STNACPINT-F

INTRODUCTION AU CONTRÔLE D'AXES B&R

Référence : **STNACPINT-F**
Formation ACPINT

Version : **0.47x.0**
FL/PM/SP/ZR/OR-10/03

DÉBUT DU SÉMINAIRE

1. INTRODUCTION

Généralités

Notre séminaire de formation Introduction au contrôle d'axe B&R vous permettra de découvrir les produits B&R et plus particulièrement la variation de vitesse, la programmation ainsi que les applications possibles.

Pendant ce séminaire, vous travaillerez conjointement avec votre formateur.

Certains d'entre vous ont déjà une grande expérience, d'autres découvrent un nouveau domaine.

Nous nous efforcerons de donner à chacun le maximum d'informations.

La réussite de ce séminaire ne dépend pas seulement des compétences techniques. L'implication personnelle de chacun et la coopération entre les participants et le formateur jouent un rôle déterminant.

Présentation du formateur

Votre formateur va se présenter. N'hésitez pas à prendre quelques notes.

_____

_____

_____

_____

Présentation des participants

Au cours de ce séminaire, vous allez travailler en petits groupes. Il est donc intéressant de connaître les autres participants (nom, société, produits, domaine d'activité, niveau de connaissances).

2. EMPLOI DU TEMPS

L'emploi du temps constitue un facteur important dans le déroulement d'un séminaire.

Début : _____

Déjeuner : _____

Fin : _____

Nous ferons de temps en temps une petite pause pour satisfaire les inconditionnels buveurs de café et de thé.

3. PROGRAMME

- Présentation générale des systèmes
- Chaîne d'entraînement
- Dimensionnement de l'entraînement
- Concept NC
- Motion Components
- Système ACOPOS
- Mise en service
- Exemple d'application

PRÉSENTATION GÉNÉRALE DES SYSTÈMES

1. SOLUTIONS B&R POUR LE CONTRÔLE D'AXES

Aujourd'hui, presque toutes les machines ou installations à automatiser doivent exécuter des tâches de positionnement plus ou moins complexes.

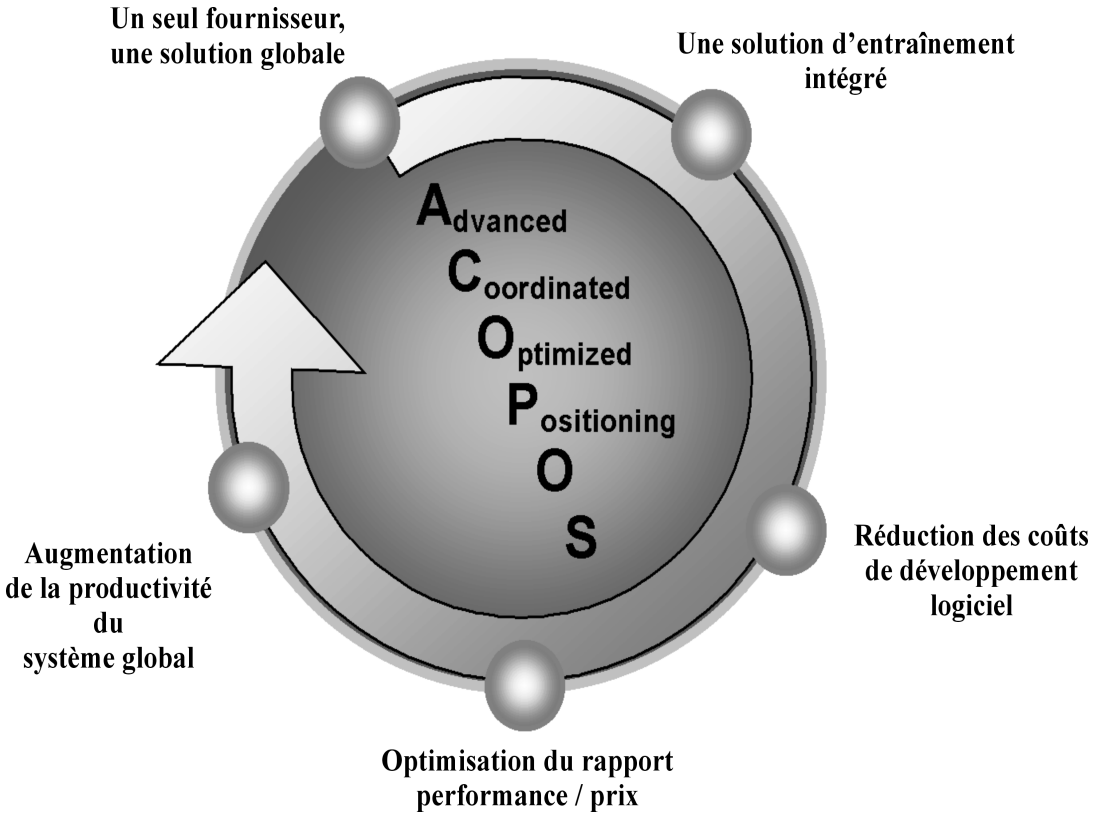
Désormais, de nombreux mouvements mécaniques doivent être exécutés à l'aide de systèmes d'entraînement électriques hautement dynamiques, faciles à installer et ne nécessitant aucune maintenance.

Pour optimiser les performances de la machine, il convient de mettre en œuvre des solutions d'entraînement de haute technicité avec un **partenaire compétent et expérimenté**.

Depuis plus de 20 ans, B&R offre des **solutions de positionnement** pour tous les secteurs d'activité et convainc avec ses **fonctions technologiques**.

Le système global et novateur développé par B&R intègre désormais toute une **gamme de produits** comprenant des variateurs, des moteurs et des codeurs.

2. EXIGENCES - OBJECTIF - SOLUTION



Tab. 2.1.: **Exigences - Objectif - Solution**

En grec, AKOPOS signifie "simple et sans efforts"

2.1 Un seul fournisseur pour une solution globale

Avec B&R, vous disposez d'un partenaire et d'un fournisseur à la hauteur de vos exigences en matière d'automatisation et de contrôle d'axes.

Un responsable de projet se chargera de répondre à vos questions dans les domaines suivants :

- **mécanique**
  Quel est le moteur, le codeur ou le réducteur le mieux adapté à mon application ?

- **électrique**
  Quelle classe de puissance faut-il choisir pour les variateurs ? Quels câbles dois-je utiliser et où puis-je me les procurer ?

- **logiciel**
  Comment dois-je concevoir mon système de régulation ? Comment faut-il organiser et programmer le contrôle du processus ?

Avec un **interlocuteur unique**, vos tâches administratives peuvent être regroupées et minimisées.

De plus, les éléments constituant la solution B&R sont parfaitement harmonisés entre eux. Ainsi, vous disposez d'une **solution globale et complète**.

2.2 Solution d'entraînement intégrée

B&R offre une **gamme complète de produits** permettant de satisfaire vos besoins d'automatisation.

La gamme de produits ACOPOS couvre toute la partie **Motion** relative au contrôle d'axes.

Les composants Motion font **totalement partie intégrante** du concept d'automatisation B&R, ce qui permet une harmonisation optimale avec les composants **Control** et **Panel** (contrôle et visualisation).

Cette solution d'entraînement s'inscrit totalement dans la tendance actuelle à la **décentralisation** et à l'encapsulation de parties de machine et d'installation, avec des liens définis entre les différents sous-ensembles.

2.3 Réduction des coûts de développement logiciel

La réalisation des tâches de contrôle et d'entraînement se déroule en trois étapes :

- **Création et configuration**
  des composants matériels et logiciels
- **Programmation**
  des tâches système et de l'applicatif
- **Mise en service et maintenance**
  des axes et de l'applicatif

Pour réduire les besoins de développement et créer un **cadre propice à l'innovation**, B&R offre un **outil global** pour l'utilisation des systèmes d'automatisation - **Automation Studio™**.

2.4 Optimisation du rapport prix-performance

La compétitivité de votre machine ou de votre installation passe par une optimisation de son rapport prix-performance. A cet effet, B&R applique les règles suivantes :

- **Architecture système optimisée**
  grâce à des composants parfaitement complémentaires
- **Décentralisation des composants**
  grâce à une structure compacte et modulaire
- **Analyse du processus et dimensionnement des composants**
  Suivi et assistance assurées par un ingénieur B&R
- **Développement et programmation**
  un seul outil – une solution globale – temps de formation réduits
- **Mise en service / Optimisation**
  Tous les outils de diagnostic sont regroupés dans un seul atelier logiciel - support sur place

2.5 Augmentation de la productivité du système global

ACOPOS, votre gage de réussite grâce à

- **une densité de puissance élevée,**
- **des cadences d'horloge élevées,**
- **une qualité supérieure,**

et ce pour l'ensemble des applications monoaxes et multiaxes, avec en plus

- **des fonctions technologiques,**
- **une interface intuitive graphique**

et l'avantage que constitue

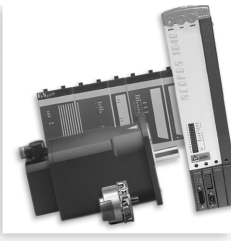
- **la grande expérience acquise par B&R.**

La solution offerte par B&R s'appelle ACOPOS !

3. SYSTÈMES B&R DE CONTRÔLE D'AXES

CHAÎNE D'ENTRAÎNEMENT

1. ELEMENTS DE LA CHAÎNE D'ENTRAÎNEMENT



Le caractère optimal de la chaîne d'entraînement dépend de la qualité de chacun de ses éléments (ces derniers figurent dans le schéma qui suit).

Il est donc important de connaître la structure des différents éléments ainsi que leurs variantes.

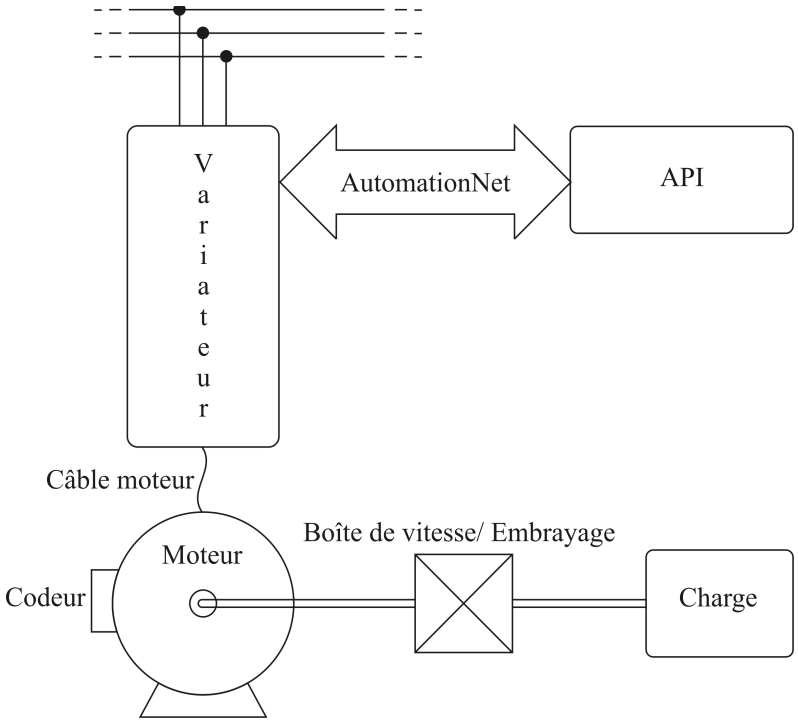


Fig. 3.1 : **Eléments d'une chaîne d'entraînement**

Dans les pages suivantes, vous trouverez des explications générales sur les différentes variantes et découvrirez les spécificités des éléments d'entraînement de B&R.

2. MOTEUR

2.1 Aperçu général

Convertissant l'énergie électrique en énergie cinétique (ou de rotation), le moteur constitue un des éléments essentiels de la chaîne d'entraînement.

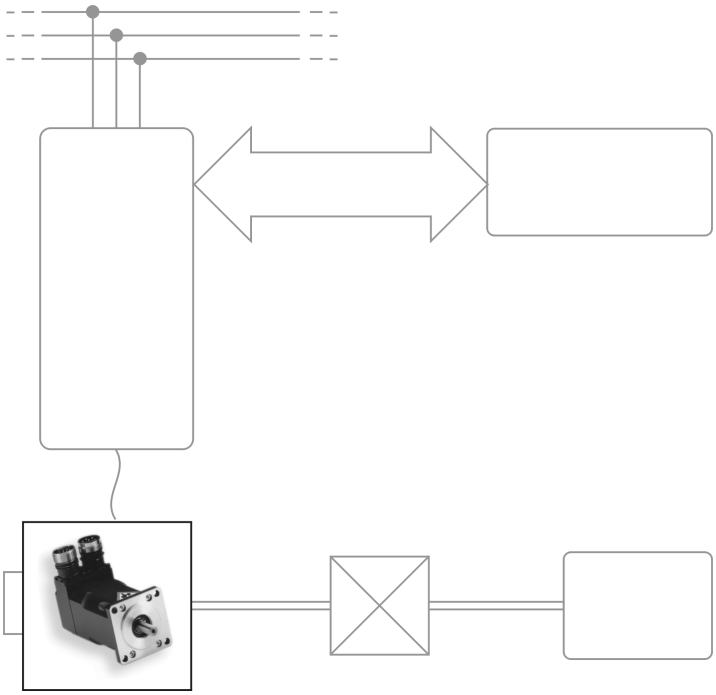


Fig. 3.2 : **Eléments d'une chaîne d'entraînement - Moteur**

2.1.1 Types de moteurs

Depuis que le développement des convertisseurs d'énergie électromécaniques a commencé au début du 19ième siècle, trois types de moteurs se sont imposés. Ces derniers se différencient par leur structure et leur mode de fonctionnement.

- les **moteurs à courant continu** (abréviation dans la suite : MDC)
- les **moteurs synchrones** (abréviation dans la suite : MS)
- les **moteurs asynchrones** (abréviation dans la suite : MAS)

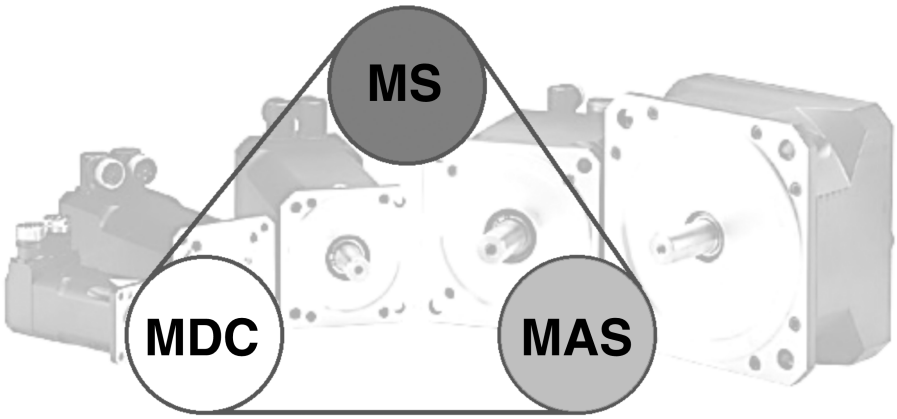


Fig. 3.3 : **Types de moteurs**

Ces trois types de bases se déclinent en de multiples variantes : moteur pas à pas, moteur à réluctance, moteur linéaire...

Quel est le moteur le plus approprié pour mon application ? Pour répondre à cette question, il faut définir précisément les performances requises en matière de positionnement, sachant que la plupart des applications exigent une dynamique très importante.

Autrefois, pour les entraînements asservis, les MDC étaient choisis en priorité en raison de la simplicité de régulation. Aujourd'hui, à une époque où des processeurs haute performance sont capables d'exécuter des algorithmes de régulation élaborés, les MS et MAS connaissent une progression soutenue.

Les avantages des MS et MAS résident surtout dans le fait qu'ils ne nécessitent aucune maintenance (absence de commutation mécanique, meilleures caractéristiques de refroidissement et plus grande robustesse).

Ce sont les MS qui ont les meilleures propriétés dynamiques.

Comparaison des différents types de moteurs

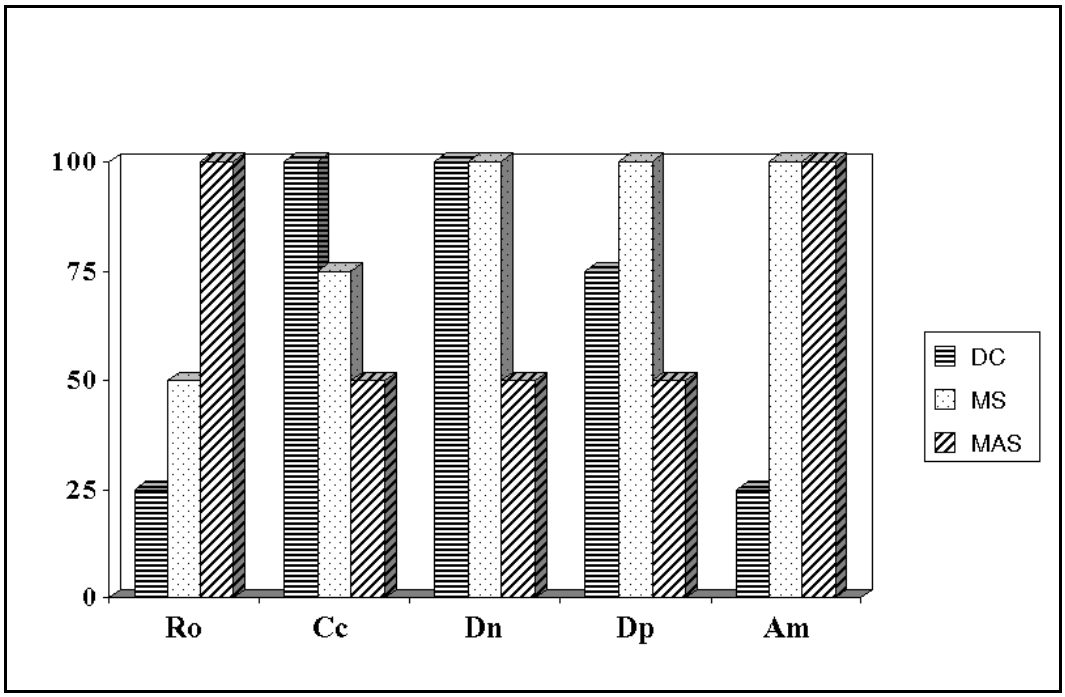


Fig. 3.4 : **Comparaison des différents types de moteurs**

MS..... Moteur synchrone
MAS.... Moteur asynchrone
DC..... Moteur DC

Ro..... Robustesse
Dn..... Dynamique
Cc..... Complexité de l'électronique de commande
Dp..... Densité de puissance (rapport puissance/taille)
Am ..... Absence de maintenance

2.1.2 Structure des moteurs synchrones

Les principaux éléments de ces moteurs à champ tournant et à pôles internes (stator + rotor) figurent dans le schéma ci-dessous :

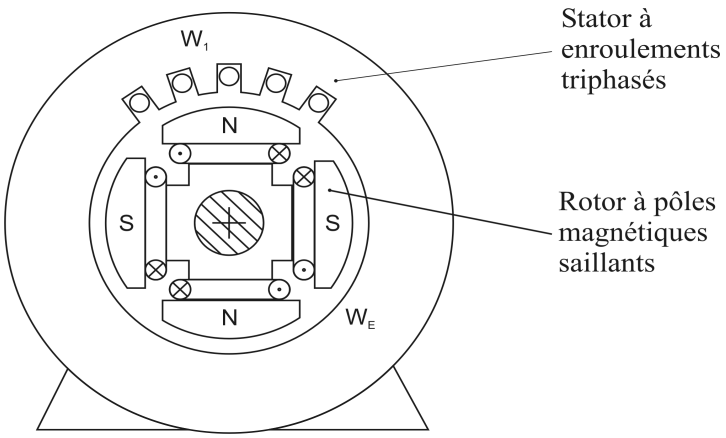


Fig. 3.5 : **Structure des moteurs synchrones**

Le couplage des enroulements du stator au réseau triphasé est en étoile. Connectés à un réseau triphasé, les enroulements du stator génèrent un champ tournant.

Dans le domaine des puissances moyennes, le rotor est généralement constitué d'aimants permanents de qualité supérieure permettant de générer un champ rotorique lié à la position du rotor.

2.1.3 Principe de fonctionnement des moteurs synchrones

La rotation d'un moteur électrique est basée sur le principe physique suivant : un conducteur traversé par un courant et se trouvant dans un champ magnétique est soumis à une force mécanique.

La règle de la main gauche et la formule qui suit permettent de déterminer la direction et l'intensité de cette force :

$$\vec{F} = (\vec{l} \times \vec{B}) \cdot I$$

F ..... Vecteur force
B ..... Vecteur induction
l ..... Vecteur du conducteur dans le champ
I ..... Courant

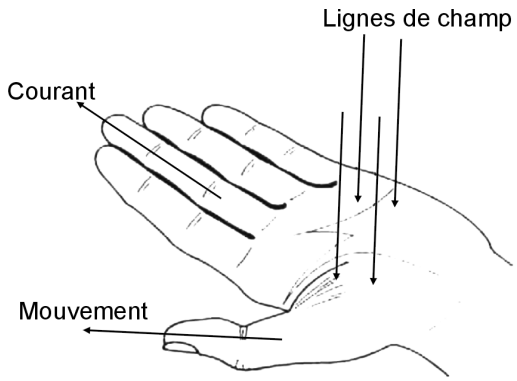


Fig. 3.6 : **Relation entre le courant, le mouvement et les lignes de champ**

La force exercée sur le conducteur à une distance r par rapport au centre géométrique (axe du moteur) génère un **couple C** selon la formule

$$C = 2 \cdot F \cdot r \cdot \sin\alpha$$

Ce couple est **fonction de l'angle alpha** entre le vecteur force et le rayon r.

On atteint le couple maximal lorsque $\alpha = 90°$ (voir schéma ci-dessous).

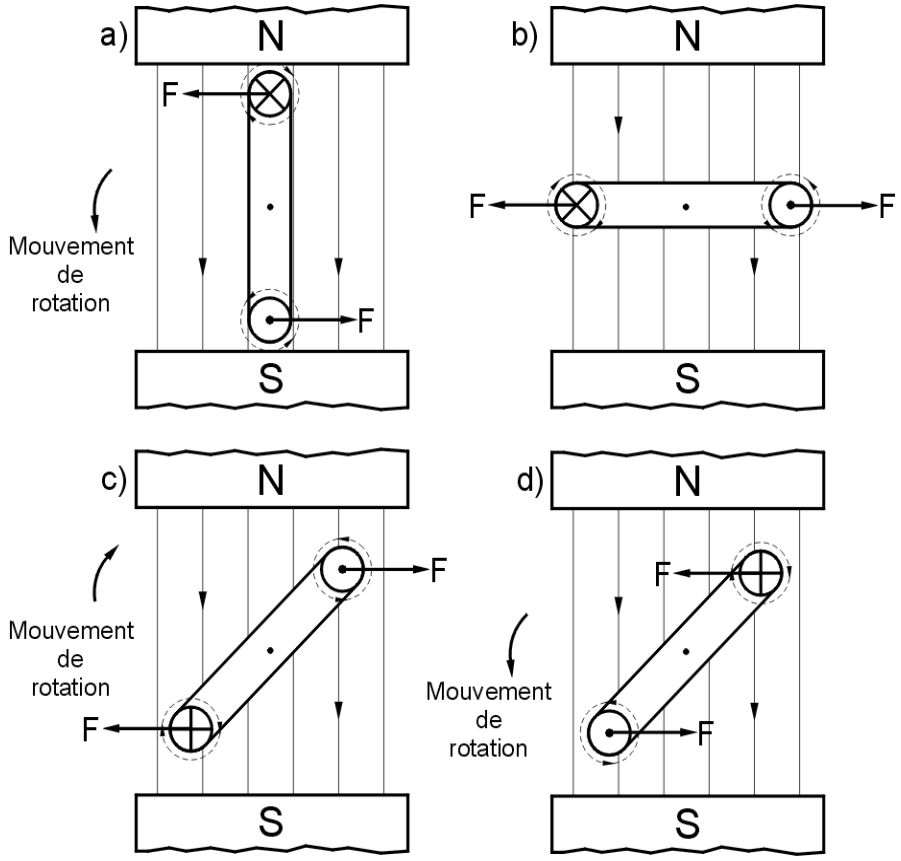


Fig. 3.7 : **Couple et angle**

2.1.4 De la commutation mécanique à la commutation électrique

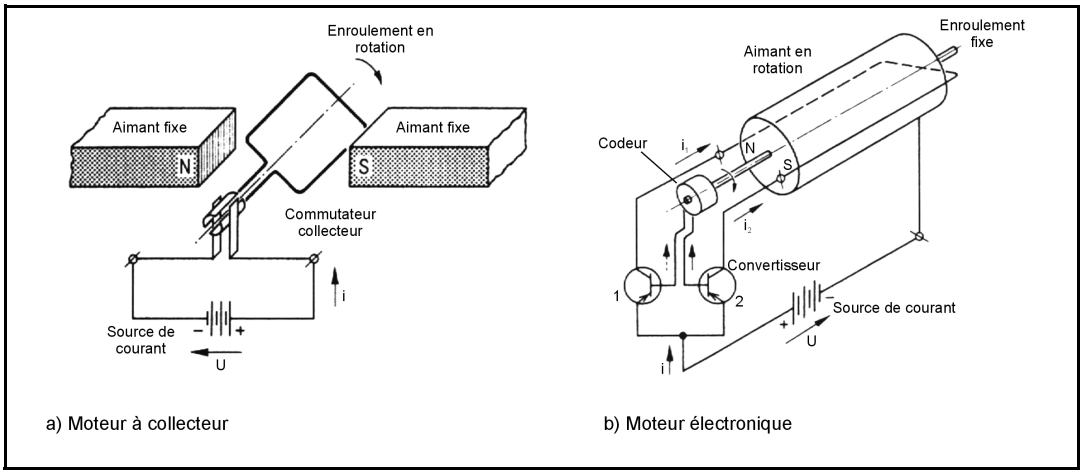


Fig. 3.8 : **Commutation mécanique / électrique**

a) moteur à collecteurs = moteur DC (MDC)
b) moteur électronique = moteur synchrone (MS)

Le rôle de la **commutation** est d'assurer qu'il y a toujours un enroulement conducteur traversé par un courant dans le champ d'excitation.

Une disposition appropriée des collecteurs, des balais et des points de raccordement des conducteurs électriques permet de réaliser la commutation sur un moteur DC. L'entrée de courant se fait ainsi par l'intermédiaire d'une commutation mécanique, comme on peut le voir sur la figure en-haut à gauche.

Le moteur à collecteurs a pour inconvénient l'usure des éléments mécaniques de commutation et l'entretien qui s'ensuit.

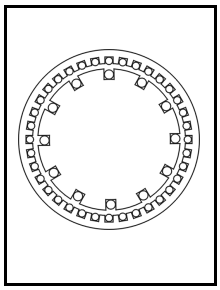
L'avantage du MS réside dans le fait qu'il ne nécessite aucune maintenance, la commutation étant purement électrique. D'où son nom "**moteur DC brushless**". La relation entre les différentes grandeurs électromagnétiques est ici inversée : le champ d'excitation est produit grâce aux aimants permanents situés sur le rotor, alors que le champ créant le couple est généré par les enroulements du stator commandés par l'électronique de puissance. Ainsi, le MS ne comporte pas de pièces mécaniques sujettes à l'usure.

Pour les variateurs dynamiques, cette **commutation électrique** est réalisée grâce à une électronique de puissance appropriée.

2.1.5 Structure et principe de fonctionnement du moteur asynchrone

Du point de vue du stator, le MAS est une machine à champ tournant avec enroulement triphasé.

Le stator du MAS se distingue de celui du MS par l'absence d'excitation continue. Le champ d'excitation est produit indirectement par le champ tournant créé au niveau du stator.



À l'arrêt, le moteur MAS agit comme un **transformateur triphasé** : conformément à la loi de Lenz, une tension est induite par le champ tournant dans les barres du rotor. Cette tension produit un courant électrique dans les barres conductrices en court-circuit (**rotor en court-circuit**). Grâce au champ du stator, le conducteur traversé par un courant est alors soumis à une force mettant le rotor en mouvement.

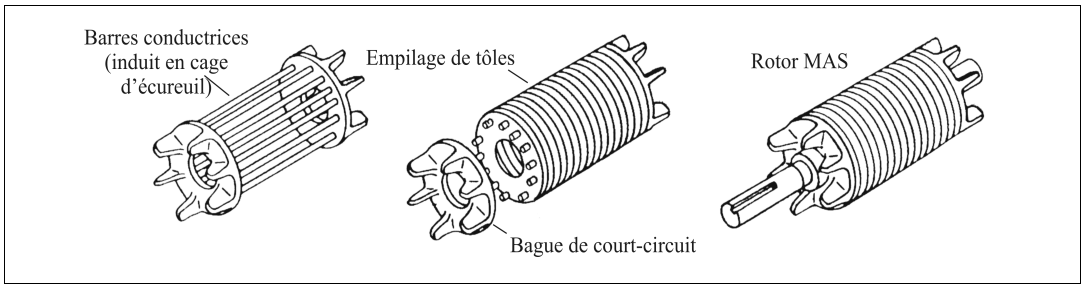


Fig. 3.9 : **Structure d'un moteur asynchrone**

Après la phase de démarrage, le rotor finit par atteindre une vitesse de rotation légèrement inférieure à celle du champ tournant. Cette différence de vitesse est nécessaire car elle engendre une **vitesse relative** permettant d'induire dans le rotor un courant suffisant pour vaincre les frottements, la résistance de l'air ou le couple résistant.

Comme le rotor ne peut jamais atteindre la vitesse du champ tournant, le mouvement est dit **asynchrone**.

La commande des MAS, tout comme celle des MS, est une commande à contrôle vectoriel, basée sur l'orientation des champs. Néanmoins, dans le cas des MAS,

les deux champs ϕ_Q et ϕ_D doivent être générés au niveau du stator et pris en compte dans la régulation.

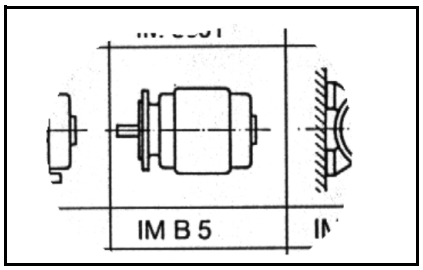
2.2 Highlights de B&R

B&R offre des moteurs de qualité se prêtant bien aux processus exigeant dynamique et précision :

- Densité de puissance élevée
- Aucune maintenance
- Possibilités d'adaptation étendues
- Très bonne concentricité

2.2.1 Réalisation des moteurs

La structure compacte des moteurs B&R (de 0,14 à 32 Nm) est de type B5.



- 2 flasques
- sans embase
- arbre lisse
- brides d'extension

La taille du moteur (diamètre et longueur) est fonction de la puissance et de la vitesse requise.

$$P_i = k \cdot D^2 \cdot l \cdot n$$

P_i ..... puissance interne
k ..... constante de construction
D..... diamètre de l'induit
l ..... longueur de l'induit
n ..... vitesse

Ainsi, vous pouvez choisir des formes de moteur, c'est-à-dire **le diamètre de la bride et la longueur**, en fonction de l'espace disponible.

Tous les moteurs ont une protection IP65 avec/sans **joint d'étanchéité**.

La **plaque signalétique électronique** embarquée du moteur permet une mise en service simple et efficace.

Remarque

Les références des moteurs ainsi que les différentes options sont exposées en détail dans le catalogue.

2.2.2 Caractéristiques des moteurs

Les moteurs B&R se prêtent particulièrement bien à la réalisation de mouvements hautement dynamiques grâce à leur **haute densité de puissance** (puissance/volume). Leur concentricité optimale garantit un positionnement extrêmement précis.

La surface extérieure lisse du moteur n'est **pas salissante** et permet ainsi un **refroidissement** par rayonnement thermique **ne nécessitant aucune maintenance**.

Un codeur résolver, un **frein de maintien** intégré et des vitesses nominales spéciales sont disponibles en option.

La sécurité de fonctionnement est assurée par de multiples **protections contre les surcharges**, même dans des conditions extrêmes.

La **mesure analogique de la température de l'enroulement** (max. 145 °C) permet une modélisation précise de la température. Les moteurs peuvent ainsi être exploités jusqu'aux limites de leurs capacités.

Les **câbles codeur/moteur** que nous choisissons sont de grande qualité. Nous les **préconfectionnons** avec des longueurs adaptées. Si la longueur de câble dépasse 5 m, les réflexions se produisant aux extrémités de câbles peuvent générer des tensions jusqu'à deux fois supérieures à la tension du bus DC. Des filtres appropriés sont prévus pour protéger le moteur.

Les longueurs de câble standard sont les suivantes : 5 m, 10 m, 15 m, 20 m, 25 m.

Il est également possible de choisir un moteur en fonction du **bout d'arbre**, ce dernier pouvant être lisse ou **rainuré** (pour clavette).

2.2.3 Option : le frein

En option, les moteurs B&R peuvent être équipés de freins à aimants permanents.

L'enclenchement du frein se fait électriquement via le 24 VDC issu des circuits de commande de la partie puissance et prélevé par l'électronique de régulation. Le bon fonctionnement du frein n'est donc assuré que si le niveau requis pour la tension de commande est respecté !

Tous les **retards** à l'**activation** ou à la **désactivation** des freins sont lus par l'électronique de commande **via l'ENDAT** et pris en compte.

La surveillance du **circuit de freinage** est **assurée** en interne.

Le frein a été conçu comme un **frein de maintien**. Il ne peut donc pas être utilisé pour réaliser un freinage permanent de mouvement actif.

IMPORTANT Le frein ne doit pas être utilisé comme un moyen de protection des personnes !!!

3. CODEURS

3.1 Aperçu général

Le retour des valeurs réelles de vitesse et de position via un codeur approprié joue un rôle essentiel dans tout système asservi.

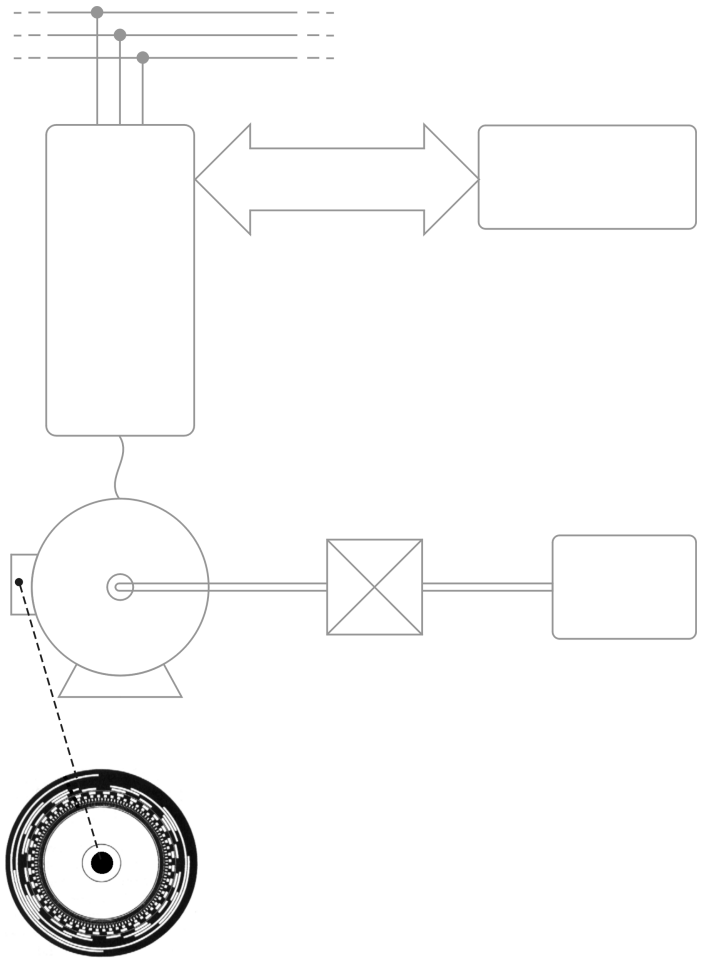


Fig. 3.10 : **Eléments d'une chaîne d'entraînement - Codeur**

Différents systèmes peuvent être choisis en fonction de l'application souhaitée :

- **codeur incrémental** optique
- **codeur absolu** optique
- codeur à induction **résolver**

Comparaison des différents types de codeurs

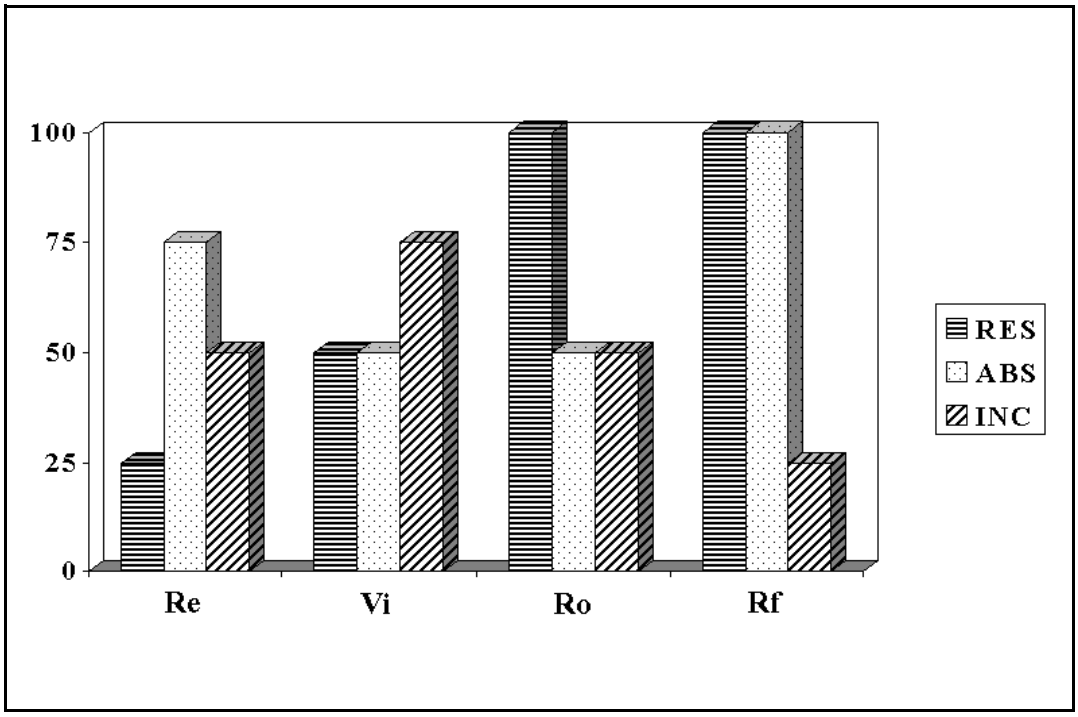


Fig. 3.11 : **Comparaison des différents types de codeurs**

RES ..... Résolver
ABS..... Codeur absolu optique
INC ..... Codeur incrémental optique (lecture de signal par circuit TTL)

Re ..... Résolution
Vi ..... Vitesse de transmission du signal
Ro..... Robustesse
Rf ..... Prise de référence (temps nécessaire)

Selon l'application, on utilise des codeurs linéaires ou des codeurs rotatifs.

3.1.1 Codeur incrémental optique

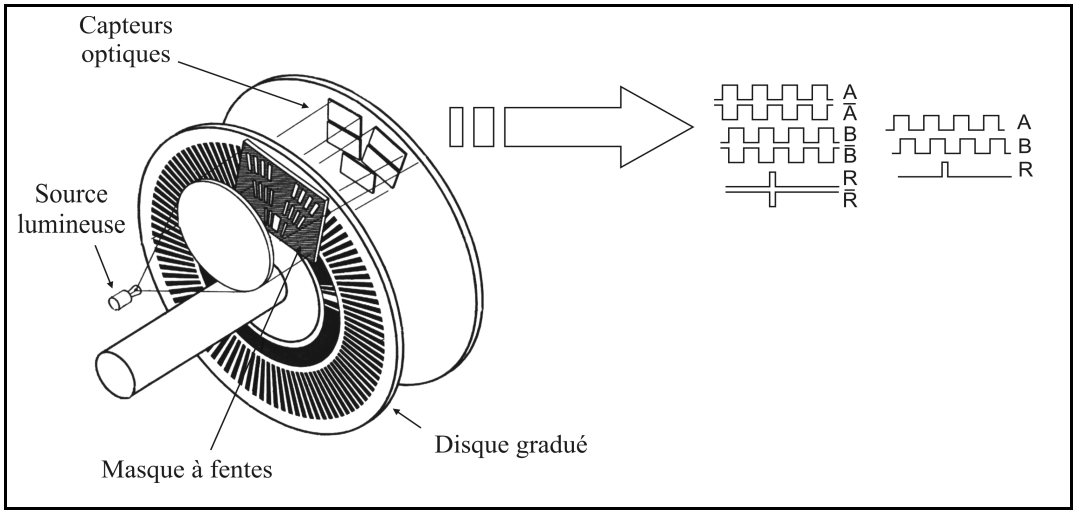


Fig. 3.12 : **Codeur incrémental optique**

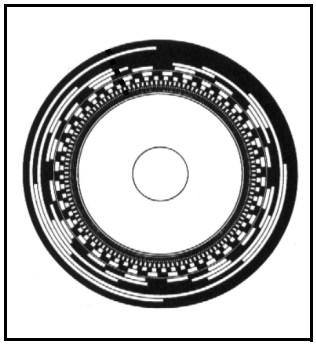
Un signal sinusoïdal issu de la superposition d'un masque à fentes et d'un disque gradué (sinus et cosinus déphasé de 90°) est transformé en signaux de codeur **rectangulaires**. Ces derniers sont ensuite utilisés par la logique d'évaluation au niveau de l'électronique de commande pour augmenter/diminuer une valeur de compteur de position.

Pour pouvoir établir une relation entre la valeur du compteur et la position actuelle, il faut effectuer une **prise de référence**.

La résolution dépend du nombre de barres, du type de mesure (1, 2 ou 4 fronts) et de la fréquence d'entrée maximale de la logique d'évaluation.

3.1.2 Codeur absolu optique

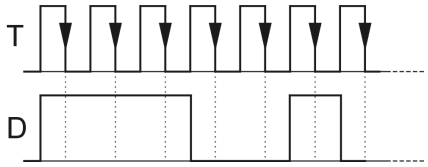
Dans un codeur absolu, chaque position correspond à une valeur unique. La résolution est donnée en bits, chaque bit correspondant à une piste sur le disque du codeur.



Si la plage de déplacement de l'axe correspond à une rotation du codeur de moins d'un tour, on utilise un codeur **simple tour**.

Si la plage de déplacement de l'axe correspond à une rotation de plus d'un tour, on utilise des codeurs **multitours** (codeur avec démultiplicateur pour pistes/ bits supplémentaires indiquant le nombre de tours).

La transmission du signal à la logique d'évaluation se fait avec le protocole SSI (interface synchrone-série).



Synchrone : les données de position sont transmises de façon synchrone par rapport à un signal d'horloge

Série : les données de position sont transmises les unes à la suite des autres, avec un débit approprié.

La relation entre la valeur du codeur et la position est toujours définie. Il n'est donc pas nécessaire d'effectuer une prise de référence.

Selon le disque utilisé, le code délivré par un codeur absolu peut être du code **binaire pur** ou du code **Gray**.

3.1.3 Codeur à induction

Les applications militaires ont nécessité le développement d'un codeur très **robuste** et de **construction simple** : le **résolver**.

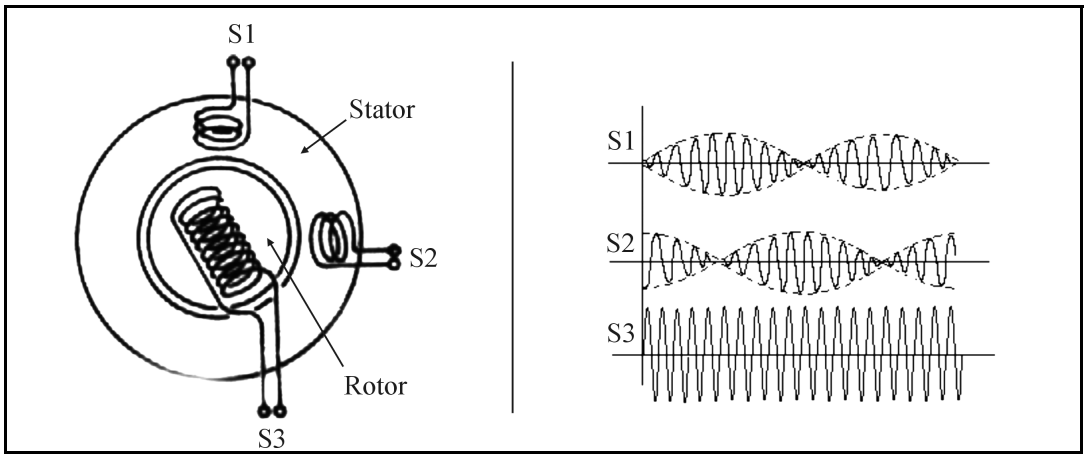


Fig. 3.13 : **Résolver**

La génération du signal codeur s'opère grâce à l'entrée d'un signal sinusoïdal de fréquence constante dans une bobine rotorique. Les tensions S1, S2 sur les bobines du stator placées à 90° sont générées par induction, suivant le **principe du transformateur**. Si le rotor tourne, on obtient une enveloppe sinusoïdale (sinus et cosinus).

Si la plage de déplacement de l'axe correspond à une rotation de moins d'un tour, on peut assigner une position unique à chaque valeur délivrée par le codeur. **Aucune prise de référence** n'est alors nécessaire.

La résolution dépend de la logique d'évaluation et de la fréquence du signal d'entrée sur la bobine du rotor.

3.2 Highlights de B&R

B&R offre en complément de ses moteurs différents systèmes de codeurs pour satisfaire les exigences les plus diverses :

- Codeur sinusoïdal optique **ENDAT**

3.2.1 ENDAT - Codeur sinusoïdal optique

Avec l'interface **EN**coder**DAT**a, B&R combine les avantages du codeur incrémental et du codeur absolu et compense aussi leurs inconvénients.

- **Codeur incrémental**
  Les avantages de ce codeur résident dans la transmission rapide du signal et dans la haute résolution grâce à l'évaluation sinusoïdale. Prise de référence nécessaire.
- **Codeur absolu**
  Le lien permanent existant entre la position logicielle et la position machine permet de s'affranchir de toute prise de référence. Limitée en débit, la transmission des données est unidirectionnelle et sérielle.

ENDAT = Combinaison d'un codeur incrémental et d'un codeur absolu :

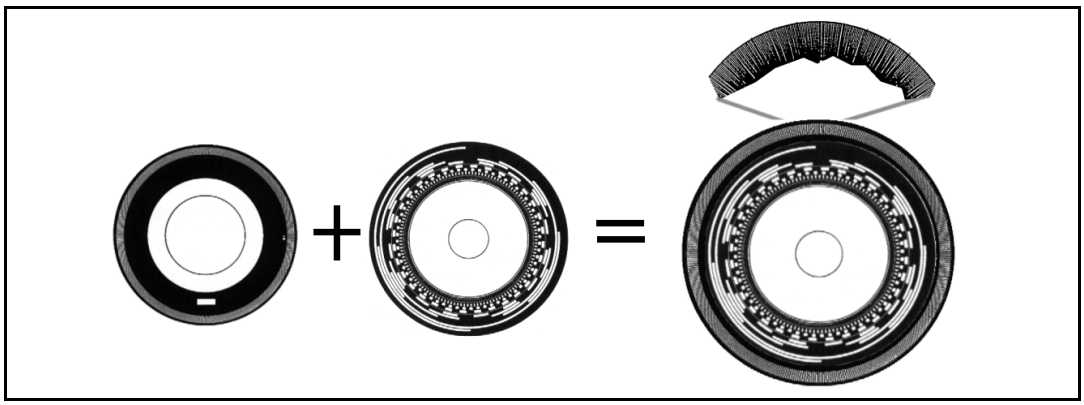


Fig. 3.14 : **ENDAT - Combinaison d'un codeur incrémental et d'un codeur absolu**

Lors de la mise sous tension de la machine, la position est lue à partir du disque absolu. Si la plage du codeur est plus grande que celle du déplacement de l'axe, il s'agit d'un système absolu. Aucune prise de référence ne doit alors être effectuée.

Pour permettre une commande hautement dynamique, le système ENDAT délivre un **signal sinus et cosinus par trait d'incrémentation**. A partir de ce signal, la logique qui suit (électronique de régulation) génère une résolution standard de plus de 500000 incréments/rotation avec un transfert instantané.

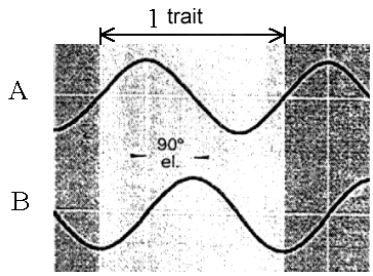


Fig. 3.15 : **ENDAT – Signal sinus et cosinus**

Ce procédé permet d'obtenir une **concentricité** et une **rigidité optimales** et donc des mouvements précis et optimisés dans le temps.

De plus, la valeur incrémentale est comparée à la valeur absolue tous les 200 µs pour détecter d'éventuels problèmes si un décalage se produit.

3.2.2 ENDAT – Plaque signalétique électronique

Le codeur ENDAT comporte une **mémoire de données** EEPROM **non volatile** et ne nécessitant aucune maintenance. Toutes les données requises pour la mise en service du variateur sont stockées dans cette mémoire.

Cette mémoire de données permet d'échanger avec l'électronique de commande toutes les informations qui ont été préprogrammées par B&R, comme par exemple la résolution du codeur ou les paramètres moteur, et ce via la ligne SSI **bidirectionnelle**. Cette lecture automatique permet une adaptation optimale au système moteur/codeur, le tout avec la technologie **Plug&Play**.

La plaque signalétique électronique que constitue cette mémoire permet de réduire la maintenance et les temps de mise en service.

Comparaison ENDAT/Résolver

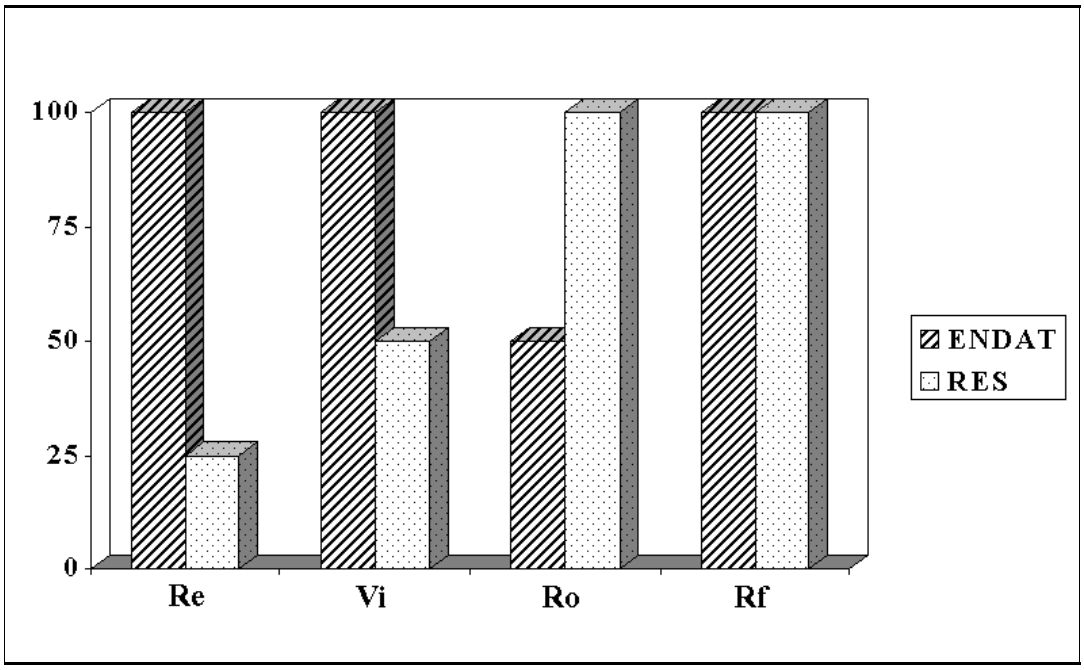


Fig. 3.16 : **Comparaison ENDAT/Résolver**

RES.....Résolver
ENDAT.....Codeur sinusoïdal optique ENDAT

Re.....Résolution
Vi.....Vitesse de transmission du signal
Ro.....Robustesse
Rf.....Prise de référence (temps nécessaire)

4. ELECTRONIQUE DE PUISSANCE

Pour atteindre la vitesse ou la position de consigne souhaitée à l'aide du variateur, il faut fournir au moteur une tension ou un courant approprié.

A cet effet, la tension ou le courant issu du réseau électrique doit être traité/converti par l'électronique de puissance dans le variateur.

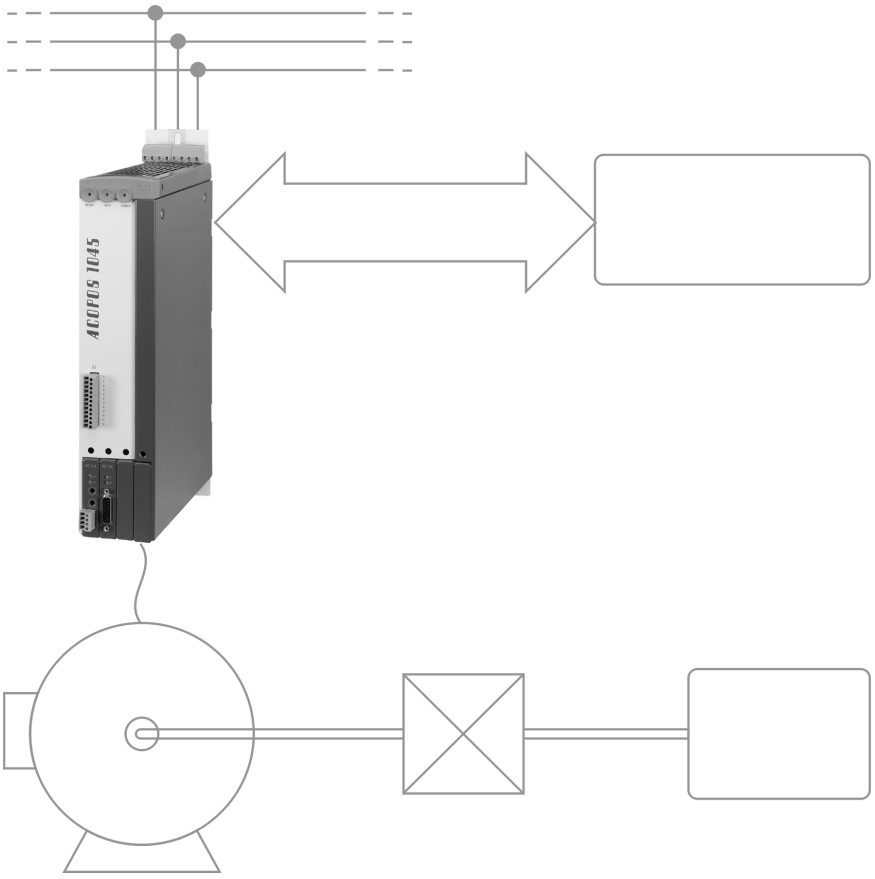


Fig. 3.17 : **Eléments d'une chaîne d'entraînement – Electronique de puissance**

4.1 Aperçu général

Dans ce domaine, nous trouvons deux types d'éléments de puissance :

- **la variateur**
- le convertisseur de fréquence = CF

Ces deux éléments diffèrent par leur capacité à générer des mouvements précis et optimisés dans le temps, lesquels nécessitent un retour rapide et exact des valeurs de position réelle dans la boucle de régulation.

Le variateur se prête bien à la réalisation de tels mouvements. Le convertisseur de fréquence joue principalement un rôle d'actionneur, le plus souvent sans aucune régulation.

4.1.1 Structure

L'électronique de puissance se subdivise en trois parties :

- le redresseur (redresseur en pont = RP)
- le bus DC avec hacheur de freinage
- l'onduleur (onduleur 6 impulsions = ON)

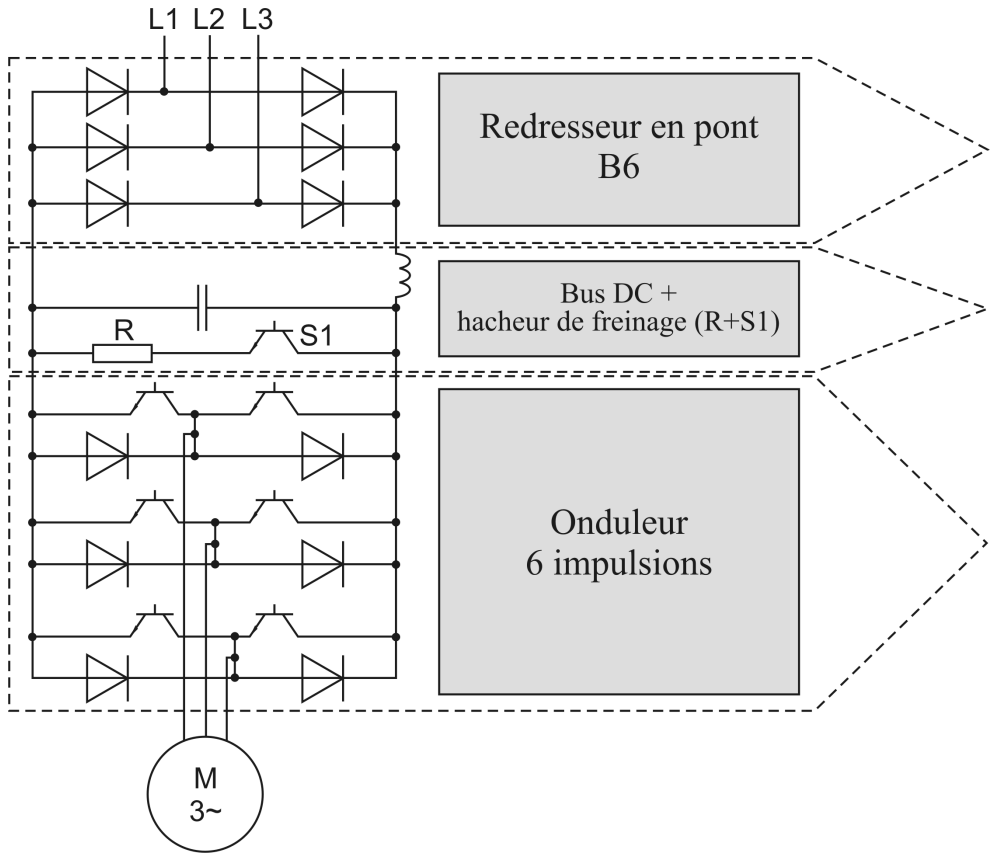


Fig. 3.18 : **Structure de l'électronique de puissance**

A partir de la tension du réseau électrique, le RP produit une tension continue pour le bus DC. Le condensateur permet un stockage temporaire de l'énergie.

La tension de moteur requise par la régulation est obtenue par un cadencement approprié de la tension du bus DC.

4.1.2 Commande de l'onduleur

La régulation prédéfinit une **tension de consigne**, laquelle doit être transformée par la **logique de commande** grâce à l'envoi d'impulsions de commande appropriées aux éléments de commutation de puissance de l'onduleur.

Le système de commande le plus simple repose sur le principe de la commutation par paquet. La tension de phase rectangulaire issue de cette commutation présente l'inconvénient d'avoir une haute teneur en harmoniques et donc de générer des pertes.

On obtient des **caractéristiques de concentricité optimales** grâce à la **M**odulation de **L**argeur d'**I**mpulsion :

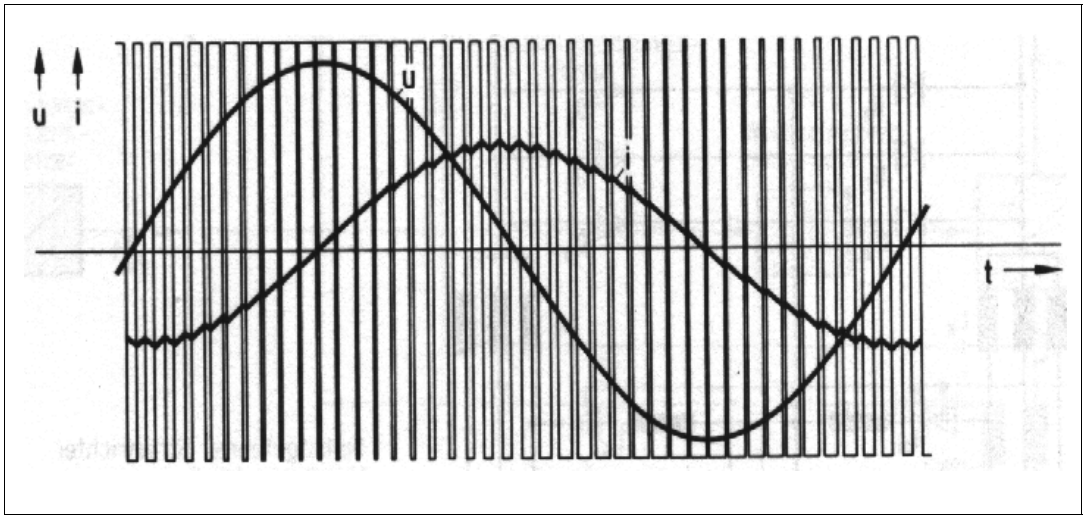


Fig. 3.19 : **Modulation de largeur d'impulsion**

L'augmentation de la fréquence a pour effet d'améliorer la précision du positionnement ainsi que la concentricité !

4.1.3 Résistance de freinage

L'énergie renvoyée par le moteur dans le circuit du bus DC peut être exploitée de deux manières :

- **Résistance de freinage**
  Lorsque la tension du bus DC augmente, un commutateur de puissance (hacheur de freinage) est commandé périodiquement en ouverture et fermeture en fonction de la valeur de cette tension. Le surplus d'énergie électrique est converti en chaleur dans une résistance refroidie de façon appropriée.

  Lorsque l'énergie de freinage maximale est atteinte, le commutateur de puissance devient passant.

- **Liaison de bus DC**
  La tension du bus DC est déviée vers les bornes de l'appareil pour pouvoir être transmise aux autres variateurs.

  Dans le cas où le moteur renvoit de l'énergie au variateur, cette énergie peut alors être détournée via la liaison de bus DC vers un autre variateur fonctionnant en mode moteur.

Remarque

Vous trouverez des informations détaillées sur le raccordement des ACOPOS dans le manuel d'utilisation.

4.2 Highlights de B&R

Pour la commande et la régulation de moteur synchrone ou asynchrone, B&R propose un variateur avec résistance de freinage et filtre CEM intégrés.

- Commande par IGBT
- Modulation MLI 10 ou 20 kHz
- Contrôle précis de la température
- Positionnement optimal du point neutre pour utiliser pleinement la tension du bus DC.

4.2.1 Commande moteur par IGBT

L'onduleur 6 impulsions est équipé de transistors **IGBT** permettant un transfert de puissance avec un haut rendement.

L'IGBT présente à la fois les avantages d'un MOSFET (**circuits de commande de faible puissance**) et d'un transistor de puissance (**pertes faibles à l'état passant**).

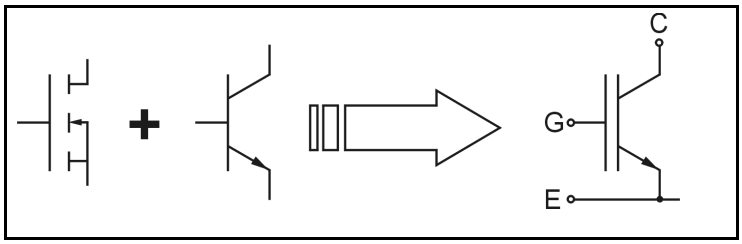


Fig. 3.20 : **IGBT**

4.2.2 Modulation MLI 20 kHz

Pour effectuer une régulation hautement dynamique, on utilise un signal MLI généré avec une horloge de 20 kHz, ce qui permet un **temps de réaction de 50 µs** !

De plus, la fréquence utilisée se trouve en-dehors du domaine audible et n'entraîne donc pas de **nuisances sonores**.

Dans le cas d'applications spéciales, lorsque l'on souhaite obtenir plus de puissance de la part du variateur avec une dynamique plus faible, une **réduction par le logiciel** de la fréquence d'horloge est prévue.

4.2.3 Contrôle de la température

Pour utiliser la puissance au maximum, la température réelle est mesurée physiquement en deux points.

- IGBT
- Enroulement du moteur

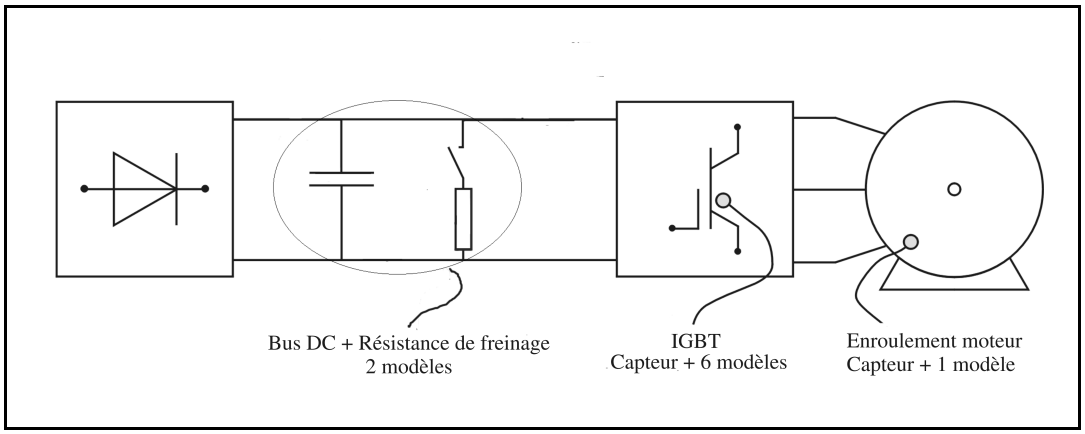


Fig. 3.21 : **Contrôle de la température**

De plus, une modélisation thermique de température est effectuée 9 points à partir du courant continu, du courant de crête, ...

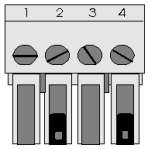
4.2.4 Filtre CEM intégré

A fréquence d'horloge élevée, l'effet capacitif introduit par les différents blindages entraîne l'apparition de courants parasites, même en l'absence de courants de charge.

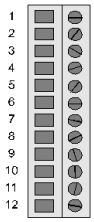
Un filtre CEM est intégré au variateur B&R pour éviter toute perturbation sur le réseau et ses différents participants.

4.2.5 Connectique

La connectique, pratique et facile à utiliser, se prête bien au câblage en milieu industriel (série 1022 – 1320). Les efforts et le temps consacrés au câblage sont ainsi réduits.



Pour éviter toute erreur de connexion, les borniers sont munis de broches codées correspondant à celles du connecteur ACOPOS.



Les lignes de signal digital sont raccordées avec un bornier.

Vous trouverez également une description des connexions dans le manuel d'utilisation ACOPOS fourni avec le variateur.

Remarque

Grâce à leur structure compacte, les variateurs ACOPOS permettent un gain de place important dans l'armoire électrique lorsqu'ils sont placés directement les uns à côté des autres.

Les cotes de montage détaillées sont indiquées dans le manuel d'utilisation ACOPOS.

DANGER

Tension de boîtier et courant de fuite élevés ! Danger de mort ou risque de blessure par électrocution !

- Avant la mise sous tension, mettre à la terre les équipements électriques, les boîtiers de tous les appareils électriques et les moteurs avec le conducteur de protection, même pour des tests de courte durée.
- Raccorder le conducteur de protection des équipements et appareils électriques au réseau d'alimentation. Le courant de fuite est supérieur à 3,5 mA.
- Important : La connexion du conducteur de protection (PE) sur le boîtier ACOPOS doit être mise à la terre avec des fils de cuivre d'au moins 10 mm2.

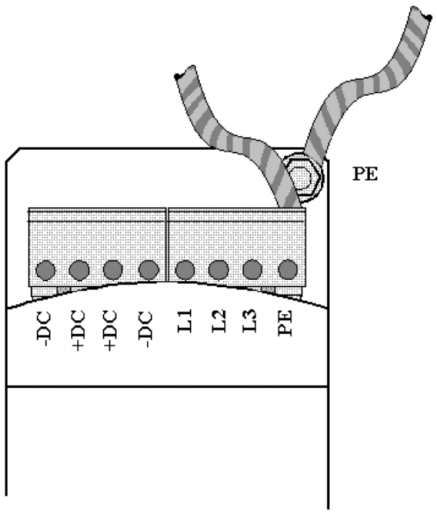


Fig. 3.22 : **Raccordement du conducteur de protection**

IMPORTANT

Vous trouverez des informations détaillées dans le manuel **ACOPOS Installation !**

5. RÉGULATION

5.1 Aperçu général

L'électronique de commande et de régulation constitue le cœur de la chaîne d'entraînement.

En matière de **qualité**, de **dynamique** et de **précision** de mouvement, cette électronique joue un rôle essentiel.

Dans la plupart des applications, les différentes composantes - le courant, la vitesse et la position - sont combinées en cascade via des circuits de régulation autonomes.

La combinaison en cascade permet un paramétrage simple et une mise en service des boucles de régulation de l'intérieur vers l'extérieur.

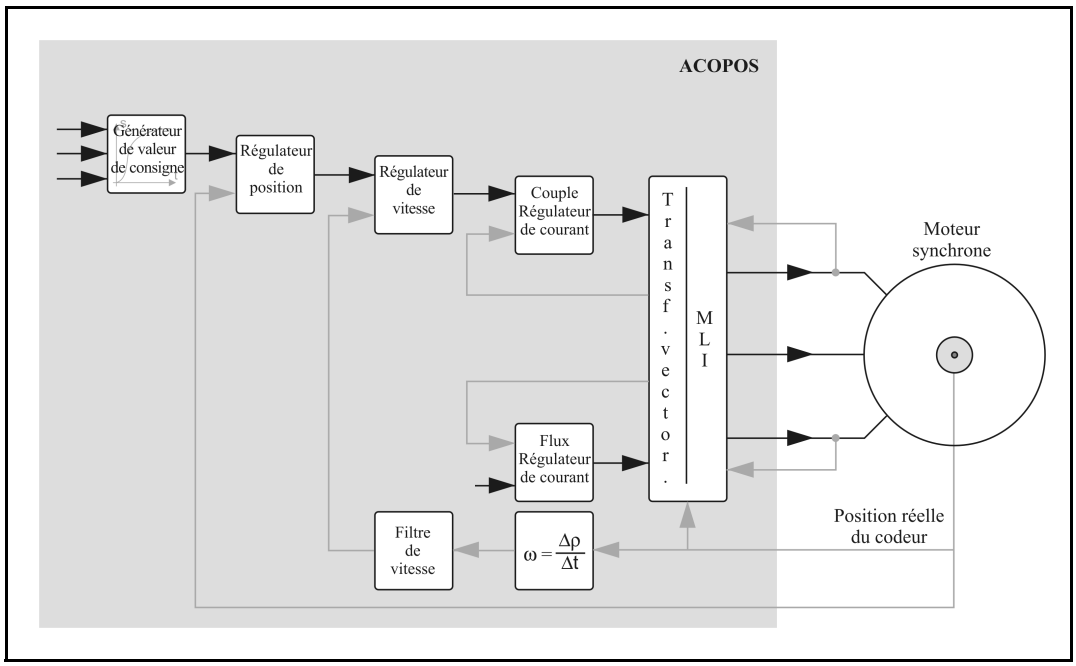


Fig. 3.23 : **Combinaison en cascade**

5.2 Highlights de B&R

5.2.1 Le concept de régulation de B&R

Les variateurs de B&R intègrent tous les éléments de commande et de régulation ainsi que les fonctions d'acquisition et de traitement des valeurs de mesure.

La mise en cascade des trois boucles de régulation (position, vitesse et couple) **simplifie le paramétrage et la mise en service** des variateurs B&R.

La **concentricité** optimale et la **rigidité** élevée des axes garantissent des temps d'échantillonnage courts dans les boucles de régulation :

- Régulateur de couple.....**50 µs**
- Régulateur de vitesse.....**200 µs**
- Régulateur de position.....**400 µs**

5.2.2 Régulateur de position prédictif

Le régulateur de position prédictif d'ACOPOS prend en compte l'inertie de l'entraînement.

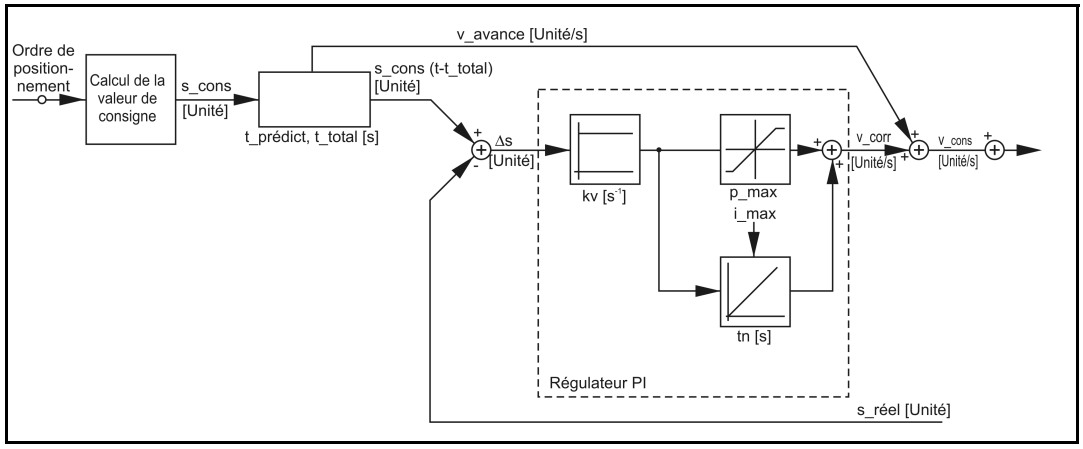


Fig. 3.24 : **Régulateur de position prédictif**

5.2.3 Electronique de régulation

Un **P**rocesseur de **S**ignal **D**igital à haut niveau d'intégration de type SHARC rend les unités de régulation hautement performantes.

Avec une vitesse de traitement de 120 Mflops/sec et un traitement des données sur 32/48 bits, ce processeur est conçu pour atteindre **les fréquences d'impulsion élevées** requises pour la commande des composants de puissance.

Le circuit **F**ield**P**rogrammable**G**ate**A**rray utilisé constitue aussi une évolution en matière de logique programmable.

Les blocs logiques paramétrables avec le logiciel du variateur permettent **une spécialisation maximale** de l'électronique tout en offrant un **maximum de flexibilité**.

Les modifications apportées à la logique ne nécessitent plus de redéfinition du matériel comme c'était le cas autrefois. Une mise à jour peut désormais être effectuée avec le **système d'exploitation B&R du variateur**.

6. COMMUNICATION AVEC LE MONDE EXTÉRIEUR

6.1 Aperçu général

Le variateur a besoin d'interfaces pour communiquer avec le monde extérieur.

- Automation Net
- Interrupteur de fin de course
- Interrupteur de position de référence
- Variateur prêt
- Quick Stop
- Interface codeur
- Frein
- Capteur de température moteur
- Sortie 24 V
- Bus DC
- Connexion réseau
- Résistance de freinage externe
- Alimentation en puissance du moteur
- LED d'état
- ...

Remarque

Manuel d'utilisation ACOPOS

6.2 Highlights de B&R

Par ailleurs, le variateur est équipé de manière standard de deux entrées de déclenchement 24 VDC (trigger) permettant la réalisation des fonctions technologiques.
Exemple : Contrôle des marques d'impression

La **modularité du système** permet de choisir la configuration optimale.

- Interface CAN / Ethernet Powerlink
- Interface codeur pour retour moteur
- Interface codeur supplémentaire
  Exemple : retour direct d'information de position
- Modules d'E/S
  Exemple : séquenceur à cames

DIMENSIONNEMENT

1. GÉNÉRALITÉS

Avant de réaliser l'automatisation des différentes séquences de mouvement, il faut d'abord dimensionner l'ensemble moteur-variateur.

Le rapport prix/performance de la machine ou de l'installation ne sera optimal que si le moteur et le variateur ont été bien choisis.

Si l'ensemble moteur/variateur est sous-dimensionné, les processus requis ne pourront pas être réalisés de manière satisfaisante. A l'inverse, un surdimensionnement de l'ensemble moteur/variateur se traduira par des surcoûts (encombrement, poids et puissance trop importants).

Dans ce chapitre, vous trouverez des informations destinées à vous guider dans le choix de l'entraînement approprié.

Analyse du mouvement
: Première étape : analyse du mouvement.

Transfert d'énergie
: Comparaison de différentes solutions mécaniques pour transmettre l'énergie du moteur à la charge.

Limites de l'entraînement
: Prise en considération des limites des différents éléments de l'entraînement

Méthode de dimensionnement
: Présentation d'une méthode de dimensionnement d'entraînement

Exemples
: Introduction au processus de dimensionnement à partir d'exemples issus de la pratique

Solutions
: Pour faciliter l'étude de ce chapitre, les solutions des différents exercices figurent aussi dans ce manuel.

2. ANALYSE DU MOUVEMENT

Pour toute application de positionnement, l'entraînement doit accomplir des mouvements spécifiques et prédéfinis.

La conception d'un entraînement - moteur + codeur + variateur - adapté aux mouvements requis passe par l'analyse des mouvements à l'aide de **diagrammes v/t**.

Dans les pages suivantes figure un récapitulatif des formules les plus importantes.

2.1 Lois de la cinématique

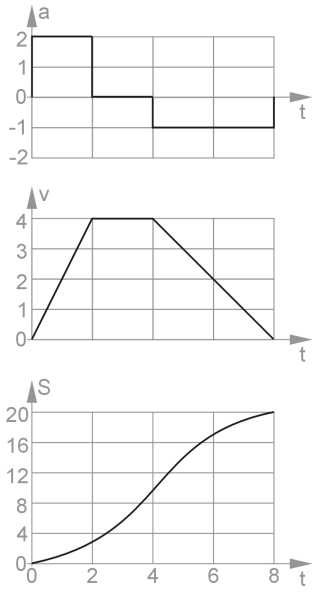
	Translation	Rotation
Distance/Angle	$s = v \cdot t$	$\varphi = \omega \cdot t$
Vitesse	$v = \frac{s}{t}$	$v = \omega \cdot r = \frac{\pi}{60 \cdot n \cdot d}$
Vitesse angulaire		$\omega = \frac{d\varphi}{dt} = \frac{2 \cdot \pi \cdot n}{60}$
Accélération	$a = \frac{dv}{dt}$	$\alpha = \frac{d\omega}{dt}$
Force	$F = m \cdot a$	
Couple	$C = F \cdot r$	$C = J \cdot \frac{d\omega}{dt}$
Puissance	$P = F \cdot v$	$P = C \cdot \omega$
Energie	$W = F \cdot s$	$W = C \cdot \varphi$
Energie	$W = \frac{1}{2} \cdot m \cdot v^2$	$W = \frac{1}{2} \cdot J \cdot \omega^2$

t ..... Temps [s]
s ..... Distance [m]
φ ..... Angle [rad]
v ..... Vitesse [m/s]
ω ..... Vitesse angulaire [rad/s]
a ..... Accélération [m/s^2]
α ..... Accélération angulaire [rad/s^2]
m ..... Masse [kg]
J ..... Moment d'inertie [$kg\ m^2$]
F ..... Force [N]
C ..... Couple [Nm]
P ..... Puissance [W]
W .... Energie/Travail [W_s]

2.2 Profils de mouvement

Diagramme v/t avec phase d'accélération constante.

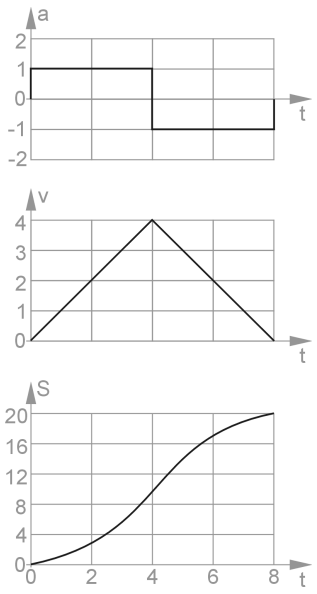
Temps d'accélération	$t_{acc} = \frac{v}{a_{acc}}$
Distance d'accélération	$s_{acc} = \frac{1}{2} \cdot v \cdot t_{acc}$ $s_{acc} = \frac{1}{2} \cdot a_{acc} \cdot t_{acc}^2$
Temps de décélération	$t_{dec} = \frac{v}{a_{dec}}$
Distance de décélération	$s_{dec} = \frac{1}{2} \cdot v \cdot t_{dec}$ $s_{dec} = \frac{1}{2} \cdot a_{dec} \cdot t_{dec}^2$
Distance avec v = constante	$s = v \cdot t$
Temps pour v = constante	$t = \frac{s}{v}$
Distance totale	$s_{tot} = s_{acc} + s_{dec} + s$
Temps total	$t_{tot} = t_{acc} + t_{dec} + t$



S'il existe une limitation de vibration, il faut en tenir compte dans les formules !

Diagramme v/t pour couple minimal

Temps d'accélération	$t_{acc} = \frac{v}{a_{acc}}$
Distance d'accélération	$s_{acc} = \frac{1}{2} \cdot v \cdot t_{acc}$ $s_{acc} = \frac{1}{2} \cdot a_{acc} \cdot t_{acc}^{2}$
Temps de décélération	$t_{dec} = \frac{v}{a_{dec}}$
Distance de décélération	$s_{dec} = \frac{1}{2} \cdot v \cdot t_{dec}$ $s_{dec} = \frac{1}{2} \cdot a_{dec} \cdot t_{dec}^{2}$
Distance totale	$s_{tot} = s_{acc} + s_{dec}$
Temps total	$t_{tot} = t_{acc} + t_{dec}$



S'il existe une limitation de vibration, il faut en tenir compte dans les formules !

3. TRANSFERT D'ÉNERGIE

L'énergie du moteur peut être transmise à la charge de différentes manières. La technique la plus appropriée est celle qui répond le mieux aux exigences de l'application de positionnement.

La transformation d'un **mouvement de rotation** en **mouvement linéaire** ou bien **l'adaptation de la vitesse ou du couple** peut constituer un des critères de choix.

Les techniques les plus courantes sont exposées dans la suite.

3.1 Entraînement direct

Dans le cas d'un entraînement direct, la charge est couplée au moteur par une liaison **rigide**.

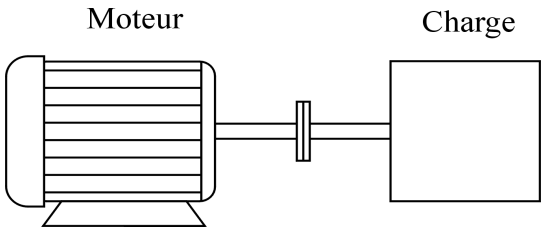


Fig. 4.1 : **Entraînement direct**

$$C_{tot} = f(t) = C_{charge} + C_{frottement} + C_{acceleration}$$

$$C_{acceleration} = C_{acc}$$

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt} + \frac{\omega}{2} \cdot \frac{dJ}{dt}$$

A inertie constante, on a :

$$J_{tot} = J_{charge} + J_{embrayage} + J_{arbre} + J_{moteur}$$

D'où le couple d'accélération :

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt}$$

$\frac{d\omega}{dt}$ .....calculé à partir des diagrammes v/t d'analyse du mouvement

J_{moteur} .....obtenu à partir des caractéristiques du moteur

$J_{embrayage}$ obtenu à partir des caractéristiques de la boîte de vitesses

J_{arbre} .....mesuré ou calculé

J_{charge} .....mesuré ou calculé

Exemple

Calcul de l'inertie d'un cylindre plein

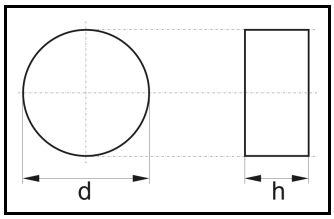


Fig. 4.2 : **Inertie d'un cylindre**

$$J = \frac{1}{2} \cdot m \cdot \left(\frac{d}{2}\right)^2$$

J..... Moment d'inertie [kgm^2]
m ..... Masse du cylindre [kg]
d ..... Diamètre du cylindre [m]
h ..... Hauteur du cylindre [m]

3.2 Entraînement indirect

Une **adaptation** du **mouvement**, du **couple** et de la **vitesse** peut être réalisée avec une boîte de vitesse. Seul le **rendement** de la boîte de vitesses a une incidence sur la puissance transmise.

Une boîte de vitesse est en quelque sorte un **transformateur mécanique**.

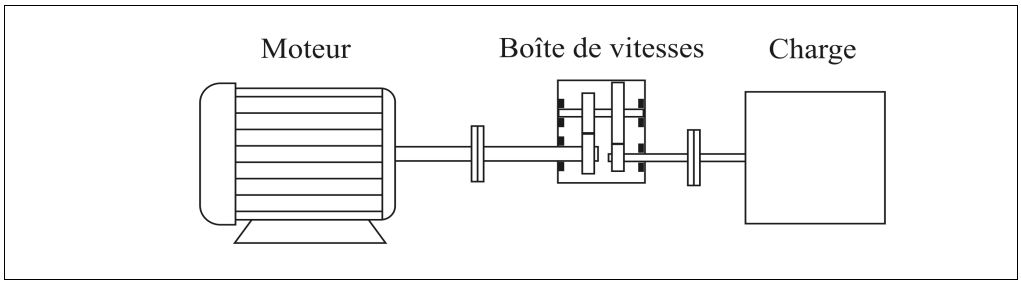


Fig. 4.3 : **Entraînement indirect**

$$J_{tot} = J_{charge}^{*} + J_{embrayage} + J_{arbre} + J_{boitedevitesse} + J_{moteur}$$

Le moment d'inertie total correspond à la somme des moments d'inertie des différents éléments de l'entraînement, à condition que ces derniers ait un axe de référence commun.

De ce fait, l'inertie de la charge est ramenée à l'arbre du moteur, par application du principe de conservation de l'énergie :

$$J_{charge}^{*} \cdot \frac{\omega_{arbremoteur}^{2}}{2} = J_{charge} \cdot \frac{\omega_{arbrecharge}^{2}}{2}$$

$$J_{charge}^{*} = J_{charge} \cdot \frac{\omega_{arbrecharge}^{2}}{\omega_{arbremoteur}^{2}}$$

En utilisant le rapport de transformation

$$i = \frac{\omega_{arbremoteur}}{\omega_{arbrecharge}} = \frac{Nbdents_{charge}}{Nbdents_{moteur}}$$

on obtient

$$\boxed{J_{charge}^{*} = J_{charge} \cdot \frac{1}{i^{2}}}$$

3.3 Vis à billes

Du fait de sa construction mécanique, une vis à billes permet de convertir un mouvement de rotation en un mouvement de translation lié au **pas du filetage**.

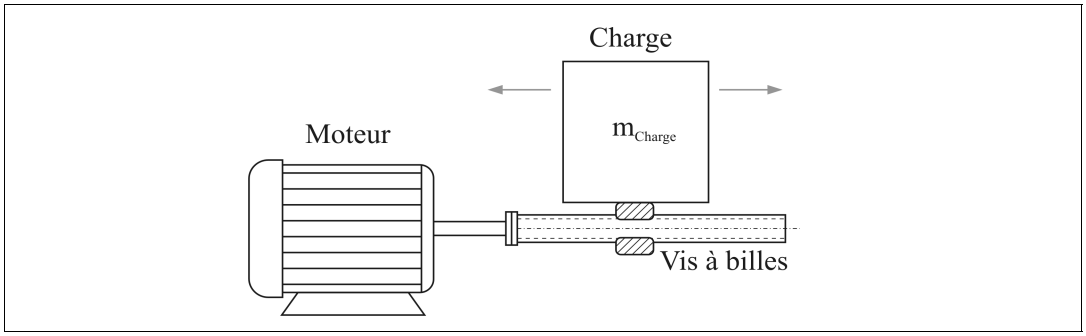


Fig. 4.4 : **Vis à billes**

Calcul du moment d'inertie :

$$J_{tot} = m_{charge} \cdot \frac{p^2}{4 \cdot \pi^2} + J_{vis}$$

p ..... Pas de la vis à billes [m/tour]

Remarque

Le **rendement** de ce maillon de la transmission dépend essentiellement du type de vis à billes utilisé (vis à billes avec roulements ou vis à billes avec filetage trapézoïdal) et doit absolument être pris en compte dans les calculs.

3.4 Crémaillère

L'association d'une **crémaillère** et d'un **pignon** permet de transformer un mouvement de rotation en un **mouvement linéaire** !

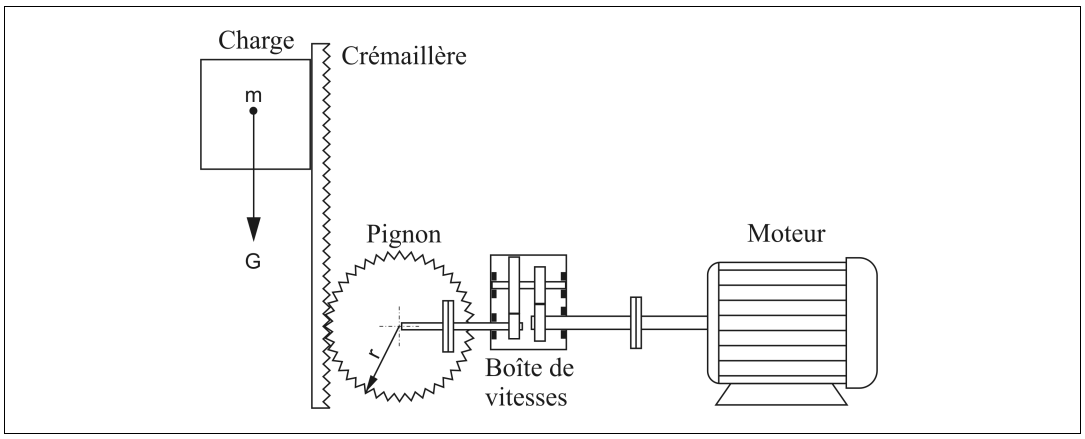


Fig. 4.5 : **Entraînement avec crémaillère**

Dans cette configuration, nous avons une charge constante G due à l'action de la pesanteur. Cette charge se calcule avec la formule

$$G = m \cdot g$$

La force d'accélération requise pour effectuer un mouvement vertical se calcule comme suit :

$$F = m \cdot a$$

$$F_{total} = F \pm G$$

$$C_{poulie} = F_{total} \cdot r$$

$$C_{moteur} = \frac{C_{poulie}}{i \cdot \eta}$$

η .....Rendement

i .....Rapport de transformation

4. LIMITES DU SYSTÈME D'ENTRAÎNEMENT

Après avoir analysé les différentes séquences de mouvement de la machine en tenant compte du type de transmission d'énergie, vous devez maintenant choisir le système d'entraînement (comprenant le codeur, le moteur et le variateur) le plus économique et donc le moins volumineux.

Lors du choix, certaines limites doivent être respectées :

Codeur :

- Résolution
- Précision

Moteur :

- Vitesse
- Couple maximal
- Charge thermique
- Rapport d'inertie
- Charge radiale et axiale sur l'arbre

Variateur :

- Courant maximal
- Charge de courant continu
- Puissance continue - résistance de freinage

Frein de maintien

- Couple de maintien

Remarque

Pour plus de détails, veuillez vous reporter au catalogue MotionSystems ou au manuel d'utilisation ACOPOS.

4.1 Limites du codeur

4.1.1 Résolution

Un codeur Endat Sinus optique standard avec 512 traits par tour peut être lu avec une résolution de plus de $(512 \cdot 4 \cdot 2^{(12-4)}) \approx 500000[\text{inc par tour}]]$ en utilisant une carte codeur B&R.

4.1.2 Précision

La précision du codeur est donnée en minutes d'angle ou secondes d'angle. Une minute d'angle = 1' correspond à 60 secondes d'angle = 60''.

4.2 Limites du moteur

4.2.1 Vitesse

Le moteur doit être capable de fournir l'ensemble des vitesses requises par la machine.
Ceci peut être vérifié dans le diagramme **Couple/Vitesse**. Ces diagrammes figurent dans le catalogue Motion.

On peut effectuer des **ajustements** en utilisant **une boîte de vitesses** ou en exploitant le moteur asynchrone avec un champ faible, en tenant compte de la caractéristique couple/vitesse.

4.2.2 Couple maximal

Le couple de crête maximal correspondant à la vitesse requise peut être déduit de la caractéristique couple/vitesse. C_{max} est le couple de crête max. admissible à l'arrêt. Lorsque la vitesse augmente, le couple de crête diminue légèrement en raison des pertes dues au frottement. Une chute importante du couple de crête se produit lorsque la force contre-électromotrice devient prépondérante.

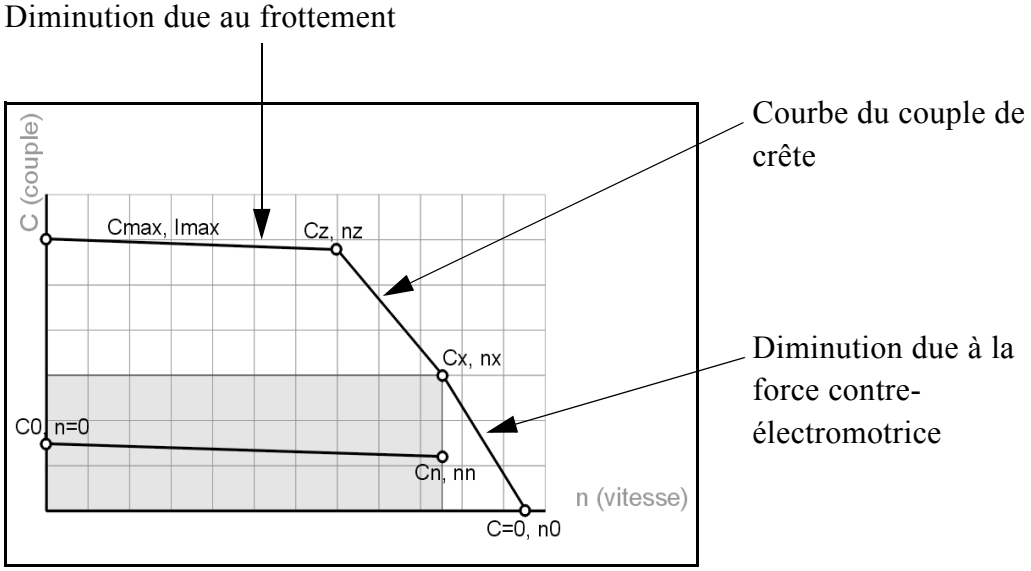


Fig. 4.6 : **Caractéristique couple/vitesse**

Tension de force contre-électromotrice

D'après la loi de l'induction électromagnétique, une tension est induite dans les enroulements du stator :

$$U = N \cdot \frac{d\Phi}{dt} \text{ avec } \Phi = \Phi_{crete} \cdot \sin(\omega \cdot t)$$

$$U = N \cdot \omega \cdot \Phi_{crete} \cdot \cos(\omega \cdot t)$$

En valeur efficace :

$$U = N \cdot 2 \cdot \pi \cdot f \cdot \frac{\Phi_{crete}}{\sqrt{2}} = 4{,}44 \cdot N \cdot f \cdot \Phi_{crete}$$

Cette tension dépend des caractéristiques de la machine (facteur de raccourcissement/d'allongement, etc.) représentées par la constante k.

$$U = k \cdot N \cdot 4{,}44 \cdot \Phi_{crete} \cdot \frac{n}{60}$$

Les valeurs fixes sont regroupées dans une constante de vitesse appelée k_E. D'où l'équation suivante pour la tension de f.c.é.m. entre phases :

$$U = k_E \cdot n$$

Lorsque la vitesse augmente, **la tension de force contre-électromotrice** augmente également.

Le courant fourni au moteur synchrone est d'autant plus faible que la tension de force contre-électromotrice se rapproche de la tension du bus DC de l'ACOPOS.

Lorsque la vitesse est égale à la vitesse à vide n_0, la tension de force contre-électromotrice est très légèrement inférieure à la tension du bus DC de l'ACOPOS. Le courant circulant dans le moteur ne sert alors plus qu'à vaincre les frottements.

4.2.3 Charge thermique

La charge thermique prend en compte l'**échauffement des enroulements**. Elle est déterminée en calculant **la valeur efficace du couple** à partir du diagramme vt. Cette méthode présuppose une période de mouvement inférieure à la constante de temps thermique du moteur.

A vitesse nominale, le moteur peut fournir en permanence le couple nominal C_n sans échauffement.

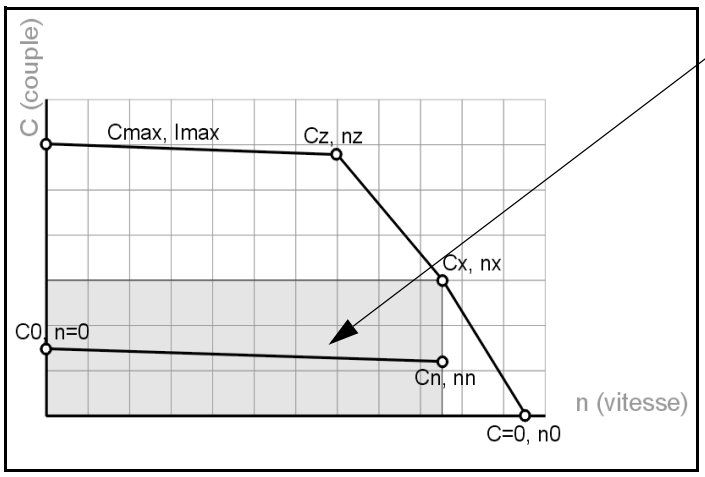


Fig. 4.7 : **Caractéristique couple/vitesse**

4.2.4 Rapport d'inertie

Le rapport d'inertie entre la charge et le moteur ne doit pas dépasser certaines limites afin de permettre une régulation dynamique et stable.

4.2.5 Charge radiale et axiale sur l'arbre

Avec les commandes à courroie, il est indispensable de tenir compte de la précontrainte et de la charge radiale correspondante qui s'exercent sur l'arbre moteur.

Des forces axiales peuvent apparaître sur des moteurs montés sur un axe donné et subissant les accélérations dues aux mouvements de l'autre axe.

4.3 Limites du variateur

4.3.1 Courant de crête

Courant maximal de courte durée pouvant être supporté par l'IGBT pour fournir le couple de crête.

4.3.2 Courant continu

Le courant continu admissible peut être délivré en permanence par le variateur tant que les conditions environnementales sont respectées (pas de surcharge thermique).

4.3.3 Puissance continue de freinage - Résistance de freinage

L'**énergie renvoyée** dans le variateur ACOPOS **lors du freinage** est transformée en **chaleur** par la résistance de freinage.

Pour que l'énergie renvoyée dans l'ACOPOS soit convertie sans surcharge thermique, il convient de comparer la puissance attendue aux caractéristiques de la résistance de freinage.

4.3.4 Frein de maintien

Le frein doit être adapté au **couple de maintien** requis.

Le frein de maintien ne doit pas être utilisé pour réaliser un **freinage actif** !

Le frein ne peut et ne doit **pas être utilisé pour la protection des personnes**.

Constante de couple

Pour comparer les valeurs issues du diagramme vt aux limites du variateur, il faut déterminer les courants correspondant aux différents couples.

La **relation entre le couple et le courant du moteur** peut être déterminée à partir de la **constante de couple k_C** spécifiée dans la fiche technique du moteur.

La constante de couple se calcule de la manière suivante :

$$C = 2 \cdot F \cdot r \cdot \sin\alpha$$

$$F = l \cdot B \cdot I$$

$$C = k \cdot l \cdot 2 \cdot r \cdot B \cdot I \cdot \sin\alpha$$ .... k tient compte des caractéristiques de construction du MS !

$$A = l \cdot 2 \cdot r$$

$$\Phi = A \cdot B$$

$$C = k \cdot \Phi \cdot I \cdot \sin\alpha$$

$\sin\alpha = 1$ ... assuré par la commutation !

$$k_C = k \cdot \Phi$$

$$\boxed{C = k_C \cdot I}$$

Le paragraphe suivant expose une méthode de dimensionnement d'entraînement permettant d'atteindre un très bon rapport performance/prix. La procédure est la suivante :

5. MÉTHODE DE DIMENSIONNEMENT

Voici une méthode de dimensionnement d'entraînement permettant d'atteindre un très bon rapport performance/prix :

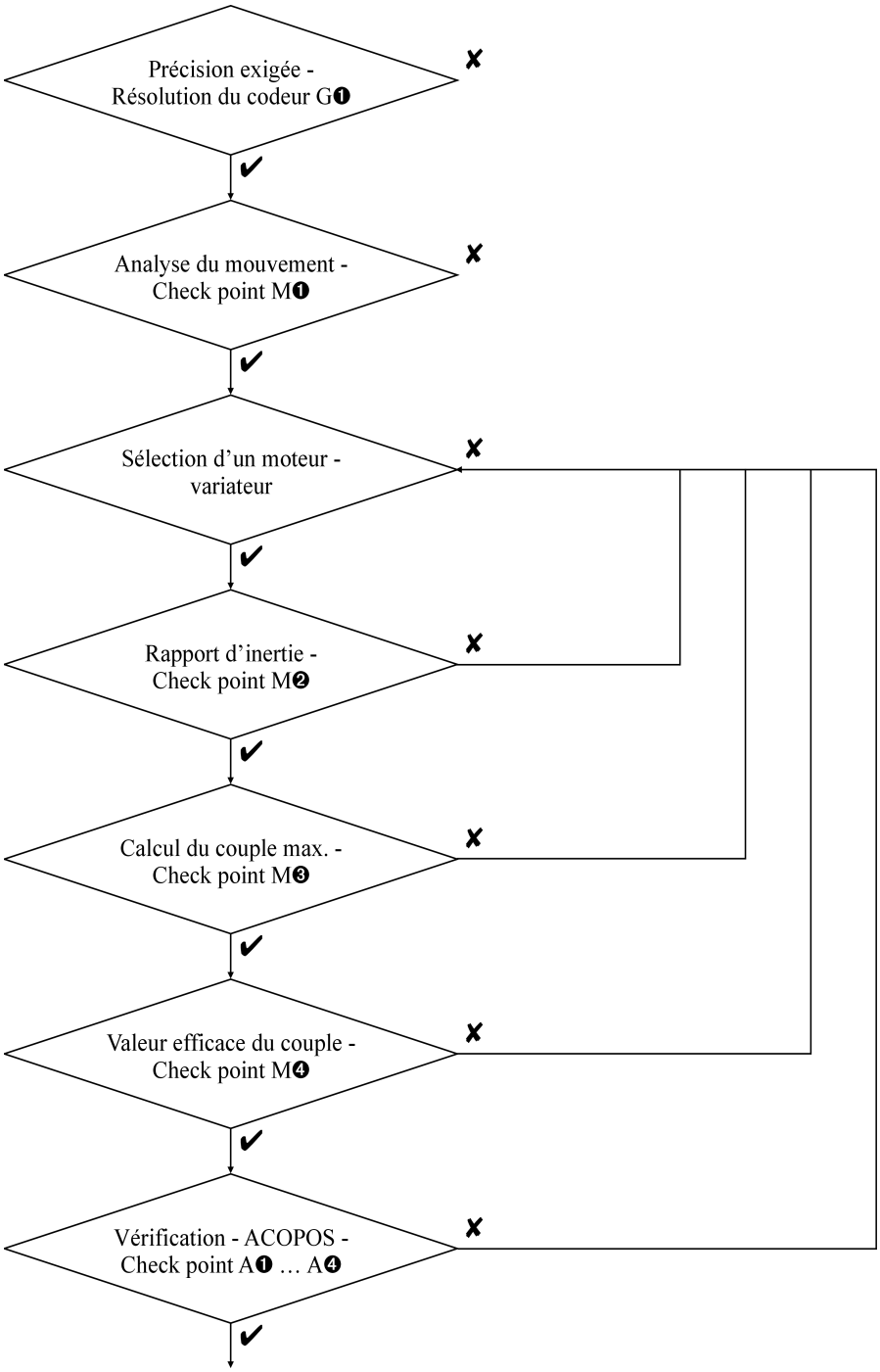


Fig. 4.8 :

5.1 Procédure itérative

5.1.1 Résolution et précision du codeur

$$Re_{req} > \frac{Re_{codeur}}{10}$$

Re_{req} Résolution minimale requise par l'application
Re_{codeur} Résolution max. du codeur

$$Pr_{req} < Pr_{codeur}$$

Pr_{req} Précision requise par l'application
Pr_{codeur} Précision du codeur

5.1.2 Analyse du mouvement

Tout dimensionnement se fait avec un diagramme vt servant à décrire le mouvement. Le diagramme vt permet de déterminer la vitesse maximale côté charge et donc aussi la vitesse maximale côté moteur du fait de l'accouplement mécanique.

5.1.3 Choix du moteur

Le moteur adéquat est alors choisi dans le catalogue en fonction de la vitesse maximale déterminée lors de l'analyse du mouvement. A ce stade, nous n'avons pas d'autres paramètres de sélection. Nous avons donc encore un large choix. C'est pourquoi nous allons d'abord sélectionner un moteur quelconque et vérifier ensuite s'il répond aux autres critères.

5.1.4 Rapport d'inertie

La **valeur empirique** indiquée peut bien sûr être dépassée dans le cas de processus moins dynamiques.

$$\frac{J_{charge}}{J_{moteur}} < 20$$

J_{charge} Moment d'inertie de charge

J_{moteur} Moment d'inertie du moteur (fiche technique)

5.1.5 Couple maximal

$$C_{totmax} < C_{crete}$$

$C_{tot\ max}$ Couple total max. déterminé à partir du diagramme vt

C_{crete} Couple de crête à la vitesse requise, déterminé à partir de la caractéristique Cn

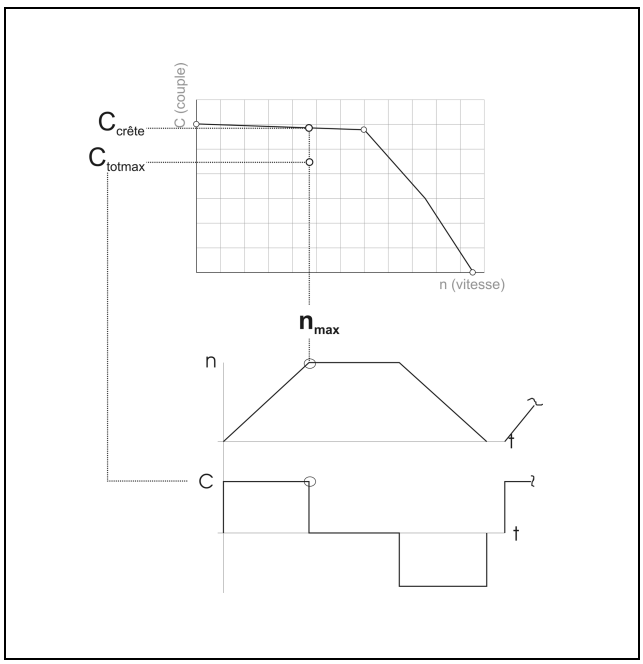


Fig. 4.9 : Diagramme

5.1.6 Valeur efficace du couple

Le couple efficace est donné par la formule :

$$C_{eff} = \sqrt{\frac{1}{T} \cdot \int_0^T C^2 \cdot dt}$$

$$C_{eff} < C_N$$

C_{eff} Valeur efficace obtenue à partir du diagramme vt

C_N Couple nominal dans la fiche technique

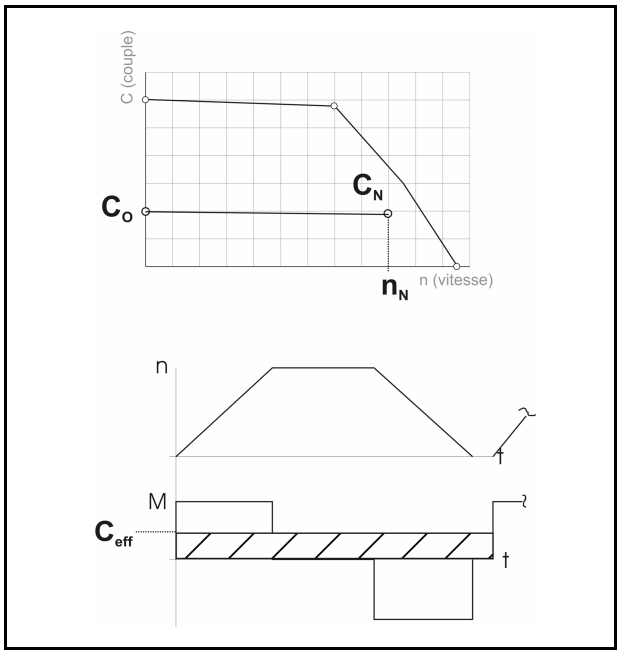


Fig. 4.10 : Couple efficace

5.1.7 Courant maximal de l'ACOPOS

$$I_{totmax} < I_{crete}$$

$I_{tot\ max}$ Valeur max. du courant obtenue en faisant le produit du couple max. du diagramme vt et de k_C

I_{crete} Courant maximal dans la fiche technique

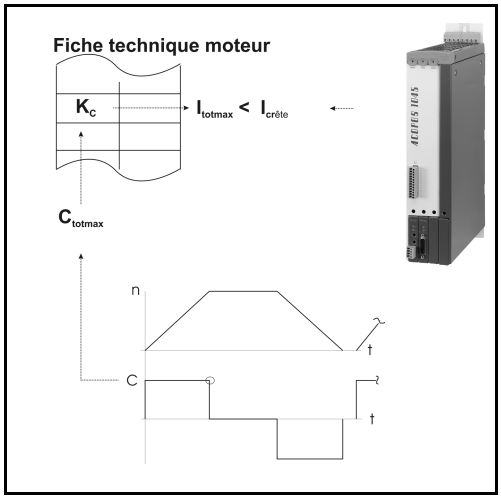


Fig. 4.11 : Courant de l'ACOPOS

5.1.8 Valeur efficace du courant de l'ACOPOS

$$I_{eff} < I_N$$

I_{eff} Valeur efficace du courant obtenue en faisant le produit du couple max. du diagramme vt et de k_C

I_N Courant continu dans la fiche technique

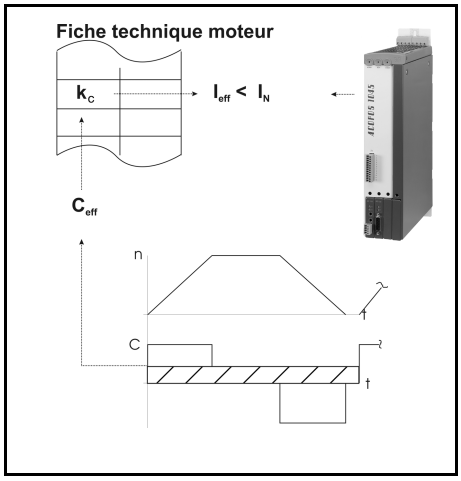


Fig. 4.12 : Courant de l'ACOPOS

5.1.9 Energie de freinage

$$P_{freinage} < P_{continue}$$

$P_{freinage}$ Puissance de freinage moyenne dissipée dans la résistance de freinage

$P_{continue}$ Puissance continue dans la fiche technique

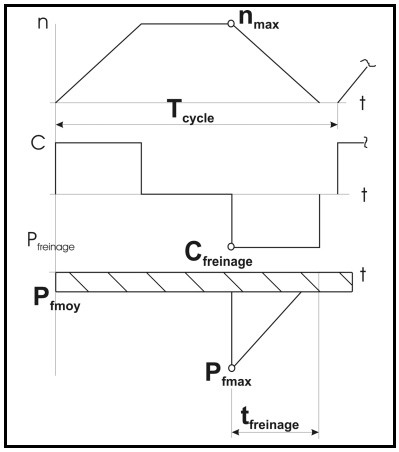


Fig. 4.13 :

$$P_{fmax} = C_{freinage} \cdot \frac{n_{max}}{60} \cdot 2\pi$$

$$P_{fmoy} = P_{fmax} \cdot \frac{t_{freinage}}{T_{cycle}}$$

Le condensateur du bus DC absorbe l'énergie de freinage du moteur. Partant de U_0, la tension du bus DC augmente alors jusqu'à U_{max} (fin de l'absorption d'énergie). Pour protéger le condensateur contre les surtensions, l'énergie de freinage résiduelle est écoulée dans la résistance de freinage grâce au hacheur IGBT.

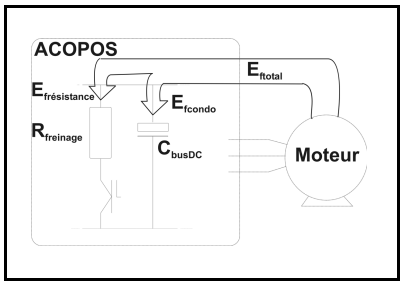


Fig. 4.14 :

$$E_{ftotal} = P_{fmoy} \cdot T_{cycle}$$

$$E_{fcondo} = C \cdot \frac{(U_{max} - U_0)^2}{2}$$

$$E_{fresistance} = E_{ftotal} - E_{fcondo}$$

$$P_{freinage} = \frac{E_{fresistance}}{T_{cycle}}$$

6. EXEMPLES

La méthode de dimensionnement indiquée précédemment est illustrée dans les deux exemples suivants.

Les données correspondent à des demandes clients typiques relatives à des tâches d'automatisation réelles. En annexe, vous trouverez d'autres exemples ainsi que les solutions.

6.1 Exemple d'application : machine à étiqueter les bouteilles

Chaîne d'entraînement d'une machine à étiqueter les bouteilles - Cahier des charges

- La longueur d'une étiquette est de 200 mm.
- L'espace entre deux étiquettes est de 200 mm.
- La cadence machine à atteindre est de 18000 bouteilles/heure.
- Le tapis roulant avec les étiquettes se déplace à une vitesse constante de 2 m/s.
- Pendant la pose de l'étiquette sur une longueur de 200 mm, le rouleau de la presse doit se déplacer à une vitesse constante de 2 m/s.
- Entre deux étiquettes, le rouleau de la presse doit s'arrêter pendant 40 ms pour le changement de bouteille.
- Le diamètre du rouleau de la presse est de 10 cm.
- Une boîte de vitesses à roues dentées avec un rapport de transformation i = 6 est utilisé.
- Le couple de friction est de 0,22 Nm.
- L'inertie de la charge est de 80 kgcm^2.
- La précision requise pour la position est de ±1 mm.

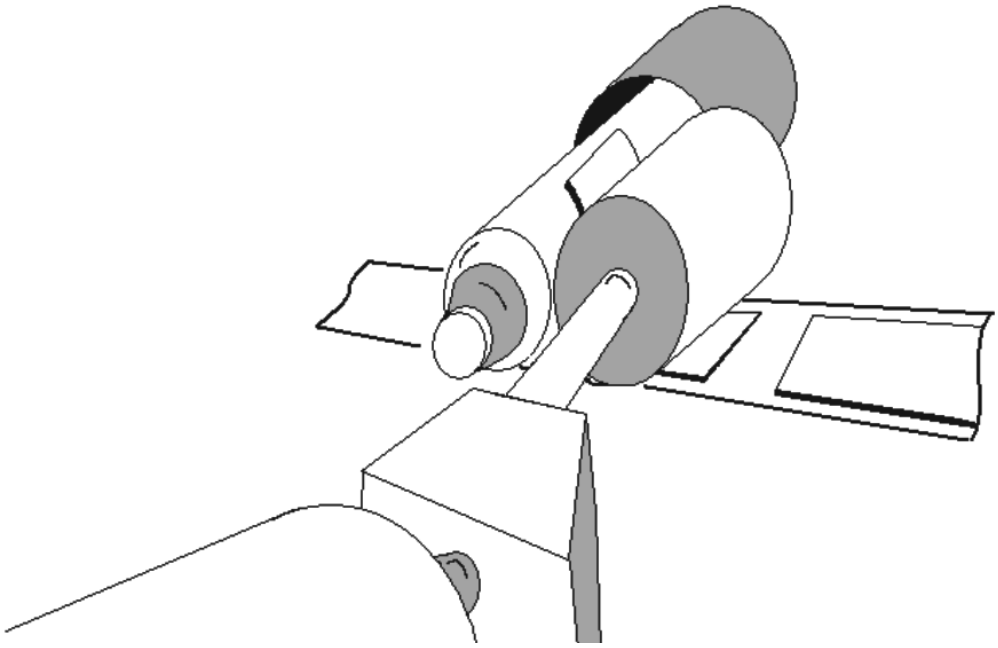


Fig. 4.15 : **Exemple de machine à étiqueter les bouteilles**

6.1.1 Résolution et précision du codeur

Il faut s'assurer que la résolution du codeur dont on dispose (500.000 inc/tour) permet d'atteindre la précision requise.

$\varnothing = 10\ \text{cm} = 100\ \text{mm}$

Périmètre = _____ mm
ex 1

Résolution de codeur minimale requise Re_{min} :

$1\ \text{mm} / (\text{Perimetre}) = Re_{chargemin} / 360°$

$Re_{chargemin} = 360° \cdot 1\ \text{mm} / p =$ _____ °
ex 2

En tenant compte de la boîte de vitesse :

$Re_{moteurmin} = i \cdot Re_{chargemin} =$ _____ °
ex 3

La précision de 1 mm est réalisable avec le codeur dont on dispose si

$$Re_{moteurmin} > \left(360° / \left(\frac{Re_{codeur}}{10}\right)\right)$$ G❶ ✔

_____ ° / Inc > 0,0072 ° / Inc
ex 4

Condition vérifiée.

De plus, la précision de la boîte de vitesse côté charge de $\varphi = \pm¼°$ est à prendre en compte :

$\varphi < Re_{chargemin}$

Enfin, il faut aussi vérifier la précision du codeur.

6.1.2 Analyse du mouvement

Le processus nécessite la séquence de mouvement suivante :

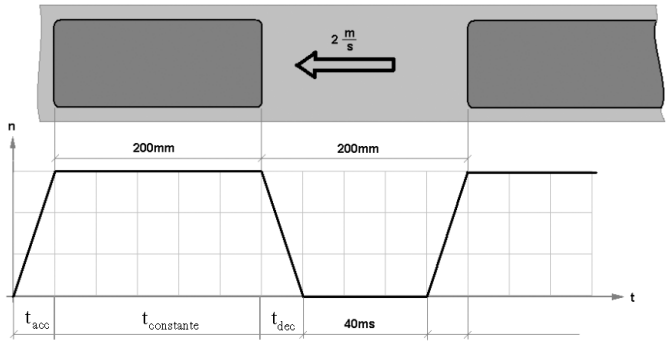


Fig. 4.16 : **Analyse du mouvement**

Détermination des temps :

$$t_{constante} = \frac{s_{longueuretiq}}{v_{tapis}}$$

$$t_{constante} = \frac{0{,}2\ m}{2\ m/s} = \quad s$$

ex 5

La période T est l'inverse de la capacité machine :

$$T = 3600 \cdot \frac{1}{1800\ bouteilles/h} = \quad s$$

ex 6

Le temps d'accélération doit être égal au temps de freinage : $t_{acc} = t_{dec}$ avec

$$T = t_{acc} + t_{constante} + t_{dec} + 40\ ms$$

donc :

$$t_{acc} = t_{dec} = \frac{T - t_{constante} - 40\ ms}{2} = \quad s$$

ex 7

La vitesse maximale côté charge se calcule de la manière suivante :

$$v_{chargemax} = \omega_{chargemax} \cdot \frac{\varnothing}{2} \text{ et } n_{chargemax} = 60 \cdot \frac{\omega}{2 \cdot \pi}$$

$$\omega_{chargemax} = \frac{2 \cdot v_{chargemax}}{\varnothing} = 2 \cdot \frac{2\ m/s}{0{,}1\ m} = \underset{\text{ex 8}}{\qquad} \text{ 1/s}$$

$$n_{chargemax} = 60 \cdot \frac{(2 \cdot v_{chargemax})/\varnothing}{2 \cdot \pi} = \underset{\text{ex 9}}{\qquad} \text{ tour/min}$$

La vitesse maximale côté moteur pour le checkpoint M❶ se calcule avec le rapport de transformation de la boîte de vitesse (i = 6) :

$$i = \frac{\omega_{moteurmax}}{\omega_{chargemax}}$$

$$\omega_{moteurmax} = i \cdot \omega_{chargemax} = \underset{\text{ex 10}}{\qquad} \text{ 1/s}$$

$$n_{moteurmax} = i \cdot n_{chargemax} = \underset{\text{ex 11}}{\qquad} \text{ tour/min}$$ M❶ ✔

6.1.3 Choix d'un moteur

Pour initier les calculs, choisissons dans la liste le moteur suivant : 8MSA3L.E0

Caractéristiques techniques	Symbole	Unité	8MSA3Ldd-eeff		
Nominal Speed	n_N	min^{-1}	3000	4500	6000
Nominal Torque	M_N	Nm	2,15	**2**	1,8
Nominal Current	I_N	A	1,64	**2,2**	**2,3**
Stall Torque	M_0	Nm	2,5	2,5	2,5
Stall Current	I_0	A	1,81	**2,61**	**3,02**
Maximum Pulse Torque	M_{max}	Nm	10	10	10
Maximum Pulse Current	I_{max}	A	8,2	**11,2**	**13**
Maximum Speed	n_{max}	min^{-1}	12000	12000	12000
Torque Constant	K_T	Nm/A	**1,39**	**0,96**	0,83
Voltage Constant	K_E	V/1000 min^{-1}	**84**	**58**	50
Resistance Ph-Ph	R_{2ph}	Ω	**15**	**7**	**5,4**
Inductance Ph-Ph	L_{2ph}	mH	**33,2**	**15,4**	**11,7**
Electrical Time Constant	t_{el}	ms	**21**	**2,2**	**2,17**
Thermal Time Constant	t_{therm}	min	**32**	**32**	**32**
Moment of Inertia without Brake	J	$kgcm^2$	1,4	1,4	1,4
Weight without Brake	m	kg	3,3	3,3	3,3

Fig. 4.17 : **Caractéristiques des servo-moteurs**

6.1.4 Rapport d'inertie

Pour pouvoir assurer une bonne régulation, on doit avoir

$$\frac{J_{charge}}{J_{moteur}} < 20$$

.

Dans la fiche technique du moteur, on a

$$J_{moteur} = \underset{\text{ex 12}}{} \text{ kgm}^2$$

$$J_{charge} = 80 \cdot 10^{-4} \text{ kgm}^2$$

Moment d'inertie de la charge ramené à l'arbre du moteur :

$$J_{charge}^* = J_{charge} \cdot \frac{1}{i^2} = \underline{2{,}22 \cdot 10^{-4} \text{ kgm}^2}$$

$$\frac{J_{charge}}{J_{moteur}} = \frac{2{,}22}{1{,}4} = \underline{1{,}59}$$ M❷ ✔

6.1.5 Couple maximal

Le couple maximal est la somme du couple d'accélération, du couple de frottement et du couple de charge. Le couple de charge et le couple de frottement sont donnés dans le cahier des charges.

Le couple impulsionnel maximal se calcule de la manière suivante :

$$C_{totmax} = C_{acc} + C_f + C_{charge}$$

Le couple de charge est négligé dans cet exemple.

Etant donné la symétrie des rampes dans le diagramme vt, les couples d'accélération et de freinage sont identiques.

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt}$$

$$J_{tot} = J_{moteur} + J_{charge}^{*} = \underline{3{,}62 \cdot 10^{-4}\ kgm^2}$$

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt} = \qquad \text{Nm}$$

ex 13

On obtient ainsi le couple total maximal pour le checkpoint M❸.

$$C_{totmax} = C_{acc} + C_f \approx C_{acc} + 0{,}22\ Nm = \underline{3{,}1\ Nm}$$

M❸ ✔

6.1.6 Valeur efficace du couple

La période (T = 0,2 s) étant inférieure à la constante de temps thermique indiquée dans la fiche moteur (= 32 min), la charge thermique sera déterminée en calculant la valeur efficace du couple.

$$C_{eff} = \sqrt{\frac{1}{T} \cdot \int_0^T C^2 \cdot dt}$$

La valeur efficace de $C_{(t)} = C_{acc} + C_f$ doit être calculée sur la période T. Attention aux signes de C_{acc} et C_f !

$$C_{eff} = \sqrt{\frac{1}{T} \cdot [(C_{acc} + C_f)^2 \cdot t_{acc} + C_f^2 \cdot t_{constante} + (C_{dec} - C_f)^2 \cdot t_{dec}]}$$

C_{eff} = _____ Nm

ex 14

On obtient ainsi la valeur pour le checkpoint M❹. M❹ ✔

6.1.7 Courant maximal de l'ACOPOS

Le courant maximal se calcule en utilisant la constante de couple k_C spécifiée dans la fiche moteur :

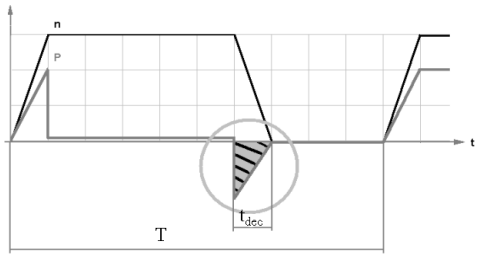
$$I_{totmax} = \frac{C_{totmax}}{k_C} =$$ ex 15 A A❶ ✔

6.1.8 Valeur efficace du courant de l'ACOPOS

La valeur efficace du courant se calcule en utilisant la constante de couple k_C spécifiée dans la fiche moteur :

$$I_{eff} = \frac{C_{eff}}{k_C} =$$ ex 16 A A❷ ✔

6.1.9 Energie de freinage



$$P_{(t)} = C_{(t)} \cdot \omega_{(t)}$$

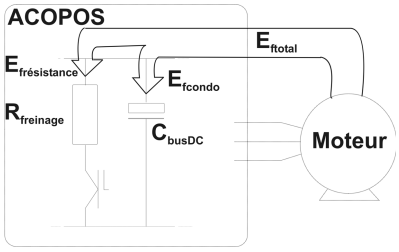
La zone hachurée à l'intérieur de la courbe de puissance correspond à l'énergie de freinage.

Pour déterminer la charge thermique dans la résistance de freinage, on calcule d'abord la puissance de freinage maximale. Ensuite, on calcule la moyenne de la puissance de freinage sur t_{total} afin de déterminer l'énergie de freinage totale. Enfin, on retranche l'énergie absorbée par le condensateur pour pouvoir calculer la puissance dissipée dans la résistance de freinage.

$$P_{fmax} = (C_{acc} - C_f) \cdot \omega_{max} = (2{,}9 - 0{,}22)\ \text{Nm} \cdot 240\ \text{1/s} = 643\ \text{W}$$

$$P_{fmoy} = \frac{P_{fmax} \cdot t_{dec}}{2 \cdot T} = \quad = \underline{\quad\quad} \text{ W}$$

ex 17



$$E_{ftotal} = P_{fmoy} \cdot T$$

$$E_{fcondo} = C \cdot \frac{(U_{max} - U_0)^2}{2}$$

$$E_{fresistance} = E_{ftotal} - E_{fcondo}$$

$$P_{freinage} = \frac{E_{fresistance}}{T}$$

A❹ ✔

6.2 Exemple d'application : carrousel

- Le carrousel se subdivise en 5 segments égaux (360° / 5)
- Le carrousel se déplace selon des cycles périodiques : à chaque cycle, le disque se déplace d'un segment en 30 ms puis s'arrête pendant 170 ms.
- La précision requise pour le positionnement est de ±0,025 mm sur le bord extérieur du disque
- Dimensions du disque : ∅ 10 cm, d = 1 cm ; matériau : acier avec une densité de ρ = 7860 kg/m^3
- Une solution d'entraînement direct ou indirect doit être proposée.

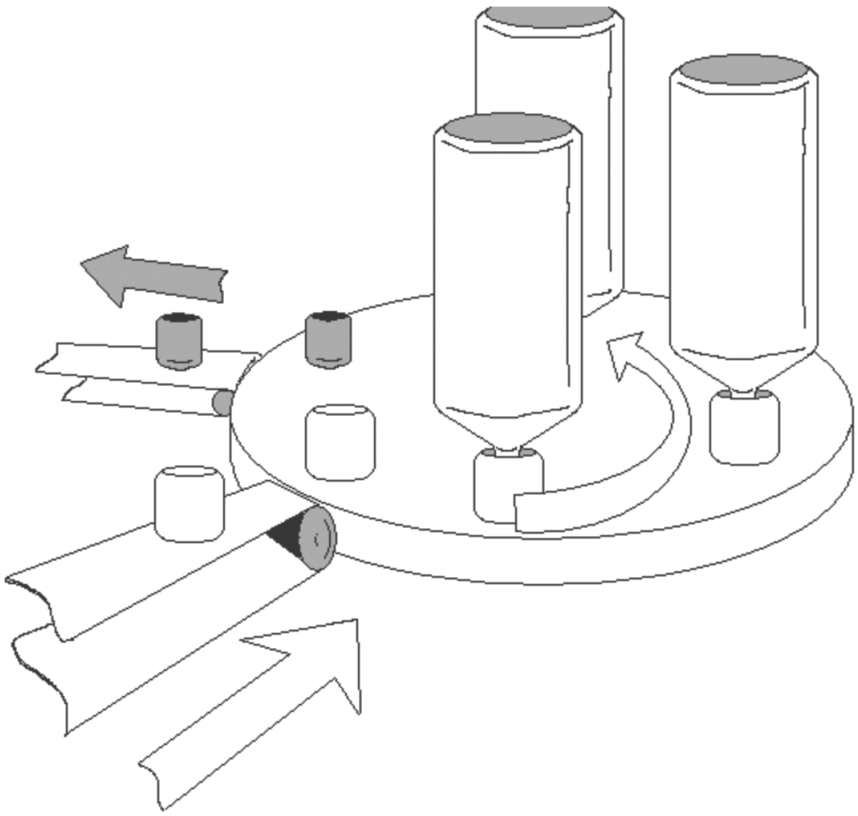


Fig. 4.18 : **Exemple d'application : carrousel**

6.2.1 Résolution et précision du codeur

Il faut s'assurer que la résolution du codeur dont on dispose (500.000 inc/tour) permet d'atteindre la précision requise.

∅ = 10 cm = 100 mm

Périmètre $p =$ _____ mm
ex 18

Résolution de codeur minimale requise Re_{min} :

$Re_{min} =$ _____ $\approx$ 12566 inc/tour
ex 19

La précision de 0,025 mm est réalisable avec le codeur dont on dispose si :

$$Re_{min} = \frac{Re_{codeur}}{10}$$

12566 inc/tour < 50000 inc/tour ✔

Ensuite, la précision du codeur doit être comparée à la précision requise.

6.2.2 Analyse du mouvement

La courbe v/t suivante permet d'obtenir un couple minimum :

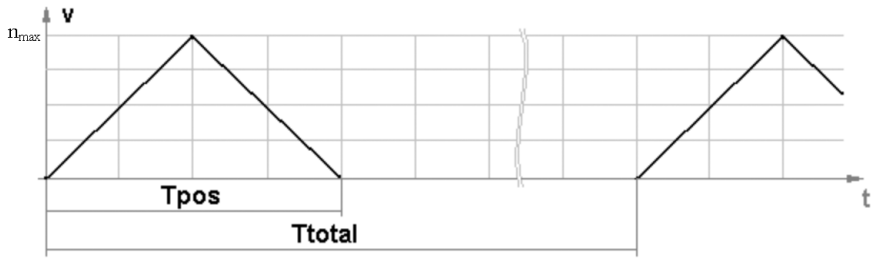


Fig. 4.19 : **Analyse du mouvement**

D'après le cahier des charges :

Tpos= t_{pos} = 0,03 s
Ttotal= t_{total} = 0,2 s

La distance (angle) parcourue par un segment pendant T_{pos} est

φ= ex 20 °

soit, en radian : φ = ex 21 [rad]

D'où :

ω_{max} = ex 22 1/s

La vitesse maximale pour le checkpoint M❶ vaut donc :

n_{max} = ex 23 800 tour/min

6.2.3 Choix d'un moteur

Pour initier les calculs, choisissons dans la liste le moteur suivant : 8MSA3L.E0

Caractéristiques techniques	Symbole	Unité	8MSA3Ldd-eeff		
Nominal Speed	n_N	min^{-1}	3000	4500	6000
Nominal Torque	M_N	Nm	2,15	**2**	1,8
Nominal Current	I_N	A	1,64	**2,2**	**2,3**
Stall Torque	M_0	Nm	2,5	2,5	2,5
Stall Current	I_0	A	1,81	**2,61**	**3,02**
Maximum Pulse Torque	M_{max}	Nm	10	10	10
Maximum Pulse Current	I_{max}	A	8,2	**11,2**	**13**
Maximum Speed	n_{max}	min^{-1}	12000	12000	12000
Torque Constant	K_T	Nm/A	**1,39**	**0,96**	0,83
Voltage Constant	K_E	V/1000 min^{-1}	**84**	**58**	50
Resistance Ph-Ph	R_{2ph}	Ω	**15**	**7**	**5,4**
Inductance Ph-Ph	L_{2ph}	mH	**33,2**	**15,4**	**11,7**
Electrical Time Constant	t_{el}	ms	**21**	**2,2**	**2,17**
Thermal Time Constant	t_{therm}	min	**32**	**32**	**32**
Moment of Inertia without Brake	J	$kgcm^2$	1,4	1,4	1,4
Weight without Brake	m	kg	3,3	3,3	3,3

Fig. 4.20 : **Caractéristiques des servo-moteurs**

6.2.4 Rapport d'inertie

Pour pouvoir assurer une bonne régulation, on doit avoir

$$\frac{J_{charge}}{J_{moteur}} < 20$$

.

Dans la fiche technique du moteur, on a

J_{moteur} = _____ kgm^2
ex 24

Calcul du moment d'inertie en prenant un cylindre pour simplifier :

J_{charge} = _____ kgm^2
ex 25

$$\frac{J_{charge}}{J_{moteur}} = \frac{J_{charge}}{1{,}2\ kg \cdot cm^2} = \underline{6{,}4}$$

6.2.5 Couple maximal

Le couple maximal est la somme du couple d'accélération, du couple de frottement et du couple de charge. Le couple de charge est négligeable dans cet exemple.

Le couple impulsionnel maximal est donc calculé à partir du couple d'accélération et du couple de frottement :

$$C_{totmax} = C_{acc} + C_f$$

Etant donné la symétrie des rampes dans le diagramme vt, les couples d'accélération et de freinage sont identiques.

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt}$$

$$J_{tot} = J_{moteur} + J_{charge} = \underline{8{,}916 \cdot 10^{-4}\ kgm^2}$$

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt} = \quad \text{Nm}$$

ex 26

Couple de frottement : 20% du couple d'accélération

$$C_f = 0{,}2 \cdot C_{acc} \approx \quad \text{Nm}$$

ex 27

On obtient ainsi le couple total maximal pour le checkpoint M❸.

$$C_{totmax} = C_{acc} + C_f \approx \underline{6\ Nm}$$

6.2.6 Valeur efficace du couple

La période (t_{total} = 0,2 s) étant inférieure à la constante de temps thermique indiquée dans la fiche moteur (= 32 min), la charge thermique sera déterminée en calculant la valeur efficace du couple.

$$C_{eff} = \sqrt{\frac{1}{T} \cdot \int_0^T C^2 \cdot dt}$$

La valeur efficace de $C_{(t)} = C_{acc} + C_f$ doit être calculée sur la période t_{total}. Attention aux signes de C_{acc} et C_f !

C_{eff} = Nm

ex 28

Ainsi, on obtient pour le Check point M❹ :

$C_{eff} \approx \underline{\underline{2\ Nm}}$

6.2.7 Courant maximal de l'ACOPOS

Le courant maximal se calcule en utilisant la constante de couple k_C spécifiée dans la fiche moteur :

$$I_{totmax} = \frac{M_{totmax}}{k_C} = \quad \text{A}$$

ex 29

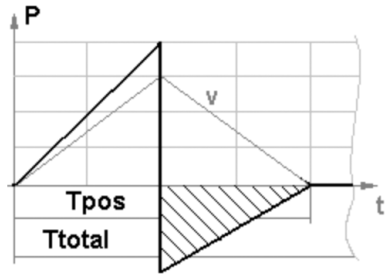
6.2.8 Valeur efficace du courant de l'ACOPOS

La valeur efficace du courant se calcule en utilisant la constante de couple k_C spécifiée dans la fiche moteur :

$$I_{eff} = \frac{C_{eff}}{k_C} = \quad \text{A}$$

ex 30

6.2.9 Energie de freinage



$$P_{(t)} = C_{(t)} \cdot \omega_{(t)}$$

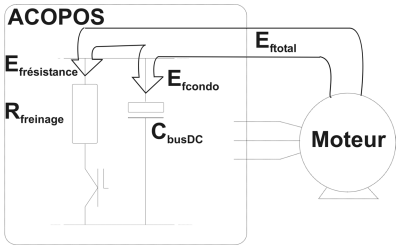
La zone hachurée à l'intérieur de la courbe de puissance représente l'énergie de freinage à dissiper sous forme de chaleur dans la résistance de freinage.

Pour déterminer la charge thermique dans la résistance de freinage, on calcule d'abord la puissance de freinage maximale. Ensuite, on calcule la moyenne de la puissance de freinage sur t_{total} afin de déterminer l'énergie de freinage totale. Enfin, on retranche l'énergie absorbée par le condensateur pour pouvoir calculer la puissance dissipée dans la résistance de freinage.

$P_{fmax} =$

$P_{fmoy} =$

ex 31 et ex 32



$$E_{ftotal} = P_{fmoy} \cdot T_{total}$$

$$E_{fcondo} = C \cdot \frac{(U_{max} - U_0)^2}{2}$$

$$E_{fresistance} = E_{ftotal} - E_{fcondo}$$

$$P_{freinage} = \frac{E_{fresistance}}{T_{total}}$$

A❹ ✔

6.2.10 Calcul du rapport de transformation optimal

Il s'agit maintenant de vérifier si l'utilisation d'une courroie permet d'avoir un moteur de plus petite dimension. A cet effet, la relation entre le couple d'accélération C_{acc} et le rapport de transformation i est représentée graphiquement.

A partir des relations

$$J_{charge}^{*} = J_{charge} \cdot \frac{1}{i^2}$$

$$J_{tot} = J_{moteur} + J_{charge}^{*}$$

$$i = \frac{\omega_{moteur}}{\omega_{charge}}$$

$$C_{acc} = J_{tot} \cdot \frac{d\omega}{dt}$$

on obtient

$$C_{acc} = f(i) =$$

ex 33

Calculez les valeurs de courbe et reportez-les dans la grille ci-dessous :

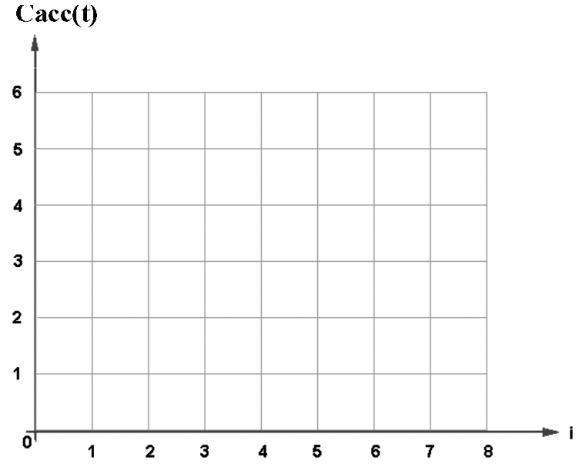


Fig. 4.21 : **Relation couple d'accélération - rapport de transformation**

Le rapport de transformation optimal est ici compris entre i = 2 et i = 3.

Dans les calculs à venir, nous utiliserons le rapport entier i = 3 !

Tous les calculs doivent de nouveau être vérifiés à chaque checkpoint, en tenant compte ici de la boîte de vitesse.

Remarque Pour les calculs, on choisira le moteur de taille immédiatement inférieure, c'est-à-dire 8MSA3M.E0 !

6.3 Exemple d'application : tapis roulant

6.3.1 Cahier des charges

Une masse m = 150 kg doit être déplacée de Δs = 1,2 m en t_{pos} = 0,8 s à l'aide d'un tapis roulant.

Le positionnement se répète avec un cycle de $t_{période}$ = 2 s.

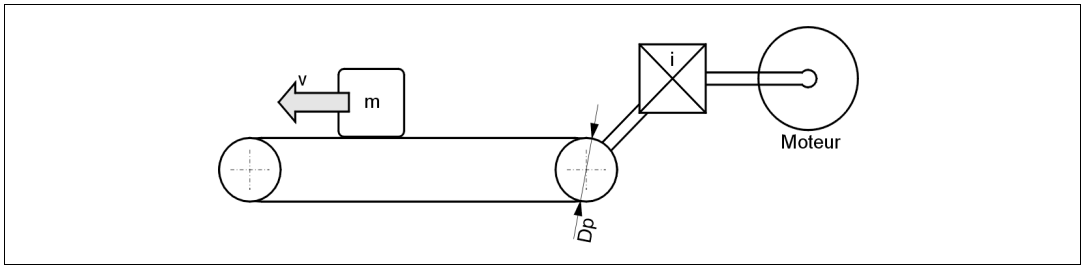


Fig. 4.22 : **Exemple d'application : tapis roulant**

Le diamètre du disque Dp est de 8 cm.

Trouvez le rapport de vitesse i optimal ainsi que le moteur et le variateur appropriés pour cette application !

Remarques

Choisissez un profil de vitesse avec un couple le plus faible possible.

Le moment d'inertie de la charge ramené à l'arbre du moteur se calcule de la manière suivante :

$$J_{charge}^{*} = \frac{1}{2} \cdot m \cdot \left(\frac{Dp}{2}\right)^2 \cdot \frac{1}{i^2}$$

6.4 Exemple d'application : vis à bille horizontale

6.4.1 Cahier des charges

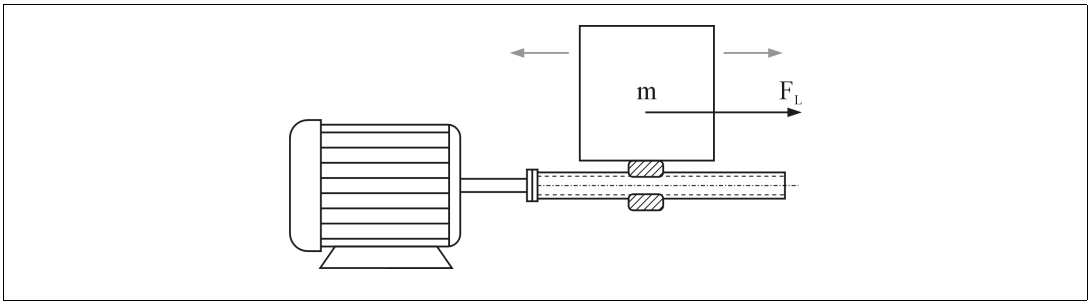


Fig. 4.23 : **Exemple d'application : vis à bille horizontale**

Une masse m = 100 kg se déplace horizontalement sous l'action d'une force d'élasticité constante F_L = 1 kN.

La vis à bille de longueur 415 mm doit déplacer le chariot avec sa masse d'une distance s = 315 mm vers la gauche en t = 0,7 s puis vers la droite avec les mêmes valeurs.

La vis à bille comporte un filetage de pas p = 12 mm pour un diamètre moyen de 32 mm et est en acier (ρ = 7860 kg/m^3)

Effectuez le dimensionnement et sélectionnez un moteur et un variateur appropriés !

Remarques

Le couple d'accélération C_{acc} se calcule de la manière suivante :

$$C_{acc} = F_{total} \cdot \frac{p}{2 \cdot \pi}$$

$$F_{total} = F_L \pm m \cdot a$$

7. LITTÉRATURE

Pour toute information complémentaire, vous pouvez consulter un des nombreux ouvrages disponibles dans la littérature spécialisée.

Suggestions :

Titre de l'ouvrage	Maison d'édition	ISBN
Das 1 x 1 der Antriebsauslegung	VDE	3-8007-2092-2
...	...	...
...	...	...

8. SOLUTIONS

Dans les pages suivantes, vous trouverez des propositions de solution pour les deux premiers exemples.

8.1 Machine à étiqueter des bouteilles

ex 1 ... $p = \varnothing \cdot \pi = 100\ \text{mm} \cdot \pi = \underline{314{,}16\ \text{mm}}$

ex 2 ... $Re_{chargemin} = 360° \cdot \frac{1\ \text{mm}}{314{,}16\ \text{mm}} = \underline{1{,}15°}$

ex 3 ... $Re_{moteumin} = i \cdot Re_{chargemin} = 6 \cdot 1{,}15° = \underline{6{,}88°}$

ex 4 ... $\underline{6{,}88°}$

ex 5 ... $t_{constante} = \underline{100\ \text{ms}}$

ex 6 ... $T = \underline{200\ \text{ms}}$

ex 7 ... $t_{acc} = t_{dec} = \underline{30\ \text{ms}}$

ex 8 ... $\omega_{chargemax} = \underline{40\ 1/\text{s}}$

ex 9 ... $n_{chargemax} = \underline{382\ \text{tour/min}}$

ex 10 ... $\omega_{moteurmax} = 6 \cdot 40\ 1/\text{s} = \underline{240\ 1/\text{s}}$

ex 11 ... $n_{moteurmax} = 6 \cdot 382\ \text{tour/min} = \underline{2292\ \text{tour/min}}$

ex 12 ... $J_{moteur} = \underline{1{,}2 \cdot 10^{-4}\ \text{kgm}^2}$

ex 13 ... $C_{acc} = 3{,}62 \cdot 10^{-4}\ \text{kgm}^2 \cdot \frac{240\ 1/\text{s}}{30\ \text{ms}} = \underline{2{,}9\ \text{Nm}}$

ex 14 ... $C_{eff} = \underline{1{,}6\ \text{Nm}}$

ex 15 ... $$I_{totmax} = \frac{M_{totmax}}{k_C} = \frac{3{,}1\ \text{Nm}}{1{,}38\ \text{Nm/A}} = \underline{2{,}25\ \text{A}}$$

ex 16 ... $$I_{eff} = \frac{C_{eff}}{k_C} = \frac{1{,}6\ \text{Nm}}{1{,}38\ \text{Nm/A}} = \underline{1{,}2\ \text{A}}$$

ex 17 ... $$P_{fmoy} = \frac{(643\ \text{W} \cdot 30\ \text{ms})}{2 \cdot 200\ \text{ms}} = \underline{48\ \text{W}}$$

8.2 Carrousel

ex 18 ... $p = \varnothing \cdot \pi = 100\ \text{mm} \cdot 3{,}14159 = \underline{314{,}159\ \text{mm}}$

ex 19 ... $Re_{min} = \frac{p}{0{,}025\ \text{mm}} = \frac{314{,}159\ \text{mm}}{0{,}025\ \text{mm}} = \underline{12566\ \text{Inc/Tour}}$

ex 20 ... $\varphi = \frac{360°}{5} = \underline{72°}$

ex 21 ... $\varphi = \pi \cdot \frac{72°}{180°} = \underline{1{,}257\ \text{rad}}$

ex 22 ... $\omega_{max} = \frac{d\varphi}{dt} = \frac{\varphi}{t_{pos}/2} = \frac{1{,}257}{0{,}015\ \text{s}} = \underline{83{,}8\ 1/\text{s}}$

ex 23 ... $n_{max} = 60 \cdot \frac{\omega_{max}}{2 \cdot \pi} = 60 \cdot \frac{83{,}8\ 1/\text{s}}{2 \cdot \pi} = \underline{800\ \text{tour/min}}$

ex 24 ... $J_{moteur} = \underline{1{,}2 \cdot 10^{-4}\ \text{kgm}^2}$

ex 25 ... $J_{charge} = \frac{1}{2} \cdot \varnothing^4 \cdot \rho \cdot d \cdot \frac{\pi}{16} = \frac{1}{2} \cdot (0{,}1\ \text{m})^4 \cdot 7860\ \text{kg/m}^3 \cdot 0{,}01\ \frac{\pi}{16}$
$= \underline{7{,}716\ \text{kgcm}^2}$

ex 26 ... $C_{acc} = (7{,}716 + 1{,}2) \cdot 10^{-4}\ \text{kgm}^2 \cdot \frac{83{,}8\ 1/\text{s}}{0{,}015\ \text{s}} = \underline{5\ \text{Nm}}$

ex 27 ... $C_f = 0{,}2 \cdot C_{acc} = 0{,}2 \cdot 5\ \text{Nm} = \underline{1\ \text{Nm}}$

ex 28 ... $C_{eff} = \sqrt{\frac{1}{T_{total}} \cdot \left[(C_{acc} + C_f)^2 \cdot \frac{T_{pos}}{2} + (C_{acc} - C_f)^2 \cdot \frac{T_{pos}}{2}\right]}$

$C_{eff} = \sqrt{\frac{1}{0{,}2\ \text{s}} \cdot \left[(5\ \text{Nm} + 1\ \text{Nm})^2 \cdot \frac{0{,}03\ \text{s}}{2} + (5\ \text{Nm} - 1\ \text{Nm})^2 \cdot \frac{0{,}03\ \text{s}}{2}\right]} = \underline{2\ \text{Nm}}$

ex 29 ...

$$I_{totmax} = \frac{C_{totmax}}{k_C} = \frac{6\ Nm}{1{,}38\ Nm/A} = \underline{4{,}4\ A}$$

ex 30 ...

$$I_{eff} = \frac{C_{eff}}{k_C} = \frac{2\ Nm}{1{,}38\ Nm/A} = \underline{1{,}45\ A}$$

ex 31 ...

$$P_{fmax} = (C_{acc} - C_f) \cdot \omega_{max} = 4\ Nm \cdot 83{,}8\ 1/s = \underline{333\ W}$$

ex 32 ...

$$P_{fmoy} = \frac{P_{fmax}}{2} \cdot \frac{t_{pos}}{2 \cdot t_{total}} = \frac{333}{2}\ W \cdot \frac{0{,}03\ s}{2 \cdot 0{,}2\ s} = \underline{12{,}5\ W}$$

ex 33 ...

$$C_{acc} = f(i) = \left(J_{charge} \cdot \frac{1}{i^2} + J_{moteur}\right) \cdot \frac{\omega_{chargemax} \cdot i}{t_{pos}/2}$$

$$C_{acc} = \left(7{,}716 \cdot \frac{1}{i^2} + 1{,}2\right) \cdot 10^{-4}\ kgm^2 \cdot \frac{83{,}8\ 1/s \cdot i}{0{,}015\ s}$$

CONCEPT NC

1. APERÇU GÉNÉRAL

Le concept NC (commande numérique) de B&R englobe tous les composants requis pour la réalisation des tâches de positionnement.

La programmation **orientée objet** des axes réduit le **temps consacré au développement** et accroît les possibilités de **réutilisation**.

Les paragraphes suivants exposent les différents éléments mis en jeu ainsi que les liens qui existent entre ces éléments.

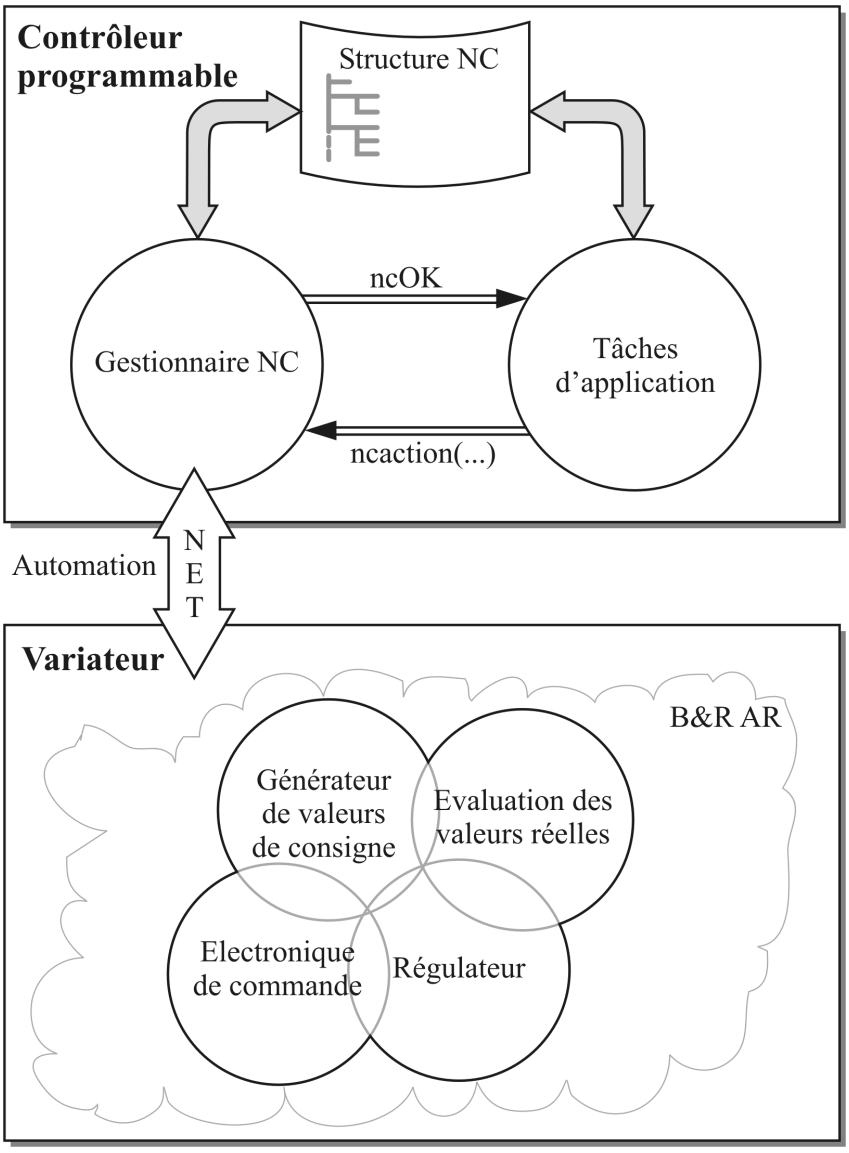


Fig. 5.1 : **Concept NC**

2. VARIATEUR

Le variateur B&R englobe les composants suivants:

- **Générateur de valeurs de consigne**
  Sa fonction est de calculer les valeurs de consigne optimales pour la position à partir des paramètres de l'application, tout en respectant les valeurs maximales définies.
- **Evaluation des valeurs réelles**
  Evaluation des données de codeur, des courants de phase et des valeurs de température.
- **Régulateur**
  Le concept de mise en cascade utilisé par B&R facilite le paramétrage de la régulation.
- **Electronique de commande**
  L'électronique de commande effectue les tâches suivantes :
  - Génération d'impulsion pour les IGBT et résistance de freinage
  - Gestion d'interface
  - Transformation de signaux numériques
  - Commande de frein

Tous ces composants sont gérés par le **B&R Automation Runtime** (système d'exploitation B&R) du variateur.

Il est possible d'effectuer une mise à jour permettant d'utiliser les **fonctions technologiques les plus récentes**. Cette mise à jour consiste en une réactualisation du contrôleur programmable avec l'atelier logiciel **AutomationStudio** (via AutomationNet).

3. LIAISON VARIATEUR – CONTRÔLEUR PROGRAMMABLE

Les données échangées entre le **variateur** et le **contrôleur programmable** via **B&R AutomationNet** sont les suivantes :

- Paramètres
- Données de processus
- Informations d'état

La **configuration** de la liaison (AutomationNet) est définie dans **AutomationStudio**. Exemple avec AutomationNet CAN / Ethernet Powerlink :

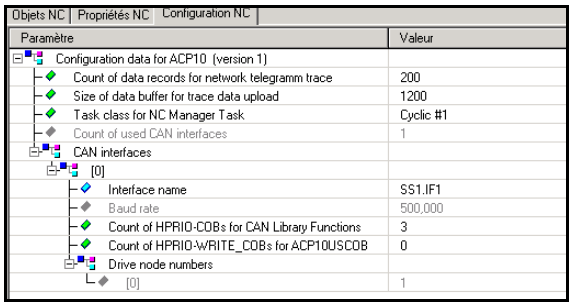


Fig. 5.2 : **Configuration NC**

- Paramètres
  Le contrôleur programmable peut écrire ou lire des paramètres sur le variateur (par exemple, des paramètres de régulateur, de codeur, de moteur, etc.)

- Données de processus
  Le contrôleur programmable peut écrire ou lire des données de processus sur le variateur (valeurs de température, puissance dynamique actuelle, …)

- Informations d'état
  Le variateur envoie automatiquement des valeurs d'état au contrôleur programmable sous forme d'avertissements ou d'erreurs.

4. CONTROLEUR PROGRAMMABLE

La coordination du processus et le contrôle de l'application de positionnement sont assurés par l'interaction des éléments suivants :

- Structure NC
- Tâche d'application
- Gestionnaire NC

Pour une réalisation **simple** et **claire** des tâches de positionnement, B&R propose une **solution orientée objet**.

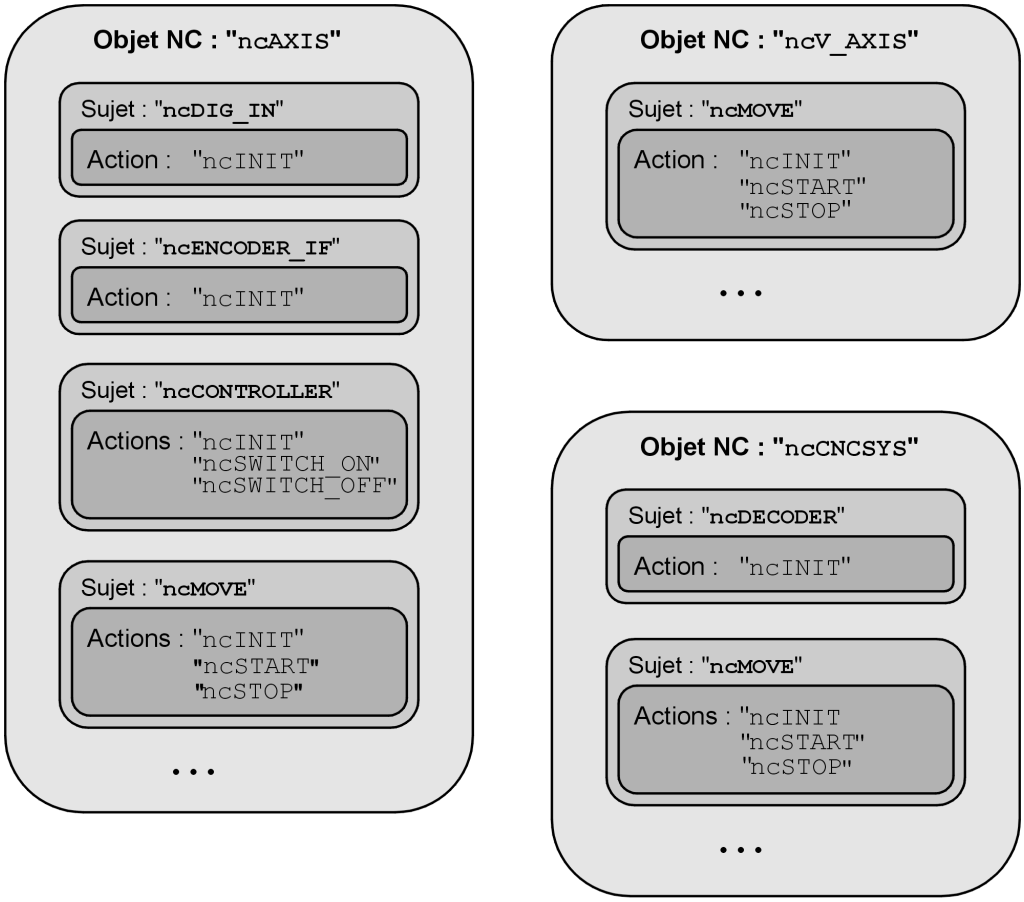


Fig. 5.3 : **Solution orientée objet NC**

4.1 Structure NC

Un axe comprend toute une série de paramètres et de données relatives au processus.

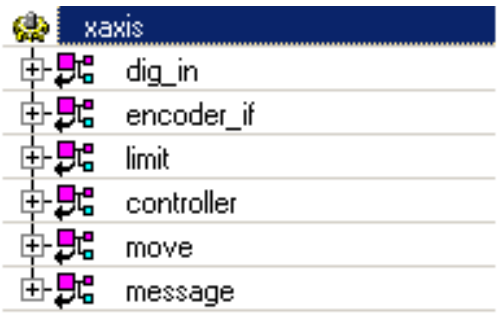
Ces données sont regroupées par thèmes dans une **structure** correspondant au type d'axe (clarté de la présentation).

Le contenu des données est fonction du type d'axe (type d'objet). Un axe réel requiert plus d'informations qu'un axe virtuel (données sur l'interface codeur, par exemple).

Par conséquent, il existe une **structure NC** spécifique pour **chaque type d'objet**.

Type d'objet	Type de structure NC
Axe réel	ACP10AXIS_typ
Axe virtuel	ACP10VAXIS_typ

Les éléments de la structure NC contiennent les paramètres relatifs à chacun des thèmes :



... paramètre d'axe réel
... entrées digitales
... interface codeur
... valeurs limites de l'axe
... paramétrage du régulateur
... paramétrage du mouvement de l'axe

Fig. 5.4 : **Eléments de la structure NC**

Les paramètres pour l'interface codeur, le régulateur etc. sont soit écrits dans la structure NC par l'intermédiaire du programme utilisateur, soit entrés dans un **module de paramètres d'initialisation** d'axe (module de données) et **automatiquement** pris en compte par le gestionnaire NC.

Module de paramètres d'initialisation d'axe :

Fig. 5.5 : **Module de paramètres d'initialisation**

La structure NC comprend également toutes les données d'état et de processus essentielles :

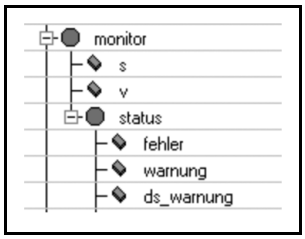


Fig. 5.6 : **Données**

4.2 Tâche d'application

Le **processus de positionnement** est programmé dans une **tâche utilisateur**.

Les données de processus et de paramètre sont entrées dans la structure NC et transférées au variateur grâce à une commande appropriée.

Les commandes de contrôleur programmable destinées à être envoyées au variateur sont lancées par l'appel de la fonction de bibliothèque NC **ncaction(...)**.

Cependant, un objet NC doit d'abord être créé, et ce afin de réserver la mémoire requise pour la structure de données NC. Ceci est réalisé par l'appel de la fonction de bibliothèque NC **ncalloc(.....)**

status= ncalloc(bus_type,module_adr,object_type,channel,adr(nc_object))		
bus_type	UINT	Code d'identification du bus : `ncACP10MAN`
module_adr	UINT	Numéro de nœud du variateur sur le bus

object_type	UINT	Type d'objet :
		`ncAXIS`
		`ncV_AXIS`
		`ncCNCSYS`
		`ncMODULE`

channel	UINT	Numéro de voie : toujours à 1
nc_object	UDINT	Objet NC : pointeur sur la mémoire réservée à la structure de données NC
status	UINT	Etat de la fonction : `ncOK` ou numéro d'erreur

Avec la fonction **ncaction(...)**, il est maintenant possible d'exécuter des actions sur cet objet.

status= ncaction(object,subject,action)		
object	UDINT	Objet NC : pointeur sur la mémoire réservée à la structure de données NC
subject	UDINT	Partie de l'objet NC = Thème
action	UDINT	Action à effectuer
status	UINT	Etat de la fonction : `ncOK` ... Instruction reportée `ncACTIVE` ... Instruction non reportée ou état d'erreur

Une action lancée par ncaction(...) est reportée dans la file d'attente du gestionnaire NC réservée aux commandes. Le bon report de l'action est confirmé à l'application par l'**état ncOK**. Dans le cas où la file d'attente des commandes est pleine et ne peut plus recevoir de nouvelles commandes venant de l'application, l'appelant de la fonction reçoit l'**état ncACTIVE**.

Ceci se répète jusqu'au moment où la file d'attente est à nouveau libre pour accepter la commande.

IMPORTANT

Lors de l'appel des fonctions ncalloc(.....) ou ncaction(...), l'**état des fonctions** doit toujours être lu pour que l'exécution du programme puisse se poursuivre !

4.3 Gestionnaire NC

Fig. 5.7 : **Fichier Gestionnaire dans l'arborescence logicielle**

Le gestionnaire NC est une extension du système d'exploitation dont la fonction est de faire le lien entre l'application de positionnement et le variateur.

Les tâches effectuées par le gestionnaire NC sont les suivantes :

- Transférer les commandes NC (actions) au variateur
- Mettre à disposition les informations d'état du variateur
- Transférer le système d'exploitation du variateur

MOTION COMPONENTS

1. INSTALLATION DES MOTION COMPONENTS

Les Motion Components font partie intégrante d'AutomationStudio™ et peuvent être installés à n'importe quel moment à partir du CD d'installation d'AutomationSoftware™.

Dans le répertoire racine du CD d'installation, on trouve le fichier "BrMenu.exe". Un double-clic sur ce fichier lance l'assistant d'installation destiné à guider l'utilisateur.

Dans la liste de sélection du fichier BrMenu.exe, on trouve également le gestionnaire ODBC pour la gestion des bases de données.

Lors de la **sélection des composants**, l'utilisateur peut choisir de **(ré)installer** ou de désinstaller certaines parties ciblées d'AutomationStudio™.

2. GESTION DE PROJET DANS AUTOMATION STUDIO

L'intégration du concept de variateur B&R à AutomationStudio™ grâce aux Motion Components permet l'élaboration, la gestion complète et la mise en service des tâches de contrôle et de positionnement.

Les coûts d'ingéniérie liés au logiciel, à la mise en service et à la maintenance sont ainsi considérablement réduits.

Les outils disponibles se répartissent en quatre catégories :

- Composants matériels
- Composants logiciels
- Mise à jour d'AR pour ACOPOS
- Centre de test

2.1 Composants matériels

Nous allons gérer dans AutomationStudio™ la configuration matérielle suivante :

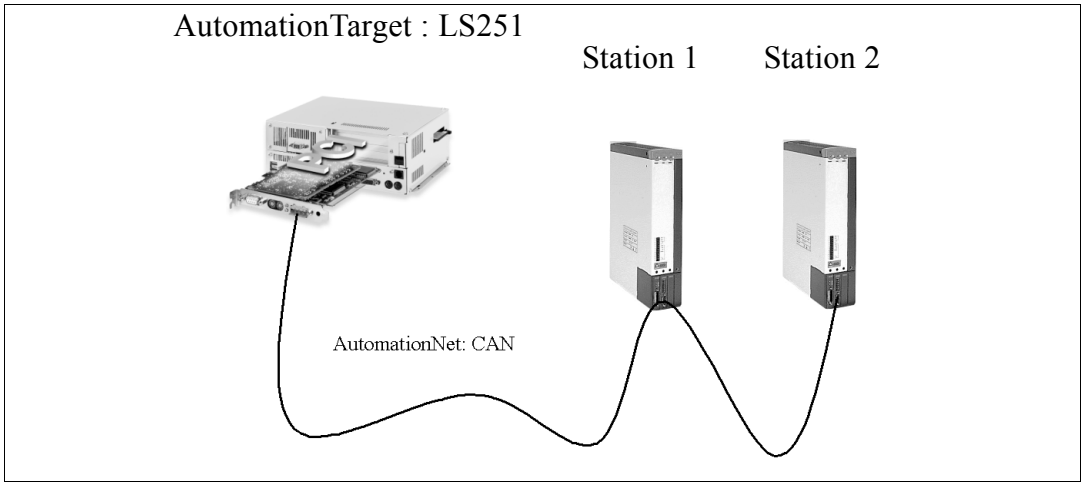


Fig. 6.1 : **Structure matérielle**

2.1.1 Choix et gestion d'un esclave d'E/S CAN

Insertion de l'ACOPOS en tant que participant au réseau Automation Net : CAN.

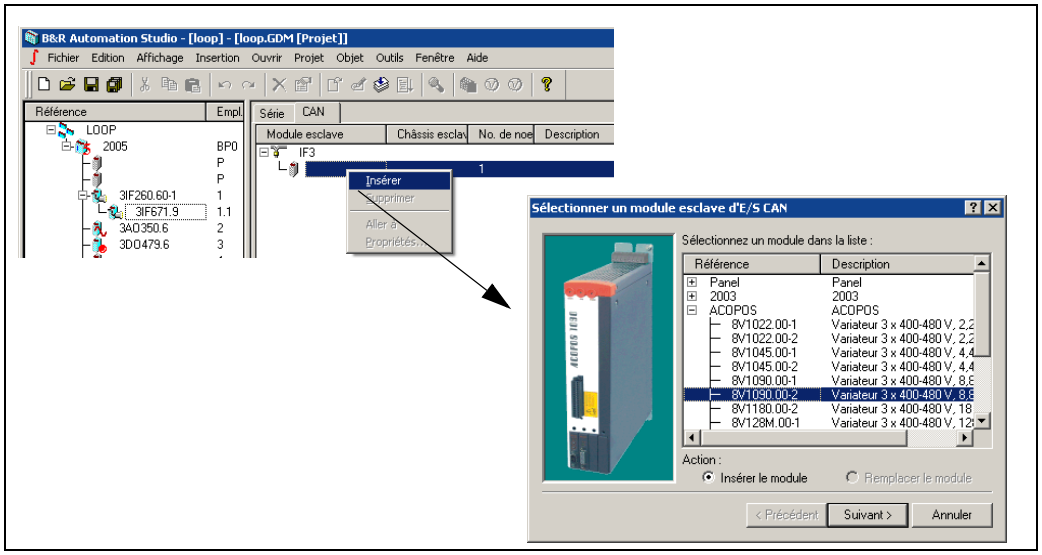


Fig. 6.2 : **Insertion de l'ACOPOS**

2.1.2 Détermination des paramètres de connexion

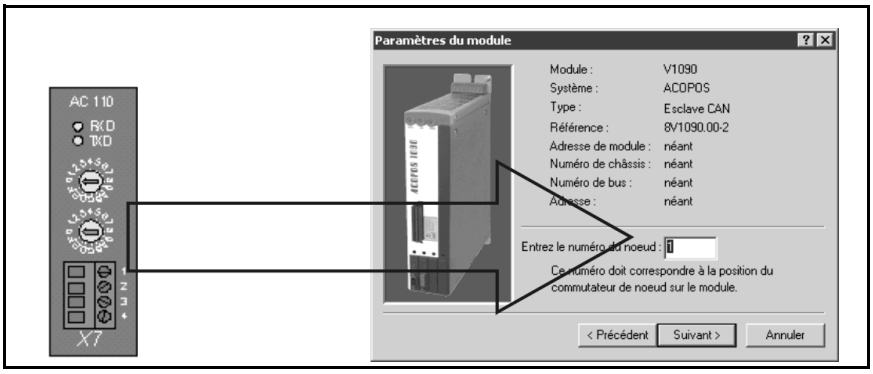


Fig. 6.3 : **Définition du numéro de station**

Le débit de transmission est préréglé à 500 kbauds. Vient ensuite le choix des modules enfichables. Il est possible d'effectuer des modifications via la commande Propriétés du menu Edition !

2.1.3 Choix des cartes enfichables pour l'ACOPOS

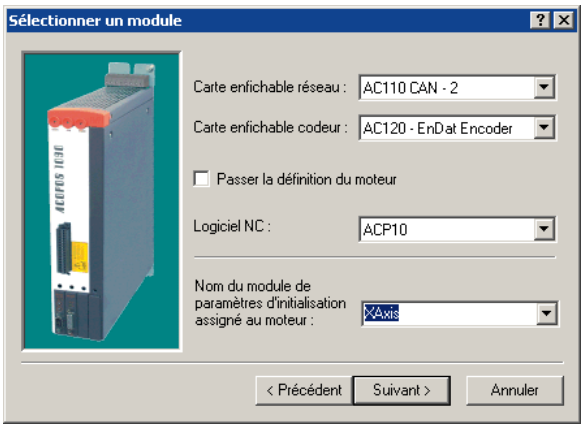


Fig. 6.4 : **Choix des modules**

2.1.4 Choix du moteur

Si vous utilisez un moteur B&R, vous pouvez le définir soit à l'aide de l'Assistant en spécifiant ses paramètres, soit en entrant directement sa référence.

Si vous utilisez un autre moteur (moteur OEM par exemple), vous pouvez entrer tous les paramètres requis pour l'ACOPOS.

Les données de description sont mémorisées dans un module de paramètres que vous aurez préalablement nommé.

Fig. 6.5 : **Choix du moteur**

L'Assistant vous guide dans la définition des paramètres du moteur et du codeur.

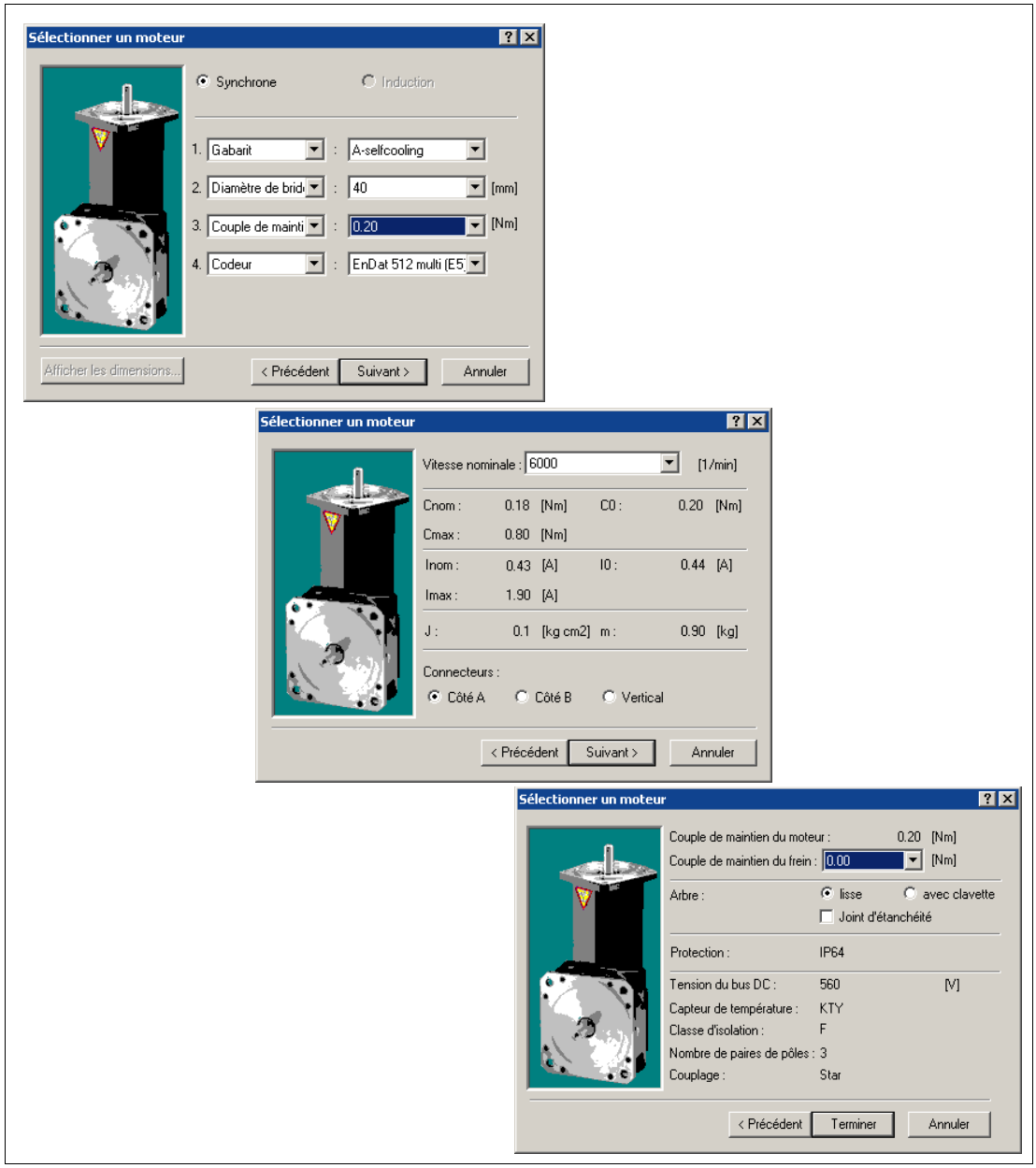
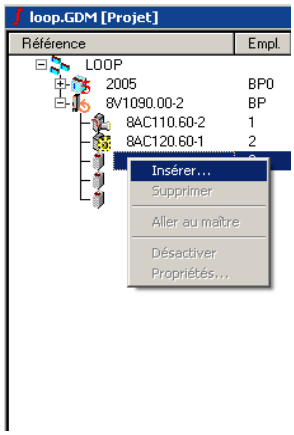


Fig. 6.6 : **Choix du moteur**

2.1.5 Sélection manuelle

- Sélection des cartes enfichables :



Exemple :

- Cartes d'interface AutomationNet
- Interface ENDAT
- Interface résolver
- Interface d'E/S digitales

Fig. 6.7 : **Insertion des cartes dans l'arborescence matérielle**

- Sélection du moteur et du codeur :

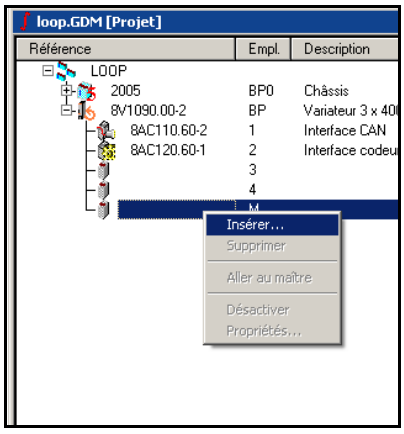


Fig. 6.8 : **Insertion d'un moteur dans l'arborescence matérielle**

Insertion automatique des fichiers système requis dans le projet :

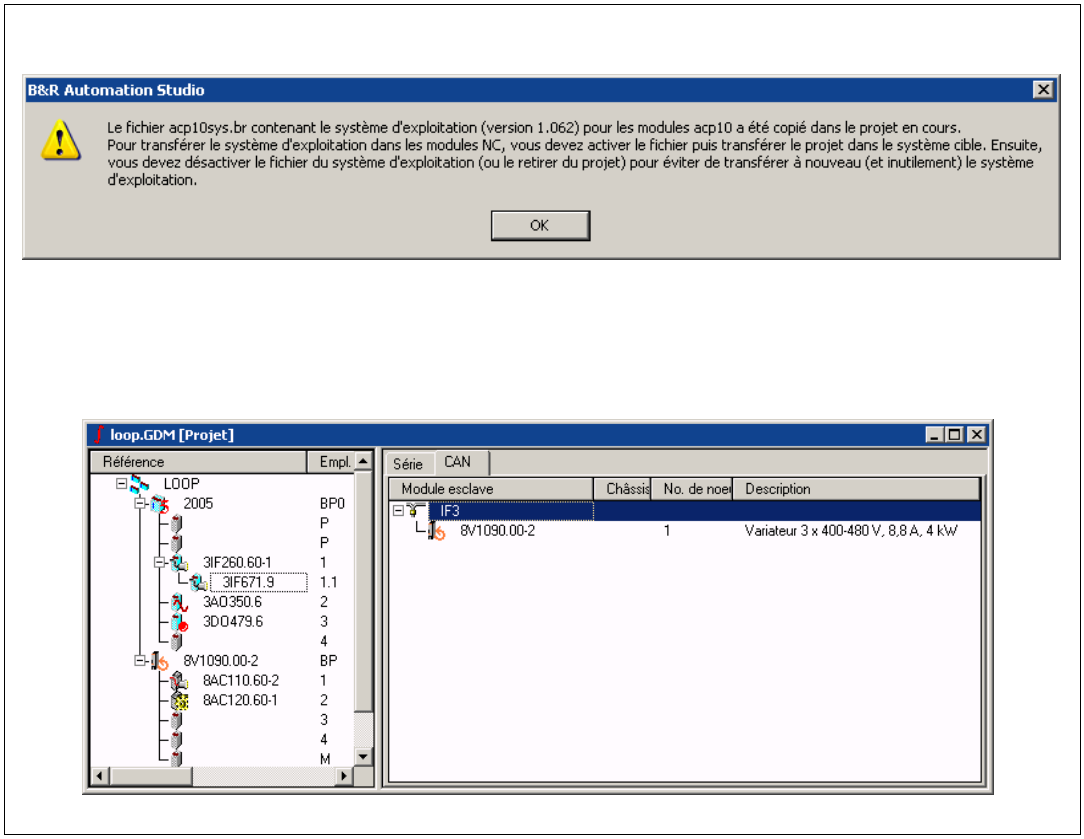


Fig. 6.9 : **Insertion automatique des fichiers système**

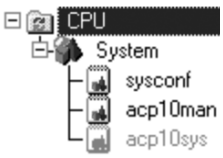
2.2 Composants logiciels

Les Motion Components permettent d'utiliser les modules logiciels appropriés pour la réalisation des tâches de positionnement.

Ces modules logiciels se répartissent en trois catégories :

- **Objets système**
- **Objets Motion (objets experts)**
- **Tâches d'application (coordination du processus)**

2.2.1 Objets système

L'ACOPOS AR est doté de fonctionnalités spéciales pour applications de positionnement.	

Le gestionnaire ACOPOS (acp10man) gère la liaison entre le contrôleur programmable et le variateur et joue un rôle d'interface sous forme de bibliothèque pour tâches d'application .

Le gestionnaire ACOPOS et l'ACOPOS AR sont automatiquement insérés dans le projet par AutomationStudio™ lors de l'insertion du matériel correspondant.

La procédure à suivre dans **AutomationStudio™** pour la **mise à jour automatique** des fichiers système est la suivante :

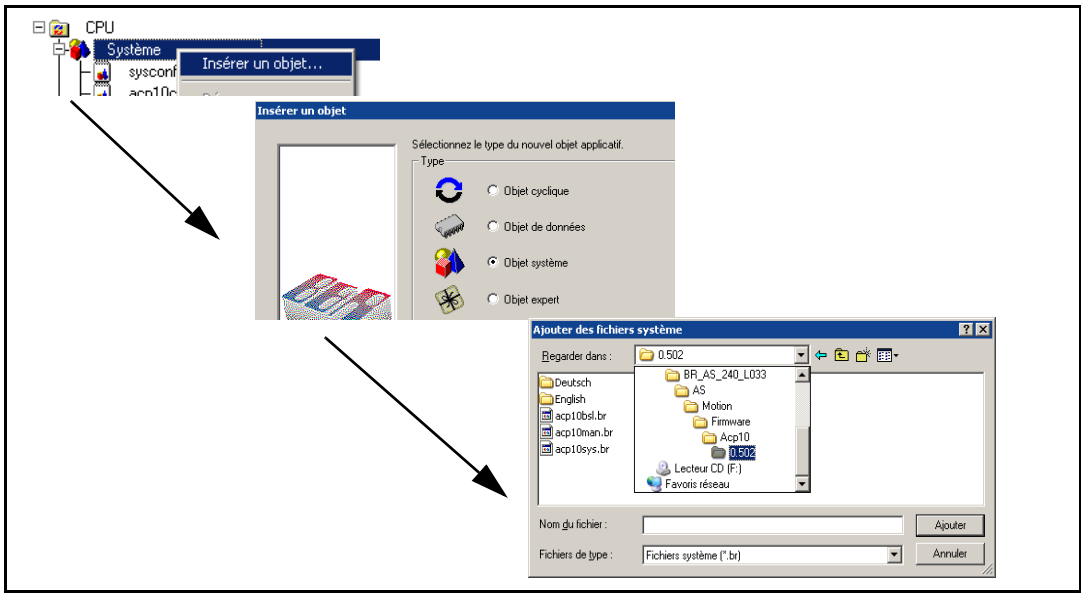


Fig. 6.10 : **Insertion d'objets logiciels**

2.2.2 Objets Motion (objets experts)

Pour simplifier l'élaboration et la gestion du projet, vous disposez d'objets Motion pour vos différentes tâches.

Ces objets peuvent être insérés comme des tâches avec le bouton droit de la souris.

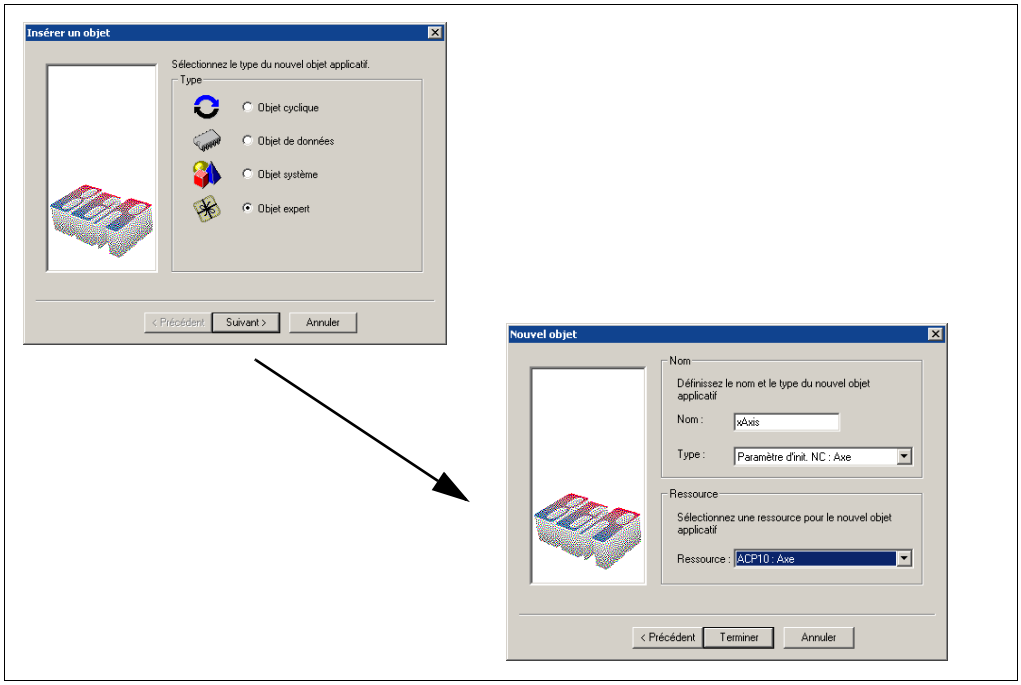
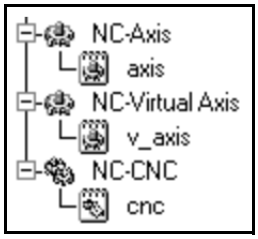


Fig. 6.11 : **Insertion d'un objet axe**

Le système cible **ACOPOS** doit être sélectionné dans le champ **Ressource** !

2.2.3 Modules de paramètres d'initialisation

Ces paramètres peuvent être déjà prédéfinis dans un groupe de modules puis transférés au contrôleur programmable.



Au **démarrage du contrôleur programmable**, le gestionnaire ACOPOS vient **lire** les paramètres dans les modules de paramètres d'initialisation et les transmet aux structures d'axe correspondantes lors de l'exécution **automatique** des **commandes d'initialisation**.

La **relation** entre **le module de paramètres d'initialisation** et l'**objet axe** est établie dans le panneau des Propriétés de l'ACOPOS en **reliant** l'objet NC à un objet axe.

Exemple

Elaboration d'un projet pour l'application du carrousel

- Créer un nouveau projet nommé "proj_rt"
- Définir la configuration matérielle du châssis principal (par téléchargement automatique)
- Insérer un ACOPOS sous CAN et régler le numéro de noeud = numéro de station à 1
- Créer un module de paramètres d'initialisation nommé "rtaxpara" dans l'arborescence logicielle
- Relier le module de paramètre d'initialisation à un axe réel sur l'ACOPOS

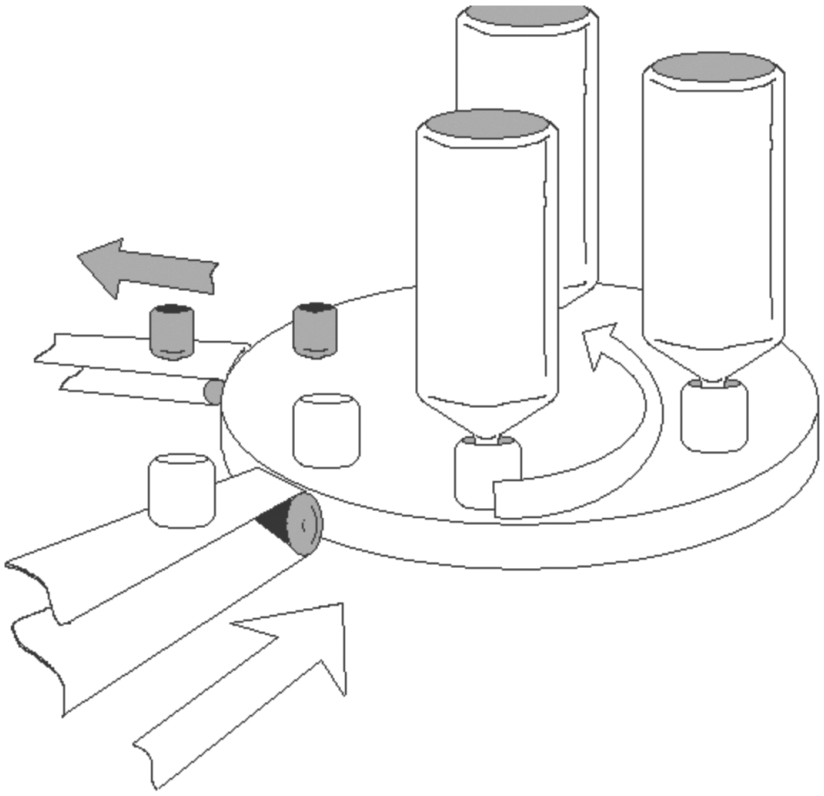


Fig. 6.12 : **Application du carrousel I**

2.3 Tâches d'application

La **coordination des séquences** de mouvements d'axe est commandée et contrôlée à partir de l'applicatif. Certaines fonctions de la bibliothèque ACOPOS gérées par le gestionnaire ACOPOS sont prévues à cet effet.

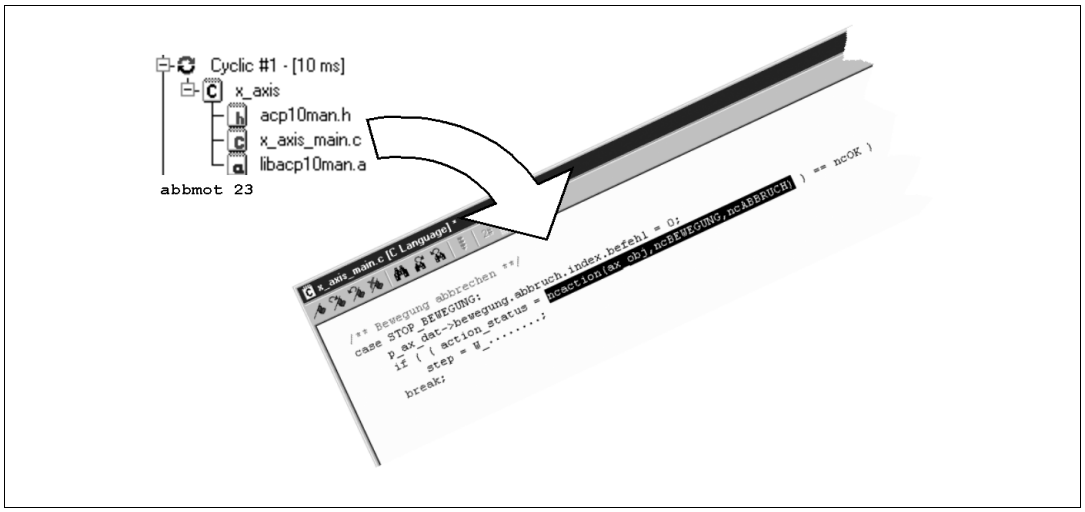


Fig. 6.13 : **Tâche avec coordination des séquences de mouvement**

Vous pouvez choisir C ANSI ou AutomationBasic comme **langage de programmation**.

Automation Studio™ ajoute automatiquement au projet en cours les **extensions** propres au langage de programmation sélectionné. Exemple : pour une tâche C, le fichier en-tête "acp10man.h" et le fichier archive "libacp10man.a" en **anglais** ou en **allemand**.

2.3.1 Changement de langue : anglais - allemand

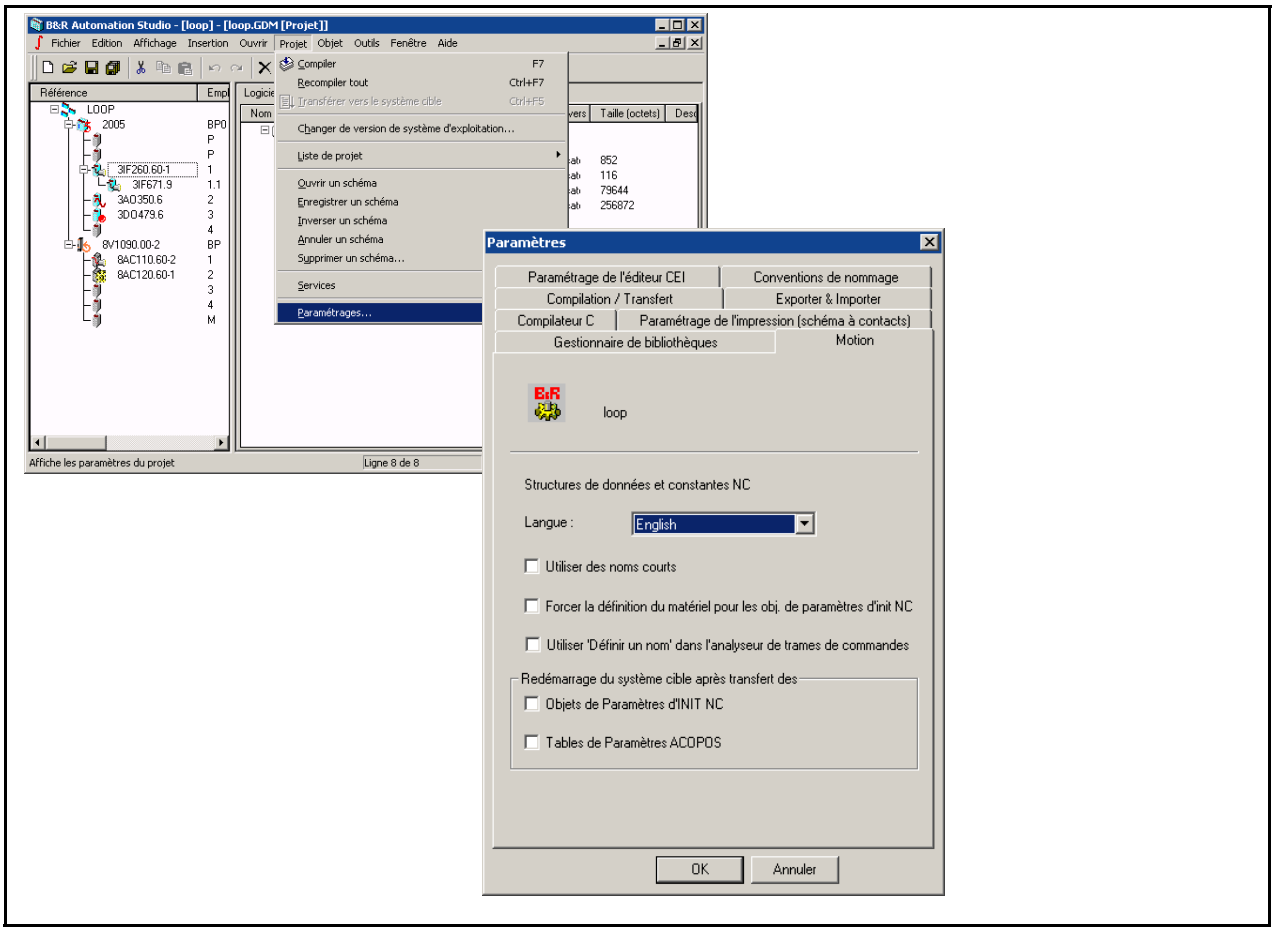


Fig. 6.14 : **Changement de langue et noms courts**

Exemple

Elaboration d'un projet pour l'application du carrousel (suite)

- Insérer une tâche C nommée "rtablauf" dans l'arborescence logicielle
- Ajouter à la tâche C le fichier en-tête ACP10 et le fichier archive (acp10man.h + libacp10man.a)
- Sélectionner la langue souhaitée (allemand/anglais)

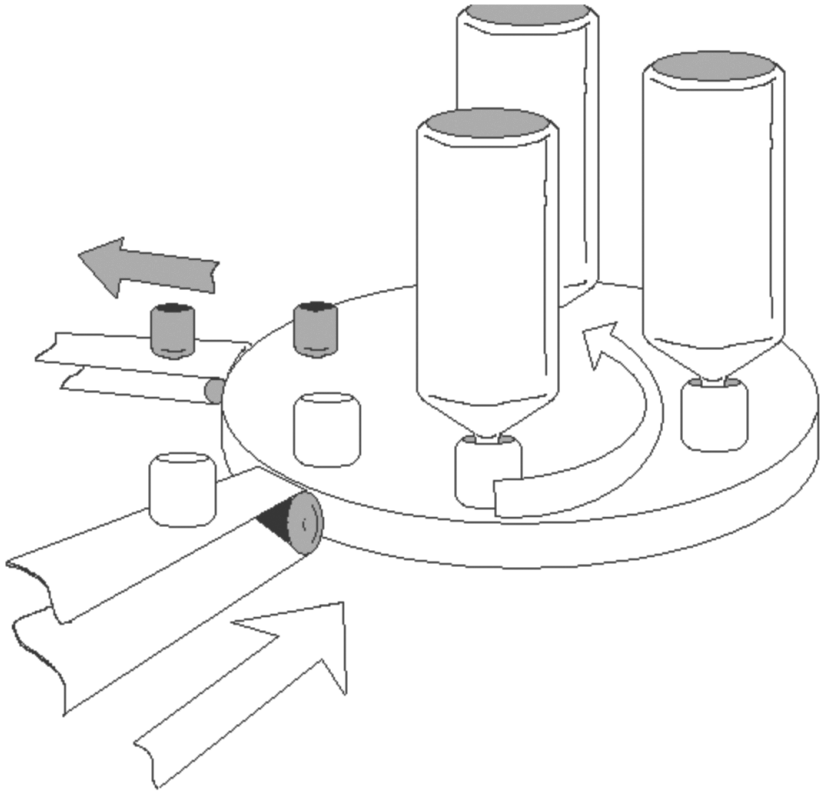


Fig. 6.15 : **Application du carrousel II**

2.4 Mise à jour d'ACOPOS AR

B&R s'efforce en permanence d'offrir à l'utilisateur de **nouvelles fonctionnalités** pour augmenter les performances et réduire les coûts d'ingéniérie.

Pour doter votre application de nouvelles fonctionnalités, vous pouvez installer un nouvel ACOPOS AR sur l'AT avec Automation Studio™.

Deux possibilités s'offrent à vous pour l'installation de l'ACOPOS AR sur le variateur :

- ACOPOS AR manuel
- ACOPOS AR embarqué

2.4.1 ACP AR manuel :

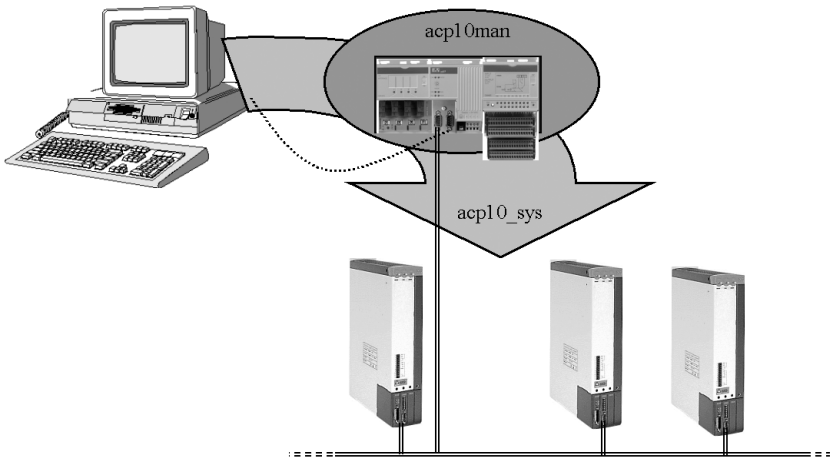


Fig. 6.16 : **Transfert de l'AR pour l'ACOPOS**

Le fichier d'ACOPOS AR (= acp10_sys) est chargé dans le contrôleur programmable (transfert) et ensuite distribué par le gestionnaire ACOPOS aux différents variateurs.

Après le transfert, le fichier d'ACOPOS AR est automatiquement supprimé dans le contrôleur programmable.

Remarque

Pour qu'une mise à jour d'ACP AR puisse être effectuée, les variateurs ACOPOS doivent avoir été préalablement **définis dans l'arborescence matérielle**.

2.4.2 ACOPOS AR embarqué

Le fichier d'ACOPOS AR est stocké dans le contrôleur sous la forme d'un module de données.

Avantage : **la mise à jour automatique** du système d'exploitation**, sans** l'aide de l'**outil de programmation, par le contrôleur programmable lui-même** lorsque l'on connecte un nouveau variateur dépourvu d'ACOPOS AR ou comportant un ACOPOS AR plus ancien.

Cependant, la mémoire requise dans le contrôleur programmable est alors beaucoup plus importante !

Exemple

Elaboration d'un projet pour l'application du carrousel (suite)

- Mettre à jour l'ACOPOS AR avec la méthode manuelle
- Vérifier les LED d'état sur l'ACOPOS
- Transférer le module de paramètres d'initialisation au contrôleur programmable

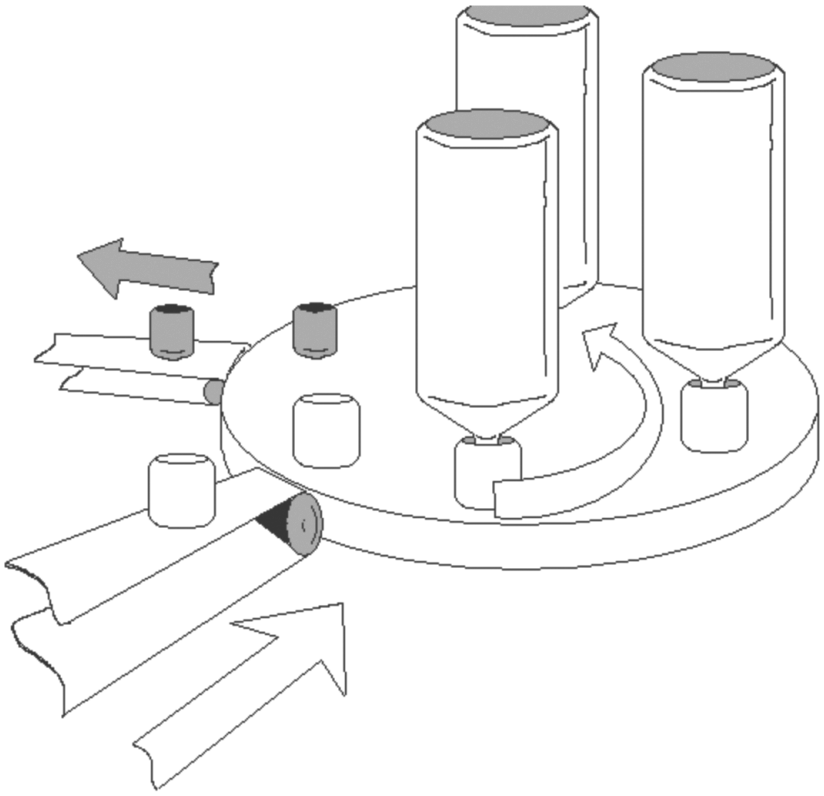


Fig. 6.17 : **Application du carrousel III**

2.5 Editeurs NC

La création des différents objets Motion se fait dans Automation Studio™ avec les éditeurs respectifs.

Des **masques d'édition** prédéfinis facilitent la saisie et permettent de définir le lien avec l'objet axe, et ce en **allemand** ou en **anglais**.

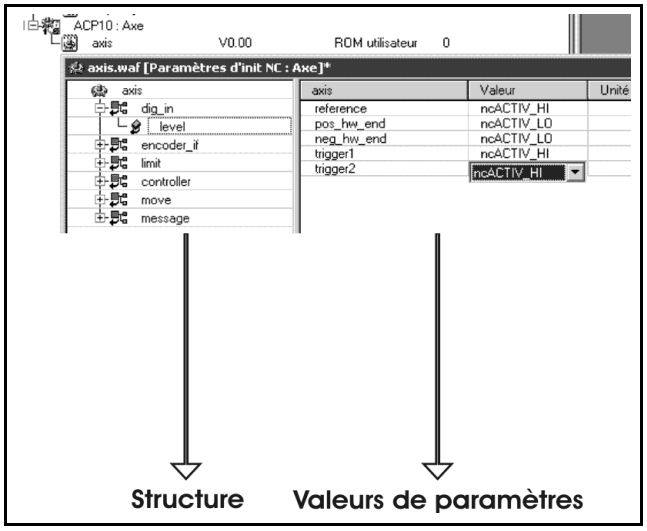


Fig. 6.18 : **Module de paramètres d'initialisation**

2.6 Diagnostic NC

Après l'élaboration du projet et la programmation vient la **mise en service** des axes. Le choix des paramètres de régulation appropriés est ici d'une importance capitale.

Les performances de la machine et donc la qualité du produit vont de pair avec une dynamique d'axe élevée, obtenue grâce au **paramétrage optimal** de la régulation.

Aussi, dans le passé, ce travail de paramétrage essentiel était effectué avec des appareils de mesure et d'analyse coûteux (oscilloscope à mémoire par exemple).

Automation Studio™, avec les Motion Components, met à la disposition de l'utilisateur toute une série d'outils de diagnostic et d'analyse intégrés. Ces derniers satisfont tout type d'exigence et permettent une mise en service rapide et précise d'axes simples ou de groupes d'axes.

Un « centre de test » regroupe les outils de diagnostic et d'analyse suivants :

- Interface de commande
- Interface de paramètre
- Moniteur de variables
- Oscilloscope intégré

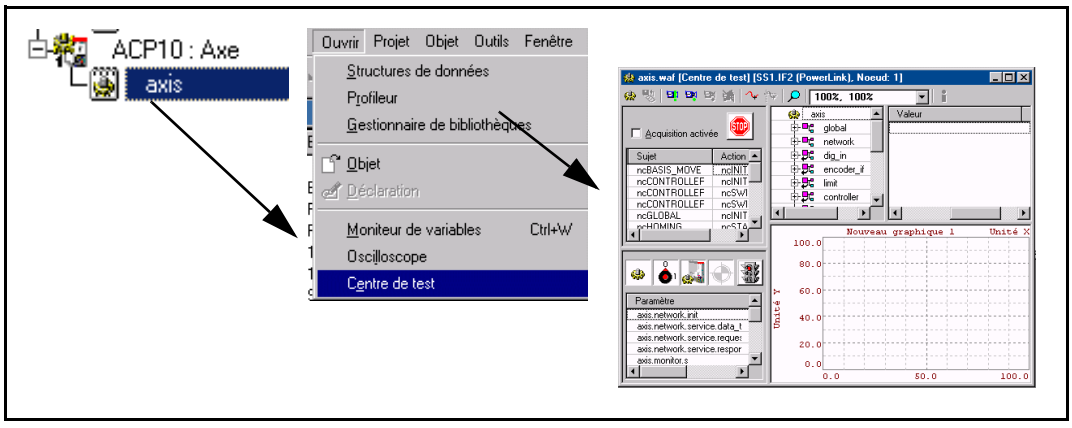
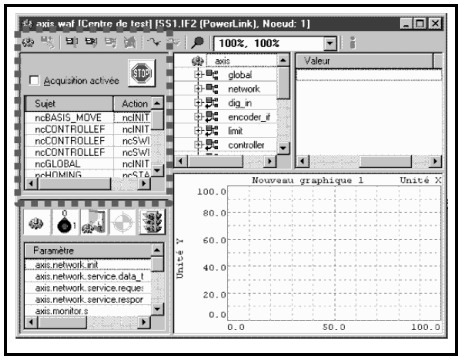


Fig. 6.19 : **Ouverture du centre de test**

2.6.1 Interface de commande



Lors de la mise en service, il est souvent plus facile de réaliser des mouvements par commande directe, sans passer par l'applicatif.

Pour cette raison, le centre de test comprend une interface permettant d'exécuter des commandes **manuellement**.

Des **"Sujets/Actions"** préprogrammés peuvent être insérés dans l'interface de commande et **exécutés** aussi souvent que nécessaire en appelant la fonction "ncaction(..)".

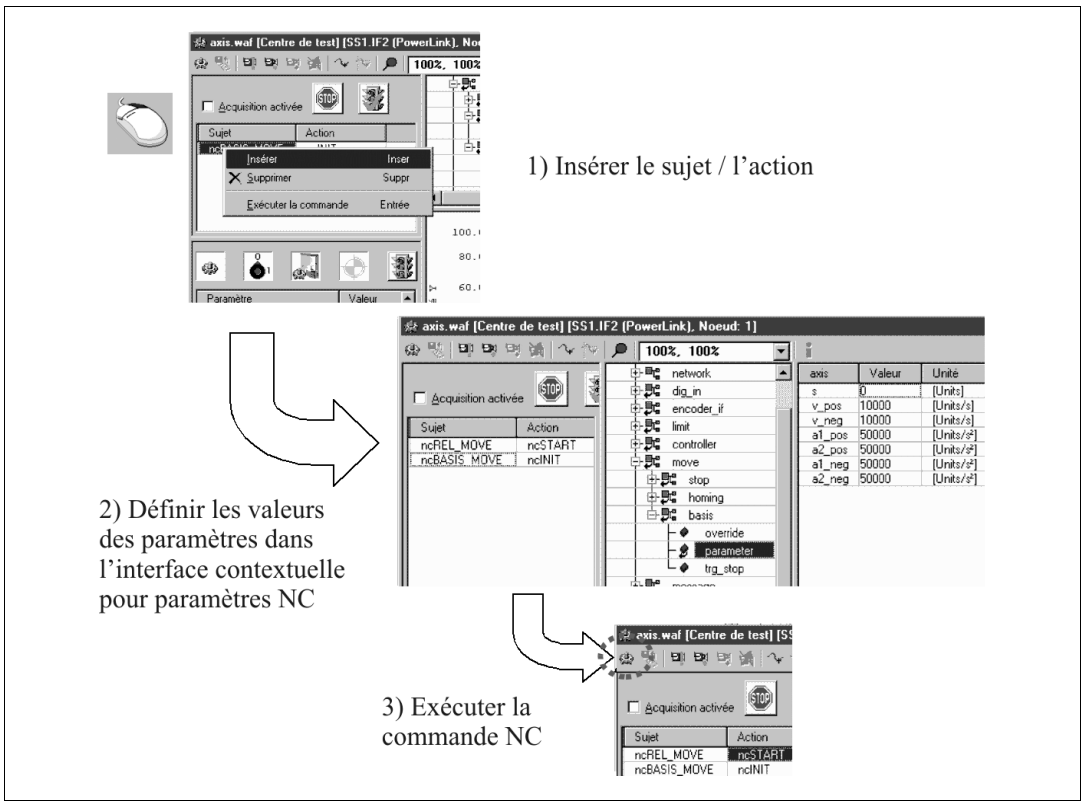


Fig. 6.20 : **Interface de commande dans le centre de test**

2.6.2 Moniteur de variables

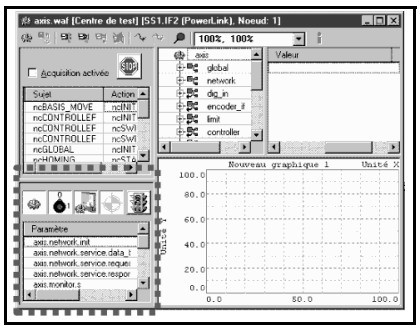


Fig. 6.21 : **Centre de test - Fenêtre du moniteur de variables**

- **Informations d'état**
  Les icônes d'état donnent un aperçu global des informations d'état les plus utilisées.

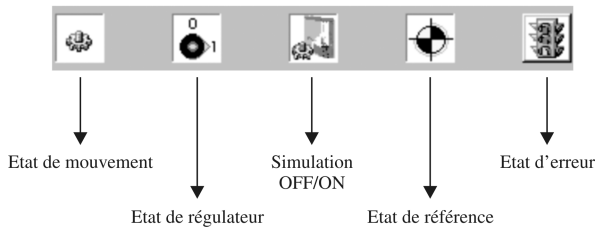
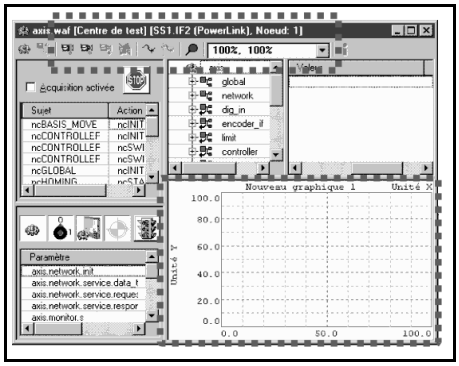


Fig. 6.22 : **Affichage d'état**

- **Champ de variables**
  Les éléments de structures d'axe peuvent être insérés et visualisés dans ce champ.

- **Noms courts ou longs**
  Avec la commande Paramétrages du menu Projet, on peut choisir une représentation des informations d'axe sous forme de texte court ou long.

2.6.3 Oscilloscope intégré



La fonction Oscilloscope des Motion Components permet d'effectuer **une acquisition de données en temps réel directement dans l'ACOPOS**. Les données d'acquisition sont chargées par l'ACOPOS dans Automation Studio™ via le contrôleur programmable, puis **affichées graphiquement**.

L'oscilloscope permet d'enregistrer des séquences de mouvements sans perte de précision ni charge additionnelle.

- **Barre d'outils de l'oscilloscope**
  Une barre d'outils regroupant les éléments de commande les plus importants facilite l'utilisation de l'oscilloscope.

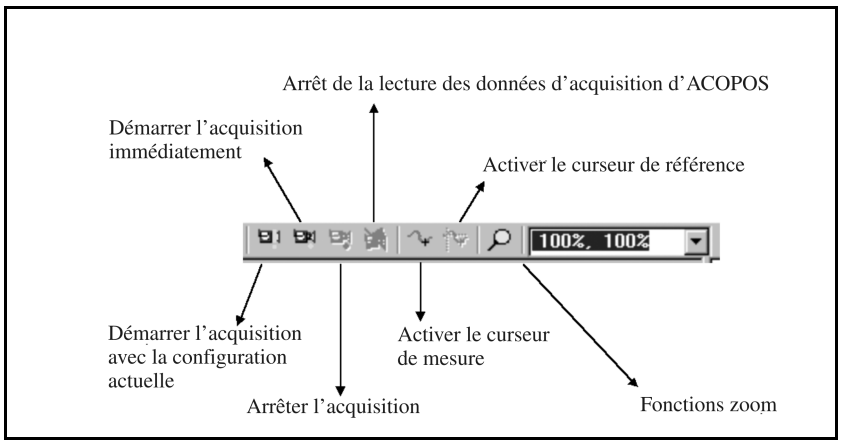


Fig. 6.23 : **Icônes de l'oscilloscope**

- **Paramétrage de l'oscilloscope intégré**
  L'utilisateur dispose de nombreuses possibilités de réglage pour les paramètres d'acquisition. Un clic droit de souris dans la fenêtre de l'oscilloscope permet d'éditer ces paramètres.

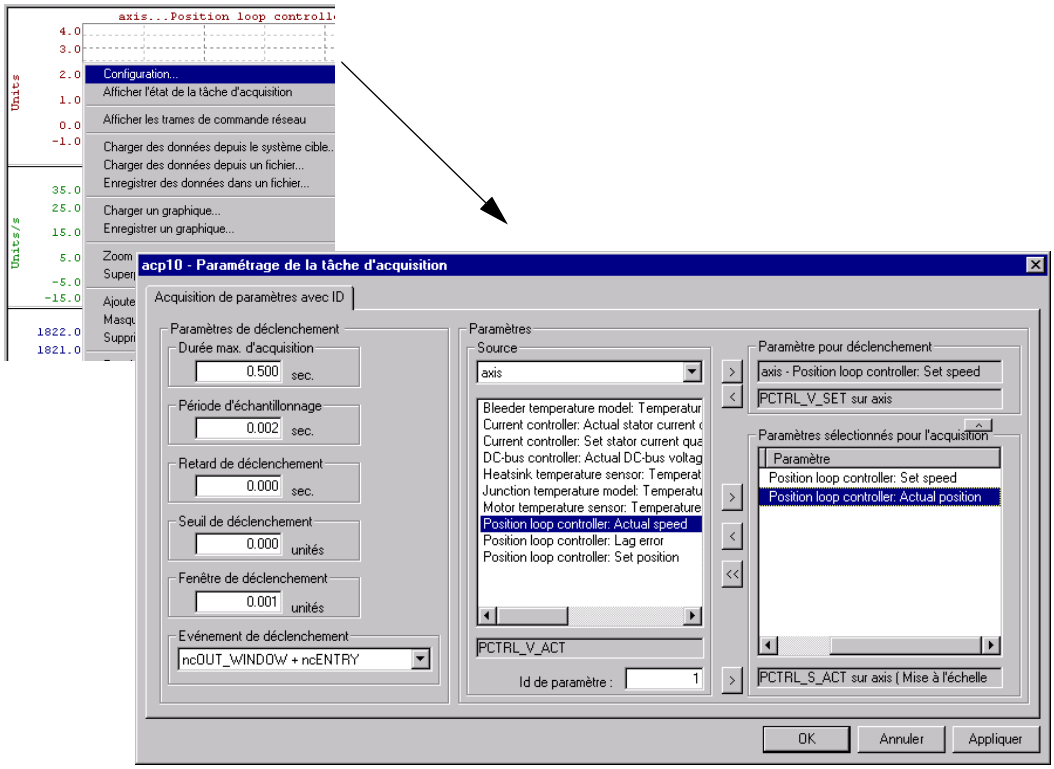


Fig. 6.24 : **Options de l'oscilloscope intégré**

Remarque

Pour plus de détails sur les possibilités de réglage, voir **Chapitre 8 : "Oscilloscope intégré"**.

- **Lancement de l'acquisition**
  L'acquisition des données peut être lancée de trois manières différentes :

Par une action NC	
En fonction du paramétrage (conditions de déclenchement)	
Directement	

- **Analyse des courbes**
  L'oscilloscope intégré vous offre des outils d'édition et d'aide appropriés pour l'interprétation et l'analyse des courbes.

  On accède à ces outils en faisant un clic droit de souris dans la fenêtre de l'oscilloscope et en sélectionnant la commande **Propriétés** du menu contextuel.

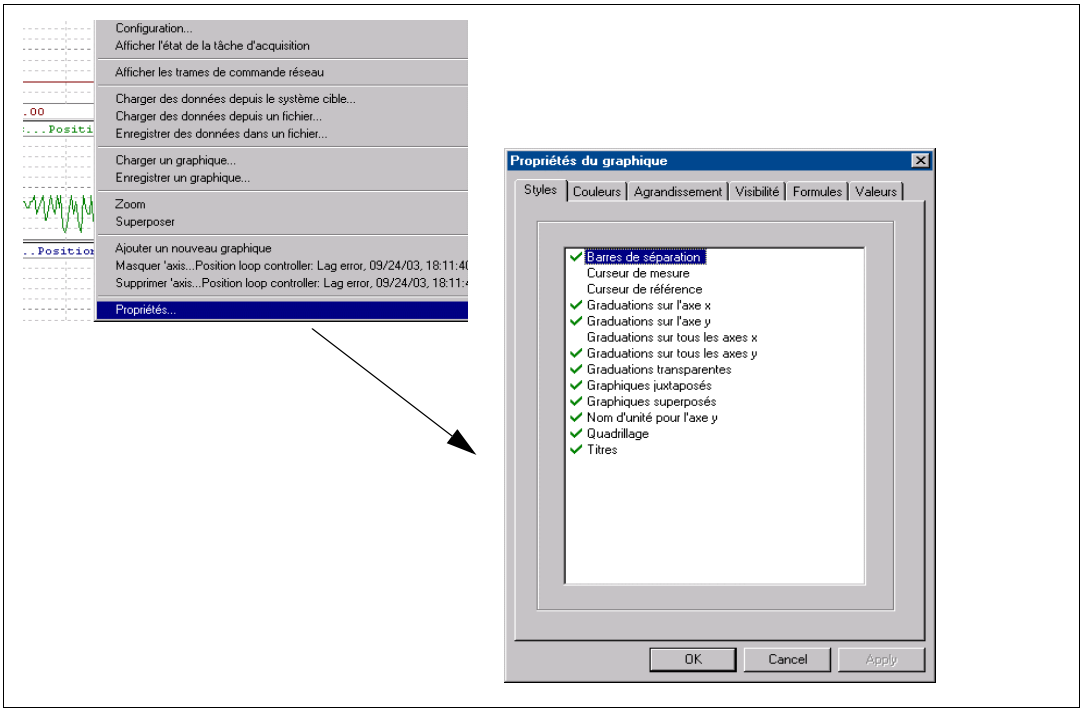


Fig. 6.25 : **Propriétés du graphique**

Dans l'intercalaire "Styles", vous pouvez activer l'affichage de certains éléments comme le **curseur de référence ou de mesure**.

Dans l'intercalaire "Couleurs", vous pouvez définir la **couleur** de l'arrière-plan et des courbes.

Dans l'intercalaire "Agrandissement", vous pouvez paramétrer une **échelle de courbe** appropriée.

- **Archivage des données d'acquisition**
  Les courbes peuvent être enregistrées dans des **fichiers**, échangées et rechargées pour être analysées ultérieurement ou insérées dans une documentation.

  Pour l'enregistrement, vous pouvez utiliser un format B&R ou bien le format **ASCII** (utilisation dans EXCEL ou MathCAD par exemple)

  Ici aussi, le mode de sélection le plus simple consiste à sélectionner la commande appropriée dans le menu contextuel (après avoir cliqué avec le bouton droit de la souris).

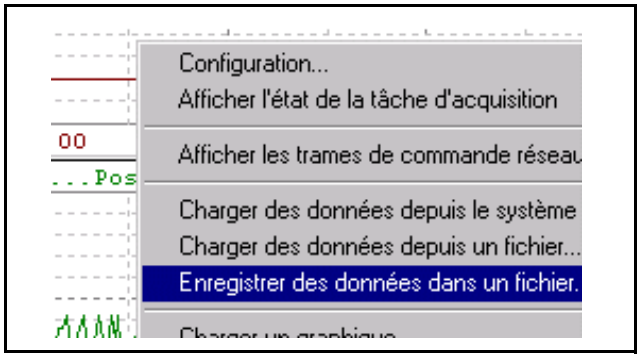


Fig. 6.26 : **Exportation des données graphiques au format ASCII**

- **Analyseur de trames**
  L'analyseur de trames permet d'analyser précisément les séquences de commandes échangées entre le gestionnaire ACOPOS résidant dans le contrôleur et l'ACOPOS lui-même.

  Le menu permettant d'accéder à cet outil s'obtient en cliquant avec le bouton droit de la souris dans la fenêtre de l'oscilloscope :

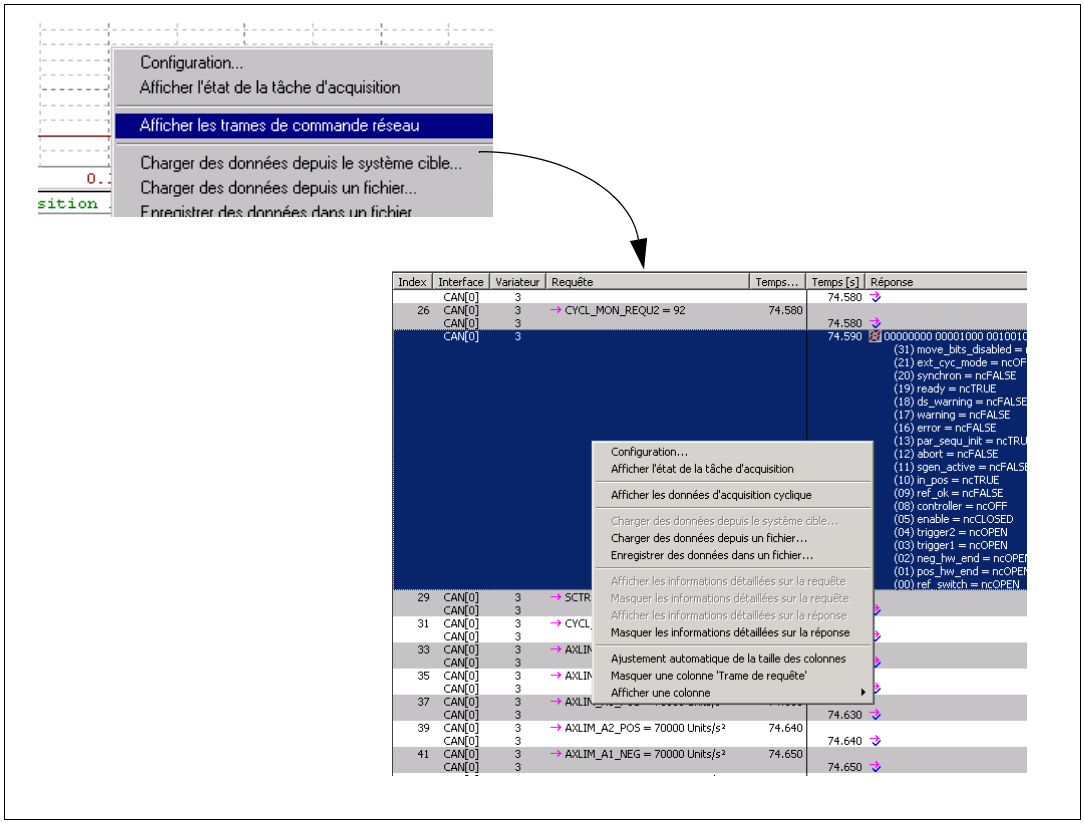


Fig. 6.27 : **Analyseur de trames**

ACP10-SW V0.47x

29 Juin 2001

1 CONCEPT NC

Pour rendre la conception des tâches de positionnement plus simple et plus claire, le logiciel NC de B&R est basé sur le concept « orienté objet ». Conséquence : des coûts de développement réduits et des possibilités de réutilisation accrues. De plus, le logiciel est plus facile à appréhender du fait de l'utilisation de "**constantes NC**" aux noms symboliques évocateurs ("`ncSWITCH_ON`", "`ncSWITCH_OFF`", …).

Les unités de commande principales appelées **objets NC** constituent la base de ce concept orienté objet :

Objet NC[1]	**Constante NC**
Axe réel	**`ncAXIS`**
Axe virtuel	**`ncV_AXIS`**
Système CNC	**`ncCNCSYS`**
...	

Ces unités de commande principales se subdivisent en sous-groupes appelés **sujets** :

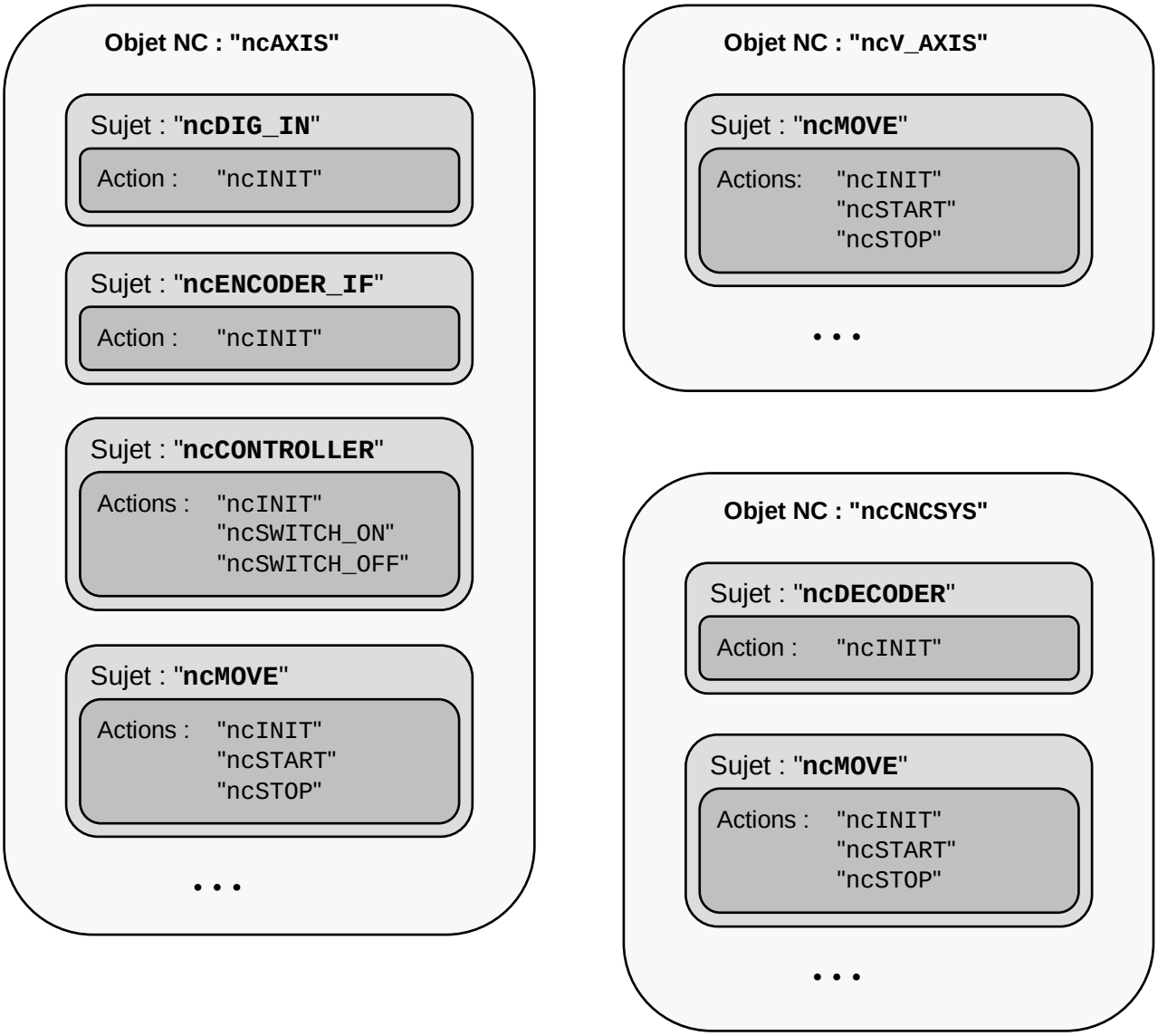
Sujet	**Constante NC**
Entrées digitales	**`ncDIG_IN`**
Interface codeur	**`ncENCODER_IF`**
Régulateur	**`NcCONTROLLER`**
Mouvement	**`ncMOVE`**
Décodeur CNC	**`ncDECODER`**
...	

Pour pouvoir agir sur ces sujets, des **actions** sont définies :

Action	**Constante NC**
Initialiser	**`ncINIT`**
Activer	**`NcSWITCH_ON`**
Désactiver	**`NcSWITCH_OFF`**
Démarrer	**`NcSTART`**
Arrêter	**`NcSTOP`**
...	

[1] A ce jour, ACP10-SW ne fournit que des axes réels en tant qu'objets NC (les axes virtuels ou le système CNC ne sont pas encore au programme)

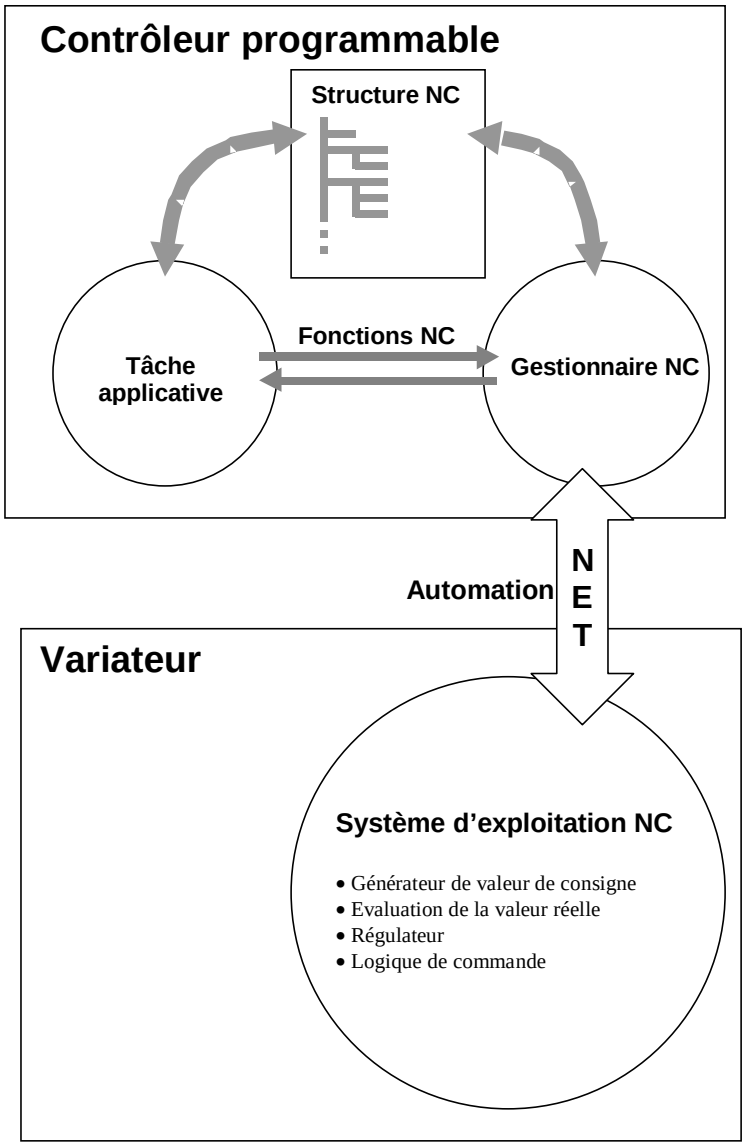
La structure hiérarchique propre au concept NC "**Objet NC → Sujet → Actions**" est illustrée ci-dessous :



La coordination du processus s'effectue dans une tâche d'application, dans le contrôleur B&R. Cette tâche d'application gère les objets NC en utilisant la structure NC et les fonctions NC fournies par le gestionnaire NC.

Le gestionnaire NC communique via AutomationNet avec le système d'exploitation NC résidant dans le variateur B&R. Ainsi, le gestionnaire NC transmet au variateur des commandes et des paramètres issus de la tâche d'application et, à l'inverse, des données de processus ainsi que des indicateurs d'état sont transmis à la tâche d'application depuis le variateur.

Le système d'exploitation NC résidant dans le variateur contient les composants nécessaires à la réalisation du positionnement.



Une mise à jour du système d'exploitation NC pour permettre l'utilisation des fonctions mécatroniques les plus récentes peut être effectuée directement avec AutomationStudio, via AutomationNet.

1.1 Structure NC

Un objet NC (par exemple un axe) englobe toute une série de données de paramétrage et d'informations relatives au processus.

Pour plus de clarté, toutes ces données et informations sont regroupées par thème dans une structure de données utilisateur appelée **structure NC** et correspondant au type de l'objet NC.

Le contenu de cette structure dépend du type d'objet NC. Un axe réel requiert plus d'informations qu'un axe virtuel (données sur l'interface codeur par exemple).

Par conséquent, il existe une structure NC spécifique pour chaque type d'objet :

Type d'objet NC	Type de données – Structure NC
Axe réel	**ACP10ACHSE_typ**
Axe virtuel[1]	**ACP10VACHS_typ**

Les éléments de la structure NC contiennent les paramètres relatifs à chacun des thèmes :

Exemple avec le type de données **ACP10ACHSE_typ** :

dig_in .....Entrées digitales
...
encoder_if .....Interface codeur
...
limit .....Valeurs limites de l'axe
...
controller .....Paramétrage du régulateur
...
move .....Paramétrage du mouvement
...

Les données de paramétrage peuvent être entrées dans un module de paramètres d'initialisation (module de données) dans AutomationStudio puis transférées au contrôleur avec ce module de données. Ensuite, le gestionnaire NC transfère automatiquement à la structure NC les paramètres contenus dans ce module de données et envoie ces paramètres au variateur.

Par ailleurs, les données de paramétrage peuvent être écrites dans la structure NC via le programme utilisateur. Exemple pour les paramètres du régulateur de vitesse :

```
p_ax_dat->controller.speed.kv = 0.5;  /* [As/Tour] */
p_ax_dat->controller.speed.tn = 0.02; /* [s]*/
```

En plus des données de paramétrage, la structure NC contient aussi toutes les données d'état et de processus essentielles :

Exemple avec le type de données **ACP10ACHSE_typ** :

...
monitor .....Moniteur d'axe
...
message .....Messages (erreurs, avertissements)
...

[1] A ce jour, ACP10-SW ne fournit pas encore d'axes virtuels en tant qu'objets NC

1.2 Tâche d'application

Le processus de positionnement se programme dans une tâche utilisateur. Les données de processus et de paramétrage sont entrées dans la structure NC puis transférées au variateur via une commande appropriée.

Les commandes à envoyer au variateur sont lancées avec la fonction NC "**`ncaction()`**" (d'où le nom d'**actions NC** pour ces commandes).

Toutefois, il faut d'abord allouer au gestionnaire NC l'objet NC approprié en appelant la fonction "**`ncalloc()`**".

1.2.1 Fonction NC ncalloc() : allocation d'un objet NC

`status = ncalloc(bus_typ,modul_adr,object_typ,channel,adr(nc_object))`		
Paramètres d'entrée		
`bus_typ`	`UINT`	Identificateur du bus du contrôleur : **`ncACP10MAN`**
`modul_adr`	`UINT`	Adresse de module : - Octet de poids fort : index de l'interface CAN dans la configuration NC - Octet de poids faible : numéro de nœud CAN
`object_typ`	`UINT`	Identificateur du type d'objet NC : **`ncAXIS`**
`channel`	`UINT`	Numéro de voie : **`1`**
Paramètres de sortie :		
`nc_object`	`UDINT`	Objet NC (pointeur sur la structure NC)
`status`	`UINT`	Etat de la fonction : **`ncOK`** ou code d'erreur

Un objet NC est alloué au gestionnaire NC avec la fonction ncalloc(). Ensuite, l'objet NC est traité de façon cyclique par le gestionnaire NC.

Dans le paramètre de sortie "`nc_object`", cette fonction retourne l'adresse de la structure NC de l'objet NC (pointeur sur structure NC). Avec cette adresse et le type de la structure NC, la tâche d'application peut ensuite accéder aux données d'objet NC et commander l'objet NC avec la fonction ncaction().

Pour des raisons de performance, la fonction ncalloc() ne devrait être appelée que dans le sous-programme d'initialisation des tâches du contrôleur programmable.

Codes d'erreur :

10600 Erreur pour laquelle des informations supplémentaires sont reportées dans les enregistrements d'erreur de l'objet NC. Le pointeur d'objet NC dans "nc_object" est valide. Pour lire tous les enregistrements d'erreur, "`ncaction(nc_object,ncMESSAGE,ncACKNOWLEDGE)`" est dans ce cas la seule action NC qui peut être appelée dans un programme applicatif, toutes les autres actions NC étant rejetées par le gestionnaire NC avec le message d'erreur correspondant.

10650 Liste de configuration NC introuvable

10703 Objet NC introuvable
(spécification erronée pour "`bus_typ`", "`modul_adr`", "`object_typ`", ou "`channel`")

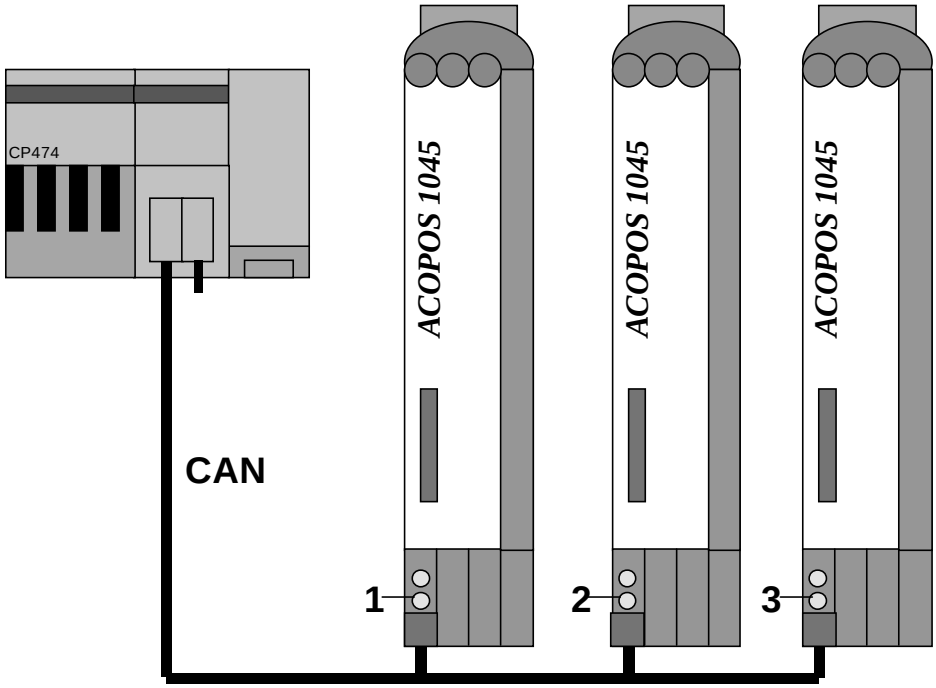
10704 Identificateur de classe de tâche non valide
(appel de ncalloc() dans une classe de tâche plus rapide que celle définie pour la tâche de gestionnaire NC)

10705 Module gestionnaire système introuvable

10706 Erreur lors de l'installation de la tâche NC pour le début de la classe de tâche (pile de tâche suffisante ?)

10707 Erreur lors de l'installation de la tâche NC pour la fin de la classe de tâche (pile de tâche suffisante ?)

Exemple :



Considérons une application avec trois variateurs "ACOPOS 1045" commandés par une CPU "CP474" via CAN. Sur les cartes CAN des variateurs, les numéros de nœud sont réglés à 1, 2 et 3. Dans une tâche d'application, la variable d'objet NC "`ax1_obj`" et le pointeur de données utilisateur associé "`p_ax1_dat`" sont déclarés pour la commande de l'axe réel du variateur portant le numéro de nœud 1 :

```
UDINT          ax1_obj;   /* Objet NC pour l'axe réel du noeud 1 */
ACP10AXIS_typ *p_ax1_dat;/* Pointeur sur données utilisateur de ax1_obj */
```

Dans le sous-programme d'initialisation, cet axe réel peut être alors être alloué au gestionnaire NC en indiquant le numéro de noeud **1** et le type d'objet NC "**`ncAXIS`**".

```
alloc_stat_ax1 = ncalloc(ncACP10MAN,1,ncAXIS,1,(UDINT)&ax1_obj);
```

Après allocation réussie (**!!!** "**`ncOK`**" pour l'état retourné par la fonction ncalloc **!!!**), l'objet NC est traité cycliquement par le gestionnaire NC. A présent, le pointeur d'objet NC peut être affecté à la variable correspondante :

```
if ( alloc_stat_ax1 == ncOK )
/* allocation d'objet NC réussie */
    p_ax1_dat = (ACP10AXIS_typ*)ax1_obj;
```

L'allocation des axes réels des variateurs portant les numéros de nœud 2 et 3 au gestionnaire NC peut être effectuée de la même façon en spécifiant les numéros de nœuds et le type d'objet NC.

1.2.2 Fonction NC ncaction() : appel d'une action NC

status = ncaction(nc_object,subject,action)		
Paramètres d'entrée :		
nc_object	UDINT	Objet NC (de ncalloc())
subject	UINT	Sujet sur lequel l'action est appliquée
action	UINT	Action à effectuer
Paramètres de sortie :		
status	UINT	Etat de la fonction : **ncOK** (commande reportée) **ncACTIVE** (commande ***non*** reportée) ou code d'erreur)

La fonction ncaction() reporte les paramètres de la commande à exécuter dans un élément d'instruction du gestionnaire NC. Ensuite, dans la tâche NC cyclique, ce dernier transmet la commande au variateur.

Codes d'erreurs :

10720 Pointeur de fonction non valide
10721 Paramètre "nc_object" non valide (pas d'objet NC)
10722 Paramètre "subject" non valide (non défini ou non autorisé pour l'objet NC spécifié)
10723 Paramètre "action" non valide (non défini ou non autorisé pour le sujet spécifié)
10724 Type d'objet NC non valide
10726 Cette action NC est bloquée à cause d'une erreur dans ncalloc()

Lorsque l'on utilise la fonction ncaction(), il faut impérativement respecter les points suivants :

- Une action NC ne peut être transmise au gestionnaire NC que si la valeur retournée est "**status == ncOK**". Dans tous les autres cas, l'applicatif ne doit pas passer à l'étape suivante.

- "**status == ncAKTIV**" signifie que l'élément d'instruction du gestionnaire NC est occupé. Dans ce cas, l'applicatif doit simplement rester en suspens à ce stade de l'exécution.

- Toutes les autres valeurs pour "**status**" indiquent une des erreurs citées ci-dessus.

- "**status == ncOK**" signifie seulement que la commande a bien été transmise au gestionnaire NC. Cela ne signifie pas que la commande a déjà été transmise au variateur ou que ce dernier a déjà exécuté la commande. Ceci est indiqué par des affichages d'état supplémentaires générés pour chaque action NC.
  Appel de fonction `status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_ON)`
  Indications(s) d'état: "controller.status = ncON"
  après activation du régulateur

  Un état de ce type peut aussi servir de condition à la réalisation d'une autre action NC :
  Appel de fonction `status = ncaction(ax_obj,ncHOMING,ncSTART)`
  Conditions : **"controller.status == ncON"**

Exemple :

Description des actions NC "Activer le régulateur" et "Démarrer la prise de référence" :

Appel de fonction	-	`status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_ON)`
Paramètres :	-	
Condition(s) :	-	...
Résultat(s) :	-	Activation du régulateur
Indications(s) d'état :	-	"**controller.status = ncON**" après activation du régulateur
Appel de fonction	-	`status = ncaction(ax_obj,ncHOMING,ncSTART)`
Paramètres :	-	...
Condition(s) :	-	"**controller.status == ncON**"
Résultat(s) :	-	Démarrage de la prise de référence avec les paramètres initialisés précédemment
Indications(s) d'état :	-	"move.mode = ncHOMING" "**move.homing.status.ok = ncFALSE**" après "**ncOK**" en tant qu'état résultant de l'action NC
	-	"**move.homing.status.ok = ncTRUE**" après que la prise de référence a bien été effectuée

Pour l'axe réel du variateur avec le numéro de nœud 1 et alloué avec succès via ncalloc() dans l'exemple précédent, les différentes étapes pour l'activation du régulateur et la prise de référence sont les suivantes :

```
switch ( step )
{
  ...
case SW_CTRL_ON: /* Activer le régulateur */
    status = ncaction(ax1_obj,ncCONTROLLER,ncSWITCH_ON);
    if ( status  == ncOK )
    {
      step = W_CTRL_ON;
    }
    break;

  case W_CTRL_ON: /* Attendre que le régulateur a été activé */
    if ( p_ax1_dat->controller.status == ncON )
    {
      step = HOMING_START;
    }
    break;

case HOMING_START: /* Démarrer la prise de référence */
    status = ncaction(ax1_obj,ncHOMING,ncSTART);
    if ( status == ncOK )
    {
      step = W_HOMING_OK;
    }
    break;

case W_HOMING_OK: /* Attendre que la prise de référence a bien été effectuée  */
    if ( p_ax1_dat->move.homing.status.ok == ncTRUE )
    {
        step = <NEXT_STEP>
    }
    break;
  ...
}
```

1.3 Gestionnaire NC

Le gestionnaire NC est une extension du système d'exploitation et constitue le lien entre l'application de positionnement et le système d'exploitation NC résidant dans le variateur. Aussi, le variateur ne peut être commandé que si le projet du contrôleur contient les objets système suivants :

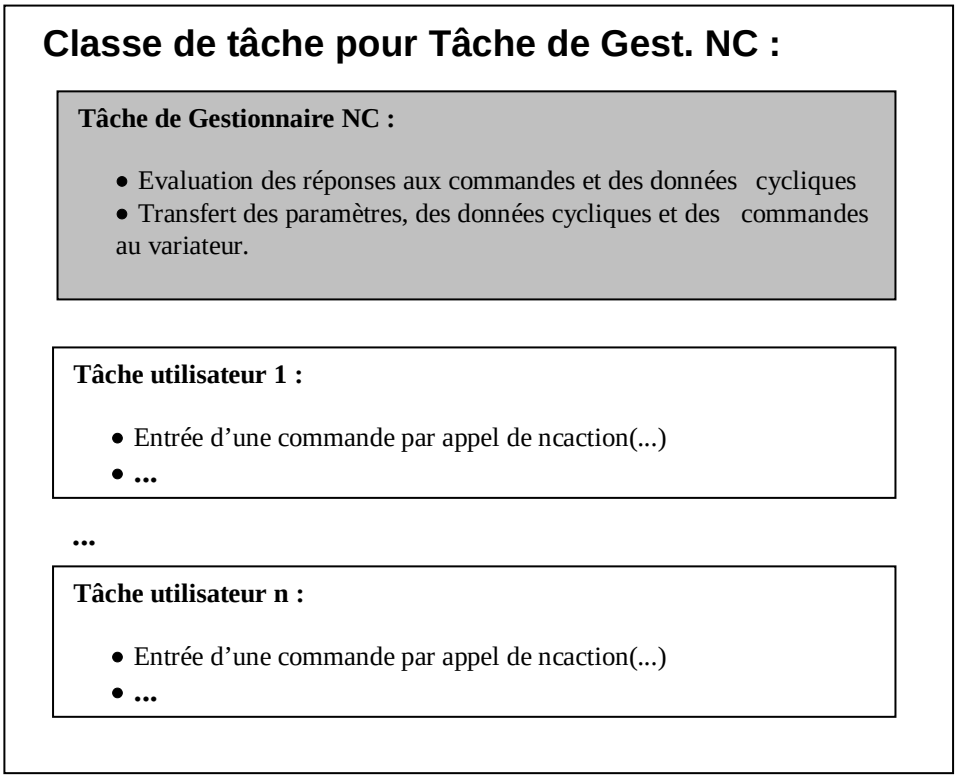
acp10man .....Gestionnaire NC du système ACP10
acp10sys[1] .....Système d'exploitation NC du système ACP10

La partie spécifique au matériel dans les fonctions NC générales ncalloc() et ncaction() est contenue dans le gestionnaire NC.

Pendant la phase de boot, le gestionnaire NC initialise la zone mémoire où sont stockées, entre autres, les structures NC pour tous les objets NC.

Lors du fonctionnement cyclique, le gestionnaire NC effectue les tâches suivantes :
- Transfert des données de paramétrage et des commandes au variateur
- Lecture des données d'état et report de ces données dans la structure NC
- Transfert du système d'exploitation NC au variateur

Lors du fonctionnement cyclique dans le contrôleur programmable, le gestionnaire NC traite les objets NC dans la classe de tâches entrée en tant que "Classe de tâches pour Tâche du Gestionnaire NC"[2] dans la configuration NC. A cet effet, le gestionnaire NC installe une tâche propre appelée ensuite au début de cette classe de tâches :



[1] Dans le projet, le transfert du système d'exploitation NC au contrôleur programmable est normalement bloqué avec "**Désactiver**" ("Désactiver" ne doit ensuite être désactivé que si le système d'exploitation NC doit être transféré au variateur)
[2] L'entrée "Classe de tâche pour Tâche de Gestionnaire NC" n'existe qu'à partir de la version V0.470. Dans les versions antérieures d'ACP10-SW, la Tâche de Gestionnaire NC était toujours installée dans la classe de tâches "Cyclique#1".

2 UTILISATION DES OBJETS NC

Le variateur ACOPOS de B&R est destiné à une utilisation dans des applications très diverses. Le concept NC doit permettre la prise en charge d'applications monoaxe ou d'applications plus complexes. Les objets NC, unités de commande dotées de zones de données spécifiques (structures NC) et de fonctions (actions NC), ont été créés à cet effet.

Les objets NC réalisés à ce jour dans le système ACP10 sont les suivants :

Objet NC	Identificateur de type	Type de données
Axe NC réel	ncAXIS	ACP10AXIS_typ

Un objet NC est alloué au gestionnaire NC avec la fonction **ncalloc().** Pour accéder directement aux données de l'objet NC, il faut affecter l'adresse du début de la structure NC à un pointeur défini avec un type spécifique à l'objet NC. Ceci peut être programmé de la manière suivante :

```
UDINT          ax1_obj;   /* Objet NC pour un axe réel */
ACP10ACHSE_typ *p_ax_dat;/* Pointeur sur données utilisateur de ax_obj */

status = ncalloc(ncACP10MAN,1,ncAXIS,1,(UDINT)&ax_obj);
if ( action_status == ncOK )
{ /* Allocation de l'objet NC réussie */
    p_ax_dat = (ACP10AXIS_typ*)ax_obj;
}
```

Ensuite, comme le montre l'exemple qui suit, l'objet NC peut être exploité grâce à la fonction **ncaction().** Les paramètres de fonction sont définis dans la structure de données NC spécifique à cet objet :

```
case REF_INIT: /* Initialiser les paramètres de prise de référence */
  p_ax_dat->move.homing.parameter.v_switch = ...;
  ...
  action_status = ncaction(ax_obj,ncHOMING,ncINIT);
  if ( action_status == ncOK )
  {
    step = W_HOMING_INIT;
  }
  break;
```

Dans la suite, on décrit en détail la façon dont sont utilisées les structures de données conjointement avec les fonctions, et ce pour tous les objets NC. Si aucune condition n'a été explicitement spécifiée pour la réalisation d'une action NC, seuls des paramètres de commande non valides ou un état d'erreur provoquera le rejet de cette action dans la partie de programme en cours d'exécution. Une commande sans effet peut toujours être exécutée (par exemple, "Désactivation du régulateur" lorsque le régulateur n'est pas activé).

Toutes les structures de données sont représentées sans les octets de remplissage.

Dans les structures de données, les composantes servant uniquement à l'affichage ou à la lecture de données en sortie sont représentées sur fond gris. Les zones mémoire correspondantes ne peuvent en aucun cas être réécrites dans le programme utilisateur !

2.1 Objet NC "ncAXIS"

Un objet NC de type "ncAXIS" est disponible pour chaque variateur B&R capable de fonctionner avec ACP10SW. Cet objet NC peut être initialisé dans un sous-programme d'initialisation de la manière suivante (voir aussi la tâche exemple) :

```
#include "acp10man.h" /* Fonctions, types de données et constantes pour ACP10-SW */

...

/* Sous-programme d'initialisation */
void _INIT ax_init(void)
{
    ACP10_NODE = ...; /* Numéro de nœud CAN du variateur */
    channel    = 1;   /* Il existe un objet NC de ce type */

    status = ncalloc(ncACP10MAN,ACP10_NODE,ncAXIS,1,(UDINT)&ax_obj);

    if ( ( alloc_status != ncOK ) && ( alloc_status != 10600 ) )
    {
    /* Le pointeur d'objet NC est non valide */
        return;
    }

    p_ax_dat = (ACP10AXIS_typ*)ax_obj;

    if ( p_ax_dat->size != sizeof(ACP10ACHSE_typ) )
    {
    /* Le type de données NC n'est pas compatible avec le gestionnaire NC */
        dattyp_error = sizeof(ACP10AXIS_typ);
        return;
    }
    else
    {
        dattyp_error = 0;
    }

    if ( (p_ax_dat->sw_version.nc_manager&0xFFF0)!=(ACP10MAN_H_VERSION&0xFFF0) )
    {
    /* La version de "acp10man.h" n'est pas compatible avec le gestionnaire NC */
        version_error = ACP10MAN_H_VERSION;
        return;
    }
    else
    {
        version_error = 0;
    }

/* Autres initialisations */
    ...

}
```

Si la valeur attribuée à une des variables "dattyp_error" ou "version_error" est différente de zéro, la version de "acp10man.h" dans le projet est erronée.

2.1.1 Vue d'ensemble

2.1.1.1 Type de données "ACP10AXIS_typ"

Size	USINT	Taille du type de données de gestionnaire NC correspondant [octets]
sw_version		**ID de version logicielle [hexadécimal]**
Nc_manager	UINT	Gestionnaire NC
Nc_system	UINT	Système d'exploitation NC
Simulation		**Mode simulation**
Status	USINT	Etat : ncON/ncOFF
Global		**Initialisation globale**
init	USINT	Initialisation globale terminée : ncFALSE/ncTRUE
Network		**Réseau**
init	USINT	Réseau initialisé : ncFALSE/ncTRUE
Service		**Interface service**
data_adr	UDINT	Adresse de données
data_text	USINT[34]	Données en format texte
Request		**Requête (au variateur)**
par_id	UINT	ID de paramètre
Response		**Réponse (du variateur)**
par_id	UINT	ID de paramètre
dig_in		**Entrées digitales**
init	USINT	Initialisation des entrées digitales : ncFALSE/ncTRUE
status		**Etat**
reference	USINT	Interrupteur position référence :: ncOPEN/ncCLOSED
pos_hw_end	USINT	Interrupteur FC positif (matériel) :: ncOPEN/ncCLOSED
neg_hw_end	USINT	Interrupteur FC négatif (matériel) :: ncOPEN/ncCLOSED
trigger1	USINT	Trigger1 : ncOPEN/ncCLOSED
trigger2	USINT	Trigger2 : ncOPEN/ncCLOSED
level		**Niveau d'entrée actif**
reference	UINT	Interrupteur position référence: ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP)
pos_hw_end	UINT	Interrupteur FC positif matériel: ncACTIV_LO/ncACTIV_HI (+ncFORCE) (+ncQUICKSTOP)
neg_hw_end	UINT	Interrupteur FC négatif matériel: ncACTIV_LO/ncACTIV_HI (+ncFORCE) (+ncQUICKSTOP)
trigger1	UINT	Trigger 1 : ncACTIV_LO/ncACTIV_HI (+ncFORCE) (+ncQUICKSTOP)
trigger2	UINT	Trigger 2 : ncACTIV_LO/ncACTIV_HI (+ncFORCE) (+ncQUICKSTOP) Avec "+ncFORCE", l'état appliqué via la fonction de forçage est évalué à la place de l'entrée matérielle digitale. Avec "+ncQUICKSTOP", le niveau défini de l'entrée digitale matérielle déclenche la fonction Quickstop.
force		**Application d'état aux entrées digitales par la fonction de forçage**
reference	USINT	Interrupteur position référence: ncOPEN/ncCLOSED
pos_hw_end	USINT	Interrupteur FC positif matériel: ncOPEN/ncCLOSED
neg_hw_end	USINT	Interrupteur FC négatif matériel: ncOPEN/ncCLOSED
trigger1	USINT	Trigger 1 : ncOPEN/ncCLOSED
trigger2	USINT	Trigger 2 : ncOPEN/ncCLOSED
encoder_if		**Interface codeur**
init	USINT	Interface codeur initialisée : ncFALSE/ncTRUE
Parameter		**Paramètre**
Count_dir	USINT	Sens de comptage : ncSTANDARD/ncINVERSE
Scaling		**Echelle**
Load		**Charge [unités/tour_moteur]**
units	UDINT	Unités sur la charge
Rev_motor	UDINT	Tours moteur

Limit		**Valeurs limites**
init	USINT	Valeurs limites initialisées : ncFALSE/ncTRUE
Parameter		**Paramètre**
v_pos	REAL	Vitesse dans la driection positive [unité/s]
v_neg	REAL	Vitesse dans la direction négative [unité/s]
a1_pos	REAL	Accélération dans la direction positifve[unité/s^2]
a2_pos	REAL	Décélération dans la direction positive [unité/s^2]
a1_neg	REAL	Accélération dans la direction négative [unité/s^2]
a2_neg	REAL	Décélération dans la direction négative [unité/s^2]
t_jolt	REAL	Temps de vibration [s]
t_in_pos	REAL	Temps d'attente avant le message "In Position" [s]
pos_sw_end	DINT	Fin de course positif logiciel [unité]
neg_sw_end	DINT	Fin de course négatif logiciel [unité]
ds_warning	REAL	Si l'erreur de traînage dépasse "ds_warning", un avertissement s'affiche
ds_stop	REAL	Si l'erreur de traînage dépasse "ds_stop", le mouvement est stoppé
Controller		**Régulateur**
init	USINT	Régulateur initialisé : ncFALSE/ncTRUE
ready	USINT	Régulateur prêt à être activé : ncFALSE/ncTRUE
status	USINT	Etat : ncOFF/ncON
Position		**Régulateur de position**
kv	REAL	Amplification proportionnelle [1/s]
tn	REAL	Temps d'action intégrale [s]
t_predict	REAL	Temps de prédiction [s]
t_total	REAL	Temps de retard total [s]
p_max	REAL	Action proportionnelle max. [unité/s]
i_max	REAL	Action intégrale max. [unité/s]
Speed		**Régulateur de vitesse**
kv	REAL	Amplification proportionnelle [A s / tour]
tn	REAL	Temps d'action intégrale [s]
Move		**Mouvement**
modus	UINT	Mode : ncOFF — Pas de mouvement actif ncSTOP — Arrêt d'un mouvement ncHOMING — Prise de référence active ncBASIS_MOVE — Mouvement de base actif ncACTIV — Mouvement actif n'ayant pas été lancé par une commande de mouvement du gestionnaire NC (par l'interface service par exemple)
detail	UINT	Détail : Général : ncOFF — Pas de mouvement actif ou aucune info détaillée requise Pour le mode "ncSTOP": ncSTOP — Arrêt après une action NC "ncMOVE, ncSTOP" ncEVENT — Arrêt après un événement dans le variateur Pour le mode "ncBASIS_MOVE": ncHALT — Suspension active d'un mouvement de base ncPOS_MOVE — Mouvement dans la direction positive (+ ncTRG_STOP) — (avec mode "Arrêt après déclenchement") ncNEG_MOVE — Mouvement dans la direction négative (+ ncTRG_STOP) — (avec mode "Arrêt après déclenchement") ncABS_MOVE — Mouvement avec position cible absolue (+ ncTRG_STOP) — (avec mode "Arrêt après déclenchement") (+ ncS_REST) — (avec variation "+ s_rest") ncREL_MOVE — Mouvement avec distance de déplacement relative (+ ncTRG_STOP) — (avec mode "Arrêt après déclenchment") (+ ncS_REST) — (avec variation "+ s_rest")

Stop		**Arrêt d'un mouvement**
init	USINT	Arrêt initialisé : ncFALSE/ncTRUE
Index		**Index de paramètres**
command	USINT	pour la commande d'arrêt suivante
parameter[4]		**Paramètres pour la configuration d'arrêt**
decel_ramp	USINT	Rampe de décélération (réglage par défaut : ncI_LIMIT): ncA_MOVE : Avec le temps de décélération défini pour le mouvement en cours (+ ncT_JOLT): (prise en compte du temps de vibration spécifié) ncA_LIMIT : Avec le temps de décélération défini dans les valeurs limites des axes (+ ncT_JOLT): (prise en compte du temps de vibration spécifié)
controller	USINT	Etat du régulateur après l'arrêt d'un mouvement : ncOFF/ncON (Réglage par défaut : ncON)
Homing		**Prise de référence**
init	USINT	Prise de référence initialisée : ncFALSE/ncTRUE
status		**Etat**
ok	USINT	Position de référence valide : ncFALSE/ncTRUE
tr_s_rel	FLOAT	Distance entre l'activation de "Déclenchement sur impulsion de référence" et l'apparition d'une impulsion de référence [tours de codeur]
Offset	DINT	Offset une fois que la prise de référence a été effectuée [unité]
parameter		**Paramètre**
s	DINT	Position de référence ou offset de prise de référence [unité]
v_switch	REAL	Vitesse pour la recherche de l'interrupteur de position de référence [unité/s]
v_trigger	REAL	Vitesse du trigger (après avoir atteint l'interrupteur de position d'origine) [unité/s]
a	REAL	Vitesse [unité/s^2]
Mode	USINT	Mode : ncDIRECT : direct ncSWITCH_GATE : avec porte générée par l'interrupteur de position de réf. ncABS_SWITCH : avec interrupteur de position de référence absolue ncEND_SWITCH : avec interrupteur de fin de course matériel ncHOME_OFFSET: application d'un offset de prise de réf. + ncCORRECTION : correction de la zone de comptage
Edge_sw	USINT	Front de l'interrupteur de position de référence :ncPOSITIVE/ncNEGATIVE
Start_dir	USINT	Sens de démarrage : ncPOSITIV/ncNEGATIV
Trig_dir	USINT	Sens de déclenchement : ncPOSITIV/ncNEGATIV
Ref_pulse	USINT	Impulsion de référence : ncOFF/ncON
tr_s_block	FLOAT	Distance pour le blocage de l'activation de "Déclenchement sur impulsion de référence" [tours codeur]
basis		**Mouvements de base**
init	USINT	Mouvements de base initialisés : ncFALSE/ncTRUE
status		**Etat**
in_pos	USINT	"In Position" (position cible atteinte) : ncFALSE/ncTRUE
override		**Consigne de vitesse/accélération en %**
v	INT	Consigne de vitesse [0.01%] (réglage par défaut : 10000) Valeurs valides : de 0 à 20000 (0% à 200%)
a	INT	Consigne d'accélération [0.01%] (réglage par défaut : 10000) Valeurs valides : de 1 à 20000 (0.01% à 200%)
parameter		**Paramètres**
s	DINT	Position cible ou distance de déplacement relative [unité]
v_pos	REAL	Vitesse dans la direction positive [unité/s]
v_neg	REAL	Vitesse dans la direction négative [unité/s]
a1_pos	REAL	Accélération dans la direction positive [unité/s^2]
a2_pos	REAL	Retard dans la direction positive [unité/s^2]
a1_neg	REAL	Accélération dans la direction négative [unité/s^2]
a2_neg	REAL	Retard dans la direction négative [unité/s^2]
trg_stop		**Mode "Arrêt après déclenchement"**
init	USINT	Mode initialisé : ncFALSE/ncTRUE
événement	USINT	Événement de déclenchement : NcOFF : Le mode "Arrêt après déclenchement" est désactivé ncTRIGGER1+ncP_EDGE : Front positif à l'entrée digitale "Trigger1" ncTRIGGER2+ncP_EDGE : Front positif à l'entrée digitale "Trigger2" ncTRIGGER1+ncN_EDGE : Front négatif à l'entrée digitale "Trigger1" ncTRIGGER2+ncN_EDGE : Front négatif à l'entrée digitale "Trigger2"
s_rest	DINT	Distance restante après déclenchement [unité]

monitor		**Moniteur**
s	DINT	Position de consigne [unité] (est égale à la position réelle si le régulateur est désactivé)
v	REAL	Vitesse de consigne [unité/s]
status		**Bits d'état**
error	USINT	Erreur apparue : ncFALSE/ncTRUE
warning	USINT	Avertissement apparu : ncFALSE/ncTRUE
ds_warning	USINT	Erreur de traînage supérieure à "limit.ds_warning": ncFALSE/ncTRUE
message		**Messages (erreurs, avertissements)**
count		**Nombre**
Error	USINT	Nombre d'erreurs non acquittées
warning	USINT	Nombre d'avertissements non acquittés
satz		**Enregistrement de message actuel**
number	UINT	Numéro
info	UDINT	Info supplémentaire (codée)
par_id	UINT	ID de paramètre (utilisé seulement pour des erreurs à la lecture ou à l'écriture de paramètres)
text		**Recherche de texte pour l'enregistrement de message actuel**
status		**Etat de la recherche du texte**
zeilen	UINT	Lignes du texte trouvé
fehler	UINT	Erreur : 0: Pas d'erreur 1: Adresse du tampon de données zéro 2: Taille du tampon de données inférieure au minimum (40) 3: Erreur avec texte (en anglais) dans le tampon de données
parameter		**Paramètre**
format	UINT	Format : ncEMPTY/ncZERO/ncBREAK
columns	UINT	Nombre de colonnes par ligne
data_modul	USINT[10]	Nom du module de données de texte d'erreur
data_len	UINT	Taille du tampon de données dans le programme utilisateur
daten_adr	UDINT	Adresse du tampon de données dans le programme utilisateur

2.1.1.2 Actions NC pour l'objet NC "ncAXIS"

2.1.1.2.1 Actions NC standard

Sujet	Action	Description
ncSERVICE	ncREAD	Lire un paramètre de variateur via l'interface service
ncSERVICE	ncSET	Appliquer un paramètre de variateur via l'interface service
ncSERVICE	ncREAD + ncDATA_TEXT	Lire un paramètre de variateur via l'interface service ***avec*** la valeur dans "data_text"
ncSERVICE	ncSET + ncDATA_TEXT	Appliquer un paramètre de variateur via l'interface service ***avec*** la valeur dans "data_text"
ncSIMULATION	NcSWITCH_ON	Activer le mode simulation
ncSIMULATION	NcSWITCH_OFF	Désactiver le mode simulation
ncGLOBAL	ncINIT	Initialisation globale
ncDIG_IN	ncINIT	Initialisation des entrées digitales
ncDIG_IN	ncFORCE	Appliquer un état aux entrées digitales par forçage
ncENCODER_IF	ncINIT	Initialisation de l'interface codeur
ncLIMITS	ncINIT	Initialisation des valeurs limites
ncCONTROLLER	ncINIT	Initialisation du régulateur
ncCONTROLLER	NcSWITCH_ON	Activer le régulateur
ncCONTROLLER	NcSWITCH_OFF	Désactiver du régulateur
ncSTOP	ncINIT	Initialiser la configuration d'arrêt
ncHOMING	ncINIT	Initialiser la prise de référence
ncHOMING	ncSTART	Démarrer de la prise de référence
ncBASIS_MOVE	ncINIT	Initialiser un mouvement de base
ncPOS_MOVE	ncSTART	Démarrer un mouvement dans la direction positive
ncNEG_MOVE	ncSTART	Démarrer un mouvement dans la direction négative
ncABS_MOVE	ncSTART	Démarrer un mouvement avec position cible absolue
ncREL_MOVE	ncSTART	Démarrer un mouvement avec distance de déplacement relatif
ncBASIS_TRG_STOP	ncINIT	Initialiser le mode "Arrêt après Déclenchement" pour mouvements de base
ncPOS_MOVE + ncTRG_STOP	ncSTART	Démarrer un mouvement dans la direction positive ***avec*** mode "Arrêt après Déclenchement"
ncNEG_MOVE + ncTRG_STOP	ncSTART	Démarrer un mouvement dans la direction négative ***avec*** mode "Arrêt après Déclenchement"
ncABS_MOVE + ncTRG_STOP	ncSTART	Démarrer un mouvement avec position cible absolue ***avec*** mode "Arrêt après Déclenchement"
ncABS_MOVE + ncTRG_STOP + ncS_REST	ncSTART	Démarrer un mouvement avec position cible absolue ***avec*** mode "Arrêt après Déclenchement" ***et*** variation "+ s_rest"
ncREL_MOVE + ncTRG_STOP	ncSTART	Démarrer un mouvement avec distance de déplacement relatif ***avec*** mode "Arrêt après Déclenchement"
ncREL_MOVE + ncTRG_STOP + ncS_REST	ncSTART	Démarrer un mouvement avec distance de déplacement relatif ***avec*** mode "Arrêt après Déclenchement" ***avec*** variation "+ s_rest"

2.1.1.2.2 Actions NC à priorité haute

Lors de l'appel d'une action NC à priorité haute, le télégramme de commande CAN correspondant est immédiatement transféré au variateur, même si le traitement d'une action NC standard appelée précédemment n'est pas encore terminé.

sujet	action	Description
ncMOVE	ncSTOP	Arrêter un mouvement
ncBASIS_BEW	ncHALT	Arrêter un mouvement de base arrêt avec les paramètres en cours

2.1.1.2.2 Actions NC internes

Les actions NC internes sont exécutées uniquement par le gestionnaire NC, sans transfert de données entre ce dernier et le variateur. Pour cette raison, ces actions NC sont toujours exécutées, même si le traitement d'une action NC appelée précédemment n'est pas encore terminé.

sujet	action	Description
ncMESSAGE	ncTEXT	Déterminer le texte pour l'enregistrement de message en cours
ncMESSAGE	ncACKNOWLEDGE	Acquitter l'enregistrement de message en cours

2.1.2 Informations générales

2.1.2.1 Structure de données

size	USINT	Taille de la structure de données du gestionnaire NC correspondant [octets]
sw_version		**ID de version logicielle [hexadécimal]**
nc_manager	UINT	Gestionnaire NC
nc_system	UINT	Système d'exploitation NC

Après allocation d'un objet NC avec la fonction ncalloc(), ces informations permettent de vérifier si les données NC incluses dans la tâche d'application (dans "C" via "`#include acp10man.h`") sont compatibles avec le gestionnaire NC résidant dans le contrôleur programmable.

L'élément "`size`" peut être utilisé pour contrôler la taille de la structure de l'objet NC :

```
if ( p_ax_dat->size != sizeof(ACP10AXIS_typ) )
{
/* La structure de données NC n'est pas compatible avec le gestionnaire NC */
    dattyp_error = sizeof(ACP10AXIS_typ);
    ...
}
```

De plus, en "C", l'élément "`sw_version.nc_manager`" peut être utilisé pour vérifier la version de "`acp10man.h`" :

```
if ( (p_ax_dat->sw_version.nc_manager&0xFFF0)!=(ACP10MAN_H_VERSION&0xFFF0) )
{
/* La version de "acp10man.h" n'est pas compatible avec le gestionnaire NC */
    version_error = ACP10MAN_H_VERSION;
    ...
}
```

L'élément "`sw_version`" est utilisé pour afficher l'ID de version du gestionnaire NC "acp10man" résidant dans le contrôleur ainsi que l'ID de version du système d'exploitation NC "acp10sys" résidant dans le variateur. La valeur hexadécimale "**16#$X_1X_2X_3X_4$**" correspond à l'ID de version logicielle "**$X_1.X_2X_3X_4$**" (dans un projet d'AutomationStudio-Projekt, le chiffre "**X_4**" de l'ID de version n'est affiché que si la valeur est différente de zéro).

Exemples :

ID de version logicielle	Affichage dans "`sw_version...`"
1.20	**16#1200**
1.201	**16#1201**

Important : Deux éléments de logiciel NC sont compatibles entre eux si les chiffres X_1, X_2 et X_3 de leur ID de version sont égaux !

2.1.3 Réseau

2.1.3.1 Structure de données

network		**Réseau**
init	USINT	Réseau initialisé : ncFALSCH/ncWAHR
service		**Interface service**
data_adr	UDINT	Adresse des données
data_text	USINT[34]	Données en format texte
request		**Requête (au variateur)**
par_id	UINT	ID de paramètre
response		**Réponse (du variateur)**
par_id	UINT	ID de paramètre

L'état "network.init = ncTRUE" est appliqué après initialisation réussie du système ACP10. Cet état n'a pas besoin d'être vérifié explicitement dans la tâche d'application car le gestionnaire NC affiche une erreur et empêche toute utilisation d'objet NC si "network.init==ncFALSE".

L'interface service ("network.service") permet à l'utilisateur d'accéder à tous les paramètres se trouvant dans le variateur (en lecture et en écriture). Dans la tâche d'application, il est ainsi possible d'appliquer et de lire des paramètres de variateur qui ne sont pas contenus dans la structure NC et qui, de ce fait, ne peuvent pas être appliqués par une action NC ou lus par le gestionnaire NC.

Dans un programme C, l'utilisateur peut faire appel à des constantes dites ID de paramètre ("**ACP10PAR**_<Define-Name>"). Les IDs de paramètres sont définis dans le fichier en-tête "acp10man.h" de la manière suivante :

```
...
#define ACP10PAR_PCTRL_LAG_ERROR        112 /* (REAL) Erreur de traînage */
...
```

Il existe deux façons de lire ou d'appliquer un paramètre :

1) La valeur du paramètre se trouve à l'adresse définie avec "network.service.**daten_adr**" :
   Dans ce cas, l'utilisateur doit recourir à l'action NC "ncSERVICE,**ncREAD/ncSET**". Dans la tâche d'application, une variable de type approprié et constituant la source ou la cible de l'opération doit être disponible. Avant l'appel de l'action NC, l'adresse de cette variable doit être entrée dans "network.service.data_adr".

2) La valeur du paramètre est entrée en format texte dans "network.service.**data_text**".
   Dans ce cas, l'utilisateur doit recourir à l'action NC "ncSERVICE,**ncREAD/ncSET+ncDATA_TEXT**". Le gestionnaire NC cherche l'ID de paramètre dans la liste de paramètres et génère soit un texte à partir de la valeur du paramètre (pour une lecture de paramètre), soit une valeur de paramètre à partir du texte (pour une écriture de paramètre), tout en prenant en compte le type de données correspondant à l'ID de paramètre.

   Exemple :

Paramètre – Type de données	Texte en entrée/sortie	Valeur de paramètre
REAL	"4.23"	4.23
UDINT	"4.23" (seulement possible en entrée)	4
UDINT	"17"	17
STR32	"Motor-Identifier-Text"	"Motor-Identifier-Text"

2.1.3.2 Actions NC

sujet	action	Description
ncSERVICE	ncREAD	Lire un paramètre de variateur via l'interface service
ncSERVICE	ncSET	Appliquer un paramètre de variateur via l'interface service
ncSERVICE	ncREAD + ncDATA_TEXT	Lire un paramètre de variateur via l'interface service ***avec*** la valeur dans "data_text"
ncSERVICE	ncSET + ncDATA_TEXT	Appliquer un paramètre de variateur via l'interface service ***avec*** la valeur dans "data_text"

Important : Les actions NC "ncSERVICE,ncREAD/ncSET+ncDATATEXT" sont avant tout conçues pour la lecture et l'écriture de paramètres avec les Motion Components. Si vous utilisez ces actions NC, notez que l'exécution des fonctions correspondantes peut prendre, à ce jour, jusqu'à 5 ms dans la classe de tâches configurée pour la Tâche de Gestionnaire NC.

2.1.3.2.1 Lecture de paramètres de variateur via l'interface service

Appel de fonction : `status = ncaction(ax_obj,ncSERVICE,ncREAD)`

Paramètre : "`network.service.request.par_id`"

Condition(s) : "`network.init == ncTRUE`"

Résultat(s) : Le paramètre spécifié est lu dans le variateur par l'intermédiaire de l'interface service. La taille et le format du paramètre sont déterminés par le gestionnaire NC à partir de l'ID de paramètre. Une fois cette opération terminée, la valeur du paramètre est transférée à la mémoire en commençant par l'adresse "`network.service.data_adr`" De ce fait, la zone de mémoire à partir de l'adresse "`network.service.data_adr`" doit être disponible pour les données résultantes.

Indications(s) d'état: "`network.service.response.par_id = 0`"
après "**ncOK**" en tant qu'état résultant de l'action NC

"`network.service.response.par_id = 0xFFFF`"
et le message d'erreur correspondant dans "`error`"
après refus de lire le paramètre dans le variateur

"`network.service.response.par_id = ...service.request.par_id`"
Données résultantes à partir de l'adresse "`network.service.data_adr`"
après lecture réussie du paramètre

Exemple :

```
switch (step)
{
...
  case PAR_READ:
    p_ax_dat->network.service.request.par_id = ACP10PAR_...;
    p_ax_dat->network.service.data_adr = (UDINT)&par_dat;
    action_status = ncaction(ax_obj,ncSERVICE,ncREAD);
    if ( action_status == ncOK )
    {
      step = W_PAR_READ;
    }
    break;

  case W_PAR_READ:
    if (p_ax_dat->network.service.response.par_id != 0)
    {
    /* Opération terminée */
      if ( p_ax_dat->network.service.response.par_id ==
           p_ax_dat->network.service.request.par_id )
      {
      /* Opération terminée avec succès */
      /* Résultat dans "par_dat" */
        step = <NEXT_STEP>
      }
      else
      {
      /* Réponse erreur : message d'erreur dans "erreur" */
        step = <ERROR_STEP>
      }
    }
    break;
...
}
```

2.1.3.2.2 Ecriture de paramètre de variateur via l'interface service

Appel de fonction : `status = ncaction(ax_obj,ncSERVICE,ncSET)`

Paramètre : "`network.service.data_adr`"

Condition(s) : "`network.init == ncTRUE`"
"`network.service.response.par_id != 0`"

Résultat(s) : Les données à partir de l'adresse "`network.service.data_adr`" sont transférées au variateur via l'interface service pour l'écriture du paramètre spécifié. La taille et le format du paramètre sont déterminés par le gestionnaire NC à partir de l'ID de paramètre.

Indications(s) d'état: "`network.service.response.par_id = 0`"
après "**`ncOK`**"en tant qu'état résultant de l'action NC

"`network.service.response.par_id = 0xFFFF`"
et le message d'erreur correspondant dans "`error`"
après refus d'écrire le paramètre dans le variateur

"`network.service.response.par_id = ...service.request.par_id`"
après écriture réussie du paramètre

Exemple :

```
switch (step)
{
...
  case PAR_SET:
    p_ax_dat->network.service.request.par_id = ACP10PAR_...;
    p_ax_dat->network.service.data_adr = (UDINT)&par_dat;
    action_status = ncaction(ax_obj,ncSERVICE,ncSET);
    if ( action_status == ncOK )
    {
      step = W_PAR_SET;
    }
    break;

  case W_PAR_SET:
    if (p_ax_dat->network.service.response.par_id != 0)
    {
    /* Opération terminée */
      if ( p_ax_dat->network.service.response.par_id ==
           p_ax_dat->network.service.request.par_id )
      {
      /* Opération terminée avec succès */
        step = <NEXT_STEP>
      }
      else
      {
      /* Réponse erreur : message d'erreur dans "erreur" */
        step = <ERROR_STEP>
      }
    }
    break;
...
}
```

2.1.3.2.3 Lecture de paramètre de variateur via l'interface service (valeur dans "data_text")

Appel de fonction : `status = ncaction(ax_obj,ncSERVICE,ncREAD+ncDATA_TEXT)`

Paramètre : "network.service.request.par_id"

Condition(s) : "network.init == ncTRUE"

Résultat(s) : Le paramètre spécifié est lu dans le variateur par l'intermédiaire de l'interface service. Le gestionnaire NC détermine la taille et le format du paramètre grâce à l'ID de paramètre. Une fois l'opération terminée, la valeur du paramètre est affiché en format texte dans "network.service.data_text".

Indications(s) d'état: "network.service.response.par_id = 0"
après "**ncOK**" en tant qu'état résultant de l'action NC

"network.service.response.par_id = 0xFFFF"
et le message d'erreur correspondant dans "error"
après refus de lire le paramètre dans le variateur

"network.service.response.par_id = ...service.request.par_id"
Valeur du paramètre (en format texte) dans "network.service.data_text"
après lecture réussie du paramètre

Exemple :

```
switch (step)
{
...
  case PAR_READ:
    p_ax_dat->network.service.request.par_id = ACP10PAR_...;
    action_status = ncaction(ax_obj,ncSERVICE,ncREAD+ncDATA_TEXT);
    if ( action_status == ncOK )
    {
      step = W_PAR_READ;
    }
    break;

  case W_PAR_READ:
    if (p_ax_dat->network.service.response.par_id != 0)
    {
    /* Opération terminée */
      if ( p_ax_dat->network.service.response.par_id ==
           p_ax_dat->network.service.request.par_id )
      {
      /* Opération terminée avec succès */
      /* Le résultat est dans "network.service.data_text" */
        step = <NEXT_STEP>
      }
      else
      {
      /* Réponse erreur : message d'erreur dans "error" */
        step = <ERROR_STEP>
      }
    }
    break;
...
}
```

2.1.3.2.4 Ecriture de paramètre de variateur via l'interface service (valeur dans "data_text")

Appel de fonction : `status = ncaction(ax_obj,ncSERVICE,ncREAD+ncDATA_TEXT)`

Paramètre : "network.service.data_txt"
"network.service.request.par_id"

Condition(s) : "network.init == ncTRUE"

Résultat(s) : La valeur du paramètre est générée à partir du texte d'entrée contenu dans "network.service.data_text" et transférée au variateur pour l'écriture du paramètre spécifié. Le gestionnaire NC détermine la taille et le format du paramètre grâce à l'ID de paramètre.

Indications(s) d'état: "network.service.response.par_id = 0"
après "**ncOK**" en tant qu'état résultant de l'action NC

"network.service.response.par_id = 0xFFFF"
et le message d'erreur correspondant dans "error"
après refus d'écrire le paramètre dans le variateur

"network.service.response.par_id = ...service.request.par_id"
après écriture réussie du paramètre

Exemple :

```
switch (step)
{
...
  case PAR_SET:
    p_ax_dat->network.service.request.par_id = ACP10PAR_...;
    <Entrée de la valeur dans "p_ax_dat->network.service.data_text">
    action_status = ncaction(ax_obj,ncSERVICE,ncREAD+ncDATA_TEXT);
    if ( action_status == ncOK )
    {
      step = W_PAR_SET;
    }
    break;

  case W_PAR_SET:
    if (p_ax_dat->network.service.response.par_id != 0)
    {
    /* Opération terminée */
      if ( p_ax_dat->network.service.response.par_id ==
           p_ax_dat->network.service.request.par_id )
      {
      /* Opération terminée avec succès */
        step = <NEXT_STEP>
      }
      else
      {
      /* Réponse erreur : message d'erreur dans "error" */
        step = <ERROR_STEP>
      }
    }
    break;
...
}
```

2.1.4 Initialisation globale

2.1.4.1 Structure de données

global		**Initialisation globale**
init	USINT	Initialisation globale terminée ncFALSE/ncTRUE

La composante "global.init" indique si tous les paramètres dans le variateur requis pour l'utilisation de cet objet NC sont initialisés. Une liste des paramètres peut être trouvée dans la description de l'action NC "ncGLOBAL,ncINIT".

2.1.4.2 Actions NC

sujet	action	Description
ncGLOBAL	ncINIT	Initialisation globale

2.1.4.2.1 Initialisation globale

Appel de fonction : `status = ncaction(ax_obj,ncGLOBAL,ncINIT)`

Paramètre : "dig_in.level"
"encoder_if.parameter"
"limit.parameter"
"controller.position", "controller.speed"
"stop.parameter"
"move.homing.parameter"
"move.basis.parameter"

Condition(s) : "network.init == ncTRUE"
"controller.status == ncON"

Résultat(s) : Initialisation globale des composantes citées ci-dessus.
A cet effet, le gestionnaire NC appelle en interne les actions NC d'initialisation correspondantes.

Indications(s) d'état : "global.init = ncFALSE"
"<...>.init = ncFALSE" pour tous les "<...>.init" des actions NC d'initilisation qui ont été appelées,
après "**ncOK**" en tant qu'état résultant de l'action NC

"global.init = ncTRUE"
"<...>.init = ncTRUE" pour tous les "<...>.init" des actions NC d'initilisation appelées
(et toutes les autres indications d'état générées une fois que ces actions NC ont été exécutées),
après que l'initialisation globale a été effectuée avec succès

Important : Tous les paramètres requis pour cette action NC peuvent être entrés dans un module de paramètres d'initialisation (module de données) dans B&R-AutomationStudio puis transférés au contrôleur avec ce module de données. Lors du démarrage du contrôleur, le gestionnaire NC reporte ensuite automatiquement dans la structure NC les paramètres contenus dans ce module de données De plus, il active automatiquement cette action NC. Dans le cas des objets NC pour lesquels un module de paramètres d'initialisation existe dans le contrôleur, il n'est donc pas nécessaire d'appeler les actions NC d'initialisation contenues dans cette action NC.

L'état "global.init" est alors généré si les différentes actions NC d'initialisation sont utilisées.

Etat :

```
switch (step)
{
...
  case GLOBAL_INIT:
  /* Initialisation des paramètres */
    p_ax_dat->... = ...;
    ...
    action_status = ncaction(ax_obj,ncGLOBAL,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_GLOBAL_INIT;
    }
    break;

  case W_GLOBAL_INIT:
    if (p_ax_dat->global.init == ncTRUE)
    { /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.5 Mode simulation

2.1.5.1 Structure de données

simulation		**Mode simulation**
status	USINT	Status : ncON/ncOFF

En mode simulation, vous disposez de toutes les fonctionnalités du système d'exploitation ACP10 sans utiliser les interfaces matérielles du variateur. Dans ce mode de fonctionnement, vous pouvez tester avec le variateur des parties du programme d'application indépendantes du matériel, sans connecter des composants matériels (moteur, codeur…).

Important: En mode simulation, la fonctionnalité d'entrée digitale est active !

Ceci a une incidence sur le processus de prise de référence (interrupteur de position de référence ou de fin de course) ou sur les différents mouvements (interrupteur de fin de course, entrée de trigger). Un changement des niveaux actifs pour les entrées ou bien un forçage des états des entrées digitales avec la valeur requise (fonction Force), peut s'avérer nécessaire.

2.1.5.2 Actions NC

sujet	**Action**	**Description**
ncSIMULATION	NcSWITCH_ON	Activer le mode simulation
ncSIMULATION	ncSWITCH_OFF	Désactiver le mode simulation

2.1.5.2.1 Activation du mode simulation

Appel de fonction : `status = ncaction(ax_obj,ncSIMULATION,ncSWITCH_ON)`

Paramètre(s) : -

Conditions : "`network.init == ncTRUE`"
"`controller.status == ncOFF`"

Résultat(s): Activation du mode simulation

Indication(s) d'état : "`simulation.status = ncON`"
après activation du mode simulation

Exemple :

```
switch (step)
{
...
  case SIM_ON:
    action_status=ncaction(ax_obj,ncSIMULATION,ncSWITCH_ON);
    if ( action_status == ncOK )
    {
      step = W_SIM_ON;
    }
    break;

  case W_SIM_ON:
    if (p_ax_dat->simulation.status == ncON)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.5.2.2 Désactivation du mode simulation

Appel de fonction : `status = ncaction(ax_obj,ncSIMULATION,ncSWITCH_OFF)`

Paramètre : -

Condition(s) : "`network.init == ncTRUE`"
"`controller.status == ncOFF`"

Résultat(s) : Désactivation du mode simulation

Indication d'état(s) : "`simulation.status = ncOFF`"
après désactivation du mode simulation

Exemple :

```
switch (step)
{
...
  case SIM_OFF:
    action_status=ncaction(ax_obj,ncSIMULATION,ncSWITCH_OFF);
    if ( action_status == ncOK )
    {
      step = W_SIM_OFF;
    }
    break;

  case W_SIM_OFF:
    if (p_ax_dat->simulation.status == ncOFF)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.6 Entrées digitales

2.1.6.1 Structure de données

dig_in		**Entrées digitales**
init	USINT	Entrées digitales initialisées : ncFALSE/ncTRUE
status		**Etat**
reference	USINT	Interrupteur de position d'origine : ncOPEN/ncCLOSED
pos_hw_end	USINT	Interrupteur de fin de course positif : ncOPEN/ncCLOSED
neg_hw_end	USINT	Interrupteur de fin de course négatif : ncOPEN/ncCLOSED
trigger1	USINT	Trigger1 : ncOPEN/ncCLOSED
trigger2	USINT	Trigger2 : ncOPEN/ncCLOSED
level		**Niveau d'entrée actif**
reference	UINT	Interrupteur de position d'origine : ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP)
pos_hw_end	UINT	Interrupteur de fin de course positif : ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP)
neg_hw_end	UINT	Interrupteur de fin de course négatif : ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP)
trigger1	UINT	Trigger1 : ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP)
trigger2	UINT	Trigger2 : ncACTIV_LO/ncACTIV_HI (+ncFORCE)(+ncQUICKSTOP) Avec "+ncFORCE", c'est l'état appliqué via la fonction de forçage est qui est évalué, et non l'entrée matérielle digitale. Avec "+ncQUICKSTOP", le niveau défini de l'entrée matérielle digitale déclenche la fonction Quickstop.
force		**Application d'états aux entrées digitales avec la fonction de forçage**
reference	USINT	Interrupteur de position d'origine : ncOPEN/ncCLOSED
pos_hw_end	USINT	Interrupteur de fin de course positif : ncOPEN/ncCLOSED
neg_hw_end	USINT	Interrupteur de fin de course négatif : ncOPEN/ncCLOSED
trigger1	USINT	Trigger1 : ncOPEN/ncCLOSED
trigger2	USINT	Trigger2 : ncOPEN/ncCLOSED

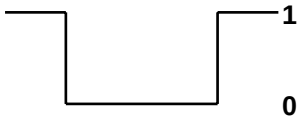
L'**état logique** d'une entrée digitale sur le variateur est indiqué dans "`dig_in.status.`<*Input*>". L'état logique résulte de l'état physique de l'entrée et de son niveau actif d'entrée initialisé "`dig_in.level.`<*Input*>".

"**`dig_in.level.`**<***Input***> **= ncACTIV_HI**" conduit à la relation suivante entre l'état physique et l'état logique :

Etat physique	Etat logique
1 (haut)	**ncCLOSED**
0 (bas)	**ncOPEN**

"**`dig_in.level.`**<***Input***> **= ncACTIV_LO**" conduit à la relation inverse :

Etat physique	Etat logique
1 (haut)	**ncOPEN**
0 (bas)	**ncCLOSED**



Pour chacune des entrées digitales où la fonction de forçage a été appelée avec "**`dig_in.level.`**<***Input***>**=... + ncFORCE**", "**`dig_in.force.`**<***Input***>**=ncOPEN/ncCLOSED**" suivi de l'appel de la **fonction de forçage** permet d'affecter à l'état la valeur souhaitée, dans la tâche d'application.

Important : La fonctionnalité d'entrée digitale est aussi active en mode simulation. Ceci a une incidence sur le processus de prise de référence (interrupteur de position de référence, interrupteur de fin de course) ou sur les différents mouvements (interrupteur de fin de course). Un changement des niveaux actifs pour les entrées ou bien un forçage des états des entrées digitales avec la valeur requise peut s'avérer nécessaire.

2.1.6.2 Actions NC

Sujet	Action	Description
ncDIG_IN	ncINIT	Initialiser les entrées digitales
ncDIG_IN	ncFORCE	Appliquer un état aux entrées digitales avec la fonction de forçage

2.1.6.2.1 Initialisation des entrées digitales

Appel de fonction : `status = ncaction(ax_obj,ncDIG_IN,ncINIT)`

Paramètre : "`dig_in.level`"

Condition(s) : "`network.init == ncTRUE`"
"`controller.status == ncAUS`"

Résultat(s) : Initialisation de la fonctionnalité d'entrée digitale sur le variateur

Indication(s) d'état "`dig_in.init = ncFALSE`"
après "**ncOK**" en tant qu'état résultant de l'action NC

"`dig_in.init = ncTRUE`"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case DIG_IN_INIT:
    p_ax_dat->dig_in.level.homing    = ncACTIV_HI;
    p_ax_dat->dig_in.level.pos_hw_end = ncACTIV_LO;
    p_ax_dat->dig_in.level.neg_hw_end = ncACTIV_LO;
    p_ax_dat->dig_in.level.trigger1   = ncACTIV_HI;
    p_ax_dat->dig_in.level.trigger2   = ncACTIV_HI + ncFORCE;

    action_status = ncaction(ax_obj,ncDIG_E,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_DIG_IN_INIT;
    }
    break;

  case W_DIG_IN_INIT:
    if (p_ax_dat->dig_e.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.6.2.2 Application d'un état aux entrées digitales via la fonction de forçage

Appel de fonction : `status = ncaction(ax_obj,ncDIG_IN,ncFORCE)`

Paramètre : "`dig_in.force`"

Condition(s) : "`network.init == ncTRUE`"

Résultat(s) : L'état reporté dans "`dig_in.force`" est transféré au variateur. Ensuite, dans le variateur, cet état est appliqué aux entrées digitales pour lesquelles le mode "+`ncFORCE`" a été initialisé avec "`dig_in.level`".

Indication(s) d'état : Etat des entrées digitales forcées dans "`dig_in.status`"
une fois l'exécution de la fonction de forçage terminée

Exemple :

```
switch (step)
{
...
  case DIG_IN_FORCE:
/* agit seulement si ".level = ...+ncFORCE" est initialisé */
    p_ax_dat->dig_in.force.trigger2 = ncCLOSE;

    action_status = ncaction(ax_obj,ncDIG_IN,ncFORCE);
    if ( action_status == ncOK )
    {
      step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.7 Interface codeur

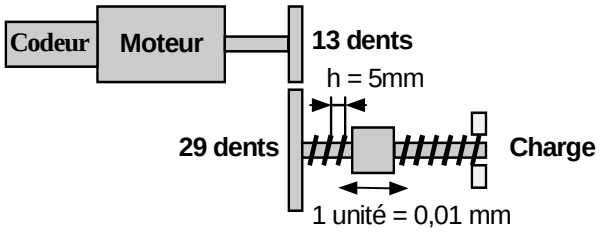
2.1.7.1 Structure de données

encoder_if		**Interface codeur**
init	USINT	Interface codeur initialisée : ncFALSE/ncTRUE
parameter		**Paramètre**
count_dir	USINT	Sens de comptage : ncSTANDARD/ncINVERSE
scaling		**Echelle**
load		**Charge [unités/tour_moteur]**
units	UDINT	Unités sur la charge
rev_motor	UDINT	Tours moteur

"encoder_if.parameter.count_dir = nc**INVERSE**" permet d'inverser le sens de comptage du codeur dans le cas où celui-ci ne correspond pas au sens de comptage souhaité. Ceci n'a aucune incidence sur l'indication du sens de comptage (UP- , DN-LED) sur le module interface codeur du variateur.

L'utilisateur peut utiliser l'unité de son choix (par exemple mm, 1/100 mm, 1/20 pouce, degré d'angle etc.) pour les positions (et les données qui en découlent comme par exemple la vitesse et l'accélération). A cet effet, il faut d'abord définir la relation entre un multiple entier de cette unité et un nombre donné de tours moteur dans la composante "encoder_if.parameter.scaling".

Exemple :



29tour$_{moteur}$ ≡ **13**tour$_{charge}$ et 1tour$_{charge}$ ≡ **5**mm (h=5mm) et 1mm ≡ **100**unités(1unité = 0,01mm)

⇒ `scaling.load.units = 13 * 5 * 100`
`scaling.load.rev_motor = 29`

2.1.7.2 Actions NC

Sujet	Action	Description
ncENCODER_IF	ncINIT	Initialiser l'interface codeur

2.1.7.2.1 Initialisation de l'interface codeur

Appel de fonction : `status = ncaction(ax_obj,ncENCODER_IF,ncINIT)`

Paramètre : "encoder_if.parameter"

Condition(s) : "network.init == ncTRUE"
"controller.status == ncOFF"

Résultat(s) : Initialisation de l'interface codeur sur le variateur

Indication(s) d'état : "encoder_if.init = ncFALSE"
après "**ncOK**" en tant qu'état résultant de l'action NC

"move.homing.status.ok = ncFALSE"
"encoder_if.init = ncTRUE"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case ENCODER_IF_INIT:
    p_ax_dat->encoder_if.parameter.count_dir = ...;

    p_ax_dat->encoder_if.parameter.scaling.load.units = ...;
    p_ax_dat->encoder_if.parameter.scaling.load.rev_motor = ...;

    action_status = ncaction(ax_obj,ncENCODER_IF,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_ENCODER_IF_INIT;
    }
    break;

  case W_ENCODER_IF_INIT:
    if (p_ax_dat->encoder_if.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.8 Valeurs limites

2.1.8.1 Structure de données

limit		**Valeurs limites**
init	USINT	Valeurs limites d'axe initialisées : ncFALSE/ncTRUE
parameter		**Paramètre**
v_pos	REAL	Vitesse dans la direction positive [unité/s]
v_neg	REAL	Vitesse dans la direction négative [unité/s]
a1_pos	REAL	Accélération dans la direction positive [unité/s^2]
a2_pos	REAL	Décélération dans la direction positive [unité/s^2]
a1_neg	REAL	Accélération dans la direction négative [unité/s^2]
a2_neg	REAL	Décélération dans la direction négative [unité/s^2]
t_jolt	REAL	Temps de vibration [s]
t_in_pos	REAL	Temps d'attente avant le message "In Position" [s]
pos_sw_end	DINT	Fin de course logiciel positif [unité]
neg_sw_end	DINT	Fin de course logiciel négatif [unité]
ds_warning	REAL	Valeur limite de l'erreur de traînage pour lancer un avertissement
ds_stop	REAL	Valeur limite de l'erreur de traînage pour stopper un mouvement

Fins de course logiciels

La plage de mouvement est délimitée par les fins de course logiciels. Le fin de course logiciel **pos**itif est défini avec "limit.parameter.**pos**_sw_end" et le fin de course **nég**atif avec "limit.parameter.**neg**_sw_end".

Tout démarrage d'un mouvement avec une position cible (par exemple "ncABS_MOVE", "ncREL_MOVE") au-delà de ces fins de course est refusé et entraîne l'apparition d'un message d'erreur.

Dans le cas des mouvements sans position cible définie (par exemple "ncPOS_MOVE", "ncNEG_MOVE"), il se produit un freinage accompagné d'un message d'erreur et entraînant l'arrêt aux fins de course logiciels.

Surveillance de l'erreur de traînage

Deux valeurs limites, "ds_warning" et "ds_stop", sont définies pour la surveillance de l'erreur de traînage (différence entre la position de consigne et la position réelle dans le régulateur de position).

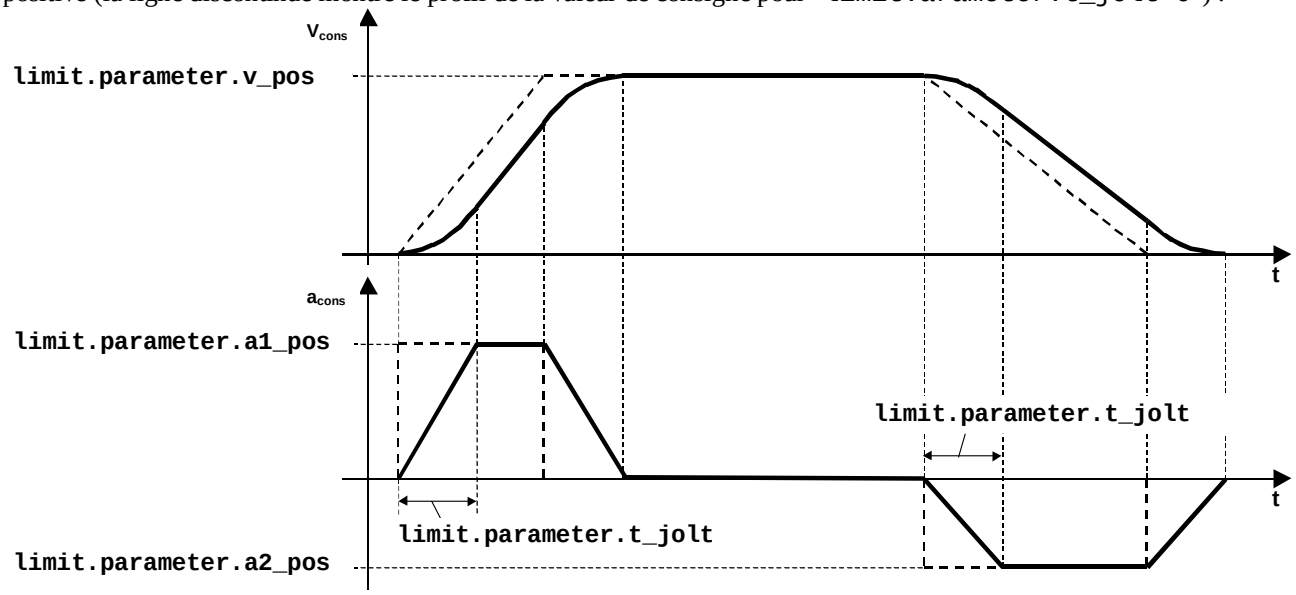
Si la valeur absolue de l'erreur de traînage est supérieure à la valeur indiquée dans "limit.parameter.**ds_warning**", alors "monitor.status.**ds_warnung**" prend la valeur "ncTRUE" (dans le cas contraire, "ncFALSE").

Dans le cas où la valeur absolue de l'erreur de traînage dépasse la valeur reportée dans "limit.parameter.**ds_warning**", tout mouvement actif est **stoppé** (freinage jusqu'à l'arrêt) avec une décélération correspondant à "limit.parameter.a2_pos/a2_neg" et ensuite, le régulateur est désactivé. De plus, un message d'erreur est généré.

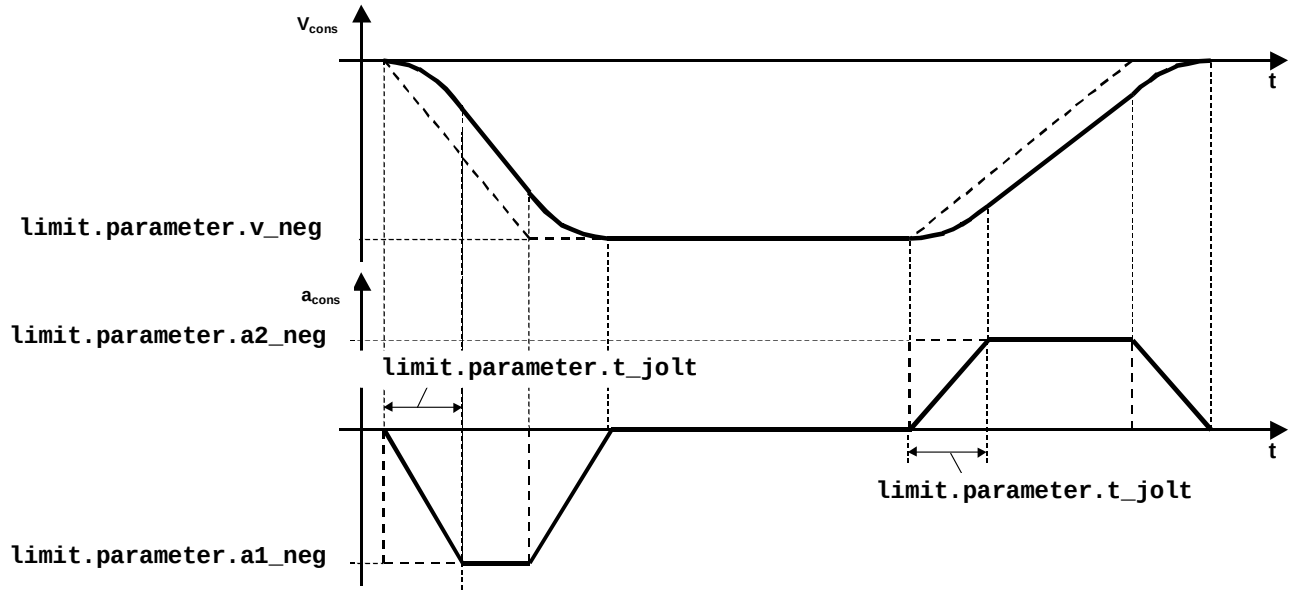
Profil de mouvement

Pour le profil du mouvement, on peut définir dans "`limit.parameter`" une limite de vitesse ("`v_`**pos**`/v_`**neg**"), d'accélération ("`a1_`**pos**`/a1_`**neg**") et de décélération ("`a2_`**pos**`/a2_`**neg**"), et ce pour les directions **pos**itive et **nég**ative. De plus, "`t_jolt`" permet de définir le temps que met l'accélération pour passer de zéro à la valeur maximale paramétrée pour le positionnement concerné. Un filtre linéaire avec un temps de filtre de "`t_jolt`" limite les vibrations. Malgré le temps requis pour la limitation des vibrations, la position cible est souvent atteinte plus tôt du fait de la réduction du temps d'établissement.

La figure qui suit montre le lien existant entre les différents paramètres pour un mouvement effectué dans la direction positive (la ligne discontinue montre le profil de la valeur de consigne pour "`limit.arameter.t_jolt=0`") :

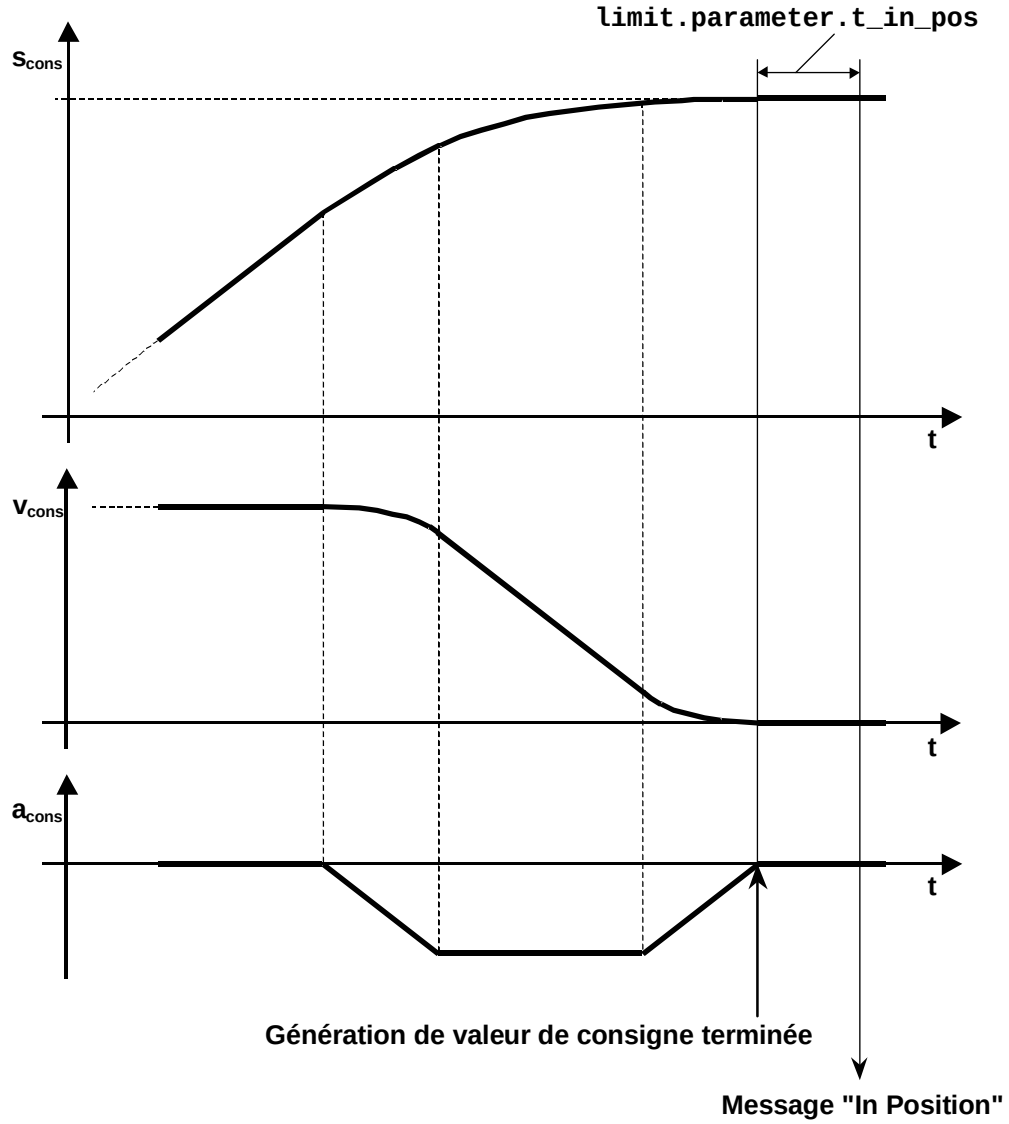


La figure qui suit montre l'action des différents paramètres pour un mouvement effectué dans la direction négative (la ligne discontinue montre le profil de la valeur de consigne pour "`limit.parameter.t_jolt=0`") :



Temps d'attente pour le message "In Position"

"`limit.parameter.t_in_pos`" permet de définir un temps d'attente devant s'écouler après que la valeur de consigne a été générée (position de consigne et position cible identiques) et avant que le message "In Position" n'apparaisse. Ce temps doit être au moins aussi long que le temps d'établissement de l'axe.



Important : Le message "In Position" n'est pas lié à la position réelle. Il dépend uniquement du moment où la génération de valeur de consigne est terminée et du temps d'attente défini avec "`limit.parameter.t_in_pos`".

2.1.8.2 Actions NC

sujet	action	Description
ncLIMITS	ncINIT	Initialiser les valeurs limites

2.1.8.2.1 Initialisation des valeurs limites

Appel de fonction : `status = ncaction(ax_obj,ncLIMITS,ncINIT)`

Paramètre : `"limit.parameter"`

Condition(s) : `"network.init == ncTRUE"`
`"move.mode == ncOFF"`

Résultat(s) : Initialisation des valeurs limites sur le variateur

Indications(s) d'état : `"limit.init = ncFALSE"`
après "**ncOK**" en tant qu'état résultant de l'action NC

`"limit.init = ncTRUE"`
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case LIMITS_INIT:
    p_ax_dat->limit.parameter.v_pos      = ...;
    p_ax_dat->limit.parameter.v_neg      = ...;
    p_ax_dat->limit.parameter.a1_pos     = ...;
    p_ax_dat->limit.parameter.a2_pos     = ...;
    p_ax_dat->limit.parameter.a1_neg     = ...;
    p_ax_dat->limit.parameter.a2_neg     = ...;
    p_ax_dat->limit.parameter.t_jolt     = ...;
    p_ax_dat->limit.parameter.t_in_pos   = ...;
    p_ax_dat->limit.parameter.pos_sw_end = ...;
    p_ax_dat->limit.parameter.neg_sw_end = ...;
    p_ax_dat->limit.parameter.ds_warning = ...;
    p_ax_dat->limit.parameter.ds_stop    = ...;

    action_status = ncaction(ax_obj,ncLIMITS,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_LIMITS_INIT;
    }
    break;

  case W_LIMITS_INIT:
    if (p_ax_dat->limit.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.9 Régulateur

2.1.9.1 Structure de données

controller		**Régulateur**
init	USINT	Initialisation du régulateur : ncFALSE/ncTRUE
ready	USINT	Régulateur prêt à être activé : ncFALSE/ncTRUE
status	USINT	Etat : ncOFF/ncON
position		**Régulateur de position**
kv	REAL	Amplification proportionnelle [1/s]
tn	REAL	Temps d'action intégrale [s]
t_predict	REAL	Temps de prédiction [s]
t_total	REAL	Retard total [s]
p_max	REAL	Action proportionnelle max. [unité/s]
i_max	REAL	Action intégrale max. [unité/s]
speed		**Régulateur de vitesse**
kv	REAL	Amplification proportionnelle [A s / tour]
tn	REAL	Temps d'action intégrale [s]

Etat "Régulateur prêt"

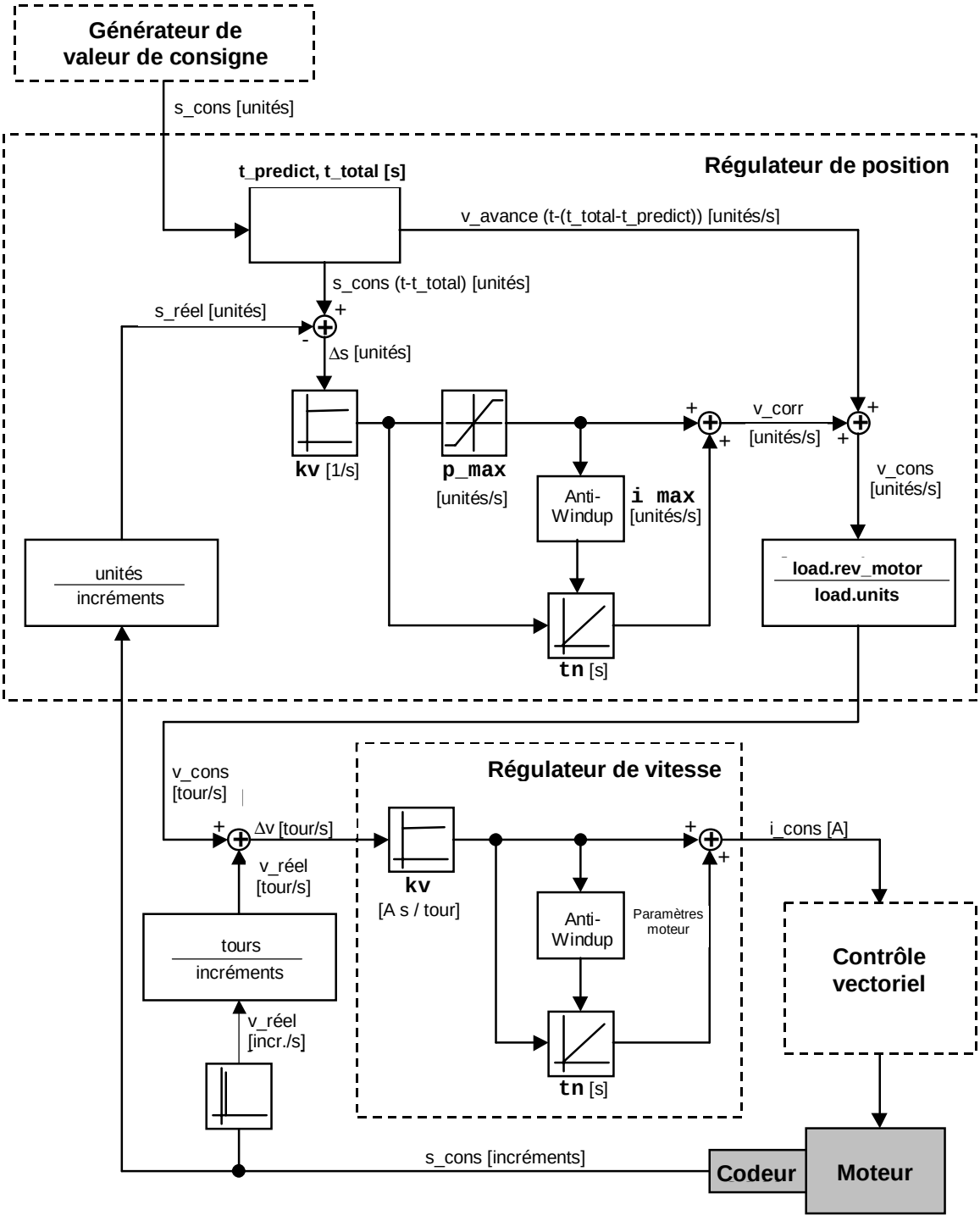
Lorsque l'état "controller.ready=ncTRUE" n'est pas vérifié, l'Acopos se trouve en état d'erreur (la LED rouge est allumée). Les états d'erreur peuvent être dus au fait que certaines conditions externes ne sont pas remplies, comme par exemple l'établissement de la tension de bus DC ou encore l'alimentation de l'entrée "controller enable" (connecteur mâle X1, broche 8) en 24 V. Le moment où ces conditions externes sont remplies n'étant pas connu du système d'exploitation de l'Acopos, aucun enregistrement d'erreur pour une description plus détaillée des erreurs n'est reporté dans la FIFO des erreurs ou indiqué dans la structure de données utilisateur.

Lorsque le programme d'application a créé toutes les conditions (tension de bus DC présente, par exemple) pour que "controller.ready=ncTRUE" soit vérifié, toutes ces conditions devraient être également vérifiées dans le système d'exploitation de l'Acopos au plus tard au bout de 5 secondes. Si "controller.ready=ncTRUE" ne s'affiche pas au bout de 5 secondes, on peut en déduire à coup sûr que l'Acopos est en état d'erreur ; le programme d'application peut déclencher le report des enregistrements d'erreurs dans la FIFO en mettant le paramètre ayant pour ID "**CMD_ERR_STATE_INTO_FIFO**" à l'état actif.

Le paramètre ayant pour ID "**CMD_ERR_STATE_INTO_FIFO**" est disponible à partir de la version V0.420 d'ACP10-SW. Dans les versions antérieures d'ACP10-SW, le report des enregistrements d'erreurs permettant de décrire plus en détail les erreurs présentes peut être déclenché en mettant le paramètre ayant pour ID "**CMD_CONTROLLER**" à l'état actif, avec le code "ncSWITCH_ON".

Schéma-bloc

Le schéma-bloc ci-dessous met en évidence le mode d'action des différents paramètres du régulateur :

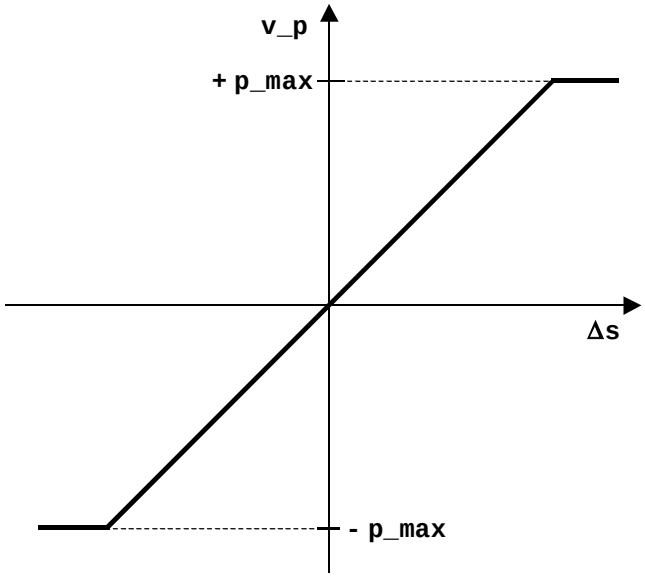


Calcul de "v_corr"

La vitesse de correction "**v_corr**" résultant de l'erreur de traînage "**Δs**" est calculée selon la séquence suivante :

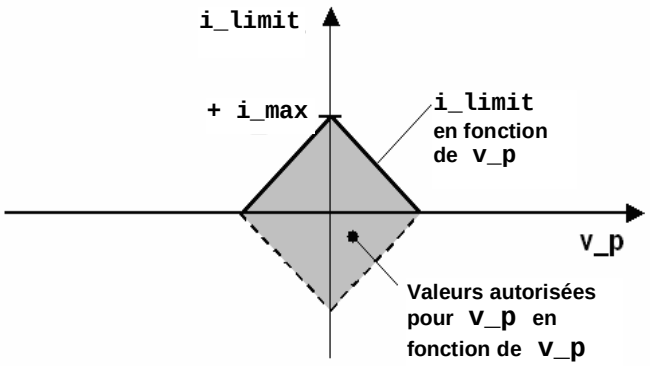
Calcul de la vitesse "**v_p**" résultant de l'amplification proportionnelle et limitation de cette vitesse à "**p_max**" :

```
v_p = kv * Δs

if ( v_p > p_max )
    v_p = p_max
else if ( v_p < -p_max )
    v_p = -p_max
```



Calcul de "**i_limit**" avec la valeur ainsi obtenue et "**i_max**", puis limitation de la vitesse "**v_i**" résultant de l'amplification intégrale à cette valeur "**i_limit**" :

```
i_limit = i_max - |v_p|

if ( i_limit < 0 )
    i_limit = 0

v_i = f(Δs,kv,tn)

if ( v_i > i_limit )
    v_i = i_limit
else if ( v_i < -i_limit )
    v_i = -i_limit
```



Enfin, "**v_corr** = **v_p** + **v_i**" peut être calculé.

2.1.9.2 Actions NC

sujet	action	Description
ncCONTROLLER	ncINIT	Initialiser le régulateur
ncCONTROLLER	ncSWITCH_ON	Activer le régulateur
ncCONTROLLER	ncSWITCH_OFF	Désactiver le régulateur

2.1.9.2.1 Initialisation du régulateur

Appel de fonction : `status = ncaction(ax_obj,ncREGLER,ncINIT)`

Paramètres : "`controller.position`"
"`controller.speed`"

Condition(s) : "`network.init == ncTRUE`"

Résultats(s) : Initialisation du régulateur avec les paramètres prédéfinis

Indications(s) d'état :"`controller.init = ncFALSE`"
après "**`ncOK`**", état résultant de l'action NC

"`controller.init = ncWAHR`"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case CONTROLLER_INIT:
    p_ax_dat->controller.position.kv       = ...;
    p_ax_dat->controller.position.tn       = ...;
    p_ax_dat->controller.position.t_voraus = ...;
    p_ax_dat->controller.position.t_gesamt = ...;
    p_ax_dat->controller.position.p_max    = ...;
    p_ax_dat->controller.position.i_max    = ...;
    p_ax_dat->controller.speed.kv   = ...;
    p_ax_dat->controller.speed.tn   = ...;

    action_status = ncaction(ax_obj,ncCONTROLLER,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_CONTROLLER_INIT;
    }
    break;

  case W_CONTROLLER_INIT:
    if (p_ax_dat->controller.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Il est également possible d'appeler cette action NC lorsque le régulateur est activé.

2.1.9.2.2 Activation du régulateur

Appel de fonction : `status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_ON)`

Paramètre : -

Condition(s) : "`controller.ready == ncTRUE`"
On ne doit pas avoir "`dig_in.status.neg_hw_end == ncCLOSED`" et en même temps "`dig_in.status.pos_hw_end == ncCLOSED`"

Résultat(s) : Activation du régulateur

Indication(s) d'état : "`controller.status = ncON`"
après activation du régulateur

Exemple :

```
switch (step)
{
...
  case W_CONTROLLER_READY:
    if (p_ax_dat->controller.ready == ncTRUE)
    {
    /* Régulateur prêt à être activé */
        step = CONTROLLER_ON;
    }
    break;

  case CONTROLLER_ON:
    action_status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_ON);
    if ( action_status == ncOK )
    {
      step = W_CONTROLLER_ON;
    }
    break;

  case W_CONTROLLER_ON:
    if (p_ax_dat->controller.status == ncON)
    /* Opération terminée avec succès */
    {
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.9.2.3 Désactivation du régulateur

Appel de fonction : `status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_OFF)`

Paramètre : -

Condition(s) : "`move.mode == ncOFF`"

Résultat(s) : Désactivation du régulateur

Indication(s) d'état : "`controller.status = ncOFF`"
après désactivation du régulateur

Exemple :

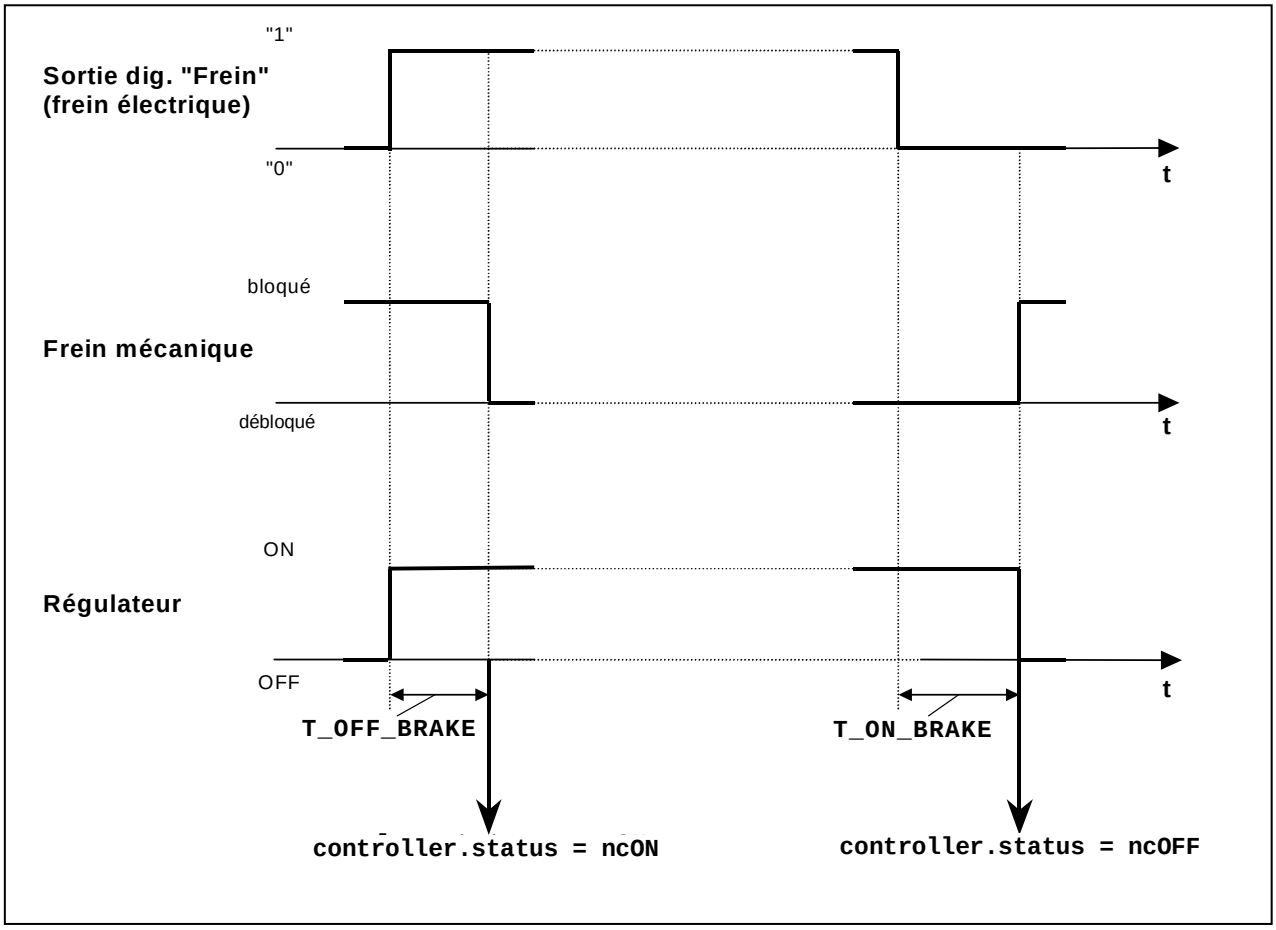
```
switch (step)
{
...
  case CONTROLLER_OFF:
    action_status = ncaction(ax_obj,ncCONTROLLER,ncSWITCH_OFF);
    if ( action_status == ncOK )
    {
      step = W_CONTROLLER_OFF;
    }
    break;

  case W_CONTROLLER_OFF:
    if (p_ax_dat->controller.status == ncOFF)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.10 Frein

Les paramètres relatifs au frein font partie des paramètres moteur. Ces derniers ne se trouvent pas dans la structure NC mais dans l'interface EnDat ou dans un module de données OEM. Le variateur vérifie si un frein est présent (paramètre "I_RATED_BRAKE" différent de zéro) et si oui, lit les paramètres requis pour la commande du frein.

Commande automatique du frein (cas normal)

Normalement, dans le cas où le moteur comporte un frein (le paramètre "BRAKE_MODE" a la valeur "1"), la sortie digitale "Frein" du variateur est automatiquement commandée par le système d'exploitation NC du variateur, par activation et désactivation du régulateur (et la surveillance de courant et de tension au niveau du relais de frein est activée) :



Après l'activation du régulateur sur le variateur, la sortie digitale "Frein" est mise à "1" pour déclencher le freinage. Une fois le temps "T_OFF_BRAKE" écoulé (temps de retard avant déclenchement du freinage), "controller.status = ncON" est appliqué à la structure NC.

Avant la désactivation du régulateur, la sortie digitale "Frein" est mise à "0" pour bloquer le frein. Une fois le temps "T_ON_BRAKE" écoulé (temps de retard avant blocage du frein), le régulateur est désactivé. Après désactivation du régulateur, "controller.status = ncOFF" est appliqué à la structure NC.

En modifiant les paramètres décrits ci-après pour la commande du frein, il est possible d'adapter la commande automatique du frein qui vient d'être décrite aux besoins de l'application.

Paramètres pour la commande du frein

- **Mode de commande pour le frein**

```
ACP10PAR_BRAKE_MODE  90 /* (UINT) Mode de commande pour le frein */
```

Bit 0 : Commande automatique par le système d'exploitation NC (0/1: Commande automatique OFF/ON)
Bit 1 : Activation de la commande manuelle, aussi avec "Controller ON" (0/1 : Activation OFF/ON)
Bit 2 : Surveillance de courant et de tension au niveau du relais de frein (***invers***) (0/1 : Surveillance ON/OFF)

Bit2	Bit1	Bit0	Mode	Effets
0	0	0	**0**	Le frein **n'**est **pas** commandé **par le système d'exploitation NC** du variateur. La **commande** manuelle du frein avec le paramètre "CMD_BRAKE" est refusée et provoque un **message d'erreur**. Si le frein n'est pas connecté au variateur, il peut être commandé indépendamment de ACP10-SW moyennant un câble approprié.
0	0	1	**1**	Le frein est commandé **automatiquement par le système d'exploitation NC** du variateur (en fonction de l'état du régulateur). Le frein ne peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE" **que si l'on a "Controller OFF"**. **Surveillance** de courant et de tension au niveau du relais de frein.
0	1	0	**2**	Le frein **n'**est **pas** commandé **par le système d'exploitation NC** du variateur. Le frein peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE", **que l'on ait "Controller OFF" ou "Controller ON"**. **Surveillance** de courant et de tension au niveau du relais de frein.
0	1	1	**3**	Le frein est commandé automatiquement **par le système d'exploitation NC** du variateur (en fonction de l'état du régulateur). Le frein peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE", **que l'on ait "Controller OFF" ou "Controller ON"**. **Surveillance** de courant et de tension au niveau du relais de frein.
1	0	0	**4**	Mêmes effets qu'en mode "0".
1	0	1	**5**	Le frein est commandé **automatiquement par le système d'exploitation NC** du variateur (en fonction de l'état du régulateur). Le frein ne peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE" **que si l'on a "Controller OFF"**. **Pas de surveillance** de courant et de tension au niveau du relais de frein.
1	1	0	**6**	Le frein **n'**est **pas** commandé **par le système d'exploitation NC** du variateur. Le frein peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE", **que l'on ait "Controller OFF" ou "Controller ON"**. **Pas de surveillance** de courant et de tension au niveau du relais de frein.
1	1	1	**7**	Le frein est commandé **automatiquement par le système d'exploitation NC** du variateur (en fonction de l'état du régulateur). Le frein peut être commandé (bloqué et débloqué) manuellement avec le paramètre "CMD_BRAKE", **que l'on ait "Controller OFF" ou "Controller ON"**. **Pas de surveillance** de courant et de tension au niveau du relais de frein.

- **Adaptation des temps de retard**

```
ACP10PAR_T_ON_BRAKE  44 /* (REAL) Temps de retard: bloquer le frein [s] */
ACP10PAR_T_OFF_BRAKE 45 /* (REAL) Temps de retard: débloquer le frein [s] */
```

Les temps de retard définis par le fabricant des freins pour le blocage et le déblocage se trouvent dans les données OEM et peuvent être modifiés si nécessaire avec ces paramètres.

- **Déblocage et blocage manuel du frein**

```
ACP10PAR_CMD_BRAKE   86 /* (UINT) Commande: débloquer/bloquer le frein */
```

L'écriture dans le variateur de la valeur de la constante "ncSWITCH_OFF" provoque le déblocage du frein. A l'inverse, l'écriture de la valeur de la constante "ncSWITCH_ON" entraîne le blocage du frein.

- **Etat des freins**

```
ACP10PAR_STATUS_BITS 178 /* (UDINT) Bits d'état général*/
```

L'état du frein est lu par le biais des bits d'état général. Si le Bit2 de ce paramètre est activé, le frein est bloqué ; dans la cas contraire, le frein est débloqué.

2.1.11 Etat général d'un mouvement

2.1.11.1 Structures de données

Mouvement		Mouvement	
mode	UINT	Mode :	
		ncOFF	Pas de mouvement actif
		ncSTOP	Arrêt d'un mouvement actif
		ncHOMING	Prise de référence active
		ncBASIS_MOVE	Mouvement de base actif
		ncACTIV	Un mouvement qui n'a pas été lancé par une commande de mouvement du gestionnaire NC est actif (mouvement lancé, par exemple, par l'interface Service)
detail	UINT	Détail :	
		Général :	
		ncOFF	Pas de mouvement actif ou détails non requis
		Pour le mode "ncSTOP" :	
		ncSTOP	Arrêt après action NC "ncMOVE, ncSTOP"
		ncEVENT	Arrêt après événement dans le variateur
		Pour le mode "ncBASIS_MOVE" :	
		ncHALT	Arrêt d'un mouvement de base actif
		ncPOS_MOVE	Mouvement dans la direction positive
		(+ ncTRG_STOP)	(avec le mode "Stop after Trigger")
		ncNEG_MOVE	Mouvement dans la direction négative
		(+ ncTRG_STOP)	(avec le mode "Stop after Trigger")
		ncABS_MOVE	Mouvement avec position cible absolue
		(+ ncTRG_STOP)	(avec le mode "Stop after Trigger")
		(+ ncS_REST)	(avec variante "+ s_rest")
		ncREL_MOVE	Mouvement avec distance de déplacement relatif
		(+ ncTRG_STOP)	(avec le mode "Stop after Trigger")
		(+ ncS_REST)	(avec variante "+ s_rest")

"**move.mode==ncOFF**" indique qu'**aucun mouvement n'est actif** pour cet objet NC et que cet objet NC est donc au repos.

Dans les autres cas, "**move.mode**" et "**move.detail**" permettent de déterminer le **type du mouvement actuellement actif**.

Exemples :

Mouvement	Mode	Détail
Prise de référence	ncHOMING	ncOFF
Mouvement dans la direction positive	ncBASIS_MOVE	ncPOS_MOVE
Mouvement avec position cible absolue	ncBASIS_MOVE	ncABS_MOVE
Arrêt d'un mouvement après appel de l'action NC "ncMOVE,ncSTOP"	ncSTOP	ncSTOP
Arrêt d'un mouvement après l'apparition d'un événement, dans le variateur, nécessitant l'arrêt du mouvement	ncSTOP	ncEVENT

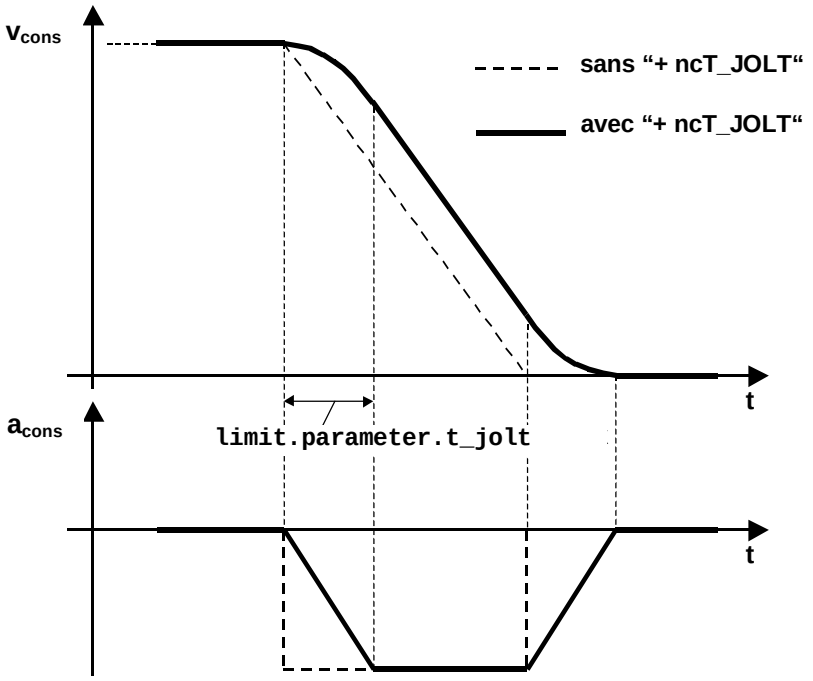
2.1.12 Arrêt d'un mouvement

2.1.12.1 Structure de données

move		**Mouvement**
...		
stop		**Arrêt d'un mouvement**
init	USINT	Arrêt initialisé : ncFALSE/ncTRUE
index		**Index d'enregistrement de paramètre**
command	USINT	Pour la commande d'arrêt suivante
parameter[4]		**Paramètres de la configuration d'arrêt**
decel_ramp	USINT	Rampe de freinage (paramétrage par défaut : ncA_LIMIT): ncA_MOVE: avec la décélération définie pour le mouvement actuel (+ ncT_JOLT): (et prise en compte du temps de vibration spécifié) ncA_LIMIT: avec la décélération définie dans les valeurs limites d'axe (+ ncT_JOLT): (et prise en compte du temps de vibration paramétré)
controller	USINT	Etat du régulateur après arrêt du mouvement : ncOFF/ncON (préréglage : ncON)

"move.stop.parameter" permet de définir plusieurs paramètres pour l'arrêt d'un mouvement. "move.stop.index.command" détermine les paramètres devant être utilisés pour arrêter un mouvement avec l'action NC "ncMOVE,ncSTOP".

"decel_rampe" permet de définir pour un jeu de paramètres donné la rampe de freinage précédant l'arrêt du mouvement. Si "ncA_**LIMIT**" est sélectionné, le freinage s'effectue avec la rampe de freinage définie dans les valeurs limites d'axes ("limit.parameter.a2_pos/a2_neg" pour la direction positive/négative). Avec "ncA_**MOVE**", la rampe de freinage utilisée est celle qui a été définie pour le **mouvement actuellement actif** (pour les mouvements de base, cela serait par exemple "move.basis.parameter.a2_pos/a2_neg" pour la direction positive/négative). Par ailleurs, "+ncT_JOLT" permet d'activer la limitation de vibration. Les effets de cette limitation sur la rampe de freinage sont illustrés dans les diagrammes ci-dessous :



Pour un jeu de paramètres, la composante "controller" détermine si le régulateur doit rester activé ("controller=ncON") ou non ("controller=ncOFF") une fois le mouvement arrêté.

2.1.12.2 Actions NC

Sujet	Action	Description
ncSTOP	ncINIT	Initialiser la configuration d'arrêt
ncMOVE	ncSTOP	Stopper le mouvement

2.1.12.2.1 Initialisation de la configuration d'arrêt

Appel de fonction : `status = ncaction(ax_obj,ncSTOP,ncINIT)`

Paramètre : "`move.stop.parameter`"

Condition(s) : "`network.init == ncTRUE`"
"`controller.status == ncOFF`"

Résultat(s) : Initialisation de la configuration d'arrêt dans le variateur

Indication(s) d'état : "`move.stop.init = ncFALSE`"
après "**ncOK**" en tant qu'état résultant de l'action NC

"`move.stop.init = ncTRUE`"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case STOP_INIT:
    p_ax_dat->move.stop.parameter[0].decel_rampe = ...;
    p_ax_dat->move.stop.parameter[0].controller  = ...;
    p_ax_dat->move.stop.parameter[1].decel_rampe = ...;
    p_ax_dat->move.stop.parameter[1].controller  = ...;
    p_ax_dat->move.stop.parameter[2].decel_rampe = ...;
    p_ax_dat->move.stop.parameter[2].controller  = ...;
    p_ax_dat->move.stop.parameter[3].decel_rampe = ...;
    p_ax_dat->move.stop.parameter[3].controller  = ...;

    action_status = ncaction(ax_obj,ncSTOP,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_STOP_INIT;
    }
    break;

  case W_STOP_INIT:
    if (p_ax_dat->stop.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.12.2.2 Arrêt d'un mouvement

Appel de fonction : `status = ncaction(ax_obj,ncMOVE,ncSTOP)`

Paramètre : "`move.stop.index.command`"

Condition(s) : "`controller.status == ncON`"
"`move.stop.init == ncTRUE`"

Résultat(s) : Freinage du mouvement avec l'enregistrement de paramètre sélectionné par "`index.command`"

Indications(s) d'état :"`move.mode = ncSTOP`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
une fois que l'arrêt du mouvement a été effectué

Exemple :

```
switch (step)
{
...
  case STOP:
    p_ax_dat->move.stop.index.command = 1;

    action_status = ncaction(ax_obj,ncMOVE,ncSTOP);
    if ( action_status == ncOK )
    {
      step = W_STOP;
    }
    break;

  case W_STOP:
    if (p_ax_dat->move.mode == ncOFF)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Contrairement à "`ncBASIS_MOVE,ncSTOP`", après "`ncMOVE,ncSTOP`", le démarrage d'un autre mouvement n'est pas autorisé pendant la rampe de décélération.

"`ncMOVE,ncSTOP`" est une action NC dont la priorité est plus haute que celle des actions NC standard. Cela signifie qu'en cas d'appel de cette action NC, le télégramme de commande CAN correspondant est immédiatement envoyé au variateur, même dans le cas où le traitement d'une action NC standard appelée précédemment n'est pas encore terminé.

2.1.13 Prise de référence

2.1.13.1 Structure de données

move		**Mouvement**
...		
homing		**Prise de référence**
init	USINT	Prise de référence initialisée : ncFALSE/ncTRUE
status		**Status**
ok	USINT	Position de référence valide : ncFALSE/ncTRUE
tr_s_rel	FLOAT	Distance parcourue entre l'activation du "déclenchement sur impulsion de référence" et l'apparition d'une impulsion de référence [tours codeur]
offset	DINT	Offset de référence une fois la prise de référence terminée [unité]
parameter		**Paramètres**
s	DINT	Position de référence ou offset de référence [unité]
v_switch	REAL	Vitesse pour la recherche de l'interrupteur de position de référence [unité/s]
v_trigger	REAL	Vitesse de déclenchement (après avoir atteint l'interrupteur de position de réf.) [unité/s]
a	REAL	Accélération [unité/s^2]
mode	USINT	Mode : ncDIRECT : direct ncSWITCH_GATE : avec porte générée par l'interrupteur de pos. de réf. ncABS_SWITCH : avec interrupteur de position de référence absolue ncREF_OFFSET : application d'un offset de prise de référence ncEND_SWITCH : avec interrupteur de fin de course matériel + ncCORRECTION : avec correction de la plage de comptage
edge_sw	USINT	Front de l'interrupteur de position de référence : ncPOSITIV/ncNEGATIV
start_dir	USINT	Direction de départ : ncPOSITIV/ncNEGATIV
trigg_dir	USINT	Direction de déclenchement : ncPOSITIV/ncNEGATIV
ref_pulse	USINT	Impulsion de référence : ncOFF/ncON
tr_s_block	FLOAT	Distance pour le blocage de l'activation du "Déclenchement sur impulsion de référence" [tours codeur]

Important : En mode simulation, les entrées digitales (interrupteur de position de référence et de fin de course matériel) sont aussi traitées en fonction du mode de prise de référence.

Position de référence / Offset de référence

La prise de référence établit la relation entre l'état actuel du compteur (position logicielle) et la position machine (position matérielle). Pour ce faire, un point de référence est assigné à une position déterminée de la machine (position de référence) :

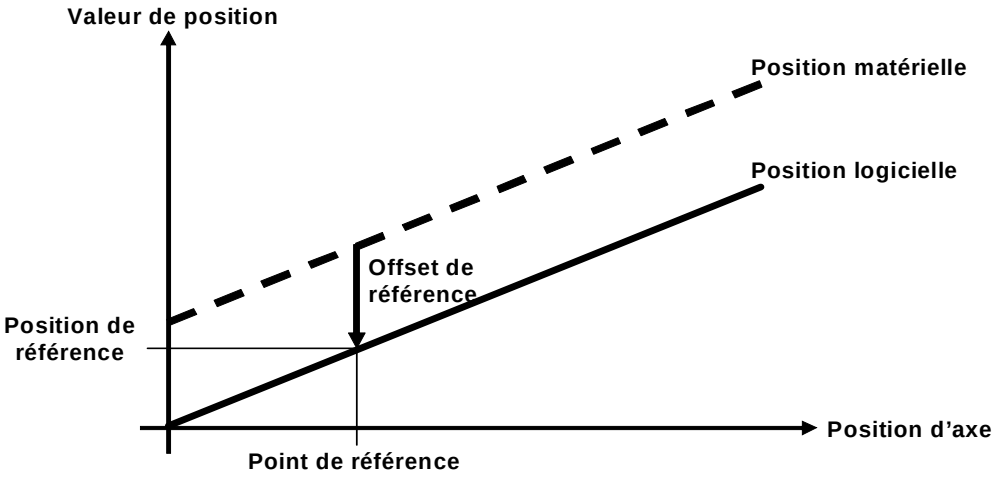
- Tout d'abord, on détermine l'offset de référence ; celui-ci correspond à la différence entre la position de référence et la position matérielle correspondant au point de référence spécifié :
  Offset de référence = Position de référence – Position matérielle$_{\text{Point de référence}}$

- Une fois la prise de référence terminée, l'addition de la position matérielle et de l'offset de référence permet d'obtenir la position logicielle :
  Position logicielle = Position matérielle + Offset de référence

- En effectuant ce calcul pour le point de référence, on obtient la position de référence exacte souhaitée :
  Pos. logicielle.$_{\text{point de réf.}}$ = Pos. matérielle $_{\text{point de réf.}}$ + Offset de référence

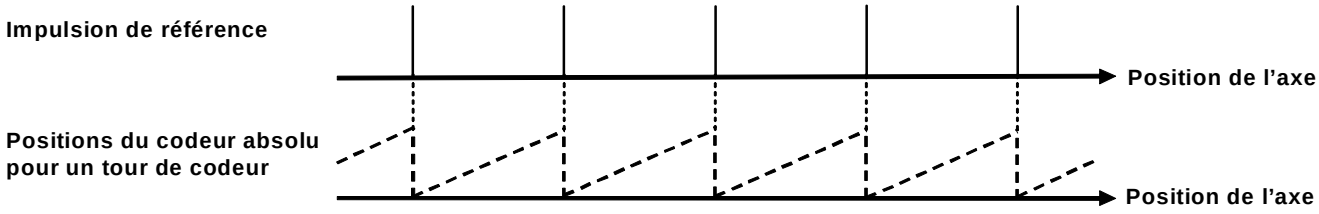
  Position de référence

Le diagramme ci-dessous illustre ce calcul :



Impulsion de référence

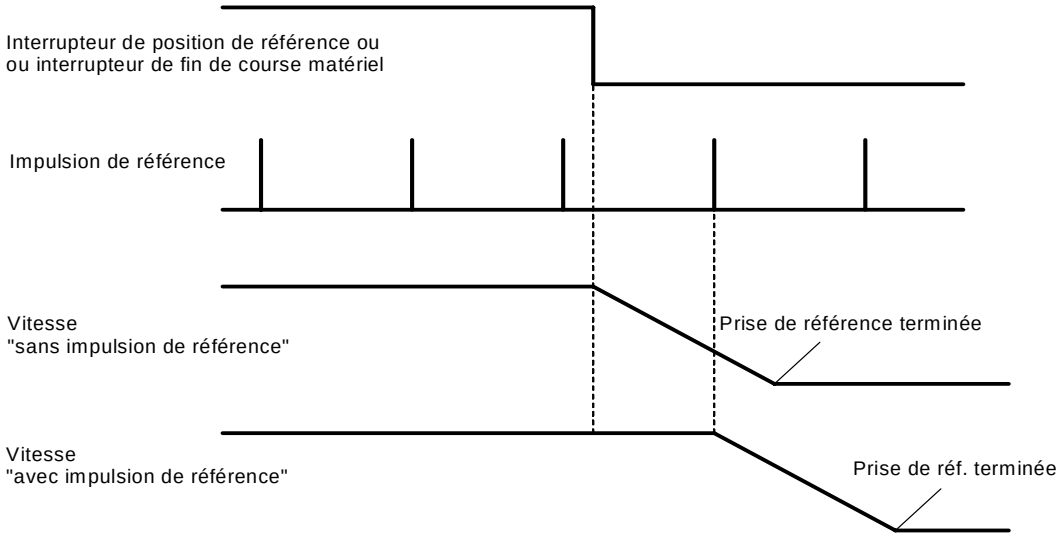
La marque zéro du codeur rotatif sert habituellement de point de référence. L'impulsion de référence (R) délivre une position angulaire reproductible à chaque tour de codeur. Les codeurs incrémentaux délivrent l'impulsion de référence sur une piste séparée. Dans les codeurs absolus, la position zéro parcourue à chaque tour de codeur est utilisée comme impulsion de référence :



Avec/Sans impulsion de référence pour les modes de prise de référence avec interrupteur de position de référence

"ref_pulse=ncOFF" La position de référence est enregistrée lorsque le front défini pour l'interrupteur de position de référence est atteint. Ensuite, le mouvement est stoppé.

"ref_pulse=ncON" La position de référence est enregistrée lorsque la première impulsion de référence apparaissant après le front défini pour l'interrupteur de position d'origine est atteinte. Ensuite, le mouvement est arrêté.

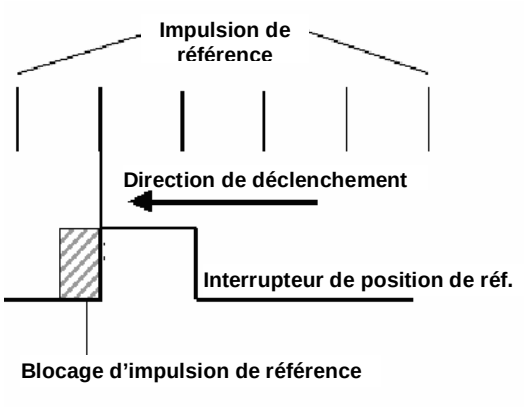


Distance d'impulsion de référence / Blocage d'impulsion de référence

La première impulsion de référence après le front de l'interrupteur de position de référence n'est pas toujours détectée si elle se produit dans la zone d'hystérésis de ce même interrupteur. Dans ce cas, c'est l'impulsion de référence suivante qui est prise en compte comme première impulsion (révolution suivante du codeur) :

Contrôler tout d'abord la **distance d'impulsion de référence** ("`homing.tr_s_rel`") avec "`homing.tr_s_block = 0.0`" (les valeurs "dangereuses" se situent autour de "`0.0`" ou "`1.0`")

Si nécessaire, mettre le paramètre de **blocage d'impulsion de référence** ("`homing.tr_s_block`") à "`0.5`" (ensuite, la détection de l'impulsion de référence est bloquée pendant un demi tour de codeur après que l'interrupteur de position de référence a été atteint)



Prise de référence directe

"**`mode = ncDIRECT`**"

"`ref_pulse=ncOFF`" La position de référence est immédiatement enregistrée (position actuelle = position de référence).

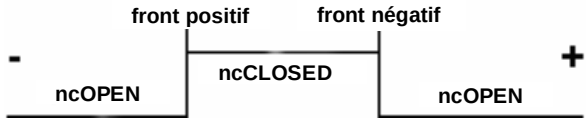
"`ref_pulse=ncON`" Un mouvement est lancé dans la direction de déclenchement. La position de référence est enregistrée lorsque la première impulsion de référence est atteinte. Ensuite, le mouvement est stoppé.

Important : Les paramètres "`edge_sw`" et "`start_dir`" sont sans effet car aucun interrupteur de position de référence n'est utilisé pour ce mode.

Prise de référence avec interrupteur de position de référence

"mode = ncSWITCH_GATE"

Definition: Le front de l'interrupteur de position de référence est positif lorsque l'état logique de cet interrupteur passe de "ncOPEN" à "ncCLOSED" lors d'un mouvement dans la direction positive

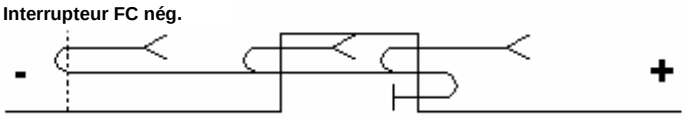


Dans ce mode de prise de référence, un mouvement est lancé avec "v_switch" pour rechercher l'interrupteur de position de référence. Après avoir atteint l'interrupteur de position de référence, l'axe se déplace vers le front qui a été défini pour cet interrupteur avec la vitesse définie dans "v_trigger".

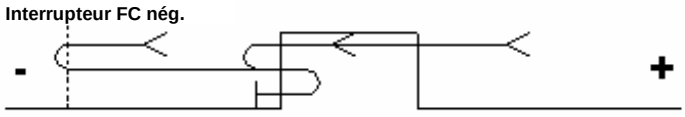
"ref_pulse=ncOFF" La position de référence est enregistrée lorsque l'on atteint le front défini pour l'interrupteur de position de référence. Ensuite, le mouvement est stoppé.

"ref_pulse=ncON" La position de référence est enregistrée lorsque l'on atteint la première impulsion de référence après le front défini pour l'interrupteur de position de référence. Ensuite, le mouvement est stoppé.

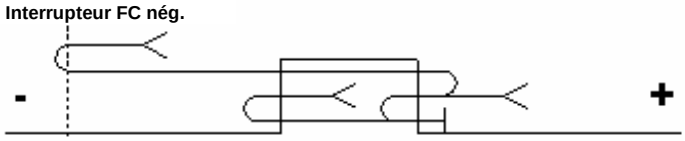
```
"start_dir = ncNEGATIVE"
"edge_sw = ncNEGATIVE"
"trigg_dir = ncNEGATIVE"
```



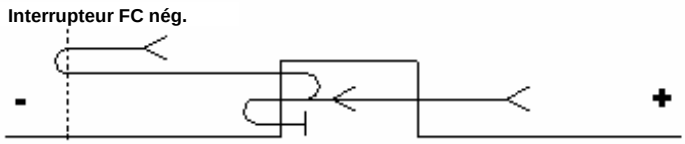

```
"start_dir = ncNEGATIVE"
"edge_sw = ncPOSITIVE"
"trigg_dir = ncNEGATIVE"
```



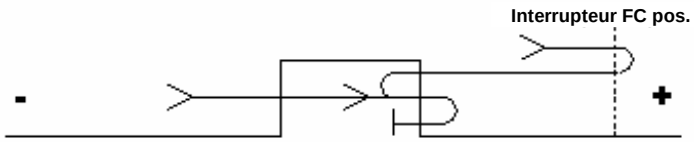

```
"start_dir = ncNEGATIVE"
"edge_sw = ncNEGATIVE"
"trigg_dir = ncPOSITIVE"
```



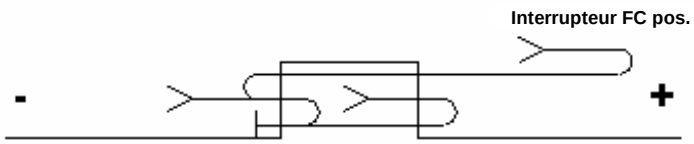

```
"start_dir = ncNEGATIVE"
"edge_sw = ncPOSITIVE"
"trigg_dir = ncPOSITIVE"
```



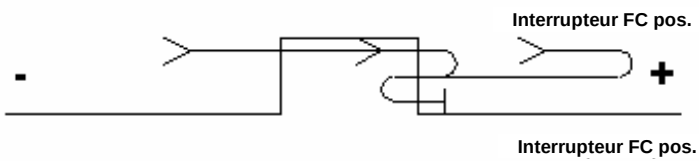

```
"start_dir = ncPOSITIVE"
"edge_sw = ncNEGATIVE"
"trigg_dir = ncNEGATIVE"
```



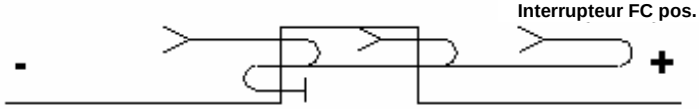

```
"start_dir = ncPOSITIVE"
"edge_sw = ncPOSITIVE"
"trigg_dir = ncNEGATIVE"
```




```
"start_dir = ncPOSITIVE"
"edge_sw = ncNEGATIVE"
"trigg_dir = ncPOSITIVE"
```



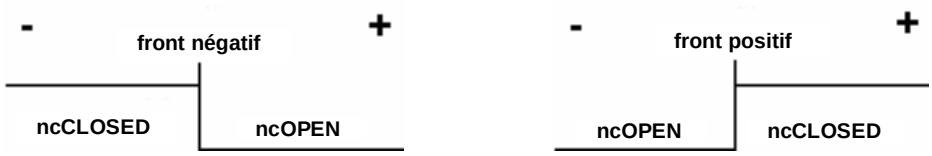

```
"start_dir = ncPOSITIVE"
"edge_sw = ncPOSITIVE"
"trigg_dir = ncPOSITIVE"
```



Prise de référence avec interrupteur de position de référence absolue

"mode = ncABS_SWITCH"

Definition: Le front de l'interrupteur de position de référence est positif lorsque l'état logique de l'interrupteur de position de référence passe de "ncOPEN" à "ncCLOSED" :



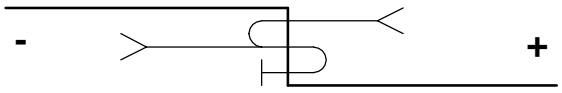
Dans ce mode de prise de référence, un mouvement est lancé avec "v_switch" pour rechercher l'interrupteur de position de référence. Après avoir atteint l'interrupteur de position de référence, l'axe se déplace vers le front qui a été défini pour l'interrupteur de position de référence avec la vitesse définie dans "v_trigger".

"ref_pulse=ncOFF" La position de référence est enregistrée lorsque l'on atteint le front défini pour l'interrupteur de position de référence. Ensuite, le mouvement est stoppé.

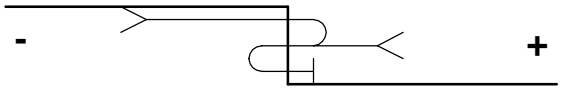
"ref_pulse=ncON" La position de référence est enregistrée lorsque l'on atteint la première impulsion de référence après le front défini pour l'interrupteur de position de référence. Ensuite, le mouvement est stoppé.

Important : La direction de démarrage dépend de l'état de l'interrupteur de position de référence. Le paramètre "start_dir" est donc ignoré.

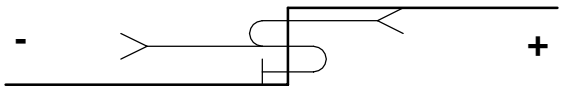
"edge_sw = ncNEGATIVE"
"trigg_dir = ncNEGATIVE"



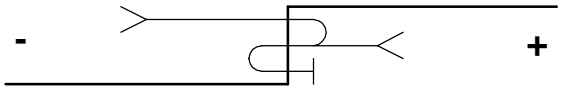
"edge_sw = ncNEGATIVE"
"trigg_dir = ncPOSITIVE"



"edge_sw = ncPOSITIVE"
"trigg_dir = ncNEGATIVE"



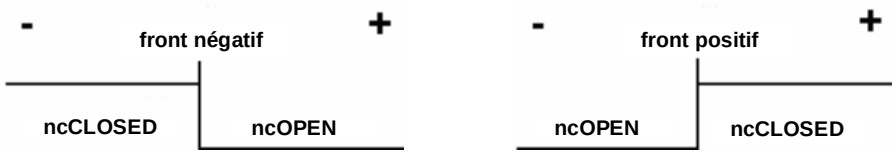
"edge_sw = ncPOSITIVE"
"trigg_dir = ncPOSITIVE"



Prise de référence avec interrupteur de fin de course

"mode = ncEND_SWITCH"

Definition: Le front de l'interrupteur de référence est positif lorsque l'état logique de l'interrupteur de fin de course passe de "ncOPEN" à "ncCLOSED" lors d'un mouvement dans la direction positive (l'interrupteur de fin de course utilisé est défini en fonction de ce front) :



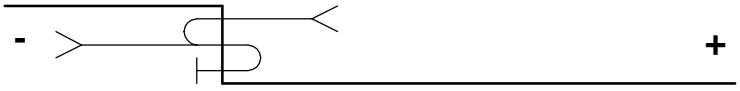
Dans ce mode de prise de référence, un mouvement est lancé avec "v_switch" pour rechercher l'interrupteur matériel de fin de course. Après avoir atteint l'interrupteur de fin de course, l'axe se déplace vers le front qui a été défini pour l'interrupteur de position de référence avec la vitesse définie dans "v_trigger".

"ref_pulse=ncOFF" La position de référence est enregistrée lorsque l'on atteint le front de l'interupteur de position de référence. Ensuite, le mouvement est stoppé.

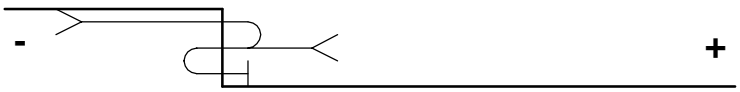
"ref_pulse=ncON" La position de référence est enregistrée lorsque l'on atteint la première impulsion de référence après le front défini pour l'interrupteur de position de référence. Ensuite, le mouvement est stoppé.

La direction de départ dépend de l'état de l'interrupteur de fin de course. Le paramètre "start_dir" est donc ignoré !

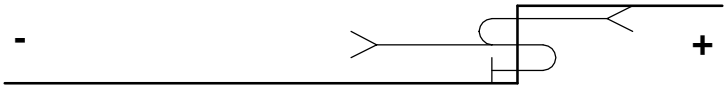
"edge_sw = ncNEGATIVE"
"trigg_dir = ncNEGATIVE"
*"**ref_pulse = ncON**" non autorisé !*



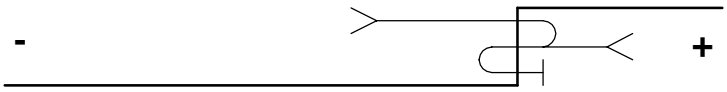
"edge_sw = ncNEGATIVE"
"trigg_dir = ncPOSITIVE"



"edge_sw = ncPOSITIVE"
"trigg_dir = ncNEGATIVE"



"edge_sw = ncPOSITIVE"
"trigg_dir = ncPOSITIVE"
*"**ref_pulse = ncON**" non autorisé !*

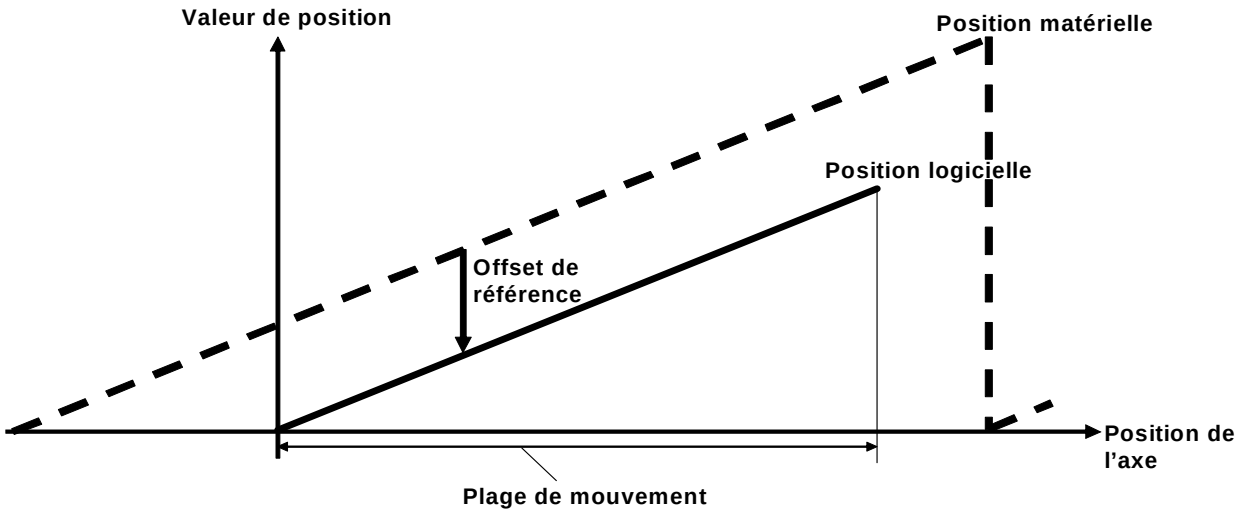


Prise de référence par définition de l'offset de référence

"mode = ncHOME_OFFSET"

Dans ce mode de prise de référence, l'offset de référence n'est pas délivré par le système d'exploitation ACP10 mais par "move.homing.parameter.s". Tous les autres paramètres de prise de référence sont ignorés.

Ce mode est particulièrement bien adapté au codeur absolu pour lequel chaque valeur de position est unique sur toute la plage de mouvement. Grâce à l'offset de référence, la position machine correcte (position logicielle) peut ici être calculée à partir de la position matérielle indiquée par le codeur.



On peut déterminer l'offset de référence en effectuant une seule fois un mouvement d'étalonnage (prise de référence avec interrupteur de référence, par exemple).

Prise de référence par définition d'un offset de référence avec correction de la plage de comptage

"modus = ncREF_OFFSET + ncKORREKTUR"

Dans ce mode de prise de référence, on vérifie en plus, après définition de l'offset de référence, si la position logicielle est comprise dans la plage de mouvement définie par les fins de course logiciels. Si ce n'est pas le cas, l'offset de référence est corrigé :

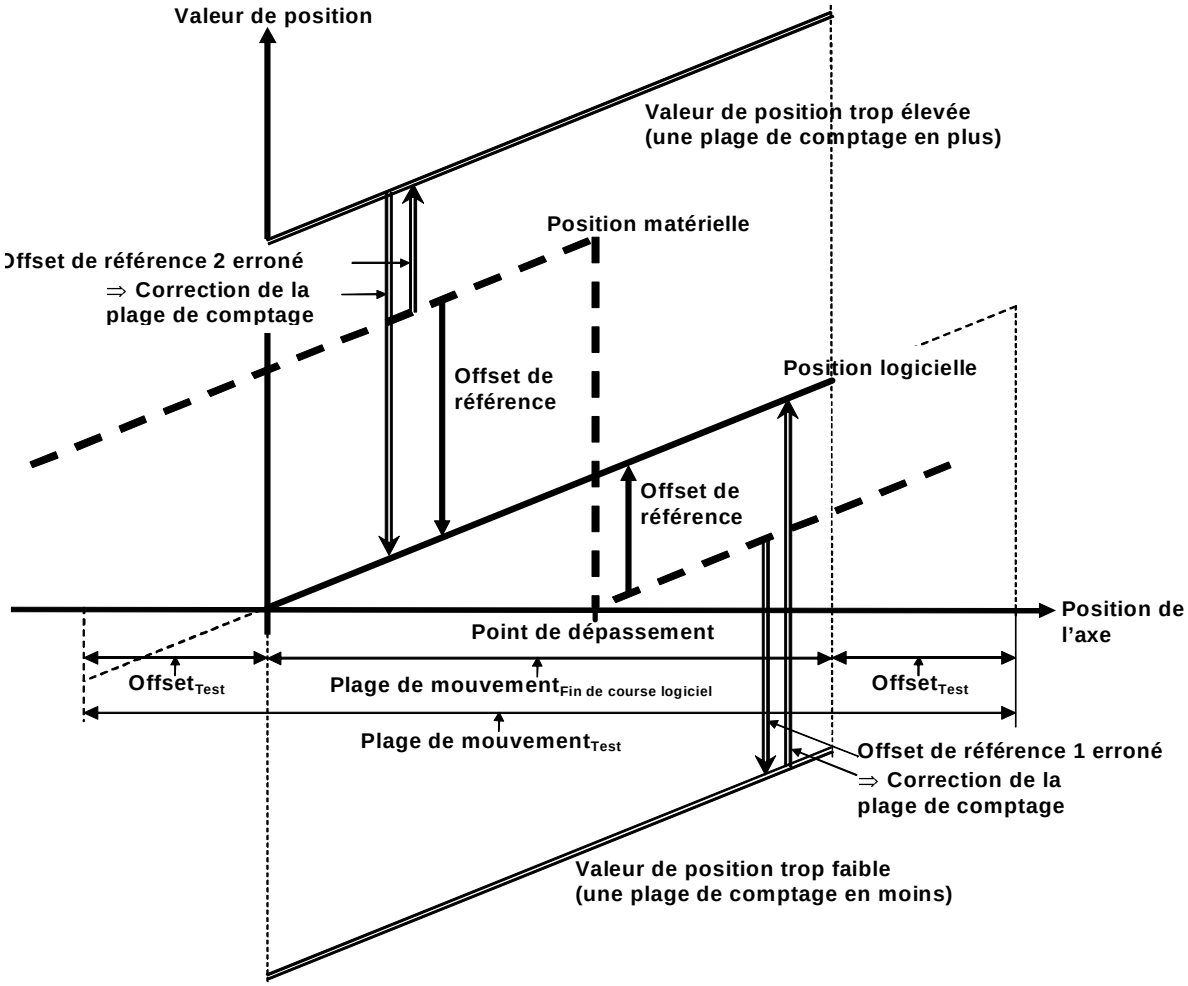
Position logicielle < Plage de mouvement$_{Test}$ Minimum
⇒ Offset de réf.$_{corrigé}$ = Offset de réf.$_{défini}$ + Plage de comptage du codeur

Position logicielle > Plage de mouvement$_{Test}$ Maximum
⇒ Offset de réf.$_{corrigé}$ = Offset de réf.$_{défini}$ + Plage de comptage du codeur

Les plages de déplacement minimum et maximum pour le test ci-dessus sont calculées à l'activation du régulateur, en centrant la plage de comptage du codeur sur la plage de mouvement définie par les fins de courses logiciels.

$Offset_{Test}$ = (Plage de comptage du codeur – Plage de $mouvement_{FC\ logiciels}$) / 2
Plage de $mouvement_{Test}$-Minimum = Fin de course logiciel négatif - $Offset_{Test}$
Plage de $mouvement_{Test}$-Maximum = Fin de course logiciel positif + $Offset_{Test}$

Dans le cas où l'on utilise des codeurs absolus, une correction de la plage de comptage est requise si le codeur délivre une valeur de position unique sur toute la plage de mouvement et si un dépassement de la plage codeur se produit dans la plage de mouvement. L'offset de référence varie selon que la machine a été mise en marche à une position à droite ou à gauche du point de dépassement



A droite du point de dépassement, l'offset de référence 1 valable pour le côté gauche conduirait à une valeur de position incorrecte. De même, à gauche du point de dépassement, l'offset de référence 2 valable pour le côté droit conduirait à une valeur de position incorrecte. Ceci peut être compensé par la correction de plage de comptage.

Important : La correction de la plage de comptage n'est possible que si la plage de comptage du codeur est plus grande que la plage de mouvement !

2.1.13.2 Actions NC

Sujet	Action	Description
ncHOMING	ncINIT	Initialiser la prise de référence
ncHOMING	ncSTART	Lancer la prise de référence

2.1.13.2.1 Initialisation de la prise de référence

Appel de fonction : `status = ncaction(ax_obj,ncHOMING,ncINIT)`

Paramètre :
"move.homing.parameter.v_switch"
"move.homing.parameter.v_trigger"
"move.homing.parameter.a"
"move.homing.parameter.tr_s_block"

Condition(s) :
"network.init == ncTRUE"
"move.mode != ncHOMING"

Résultat(s) : Initialisation des paramètres de prise de référence dans le variateur

Indication(s) d'état : "move.homing.init = ncFALSE"
après "**ncOK**" en tant qu'état résultant de l'action NC

"move.homing.init = ncTRUE"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case REF_INIT:
    p_ax_dat->move.homing.parameter.v_switch = ...;
    p_ax_dat->move.homing.parameter.v_trigger  = ...;
    p_ax_dat->move.homing.parameter.a          = ...;
    p_ax_dat->move.homing.parameter.tr_s_block = ...;

    action_status = ncaction(ax_obj,ncHOMING,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_REF_INIT;
    }
    break;

  case W_HOMING_INIT:
    if (p_ax_dat->move.homing.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.13.2.2 Lancement de la prise de référence

Appel de fonction : `status = ncaction(ax_obj,ncHOMING,ncSTART)`

Paramètre :
"move.homing.parameter.s"
"move.homing.parameter.mode"
"move.homing.parameter.edge_sw"
"move.homing.parameter.start_dir"
"move.homing.parameter.trig_dir"
"move.homing.parameter.ref_pulse"

Condition(s) :
"controller.status == ncON"
"move.mode == ncOFF"
"move.homing.init == ncTRUE"

Résultat(s) : Lancement de la prise de référence avec les paramètres initialisés précédemment

Indication(s) d'état : "move.mode = ncHOMING"
"move.homing.status.ok = ncFALSE"
après "**ncOK**" en tant qu'état résultant de l'action NC

"move.mode = ncOFF"
après arrêt réussi du mouvement ou
après exécution réussie de la prise de référence

"move.homing.status.ok = ncTRUE"
Distance d'impulsion de référence dans "move.homing.status.tr_s_rel"
Offset de référence dans "move.homing.status.offset"
après exécution réussie de la prise de référence

L'action NC "ncHOMING,ncSTART" peut aussi être appelée avec "controller.status==ncOFF" si le mode spécifié pour la prise de référence ne nécessite aucun mouvement (comme c'est le cas par exemple pour "mode=ncDIRECT" avec "refpulse=ncOFF" ou "mode=ncREF_OFFSET"). Bien qu'aucun mouvement ne soit effectué dans ce cas, l'état "move.mode" est modifié par le gestionnaire NC comme cela est décrit ci-dessus.

Exemple :

```
switch (step)
{
...
  case REF_START:
    p_ax_dat->move.homing.parameter.s          = ...;
    p_ax_dat->move.homing.parameter.mode       = ...;
    p_ax_dat->move.homing.parameter.edge_sw    = ...;
    p_ax_dat->move.homing.parameter.start_dir  = ...;
    p_ax_dat->move.homing.parameter.trig_dir   = ...;
    p_ax_dat->move.homing.parameter.ref_pulse  = ...;

    action_status = ncaction(ax_obj,ncHOMING,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_REF_OK;
    }
    break;

  case W_REF_OK:
    if (p_ax_dat->move.homing.status.ok == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.14 Mouvements de base

2.1.14.1 Structure de données

move		**Mouvement**
...		
basis		**Mouvements de base**
init	USINT	Mouvements de base initialisés : ncFALSE/ncTRUE
status		**Status**
in_pos	USINT	"In Position" (position cible atteinte) : ncFALSE/ncTRUE
override		**Override**
v	INT	Consigne de vitesse en % [0.01%] (paramétrage par défaut : 10000) Valeurs valides : de 0 à 20000 (0% à 200%)
a	INT	Consigne d'accélération [0.01%] (paramétrage : 10000) Valeurs valides : de 1 à 20000 (0,01% à 200%)
parameter		**Paramètre**
s	DINT	Position cible ou distance de déplacement relative [unité]
v_pos	REAL	Vitesse dans la direction positive [unité/s]
v_neg	REAL	Vitesse dans la direction négative [unité/s]
a1_pos	REAL	Accélération dans la direction positive [unité/s^2]
a2_pos	REAL	Décélération dans la direction positive [unité/s^2]
a1_neg	REAL	Accélération dans la direction négative [unité/s^2]
a2_neg	REAL	Décélération dans direction négative [unité/s^2]
trg_stop		**Mode "Arrêt après Déclenchement"**
init	USINT	Mode initialisé : ncFALSE/ncTRUE
event	USINT	Evénement de déclenchement : ncOFF: Mode "Arrêt après déclenchement" désactivé ncTRIGGER1+ncP_EDGE: Front positif à l'entrée digitale "Trigger1" ncTRIGGER2+ncP_EDGE: Front positif à l'entrée digitale "Trigger2" ncTRIGGER1+ncN_EDGE: Front négatif à l'entrée digitale "Trigger1" ncTRIGGER2+ncN_EDGE: Front négatif à l'entrée digitale "Trigger2"
s_rest	DINT	Distance restante après déclenchement [unité]

Override (consigne de vitesse ou d'accélération en %)

La consigne de vitesse Override permet de changer la **vitesse** (avec "`move.basis.override.v`") et l'**accélération** (avec "`move.basis.override.a`") à tout moment dans une plage allant de 0% à 200% de la valeur maximale spécifiée.

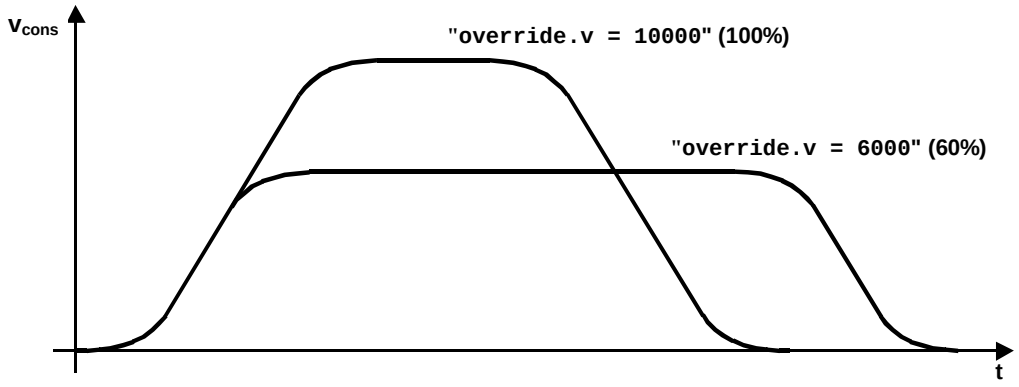
Important : Aucune action NC n'est nécessaire pour transférer l'Override au variateur.

Tout changement de valeur pour les deux composantes Override est contrôlé dans la tâche NC cyclique. En cas de changement, le transfert des deux composantes au variateur est automatiquement activé. Néanmoins, le télégramme CAN correspondant n'est transféré au variateur que lorsque le traitement l'action NC standard appelée précédemment est terminé.

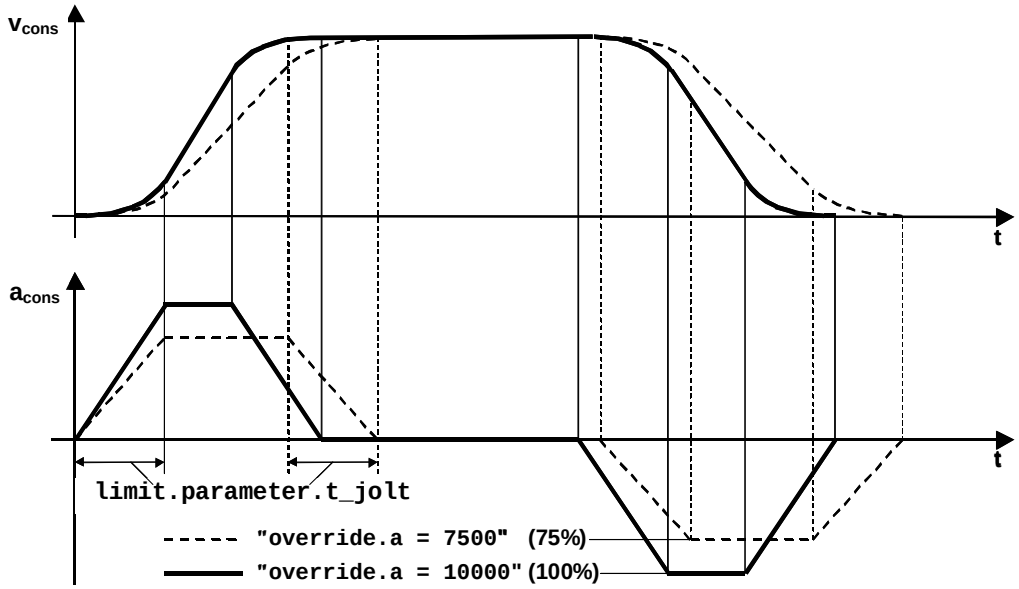
Des changements d'override trop fréquents dans la tâche d'application peuvent fortement perturber le transfert des télégrammes CAN associés à des actions NC standard. En effet, un seul télégramme CAN de ce type est transféré par variateur et par cycle de tâche NC.

Le changement de vitesse ou d'accélération dû à un changement des valeurs d'override correspondantes, pour un mouvement de base actif, ne se fait pas immédiatement.
Si "`move.basis.override.v`" est modifié, le changement de vitesse s'opère, pour les mouvements de base, avec les valeurs d'accélération ("`a1_pos/a1_neg`") ou de décélération ("`a2_pos/a2_neg`") spécifiées.
Si "`move.basis.override.a`" est modifié, la rampe d'accélération ou de décélération est modifiée en fonction du temps de vibration spécifié ("`limit.parameter.t_jolt`").

Comme le montre le diagramme suivant, les profils de mouvement varient en fonction de "`move.basis.override.v`" :

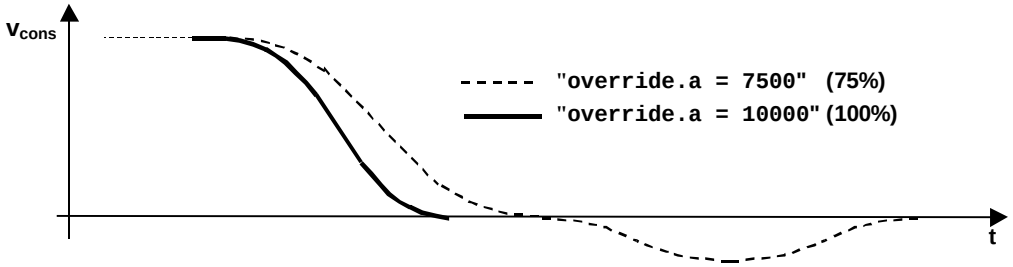


Comme le montre le diagramme suivant, les profils de mouvement varient en fonction de "`move.basis.override.a`" :



Le temps de montée de l'accélération est le même dans les deux cas ("`limit.parameter.t_jolt`") car la limitation de vibration est effectuée avec un filtre linéaire.

Important : Lorsque l'on diminue "`override.a`" dans la plage de valeurs "`s = v/2*(v/(a*override.a`$_{nouveau}$`)+t_jolt)`" (nouvelle distance de freinage après modification de "`override.a`") avant que la position cible soit atteinte, le mouvement se poursuit au-delà de la position cible, puis un retour vers cette position cible est effectué :

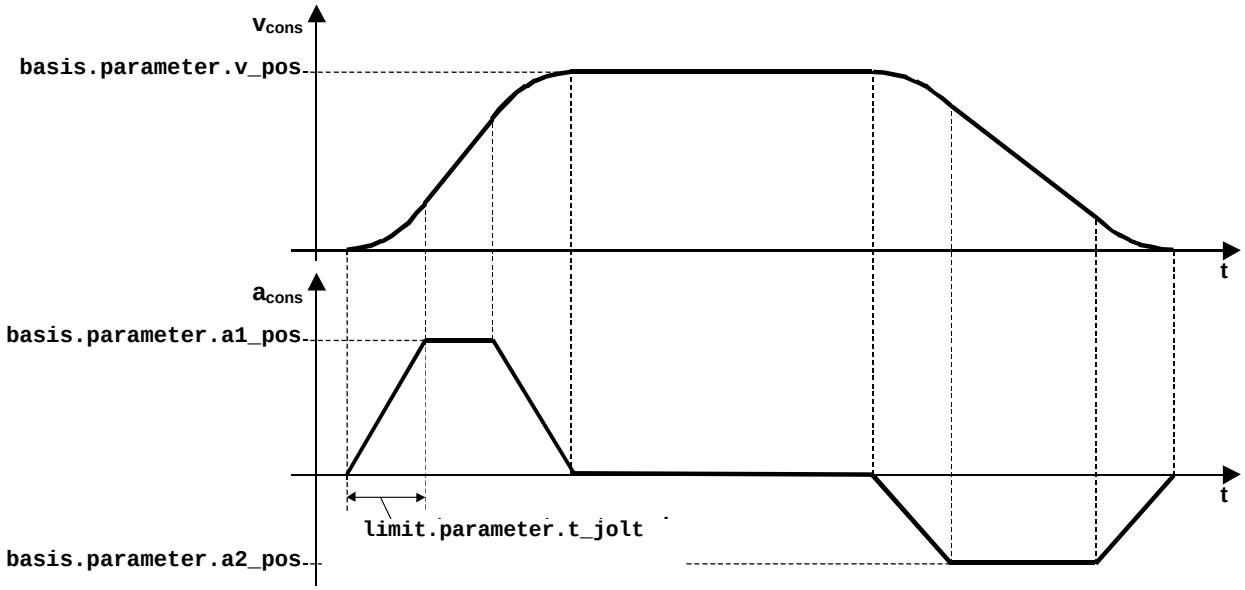


:

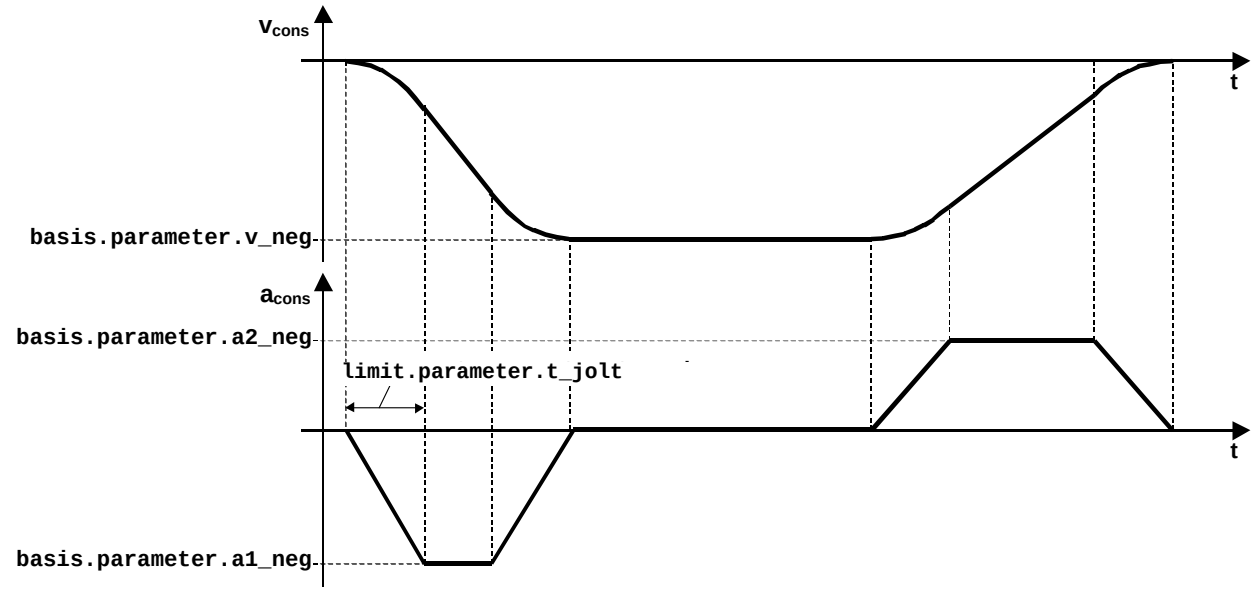
Profil de mouvement

Pour le profil de mouvement, la valeur de la vitesse ("**v_pos**/**v_neg**"), de l'accélération ("**a1_pos**/**a1_neg**") et de la décélération ("**a2_pos**/**a2_neg**") peut être initialisée dans "move.basis.parameter", et ce pour les mouvements dans les directions **pos**itive et **nég**ative.

Le diagramme suivant met en évidence le mode d'action des différents paramètres pour un mouvement dans la direction positive :



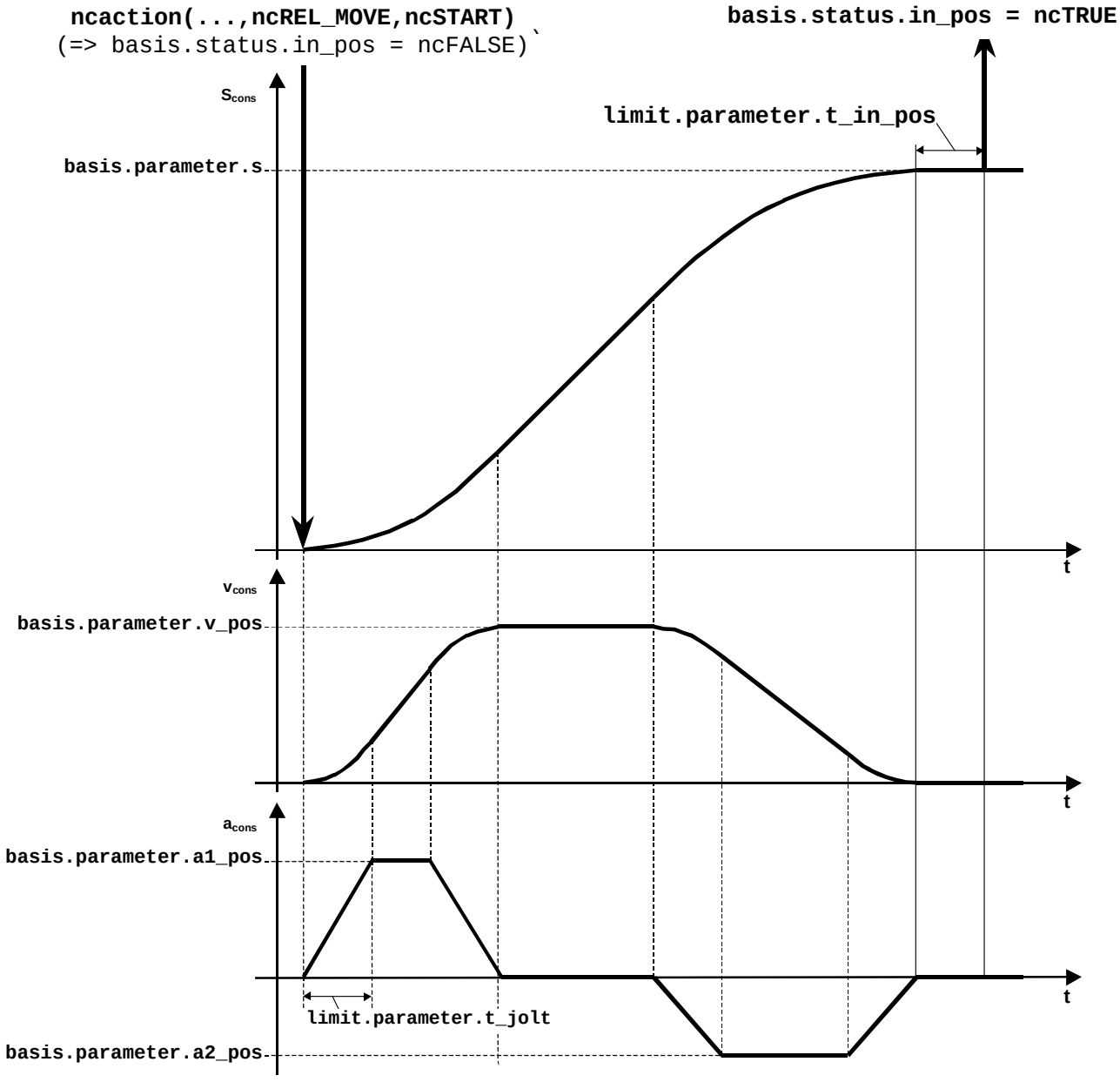
Le diagramme suivant met en évidence le mode d'action des différents paramètres pour un mouvement dans la direction négative :



Important : Les paramètres initialisés avec l'action NC "ncBASIS_MOVE,ncINIT" dans le variateur pour des mouvements de base sont aussi modifiés en partie par des actions NC associées aux mouvements de base (par exemple, "v_pos" pour "ncPOS_MOVE,ncSTART"). Attention donc aux paramètres spécifiés dans les actions NC.

Etat pour mouvements de base avec position cible

Outre l'état de mouvement général dans "`move.mode`" et "`move.detail`", l'état "`move.basis.status.in_pos`" est aussi significatif pour les mouvements de base conduisant à une position cible définie (par exemple "`ncABS_MOVE`" et "`ncREL_MOVE`"). Lorsqu'un mouvement de ce type a été **effectué correctement**, alors "**`move.basis.status.in_pos==ncTRUE`**". Ceci est illustré dans le diagramme qui suit pour un positionnement relatif dans la direction positive :

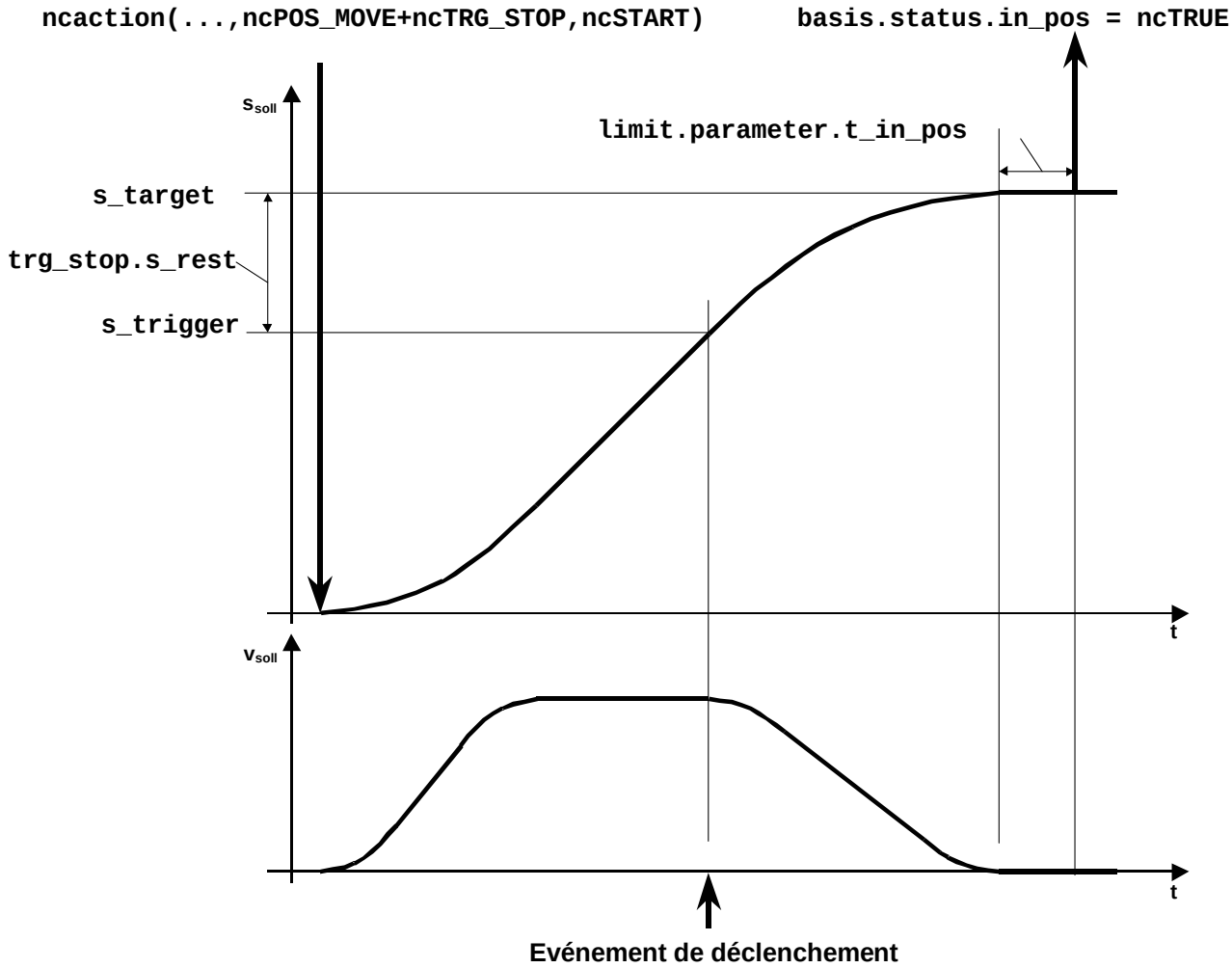


Important : L'état "`move.basis.status.in_pos==ncTRUE`" ne dépend pas de la position réelle, mais seulement du moment où la génération de valeur de consigne pour le positionnement lancé en dernier se termine et du temps d'attente défini avec "`limit.parameter.t_in_pos`".

.

Mode "Arrêt après Déclenchement"

Le mode "**Arrêt après Déclenchement**" peut être activé pour chaque mouvement de base lorsque "**ncTRG_STOP**" est ajouté au code du sujet de l'action NC (par exemple, "ncPOS_MOVE+ncTRG_STOP,ncSTART" pour le démarrage d'un mouvement dans la direction positive). Si ce mode est activé pour un mouvement de base, la position réelle "s_trigger" est flashée (latch) à l'arrivée de l'événement de déclenchement défini "trg_stop.event". Ensuite, avec la position réelle ainsi obtenue et la distance restante définie "trg_stop.s_rest", une position cible "s_target" est calculée ; le mouvement est ensuite stoppé à cette position cible. Ce processus est illustré dans la figure suivante :



Important : Si l'événement de déclenchement paramétré se produit à la position actuelle "s_trigger", la position cible "s_target" est calculée comme suit, quelle que soit la direction du mouvement :

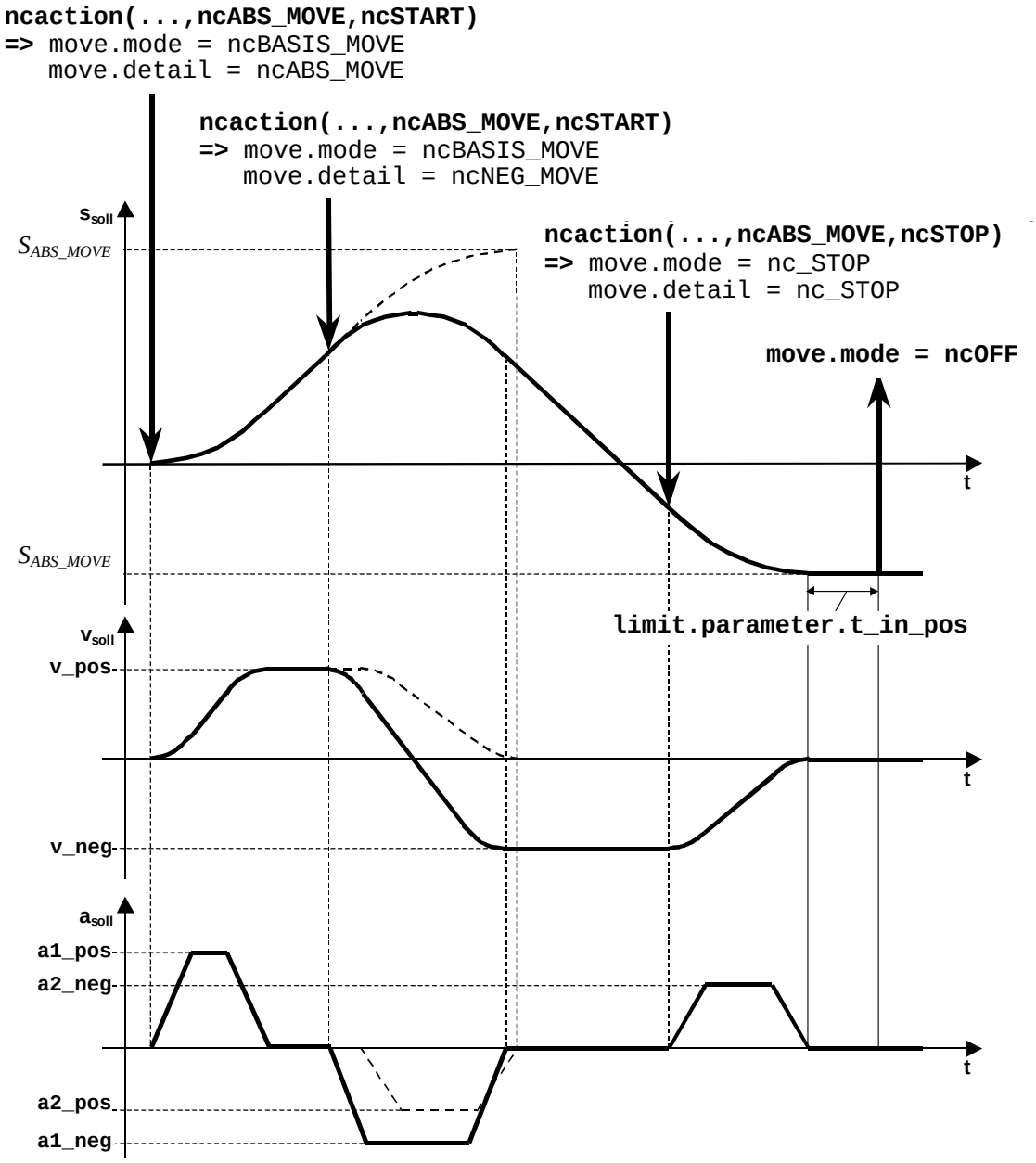
```
s_target = s_trigger + trg_stop.s_rest
```

L'arrivée à la position cible peut également entraîner une inversion de la direction du mouvement.

2.1.14.2 Actions NC

Sujet	**Action**	**Description**
ncBASIS_MOVE	ncINIT	Initialiser des mouvements de base
ncBASIS_MOVE	ncHALT	Arrêter un mouvement de base (avec les paramètres actuels)
ncPOS_MOVE	ncSTART	Démarrer un mouvement dans la direction positive
ncNEG_MOVE	ncSTART	Démarrer un mouvement dans la direction négative
ncABS_MOVE	ncSTART	Démarrer un mouvement avec position cible absolue
ncREL_MOVE	ncSTART	Démarrer un mouvement avec distance de déplacement relatif
ncBASIS_TRG_STOP	ncINIT	Initialiser le mode "Arrêt après Déclenchement" pour les mouvements de base
ncPOS_MOVE **+** ncTRG_STOP	ncSTART	Démarrer un mouvement dans la direction positive ***en*** mode "Arrêt après Déclenchement"
ncNEG_MOVE **+** ncTRG_STOP	ncSTART	Démarrer un mouvement dans la direction négative ***en*** mode "Arrêt après Déclenchement"
ncABS_MOVE **+** ncTRG_STOP	ncSTART	Démarrer un mouvement avec position cible absolue ***en*** mode "Arrêt après Déclenchement"
ncABS_MOVE **+** ncTRG_STOP **+** ncS_REST	ncSTART	Démarrer un mouvement avec position cible absolue ***en*** mode "Arrêt après Déclenchement" ***avec*** variante "+ s_rest"
ncREL_MOVE **+** ncTRG_STOP	ncSTART	Démarrer un mouvement avec distance de déplacement relatif ***en*** mode "Arrêt après Déclenchement"
ncREL_MOVE **+** ncTRG_STOP **+** ncS_REST	ncSTART	Démarrer un mouvement avec distance de déplacement relatif ***en*** mode "Arrêt après Déclenchement" ***avec*** variante "+ s_rest"

Important : Le démarrage d'un mouvement de base est également autorisé pendant qu'un autre mouvement de base est actif. La figure suivante montre un exemple avec les indications d'état correspondantes :



2.1.14.2.1 Initialisation des mouvements de base

Appel de fonction : `status = ncaction(ax_obj,ncBASIS_MOVE,ncINIT)`

Paramètre : "`move.basis.parameter`"

Condition(s) : "`network.init == ncTRUE`"

Résultat(s) : Initialisation des mouvements de base dans le variateur

Indication d'état(s) : "`move.basis.init = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.basis.init = ncTRUE`"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case BASIS_MOVE_INIT:
    p_ax_dat->move.basis.parameter.v_pos  = ...;
    p_ax_dat->move.basis.parameter.v_neg  = ...;
    p_ax_dat->move.basis.parameter.a1_pos = ...;
    p_ax_dat->move.basis.parameter.a2_pos = ...;
    p_ax_dat->move.basis.parameter.a1_neg = ...;
    p_ax_dat->move.basis.parameter.a2_neg = ...;

    action_status = ncaction(ax_obj,ncBASIS_MOVE,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_BASIS_MOVE_INIT;
    }
    break;

  case W_BASIS_MOVE_INIT:
    if (p_ax_dat->move.basis.init == ncTRUE)
    /* Opération terminée avec succès */
    {
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Si cette action NC est appelée alors qu'un mouvement de base est actif, les paramètres ne sont alors appliqués qu'au prochain démarrage d'un mouvement de base.

Les paramètres initialisés avec cette action NC dans le variateur pour des mouvements de base sont partiellement modifiés par d'autre actions NC associées à des mouvements de base (par exemple "`v_pos`" pour "`ncPOS_MOVE,ncSTART`"). Attention donc aux paramètres spécifiés dans les actions NC.

2.1.14.2.2 Arrêt d'un mouvement de base avec les paramètres en cours

Appel de fonction : `status = ncaction(ax_obj,ncBASIS_MOVE,ncHALT)`

Paramètre : -

Condition(s) : "`regler.status == ncON`"
"`move.mode == ncBASIS_MOVE`" ou "`move.mode == ncOFF`"

Résultat(s) : Freinage d'un mouvement de base avec décélération paramétrée pour ce mouvement et prise en compte du temps de vibration paramétré

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncHALT`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement

Exemple :

```
switch (step)
{
...
  case HALT:
    action_status = ncaction(ax_obj,ncBASIS_MOVE,ncHALT);
    if ( action_status == ncOK )
    {
      step = W_HALT;
    }
    break;

  case W_HALT:
    if (p_ax_dat->move.mode == ncOFF)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Contrairement à "`ncMOVE,ncSTOP`", le démarrage d'un autre mouvement de base dans la rampe de freinage est autorisé après "`ncBASIS_MOVE,ncHALT`" .

"`ncBASIS_MOVE,ncHALT`" est une action NC à priorité plus haute que les actions NC standard. Cela signifie qu'en cas d'appel de cette action NC, le télégramme de commande CAN correspondant est immédiatement envoyé au variateur, même si le traitement d'une action NC standard appelée précédemment n'est pas encore terminé.

2.1.14.2.3 Démarrage d'un mouvement dans la direction positive

Appel de fonction : `status = ncaction(ax_obj,ncPOS_MOVE,ncSTART)`

Paramètre : "`move.basis.parameter.v_pos`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`"
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement dans la direction positive

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncPOS_MOVE`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après freinage du mouvement jusqu'à "`move.basis.parameter.v_pos = 0`"

Exemple :

```
switch (step)
{
...
  case POS_MOVE_START:
    p_ax_dat->move.basis.parameter.v_pos = ...;
    action_status = ncaction(ax_obj,ncPOS_MOVE,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_STOP;
    }
    break;

  case W_MOVE_STOP:
    if (move_stop)
    {
      p_ax_dat->move.stop.index.command = 1;
      action_status = ncaction(ax_obj,ncMOVE,ncSTOP);
      if ( action_status == ncOK )
      {
        move_stop = 0;
        step = W_STOP;
      }
    }
    break;

  case W_STOP:
    if (p_ax_dat->move.mode == ncOFF)
    {
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Le paramètre "`move.basis.parameter.v_pos`" initialisé précédemment dans le variateur avec l'action NC "`ncBASIS_MOVE,ncINIT`" est réinitialisé avec cette action NC.

2.1.14.2.4 Démarrage du mouvement dans la direction négative

Appel de fonction : `status = ncaction(ax_obj,ncNEG_MOVE,ncSTART)`

Paramètre : "`move.basis.parameter.v_neg`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.neg_hw_end == ncOPEN`"
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement dans la direction négative

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncNEG_MOVE`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.modus = ncOFF`"
après arrêt réussi du mouvement ou
après freinage du mouvement jusqu'au moment où "`move.basis.parameter.v_pos = 0`"

Exemple :

```
switch (step)
{
...
  case NEG_MOVE_START:
    p_ax_dat->move.basis.parameter.v_neg = ...;
    action_status = ncaction(ax_obj,ncNEG_MOVE,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_STOP;
    }
    break;

  case W_MOVE_STOP:
    if (move_stop)
    {
      p_ax_dat->move.stop.index.command = 1;
      action_status = ncaction(ax_obj,ncMOVE,ncSTOP);
      if ( action_status == ncOK )
      {
        move_stop = 0;
        step = W_STOP;
      }
    }
    break;

  case W_STOP:
    if (p_ax_dat->move.mode == ncOFF)
    {
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Le paramètre "`move.basis.parameter.v_neg`" initialisé précédemment dans le variateur avec l'action NC "`ncBASIS_MOVE,ncINIT`" est réinitialisé avec cette action NC.

2.1.14.2.5 Démarrage d'un mouvement avec position cible absolue

Appel de fonction : `status = ncaction(ax_obj,ncABS_MOVE,ncSTART)`

Paramètre : "`move.basis.parameter.s`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`" (pour un mouvement dans la direction pos.)
"`dig_e.status.neg_hw_end == ncOPEN`" (pour un mouvement dans la direction nég.)
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement avec position cible absolue

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncABS_MOVE`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case ABS_MOVE_START:
    p_ax_dat->move.basis.parameter.s = ...;
    action_status = ncaction(ax_obj,ncABS_MOVE,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

2.1.14.2.6 Démarrage d'un mouvement avec distance de déplacement relatif

Appel de fonction : `status = ncaction(ax_obj,ncREL_MOVE,ncSTART)`

Paramètre : "`move.basis.parameter.s`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`" (mouvement dans la direction positive)
"`dig_e.status.neg_hw_end == ncOPEN`" (mouvement dans la direction négative)
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement avec distance de déplacement relatif

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncREL_MOVE`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case REL_MOVE_START:
    p_ax_dat->move.basis.parameter.s = ...;
    action_status = ncaction(ax_obj,ncREL_MOVE,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Si à l'appel de cette action NC un mouvement avec position cible définie "`s_target`$_{old}$" est actif (par exemple "`ncABS_MOVE`" ou "`ncREL_MOVE`"), la nouvelle position cible "`s_target`$_{new}$" se calcule de la façon suivante :

$$s\_target_{new} = s\_target_{old} + move.basis.parameter.s$$

Dans tous les autres cas, la nouvelle position cible est calculée avec la position de consigne "`s_cons`" actuelle du régulateur de position :

$$s\_target_{new} = s\_target + move.basis.parameter.s$$

2.1.14.2.7 Initialisation du mode "Arrêt après Déclenchement" pour des mouvements de base

Appel de fonction : `status = ncaction(ax_obj,ncBASIS_TRG_STOP,ncINIT)`

Paramètre : "`move.basis.trg_stop`"

Condition(s) : "`network.init == ncTRUE`"

Résultat(s) : Initialisation du mode"Arrêt après Déclenchement" dans le variateur pour des mouvements de base

Indication(s) d'état : "`move.basis.trg_stop.init = ncFALSE`"
après "**ncOK**" en tant qu'état résultant de l'action NC

"`move.basis.trg_stop.init = ncTRUE`"
après initialisation réussie

Exemple :

```
switch (step)
{
...
  case BASIS_TRG_STOP_INIT:
    p_ax_dat->move.basis.trg_stop.event  = ...;
    p_ax_dat->move.basis.trg_stop.s_rest    = ...;
    action_status = ncaction(ax_obj,ncBASIS_TRG_STOP,ncINIT);
    if ( action_status == ncOK )
    {
      step = W_BASIS_TRG_STOP_INIT;
    }
    break;

  case W_BASIS_TRG_STOP_INIT:
    if (p_ax_dat->move.basis.trg_stop.init == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Si cette action NC est appelée alors qu'un mouvement de base est actif, les paramètres ne sont alors appliqués qu'au prochain démarrage d'un mouvement de base.

2.1.14.2.8 Démarrage d'un mouvement dans la direction positive en mode "Arrêt après Déclenchement"

Appel de fonction : `status = ncaction(ax_obj,ncPOS_MOVE+ncTRG_STOP,ncSTART)`

Paramètre : "`move.basis.parameter.v_pos`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`"
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement dans la direction positive en mode "Arrêt après Déclenchement"

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncPOS_MOVE+ncTRG_STOP`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case POS_BEW_START_TRG_STOP:
    p_ax_dat->move.basis.parameter.v_pos = ...;
    action_status = ncaction(ax_obj,ncPOS_MOVE+ncTRG_STOP,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Le paramètre "`move.basis.parameter.v_pos`" initialisé précédemment dans le variateur avec l'action NC "`ncBASIS_MOVE,ncINIT`" est réinitialisé avec cette action NC.

2.1.14.2.9 Démarrage d'un mouvement dans la direction négative en mode "Arrêt après Déclenchement"

Appel de fonction : `status = ncaction(ax_obj,ncNEG_MOVE+ncTRG_STOP,ncSTART)`

Paramètre : "`move.basis.parameter.v_neg`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.neg_hw_end == ncOPEN`"
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement dans la direction négative en mode "Arrêt après Déclenchement"

Indication(s) d'état : "`move.mode = ncBASIS_MOVE`"
"`move.detail = ncNEG_MOVE+ncTRG_STOP`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case NEG_BEW_START_TRG_STOP:
    p_ax_dat->move.basis.parameter.v_neg = ...;
    action_status = ncaction(ax_obj,ncNEG_MOVE+ncTRG_STOP,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

Important : Le paramètre "`move.basis.parameter.v_neg`" initialisé précédemment dans le variateur avec l'action NC "`ncBASIS_MOVE,ncINIT`" est réinitialisé avec cette action NC.

2.1.14.2.10 Démarrage d'un mouvement avec position cible absolue en mode "Arrêt après Déclenchement"

Appel de fonction : `status = ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP,ncSTART)`

Paramètre : "`move.basis.parameter.s`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`" (mouvement dans la direction positive)
"`dig_e.status.neg_hw_end == ncOPEN`" (mouvement dans la direction négative)
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement avec position cible absolue en mode "Arrêt après Déclenchement"

Indicartion(s) d'état :"`move.mode = ncBASIS_MOVE`"
"`move.detail = ncABS_MOVE+ncTRG_STOP`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case ABS_MOVE_START_TRG_STOP:
    p_ax_dat->move.basis.parameter.s = ...;
    action_status = ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

La position cible de base "**`s_target`**$_{\text{ABS\_MOVE}}$" se calcule comme pour un mouvement "normal" avec la position cible absolue.

Variantes : **"`ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP,ncSTART)`" :**
L'axe se déplace au maximum jusqu'à la position cible de base "**`s_target`**$_{\text{ABS\_MOVE}}$" !

"`ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP+ncS_REST,ncSTART)`" :
Dans le cas où un événement de déclenchement se produit, avant que l'axe n'atteigne "**`s_target`**$_{\text{ABS\_MOVE}}$", l'axe se déplace jusqu'à "**`s_target`**$_{\text{TRG\_STOP}}$ **`= s_trigger + trg_stop.s_rest`**", même si cette position se trouve au-delà de la position cible de base "**`s_target`**$_{\text{ABS\_MOVE}}$" !

2.1.14.2.11 Démarrage d'un mouvement avec distance de déplacement relatif en mode "Arrêt après Déclenchement"

Appel de fontion : `status = ncaction(ax_obj,ncREL_MOVE+ncTRG_STOP,ncSTART)`

Paramètre : "`move.basis.parameter.s`"

Condition(s) : "`controller.status == ncON`"
"`dig_e.status.pos_hw_end == ncOPEN`" (mouvement dans la direction positive)
"`dig_e.status.neg_hw_end == ncOPEN`" (mouvement dans la direction négative)
"`move.homing.status.ok == ncTRUE`"

Résultat(s) : Démarrage d'un mouvement avec distance de déplacement relatif en mode "Arrêt après Déclenchement"

Indications(s) d'état :"`move.mode = ncBASIS_MOVE`"
"`move.detail = ncREL_MOVE+ncTRG_STOP`"
"`move.basis.status.in_pos = ncFALSE`"
après "**`ncOK`**" état résultant de l'action NC

"`move.mode = ncOFF`"
après arrêt réussi du mouvement ou
après mouvement correctement effectué

"`move.basis.status.in_pos = ncTRUE`"
après mouvement correctement effectué

Exemple :

```
switch (step)
{
...
  case REL_BEW_START_TRG_STOP:
    p_ax_dat->move.basis.parameter.s = ...;
    action_status = ncaction(ax_obj,ncREL_MOVE+ncTRG_STOP,ncSTART);
    if ( action_status == ncOK )
    {
      step = W_MOVE_END;
    }
    break;

  case W_MOVE_END:
    if (p_ax_dat->move.basis.status.in_pos == ncTRUE)
    {
    /* Opération terminée avec succès */
        step = <NEXT_STEP>
    }
    break;
...
}
```

La position cible de base "**`s_target`**$_{\text{REL\_MOVE}}$" se calcule comme pour un mouvement "normal" avec distance de déplacement relatif.

Variantes : **"`ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP,ncSTART)`" :**
L'axe se déplace au maximum jusqu'à la position cible de base "**`s_target`**$_{\text{ABS\_MOVE}}$" !

"`ncaction(ax_obj,ncABS_MOVE+ncTRG_STOP+ncS_REST,ncSTART)`" :
Dans le cas où un événement de déclenchement se produit, avant que l'axe n'atteigne "**`s_target`**$_{\text{ABS\_MOVE}}$", l'axe se déplace jusqu'à "**`s_target`**$_{\text{TRG\_STOP}}$ **`= s_trigger + trg_stop.s_rest`**", même si cette position se trouve au-delà de la position cible de base "**`s_target`**$_{\text{ABS\_MOVE}}$" !

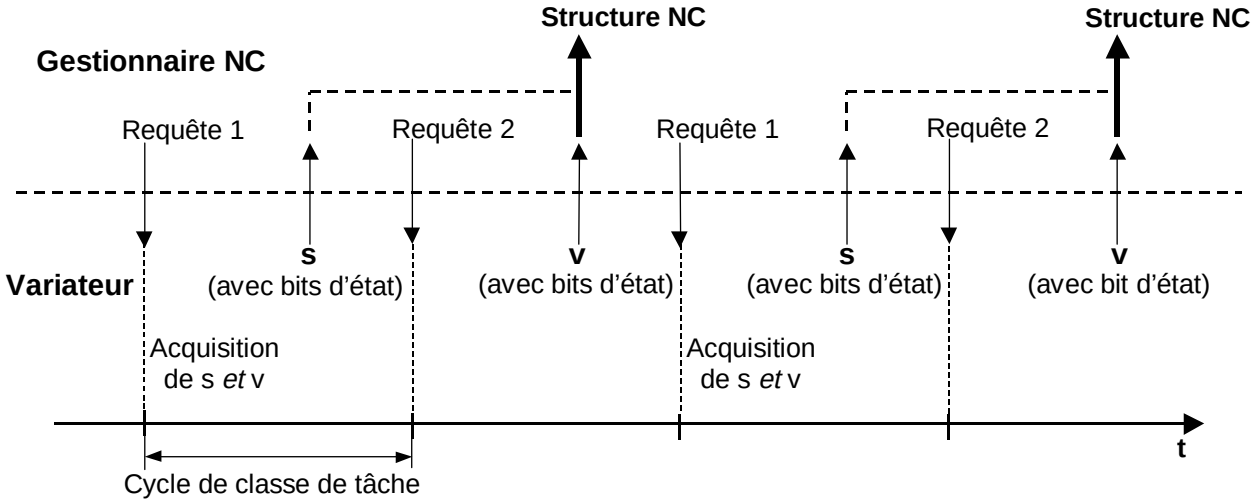
2.1.15 Moniteur

2.1.15.1 Structure de données

monitor		**Moniteur**
s	DINT	Position de consigne [unité] (est égale à la position réelle si le régulateur n'est pas enclenché)
v	REAL	Vitesse de consigne [unité/s]
status		**Bits d'état**
error	USINT	Erreur apparue : ncFALSE/ncTRUE
warning	USINT	Avertissement apparu : ncFALSE/ncTRUE
ds_warning	USINT	Erreur de traînage plus grande que "limit.ds_warning" : ncFALSE/ncTRUE

Pour que le contrôleur agisse de façon cyclique, le gestionnaire NC installe la tâche de gestionnaire NC dans une classe de tâche. Dans cette tâche de gestionnaire NC (appelée au début de la classe de tâche), les données « monitor » de tous les variateurs reliés au réseau CAN font l'objet d'une requête à chaque cycle. Cette requête est réalisée via un télégramme de diffusion unique et reçue simultanément par tous les variateurs. La cohérence des données est ainsi assurée car l'acquisition de position et de vitesse se fait en même temps dans tous les variateurs.

Après cette requête au gestionnaire NC, les données "`monitor.s`" et "`monitor.v`" sont alternativement envoyées au gestionnaire NC depuis le variateur, et ce en l'espace de deux cycles de classe de tâches successifs. La première requête déclenche l'acquisition de ces données dans le variateur. A réception du second télégramme, les données « monitor » sont transférées à la structure NC. Dans la structure NC, la tâche d'application dispose donc toujours de valeurs pour "`monitor.s`" et le "`monitor.v`", lesquelles ont été acquises simultanément dans le variateur.



Chaque télégramme « monitor » provenant du variateur contient des bits d'état, ce qui permet à la tâche d'application de réagir aux événements se produisant dans le variateur le plus rapidement possible

L'apparition d'une erreur ou d'un avertissement est signalée dans "`monitor.error`" ou "`monitor.warning`".

Important : L'apparition d'une erreur ou d'un avertissement dans le variateur est immédiatement signalée dans "`monitor.error`" ou "`monitor.warning`". Ensuite, l'enregistrement de message correspondant est lu par le gestionnaire NC et apparaît dans la composante "`message.record`".

Si la valeur absolue de l'erreur de traînage est supérieure à la valeur indiquée dans "`limit.parameter.ds_warning`", aucun message d'erreur n'est généré mais "`monitor.status.ds_warning`" est mis à la valeur "`ncTRUE`". Dans le cas contraire, c'est "`ncFALSE`" qui est reporté dans "`monitor.status.ds_warning`".

2.1.16 Traitement des messages (erreurs, avertissements)

2.1.16.1 Structure de données

message		**Messages (erreurs, avertissements)**
count		**Nombre**
error	USINT	Nombre d'erreurs non acquittées
warning	USINT	Nombre d'avertissements non acquittés
record		**Enregistrement de message en cours**
number	UINT	Numéro
info	UDINT	Info additionnelle (codée)
par_id	UINT	ID de paramètre (utilisé seulement pour les erreurs lors des opérations de lecture ou d'écriture de paramètres)
text		**Détermination du texte pour l'enregistrement de message actuel**
status		**Etat**
lines	UINT	Lignes du texte attribué
error	UINT	Erreur : 0: Pas d'erreur 1: Adresse du tampon de données zéro 2: Taille du tampon de données inférieure au minimum (40) 3: Erreur avec texte (en anglais) dans le tampon de données
parameter		**Paramètre**
format	UINT	Format : ncEMPTY/ncZERO/ncBREAK
columns	UINT	Nombre de colonnes par ligne (longueur de ligne)
data_modul	USINT[10]	Nom du module de données pour textes d'erreur
dat_len	UINT	Taille du tampon de données dans le programme d'application
Data_adr	UDINT	Adresse du tampon de données dans le programme d'application

La composante "message.count" contient le nombre d'erreurs et d'avertissements qui se sont produits et qui n'ont pas encore été acquittés par l'utilisateur avec l'action NC "ncMESSAGE,ncACKNOWLEDGE".

La composante "message.record" contient l'enregistrement de message en cours. Il s'agit du message le plus ancien qui n'a pas encore été acquitté par l'utilisateur avec l'action NC "ncMESSAGE,ncACKNOWLEDGE".

Les textes en anglais correspondant à "message.record.number" se trouvent dans le fichier de textes d'erreur "acp10err001.txt" :

```
...
4007: Lag error stop limit exceeded
      Info(REAL): Current lag error
4008: Positive limit switch reached
4009: Negative limit switch reached
4010: Switch controller on not possible: Both limit switches are closed
4011: Switch controller off not possible: Movement active
4012: Switch controller on not possible: INIT parameter missing or not valid
      Info('PARID'): Parameter ID
...
```

Si un message d'erreur contient une information additionnelle, une ligne de texte supplémentaire figure sous le texte d'erreur, pour le numéro d'erreur en cours. Cette ligne commence par "**Info(<T*ype de données*>)**" (voir plus haut). L'information addtionnelle correspondant à l'enregistrement de message en cours se trouve dans "message.record.info". Cependant, cette information ne peut pas être lue directement car la valeur contenue dans la composante définie avec UDINT peut être de n'importe quel type.

La valeur de l'info additionnelle pour l'enregistrement de message en cours peut être déterminée à l'aide de l'action NC "ncMESSAGE,ncTEXT". Si à l'appel de cette action NC, "message.text.parameter.datenmodul[0]=0" est appliqué, un texte comme ceux qui figurent dans les exemples suivants est reporté dans le tampon de données du programme d'application :

"Info: 12.345"	Valeur d'information de type REAL
"Info: 678"	Valeur d'information de type UINT, DINT ou UDINT
"Info: Not existing"	Pas d'information pour ce numéro d'erreur

Exemple :

Après arrêt d'un mouvement, les données suivantes apparaissent dans l'enregistrement de message en cours :

```
message.record.number = 4007
message.record.par_id = 0
```

Avec l'action NC "ncMESSAGE,ncTEXT", le texte déterminé pour cet enregistrement de message est le suivant :

"Info: **45.678**"

La liste d'erreurs exposée précédemment permet de déduire que l'erreur de traînage maximale autorisée a été dépassée ; de plus, elle indique la valeur de l'erreur de traînage qui a entraîné l'arrêt du mouvement :

```
4007: Lag error stop limit exceeded
      Info(REAL): Current lag error 45.678
```

Si le gestionnaire NC génère une erreur lors d'une opération de lecture ou d'écriture de paramètre, l'ID de paramètre correspondant est alors reporté dans "message.record.par_id".

Exemple :

Après exécution de l'action NC "ncLIMIT,ncINIT" avec "t_jolt=0.5", les données suivantes apparaissent dans l'enregistrement de message en cours :

```
message.record.number = 5
message.record.par_id = 125
```

Pour cet enregistrement de message, l'action NC "ncMESSAGE,ncTEXT" délivre le texte suivant :

"Info: **0.02**"

La liste d'erreur et la liste de paramètres (par exemple dans "acp10man.h") permet de déduire que la valeur définie pour le temps de vibration n'a pas été acceptée par le variateur, étant donné qu'elle est supérieure à la valeur maximale autorisée pour ce paramètre :

```
5:  Value of parameter higher than maximum value
    Info('REAL'): Maximum value 0.02

..._AXLIM_T_JOLT 125 /* (REAL) Jolt time */
```

Erreurs / Avertissements

record.number <= 0x7FFF: (32767) **Erreur**

Une erreur en réponse à une écriture de paramètre signifie par exemple que la valeur de ce paramètre n'a pas été acceptée par le variateur et que le paramètre n'est donc "**pas initialisé**".

record.number >= 0x8000: (32768) **Avertissement**

Un avertissement en réponse à une écriture de paramètre signifie par exemple que la valeur de ce paramètre a été acceptée par le variateur (éventuellement modifiée) et que le paramètre est "**initialisé**".

Détermination du texte pour l'enregistrement de message en cours

Avec l'action NC "`ncMESSAGE,ncTEXT`", le texte destiné au message en cours (contenu dans "`message.record`") peut être délivré dans un format défini par l'utilisateur s'il existe un module de données pour textes d'erreur approprié dans le contrôleur programmable. Le nom de ce module de données doit être entré dans "`message.text.parameter.datamodul`". Le nombre de colonnes pour une ligne de texte (longueur de ligne) peut être défini avec le paramètre "`message.text.parameter.column`". Le format souhaité pour le texte à obtenir est défini à l'aide de "`message.text.parameter.format`" :

ncEMPTY : Si un texte est plus long que la longueur de ligne qui a été définie, il est tronqué à la fin de la ligne. La suite de ce texte est reportée à la ligne suivante. Si la fin de la ligne n'est pas encore atteinte à la fin d'un texte, des espaces sont placés dans toutes les colonnes restantes après ce texte.

ncZERO : Si un texte est plus long que la longueur de ligne qui a été définie, il est tronqué à la fin de la ligne. La suite de ce texte est reportée à la ligne suivante. Un caractère ZERO est reporté à la fin de chaque texte et à la fin de chaque ligne. Si la fin de la ligne n'est pas encore atteinte à la fin d'un texte, des espaces sont placés dans toutes les colonnes restantes de cette ligne après la marque de fin zéro.

ncSTOP : Si un texte est plus long que la longueur de ligne qui a été définie, il est tronqué à la fin d'un mot, avant la fin de la ligne. La suite de ce texte s'affiche à la ligne suivante en commençant par le début du mot suivant. Si la fin de la ligne n'est pas encore atteinte à la fin d'un texte, des espaces sont placés dans toutes les colonnes restantes après ce texte.

ncSTOP + ncZERO (combinaison de "ncSTOP" et "ncNULL") **:**
Si un texte est plus long que la longueur de ligne qui a été définie, il est tronqué à la fin d'un mot, avant la fin de la ligne. La suite de ce texte s'affiche à la ligne suivante en commençant par le début du mot suivant. Un caractère ZERO est reporté à la fin de chaque texte et à la fin de chaque ligne. Si la fin de la ligne n'est pas encore atteinte à la fin d'un texte, des espaces sont placés dans toutes les colonnes restantes de cette ligne après la marque de fin zéro.

Il est possible d'obtenir jusqu'à 4 textes différents pour le message en cours. Chacun de ces textes est reporté dans le tampon de données à partir de l'adresse indiquée, en accord avec les paramétrages décrits plus haut (voir exemple sur la page suivante). Si la détermination de texte a été effectuée correctement, le nombre des lignes de texte obtenues apparaît dans "`message.text.status.line`".

Exemple :

Les données suivantes apparaissent dans l'enregistrement de message en cours :

```
message.record.number = 4010
message.record.par_id = 93
```

Pour cet enregistrement de message, l'action NC "ncMESSAGE,ncTEXT" fournit le texte suivant :

```
1: "Command: Switch controller on/off"
2: "Switch controller on not possible: Both limit switches are closed"
```

Avec "columns=38" et "format=ncEMPTY", on obtiendrait les trois lignes suivantes :

```
1: "Command: Switch controller on/off     "
2: "Switch controller on not possible: Both"
3: "limit switch are closed                "
```

Avec "columns=38" et "format=ncZERO", on obtiendrait les trois lignes suivantes :

```
1: "Command: Switch controller on/off\0"
2: "Switch controller on not possible: Both\0"
3: "limit switch are closed\0"
```

Avec "columns=38" et "format=ncBREAK", on obtiendrait les trois lignes suivantes :

```
      12345678901234567890123456789012345678
1: "Command: Switch controller on/off      "
2: "Switch controller on not possible:     "
3: "Both limit switch are closed          "
```

Avec "columns=38" et "format=ncBREAK+ncZERO", on obtiendrait les trois lignes suivantes :

```
1: "Command: Switch controller on/off\0"
2: "Switch controller on not possible:\0"
3: "Both limit switch are closed\0"
```

2.1.16.2 Actions NC

Sujet	Action	Description
ncMESSAGE	ncTEXT	Déterminer le texte pour l'enregistrement de message en cours
ncMESSAGE	ncACKNOWLEDGE	Acquitter l'enregistrement de message en cours

2.1.16.2.1 Détermination du texte pour l'enregistrement de message en cours

Appel de fonction : `status = ncaction(ax_obj,ncMESSAGE,ncTEXT)`

Paramètre : "`message.text.parameter`"
"`message.record`"

Condition(s) : -

Résultat(s) : Détermination du texte pour l'enregistrement de message en cours, avec les paramètres spécifiés.

Indication(s) d'état : "`message.text.status.lines = 0`"
"`message.text.status.error = 0`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC

"`message.text.status.error = <`***`Error Number`***`>`"
après une erreur lors de la détermination du texte

Nombre de lignes du texte qui a été déterminé : "`message.text.status.lines`"
Adresse où se trouve le texte qui a été déterminé : "`message.text.parameter.data_adr`"
après détermination de texte effectuée avec succès

Important : "`ncMESSAGE,ncTEXT`" est une action NC interne sans transfert de données entre le gestionnaire NC et le variateur. Cela signifie que cette action NC peut toujours être exécutée, même si le traitement d'une action NC appelée précédemment (hormis "`ncMESSAGE,ncTEXT`") n'est pas encore terminé.

Lorsque "`message.text.parameter.datamodul[0]=0`" est appliqué, seul le texte pour l'information additionnelle est déterminé, et ce dans un format prédéfini avec "`ncMESSAGE,ncTEXT`". Les paramètres "`message.text.parameter.format`" et "`...columns`" sont ici ignorés.

2.1.16.2.2 Acquittement de l'enregistrement de message en cours

Appel de fonction : `status = ncaction(ax_obj,ncMESSAGE,ncACKNOWLEDGE)`

Paramètre : -

Condition(s) : -

Résultat(s) : Si un enregistrement de message est encore présent, le gestionnaire NC le transfère à l'objet NC ; dans le cas contraire, l'enregistrement de message de l'objet NC est supprimé.

Indication(s) d'état : "`message.count.error--`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC,
si un enregistrement d'erreur ("`message.record.number<= 32767`") a été acquitté

"`message.count.warning--`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC,
si un enregistrement d'avertissement ("`message.record.number>= 32768`") a été acquitté

Nouvel enregistrement de message dans "`message.record`"
après "**`ncOK`**" en tant qu'état résultant de l'action NC, si tous les messages n'ont pas encore été acquittés

```
"message.record.number    = 0"
"message.count.warning    = 0"
"message.count.error      = 0"
```

après "**`ncOK`**" en tant qu'état résultant de l'action NC, si tous les messages ont été acquittés

Important : "`ncMESSAGE,ncACKNOWLEDGE`" est une action NC interne sans transfert de données entre le gestionnaire NC et le variateur. Cela signifie que cette action NC peut être exécutée à n'importe quel moment, même si le traitement d'une action NC appelée précédemment n'est pas encore terminé.

Exemple :
```
if ( p_ax_dat->monitor.status.error == ncTRUE )
    /* ici, pas avec "p_ax_dat->monitor.status.warning == ncTRUE" */
      step = ERROR_STEP;

    switch (step)
    {
    ...

    /* Arrêt d'un mouvement, si souhaité et nécessaire */
      case ERROR_STEP:
        if ( error_stop == ncON &&
             p_ax_dat->move.mode != ncOFF )
        {
          p_ax_dat->move.stop.index.command=ind_error_stop;
          action_status = ncaction(ax_obj,ncMOVE,ncSTOP);
          if ( action_status == ncOK )
          {
            step = W_MESSAGE_NUMBER;
          }
        }
        else
        {
          step = W_MESSAGE_NUMBER;
        }
        break;

    /*  Il n'est plus nécessaire de désactiver le régulateur avec une */
    /*   action  NC  puisqu'un  mode  d'arrêt  peut  être  défini  avec  */
    /* "Switch controller off" !*/

    /* Attendre jusqu'à ce que le numéro soit reporté dans l'enregistrement*/
    /* de message en cours */
      case W_MESSAGE_NUMBER:
        if ( p_ax_dat->message.record.number != 0 )
        {
        /* "monitor.error/warning" sont des bits d'état cyclique */
        /*   L'enregistrement  de  message  est  lu  par  le  variateur*/
        /* ultérieurement*/
          step = MESSAGE_TEXT;
        }
        break;
```

```
/* Détermination du texte pour l'enregistrement de message en cours */
  case MESSAGE_TEXT:
/* Initialisation des paramètres dans le sous-programme d'init. INIT-UP*/
    action_status = ncaction(ax_obj,ncMESSAGE,ncTEXT);
    if ( action_status == ncOK )
    {
      step = W_MESSAGE_TEXT;
    }
    break;

/* Attente jusqu'à ce que la détermination du texte soit terminée*/
  case W_MESSAGE_TEXT:
    if ( p_ax_dat->message.text.status.lines != 0 ||
         p_ax_dat->message.text.status.error != 0 )
    {
      step = MESSAGE_QUIT;
    }
    break;

/* Acquittement de l'enregistrement de message en cours */
  case MESSAGE_ACK:
    if (acknowledge)
    {
      action_status = ncaction(ax_obj,ncMESSAGE,ncACKNOWLEDGE);
      if ( action_status == ncOK )
      {
        acknowledge = 0;
        if ( p_ax_dat->error.number == 0 )
        {
        /* Tous les enregistrements de messages sont acquittés */
          step = ALL_MESSAGE_ACKN;
        }
        else
        {
          step = MESSAGE_TEXT;
        }
      }
    }
    break;

/* Tous les enregistrements de message sont acquittés */
  case ALL_MESSAGE_ACKN:
    ...
    break;

...
}
```

OSCILLOSCOPE

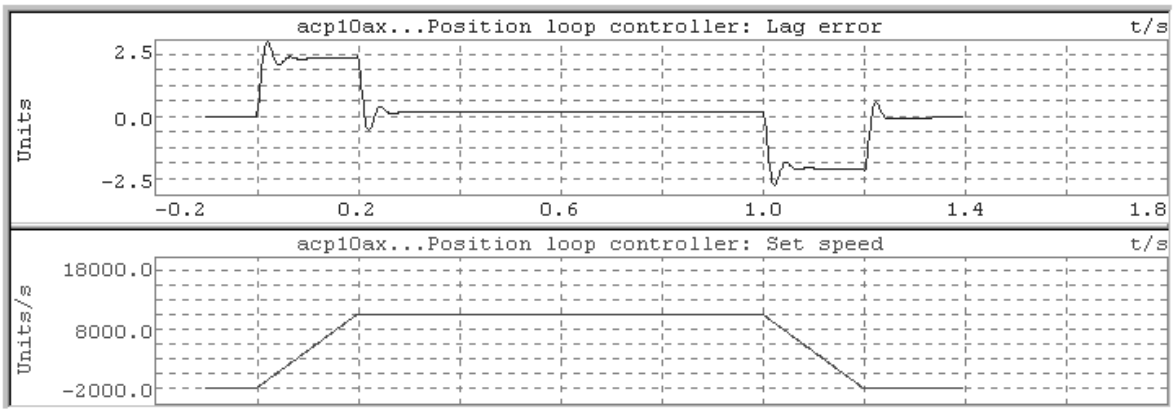
(à partir de ACP10-SW V0.23)

23 Novembre 2000

1 GENERALITES

Avec l'oscilloscope intégré, l'utilisateur peut enregistrer cycliquement l'évolution de certains paramètres dans le variateur en vue d'une analyse ultérieure. Le lancement d'acquisition et l'évaluation des données qui s'ensuit sont effectués le plus souvent avec les Motion Components (l'utilisateur peut aussi lancer l'acquisition des données avec un programme applicatif et évaluer ensuite ces données avec MCAD).

Les deux diagrammes ci-dessous illustrent une acquisition de données effectuée avec les Motion Components pour l'erreur de traînage et la vitesse de consigne.



2 MANIPULATION DE L´OSCILLOSCOPE

Dans le cas d'une utilisation de l'oscilloscope avec le gestionnaire NC "ACP10MAN", les Motion Components écrivent la configuration d'acquisition dans la structure de données utilisateur et transmettent ensuite l'action "`ncTRACE,ncSTART`" au gestionnaire NC. Ensuite, les Motion Components lisent l'état d'acquisition cycliquement dans la structure de données utilisateur. Une fois l'acquisition effectuée, les données d'acquisition sont transférées depuis le variateur jusqu'aux Motion Components sur le PC, via le contrôleur programmable.

2.1 Configuration d'acquisition

Les paramètres pouvant être utilisés pour configurer l'acquisition des données sont les suivants :

Paramètres dans les Motion Components	**Paramètres dans la structure de données utilisateur**	**Description**
Durée max. d'acquisition	`.t_trace`	Temps d'enregistrement
Période d'échantillonnage	`.t_sampling`	Période d'échantillonnage
Retard de déclenchement	`.t_delay`	Temps de retard (par rapport à l'événement de déclenchement)
Seuil de déclenchement	`.trigger.threshold`	Seuil de déclenchement
Fenêtre de déclenchement	`.trigger.window`	Fenêtre de déclenchement
Evénement de déclenchement	`.trigger.event`	Evénement de déclenchement
Paramètres pour déclenchement	`.trigger.par_id`	Paramètre de déclenchement
Paramètres sélectionnés pour l'acquisition	`.test_dat[i].par_id`	Paramètre de test

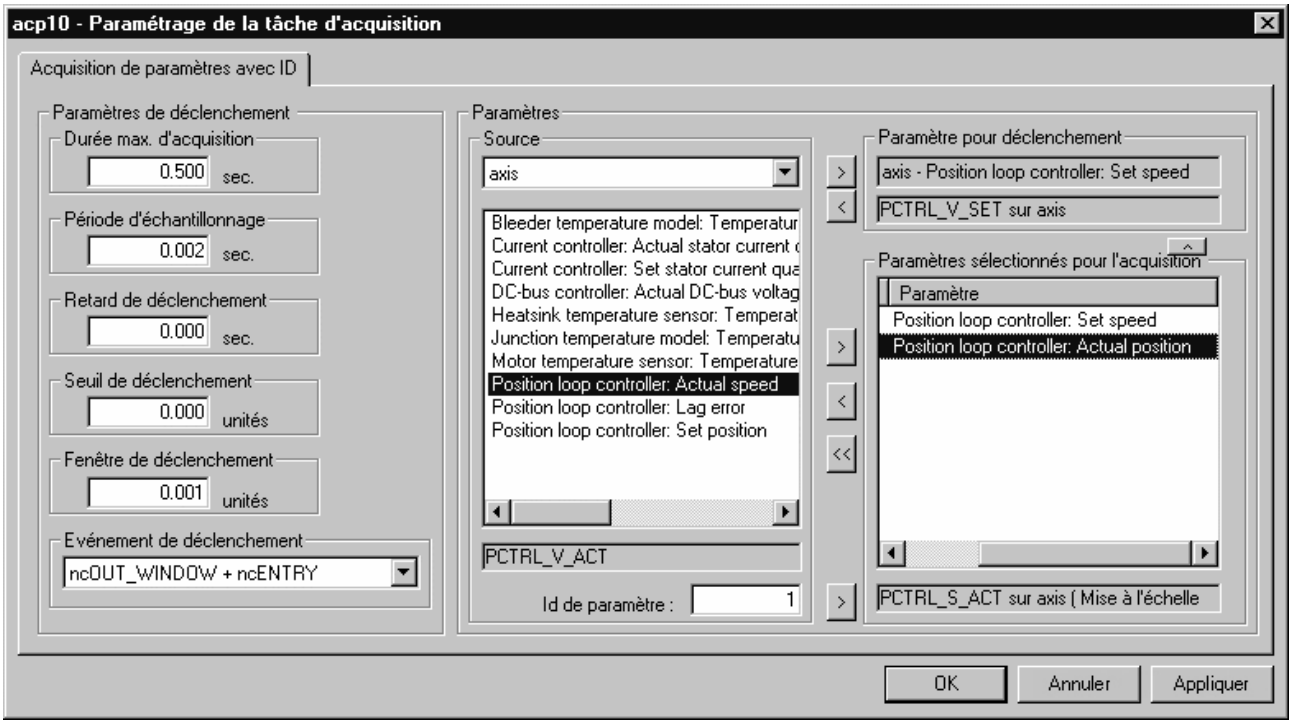
2.1.1 Configuration d'acquisition dans la structure de données utilisateur

Dans le système ACP10-SW, la structure de données pour l'objet NC de type "ncMODUL" contient les composantes suivantes pour la définition de la configuration d'acquisition :

trace		**Acquisition**
status	USINT	Etat :
		ncTR_START: Acquisition lancée par l'utilisateur
		ncTR_END: Acquisition terminée par le variateur
		ncTR_TRIGG: Attente du déclenchement de démarrage (par trigger)
		ncTR_DELAY: Attente de la fin de la temporisation avant démarrage
		ncTR_TRACE: Enregistrement jusqu'à la fin du temps d'enregistrement
		ncTR_RING: Enregistrement dans un tampon circulaire
		ncTR_REST: Enregistrement jusqu'à la fin du temps résiduel
buf_size	UDINT	Taille du tampon de données d'acquisition dans le variateur [octets]
t_trace	REAL	Temps d'enregistrement [s]
t_sample	REAL	Temps d'échantillonnage [s]
t_delay	REAL	Temps de retard (par rapport à l'événement de déclenchement) [s]
trigger		**Trigger**
par_id	UINT	ID de paramètre (ou ncOFF)
event	USINT	Evénement :
		NcOFF : Pas d'événement de déclenchement
		ncIN_WINDOW : Zone "A l'intérieur de la fenêtre"
		ncOUT_WINDOW : Zone "A l'extérieur de la fenêtre"
		ncABOVE_WINDOW : Zone "Au-dessus de la fenêtre"
		ncBELOW_WINDOW : Zone "Au-dessous de la fenêtre"
		+ ncENTRY : Lors de l'entrée dans la zone définie
threshold	REAL	Seuil [unité fonction du paramètre de déclenchement]
window	REAL	Fenêtre [unité fonction du paramètre de déclenchement]
test_dat[10]		**Données de test**
par_id	UINT	ID de paramètre (ou ncOFF)

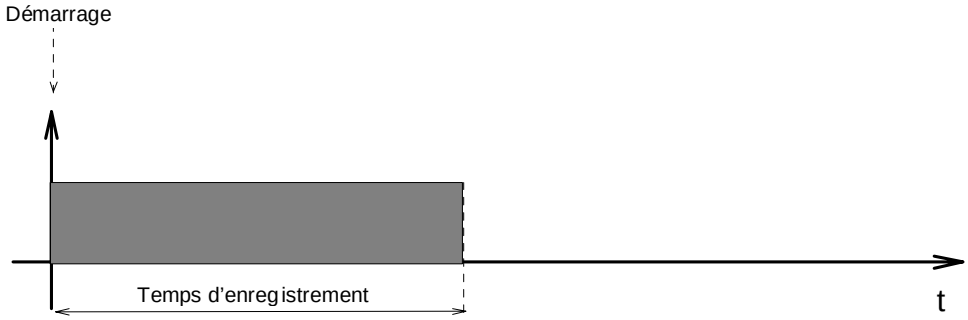
2.1.2 Configuration d'acquisition dans les Motion Components

Dans les Motion Components, la configuration d'acquisition est définie dans la boîte de dialogue "Paramétrage de la tâche d'acquisition".

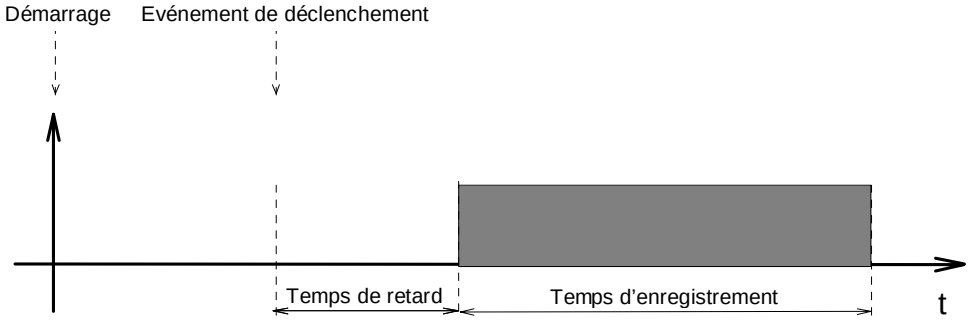


2.2 Modes d'enregistrement

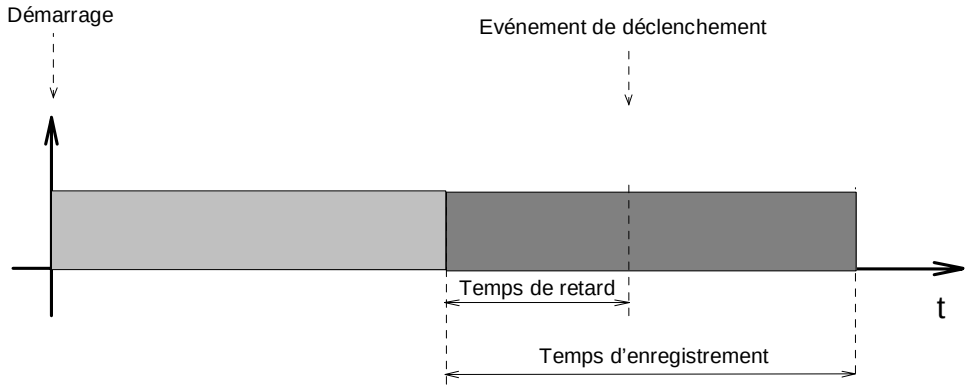
2.2.1 Démarrage manuel (sans événement de déclenchement)



2.2.2 Démarrage avec événement de déclenchement et temps de retard positif (post-déclenchement)

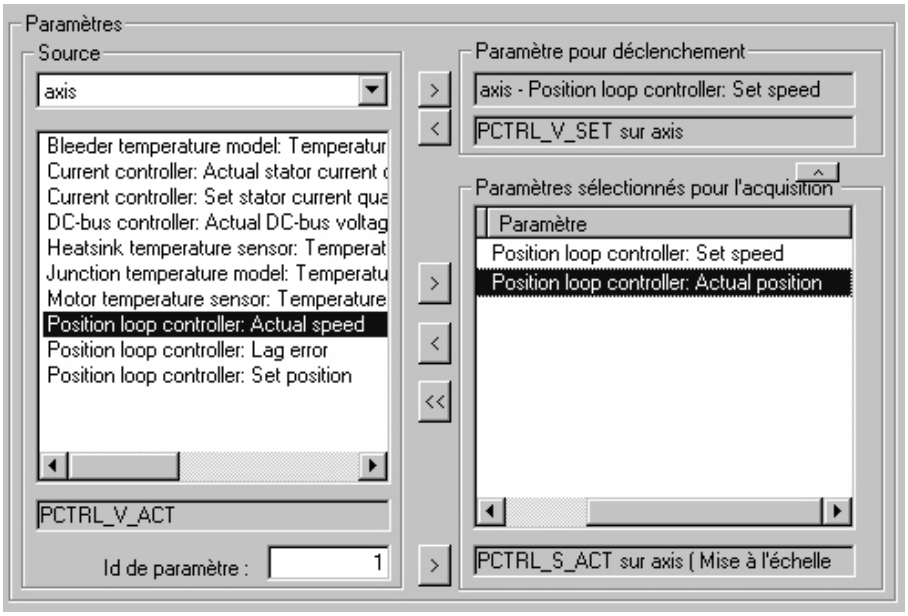


2.2.3 Démarrage avec événement de déclenchement et temps de retard négatif (prédéclenchement)



2.3 Paramètres de déclenchement et de test

Dans les Motion Components, tous les paramètres pouvant être sélectionnés en tant que paramètres de déclenchement et de test pour une acquisition de données et pour l'objet NC spécifié dans Source sont affichés dans le dialogue de configuration. Les boutons "**>**" et "**<**" permettent respectivement d'activer et de désactiver des paramètres en tant que paramètre de déclenchement (ou de test). Le bouton "**<<**" permet de désactiver tous les paramètres de test.



De plus, les paramètres de test peuvent être définis en entrant le numéro d'"Id de paramètre".

2.4 Evénements de déclenchement

Les événements de déclenchement pouvant être définis sont les suivants :

Événement de déclenchement	**Description**
`ncOFF`	Pas d'événement de déclenchement
`ncIN_WINDOW` `ncIN_WINDOW + ncENTRY`	Zone "A l'intérieur de la fenêtre" (réaction sur niveau) Entrée dans cette zone (réaction sur front)
`ncOUT_WINDOW` `ncOUT_WINDOW + ncENTRY`	Zone "A l'extérieur de la fenêtre" (réaction sur niveau) Entrée dans cette zone (réaction sur front)
`ncABOVE_WINDOW` `ncABOVE_WINDOW + ncENTRY`	Zone "Au-dessus de la fenêtre" (réaction sur niveau) Entrée dans cette zone (réaction sur front)
`ncBELOW_WINDOW` `ncBELOW_WINDOW + ncENTRY`	Zone "Au-dessous de la fenêtre" (réaction sur niveau) Entrée dans cette zone (réaction sur front)

L'événement de déclenchement est désactivé avec "**`ncOFF`**". Dans ce cas, l'enregistrement commence immédiatement au lancement de l'acquisition ("Démarrage manuel"). L'événement de déclenchement est également désactivé si aucun paramètre de déclenchement n'est activé ("`trigger.par_id = ncOFF`").
Pour tous les autres modes d'enregistrement, l'enregistrement ne démarre qu'à l'apparition de l'événement de déclenchement. Les événements de déclenchement se rapportent tous à la valeur du paramètre de déclenchement spécifié. Les événements comme "*Début d'un mouvement*" ou "*Fin d'un mouvement*" ne font pas l'objet d'une définition propre (comme le montrent les exemples ci-dessous, ces événements se définissent simplement avec le paramètre "Vitesse de consigne").

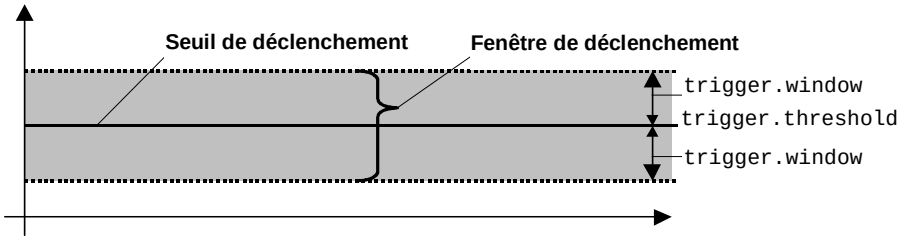
2.4.1 Evénement de déclenchement "ncIN_WINDOW"

ncIN_WINDOW (dans la zone "à l'intérieur de la fenêtre" ; réaction sur niveau) :

L'événement de déclenchement se produit si la valeur "x" du paramètre de déclenchement vérifie la condition suivante :

"x ≤ trigger.threshold + trigger.window" **et**

"x ≥ trigger.threshold - trigger.window"



Avec "trigger.window==0", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement correspond exactement à la valeur "trigger.threshold" (ce mode de déclenchement ne convient que pour des paramètres INT ou BOOL).

ncIN_WINDOW + ncENTRY (A l'entrée dans la zone "À l'intérieur de la fenêtre" ; réaction sur front) :

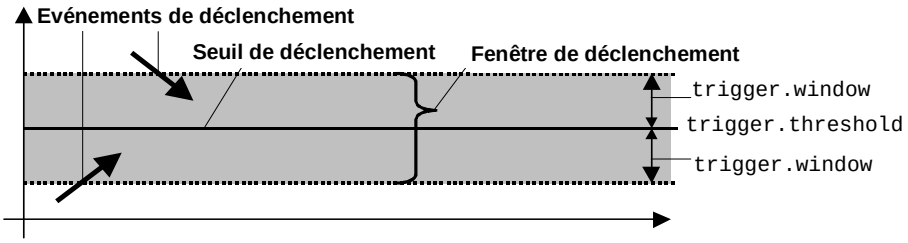
L'événement de déclenchement se produit si la valeur du paramètre de déclenchement

passe au-dessous de la limite supérieure de la fenêtre "trigger.threshold + trigger.window"

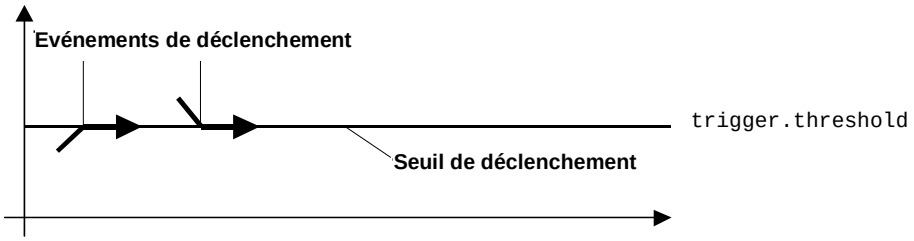
ou

passe au-dessus de la limite inférieure de la fenêtre "trigger.threshold + trigger.window"

(après avoir été précédemment en-dehors de la fenêtre) et se trouve ensuite dans la fenêtre. Si la valeur du paramètre de déclenchement au lancement de l'acquisition est à l'intérieur de la fenêtre, l'enregistrement n'est pas lancé.



Avec "trigger.window==0", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement atteint de façon exacte la valeur "trigger.threshold", par le haut ou par le bas (ce mode de déclenchement ne convient que pour des paramètres INT ou BOOL).

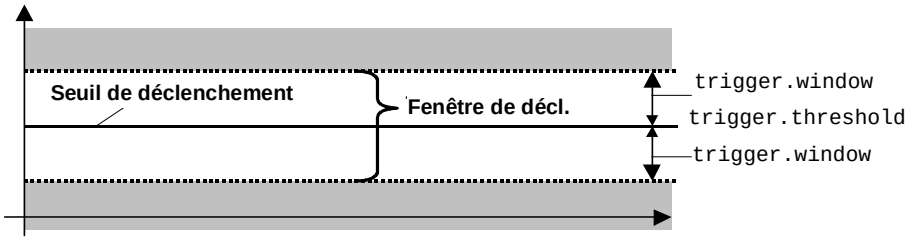


2.4.2 Evénement de déclenchement "ncOFF_WINDOW"

ncOUT_WINDOW (dans la zone "à l'extérieur de la fenêtre" ; réaction par rapport au niveau) :

L'événement de déclenchement se produit si la valeur "x" du paramètre de déclenchement vérifie la condition suivante :

"`x > trigger.threshold + trigger.window`" **ou**
"`x < trigger.threshold - trigger.window`"



Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement est différente de la valeur "`trigger.threshold`".

ncIN_WINDOW + ncENTRY (à l'entrée dans la zone "à l'intérieur de la fenêtre" ; réaction sur front) :

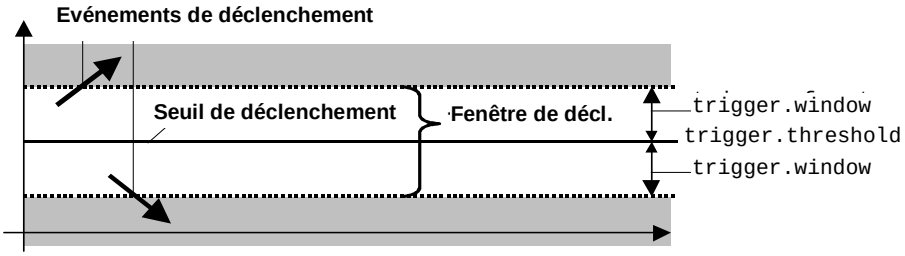
L'événement de déclenchement se produit si la valeur du paramètre de déclenchement

passe au-dessus de la limite supérieure de la fenêtre "`trigger.threshold + trigger.window`"

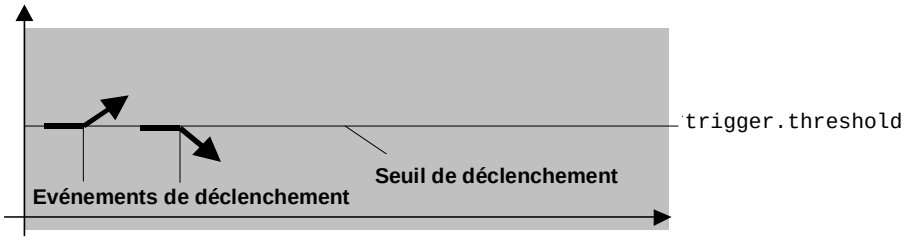
ou

passe au-dessous de la limite inférieure de la fenêtre "`trigger.threshold + trigger.window`"

(après avoir été précédemment dans la fenêtre) et se trouve ensuite hors de la fenêtre. Si la valeur du paramètre de déclenchement au lancement de l'acquisition est à l'extérieur de la fenêtre, l'enregistrement n'est pas lancé.



Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement atteint de façon exacte la valeur "`trigger.threshold`" (ce mode de déclenchement ne convient que pour des paramètres INT ou BOOL).

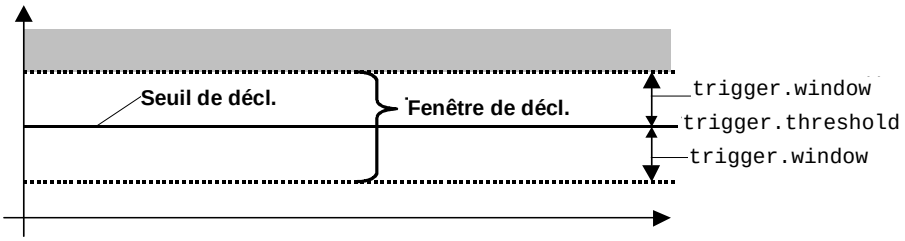


2.4.3 Evénement de déclenchement "ncABOVE_WINDOW"

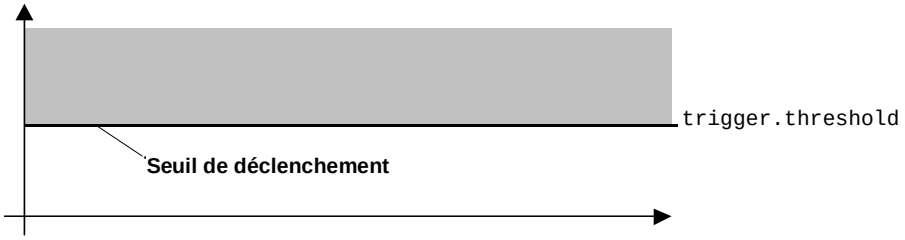
ncABOVE_WINDOW (dans la zone "au-dessus de la fenêtre" ; réaction par rapport au niveau) :

L'événement de déclenchement se produit si la valeur "x" du paramètre de déclenchement vérifie la condition suivante :

"`x > trigger.threshold + trigger.window`"

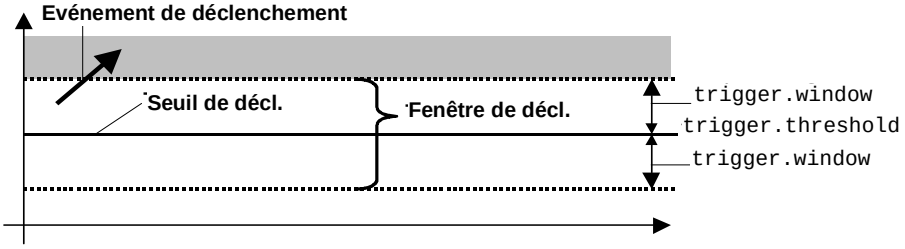


Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement est supérieure la valeur "`trigger.threshold`".

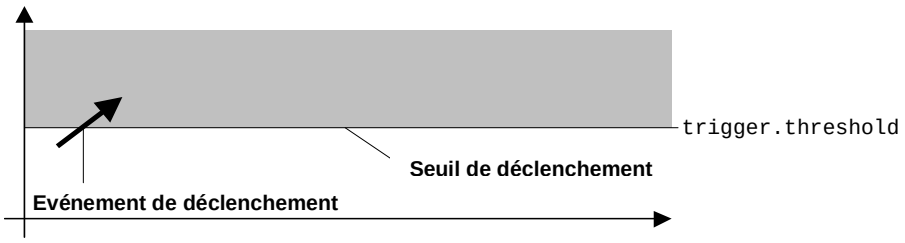


ncABOVE_WINDOW + ncENTRY (à l'entrée dans la zone "au-dessus de la fenêtre" ; réaction sur front) :

L'événement de déclenchement se produit si la valeur du paramètre de déclenchement passe au-dessus de la limite supérieure de la fenêtre "`trigger.threshold + trigger.window`". Si la valeur du paramètre de déclenchement au lancement de l'acquisition est au-dessus de la fenêtre, l'enregistrement n'est pas lancé.



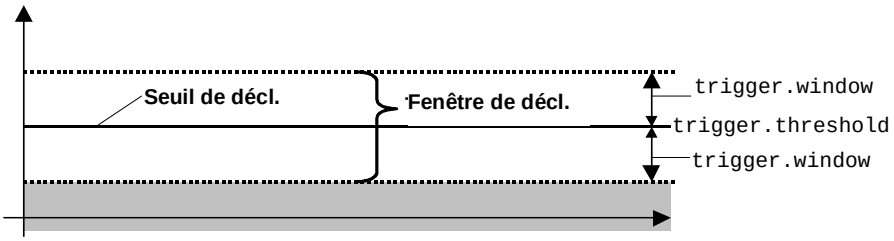
Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement est supérieure à la valeur "`trigger.threshold`".



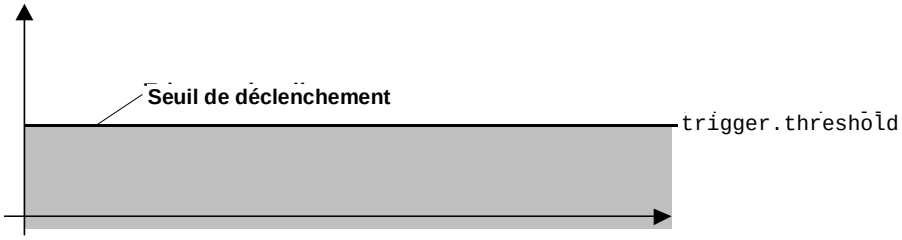
2.4.4 Evénement de déclenchement "ncBELOW_WINDOW"

ncABOVE_WINDOW (dans la zone "au-dessus de la fenêtre" ; réaction par rapport au niveau) :

L'événement de déclenchement se produit si la valeur "x" du paramètre de déclenchement vérifie la condition suivante :
"`x < trigger.threshold - trigger.window`"

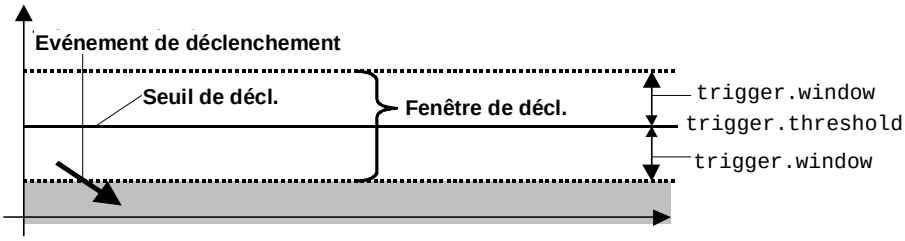


Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement est inférieure la valeur "`trigger.threshold`".

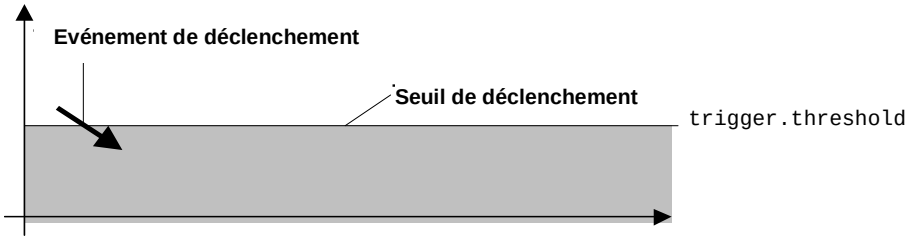


ncBELOW_WINDOW + ncENTRY (à l'entrée dans la zone "au-dessous de la fenêtre" ; réaction sur front) :

L'événement de déclenchement se produit si la valeur du paramètre de déclenchement passe au-dessous de la limite inférieure de la fenêtre "`trigger.threshold + trigger.window`". Si la valeur du paramètre de déclenchement au lancement de l'acquisition est au-dessous de la fenêtre, l'enregistrement n'est pas lancé.



Avec "`trigger.window==0`", l'événement de déclenchement se produit si la valeur du paramètre de déclenchement passe en-dessous de la valeur "`trigger.threshold`".



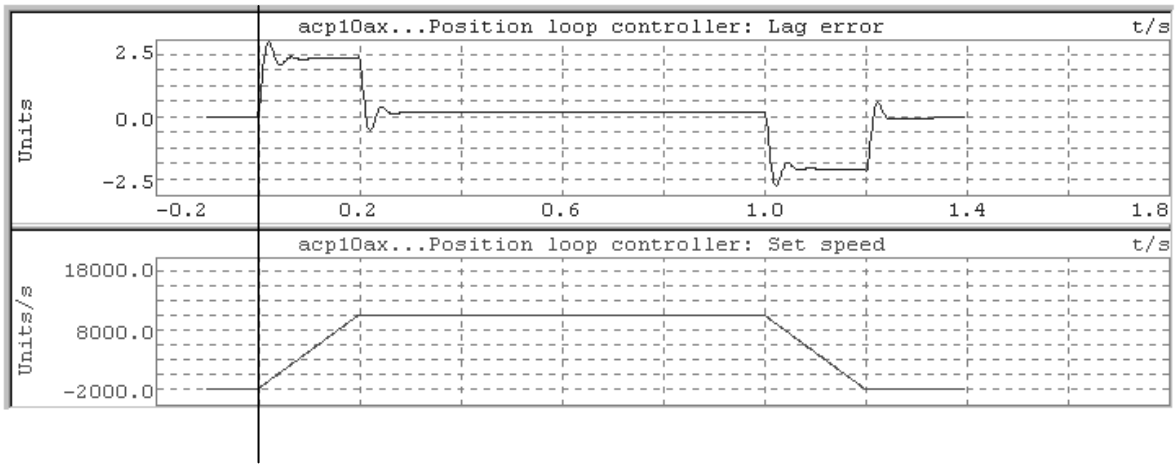
2.5 Exemples de configuration

2.5.1 Début d'un mouvement

Avec la configuration ci-dessous dans les Motion Components, la vitesse de consigne et l'erreur de traînage sont enregistrées pendant 1,5 secondes. L'acquisition des données commence 0,1 secondes avant le début du mouvement :

Durée maximale d'acquisition	`1.500`
Période d'échantillonnage	`0.002`
Retard de déclenchement	`-0.100`
Seuil de déclenchement	`0.000`
Fenêtre de déclenchement	`0.001`
Evénement de déclenchement	`ncOFF_WINDOW + ncENTRY`
Paramètre à déclencher	Régulateur de position : vitesse de consigne
Paramètres sélectionnés	Régulateur de position : erreur de traînage Régulateur de position : vitesse de consigne

Avec cette configuration, les données d'acquisition pourraient se présenter de la manière suivante :



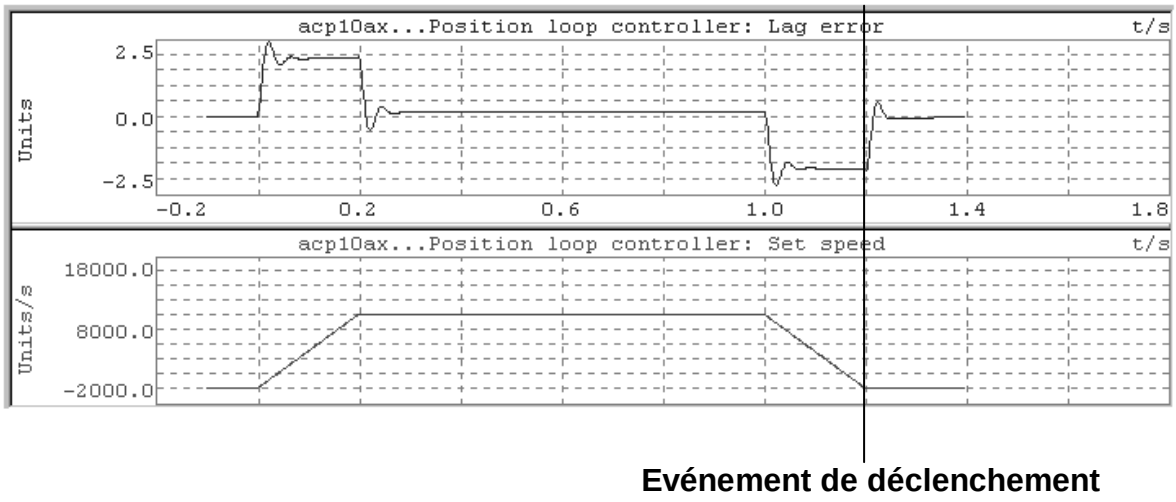
Evénement de déclenchement

2.5.2 Fin d'un mouvement

Avec la configuration ci-dessous dans les Motion Components, la vitesse de consigne et l'erreur de traînage sont enregistrées pendant 1,5 secondes. L'acquisition des données commence 1,4 secondes avant la fin du mouvement :

Durée maximale d'acquisition	`1.500`
Période d'échantillonnage	`0.002`
Retard de déclenchement	`-1.400`
Seuil de déclenchement	`0.000`
Fenêtre de déclenchement	`0.001`
Evénement de déclenchement	`ncIN_WINDOW + ncENTRY`
Paramètre à déclencher	Régulateur de position : vitesse de consigne
Paramètres sélectionnés	Régulateur de position : erreur de traînage Régulateur de position : vitesse de consigne

Avec cette configuration, les données d'acquisition pourraient se présenter de la manière suivante :



Evénement de déclenchement

2.6 Actions NC

sujet	action	Description
ncTRACE	ncSTART	Démarrer l'acquisition des données
ncTRACE	ncSTOP	Arrêter l'acquisition des données

2.6.1 Lancement de la tâche d'acquisition

Appel de fonction : `status = ncaction(<nc_object>,ncTRACE,ncSTART)`

Paramètre : "`.trace`" (sauf "`.trace.status`")

Condition(s) : "`.trace.status <= ncTR_END`"

Effet(s) : Démarrage de la tâche d'acquisition avec les paramètres spécifiés

Indications(s) d'état: "`.trace.status = ncTR_START`"
après réception de l'état `ncOK` pour l'action NC (s'il n'y a pas violation des conditions de démarrage)

Indication de l'état de la tâche d'acquisition en cours dans "`.trace.status`"

Si "`.trace.status == ncTR_END`" est vérifié, cela signifie que l'acquisition des données sur le variateur est terminée !
Si "`.trace.status == ncOFF`" est vérifié, cela signifie que l'acquisition des données sur le variateur n'a pas été lancée (à cause par exemple d'une erreur d'initialisation).

Lors du traitement de cette action NC, le gestionnaire NC met l'état (status) à "**`ncTR_START`**". La configuration d'acquisition est transférée à partir des données utilisateur jusqu'au variateur, avec les ID de paramètre correspondants. Après transfert réussi de la configuration d'acquisition, le gestionnaire NC envoie la commande de démarrage au variateur. "**`ncTR_END<trace.status<ncTR_START`**" indique si le variateur est en train d'exécuter la tâche d'acquisition de données. "**`trace.status=ncTR_END`**" indique si le variateur a terminé l'exécution de la tâche d'acquisition.

2.6.2 Arrêt de la tâche d'acquisition

Appel de fonction : `status = ncaction(<nc_object>,ncTRACE,ncSTOP)`

Structure de données : -

Condition(s) : -

Effet(s) : Arrêt (annulation) de la tâche d'acquisition

Indications(s) d'état: "`.trace.status <= ncTR_END`"
après arrêt réussi de la tâche d'acquisition

Après cette action NC, le gestionnaire NC transfère la commande d'arrêt au variateur. "**`trace.status=ncTR_END`**" dans les données utilisateur indique que le variateur a acquitté la commande d'arrêt.

2.7 Traitement de la tâche d'acquisition dans le gestionnaire NC

Lors du traitement de l'action NC "ncTRACE,ncSTART", le gestionnaire NC transfère la configuration d'acquisition depuis les données utilisateur jusqu'au variateur, avec les ID de paramètres suivants (dans l'ordre indiqué) :

Données utilisateur	Type	ID de paramètre	Description
0 pour test_dat[**i**]	UINT	TRACE_TEST_INDEX	Indice pour donnée de test
test_dat[**i**].par_id	UINT	TRACE_TEST_PARID	ID de paramètre pour le point de données de test
...	...	...	...
n pour test_dat[**m**]	UINT	TRACE_TEST_INDEX	Indice pour donnée de test
test_dat[**m**].par_id	UINT	TRACE_TEST_PARID	ID de paramètre pour point de données de test
trigger.par_id	UINT	TRACE_TRIGGER_PARID	ID de paramètre pour déclenchement
trigger.event	USINT	TRACE_TRIGGER_EVENT	Événement de délenchement
trigger.threshold	REAL	TRACE_TRIGGER_THRESHOLD	Seuil de déclenchement
trigger.window	REAL	TRACE_TRIGGER_WINDOW	Fenêtre de déclenchement
t_sampling	REAL	TRACE_T_SAMPLING	Temps d'échantillonnage
t_delay	REAL	TRACE_T_DELAY	Retard de déclenchement
t_trace	REAL	TRACE_T_TRACE	Temps d'enregistrement

Le transfert des données dans cet ordre garantit que chaque paramètre dans le variateur peut être vérifié immédiatement à la réception d'un télégramme de requête d'écriture.

A ce jour, 10 points de données de test au plus peuvent être configurés pour une acquisition de données. Concernant la configuration des points de données de test, les espaces vides dans les données utilisateur sont autorisés avec "test_dat[i].par_id=ncOFF". Néanmoins, ces points de données de test doivent être transférés au variateur sans espaces vides. En effet, si l'on entre "par_id=ncOFF" (ou "par_id=0") pour un point de donnée de test dans le variateur, aucun autre point de données ne suit et seuls les points de données dont l'indice est plus petit sont valides.

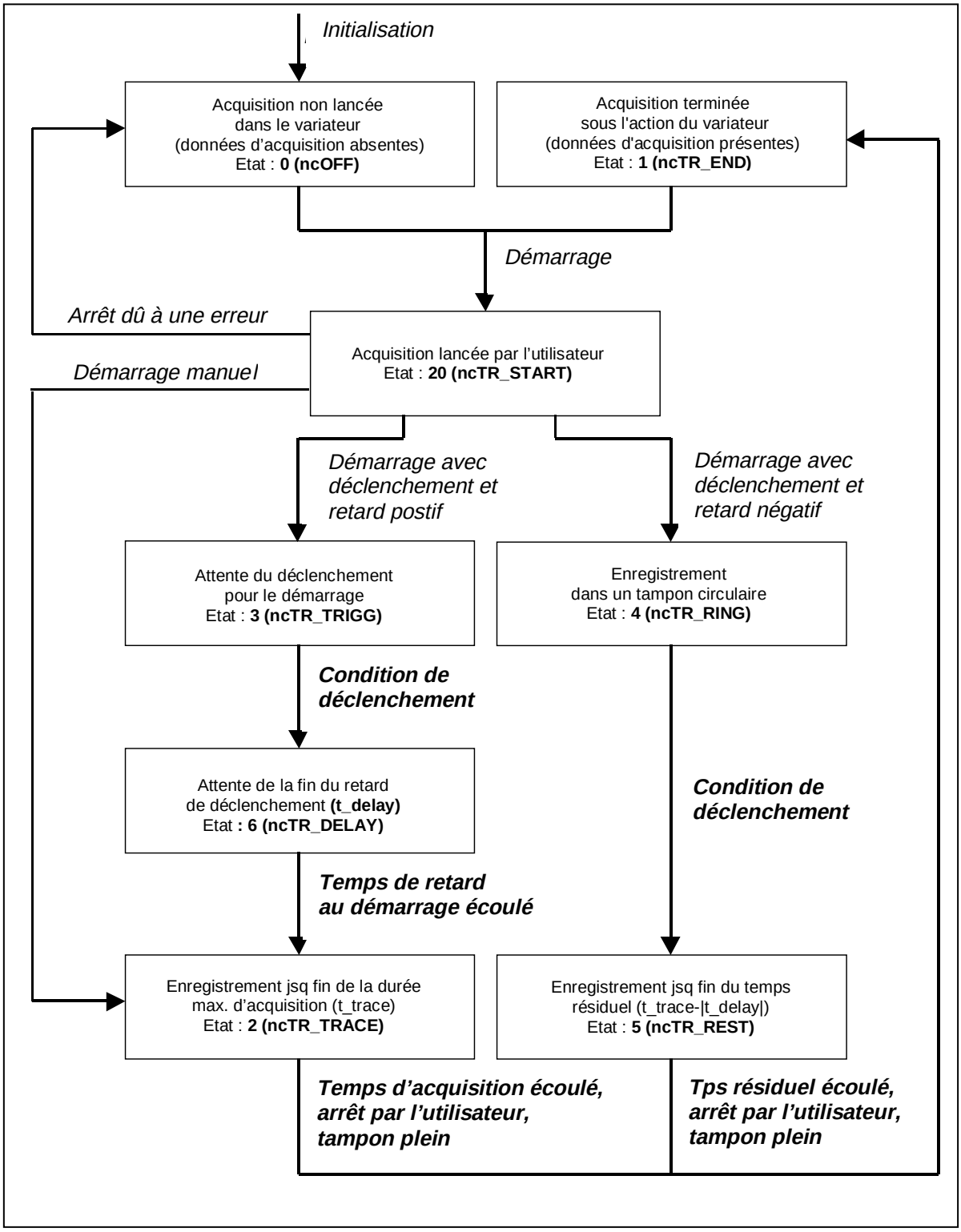
Exemple :

Si l'on effectue les entrées suivantes dans les données utilisateur,	alors ceci est transféré au variateur

`test_dat[0].par_id = PARID_S_SET;`
`test_dat[1].par_id = ncOFF;`
`test_dat[2].par_id = PARID_V_SET;`
`test_dat[3].par_id = ncOFF;`
`...`
`test_dat[9].par_id = ncOFF;`
	`TRACE_TEST_INDEX = 0`
	`TRACE_TEST_PARID = PARID_S_SET`
	`TRACE_TEST_INDEX = 1`
	`TRACE_TEST_PARID = PARID_V_SET`
	`TRACE_TEST_INDEX = 2`
	`TRACE_TEST_PARID = 0...`**Marque de fin**

Lors du traitement de l'action NC "ncTRACE,ncSTART", le gestionnaire NC met l'état (status) à **"ncTR_START"** dans les données utilisateur. La configuration d'acquisition est transférée à partir des données utilisateur jusqu'au variateur, avec les ID de paramètre correspondants. Après transfert réussi de la configuration d'acquisition, le gestionnaire NC envoie la commande de démarrage au variateur. Après démarrage réussi, le gestionnaire NC lit cycliquement (environ tous les 100 ms) l'état d'acquisition avec l'ID de paramètre "**TRACE_STATUS**" et indique cet état dans la structure de données utilisateur. **"ncTR_END<trace.status<ncTR_START"** indique si le variateur est en train d'exécuter la tâche d'acquisition de données.

2.8 Etats de la tâche d'acquisition



MISE EN SERVICE

1. GENERALITES

Dans une application d'automatisation, la mise en service constitue l'étape finale après l'élaboration du projet et la programmation.

Un réglage optimal de la boucle de régulation contribue grandement à améliorer la qualité et la dynamique du système.

Les pages suivantes esquissent la théorie relative aux boucles de régulation et exposent une méthode permettant de paramétrer ces boucles.

Les différentes parties seront abordées étape par étape.

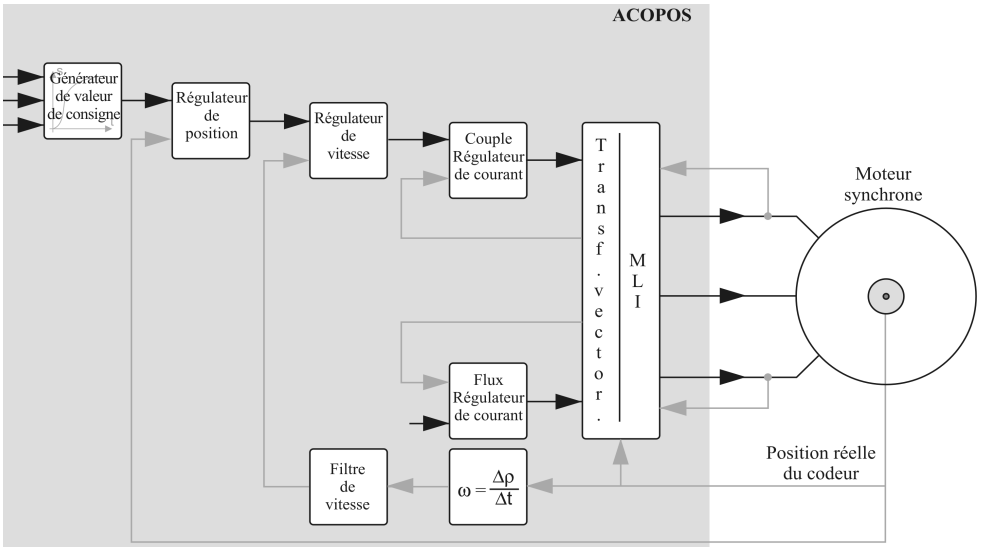


Fig. 9.1 : **Vue d'ensemble du régulateur**

2. SECURITE

C'est la sécurité qui doit prévaloir lors de toute mise en service !

Vérifier les protections

- Protection électrique (câblage, mise à la terre, tensions, ...)
- Protection mécanique
- Aucune personne dans la zone dangereuse
- Contact visuel avec l'axe
- Fonction arrêt d'urgence
- Vérifier le butoir en caoutchouc à l'extrémité mécanique de l'axe

Limiter les risques

- Centrer l'axe
- Supprimer les outils de machine onéreux (en fonction des possibilités)
- Retirer les obstacles de la zone de travail
- Rapprocher l'interrupteur matériel de fin de course (en fonction des possibilités)

Remarque

Vous trouverez des informations détaillées sur la sécurité dans le manuel ACOPOS Consignes de sécurité !

3. REGULATEUR DE COURANT

Le régulateur de courant utilisé (élément PI) applique le courant de consigne issu du régulateur de vitesse ou de flux, en l'adaptant aux caractéristiques électriques de l'enroulement.

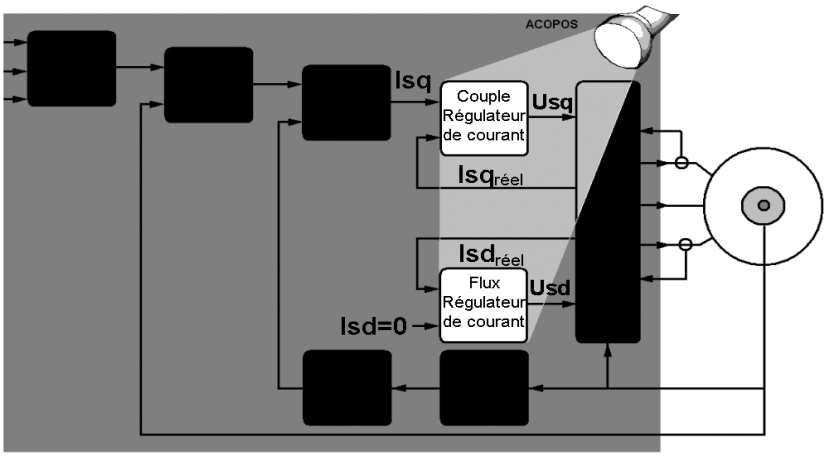


Fig. 9.2 : **Régulateur de courant**

3.1 Règles de paramétrage

- Auto Tuning – utilisation de moteurs B&R
- Manuel – moteurs d'autres fabricants

4. REGULATEUR DE VITESSE

Le régulateur de vitesse utilisé (élément de type PI) transforme la valeur de consigne issue du régulateur de position en un courant approprié, lequel est multiplié par k_C (= couple).

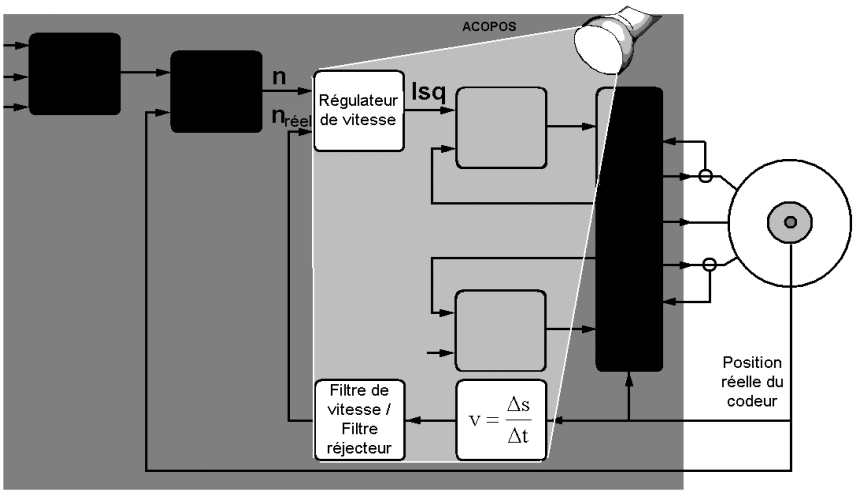


Fig. 9.3 : **Régulateur de vitesse**

4.1 Théorie

Le régulateur de vitesse est un élément PI. La théorie correspondante est traitée dans de nombreux ouvrages spécialisés.

4.1.1 Filtre de vitesse

Après évaluation de la position réelle par rapport au temps, le signal de vitesse réelle transmis au régulateur PI de vitesse est perturbé par un bruit de haute fréquence du fait de la transmission dans la ligne de signal.

Un filtre de vitesse doit éliminer ce bruit dans le signal de vitesse pour garantir la qualité de la régulation.

Le temps de filtre doit être réglé de telle sorte qu'il soit "suffisamment" supérieur (au moins d'un facteur 10) à l'inverse de la fréquence limite du bruit de signal et "suffisamment" **inférieur à l'inverse de la fréquence limite du système mécanique**.

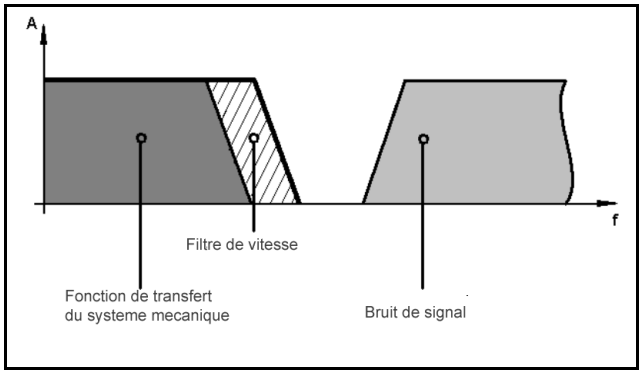


Fig. 9.4 : **Filtre de vitesse**

4.1.2 Filtre réjecteur de fréquence

Dans une mécanique d'entraînement exempte de distorsions, le système mécanique Rotor - Liaison mécanique - Charge est sujet aux oscillations mécaniques du fait du retour de position et de vitesse dans la boucle de régulation fermée. Ce comportement d'"**oscillateur bi-masse**" est observé dans la plage de fréquence 400-1500 Hz.

La **fréquence de résonance** due à l'"oscillation bi-masse" peut être atténuée de façon sélective grâce à un **filtre réjecteur de fréquence** (notch en anglais).

On obtient ainsi une meilleure précision de trajectoire et des temps de cycle plus petits pour les processus de positionnement, tout en assurant une bonne stabilité.

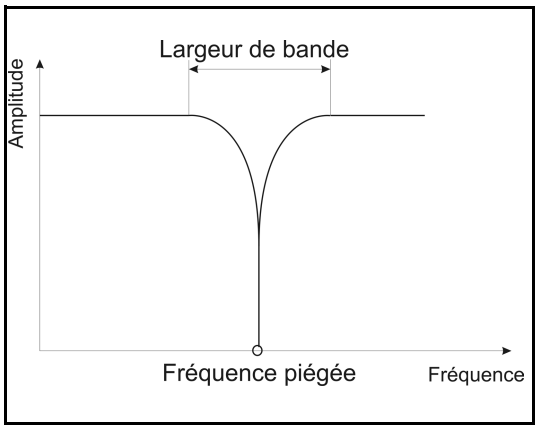


Fig. 9.5 : **Filtre réjecteur de fréquence**

A la fréquence "piégée" qui doit être réglée par la suite à la fréquence de résonance, l'amplification du filtre réjecteur est égale à zéro. A des fréquences plus hautes ou plus basses, l'amplification augmente pour atteindre une valeur constante.

4.2 Règles de paramétrage

4.2.1 Commandes NC

ncCONTROLLER	ncINIT
ncCONTROLLER	ncSWITCH_OFF
ncCONTROLLER	ncSWITCH_ON

Fig. 9.6 : **Commandes**

4.2.2 Amplification proportionnelle

Nom de paramètre	kv
Type de données	REAL (4 octets)
Unité	Ampère seconde / tour
Réglage	• Effectuer le réglage en corrélation avec le filtre de vitesse • Doubler à chaque fois la valeur jusqu'à ce que l'axe commence à osciller • Diminuer la valeur d'1/3 lorsque l'axe oscille
Valeur initiale	0,1 As/tour

4.2.3 Filtre de vitesse

Nom de paramètre	TI_SPEED_FIL
ID de paramètre pour Service Channel	283
Type de données	REAL (4 octets)
Unité	Secondes
Réglage	• Effectuer le réglage en corrélation avec l'amplification proportionnelle • Doubler à chaque fois la valeur jusqu'à ce que l'axe commence à osciller
Valeur initiale	0,2 ms
Valeur max.	5 ms

4.2.4 Filtre réjecteur de fréquence

- Fréquence piégée

Nom de paramètre	TI_SPEED_FIL
ID de paramètre pour Service Channel	226
Type de données	REAL (4 octets)
Unité	1/seconde
Réglage	• Mettre le gain proportionnel du régulateur de position à 0 ; augmenter le gain proportionnel du régulateur de vitesse jusqu'à ce que l'axe commence à osciller. • Déterminer la fréquence de résonance en appliquant la transformation de Fourier rapide (FFT) au signal du courant transversal réel dans le Centre de Test.
Valeur initiale	400 Hz
Valeur max.	1600 Hz

- Filtre réjecteur de fréquence – Bande passante

Nom de paramètre	TI_SPEED_FIL
ID de paramètre pour Service Channel	227
Type de données	REAL (4 octets)
Unité	1/seconde
Réglage	• Augmenter par paliers jusqu'à ce que l'axe n'oscille plus.
Valeur initiale	20 Hz
Valeur max.	300 Hz

4.2.5 Gain intégral

Nom de paramètre	Tn
Type de données	REAL (4 octets)
Unité	Secondes
Réglage	• Réduire à chaque fois la valeur de moitié jusqu'à ce que l'axe commence à osciller • Augmenter la valeur d'1/3 lorsque l'axe oscille
Valeur min.	5 ... 10 x temps de filtre
Valeur initiale	Dans le domaine des secondes

La mise en cascade du régulateur P avec le système prédictif permet de se passer de la partie intégrale dans la plupart des applications (appliquer Tn = 0).

5. REGULATEUR DE POSITION

Le régulateur de position PI avec consigne prédictive de vitesse convertit la valeur de consigne issue du générateur de position en une vitesse appropriée.

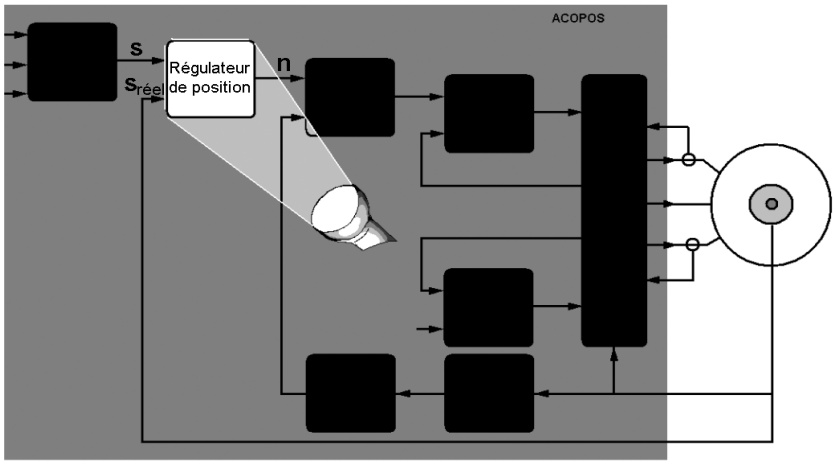


Fig. 9.7 : **Régulateur de position**

5.1 Théorie

Pour atteindre une précision de positionnement suffisante et garantir ainsi la qualité de vos produits, nous devons effectuer trois tâches concrètes :

Tâche 1)

Lorsque l'on utilise un simple régulateur PI, il se produit toujours une erreur de traînage = position de consigne - position réelle. L'amplification de cette erreur avec le facteur kv du régulateur de position donne la vitesse de consigne. L'erreur de traînage est donc fonction de la vitesse.

Solution 1 :

Utilisation d'une consigne prédictive pour éviter toute erreur de traînage pendant la phase de mouvement à vitesse constante.

Tâche 2)

En intégrant la vitesse à la boucle de régulation en plus de la position, on obtient à nouveau une erreur de traînage pendant les phases d'accélération et de décélération. Cette erreur est fonction du gain proportionnel du régulateur de position.

Solution 2 :

Une procédure spécifique permettant de prédéfinir la valeur de consigne de position en fonction de la réactivité de l'axe permet d'éviter toute erreur de traînage. La précision s'en trouve augmentée.

Tâche 3)

Dans le cas d'axes solidaires (CNC, profils de cames électroniques,…), il se produira un écart entre la position réelle et la position de consigne car les axes à mettre en mouvement ont des inerties différentes, ce qui a une incidence négative en terme de qualité.

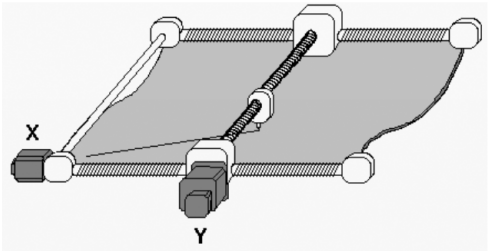
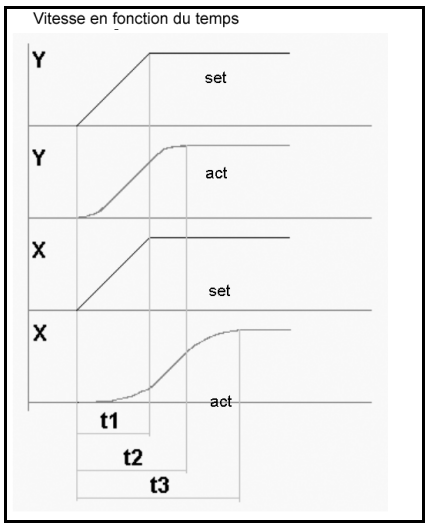


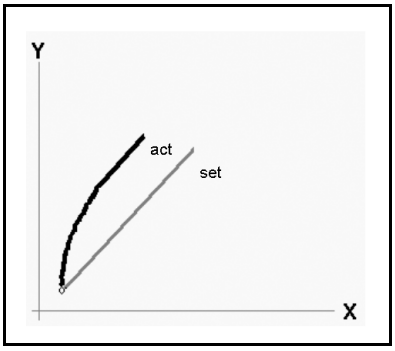
Fig. 9.8 : **Modèle multiaxes**

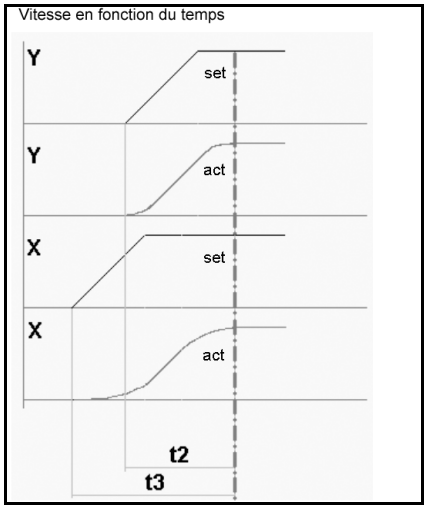


Deux axes - X et Y - doivent être mis en mouvement avec le même profil de vitesse de consigne pour provoquer un mouvement linéaire dans le plan XY.

La décélération de l'axe X étant différente de celle de l'axe Y, Y atteint la vitesse de consigne plus rapidement que X.

De ce fait, la trajectoire réelle ne correspond pas en tout point à la trajectoire recherchée, ce qui a une incidence négative en terme de qualité.





Pour que la trajectoire réelle corresponde le plus précisément possible à la trajectoire recherchée, il convient de décaler dans le temps le démarrage des deux axes en tenant compte des différences d'inertie.

Solution 3 :

Avec le temps de prédiction, le régulateur de position prédictif de B&R tient compte des différents temps de réaction des axes.

La **prédiction** permet d'assurer qu'**aucune erreur de traînage** ne se produira pendant les phases d'accélération, de vitesse constante et de décélération.

La vitesse de consigne est calculée à partir de la vitesse prédictive et de la vitesse de correction.

La vitesse prédictive est calculée à partir des positions de consigne futures et constitue la plus grande part de la grandeur réglante (vitesse de consigne).

L'amplification proportionnelle intégrale des écarts de position dûs à des perturbations donne la vitesse de correction.

Si le temps de prédiction et le variateur sont correctement paramétrés et si l'on évite ainsi les écarts de position sans que le régulateur PI n'intervienne, l'amplification intégrale n'entraîne pas de suroscillations non sollicitées au-delà de la position cible.

L'avantage que procure l'amplificateur PI, c'est-à-dire la grande rigidité statique de la charge, peut donc toujours être mis à profit.

Le temps de prédiction génère un temps de retard. Pour générer, dans les applications multiaxes, le même retard et assurer ainsi le synchronisme entre tous les axes, on peut définir un temps de retard total. Ce dernier doit être identique pour tous les axes et déterminé en fonction de l'axe correspondant au temps de prédiction le plus long.

L'action du régulateur de position peut être limitée pour empêcher l'effet de repliement. Cet effet apparaît si le variateur est entraîné jusqu'aux limites de vitesse et de couple et si l'écart de régulation ne peut plus être corrigé. L'intégrateur de l'amplificateur PI fonctionne alors en zone limite et ne revient en position stable qu'après l'apparition d'un écart de régulation négatif (suroscillation).

5.2 Règles de paramétrage

5.2.1 Commandes NC

ncCONTROLLER	ncINIT
ncCONTROLLER	ncSWITCH_OFF
ncCONTROLLER	ncSWITCH_ON

Fig. 9.9 : **Commandes**

5.2.2 Gain proportionnel

Nom de paramètre	kv
Type de données	REAL (4 octets)
Unité	1/seconde
Réglage	• Doubler à chaque fois la valeur jusqu'à ce que l'axe commence à osciller • Diminuer la valeur d'1/3 lorsque l'axe oscille
Valeur initiale	1/s

5.2.3 Temps de prédiction

Nom de paramètre	t_prédiction
Type de données	REAL (4 octets)
Unité	Secondes
Réglage	• Augmenter la valeur • Enregistrer et visualiser un mouvement (oscilloscope) • Répéter ce processus jusqu'à ce que le comportement souhaité soit atteint
Valeur initiale	1 cycle d'échantillonnage de régulateur de position, mais toujours supérieur à t_filtre

Le temps de retard total correspond à celui de l'axe le plus lent (temps de prédiction le plus élevé) dans un groupe d'axes solidaires.

Pour un axe unique, t_total = t_prédiction !

5.2.4 Gain intégral

Nom de paramètre	Tn
Type de données	REAL (4 octets)
Unité	Secondes
Réglage	• Doubler ou réduire de moitié la valeur • Enregistrer et visualiser un mouvement (oscilloscope) • Répéter ce processus jusqu'à ce que le comportement souhaité soit atteint
Valeur initiale	3/kv

On utilise une partie intégrale pour éviter les écarts de position lorsque l'axe est à l'arrêt (axe de levage par exemple). Néanmoins, ce procédé entraîne systématiquement l'apparition d'une suroscillation !

5.2.5 Limites du régulateur de position

Nom de paramètre	p_max
Type de données	REAL (4 octets)
Unité	unités/seconde
Réglage	• Tenir compte du courant impulsionnel maximal du moteur ! • $p\_max = \frac{I_{ImpulsMoteur}}{kv_{vitesse}} + 10\%$

• Anti-repliement

Nom de paramètre	i_max
Type de données	REAL (4 octets)
Unité	unités/seconde
Réglage	• Pour atteindre le couple de maintien requis (axe vertical par exemple) • $i\_max = \frac{\frac{C_{maintien}}{k_C}}{kv_{vitesse}} + 10\%$

Exemple

Pour l'application du carrousel, la mise en service doit être effectuée de la façon suivante :

- Régler le gain proportionnel pour les régulateurs de position et de vitesse
- Paramétrer le filtre réjecteur de fréquence et le filtre de vitesse
- Si l'axe est au repos, pas de mouvement

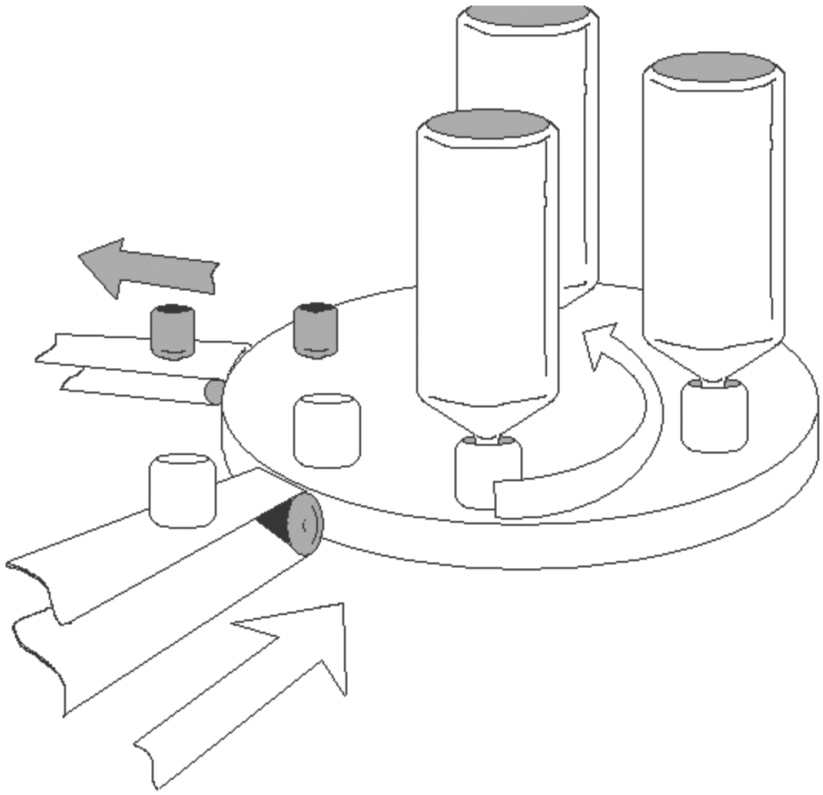


Fig. 9.10 : **Carrousel**

6. METHODE DE PARAMETRAGE D'AXE

Il existe deux méthodes pour paramétrer un axe :

- analyse mathématique
- méthode heuristique par approximations successives

Pour faciliter ce travail de paramétrage, nous présentons la marche à suivre sous la forme d'un diagramme. Vous trouverez des explications sur ce diagramme dans les pages suivantes !

6.1 Diagramme

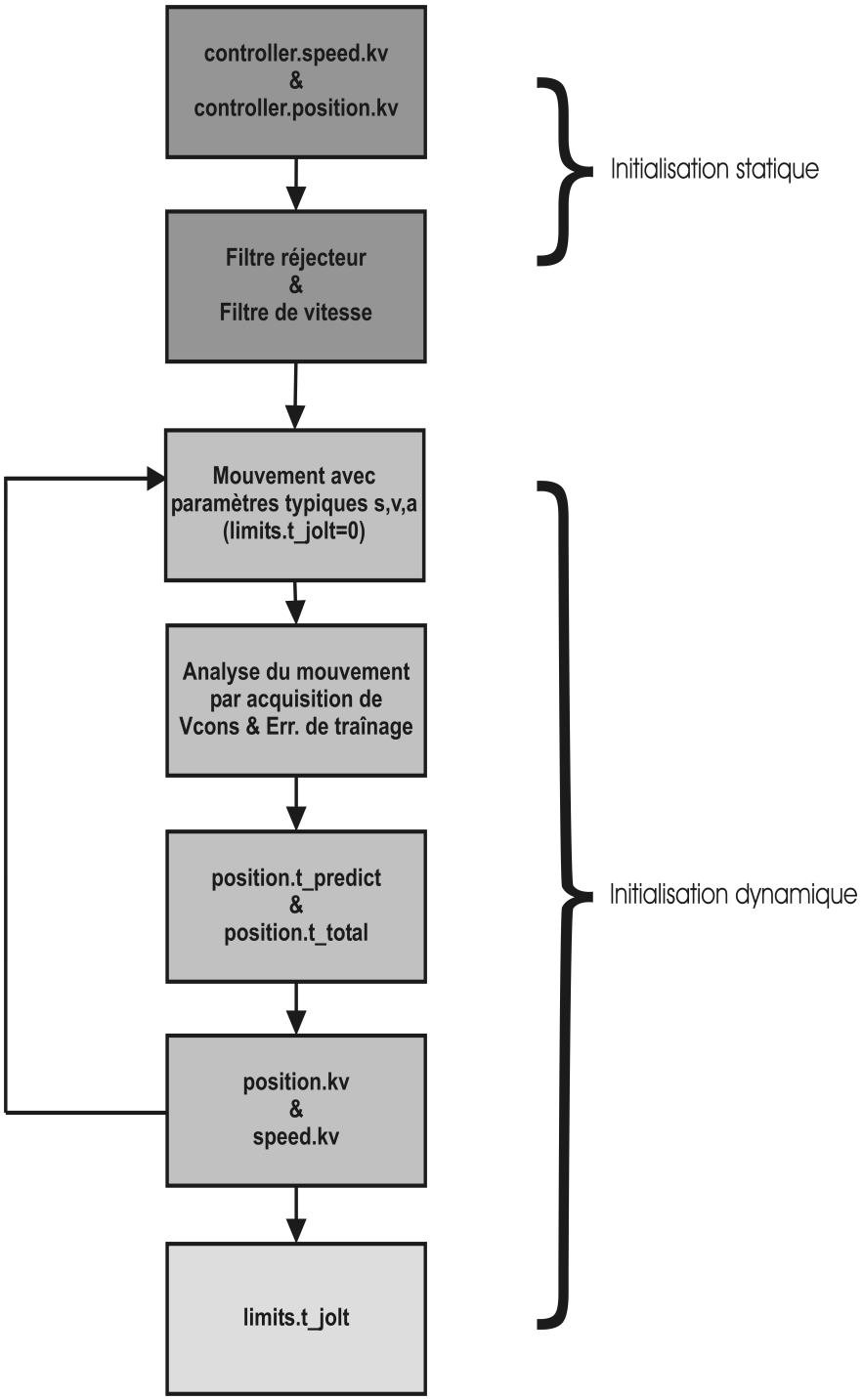


Fig. 9.11 : **Méthode de paramétrage d'axe**

Exemple

Exécutez un mouvement avec les paramètres suivants et optimisez le régulateur en minimisant l'erreur de traînage !

.s = 8000

.v = 10000

.a = 50000

.t_vibration = 0

1 tour moteur = 1000 unités

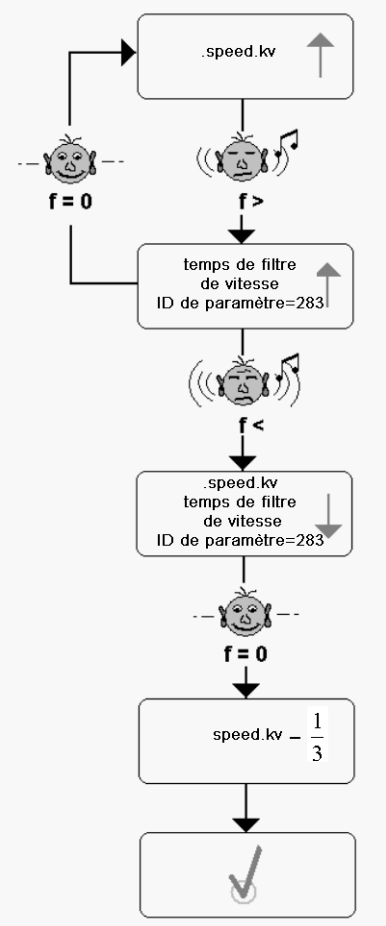
$t_{échantillonnage}$ = 0,001s (oscilloscope)

Nom du groupe				
position.kv				
position.t_predict				
speed.kv				
speed.tn				
Tfiltre				
Δs (erreur de traînage)				

6.2 Initialisation statique

Avec l'initialisation statique, nous cherchons à déterminer des valeurs initiales pour le gain proportionnel et le temps de filtre de vitesse nous permettant d'avoir un axe stable, exempt d'oscillations (bruit). Ensuite, nous pourrons effectuer un réglage dynamique pour affiner le réglage précédent et attribuer des valeurs définitives aux paramètres.

6.2.1 position.speed.kv & temps de filtre de vitesse



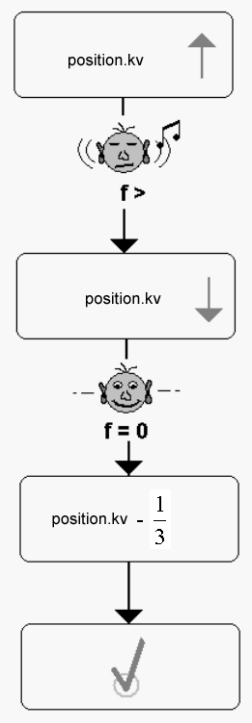
Doubler la valeur jusqu'à l'apparition d'oscillations

Augmenter le temps – si aucun bruit (f = 0), retour à l'étape précédente
si l'on entend un bruit grave (f <), aller à l'étape suivante

Diminuer kv & temps de filtre jusqu'à ce que l'on n'entende plus aucun bruit

Diminuer kv & temps de filtre d'$\frac{1}{3}$

6.2.2 controller.position.kv



Doubler jusqu'à ce que l'axe se mette à osciller

Diminuer la valeur jusqu'à ce que l'axe soit stable

Retrancher du facteur kv.

Terminé !

6.3 Initialisation dynamique

Maintenant, il s'agit d'optimiser les paramètres pour obtenir un **axe dynamique et stable** avec une **erreur de traînage la plus faible possible**.

L'enregistrement et l'analyse des mouvements à l'aide des Motion Components permet d'atteindre cet objectif.

Grâce aux possibilités de manipulation et de test qu'offrent les **Motion Components**, vous pourrez vous familiariser rapidement à ce type de tâche.

Le **moyen le plus rapide et le plus simple** de paramétrer un axe consiste à modifier un paramètre, à visualiser le mouvement avec l'oscilloscope intégré et à constater si le résultat obtenu est meilleur ou non !

Vous obtiendrez un axe dynamique et stable en appliquant les trois règles suivantes :

- Seulement 2 suroscillations – pas d'oscillation constante
- Eliminer l'erreur de traînage
- Eviter l'apparition d'une erreur de traînage négative une fois la position de consigne atteinte

6.3.1 Seulement 2 x suroscillations – pas d'oscillation constante

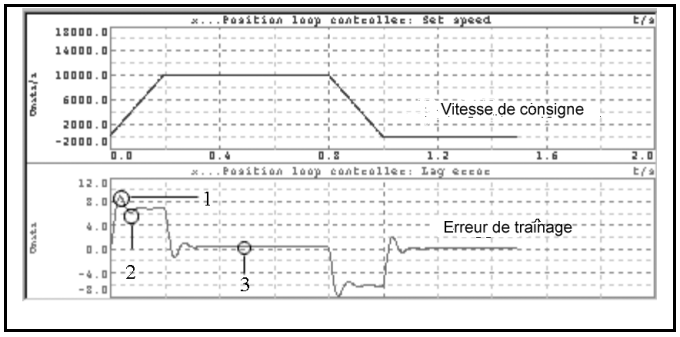


Fig. 9.12 : **Visualisation de l'erreur de traînage avec l'oscilloscope intégré**

Essayez de modifier le gain proportionnel du régulateur de vitesse et du régulateur de position jusqu'au moment où deux suroscillations seulement se produisent.

Plus le gain kv du régulateur de position est élevé, plus le gain kv du régulateur de vitesse peut être augmenté.

Toutefois, essayez de maintenir l'axe à un état stable, sans oscillations constantes!

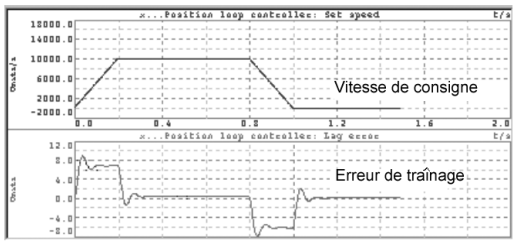
Pendant une phase de mouvement à vitesse constante, vous pouvez éliminer l'erreur de traînage en augmentant le gain kv du régulateur de position.

La mise en cascade du régulateur de vitesse et du régulateur de position permet de corriger les écarts de position pendant les phases de mouvement à vitesse constante. Ce procédé se substitue donc à la partie intégrale dans le régulateur de vitesse sans en présenter les inconvénients (suroscillation accentuée).

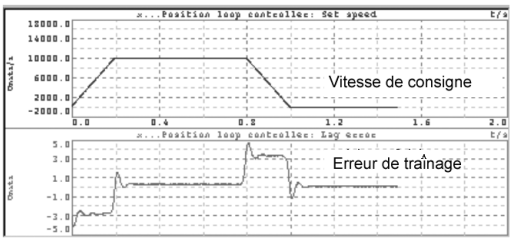
6.3.2 Eliminer l'erreur de traînage

En paramétrant le temps de prédiction t_prédiction, nous allons maintenant essayer de réduire à 0 l'erreur de traînage se produisant en phase d'accélération et de décélération.

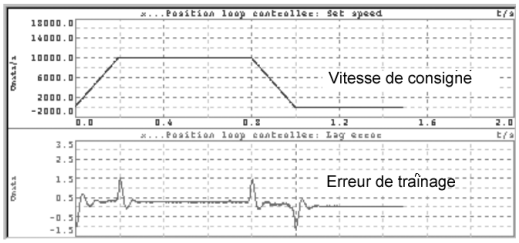
Si le temps de prédiction est trop petit, on obtient une erreur de traînage positive pendant la phase d'accélération



Si le temps de prédiction est trop grand, on obtient une erreur de traînage négative pendant la phase d'accélération



L'objectif est atteint lorsque l'erreur de traînage est quasi nulle, exception faite des pics brefs se produisant lors des changements de vitesse de consigne.



6.3.3 Eviter l'apparition d'une erreur de traînage négative une fois la position de consigne atteinte

Lors du réglage affiné des gains dans les régulateurs de vitesse et de position, il faut veiller à ce que l'erreur de traînage ne présente aucune suroscillation négative après l'achèvement du profil de vitesse de consigne. Sinon, l'axe irait trop loin et il faudrait alors effectuer un mouvement de retour !

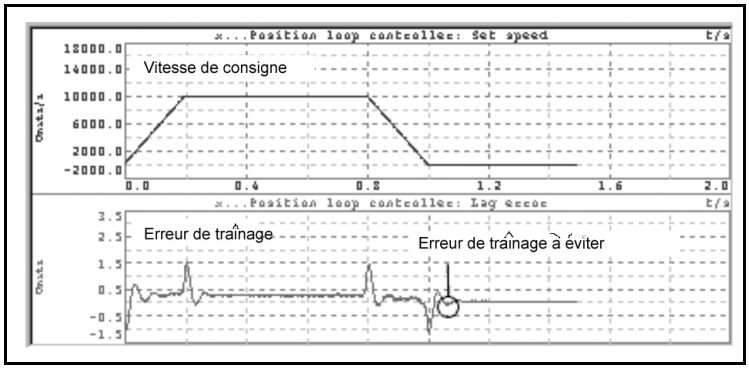


Fig. 9.13 : **Visualisation du mouvement avec l'oscilloscope intégré**

7. LITTERATURE

Pour toute information complémentaire, vous pouvez consulter un des nombreux ouvrages disponibles dans la littérature spécialisée. Exemple :

Titre de l'ouvrage	**Maison d'édition**	**ISBN**
Control of Electrical Drives	Springer	3-540-59380-2
...	...	...
...	...	...

EXEMPLE D'APPLICATION

1. CONTRÔLE MONOAXE D'UN CARROUSEL

Cahier des charges :

- Le carrousel se subdivise en 5 parties égales (360°/5)
- Le carrousel se déplace par cycles d'horloge : à chaque cycle, le disque se déplace d'un segment en 200 ms puis s'arrête pendant 1 seconde.
- La précision de positionnement requise est de ±1°.
- La solution offerte est une solution d'entraînement direct.

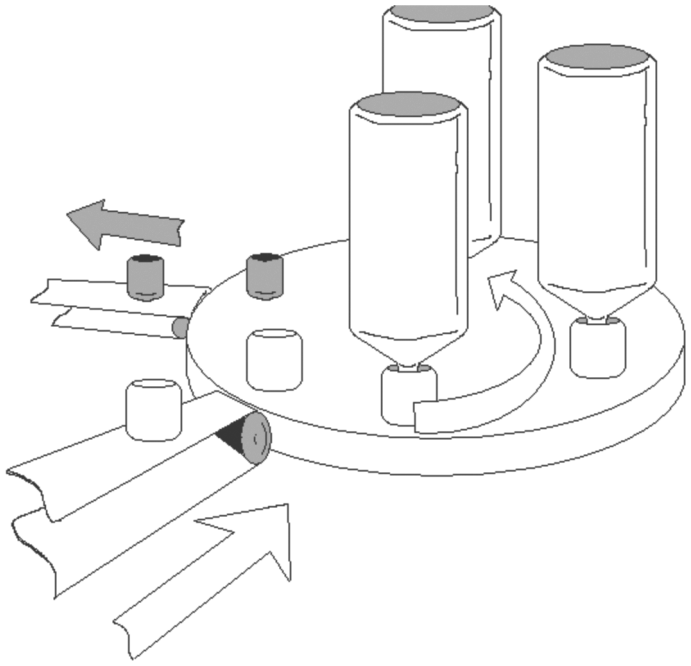


Fig. 10.1 : **Carrousel**

1.1 Solution

Etape 1 : trouver un profil de vitesse approprié (diagramme vt). Celui-ci permettra d'élaborer ensuite avec B&R Automation Basic la tâche logicielle pour l'automatisation des séquences de mouvement.

Ensuite, il faut définir une mise à l'échelle appropriée pour l'interface codeur ainsi que les paramètres de mouvement respectifs. Ces paramètres doivent être stockés dans un "Module de Paramètres d'Initialisation".
Ce procédé est d'un accès facile, même pour les programmeurs non expérimentés.

Etape 2 : définir des paramètres de régulation optimaux pour garantir la qualité du produit. Le "Centre de Test" vous aidera dans cette tâche !

Etape 3 : créer la tâche logicielle, dans le "sous-programme d'initialisation", qui appelle la fonction ncalloc(...) en utilisant le numéro de nœud adéquat !

Etape 4 : créer la séquence d'instructions dans la partie cyclique, en commençant par l'activation du régulateur.

Etape 5 : configurer la prise de référence avec les "interrupteurs de position de référence absolue". Cette configuration doit être transférée via le simulateur automate à l'ACOPOS (sur une carte d'entrée digitale), en utilisant "ncDIG_IN, ncFORCE" !

Dernière étape : programmer du mouvement cyclique

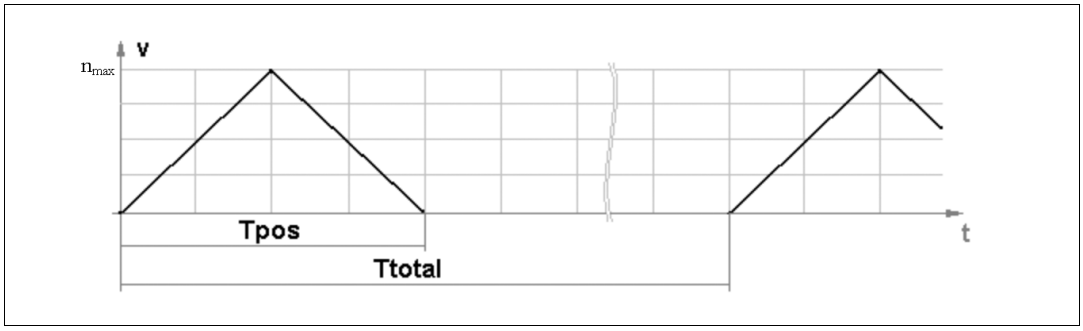
1.2 Paramètres pour une solution possible

Pour atteindre la précision de position souhaitée de ±1°, la plus petite unité doit être 0,1°.
A cet effet, on a effectué les paramétrages suivants pour l'interface codeur.

tour_mot = 1
unités_charge = 3600

-> 1 unité correspond à 0,1°

Les paramètres requis pour la séquence de mouvements ont été calculés à partir du diagramme v/t.



s= 720 unités, correspond à la surface à l'intérieur de la courbe v/t
v_max = 7200 unités/s
a_max = 72000 unités/s

RÉCAPITULATIF DU SÉMINAIRE

1. RÉCAPITULATIF DU SÉMINAIRE

- Présentation générale des systèmes

- Chaîne d'entraînement

- Dimensionnement de l'entraînement

- Concept NC

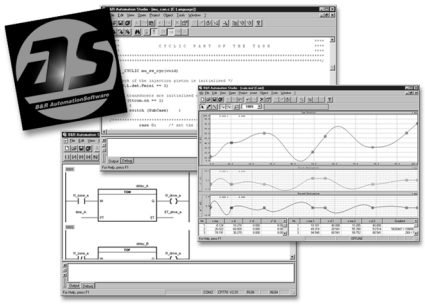
- Motion Components

- Système ACOPOS

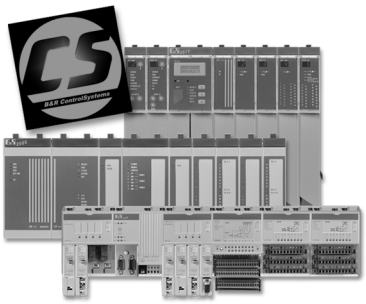
- Mise en service

- Exemple d'application

2. PROGRAMME



B&R AUTOMATIONSOFTWARE



B&R CONTROLSYSTEMS

B&R MOTIONSYSTEMS

B&R PANELSYSTEMS

3. AGENCES

Nos adresses et informations produit sont régulièrement mises à jour. Vous pouvez les trouver à l'adresse Internet suivante :

HTTP://WWW.BR-AUTOMATION.COM