l'actionnement de l'**axe** indiqué et l'enregistre dans la variabile **donné**. Le paramètre **erreur** contienne les codes des éventuelles erreurs engendrés pendant l'opération. Les codes d'erreur doivent être gérés par Gpl comme erreurs de cycle.
Les codes d'erreurs rendus sont:

Code Erreur	**Message**
-40	Commande non permise dans les actuelles conditions de fonctionnement
-41	Erreur de timeout pendant l'exécution d'une commande MECHATROLINK-II
-44	Erreur de timeout pendant l'exécution d'une souscommande MECHATROLINK-II
-45	Erreur de connexion de l'actionnement

Pour les valeurs à assigner aux variables **paramètres** et **dimension** faire référence à la documentation officielle Yaskawa MECHATROLINK-II. Cette instruction peut être utilisée seulement avec cartes AlbMech, DualMech et DualMech Mono. Pour toute information complémentaire inhérente à l'utilisation de cette instruction, consulter TPA.

MECGETSTATUS

Syntaxe

MECGETSTATUS	**axe,état,inout,erreur**

Arguments

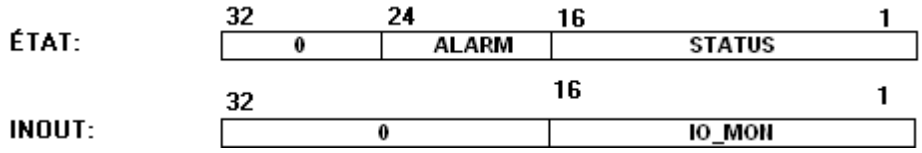
axe	noms de dispositifs type axe numérique
état	constant ou variable integer
inout	constant ou variable integer
erreur	variable integer. Code d'erreur

Description
Lit et enregistre dans la variable **état** la valeur de STATUS et de ALARM et dans la variable **inout** la valeur de IO_MON reliées à l'**axe** spécifié. Pour les valeurs de STATUS, ALARM, IO_MON faire référence à la documentation officielle Yaskawa MECHATROLINK-II.
Le paramètre **erreur** contienne les codes des éventuelles erreurs engendrés pendant l'opération. Les codes d'erreur doivent être gérés par Gpl comme erreurs de cycle.
Les codes d'erreurs rendus sont :

Code Erreur	**Message**
-40	Commande non permise dans les actuelles conditions de fonctionnement
-41	Erreur de timeout pendant l'exécution d'une commande MECHATROLINK-II
-44	Erreur de timeout pendant l'exécution d'une sous-commande MECHATROLINK-II

-45 Erreur de connexion de l'actionnement

Elle est définie une serie de catégories de erreurs. Les catégories représentent la valeur du nibble le plus haut de ALARM.
Pour sortir de alarmes qui font partie d'une de suivantes catégories 0x30,0x70,0xD0,0xF0 une commande de CLEAR (0x06) doit être envoyée. Alarms qui font partie d'une de suivantes catégories 0x00,0x10,0x40,0xB0 ne peuvent pas être éliminés avec une commande. Il faut résoudre le problème qui génère l'alarme, éteindre le servo entraînement et le rallumer.
La structure des variables **état** et **inout** est un masque de bit structurée comme dans la représentation ci-dessous :



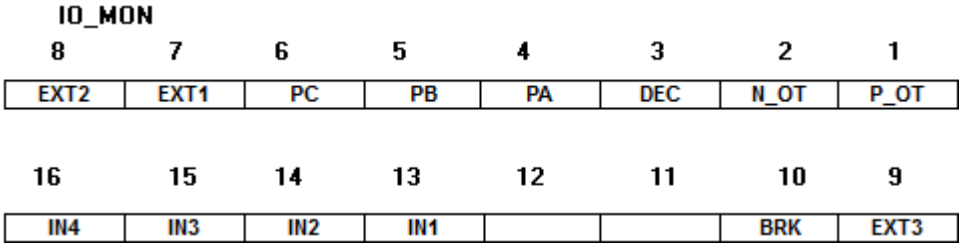
ÉTAT

8	7	6	5	4	3	2	1
PSET V_CMP	ZPOINT	Réservé	PON	SVON	CMDRDY	WARNG	ALM

16	15	14	13	12	11	10	9
Réservé	Réservé	N-SOT	P-SOT	NEAR V_LIM	L_CMP	T_LIM	DEN ZSPD

Que signifient les bits d'ÉTAT

Bit	**Commande**	**Broches physiques pouvant être reliés en Virtuel-Physique**
1	ALM (Alarm)	Entrée digitale
2	WARNG (Warning)	Entrée digitale
3	CMDRDY (Command Ready)	
4	SVON (Servo ON)	Sortie digitale
5	PON (Main Power ON)	Entrée digitale
6	Reservé	
7	ZPOINT (Zero Point)	
8	PSET (Position Complete) V_CMP (Velocity Agreement))	
9	DEN (Command Distribution Completed Flag) ZSPD (Zero Velocity)	
10	T_LIM (Torque Limit)	
11	L_CMP (Latch Completed)	
12	NEAR (Position Proximity) V_LIM (Velocity Limit))	
13	P-SOT (Forward-direction Software Limit)	
14	N-SOT (Reverse-direction Software Limit)	
15	Réservé	
16	Réservé	



Que signifient les bits d'IO_MON

Bit	Commande	Broches physiques pouvant être reliés en Virtuel-Physique
1	P_OT (Forward Over Travel)	
2	N_OT (Reverse Over Travel)	
3	DEC (Deceleration Limit Switch)	
4	PA (Phase A)	
5	PB (Phase B)	
6	PC (Phase C)	Entrée digitale
7	EXT1 (First external latch input)	Entrée digitale
8	EXT2 (Second external latch input) V_CMP (Velocity Agreement)	Entrée digitale
9	EXT3 (Third external latch input) ZSPD (Zero Velocity)	
10	BRK (Brake output)	
11		
12		
13	IN1 (General-purpose input 1)	
14	IN2 (General-purpose input 2)	
15	IN3 (General-purpose input 3)	
16	IN4 (General-purpose input 4)	

Cette instruction peut être utilisée seulement avec cartes AlbMech, DualMech et DualMech Mono.
Pour toute information complémentaire inhérente à l'utilisation de cette instruction, consulter TPA.

MECSETPARAM

Syntaxe

MECSETPARAM **axe,paramètre,dimension,donnée,erreur**

Arguments

axe noms de dispositifs type axe numérique.
paramètre constant ou variable integer
dimension constant ou variable integer
donnée variable integer
erreur variable integer. Code d'erreur

Description

Écrit une **donnée** dans le **paramètre** de l'**axe** indiqué.
Pour les valeurs à assigner aux variables **paramètres** et **dimension** faire référence à la documentation officielle Yaskawa MECHATROLINK-II. Le paramètre **erreur** contient les codes des

éventuelles erreurs engendrés pendant l'opération. Les codes d'erreur doivent être gérés par Gpl comme erreurs de cycle.
Les codes d'erreurs rendus sont :

Code Erreur	**Message**
-40	Commande non permise dans les actuelles conditions de fonctionnement
-41	Erreur de timeout pendant l'exécution d'une commande MECHATROLINK-II
-44	Erreur de timeout pendant l'exécution d'une souscommande MECHATROLINK-II
-45	Erreur de connexion de l'actionnement

Cette instruction peut être utilisée seulement avec cartes AlbMech, DualMech et DualMech Mono.
Pour toute information complémentaire inhérente à l'utilisation de cette instruction, consultez TPA.

Remarque

Cette instruction agit sur le comportement de axes numériques et donc elle doit être utilisée dans un contexte contrôlé. Pour écrire une donnée dans la mémoire pas-volatile on doit utiliser l'instruction MECCOMMAND

10.3.15 Bus de Champ Standard

AXCONTROL

Syntaxe

AXCONTROL **axe, donnée**

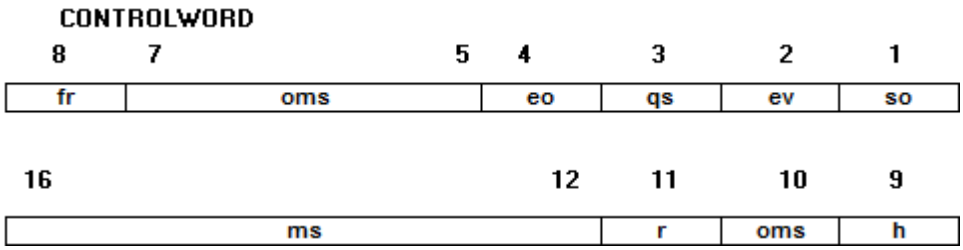
Arguments

axe nom de dispositif type axe
donnée variable ou constante integer. Il affiche la ControlWord

Description

Il affiche la **donnée** de ControlWord, d'une manière conforme au rendement actif, selon "CiA 402 CANopen device profile".

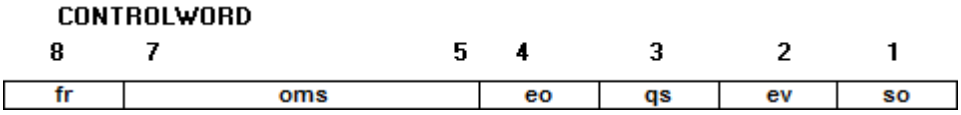
Table définition des valeurs pour EtherCAT



Bit	Signification	Nom en-virtuel physique
1	so=Switch ON	CW1
2	ev=Enable voltage	EV
3	qs=Quick stop	STOP
4	eo=Enable operation	SVON
5	oms=Operation mode specific	CW5
6	oms=Operation mode specific	CW6
7	oms=Operation mode specific	CW7
8	fr=Fault reset	RESALM
9	h=Halt	CW9
10	oms=Operation mode specific	CW10

11	r=Reserved	CW11
12	ms=Manufacturer specific	CW12
13	ms=Manufacturer specific	CW13
14	ms=Manufacturer specific	CW14
15	ms=Manufacturer specific	CW15
16	ms=Manufacturer specific	CW16

Table définition des valeurs pour S-CAN



Bit	Signification	Nom en virtuel-physique
1	Ten_cmd=commande activation torque 1: torque axe 0: axe libre	SVON
2	Ten_cmd=commande activation mouvements 1: mouvements activés 0: axe en arrêt	ENMOVE
3	Stp_cmd=commande arrêt. 1:commande arrêt active 0:commande arrêt non active	STOP
4	Alm_rst=état alertes 1:commande remise à zéro alertes	RESALM
5	Ltc_rst:reset bit 5 di StatusWord	CW5
6	oms=spécifique du mode sélectionné	CW6
7	oms=spécifique du mode sélectionné	CW7
8	oms=spécifique du mode sélectionné	CW8

AXSTATUS

Syntaxe

AXSTATUS **axe,valeur**

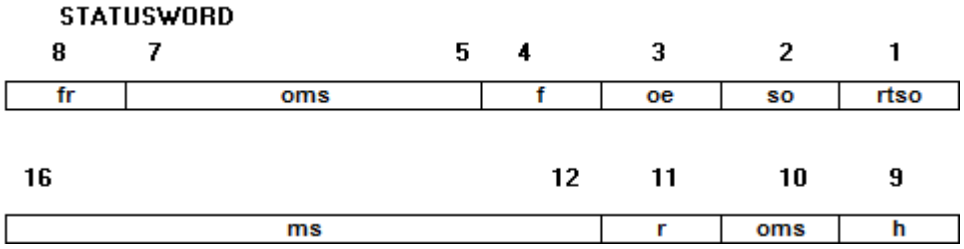
Arguments

axe nom d'un dispositif type axe
valeur variabile integer.

Description

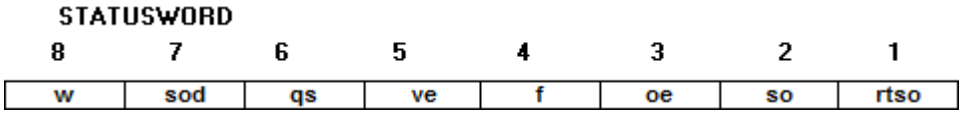
Il rend la valeur contenue dans StatusWord d'une manière conforme à "CiA 402 CANopen device profile".

Table définition des valeurs pour EtherCAT



Bit	Signification	Nom en virtuel-physique
1	rtso=Ready to switch on	RTSO
2	so=Switched on	SW2
3	oe=Operation enabled	OE
4	f=Fault	ALM
5	ve=Voltage enable	VE
6	os=Quick stop	QS
7	sod=Switch on disabled	SOD
8	w=Warning	WARNG
9	ms=Manufacturer specific	SW9
10	rm=Remote	SW10
11	tr=Target reached or reserved	SW11
12	ila=Internal limit active	SW12
13	oms=Operation mode specific	SW13
14	oms=Operation mode specific	SW14
15	ms=Manufacturer specific	SW15
16	ms=Manufacturer specific	SW16

Table définition des valeurs pour S-CAN



Bit	Signification	Nom en virtuel-physique
1	Ten_st=état activation torque 1: torque axe 0: axe libre	SW1
2	Ien_st=état activation mouvements 1: mouvements activés 0: axe en arrêt	SW2
3	Stp_st=état arrêt 1: rampe d'arrêt en cours 0:arrêt non activé ou rampe terminée	SW3
4	Alm_rst=état alertes	ALM

	1:machine en alerte 0: on n'a trouvé aucune alerte	
5	Ltc_st=état latch position 1:latch de position exécuté, registre prêt pour la lecture 0:on n'a trouvé aucun latch de position	SW5
6	oms=spécifique du mode sélectionné	SW6
7	oms=spécifique du mode sélectionné	SW7
8	oms=spécifique du mode sélectionné	SW8

CNBYDEVICE

Syntaxe

CNBYDEVICE **dispositif,carte,cn**

Arguments

dispositif nom du dispositif.
carte variable integer. Numéro de carte rendu
cn variable integer. Numéro de nœud rendu

Description

Il rend le numéro de carte et le numéro de nœud du dispositif défini dans le paramètre **dispositif**.
Cette instruction sert à utiliser les instructions, qui n'ont pas des références directes à dispositifs, par exemple les instructions READDICTIONARY et WRITEDICTIONARY.
Cette instruction peut être utilisée pour des dispositifs sur des bus CAN, S-CAN, GreenBUS ou EtherCAT.
Si le numéro de nœud renvoyé est une valeur négative, cela signifie que le nœud est désactivé.

READDICTIONARY

Syntaxe

READDICTIONARY **carte,cn,indice,sous-indice,dimdonnée, donnée,err**

Arguments

carte constante ou variable integer. Numéro de carte
cn constante ou variable integer. Numéro nœud
indice constante ou variable integer. Indice objet dans le dictionnaire
sous-indice constante ou variable integer. Sous-indice objet dans le dictionnaire
dimdonnée variable integer. Grandeur de la donnée lue
donnée variable char, integer, float,double, vecteur de char, string. Variable qui reçoive la donnée
err variable integer. Code d' erreur rendu du nœud.

Description

Il lit le contenu d' un objet dans le dictionnaire des objet contenus dans le nœud. L'instruction permet di lire au moyen du protocole SDO tous les objets définis selon "CiA 402 CANopen device profile" en plus de tous les objets rendus disponibles du Constructeur du nœud. Pour connaître la signification des paramètres **indice**, **sous-indice** et **dimdonnée** on doit se référer à "CiA 402 CANopen device profile" ou à la spécification du Constructeur du nœud. Pour les dispositifs S-CAN le paramètre sous-indice doit toujours être programmé à zéro.

WRITEDICTIONARY

Syntaxe

READDICTIONARY **carte,cn,indice,sous-indice,dimdonnée,donnée,err**

Arguments

carte constante ou variable integer. Numéro de carte

cn	constante ou variable integer. Numéro nœud
indice	constante ou variable integer. Indice objet dans le dictionnaire
sous-indice	constante ou variable integer. Sous-indice objet dans le dictionnaire
dimdonnée	constante ou variable integer. Grandeur de la donnée lue
donnée	variable char, integer, float, double,vecteur de char, string. Variable que reçoive la donnée
err	variable integer. Code d' erreur rendu du nœud

Description

Il écrit le contenu d'un objet dans le dictionnaire des objet contenus dans le nœud. L'instruction permet d' écrire au moyen du protocol SDO tous les objets définis selon "CiA 402 CANopen device profile" en plus de tous les objets rendus disponibles du Constructeur du nœud. Pour connaître la signification des paramètres **indice**, **sous-indice** et **dimdonnée** on doit se référer à "CiA 402 CANopen device profile" ou à la spécification du Constructeur du nœud. Pour les dispositifs S-CAN le paramètre sous-indice doit toujours être programmé à zéro.

10.3.16 EtherCAT

ACTIVATEMODE

Syntaxe

ACTIVATEMODE **axe,donnée,err**

Arguments

axe	nom de dispositif type axe
donnée	constante ou variable integer. Mode d' exploitation
err	integer variable. Code d'erreur rendu du servo contrôle.

Description

Il affiche le mode d'exploitation défini dans **donnée** d'une manière conforme à "CiA 402 CANopen device profile". Le mode d'exploitation de l'axe de départ correspond à la valeur de la **donnée**=9, c'est à dire "Configuration de vitesse synchrone". L'instruction rend la valeur **err=** 0, si la commande a réussi, autrement elle rend un code d'erreur.
Le tableau suivant montre des valeurs à assigner à **donnée** pour choisir Mode d'exploitation.

Value	Definition
+6	Homing mode
+9	Cyclic synchronous velocity mode

ECATGETREGISTER

Syntaxe

ECATGETREGISTER **nœud,adresse,dim, donnée,erreur**

Arguments

nœud	constante ou variable type integer. Position occupée par l'esclave dans la chaine EtherCAT (à partir d'1)
adresse	constante ou variable type integer. Adresse du registre de l'ESC (EtherCAT Slave Controller) à lire (à partir de 0)
dim	constante ou variable type integer. Numero di byte à lire (1, 2 ou 4)
donnée	constante ou variable integer. Conteneur de la donnée lue
erreur	variable integer. Code d'erreur

Description

Cette instruction renvoie **[donnée]** le contenu d'un registre de l'ESC (EtherCAT Slave Controller) du nœud EtherCAT indiqué. Le paramètre **erreur** contiendra le code numérique de l'erreur, si aucune erreur ne s'est produite.

ECATSETREGISTER

Syntaxe

ECATSETREGISTER **nœud, adresse, dim, donnée, erreur**

Arguments

nœud	constante ou variable type integer. Position occupée par l'esclave dans la chaîne EtherCAT (à partir d'1)
adresse	constante ou variable type integer. Adresse du registre de l'ESC (EtherCAT Slave Controller) à lire (à partir de 0)
dim	constante ou variable type integer. Nombre d'octets à écrire (1, 2 ou 4)
donnée	constante ou variable integer. Donnée qui doit être écrite
erreur	variable integer. Code d'erreur

Description

Cette instruction écrit le contenu [donnée] d'un registre de l'ESC (EtherCAT Slave Controller) du nœud EtherCAT indiqué. Le paramètre **erreur** contiendra le code numérique de l'erreur, 0 si aucune erreur ne s'est produite.

GETPDO

Syntaxe

GETPDO **carte, nœud,nPDO,nObj,donnée,[erreur]**

Arguments

carte	constante ou variable type integer. Numéro de carte
nœud	constante ou variable type integer. Position occupée par l'esclave dans la chaîne EtherCAT (à partir d'1)
nPDO	constante ou variable type integer. Identificateur du PDO (es. $1600h) ou sa position dans la liste de PDOs définis dans la configuration matérielle pour le nœud concerné (de 1 à 16)
nObj	constante ou variable type integer. Identificateur de l'objet (ex. $6040h) ou sa position dans la liste des objets configurés dans le PDO (de 1 à 32)
donnée	variable integer. Elle reçoit la valeur.
erreur	variable integer. Code d'erreur.

Description

Cette instruction rend en **[données]** le contenu d'un objet échangé au moyen des PDOs configurés pour le nœud EtherCAT. Si les arguments passés sont erronés et si le paramètre **erreur** n'a pas été configuré, un erreur de système survient. Si un paramètre **erreur** a été configuré, il contiendra le code numérique de l'erreur de système correspondante.

SETEOE

Syntaxe

SETEOE **état,[erreur]**

Arguments

état	constante prédéfinie. Les valeurs admises sont les suivantes: - ON active la gestion du protocole EoE - OFF désactive gestion du protocole EoE
erreur	variable integer. Code d'erreur.

Description

Cette instruction permet d'activer et de désactiver la prise en charge du protocole EoE (Ehernet over EtherCAT). Cette fonction utilise le logiciel auxiliaire SnifferCE ou SnifferRTX qui gère la communication EoE entre la commande numérique et le PC superviseur. La variable d'erreur peut prendre les valeurs suivantes :
0 si Albatros a été fermé correctement,
-1 si la commande n'est pas terminée dans le délai maximal prévu de 5/2 secondes
-2 si une erreur de communication a eu lieu.

SETPDO

Syntaxe

SETPDO **carte, nœud,nPDO,nObj,donnée,[erreur]**

Arguments

carte	constante ou variable type integer. Numéro de carte

nœud	constante ouvariable type integer. Position occupée par l'esclave dans la chaine EtherCAT (à partir d'1)
nPDO	constante ouvariable type integer. Identificateur du PDO (es. $1600h) ou sa position dans la liste de PDOs définis dans la configuration matérielle pour le nœud concerné (de 1 à 16)
nObj	constante ouvariable type integer. Identificateur de l'objet (ex. $6040h) ou sa position dans la liste des objets configurés dans le PDO (de 1 à 32)
donnée	variable integer. Valeur configurée
erreur	variable integer. Code d'erreur.

Description

Cette instruction configure le contenu **[donnée]** d'un objet échangé au moyen des PDOs configurés pour le nœud EtherCAT. Si les arguments passés sont erronés et si le paramètre **erreur** n'a pas été configuré**,** un erreur de système survient. Si un paramètre **erreur** a été configuré, il contiendra le code numérique de l'erreur de système correspondante.

10.3.17 Cartes TMSBus avec contrôle CAN

GETCNSTATE

Syntaxe

GETCNSTATE **carte, nœud, état**

Arguments

carte	constante ou variable type integer. Numéro de la carte
nœud	constante ou variable type integer. Numéro du nœud
état	constante ou variable integer.

Description

Renvoie l'état du protocole NMT pour le **nœud** de la **carte** indiquée. Pour toute information complémentaire sur la signification de ces paramètres, ainsi que sur le type et la dimension des données, voir directement la documentation du dispositif.

GETSDOERROR

Syntaxe

GETSDOSERROR **carte, erreur**

Arguments

carte	constante ou variable type integer. Numéro de la carte (de 1 à 4)
erreur	variable integer. Code d'erreur

Description

Renvoie la dernière **erreur** qui s'est produite et qui concerne la communication SDO pour la **carte** indiquée. Pour toute information complémentaire sur la signification de ces paramètres, ainsi que sur le type et la dimension des données, voir directement la documentation du dispositif.

GETMNSTATE

Syntaxe

GETMNSTATE **carte, état**

Arguments

carte	constante ou variable type integer. Numéro de la carte (de 1 à 4)
état	constante ou variable integer.

Description

Renvoie l'état du protocole NMT pour le nœud maître de la **carte** indiquée. Pour toute information complémentaire sur la signification de ces paramètres, ainsi que sur le type et la dimension des données, voir directement la documentation du dispositif.

RECEIVEPDO

Syntaxe

RECEIVEPDO **carte, nœud, numéroPDO**

Arguments

carte constante ou variable integer. numéro de la carte (de 1 à 4)
nœud constante ou variable integer. Numéro du nœud
numéroPDO constante ou variable integer. Numéro du PDO

Description

Il lit le contenu du PDO indiqué par **numéroPDO** pour le nœud spécifie. Cette instruction est utilisée pour la lecture de PDOs asynchrones (c'est à dire les PDOs qui dans la configuration matérielle ont activé l'option **Asynchrone**).
La donnée lue est copiée dans les dispositifs connectés en virtuel-physique.
On ne peut utiliser cette instruction qu'avec des cartes TMSCan et TMSCan+.

SENDPDO

Syntaxe

SENDPDO **carte, nœud, numéroPDO**

Arguments

carte constante ou variable integer. Numéro de la carte
nœud constante ou variable integer. Numéro du nœud
numéroPDO constante ou variable integer. Numéro du PDO

Description

Il écrit le contenu du PDO indiqué par **numéroPDO** pour le nœud spécifie. Cette instruction est utilisée pour l'écriture de PDOs asynchrones (c'est à dire les PDOs qui dans la configuration matérielle ont activé l'option **Asynchrone**).
On ne peut utiliser cette instruction qu'avec des cartes TMSCan et TMSCan+.

SETNMTSTATE

Syntaxe

SETMNSTATE **carte, nœud, état**

Arguments

carte constante ou variable type integer. Numéro de la carte (de 1 à 4)
nœud constante ou variable type integer. Numéro du nœud
état constante ou variable integer.

Description

Configure l'état du protocole NMT pour le **nœud** de la **carte** indiquée. Si la valeur du nœud est égal à 0 (zéro) ou supérieure à 126, la configuration est appliquée à toutes les nœuds configurés et qui se trouvent dans le canal. Pour toute information complémentaire sur la signification de ces paramètres, ainsi que sur le type et la dimension des données, voir directement la documentation du dispositif.

Valeur	État du protocole
1	Opérationnel
128	Pre-Opérationnel

10.3.18 Simulation

DISABLE

Syntaxe

DISABLE **axe1,[...axe6]**

Arguments

axe1...[...axe6] noms de dispositifs type axe

Description
Cette instruction désactive les axes indiqués. La désactivation permet d'effectuer des simulations du cycle de machine sans déplacer vraiment les axes. Un axe désactivé ne lit pas les informations qui proviennent du codeur, mais il simule une erreur de boucle proportionnelle à la vitesse courante. La désactivation de l'axe ne désactive pourtant pas la référence de vitesse. Il y a donc un voltage différent de zéro sur le connecteur de l'axe même pendant les mouvements simulés. Il est donc nécessaire de désalimenter les actionnements ou de les déconnecter de la carte des axes pendant les mouvements simulés, c'est-à-dire avec les axes désactivés. Voir également ENABLE.

Remarque
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX.

DISABLEFORCEDINPUT

Syntaxe
DISABLEFORCEDINPUT

Arguments
aucun argument

Description
Cette instruction empêche d'utiliser les fonctions pour le forcement des entrées. Si des entrées avaient été forcées précédemment, l'exécution de cette instruction les remet dans l'état réel. Voir également ENABLEFORCEDINPUT, DISABLEFORCEDINPUT, SETFORCEDINPUT, RESETFORCEDINPUT, SETFORCEDPORT, SETFORCEDANALOG.

ENABLE

Syntaxe
ENABLE **axe1,[...axe6]**

Arguments
axe1...[...axe6] noms de dispositif type axe

Description
Cette instruction active les axes indiqués. Les axes en cours d'initialisation sont toujours activés. Cette instruction ne doit être appelée que si les axes ont été précédemment désactivés par une instruction DISABLE.

Remarque
Axes de type pas-pas peuvent être utilisés dans cette instruction uniquement s'ils sont gérés par un contrôle à distance TRS-AX.

ENABLEFORCEDINPUT

Syntaxe
ENABLEFORCEDINPUT

Arguments
aucun argument

Description
Cette instruction active le forcement des entrées. Avant d'utiliser les instructions pour activer ou désactiver les dispositifs d'entrée de façon forcée, il est nécessaire d'exécuter cette instruction. Dans le cas contraire, les instructions de forcement des entrées n'entraînent aucun effet.
Voir également DISABLEFORCEDINPUT, SETFORCEDINPUT, RESETFORCEDINPUT, SETFORCEDPORT, SETFORCEDANALOG.

RESETFORCEDINPUT

Syntaxe
RESETFORCEDINPUT **nomentrée**

Arguments

nomentrée	nom entrée numérique

Description

Cette instruction met sur OFF l'entrée indiquée par **nomentrée**.
Pour pouvoir utiliser cette instruction, il est nécessaire d'avoir déjà activé le forcement des entrées, avec l'instruction ENABLEFORCEDINPUT.
Voir également DISABLEFORCEDINPUT, SETFORCEDINPUT, SETFORCEDPORT, SETFORCEDANALOG.

SETFORCEDANALOG

Syntaxe

SETFORCEDANALOG	**entréeanalogique, valeur**

Arguments

entréeanalogique	nom de dispositif entrée analogique
variable	constante ou variable integer ou float ou double

Description

Cette instruction force la **valeur** dans l'entrée analogique indiquée par **entréeanalogique**.
Pour pouvoir utiliser cette instruction, il est nécessaire d'avoir déjà activé le forcement des entrées, avec l'instruction ENABLEFORCEDINPUT. Voir également DISABLEFORCEDINPUT, SETFORCEDINPUT, RESETFORCEDINPUT, SETFORCEDPORT .

SETFORCEDINPUT

Syntaxe

SETFORCEDINPUT	**nomentrée**

Arguments

nomentrée	nom entrée numérique

Description

Cette instruction met sur ON l'entrée indiquée par **nomentrée**.
Pour pouvoir utiliser cette instruction, il est nécessaire d'avoir déjà activé le forcement des entrées, avec l'instruction ENABLEFORCEDINPUT.
Voir également DISABLEFORCEDINPUT, RESETFORCEDINPUT, SETFORCEDPORT, SETFORCEDANALOG.

SETFORCEDPORT

Syntaxe

SETFORCEDPORT	**nomport, valeur**

Arguments

nomport	nom de dispositif port d'entrée
variable	constante ou variable integer ou char

Description

Cette instruction force la **valeur** dans le port d'entrées exprimée par **nomport**. Le port d'entrées est considéré comme un masque de bit. Si un bit vaut 1, l'entrée correspondante est forcée sur "ON".
Pour pouvoir utiliser cette instruction, il est nécessaire d'avoir déjà activé le forcement des entrées, avec l'instruction ENABLEFORCEDINPUT.
Voir également DISABLEFORCEDINPUT, SETFORCEDINPUT, RESETFORCEDINPUT, SETFORCEDANALOG.

10.3.19 BlackBox

Le but de la fonctionnalité "BlackBox" est d'enregistrer dans une base de données toute l'activité d'une machine, c'est-à-dire un module local ou à distance. La "activité d'une machine" est la variation dans le tempos d'un sous-ensemble de tous les dispositifs logiques utilisables en GPL. De cette façon il est possible d'analyser par la suite le comportement de la machine, en corrélant les états des dispositifs stockés. La base de données possède une table contenant un horodotage et l'état de tous les

périphériques de cet instant, l'un pour chaque colonne. Dans la langue GPL de nouvelles instructions ont été introduites pour démarrer, terminer et interroger l'activité de l'enregistrement; elles sont décrites ci-dessous.

Chaque fichier de BlackBox est une base de données SQLite et il contient des informations concernant un module seul. Le nom du fichier inclut le numéro du module, la date e l'heure du démarrage de l'enregistrement.

L' adjonction d'enregistrements dans la base de données est effectuée de manière transactionnelle. Chaque transaction contient au plus les enregistrement générés dans 1 seconde. En cas de blackout la cohérence du fichier est garantie et la dernière transaction peut se perdre. La durée limite de la transaction peut être modifiée par une entrée in Tpa.ini (pour plus d'informations veuillez contacter TPA).

Nous avons inséré la limite de 12 heures à la durée de l'enregistrement, c'est-à-dire que chaque base de données contiendra toujours seulement les dernières 12 heures d'enregistrement, pendant l'enregistrement les enregistrements les plus anciens seront enlevés de la base de données. La durée limite de l'histoire enregistrée dans la base de données peut être modifiée par une entrée en Tpa.ini (pour plus d'informations veuillez contacter TPA).

Cette fonctionnalité est disponible pour des dispositifs physiques en bus GreenBUS, EtherCAT, CAN, S-CAN, MECHATROLINK-II, qui sont connectés par TMSbus, TMSbus+, TMScan, TMScan+, DualMech, DualMech Mono, AlbMech.

ENDBLACKBOX

Syntaxe

ENDBLACKBOX

Description

Cette instruction termine la fonctionnalité d'enregistrement dans le fichier de toute l'activité d'un module local ou à distance. Voyez aussi STARTBLACKBOX et PAUSEBLACKBOX.

PAUSEBLACKBOX

Syntaxe

PAUSEBLACKBOX

Description

Cette fonction suspend la fonctionnalité d'enregistrement dans le fichier de toute l'activité d'un module local ou à distance. Pour reprendre l'enregistrement il faut exécuter l'instruction STARTBLACKBOX sans arguments. Voyez aussi ENDBLACKBOX.

STARTBLACKBOX

Syntaxe

STARTBLACKBOX **[valeur][,erreur]**

Arguments

nœud	constante ou variable integer. Période d'enregistrement
erreur	variable integer. Code d'erreur

Description

Cette instruction active la fonctionnalité d'enregistrement dans le fichier de toute l'activité d'un module local ou à distance. L'activité d'un module est la variation dans le temps de l'état des dispositifs logiques, à l'exclusion des dispositifs du type indicateur.
La période d'enregistrement (**valeur**) est exprimée en millisecondes. Elle ne peut pas être inférieure à 10 et elle doit être un multiple de la période de real-time. Si tel n'était pas le cas, l'erreur no. 4399 se produirait (paramètre hors de la plage).
Si l'instruction démarre un enregistrement et la **valeur** est omise, la valeur considéré par défaut est 20.
Si l'instruction reprend un enregistrement interrompue précédemment, aucune **valeur** configurée n'est considérée.
S'il n'a pas été possible de faire partir l'enregistrement, **erreur** contient une valeur différente de 0, autrefois il contient 0.

Code d' erreur	Description
0	Aucune erreur
1	Il y a des différences dans la configuration des dispositifs dans le contrôle numérique et en Albatros
2	Le nombre des dispositifs à enregistrer dépasse le nombre max. prévu pour le système.
3	Aucun dispositif dans la configuration
4	Le logiciel de communication dans le module à distance ne prends pas en charge la fonctionnalité de BlackBox (seulement modules à distance)
5	Le contrôle numérique empêche le démarrage de l'enregistrement
6	Erreur lors du téléchargement de la librairie de gestion de la base de données.
7	Le nombre des colonnes pour la table dépasse le nombre max, de colonne qu'on peut gérer de la base de données.
8	Il n'a pas été possible d'ouvrir la base de données sur le disque
9	Il n'a pas été possible de créer dans la base de données la tables des enregistrements
10	Erreur dans l'adresse IP pour la communication avec le module à distance (seulement modules à distance)
11	Il n'a pas été possible de créer le socket de communication pour recevoir les données (seulement modules à distance)
12	Il n'a pas été possible d'associer une adresse locale au socket de communication (seulement modules à distance)
13	Il n'a pas été possible de se connecter au socket à distance (seulement modules à distance)
14	Il n'a pas été possible d'accéder à la zone de mémoire partagée avec le contrôle numérique
15	La configuration du matériel empêche l'utilisation de la fonctionnalité "BlackBox"
16	La fonctionnalité a été désactivée en tpa.ini

Voir aussi PAUSEBLACKBOX et ENDBLACKBOX.

10.3.20 ISO

ISOG0

Syntaxe

ISOG0 **étiquette, axe1, cote1, axe2, cote2, axe3, cote3, axe4, cote4,axe5, cote5, [valeur]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identife un bloc de déplacement. N dans le standard ISO
axe1	nom de dispositif type axe (X dans le standard ISO)
cote1	constante ou variable. Cote espace opérationnel de l'axe1
axe2	nom de dispositif type axe (Y dans le standard ISO)
cote2	constante ou variable. Cote espace opérationnel de l'axe2
axe3	nom de dispositif type axe (Z dans le standard ISO)
cote3	constante ou variable. Cote espace opérationnel de l'axe3
axe4	nom de dispositif type axe (C dans le standard ISO)
cote4	constante ou variable. Cote espace des joints de l'axe4
axe5	nom de dispositif type axe (B dans le standard ISO)
cote5	constante ou variable. Cote espace des joints de l'axe5
valeur	constante ou variable double. Elle représente le pourcentage de feed rate. (F dans le standard ISO)

Description

Il programme le mouvement rapide. Les traits de mouvement rapide sont gérés en synchronisation. Les points définis de l'utilisateur sont les extremums du trait individuel de déplacement, qui est parcouru de façon à ce que les axes soient synchronisés entre eux. Ça signifie que les axes physiques se déplacent de façon indépendante l'un de l'autre et cependant ils démarrent et arrivent dans le même temps, de la même manière que dans les instructions MULTIABS et MULTIINC. Le bec de l'outil ne parcourt pas une ligne dans l'espace opérationnel et sa trajectoire n'est pas contrôlée. Le paramètre **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'un façon univoque le bloc de déplacement. Les premières trois **cotes** identifient la position du bec dans l'espace opérationnel, tandis que les deux suivantes définissent la valeur des axes rotatifs dans

l'espace des joints. La **valeur** de FeedRate définit le pourcentage de réduction par rapport à la vitesse maximale possible (In ISO: F0 vitesse maximale, F100 FeedRate nul, donc axes arrêtés). L'instruction génère un erreur de système (4105 Instruction non exécutable sur l'axe NomAxe), si utilisés sur axes pas-pas.
Il n'est pas possible d'utiliser l'instruction WAITCOLL, car à partir de la collision on perdrait le lien d'interpolation avec les autres axes, qui contribuent au mouvement et génèrent un profil différent du celui prévu. Si cette instruction est utilisée, se génère l'erreur de système "4101 – Gestion non congruente de l'axe NomAxe".

ISOG1

Syntaxe

ISOG1 **étiquette, axe1, cote1, axe2, cote2, axe3, cote3, axe4,cote4,asse5, quota5, [valeur]**

Arguments

étiquette	constante ou variable type integer. Etiquette qui identifie un bloc de déplacement. N dans le standard ISO)
axe1	nom de dispositif type axe (X dans le standard ISO)
cote1	constante ou variable. Cote espace opérationnel de l'axe1
axe2	nom de dispositif type axe (Y dans le standard ISO)
cote2	constante ou variable. Cote espace opérationnel de l'axe2
axe3	nom de dispositif type axe (Z dans le standard ISO)
cote3	constante ou variable. Cote espace opérationnel de l'axe3
axe4	nom de dispositif type axe (C dans le standard ISO)
cote4	constante ou variable. Cote espace des joints de l'axe4
axe5	nom de dispositif type axe (B dans le standard ISO)
cote5	constante ou variable. Cote espace des joints de l'axe5
valeur	constante ou variable double. Il représente la valeur de feed rate. (F dans le standard ISO)

Description

Il définit le point dans l'espace opérationnel que la pointe de l'outil doit joindre a la fin de l'interpolation du bloc courant. Le paramètre **étiquette** est utilisé en association avec l'instruction SETLABELINTERP pour identifier d'un façon univoque le bloc de déplacement. Les premières trois **cotes** identifient la position du bec dans l'espace opérationnel, tandis que les deux suivantes définissent la valeur des axes de pivotement dans l'espace des joints. La **valeur** Feed définit la vitesse de la pointe comme unité de mesure (millimètres ou degrés) par minute (en présence de l'instruction ISOG94) comme inverse du temps d'exécution (en présence de l' instruction ISOG93). Le paramètre **valeur** est obligatoire pour la première instruction ISOG1 du mouvement d'interpolation. L'instruction génère un erreur de système (4105 Instruction non exécutable sur l'axe NomAxe), si utilisés sur axes pas-pas.
Il n'est pas possible d'utiliser l'instruction WAITCOLL, car à partir de la collision on perdrait le lien d'interpolation avec les autres axes, qui contribuent au mouvement et génèrent un profil différent du celui prévu. Si cette instruction est utilisée, se génère l'erreur de système 4101 - Gestion non congruente de l'axe NomAxe.

ISOG9

Syntaxe

ISOG9 **Axe**

Arguments

Axe	nom de dispositif type axe

Description

Il permet l'arrête forcé du mouvement. En présence de cette instruction l'interpolation ou le mouvement rapide sont arrêtés avant de passer au bloc suivant. Toutefois il ne s'agit pas d'une instruction de blocage, telle que l'instruction WAITSTILL. Le contrôle est informé sur un arrête forcé et le procédé de saisie des blocs de mouvement avance jusqu'à le remplissage du lookahead. Le paramètre **axe** identifie le canal d'interpolation 5 Axes à stopper à la fin du bloc calculé auparavant. Dans ce cas il n'y a pas de différence si on est en train d'exécuter une instruction ISOG1 ou une instruction ISOG0.

ISOG90

Syntaxe

ISOG90 **axe**

Arguments

axe nom de dispositif type axe

Description

Il établit l'interprétation des cotes comme cotes absolues. Le paramètre **axe** identifie le canal d'interpolation 5 Axes qui à partir de cette instruction interprétera les cotes des axes comme cotes absolues (condition de défaut). Dans ce cas il n'y a pas de différence si on est en train d'exécuter une instruction ISOG1 ou une instruction ISOG0.

ISOG91

Syntaxe

ISOG91 **axe**

Arguments

axe nom de dispositif type axe

Description

Il établit l'interprétation des cotes comme cotes absolues. Le paramètre **axe** identifie le canal d'interpolation 5 Axes qui à partir de cette instruction interprétera les cotes des axes comme cotes relatives. Dans ce cas il n'y a pas de différence si on est en train d'exécuter une instruction ISOG1 ou une instruction ISOG0.

ISOG93

Syntaxe

ISOG93 **axe**

Arguments

axe nom de dispositif type axe

Description

Il établit l'interprétation des vitesses comme inverse du temps d'exécution. Le paramètre **axe** identifie le canal d'interpolation 5 Axes qui à partir de cette instruction interprétera la valeur passée à travers les paramètres F de l'instruction ISOG1 comme inverse du temps d'exécution exprimé en minutes. Grâce a cela le contrôle est à même de déterminer la vitesse que la pointe de l'outil doit tenir dans les blocs d'interpolation.

ISOG94

Syntaxe

ISOG94 **axe**

Arguments

axe nom de dispositif type axe

Description

Il établit l'interpretation des vitesses comme unité de mesure par minute. Le paramètre **axe** identifie le canal d'interpolation 5 Axes qui à partir de cette instruction interprétera les vitesses comme unités de mesure par minute (condition de défaut).

ISOG216

Syntaxes

ISOG216 **nomMatriceRotatifs, nomMatricePorte-Outils, nomMatriceOutils, masqueHabilitation axe1, axe2,axe3,axe4,axe5**

Arguments

nomMatriceRotatifs	nom de la matrice. Il contient les données relatives aux axes rotatifs.
nomMatricePorte-Outils	nom de la matrice. Il contient les données relatives au porte-outils
nomMatriceOutils	nom de la matrice. Il contient les données relatives aux outils
masqueValidation	variable ou constante integer. Masque de habilitation des axes C et B
axe1	nom de dispositif type axe (X dans le standard ISO)
axe2	nom de dispositif type axe (Y dans le standard ISO)
axe3	nom de dispositif type axe (Z dans le standard ISO)
axe4	nom de dispositif type axe (C dans le standard ISO)
axe5	nom de dispositif type axe (B dans le standard ISO)

Description

Il identifie les trois matrices pour la parametrisation de la machine et le cinq dispositifs de type axe, qui composent la machine même. Telle instruction **doit** être exécutée avant toutes les autres instructions ISO. Le paramètre **masqueHabilitation** définit les axes rotatifs (C et/ouB) à valider. Pour les valeurs à programmer, il est fait référence à la table suivante :

masqueValidation	Description
31	Habilitation des axes C et B
23	Habilitation du seulement l'axe B
15	Habilitation du seulement l'axe B
7	Déshabilitation des axes C et B

Note

L'unité de mesure qui exprime, dans la configuration, les valeurs des axes rotatifs, doit être degrés.

Le lien entre les axes physiques et les axes ISO virtuels, réglé par cette instruction, est dissous lorsque la tâche est terminée, dans la quelle l'instruction est définie ou par l'instruction ISOM2. Donc, les axes peuvent être utilisés pour mouvements de type classique.

ISOG217

Syntaxe

ISOG217

axe1,axe2,axe3,axe4,axe5,axeVirtuel1,axeVirtuel2,axeVirtuel3,axeVirtuel4,axeVirtuel5.

Arguments

axe1	nom du dispositif type axe
axe2	nom du dispositif type axe
axe3	nom du dispositif type axe
axe4	nom du dispositif type axe
axe5	nom du dispositif type axe
axeVirtuel1	nom du dispositif type axe virtuel (X dans la norme ISO)
axeVirtuel2	nom du dispositif type axe virtuel (Y dans la norme ISO)
axeVirtuel3	nom du dispositif type axe virtuel (Z dans la norme ISO)
axeVirtuel4	nom du dispositif type axe virtuel (C dans la norme ISO)
axeVirtuel5	nom du dispositif type axe virtuel (B dans la norme ISO)

Description

Décrit les axes physiques et les axes virtuels, qui composent la machine Les axes virtuels décrivent la position et l'orientation de l'outil et ils doivent être déclarés de type virtuel en configuration de Albatros. Les premiers cinq axes spécifiés doivent être des axes physiques et ils sont réglés par l'interpolateur. Les cinq suivants doivent être des axes virtuels et ils sont les axes qui son utilisés dans les instructions ISOG0 et ISOG1.
Cette instruction **doit** être exécutée avant toutes les autres instructions ISO.
Les formules de cinématique directe et inverse pour passer d'une position dans l'espace des joints (axes physiques) à l'espace opérationnel (axes virtuels) doivent être spécifiées par l'instruction KINEMATICEXPR pour chacun des dix axes définis dans l'instruction ISOG217.
L'instruction génère une erreur de système (4105 Instruction non exécutable sur l'axe NomAxe), si elle est utilisée dans les axes pas-pas.

Remarque

Le lien entre les axes physiques et les axes ISO virtuels, réglé par cette instruction, est dissous lorsque la tâche est terminée, dans la quelle l'instruction est définie ou lorsque ISOM2 est exécuté. Donc, les axes peuvent être utilisés pour mouvements de type classique.

ISOM2

Syntaxe

ISOM2 **axe**

Arguments

axe nom de dispositif type axe

Description

Il libère les axes du mouvement ISO, établi par l'instruction ISOG216 ou par l'instruction ISOG217.

ISOM6

Syntaxe

ISOM6 **axe, indiceLigneMatriceRotatifs, indiceLigneMatricePorte-Outils, indiceLigneMatriceOutils**

Arguments

axe	Nom du dispositif type axe
indiceLigneMatriceRotatifs	Constante ou variable type integer. Indice de ligne de la matrice des axes rotatifs
indiceLigneMatricePorte-Outils	Constante ou variable type integer. Indice de ligne de la matrice du porte-outils
indiceLigneMatriceOutils	Constante ou variable type integer. Indice de ligne de la matrice des outils

Description

Il programme l'utilisation d'un groupe de paramètres qui décrivent la cinématique de la machine. Les **indices** se réfèrent à trois matrices dont le nom est établi de l'utilisateur et sont déclarées dans le fichier de la variable globale de Albatros. Le paramètre **axe** identifie le canal d'interpolation correspondant. Les tableaux suivants décrivent comment les trois matrices dans le fichier des variables globales doivent être déclarées :

Champs Matrice	Matrice Axes Rotatifs
Offset X	Offset le long de X entre le point fixe et le point de contrôle de la tête
Offset Y	Offset le long de Y entre le point fixe et le point de contrôle de la tête
Offset Z	Offset le long de Z entre le point fixe et le point de contrôle de la tête
Désaxement X	Ecart en X entre l'axe de rotation et celui de pointage (lorsque la position de l'axe C = 0)
Désaxement Y	Ecart en Y entre l'axe de rotation et celui de pointage (lorsque la position de l'axe C = 0)
Désaxement Z	Distance Nez-point fixe
Angle δ	Angle autour de Z pour le correct positionnement de la tête par rapport au zéro machine.
Angle γ	Angle entre l'espace de rotation et celui de pointage

Champs Matrice	Matrice Porte-Outils
OffsetPU X	Offset en X entre le point d'accrochage du porte-outil au moteur et le point d'accrochage au porte-outil (lorsque la position de l'axe C = 0 et moteur vertical)
OffsetPU Y	Offset en Y entre le point d'accrochage du porte-outil au moteur et le point d'accrochage au porte-outil (lorsque la position de l'axe C = 0 et moteur vertical)

OffsetPU Z	Offset en Z entre le point d'accrochage du porte-outil au moteur et le point d'accrochage au porte-outil (lorsque la position de l'axe C = 0 et moteur vertical)
Angle α	Angle de déphasage entre l'axe du moteur et celui de l'outil (par rapport à Z)
Angle β	Angle de déphasage entre l'axe du moteur et celui de l'outil (par rapport à Y)

Champs Matrice	Matrice Outils
Longueur outil	Longueur de l' outil

ISOSETPARAM

Syntaxe

ISOSETPARAM **NuméroIdentificationParamètre, valeur**

Arguments

NuméroIdentificationParamètre constante ou variable type integer. Il s'agit du nombre qui identifie un paramètre

Valeur constante ou variable float. Il s'agit du valeur à programmer.

Description

Il programme des paramètres, qui caractérisent la fluidité du mouvement interpolé ISO. Le tableau suivant explique la valeur de chaque **NuméroIdentificationParamètre,** les valeurs dans lesquelles la variable valeur doit être inclue et les valeurs de défaut.

NuméroIdentificationParamètre	Gamme	Défaut	Valeur
0	0.0-100.0	50.0	Pourcentage de décélération sur les axes linéaires en cas de point anguleux. (0=pas de décélération, 100=valeur maximale de décélération permise par l'interpolateur)
1	0.0-100.0	50.0	Pourcentage de décélération sur les axes rotatifs en cas de point anguleux. (0=pas de décélération, 100=valeur maximale de décélération permise par l'interpolateur)
2	0.5-1.0	0.9	Facteur de réduction vitesse en abscisse curviligne dans le cas de point anguleux (1=pas de réduction,0.5= réduction maximal)
3	0.0-100.0	60.0	Pourcentage de décélération dans le cas de discontinuités rapprochées. (0=pas de décélération, 100=valeur maximale de décélération permise par l'interpolateur)
4	0.0-100.0	10.0	Pourcentage de nombre lisse dans la trajectoire
5	(Supérieur à 0.0)-1.0	0.2	Dimension minimal du trait à parcourir avec seulement axes linéaires. La valeur est exprimée in millimètres.
6	(Supérieur à 0.0)-1.0	0.1	Dimension minimale du trait à parcourir avec seulement

			deux axes rotatifs. La valeur est exprimée en degrés.
7	0.0-100.0	100.0	Limite inférieure du filtre de smooth
8	1.0.100.0	1.0	Facteur de multiplication appliqué aux accélérations et à les décélérations définies dans la configuration. Il incrémente l'accélération et la décélération maximales des seules axes linéaires. Valeurs en dehors de la plage génèrent l'erreur de système 4399 - Paramètre hors de la plage.
9	1.0-100.0	1.0	Facteur de multiplication appliqué aux accélérations et à les décélérations définies dans la configuration. Il incrémente l'accélération et la décélération maximales des seules axes rotatifs. Valeurs en dehors de la plage génèrent l'erreur de système 4399 - Paramètre hors de la plage.
10	0.0-1.0	0.0	Flag pour activer (0.0) ou désactiver (1.0) la réduction de vitesse entre blocs consécutifs dans le cas de point anguleux. La désactivation peut également être obtenue en utilisant l'instruction avec les paramètres suivants : ISOSETPARAM 0 0.0 ISOSETPARAM 1 0.0 ISOSETPARAM 2 1.0

KINEMATICEXPR

Syntaxe

KINEMATICEXPR **axe = expression**

Arguments

axe nom du dispositif type axe physique ou virtuel
expression ensemble d' opérateurs

Description

Permet de définir les expressions individuelles de cinématique directe et inverse. Avant d' exécuter cette instruction on doit appeler l'instruction ISOG217 qui décrit les axes physiques et les axes virtuels, qui composent la machine. Pour chaque axe défini en ISOG217 on doit appeler l'instruction KINEMATICEXPR. L'expression de cinématique d'un axe dans l'espace des joints (cinématique inverse) peut être fonction de variables, constantes et des cotes des axes dans l'espace opérationnel. L'expression de cinématique d'un axe dans l'espace opérationnel (cinématique directe) peut être fonction de variables, de constantes et des cotes des axes dans l'espaces des joints.
La syntaxe de l'**expression** est la même que celle qui a été décrite dans l'instruction EXPR avec la différence qui les variables locales ne peuvent pas être utilisées. En outre, on ne peut pas utiliser des axes du même type que l'axe déclaré dans l'**axe** et non déclarés dans l'instruction ISOG217. Par exemple, lorsqu'on définit la cinématique d'un axe virtuel, qui est déjà déclaré dans l'instruction ISOG217, dans l'expression on ne peut utiliser que le cinq axe physiques, qui ont été déclarés dans la ISOG217.

Exemple

```
ut as  double ;numéro outil
offset  ;offset Y nez point d'appui
y as
double
offset  ;offset Z nez point d'appui
z as
double


Function ISO5Ax


     setval 100.ut
     setval 120.0.offsety
     setval 60.0.offsetz
     ; CINEMATIQUE EXPLICITE
     ISOG217 Rx Ry Rz Rc Rb X Y Z C B

     ; DÉFINITION DES EXPRESSIONS DE CYNÉMATIQUE
     ; CYNÉMATIQUE EXPLICITE INVERSE AXE physique Rx
     KinematicExpr Rx = X - 135 + ut * sin ( B ) * cos ( C )

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE physique Ry
     KinematicExpr Ry = Y + offsety + ut * sin ( B ) * sin ( C )

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE Rz
     KinematicExpr Rz = Z + offsetz + ut * cos ( B )

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE Rc
     KinematicExpr Rc = C

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE Rb
     KinematicExpr Rb = B

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE X
     KinematicExpr X = Rx + 135 - ut * sin ( Rb ) * cos ( C )

     ; CYNÉMATIQUE EXPLICITE INVERSE AXE Y
     KinematicExpr Y = Ry - offsety - ut * sin ( Rb ) * sin ( C )

     ; CYNÉMATIQUE EXPLICITE DIRECTE AXE Z
     KinematicExpr Z = Rz - offsetz - ut * cos ( Rb )

     ; CYNÉMATIQUE EXPLICITE DIRECTE AXE C
     KinematicExpr C = Rc

     ; CYNÉMATIQUE EXPLICITE DIRECTE AXE B
     KinematicExpr B = Rb


     ;MOUVEMENT
     ISOG0 1001,X 998.0,Y 600.0,Z 0.0,C 90.0,B 45.0,0.0
     ISOG1 1001,X 998.0,Y 600.0,Z 0.0,C 90.0,B 45.0,1000.0
     ISOG1 1001,X 998.0,Y 600.0,Z 0.0,C 90.0,B 45.0,1000.0
     ISOG1 1003,X 996.0,Y 600.0,Z 0.0,C 90.0,B 45.0,1000.0
     WAITSTILL X
     ISOM2 X
FRet
```

10.3.21 Instructions n'étant plus disponibles

INPBCD	lit une série de nibbles numériques au format BCD
OUTBCD	modifie une série de nibbles numériques au format BCD
SETFORCEDBCD	force une série de nibbles au format BCD
CANOPENDRIVER	ouvre un canal de communication CANopen
CANCLOSEDRIVER	ferme un canal de communication CANopen
CANRESETBOARD	exécute la réinitialisation d'une carte CANopen
CANSETOBJECT	écrit un objet CANopen
CANGETOBJECT	lit un objet CANopen
SLMCOMMAND	exécute une commande SLM
SLMEEPROMDISABLE	exécute une commande de désactivation à l'écriture EEPROM
SLMEEPROMENABLE	exécute une commande d'activation en écriture EEPROM
SLMGETEEPROM	lit un emplacement de mémoire EEPROM
SLMGETPARAM	lit un paramètre SLM
SLMGETREGISTER	lit un registre SLM
SLMGETSTATUS	lit une grandeur de l'actionnement
SLMSETEEPROM	écrit un emplacement de la mémoire EEPROM
SLMSETPARAM	règle un paramètre SLM
SLMSETREGISTER	règle un registre SLM
HOMING	exécute la "recherche du zéro"
SYNCROOPEN	ouvre un canal de déplacement synchronisé.
SYNCROCLOSE	ferme un canal de déplacement synchronisé
SYNCROMOVE	attribue un point de mouvement synchronisé
SYNCROSETACC	règle l'accélération pour les mouvements synchronisés
SYNCROSETDEC	règle la décélération pour les mouvements synchronisés
SYNCROSETVEL	règle la vitesse pour les mouvements synchronisés
SYNCROSETFEED	ajuste la vitesse des axes dans un mouvement synchronisé
SYNCROSTARTMOVE	commence l'élaboration d'un mouvement synchronisé
GETVF	lit la valeur du convertisseur tension/fréquence

10.3.22 Instructions inutilisables sur Interrupt

Les instructions suivantes ne peuvent pas être utilisées dans les fonctions qui sont rappelées par les instructions ONFLAG, ONINPUT et ONERRSYS. Leur utilisation est interdite même dans les tâches en real-time.

Instructions qui, à leur tour, effectuent un appel de fonction sur interrupt :

- ONFLAG
- ONINPUT
- ONERRSYS

Instructions impliquant une attente :

- WAITINPUT
- WAITFLAG
- WAITACC
- WAITCOLL
- WAITDEC
- WAITREG
- WAITTARGET
- WAITWIN
- WAITSTILL
- WAITTASK
- WAITRECEIVE
- WAITPERSISTINPUT
- MULTIWAITFLAG
- MULTIWAITINPUT

Instructions de communication :

- SEND
- RECEIVE
- CLEARRECEIVE
- COMOPEN
- COMCLOSE
- COMREAD
- COMREADSTRING
- COMWRITE
- COMWRITESTRING

- COMGETERROR
- COMCLEARRXBUFFER
- COMGETRXCOUNT

Les instructions suivantes liées au mouvement des axes :
- MOVINC
- MOVABS
- LINEARINC
- LINEARABS
- CIRCLE
- CIRCINC
- CIRCABS
- HELICINC
- HELICABS
- COORDIN
- SETRIFLOC
- RESRIFLOC
- SETPFLY
- SETPZERO
- SETINDICEINTERP
- STARTINTERP
- FASTREAD
- ENABLE
- DISABLE
- ENDMOV

Instructions pour la gestion ISO :
- ISOG0
- ISOG1
- ISOG9
- ISOG90
- ISOG91
- ISOG93
- ISOG94
- ISOG216
- ISOG217
- ISOM2
- ISOM6
- ISOSETPARAM
- KINEMATICEXPR

Instructions BlackBox:
- ENDBLACKBOX
- PAUSEBLACKBOX
- STARTBLACKBOX

Les instructions suivantes liées à la gestión EtherCAT :
- READDICTIONARY
- WRITEDICTIONARY

Instructions liées au multitâche :
- SENDMAIL
- WAITMAIL
- ENDMAIL
- SENDIPC
- WAITIPC
- TESTMAIL
- TESTIPC

Instructions impliquant une élaboration longue :
- SORT
- FIND
- FINDB
- MOVEMAT

10.4 Exemples

10.4.1 Zérotage sur Interrupt

```
;-----------------------------------------------------------------------
;  Exemple de routine de zérotage éclair
;
;  La fonction exécute les opérations suivantes :
;
;     1) Règle l'axe en désactivant les limites de logiciel
;        et en remettant la cote à zéro
;     2) S'assure que le capteur n'est pas déjà en état ON.
;        S'il est en état ON, déplace l'axe et attend qu'il se
;        remette en état OFF. Si cela n'a pas lieu dans 30
;        secondes, génère un message d'erreur.
;     3) Règle la vitesse de recherche du capteur
;     4) Lance un mouvement de l'axe et active le zérotage
;        éclair pour cet axe. Au déclenchement de l'interrupt,
;        la cote de l'axe est mise à zéro et un mouvement à une cote de
;        dégagement part automatiquement.
;     5) Attend l'arrivée de l'axe à la cote de dégagement
;     6) Rétablit les limites de l'axe
;
;   © TPA.
;-----------------------------------------------------------------------
Function Fast_Zérotage

      ResLimPos   axe                          ; Initialisation Axe
      ResLimNeg   axe
      SetQuote    axe,0

      IfInput     FastInput,OFF,Goto Continue  ; Test Capteur occupé
      SetVel      axe,5                        ; Règle vitesse de
                                               ; dégagement
      MovAbs      axe,30                       ; déplace l'axe
      WaitInput   FastInput,OFF,30,Call Error  ; Contrôle dégagement
                                               ; micro,
                                               ; erreur après TimeOut=30

     EndMov       axe                          ; Stop axe
     WaitStill    axe                          ; Attente axe arrêté


Continue:
     SetVel       axe,10                       ; Vitesse de recherche
                                               capteur de zérotage
     MovAbs       axe,-1000                    ; Mouvement en négatif de
                                               recherche capteur
                  axe,ON,10,0                  ; Enclenchement Interrupt
                                               ; et réglage de la cote et
                                               de la vitesse de
                                               ; dégagement
     WaitStill    axe                          ; Attente axe arrêté


     SetLimPos    axe                          ; Rétablissement des
                                               limites de l'axe
     SetLimNeg    axe

; sous-procédure d'envoi de message d'erreur
Error:


     Error        ERR_SETP          ; Message d'erreur impossible de
                                    poursuivre
     Ret
```

10.4.2 Serveur de déplacement des axes

```
;-------------------------------------------------------------
; Exemple d'un serveur de déplacement des axes :
;
;  Le serveur déplace les axes de la machine
;  pour le compte d'autres tâches.
;
;  Les tâches client envoient les commandes sous la forme de
;  messages (mail) à une boîte aux lettres.
;
;  Le serveur prélève les commandes de la boîte aux lettres et
;  les exécute.
;
;  Les demandes sont mises à la queue dans la boîte. Donc,
;  si une demande arrive pendant que le serveur est occupé,
;  elle n'est pas perdue et elle sera exécutée dès que possible.
;
;  Le serveur est la seule tâche à déplacer les axes. Tout
;  conflit est donc ainsi évité.
;
;  Le serveur est implémenté par la fonction Master_axes.
;
;  Un exemple de client est implémenté par la fonction Check_flag.
;  Cette fonction contrôle régulièrement l'état d'un flag et
;  lorsqu'elle le trouve en état ON, elle envoie au serveur
;  la commande d'exécution de zérotage des axes.
;  Le flag sera vraisemblablement réglé à la Main sur ON
;  par l'opérateur qui utilisera, par exemple, un tableau synoptique.
;
;-------------------------------------------------------------

;-----------------------------------
; -- CONSTANTES GLOBALES DE MACHINE --
;-----------------------------------
Const MBOX = 101     ; identifie la boîte aux lettres
                     ; commandes
Const SETP = 10      ; zérotage des axes
Const CHG  = 11      ; changement d'outil
Const Perçage = 12   ; exécution d'un perçage


;---------------------
; --- GROUPE AXES ---
;---------------------

; définition des messages d'erreur
Defmsg ERR_CMD "Commande groupe axes inconnu"

; --- Serveur ---
Function Master_axes autorun

   local cmd as integer                        ; commande
   local cote_X as double                      ; cote X trou
   local cote_Y as double                      ; cote Y trou

loop:
   waitmail MBOX,cmd,cote_X,cote_Y             ; attente commande

   ; La commande étant reçue, l'action opportune
   ; est identifiée et exécutée
   Select cmd
```

```
   case SETP
            fcall Zérotage_axes              ; Zérotage des axes
   case CHG
            fcall changement d'outil         ; Exécute un changement
                                             ; d'outil
   case FORO
            fcall perçage cote_X,cote_Y      ; perçage aux cotes indiquées
   case else
            call erreur
   endselect

   endmail MBOX                              ; notification de l'exécution

                                             ; de la commande
   goto loop                                 ; retour en attente d'une
                                             ; nouvelle commande


   fret

; sous-procédure d'envoi d'un message d'erreur
erreur:
   error ERR_CMD
   ret


;-------------------------
; --- GENERIQUE GROUPE ---
;-------------------------

; --- Client ---
Function Check_flag

loop:

   ifflag Setp_aaxes,OFF, goto loop          ; test état flag

   ; OK le flag est sur ON, envoi de la commande
   sendmail MBOX,WAITTACK,SETP,0.0,0.0

   resetflag Setp_axes                       ; réinitialisation du flag

   goto loop                                 ; retour en attente

   fret

   ; NOTER QUE :
   ; - Après la commande "SETP", il est nécessaire d'indiquer
   ;   les deux paramètres "cote_X" et "cote_Y" même si cela
   ;   n'a pas de sens pour l'opération de zérotage.
   ;   Toutefois, le serveur ignore a priori quelle commande
   ;   lui sera envoyée. Il faut donc indiquer deux valeurs
   ;   du même type que celles que le serveur s'attend à recevoir,
   ;   dans ce cas deux DOUBLE. Les valeurs passées sont "0.0" et "0.0".
   ; - Le paramètre "WAITACK" fait en sorte que le client attende
   ;   l'exécution de la commande de la part du Serveur.
   ;   Le client ne poursuit son exécution que lorsque le Serveur
```

```
;   a exécuté un ENDMAIL ou qu'il a commencé à traiter une nouvelle
;   commande (WAITMAIL).
```

10.4.3 Cycle de Main avec gestion des erreurs

```
;-------------------------------------------------------------
; Fonction principale hypothétique
; initialise la machine et exécute un cycle de contrôle
;-------------------------------------------------------------
Function Main
      OnErrSys GestErrSys           ; active la gestion des erreurs

      StartTask   Urgences          ; initialise
      StartTask   Calculateur
      ActiveAxes

loop:
      IfFlag Flag,OFF, RétablissementUrgences
      .........
      Goto loop
Fret


;-----------------------------------
; fonction de gestion des erreurs
;-----------------------------------
Function GestErrSys
      Param nerreur as integer
      Param tâche as function
      Param typedevice as device

       EndT Task                ; Achève tâche Calculateur
      ask
       If   nerreur, >, 5, goto ; Les 5 premières erreurs
            noerraxis
                                ; sont relatives
                                ; aux axes

       Rese Flag
tFlag
DésactiveA
xes

noerraxis:
Fret
```

10.4.4 Opérations sur les chaînes

```
;-----------------------------------------------------------------
; Exemple de manipulation des chaînes
;-----------------------------------------------------------------
Function Exemple
  Local       chaîne1 as string
  Local       chaîne2 as string
  Local       chaîne3 as string
  Local       longueur as integer
  Local       position as integer

  SetString   "chaîne d'",chaîne1        ; chaîne1 contient
                                         ; maintenant "Chaîne d'"
  SetString   " essai",chaîne2

  AddString   chaîne1,chaîne2,chaîne3    ; chaîne3 contient
```

```
                                        ; "Chaîne d'essai"

  Search      chaîne3,'h',position      ; position vaut 2
  Search      chaîne3,'Z',position      ; position vaut -1

  Left        chaîne3,7,chaîne1         ; chaîne1
                                        ; contient "Chaîne"

  Right       chaîne3,2,chaîne2         ; chaîne2
                                        ; contient "ai"

  Mid         chaîne3,10,2,chaîne3      ; chaîne3
                                        ; contient "es"

  ControlChar 65,chaîne1                ; chaîne1
                                        ; contient "A"

  Len         chaîne3,longueur          ; longueur vaut 2

  Str         longueur,chaîne3          ; chaîne3
                                        ; contient "2"

  Val         position,chaîne1          ; chaîne1
                                        ; contient "-1"

  AddString   "Le résultat est ",chaîne1,chaîne2

  ; chaîne2 contient "Le résultat est -1"

Fret
```

10.4.5 Exécution Séquentielle / Parallèle

```
;-----------------------------------------------
; Exemple de routine qui contrôle le zérotage
; d'une machine à 3 axes en évitant d'éventuelles
; interférences mécaniques.
;
; Les zérotages des différents axes sont
; implémentés par des fonctions dont le texte est omis.
; Voir l'exemple "Routine de zérotage".
;
; D'abord, exécution uniquement du zérotage
; de l'axe Z (qui est censé ne pas pouvoir être
; effectué en même temps que les autres).
; Ensuite, les zérotages des axes
; X et Y sont exécutés en même temps.
;-----------------------------------------------

; message pour l'opérateur (traduit en langue)
DefMsg      MSG_SETP    ITA   "Azzeramento assi in corso ..."
                        FRA   "Zérotage axes en cours ..."

Function ZérotageAxes

  Message     MSG_SETP              ; informe l'opérateur

  Fcall       ZérotageaxeZ          ; zérotage de l'axe Z

                                    ; OK le zérotage axe Z est terminé

  StartTask   ZérotageAxeX          ; lance le zérotage X et Y
  StartTask   ZérotageAxeY
```

```
  WaitTask    ZérotageAxeX          ; attend la fin
  WaitTask    ZérotageAxeY

  DelMessage  MSG_SETP              ; élimine le message
                                    ; pour l'opérateur
Fret
```

10.4.6 Routine de Zérotage

```
;-----------------------------------------------------------------
;  Exemple de routine de zérotage d'un axe
;
;  La fonction exécute les opérations suivantes :
;    1) désactive les limites de logiciel de l'axe
;    2) règle la vitesse de recherche du switch
;    3) déplace l'axe à une cote incrémentielle qui
;       garantit que le switch soit atteint
;    4) attend que l'axe fasse déclencher le switch
;    5) arrête l'axe et attend la fin du mouvement
;    6) règle la vitesse (basse) de dégagement du switch
;    7) fait reculer l'axe de juste ce qu'il faut
;       pour dégager le switch
;    8) attend le dégagement du switch
;    9) attribue la nouvelle cote à l'axe
;   10) rétablit la vitesse par défaut et les limites de logiciel
;
;   © TPA.
;-----------------------------------------------------------------

Function Zérotage

   ResLimPos   axe          ; désactive les limites de logiciel
   ResLimNeg   axe

   SetVel      axe,10       ; règle la vitesse

   MovInc      axe,10000    ; déplace l'axe

   WaitInput   Switch,ON    ; attente du switch

   EndMov      axe          ; arrête l'axe
   waitStill   axe          ; attente de l'axe arrêté

   SetVel      axe, 0.1     ; règle la vitesse de dégagement

   MovInc      axe,-100     ; déplace l'axe

   WaitInput   Switch,OFF   ; attente de dégagement du switch

   SetQuote    axe,0        ; attribue la nouvelle cote

   SetVel      axe          ; rétablissement de la vitesse
   SetLimpos   axe          ; rétablissement des limites de logiciel
   SetLimneg   axe

Fret
```

10.4.7 Mouvements ISO

```
-----------------------------------------------------------------

; Exemple de mouvement ISO
;
```

```
; Un profil est généré avec les instructions ISOG0 et ISOG1
;
;   © TPA.
;-------------------------------------------------------------------
0150
;  Déclaration des matrices ISO
;  Matrice axes rotatifs
MxRot[5] as double:off_X double:off_Y double:off_Z double:dis_X double: _
dis_Y double:dis_Z double:delta double:gamma
;  Matrice porte-outil
MxPorte[1] as double:off_X double:off_Y double:off_Z double: _
alpha double:beta
;  Matrice outils
MxOutils[10] as double:ut double

FonctionInterpolationISO

   ; réglage des valeurs standard de parametrisation machine
   setval 90.0 MxRot[5].gamma
   setval 260.3 MxOutils[10].ut
   setval MxOutils[10].ut ut

   ; réglage des paramétres de l'algorithme
   IsosetParam 0 50
   IsosetParam 1 50
   IsosetParam 2 0.9
   IsosetParam 3 60
   IsosetParam 4 30

   ; réglage de la machine : Il déclare les trois matrices utilisées
   ; par la parametrisation de la machine et les axes physiques1458
   ; utilisés dans le mouvements ISO.
   isoG216 MxRot MxPorte MxOutils 31 X Y Z C B ; CINEMATIQUE IMPLICITE

   ; réglage du groupe de paramètres qui décrivent la cinématique
   ; de la machine.
   isoM6 X 5 1 10 ; CINEMATIQUE IMPLICITE

   ; réglage des valeurs initials
   setquote x 500
   setquote y 300
   setquote z 0
   setquote c 0
   setquote b 0
   setvel x
   setvel y
   setvel z
   setvel c
   setvel b
   setveli x y z c b

   ; exécution du profil
   isoG0 1001,X 998.0,Y 600.0,Z 0.0,C 90.0,B 45.0,50.0
   isoG1 1001,X 998.0,Y 600.0,Z 0.0,C 90.0,B 45.0,10000.0
   isoG1 1003,X 996.0,Y 600.0,Z 0.0,C 90.0,B 45.0,10000.0
   isoG1 1002,X 600.0,Y 600.0,Z 0.0,C 90.0,B 45.0,10000.0
   isoG1 1004,X 599.131759111665,Y 599.924038765061,Z 0,C 100,B
   45.0,10000.0
   isoG1 1006,X 598.289899283372,Y 599.69846310393,Z 0,C 110,B 45.0,10000.0
   isoG1 1005,X 597.5,Y 599.330127018922,Z 0,C 120,B 45.0,10000.0
   isoG1 1003,X 596.786061951567,Y 598.830222215595,Z 0,C 130,B
   45.0,10000.0
   isoG1 1002,X 596.169777784405,Y 598.213938048433,Z 0,C 140,B
   45.0,10000.0
   isoG1 1012,X 595.669872981078,Y 597.5,Z 0,C 150,B 45.0,10000.0
   isoG1 1011,X 595.301536896071,Y 596.710100716628,Z 0,C 160,B
   45.0,10000.0
```

```
    isoG1 1031,X 595.075961234939,Y 595.868240888335,Z 0,C 170,B
    45.0,10000.0
    isoG1 1102,X 595.0,Y 0.0,Z 0.0,C 180.0,B 45.0,10000.0
    waitstill X Y Z C B
fret
```

Tecnologie e Prodotti per l'Automazione S.r.l.
Via Carducci 221
I - 20099 Sesto S.Giovanni (MI)
Tel. +39 02.36527550
Fax. +39 02.2481008
www.tpaspa.com

February 2021